UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Uma Contribuição ao Emprego de Ontologias no Sensoriamento de Contexto na Computação Ubíqua

por
Anderson Medeiros Gomes

Projeto de Graduação submetido como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Adenauer Yamin
Co-orientador: Luthiano Venecian (PPGINF/UCPEL)

Pelotas, dezembro de 2009

SUMÁRIO

LISTA DE FIGURAS 5

LISTA DE TABELAS 6

LISTA DE ABREVIATURAS E SIGLAS 7

RESUMO 9

ABSTRACT 10

1 INTRODUÇÃO 11
1.1 Objetivos 12
1.2 Motivação 13
1.3 Estrutura do Texto 13

2 COMPUTAÇÃO UBÍQUA REVISITADA 14
2.1 Definições 14
2.2 Computação Móvel e Computação Pervasiva 15
2.3 Características de um cenário ubíquo 16
2.4 Aplicações da Computação Ubíqua 17
2.4.1 O vôo de Luciana 17
2.4.2 O geólogo 17
2.5 Considerações sobre o capítulo 18

3 SENSIBILIDADE AO CONTEXTO: PRINCIPAIS CONCEITOS 19
3.1 Definição de contexto 20
3.2 Adaptação e sensibilidade ao contexto 21
3.3 Desafios na concepção do suporte a sensibilidade ao contexto 22
3.3.1 Identificação dos elementos de contexto 23
3.3.2 Características das informações contextuais 24

3.3.3 Dimensões de informação de contexto 26
3.3.4 Aquisição de contexto 28
3.3.5 Modelagem de contexto 29
3.3.6 Interpretação de contexto 31
3.3.7 Processamento e raciocínio sobre o contexto 31
3.3.8 Armazenamento de informações contextuais 32
3.4 Considerações sobre o capítulo 33

4 ONTOLOGIAS: FUNDAMENTOS E TECNOLOGIAS UTILIZADAS . . 34
4.1 Linguagens para a representação de ontologias 35
4.1.1 XML - Extensible Markup Language 36
4.1.2 RDF e RDF Schema 37
4.1.3 OWL - Web Ontology Language 38
4.2 Tecnologias para a manipulação de ontologias 39
4.2.1 Protégé 39
4.2.2 Jena API 39
4.2.3 SPARQL 41
4.3 Considerações sobre o capítulo 41

5 TRABALHOS RELACIONADOS A SENSIBILIDADE AO CONTEXTO 43
5.1 SOCAM 43
5.2 Flame2008 44
5.3 JCAF 44
5.4 LOTUS 45
5.5 Considerações sobre o capítulo 46

6 MIDDLEWARE EXEHDA E SUAS FUNCIONALIDADES 47
6.1 Organização do EXEHDA 49
6.2 O núcleo do EXEHDA 51
6.3 Serviços do ambiente EXEHDA 51
6.3.1 Subsistema de Execução Distribuída 51
6.3.2 Subsistema de Reconhecimento de Contexto e Adaptação 53
6.3.3 Subsistema de Comunicação 54
6.3.4 Subsistema de Acesso Pervasivo 55
6.4 Considerações sobre o capítulo 55

7 EXEHDA-SS: MODELAGEM E IMPLEMENTAÇÃO 57
7.1 Módulo de sensoriamento 57
7.1.1 Sensor de processamento 58

7.1.2 Sensor de memória 58
7.1.3 Sensor de temperatura 59
7.1.4 Sensor de uso de espaço em disco 59
7.2 Módulo de sensibilidade ao contexto 59
7.2.1 Arquitetura básica 59
7.2.2 Implementação 61
7.2.3 Representação das informações contextuais do ambiente ubíquo 62
7.2.4 XML de descrição de nodos 63
7.3 Fluxo de execução do EXEHDA-SS 67
7.4 Considerações sobre o capítulo 68

8 CONSIDERAÇÕES FINAIS 69
8.1 Principais conclusões 69
8.2 Trabalhos futuros 70

REFERÊNCIAS 71

LISTA DE FIGURAS

Figura 2.1 Relação entre Computação Pervasiva, Ubíqua e Móvel 15

Figura 4.1 Tela do ambiente Protégé, exibindo a hierarquia de classes de uma ontologia 40
Figura 4.2 Exemplo de consulta escrita na linguagem SPARQL 41

Figura 6.1 Organização do ambiente ubíquo provido pelo EXEHDA (YAMIN, 2004) 50
Figura 6.2 Organização do núcleo do EXEHDA (YAMIN, 2004) 52
Figura 6.3 Serviços do *middleware* EXEHDA (YAMIN, 2004) 52

Figura 7.1 Visão geral da arquitetura do módulo de sensibilidade do EXEHDA-SS 60
Figura 7.2 Diagrama de classes e relacionamentos da OntUbi 64
Figura 7.3 Diagrama de classes da OntContext 65
Figura 7.4 Arquivo XML de descrição de nodos 66

LISTA DE TABELAS

Tabela 3.1 Avaliação das abordagens para Modelagem de Contexto 31

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AVU	Ambiente Virtual do Usuário
BDA	Base de Dados Pervasiva das Aplicações
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
EUA	Estados Unidos da América
EXEHDA	Execution Environment for Highly Distributed Applications
G3PD	Grupo de Pesquisa em Processamento Paralelo e Distribuído
GNU	GNU’s not Unix!
GPS	Global Positioning System
HP	Hewlett-Packard
HUD	Head-up Display
IBM	International Business Machines Corporation
JCAF	Java Context Awareness Framework
MG	Minas Gerais
ORM	Object-relational mapping
OWL	Web Ontology Language
OX	Objeto EXEHDA
PC	Personal Computer
RDF	Resource Description Framework
RDFS	RDF Schema

RMI	Remote Method Invocation
RPC	Remote Procedure Call
SGML	Standard Generalized Markup Language
SOCAM	Service-Oriented Context-Aware Middleware
SPARQL	SPARQL Protocol and RDF Query Language
TCP	Transmission Control Protocol
UCPEL	Universidade Católica de Pelotas
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

RESUMO

A Computação Ubíqua é o terceiro grande paradigma computacional, precedido pelo império dos *mainframes* e pela onda da computação pessoal. Nesse paradigma, os sistemas computacionais estarão presentes em todos os lugares e a todo momento, integrados ao ambiente do usuário e interagindo naturalmente com estes. Porém, infra-estrutura para o desenvolvimento e mecanismos de auxílio em tempo de execução são, neste momento, focos a serem pesquisados dentro da área da Computação Ubíqua. Soluções genéricas para o desenvolvimento e uso de aplicações ubíquas ainda não são disponíveis.

As aplicações ubíquas, em primeiro lugar, deverão ser sensíveis ao contexto em que se situam e adaptar-se a ele automaticamente. Diante dessa premissa, surge a necessidade de concentrar esforços na área de sensibilidade e adaptação ao contexto.

O EXEHDA é um *middleware* direcionado às aplicações distribuídas, móveis e conscientes do contexto da Computação Ubíqua. Este *middleware* é baseado em serviços, que tem por objetivo criar e gerenciar um ambiente ubíquo. O *middleware* vem sendo desenvolvido na UCPEL dentro do Grupo de Pesquisa em Processamento Paralelo e Distribuído (G3PD), em conjunto com outras universidades gaúchas.

Este documento registra de forma resumida o trabalho desenvolvido nesse Projeto de Graduação, que teve por foco central participar na modelagem e implementação de um *framework* de sensibilidade ao contexto para utilização no *middleware* EXEHDA.

Palavras-chave: Computação Ubíqua, Sensibilidade ao Contexto, Ontologia, EXEHDA.

TITLE: “A CONTRIBUTION TO ADAPTATION CONTROL IN UBIQUITOUS COMPUTING”

ABSTRACT

Ubiquitous Computing is the third major computing paradigm, preceded by the empire of mainframes and the wave of personal computing. In this paradigm, computer systems will be present in all places and all times, integrated in user’s environment and interacting with them naturally. However, now, development infrastructure and runtime aid mechanisms are research topics in Ubiquitous Computing. Generic solutions for ubiquitous applications developing and using are not yet available.

Ubiquitous applications must be context-aware and self-adapting. So, efforts in context sensitivity and context adaptation must be massive.

EXEHDA is an execution environment for highly distributed, mobile and context-aware applications of ubiquitous computing. EXEHDA is a service-based middleware, which aims to create and manage a ubiquitous environment. The middleware is being developed in UCPEL, within *Grupo de Pesquisa em Processamento Paralelo e Distribuído*, together with other universities in *Rio Grande do Sul*, Brazil.

This paper presents the work which was done during this Graduation Project. Its central aim was to collaborate in design and development of a context-sensitivity module for EXEHDA.

Keywords: ubiquitous computing, context-sensitivity, ontology, EXEHDA.

1 INTRODUÇÃO

Este capítulo objetiva fazer uma apresentação sobre os conceitos presentes nesta monografia, bem como as motivações que provocaram a necessidade de estudá-los durante o período de Projeto de Graduação.

