UNIVERSIDADE FEDERAL DE PELOTAS
Bacharelado em Ciência da Computação

Trabalho Acadêmico

Usando *Web Services* para acesso a informação de contexto em um Portal *Web*

Roberto Silva Vargas

Pelotas, 2008

ROBERTO SILVA VARGAS

USANDO *WEB SERVICES* PARA ACESSO A INFORMAÇÃO DE CONTEXTO EM UM PORTAL *WEB*

Trabalho de conclusão de curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profª. Ana Marilza P. Fleischmann, MSc.
Co-orientador: Prof. João Ladislau B. Lopes, MSc.

Pelotas, 2008

Dados de catalogação na fonte:
Maria Beatriz Vaghetti Vieira – CRB-10/1032
Biblioteca de Ciência & Tecnologia - UFPel

V297u Vargas, Roberto Silva

Usando web services para acesso a informação de contexto em um Portal Web / Roberto Silva Vargas ; orientador Ana Marilza P. Fleischmann; co-orientador João Ladislau B. Lopes. – Pelotas, 2008. – 79f. - Monografia (Conclusão de curso). Curso de Bacharelado em Ciência da Computação. Departamento de Informática. Instituto de Física e Matemática. Universidade Federal de Pelotas. Pelotas, 2008.

1.Informática. 2.Computação pervasiva. 3. Computação sensível ao contextol. 4.Web services. I.Fleischmann, Ana Marilza P. II. Lopes, João Ladislau B. II.Título.

CDD: 004.65

Banca examinadora:

.....
Profª. MSc. Ana Marilza Pernas Fleischmann (DINFO/UFPel)

.....
Prof. Dr. Leomar Soares da Rosa Jr

.....
Prof. MSc. Marcello Macarthy

Dedico este trabalho aos meus pais, Isabel e Francisco, que nunca mediram esforços para que eu sempre tivesse tudo na vida e pudesse, através da dedicação, do exemplo e do amor que me foi dado, chegar até aqui.

Agradecimentos

Primeiramente gostaria de agradecer à minha família, em especial aos meus pais, Francisco e Isabel, por todo amor e compreensão que me dedicaram sempre. Vocês sempre foram a minha base para que eu alcançasse todos os meus objetivos e os meus ídolos os quais sempre quis seguir. As minhas irmãs Ângela, Francine e Márcia por estarem ao meu lado e torcendo por mim a todo instante. Aos meus cunhados Júnior e Artur, depois de alguns meses tenho um convite pra lhes fazer: WE amanhã lá em casa, ok? Ao vô Firmino e a vó Cida, ao vô Paulo e a vó Carmen, e a minha tia-avó Nair gostaria de mencionar a saudade enorme que sinto de vocês. Amo demais vocês todos.

Agradeço também a minha namorada Júlia que tanto amo. Você apareceu em minha vida no momento certo e, com seu companheirismo, carinho, amor, paciência e dedicação foi essencial para que eu alcançasse esta vitória.

Aos meus amigos que estão há muito tempo ao meu lado e dos quais eu adoro a companhia, me desculpem pela ausência nos últimos meses. Vocês sempre estiveram presentes nos melhores momentos da minha vida e quero brindar com vocês esta conquista.

Aos meus colegas, gostaria de dizer que a parceria, as madrugadas viradas estudando e fazendo trabalhos, as massas, os mates, enfim, a companhia dispensada durante estes cinco anos fez de vocês meus grandes amigos. E do fundo do meu coração, espero que agora o tempo não nos afaste para que eu possa continuar desfrutando da amizade de vocês para sempre.

A “turma do barulho”, sempre parceira para trocar os estudos por uma cerveja bem gelada. Saibam que vocês são pessoas ímpares as quais eu admiro e respeito muito.

Aos colegas de estágio e de trabalho, saibam que vocês me fizeram crescer muito durante este período.

Aos colegas do site Acontece, agradeço por abraçarem o projeto e por me ajudarem a mantê-lo quando não tive tempo.

Agradeço também aos meus mais que orientadores, professora Ana Marilza Pernas Fleischmann e professor João Ladislau Barbará Lopes, que estiveram sempre ao meu lado, dispostos a me ajudar no decorrer deste projeto.

A todas as pessoas que direta ou indiretamente contribuíram de alguma forma para que fosse possível a conclusão desta etapa.

A todos vocês, o meu sincero muito obrigado.

“Our deepest fear is not that we are inadequate.
Our deepest fear is that we are powerful beyond measure.
It is our light, not our darkness, that most frightens us.
Your playing small does not serve the world. There is nothing enlightened about shrinking so that other people won't feel insecure around you.
We are all meant to shine as children do.
It's not just in some of us; it is in everyone.
And as we let our own lights shine, we unconsciously give other people permission to do the same.
As we are liberated from our own fear, our presence automatically liberates others.”

Marianne Williamson

Resumo

VARGAS, Roberto Silva. **Usando *web services* para acesso a informação de contexto em um Portal *Web*.** 2008, 79f. Monografia – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas.

A computação pervasiva é um paradigma computacional bastante novo, apontado pela SBC (Sociedade Brasileira de Computação) como um dos cinco grandes desafios para a pesquisa na área de computação para os próximos 10 anos (LOPES, 2008) o qual permite que o usuário acesse um ambiente computacional em qualquer lugar a qualquer hora, por meio de um dispositivo computacional (COSTA, YAMIN, GEYER, 2008), até mesmo sem a sua própria intervenção. Entretanto, para que isto seja possível, é necessário transpor uma série de desafios, como por exemplo: heterogeneidade de equipamentos, mobilidade, escalabilidade e sensibilidade ao contexto. Este projeto é focado na sensibilidade ao contexto, e utiliza o projeto GRADEp (Ambiente de Grade Pervasiva) como estudo de caso. O projeto GRADEp consiste de um esforço no sentido de solucionar estes desafios, disposto na forma de um *middleware* adaptativo ao contexto, baseado em serviços. O GRADEp possui como objetivo criar e gerenciar um ambiente pervasivo, bem como promover a execução, sob este ambiente, das aplicações que expressam semântica siga-me. Estas aplicações são distribuídas, móveis e adaptativas ao contexto em que seu processamento ocorre, estando disponíveis de qualquer lugar, todo o tempo (LOPES, 2008). Desta forma, o presente projeto tem por objetivo facilitar a forma com que informações de contexto são disseminadas, provendo, para isto, um portal *web* para acesso a informações de contexto por meio de *web services*. O projeto desenvolvido provê um portal *web* para disseminação destas informações de contexto do *middleware* GRADEp. Estas informações são processadas pelo servidor de contexto baseado em ontologias denominado EXEHDA-ON, o qual faz parte do Subsistema de Adaptação e Reconhecimento de Contexto.

Palavras-chave: Computação pervasiva. Computação sensível ao contexto. W*eb services*.

Abstract

VARGAS, Roberto Silva. **Usando *web services* para acesso a informação de contexto em um Portal *Web*.** 2008, 79f. Monografia – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas.

The pervasive computing is a recent computational paradigm, pointed by SBC (Brazilian Computer Society) as one of the five major challenges for computing research for the next 10 years (LOPES, 2008), which allows the user to access a computer environment from anywhere and at anytime, by a computational device (COSTA, YAMIN, GEYER, 2008), even without his intervention. However, to make it possible, it is necessary to overcome some challenges like heterogeneity of equipments, mobility, scalability and context-awareness. This work focus on context-awareness and it uses the GRADEp project (Pervasive Grid Environment) as a case study. The GRADEp project consists of an effort to solve these challenges, in the form of a context-adaptive middleware, based on services. The GRADEp objectives are to create and to manage a pervasive environment, and to promote the implementation, in this environment, of the applications that express semantics follow-me. These applications are distributed, mobile and adaptive to the context in which their processing occurs, being available from anywhere, at any time (LOPES, 2008). Thus, this project aims to facilitate the way that context-information is disseminated, providing for it, a web site to access the context information through web services. These informations are processed by the context-server based on ontologies called EXEHDA-ON, which is part of the Adaptation and Context Recognition Subsystem.

Keywords: Pervasive computing. Context-awareness computing. Web services.

Lista de Figuras

Figura 1 - Visão geral sobre a computação pervasiva.....22
Figura 2 - Ambiente Pervasivo. 25
Figura 3 - Subsistemas do GRADEp. 27
Figura 4 - Um WS básico. 32
Figura 5 - Exemplo de documento XML. 35
Figura 6 - Documento XML utilizando namespaces. 36
Figura 7 - Visão parcial do esquema XML de um documento. 37
Figura 8 - Arquitetura de um WS. 38
Figura 9 - Estrutura de uma mensagem SOAP. 41
Figura 10 - Exemplo de utilização de WS. 43
Figura 11 – Árvore de conceitos da ontologia do ambiente pervasivo. 46
Figura 12 - Visão geral dos serviços do EXEHDA-ON. 48
Figura 13 - Visão geral do projeto que está sendo desenvolvido. 49
Figura 14 - Parte do WSDL referente operação de monitoramento. 53
Figura 15 - Parte do WSDL referente operação de consulta. 54
Figura 16 - Parte do WSDL referente operação de instanciação. 56
Figura 17 - Tela do Portal Web de apresentação do projeto.....57
Figura 18 - Tela do Portal Web que detalha o projeto GRADEp. 57
Figura 19 - Tela do portal web com outros links sobre Computação Pervasiva. 58
Figura 20 - Tela do Portal Web referente a escolha das operações. 59
Figura 21 - Tela do Portal Web referente a operação de consulta. 60
Figura 22 - Exemplo de tela de monitoramento com contagem regressiva. 61
Figura 23 - Exemplo de tela de instanciação de novos nodos. 62
Figura 24 - Primeiro teste sobre a operação de consulta. 64

Figura 25 - Segundo teste sobre a operação de consulta. 65
Figura 26 - Teste do monitoramento. 66
Figura 27 - Teste da instanciação e posterior resultado no monitoramento. 67

Lista de Tabelas

Tabela 1 Dados dos nodos e suas propriedades na ontologia modificada..... 63

Lista de abreviaturas e siglas

API	Application Programming Interface
CPU	Unidade Central de Processamento
CSS	Cascading Style Sheets
DUNS	Data Universal Numbering System
EXEHDA	EXecution Enviroment for Highly Distributed Applications
GB	Gigabyte
GRADEp	Ambiente de Grade Pervasiva
HTML	HyperText Markup Language
IDE	Integrated Development Environment
ISAM	Infra-estrutura de Suporte às Aplicações Móveis
MB	Megabyte
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Ontology Web Language
PDA	Personal Digital Assistant
PHP	Hypertext Preprocessor
RPC	Remote Procedure Call
SAJAX	Simple Asynchronous Javascript And XML
SAML	Security Assertion Markup Language
SDK	Software Development Kit
SOAP	Simple Object Acess Protocol
UCPel	Universidade Católica de Pelotas
UDDI	Universal Description Discovery and Integration
UFPel	Universidade Federal de Pelotas
UFRGS	Universidade Federal do Rio Grande do Sul
UFSM	Universidade Federal de Santa Maria

URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Services Description Language
WS-I	Web Services Interoperability Organization
XML	eXtensible Markup Language

Sumário

1 Introdução 17
1.1 Motivação 18
1.2 Objetivos 19
1.3 Contribuição Esperada 19
1.4 Estrutura do Projeto 20

2 Computação Pervasiva 21
2.1 Conceitos Iniciais 21
2.2 Desafios da Computação Pervasiva 23
2.3 Projeto GRADEp 24
2.3.1 Estrutura física 25
2.3.2 Estrutura de serviços 26
2.3.3 Sensibilidade de Contexto no GRADEp 28
2.4 Demais Projetos em Computação Pervasiva 29
2.4.1 Aura 29
2.4.2 Gaia 29
2.4.3 Oxygen 29
2.4.4 Endeavour 30

3 Web Services 31
3.1 Conceitos Iniciais 31
3.2 A linguagem XML 34
3.2.1 Namespaces 35
3.2.2 Esquemas XML 36
3.3 Arquitetura de um Web Service 37

3.4 Operações 38
3.4.1 WSDL – Web Services Description Language 39
3.4.2 UDDI – Universal Description Discovery and Integration 40
3.4.3 SOAP – Simple Object Access Protocol 40
3.5 Funcionamento dos Web Services 42

4 Web Service para consulta ao contexto 45
4.1 Subsistema de Reconhecimento de Contexto 45
4.1.1 EXEHDA-ON 45
4.2 Desenvolvimento do Web Service 50
4.2.1 Análise de requisitos 50
4.2.2 Infra-estrutura de Desenvolvimento 50
4.2.2.1 Infra-estrutura de implementação do Web Service Servidor 50
4.2.2.2 Infra-estrutura de implementação do Web Service Cliente 51
4.2.3 Implementação dos Serviços 52
4.2.3.1 Servidor 53
4.2.3.2 Cliente e Portal Web 56

5 Análise dos Resultados 63
5.1 Consulta 64
5.1.1 Primeiro teste 64
5.1.2 Segundo teste 65
5.2 Monitoramento 65
5.3 Instanciação 66

6 Conclusão e Trabalhos Futuros 68

Referências 70

Apêndices 73

1 Introdução

Com o crescimento da utilização de dispositivos móveis e com o avanço das tecnologias disponíveis para estes dispositivos, um novo paradigma torna-se presente para integrar a onipresença destes na vida cotidiana das pessoas: a Computação Pervasiva.

O termo foi usado pela primeira vez por Mark Weiser, cientista chefe do Centro de Pesquisa Xérox PARC, em seu artigo “O computador do Século 21”. Weiser publicou este artigo em 1991 e, já nesta época, previa um aumento nas funcionalidades e na disponibilidade de serviços de computação para os usuários finais, apresentando estes serviços a menor visibilidade possível aos usuários. Para o autor, a computação não seria exclusividade de um computador e, sim, de diversos dispositivos conectados entre si (WEISER, 1991). Para permitir a transparência destes serviços, faz-se necessário transpor uma série de desafios. Entre os desafios, destaca-se neste trabalho a Sensibilidade ao Contexto. É por meio de sensibilidade ao contexto que uma aplicação modifica o seu estado de funcionamento, sendo o seu estado adaptado de acordo com o ambiente. Para esta modificação, dados são obtidos por meio de sensores, por exemplo, e processados por serviços do ambiente pervasivo (NUNES, 2008).

Neste contexto, o *middleware* GRADEp é um projeto que provê uma solução para suporte a execução de aplicações pervasivas e contém, entre outros componentes, um servidor de contexto baseado em ontologias, que armazena informações atuais a respeito do ambiente pervasivo e de seu funcionamento. Portanto, neste projeto foi desenvolvido um protótipo de um portal *web* que irá agregar ao projeto GRADEp funcionalidades de monitoramento, consulta e instanciação de informações de contexto, acessando diretamente o Subsistema de

Adaptação e Reconhecimento de Contexto. Para realizar a comunicação entre o portal *web* e o servidor de contexto foi utilizada a tecnologia de *Web Services* (WS), uma vez que estes são utilizados para integração de sistemas e comunicação entre aplicações heterogêneas. Ou seja, com eles é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

1.1 Motivação

A Computação Pervasiva é uma área recente de pesquisa, considerada o novo paradigma do século 21. Através dela possibilita-se um ambiente computacional onde o usuário obtém as informações as quais desejar, no momento em que desejar e onde desejar (GEYER, 2004).

Neste contexto, está inserido o projeto GRADEp, o qual objetiva, na forma de um *middleware*, criar e gerenciar um ambiente de grade pervasiva (YAMIN, 2004). Neste ambiente estão incluídos os seguintes serviços: acesso pervasivo aos recursos e serviços; adaptação; reconhecimento do contexto; execução distribuída; comunicação; e a descoberta e gerenciamento de recursos/serviços (GEYER, 2004).

Atualmente, as pesquisas relacionadas ao *middleware* GRADEp têm se direcionado a provê-lo de maior expressividade para a descrição do funcionamento dos dispositivos, o que se tornou possível através do uso de ontologias. A utilização de ontologia para descrição semântica de um determinado vocabulário proporciona um entendimento amplo das características e propriedades das classes pertencentes a um domínio, assim como seus relacionamentos (PERNAS, 2004).

No que diz respeito ao reconhecimento de contexto, está associado ao Subsistema de Adaptação e Reconhecimento de Contexto, o qual inclui o servidor de contexto baseado em ontologias, denominado EXEHDA-ON, no qual as informações são monitoradas constantemente, de forma a permitir que tanto a aplicação como ele próprio utilizem estas informações para gerência da adaptação.

