Paulo Rodolfo da Silva Leite Coelho

Uma Arquitetura Orientada a Serviços para Laboratórios de Acesso Remoto

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientador: Eleri Cardozo

Co-orientadora: Eliane Gomes Guimarães

Campinas, SP
2006

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Coelho, Paulo Rodolfo da Silva Leite
C65a Uma Arquitetura Orientada a Serviços para Laboratórios de Acesso Remoto. / Paulo Rodolfo da Silva Leite Coelho. – Campinas, SP: [s.n.], 2006.

Orientadores: Eleri Cardozo; Eliane Gomes Guimarães.
Tese (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Serviços Web. 2. Redes de computação. 3. Ensino a distância. 4. Robótica. I. Cardozo, Eleri. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Título em Inglês:	A Service Oriented Architecture for Remote Access Laboratories
Palavras-chave em Inglês:	WebLab, Service oriented computing SOC, Web service, Mobile robotic
Área de concentração:	Engenharia de Computação
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Alice Maria Bastos Hubinger Tokarnia, José Raimundo de Oliveira e Josué Junior Guimarães Ramos
Data da defesa:	01/12/2006

Paulo Rodolfo da Silva Leite Coelho

Uma Arquitetura Orientada a Serviços para Laboratórios de Acesso Remoto

Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.
Aprovação em 01/12/2006

Banca Examinadora:
Profa. Dra. Alice Maria Bastos Hubinger Tokarnia - UNICAMP
Prof. Dr. Eleri Cardozo - UNICAMP
Prof. Dr. José Raimundo de Oliveira - UNICAMP
Prof. Dr. Josué Junior Guimarães Ramos - CenPRA

Campinas, SP
2006

Resumo

Este trabalho apresenta uma arquitetura para a construção de laboratórios de acesso remoto (conhecidos como *WebLabs*). A arquitetura segue o paradigma de computação orientada a serviço (COS). Nesta abordagem cada recurso físico ou lógico do laboratório é modelado como um Serviço Web. Desta forma, os experimentos podem ser disponibilizados pela composição destes serviços. Um modelo conceitual para *WebLabs*, bem como a implementação deste modelo são apresentados. Experimentos em robótica móvel também foram desenvolvidos como exemplos de utilização deste laboratório.

Palavras-chave: WebLab, COS, Serviço Web, Robótica Móvel.

Abstract

This work presents an architeture for building remote access laboratories (also known as *WebLabs*). The architecture follows the service oriented computing (SOC). In this approach each logical or physical resource of the laboratory is modeled as a Web Service. In this way, the experiments are built through the composition of such services. A conceptual model of WebLabs, as well as an implementation of this model are presented. Experiments in mobile robotic were also developed for this laboratory.

Keywords: WebLab, SOC, Web Service, Mobile Robotic.

Agradecimentos

A Deus, Senhor e Criador, pelo dom maior da vida.

Aos meus pais, Paulo e Jôze, pelo exemplo de vida, ensinando valores que jamais serão esquecidos; pelo estímulo, incentivando nas dificuldades e reconhecendo os méritos; e pelo apoio incondicional, sem o qual este trabalho não seria possível.

À minha noiva Patrícia e meu filho Gabriel, minha nova família, motivação de meus esforços e estudos.

Aos meus familiares, em especial minha avó Lucrécia e meus irmãos, pelo encorajamento e incentivo.

Ao meu orientador, Prof. Dr. Eleri Cardozo, e minha co-orientadora, Dra. Eliane Guimarães, meus sinceros agradecimentos e gratidão pela orientação e oportunidade de desenvolver este trabalho.

Aos meus colegas de laboratório, em especial ao Sassi e Alex, verdadeiros irmãos e companheiros nos bons e maus momentos.

Aos alunos de iniciação científica, Victor, Daniele e Ewerton, pela dedicação ao projeto GigaBOT.

À CAPES, pelo apoio financeiro.

Dedico este trabalho à minha noiva Patrícia, pelo amor e paciência, e a meu filho Gabriel, pelo despertar do excelso sentido da paternidade.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

API - Application Programming Interface

BPEL - Business Process Execution Language

BPEL4WS - Business Process Execution Language for Web Service

CDT - C/C++ Development Tool

CGI - Common Gateway Interface

COS - Computação Orientada a Serviço

FTP - File Transfer Protocol

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IDE - Integrated Development Environment

IP - Internet Protocol

JNLP - Java Network Launching Protocol

JSP - Java Server Pages

LAN - Local Area Network

LAR - Laboratório de Acesso Remoto

MIT - Massachussetts Institute of Technology

PTZ - Pan, Tilt e Zoom

QoS - Quality of Service

SLA - Service Level Agreement

SMTP - Simple Mail Transfer Protocol

SOA - Service Oriented Architecture

SOAP - Simple Object Access Protocol

SOC - Service Oriented Computing

TCP - Transmission Control Protocol

Tidia-Ae - Tecnologia da Informação no Desenvolvimento da Internet Avançada - Aprendizado Eletrônico

UDDI - Universal Description, Discovery and Integration

URI - Unified Resource Identifier

URL - Unified Resource Locator

VPN - Virtual Private Network

W3C - World Wide Web Consortium

WSDL - Web Service Description Language

XML - eXtensible Markup Language

XPath - XML Path Language

XSD - XML Schema Description

Trabalhos Publicados Pelo Autor

1. R. P. Pinto, E. Cardozo, A. Z. Lima, P. R. S. L. Coelho, E. G. Guimarães, R. F. Sassi, L. F. Faina. "Uma Arquitetura para Suporte a Aplicações Sensíveis a Contexto Baseada em Componentes de Software". *XXXII Conferência Latinoamericana de Informática* (CLEI'2006). Santiago. Chile. Agosto 2006.

2. R. F. Sassi, P. R. S. L. Coelho, E. Cardozo, E. G. Guimarães, L. F. Faina, A. Z. Lima, R. P. Pinto. "Uma Plataforma de Middleware para Geração de Sotfware Orientado a Componentes". *Workshop de Desenvolvimento Baseado em Componentes 2006* (WDBC'2006). Recife. Brasil. Dezembro 2006.

3. P. R. S. L. Coelho, R. F. Sassi, E. Cardozo, E. G. Guimarães, L. F. Faina , A. Z. Lima, R. P. Pinto. "A Web Lab for Mobile Robotic Education". *IEEE International Conference on Robotics and Automation 2007* (ICRA'2007). Roma. Itália. Abril 2007.

Capítulo 1

Introdução

Este capítulo apresenta as principais motivações para a adoção do paradigma da Computação Orientada a Serviço (COS) na elaboração de uma arquitetura para Laboratórios de Acesso Remoto. Contribuições para este trabalho, assim como trabalhos correlatos são também descritos. Por fim, são apresentados o contexto do trabalho e um panorama da distribuição dos tópicos ao longo dos capítulos seguintes.

1.1 Motivações

Laboratórios de Acesso Remoto (LAR) ou WebLabs têm sido propostos como poderosas ferramentas de suporte ao ensino, tanto presencial quanto a distância. Apesar da literatura apresentar um grande número de implementações, o uso deste recurso tem se limitado a um pequeno número de usuários, na maioria das vezes seus próprios desenvolvedores. Além disso, é mínimo o número destas implementações que seguem o paradigma de COS. As principais razões que limitam uma proliferação mais acentuada de WebLabs são:

1. Disponibilidade apenas em ambiente local, seja por razões de segurança, seja pela impossibilidade dos protocolos empregados pelo WebLab operarem por meio da Internet pública, uma vez que geralmente utilizam protocolos que são bloqueados por *firewalls*.

2. Disponibilidade limitada de recursos ou ausência de controle de acesso, o que impede a utilização do WebLab por um grupo maior de usuários.

3. Necessidade de utilização de sistemas de software proprietários no terminal do usuário, inviabilizando o custo desta utilização.

4. Dificuldade de incorporar novos experimentos ou alterar os existentes a fim de atender a um determinado perfil de usuários.

5. Falta de pessoal de apoio para manutenção e operação dos recursos empregados no WebLab.

Será demonstrado nesta dissertação que, utilizando COS, as restrições apresentadas acima (exceto a última) podem ser eliminadas ou fortemente atenuadas. O elemento que contribui para tal é a

composição de serviços, somado a protocolos com capacidade de cruzar diferentes domínios, como o protocolo HTTP (*HyperText Transfer Protocol*).

A disponibilidade de recursos propiciado pelo projeto GigaBOT (seção 1.4.3), tais como a rede de alta velocidade, máquinas de grande poder de processamento, robôs com suporte a rede sem fio e câmeras panorâmica e de bordo, serviram também de incentivo ao desenvolvimento de um WebLab para cursos na área de robótica móvel.

1.2 Contribuições

As seguintes contribuições foram identificadas no âmbito desta dissertação:

- Definição de um modelo independente de tecnologia para o desenvolvimento de laboratórios de acesso remoto.
- Separação dos planos de gerência, de comunicação e de interação.
- Disponibilização de recursos e experimentos em robótica na forma de serviços.
- Possibilidade de federação de WebLabs.

A primeira contribuição é conseqüência da própria natureza de COS e Serviço Web, uma vez que utilizam protocolos abertos baseados em XML. A descrição de serviços por meio desta linguagem permite a tradução desta descrição para qualquer linguagem de programação, o que torna o desenvolvimento independente da tecnologia, tanto no desenvolvimento do serviço, como na criação de clientes para os serviços.

Outra contribuição é a separação dos planos de gerência, de comunicação e de interação. Com isto, a implementação de um WebLab pode ser realizada de forma a enfatizar apenas os experimentos inerentes ao próprio laboratório (que correspondem ao plano (ou serviços) de interação). Os serviços de gerência e comunicação são comuns a qualquer WebLab, pois foram desenvolvidos de tal forma que provêm interfaces genéricas para uso dos mesmos.

Os diversos recursos (robôs, câmeras, etc) dos experimentos que compõem o laboratório podem ser acessados e combinados de forma simples e rápida, uma vez que são vistos como Serviços Web. Isto possibilita o uso dos mesmos sem necessidade de nenhum conhecimento de informações técnicas de API (*Application Programming Interface*) dos recursos.

O desacoplamento do uso e da gerência de WebLabs, permite a criação e manipulação de federações de WebLabs, também modeladas por meio de Serviços Web. Para esse controle deve ser desenvolvida uma nova sessão de gerência capaz de se comunicar com as sessões de gerência (acesso) de cada WebLab. Além disso, deve-se proceder um controle sobre o compartilhamento de recursos e experimentos, bem como o controle de acesso, em que devem ser definidas regras e condições para autenticação de usuários por meio de diferentes WebLabs e federações.

1.3 Trabalhos Correlatos

Diversas abordagens para o desenvolvimento de laboratórios de acesso remoto são disponíveis na literatura. Inicialmente eram requeridos softwares específicos dependentes de plataforma no

computador do cliente [1], [2], [3], [4]. Posteriormente, houve uma mudança para tecnologias capazes de operar no navegador do cliente, incluindo Java *applets* [5], páginas HTML (*HyperText Markup Language*) estáticas e dinâmicas [6], e *scripts* CGI (*Common Gateway Interface*) [7]. Soluções baseadas em HTML geralmente resultam em clientes com pouca carga de processamento e dependem fortemente de tecnologias do lado servidor, tal qual CGI, que acoplam fortemente o desenvolvimento do WebLab.

Muitos dos projetos atuais empregam *applet* Java para o lado cliente, devido à capacidade de processamento do Java e à sua independência de plataforma. Muitos desses sistemas dependem de *sockets* TCP/IP (*Transmission Control Protocol / Internet Protocol*) para comunicação [8], [9], os quais requerem que clientes trabalhem com um estilo de programação radicalmente diferente do paradigma orientado a objetos. O projeto iLab [10], [11] do MIT (*Massachussetts Institute of Technology*) é o que mais se aproxima dessa arquitetura. São características semelhantes:

- O uso de SOAP (*Simple Object Access Protocol*) e Serviços Web, o que garante independências de plataforma e linguagem de programação.
- Separação do plano gerência de usuários e experimentos da execução e acompanhamento de experimentos.
- Integração de diversos WebLabs (federação).

Apesar da semelhança nos aspectos citados, há diferenças fundamentais:

- iLab tem como principal alvo a interligação de WebLabs, enquanto neste trabalho é enfatizado a composição de experimentos.
- Neste trabalho, federação de WebLabs se dá por meio da composição de diversos WebLabs. No iLab ela é realizada por meio de um *Service Broker* (figura 1.1), o qual representa uma entidade centralizadora na relação entre os WebLabs que compõe a federação.

Tanto a construção de experimentos, como a possibilidade de federação de laboratórios são elaboradas por meio da composição de Serviços Web. Dada que esta é a essência de COS, esta proposta é muito mais aderente a este paradigma que a proposta do projeto iLab.

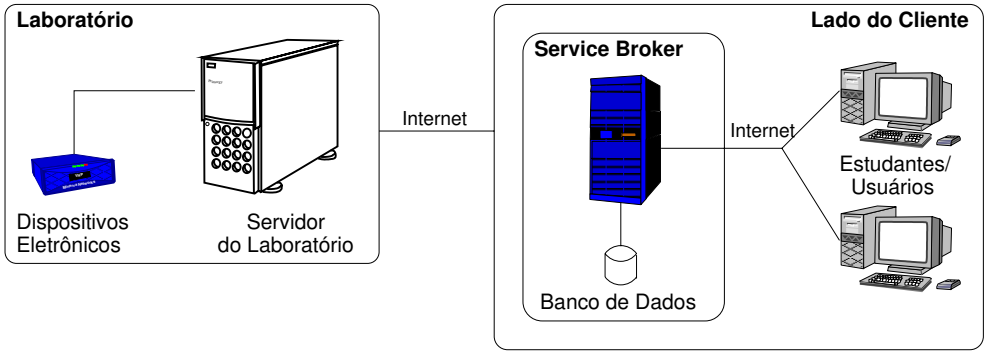


Fig. 1.1: Visão geral da arquitetura do sistema iLab

1.4 Contexto do Trabalho

O trabalho apresentado nesta dissertação faz parte do projeto GigaBOT e é uma nova abordagem para a criação de WebLabs. Um breve histórico da experiência do grupo no desenvolvimento de WebLabs e uma descrição do projeto GigaBOT são apresentados a seguir.

1.4.1 REAL WebLab

O trabalho do grupo na área de Laboratórios de Acesso Remoto (WebLabs) teve início no Centro de Pesquisas Renato Archer (CenPRA) por meio do projeto REAL [12], (*Remotely Accessible Laboratory*), em 1996, e foi tema de uma dissertação de mestrado no Instituto de Computação da UNICAMP [13]. Com este desenvolvimento inicial tornou-se clara a necessidade de uma nova infra-estrutura que pudesse atenuar as complexidades de implementação dos serviços providos pelo WebLab, nos aspectos relacionados ao desenvolvimento e controle de acesso.

Em 1997 foi criado um acordo de cooperação entre o CenPRA e a Faculdade de Engenharia Elétrica e de Computação da UNICAMP com o objetivo de construir plataformas baseadas em padrões abertos para o desenvolvimento de WebLabs. Tal acordo resultou na implementação da segunda versão do WebLab REAL, incorporando novas tecnologias tais como a arquitetura de objetos distribuídos CORBA. Esta versão incorporou funcionalidades extras como, por exemplo, o controle do robô pelo usuário por algoritmos de navegação desenvolvidos pelo próprio usuário. Adicionalmente, permitiu uma melhor interação do usuário com o laboratório e o acompanhamento de uma missão por meio de dois canais de vídeo em tempo real. Um primeiro canal, alimentado por uma câmera panorâmica, envia imagens de vídeo do ambiente onde se localizam os robôs móveis e um segundo canal, alimentado por uma câmera embarcada no computador de bordo do robô móvel, envia imagens de vídeo a partir deste robô.

A partir de 2000, iniciou-se o desenvolvimento da terceira versão do REAL WebLab [14] [15], uma evolução da anterior, utilizando o modelo de componentes CM-tel [16] [17] e a plataforma CCM-tel [18]. Esta versão utiliza páginas dinâmicas (JSP - *Java Server Pages*), *servlets* e *javascript* para acesso às funcionalidades do WebLab. Foi introduzido o conceito de sessões, as quais foram implementadas segundo o modelo de componentes CM-tel e agrupadas em três categorias: componentes de acesso, de interação e de comunicação. Cada categoria suporta uma sessão correspondente.

A sessão de acesso suporta as interações necessárias para iniciar e gerenciar a comunicação entre o usuário e o provedor de serviço, tais como: subscrição de usuários, reserva de recursos e gerência do serviço (acesso e uso do serviço).

A sessão de interação corresponde à lógica do serviço e suporta as interações que controlam, manipulam e gerenciam o comportamento do serviço, bem como o fluxo de informação associado.

A sessão de comunicação suporta a comunicação multimídia com QoS (*Quality of Control*) entre os componentes da aplicação.

Foram desenvolvidos componentes para três modos de interação:

Básico: o modo de interação básico é um serviço do REAL dedicado a usuários leigos ou com conhecimentos limitados na área de robótica, que queiram manipular os robôs por meio de teleoperação. Neste modo, os usuários interagem com o robô em um alto nível de abstração,

em que o robô é visto como um veículo capaz de se mover e de seguir uma seqüência de comandos simples.

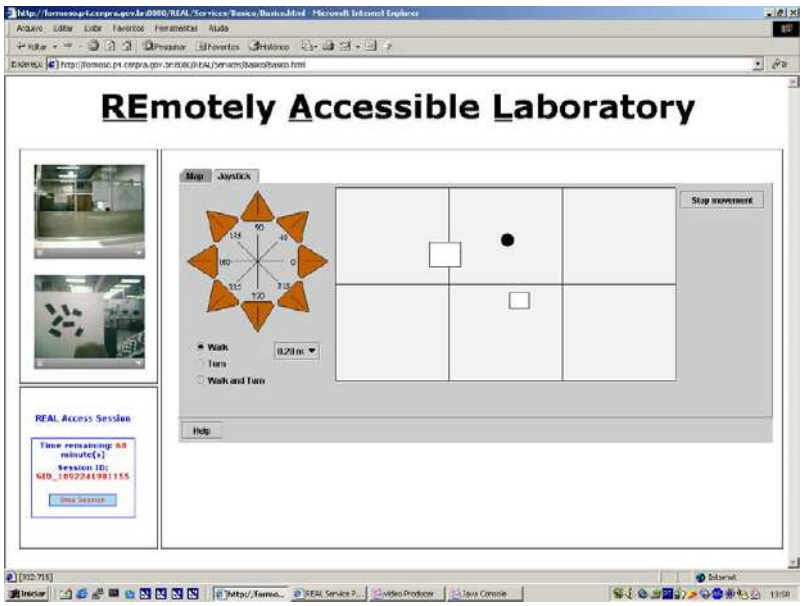


Fig. 1.2: Interface do Modo de Interação Básico do REAL.

Este serviço do REAL fornece uma interface (figura 1.2) com duas formas simplificadas de interação com o robô, por meio das quais o usuário pode enviar comandos para o robô. A primeira possibilita o controle através de um *joystick*, enquanto a segunda proporciona a determinação do destino através de cliques do *mouse* sobre o ponto desejado no mapa.

Avançado: o modo de interação avançado é dirigido a pesquisadores e estudantes com experiência em robótica que queiram desenvolver, planejar, executar e testar os seus próprios algoritmos, métodos de navegação e controle em robôs móveis, bem como experimentos robóticos complexos que exploram as funcionalidades oferecidas pelos robôs.

Assistido: o modo de interação assistido é dedicado ao aprendizado remoto pela Internet. Este modo oferece um ambiente colaborativo, que permite a múltiplos usuários o acesso simultâneo ao WebLab REAL. Este modo visa, principalmente, proporcionar um primeiro contato de estudantes com tecnologias antes inacessíveis, por exemplo, na área de robótica.

Existem dois tipos de usuários, o instrutor e os estudantes. Um usuário principal, o instrutor, pode interagir diretamente com o robô, enquanto que os estudantes são capazes apenas de acompanhar o experimento realizado por este instrutor no robô. Este modo de interação disponibiliza no navegador dos usuários dois tipos de interfaces, cada uma para um tipo de usuário. A interface do instrutor permite que este acesse o robô empregando as interfaces com todas as funcionalidades permitidas dos modos básico e avançado, enquanto que a dos estudantes consiste essencialmente da interface do modo básico, porém com as suas funcionalidades de interação desabilitadas. Assim, é possível que os estudantes acompanhem os experimentos realizados no REAL sem interferir no rumo das atividades.

Adicionalmente ao modo básico, as interfaces apresentadas para os estudantes e para o instrutor incorporam e disponibilizam três características extras: um canal de áudio, uma funcionalidade de interação e uma de apresentação de material didático. A primeira característica consiste em transportar um fluxo de áudio que permite aos estudantes ouvir a voz do instrutor em tempo real. Da mesma forma que os fluxos de vídeo, os fluxos de áudio também são diretamente suportados pelas portas de fluxo contínuo do modelo CM-tel. A segunda característica, ilustrada na figura 1.3(a), é uma funcionalidade de *chat* baseada em texto que suporta um mecanismo de comunicação e retorno entre o instrutor e os alunos. Nesta interface encontram-se os campos para a edição e visualização de texto. Dois botões, disponibilizados apenas na interface do instrutor, permitem habilitar e desabilitar o serviço de *chat*. Finalmente, um serviço de apresentação de material didático é disponibilizado, conforme ilustrado na figura 1.3(b). O lado esquerdo desta figura, disponibilizado apenas na interface do instrutor, permite ao mesmo selecionar um endereço (URL) em uma lista de URLs, contendo o conteúdo a ser apresentado (texto, figura, mídia gravada, etc.). Este conteúdo é transferido via HTTP e apresentado no navegador do aluno (lado direito da figura 1.3(b)).

A quarta versão do REAL WebLab foi desenvolvida em 2003 utilizando o modelo de componentes CM-tel e a plataforma MECM-tel [19]. Esta versão utiliza desenvolvimento de *software* baseado em componentes para dispositivos móveis com baixo poder computacional para acesso ao modo Básico (figura 1.4).

A versão atual, utilizando redes de alta velocidade (Projeto GigaBOT) é descrita na seção 1.4.3.

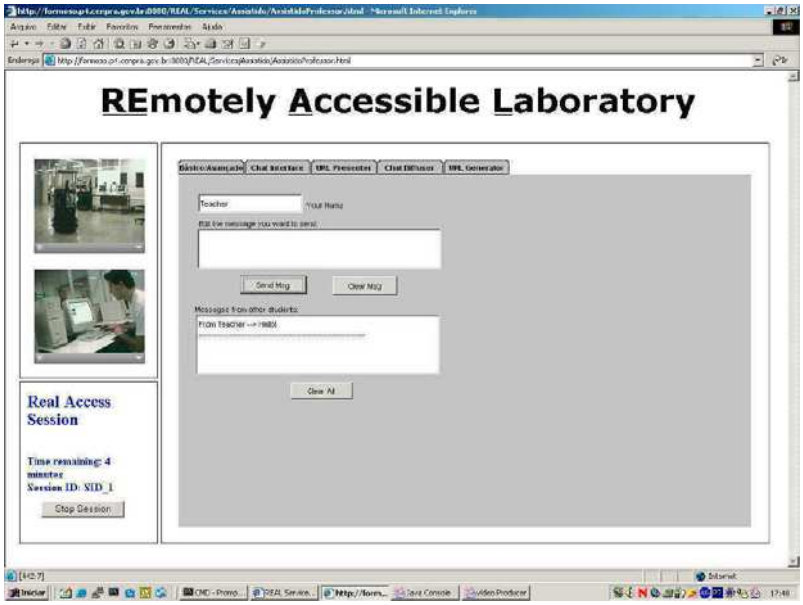
1.4.2 REAL-CT WebLab

Real-CT (Conforto Térmico) WebLab utiliza os módulos do REAL e adiciona novos módulos que serão incorporados na infra-estrutura de robótica devido a sua generalidade. Os três módulos correspondem ao módulo do robô, módulo da câmera panorâmica, e o novo módulo de servo-posicionamento. O robô tem como carga um modelo em escala de uma casa simples. O robô gira em passos de 45 graus a fim de expor o modelo a uma fonte de fluxo de ar (vento) de diferentes ângulos (representando a implantação da direcionada para o norte, nordeste, leste, sudeste, sul, sudoeste, oeste e noroeste). Os métodos relacionados à movimentação do robô apresentados na interface WSDL do Serviço Web do robô são empregados nesta tarefa.

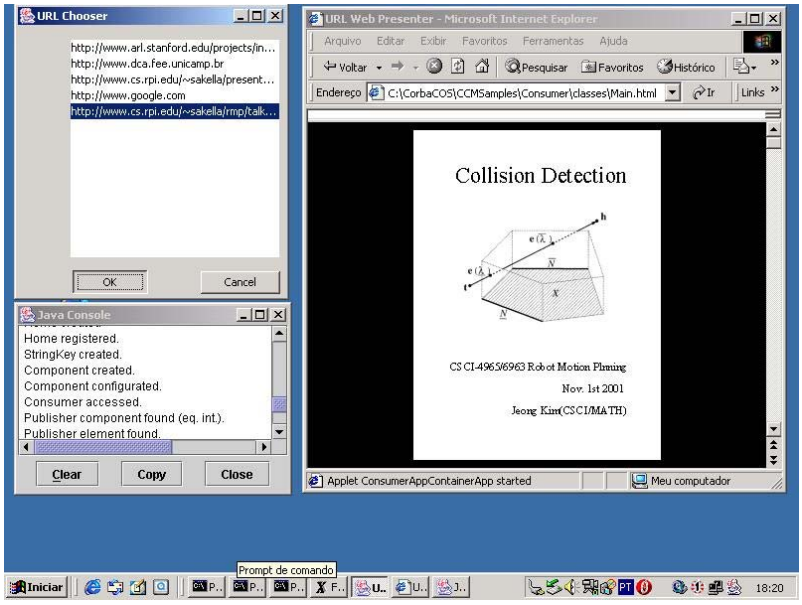
Uma câmera panorâmica permite que o estudante remoto inspecione a circulação de ar (marcada pelas fitas dentro do modelo), tire fotografias, e grave um vídeo de partes do experimento. Os métodos expostos pela interface WSDL do Serviço Web da câmera panorâmica executam tais operações.

O controle de janelas e portas no modelo são remotamente controlados em conjuntos pré-configurados, assim como o acionamento do ventilador (fluxo de ar). Um novo módulo, denominado módulo de servo-posicionamento, controla até oito servos de posicionamento e interruptores. Este módulo também expõe uma interface WSDL, permitindo ao cliente a seleção de um servo ou uma chave e a operação sobre este.

A figura 1.5 mostra a interface do usuário para o experimento do REAL-CT WebLab. Os *frames* à esquerda permitem o controle de posição do modelo por meio de rotações do robô e a seleção de combinações de portas e janelas, bem como ligar e desligar o ventilador. O *frame* à direita permite ao



(a) Interface Gráfica com Campos para *Chat* e Botões do Modo Assistido no REAL WebLab



(b) Interface Gráfica para Apresentação de Transparência da *URL* Especificada no Modo Assistido

Fig. 1.3: Interfaces Gráficas do Modo Assistido no REAL WebLab

usuário operar a câmera panorâmica, controlando seu *pan/tilt* (eixo X-Y), aumentando e diminuindo seu *zoom*, gravando figuras e vídeos.

Fig. 1.4: Modo Básico em Dispositivo Móvel usando Plataforma MECM-tel

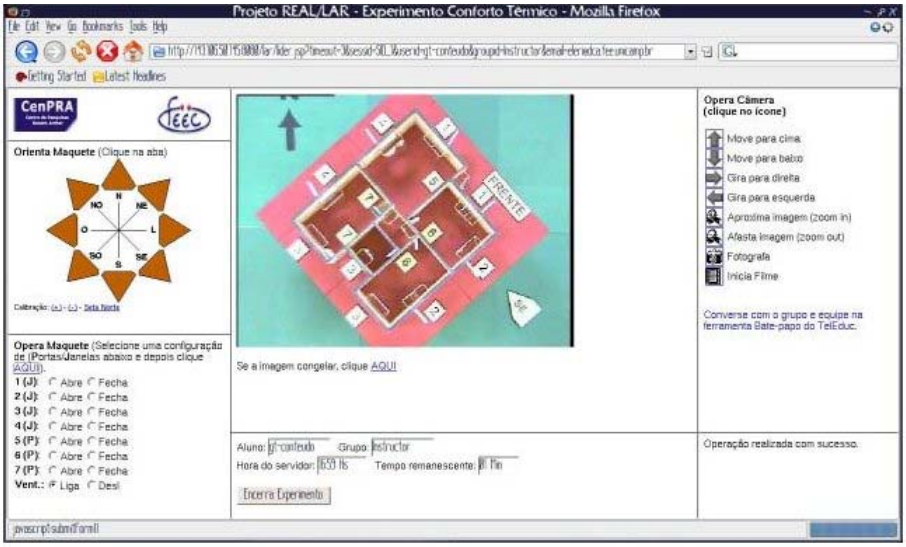


Fig. 1.5: Interface do Experimento do REAL-CT WebLab

1.4.3 Projeto GigaBOT

O projeto ”GigaBOT - Laboratórios de Acesso Remoto Sobre Redes Avançadas” é um projeto de estudo em Aplicações Multimídia de Tempo Real em Redes Avançadas (Serviços e Aplicações Científicas) e Gerenciamento de Redes Avançadas (Protocolos e Serviços de Rede). Foi proposto pela Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas (Unicamp) por meio da Fundação de Desenvolvimento da Unicamp (FUNCAMP) e é coordenado pelo Prof. Dr. Eleri Cardozo. Apresenta ainda como instituições co-executoras o Instituto de Computação (IC) da Unicamp, a Universidade Federal do Rio de Janeiro (UFRJ), o Centro de Pesquisas Renato Archer (CenPRA), o Departamento de Engenharia Elétrica da Faculdade

de Engenharia da Pontifícia Universidade Católica do Rio Grande do Sul (PUC/RS) e a Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia (UFU).

O objetivo principal do projeto é o desenvolvimento de uma plataforma para suporte a aplicações multimídia de tempo real para redes avançadas, uma aplicação de laboratório de acesso remoto para robótica e soluções de gerência integrada de redes e aplicações.