As tecnologias mais profundas são aquelas que desaparecem, elas se integram na vida cotidiana até se tornarem indistinguíveis da mesma (WEISER, 1991). Esta frase, extraída do clássico artigo de Mark Weiser sobre a computação para o século 21, sintetiza um pouco do que é esperado com a Computação Ubíqua. O termo trata do acesso ao ambiente computacional do usuário em qualquer lugar, todo o tempo, com qualquer dispositivo. A computação e seus diversos sistemas interagem de maneira natural com o ser humano a todo o momento, não importando onde ele esteja, em casa, no trabalho e na rua, constituindo um ambiente altamente distribuído, heterogêneo, dinâmico, móvel, mutável e com forte interação entre homem e máquina (AUGUSTIN, 2004).

Contexto é o conjunto de quaisquer informações que podem ser usadas para caracterizar a situação de uma entidade. Uma entidade é qualquer pessoa, lugar ou objeto que é considerado relevante para a interação entre usuário e a aplicação, incluindo o próprio usuário e a própria aplicação (DEY; ABOWD; SALBER, 2001).

As aplicações ubíquas precisam compreender o contexto em que estão inseridas e se adaptar a ele (MACIEL; ASSIS, 2004). Essa nova classe de sistemas computacionais, adaptativos ao contexto, abre perspectivas para o desenvolvimento de aplicações mais ricas, elaboradas e complexas, que exploram a natureza dinâmica e a mobilidade do usuário. Entretanto, o desenvolvimento de aplicações que se adaptem continuamente ao ambiente e permaneçam funcionando, mesmo quando o indivíduo se movimentar ou trocar de dispositivo, continua um desafio de pesquisa em aberto (GRIMM; BERSHAD, 2003). Ainda existem limitações para o desenvolvimento de tais softwares, pois poucas linguagens e ferramentas estão disponíveis para a programação de aplicações adaptáveis às mudanças de contexto (WANT; PERING, 2005).

O contexto pode ser representado nas aplicações com o uso de ontologias. Ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um

domínio e os relacionamentos entre estes. Ontologias são utilizadas em Inteligência Artificial, Web Semântica e em diversas outras áreas da Ciência da Computação, como uma forma de representação de conhecimento sobre o mundo ou alguma parte deste.

O EXEHDA, de *Execution Environment for Highly Distributed Applications*, é um *middleware* direcionado às aplicações distribuídas, móveis e conscientes do contexto da Computação Ubíqua (YAMIN, 2004; YAMIN et al., 2005). Este *middleware* é baseado em serviços, que tem por objetivo criar e gerenciar um ambiente ubíquo. As aplicações deste ambiente são distribuídas, móveis e adaptativas ao contexto.

Por sua vez, o EXEHDA-SS é um *framework* direcionado ao *middleware* EXEHDA para suporte à sensibilidade ao contexto na Computação Ubíqua, em tempo de execução. É um componente que realiza tarefas relacionadas a captura, processamento e notificação, contemplando o emprego de ontologia na representação de elementos internos de trabalho. Essas tarefas compreendem aquisição de contexto, representação de informações contextuais, tradução, raciocínio e inferência sobre as informações, persistência de dados e remessa de notificações.

1.1 Objetivos

O objetivo geral deste projeto de graduação foi contribuir com a modelagem e a implementação de um *framework* direcionado ao *middleware* EXEHDA para suporte à sensibilidade ao contexto na Computação Ubíqua, em tempo de execução.

Como objetivos específicos, destacaram-se:

- Mapear as características desejáveis em um *framework* para sensibilidade ao contexto, considerando o cenário da Computação Ubíqua;

- Conceber as funcionalidades do *framework*, procurando contemporizar flexibilidade e simplicidade de uso;

- Estreitar a compatibilidade com o projeto EXEHDA, procurando explorar, deste modo, as funcionalidades que o mesmo disponibiliza no que diz respeito a Computação Ubíqua;

- Perseguir compatibilidade também com os padrões de mercado, particularmente com a plataforma Java, o que potencializou a interoperabilidade da solução proposta.

1.2 Motivação

A Computação Ubíqua é um novo paradigma e uma tendência mundial, em função do rápido avanço tecnológico dos dispositivos móveis, redes sem fio, grades computacionais, *Web Services*, *Web* Semântica, Sistemas Autônomos e outras tecnologias relacionadas a integração de sistemas. Na Computação Ubíqua, a sensibilidade ao contexto é um componente importante e fundamental, porque aplicações que se adaptam ao espaço ubíquo atendem as necessidades dos usuários com maior facilidade, muitas vezes até, sem que ele mesmo tenha que fazer a solicitação, proporcionando, assim, uma melhora em sua usabilidade.

Não existem linguagens de programação para aplicações que se adaptem ao espaço ubíquo, nem modelos para desenvolvimento, de maneira genérica, de aplicações sensíveis ao contexto. Prover soluções adaptáveis a esses contextos tão diversos, que facilitem o desenvolvimento de aplicações baseadas no reuso, constitui um grande desafio para a Engenharia de Software (WARKEN, 2008).

1.3 Estrutura do Texto

O texto é composto por oito capítulos.

No capítulo 1, encontra-se a introdução, onde é mostrado o tema do trabalho, a motivação para escolha do tema e os objetivos perseguidos no andamento deste trabalho.

No capítulo 2, é descrita a Computação Ubíqua; suas características e conceitos relacionados ao tema.

No capítulo 3, são descritos alguns dos problemas, desafios e características da sensibilidade ao contexto na Computação Ubíqua.

No capítulo 4, são apresentados os fundamentos teóricos sobre ontologias com seus principais conceitos, categorias e as tecnologias e linguagens para tratamento de ontologias.

No capítulo 5, são apresentados alguns dos projetos em andamento que são relacionados à área de Computação Ubíqua.

No capítulo 6, é apresentado o *middleware* EXEHDA; suas características e os serviços que o compõem.

No capítulo 7, é detalhado o EXEHDA-SS, *framework* de sensibilidade ao contexto que foi desenvolvido durante o Projeto de Graduação.

No capítulo 8, são feitas considerações a respeito do trabalho.

2 COMPUTAÇÃO UBÍQUA REVISITADA

Este capítulo tem o objetivo de caracterizar a área da Computação Ubíqua e, também, discorrer sobre os principais conceitos associados.

2.1 Definições

A Computação Ubíqua, também conhecida como Computação Invisível, é definida como sendo o terceiro grande paradigma computacional, precedido pelo império dos *mainframes* e pela onda da computação pessoal (WARKEN, 2008). Nesse paradigma, a computação é onipresente e imperceptível; sistemas computacionais estarão presentes em todos os lugares e a todo momento, integrados ao ambiente do usuário e interagindo naturalmente com estes.

A Computação Ubíqua surge dos avanços da Computação Móvel e da Computação Pervasiva, reunindo a necessidade de se integrar a mobilidade e a adaptabilidade nos serviços computacionais. Dessa forma, os serviços passam a suportar o estilo *siga-me*, dando ao usuário a facilidade de acessar seu ambiente computacional independente da localização, do tempo e do dispositivo utilizado (ARAUJO, 2003).

O termo Computação Ubíqua foi definido pela primeira vez pelo cientista Mark Weiser que, no início dos anos 90, já previa um aumento nas funcionalidades e na disponibilidade de serviços de computação para os usuários finais, com a menor visibilidade possível. A computação não seria exclusividade de um computador, mas de diversos dispositivos conectados entre si (WARKEN, 2008).

A Computação Ubíqua não significa um computador que possa ser transportado para a praia, o campo ou o aeroporto. Mesmo o mais poderoso notebook com acesso a Internet ainda foca a atenção do usuário numa simples caixa. Comparando à escrita, carregar um super-notebook é como carregar um livro muito importante: personalizar este livro, mesmo escrevendo milhões de outros livros, não significa capturar o real poder da Literatura (DOMINGUES, 2008).

Hoje, por exemplo, os motores elétricos participam das vidas das pessoas de forma

onipresente - eles estão instalados nas portas, nos ventiladores, nos computadores, nos telefones celulares, nos automóveis, no chão das esteiras rolantes, etc. A Computação Ubíqua atingirá a sua plenitude quando, da mesma forma que os motores, os sistemas computacionais estiverem inseridos no cotidiano das pessoas sem que estas necessitem ter ciência da sua existência.

2.2 Computação Móvel e Computação Pervasiva

A Computação Móvel é a capacidade de mover serviços computacionais sempre que preciso e mantê-los conectados à rede ou a Internet, ou seja, o computador torna-se um dispositivo sempre presente que expande a capacidade de um usuário móvel utilizar os seus serviços (ARAUJO, 2003; WARKEN, 2008). Uma importante limitação da Computação Móvel é que o modelo computacional não muda durante a moção, isto é, o dispositivo não é capaz de obter flexivelmente informações sobre o contexto no qual a computação ocorre e ajustá-la corretamente. Numa solução para acomodar a mudança de ambiente, os usuários poderiam manualmente controlar e configurar a aplicação à medida que se movem, mas isso seria inviável e inaceitável pela maioria dos usuários (ARAUJO, 2003).