Desta forma, neste trabalho foi desenvolvido um protótipo de um portal *web*, utilizando-se da linguagem PHP 5 (*Hypertext Preprocessor*), para monitoramento, consulta e instanciação das informações referentes aos dispositivos registrados no EXEHDA-ON. Por se estar utilizando um WS, as informações de contexto, registradas através de uma ontologia, podem ser manipuladas por meio de um

cliente. O WS desenvolvido reflete o estado atual do contexto celular, uma vez que o processo cliente se comunica diretamente com o processo servidor presente na estação base da célula. No lado cliente, foi desenvolvido um WS que irá acionar os métodos instalados no servidor. Estes métodos realizam o processamento das informações mantidas pelo modelo ontológico existente no servidor de contexto.

O uso de WS deve-se ao fato destes possibilitarem a integração de sistemas. Eles permitem as aplicações se comunicarem umas com as outras, combinando funcionalidades de forma independente de plataforma ou linguagem. Os WSs são construídos com a utilização de padrões da Internet e são independentes de plataforma, assim esta característica os confere um potencial de interoperabilidade que nenhuma outra tecnologia até hoje proposta conseguiu atingir (MARQUEZAN, CARISSIMI, NAVAUX, 2006).

1.2 Objetivos

O objetivo central deste trabalho é o desenvolvimento de um WS para disseminação de informações de contexto aos consumidores, independente de plataforma.

Os dados manipulados pelo serviço desenvolvido são fornecidos aos consumidores através de um portal *web*, desenvolvido em PHP 5, para monitoramento, consulta e instanciação das informações referentes ao contexto celular.

1.3 Contribuição Esperada

No presente trabalho, pretende-se disseminar as informações de contexto do *middleware* GRADEp para seus consumidores, através da construção de um WS que irá viabilizar a disponibilização das informações em um portal *web*. Além disso, através do portal, será possível executar operações sobre o servidor de contexto EXEHDA-ON.

Ainda, deve ser notado que o trabalho contribuirá potencialmente ao projeto GRADEp, adicionando novos recursos ao *middleware* do projeto, e contribuindo assim para a comunidade usuária.

1.4 Estrutura do Projeto

Este projeto está subdividido em seis seções, sendo o primeiro referente a esta introdução, a qual apresenta a motivação, os objetivos, contribuição esperada e a estrutura do trabalho.

Na seção dois, é introduzido o conceito de computação pervasiva, mostrando seus conceitos bem como o que a sua utilização representa. São discutidos também os desafios da área que se necessitam ser transpostos para a implantação de ambientes pervasivos.

A seção três apresenta a tecnologia de *Web Services* a qual será utilizada neste trabalho para realização da comunicação entre o portal *web* e o Servidor de Contexto do *middleware* GRADEp (EXEHDA-ON). São apresentados alguns conceitos iniciais, a linguagem XML e também a arquitetura e as operações básicas de um WS ao longo desta seção.

A partir da seção quatro, é iniciada a abordagem prática deste projeto. Nela são descritos todos os passos efetuados para construção do WS, tanto do lado cliente quanto do lado servidor, os quais foram construídos seguindo os padrões recomendados pela W3C (*World Wide Web Consortium*) e pela OASIS (Organização para o Avanço de Padrões em Informação Estruturada). Estas instituições são responsáveis pela padronização dos WS, possibilitando assim a independência de plataforma e de linguagem de programação a qual se pretende atingir.

A seção cinco diz respeito ao portal w*eb,* o qual se comunica com o *Web Service* para interfacear as operações de monitoramento, consulta e instanciação sobre o EXEHDA-ON. Desta forma, todos os detalhes referentes à construção deste portal serão documentados nesta seção e, também, através dele é efetuada a análise de resultados das operações, comprovando a sua funcionalidade.

Na seção seis o trabalho é finalizado, sendo apresentadas as principais conclusões e sugestões de trabalhos futuros.

2 Computação Pervasiva

Nesta seção será feita uma breve apresentação sobre o paradigma escopo deste trabalho: a computação pervasiva. Serão discutidos os principais desafios da área e também serão apresentados alguns projetos já existentes, em especial o projeto GRADEp, o qual é focado nesta monografia.

2.1 Conceitos Iniciais

Em 1991, Mark Weiser descreveu no clássico artigo "*The computer for the 21st century*" (O computador do século 21) sua visão do que seria a computação no século 21, prevendo a ubiqüidade dos computadores pessoais. Ele defendia a teoria de que as tecnologias mais profundas são aquelas que desaparecem, misturadas com os objetos do dia-a-dia (WEISER, 1991).

Atualmente, o computador ainda é visto como uma ferramenta que executa programas em um mundo virtual, no qual os usuários entram para executar uma tarefa, e saem dele quando terminam (SAHA; MURKHERJEE, 2003). Na computação ubíqua descrita por Weiser, ocorre o contrário: a ferramenta é quem se integra ao mundo do usuário, fazendo com que o mesmo não precise intervir para operá-la. As pessoas simplesmente fazem uso da computação de forma transparente.

Com o constante avanço das tecnologias e a popularização de dispositivos móveis, como os *notebooks*, PDA*s* (*Personal Digital Assistants*), telefones celulares e câmeras fotográficas digitais, a computação esta cada dia mais presente no cotidiano da população. A computação ubíqua beneficia-se disto visto que ela é a integração desta onipresença dos dispositivos com suas aplicações, para o benefício

dos usuários.

A computação ubíqua é um termo definido por Weiser que dizia que, em um futuro próximo, iria ocorrer a proliferação dos dispositivos computacionais, de forma com que eles estariam onipresentes na vida da população. Logo, um dos termos que se faz necessário estudar para entender o funcionamento desta ubiqüidade é a computação móvel.

O conceito de computação móvel implica que os meios de computação e os serviços associados a eles são móveis, ou seja, um computador possui características que o torna possível de ser carregado de forma prática pelo usuário. Dessa forma, o computador torna-se um dispositivo sempre presente, expandindo a flexibilidade do usuário em utilizar os seus serviços, independente de localização. Assim, a computação móvel permite manter uma conectividade “a qualquer hora, em qualquer lugar”.

Já na computação pervasiva, este conceito se estende para “o tempo todo, em qualquer lugar” pela integração de tecnologias que dão suporte a esta onipresença dos dispositivos computacionais (SAHA; MURKHERJEE, 2003). A Fig. 1 ilustra esta visão:

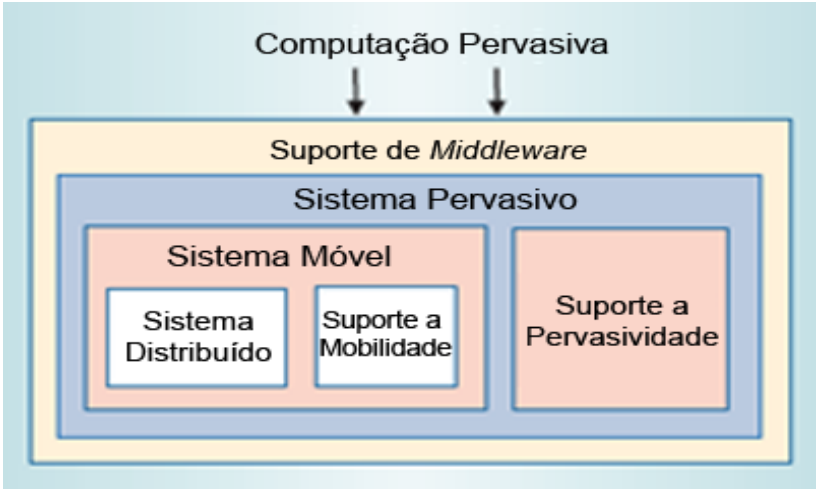

Figura 1 - Visão geral sobre a computação pervasiva.
FONTE: Traduzido de SAHA; MURKHERJEE, 2003.

A premissa central na computação pervasiva consiste em permitir ao usuário o acesso ao seu ambiente de trabalho a partir de qualquer lugar, o tempo todo, usando vários tipos de dispositivos (móveis ou não), contemplando funcionalidades que prevêem uma mobilidade física (equipamentos e/ou usuários) e de *software*.

Nesta proposta, a aplicação ou o ambiente de execução pró-ativamente

monitoram e controlam as condições do contexto e a aplicação reage às alterações no contexto através de um processo de adaptação (YAMIN, 2004).

Ainda em (YAMIN, 2004), o autor afirma que a computação pervasiva pode ser construída pela integração da computação móvel, computação em grade e da computação sensível de contexto. Ser sensível de contexto significa estar consciente das modificações que ocorrem no ambiente para posteriormente adaptar-se a elas, sem exigir a atenção do usuário. Ou seja, além de se ter o conhecimento sobre as informações em determinados momentos, as aplicações sensíveis de contexto possuem a habilidade de interpretar e usar o contexto como base para um comportamento adaptativo. Para isto, as informações de contexto dos dispositivos em execução são monitoradas constantemente através de sensores, sendo repassadas a um servidor de contexto.

Atualmente, as pesquisas relacionadas aos servidores de contexto têm se direcionado a prover este de maior expressividade para descrição do funcionamento dos dispositivos, o que se tornou possível através do uso de ontologias.

Ontologias vêm sendo utilizadas por várias áreas da Ciência da Computação, principalmente com o intuito de dotar os sistemas de meta-conhecimento. A utilização de ontologias para descrição semântica de um determinado vocabulário proporciona um entendimento amplo das características e propriedades das classes pertencentes a um domínio, assim como seus relacionamentos (LOPES, 2008).

2.2 Desafios da Computação Pervasiva

Para que seja possível o paradigma de um ambiente computacional pervasivo, é necessário transpor uma série de desafios (COSTA et al, 2008):

- Heterogeneidade de equipamentos: com a gama de dispositivos existentes, a heterogeneidade de *hardware* aparece. Além disso, os sistemas operacionais e as interfaces com o usuário também podem ser variadas. Logo, programas devem esconder do usuário essas diferenças na infra-estrutura e gerenciar as conversões necessárias de um ambiente para o outro.
- Mobilidade: depois da popularização dos PDAs, *notebooks*, celulares e outros dispositivos portáteis, passou a ser possível ao usuário acessar seus aplicativos e informações independente de onde estiverem. Cabe salientar também, que a

mobilidade lógica também deve ser suportada, ou seja, aplicativos devem se mover de um dispositivo para o outro, e o acesso aos dados deve ser mantido.

- Escalabilidade: o aumento da capacidade dos dispositivos, a intensidade e a qualidade das interações homem máquina tendem a aumentar (SAHA; MURKHERJEE, 2003). Um sistema escalável é aquele que continua efetivo quando existe um aumento significativo no número de recursos e no número de usuários.
- Sensibilidade de Contexto: refere-se a habilidade dos sistemas computacionais de obterem vantagem das informações ou condições existentes em um ambiente dinâmico para adicionar valor aos serviços ou executar tarefas mais complexas, ou seja, permite que os aplicativos detectem mudanças para posterior reação dos programas.
- Invisibilidade: a idéia de invisibilidade significa fazer com que as pessoas não precisem intervir no sistema (ou que precisem intervir minimamente). Desta forma, o que acontece atualmente, quando o usuário entra no mundo virtual do computador, não irá ocorrer mais. Pelo contrário, o computador que irá participar do mundo real do usuário.

Para solucionar estes desafios, diversos projetos propõem do desenvolvimento de *middlewares*. Como o alvo deste trabalho está no aumento das funcionalidades de um *middleware* especifico para computação pervasiva, o *middleware* GRADEp é o que será tratado a seguir.

2.3 Projeto GRADEp

O projeto GRADEp, representa uma solução para os desafios citados na seção anterior, disposta na forma de um *middleware* adaptativo ao contexto, baseado em serviços. Ele realiza a comunicação entre os diferentes dispositivos e o usuário final das aplicações, promovendo a execução, sob este ambiente, das aplicações que expressam a semântica siga-me. Esta semântica é quem permite o uso da computação a qualquer lugar, em qualquer dispositivo computacional, em qualquer momento.

O acrônimo GRADEp refere-se a grade pervasiva. O projeto foi criado com a intenção de desenvolver um *middleware* para gerenciar um ambiente de grade pervasiva. É oriundo de um outro projeto chamado ISAM (Infra-estrutura de Suporte às Aplicações Móveis), o qual começou em 2001, liderado pela Universidade Federal

do Rio Grande do Sul (UFRGS), com o objetivo de definir uma arquitetura de *software* para computação pervasiva (YAMIN et al, 2004).

O *middleware* foi inicialmente proposto sob o nome de EXEHDA (*EXecution Enviroment for Highly Distributed Applications*), mais tarde sendo distribuído com o nome de GRADEp. Atualmente, o projeto conta também com a colaboração de outras instituições como a Universidade Católica de Pelotas (UCPel), Universidade Federal de Santa Maria (UFSM) e, mais recentemente, da Universidade Federal de Pelotas (UFPel).

2.3.1 Estrutura física

A estrutura física gerenciada pelo GRADEp é mapeada em uma organização composta pela agregação de células de execução, conforme pode ser visto na Fig. 2:

Figura 2 - Ambiente Pervasivo.
FONTE: YAMIN, 2004.

Os recursos da infra-estrutura física são classificados em três abstrações básicas, as quais são utilizadas na composição do Ambiente Pervasivo (YAMIN, 2004):

- Celúla: denota a área de atuação de um nodo base, sendo composta por este

e pelos demais nodos. Os principais aspectos considerados na definição da abrangência de uma célula são: o escopo institucional, a proximidade geográfica e o custo de comunicação;

- Nodo base: é o ponto de contato para os nodos. É responsável por todos os serviços básicos do Ambiente Pervasivo e, embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários equipamentos;
- Nodo: são os equipamentos de processamento disponíveis no Ambiente Pervasivo, sendo responsáveis pela execução das aplicações. Um subcaso deste tipo de recurso é o nodo móvel, o qual possui elevada portabilidade, tipicamente dotado de interface de rede para operação sem fio, e neste caso, integra a célula a qual seu ponto-de-acesso está subordinado. Os nodos móveis são funcionalmente análogos aos nodos convencionais, porém eventualmente com uma capacidade mais restrita (por exemplo, PDAs).

O GRADEp não possui mecanismos de gerência específicos para recursos especializados, como impressoras, scanners, etc., por este motivo, ele permite a catalogação de tais recursos como integrantes de uma determinada célula do Ambiente Pervasivo, tornando-os, desta forma, passíveis de serem localizados dinamicamente, e, deste modo, podendo ser utilizados pelas aplicações pervasivas (YAMIN, 2004).

2.3.2 Estrutura de serviços

No ambiente proposto no GRADEp, estão incluídos os seguintes serviços: acesso pervasivo aos recursos e serviços; adaptação; reconhecimento do contexto; execução distribuída; comunicação; e a descoberta e gerenciamento de recursos/serviços organizados em subsistemas conforme ilustra a Fig. 3 (YAMIN, 2004):

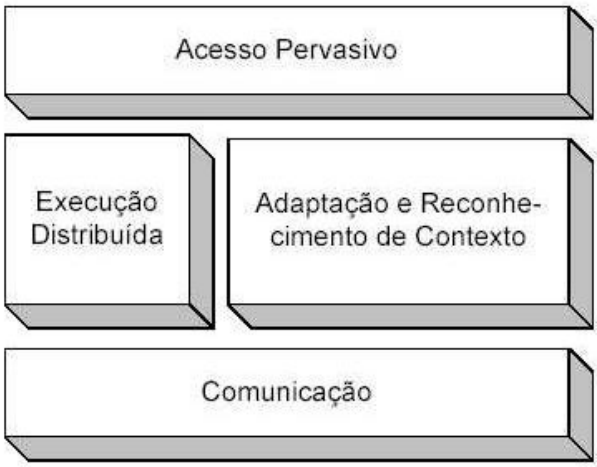

Figura 3 - Subsistemas do GRADEp.
FONTE: YAMIN, 2004.