Para demonstrar a viabilidade da proposta, este projeto irá disponibilizar na rede do projeto GIGA um laboratório de acesso remoto que permitirá explorar o controle de robôs (ou outros dispositivos) à distância. Com a Internet atual, apenas a tele-operação e o monitoramento remoto são factíveis no campo da robótica. O controle descentralizado é inviável dado o indeterminismo da rede de comunicação e a baixa largura de banda para o tráfego de grandes volumes de informação (por exemplo, imagens captadas pelo robô em tempo real). Com as redes óticas e infra-estrutura de software adequada será possível descentralizar o controle, ou seja, distribuir o processamento do controle por meio de vários nós da rede. Isto viabiliza, por exemplo, o compartilhamento de recursos de custo elevado por parte de empresas e centros de pesquisa. Em outras palavras, viabiliza a formação de federações para o uso de recursos em torno de uma rede de alto desempenho.

1.4.4 Créditos

Este trabalho não seria possível sem a grande contribuição dos participantes dos projetos REAL e GigaBOT.

O modelo conceitual para WebLabs foi desenvolvido no âmbito do projeto Tidia-Ae [20] (Tecnologia da Informação no Desenvolvimento da Internet Avançada - Aprendizado Eletrônico) pela co-orientadora e pelo orientador.

O desenvolvimento do Serviço de Difusão (seção 4.1.3) deve-se ao aluno Rodrigo Fernandes Sassi, serviço este que faz parte de seu trabalho de mestrado.

O Serviço de Acesso (seção 4.1.1) e Interfaces de Administração de WebLabs foram desenvolvidos pelo orientador, pela co-orientadora e pelo Prof. Luís Faina (UFU) .

Os alunos Alex Z. Lima, Victor Perticarrari, Daniele dos Santos e Ewerton Barbosa contribuíram na codificação de três experimentos do WebLab apresentado no capítulo 5.

1.5 Organização do Texto

O texto está dividido em cinco capítulos. O Capítulo 2 apresenta os principais conceitos pertinentes ao trabalho desenvolvido e define termos como Serviço Web, SOA (*Service Oriented Architecture*), Composição e BPEL (*Business Process Execution Language*).

O Capítulo 3 traz definições do modelo conceitual de Laboratório de Acesso Remoto e apresenta uma arquitetura baseada em SOA para o mesmo. A implementação desta arquitetura é descrita no Capítulo 4, em que são apresentados os serviços de acesso, de interação e de comunicação desenvolvidos, bem como a infra-estrutura de hardware e software utilizada.

Um WebLab utilizando a arquitetura implementada é descrito no Capítulo 5, onde são apresentadas as interfaces de interação e os mecanismos de composição de serviços, tanto de forma "*ad-hoc*" como por meio da linguagem BPEL. O capítulo apresenta ainda exemplos para cada composição desenvolvida.

Por fim, as conclusões são tratadas no Capítulo 6. Neste capítulo é fornecido um panorama geral do trabalho desenvolvido, os possíveis aprimoramentos imediatos e propostas para continuidade do trabalho.

Capítulo 2

Serviços Web e Orquestração

Neste capítulo serão apresentados os conceitos fundamentais para desenvolvimento de aplicações que utilizem este paradigma de Computação Orientada a Serviço.

Serviços provêm alto nível de abstração para organizar aplicações de larga escala e ambientes abertos. Desta forma, eles auxiliam a implementação e configuração de aplicações de tal modo que aumentam sua qualidade e a produtividade. Estas aplicações utilizam os serviços por meio de composição de serviços. Uma arquitetura para aplicações baseadas em serviços é composta de três partes principais: um provedor, um consumidor e um registro de serviços (figura 2.1). Provedores publicam seus serviços em registros, onde consumidores os encontram e os utilizam.

2.1 Computação Orientada a Serviço

Computação Orientada a Serviço, ou COS, é o paradigma que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações/soluções [21]. Serviços são elementos computacionais auto-descritíveis e independentes de plataforma que suportam uma composição rápida e de baixo custo de aplicações distribuídas. Serviços realizam operações que vão desde simples requisições a complexos processos de negócios. Serviços permitem a organizações a exposição de suas competências básicas na Internet (ou Intranet) usando linguagens e protocolos padrões (baseados em XML) e a implementação por meio de interfaces descritas utilizando padrões abertos.

Uma vez que serviços provêm uma distribuição de informação uniforme e ubíqua para um grande número de dispositivos computacionais (como *handhelds*, PDAs, telefones celulares) e plataformas (como UNIX e Windows), eles constituem o próximo passo em computação distribuída.

Serviços são oferecidos por provedores de serviços [22], os quais fornecem a descrição do serviço, bem como suporte técnico e de negócio. Desta forma, provê-se uma infra-estrutura de computação distribuída para integração e colaboração de aplicações intra-domínio e inter-domínio [23]. Os clientes desses serviços podem também ser soluções ou aplicações intra-domínio ou inter-domínio. Conseqüentemente, para satisfazer estes requerimentos os serviços devem possuir [24]:

- Neutralidade de tecnologia: devem ser invocáveis por meio das diversas tecnologias disponíveis. Isto implica que os mecanismos de invocação (protocolos, descrições e mecanismos de descoberta) devem utilizar padrões largamente aceitos.

- Fraco acoplamento: alcançado em diferente níveis:
  - Apenas as interfaces expostas pelos serviços são utilizadas nos mecanismos de composição de serviços.
  - Com o uso de mecanismos de registro e descoberta desacopla-se o uso da localização do serviço.
  - Acoplamento à linguagem e plataforma pode ser evitado usando um protocolo de transporte independente de plataforma.
  - Acoplamento devido ao estilo de invocação (requisição-resposta) pode ser atenuado utilizando troca de mensagens assíncronas.
- Transparência de localização: serviços devem ter sua informação de definições e localização armazenadas em um repositório e ser acessíveis por uma variedade de clientes que podem localizá-los e invocá-los independente da localização.
- Capacidade de composição: serviços podem ser compostos para formar outro serviço, provendo um modo de criação de novos serviços flexível, rápido e de baixo custo.
- Ubiqüidade: serviços podem ser acessados por meio da Internet de qualquer lugar a qualquer momento.
- Maior granularidade: quando comparados a objetos e componentes, o que facilita o reuso de serviços.

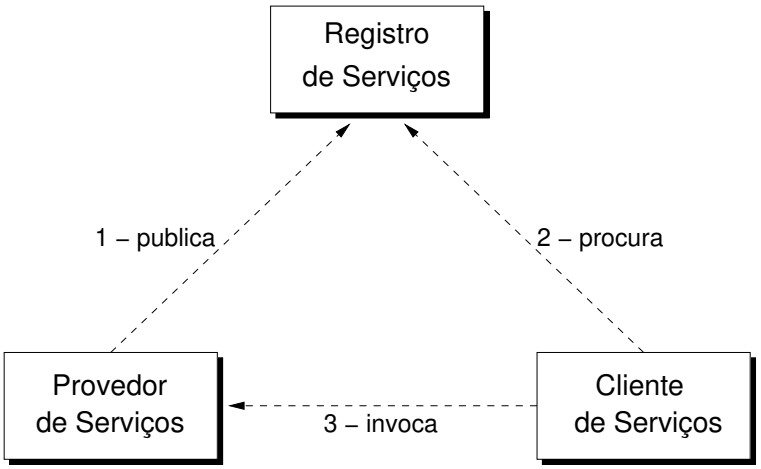


Fig. 2.1: Cenário de utilização de serviços

Existem duas classificações de serviços: simples e compostos. Serviços compostos envolvem a utilização de serviços existentes de maneira a acessar e combinar informações e funções de múltiplos provedores de serviço.

Aplicações baseadas em serviços são desenvolvidas como conjuntos independentes de serviços que interagem entre si, oferecendo interfaces bem definidas para seus usuários. Isto é alcançado sem a necessidade de um forte acoplamento de aplicações distribuídas entre parceiros, e sem o requerimento de acordos pré-determinados antes do uso de um serviço oferecido.

2.2 Serviços Web

Uma vez que os serviços encapsulam a funcionalidade do negócio, alguma forma de infra-estrutura intra-serviço é necessária para facilitar suas interações e comunicações. Diferentes formas desta infra-estrutura são possíveis, porque serviços podem ser implementados em uma única máquina, distribuídos por meio de um conjunto de computadores em uma rede local, ou distribuídos mais amplamente por meio de muitas redes. Um caso particularmente interessante ocorre quando serviços utilizam a Internet como meio de comunicação e padrões abertos. Um Serviço Web, ou *Web Service*, é um tipo específico de serviço que é identificado por uma URI (*Unified Resource Identifier*) e apresenta as seguintes características:

- Expõe seus recursos programaticamente por meio da Internet usando linguagens e protocolos padrões da Internet, e
- Pode ser implementado por meio de uma interface auto-descritiva baseada em padrões abertos da Internet.

Os protagonistas no desenvolvimento de Serviços Web foram IBM e Microsoft. A criação de linguagem XML e a aceitação como padrão pela W3C (*World Wide Web Consortium*) em 1998 abriram caminho para o desenvolvimento de Serviços Web com trabalhos no sentido de criar uma comunicação serviço-a-serviço padronizada.

Uma vez desenvolvido o protocolo básico, o próximo passo foi o acordo na criação de um protocolo padronizado para troca de mensagem baseado em XML. A Microsoft desenvolveu o protocolo SOAP (*Simple Object Access Protocol*) - flexível, independente de plataforma e de propósito geral. IBM adotou o protocolo e passou a trabalhar na versão 1.1. Ao mesmo tempo, IBM e Microsoft iniciavam trabalhos no sentido de obter um método de descrever programaticamente como se conectar a um Serviço Web. Desse esforço surgiu a linguagem WSDL (*Web Service Description Language*).

Com SOAP e WSDL, passou a ser possível a criação e descrição de Serviços Web. A partir daí, IBM, Microsoft e Ariba passaram a se empenhar na tarefa de prover mecanismos de descoberta de Serviços Web. Assim, foi apresenta a versão 1.0 do UDDI (*Universal Description, Discovery and Integration*).

Definidos os protocolos para troca de mensagem (SOAP), descrição (WSDL) e descoberta (UDDI), as maiores empresas de *software* como Oracle, HP, Sun, IBM, BEA e Microsoft, entre outras, passaram a demonstrar interesse no suporte e instalação de Serviços Web em seus produtos.

2.2.1 Tecnologias utilizadas em Serviços Web

Como descrito na seção 2.2, um Serviço Web é capaz de descrever a si mesmo, ser localizado e, por fim, ser invocado. Sua descrição é provida pela interface WSDL, sua localização, pelo serviço UDDI e suas funcionalidades são invocadas pelo uso do protocolo SOAP, sendo que estas três tecnologias são construídas fundamentalmente sobre o mesmo padrão de descrição - XML.

```
1 <pessoa>
2 <nome>Paulo</nome>
3 <endereço>
4   <rua>Rua da Paz</rua>
5   <número>500</número>
6   <cidade>Campinas</cidade>
7 </endereço>
8 </pessoa>
```

Fig. 2.2: Um exemplo de XML

XML - *eXtensible Markup Language*

XML [25] é uma especificação padrão para definição de conteúdo de informação computacional. XML possibilita o entendimento entre quem gerou e quem interpreta a mensagem por meio do uso de marcações (ou *tags*). As marcações 'delimitam' o conteúdo.

A figura 2.2 mostra um fragmento de um documento XML de exemplo em que se pode verificar a importância das marcações na especificação do conteúdo da informação.

A importância de XML para a computação pode ser comparada à importância da linguagem natural para os humanos, uma vez que ela define uma forma, uma estrutura, uma sintaxe e semântica para que computadores se comuniquem. Se duas entidades usam diferentes linguagens, o único meio de comunicação entre elas é por meio de uma tradução entre as duas linguagens. Em computação, o uso de uma linguagem comum pode reduzir drasticamente o custo de desenvolvimento e manutenção, bem como aumentar a performance [26], [27].

Muitas tentativas de desenvolver essa linguagem padrão ocorreram anteriormente, mas não obtiveram êxito. Entretanto, o sucesso da Internet e o uso de HTML como uma linguagem comum encorajou a rápida aceitação e adoção do XML como essa linguagem.

SOAP - *Simple Object Access Protocol*

Com XML, tem-se um maneira comum de representação do conteúdo de informação enviada entre um componente de *software* e outro. Dessa forma, SOAP [28] permite um componente de *software* invocar as funcionalidades de outro componente, usando mensagens trocadas entre os dois como mecanismo de invocação.

O protocolo SOAP inclui [29]:

- Um "envelope" que define um *framework* para descrever o conteúdo da mensagem e como processá-la.
- Um conjunto de regras de codificação para expressar instâncias de tipos de dados dentro da mensagem.
- Uma convenção para representar a maneira na qual os procedimentos (ou métodos) do componente de *software* pode ser chamado, e suas respostas.
- Uma convenção de ligação para troca de mensagem utilizando um protocolo de comunicação.

A versão 1.1 da especificação do protocolo SOAP foi proposta em 2000 pelo W3C e inclui mapeamento para o protocolo HTTP. Apesar de mensagens poderem ser transferidas utilizando outros protocolos, a escolha do HTTP reflete a intenção de tornar o protocolo disponível na Internet. Em 2003 foi lançada a versão 1.2.

SOAP utiliza um mecanismo de requisição-resposta, em que um componente de *software* faz uma requisição para outro que então provê uma resposta. Ambas requisição e resposta são transportadas na forma de documentos XML.

```
1  POST /StockQuote HTTP/1.1
2  Host:www.stockquoteserver.com
3  Content-Type: text/xml; charset=''utf-8''
4  Content-Length: nnnn
5  SOAP Action: ''Some-URI''
6
7  <SOAP-ENV:Envelope
8    xmlns:SOAP-ENV=``http://schemas.xmlsoap.org/soap/envelope/''
9    SOAP-ENV:encodingStyle=``http://schemas.xmlsoap.org/soap/encoding/''>
10   <SOAP-ENV:Body>
11     <m:GetLastTradePrice xmlns:m=``Some-URI''>
12       <symbol>STOCKSYMBOL</symbol>
13     </m:GetLastTradePrice>
14   </SOAP-ENV:Body>
15 </SOAP-ENV:Envelope>
```

Fig. 2.3: Um exemplo de requisição SOAP

Um exemplo de uma requisição SOAP pode ser visto na figura 2.3. Observa-se que a mensagem é enviada por meio de uma requisição HTTP POST. As linhas 7 a 15 definem o envelope SOAP, que contém as partes principais do protocolo - definindo o *framework* para troca de mensagens, os tipos de dados e os procedimentos (métodos). A linha 8 e 9 mostram as localizações de *namespace* e regras de codificação. As linhas de 10 a 14 definem o corpo (*BODY*) da requisição SOAP, no qual se encontra o método e seus parâmetros.

Assim nota-se que, excluindo-se o cabeçalho, a requisição SOAP consiste de uma chamada direta a um método e, uma vez que é baseada em XML, pode ser facilmente lida tanto por computadores quanto pelo homem. Além disso, a mensagem é embutida na requisição HTTP, o que a permite ser transportada por meio da Internet. A mensagem de resposta SOAP tem um formato similar, como pode ser visto na figura 2.4. A diferença principal reside na mensagem ser agora uma resposta HTTP e no corpo da mensagem SOAP conter a resposta à requisição.

SOAP provê um simples, mas poderoso meio de um componente de *software* invocar ações em outros por meio de interações de mensagens. Todos as principais empresas vendedoras de plataformas de *software* concordaram em implementar a especificação do SOAP de acordo com a W3C. Com isso, um componente de *software* instalado em uma plataforma pode se comunicar com outro em qualquer outro tipo de plataforma.

WSDL - *Web Service Description Language*

HTTP, XML e SOAP provêm mecanismos para que um componente de *software* invoque a funcionalidade de outro por meio da Internet. De fato, essas tecnologias podem ser usadas com

```
1  HTTP/1.1 200 OK
2  Content-Type:text/xml; charset=''utf-8''
3  Content-Length: nnnn
4
5  <SOAP-ENV:Envelope
6    xmlns:SOAP-ENV=''http://schemas.xmlsoap.org/soap/envelope/''
7    SOAP-ENV:encodingStyle=''http://schemas.xmlsoap.org/soap.encoding/''/>
8    <SOAP-ENV:Body>
9      <m:GetLastTradePriceResponse xmlns:m=''Some-URI''>
10       <Price>34.5</Price>
11     </m:GetLastTradePriceResponse>
12   </SOAP-ENV:Body>
13 </SOAP-ENV:Envelope>
```

Fig. 2.4: Um exemplo de resposta SOAP

tal finalidade em qualquer rede que suporte o protocolo HTTP. Para tirar proveito desta integração, são também necessários:

- Informações sobre todas as funções (métodos) disponíveis, incluindo seus parâmetros.
- Informações sobre tipos de dado para todas as mensagens XML, incluindo as especificações de valores.
- Informação sobre o protocolo de transporte específico a ser usado.
- Informação sobre endereço para a localização serviço especificado.

Um documento WSDL [30] é um documento XML que fornece as informações listadas acima a respeito da um componente de *software*[29]. Assim, usando este documento pode-se interagir com qualquer das funcionalidades disponíveis de um Serviço Web. Com ferramentas específicas, este processo pode ser inteiramente automatizado, habilitando aplicações a facilmente se integrarem a novos Serviços Web com muito pouco ou nenhum código manual.

Um exemplo de WSDL é listado na figura 2.5. A primeira seção (linhas 1 a 7) define os vários espaços de nomes requeridos, incluindo a URL (*Unified Resource Locator*) do próprio documento WSDL. A segunda seção (linhas 8 a 13) define as mensagens usadas como parâmetros de entrada e saída. A terceira seção (linhas 14 a 19) define a operação (o método ou procedimento), que é encapsulado como um *port type*. A quarta seção (linhas 20 a 36) declara as regras de codificação para as entradas e saídas para o *port type*, colocando-o como uma operação (*operation*). A última seção (linhas 37 a 44) declara o serviço, ligando-o a um *port type*.

Do exemplo pode-se observar que a linguagem WSDL fornece descrição suficiente de um Serviço Web, de tal forma que uma aplicação pode interpretá-la e determinar as informações necessárias para sua invocação.

UDDI - *Universal Description, Discovery and Integration*

Se a localização do documento WSDL que descreve um serviço desejado é conhecida, pode-se simplesmente incluir esta localização no *software* de desenvolvimento e implementar a integração

```
<?xml version=''1.0'' encoding="UTF-8"?>
<definitions name="WeatherService"tions name="WeatherService"
  targetNamespace="http://www.townweather.com/wsdl/WeatherService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.townweather.com/wsdl/WeatherService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="getWeatherRequest">
    <part name="town" type="xsd:string"/>
  </message>
  <message name="getWeatherResponse">
    <part name="temperature" type="xsd:int"/>
  </message>
  <portType name="WeatherPortType">
    <operation name="getWeather">
      <input message="tns:getWeatherRequest"/>
      <output message="tns:getWeatherResponse"/>
    </operation>
  </portType>
  <binding name="WeatherBinding" type="tns:WeatherPortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getWeather">
      <soap:operation soapAction=""/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap.encoding/"
          namespace="urn:examples:weatherservice" use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap.encoding/"
          namespace="urn:examples:weatherservice" use="encoded"/>
      </output>
    </operation>
  </binding>
  <service name="WeatherService">
    <documentation>WSDL File for Weather Service</documentation>
    <port binding="tns:WeatherBinding" name="WeatherPort">
      <soap:address
        location="http://localhost:8080/soap.servlet.rpcrouter"/>
    </port>
  </service>
</definitions>
```

Fig. 2.5: Um exemplo de arquivo WSDL

deste Serviço. Entretanto, se é necessário criar uma aplicação, levando-se em conta disponibilidade, performance e outros atributos não-funcionais, e deseja-se comparar ou testar componentes de *software* fornecidos por diferentes companhias, UDDI [31] provê extensão para as tecnologias básicas de Serviços Web, permitindo a criação de um registro de Serviços Web [29]. Isto possibilita que companhias façam negócios entre si por meio da Internet, A especificação do UDDI permite rápida, fácil e dinâmica localização e interação entre serviços dessas companhias.

UDDI traz a possibilidade de:

- Descrever seus negócios e serviços.
- Descobrir outros fornecedores de serviços
- Integrar-se com estes outros serviços.

As especificações para UDDI permitem a criação e uso de um registro contendo informação sobre negócios e serviços que eles oferecem. A informação é organizada da seguinte forma:

Entidade de Negócio (*Business Entity*): Uma entidade de negócio representa informação sobre uma companhia. Cada entidade de negócio contém um identificador único, o nome da companhia, um breve descrição da companhia, algumas informações básicas de contato, uma lista de categorias e identificadores que descrevem a companhia, e uma URL apontando para mais informações sobre a companhia.

Serviço de Negócio (*Business Service*): Associado com uma entidade de negócio está uma lista de *business services* oferecidos por ela. Cada entrada nesta lista contém uma descrição do serviço, uma lista de categorias que descrevem o serviço, e uma lista de apontadores para referências e informações relacionadas ao serviço.

Ponteiros de Especificação: Associada com cada serviço de negócio está uma lista de *binding templates* que apontam para especificações e outras informações técnicas sobre o serviço. Por exemplo, um *binding template* pode apontar para uma URL que fornece informações sobre como invocar o serviço. Também é possível usar estes apontadores para acessar SLAs (*Service Level Agreements*) que descrevem a natureza contratual do uso do serviço. Os ponteiros de especificação também associam o serviço com um tipo de serviço.

Tipos de Serviço: Um tipo de serviço é definido por um *tModel*. Um *tModel* especifica informações tais como o nome do Serviço, o nome da organização que o publicou, a lista de categorias que descrevem o tipo de serviço e ponteiros para especificações técnicas para o tipo de serviço, entre elas: definições de interface, formatos de mensagens e protocolos de segurança.

É possível ainda o uso de UDDI para criar registros privados que residem em redes privadas, oferecendo funcionalidade para um conjunto específico de usuários. As especificações para UDDI são administradas pela OASIS (*Organization for the Advancement of Structured Information Services*) e, apesar de disponíveis desde 1999, seu uso para localização de Serviços Web se mantém relativamente imaturo. Além dos registros UDDI, existem ainda listagens publicamente disponíveis de Serviços Web, por exemplo XMethods [32] e BindingPoint [33].

Integração das tecnologias

As tecnologias descritas anteriormente provêm mecanismo para que o *software* descreva a si próprio, seja descoberto por outro *software* e possa ser invocado por meio da rede. A indústria de Tecnologia da Informação abraçou as especificações para estas tecnologias de tal forma que o uso de Serviços Web foi largamente disseminado.

Com relação a serviços na Internet, até recentemente temos apenas pequenas ilhas de crescimento (por exemplo, serviços bancários, serviços de viagem, etc), mas muito pouca tecnologia comum

entre eles. Consumidores ainda preferem tecnologias mais antigas, tal como formulários HTML. Tecnologias de Serviços Web fornecem a padronização que permitirá fácil interoperação entre serviços oferecidos na Internet.

2.3 Composição

Na criação de aplicações que utilizam o paradigma da Computação Orientada a Serviços, torna-se cada vez mais comum o emprego da composição de serviços. Desenvolvedores e usuários podem, via composição, resolver problemas complexos por meio da combinação de serviços básicos disponíveis, ordenando-os de forma a obter um melhor ajuste para os requerimentos de suas soluções. A composição de serviços acelera o desenvolvimento de aplicações, o reuso de serviços e a implementação de novos serviços mais complexos.

A composição de serviços, utilizada na criação de processos de negócios, pode ser descrita sob dois aspectos: orquestração e coreografia. A figura 2.6 ilustra a relação entre esses aspectos em um nível mais alto.

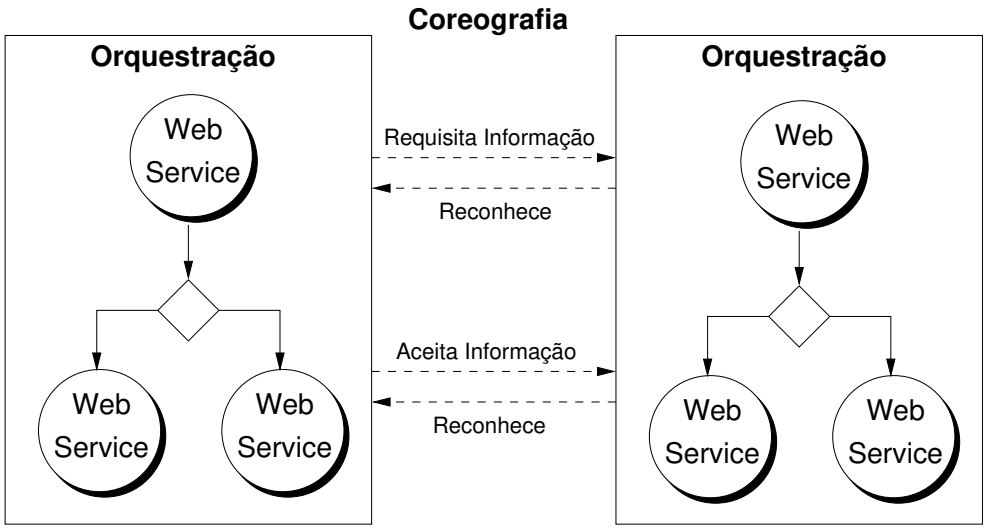


Fig. 2.6: Orquestração e Coreografia

Orquestração refere-se a um processo de negócio executável que pode interagir com Serviços Web tanto internos quanto externos. As interações ocorrem em nível de mensagens e incluem a lógica do negócio e a ordem de execução das tarefas, podendo abranger aplicações e organizações em um modelo de processo transacional, de longa duração e muitos passos. A orquestração sempre representa o controle da perspectiva de uma parte envolvida. Isto difere da coreografia, a qual é mais colaborativa e permite que cada parte envolvida descreva sua própria parte na interação. Dessa forma, a coreografia age sobre as seqüências de mensagens entre múltiplas partes e fontes - tipicamente as trocas de mensagens públicas entre Serviços Web - e não sobre um processo de negócio específico definido por uma única parte.

2.3.1 Requerimentos da Composição de Serviço

As complexidades de sistemas distribuídos e as crescentes barreiras de segurança têm influenciado a evolução de COS nos níveis de *hardware*, sistemas operacionais e aplicações. Na perspectiva do desenvolvedor, a composição de serviços provê possibilidades de reuso. Na perspectiva do usuário, oferece acesso fácil e transparente a uma variedade de serviços complexos.

Requerimentos para a composição de serviços diferem daqueles necessários ao desenvolvimento de software baseado em componentes: em contraposição ao acesso a documentação ou código (binário ou fonte), tanto desenvolvedores quanto usuários de aplicações COS têm acesso apenas às descrições funcionais providas pela linguagem WSDL. Serviços são executados em diferentes contêineres, separados por *firewalls* e outras barreiras de segurança. Um mecanismo de composição deve, por isso, satisfazer diversos requerimentos: conectividade, propriedades não-funcionais de qualidade de serviço, corretude e escalabilidade.

Primeiramente, a invocação de serviços de forma assíncrona é vital para que seja atingida a segurança e escalabilidade que os ambientes de Tecnologia da Informação atuais requerem. A possibilidade de se invocar serviços concorrentemente contribuem também para aumentar a performance do processo. A implementação de Serviços Web assíncronos requer um mecanismo capaz de correlacionar requisições e respostas de cada um. Isto geralmente é conseguido com a utilização de identificadores de correlação.

Um outro requerimento diz respeito à capacidade de gerenciar exceções e integridade transacional. Em adição ao tratamento de erros e restrições de tempo, Serviços Web orquestrados devem garantir disponibilidade de recursos para transações distribuídas de longa duração. Transações tradicionais tipo ACID (atomicidade, consistência, isolamento e durabilidade) não são suficientes para transações distribuídas de longa duração, uma vez que não podem "travar" recursos em uma transação que dura um grande período de tempo. A noção de transações de compensação oferece uma maneira de desfazer uma ação caso um processo ou usuário a cancele. Por meio de transações de compensação, cada método expõe uma operação de *undo* (desfazer) que o coordenador da transação pode invocar se necessário [34].

A orquestração de Serviços Web deve ser dinâmica, flexível e adaptável de forma a atender mudanças nas necessidades do negócio. Uma separação clara entre a lógica do processo (conhecido como processo de negócio) e os Serviços Web utilizados promovem essa flexibilidade. Uma máquina de orquestração pode geralmente alcançar esta separação. A máquina manipula o fluxo do processo, chamando os Serviços Web apropriados e determinando que passos completar (figura 2.7).

Por fim, deve ser possível compor serviços de nível mais alto a partir de processos orquestrados existentes. A exposição desses processos por meio de suas interfaces de Serviços Web cumpre o objetivo deste combinação recursiva.

2.4 BPEL - *Business Process Execution Language*

Microsoft, IBM, Siebel Systems, BEA e SAP foram co-autores da versão 1.1 da especificação BPEL4WS [35] (*Business Process Execution Language for Web Service*) lançada em maio de 2003. A especificação - chamada também de BPEL - modela o comportamento de Serviços Web em uma interação de processo de negócio. Ela provê uma gramática baseada em XML para descrever a lógica de controle requerida para coordenar Serviços Web participantes do fluxo de um processo. Uma

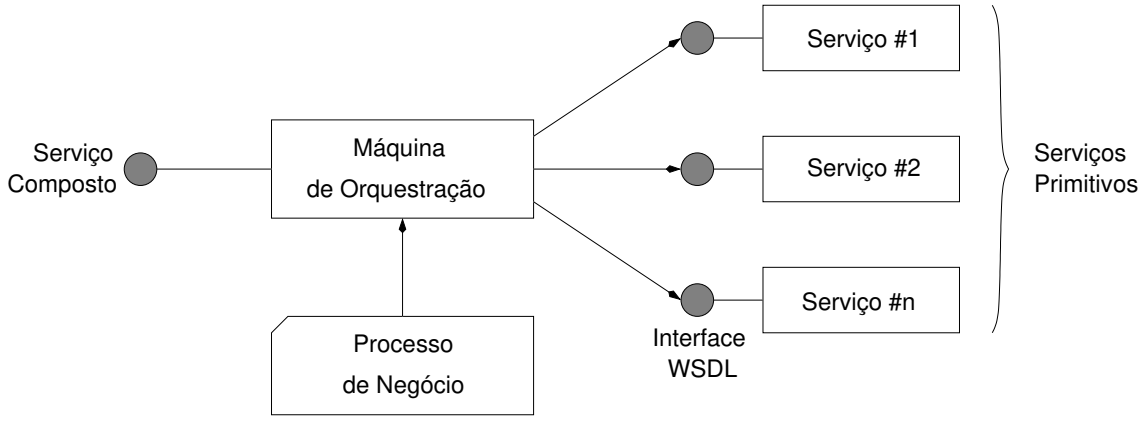


Fig. 2.7: Composição de serviço

máquina de orquestração pode então executar esta gramática, coordenando atividades e compensando o processo como um todo caso erros ocorram.