Na Computação Pervasiva, os sistemas computacionais estão embarcados no ambiente de forma invisível para o usuário. Nesta concepção, computadores têm a capacidade de obter informações do ambiente no qual ele está inserido e utilizá-las para dinamicamente construir modelos computacionais, ou seja, controlar, configurar e ajustar as aplicações para melhor atender as necessidades do usuário (ARAUJO, 2003).

É preciso observar que em um ambiente pervasivo, nem todos dispositivos são móveis. Por outro lado, na Computação Móvel, nem todos os dispositivos assumem a característica de serem pervasivos. Diante disso, define-se a Computação Ubíqua como uma interseção das áreas de Computação Móvel e Computação Pervasiva, aliando o alto grau de mobilidade da primeira com o alto grau de dispositivos embarcados da segunda, conforme exposto na Figura 2.1 (ARAUJO, 2003).

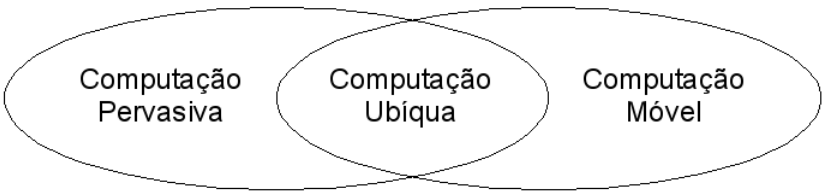

Figura 2.1: Relação entre Computação Pervasiva, Ubíqua e Móvel

2.3 Características de um cenário ubíquo

As características descritas nesta seção estão presentes em um cenário típico de Computação Ubíqua (ARAUJO, 2003; WARKEN, 2008).

Invisibilidade

Quanto mais presente uma tecnologia estiver, menos perceptível ela deve ser. O computador torna-se fundamental nas atividades cotidianas, mas dilui-se no mundo físico, tornando-se onipresente e imperceptível. A tecnologia não deve exigir mais que a atenção periférica dos usuários.

Pró-atividade

O sistema ubíquo deve ser capaz de antecipar a intenção do usuário, por exemplo, como visto na seção 2.4.1.

Sensibilidade ao Contexto

Para que um sistema ubíquo tenha o mínimo de intrusão no mundo real, sua infra-estrutura de software deve ser adaptativa ao contexto, ou seja, deve possuir uma base extensa de informações a respeito dos usuários e do ambiente, e, com isso, otimizar o sistema para a satisfação desses usuários.

Interfaces naturais

Uma das propostas da Computação Ubíqua é a de celebrar a comunicação natural entre pessoas e sistemas computacionais. Diante disso, surge a necessidade de se buscar técnicas para que os recursos de comunicação utilizados no dia a dia de uma sociedade, como gestos, voz e mesmo olhares, continuem sendo utilizados na interação entre o homem e a máquina.

Descentralização

O computador pessoal é um dispositivo de propósito geral que atende a muitas necessidades dos usuários, tais como: edição de texto, contabilidade, navegação na web, produção de apresentações, entretenimento, etc. Nos dispositivos ubíquos, entretanto, a limitação de espaço físico impõe a limitação de recursos computacionais que, por sua vez, produz o objetivo de focalizar poucas necessidades específicas. Em um cenário ubíquo, as necessidades gerais passam a ser supridas através da colaboração mútua entre as várias entidades computacionais.

2.4 Aplicações da Computação Ubíqua

As subseções a seguir discutem cenários de aplicação da Computação Ubíqua, com o intuito de caracterizar a sua aplicabilidade.

2.4.1 O vôo de Luciana

Luciana está no portão 5 do Aeroporto Internacional de Brasília a espera de seu vôo para Patos de Minas, MG. Neste período, pelo *notebook*, ela revisa sua dissertação de mestrado, pois necessita enviá-la por e-mail, através da rede *wireless* do aeroporto, ao seu orientador para correções. Entretanto, devido à limitação da largura de banda disponível naquele instante, causada pela alta taxa de utilização da rede no perímetro do portão 5, o envio imediato do documento de Luciana, que contém vários megabytes de texto, imagens e planilhas, é inviabilizado pela iminência de seu embarque. Em um ambiente ubíquo, seria possível verificar a impossibilidade de transferência completa do documento de Luciana antes da partida de seu vôo e, através de observações na programação de decolagens do aeroporto e avaliações climáticas e de tráfego aéreo para previsão de atrasos, orientar a usuária a dirigir-se ao portão mais próximo cujo fluxo na rede não impeça a total conclusão de suas atividades. Esta pró-atividade do sistema agrega conforto e segurança a usuária, na medida em que garante o cumprimento de seus compromissos e proporciona-lhe maior comodidade (WARKEN, 2008).

2.4.2 O geólogo

Um geólogo do Instituto Americano de Geologia foi mandado para um local remoto no oeste dos EUA para examinar os efeitos de um terremoto recente. Usando um PC e um sistema denominado neste exemplo por MANNA, que suporta as atividades do geólogo em qualquer dispositivo que ele usar, o geólogo baixa mapas e relatórios existentes sobre a área de modo a se preparar para a visita. Como o PC não é um equipamento móvel e o geólogo precisa viajar, os documentos são transferidos para um *laptop* e o geólogo pega um avião até o local onde ocorreu o terremoto.

Nesse avião, não há suporte para rede, portanto o *laptop* desabilita a conexão de rede e oferece apenas computação local. Quando o geólogo examina os vídeos do local, a interface do usuário chaveia automaticamente para monitor branco e preto e reduz a taxa de quadros por segundo, para ajudar a conservar a energia da bateria. Ao chegar ao aeroporto, o geólogo aluga um carro e dirige até o local. Ele recebe uma mensagem através do sistema MANNA pelo celular, alertando-o para examinar um local em particular. Como o fone celular oferece espaço de tela extremamente limitado, o mapa da região não é mostrado. Em vez disso, o celular mostra a localização geográfica, orientações de como chegar lá e a posição atual do geólogo, via GPS. Um recurso que permite ao geólogo

responder a mensagem também é oferecido pelo software.

Chegando no local, o geólogo usa o seu *palmtop* para tomar notas sobre a região. Como o *palmtop* conta com uma caneta de toque (*Stylus*) para interação, não é permitida a interação através de cliques duplos ou cliques específicos para o lançamento de menus de contexto. Mais ainda, como o tamanho da tela também é um problema no *palmtop*, um layout mais conservador é adotado para ser mostrado ao geólogo. Ao completar a investigação, o geólogo prepara uma apresentação em dois formatos. Primeiro uma apresentação do seu percurso pelo local, com anotações, é feita para o HUD. Como o HUD tem capacidade limitada para tratar entradas de texto, a aplicação MANNA oferece interação baseada em voz. Uma outra apresentação, mais convencional, é preparada para ser visualizada em telão. Como o propósito da apresentação no telão não é o de interação, mecanismos de interação são removidos (ARAUJO, 2003).

2.5 Considerações sobre o capítulo

Este capítulo caracterizou a área da Computação Ubíqua, enumerou alguns dos conceitos existentes dessa área e exemplificou casos de utilização.

A Computação Ubíqua reúne uma fração dos esforços nas áreas de Computação Pervasiva e Computação Móvel, aliando onipresença, invisibilidade e alta mobilidade. Um cenário típico de aplicação da Computação Ubíqua é caracterizado pela invisibilidade, pró-atividade, sensibilidade ao contexto, predominância de interfaces homem-máquina naturais e descentralização de tarefas.

O capítulo seguinte introduz o tema de Sensibilidade ao Contexto, foco do trabalho a ser implementado no Projeto de Graduação.

3 SENSIBILIDADE AO CONTEXTO: PRINCIPAIS CONCEITOS

Este capítulo resume o estudo realizado durante o desenvolvimento desta monografia, referente à área de conhecimento nuclear ao trabalho, que é sensibilidade ao contexto.

Nas mais diversas situações do dia a dia, as pessoas fazem uso do conhecimento do contexto para delimitar e direcionar ações e comportamentos. As mensagens trocadas para comunicação trazem junto um contexto associado, que apóia a compreensão do seu conteúdo. O contexto ajuda a melhorar a qualidade de conversação e a compreender certas situações, ações ou eventos.

O contexto desempenha um papel importante em qualquer domínio que envolva requisitos como compreensão, raciocínio, resolução de problemas ou aprendizado. Contexto é uma importante ferramenta para apoiar a comunicação entre pessoas e sistemas computacionais, porque ajuda a diminuir ambigüidade e conflitos, aumenta a expressividade dos diálogos e possibilita a melhoria dos serviços e informações oferecidos pela aplicação. A *consciência do contexto* faz com que as aplicações se tornem mais amigáveis, flexíveis e fáceis de usar.