- Execução Distribuída: O Subsistema de Execução Distribuída é responsável pelo suporte ao processamento distribuído no GRADEp. No intuito de promover uma execução efetivamente pervasiva, este subsistema interage com outros subsistemas do GRADEp. Em especifico, interage com o subsistema de reconhecimento de contexto e adaptação, de forma a prover comportamento distribuído e adaptativo as aplicações.
- Adaptação e Reconhecimento de Contexto: O suporte a adaptação no GRADEp está associado a operação do subsistema de reconhecimento de contexto e adaptação. Este subsistema inclui serviços que tratam desde a extração da "informação bruta" sobre as características dinâmicas e estáticas dos recursos que compõe o Ambiente Pervasivo, passando pela identificação em alto nível dos elementos de contexto, até o disparo das ações de adaptação em reação a modificação no estado de tais elementos de contexto.
- Comunicação: A natureza da mobilidade do *hardware* e, na maioria das vezes, também a do *software*, não garante a interação continua entre os componentes da aplicação distribuída. As desconexões são comuns, não somente devido à existência de alguns *links* sem fio, mas, sobretudo como uma estratégia para economia de energia nos dispositivos móveis. O subsistema de comunicação do GRADEp disponibiliza mecanismos que atendem estes aspectos da computação pervasiva.
- Acesso Pervasivo: O ambiente de computação pervasiva tem por premissas (i) a possibilidade de o usuário disparar aplicações a partir de qualquer nodo

integrante do sistema e, após o disparo, (ii) a mobilidade parcial ou integral de tais aplicações em resposta a modificações em seu contexto de execução, como, por exemplo, a movimentação do usuário ou alteração na condição de carga dos dispositivos atualmente em uso pela aplicação. Com isso, o subsistema de acesso pervasivo, suporta a premissa de acesso em qualquer lugar, o tempo todo, a dados e códigos da computação pervasiva.

2.3.3 Sensibilidade de Contexto no GRADEp

A computação sensível de contexto é um paradigma computacional que se propõe a permitir que as aplicações tenham acesso e tirem proveito de informações que digam respeito às computações que realizam, buscando otimizar seu processamento. Está relacionada com a habilidade dos sistemas computacionais obterem vantagem das informações ou condições existentes em um ambiente dinâmico para adicionar valor aos serviços ou executar tarefas mais complexas.

Em (YAMIN et al., 2003), o contexto é definido como toda a informação relevante para a aplicação que possa ser obtida da infra-estrutura computacional, cuja alteração em seu estado dispara um processo de adaptação na aplicação. Nessa visão, o contexto permite enfocar os aspectos relevantes para uma situação particular e ignorar outros. A aplicação explicitamente identifica e define as entidades que caracterizam uma situação e essas passam a integrar o seu contexto.

Uma das mais citadas definições é a encontrada em (DEY; ABOWD, 2000). Segundo os autores, entende-se por contexto qualquer informação que possa ser usada para caracterizar a situação de uma entidade, entendendo-se por entidade uma pessoa, um lugar ou um objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo os próprios usuário e a aplicação

No GRADEp, as informações de contexto são pró-ativamente monitoradas. Ou seja, o *middleware* está constantemente monitorando as informações através de sensores. Estas informações são processadas pelo servidor de contexto baseado em ontologias, de forma a permitir que tanto a aplicação como ele próprio utilize estas informações para gerência da adaptação

O serviço de contexto (Subsistema de Reconhecimento de Contexto) será definido em maiores detalhes na seção 4 deste trabalho, onde é descrito o serviço desenvolvido.

2.4 Demais Projetos em Computação Pervasiva

Além do *middleware* GRADEp, tratado de forma mais especifica neste trabalho, existem muitas outras soluções, também na forma de *middlewares* ou não, para gerenciamento de um ambiente pervasivo e também com suporte a programação de aplicações neste ambiente. Alguns deles serão apresentados a seguir:

2.4.1 Aura

Proposto na Universidade de Carnegie Mellon considera que o mais valioso recurso de um sistema computacional é o usuário e com isso, foca-se nele, nas suas tarefas e preferências. Possui como objetivo principal, prover a cada usuário uma "aura de informação pessoal", invisível de computação com serviços que persistem independentes de localização (AURA, 2008).

2.4.2 Gaia

No projeto Gaia (GAIA, 2008) é defendida uma visão de futuro onde o espaço habitado pelas pessoas é interativo e programável, sendo chamado de "espaços ativos" (*Active Spaces*). Os usuários interagem com seus escritórios, casa, carros, etc., para requisitar informações, beneficiar-se dos recursos disponíveis e configurar o comportamento de seu habitat. Dados e tarefas estão sempre acessíveis e são mapeados dinamicamente para os recursos convenientes e presentes na localização corrente.

2.4.3 Oxygen

O projeto Oxygen é desenvolvido pelo Laboratório de Ciência da Computação e Inteligência Artificial do MIT (Massachusetts Institute of Technology), e tenta atingir os objetivos da computação pervasiva, através da combinação de tecnologias de "usuário" e tecnologias de "sistema". As tecnologias de "usuário" atendem diretamente as necessidades humanas. Já as tecnologias de "sistema" referem-se ao suporte as mudanças que ocorrem no ambiente, ou seja, a captação

das informações presentes no ambiente e o disparo das operações referentes a elas (OXYGEN, 2008).

2.4.4 Endeavour

O projeto Endeavour[1], do departamento de ciência da computação e engenharia eletrônica da Universidade de Berkeley, foi iniciado em 1999, e leva o nome do navio comandado pelo Capitão Hook em suas expedições pelo Oceano Pacífico. No projeto Endeavour usa-se a metáfora do mar e da maneira como ele "interconecta" o mundo (KATZ, 1999).

Sob o ponto de vista do projeto, os sistemas de informação são "fluídos", que existem todo o tempo e estão em todo lugar, são componentes que "fluem" através das infra-estruturas, transformando-se para adaptar-se ao seu uso e para cooperação em tarefas. O objetivo é criar um "utilitário" para informação pervasiva, baseado em uma nova tecnologia de sistemas "fluídicos", trazendo novas alternativas para a resolução de problemas e aprendizado.

Percebe-se então que a computação pervasiva vem sendo cada vez mais fonte de pesquisa nas universidades e, inevitavelmente, em breve estará a disposição de todos fornecendo facilidade de computação e transformando a área da informática.

[1] http://endeavour.cs.berkeley.edu/

3 *Web Services*

O tema abordado ao longo desta seção é *Web Service* (WS). WS será utilizado neste trabalho como mecanismo para acessar o servidor de contexto do *middleware* GRADEp e repassar as informações para o portal *web* (como tratado na seção de introdução desta monografia).

3.1 Conceitos Iniciais

Web Service (WS) é um paradigma de computação distribuída baseado em padrões da Internet (BONA, 2004). Pode-se dizer então que WS é uma tecnologia que se constitui de outras tecnologias e padrões que estão sendo desenvolvidos há pouco tempo. São considerados um novo tipo de aplicação para a Internet e, na prática, permitem uma comunicação fácil entre diferentes aplicações, de maneira rápida e eficiente, utilizando a estrutura da *web* já existente.

Estas aplicações usam mensagens em formato XML (*eXtensible Markup Language*) padronizadas, podendo ser acessadas remotamente, independentes do sistema operacional ou linguagem de programação adotados nos computadores que estão se comunicando, conforme ilustrado na Fig. 4.

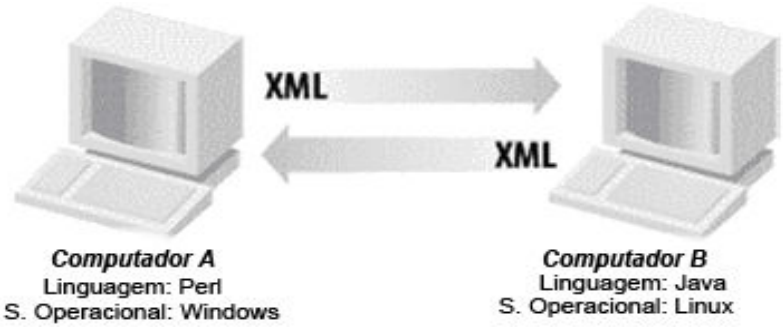

Figura 4 - Um WS básico.
FONTE: Traduzido de CERAMI, 2002.

Um WS também deve possuir duas outras propriedades. Segundo Cerami (2002), um WS deve:

- ser auto-descritivo: quando um novo WS é desenvolvido, deve-se publicar também uma interface pública para o serviço. No mínimo, ele deve conter uma documentação para que outros desenvolvedores possam mais facilmente integra-lo.
- ser possível de ser encontrado: quando um novo WS é criado, deve haver um mecanismo relativamente simples para publicá-lo. Da mesma forma deve haver algum mecanismo através do qual os consumidores interessados possam encontrar o serviço e localizar a sua interface pública.

Portanto, WS é uma interface que descreve uma coleção de operações que são acessíveis pela rede através de mensagens em formato XML padronizadas (KREGER, 2001), a qual deve ser auto-descritiva e possível de ser encontrada.

WSs combinam as características de execução das aplicações programáticas com a abstração da internet (NEWCOMER, 2002), portanto, permitem um fraco acoplamento entre sistemas. Desta maneira, existe um baixo grau de interdependência entre as partes que se comunicam e as modificações internas em cada parte não são percebidas externamente. Sendo assim, é possível a comunicação entre diferentes plataformas e ambientes, desde que estas aplicações adotem o mesmo conceito representado pelos WSs.

WSs representam uma importante evolução para a comunicação entre os sistemas, pois a comunicação se baseia em padrões amplamente aceitos e utilizados em um âmbito global, que são os padrões utilizados pela internet.

Apesar de não ser a primeira tecnologia a permitir este tipo de serviço, ela se diferencia por usar XML na comunicação, escondendo a implementação em ambas as pontas (cliente/servidor), o que a torna independente de plataforma e de linguagem. É a integração ideal para um ambiente tão diversificado quanto à

internet, na qual se conectam computadores pessoais, sejam eles *desktops*, *notebooks* ou PDAs, além de servidores que possuem um número enorme de diferentes arquiteturas e plataformas. Em suma, essa tecnologia realmente habilita uma utilização da internet em nível global (BREITMAN, 2005).

Outra importante característica dos WSs é que, juntamente com os mesmos, existem informações sobre o formato das mensagens que serão trocadas entre as aplicações; a forma como estas mensagens serão transportadas, a localização do serviço, entre outras informações. Todas estas informações são baseadas na linguagem XML, permitindo desta forma uma descrição completa do serviço e possibilitando que outras aplicações possam usar este serviço de maneira adequada (SILVA, 2004).

Existem atualmente algumas organizações envolvidas na criação e na especificação de padrões para os WSs. O W3C (*World Wide Web Consortium*) é responsável por uma série de padrões diretamente ligados aos WSs, como por exemplo: a linguagem XML; o protocolo SOAP (*Simple Object Acess Protocol*) que define o formato das mensagens utilizadas; a linguagem de descrição WSDL (*Web Services Description Language*); entre outras. Em linhas gerais, pode-se dizer que as tecnologias básicas ligadas aos WSs são tratadas pelo W3C.

Outra importante organização é a OASIS (*Organization for the Advancement of Structured Information Standards*). Entre os vários padrões desenvolvidos por esta organização pode-se citar: UDDI (*Universal Description Discovery and Integration*) que consiste em uma série de métodos para a descoberta e chamada de forma dinâmica de WS; *WS-Security* (*Web Services Security*) que contém especificações cujo objetivo é garantir a integridade e a confidencialidade na troca de mensagens SOAP entre WS; a linguagem SAML (*Security Assertion Markup Language*) cujo objetivo é criar e trocar informações sobre segurança, como uma autenticação bem sucedida, por exemplo, entre WS. Comparando com o W3C, o trabalho da OASIS está ligado a questões de mais alto nível nos WSs.

Uma outra organização importante no contexto dos *Web Services* é a WS-I (*Web Services Interoperability Organization*). Esta organização tem por foco promover a interoperabilidade entre os WS nas diversas plataformas, sistemas operacionais e linguagens de programação.

3.2 A linguagem XML

XML (*eXtensible Markup Language*) é uma linguagem de marcação, assim como a HTML (*HyperText Markup Language*), recomendada pela W3C para a representação de documentos de forma estruturada (HAROLD, MEANS, 2002), independente da plataforma onde é utilizada. Descreve uma classe de objetos de dados chamados documentos XML e parcialmente descreve o comportamento dos programas de computador que os processa (BRAY, PAOLI, SPERBERG-MCQUEEN,1998). Uma grande vantagem da linguagem XML, além de oferecer interoperabilidade, tanto de plataforma como de linguagem, é que ela não se limita a um certo número de *tags* (marcações)*,* ou seja, você pode criar suas próprias *tags*.

Os documentos XML são escritos conforme as regras desta linguagem. Entre essas regras pode-se citar, entre outras: os documentos devem ser simples de ser utilizados através da internet; podendo ser processados por computadores e legíveis a seres humanos.

Um documento XML é composto de marcações (*tags*) e dentro destas *tags* é possível a existência de dados e atributos. Os atributos são usados para fornecer uma informação adicional sobre os elementos, e como já dito anteriormente, o conjunto de *tags* é criado pelos usuários conforme suas necessidades e sua semântica advêm de um acordo entre as partes que utilizarão estes dados, visto que a especificação da linguagem não define necessariamente a sintaxe e a semântica a serem utilizadas.

Devido a estas características, a linguagem XML é um padrão para troca de dados através da internet. A Fig. 5 mostra um exemplo de documento XML válido, ou seja, aquele que está de acordo com as regras da linguagem XML.

```
<?xml version=”1.0”?>
<exposicao>
        <quadro id=”001”>
                <nome>Mona Lisa</nome>
                <pintor>Leonardo Da Vinci</pintor>
        </quadro>
        <quadro id=”002”>
                <nome>Guernica</nome>
                <pintor>Pablo Picasso</pintor>
        </quadro>
</exposicao>
```

Figura 5 - Exemplo de documento XML.

Como se pode ver na Fig. 5, em um documento XML cada *tag* inicial, por exemplo, <exposicao>, precisa de uma *tag* final correspondente: </exposicao>. Na linguagem XML um par constituído pela *tag* inicial e final, pode ser chamado de elemento. O elemento pode conter outros elementos filhos e desta forma o documento XML é estruturado na forma de uma árvore. E, como já foi dito, um elemento também pode ter um atributo. Por exemplo, o elemento <quadro> possui um atributo denominado *id*, o qual representa um número de identificação da obra.

3.2.1 *Namespaces*

Namespaces é a solução adotada para distinguir *tags* iguais. São necessários, pois em XML o conjunto de *tags* pode ser criado livremente, e desta maneira pode ser necessária a criação de *tags* iguais para representar diferentes dados. Portanto, eles permitem que cada elemento e atributo em um documento sejam colocados em *namespaces* diferentes (HAROLD, 1999).

Na Fig. 5, a *tag* <exposicao> é utilizada para representar uma exposição de quadros. Porém, esta mesma *tag* poderia ser utilizada para representar, por exemplo, uma exposição de carros antigos.

Para isto, *namespace* é utilizado como um prefixo colocado em cada *tag* para diferenciá-la de outras *tags* que possam haver em um mesmo documento. Este prefixo deve ser único, visto que prefixos iguais levariam ao mesmo problema inicial. A idéia é possuir um nome único que posteriormente será substituído pelo programa que processará o documento XML.

Para acrescentar um *namespace* a uma *tag* é necessário utilizar o atributo “xmlns” seguido do prefixo que o identifica e sua respectiva URI (*Uniform Resource Identifier*). Um *namespace* URI nada mais é do que um identificador formal que ajuda a distinguir os elementos com o mesmo nome, mas com diferentes significados. O exemplo a seguir mostra o código da Fig. 5, utilizando o conceito de *namespaces*.

```
<?xml version=”1.0”?>
<pinturas:exposicao xmlns:pinturas=”www.servidor.com.br/exposicao1”>
        <quadro id=”001”>
                <nome>Mona Lisa</nome>
                <pintor>Leonardo Da Vinci</pintor>
        </quadro>
        <quadro id=”002”>
                <nome>Guernica</nome>
                <pintor>Pablo Picasso</pintor>
        </quadro>
</pinturas:exposicao>
<veiculos:exposicao xmlns:veiculos=”www.servidor.com.br/exposicao2”>
        <carro id=”001”>
                <nome>Fusca</nome>
        </carro>
        <carro id=”002”>
                <nome>Opala</nome>
        </carro>
</veiculos:exposicao>
```

Figura 6 - Documento XML utilizando namespaces.