As diversas camadas que compõe Serviços Web e composição podem ser vistas na figura 2.8, na qual é possível notar que BPEL é uma camada sobre WSDL. A interface WSDL define as operações permitidas e BPEL define como seqüenciá-las. WSDL também descreve os pontos de entrada e saída públicos para todo processo BPEL. Além disso, os tipos de dados descritos na interface WSDL, fornecem a informação que é passada entre requisições de processos. A interface WSDL pode ainda referenciar serviços externos de que o processo BPEL necessita.

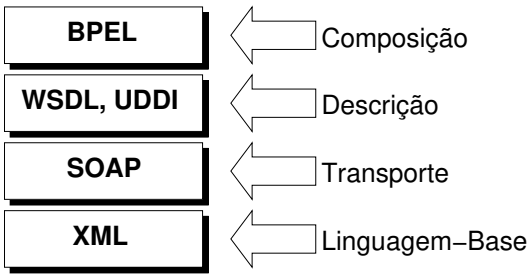


Fig. 2.8: Camadas componentes de Serviços Web

BPEL provê suporte para processos de negócio abstratos e executáveis. Um processo executável modela o comportamento de participantes em uma interação de negócio específica, essencialmente modelando uma fluxo de trabalho privado. Um processo abstrato ou protocolo de negócio especifica as trocas de mensagens públicas entre as partes. Protocolos de negócio não são executáveis e transmitem os detalhes internos do fluxo do processo. Essencialmente, processos executáveis modelam orquestração, enquanto processos abstratos modelam a coreografia de serviços.

A especificação do BPEL inclui suporte tanto para atividades básicas quanto estruturadas. Uma atividade básica é uma instrução que interage com algo externo ao próprio processo. Por exemplo, atividades básicas manipulariam ações como receber, responder ou invocar chamadas de Serviços Web. Em um cenário típico, um processo executável BPEL recebe uma mensagem. Em conseqüência, ele pode invocar uma série de serviços para colher dados adicionais e subseqüentemente responder

ao requisitante. Na figura 2.9 [34], as mensagens de invocação, recebimento e resposta representam atividades básicas do processo [36].

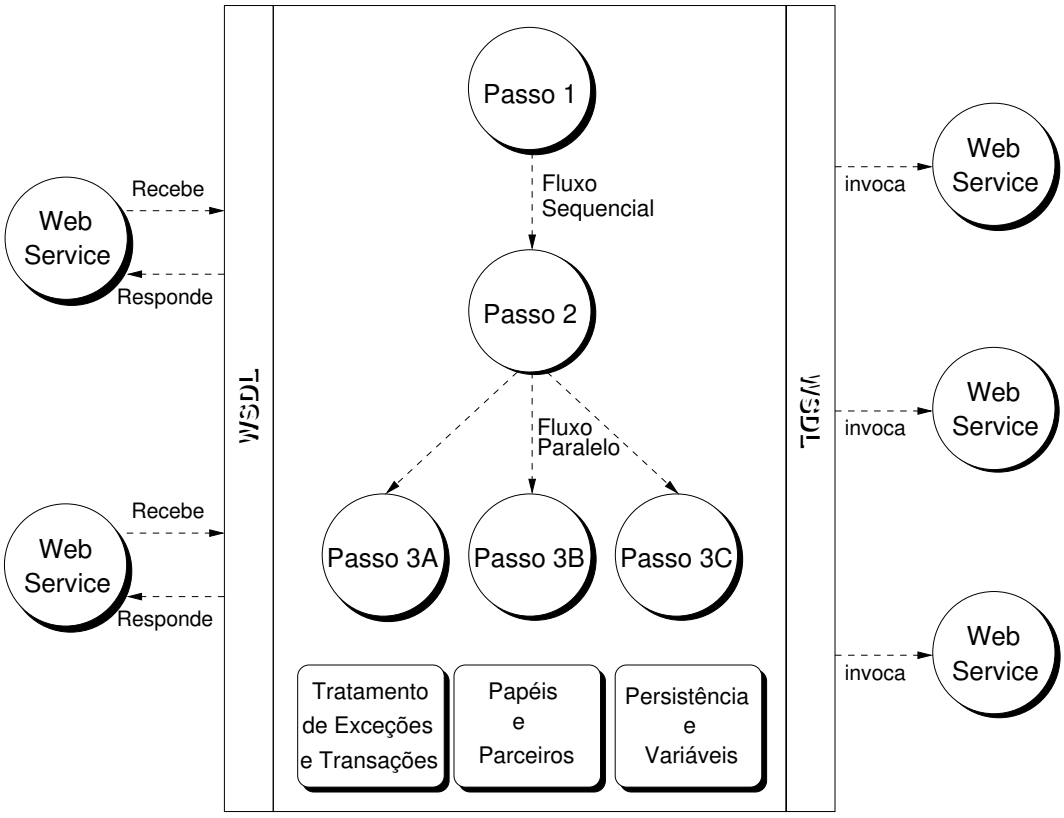


Fig. 2.9: Fluxo de um processo BPEL4WS, ou BPEL

Atividades estruturadas gerenciam o fluxo do processo como um todo, especificando a seqüência para Serviços Web referenciados. Estas atividades também suportam laços e condicionais, que formam essencialmente a lógica de programação para a linguagem BPEL [37].

Variáveis e parceiros (*partnerLinks*) são outros dois elementos importantes que compõem a linguagem BPEL:

- Uma variável identifica um dado específico trocado em um fluxo de mensagem. Quando um processo BPEL recebe uma mensagem, ele atualiza a variável apropriada de forma que as requisições subseqüentes possam acessar o dado. Variáveis são, assim, usadas para gerenciar a persistência de dados (estados) por meio das diversas requisições de Serviços Web.

- Um parceiro pode ser qualquer serviço que o processo invoca ou qualquer serviço que o invoca. Cada parceiro é mapeado para um papel específico que ele desempenha no processo.

BPEL também provê um robusto mecanismo para tratamento de transações e exceções [36], [37], o que é tratado nas especificações de *WS-Coordination* e *WS-Transaction*. Desenvolvido juntamente por Microsoft, IBM e BEA, estas especificações provêm o suporte necessário para gerenciar e coordenar as operações de atividade do negócio.

Para agrupar um conjunto de atividades em uma única transação, BPEL usa a marcação *scope* (escopo). Esta marcação significa que os passos nela contidos devem ser completados com sucesso ou falharem em sua totalidade. Dentro deste escopo pode-se especificar mecanismos de compensação que a máquina de orquestração pode invocar em caso de erro [36].

Para o tratamento de exceções, BPEL utiliza, de forma similar à linguagem de programação Java, as cláusulas *throw* e *catch* [36].

2.5 Considerações Finais

Este capítulo apresentou a variedade de tecnologias envolvidas no âmbito desta dissertação. A principal vantagem das linguagens e tecnologias destacadas é a independência de plataforma e linguagem de programação, resultado da neutralidade própria do XML. No capítulo seguinte é introduzido o modelo de WebLab, seguido da descrição das três sessões que compõem o modelo.

Capítulo 3

Uma Arquitetura para WebLab Baseada em SOA

Este capítulo descreve o modelo conceitual que define o Laboratório de Acesso Remoto a ser construído e fornece uma arquitetura para o desenvolvimento do laboratório, destacando os elementos importantes para o funcionamento e construção do laboratório.

3.1 Modelo Conceitual de WebLab

O modelo conceitual foi proposto inicialmente no projeto TIDIA-Ae [20]. A figura 3.1 ilustra os elementos que compõem um WebLab, bem como as relações entre os elementos. Um *participante* pode se constituir de um *usuário* individual, um *grupo* de usuários ou (recursivamente) de participantes. Um WebLab agrega *experimentos* e a relação de um WebLab para com si próprio indica que WebLabs podem ser federados para aumentar a gama de experimentos oferecidos e/ou para beneficiar a uma gama maior de usuários. Experimentos são atividades executadas a distância que utilizam *serviços* como elementos de constituição. Serviços disponibilizam tanto funcionalidades específicas como manipulação de robôs e câmeras quanto funcionalidades de propósito geral como controle de acesso e comunicação inter-pessoal. Serviços fazem uso de *recursos*, tanto físicos (robôs, câmeras, etc.) quanto lógicos (pacotes de simulação, de visualização, etc.).

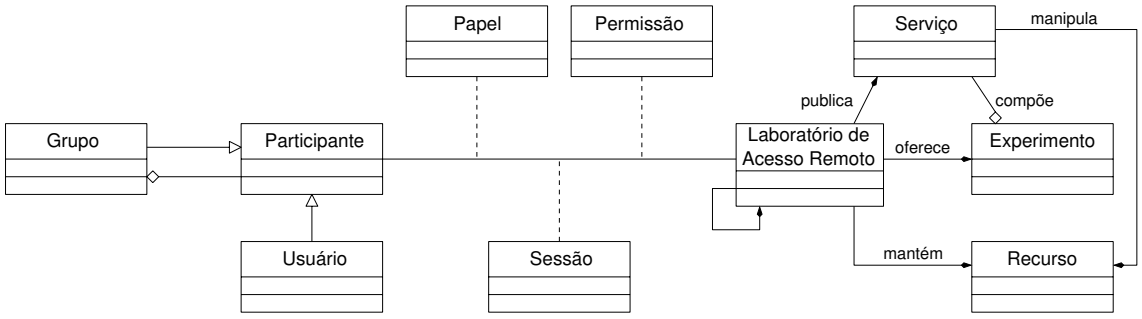


Fig. 3.1: Modelo conceitual para WebLabs.

Cada participante possui um *papel* e uma *permissão* para utilizar um WebLab. São exemplos

de papéis aluno, professor, pesquisador, auxiliar didático e administrador. Permissões definem atribuições, usualmente associadas a papéis, tais como permissão para cadastrar usuários, para disponibilizar experimentos e para executar experimentos. A interação do participante com o WebLab é regida por uma ou mais *sessões*. Uma sessão armazena o estado da interação do participante com o WebLab. Serviços interativos como WebLabs necessitam de pelo menos três classes de sessões:

1. Sessão de Acesso: controla o acesso do participante ao WebLab de acordo com seus papéis, permissões e credenciais.

2. Sessão de Interação: controla o uso dos recursos oferecidos pelo WebLab, propiciando ações de configuração e acionamento remoto de equipamentos, submissão remota de tarefas, aquisição remota de dados, dentre outras.

3. Sessão de Comunicação: controla o uso de recursos de comunicação tais como câmeras, microfones, sistemas de difusão, etc.

3.2 Arquitetura Baseada em SOA para WebLabs

A arquitetura que viabiliza a criação de WebLabs tem como peça fundamental os diversos serviços oferecidos. Estes serviços provêm mecanismos para controle de acesso, para comunicação e para criação de experimentos por meio dos serviços representativos dos recursos (tanto lógicos quanto físicos) disponíveis. A combinação e composição adequadas destes diversos serviços possibilitam, de fato, a criação de um WebLab.

O diagrama de pacotes na figura 3.2 apresenta os elementos que compõe uma arquitetura orientada a serviço para gerência de WebLabs. Os cinco pacotes representam os componentes gerais que provêm suporte a WebLabs.

Os Serviços de Gerência de Laboratório e de Participantes relacionam-se com informações de longa duração e devem prover tanto interfaces programáticas quanto de interação humana. A gerência do laboratório corresponde à gerência de recursos e experimentos disponíveis, incluindo o registro e a disponibilização destes. A gerência de participantes corresponde às atividades para inclusão e exclusão de participantes (usuários e grupos), bem como atribuições de credenciais (papéis e permissões) aos mesmos.

Os demais serviços correspondem aos Serviços de Gerência de Sessões e expõe interfaces para gerência da interação entre um participante e o WebLab. A arquitetura identifica três sessões: acesso, interação e comunicação. Foi definido, assim, um mapeamento das funcionalidades destas sessões de tal forma que possam ser disponibilizadas e acessadas na forma de Serviços Web.

3.2.1 Sessão de Acesso

A necessidade do controle de acesso é fundamental para qualquer serviço interativo que queira delegar papéis e permissões a diversas categorias de usuários. Por isso, a apresentação de um serviço que possibilite tal controle é necessário para que a construção de um *WebLab* possa ser realmente efetiva.

Este serviço deve proporcionar ao usuário ações básicas, tais como:

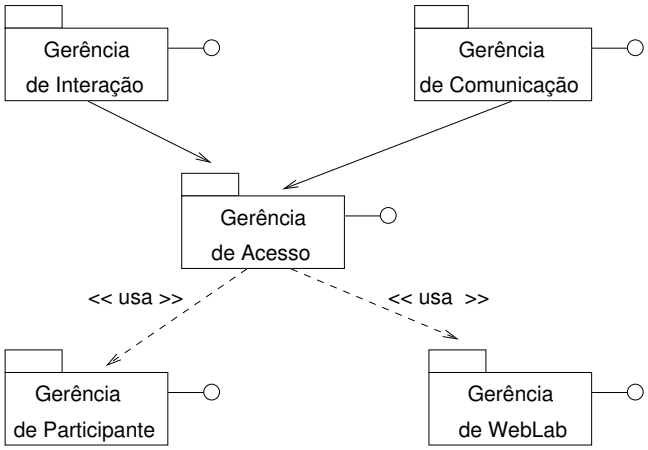


Fig. 3.2: Componentes de uma Arquitetura Orientada a Serviço para Gerência de WebLabs

Autenticação: mecanismos de autenticação devem garantir que somente usuários cadastrados tenham acesso ao laboratório.

Autorização: decide se as credenciais do usuário (papel e permissão) são suficientes para garantir acesso aos recursos do WebLab.

Criação de Sessão de Acesso: a cada execução de um experimento em um laboratório, o usuário deve criar uma sessão de acesso. Isto faz com que os recursos necessários ao experimento sejam devidamente alocados e indisponibilizados para outras atividades que possam requerer uso de algum destes recursos.

Destruição de Sessão de Acesso: ao término da execução do experimento, a destruição da sessão de acesso corrente libera os recursos capturados pelo experimento, tornando viável a realização de outras atividades que dependam do(s) recurso(s) em questão.

Verificação da Sessão de Acesso: durante a execução de alguma atividade dentro do laboratório, a verificação de inatividade deve levar ao cancelamento da sessão de acesso corrente, liberando os recursos. Para que isso ocorra, deve-se fornecer um mecanismo de monitoramento de atividade dentro de uma sessão.

Para execução destas ações, o Serviço de Acesso utiliza as informações previamente fornecidas aos Serviços de Gerência de Participantes e WebLabs para verificação de permissões e credenciais dos participantes, bem como disponibilidade e reserva prévia de recursos.

A figura 3.3 mostra um diagrama com as atribuições desejadas do serviço de acesso.

3.2.2 Sessão de Interação

Cada WebLab apresenta recursos e ferramentas específicas para a sua necessidade e finalidade. O fornecimentos destes instrumentos de maneira adequada é papel da sessão de interação. No caso

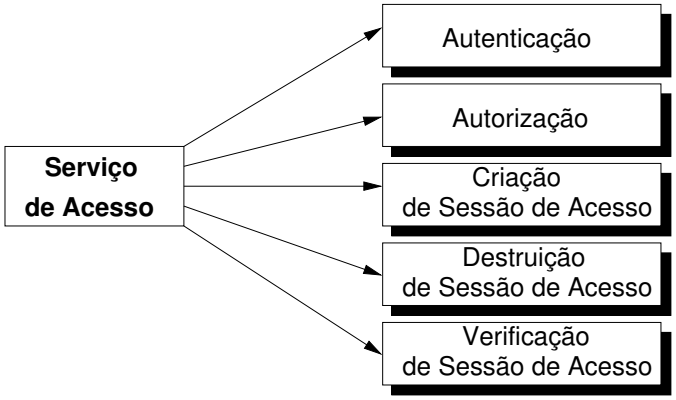


Fig. 3.3: Atribuições do Serviço de Acesso

particular desta arquitetura baseada em COS, tais instrumentos são disponibilizados na forma de serviços.

Assim, antes do desenvolvimento dos serviços propriamente, deve-se fazer um levantamento detalhado das características e funcionalidades dos recursos que se deseja prover. Além disso, um estudo das ferramentas necessárias aos experimentos disponibilizados pelo WebLab também é fundamental. A figura 3.4 mostra o processo de disponibilização dos recursos na forma de Serviços Web.

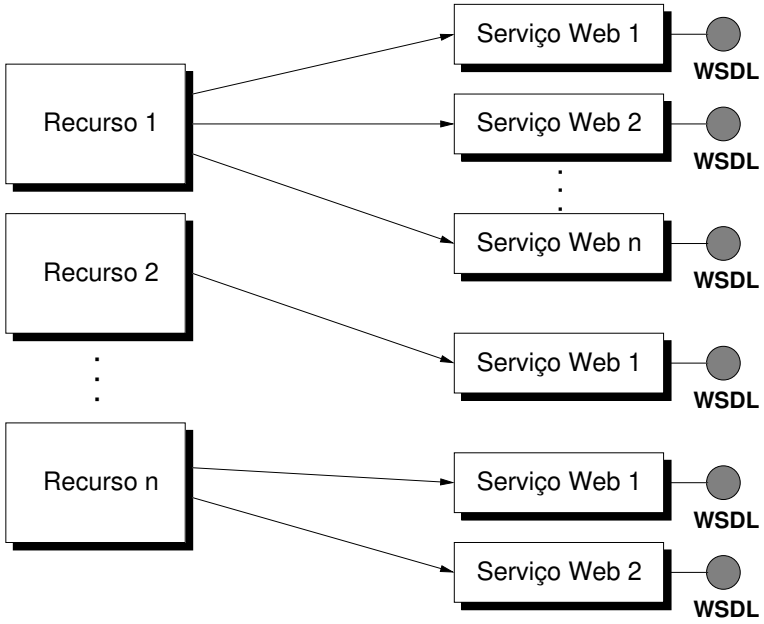


Fig. 3.4: Serviços Representativos dos Recursos

A composição adequada dos serviços disponibilizados pela Sessão de Interação, previamente autenticada pela Sessão de Acesso (seção 3.2.1), dão origem aos experimentos componentes do WebLab. Esta composição pode se dar tanto de maneira *"ad-hoc"* (os diversos serviços são compostos diretamente no código da aplicação), quanto por meio da criação de um *Processo de*

Negócio (seção 2.4) em um esquema de composição descrito na linguagem BPEL, devidamente instalado em uma máquina de composição.

Na figura 3.5 é possível visualizar a composição dos serviços de forma a prover novos serviços mais complexos que facilitem a tarefa do usuário na criação de experimentos. A vantagem desta abordagem, além da facilidade já citada, é a separação da implementação do serviços dos recursos da lógica da composição, o que facilita possíveis correções e alterações tanto nos serviços quanto nos processos de negócio.

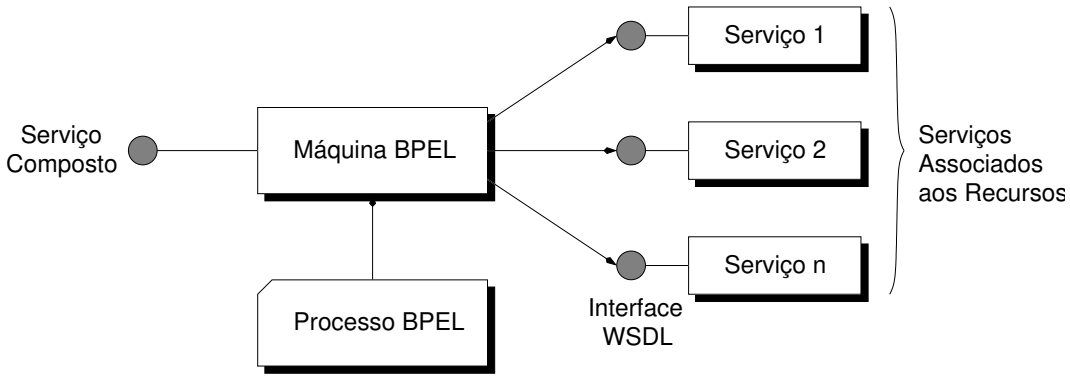


Fig. 3.5: Composição dos Serviços de Interação

As funções da gerência de interação incluem o início e encerramento de sessões de comunicação, e funções de *logging*.

3.2.3 Sessão de Comunicação

A terceira classe de sessão compreende serviços de suporte à comunicação. Responsáveis pela troca de informações entre os participantes e o WebLab, estes serviços de base devem prover mecanismos para:

Subscrição de fornecedores e consumidores de informação: registro de pontos de acessos a consumidores e fornecedores para envio de mensagens, bem como monitoramento do tráfego de informações entre estes. Possibilidade de encaminhamento automático de mensagens baseado nas escolhas de consumidores durante a subscrição.

Troca de mensagens: mecanismo de troca de mensagens entre consumidores e fornecedores, como forma de controle e negociação de parâmetros necessários à comunicação entre ambos.

Notificação de eventos: envio assíncrono de mensagens (eventos) como forma de notificação de anormalidades e atividades de reconfiguração dos fornecedores de informação.

Gerência de conexão de fluxos de dados: criação, negociação e encerramento de conexão entre fornecedores e consumidores de fluxos de dados, com a intermediação no estabelecimento de parâmetros de configuração, bem como definição de controle de QoS sobre a rede.

Monitoramento: possibilidade de acompanhamento gerencial da utilização do serviço, monitorando trocas de mensagens. Útil para identificação de funcionamento anormal do serviço, bem como deteção de falhas.

3.3 Considerações Finais

Este capítulo apresentou o modelo conceitual para a arquitetura proposta. As três sessões descritas regem a relação entre participantes e WebLab, e são a base para a implementação da arquitetura. O capítulo seguinte descreve o processo de implementação do modelo utilizando COS.

Capítulo 4

Implementação da Arquitetura

Este capítulo descreve uma implementação da arquitetura proposta no capítulo 3. Inicialmente, são apresentados detalhes do desenvolvimento das três sessões, com ênfase na Sessão de Interação (seção 4.1.2); esta sessão será a base para o desenvolvimento das aplicações e experimentos fornecidos pelo laboratório. Por fim, são exibidas as infra-estruturas de *hardware* e de *software*.

4.1 Serviços

Os serviços constituem a infra-estrutura básica para a implementação do WebLab. Uma implementação do modelo conceitual de WebLab foi realizada utilizando Serviços Web classificados em três categorias: serviços de acesso, de interação e de comunicação. Estes serviços suportam o estabelecimento das três sessões apresentadas no capítulo 3. O WebLab permite a criação de experimentos a partir da composição dos serviços oferecidos. Novos serviços podem ser adicionados sem qualquer interferência àqueles já disponíveis. Mais importante, a composição pode fazer uso de serviços oferecidos por outros WebLabs.

Serviços de acesso são responsáveis pelo gerenciamento de usuários, grupos, papéis, permissões, recursos, experimentos e WebLabs. O controle de acesso e a autenticação de usuários também é de responsabilidade destes serviços. A concepção dos serviços de acesso os tornam gerais para serem empregados em qualquer aplicação de WebLab, independentemente do seu domínio.

Serviços de interação suportam a execução remota de experimentos, oferecendo interfaces de manipulação de recursos e ferramentas necessárias aos experimentos oferecidos pelo WebLab. Estas interfaces permitem a seleção de formas de interação, a configuração e manipulação remota de equipamentos robóticos, a submissão remota de tarefas (código executável), a aquisição remota de dados e o acompanhamento e registro da interação.

Serviços de comunicação suportam os diversos estilos de comunicação um-para-muitos tais como comunicação multimídia em tempo real, notificação assíncrona de eventos, comunicação em grupo e difusão de mensagens.

Os serviços de acesso, interação e comunicação têm suas interfaces especificadas em WSDL. O acesso a estes serviços via interfaces WSDL é importante no caso de federações onde seus serviços são utilizados por clientes específicos localizados em outros domínios da federação, eventualmente utilizando esquemas de composição de Serviços Web.

4.1.1 Serviço de Acesso

Os serviços de Acesso compreendem os serviços necessários para controle de acesso a laboratórios e experimentos, bem como controle da sessão de uso do experimento.

Para utilizar um experimento em um laboratório, o participante utiliza-se do Serviço de Acesso para ser autenticado e, só então, seleciona um serviço da lista de serviços oferecidos. Portanto é de responsabilidade deste serviço o estabelecimento de uma relação segura e gerenciável entre os seus usuários e o provedor de serviço. Esta relação implica, genericamente, em uma relação contratual entre os envolvidos, permitindo ao usuário o acesso aos serviços disponibilizados pelo provedor de serviços.

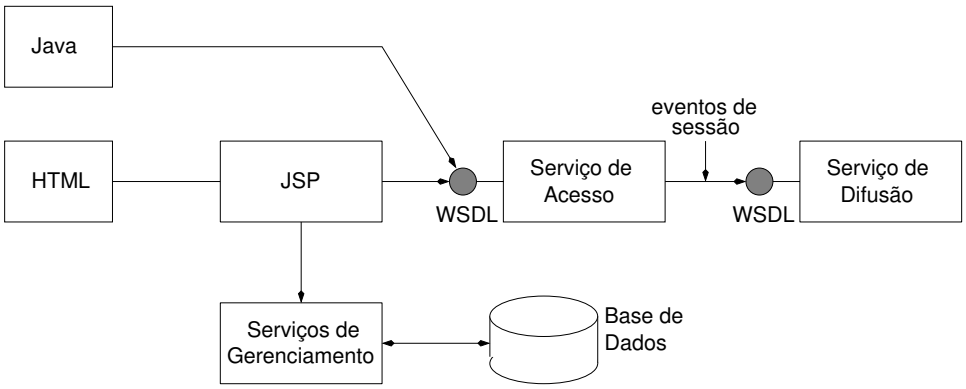


Fig. 4.1: Serviços de Acesso ao GigaBOT WebLab.

Dentre os serviços contemplados na sessão de acesso, destacamos três grupos:

- Gerenciamento de participantes, papéis e permissões.
- Gerenciamento de laboratórios, experimentos e recursos.
- Gerenciamento de reserva, de uso e de restrições.

Estes serviços são implementados por meio de páginas dinâmicas JSP (*Java Server Pages*) e Java *beans*. Páginas JSPs interagem com um sistema gerenciador de base de dados que, por sua vez, armazena informações acerca dos participantes (p.ex., usuários e grupos), recursos (p.ex., câmeras, robôs), experimentos, laboratórios, reservas de experimentos e restrições associadas aos recursos ou às reservas de participantes.

Para restringir o uso do WebLab, o serviço de gerenciamento de participantes possibilita o cadastramento, atualização e remoção dos mesmos. Usuário e Grupo constituem especializações de Participante. Grupos são constituídos por usuários ou por combinação de usuários e grupos. Cada participante utiliza o WebLab segundo o papel e permissão atribuído. Papel define o escopo de funções dentro do WebLab. Dentre os papéis que um participante pode assumir, destacamos: administrador, pesquisador, professor, técnico, aluno e visitante. Já permissões, definem atribuições, usualmente associadas a papéis, como por exemplo: para um usuário aluno, as permissões definem quais permissões o aluno poderá executar no momento.

Com funcionalidades semelhantes, o serviço de gerenciamento de laboratórios, experimentos e recursos possibilita o cadastramento, atualização e remoção de laboratórios, experimentos e recursos,

bem como a associação de experimentos a laboratórios e de recursos a experimentos. Por não ser temporal, um recurso pode ser associado a mais de um experimento assim como um experimento a mais de um laboratório. Só depois dessas associações terem sido estabelecidas é que um experimento pode ser disponibilizado, ou seja, a relação de tempo é estabelecida. Assim, dois experimentos que utilizam os mesmos recursos não podem ser disponibilizados simultaneamente, a menos que o recurso seja compartilhável.

Serviços Interativos como WebLab necessitam de acesso exclusivo para certos recursos (p.ex., robôs, câmeras) e, assim, demandam da Sessão de Acesso serviços que possibilitem a reserva dos mesmos por um determinado tempo. Naturalmente, que ao usuário cabe somente a reserva do experimento, enquanto que ao sistema cabe a garantia que os recursos associados ao experimento estarão disponíveis para a reserva do experimento em questão. Por outro lado, recursos demandam manutenção e, portanto, também é necessário disponibilizar aos mantenedores do WebLab serviços que possibilitem restringir o acesso a dados recursos em um dado período, como meio de permitir a manutenção do mesmo.

Um WebLab deve ser compartilhado entre um grupo de usuários com o mesmo objetivo, por exemplo, realizar um experimento relativo a uma seqüência de experimentos de um dado curso. Nestes casos, é interessante limitar o mínimo e o máximo tempo de reserva que um participante poderá contratar no WebLab em um dado período, possibilitando uma melhor utilização dos recursos.

Interfaces de Acesso ao Serviço

Os serviços de acesso podem ser acessados tanto por meio das páginas JSP, quanto por meio da interface WSDL do serviço *Access Service*. O primeiro método de acesso é utilizado para as três formas de gerência citadas. O Serviço Web possibilita ao experimento em execução a criação, a manutenção e o encerramento de sessões de acesso para realização dos experimentos.

Este Serviço Web verifica a existência de reserva para determinado usuário, e em caso positivo retorna o tempo restante. Para que a sessão de uso seja mantida é necessário uma verificação de atividade por parte do usuário a cada minuto, com tolerância de mais um minuto. Caso seja constatada inatividade por um período superior a dois minutos a sessão é encerrada e o experimento é terminado. Isto previne a alocação de recursos que não estejam sendo efetivamente usados e possibilita o encerramento da sessão caso o usuário esqueça de fazê-lo.