O reconhecimento da importância do contexto motivou pesquisadores de diversas áreas da computação, como Inteligência Artificial, Interface Homem-Máquina, Computação Ubíqua, Engenharia de Software, Banco de dados e Sistemas Colaborativos, a estudar esse conceito e entender como o mesmo pode ser formalizado e utilizado nos sistemas computacionais (VENECIAN, 2009).

A Computação Sensível ao Contexto investiga o emprego de informações que caracterizam a situação de uma interação usuário-computador no sentido de fornecer serviços adaptados a usuários e aplicações.

3.1 Definição de contexto

A palavra *contexto* no dicionário Houaiss significa a *inter-relação de circunstâncias que acompanham um fato ou uma situação*. Por mais que essa definição forneça uma noção genérica do significado de contexto, ela não mostra de que maneira esse conceito está relacionado com ambientes computacionais e sistemas de Tecnologia da Informação. A abrangência desse conceito leva a entender que, intuitivamente, contexto pode ser entendido como tudo que está ao redor de um sistema em questão, tudo que ocorre em um determinado ambiente (VENECIAN, 2009).

Alguns pesquisadores, com o intuito de limitar a abrangência desse conceito, elaboraram definições diferentes. Schilit (SCHILIT, 1995) (SCHILIT; ADAMS; WANT, 1994) divide contexto em três categorias:

- Contexto Computacional: conectividade de rede, custos de comunicação, largura de banda e recursos disponíveis, como impressoras, processadores e memória;
- Contexto do Usuário: perfil do usuário, localização, pessoas próximas a ele, humor e outros;
- Contexto Físico: luminosidade, níveis de barulhos, condições do trânsito, temperatura e outros.

Além disso, (CHEN; KOTZ, 2002) defende a inclusão do tempo (tais como hora do dia, da semana, do mês e a estação do ano) como uma quarta categoria de contexto e introduz o conceito de *Histórico de Contexto* e a necessidade de armazenamento de informações contextuais como fonte de tomada de decisões e construção de aplicações sensíveis ao contexto.

A definição mais referenciada na literatura de Computação Ubíqua para contexto é a defendida por (DEY, 2000):

"*Contexto é qualquer informação que pode ser usada para caracterizar uma situação de uma entidade. Uma entidade é uma pessoa, um lugar, ou um objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação.*"

(DEY, 2000) destaca que os contextos mais relevantes para um ambiente computacional são a localização, a identidade, o tempo e a atividade de uma entidade, ou seja, a enumeração de exemplos de contexto ainda é bastante usada na literatura. Considerando a importância do ambiente ao seu redor e o quanto ele determina o comportamento de uma aplicação sensível ao contexto, (CHEN; KOTZ, 2002) define o contexto da seguinte maneira:

"*Contexto é o conjunto de estados e características de um ambiente que determina o comportamento de uma aplicação ou no qual um evento de uma aplicação ocorre e*

interessa ao usuário."

Pode-se ainda entender o termo *contexto* como *circunstâncias ou situações em que uma tarefa computacional está inserida* (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2002).

Freqüentemente usadas como sinônimo de contexto, as *informações contextuais* são nada mais que as informações que caracterizam um determinado contexto. São elas as informações relevantes para se determinar o estado atual de um contexto em questão. Considerando o contexto *localização*, por exemplo, as informações contextuais referentes a esse contexto poderiam ser a latitude e a longitude de uma entidade ou em que sala de um edifício se encontra uma determinada pessoa (VENECIAN, 2009).

3.2 Adaptação e sensibilidade ao contexto

Para que um sistema ubíquo tenha o mínimo de intrusão no mundo real e, ao mesmo tempo, satisfaça as expectativas dos usuários, sua infra-estrutura de software deve ser sensível ao contexto. As aplicações devem possuir uma base de informações extensa a respeito do ambiente e, com isso, adaptarem seu comportamento da melhor forma possível. Este processo requer a existência de múltiplos caminhos de execução ou de configurações alternativas, com diferentes perfis de utilização dos recursos computacionais (YAMIN et al., 2005).

As modificações no sistema ubíquo ocorrem no comportamento, na estrutura ou na interface das aplicações. Tais modificações são governadas por políticas de adaptação gerenciadas pelo ambiente de execução, tendo, opcionalmente, cooperação com as aplicações. Os tipos de alterações, provocadas em um sistema ubíquo em resposta às mudanças ocorridas no contexto, dependem da natureza das aplicações e dos recursos exigidos por elas (WARKEN, 2008). Dentre as transformações produzidas, citam-se, por exemplo:

1. Variação de filtros (compressão, omissão, conversão de formato) inseridos entre o servidor e o cliente;
2. Alteração da fidelidade de saída;
3. Alterações nas dimensões da interface;
4. Migração de componentes para máquinas mais adequadas (com maior poder computacional, com maior memória, com rede de comunicação mais veloz, etc.);
5. Funcionalidade conectada em face à desconexão utilizando *buffering* e *prefetching*.

A adaptação ao contexto é um processo que abrange as seguintes etapas (WARKEN, 2008):

- Detecção das alterações: engloba monitoração, interpretação e notificação; visa observar o ambiente para detectar e notificar alterações significativas. Este subprocesso pode ser realizado de duas formas:
  - *Pull*: a informação é solicitada por uma requisição, o cliente busca a informação sempre que é preciso;
  - *Push*: a informação é enviada ao cliente que se inscreveu no serviço, sem que ele a tenha explicitamente requisitado. A informação é entregue ao cliente sem que este tenha controle de quando isto ocorre;
- Escolha da ação: engloba a seleção da ação adaptativa entre alternativas pré-definidas pelo programador. A seleção pode ser realizada periodicamente, ou quando ocorre o evento de notificação de alteração;
- Ativação da ação: refere-se à execução da ação adaptativa selecionada.

O processo de detecção das alterações, isto é, o suporte a sensibilidade ao contexto, é uma etapa que traz aos desenvolvedores de aplicações ubíquas uma série de questões e desafios para solucionar e, por isso, merece um tratamento à parte.

3.3 Desafios na concepção do suporte a sensibilidade ao contexto

Como comentado na seção anterior, a construção do suporte à sensibilidade ao contexto para as aplicações apresenta inúmeros desafios. Tais desafios compreendem (VENECIAN, 2009):

- A caracterização dos elementos de contexto para uso na aplicação;
- A aquisição do contexto a partir de fontes heterogêneas, tais como sensores físicos, base de dados, agentes e aplicações;
- A representação de um modelo semântico formal de contexto;
- O processamento e interpretação das informações de contexto adquiridas;
- A disseminação do contexto a entidades interessadas de forma distribuída e na hora certa;
- O tratamento da qualidade da informação contextual.

As subseções a seguir apresentam o estudo sobre os fundamentos inerentes para o suporte à sensibilidade ao contexto.

3.3.1 Identificação dos elementos de contexto

Dois grandes desafios lidados ao se desenvolver um sistema sensível ao contexto são o de delimitar as ações dependentes de contexto nesses sistemas e o de identificar os elementos contextuais que caracterizam a situação na qual essas ações são executadas. Estudos mostram que a identificação dos elementos de contexto depende fortemente do tipo da tarefa e domínio em questão (OZTURK; AAMODT, 1997). Por outro lado, o papel que o contexto desempenha pode ser generalizado sobre tarefas e domínios específicos (VENECIAN, 2009).

O contexto é uma construção dinâmica: embora em algumas situações, o contexto seja relativamente estável e previsível, existem muitas outras onde isso não acontece. Na maioria dos casos, é muito difícil - ou mesmo impossível - para o projetista de uma aplicação sensível ao contexto enumerar o conjunto de estados contextuais que podem existir na aplicação, identificar que informações podem determinar com precisão um estado contextual desse conjunto e definir que ações devem ser executadas em um estado particular (OZTURK; AAMODT, 2001).

Com o propósito de apoiar a identificação dos elementos de contexto, algumas classificações para as informações contextuais foram propostas na literatura. Uma dessas classificações divide as informações contextuais em: (1) contexto primário, ou básico, ou de baixo nível; e (2) contexto complexo, ou de alto nível (WANG et al., 2004). A primeira indica elementos contextuais que podem ser percebidos, automaticamente, por sensores físicos ou lógicos e a segunda refere-se a informações de contexto fornecidas pelo próprio usuário ou deduzidas por motores de inferência a partir de um conjunto de informações de contexto de baixo nível.

Exemplos de contextos básicos incluem identidade (de atores ou dispositivos), atividade atual (que pode ser uma etapa em um processo ou um passo em um *workflow*), localização (geográfica ou virtual), tempo (dia, hora, estação do ano), condições ambientais (temperatura, qualidade do ar, luz, som), disponibilidade de recursos (bateria, largura de banda, tamanho da tela), recursos próximos (dispositivos acessíveis, impressoras, hosts), medidas fisiológicas (pressão sanguínea, batimento cardíaco, atividade muscular, tom de voz), entre outros.