3.2.2 Esquemas XML

Esquemas XML (*XML Schema*) são utilizados para descrever a estrutura de um documento XML visto que eles podem conter muitos elementos, inclusive elementos aninhados. Portanto, é necessário um meio para validar um documento XML, fazendo com que, além de seguir as regras da linguagem, o documento esteja estruturado conforme acordado entre as partes. Assim, um Esquema XML é a constituição de regras ou de modelos de estrutura, que se aplicam a uma classe de

documentos XML (VLIST, 2002) para garantir que eles estejam na forma correta, ou seja, servem para descrever a estrutura que deve ser seguida pelos documentos escritos segundo este esquema. Este mecanismo garante então que duas partes distintas possam trocar um documento XML e cada uma delas possa verificar se o mesmo é válido.

O elemento <exposicao> mostrado anteriormente pode ser definido utilizando-se o esquema mostrado na Fig. 7.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://w3.org/2001/XMLSchema"
        targetNameSpace="http://www.dominio.com"
        elementFormDefault="qualified">
<xs:element name="exposicao" minOccurs="1" type="pinturas:exposicaoTipo">
    <xs:complexType name="exposicaoTipo">
        <xs:sequence>
            <xs:element name="quadro" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Figura 7 - Visão parcial do esquema XML de um documento.

3.3 Arquitetura de um *Web Service*

Em um modelo típico de funcionamento de um WS, ilustrado na Fig. 8, podem-se identificar três tipos básicos de papéis realizados por uma das entidades participantes de sua arquitetura (MARQUEZAN; CARISSIMI; NAVAUX, 2006):

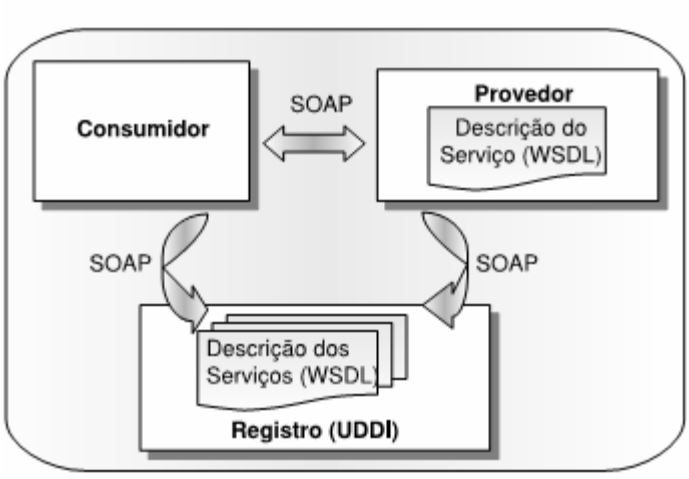

Figura 8 - Arquitetura de um WS.
FONTE: MARQUEZAN; CARISSIMI; NAVAUX, 2006.

- Provedor de Serviço (*Service Provider*): É a entidade que cria o WS e que disponibiliza um serviço remotamente. Para que o Provedor de Serviço atinja o máximo do seu potencial, ele precisa descrever as suas operações em algum formato padrão e publicar a existência do serviço em algum lugar.
- Consumidor de Serviço (*Service Consumer*): É a entidade cliente que utiliza os serviços de um Provedor de Serviço. O Consumidor de Serviço pode saber quais as funcionalidades disponíveis por meio da descrição fornecida pelo Provedor de Serviço. Uma vez localizado o serviço, o cliente se conecta a ele por meio de uma vinculação (*bind*) transparente ao usuário.
- Registro de Serviço (*Service Registry*): É o lugar onde os *Service Providers* listam os seus serviços. Ele agrega e lista vários WS disponíveis na Internet, fornecendo informação sobre a companhia responsável pelo serviço, além de detalhes sobre a sua funcionalidade.

3.4 Operações

Segundo Kreger (2003), é possível identificar três operações básicas no funcionamento de um WS: a descrição, a localização/publicação e a vinculação (*describe, find/publish, bind*).

- Descrição – Um padrão chamado WSDL também usa o formato XML para descrever WS, definindo os métodos (funções) presentes, os parâmetros e as saídas de cada método e os tipos de dados usados.
- Publicação – O protocolo fornece uma maneira para que Provedores de

Serviço possam ser publicados.

- Vinculação – A vinculação é feita a partir do protocolo SOAP, criado a partir de um padrão definido em XML.

Nas subsecções a seguir, serão abordados cada uma destas tecnologias que permitem a implementação dos WS.

3.4.1 WSDL – *Web Services Description Language*

O WSDL foi criado para descrever e publicar os formatos e protocolos de um WS de uma forma padrão. Ele estabelece um modelo comum para a descrição e publicação do WS. Essa tecnologia permite que exista uma separação na descrição de funcionalidades abstratas oferecidas por um serviço dos detalhes concretos da descrição do mesmo, como por exemplo, de que forma e onde as funcionalidades do serviço são oferecidas (CHINNICI et al, 2003). Na prática, as definições em WSDL servem como uma receita para automatizar os detalhes de comunicação entre as aplicações.

O WSDL define vários componentes que descrevem um serviço (NEWCOMER, 2002):

- Tipos de dados: os tipos de dados, na forma de esquemas XML ou eventualmente algum outro mecanismo, a serem utilizados nas mensagens.
- Mensagem: uma definição abstrata dos dados, sob a forma de uma mensagem apresentada, seja como um documento inteiro ou como argumentos para serem mapeados para um método de invocação.
- Operação: a definição abstrata da operação de uma mensagem, como a nomeação de um método, fila de mensagem, ou processo comercial, que vai aceitar e processar a mensagem.
- *Port type*: um conjunto de operações.
- *Binding*: o protocolo concreto e os formatos de dados para as operações e mensagens definidas para um determinado *port type*.
- *Port*: uma combinação de um *binding* e um endereço de rede, fornecendo o endereço alvo do serviço da comunicação.
- Serviço: uma coleção de *port types*.

3.4.2 UDDI – *Universal Description Discovery and Integration*

UDDI é um registro de WSs e fornece uma especificação técnica para descrição, descoberta e integração de WS. Essa tecnologia permite que instituições publiquem e encontrem WSs (UDDI, 2000). Funciona de maneira análoga a um catálogo, pois permite que um cliente fique sabendo sobre as funcionalidades oferecidas e possa obter mais informações sobre os serviços disponíveis.

Em sua essência, UDDI é composto por duas partes (JONES, 2003): a primeira parte diz respeito à especificação técnica para a construção de um diretório distribuído de negócios e WS, onde os dados são armazenados em um formato XML especifico e a especificação UDDI inclui APIs (*Application Programming* Interface) detalhadas para a busca de dados cadastrados e para a distribuição de novos dados. A segunda parte consiste no Registro de Negócios UDDI (UDDI *Bussines Registry*), uma implementação completamente operacional da especificação UDDI.

Em UDDI, as informações descritas são geralmente divididas em 3 categorias principais (COLAN, 2002), apresentadas a seguir:

- *White Pages* (Páginas Brancas) – Essa categoria contem informações referentes a: nomes e endereços de negócios, informação para contato, nome do *Web Site*, DUNS (*Data Universal Numbering System*) ou algum outro número de identificação;
- *Yellow Pages* (Páginas Amarelas) – Contém informações relacionadas ao tipo de negócio, localização e produtos;
- *Green Pages* (Páginas Verdes) – Essa categoria contem informações técnicas sobre os WS (incluindo as especificações em WSDL).

A tecnologia UDDI utiliza os próprios padrões empregados em WSs (SOAP e WSDL) para cumprir com suas funcionalidades de registro e disponibilização de serviços. Sendo que esta característica garante sua universalidade (MARQUEZAN, CARISSIMI, NAVAUX, 2006).

3.4.3 SOAP – *Simple Object Access Protocol*

O SOAP é um protocolo baseado em XML, que permite a comunicação entre aplicações (BECKER, CLARO, SOBRAL, 2001) e, talvez seja a mais significante de

todas as tecnologias dos *Web Services*. É verdade que WSs não existiram sem uma forma abstrata para representar os dados e sem a sua publicação. Mas o SOAP realiza indiscutivelmente o mais importante aspecto dos WS: transportar os dados de um local para outro através da rede (NEWCOMER, 2002).

Mesmo podendo ser utilizado em uma grande variedade de sistemas de mensagens e poder ser transmitido por diversos protocolos, ele é projetado para ser um mecanismo simples, que pode ser alargado para englobar outras funções, funcionalidades e tecnologias. É basicamente um protocolo de um sentido só, ou seja, as mensagens transmitidas através do protocolo SOAP não são requisições que necessitam de respostas. Elas são simplesmente mensagens que contêm dados a serem enviados a um receptor. Entretanto, o SOAP possibilita a construção de padrões de interações complexos, como por exemplo, requisição/resposta, ou requisição/múltiplas-respostas.

Uma mensagem SOAP possui uma estrutura simples, contendo basicamente os seguintes componentes: envelope, cabeçalho (*header*) e corpo (*body*) (CURBERA et al, 2002). A estrutura de uma mensagem SOAP é apresentada na Fig. 9.

```
<SOAP: Envelope
        xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP: Cabecalho>
        <!- conteúdo do cabeçalho -->
    </SOAP: Cabecalho>
    <SOAP: Corpo>
        <!-- conteúdo do corpo -->
    </SOAP: Corpo>
</SOAP: Envelope>
```

Figura 9 - Estrutura de uma mensagem SOAP.
FONTE: Traduzido de CURBERA et al, 2002.

O envelope é o elemento obrigatório de uma mensagem SOAP, que designa o escopo da mensagem através da definição da "versão" utilizada. Ao contrário do que ocorre em HTML e XML, a versão em SOAP é definida pelo espaço de nomes utilizado, sendo ele o responsável por especificar os elementos e os atributos SOAP.

O cabeçalho é um componente opcional, podendo haver mais de um cabeçalho em uma mensagem SOAP. A inserção de um cabeçalho não pode

comprometer a capacidade de processamento da mensagem SOAP, isto é, se um nodo SOAP não é capaz de entender o conteúdo de um cabeçalho ele deverá continuar processando a mensagem. Entretanto, pode-se forçar os processadores SOAP a interpretar o conteúdo de um cabeçalho. Neste caso, se um nodo não compreender seu conteúdo ele irá parar o processamento da mensagem.

O corpo de uma mensagem SOAP é quem realmente possui o conteúdo que deverá ser processado no receptor SOAP apropriado. Ele é um elemento obrigatório em qualquer mensagem. Devido a flexibilidade do protocolo SOAP, o corpo da mensagem pode servir para diversos propósitos. Por exemplo, pode servir para uma transmissão de documentos XML ou então para a chamada de um procedimento remoto, através das convenções SOAP-RPC (SOAP-*Remote Procedure Call*). Em uma mensagem podem existir mais de um corpo, assim podem existir vários cabeçalhos. Entretanto, o funcionamento dessas múltiplas entradas do corpo SOAP é diferenciado. A primeira entrada é a que realmente contém o RPC ou o documento XML literal, as outras entradas possuirão referencias que dão suporte à primeira entrada.

3.5 Funcionamento dos *Web Services*

Para facilitar e deixar mais claro alguns dos possíveis usos para os WSs, nesta subseção será apresentado uma variação de um exemplo clássico desta área.

Imagine uma agência de turismo *online* que permite aos seus clientes comprarem um pacote turístico, o qual inclui serviços como: reservas em hotéis; passagens aéreas; e espetáculos culturais e esportivos.

O cliente pode acessar o site desta agência e, através de uma plataforma disponibilizada ali por ela, a qual se comunica com outras empresas através do uso dos WSs, adquirir o seu pacote de forma totalmente personalizada, conforme a disponibilidade dos serviços oferecidos.

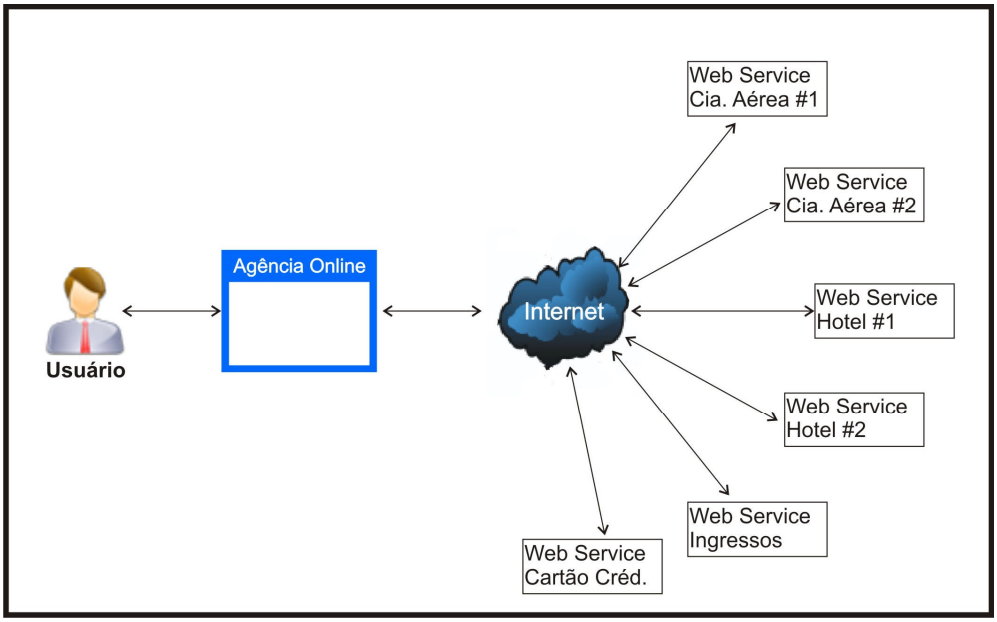

Figura 10 - Exemplo de utilização de WS.

Considerando que o cliente finalize a compra do pacote, uma série de passos deverá ser executada a fim de confirmar a compra de todos esses serviços. Neste caso, o cliente deverá confirmar então a compra de todo o pacote fornecendo os dados de seu cartão de crédito, na plataforma da agência de turismo. Logo, é necessário que a comunicação entre a plataforma da agência de turismo e os WS da administradora de cartão de crédito seja feita de alguma forma segura, bem como entre a agência de turismo e seus fornecedores a fim de se garantir a integridade e confidencialidade das mensagens.

Outro detalhe importante: supondo que a plataforma realize as reservas das passagens e do hotel e, em seguida, ao tentar realizar a compra de ingressos para uma partida de futebol, esta compra não possa ser realizada, devido ao fim dos ingressos, por exemplo. Neste caso, a solicitação do cliente não poderá ser contemplada, porém foi parcialmente realizada.

Sendo assim, a plataforma deverá possuir um mecanismo semelhante a uma transação, a fim de que seja possível desfazer as operações realizadas, visto que é necessário que todas sejam concluídas com sucesso e não apenas algumas.

Finalmente, a seqüência de todo o processo de compra dos vários serviços, e a comunicação com a empresa de cartões de crédito a fim de realizar cada transação precisa ser organizada e gerenciada de alguma forma, ou seja, é necessária uma orquestração de vários WSs a fim de que o resultado esperado aconteça. Neste exemplo isto é importante visto que o cliente poderá adquirir um ou

mais serviços e em cada situação, a ordem e a seqüência das operações será diferente uma das outras.

Considerando que a comunicação se dá através de WSs, é necessária a presença de mecanismos que garantam todos os comportamentos especificados.

4 *Web Service* para consulta ao contexto

A partir desta seção o foco passa a ser o projeto desenvolvido, ou seja, será detalhada a construção do WS para comunicação com o Subsistema de Reconhecimento de Contexto do *middleware* GRADEp para passagem dos dados de contexto para o portal *web*.

4.1 Subsistema de Reconhecimento de Contexto

Como dito anteriormente na Introdução, é no Subsistema de Adaptação e Reconhecimento de Contexto que ocorre o gerenciamento da informação de contexto e a sua adaptação. Portanto, ele inclui serviços que tratam desde a extração da "informação bruta", ou seja, as informações vinda diretamente dos sensores, sobre as características dinâmicas e estáticas dos recursos que compõe o Ambiente Pervasivo, passando pela identificação em alto nível dos elementos de contexto, até o disparo das ações de adaptação em reação a modificações no estado de tais elementos de contexto. Sendo assim, o subsistema é o responsável pela absorção das informações e a posterior reação a elas (adaptação).