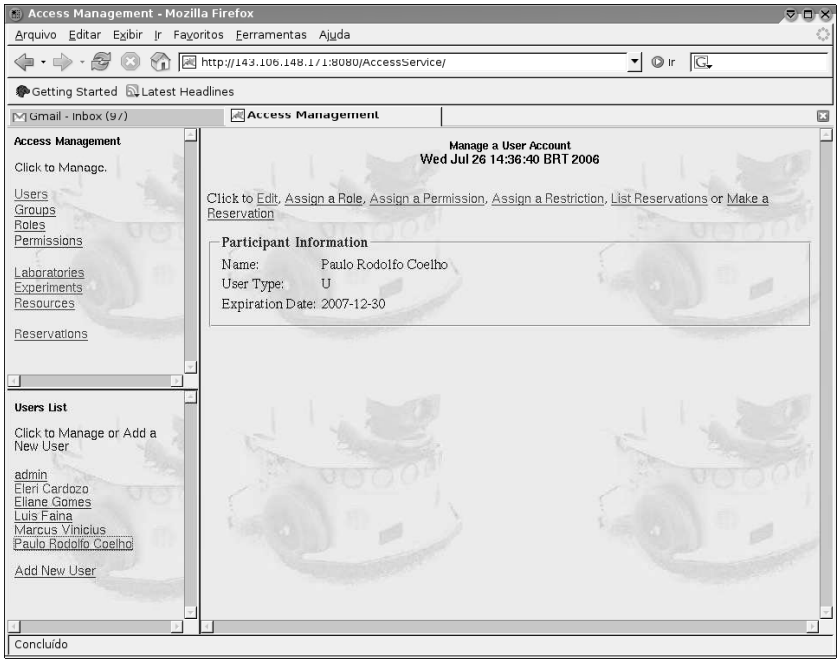
Os métodos oferecidos pelos Serviços Web, bem como as interfaces JSP de gerência podem ser vistas na figuras 4.2(a) e 4.2(b).

4.1.2 Serviço de Interação

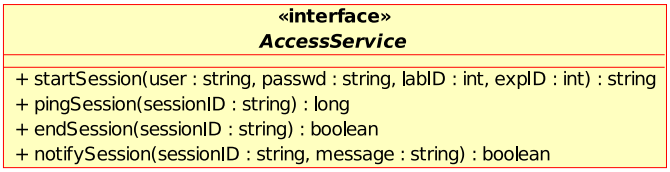
Os Serviços de Interação correspondem àqueles que efetivamente tornam o WebLab funcional do ponto de vista do uso por parte do usuário final.

Tendo em vista o *hardware* disponibilizado para o projeto (seção 4.2), optou-se pela área de robótica. Desta forma, foram implementados serviços que possibilitem interação com os robôs e com as câmeras disponíveis.

Os serviços correspondem ao mapeamento de recursos oferecidos pelos *hardwares* em métodos que possam ser acessados de forma clara e intuitiva, permitindo controle e acompanhamento do resultado das interações sobre estes. Este mapeamento tem como principais vantagens:



(a) Interface JSP de Gerência



(b) Interface para Controle de Sessão de Uso

Fig. 4.2: Interfaces de Utilização dos Serviços de Acesso

- Esconder a complexidade do *hardware* envolvido uma vez que o usuário final acessa apenas a interface WSDL que descreve o serviço implementado.

- Facilitar a atualização de componentes do *hardware* e do próprio *hardware*, uma vez que alterações na implementação do serviço não alteram o uso do ponto de vista do usuário final.

Controle do Robô

Os robôs utilizados no WebLab possuem uma API própria desenvolvida em C++. Para disponibilizar os principais recursos oferecidos por eles, foram definidos serviços que possibilitam, na forma de Serviços Web, acesso às suas funcionalidades.

A comunicação com o robô pode ser feita ou por meio de sua porta serial, ou por meio da rede, por meio de um aplicativo fornecido pelo fabricante denominado *IPTHRU*. Para a criação dos Serviços Web a segunda opção teve de ser adotada, pois seria inviável instalar os serviços no próprio robô.

O aplicativo utilizado, entretanto, possui a restrição de aceitar apenas uma conexão, pois seu papel é simular uma conexão feita pela porta serial do robô. Com a restrição imposta, a utilização dos serviços se tornaria inviável ou, no mínimo, muito limitada. Cada invocação ao serviço teria de se

conectar ao robô, realizar sua operação específica, e desconectar-se para que outra operação pudesse ser invocada. Além do tempo perdido com conexões e desconexões, a possibilidade de invocações em paralelo dos serviços relativos ao robô seria vetada por esta limitação.

Para solucionar tal restrição, optou-se por criar um intermediário entre o robô e os serviços, de forma que este possa aceitar conexões de vários clientes, inclusive concorrentemente. Foi desenvolvido, assim, um Operador do Robô *multi-threaded*. Seu papel não é apenas manter uma única conexão com o robô, mas possibilitar que as diversas *threads* (correspondentes aos diversos serviços) sejam invocadas independentemente, possibilitando aos serviços acesso simultâneo às operações do robô. A figura 4.3 ilustra o esquema descrito.

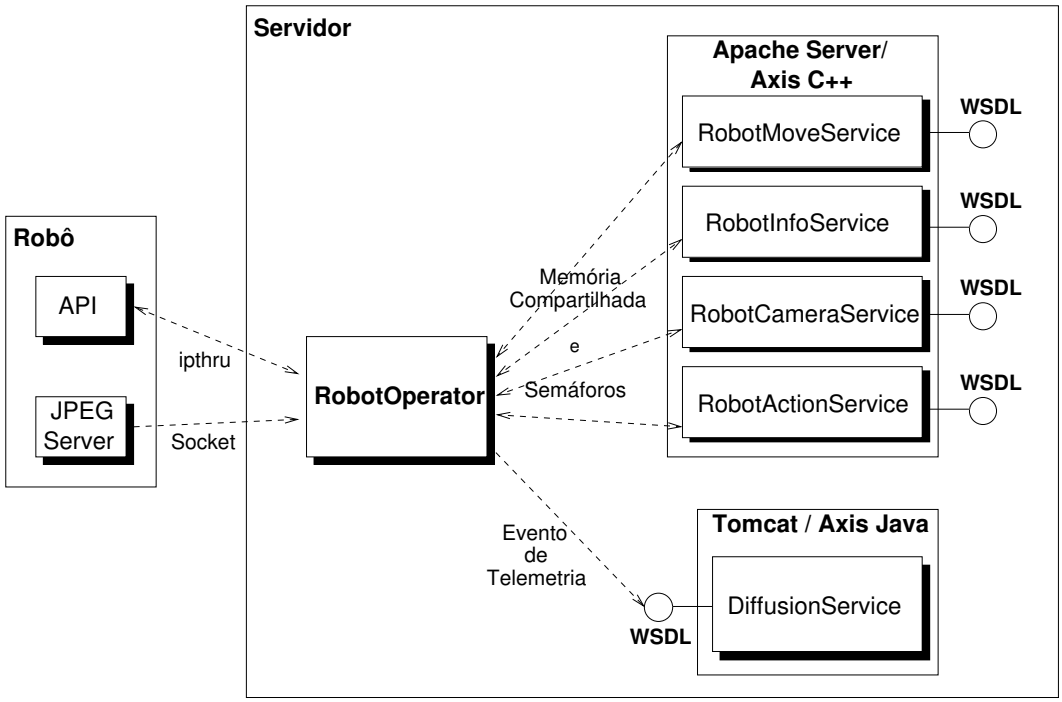


Fig. 4.3: Operador do Robô

A comunicação entre o Operador e cada serviço ocorre por meio de uma região de memória compartilhada com cada um. Além disso, para impor a ordem certa de acesso a esta memória, e garantir que o serviço aguarde o término da operação requerida sobre o robô antes de resgatar o retorno na região compartilhada, foi adotado um esquema de sinalização entre Operador e serviços por meio de semáforos.

Desta forma, o Serviço Web acessa a região de memória disponibilizada pelo Operador, e marca a operação desejada no atributo *opCode* (*Struct operation* na figura 4.4). Feito isto, o semáforo do Operador é liberado, ficando o serviço ”travado” aguardando o retorno. O Operador executa a operação requerida, colocando (quando pertinente) os valores de retorno nesta mesma região compartilhada e ”liberando” o semáforo do serviço. Por fim, o serviço retorna ao cliente os valores obtidos da região compartilhada.

Além de fazer o papel de intermediário entre Serviços Web e robô, o Operador tem o papel

fundamental de difundir informações de telemetria por meio do Serviço de Difusão (seção 4.1.3), possibilitando o acompanhamento, em tempo real, da movimentação do robô por clientes subscritos neste Serviço.

As imagens capturadas pelo robô com câmera de bordo também são intermediadas pelo Operador do Robô. Um aplicativo servidor em execução no robô as obtém da placa de captura instalada. O Operador funciona como um cliente, que, por *socket*, obtém a cadeia de bytes correspondente à imagem capturada, repassando-a ao Serviço Web correspondente.

O programa é composto por sete *threads*, sendo que quatro interagem com os serviços do robô:

1. Classe *Move*: *Thread* que interage com o Serviço de Movimentação do Robô, ficando 'travada' pelo semáforo (Classe *Sem*), até que uma invocação do serviço informe a operação desejada e 'libere' a *thread* para a obtenção dos parâmetros na região compartilhada de memória. A região compartilhada possui a estrutura da Classe *Operation*, sendo composta de um número inteiro que define a operação em execução e um vetor para troca de parâmetros de entrada/retorno.

2. Classe *Action*: interage com o Serviço de Ações, operando de maneira análoga a descrita para o Classe *Move*.

3. Classe *Info*: interage com o Serviço de Informação, capturando do robô tanto valores de telemetria, quanto valores de carga da bateria, de quantidade de sonares, etc. Opera da mesma forma das demais classes descritas.

4. Classe *View*: interage com o Serviço de Visão, oferecendo o controle de direção e *zoom* da câmera de bordo, operando também por meio de semáforos e memória compartilhada. Esta classe também é cliente da aplicação *JpegServer*, mostrada na figura 4.3, e, por isso, tem a função de obter as imagens capturadas, convertendo-as na estrutura necessária ao Serviço de Visão.

5. Classe *InfoFlowProducer*: cliente do Serviço de Difusão. É responsável pelo envio de informações de telemetria. Essa *thread* envia tais informações, na forma de um evento, em uma taxa de oito eventos por segundo. Desta forma, é possível um acompanhamento em tempo real da movimentação do robô.

6. Classe *Keyhandler*: responsável pelo tratamento de eventos do teclado. Quando a tecla <ESC> é pressionada, as demais *threads* são encerradas e o programa é finalizado.

7. Classe *RobotOperator*: trata-se da *thread* principal do programa, responsável pela comunicação inicial com o robô, pelo acionamento do motor e da câmera de bordo, pela instanciação e disparo das demais classes (*threads*), e pela desconexão do robô ao término da execução.

A figura 4.4 exibe as classes que compõe o Operador do Robô e a relação entre elas.

Serviço de Movimentação

Para controle de movimentação do robô foi implementado um serviço em C++ que troca informações com o Operador do Robô para realização das operações.

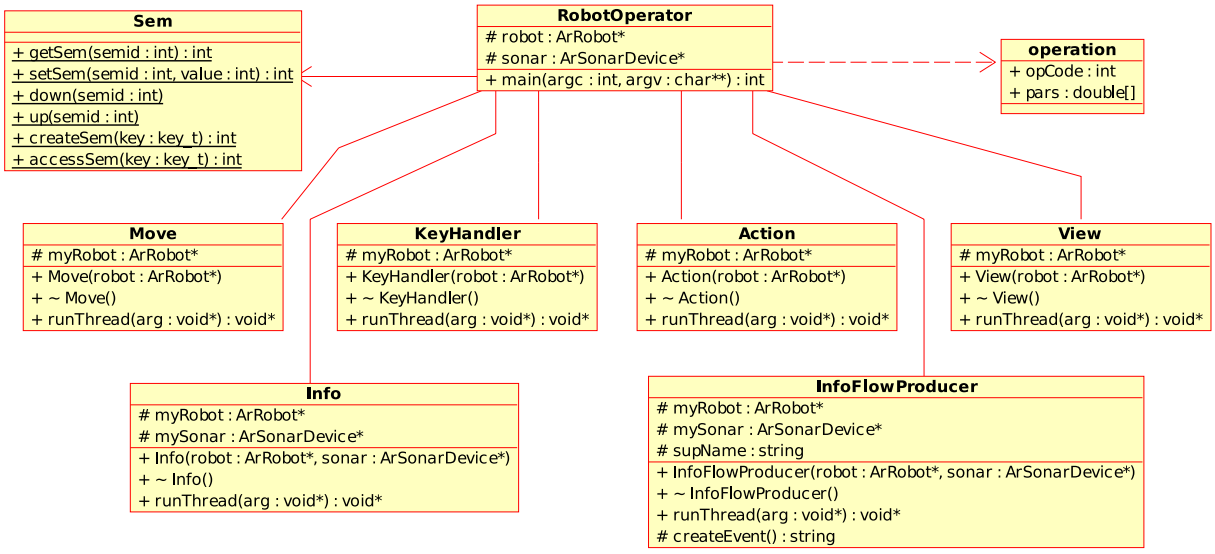


Fig. 4.4: Diagrama de Classes do Operador do Robô

Ele provê as operações básicas de movimentação por meio de movimentação para uma posição definida, tanto de forma síncrona (*moveSync*), quanto assíncrona (*move*). Permite ainda deslocamentos e giros. O acompanhamento da movimentação é possível por meio dos métodos *isMoveDone* para deslocamentos e *isHeadingDone* para rotações.

Com este serviço pode-se ainda definir e obter valores para velocidades e acelerações, tanto lineares quanto rotacionais.

Por fim, pode-se definir e obter a posição atual do robô, além de distância e ângulo para um posição objetivo.

A figura 4.5 expõe o diagrama de classes do serviço, no qual é possível perceber os métodos descritos.

Serviço de Ações

O Serviço de Ações é responsável pelo mapeamento de um conjunto básico de ações fornecidas pela API do robô, a fim de que possa ser acessadas fácil e diretamente por meio de Serviços Web.

Ações correspondem a um conjunto de tarefas mais comuns pré-definidas e fornecidas pela robô e visam facilitar operações com o robô tornando tarefas repetitivas e rotineiras facilmente acessadas.

Diversas ações podem ser executadas concorrentemente. O robô provê, para tal, um mecanismo de prioridades que define a procedência e a escolha da ação executada em cada situação. Desta forma, ações de recuperação e proteção contra colisão costumam ter prioridade maior que ações de movimentação.

São nove as ações disponibilizadas pelo Serviço de Ações (figura 4.6). Cada qual possui parâmetros próprios de configuração, de acordo com a função a ser exercida. São elas:

Avoid Front : este método tem o objetivo de evitar colisões frontais. Ele utiliza os dispositivos de alcance que o robô possui (sonar, neste caso) para detectar a aproximação de um obstáculo e

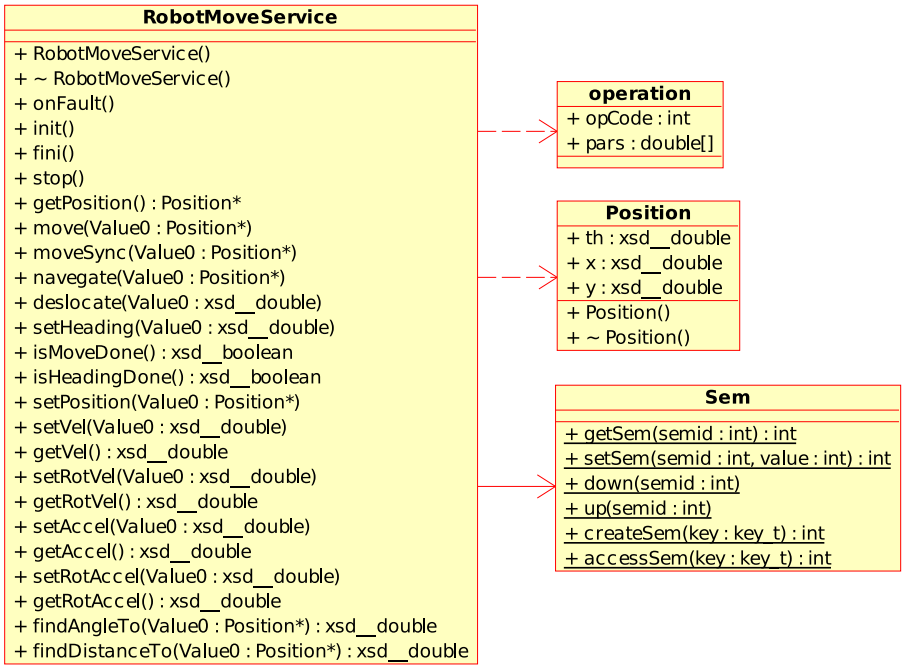


Fig. 4.5: Diagrama de Classes do Serviço de Movimentação

tomar medida preventiva. Seus parâmetros de funcionamento são a distância definida como limiar para tomar a ação (em mm), a velocidade durante sua ação (em mm/s) e o ângulo a ser rotacionado na tentativa de afastar-se do obstáculo (em graus).

Avoid Side : Trabalha de maneira semelhante à ação anterior, prevenindo, entretanto, colisões laterais. Seus parâmetros de controle são a distância mínima do obstáculo (em mm) e o variação de ângulo para desvio (em graus).

Bumpers : Define a ação a ser tomada em caso de colisão com os *bumpers*. Se ocorre colisão frontal, o robô deve retornar e virar-se; em caso de colisão traseira, o robô deve ir para frente e virar. É configurado pelos parâmetros de velocidade de retorno (em mm/sec), tempo de retorno (em ms), e tempo de rotação (também em ms).

Goto : Esta ação move o robô para a posição definida pelo usuário. Esse movimento é realizado sem qualquer ação de desvio de obstáculos ou recuperação de colisões. Pode, por isso, ser combinado com ações deste tipo. Seus parâmetros são a posição desejada (com coordenadas x e y em mm), a distância mínima do objetivo (em mm), a velocidade de deslocamento (em mm/s), a velocidade durante a rotação na direção do objetivo (em mm/s) e o ângulo de rotação (em graus).

Goto Straigth : Esta ação é semelhante à anterior, diferindo apenas na ausência da rotação em direção ao objetivo. Deste modo, o robô, sem virar, chega o mais próximo possível da posição final. Por isso, o único parâmetro de controle é a velocidade de locomoção (em mm/s).

Limiter Backwards : Ação que limita o movimento traseiro do robô, baseado na leitura obtida dos sonares. Seus parâmetros são a distância em que se deve parar (em mm), a distância na qual se deve reduzir a velocidade (em mm), e a velocidade máxima do movimento.

Limiter Forwards : É equivalente à ação anterior para movimentos para frente, possuindo os mesmos parâmetros de controle.

Stall Recover : Tenta recuperar, por meio de uma série de ações pré-cofiguradas, caso ocorra um travamento do robô ou de suas rodas. Seus parâmetros são a distância do obstáculo (em mm) a partir da qual uma ação deve ser tomada, o número máximo de ações de recuperação, a velocidade durante a ação (em mm/s) e o ângulo de rotação (em graus).

Constant Velocity : Trata-se de uma ação básica que coloca o robô em movimento retilíneo uniforme. Por isso, seu único parâmetro é a velocidade de deslocamento (em mm/s).

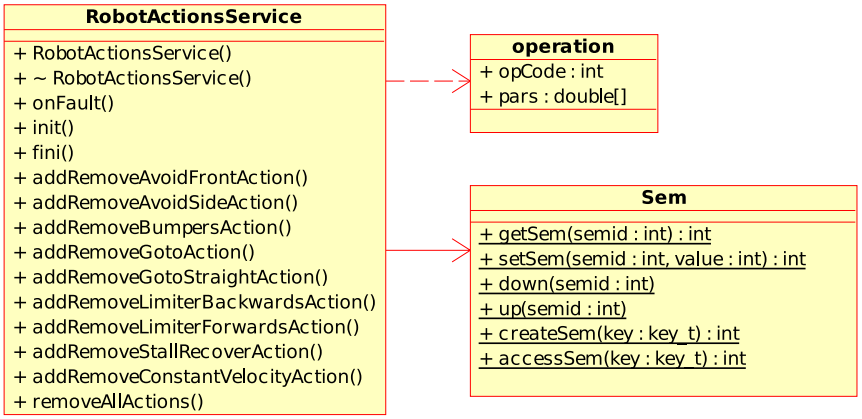


Fig. 4.6: Diagrama de Classes do Serviço de Ações

Serviço de Visão

Para acompanhamento de ações sobre o robô, foi desenvolvido um Serviço Web que provê controle sobre sua câmera. Com ele é possível obter imagens e configurar a posição da câmera.

A câmera operada pelo robô é fixada sobre este (figura 4.14(a)). Trata-se da câmera Canon VC-C4, descrita na seção 4.2, na qual é possível obter os valores de PTZ (*pan*, *tilt* e *zoom*), bem como reajustá-los, tanto relativa quanto absolutamente.

O serviço fornece ainda a imagem capturada pela câmera por meio do método *getJpegFrame*. Ele retorna a classe *Image*, que possui como atributos um vetor de byte representando a imagem e um número inteiro que expressa o tamanho em *bytes* da imagem capturada. Para obtenção da imagem, o Operador do Robô se conecta ao servidor Jpeg em execução no robô e repassa a imagem obtida para o serviço.

Antes da operação da câmera, entretanto, é necessário que o método *initDevice* seja invocado para que seja executada sua rotina de inicialização e esta esteja apta às operações requisitadas.

A figura 4.7 exibe o diagrama das classes que compõem o Serviço de Visão.

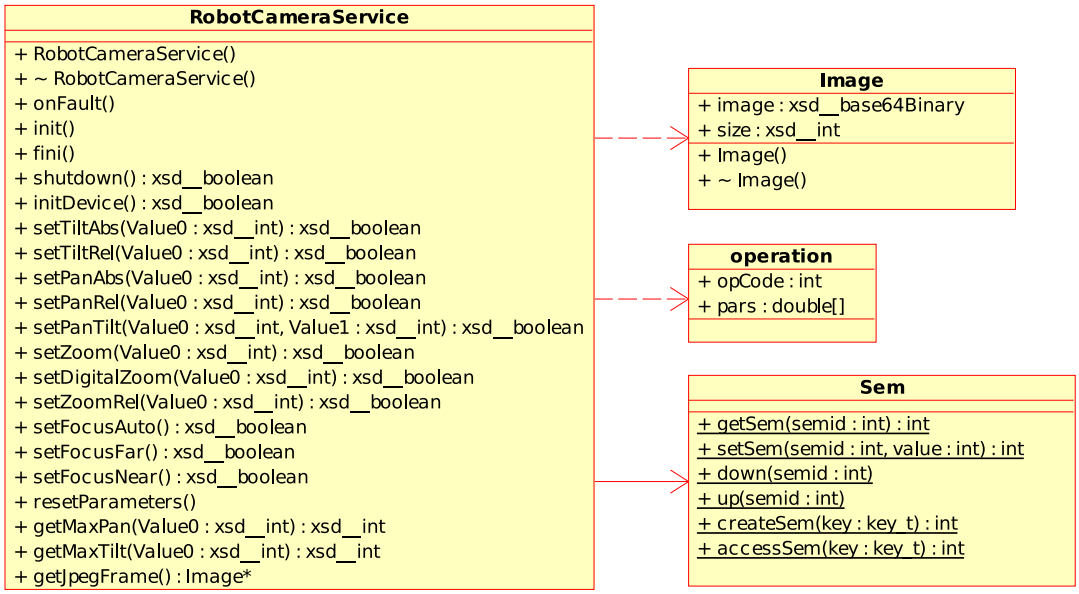


Fig. 4.7: Diagrama de Classes do Serviço de Visão

Serviço de Telemetria

O Serviço de Telemetria oferece ao usuário a possibilidade de acompanhamento de informações sobre o robô, tornando possível a verificação de resultados de ações sobre ele por meio dos valores obtidos.

Podem ser obtidas as mais diversas informações, entre as quais temos:

- Número de Sonares instalados no robô.
- Alcance dos sonares, tanto valores individuais quanto um vetor com a leitura de todos os sonares.
- Ângulo do sonar em relação ao robô, também individual ou globalmente.
- A posição, em relação ao referencial local, de cada sonar.
- A posição, em relação ao robô, de cada sonar.
- A posição, em relação ao referencial local, da leitura de cada sonar.
- Valor da tensão nas baterias do robô.
- Comprimento e largura do robô.

A classe *Position* é retornada para métodos que requerem informações de posição, enquanto a classe *SonarStruct* retorna informações sobre todos os sonares instalados. Na figura 4.8 é apresentado um diagrama de classes detalhado do serviço implementado.

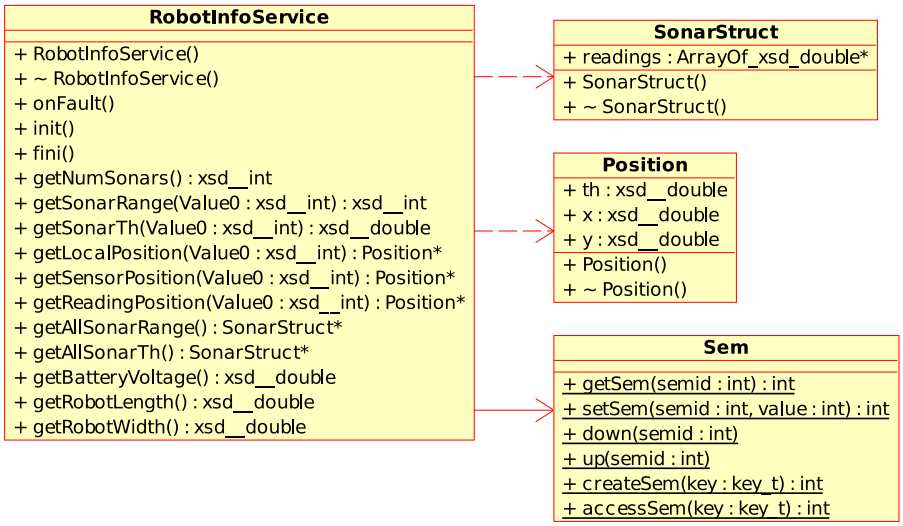


Fig. 4.8: Diagrama de Classes do Serviço de Telemetria

Eventos de Telemetria

Além da intermediação entre robô e serviços, o Operador tem o papel fundamental de difundir informações de Telemetria. Essa difusão ocorre por meio da submissão de eventos contendo tais informações por meio do Serviço de Difusão (4.1.3). Para isso, o Operador se registra como um produtor de eventos neste serviço e, feito isto, dispara eventos de telemetria em uma taxa de oito eventos por segundo.

Os usuários que necessitam informações de posicionamento em tempo real devem se subscrever como consumidores de eventos no Serviço de Difusão. Um exemplo de evento de telemetria pode ser observado na figura 4.9. As informações disponibilizadas no evento são praticamente as mesmas fornecidas pelo Serviço de Telemetria.

O que caracteriza a necessidade por eventos de telemetria, em substituição à obtenção direta por meio do Serviço de Telemetria, é a importância da informação em tempo real, de grande utilidade para ações de correção de controle e para acompanhamento efetivo do resultado da execução de ações sobre o robô.

Serviço de Controle da Câmera Panorâmica

O Serviço de Controle da Câmera Panorâmica apresenta uma interface similar à apresentada para o Serviço de Visão do robô. Isto facilita a utilização do serviço e a adaptação do código do cliente para uso deste recurso. Para o serviço em questão foi utilizada a câmera Axis 213PTZ (seção 4.2).

Além dos funções básicas citados para o Serviço de Visão, foram adicionadas duas novas utilidades a este serviço. A primeira é a possibilidade de definir parâmetros de qualidade e dimensão da figura, passados como argumentos para o método *getUserDefinedJpegFrame*; enquanto a outra possibilita a gravação de vídeos.

A gravação de vídeos é controlada pelos métodos *startRecording* e *stopRecording*. Ao ser invocado o primeiro método, a gravação é iniciada. A invocação do segundo, que requer

```
<?xml version="1.0" encoding="UTF-8"?>
<event>
  <name>TelemetryEvent</name>
  <supplier>RobotInfoFlowProducer</supplier>
  <timeout>1000</timeout>
  <logrecord>false</logrecord>
  <battery>13.0</battery>
  <numSonar>16</numSonar>
  <position>
    <x>0.0</x>
    <y>0.0</y>
    <th>0.0</th>
  </position>
  <sonars>
    <localPosition sonar="0">
      <x>69.0</x>
      <y>3987.0</y>
      <th>0.0</th>
    </localPosition>
    <readingPosition sonar="0">
      <x>69.0</x>
      <y>3987.0</y>
      <th>0.0</th>
    </readingPosition>
    <sensorPosition sonar="0">
      <x>69.0</x>
      <y>136.0</y>
      <th>90.0</th>
    </sensorPosition>
        ...
        ...
        ...
    <localPosition sonar="15">
      <x>-157.0</x>
      <y>3986.0</y>
      <th>0.0</th>
    </localPosition>
    <readingPosition sonar="15">
      <x>-157.0</x>
      <y>3986.0</y>
      <th>0.0</th>
    </readingPosition>
    <sensorPosition sonar="15">
      <x>-157.0</x>
      <y>136.0</y>
      <th>90.0</th>
    </sensorPosition>
  </sonars>
</event>
```

Fig. 4.9: Exemplo de um Evento de Telemetria

nome do arquivo e comentários sobre o vídeo como parâmetros de entrada, finaliza a gravação, disponibilizando o resultado do processo de gravação, no formato XVID, em um diretório público acessível pelo usuário, juntamente com os comentários passados como parâmetros.

A figura 4.10 ilustra o diagrama de classes representativo do serviço implementado. A classe *ConsoleColor* tem apenas a função de alterar a cor do texto no *log* de saída, para fins de acompanhamento das invocações ao serviço.