Contextos complexos podem ser, por exemplo, a situação do indivíduo (se está falando, lendo, caminhando ou escrevendo), situações sociais (com quem o usuário está, quem são as pessoas próximas), atividades sociais (se o usuário está em reunião ou ministrando aula), entre outras. Para identificar se um usuário está ministrando uma aula, por exemplo, pode ser criada uma regra lógica que obtenha como contextos básicos a identificação da sala onde ele se encontra, a indicação da existência de outras pessoas na sala, a posição do usuário em relação a essas pessoas e a indicação da existência de uma instância de algum software de apresentação (VENECIAN, 2009).

(OZTURK; AAMODT, 2001) indica que o contexto varia em 5 dimensões: período de tempo, episódios de uso anteriores conhecidos pela pessoa, estado das interações sociais, mudanças nos objetivos internos, e influências do local onde a pessoa se encontra.

Uma outra classificação para as informações de contexto separa-o em porções estáticas e dinâmicas (GAUVIN; BOURRY-BRISSET; AUGER, 2004). A porção estática inclui os parâmetros externos, gerais e relativamente fixos, relacionados ao trabalho do usuário. A porção dinâmica é composta por uma memória dinâmica das ações do usuário, continuamente atualizada quando mudanças, eventos, atividades ou padrões aprendidos de ações ocorrem durante a execução do trabalho.

(KORPIPAA et al., 2003) definiu alguns critérios para a identificação do que escolher como elemento contextual ao construir uma aplicação ciente de contexto:

- Habilidade para descrever propriedades úteis do mundo real;
- Habilidade para inferência de contextos complexos;
- Facilidade ou viabilidade de ser medido ou reconhecido, automaticamente, de forma o mais precisa e não ambígua possível.

(ROSA; BORGES; SANTORO, 2003) propõe um *framework* conceitual de contexto para sistemas colaborativos que classifica as informações de contexto em cinco categorias principais:

- Informações sobre as pessoas e os grupos;
- Informações sobre tarefas agendadas;
- Informações sobre o relacionamento entre pessoas e tarefas;
- Informações sobre o ambiente onde ocorre a interação;
- Informações sobre tarefas e atividades concluídas.

De uma maneira genérica, as informações de contextos referentes a uma ação ou tarefa podem ser identificadas a partir das respostas às seis perguntas apresentadas na seção 3.3.3.

3.3.2 Características das informações contextuais

As informações contextuais possuem características bem peculiares, que devem ser ressaltadas. De acordo com (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2002), a natureza da informação contextual deve ser levada em conta ao se construir sistemas ubíquos e de computação sensível ao contexto. Eis alguns pontos que cabe destacar:

- **Características temporais da informação contextual** - pode-se caracterizar uma informação contextual como sendo estática ou dinâmica. Informações contextuais estáticas descrevem aspectos invariáveis dos sistemas, como a data de aniversário de uma pessoa, por exemplo. Já as informações dinâmicas, que são as mais comuns em sistemas ubíquos, variam freqüentemente. O tempo de validade de uma informação contextual dinâmica é altamente variável - por exemplo, relações entre colegas ou amigos podem durar por meses e anos, enquanto a localização e a atividade de uma pessoa podem mudar a cada minuto. A característica temporal influencia e determina quando uma determinada informação deverá ser adquirida. Enquanto informações estáticas podem ser facilmente obtidas de maneira direta dos usuários das aplicações, mudanças freqüentes no contexto são detectadas através de meios indiretos, como sensores.

- **Imperfeições da informação contextual** - a informação pode estar incorreta, se não refletir o verdadeiro estado do mundo que ela modela, inconsistente, se contém informação contraditória, ou incompleta, se, sob alguns aspectos, o contexto não é reconhecido. Em ambientes extremamente dinâmicos, como o da Computação Ubíqua, a informação contextual rapidamente se torna obsoleta, não refletindo o ambiente que deveria representar. Isso ocorre pelo fato de que, freqüentemente, as fontes, os repositórios e os consumidores de contexto estão distribuídos, gerando, muitas vezes, atrasos entre o envio e a entrega das informações contextuais. Além disso, os produtores de contextos, como sensores, algoritmos de derivação e usuários, podem prover informação imperfeita. Esse é, particularmente, um problema que ocorre quando uma informação contextual é inferida a partir de sensores de mais baixo nível; por exemplo, quando a atividade de uma pessoa é inferida indiretamente, a partir de sua localização e do nível de ruído ao seu redor. Quedas de canais, interferências e outras falhas na comunicação também podem ocorrer no caminho entre e o envio e a entrega da informação contextual, perdendo ou corrompendo parte do que foi enviado ou a informação por completo.

- **Contexto tem representações alternativas** - a maioria das informações contextuais em sistemas sensíveis ao contexto é proveniente de sensores. Geralmente, existe uma grande diferença entre aquilo que é lido nos sensores e a abstração entendida pelas aplicações. Essa diferença de abstração se deve aos tratamentos e processamentos que uma informação contextual deve passar - por exemplo, um sensor de localização fornece as coordenadas geográficas de uma pessoa ou de um dispositivo, enquanto uma aplicação está interessada na identidade do prédio ou da sala em que o usuário está. Os requisitos e níveis de abstração que uma informação contextual exige podem variar de uma aplicação para a outra e, portanto, um modelo de

contexto deve suportar múltiplas representações do mesmo contexto em diferentes formas e níveis de abstração e, ainda, ser capaz de entender os relacionamentos entre essas representações alternativas. Informações contextuais são extremamente inter-relacionadas. Diversos relacionamentos entre as informações contextuais são evidentes, por exemplo, proximidade entre usuários e seus dispositivos, mas outros tipos de relacionamentos podem não ser tão óbvios. As informações contextuais podem estar relacionadas entre si através de regras de derivação, que descrevem como uma informação contextual é obtida a partir de uma ou mais informações.

3.3.3 Dimensões de informação de contexto

A partir das definições apresentadas nas seções anteriores, percebe-se a existência de uma grande diversidade de informações que podem ser utilizadas como informações de contexto - diversidade essa que depende do domínio da aplicação em questão. Muitas aplicações sensíveis ao contexto têm explorado informações de identidade e de localização de pessoas e objetos para proverem algum serviço útil a usuários, como as aplicações pioneiras *Active Badge* (SCHILIT, 1995) e *ParcTab* (SCHILIT; ADAMS; WANT, 1994). Ambos os protótipos utilizavam mecanismos emissores de sinais que forneciam a localização de pessoas em um edifício, além de identificarem essas pessoas em mapas eletrônicos atualizados periodicamente. Com tais informações era possível, por exemplo, realizar transferências automáticas de chamadas telefônicas.

Aplicações sensíveis ao contexto mais recentes passaram a utilizar as facilidades do sistema de localização GPS (*Global Positioning System*), bastante utilizado no monitoramento de automóveis em cidades e rodovias. Por exemplo, o sistema CyberGuide (ABOWD; MYNATT; RODDEN, 2002) é utilizado como um guia turístico capaz de escolher conteúdos audiovisuais para serem exibidos conforme as informações de localização de pessoas. Com os avanços na área de comunicação por redes sem fio, novos sistemas sensíveis ao contexto passaram a explorar informações de localização, como o sistema Guide, que utiliza sinais de redes 802.11 para identificar a localização de turistas ao longo de uma cidade e, a partir de sua localização, gerar roteiros personalizados (VENECIAN, 2009).

No entanto, existem outras informações de contexto além de localização e identificação de pessoas e objetos. A maioria dos sistemas sensíveis ao contexto não incorpora várias das informações disponíveis em um ambiente, como noções de tempo, histórico e dados de outros usuários. Em combinação com as características de aplicações sensíveis ao contexto, (ABOWD; MYNATT; RODDEN, 2002) discute a utilização de cinco dimensões semânticas de informações de contexto para auxiliar projetistas e desenvolvedores na especificação, modelagem e estruturação de informações de contexto de suas aplicações. Essas cinco dimensões semânticas são:

- *Who* (quem) - seres humanos realizam suas atividades e se recordam de fatos passados com base na presença de pessoas e/ou objetos. Aplicações sensíveis ao contexto devem, portanto, controlar a identificação de todas as entidades participantes de uma atividade no intuito de atender às necessidades de usuários. Informações de contexto de identificação podem incluir, entre outras, nome, endereço eletrônico, senha, voz e impressão digital.

- *Where* (onde) - a mais explorada das dimensões de informações de contexto, a localização de entidades em ambientes físicos é normalmente associada a outras dimensões, como a dimensão temporal *When* (quando). Ao combinar essas duas dimensões, é possível explorar não apenas a mobilidade de usuários, mas também informações sobre sua orientação em um ambiente físico e, conseqüentemente, fornecer serviços ou informações adaptados ao comportamento desses usuários. Informações de contexto de localização incluem, entre outras, latitude, longitude, altitude, cidade e posição relativa a objetos e pessoas.