4.1.1 EXEHDA-ON

Uma questão importante na sensibilidade ao contexto é o grau de expressividade de uma informação. Quanto maior for o grau de expressividade, maior é a capacidade da informação ser interpretada e representada da forma correta. Por isto, LOPES (2008) propôs uma abordagem baseada em ontologias para reconhecimento de contexto através do projeto EXEHDA-ON, o qual tem como

premissa a possibilidade de empregar uma semântica de maior expressividade que a usualmente praticada na coleta dos dados de contexto. No processamento dos dados sensorados, permite-se também atingir melhores níveis de descrição das informações que caracterizam o contexto do ambiente computacional (LOPES, 2008). Na Fig. 11, pode-se visualizar a árvore de classificação de conceitos, onde é mostrada a estrutura hierárquica das classes definidas para a ontologia do ambiente pervasivo, tratado no EXEHDA-ON. Como se pode perceber na Fig. 11, a ontologia descreve, até o momento, conceitos ligados ao ambiente físico de cada célula.

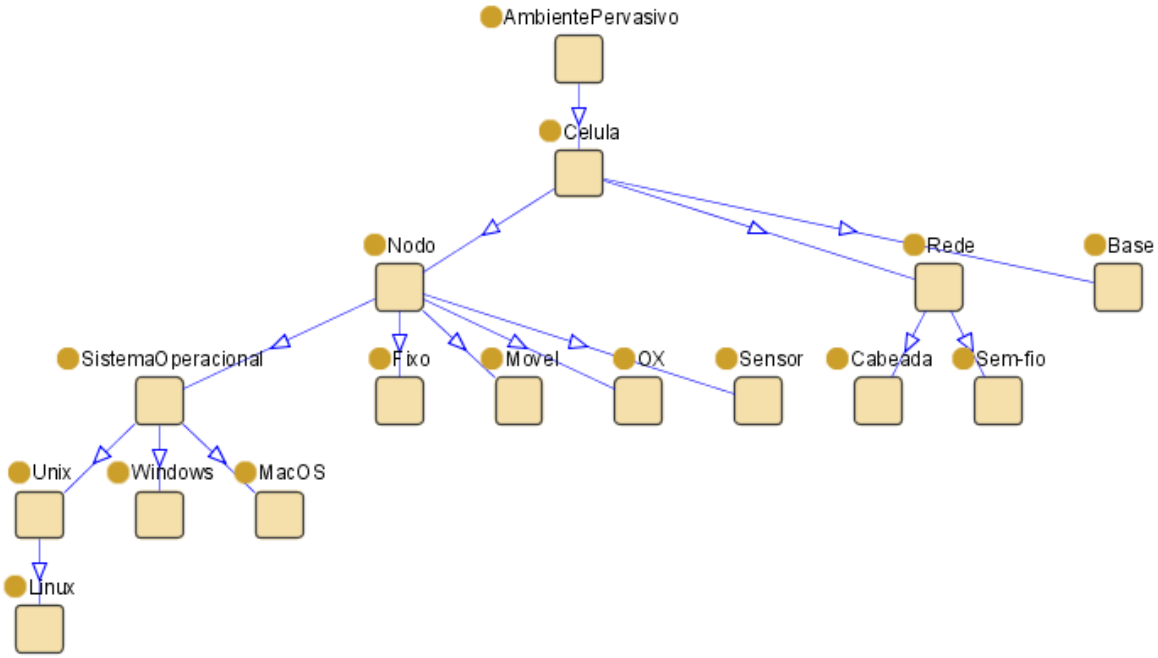

Figura 11 – Árvore de conceitos da ontologia do ambiente pervasivo.
FONTE: LOPES, 2008.

As ontologias já vêm sendo utilizadas por várias áreas da ciência da computação, principalmente com o intuito de dotar os sistemas de meta-conhecimento. A utilização de ontologias para descrição semântica de um determinado vocabulário proporciona um entendimento amplo das características e propriedades das classes pertencentes a um domínio, assim como seus relacionamentos e restrições (PERNAS, 2004).

Atualmente, a definição mais amplamente aceita e citada pelos autores da área de computação é a que define ontologia como uma “especificação formal e explicita de uma conceituação compartilhada” (GRUBER, 1993) (FENSEL, 2000) onde: (a) conceituação se refere ao modelo abstrato do mundo real. (b) explicita

significa que conceitos e seus requisitos são definidos explicitamente. (c) formal indica que a ontologia é processável por maquina, permite raciocínio automático e possui semântica lógica formal. (d) compartilhada significa que uma ontologia captura o conhecimento apresentado não apenas por um único indivíduo, mas por um grupo.

A necessidade de confiabilidade com relação aos conceitos do vocabulário ou linguagem que se esta utilizando em certo ambiente é outro forte motivo para utilização de ontologias, pois a representação formal adquirida com a aplicação das mesmas pode tornar possível a automação da checagem de consistência, resultando em ambientes mais confiáveis (GRUNINGER, 1996).

Além do exposto acima, pode-se destacar algumas vantagens da utilização de ontologias na área de Ciência da Computação (LOPES, 2008):

- conhecimento representado através de um vocabulário, o qual possui uma conceituação que o sustenta e evita interpretações ambíguas;
- compartilhamento e reuso da ontologia, que modele adequadamente certo domínio por pessoas que desenvolvam aplicações dentro desse domínio;
- descrição exata do conhecimento fornecido por uma ontologia, em função de sua escrita em linguagem formal, a qual evita o *gap* semântico existente na linguagem natural, na qual as palavras podem ter semântica totalmente diferente conforme o seu contexto. Por exemplo, a palavra “memória” pode estar associada a um dispositivo de armazenamento de dados em um computador, bem como pode se referir a memória humana (capacidade de natureza psicológica de adquirir, armazenar e evocar informações). A interpretação da palavra pode ser atribuída a um conceito ou outro conforme o estado mental do individuo. Então, no exemplo, se existir uma conceituação comum e as pessoas envolvidas concordarem em uma ontologia sobre o domínio “computadores”, possivelmente não haverá mal entendido;
- possibilidade de fazer o mapeamento da linguagem da ontologia sem que com isso seja alterada a sua conceituação, ou seja, uma mesma conceituação pode ser expressa em varias línguas;
- possibilidade de estender o uso de uma ontologia genérica de forma a que ela torne-se adequada a um domínio especifico.

O EXEHDA-ON possui diversos serviços encarregados de processar e distribuir as informações de contexto com base na ontologia de LOPES (2008). A Fig. 12 mostra uma visão desses serviços e de seus relacionamentos.

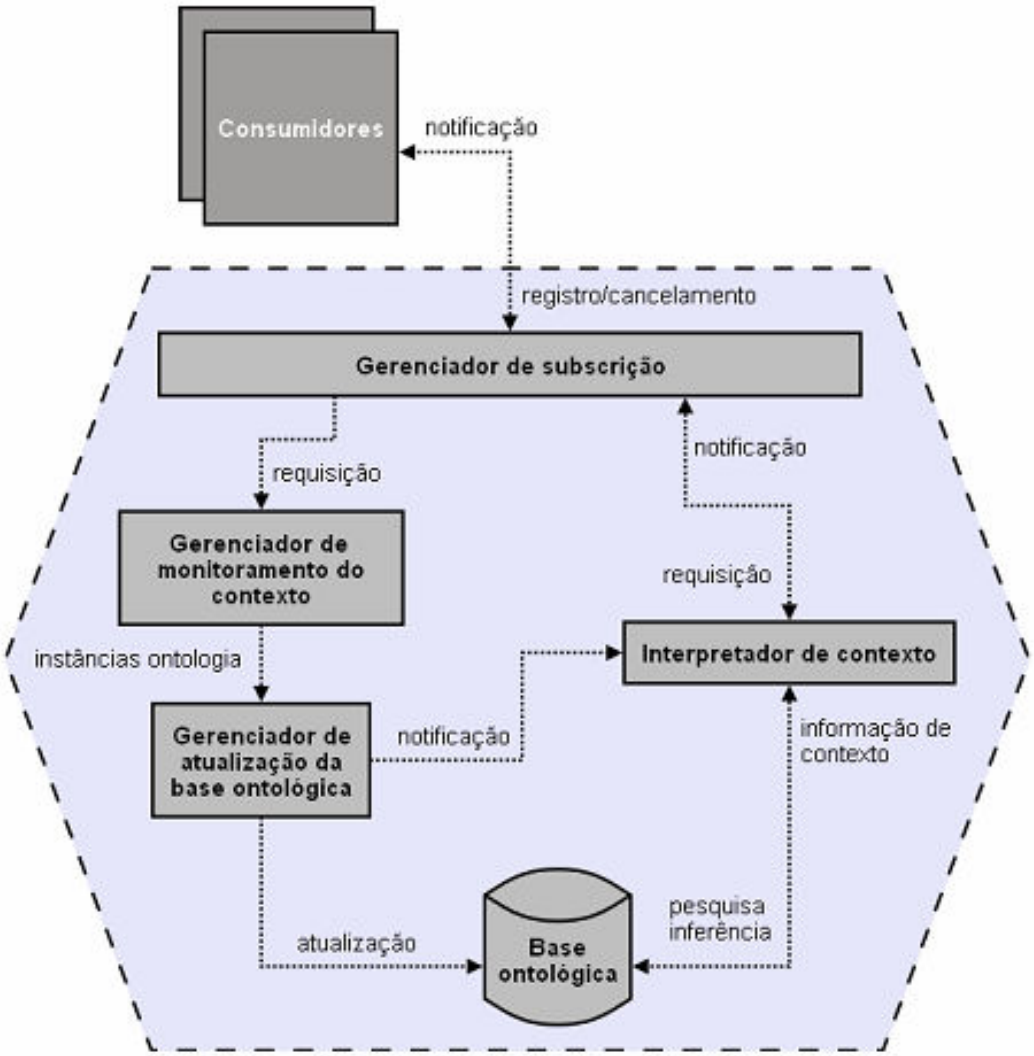

Figura 12 - Visão geral dos serviços do EXEHDA-ON.
FONTE: LOPES, 2008.

No EXEHDA-ON o consumidor registra suas requisições aos serviços de contexto através do Gerenciador de subscrição, o qual também é responsável pelo cancelamento da subscrição quando o consumidor não tiver mais interesse em alguma informação de contexto. Enfim, cabe ao Gerenciador de subscrição gerenciar as solicitações enviadas pelos consumidores e distribuí-las aos demais serviços que formam a arquitetura do EXEHDA-ON (LOPES, 2008).

No serviço Gerenciador de monitoramento do contexto é onde ficam os sensores que detectam alterações nos estados de contexto, passando para o Gerenciador de atualização da base ontológica a incumbência de atualizar a Base

ontológica criando uma nova instância ou então atualizando uma instância já existente. O Gerenciador de atualização da base ontológica também deve notificar o Interpretador de Contexto de que ocorreu uma atualização na Base ontológica, para que ele faça uma consulta a ela, verificando o novo estado e disparando as notificações necessárias para os consumidores. O Interpretador de contexto também pode realizar consultas requisitadas pelos Consumidores, as quais o Gerenciador de subscrição encaminha para ele que após realizá-la devolve uma notificação para este (LOPES, 2008).

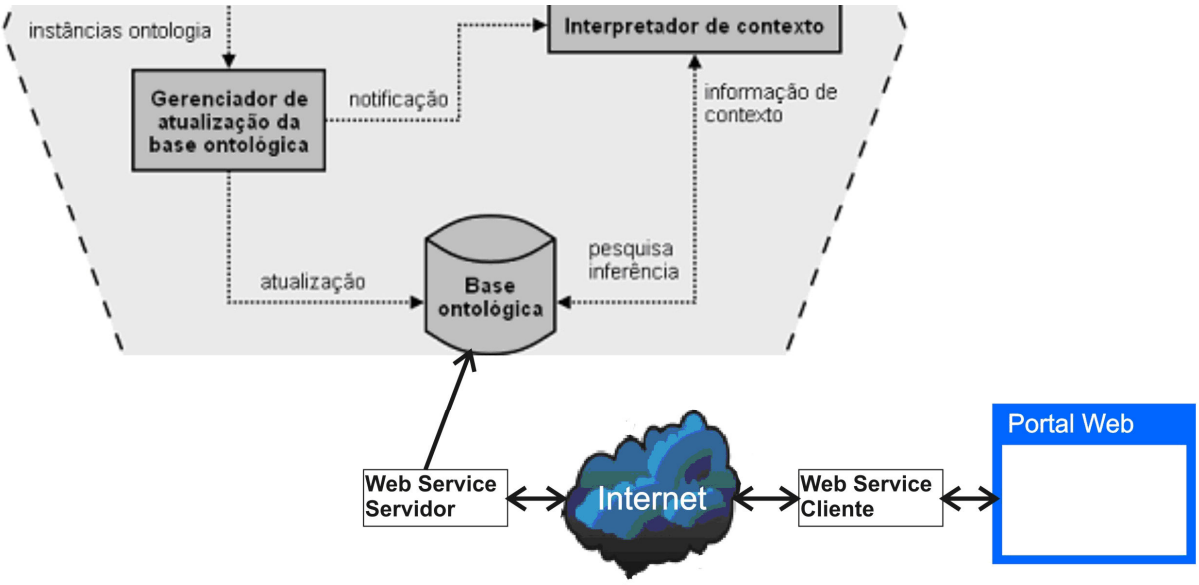

Figura 13 - Visão geral do projeto desenvolvido.

O WS construído (e descrito a partir da próxima seção) possui dois serviços complementares: servidor e cliente. No lado do serviço cliente, o WS irá capturar informações existentes no portal (providas pelos usuários, por exemplo) e as repassará através da Internet para o serviço competente localizado no lado servidor do WS. Este serviço existente no servidor irá acessar a Base ontológica para realização das operações de consulta, monitoramento e instanciação. Posteriormente, será feito o caminho inverso até que as informações sejam exibidas no portal *web*. Todo este processo de captura e repasse de informações é detalhado nas próximas seções deste trabalho.

4.2 Desenvolvimento do *Web Service*

O desenvolvimento do WS foi realizado em três etapas: análise de requisitos; definição da infra-estrutura utilizada para criação do WS, tanto no lado cliente como no lado servidor; e implementação propriamente dita, onde foram desenvolvidos os serviços providos pelo servidor e o portal acessado pelo cliente. Estes passos estão detalhados nas próximas seções.

4.2.1 Análise de requisitos

Para projetar o sistema, alguns requisitos básicos foram considerados. São eles:

- A entrada dos dados deve ser validada quando estes forem informados no portal *web*, fazendo com que somente dados válidos sejam repassados ao WS.
- O sistema deve possuir acesso à ontologia no momento de sua inicialização e somente poderá seguir com sua execução normal caso este acesso seja realizado com sucesso. Caso contrário, o sistema deve enviar uma mensagem indicando o erro e a não execução de qualquer operação.

4.2.2 Infra-estrutura de Desenvolvimento

Antes de retratar o desenvolvimento do WS, é preciso falar da infra-estrutura utilizada para desenvolvimento dos serviços.

4.2.2.1 Infra-estrutura de implementação do *Web Service* Servidor

Para o desenvolvimento do WS servidor, foram utilizadas uma série de ferramentas, em especial a linguagem de programação Java, utilizada também na construção do EXEHDA-ON e do GRADEp. Esta e outras ferramentas são todas descritas a seguir.

Linguagem de Programação: Foi utilizada a linguagem JAVA SDK (*Software Development* Kit) versão 6, *update* 10.

Ambiente de Desenvolvimento: Foi utilizado o IDE (*Integrated*

Development Environment) NetBeans[2] 6.1 instalado com Java versão 6, update 10 para escrita do código relativo a este projeto. A escolha desse ambiente de desenvolvimento se deve ao fato de que o NetBeans possui integração direta com a linguagem Java e com o servidor GlassFish[3].

Servidor: Foi utilizado ao longo do desenvolvimento do WS do lado servidor o servidor de aplicação GlassFish em sua versão 2 sobre o qual era executada e testada a aplicação. Foi do servidor também a responsabilidade de gerar o arquivo WSDL com a descrição do WS. O GlassFish é um servidor *open source* o qual vêm integrado com o pacote de instalação da IDE Netbeans 6.1. É desenvolvido pela Sun Microsystems para a plataforma Java EE (Java *Enterprise Edition*) e a sua versão comercial é chamada Sun Java *System Application Server*.

API Jena[4]: O Jena é um *framework* desenvolvido para construir aplicações de Web Semântica, provendo um ambiente de programação para várias linguagens associadas à Web Semântica. Esse *framework* é utilizado tanto pelo EXEHDA-ON para manipular a ontologia de acordo com os dados passados dos diversos sensores espalhados pelo ambiente pervasivo, como também pelo próprio WS servidor para monitorar, instanciar e consultar a ontologia.