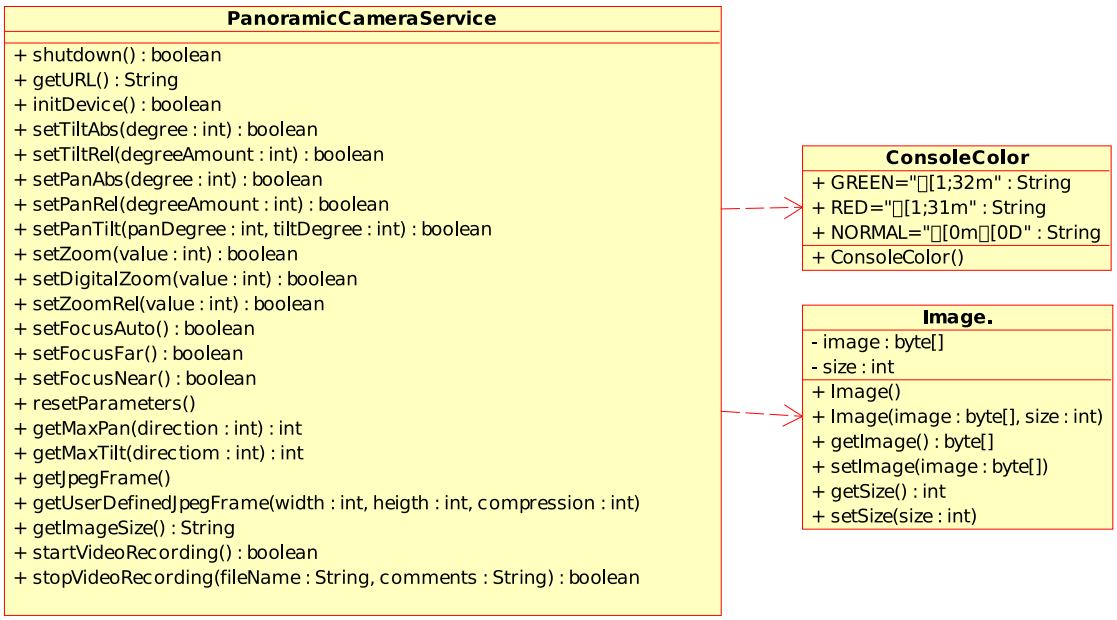


Fig. 4.10: Diagrama de Classes do Serviço da Câmera Panorâmica

Serviço de Submissão de Código

Para usuários avançados, foi desenvolvido um serviço capaz de executar um programa com comunicação direta com o robô, realizando tarefas diretamente sobre este. O serviço, implementado em Java e instalado no Axis, fornece a infra-estrutura necessário para a submissão, execução, acompanhamento e encerramento do programa.

Para isto, é necessário conhecimento da API do robô, além de conhecimentos da linguagem de programação C++, na qual a API foi desenvolvida.

O código executável previamente compilado deve ser enviado ao servidor por meio do método *transferData*, sendo armazenado em um espaço reservado ao usuário. Uma vez transferido, sua execução pode ser iniciada (método *execute*) definindo-se o nome do programa a ser executado e seus parâmetros de entrada.

O acompanhamento da execução é realizado pelo método *getOutput*, que pode ser invocado a qualquer momento. Com ele é possível obter as mensagens recebidas tanto na saída padrão, quanto na saída de erro.

Caso alguma operação anormal seja observada ou o usuário deseje encerrar a execução, é possível invocar, a qualquer momento, a operação *kill*.

Devido à restrição já citada de o robô aceitar apenas uma conexão, o uso deste serviço impossibilita o uso dos demais serviços que utilizam o robô. Isto ocorre, porque Operador do

robô deve estar desconectado deste para que o programa do cliente possa conectar-se. A câmera panorâmica, entretanto, pode ser normalmente utilizada para acompanhamento do experimento.

A figura 4.11 apresenta o diagrama de classes do serviço implementado. Nele é possível observar que os métodos tratam individualmente de cada usuário.

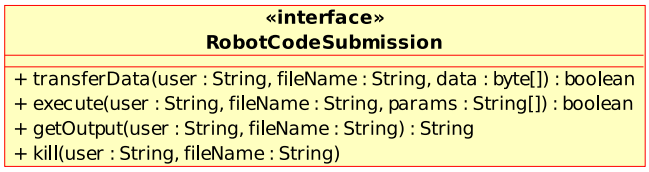


Fig. 4.11: Diagrama de Classes do Serviço de Submissão de Código

4.1.3 Serviço de Comunicação

O Serviço de Comunicação baseia-se em um serviço de notificação de eventos - descritos segundo a linguagem XML - para transportar informações que identificam, por exemplo, se um fornecedor está começando/encerrando a transmissão de um fluxo ou se o evento traz uma notificação simples para um consumidor (seja um alarme, um erro ou um dado qualquer). O serviço responsabiliza-se pela entrega dos eventos submetidos aos consumidores interessados.

Essa demonstração de interesse é feita por meio de um filtro, na forma de uma expressão XPath (*XML Path Language*), que descreve um determinado tipo de informação a ser procurada no evento.

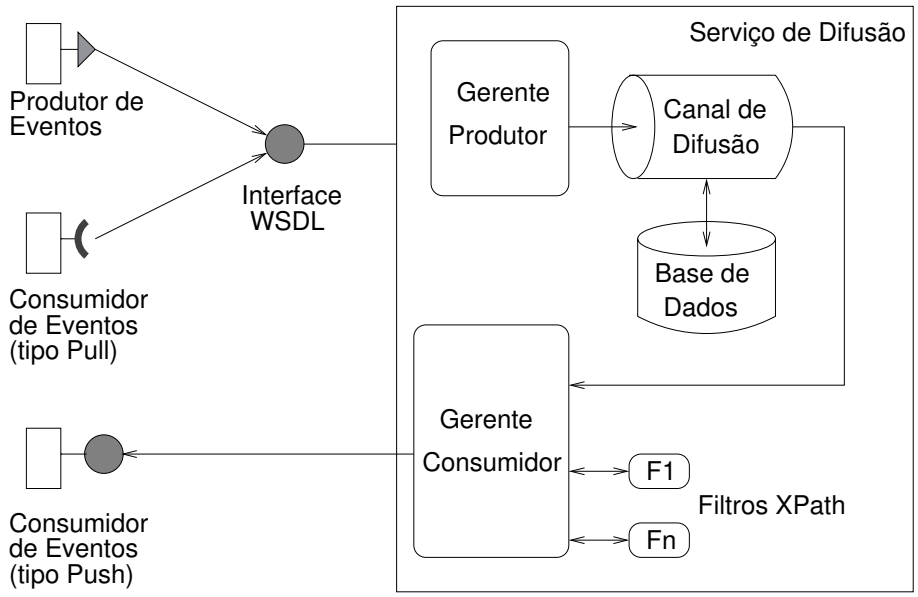


Fig. 4.12: Modelo do Serviço de Difusão

O Serviço de Difusão (figura 4.12) aborda os modelos *push* e *pull*. Um consumidor do tipo *push* deve obrigatoriamente informar uma porta receptora de dados (interface de Serviço Web). Um

consumidor do tipo *pull* não tem essa obrigação. Dessa forma, o consumidor *push* irá receber um evento sempre que o serviço identificar que seu filtro combina com um evento válido. Já um consumidor *pull*, tomará a iniciativa de acessar o serviço a procura de um evento válido, sempre que lhe for conveniente, informando o filtro desejado.

Independente do tipo, um consumidor pode especificar um fornecedor de dados durante sua subscrição. Assim, somente os eventos válidos desse fornecedor específico serão repassados ao consumidor. Isso cria uma associação entre as partes. Caso o consumidor não exponha a necessidade de associar-se a um fornecedor específico, ele terá acesso a todos os eventos submetidos, validando apenas aqueles que coincidirem com seu filtro. Existe ainda a possibilidade de um consumidor requisitar um fornecedor que ainda não esteja cadastrado no serviço. Isso fará com que ele seja colocado em um lista de espera, onde ficará aguardando até que o fornecedor específico entre no serviço e submeta eventos.

O Serviço de Difusão é composto por um Gerente Produtor, um Gerente Consumidor, um Canal de Fluxo, uma Base de Dados e pelos Filtros. O Gerente Produtor aceita os eventos submetidos ao serviço em sua interface, acrescenta uma marca de tempo e os transfere para o Canal de Fluxo. No canal, é averiguado se um evento é persistente ou transiente (necessidade de persistência na base de dados) e se é preciso armazená-lo no log. Através do canal também são feitas consultas, remoções e atualizações de eventos na Base de Dados. Os eventos são descritos por meio da linguagem XML, conforme um arquivo de descrição XSD (*XML Schema Description*), e as consultas por meio dos Filtros, que correspondem a expressões XPath, utilizadas para selecionar nós em um documento XML. Os Filtros são associados a um determinado consumidor, e por conseqüência à sua interface consumidora de eventos (caso exista). Os consumidores são administrados pelo Gerente Consumidor, que gerencia os Filtros e analisa a validade de cada evento, a fim de entregar somente aqueles que ainda não tenham validade. Um exemplo de um evento é exibido na figura 4.9.

O Serviço de Difusão é responsável por prover comunicação multi-ponto entre os diversos serviços. Ele pode ser empregado para notificação de eventos (por exemplo, alarmes), difusão de mensagens textuais (por exemplo, em aplicativos tipo bate-papo), difusão de informação gráfica (por exemplo, em aplicativos de quadro branco) e suporte a controle de dispositivos multimídia (por exemplo, sincronizar produtores e consumidores de áudio ou vídeo). Permite ainda a negociação de parâmetros multimídia entre produtor e consumidores de fluxo.

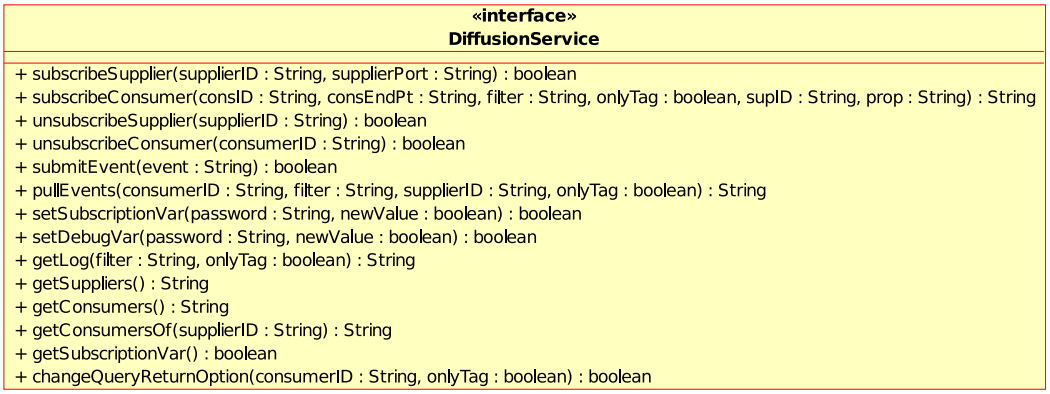
O Serviço provê uma interface WSDL que possibilita o cadastro e o descadastramento de consumidores e produtores, o envio de eventos, a consulta a eventos submetidos, a verificação do *log*. Permite ainda tarefas administrativas como determinar se a subscrição, a validação dos eventos submetidos, ou o modo de *debug* devem ser ativados ou desativados. É possível também a consulta dos produtores e consumidores registrados.

A figura 4.13(a) exibe a classe que representa o serviço implementado.

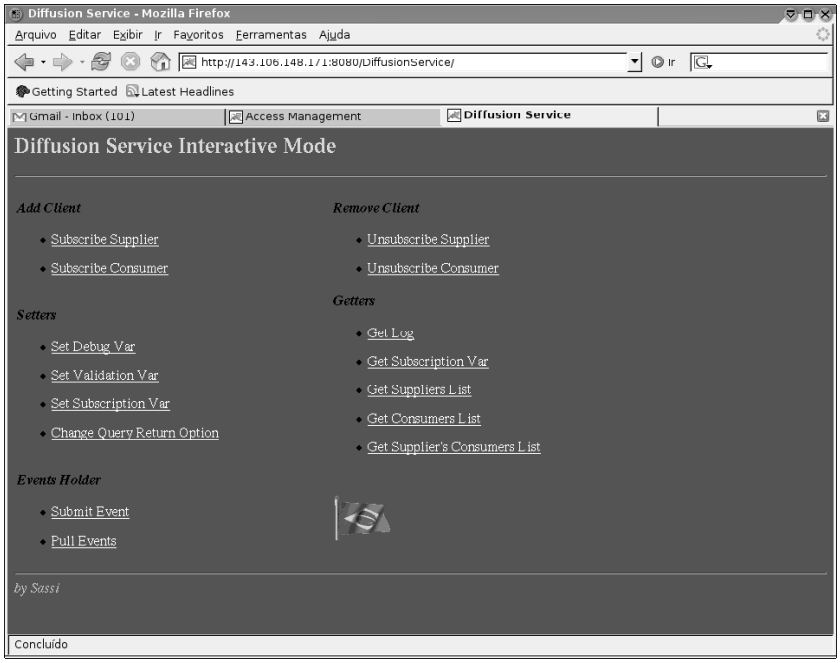
O serviço apresenta um *front-end* desenvolvido em JSP para controle e consulta ao serviço. A figura 4.13(b) ilustra este *front-end*.

4.2 Infra-estrutura de *Hardware*

Para o desenvolvimento dos serviços de interação foram utilizados diversos recursos de *hardware*. Tais recursos foram estudados e mapeados para Serviços Web (4.1.2) de forma que possam ser



(a) Métodos oferecidos pelo Serviço de Difusão



(b) Interface JSP para Administração do Serviço

Fig. 4.13: Acesso ao Serviço de Difusão

facilmente acessados e combinados para criação de experimentos na área de robótica.

Os recursos de *hardware* utilizados são:

- Dois robôs Pioneer P3-DX.
- Uma câmera Axis PTZ213.
- Uma câmera Axis 206W;
- Um computador de mão iPaq da Hewlett-Packard.
- Um ponto de acesso Sem Fio LinkSys WAP54G.

- Um roteador Summit 200-24.
- Dois servidores para instalação dos serviços.
- Um computador para implementação da arquitetura.

Robôs Pioneer P3-DX

Para execução dos experimentos em robótica o laboratório conta com dois robôs modelo P3-DX da Pioneer [38] (figura 4.14). Embora do mesmo modelo, eles diferem quanto ao acessórios componentes de cada um.

(a) Robô com Computador e Câmera de Bordo

(b) Robô Controlado pelo iPaq

Fig. 4.14: Robôs Pioneer P3-DX

O primeiro deles apresenta os seguintes acessórios:

***Bumpers* de proteção:** provêm um mecanismo de segurança contra colisão, desligando automaticamente os motores do robô em caso de choque.

Sonares: São 16 sonares ao redor do robô, oferecendo um mecanismo para mapeamento do ambiente e prevenção de colisão.

Computador de Bordo: O controle do robô se dá por meio deste computador. Trata-se de um Pentium 3 de 700 Mhz que fornece os seguintes recursos.

- Conexão sem fio.
- Placa de Captura de vídeo.

Câmera de Bordo: O robô fornece ainda uma câmera da Canon (figura 4.15), modelo VCC4 [39], diretamente ligada à placa de captura de vídeo do computador de bordo. Esta câmera oferece recursos de *pan*, *tilt* e *zoom*. Além disso, tanto seu controle quanto a aquisição de imagens são totalmente integrados à API do robô, sendo acessados diretamente por meio desta.

Fig. 4.15: Câmera de Bordo do Robô

O outro robô oferece apenas os *Bumpers* e os Sonares como acessórios. Para controle do mesmo deve ser efetuada uma conexão por meio de um porta serial disponível. Essa conexão é feita utilizando-se um computador de mão da HP, modelo HP5550 [40] (figura 4.16(a)). Este computador de mão, uma vez conectado ao robô, possui uma interface de rede sem fio que possibilita controlá-lo remotamente.

Para suprir a ausência de câmera de bordo sobre este robô foi acoplada a este uma câmera com interface de rede sem fio Axis 206W [41] (figura 4.16(b)). Esta câmera não provê controle de *pan* e *tilt*, mas possibilita alterar o *zoom*.

(a) Computador de mão para Controle do Robô

(b) Câmera de Bordo

Fig. 4.16: Acessórios Utilizados no Robô

Câmera Axis PTZ213

A câmera Axis PTZ213 [42] (figura 4.17) é uma câmera capaz de oferecer imagem de grande qualidade e controles avançados. Dentre seus diversos recursos destaca-se:

Controle de *pan*, *tilt* e *zoom*: oferece *zoom* óptico de 26× acrescido de 12× de *zoom* digital.

Modo Infra-Vermelho: a câmera pode ser manualmente controlada ou programada para alternar para o modo infra-vermelho em situações de baixa luminosidade.

Grande Qualidade de Imagem: quadros de até 768 × 576 a 25 fps (*frames* por segundo).

Formato de vídeo: é possível a transmissão de vídeo tanto no formato Motion JPEG quanto no formato MPEG-4.

API disponibilizada em HTML: seu controle é todo disponibilizado por meio de requisições HTML, facilitando sua operação.

Fig. 4.17: Câmera Panorâmica de Acompanhamento dos Experimentos

Esta câmera é responsável pelo fornecimento da visão panorâmica do ambiente em que o robô opera, dando uma noção mais completa ao usuário do resultado da execução dos experimentos do laboratório.

Servidores

Toda a infra-estrutura de serviços desenvolvida foi instalada em dois servidores. O principal, mais robusto, possui processador *Dual Core Xeon* da Intel, com freqüência de 3.0 GHz, 2 GB de memória RAM e disco com capacidade de 120 GB. Além disso ele conta com uma interface de rede Fast Ethernet.

Nesta máquina estão instalados os bancos de dados utilizados pelos serviços e pela máquina de orquestração BPEL, bem como os serviços mais básicos da Sessão de Acesso (seção 4.1.1) e de Comunicação (seção 4.1.3). Os serviços da câmera e o serviço de busca de Serviços Web (UDDI) também estão instalados neste servidor.

O segundo servidor - com processador Pentium 4 *Hyper Threading* de 3.4 GHz, 2 GB de memória RAM, 80 GB de disco rígido e placa de rede Fast Ethernet - é responsável basicamente pela comunicação e pelo controle dos robôs. Nele estão instalados os Serviços Web que operam estes recursos, bem como a câmera de bordo Canon VCC4 (integrada a API do robô).

4.2.1 Integração dos Recursos de *Hardware*

Os diversos recursos de *hardware* descritos são interconectados conforme diagrama apresentado na figura 4.18.

Os dois robôs conectam-se ao ponto de acesso da LinkSys WAP54G; um por meio de seu computador de bordo, outro por meio de um iPaq conectado à sua porta serial. A câmera Axis 206W também se liga a este ponto de acesso.

Nessa mesma rede local estão ligados ainda os dois servidores e a câmera Axis 213PTZ. Esta rede local conecta-se à rede Giga por meio do roteador Summit 200-24.

A câmera Canon VCC4 é acessível apenas por meio do robô ao qual ela se conecta por uma placa de captura de vídeo.

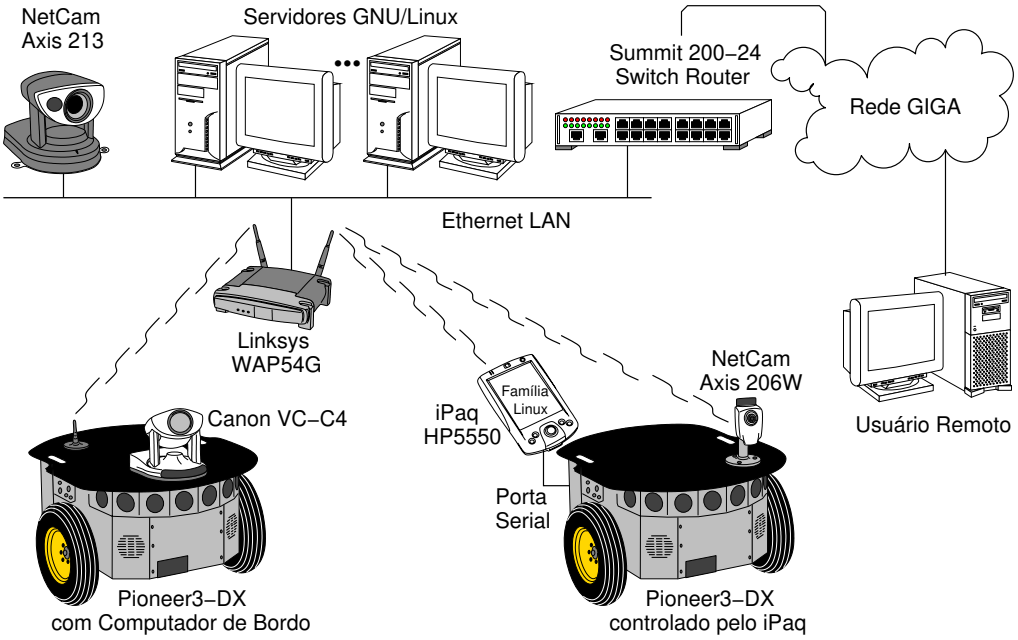


Fig. 4.18: Integração da Infra-estrutura de *Hardware* Disponível

4.3 Infra-estrutura de *Software*

Os *softwares* empregados na implementação da arquitetura proposta podem ser divididos em quatro grupos:

***Softwares* de Instalação e Descoberta de Serviços:** os *softwares* a partir do qual Serviços Web e Processos de Negócios são encontrados e acessados por meio de suas interfaces WSDL são a base para a arquitetura proposta.

No caso dos Serviços Web implementados na linguagem Java, estes *softwares* correspondem basicamente ao Servidor de Aplicação conhecido como Apache Tomcat [43] e ao Contêiner para instalação dos serviços denominado Apache Axis [44]. Este último é instalado no próprio Tomcat e provê ferramentas que auxiliam na criação da infra-estrutura necessária ao serviço, por meio da geração automática de:

- *Stubs* e *Skeletons* a partir de uma interface (em Java, no caso) que descreve os métodos do serviço.
- Interfaces WSDL para o serviço.
- Descritor de *Deployment* (em XML) do serviço, a fim de facilitar sua instalação.

Uma vez instalados, suas interfaces WSDL são facilmente acessadas por meio da Web.

Para *deployment* de Serviços Web implementados em C++ foi instalada uma versão do Axis desenvolvida para esta linguagem. Esta versão é instalada sobre o Servidor HTTP da Apache

[45] e provê ferramentas para geração automática de código. A interface WSDL e a instalação do serviço, contudo, devem ser feitas manualmente.

Para oferecimento dos Processos de Negócio foi instalada a máquina de orquestração da Oracle, Oracle BPEL *Process Manager* [46]. Entre seus principais recursos, destacam-se o suporte a:

- Padrões de Serviços Web, tais como XML, SOAP e WSDL.
- Armazenamento em banco de dados de processos de longa duração e correlação de mensagens assíncronas.
- Arquitetura Orientada a Serviço.
- Processamento paralelo de tarefas.
- Manipulação de faltas e gerenciamento de exceções durante o *design* e a execução.
- Notificações e *timeouts* de eventos.
- Mecanismos de compensação para a implementação de processos de longa duração.
- Escalabilidade e confiabilidade de processos.
- Gerenciamento e Administração de processos.
- Controle de versão.
- Trilha de auditoria para acompanhamento do histórico do fluxo de negócio (figura 4.19).

Fig. 4.19: Acompanhamento do Fluxo de Execução de um Processo de Negócio

Para facilitar a descoberta dos serviços desenvolvidos foi instalado um serviço de registro, também da Apache, conhecido como jUDDI [47]. Trata-se de uma implementação em código aberto, escrita em Java, das especificações do protocolo UDDI para Serviços Web. Desta forma, possibilita-se que aplicações encontrem e utilizem Serviços Web de forma rápida, fácil e dinâmica.

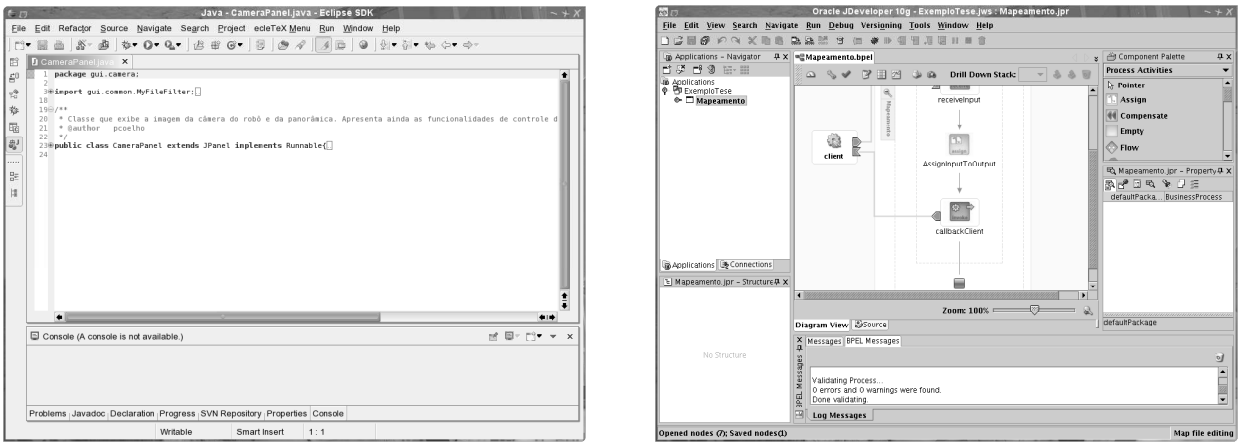
IDEs (*Integrated Development Environment*): para o desenvolvimento dos diversos Serviços Web, bem como as aplicações componentes do laboratório que se utilizam dos serviços implementados foi utilizado o *software* Eclipse [48]. Trata-se de uma IDE de código aberto que tem por objetivo prover uma plataforma de desenvolvimento e *frameworks* de aplicação para criação de *software*.

Eclipse (figura 4.20(a)) provê suporte nativo para desenvolvimento de aplicações em Java, oferecendo ferramentas que facilitam a compilação e execução destes aplicativos. Além disso, é possível estender o suporte para outras linguagens de programação com a instalação de *plugins*. *Plugins* são o grande diferencial do Eclipse, e são responsáveis por torná-lo tão largamente aceito e utilizado.

Para os serviços e aplicações desenvolvidos em C++, foi instalado no Eclipse o *plugin* CDT (C/C++ Development Tools). Com isso, o desenvolvimento de código pôde ser totalmente realizado no Eclipse.

Os Processos de Negócio, entretanto, foram criados em uma IDE fornecida pela Oracle juntamente com sua Máquina de Orquestração chamada Oracle JDeveloper (figura 4.20(b)). Entre sua principais vantagens estão:

- Facilidade na criação de Processos de Negócios: as atividades e parceiros são adicionados ao processo de maneira *drag and drop*.
- Assistentes para criação de parceiros e configuração das atividades.
- Integração com a Máquina de Orquestração: possibilita compilação e instalação do processo diretamente por meio da IDE.
- Assistente para criação de "Adaptadores": são ferramentas que permitem que informações do processo (valores de variáveis, resultados de atividades, etc) sejam enviados diretamente por SMTP (*Simple Mail Transfer Protocol*), FTP (*File Transfer Protocol*), HTTP ou para um arquivo local.



(a) Eclipse (b) Oracle JDeveloper

Fig. 4.20: IDEs utilizadas na implementação da arquitetura

Bancos de Dados: a arquitetura implementada utiliza três bancos de dados: dbXML [49], MySQL [50] e o Oracle *Database* 10g *Express Edition* [51].

O Serviço de Difusão (Seção 4.1.3), utiliza dbXML para persistência de eventos e manutenção de *logs*. Sua adoção deve-se ao fato de eventos e logs consistirem de documentos XML.

Para jUDDI, utilizado para registro dos Serviços Web implementados, foi adotado o banco de dados MySQL, sugerido pela própria Fundação Apache.

A Máquina de Orquestração da Oracle não provê suporte direto para MySQL. Por conta disso, foi instalada seu próprio banco de dados. Entretanto, visto que o armazenamento não é um ponto crítico, foi instalada uma versão menos robusta, conhecida como *Express Edition* (preferida sobre a versão *Enterprise Edition*, que provê funcionalidades específicas para servidores que necessitem de alto desempenho).

Outros *Softwares*: Diversos outros *softwares* foram utilizados como auxiliares no desenvolvimento e utilização da arquitetura. Entre eles, pode-se destacar:

- BoUML e Umbrello: modelagem UML da arquitetura.
- Xfig: criação de diagramas.
- Quanta Plus: criação de páginas HTML.
- Motion e Mplayer: gravação de vídeos a partir de fluxos de figuras obtidas da câmera Axis 213PTZ.

4.4 Considerações Finais

Este capítulo apresentou uma implementação para a arquitetura apresentada no capítulo anterior, bem como a infra-estrutura de *hardware* e *software* empregada no seu desenvolvimento. O próximo capítulo descreve um WebLab para o domínio da robótica móvel, com os experimentos desenvolvidos e avaliação de desempenho.

Capítulo 5

Um WebLab para Robótica Móvel

Para validação da arquitetura proposta foi desenvolvido um WebLab para Robótica Móvel. Trata-se do GigaBOT WebLab, o qual permite que participantes realizem remotamente, seja por meio da Internet, seja por meio de redes avançadas como a rede Giga da RNP, os experimentos na área de robótica móvel. Os serviços de acesso, de interação e de comunicação implementados (seção 4.1) foram utilizados na elaboração e disponibilização do WebLab. São apresentadas as interfaces de interação desenvolvidas para este WebLab, e as composições "*ad-hoc*" e por meio de BPEL, bem como exemplos de utilização destes mecanismos de composição para criação de experimentos e novos Serviços Web mais complexos.