- *When* (quando) - informações de contexto temporais podem ser usadas para situar eventos em uma linha do tempo ou auxiliar na interpretação de atividades humanas e no estabelecimento de padrões de comportamento. Por exemplo, uma visita breve a uma página Web pode indicar falta de interesse do usuário com relação ao conteúdo da página. Já no caso de uma aplicação de monitoramento de pessoas idosas, essa aplicação verifica se os instantes ou intervalos de tempo das atividades do paciente são compatíveis com a rotina diária do mesmo. Nos casos em que há desvios de padrão, a aplicação deve notificar o médico de plantão. Informações de contexto temporais incluem, entre outras, data, hora, intervalos de tempo, dia da semana, mês e ano.

- *What* (o que) - identificar o que um usuário está fazendo em um determinado momento pode ser uma tarefa complicada para uma aplicação em que atividades, não-previstas pelo projeto da aplicação, podem ser realizadas de forma concorrente. Configura-se, assim, como um dos principais desafios na computação sensível ao contexto a obtenção de informações de contexto que possibilitem a interpretação correta da atividade de um usuário. Informações de contexto de atividades variam de aplicação para aplicação, por exemplo, escrever na lousa, anotar em um caderno, trabalhar em grupo e participar de uma reunião, palestra, ou operação cirúrgica.

- *Why* (por que) - mais desafiador ainda que perceber e interpretar o que um usuário está fazendo, é entender o porquê de sua ação. Em geral, as informações de contexto de atividade (*What*) e de motivação (*Why*), por serem mais complexas, são obtidas por meio da combinação de informações de outras dimensões. O estado

emocional de um usuário pode também ser indicativo de sua motivação para a realização de uma tarefa. Aplicações sensíveis ao contexto podem obter, via sensores, informações que possam dar uma indicação do estado emocional de um usuário, por exemplo, o foco de atenção, a expressão facial, características de batimento cardíaco, níveis de pressão arterial, entonação vocal e ondas cerebrais do tipo *alfa*.

Essas cinco dimensões semânticas discutidas em (ABOWD; MYNATT; RODDEN, 2002) não sugerem completeza, mas sim, um conjunto básico de diretrizes a ser seguido no processo de construção de uma aplicação sensível a contexto. (TRUONG; ABOWD; BROTHERTON, 2001) discute uma dimensão semântica, originada do domínio de aplicações de captura e acesso:

- *How* (como) - no contexto de aplicações de captura e acesso, esta dimensão fornece informações relativas a como os recursos de um ambiente físico podem ser capturados e acessados. É importante que aplicações sensíveis ao contexto tenham informações não apenas do número e do papel dos dispositivos disponíveis para captura e acesso em um ambiente, mas também de suas características funcionais. Essas informações podem ser utilizadas, por exemplo, para a personalização de acesso a informações capturadas via dispositivos - por exemplo, os *handhelds* - com características de acesso bastante restritas, como tamanho de tela, quantidade de energia em bateria e suporte à entrada e saída de dados.

3.3.4 Aquisição de contexto

A aquisição de contexto está associada com a forma na qual as informações contextuais são obtidas, podendo ser sentida, derivada ou explicitamente provida (MOSTEFAOUI; ROCHA; BREZILLON, 2004):

- **Aquisição sentida** - este tipo compreende as informações que podem ser adquiridas do ambiente por meio de sensores (temperatura, nível de ruído, dispositivos presentes).

- **Aquisição derivada** - este é o tipo de informação que pode ser obtida em tempo de execução, via inferência. Por exemplo, é possível calcular a idade de uma pessoa a partir de sua data de nascimento e de uma informação de data e hora atuais.

- **Aquisição provida** - informação que é explicitamente fornecida à aplicação. Por exemplo, os dados cadastrais de um usuário a serem diretamente fornecidos à aplicação por meio de um formulário.

Em (VENECIAN, 2009), o processo de aquisição de contexto é dividido em modo explícito ou modo implícito. O modo explícito é quando o usuário informa diretamente

ao sistema o seu contexto atual através de interfaces específicas como, por exemplo, um formulário ou botões de ação. O modo implícito é realizado por sensores, físicos ou lógicos, que monitoram continuamente o ambiente físico ou virtual do usuário e capturam informações contextuais desses ambientes.

A etapa de aquisição não é uma tarefa fácil, principalmente quando a informação é sentida. Isso ocorre devido à grande variedade de sensores. Além disso, informação contextual possui uma natureza dinâmica, sendo necessário que a aplicação gerencie todos esses aspectos.

3.3.5 Modelagem de contexto

Atualmente, o desenvolvimento de aplicações sensíveis ao contexto é uma tarefa complexa, o que torna o uso de técnicas de modelagem extremamente úteis. Contudo, os atuais modelos para desenvolvimento de software não oferecem suporte para o projeto de tais aplicações e, sobretudo, tais técnicas de modelagem não provêem suporte para modelagem das informações contextuais (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2002).

Existem inúmeras abordagens para modelar informações contextuais, dentre as quais pode-se ressaltar:

- **Modelos com métodos de marcação** - seguem uma estrutura hierárquica de marcação com atributos e conteúdo. Em geral, utilizam-se de linguagens de marcação derivadas da SGML, *Standard Generic Markup Language*.

- **Modelos chave-valor** - utilizam um modelo simples de atributo e valor, sendo fáceis de gerenciar, contudo têm pouco poder de expressão.

- **Modelos gráficos** - baseados em notações gráficas, em geral são derivados de adaptações e extensões de modelos gráficos já difundidos, como UML, ORM ou o modelo Entidade-Relacionamento.

- **Modelos orientados a objetos** - esta abordagem tem a intenção de aplicar os principais benefícios do modelo orientado a objetos, notadamente, encapsulamento e reusabilidade, à modelagem de contexto. Nesses casos, o acesso às informações contextuais é feito somente através de interfaces bem definidas.

- **Modelos baseados em lógica** - define-se o contexto de modo que se possa inferir expressões ou fatos a partir de um conjunto de outros fatos e expressões. Em geral, este modelo possui um alto grau de formalismo.

- **Modelos baseados em ontologias** - uma ontologia é uma especificação de uma conceituação, isto é, uma descrição de conceitos e relações que existem entre eles,

em um domínio de interesse. Nesse modelo o contexto é modelado em ontologias, construindo uma base de conhecimento do contexto.

Em (STRANG; LINNHOFF-POPIEN, 2004), são avaliadas as abordagens citadas acima para modelagem de contexto, quando os seguintes critérios são considerados:

1. **Composição distribuída (cp)** - a composição do modelo de contexto deve ser altamente dinâmica em termos de tempo, topologia de rede e origem, podendo estar distribuído em diversas localidades ao longo do tempo.

2. **Validação parcial (vp)** - deve ser possível validar parcialmente o modelo, dado que nem todas as informações podem estar disponíveis ao mesmo tempo e que o conhecimento do contexto pode ser derivado da composição de outras informações distribuídas.

3. **Riqueza e qualidade da informação (rqi)** - a qualidade e a riqueza das informações podem variar de acordo com o tempo e com o tipo de sensor. Por esse motivo, o modelo deve permitir anotações de qualidade e riqueza da informação de contexto representada.

4. **Incompletude e ambigüidade (ia)** - as informações contextuais disponíveis em um dado momento podem ser incompletas ou ambíguas. Estas características devem ser cobertas pelo modelo.

5. **Nível de formalidade (nf)** - a formalidade objetiva dar um visão única do modelo; é altamente desejável que todos os participantes tenham a mesma interpretação. A formalidade permite, também, o processamento automático das informações de contexto diretamente do modelo, por exemplo, para validação.

6. **Aplicabilidade nos ambientes existentes (aae)** - para uma boa aceitação, é importante que o modelo seja aplicável às infra-estruturas de suporte a contexto já existentes.

Os resultados da análise realizada por (STRANG; LINNHOFF-POPIEN, 2004) podem ser observados na Tabela 3.1. A notação apresentada atribui o sinal “-” para o critério não-satisfeito pelo modelo, o sinal “+” para o critério atendido de maneira satisfatória e o sinal “++” para o critério que é completamente satisfeito. Nota-se, a partir dos dados presentes na tabela indicada, que modelos baseados em ontologias são os que melhor atendem aos critérios existentes no processo de modelagem de infomações contextuais.

Tabela 3.1: Avaliação das abordagens para Modelagem de Contexto

Modelo	cp	vp	rqi	ia	nf	aae
Chave-Valor	-	-	-	-	-	+
Método de marcação	+	++	-	-	+	++
Gráficos	-	-	+	-	+	+
Orientação a objetos	++	+	+	+	+	+
Baseados em lógica	++	-	-	-	++	-
Baseados em ontologia	++	++	+	+	++	+

3.3.6 Interpretação de contexto

A interpretação de contexto pode ser entendida como o conjunto de métodos e processos que realizam a abstração, o mapeamento, a manipulação, a agregação, a derivação, a inferência e demais ações sobre as informações contextuais, com o propósito de facilitar o entendimento de um determinado contexto pelas aplicações e auxiliá-las na tomada de decisões. O processo de interpretação de contexto consiste na manipulação e refinamento das informações contextuais de um ambiente (VENECIAN, 2009).