4.2.2.2 Infra-estrutura de implementação do *Web Service* Cliente

Na construção do WS cliente, que consiste de um portal, cabe destacar a importância da utilização da linguagem de programação PHP 5, a qual facilita o desenvolvimento da interface e tem plena aceitação de WSs. A seguir são descritas estas e outras ferramentas utilizadas no desenvolvimento do WS cliente.

Linguagem de Programação: Para o desenvolvimento do WS Cliente foi utilizada a linguagem PHP (*Hypertext Preprocessor*) na sua versão 5. O PHP é uma linguagem de programação interpretada, livre e muito utilizada para gerar conteúdo dinâmico na *web*.

Ambiente de Desenvolvimento: Foi utilizado o software Dreamweaver[5] na versão 8 *trial*, quando ainda era produzido pela Macromedia. Atualmente, o *software* é fabricado pela Adobe Systems, e encontra-se na versão CS 4.

---

2 http://www.netbeans.com
3 https://glassfish.dev.java.net/
4 http://jena.sourceforge.net/
5 http://www.adobe.com/products/dreamweaver/

Servidor: Foi utilizado para execução do WS do lado cliente o servidor Apache da *Apache Software Foundation* o qual é um servidor *web* livre que interpreta a linguagem PHP.

Classe NuSOAP: A classe NuSOAP, desenvolvida em PHP, foi utilizada na sua versão 0.73, para permitir a comunicação do WS cliente com o WS servidor através do protocolo SOAP. Permite não só consumir WS como também criar novos WS para serem consumidos.

Já para a criação do portal *web,* foi necessário a utilização de outras tecnologias voltadas para o desenvolvimento de interfaces *web*. São elas:

Linguagens de Programação: HTML (*HyperText Markup Language*) e CSS (*Cascading Style Sheets*) para a formatação da interface, JavaScript para a validação dos dados de entrada e PHP para formatação dos dados para comunicação com o WS.

Framework SAJAX[6]: Foi utilizado o framework SAJAX (*Simple Asynchronous Javascript And XML*) para realizar a requisição do WS via PHP sem a necessidade da página ser recarregada.

Ambiente de Desenvolvimento: Assim como no WS Cliente, foi utilizado o software Dreamweaver na versão 8 *trial*, quando ainda era produzido pela Macromedia. Atualmente, o software é fabricado pela Adobe Systems, e encontra-se na versão CS 4.

4.2.3 Implementação dos Serviços

Como mencionado anteriormente na seção 3, para desenvolver um WS ele precisa ser vinculado, descrito e publicado.

A vinculação ocorre através do protocolo SOAP, com o consumidor chamando cada uma das operações disponíveis separadamente, quando for de seu interesse.

Nesta seção serão detalhados os desenvolvimentos dos serviços no lado servidor e cliente. No lado servidor este é mais complexo, pois é quem oferece todo acesso aos dados de contexto aos clientes. Já no lado cliente o serviço é mais simples, pois ele requisita os serviços prestados pelo servidor.

[6] http://www.modernmethod.com/sajax/

4.2.3.1 Servidor

São três as operações que o WS servidor realiza sobre as ontologias que descrevem o contexto do *middleware* GRADEp: monitoramento, consulta e instanciação. A seguir, será detalhado o funcionamento de cada uma destas operações.

- Monitoramento: O monitoramento é uma operação relativamente simples, a qual não recebe nenhum parâmetro de entrada. Ela é a operação responsável somente por mostrar ao cliente os estados de cada nodo de contexto do ambiente pervasivo. Isto é feito listando as instâncias presentes nas classes “Fixo” e “Móvel”, as quais representam os nodos de cada célula do GRADEp. Para cada instância encontrada, são listadas, então, as suas propriedades. A seguir, está apresentada parte da descrição WSDL referente a esta operação.

```
- <operation name="monitora">
    <soap:operation soapAction=""/>
  - <input>
      <soap:body use="literal"/>
    </input>
  - <output>
      <soap:body use="literal"/>
    </output>
  - <fault name="FileNotFoundException">
      <soap:fault name="FileNotFoundException" use="literal"/>
    </fault>
  </operation>
```

Figura 14 - Parte do WSDL referente operação de monitoramento.

- Consulta: A consulta talvez seja a operação mais interessante disponibilizada, visto que, com ela pode-se consultar no ambiente pervasivo somente os nodos com as características que se tem interesse. Esta operação ocorre através de mecanismos de filtragem das propriedades a serem apresentadas pelos nodos, de forma que são passados como parâmetros dez valores, organizados da seguinte forma:
  - 1º parâmetro: operação referente à carga da CPU.
  - 2º parâmetro: valor referente à carga da CPU.
  - 3º parâmetro: operação referente à total de memória.

- 4º parâmetro: valor referente à ocupação de memória.
- 5º parâmetro: operação referente à ocupação de memória.
- 6º parâmetro: valor referente à ocupação de memória.
- 7º parâmetro: operação referente ao espaço total do disco.
- 8º parâmetro: valor referente ao espaço total do disco.
- 9º parâmetro: operação referente ao espaço disponível em disco.
- 10º parâmetro: valor referente ao espaço disponível em disco.

De posse destes dados, a operação “varre” todos os nodos presentes no ambiente pervasivo em busca dos que satisfaçam estes valores. A descrição WSDL para esta operação pode ser vista na Fig. 15.

```
- <operation name="consulta">
    <soap:operation soapAction=""/>
  - <input>
      <soap:body use="literal"/>
    </input>
  - <output>
      <soap:body use="literal"/>
    </output>
  - <fault name="FileNotFoundException">
      <soap:fault name="FileNotFoundException" use="literal"/>
    </fault>
  </operation>
```

Figura 15 - Parte do WSDL referente operação de consulta.

- Instanciação: A operação de instanciação permite que novos nodos sejam inseridos no ambiente pervasivo. Recebe como parâmetro dois valores do tipo *String*. Um deles representa o tipo de nodo que vai ser inserido. Para este parâmetro, somente dois valores são possíveis: móvel ou fixo. A partir deste parâmetro é verificado o atributo de identificação, “ID”, que o nodo irá receber. Por exemplo, se já existir cinco nodos móveis definidos na ontologia, o próximo nodo móvel a ser inserido será o nodo de número 6, e receberá como atributo de identificação o valor “Movel_6”. No outro parâmetro é recebida também uma *string* com o nome das propriedades a serem inseridas, bem como seus valores, de forma concatenada. Na instanciação, portanto, esta *string* é decomposta e, para determinadas instâncias “Fixo_x” ou “Movel_x” a serem criadas, são adicionadas as propriedades recebidas como parâmetro. É possível a instanciação de até

seis propriedades:

- Tipo de Dispositivo (*string*): Indica o tipo de dispositivo que está sendo instanciado. Por exemplo, pode ser adicionado ao ambiente pervasivo um computador, um supercomputador, um PDA (*Personal Digital Assistant*), um *notebook*, ou um outro dispositivo computacional qualquer.
- Carga da CPU (*float*): Indica a porcentagem de carga de processamento que o nodo está recebendo o nodo.
- Total de Memória: Indica a memória total, em *Megabytes* (MB), do dispostivo.
- Ocupação de Memória: Indica o quanto desta memória, também em *Megabytes* (MB), está ocupada.
- Tamanho Total de Disco: Indica a capacidade de armazenamento, em *Gigabytes*, do dispositivo.
- Espaço Disponível em Disco: Indica quanto espaço de armazenamento, também em *Gigabytes*, ainda está disponível no dispositivo.

De conhecimento destas propriedades, é possível entender o valor do segundo parâmetro de forma mais clara. Este parâmetro pode receber, por exemplo,

“TipoNodo##Notebook;;CargaCPU##20;;TotMemoria##512;;OcupMemoria ##256;;TotDisco##80;;DiscoDisponivel##50“ onde “;;” é um delimitador que separa as propriedades e, “##” é um delimitador que separa o nome da propriedade do seu respectivo valor. A Fig. 16 mostra o WSDL referente a esta operação.

```
- <operation name="instancia">
    <soap:operation soapAction=""/>
  - <input>
      <soap:body use="literal"/>
    </input>
  - <output>
      <soap:body use="literal"/>
    </output>
  - <fault name="FileNotFoundException">
      <soap:fault name="FileNotFoundException" use="literal"/>
    </fault>
  </operation>
```

Figura 16 - Parte do WSDL referente operação de instanciação.

A descrição completa de todo o documento WSDL referente ao WS criado encontra-se no Apêndice A.

A publicação, como já dito, ocorre através do protocolo UDDI. Porém, como este projeto trata-se de um protótipo, este WS não foi publicado.

4.2.3.2 Cliente e Portal *Web*

A proposta inicial deste projeto previa a construção de um portal *web*, o qual serviria como interface para serem feitas as requisições dos usuários e também para devolver o resultado destas requisições. Portanto, são descritos a seguir os detalhes da implementação desta interface.

O portal *web* foi concebido para a exibição de quatro telas. Cada uma delas é descrita a seguir:

Apresentação: A tela de apresentação mostra um resumo referente a este projeto, introduzindo alguns conceitos abordados e explicando o que ele provê.

Figura 17 - Tela do Portal Web de apresentação do projeto.

GRADEp: Tela de apresentação do *middleware* GRADEp a qual revela a motivação do projeto e descreve simplificadamente o que é o GRADEp.

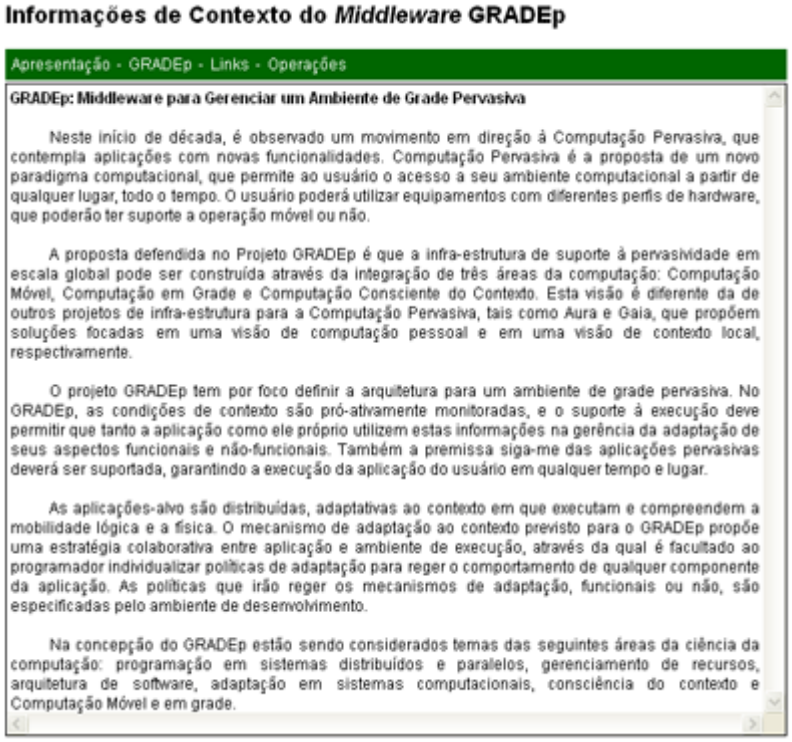

Figura 18 - Tela do Portal Web que detalha o projeto GRADEp.

Links: Esta tela apresenta para *sites*, artigos científicos, teses e dissertações relacionadas com o assunto foco deste trabalho: a Computação Pervasiva.

Figura 19 - Tela do portal web com outros links sobre Computação Pervasiva.

Operações: A tela referente às operações apresenta para o usuário a possibilidade de realizar o monitoramento das informações de contexto, onde poderá enxergar o estado de cada nodo presente no ambiente pervasivo. O usuário poderá também realizar consultas para verificar quais nodos satisfazem uma determinada situação. Além disto, é permitido, através desta mesma tela, a instanciação de novos nodos no ambiente pervasivo.

Figura 20 - Tela do Portal Web referente a escolha das operações.

De acordo com a operação selecionada, outras três telas poderão ser exibidas referentes a cada operação.

Na tela referente à consulta, o usuário pode preencher, ou não, até cinco opções de pesquisa. Para cada propriedade, ele pode escolher a condição (são cinco: >, >=, =, <= e <) e o valor que deseja comparar. Por exemplo, na Fig. 21, é feita uma consulta por nodos que tenham a carga da CPU maior que 30%, e total de memória maior ou igual a 512MB, e ocupação de memória igual a 256MB, e tamanho total de disco menor ou igual a 40GB, e, por fim, espaço disponível em disco menor que 10GB.

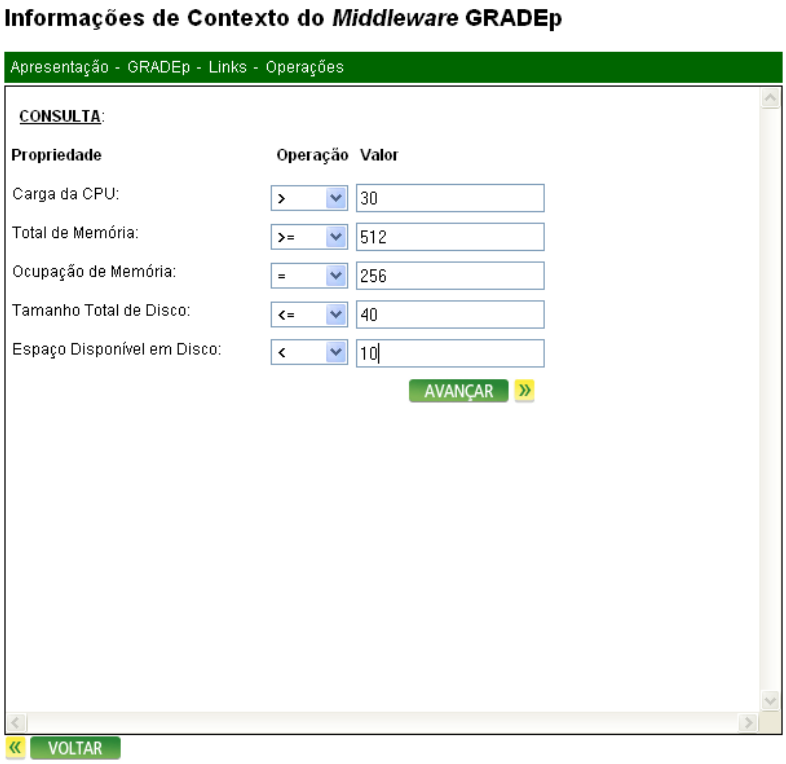

Figura 21 - Tela do Portal Web referente a operação de consulta.

Na tela referente ao monitoramento é fornecido o retorno ao usuário sobre quais nodos estão presentes no ambiente pervasivo e as suas características (tipo de nodo, tipo de dispositivo, carga da CPU, quantidade total de memória, ocupação de memória, tamanho total de disco e espaço disponível em disco), quando definidas. Como trata-se de uma tela de monitoramento, enquanto acessada, ela faz uma verificação do ambiente pervasivo a cada cinco minutos para verificar se ocorreram mudança no cenário, apresentando, inclusive uma contagem regressiva na parte inferior direita da tela, indicando quanto tempo falta para realizar esta verificação e atualizar as informações.

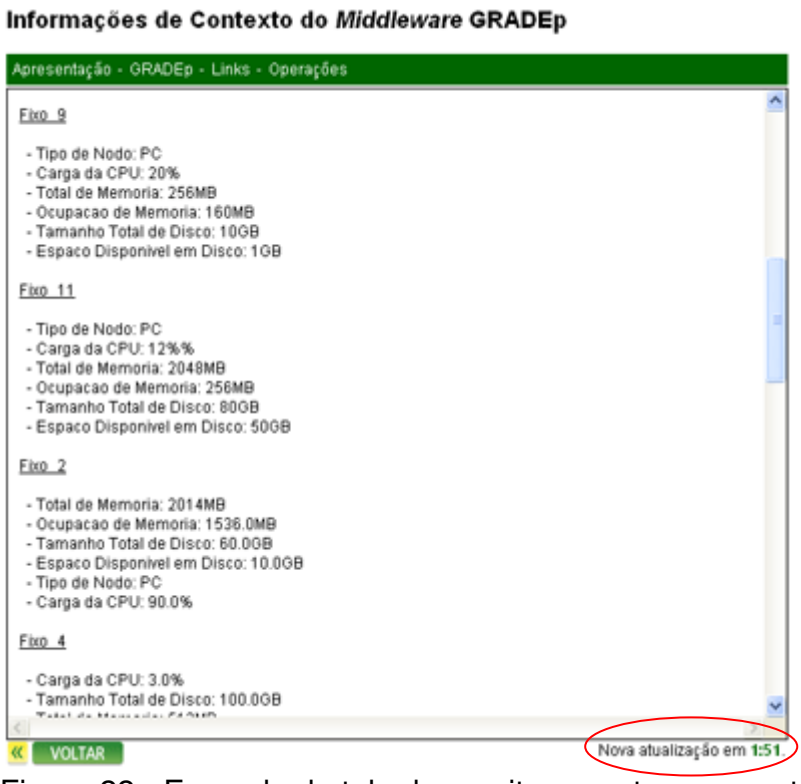

Figura 22 - Exemplo de tela de monitoramento com contagem regressiva.