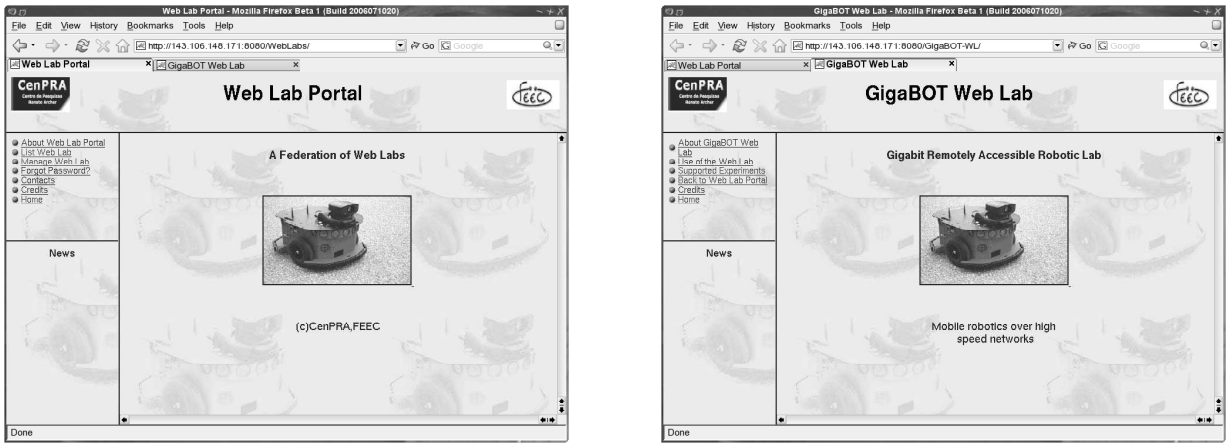
5.1 Interfaces de Interação

Para utilização dos serviços descritos na seção 4.1, foram desenvolvidas interfaces de interação com o usuário do WebLab. Tais interfaces são apresentadas basicamente sob dois formatos:

Interfaces HTML/JSP: compostas por interfaces acessíveis por meio do navegador. Correspondem a funcionalidades básicas e preliminares ao uso efetivo dos experimentos. Dentre tais funcionalidades estão a gerência de experimentos, recursos, usuários e laboratórios; a reserva e listagem de horários dos experimentos; atalhos para acesso efetivo ao experimentos e páginas descritivas dos mesmos.

O domínio *WebLabs* exibe os Laboratórios disponibilizados. Atualmente existe apenas o Laboratório de Robótica denominado "GigaBOT WebLab". Este pode ser acessado no domínio *GigaBOT-WL*. Por fim, os serviços de gerência podem ser contemplados no domínio *Access Service*. As figuras 5.1(a) e 5.1(b) exibem as interfaces dos dois primeiros domínios citados. A interface de gerência pode ser vista na figura 4.2(a).

Interfaces Gráficas em Java: as interfaces gráficas utilizadas correspondem aos experimentos propriamente ditos, que por sua vez são acessados por meio de um atalho na página JSP do GigaBOT WebLab. Este atalho é tratado por um *servlet*, o qual tem como função a adição de parâmetros de informações do usuário, retornando ao usuário um arquivo JNLP (*Java Network Launching Protocol*). Trata-se de um arquivo XML que descreve a forma como a aplicação deve ser iniciada. Este arquivo é aberto com um aplicativo componente do Java *Runtime Edition*



(a) Interface JSP dos WebLabs Disponíveis (b) Interface JSP do GigaBOT WebLab

Fig. 5.1: Interfaces JSP de Acesso aos WebLabs

denominado *Java Web Start* [52]. Para execução dos experimentos, entretanto, o usuário deve realizar uma reserva de horário prévia.

Uma vez em execução, as aplicações exibem suas interfaces de operação juntamente com uma interface básica de acompanhamento da movimentação do robô e uma interface de informações da sessão de uso corrente.

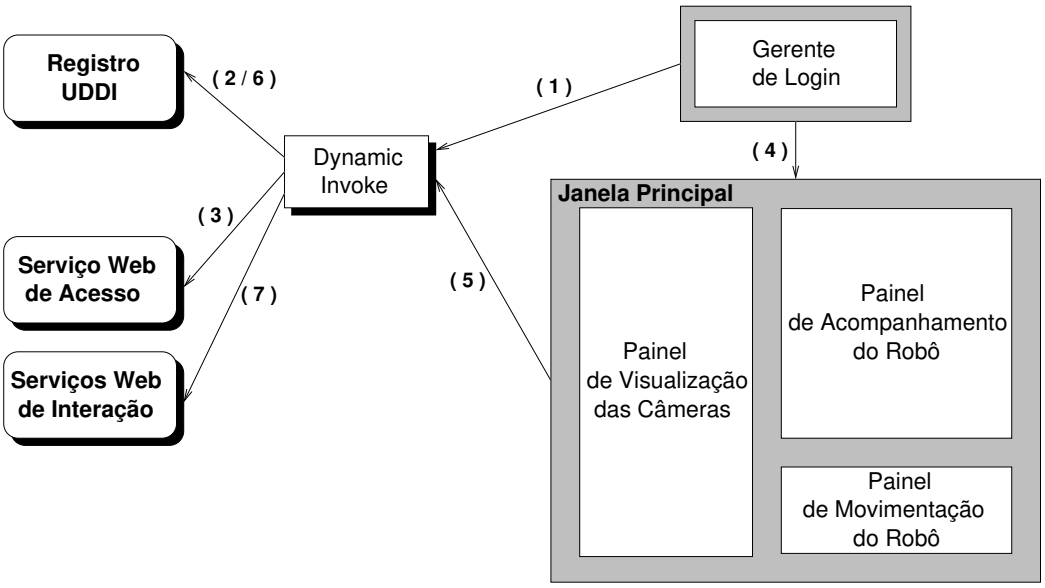
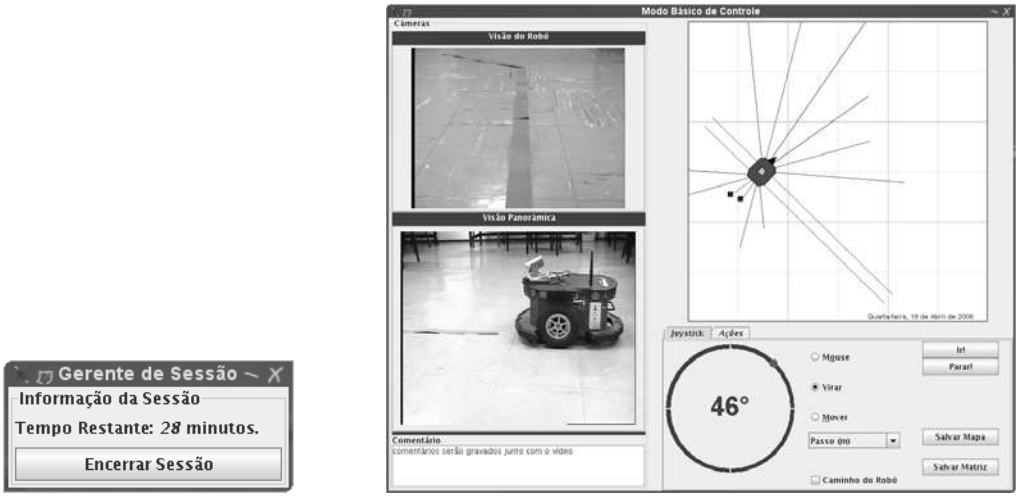


Fig. 5.2: Ciclo de Inicialização de um Experimento

A seqüência de inicialização do experimento pode ser acompanhado na figura 5.2. A interface de gerência tem a responsabilidade de, utilizando a classe *DynamicInvoke* (1) e após consultar o Registro UDDI e descobrir o endereço do Serviço de Acesso (2), iniciar a sessão de

acesso (3) por meio do método *startSession*. O serviço verifica a existência da reserva, retornando um identificador da sessão em caso de sucesso, ou vazio, caso contrário. O sucesso desta operação possibilita a inicialização das outras interfaces do experimento corrente (4). Primeiramente é iniciada a interface básica de acompanhamento, a qual, também por meio da classe *DynamicInvoke* (5) realiza a descoberta (no Registro UDDI) dos endereços dos diversos serviços (6) que representam os recursos, e invoca-os (7), exibindo uma mapa da localização do robô, juntamente com suas leituras de sonares, bem como as imagens das câmeras panorâmica e de bordo. Posteriormente, a interface principal do experimento é exibida. Para manter a sessão de acesso ativa, a interface de gerência invoca o método *pingSession* do Serviço de Acesso, a cada minuto, evitando o encerramento da sessão por inatividade; este método retorna o tempo restante para o término da reserva do usuário.

As interfaces trazendo informação da sessão de uso corrente e de acompanhamento podem ser visualizadas nas figuras 5.3(a) e 5.3(b).



(a) Gerente de Login (b) Acompanhamento e Operação Básica do Robô

Fig. 5.3: Interfaces Gráficas de Uso do WebLab

A interface de gerência e sua relação com a interface de acompanhamento pode ser observado no diagrama de classe apresentado na figura 5.4. Neste mesmo diagrama, nota-se que a classe *RobotIf* é composta por classes contidas nos pacotes *action*, *camera*, *codesubmission* e *robot*. A classe interna *PingPanel* implementa uma *thread* responsável pela manutenção de sessão de acesso corrente por meio de "*pings*" no Serviço de Acesso.

O pacote *action* corresponde à implementação da interface gráfica de acesso a um conjunto de nove ações pré-programadas no robô, disponibilizadas na forma de Serviço Web (seção 4.1.2). Este painel possibilita ao usuário a escolha das ações a serem executadas, bem como a configuração dos parâmetros de cada ação. Um diagrama das classes que fornecem tal infra-estrutura pode ser visto na figura 5.5.

O acompanhamento das imagens das câmeras panorâmica e de bordo é implementado no pacote *camera*. Trata-se de um painel por câmera, fornecendo a infra-estrutura de acesso aos Serviços

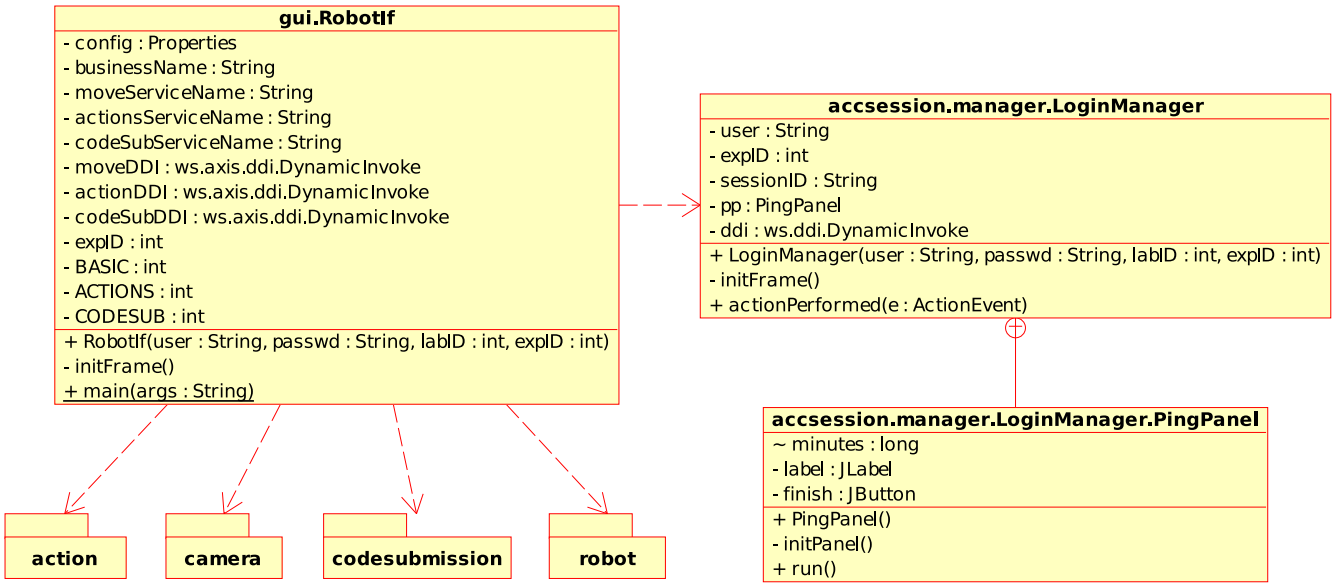


Fig. 5.4: Diagrama de Classes das Interfaces de Gerência e Acompanhamento

Web de cada câmera. Deste modo, pode-se controlar e configurar tanto a movimentação quanto o *zoom* das câmeras. A interface provê ainda funcionalidades extras, tais como salvar a imagem das câmeras localmente, ou ainda gravar um vídeo da execução do experimento corrente (apenas para a câmera panorâmica). Um diagrama de classes é exibido na figura 5.6, no qual pode-se observar as classes responsáveis por cada tarefa descrita.

Para a submissão de código, a interface provê um painel próprio no qual é possível ao usuário o envio do código, o disparo da execução, a obtenção das saídas padrão e de erro, bem como a interrupção da execução a qualquer instante. A figura 5.7(a) exibe este painel, enquanto a figura 5.7(b) exibe seu diagrama de classes.

Finalmente, o mapa com as leituras dos sonares e os controles do robô foram implementados no pacote *robot*. A classe *Map* é um painel responsável pela apresentação de informações de posição e de leituras de sonares do robô. As informações de telemetria são obtidas por meio do Serviço de Difusão (seção 3.2.3): a classe *Parser* registra um consumidor (classe *EventConsumer*) do tipo *pull* no Serviço de Difusão, e obtém do serviço os eventos de telemetria. Os eventos são submetidos e requeridos ao canal de eventos pelo Operador do Robô (seção 4.1.2) e pelo Consumidor a uma taxa de oito a dez eventos por segundo.

Dos eventos de telemetria, que tem a estrutura exibida na figura 4.9, extrai-se as informações de alcance dos sonares e posicionamento do robô. Para tal extração são aplicadas expressões XPath sobre o documento XML que representa o evento.

A classe *Map* possui ainda duas outras importantes funções: o mapeamento do ambiente e o tratamento de eventos do *mouse*. O mapeamento é resultado da aproximação do robô, durante sua movimentação, de algum obstáculo. Desta forma, sempre que algum obstáculo está a uma distância inferior a um metro do robô, o contorno deste obstáculo, conforme detectado

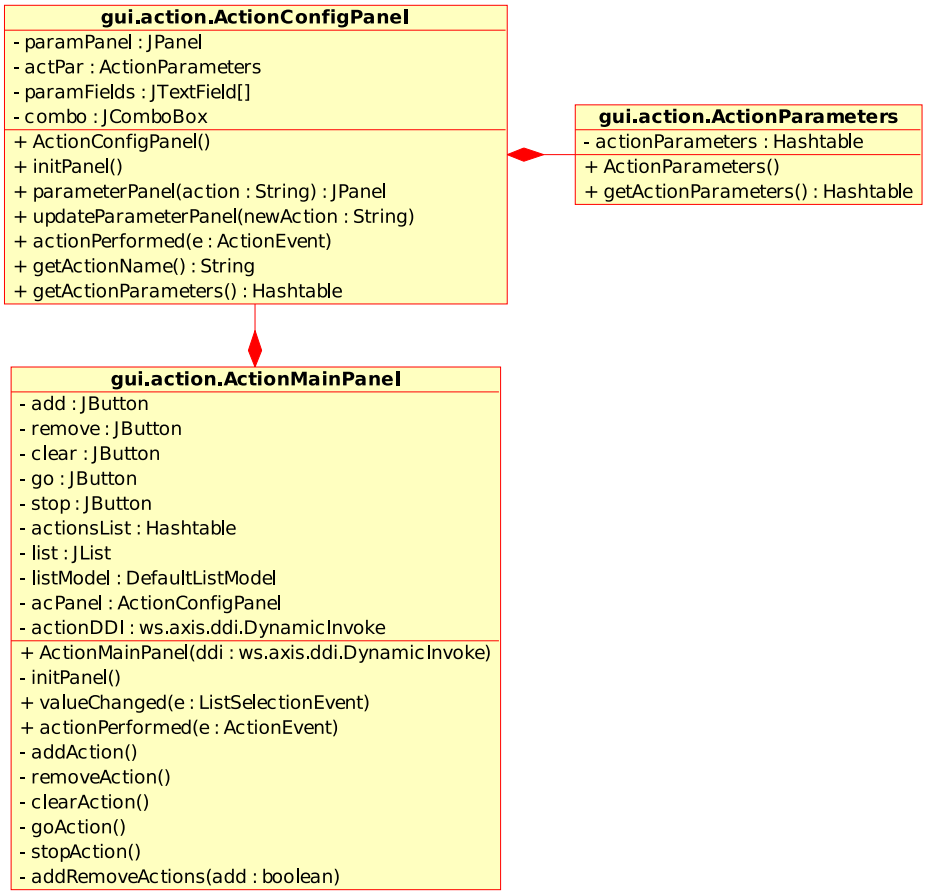


Fig. 5.5: Diagrama de Classes do Painel de Controle de Ações

pelos sonares, é exibido. A segunda função corresponde ao tratamento, quando habilitada a locomoção por meio do *mouse*, de eventos de clique do mesmo. Tais eventos são então transformados em coordenadas reais servindo como pontos-objetivos ao robô, possibilitando, desta forma, a definição de trajetórias por meio do *mouse*.

Ainda no pacote *robot* encontram-se as classes que tornam possível a movimentação básica do robô. São três os tipos de movimentos permitidos: a movimentação por meio de uma bússola, a movimentação por meio de passos definidos pelo usuário e a movimentação por meio de uma trajetória definida com o *mouse*.

A classe *Joystick* é responsável pelo controle da movimentação do robô. A movimentação por meio da bússola é definida ao se escolher a opção **Virar**. A classe *Compass* implementa o painel que exibe a bússola e passa à classe principal *Joystick* os valores definidos pelo usuário. A movimentação por meio de passos ocorre ao se escolher a opção **Mover** e definir o valor do passo (em metros). Essas duas primeiras movimentações são executadas na *thread MoveRobot*, que corresponde a uma classe interna à classe *Joystick*.

A movimentação por meio de uma trajetória definida pelo *mouse*, habilitada ao se definir a opção **Mouse**, é resultado da interação entre a classe *Map* e *Joystick*. A primeira armazena

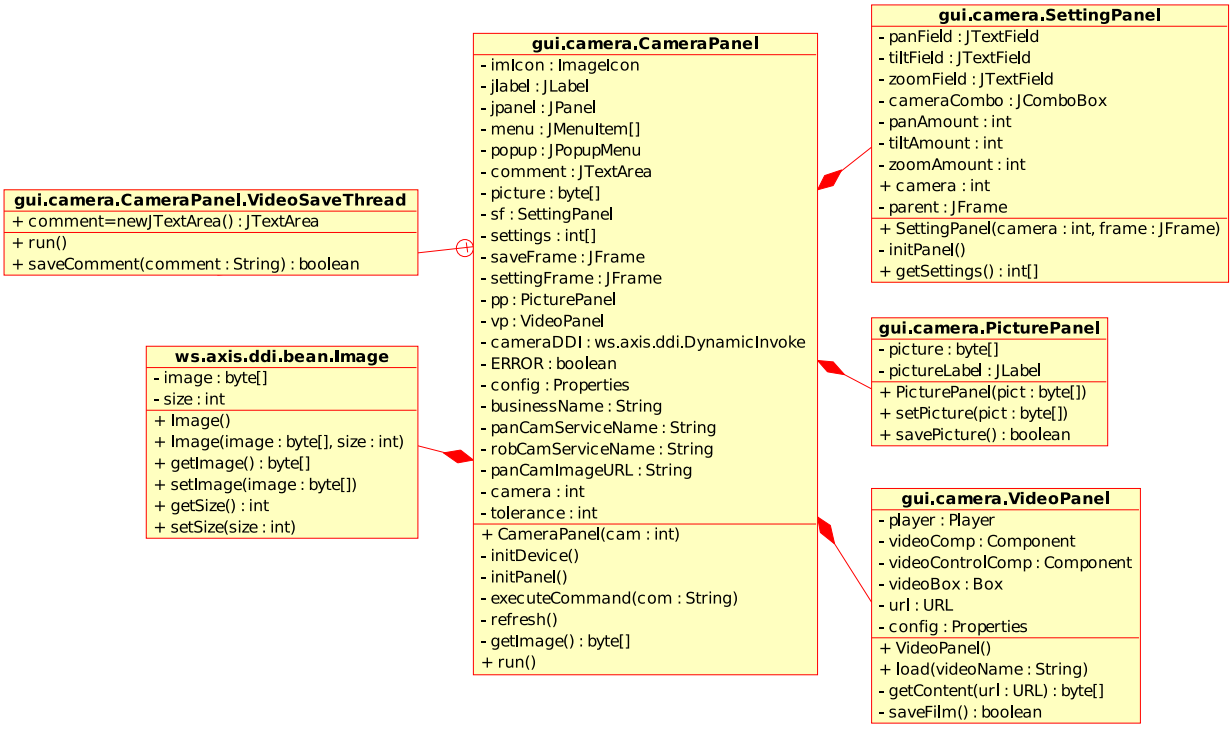
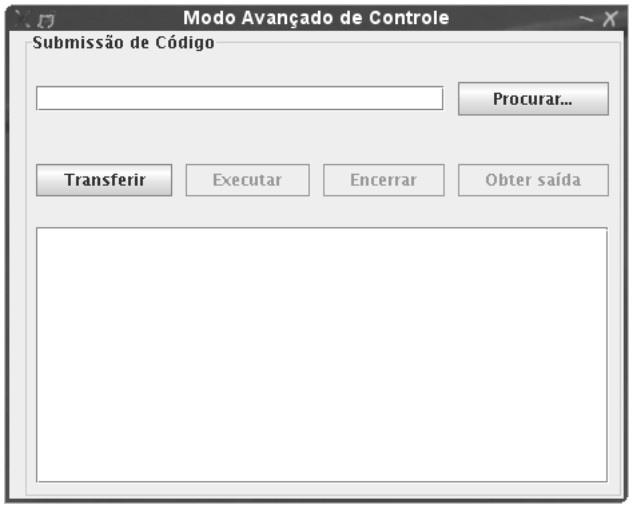
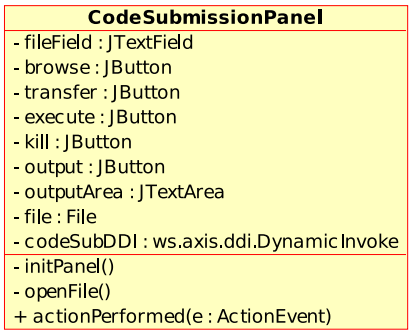


Fig. 5.6: Diagrama de Classes do Painel de Controle das Câmeras



(a) Painel de Submissão de Código



(b) Acompanhamento e Operação Básica do Robô

Fig. 5.7: *Front-end* para Serviço de Submissão de Código

em uma lista ligada a seqüência de destinos definidos pelo usuário, enquanto a segunda faz a interpretação dos dados desta lista, definindo ângulo e distância necessárias para cada objetivo. Esta análise é realizada pela *thread FollowMouseTarget*, uma classe interna à classe *Joystick*.

A relação entre as diversas classes citadas tanto para o mapeamento quanto para a locomoção do robô pode ser observada na figura 5.8. A interface com o mapa e o painel de movimentação básica é exibido na figura 5.3(b).

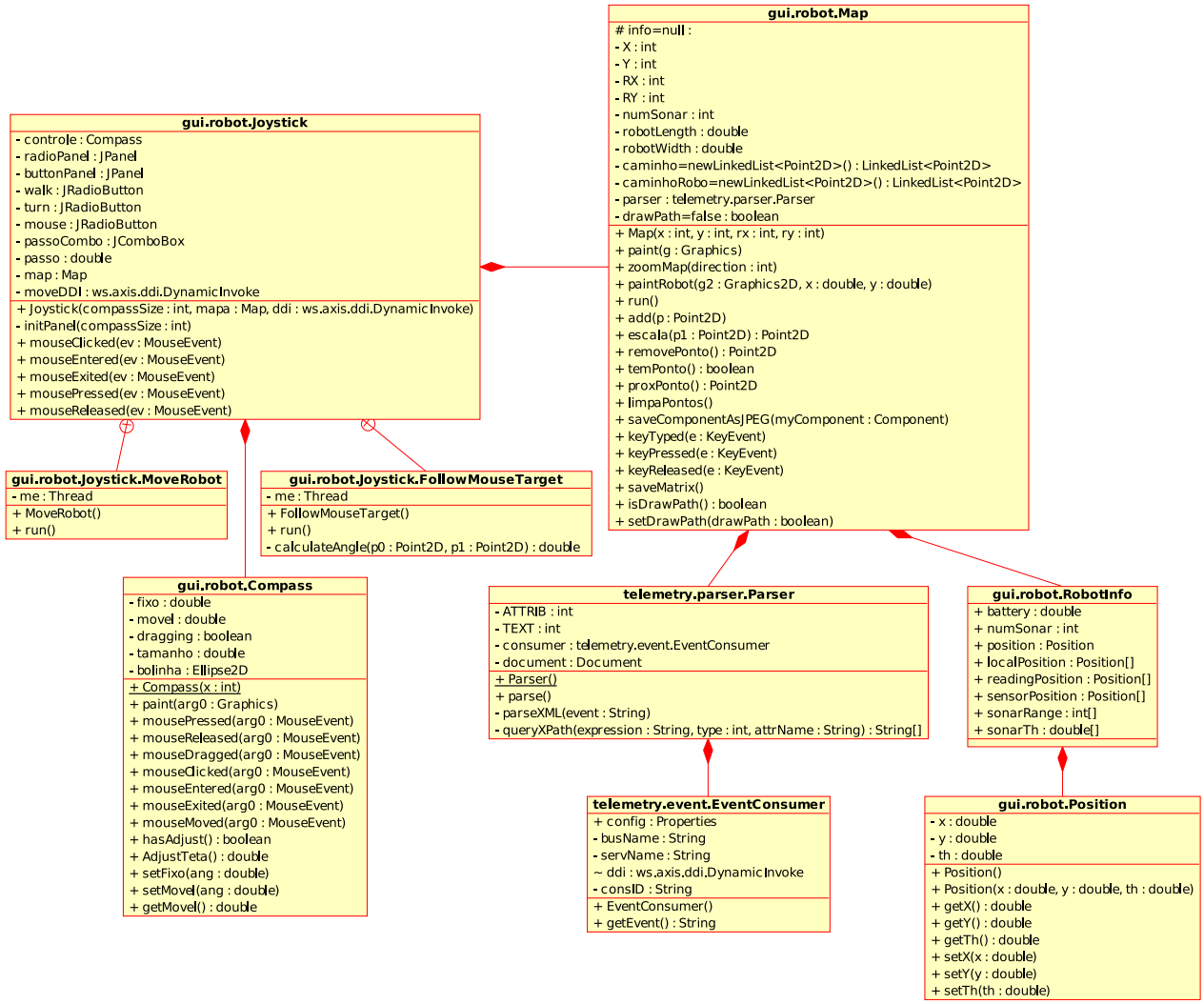


Fig. 5.8: Diagrama de Classes do Painel de Controle e Acompanhamento do Robô

5.2 Composição de Serviços

Do ponto de vista prático, um único Serviço Web isolado dificilmente fornecerá todas as funcionalidades necessárias à aplicação em desenvolvimento. Para satisfazer tais necessidades é fundamental a utilização de diversos serviços, de forma a complementar as funcionalidades requeridas.

Para isso, deve-se adotar um mecanismo de composição dos Serviços que consiga integrar os diversos serviços definindo um fluxo de execução que possa tirar proveito dos Serviços necessários.

Tal composição pode ser feita de maneira "*ad-hoc*", deixando toda a lógica do fluxo da composição no próprio código da aplicação em desenvolvimento.

Para composições que envolvem um pequeno número de Serviços Web esta abordagem é satisfatória e provê um mecanismo direto de obtenção dos resultados esperados.

À medida que a aplicação cresce, entretanto, pode tornar-se inviável a coordenação do fluxo de execução diretamente no código da aplicação. Além disso, a separação da lógica de fluxo de informações entre os diversos serviços da lógica da aplicação pode facilitar o desenvolvimento da mesma.

Desta forma, esta composição pode ser realizada por meio da linguagem BPEL. A linguagem BPEL, descrita na seção 2.4, fornece uma maneira direta de implementar e gerenciar o fluxo de interações com os diversos Serviços Web envolvidos (parceiros), bem como a troca de informações entre tais Serviços. Este fluxo, denominado Processo de Negócio, pode, então, ser acessado por meio de sua interface WSDL, sendo facilmente incorporado à aplicação em desenvolvimento. Além disso, o processo de negócio pode ser iniciado de forma assíncrona, facilitando a continuidade da parcela do fluxo lógico da aplicação que independe do processo de negócio disparado.

5.3 Exemplos

Para ilustrar a utilização dos mecanismos de composição descritos, são apresentados exemplos de experimentos implementados usando a composição dos serviços de interação desenvolvidos. Trata-se de um exemplo para a composição "*ad-hoc*" que contempla um experimento de deteção do caminho e de um experimento para a composição por meio de BPEL com a finalidade de realizar um mapeamento fotográfico do ambiente em que o robô está imerso.

Além dos dois experimentos que serão citados como exemplos, foram desenvolvidos ainda seis outros experimentos para o laboratório de robótica implementado. Trata-se dos experimentos de:

Navegação Básica: ilustrado na figura 5.3(b), representa a interface básica para acompanhamento de todos os experimentos.

Ações: Conjunto de nove ações disponibilizadas no robô também acessíveis e configuráveis por meio da interface de navegação básica.

Navegação em Ambientes Não-Estruturados: experimento para movimentação em ambientes não mapeados. Utiliza o algoritmo Campos Potenciais [53], onde o ponto de destino atrai o robô e os obstáculos o repelem. A resultante destas duas forças opostas definem dinamicamente o caminho seguido pelo robô.

Navegação em Ambientes Estruturados: experimento empregando o algoritmo A* [54] para locomoção em ambientes previamente mapeados. Estabelece a trajetória de menor distância entre o ponto de origem e o destino.

Grade de Ocupação: mapeamento estatístico do ambiente baseado nos valores de leituras dos sonares. Provê dois tipos de informações: regiões com probabilidades de existência de obstáculos e regiões com probabilidades de ausência de obstáculos.

Submissão de Código: representado pela figura 5.7(a), trata-se de um experimento avançado no qual o usuário pode submeter código previamente compilado para execução direta no servidor ligado ao robô.

5.3.1 Deteção do Caminho

O experimento de deteção de caminho permite a determinação da rota a ser percorrida pelo robô por meio da escolha da cor que este deve seguir.

São utilizados neste experimento os serviços de visão e de movimentação do robô. Ambos os serviços são descobertos dinamicamente por meio do Serviço de Descoberta (UDDI). O serviço de visão tem a função de fornecer as imagens utilizadas para a definição das tomadas de decisões do robô. O serviço de movimentação é invocado quando deseja-se alterar a velocidade do robô, ou ainda, corrigir sua trajetória em busca da fita na cor escolhida.