Em (DEY, 2000), a interpretação de contexto é vista como o processo de se elevar o nível de abstração das informações contextuais de um ambiente, ou seja, gerar uma informação contextual mais elaborada a partir de outra mais primitiva.

O processo de interpretação de contexto pode ser bastante simples, como, por exemplo, no caso de derivar o nome de uma rua a partir de suas coordenadas geográficas, ou ser bastante complexo e oneroso, como, por exemplo, no caso de inferir o humor de um usuário a partir de seu perfil e da atividade em que ele está realizando. Além disso, o ambiente em questão, o da Computação Ubíqua, é extremamente dinâmico e complexo. As informações contextuais podem estar espalhadas e distribuídas em qualquer lugar e com alto grau de mobilidade. Essa complexidade faz com que haja a necessidade de um suporte computacional às aplicações, de maneira a auxiliá-las na realização de interpretações de contextos. Tais atividades onerosas devem ser abstraídas das aplicações e o módulo *Interpretador de Contexto* torna-se, portanto, um componente essencial em uma plataforma de suporte a tais aplicações. Ele deve ser capaz de obter e prover informação contextual em diferentes níveis de abstração, conforme o desejo do usuário e de suas aplicações. Uma aplicação pode desejar tanto informações mais brutas, de mais baixo nível ou informações mais abstratas e elaboradas, de mais alto nível, provenientes de um processo de refinamento e interpretação (VENECIAN, 2009).

3.3.7 Processamento e raciocínio sobre o contexto

Um dos principais problemas na utilização de informações contextuais é como obter contexto realmente significativo para quem precisa utilizar essa informação, a partir

de um conjunto de informações dispersas e desconexas, obtidas por mecanismos heterogêneos de aquisição. Para isso, funcionalidades de processamento e raciocínio sobre a informação contextual devem ser disponibilizadas. Questões como tratamento da incerteza devem ser consideradas, pois como o contexto evolui bastante com o tempo, é difícil inferir com precisão qual é, de fato, o contexto atual da situação (VENECIAN, 2009).

Os termos *raciocínio* e *inferência* são geralmente utilizados para indicar qualquer processo pelo qual conclusões são alcançadas (RUSSELL; NORVIG, 2003). O projeto e implementação de um mecanismo para raciocínio de contextos pode variar bastante, dependendo do tipo do conhecimento contextual envolvido. Idealmente, o processamento do contexto deve ser implementado separadamente do comportamento do sistema e não embutido no código da aplicação (BELOTTI et al., 2004).

O raciocínio é utilizado para examinar a consistência do contexto e para inferir contexto implícito de alto nível, a partir de contextos explícitos, de baixo nível (WANG et al., 2004). A consistência é necessária pois, muitas vezes, a informação contextual adquirida automaticamente pode apresentar erros e ambigüidades. Por exemplo, um sensor de presença pode detectar o celular de um usuário em sua casa e deduzir que o mesmo está em casa. Porém, um outro sensor de presença baseado em câmeras detecta a presença do usuário em seu escritório. Essas duas informações são conflitantes e precisam ser resolvidas.

Existem duas diferentes abordagens para inferir contexto de alto nível (RANGANATHAN; CAMPBELL, 2003). A primeira utiliza regras estáticas, pré-definidas pelos desenvolvedores em lógica de primeira ordem, lógica de mais alta ordem, lógica descritiva, lógica temporal ou lógica *fuzzy*. A segunda utiliza técnicas de aprendizagem, como redes bayesianas, redes neurais e raciocínio baseado em casos.

No raciocínio baseado em regras, cada regra possui uma prioridade associada, de modo que uma delas possa ser escolhida caso mais de uma regra seja verdadeira ao mesmo tempo. Esse tipo de raciocínio tem a desvantagem de exigir uma definição explícita das regras por humanos, além de não ser flexível e não se adaptar a circunstâncias que mudam freqüentemente (RANGANATHAN; CAMPBELL, 2003). A utilização de técnicas de aprendizado automático permite que a máquina aprenda um padrão ou uma regra a partir de um conjunto de dados de teste, em uma fase chamada de treinamento.

3.3.8 Armazenamento de informações contextuais

A necessidade de manter o histórico de informações de contexto é um requisito ligado à aquisição de informações de contexto, bem como à disponibilidade contínua dos componentes de captura de informações de contexto. Um histórico de contexto pode ser utilizado para estabelecer tendências e predizer valores futuros de informações contexto. Sem o armazenamento dessas informações, esse tipo de análise não seria possível de ser

realizado.

3.4 Considerações sobre o capítulo

Este capítulo definiu o significado do termo “contexto”, dentro da área de Computação Ubíqua, bem como discutiu sobre aspectos de identificação, aquisição, modelagem, interpretação, raciocínio e processamento de informações contextuais.

Contexto é o conjunto de estados e características de um ambiente que determina o comportamento de uma aplicação ou no qual um evento de uma aplicação ocorre e interessa ao usuário. O suporte a sensibilidade ao contexto, por parte das aplicações, leva à necessidade de se responder uma série de interrogações: como identificar e adquirir elementos contextuais, que características e dimensões tais elementos possuem, de que maneira representá-los e armazená-los na aplicação e de que forma ela raciocinará e interpretará esses elementos.

No capítulo seguinte, será dada ênfase em ontologias, que, segundo os resultados listados na tabela 3.1, é a abordagem de modelagem de contexto mais adequada para o trabalho.

4 ONTOLOGIAS: FUNDAMENTOS E TECNOLOGIAS UTILIZADAS

Este capítulo tem o objetivo de efetuar uma discussão sobre ontologias, caracterizando conceitos fundamentais e as principais tecnologias relacionadas, as quais foram empregadas no desenvolvimento deste Projeto de Graduação.

A definição mais aceita e citada pelos autores da área de Computação é a que define ontologia como uma especificação formal e explícita de uma conceituação compartilhada (FENSEL, 2000), onde:

- Conceituação - se refere ao modelo abstrato do mundo real;
- Explícita - significa que os conceitos e seus requisitos são definidos explicitamente: definições de conceitos, instâncias, relações, restrições e axiomas;
- Formal - indica que a ontologia é processável por máquina, permite raciocínio automático e possui semântica lógica formal;
- Compartilhada - significa que uma ontologia captura o conhecimento apresentado não apenas por um único indivíduo, mas por um grupo, sendo uma terminologia comum da área modelada ou acordada entre os desenvolvedores.

Uma ontologia não se resume somente a um vocabulário. Taxonomia, que é o tipo de ontologia mais básico, consiste em especificações definidas somente com relacionamentos hierárquicos “é um”, por exemplo *Maria é uma professora* ou *João é um piloto* (AFONSO, 2007).

- Ontologias podem incluir relacionamentos não-hierárquicos. Por exemplo, pode-se ter relacionamentos do tipo “tem-interesse-em” entre os conceitos *pessoa* e *interesse*, sem que se trate de um relacionamento hierárquico.
- Ontologias também podem incluir restrições. Em uma ontologia sobre pessoas, por exemplo, pode-se construir uma restrição sobre o conceito *pessoa* baseada no relacionamento “tem-nome” que impõe que “uma pessoa terá exatamente um nome”.

Por vezes, processos de inferência permitem a descoberta de novas informações. Por exemplo, em uma ontologia em que *parente* é um relacionamento mais geral do que *mãe*, se **Maria** é mãe de **João**, o sistema será capaz de concluir que **Maria** é parente de **João**. Um usuário que consulta a ontologia buscando pelos parentes de **Maria** visualizará a resposta de que **João** atende a esse requisito, sem que esse fato tenha sido explicitamente declarado (AFONSO, 2007).

As ontologias podem ser classificadas em diferentes tipos: (AFONSO, 2007)

- **Ontologias de domínio**: capturam o conhecimento válido para um tipo particular de domínio (como mecânica, medicina, biologia, entre outros). Expressam o vocabulário relativo a um domínio particular, descrevendo situações reais deste domínio.

- **Ontologias genéricas**: são similares as ontologias de domínio, entretanto os conceitos definidos por elas são considerados genéricos entre diversas áreas. Descrevem conceitos tipicamente gerais, como: estado, espaço, tempo, processo, evento, importância, ação, entre outros, os quais são independentes de um domínio ou problema particular. Conceitos em ontologias de domínio são freqüentemente definidos como especializações de conceitos de ontologias genéricas.

- **Ontologias de aplicação**: contém todos os conceitos necessários para modelagem do conhecimento requerido por uma aplicação em particular. Esses conceitos correspondem freqüentemente aos papéis desempenhados por entidades do domínio enquanto executam certa atividade.

- **Ontologias de representação**: não se comprometem com nenhum domínio em particular. Determinam entidades representacionais sem especificar o que deve ser representado. Ontologias de domínio e ontologias genéricas são descritas por meio das primitivas fornecidas pelas ontologias de representação.