Na tela referente à instanciação o usuário deverá selecionar, portanto, o tipo de nodo que ele deseja inserir no ambiente pervasivo. De acordo com o tipo escolhido, o nodo poderá ter um atributo de identificação “Fixo_x” ou “Movel_x” conforme explicado na sessão anterior. Em seguida, o usuário terá a oportunidade de definir as propriedades do nodo em questão. Caso alguma propriedade seja mantida com valor em branco, ela não será atribuída ao nodo na ontologia do ambiente pervasivo.

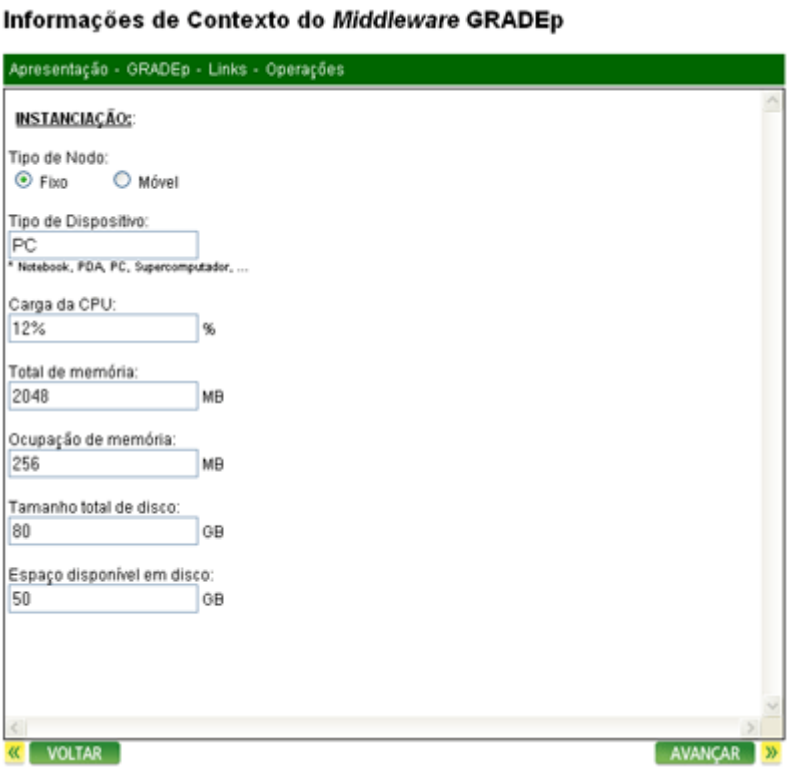

Figura 23 - Exemplo de tela de instanciação de novos nodos.

5 Análise dos Resultados

Neste capítulo serão apresentados os resultados obtidos com o desenvolvimento do WS, assim como a maneira com que estes resultados são apresentados ao consumidor através do portal *web*.

O WS construído foi testado através do portal *web a* partir de uma ontologia baseada no modelo do EXEHDA-ON, simulada com instâncias fictícias, produzidas manualmente por NUNES (2008) utilizando-se do editor de ontologias Protege[7].

A ontologia modificada possui informações sobre oito nodos computacionais, descritos na Tabela 1 a seguir:

Tabela 1 - Dados dos nodos e suas propriedades na ontologia modificada

ID do nodo	Tipo do nodo	Carga da CPU (%)	Total de memória (MB)	Ocupação de memória (MB)	Tamanho total de disco (GB)	Espaço disponível em disco (GB)
Fixo_1	PC	50	1024	768	80	40
Fixo_2	PC	90	2014	1536	60	10
Fixo_3	Servidor	100	1556	1536	80	N.I.
Fixo_4	PC	3	512	128	100	0,5
Fixo_5	Supercomputador	20	4096	256	N.I.	N.I.
Movel_1	PDA	N.I.	128	64	20	0,2
Movel_2	Notebook	N.I.	128	0	N.I.	N.I.
Movel_3	Notebook	10	512	100	N.I.	N.I.

Na Tabela 1, N.I. indica que a informação sobre a propriedade não foi

7 http://protege.stanford.edu/

inserida na ontologia.

A partir desta ontologia, foram realizados quatro testes através do portal *web* desenvolvido. Os dois primeiros referem-se à operação de consulta. Já no terceiro é realizado o teste de monitoramento, enquanto no último é realizado o teste da instanciação, seguido de uma operação de monitoramento para comprovar que a instanciação foi realizada.

5.1 Consulta

5.1.1 Primeiro teste

Neste primeiro teste, é feita apenas uma consulta simples sobre a ontologia em questão. Neste caso, a consulta requer somente os nodos com Carga de CPU maior que 30% e que tenham ocupação de memória igual a 1536MB. Como pode-se visualizar na Fig. 24, a consulta trás como resultado somente os nodos Fixo_2 e Fixo_3.

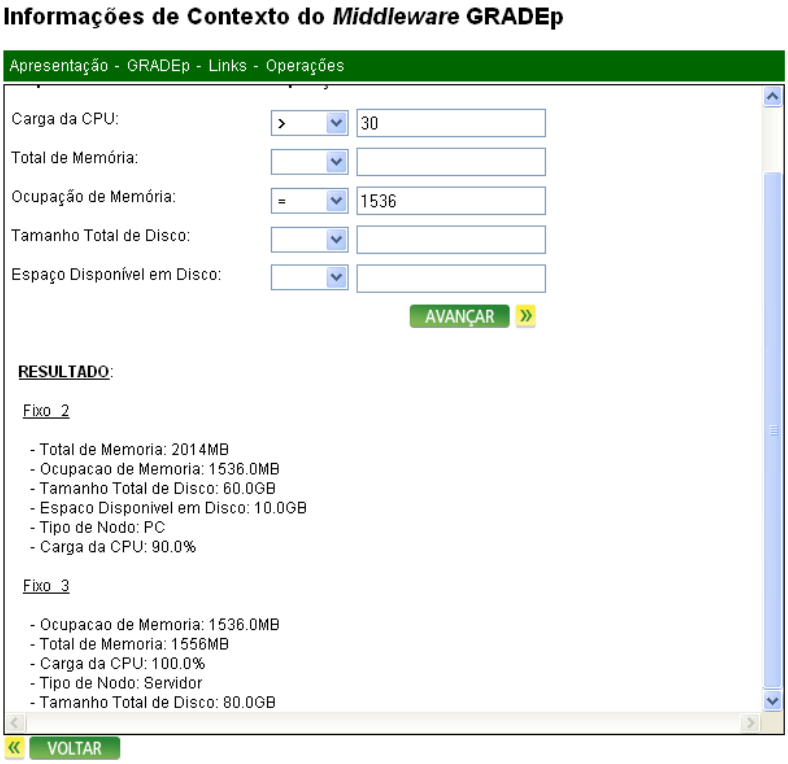

Figura 24 - Primeiro teste sobre a operação de consulta.

Neste teste percebe-se que não é preciso preencher todas as propriedades.

O usuário pode escolher somente as que desejar para preencher com os dados que refinaram a operação de consulta.

5.1.2 Segundo teste

O segundo teste da consulta realiza uma busca que envolve todas as propriedades disponíveis sobre os nodos. Neste teste, é feita uma consulta para buscar os nodos que possuam carga de CPU maior ou igual a 10%, total de memória menor que 2048MB, ocupação de memória maior que 100MB, tamanho total de disco maior que 60GB e espaço disponível em disco igual a 40GB. Portanto, o único nodo que atende a todos estes requisitos é o Fixo_1.

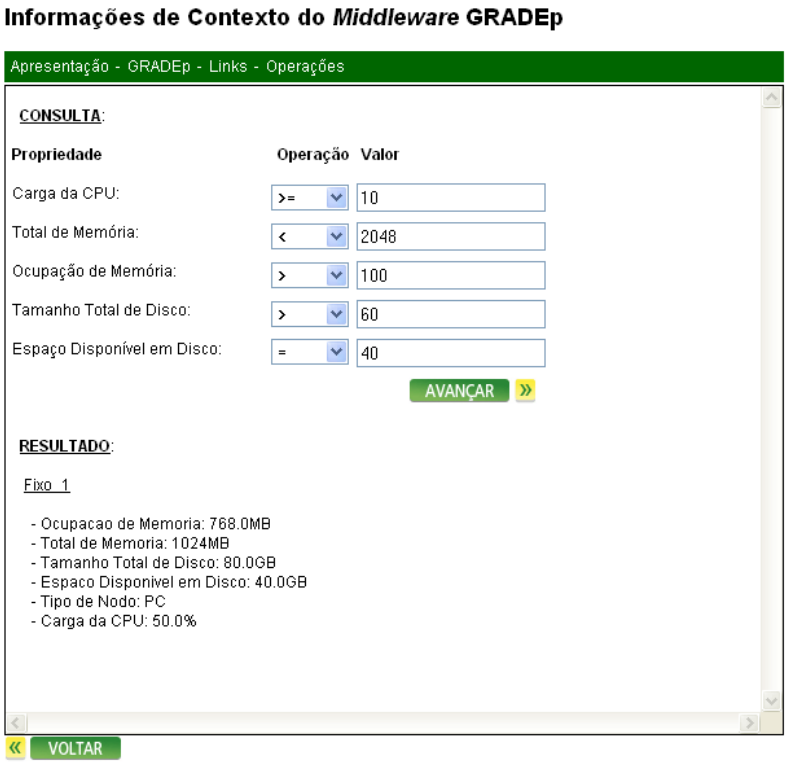

Figura 25 - Segundo teste sobre a operação de consulta.

5.2 Monitoramento

No teste do monitoramento, é feita uma requisição para a operação de monitoramento do portal *web*, a qual irá buscar, via WS, o estado atual do ambiente pervasivo. Como resultado, foi obtida uma lista dos nodos fixos e móveis instanciados no ambiente pervasivo que retratam um cenário igual ao apresentado na Tabela 1. Desta forma, a operação de monitoramento foi executada sem erros.

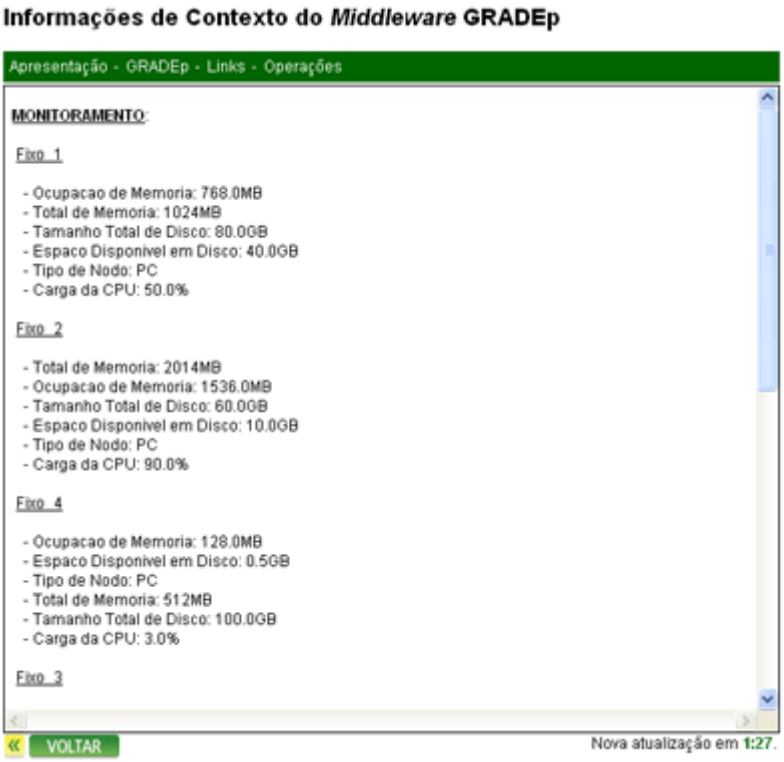

Figura 26 - Teste do monitoramento.

A listagem completa de todos os nodos encontrados no ambiente pervasivo pela operação de monitoramento encontra-se no Apêndice B.

5.3 Instanciação

O teste de instanciação está dividido em dois momentos. No primeiro, onde ocorre a instanciação de um nodo **móvel** com as seguintes características:

- Tipo de dispositivo: *Notebook*.
- Carga da CPU: 30%.
- Total de memória: 1024 MB.
- Ocupação de memória: 392 MB.
- Tamanho total de disco: 250GB.
- Espaço disponível em disco: 210GB.

Depois de instanciado o nodo, é feita uma operação de monitoramento, a qual serve para mostrar o novo cenário do ambiente pervasivo, já contando com o nodo móvel inserido e provando o sucesso da operação de instanciação. Este teste é retratado a seguir na Fig. 27.

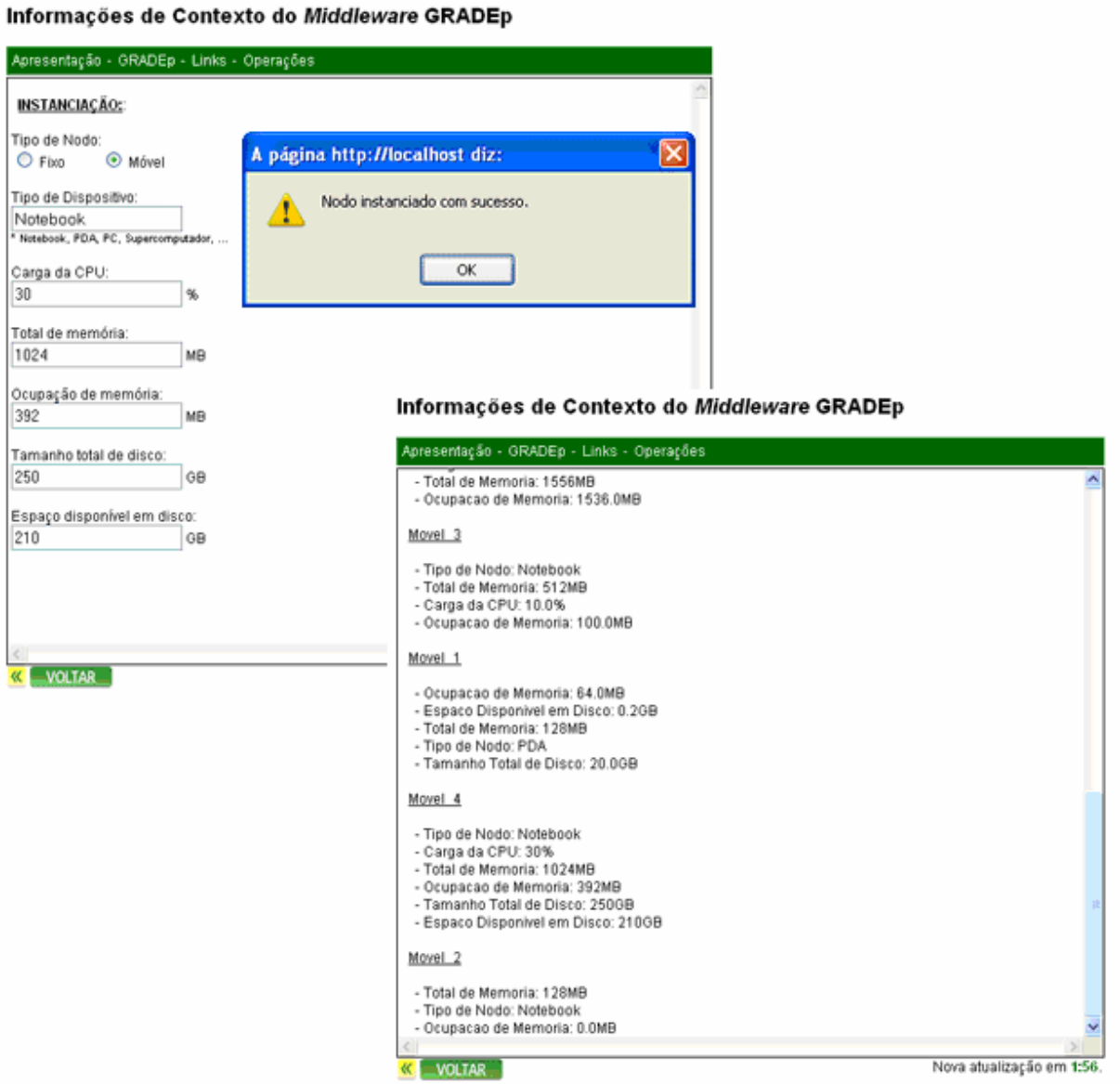

Figura 27 - Teste da instanciação e posterior resultado no monitoramento.