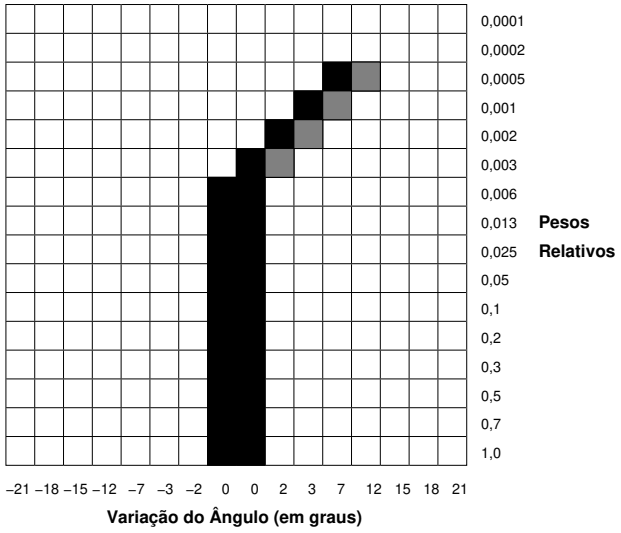


Fig. 5.9: Matriz para Cálculo da Variação do Ângulo

O objetivo deste experimento é manter a fita no centro da imagem da câmera. Deste objetivo resulta, então, a definição da velocidade e de ângulos de correção. A atualização destes parâmetros ocorre uma vez a cada segundo. O ciclo de controle pode ser descrito da seguinte forma:

1. Uma imagem é obtida do serviço de visão.

2. A imagem obtida é processada com o intuito de extrair a borda da fita que indica o caminho.

3. O resultado obtido é mapeado em uma matriz de navegação, à qual são aplicados vetores de pesos e ângulos.

4. O processamento desta matriz retorna a variação de ângulo e a velocidade a serem aplicadas no robô.

A borda fita na imagem obtida pela câmera de bordo é extraída por meio de um algoritmo para extração de borda [55]. Este algoritmo tem como entrada a imagem e uma matriz de transformação quadrada de dimensão três com valores do tipo ponto flutuante. A atribuição dos valores corretos aos elementos desta matriz determinam a qualidade da deteção. Esses valores variam em função da diferença de cor e contraste entre a fita e o piso do ambiente onde está sendo realizado o experimento. O resultado desta extração de borda é uma imagem monocromática em que a borda é mapeada para a cor preta, enquanto o restante da figura apresenta-se na cor branca.

Ao mesmo tempo, a imagem original é mapeada para uma nova imagem monocromática na qual a ocorrência da cor escolhida (com tolerância de 20%) é convertida para a cor preta e as demais cores são transformadas em branco. Este imagem monocromática e a imagem com a borda extraída são divididas em uma matriz de 16 por 16 cada. Em cada elemento da matriz é colocada a porcentagem relativa da ocorrência da cor preta (que corresponde à cor escolhida pelo usuário) . Os elementos dessas duas matrizes são multiplicados e a matriz resultante é normalizada em relação ao seu maior elemento. A somatória da multiplicação de cada elemento desta matriz pelo correspondente ângulo e peso (figura 5.9) define a variação de ângulo aplicada sobre o robô. A velocidade é escolhida pelo usuário antes do início do movimento, e é atenuada para 60% do valor definido quando a variação do ângulo é superior a 20 graus. Os valores dos pesos e ângulos apresentados na figura 5.9 foram determinados a partir de testes reais com o robô.

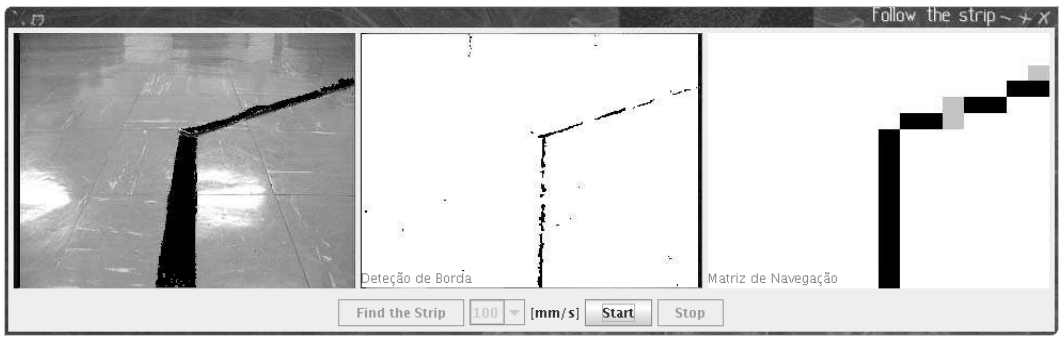


Fig. 5.10: Interface de Execução do Experimento

A composição dos serviços utilizados, neste caso, é realizada diretamente no código da aplicação, ficando a cargo do desenvolvedor todo a localização, a configuração e a utilização de cada Serviço Web envolvido.

5.3.2 Mapeamento Fotográfico

Um outro experimento desenvolvido refere-se a um mapeamento fotográfico do ambiente com a finalidade de reconhecer o local onde o robô está inserido. Tal reconhecimento é realizado por meio de rotações do robô seguidas de obtenção de figuras e tratamento de parâmetros capturados nos sonares. Além disso, um vídeo da realização do experimento também é gravado pela câmera panorâmica.

Este experimento foi desenvolvido utilizando a linguagem BPEL como mecanismo de composição. As figuras 5.11, 5.13 e 5.15 exibem o digrama que descreve o fluxo de execução do

processo de negócio implementado.

Trata-se de um processo de negócio assíncrono que disponibiliza, por meio de sua interface WSDL, portas de início de sua execução e portas de *callback*, para verificação do término.

Para a execução deste experimento são combinados os seguintes Serviços Web (parceiros): serviço de movimentação do robô, serviço de telemetria, serviço de visão do robô e serviço de câmera panorâmica. Existe ainda o parceiro que representa o cliente do processo de negócio.

O processo de negócio é formado por dois escopos (figura 5.11): um escopo de inicialização dos dispositivos de captura de vídeo (câmeras panorâmica e embarcada) e o escopo principal, no qual encontra-se, de fato, a lógica da composição.

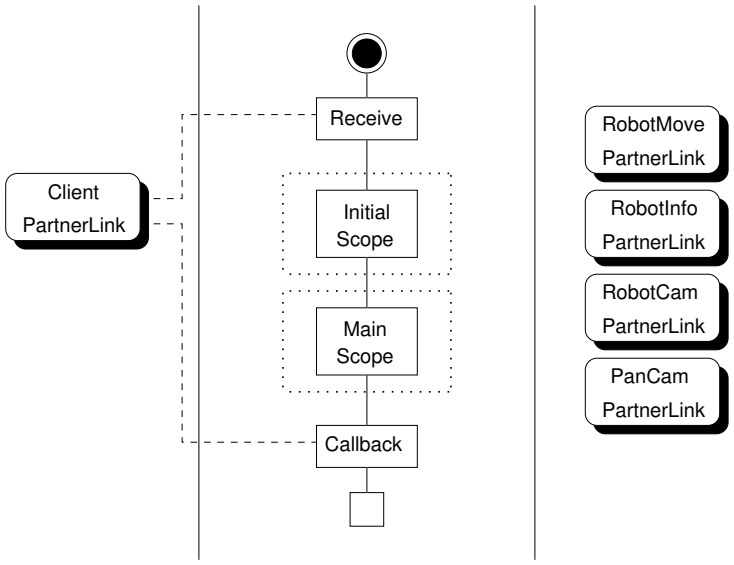


Fig. 5.11: Mapeamento do Ambiente - Visão Geral

O usuário define como parâmetros de entrada à execução do processo os valores do deslocamento angular do robô a cada interação, o intervalo de espera entre interações, bem como o nome que deseja dar ao vídeo obtido deste mapeamento fotográfico. O arquivo XSD que descreve a entrada do experimento pode ser visto na figura 5.12.

```
1 <?xml version="1.0" encoding="ISO-8859-15"?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3             xmlns="http://www.gigabot.org/input"
4             targetNamespace="http://www.gigabot.org/input"
5             elementFormDefault="qualified" >
6   <xsd:element name="turnAmount" type="xsd:double"/>
7   <xsd:element name="turnInterval" type="xsd:int"/>
8   <xsd:element name="videoName" type="xsd:string"/>
9 </xsd:schema>
```

Fig. 5.12: Definição da Entrada do Processo de Negócio

Uma vez em execução, o escopo de inicialização (figura 5.13) invoca o método *initDevice* nos

parceiros *PanCam PartnerLink* e *RobotCam PartnerLink*. O sucesso nesta fase dá continuidade ao fluxo. Caso algum erro seja detectado, uma exceção é lançada e o tratamento desta exceção atualiza a variável de retorno com uma mensagem de erro, para que o *callback* realizado pelo cliente seja notificado do ocorrido.

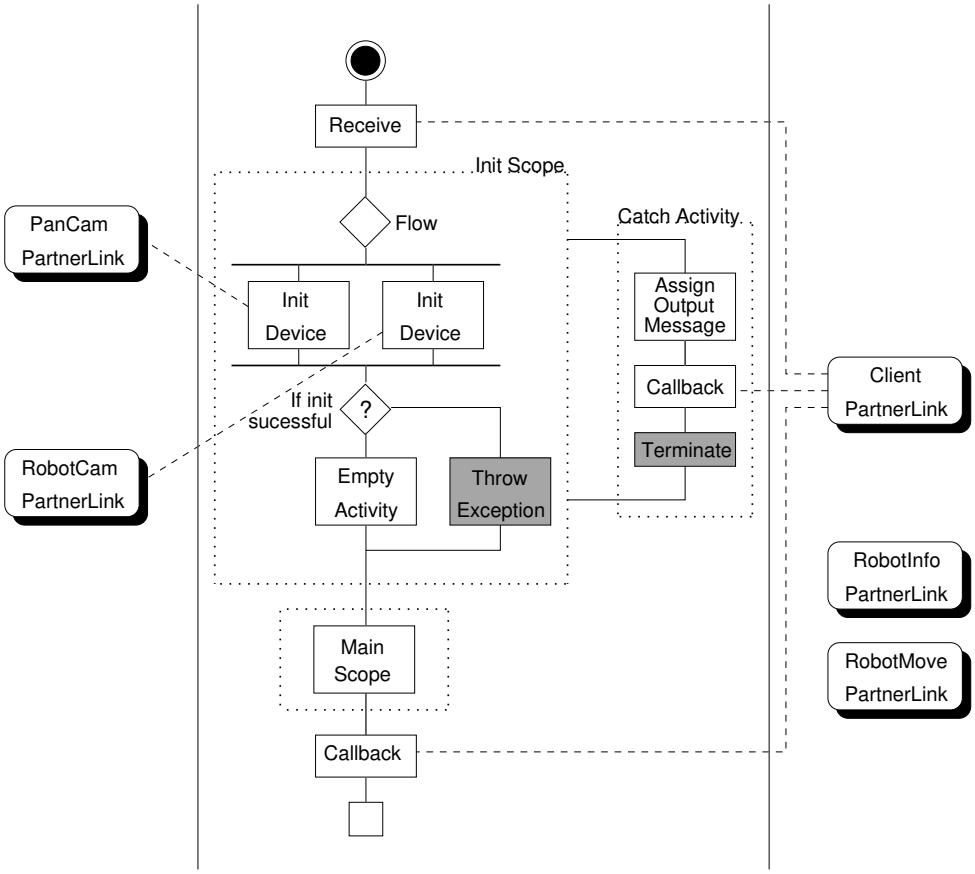


Fig. 5.13: Mapeamento do Ambiente - Escopo de Inicialização

O escopo principal (figura 5.15) é responsável pelo ciclo de execução do mapeamento fotográfico. Inicialmente é armazenado, em uma variável local ao escopo, o valor da variação de ângulo para cada ciclo. Em seguida é invocado o método *getInfo* do parceiro *RobotInfo PartnerLink*. As informações obtidas são gravadas em um vetor, representando o alcance dos 16 sonares presentes no robô. Esses dados são então analisados por um trecho de código Java, para a determinação dos alcances máximo e mínimo. A gravação do vídeo é iniciada, bem como é obtida a primeira figura da câmera embarcada. Por fim, entra-se no laço de controle (até atingir 360 graus), que espera o intervalo definido pelo usuário e executa novamente a obtenção de informação (método *getInfo*), a atualização do vetor de alcances dos sonares e do alcances máximo e mínimo, e uma nova captura da imagem da câmera embarcada. Ao término da atividade *while*, a gravação é encerrada e ao vídeo codificado é atribuído o nome escolhido pelo usuário. A variável global de saída é então atualizada com a URL do vídeo de acompanhamento da execução do experimento, os valores de alcances máximo e mínimo dos sonares, bem como uma variável que informa o sucesso ou fracasso das atividades do experimento.

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.gigabot.org/output"
            targetNamespace="http://www.gigabot.org/output"
            elementFormDefault="qualified" >
  <xsd:element name="success" type="xsd:boolean"/>
  <xsd:element name="videoURL" type="xsd:string"/>
  <xsd:complexType name="minMaxRanges">
    <xsd:sequence>
      <xsd:element name="minRange" type="xsd:double"/>
      <xsd:element name="maxRange" type="xsd:double"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Fig. 5.14: Definição do Retorno do Processo de Negócio

Ao usuário cabe desenvolver um cliente para o Serviço Web que representa o processo de negócio, e invocar o método de inicialização (com os parâmetros de entrada) e o método de *callback* (para verificação do encerramento da instância do processo de negócio). A variável retornada por este método (composta pelos parâmetros já citados) possui a estrutura descrita pelo arquivo XSD apresentado na figura 5.14.

5.4 Avaliação

WebLabs são aplicações onde altas taxas de informação são trocadas entre o usuário remoto e o WebLab. O usuário recebe do WebLab fluxos de vídeo, áudio, imagens e telemetria. O WebLab recebe do usuário fluxos de controle e, em certos casos, fluxos de vídeo e áudio. O atraso imposto pela rede tem influência marcante no experimento, notadamente quando o controle do experimento executa no computador do usuário. A fim de avaliar a interatividade do WebLab apresentado acima, um experimento foi instrumentado e avaliado em quatro cenários de acesso:

1. Acesso local: neste cenário o usuário se encontra nas dependências do Web Lab, acessando-o por meio de uma rede local cabeada ou sem fio.

2. Acesso no campus: neste cenário o usuário se encontra no campus da instituição que mantém WebLab, acessando-o por meio de uma rede de campus sem priorização de tráfego.

3. Acesso via rede privada virtual (VPN): neste cenário o usuário se encontra em outra instituição conectada à instituição que mantém o WebLab por VPN com enlaces dedicados.

4. Acesso residencial: neste cenário o usuário se encontra em sua residência e acessa o WebLab por meio de cabo ou ADSL.

O experimento de navegação por visão foi instrumentado para monitorar a banda de rede utilizada pelo experimento e o limite de desempenho do mesmo. Este limite é dado pela velocidade máxima com que o robô consegue seguir uma fita com dois segmentos retos dispontos em ângulo de 45^o.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
150	748,90	716,86	125,64	22.000	23.591,40
160	717,19	680,24	125,54	22.000	23.522,97
170	741,06	561,95	125,21	22.000	23.428,22
180 (max)	731,80	553,86	126,04	22.000	23.411,70
Média	734,74	628,23	125,61	22.0000	23.488,57

Tab. 5.1: Desempenho para Rede Ethernet de 100 Mbps.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
100	495,01	507,28	122,16	15.200	16.324,46
150	494,33	484,53	122,51	15.200	16.301,37
170 (max)	502,74	416,32	122,72	15.200	16.241,78
180	475,93	436,66	123,23	15.200	16.235,81
Média	492,00	461,20	122,66	15.200	16.275,86

Tab. 5.2: Desempenho para Rede 802.11g de 54 Mbps.

Acima desta velocidade, o robô não consegue processar imagens com freqüência suficiente para corrigir a rota na junção dos dois segmentos da fita (situação em que o robô "perde" a fita). Esta velocidade máxima depende da freqüência do ciclo de controle, que por sua vez depende da vazão e do atraso da rede, bem como da capacidade de processamento no terminal do usuário. A vazão determina o fluxo máxmo de telemetria que se consegue obter do robô. O atraso da rede combinado com a capacidade de processamento do terminal do usuário determinam a atraso na atuação do controle. Como terminal de acesso foi utilizado em todos os cenários um computador portátil Dell D520 com processador Intel Centrino Duo de 1.8 Mhz e 2GB de memória RAM.

Os resultadas para os cenários avaliados estão apresentados nas tabelas 5.1, 5.2, 5.5 e 5.6. Na tabela 5.1, acesso via conexão cabeada à rede local do WebLab, a velocidade máxima do robô foi 180 mm/s, com freqüência de controle de 3 ciclos/s. Na tabela 5.2, acesso via conexão sem fio à rede local do Web Lab, a velocidade máxima foi de 170 mm/s, com freqüência de controle também de de 3 ciclos/s. Como esperado, o desempenho do experimento para redes locais Ethernet com e sem fio foram semelhantes, dado que ambas possuem largura de banda e atraso adequados para aplicações de Web Labs. Graças a boa largura de banda das redes locais, a apresentação do fluxo do fluxo multimídia não impacta no desempenho do experimento nestas redes.

Na tabela 5.3, acesso via rede de campus, obteve-se velocidade máxima de 90 mm/s com freqüência do ciclo de controle de 2 ciclos/s. A apresentação do fluxo do fluxo multimídia teve pouco impacto no desempenho do experimento.

A rede de campus utilizada foi a Uninet que interliga as unidades de ensino, pesquisa e administração no campus da Universidade Estadual de Campinas. Esta rede possui núcleo de roteadores conectados via enlaces Ethernet de 1 Gbit/s. As conexões com as unidades se dá por meio de enlaces Ethernet de 100 Mbit/s. Esta rede não permite ao usuário reservar recursos.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
50	60,76	154,95	46,90	1.000	1.262,61
80	57,46	186,85	69,17	1.000	1.313,48
90 (max)	67,18	124,88	61,86	1.000	1.253,92
100	60,33	221,29	77,22	1000	1.358,84
120	71,32	140,78	50,01	1.000	1.262,11
Média	63,41	165,75	61,03	1.000	1.290,19

Tab. 5.3: Desempenho para Rede de Campus.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
150	680,19	604,76	285,58	22.000	23.570,53
180	671,66	551,21	290,11	22.000	23.512,98
190	706,42	561,58	291,90	22.000	23.559,90
200 (max)	735,29	692,21	285,18	22.000	23.712,68
210	681,87	543,89	297,89	22.000	23.523,65
Média	695,08	590,73	290,13	22.000	23.575,95

Tab. 5.4: Desempenho para VPN (rede Giga).

Na tabela 5.4, acesso via VPN, observa-se um desempenho ligeiramente superior àquele obtido para rede local. Aumentando-se a freqüência do ciclo de controle para 4 ciclos/s, obteve-se uma velocidade máxima de 200 mm/s. O aumento da freqüência do ciclo de controle foi possível graças ao baixíssimo atraso propiciado pela VPN. Graças à grande largura de banda e enlaces dedicados, a apresentação do fluxo do fluxo multimídia não impacta no desempenho do experimento para esta rede.

A rede utilizada para teste foi a rede Giga operada pela Rede Nacional de Ensino e Pesquisa (RNP) em parceria com a Fundação CPqD. A rede Giga conecta instituições no eixo São Paulo-Rio de Janeiro com enlaces em fibra ótica de 1 Mbps (Gigabit Ethernet). O acesso foi realizado a partir do Instituto de Computação da Unicamp, estando o Web Lab conectado ao nó da rede Giga na Faculdade de Engenharia Elétrica e de Computação da Unicamp. No roteador de acesso à rede Giga foram utilizadas 4 portas de 100 Mbps. Estas portas conectam o servidor do Web Lab, o ponto de acesso IEEE 802.11g para conexão ao robô, a câmera panorâmica e o recurso multimídia.

Na tabela 5.5, acesso residencial via ADSL de 2 Mbps, a velocidade máxima foi de apenas 30 mm/s, com freqüência de controle de 0,5 ciclo/s. Nota-se aqui o efeito da baixa largura de banda e o elevado atraso do acesso residencial. Apesar da velocidade máxima de acesso ser de 2 Mbps, a banda máxima utilizada foi de 1 Mbps, devido, provavelmente, a gargalos nas diversas redes que separam o provedor de acesso e o Web Lab.

Na tabela 5.6 a velocidade máxima foi de 70 mm/s, com freqüência de controle de 1 ciclo/s. Este desempenho superior ao caso anterior se deve à não apresentação do fluxo multimídia no terminal do usuário. Diferentemente da rede de campus, a apresentação de fluxo multimídia teve impacto significativo do desempenho do experimento.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
30 (max)	35,90	92,42	51,28	800	979,61
40	62,03	128,85	63,21	800	1.054,09
60	47,37	106,13	68,80	800	1.022,30
80	31,59	54,30	23,60	800	909,48
100	35,31	120,09	46,51	800	1001,90
Média	42,44	100,36	50,38	800	993,48

Tab. 5.5: Desempenho para Rede ADSL de 2 Mbps com Recurso Multimídia.

Vel. (mm/s)	Telemetria (Kbps)	Câm. Robô (Kbps)	Câmera Pan. (Kbps)	Rec. Mutimídia (Kbps)	Total (Kbps)
70 (max)	49,09	96,92	54,34	0	200,34
80	54,28	94,71	49,63	0	198,62
100	50,25	85,96	35,68	0	171,90
Média	51,20	92,53	46,55	0	190,29

Tab. 5.6: Desempenho para Rede ADSL de 2 Mbps Sem Recurso Multimídia.

Em redes com restrições de largura de banda e atraso, como as redes roteadas de campus sem capacidade de reserva de recursos e redes de acesso residenciais ADSL, pode-se empregar o esquema da figura 5.16 (parte superior). Para estas redes, o controle do experimento deve executar no WebLab que disponibiliza uma interface Web para o usuário. Por meio desta interface, o usuário pode controlar o experimento e acompanhar sua execução. Apesar desta solução ser comum nas implementações de WebLabs, o controle descentralizado no domínio do usuário (parte inferior da figura 5.16) se constitui numa solução muito mais rica, onde o usuário pode programar seu próprio experimento por meio de composição de serviços Web.

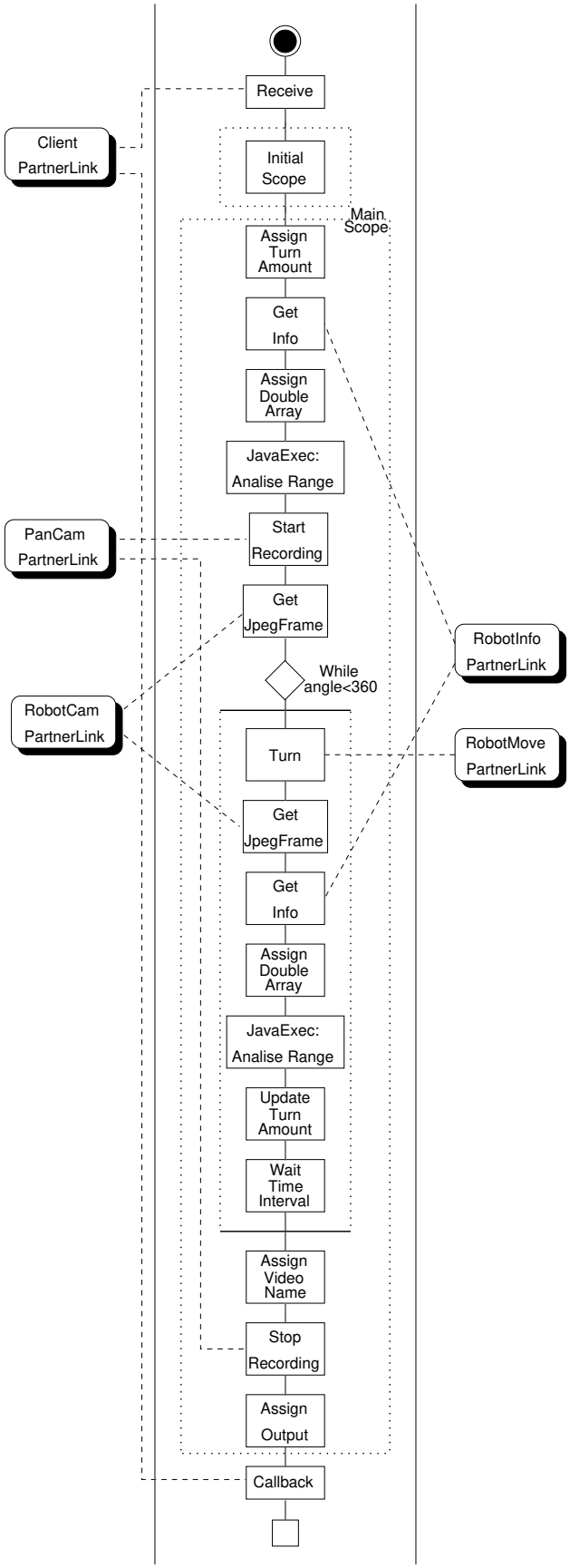


Fig. 5.15: Mapeamento do Ambiente - Escopo Principal

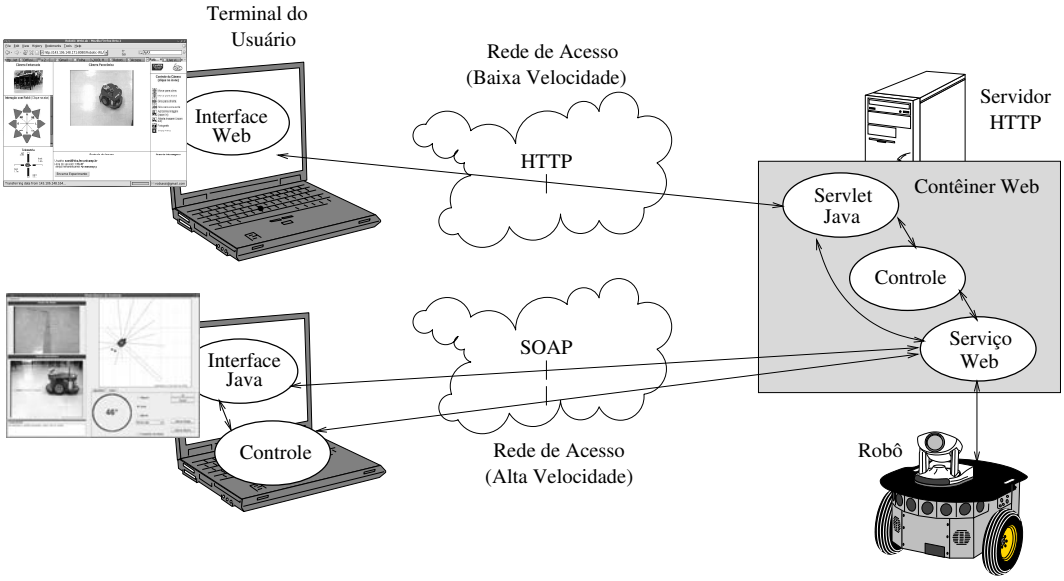


Fig. 5.16: Formas de Acesso ao WebLab em Baixa Velocidade (acima) e em Alta Velocidade (abaixo).

Capítulo 6

Conclusões

Este capítulo apresentas as considerações finais sobre o trabalho. Inicialmente é feita uma retrospectiva das contribuições do trabalho desenvolvido. Em seguida, é descrita uma avaliação da arquitetura proposta e por fim são apresentados possibilidades de trabalhos futuros.

6.1 Retrospectiva das Contribuições

O desenvolvimento de uma arquitetura orientada a serviços para WebLabs contribui, em termos gerais, para a concepção e o desenvolvimento de aplicações capazes de incentivar e disseminar o ensino e aprendizado eletrônico como grande mecanismo de auxílio na formação superior. Além disso, o compartilhamento de recursos possibilitaria seu acesso por parte de instituições educacionais incapazes de adquirirem tais recursos ou equipamentos. Outras contribuições podem ainda ser destacadas:

1. Criação de um modelo para implementação de WebLabs, onde o foco pode ser direcionado para a sessão de interação, uma vez que as sessões de acesso e comunicação independem do domínio do WebLab.

2. A arquitetura apresentada utiliza tecnologias neutras.

3. Validação da arquitetura com a construção do GigaBOT-WL, apresentado no capítulo 5. Trata-se de um WebLab para o domínio da robótica, com diversos experimentos e serviços desenvolvidos.

6.2 Avaliação

Com a modelagem e o desenvolvimento da arquitetura proposta nesta dissertação foi possível avaliar as diversas possibilidades e abordagens que ensino a distância pode propiciar.

A implementação do modelo apresentado possibilitou validação do mesmo, bem como uma confirmação de sua robustez, por meio do uso de sessões e credenciais para coordenar a interação entre participantes e o WebLab.

O desenvolvimento de um WebLab específico para o domínio de robótica trouxe, além do conhecimento adquirido na área, a possibilidade de tornar o GigaBOT-WL um modelo para o aprendizado de robótica a distância.

Além disso, pôde-se demonstrar que a utilização do paradigma de computação orientada a serviço certamente irá impactar positivamente em muitas aplicações, notadamente aplicações que se distribuem por múltiplos domínios administrativos (federações). No caso de WebLabs, as desvantagens das implementações atuais identificadas na seção 1.1 podem ser eliminadas ou minimizadas:

1. A disponibilidade apenas em ambiente local pode ser eliminada com o uso de protocolos universalmente aceitos como HTTP.

2. HTTP Seguro (HTTPS) apresenta um solução relativamente simples e bem estabelecida para segurança.

3. Apesar do protocolo HTTP ser ineficiente para o transporte de fluxos multimídia em tempo real, esta deficiência é minimizada pelo aumento constante da capacidade das redes de comunicação.

4. A disponibilidade limitada de recursos ou ausência de controle de acesso pode ser minimizada pela implementação de serviços de acesso capazes de coordenar múltiplas instâncias de um mesmo recurso e oferecer políticas de acesso capazes de gerenciar uso mediante reserva, papéis, permissões, grupos e sessões.