A listagem completa de todos os nodos encontrados no ambiente pervasivo pela operação de monitoramento logo após a instanciação do nodo **Movel_4** encontra-se no Apêndice C.

6 Conclusão e Trabalhos Futuros

Ao longo deste projeto, foram estudadas as áreas de computação pervasiva e também de *Web Services*, com o objetivo de se desenvolver uma forma mais abrangente de disseminação de informação de contexto a usuários finais. Este estudo foi realizado no âmbito do projeto GRADEp, realizando acesso a informações de contexto do *middleware* GRADEp através de um portal *web.*

No desenvolvimento deste trabalho, pode-se perceber que a utilização de *web services* para apresentação e manipulação de dados de contexto foi uma boa escolha, uma vez que os protocolos e tecnologias ligados a *web services* permitem uma boa utilização destes, provendo uma grande quantidade de ferramental de apoio. Além disso, a integração destes com a linguagem PHP 5 é simples, existindo algumas bibliotecas, como a NuSOAP, além de uma classe nativa do PHP 5, que servem para realizar a comunicação através do protocolo SOAP com outras aplicações, estando elas em qualquer linguagem, como por exemplo o Java, utilizado para a construção do WS servidor.

A apresentação dos dados de contexto do *middleware* GRADEp por meio do portal *web* desenvolvido também foi satisfatória, pois pode ocorrer de usuários finais necessitarem de dados de contexto sobre a execução dos equipamentos mesmo não estando conectados fisicamente ao ambiente pervasivo. Isto também pode ocorrer para a consulta de equipamentos específicos ou para a instanciação de novos equipamentos ao ambiente.

Como resultado deste trabalho, obteve-se então um sistema que, embora consista de um protótipo, com poucas adaptações poderá ser utilizado de forma integrada em uma EXEHDA Base. Além disso, agora outras aplicações poderão ser desenvolvidas para, utilizando-se deste WS, ter também agregada às

funcionalidades de monitoramento, consulta e instanciação sobre estas informações de contexto.

Finalmente, acredita-se que os objetivos deste trabalho foram atingidos, uma vez que o WS foi desenvolvido, assim como o portal *web* que se comunica através do WS com o servidor de contexto. Da mesma forma, os resultados obtidos com as operações a partir dos testes realizados no portal retornaram os valores corretos.

Como trabalho futuro, sugere-se a publicação efetiva do servidor, para que o serviço se torne parte de uma EXEHDA Base. Podem ser também adicionados novos serviços ao servidor, como mensagens de erro a cada carga excedente de equipamentos.

Outro trabalho que pode ser desenvolvido é fazer com que as propriedades disponibilizadas nas operações de consulta, instanciação e monitoramento, não sejam fixas, podendo assim serem ajustadas em função da ontologia em questão. Neste caso, as operações não ficariam restritas apenas as propriedades de carga da CPU, espaço total de memória, espaço ocupado de memória, espaço total de disco e espaço disponível de disco, e sim, teriam disponibilizadas dinamicamente todas as propriedades da ontologia considerada.

Referências

AURA. **Project Aura Home Page**. Disponível em http://www.cs.cmu.edu/~aura/. Acessado em agosto de 2008.

BECKER, A. K.; CLARO, D. B. ; SOBRAL, J. B. M. **Web Services e XML: um novo paradigma da Computacao Distribuida.** Objetos Distribuídos, 2001, São Paulo. I Workshop de Tendências em Objetos Distribuídos WTOD'2001, 2001. p. 51-66.

BONA, L. C. E. ; DUARTE JR, E. P. ; FONSECA, K. O. **HyperGrid: Uma Plataforma Distribuída e Confiável para Computação em Grade**. 2004.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M. **Extensible Markup Language (XML) 1.0 W3C Recommendation**. W3C, 1998.

BREITMAN, K. K. **Web semântica: a internet do futuro**. Rio de Janeiro: LTC, 2005.

CERAMI, E,. **Web Services Essentials**. O'Reilly, 2002.

CHINNICI R.; GUDGIN M.; MOREAU, J.; WEERAWARANA, S. **Web Services Description Language (WSDL) version 1.2. Technical report**, W3C, 2003.

COLAN, M. **A Technical Overview of Web Services**. 2002.

COSTA, Cristiano A.; GEYER, Cláudio; YAMIN, Adenauer. **Toward a general Software Infrastructure for Ubiquitous Computing**. In: Pervasive Computing, 2008. p.64–73.

CURBERA, F., KHALAF, R., MUKHI, N., TAI, S., and WEERAWARANA, S. **The next step in web services**. Communications ACM, 46(10):29–34. 2003.

DEY, A. K.; ABOWD, G. D. **Towards a better understanding of context and context-awareness**. Conference on Human Factors in Computing Systems, Netherlands. 2000.

FENSEL, D. **Ontologies: Silver Bullet for Knowledge Management and Eletronic Commerce**, Springer – Verlag, Berlin, 2000.

GAIA**. Web Site do Projeto Gaia**. Disponível em http://gaia.cs.uiuc.edu/. Acessado em agosto de 2008.

GEYER, Cláudio F. R. **Projeto GRADEp: *Middleware* para gerenciar um ambiente de grade pervasiva**. Instituto de Informática/UFRGS, Porto Alegre, agosto de 2004. Disponível em www.rnp.br/_arquivo/gt/2004/Proposta_GT_Grade_Pervasiva.pdf. Acesso em Julho de 2008.

GRUBER, T. **A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition**. 1993.

GRUNINGER, M. **Designing and Evaluating Generic Ontologies**. In: European Conference Of Artificial Intelligence, 12., 1996. Anais, 1996.

HAROLD, E. R.. **XML bible**. IDG Books Worldwide, 1999.

HAROLD, E. R.; MEANS, W. Scott. **XML in a Nutshell**. O'Reilly, 2002.

JONES ,J. B. **A Technical Overview of Web Services**. GSE Nordic Conference, 2003.

KATZ., R. H. **The endeavour expedition: Charting the fluid information utility**. Disponível em http://endeavour.cs.berkeley.edu/proposal/. Acessado em: agosto. 2008.

KREGER H., **Web services Conceptual Architecture (WSCA 1.0)**. 2001.

LOPES, João Ladislau Barbará. **EXEHDA-ON: Uma Abordagem Baseada em Ontologias para Sensibilidade ao Contexto na Computação Pervasiva**. 2008. Dissertação (Mestrado em Ciência da Computação) - Escola de Informática, Universidade Católica de Pelotas.

MARQUEZAN, Clarissa M.; CARISSIMI, Alexandre S. NAVAUX, Philippe O. **Web Services para Computação de Alto Desempenho**. Instituto de Informática - UFRGS, Porto Alegre, 2006.

NEWCOMER, Eric. **Understanding Web Services: XML, WSDL, SOAP and UDDI**. Addison-Wesley, 2002.

NUNES, L. D. **Sistema de consulta de Informações de Contexto para apoio a adaptação de aplicações pervasivas.** Monografia (Conclusão de curso). Universidade Federal de Pelotas. Pelotas, 2008.

OXYGEN. **OXYGEN Project Home Page**. Disponível em: http://oxygen.csail.mit.edu/ Acessado em: agosto. 2008.

PERNAS, Ana M. **Ontologias Aplicadas à Descrição de Recursos em *Grids* Computacionais**. Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2004.

SAHA, D.; MUKHERJEE, A. **Pervasive Computing: a Paradigm for the 21st Century**, IEEE Computer, New York:IEEE Computer Society, v.36, n.3, p.25-31. 2003.

SILVA, Flavio de Oliveira. **Controle de Acesso a Web Services baseado em um protocolo de autenticação segura**. Dissertação (Mestrado em Ciência da Computação). Uberlândia, 2004.

UDDI. **UDDI Technical White Paper**. UDDI, 2006.

VLIST, Eric van der. **XML Schema**. O'Reilly, 2002

WEISER, M. **The Computer for the Twenty-First Century**. Scientific American. 1991.

YAMIN, Adenauer; AUGUSTIN, Iara; BARBOSA, Jorge; SILVA, Luciano da; REAL, Rodrigo; CAVALHEIRO, Gerson; GEYER, Cláudio. **Towards Merging Context-aware, Mobile and Grid Computing**. In: INTERNATIONAL JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS, 2003, Londres. Anais do... 2003. p.191–203.

YAMIN, Adenauer. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. Tese (Doutorado em Ciência da Computação) – UFRGS – 2004.

YAMIN, Adenauer; AUGUSTIN, Iara; SILVA, Luciano; REAL, Rodrigo; BARBOSA, Jorge; GEYER, Cláudio. **ISAM: Uma Arquitetura de Software para Pervasive Computing**. Conferencia Latinoamericano de Informatica. Arequipa, Peru. Anais, 2004, p.347-358.

Apêndices

APÊNDICE A

O presente apêndice apresenta a descrição completa em WSDL do WS desenvolvido contendo as operações de monitoramento, consulta e instanciação.

```
<definitions targetNamespace="http://WS/" name="serverService">
    <types>
        <xsd:schema>
            <xsd:import namespace="http://WS/"
        schemaLocation="http://localhost:8080/WS/serverService?xsd=1"/>
        </xsd:schema>
    </types>
    <message name="consulta">
        <part name="parameters" element="tns:consulta"/>
    </message>
    <message name="consultaResponse">
        <part name="parameters" element="tns:consultaResponse"/>
    </message>
    <message name="FileNotFoundException">
        <part name="fault" element="tns:FileNotFoundException"/>
    </message>
    <message name="monitora">
        <part name="parameters" element="tns:monitora"/>
    </message>
    <message name="monitoraResponse">
        <part name="parameters" element="tns:monitoraResponse"/>
    </message>
    <message name="instancia">
        <part name="parameters" element="tns:instancia"/>
    </message>
    <message name="instanciaResponse">
        <part name="parameters" element="tns:instanciaResponse"/>
    </message>
    <portType name="server">
        <operation name="consulta">
            <input message="tns:consulta"/>
            <output message="tns:consultaResponse"/>
            <fault message="tns:FileNotFoundException"
            name="FileNotFoundException"/>
        </operation>
        <operation name="monitora">
            <input message="tns:monitora"/>
            <output message="tns:monitoraResponse"/>
            <fault message="tns:FileNotFoundException"
            name="FileNotFoundException"/>
        </operation>
        <operation name="instancia">
            <input message="tns:instancia"/>
            <output message="tns:instanciaResponse"/>
            <fault message="tns:FileNotFoundException"
```

```
                name="FileNotFoundException"/>
            </operation>
        </portType>
        <binding name="serverPortBinding" type="tns:server">
                <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document"/>
            <operation name="consulta">
                <soap:operation soapAction=""/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
                <fault name="FileNotFoundException">
                    <soap:fault name="FileNotFoundException" use="literal"/>
                </fault>
            </operation>
            <operation name="monitora">
                <soap:operation soapAction=""/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
                <fault name="FileNotFoundException">
                    <soap:fault name="FileNotFoundException" use="literal"/>
                </fault>
            </operation>
            <operation name="instancia">
                <soap:operation soapAction=""/>
                <input>
                    <soap:body use="literal"/>
                </input>
                <output>
                    <soap:body use="literal"/>
                </output>
                <fault name="FileNotFoundException">
                    <soap:fault name="FileNotFoundException" use="literal"/>
                </fault>
            </operation>
        </binding>
        <service name="serverService">
            <port name="serverPort" binding="tns:serverPortBinding">
                <soap:address
                location="http://localhost:8080/WS/serverService"/>
            </port>
        </service>
</definitions>
```

APÊNDICE B

O presente apêndice contém a listagem completa dos nodos presentes no ambiente pervasivo resultante da operação de monitoramento.

Fixo_5

- Ocupacao de Memoria: 256.0MB
- Total de Memoria: 4096MB
- Carga da CPU: 20.0%
- Tipo de Nodo: Supercomputador

Fixo_1

- Ocupacao de Memoria: 768.0MB
- Total de Memoria: 1024MB
- Tamanho Total de Disco: 80.0GB
- Espaco Disponivel em Disco: 40.0GB
- Tipo de Nodo: PC
- Carga da CPU: 50.0%

Fixo_2

- Total de Memoria: 2014MB
- Ocupacao de Memoria: 1536.0MB
- Tamanho Total de Disco: 60.0GB
- Espaco Disponivel em Disco: 10.0GB
- Tipo de Nodo: PC
- Carga da CPU: 90.0%

Fixo_4

- Ocupacao de Memoria: 128.0MB
- Espaco Disponivel em Disco: 0.5GB
- Tipo de Nodo: PC
- Total de Memoria: 512MB
- Tamanho Total de Disco: 100.0GB
- Carga da CPU: 3.0%

Fixo_3

- Ocupacao de Memoria: 1536.0MB
- Total de Memoria: 1556MB
- Carga da CPU: 100.0%
- Tipo de Nodo: Servidor
- Tamanho Total de Disco: 80.0GB

Movel_3

- Ocupacao de Memoria: 100.0MB
- Carga da CPU: 10.0%
- Total de Memoria: 512MB
- Tipo de Nodo: Notebook

Movel_1

- Tamanho Total de Disco: 20.0GB
- Tipo de Nodo: PDA
- Total de Memoria: 128MB
- Espaco Disponivel em Disco: 0.2GB
- Ocupacao de Memoria: 64.0MB

Movel_2

- Ocupacao de Memoria: 0.0MB
- Tipo de Nodo: Notebook
- Total de Memoria: 128MB

APÊNDICE C

O presente apêndice contém a listagem completa dos nodos presentes no ambiente pervasivo resultante da operação de monitoramento após a operação de instanciação adicionar o nodo Movel_4.

Fixo_5

- Tipo de Nodo: Supercomputador
- Carga da CPU: 20.0%
- Total de Memoria: 4096MB
- Ocupacao de Memoria: 256.0MB

Fixo_1

- Ocupacao de Memoria: 768.0MB
- Total de Memoria: 1024MB
- Tamanho Total de Disco: 80.0GB
- Espaco Disponivel em Disco: 40.0GB
- Tipo de Nodo: PC
- Carga da CPU: 50.0%

Fixo_2

- Total de Memoria: 2014MB
- Ocupacao de Memoria: 1536.0MB
- Tamanho Total de Disco: 60.0GB
- Espaco Disponivel em Disco: 10.0GB
- Tipo de Nodo: PC
- Carga da CPU: 90.0%

Fixo_4

- Carga da CPU: 3.0%
- Tamanho Total de Disco: 100.0GB
- Total de Memoria: 512MB
- Tipo de Nodo: PC
- Espaco Disponivel em Disco: 0.5GB
- Ocupacao de Memoria: 128.0MB

Fixo_3

- Tamanho Total de Disco: 80.0GB
- Tipo de Nodo: Servidor
- Carga da CPU: 100.0%
- Total de Memoria: 1556MB
- Ocupacao de Memoria: 1536.0MB

Movel_3

- Tipo de Nodo: Notebook
- Total de Memoria: 512MB
- Carga da CPU: 10.0%
- Ocupacao de Memoria: 100.0MB

Movel_1

- Ocupacao de Memoria: 64.0MB
- Espaco Disponivel em Disco: 0.2GB
- Total de Memoria: 128MB
- Tipo de Nodo: PDA
- Tamanho Total de Disco: 20.0GB

Movel_4

- Tipo de Nodo: Notebook
- Carga da CPU: 30%
- Total de Memoria: 1024MB
- Ocupacao de Memoria: 392MB
- Tamanho Total de Disco: 250GB
- Espaco Disponivel em Disco: 210GB

Movel_2

- Total de Memoria: 128MB
- Tipo de Nodo: Notebook
- Ocupacao de Memoria: 0.0MB