5. Serviços Web eliminam a necessidade de utilização de sistemas de software proprietários no terminal do usuário. Além de empregar padrões 100% abertos, Serviços Web podem ser acessados a partir de múltiplos terminais, inclusive terminais móveis como computadores de mão e telefones celulares.

6. Graças a um mecanismo padrão de composição (ausente nos demais paradigmas de computação distribuída), computação orientada a serviço permite a criação de novos serviços por meio da combinação de serviços existentes. No caso de WebLabs, novos experimentos podem ser criados via composição de serviços existentes como demonstrado no capítulo 5.

6.3 Utilização da Arquitetura Proposta em Outros Domínios

A apresentação do modelo conceitual para criação de WebLabs (capítulo 3), no qual a relação entre participantes e laboratório é controlada pelas três sessões introduzidas (acesso, interação e comunicação), mostra claramente a transparência deste modelo no que diz respeito ao domínio do WebLab desenvolvido. A independência de domínio das sessões de acesso e comunicação facilitam e simplificam a criação de novos WebLabs.

A tarefa de desenvolvimento de um novo WebLab é, desta forma, basicamente a atividade de criação de experimentos a partir dos serviços representativos dos equipamentos/ferramentas específicos para o domínio do WebLab. Para isto, deve-se criar mecanismos de conexão entre os equipamentos que devem ser apresentados na forma de serviços, incluindo protocolos e padrões de acionamento, acompanhamento e obtenção de dados deste equipamento. A partir daí deve-se mapear

as funcionalidades e informações destes equipamentos para o acesso através de Serviços Web. A figura 6.1 ilustra o mecanismo de criação de um experimento a partir dos Serviços Web que representa um equipamento.

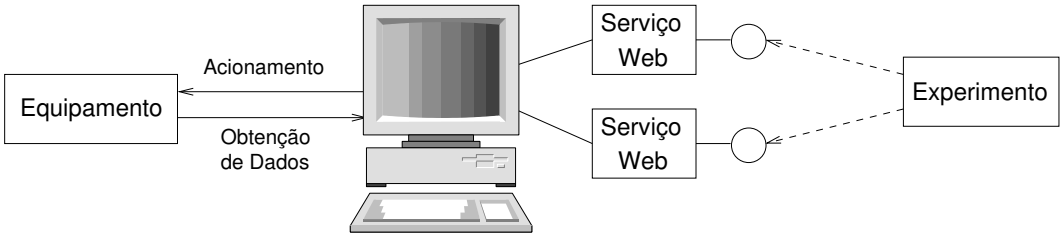


Fig. 6.1: Processo de Desenvolvimento da Sessão de Interação

Uma vez desenvolvidos os experimentos, tem-se implementada a Sessão de Interação específica para o domínio desejado. Para finalizar a implementação do WebLab, deve-se integrar os experimentos à Sessão de Acesso através da instanciação da classe *LoginManager*, apresentada no capítulo 5, em cada experimento. Esta classe é responsável pela verificação de reserva para acesso ao experimento, feita pela mesma interface Web apresentada para o WebLab de robótica. Os experimentos podem ainda utilizar o Serviço de Difusão para notificação de eventos e comunicação com dispositivos multimídias.

Desta forma, a arquitetura proposta possibilita a criação de WebLabs de maneira direta, desacoplando a implementação das tarefas administrativa (acesso) e de comunicação.

6.4 Trabalhos Futuros

Apesar do desenvolvimento da arquitetura proposta apresentar-se como uma solução real e bem estabelecida para a criação de WebLabs, existem algumas possibilidades de extensão e melhoramentos que podem tornar o uso de WebLabs mais largamente aceito.

Um primeiro passo seria a adaptação do modelo apresentado para a operação de maneira federada. Embora o modelo já preveja esta possibilidade por meio da relação do WebLab com ele próprio (figura 3.1), alterações devem ser feitas no sentido de adaptar as sessões. Além disso, deve-se adicionar a entidade representativa da federação, a qual deverá ter a responsabilidade de organizar a relação entre WebLabs, bem como a troca de informações entre estes, no que diz respeito, por exemplo, a mapeamentos de credenciais de um WebLab para outro (ontologias). Este trabalho encontra-se em andamento no escopo de uma dissertação de mestrado.

Outra melhoria seria a criação de mecanismos que possibilitassem a criação e edição de experimentos de maneira intuitiva. Isto abriria o leque de serviços disponíveis e tornaria mais rico o uso do WebLab. Para tanto, deve-se prover uma forma de mapear as informações necessárias à criação de novos experimentos para uma linguagem que possa representar o fluxo do processo de negócio correspondente ao novo experimento.

A análise de desempenho no terminal do usuário, no que diz respeito a fatores como carga de CPU, tempo de *download* dos arquivos JAR, consumo de banda durante a execução do experimento, tempo de resposta, dentre outros, não foi realizada. Tal análise poderia fornecer dados importantes

para melhoras e adaptações na arquitetura, sendo, por isso, um estudo que não dever ser deixado de lado.

Um último trabalho futuro contempla a possibilidade de uma avaliação pedagógica da arquitetura, por exemplo, por meio de sua utilização em um curso de graduação avançado ou de pós-graduação.

Referências Bibliográficas

[1] M. Exel, S. Gentil, and D. Rey. Simulation workshop and remote laboratory: two web-based trainig approaches for control. In *Proceedings of the 2000 American Control Conference*, June 2000.

[2] D. A. Miele, B. Potsaid, and J. T. Wen. An internet-base remote laboratory for control education. In *Proceedings of the 2001 American Control Conference*, June 2001.

[3] M. Casini, D. Prattichizzo, and A. Vicino. The automatic control telelab. In *IEEE Control Systems Magazine*, volume 24, June 2004.

[4] D. Z. Deniz, A. Bulancak, and G. Özcan. A novel approach to remote laboratories. In *ASEE/IEEE Frontiers in Education Conference*, November 2003.

[5] J. Sanchéz, S. Dormido, R. Pastor, and F. Morilla. A java/matlab based envorinment for remote control system laboratories: Illustrated with an inverted pendulum. In *IEEE Transations on Education*, volume 47, pages 321–329, August 2004.

[6] A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani. Remlab: a java-based remote, didatic measurement laboratory. In *Internal Symposium on Virtual and Intelligent Measurements Systems*, May 2002.

[7] M. L. Corradini, G. Ippoliti, T. Leo, and S. Longhi. An internet-based laboratory for control education. In *Proceedings of the IEEE Conference on Decision and Control*, December 2001.

[8] C. C. Ko, B. M. Chen, J. Chen, Y. Zhuang, and K. C. Tan. Development of a web-based laboratory for control experiments on a coupled tank apparatus. In *IEEE Transactions on Education*, volume 44, pages 76–86, February 2001.

[9] H. H. Hahn and M. W. Spong. Remote laboratories for control education. In *Proceedings of the IEEE Conference on Decision and Control*, December 2000.

[10] S. Lerman and J. del Alamo. ilab: remote online laboratories. `http://icampus.mit.edu/projects/iLab.shtml`.

[11] V. J. Harward, J. A. del Alamo, and V. S. Choudhary. ilab: a scalable architeture for sharing online experiments. In *Internation Conference on Engineering Education*, October 2004.

[12] E. G. Guimarães, A. T. Maffeis, J. L. Pereira, B. G. Russo, M. Bergerman, E. Cardozo, and M. F. Magalhães. REAL: A Virtual Laboratory for Mobile Robot Experiments. In *First IFAC Conference on Telematics Applications in Automation and Robotics*, volume I, pages 209–214, Weingarten, Germany, July 2001.

[13] L. R. Queiroz. Um Laboratório Virtual de Robótica e Visão Computacional. Master's thesis, Instituto de Computação, UNICAMP, Novembro 1998.

[14] E. G. Guimarães, A. T. Maffeis, J. L. Pereira, B. G. Russo, E. Cardozo, and M. Bergerman M. F. Magalhães. REAL: A Virtual Laboratory for Mobile Robot Experiments. *IEEE Transactions on Education*, 46(1):37–42, February 2003.

[15] E. G. Guimarães, A. T. Maffeis, R. P. Pinto, C. A. Miglinski, E. Cardozo, M. Bergerman, and M. F. Magalhães. REAL: A Virtual Laboratory Built from Software Components. *Proceedings of the IEEE*, 91(3):440–448, March 2003.

[16] E. G. Guimarães, E. Cardozo, M. Magalhães, M. Bergerman, A. T. Maffeis, J. L. Pereira, B. G. Russo, C. A. Miglinsk, and R. P. Pinto. Desenvolvimento de Software Orientado a Componentes para Novos Serviços de Telecomunicações. In *19 Simpósio Brasileiro de Redes de Computadores (SBRC01)*, volume I, Florianópolis, SC, Maio 2001. SBC.

[17] Eliane Gomes Guimarães. *Um modelo de componente para aplicações telemáticas e ubíquias*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação - UNICAMP, 2004.

[18] E. G. Guimarães, E. Cardozo, M. F. Magalhães, W. P. Gomes, R. P. Pinto, and L. F. Faina. CCM-tel- uma Plataforma para Aplicações Telemáticas e Ubíquas. In *22 Simpósio Brasileiro de Redes de Computadores (SBRC04)*, Gramado, RS, Maio 2004. SBC.

[19] Wander E. Carneiro Pimentel Gomes. Uma plataforma de desenvolvimento de software baseado em componentes para dispositivos móveis. Master's thesis, Faculdade de Engenharia Elétrica e de Computação - UNICAMP, 2005.

[20] E. Cardozo and E. G. Guimarães. Relatório do projeto tidia-ae: Tecnologia da informação no desenvolvimento da internet avançada - aprendizado eletrônico. Technical report, FAPESP, 2005.

[21] M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the 4th International Conference on Web Information System Engineering*, 2003.

[22] Thomas Erl. *Service-Oriented Architeture - Concepts, Technology and Design*. Prentice Hall, 1 edition, 2005.

[23] F. Leymann, D. Roller, and M. T. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2):198–211, 2002.

[24] M. N. Huhns and M. P. Singh. Service-oriented computing: key concepts and principles. In *IEEE Internet Computing*, February 2005.

[25] E. Maler et al. Extensible markup language (xml) 1.1. Technical report, W3C, 2004. `http://www.w3.org/TR/2004/REC-xml11-20040204/`.

[26] F. Curbera et al. Unraveling the web services web. In *IEEE Internet Computing*, volume 6, pages 86–93, 2002.

[27] R. Khalaf et al. Understanding web services. In *Practical Handbook of Internet Computing*, chapter 27. Chapman Hall CRC Press, 2005.

[28] M. Gudgin et al. Simple object access protocol (soap). Technical report, W3C, 2003. `http://www.w3.org/TR/soap12`.

[29] H. M. Deitel et al. *Java Web Services for Experienced Programmers*. Prentice Hall, 2003.

[30] E. Christensen et al. Web service description language (wsdl). Technical report, W3C, 2001. `http://www.w3.org/TR/wsdl.html`.

[31] T. Rogers and L. Clement. Oasis uddi specification tc. Technical report, OASIS, 2003. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec`.

[32] Xmethods. `http://www.xmethods.com`.

[33] Bindingpoint. `http://www.bindingpoint.com/default.aspx`.

[34] C. Peltz. Web services orchestration and choreography. In *IEEE Internet Computing*, pages 46–52, October 2003.

[35] T. Andrews et al. Business process execution language for web services version 1.1. Technical report, Microsoft, IBM, BEA, Siebel System, SAP, 2003. `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`.

[36] N. Milanovic and M. Malek. Current solutions for web services composition. In *IEEE Internet Computing*, pages 51–59, November/December 2004.

[37] J. Pasley. How bpel and soa are changing web services development. In *IEEE Internet Computing*, pages 60–67, May/June 2005.

[38] Mobile Robotics. *Pioneer P3-DX*. `http://www.activrobots.com/ROBOTS/p2dx.html`.

[39] Canon. *Network Video Camera VC-C4*. `http://www.usa.canon.com/consumer/controller?act=ModelTechSpecsAct\&fca%tegoryid=158\&modelid=7402`.

[40] Hewlett-Packard. *HP iPAQ h5550*. `http://h20000.www2.hp.com/bizsupport/TechSupport/Home.jsp?\&lang=en\&cc%=us\&prodTypeId=215348\&prodSeriesId=322916\&lang=en\&cc=us`.

[41] Axis Communications. *Network Video Camera Axis 206W*. http://www.axis.com/products/cam_206/index.htm.

[42] Axis Communications. *Network Video Camera Axis 213PTZ*. http://www.axis.com/products/cam_213/index.htm.

[43] Apache Software Foundation. *Apache Tomcat*. http://tomcat.apache.org/.

[44] Apache Software Foundation. *Apache Axis*. http://ws.apache.org/axis/.

[45] Apache Software Foundation. *Apache HTTP Server*. http://httpd.apache.org/.

[46] Oracle. *Oracle BPEL Process Manager*. http://www.oracle.com/technology/products/ias/bpel/index.html.

[47] Apache Software Foundation. *Apache jUDDI*. http://ws.apache.org/juddi/.

[48] Eclipse Foundation. *Eclipse*. http://www.eclipse.org/.

[49] dbXML Group. *dbXML*. http://www.dbxml.com/.

[50] MySQL AB. *MySQL Database*. http://www.mysql.com/.

[51] Oracle. *Oracle Database 10g Express Edition*. http://www.oracle.com/technology/products/database/oracle10g/index.html%.

[52] Sun Microsystems. *Java Web Start*. http://java.sun.com/products/javawebstart.

[53] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1990.

[54] P.H. Winston. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1984.

[55] S.E. Umbaugh. *Computer vision and image processing*. Prentice Hall, 1998.

Apêndice A

Apêndices

A.1 Script BPEL do Mapeamento Fotográfico

```
1  <!--
2      ////////////////////////////////////////
3      // Oracle JDeveloper BPEL Designer
4      //
5      // Created: Wed Jun 21 11:38:16 BRT 2006
6      // Author:  pcoelho
7      // Purpose: Asynchronous BPEL Process
8      ////////////////////////////////////////
9      -->
10 <process name="Mapeamento"
11          targetNamespace="http://bpel.project.gigabot/Mapeamento"
12          xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
13          xmlns:xp20="http://www.oracle.com/XSL/Transform/java/
14                 oracle.tip.pc.services.functions.Xpath20"
15          xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
16          xmlns:ns4="http://10.10.1.24:8080/axis/services/
17                 PanoramicCameraService"
18          xmlns:ns7="http://10.10.1.38/axis/RobotInfoService"
19          xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
20          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
21          xmlns:ns5="http://10.10.1.38/axis/RobotCameraService"
22          xmlns:client="http://bpel.project.gigabot/Mapeamento"
23          xmlns:ns6="http://www.gigabot.org/input"
24          xmlns:ora="http://schemas.oracle.com/xpath/extension"
25          xmlns:ns9="http://10.10.1.38/axis/MoveInfoService"
26          xmlns:ns1="http://10.10.1.38/axis/RobotMoveService"
27          xmlns:ns3="http://www.gigabot.org/output"
28          xmlns:ns2="http://www.gigabot.org"
29          xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
30          xmlns:orcl="http://www.oracle.com/XSL/Transform/java/
31                 oracle.tip.pc.services.functions.ExtFunc"
32          xmlns:ns8="http://www.example.org">
33
34    <!-- ================================================================= -->
35    <!-- PARTNERLINKS                                                      -->
36    <!-- List of services participating in this BPEL process               -->
37    <!-- ================================================================= -->
```

```
38  <partnerLinks>
39    <!-- The 'client' role represents the requester of this service. It is
40         used for callback. The location and correlation information
41         associated with the client role are automatically set using
42         WS-Addressing. -->
43    <bpelx:exec import="org.w3c.dom.Element"/>
44    <partnerLink name="client" partnerLinkType="client:Mapeamento"
45                 myRole="MapeamentoProvider"
46                 partnerRole="MapeamentoRequester"/>
47    <partnerLink name="movePartner" partnerRole="RobotMoveService_Role"
48                 partnerLinkType="ns1:RobotMoveService_PL"/>
49    <partnerLink name="panCamPartner"
50                 partnerRole="PanoramicCameraService_Role"
51                 partnerLinkType="ns4:PanoramicCameraService_PL"/>
52    <partnerLink name="robotCamPartner" partnerRole="RobotCameraService_Role"
53                 partnerLinkType="ns5:RobotCameraService_PL"/>
54    <partnerLink name="infoPartner" partnerRole="MoveInfoService_Role"
55                 partnerLinkType="ns9:MoveInfoService_PL"/>
56  </partnerLinks>
57
58  <!-- ================================================================= -->
59  <!-- VARIABLES                                                         -->
60  <!-- List of messages and XML documents used within this BPEL process  -->
61  <!-- ================================================================= -->
62  <variables>
63          <!-- Reference to the message passed as input during initiation -->
64          <!-- Reference to the message that will be sent back to the
65               requester during callback                                  -->
66    <variable name="inputVariable"
67              messageType="client:MapeamentoRequestMessage"/>
68    <variable name="outputVariable"
69              messageType="client:MapeamentoResponseMessage"/>
70  </variables>
71
72  <!-- ================================================================= -->
73  <!-- ORCHESTRATION LOGIC                                               -->
74  <!-- Set of activities coordinating the flow of messages across the    -->
75  <!-- services integrated within this business process                  -->
76  <!-- ================================================================= -->
77  <sequence name="main">
78    <receive name="receiveInput"
79             partnerLink="client"
80             portType="client:Mapeamento"
81             operation="initiate"
82             variable="inputVariable"
83             createInstance="yes"/>
84
85    <scope name="Init">
86      <variables>
87        <variable name="initPanCam_OutputVariable"
88                  messageType="ns4:initDeviceResponse"/>
89        <variable name="initPanCam_InputVariable"
90                  messageType="ns4:initDeviceRequest"/>
91        <variable name="initRobotCam_InputVariable"
92                  messageType="ns5:initDeviceRequest"/>
93        <variable name="initRobotCam_OutputVariable"
94                  messageType="ns5:initDeviceResponse"/>
95        <variable name="panCamReturn" type="xsd:boolean"/>
```

```
96        </variables>
97        <faultHandlers>
98          <catchAll>
99            <sequence name="catchSequence">
100             <assign name="AssignOutputMessage">
101               <copy>
102                 <from expression="''"/>
103                 <to variable="outputVariable" part="videoURL"/>
104               </copy>
105               <copy>
106                 <from expression="0"/>
107                 <to variable="outputVariable" part="minMaxRanges"
108                     query="/minMaxRanges/ns3:minRange"/>
109               </copy>
110               <copy>
111                 <from expression="0"/>
112                 <to variable="outputVariable" part="minMaxRanges"
113                     query="/minMaxRanges/ns3:maxRange"/>
114               </copy>
115               <copy>
116                 <from expression="false()"/>
117                 <to variable="outputVariable" part="success"
118                     query="/ns3:success"/>
119               </copy>
120             </assign>
121             <invoke name="callBackClient"
122                     partnerLink="client"
123                     portType="client:MapeamentoCallback"
124                     operation="onResult"
125                     inputVariable="outputVariable"/>
126             <terminate name="Terminate"/>
127           </sequence>
128         </catchAll>
129       </faultHandlers>
130       <sequence name="MainScopeSequence">
131         <flow name="InitDevices">
132           <sequence name="InitPanoramicCamera">
133             <invoke name="InitDevice" partnerLink="panCamPartner"
134                     portType="ns4:PanoramicCameraService"
135                     operation="initDevice"
136                     inputVariable="initPanCam_InputVariable"
137                     outputVariable="initPanCam_OutputVariable"/>
138             <assign name="AssignPanCamReturn">
139               <copy>
140                 <from variable="initPanCam_OutputVariable" part="parameters"
141                       query="/ns4:initDeviceResponse/ns4:initDeviceReturn"/>
142                 <to variable="panCamReturn"/>
143               </copy>
144             </assign>
145           </sequence>
146           <sequence name="InitRobotCamera">
147             <invoke name="InitDevice" partnerLink="robotCamPartner"
148                     portType="ns5:RobotCameraService" operation="initDevice"
149                     inputVariable="initRobotCam_InputVariable"
150                     outputVariable="initRobotCam_OutputVariable"/>
151           </sequence>
152         </flow>
153         <switch name="IfInitSuccess">
```

```
154           <case condition=
155                 "(bpws:getVariableData('initRobotCam_OutputVariable',
156                 'parameters','/ns5:initDeviceResponse/initDeviceReturn') =
157                 'true') and
158                 (bpws:getVariableData('panCamReturn') = true())">
159             <empty name="Empty"/>
160           </case>
161           <otherwise>
162             <throw name="forcedTermination"
163                    faultName="bpws:forcedTermination"/>
164           </otherwise>
165         </switch>
166       </sequence>
167     </scope>
168     <scope name="MainScope">
169       <variables>
170         <variable name="turnAmount" element="ns6:turnAmount"/>
171         <variable name="startVideoRecording_InputVariable"
172                   messageType="ns4:startVideoRecordingRequest"/>
173         <variable name="startVideoRecording_OutputVariable"
174                   messageType="ns4:startVideoRecordingResponse"/>
175         <variable name="getJpegFrame_InputVariable"
176                   messageType="ns5:getJpegFrameRequest"/>
177         <variable name="getJpegFrame_OutputVariable"
178                   messageType="ns5:getJpegFrameResponse"/>
179         <variable name="setHeading_InputVariable"
180                   messageType="ns1:setHeadingRequest"/>
181         <variable name="setHeading_OutputVariable"
182                   messageType="ns1:setHeadingResponse"/>
183         <variable name="stopVideoRecording_InputVariable"
184                   messageType="ns4:stopVideoRecordingRequest"/>
185         <variable name="stopVideoRecording_OutputVariable"
186                   messageType="ns4:stopVideoRecordingResponse"/>
187         <variable name="getMoveInfo_InputVariable"
188                   messageType="ns9:getMoveInfoRequest"/>
189         <variable name="getMoveInfo_OutputVariable"
190                   messageType="ns9:getMoveInfoResponse"/>
191         <variable name="doubleArray" element="ns8:doubleArray"/>
192         <variable name="minRange" type="xsd:double"/>
193         <variable name="maxRange" type="xsd:double"/>
194       </variables>
195       <sequence name="MainScopeSequence">
196         <assign name="AssignTurnAmount">
197           <copy>
198             <from variable="inputVariable" part="turnAmount"
199                   query="/ns6:turnAmount"/>
200             <to variable="turnAmount" query="/ns6:turnAmount"/>
201           </copy>
202         </assign>
203         <invoke name="getInfo"
204                 partnerLink="infoPartner"
205                 portType="ns9:MoveInfoService"
206                 operation="getMoveInfo"
207                 inputVariable="getMoveInfo_InputVariable"
208                 outputVariable="getMoveInfo_OutputVariable"/>
209         <assign name="AssignDoubleArray">
210           <copy>
211             <from variable="getMoveInfo_OutputVariable" part="parameters"
```

```
                query="/ns9:getMoveInfoResponse//ns9:sonarRange"/>
            <to variable="doubleArray" query="/ns8:doubleArray"/>
          </copy>
        </assign>
        <bpelx:exec name="analiseRanges" language="Java" version="1.4">
                <![CDATA[System.out.println("Starting Ranges Overview");
                try{
                  Element array = (Element)
                    getVariableData("doubleArray","/ns8:doubleArray");
                  String tmp = "";
                  int i = 0;
                  double MIN = 100000.0, MAX = 0.0;

                  setVariableData("minRange", new Double(MIN));
                  setVariableData("maxRange", new Double(MAX));

                  tmp = array.getNodeValue();
                  String[] values = tmp.trim().split("\n");

                  while(i < values.length) {
                    double value = Double.parseDouble(values[i]);
                    if(MAX < value) MAX = value;
                    if(MIN > value) MIN = value;
                    i++;
                  }
                  setVariableData("maxRange", new Double(MAX));
                  setVariableData("minRange", new Double(MIN));

                  addAuditTrailEntry("Max Range: " + MAX);
                  addAuditTrailEntry("Min Range: " + MIN);

                } catch (Exception e){
                  setVariableData("maxRange", new Double(-1));
                  setVariableData("minRange", new Double(-1));
                  addAuditTrailEntry(e);
                  e.printStackTrace();
                }

                System.out.println("Finished Ranges Overview");]]>
        </bpelx:exec>
        <invoke name="startRecording"
                partnerLink="panCamPartner"
                portType="ns4:PanoramicCameraService"
                operation="startVideoRecording"
                inputVariable="startVideoRecording_InputVariable"
                outputVariable="startVideoRecording_OutputVariable"/>

        <invoke name="getJpegFrame"
                partnerLink="robotCamPartner"
                portType="ns5:RobotCameraService"
                operation="getJpegFrame"
                inputVariable="getJpegFrame_InputVariable"
                outputVariable="getJpegFrame_OutputVariable"/>

        <while name="WhileTurning"
               condition=
               "bpws:getVariableData('turnAmount','/ns6:turnAmount') &lt;360">

```

```
270           <sequence name="WhileSequence">
271                 <wait name="WaitInterval"
272                 for="concat('PT',bpws:getVariableData(
273                 'inputVariable','turnInterval',
274                 '/ns6:turnInterval'),'S')"/>
275                 <assign name="AssignHeadingAmount">
276                 <copy>
277                 <from variable="turnAmount"
278                         query="/ns6:turnAmount"/>
279                 <to variable="setHeading_InputVariable"
280                         part="parameters"
281                         query="/ns1:setHeading/ns1:in0"/>
282                 </copy>
283                 </assign>
284                 <invoke name="turn"
285                         partnerLink="movePartner"
286                         portType="ns1:RobotMoveService"
287                         operation="setHeading"
288                         inputVariable="setHeading_InputVariable"
289                         outputVariable="setHeading_OutputVariable"/>
290                 <invoke name="getJpegFrame"
291                         partnerLink="robotCamPartner"
292                         portType="ns5:RobotCameraService"
293                         operation="getJpegFrame"
294                         inputVariable="getJpegFrame_InputVariable"
295                         outputVariable="getJpegFrame_OutputVariable"/>
296                 <invoke name="getInfo"
297                         partnerLink="infoPartner"
298                         portType="ns9:MoveInfoService"
299                         operation="getMoveInfo"
300                         inputVariable="getMoveInfo_InputVariable"
301                         outputVariable="getMoveInfo_OutputVariable"/>
302                 <assign name="AssignDoubleArray">
303                 <copy>
304                 <from variable="getMoveInfo_OutputVariable"
305                         part="parameters"
306                         query="/ns9:getMoveInfoResponse//ns9:sonarRange"
307                 />
308                 <to variable="doubleArray"
309                         query="/ns8:doubleArray"/>
310                 </copy>
311                 </assign>
312                 <bpelx:exec name="analiseRanges" language="Java"
313                         version="1.4">
314                 <![CDATA[System.out.println("Starting Ranges Overview");
315                 try{
316                   Element array = (Element)
317                     getVariableData("doubleArray","/ns8:doubleArray");
318                   String tmp = "";
319                   int i = 0;
320                   double MIN = 100000.0, MAX = 0.0;
321
322                   setVariableData("minRange", new Double(MIN));
323                   setVariableData("maxRange", new Double(MAX));
324
325                   tmp = array.getNodeValue();
326                   String[] values = tmp.trim().split("\n");
327
```

```
328                   while(i < values.length) {
329                     double value = Double.parseDouble(values[i]);
330                     if(MAX < value) MAX = value;
331                     if(MIN > value) MIN = value;
332                     i++;
333                   }
334                   setVariableData("maxRange", new Double(MAX));
335                   setVariableData("minRange", new Double(MIN));
336
337                   addAuditTrailEntry("Max Range: " + MAX);
338                   addAuditTrailEntry("Min Range: " + MIN);
339
340                 } catch (Exception e){
341                   setVariableData("maxRange", new Double(-1));
342                   setVariableData("minRange", new Double(-1));
343                   addAuditTrailEntry(e);
344                   e.printStackTrace();
345                 }
346
347                 System.out.println("Finished Ranges Overview");]]>
348                 </bpelx:exec>
349                 <assign name="UpdateTurnAmount">
350                   <copy>
351                     <from expression="bpws:getVariableData(
352                           'turnAmount','/ns6:turnAmount') +
353                           bpws:getVariableData('inputVariable',
354                           'turnAmount','/ns6:turnAmount')"/>
355                     <to variable="turnAmount"
356                         query="/ns6:turnAmount"/>
357                   </copy>
358                 </assign>
359           </sequence>
360         </while>
361         <assign name="AssignStopRecordingInput">
362           <copy>
363             <from variable="inputVariable" part="videoName"
364                   query="/ns6:videoName"/>
365             <to variable="stopVideoRecording_InputVariable"
366                 part="parameters"
367                 query="/ns4:stopVideoRecording/ns4:videoName"/>
368           </copy>
369           <copy>
370             <from expression="'Video representing the robot
371                   envorinment visual mapping'"/>
372             <to variable="stopVideoRecording_InputVariable"
373                 part="parameters"
374                 query="/ns4:stopVideoRecording/ns4:comments"/>
375           </copy>
376         </assign>
377         <invoke name="stopRecording"
378                 partnerLink="panCamPartner"
379                 portType="ns4:PanoramicCameraService"
380                 operation="stopVideoRecording"
381                 inputVariable="stopVideoRecording_InputVariable"
382                 outputVariable="stopVideoRecording_OutputVariable"/>
383         <assign name="AssignOutput">
384           <copy>
385             <from expression="concat('http://143.106.148.171:8080/videos/',
```

```
                  bpws:getVariableData('inputVariable','videoName'),'.avi')"/>
            <to variable="outputVariable" part="videoURL"/>
          </copy>
          <copy>
            <from expression="true()"/>
            <to variable="outputVariable" part="success"
                query="/ns3:success"/>
          </copy>
          <copy>
            <from variable="minRange"/>
            <to variable="outputVariable" part="minMaxRanges"
                query="/minMaxRanges/ns3:minRange"/>
          </copy>
          <copy>
            <from variable="maxRange"/>
            <to variable="outputVariable" part="minMaxRanges"
                query="/minMaxRanges/ns3:maxRange"/>
          </copy>
        </assign>
      </sequence>
    </scope>
    <invoke name="callbackClient"
            partnerLink="client"
            portType="client:MapeamentoCallback"
            operation="onResult"
            inputVariable="outputVariable"/>
  </sequence>
</process>
```