4.1 Linguagens para a representação de ontologias

As linguagens para ontologias devem permitir a seus usuários escrever conceitualizações formais explícitas sobre modelos de domínios. Seus principais requisitos são: sintaxe bem definida, suporte eficiente ao raciocínio, semântica formal, suficiente potencial de expressividade e conveniência de expressão (WARKEN, 2008).

Expressividade X Raciocínio:

- Quanto mais rica a linguagem, mais ineficiente se torna o suporte ao raciocínio;

- Às vezes pode ultrapassar as fronteiras da computabilidade;

- É necessário um compromisso entre expressividade e o suporte ao raciocínio.

Raciocínio sobre o conhecimento:

- Pertinência a Classes: se **X** é uma instância de uma classe **C** e **C** é subclasse de outra classe **D**, então é possível inferir que **X** é uma instância de **D**.
- Equivalência de Classes: se a classe **A** é equivalente à classe **B** e **B** é equivalente à classe **C**, então **A** também é equivalente a **C**.
- Consistência: se **X** é instância das classes **A** e **B**, mas **A** e **B** são disjuntas, então a ontologia está inconsistente.
- Classificação: certos pares propriedade-valor são condição suficiente para pertinência em determinadas classes. Se um indivíduo **X** satisfaz tais condições para uma classe **A**, pode-se concluir que **X** deve ser uma instância de **A**.

Existem diversas linguagens formais definidas para permitir a representatividade de ontologias. O *World Wide Web Consortium* (W3C) afirma que a maneira mais completa de representar ontologias é com a OWL (*Web Ontology Language*). Tal afirmação do W3C associada à variedade de tecnologias capazes de interagir com informações em OWL (dentre as quais algumas são listadas na seção 4.2) fazem da OWL a linguagem de manipulação de ontologias que melhor se adapta ao trabalho.

Esta seção caracteriza a *Web Ontology Language*, bem como os componentes RDF Schema, RDF e XML, que a constituem.

4.1.1 XML - Extensible Markup Language

XML, *eXtensible Markup Language*, é uma recomendação do W3C para gerar linguagens de marcação para necessidades especiais. É um subtipo de SGML, *Standard Generalized Markup Language*, ou Linguagem Padronizada de Marcação Genérica, capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet (WARKEN, 2008).

Os seguintes princípios são identificados na linguagem XML:

- Separação entre o conteúdo e a formatação;
- Simplicidade e legibilidade, tanto para humanos quanto para computadores;
- Possibilidade ilimitada de criação de *tags*;
- Criação de arquivos para validação de estrutura, chamados DTDs;
- Concentração na estrutura da informação, e não na sua aparência.

4.1.2 RDF e RDF Schema

O RDF (*Resource Description Framework*) é uma arquitetura de metadados, utilizada para descrever recursos no contexto Web. O RDF é estruturado sobre a linguagem XML e também é uma recomendação do W3C. O RDF propõe-se a resolver um dos principais desafios encontrados pelas diferentes comunidades de descrição de recursos: fornecer interoperabilidade entre os diversos padrões de metadados (WARKEN, 2008; JARDIM, 2009).

O modelo de dados RDF é composto pelos objetos descritos a seguir:

- *Resources*: um recurso pode ser caracterizado como qualquer coisa representada dentro dos padrões de descrição de uma expressão RDF. Exemplo de recurso: uma página WEB (www.ucpel.tche.br), uma parte de uma página WEB (www.ucpel.tche.br/vestibular). É importante salientar que todos os recursos devem sempre estar nomeados e identificados por um URI (*Uniform Resource Identifier*), permitindo sua identificação entre os recursos RDF;

- *Properties*: uma propriedade serve para descrever uma característica, um atributo ou uma relação qualquer utilizada em um recurso. Geralmente, essa descrição possui um significado específico e permite a interligação com outras propriedades, assemelhando-se ao esquema do modelo de entidade-relacionamento;

- *Literals*: um valor qualquer que um objeto pode assumir, de acordo com a propriedade;

- *Statements*: uma espécie de declaração de um recurso contendo um nome, uma propriedade e um valor agregado a ela. Estas três partes da indicação são chamadas respectivamente de: sujeito, predicado e objeto. Com isso, os *statements* conseguem representar essas interligações entre os recursos, suas propriedades e seus valores.

O recurso pode ser caracterizado como um objeto e é sempre identificado por um URI que, através de *statements*, possibilita a construção básica de um modelo em RDF. Para tal, utiliza-se a indicação de uma tupla, (predicado, sujeito, objeto), permitindo a ligação entre os valores e os recursos, cuja função é a de prover uma maior representação de modelos complexos. Além de ser apresentado como tupla, o modelo RDF permite uma visualização através de grafos, convencionados pela W3C, que possibilitam representar recursos através de nós, propriedades através de arcos e literais por retângulos.

A viabilidade para representar a interligação das propriedades e os outros recursos mostra-se limitada. Com isso, surgiu o RDF Schema (BRICKLEY; GUHA, 2004), uma extensão de RDF, que veio fornecer descrição de grupos de recursos e os relacionamentos

existentes entre eles. Existe a flexibilidade de criar vocabulários, representados por classes e propriedades com características restritas, podendo serem reaproveitados em outros modelos.

Uma característica bem clara da forma de reuso dos recursos e das propriedades já existentes está na utilização de sintaxes de outras linguagens. A especificação de uma linguagem não tenta enumerar um vocabulário específico para descrição das classes e propriedades de um documento RDF. Mas a característica do RDF Schema de utilizar apenas as características (recursos, propriedades) que o convém facilita a descrição de documentos RDF com a utilização de várias sintaxes.

4.1.3 OWL - Web Ontology Language

A maneira mais completa de representar ontologias, segundo o W3C, é com a OWL (*Web Ontology Language*). Ela fornece suporte a abstrações de classes, generalização, agregação, relações de transitividade, simetria e detecção de inconsistências. O W3C lançou a OWL como um padrão no uso de ontologias em 2008. A OWL é baseada em XML e, devido a isso, a informação pode ser facilmente trocada entre sistemas heterogêneos. Por ter sido projetada para ser lida por aplicações computacionais, algumas vezes considera-se que a linguagem não possa ser facilmente lida por humanos, porém esta é uma questão de ferramentas. OWL vem sendo usada para criar padrões que forneçam um *framework* para gerenciamento de ativos, integração empresarial e compartilhamento de dados na Web (WARKEN, 2008).

OWL, atualmente, tem três sub-linguagens, também chamadas espécies:

- **OWL Lite**: suporta aqueles usuários que necessitam, principalmente, de uma classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. É a linguagem que possui menor complexidade formal.

- **OWL DL**: suporta aqueles usuários que querem a máxima expressividade, enquanto mantém a computabilidade (garante-se que todas as conclusões sejam computáveis) e decidibilidade (todas as computações terminarão em tempo finito). OWL DL inclui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições (por exemplo, embora uma classe possa ser subclasse de muitas classes, uma classe não pode ser instância de outra classe).

- **OWL Full**: é direcionada àqueles usuários que querem a máxima expressividade e a liberdade sintática, mesmo sem qualquer garantia computacional. Por exemplo, em OWL Full, uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um indivíduo por si mesma. É improvável que algum software

de inferência venha a ser capaz de suportar completamente cada recurso da OWL Full.

Cada uma destas sub-linguagens é uma extensão da espécie anterior, tanto em relação ao que pode ser expressado, como em relação ao que pode ser inferido (WARKEN, 2008).

4.2 Tecnologias para a manipulação de ontologias

As principais tecnologias utilizadas para o tratamento de ontologias são descritas nesta seção.

4.2.1 Protégé

Protégé é um ambiente extensível e independente de plataforma escrito em Java e desenvolvido na Universidade de Stanford. É uma das ferramentas mais utilizadas para criação e edição de ontologias e bases de conhecimento, suportando a criação, visualização e manipulação de ontologias em vários formatos. Esse ambiente foi fundamental para o trabalho de concepção, avaliação e instanciação das ontologias empregadas no Projeto de Graduação.

O Protégé possui uma vasta quantidade de *plugins*, importa e exporta ontologias em diversos formatos, principalmente OWL, facilitando a reutilização e o intercâmbio de ontologias, além de incorporar diversas outras funcionalidades, como visualizadores e integradores, além de possibilitar o uso de *plugins* desenvolvidos por usuários (WARKEN, 2008).

Uma ilustração do ambiente Protégé é exibida na figura 4.1.

4.2.2 Jena API

Jena é uma API (*Application Program Interface*) Java desenvolvida pela HP, Hewlett-Packard Company, utilizada para desenvolver aplicações baseadas em Semântica Web (MCBRIDE, 2008). A Jena permite manipulação dinâmica de modelos ontológicos e o desenvolvimento de aplicações baseadas em Semântica Web.

Esta API está dividida essencialmente 5 áreas relativas ao processamento de ontologias:

- Processamento e manipulação de modelos RDF;
- Implementação da linguagem de consulta SPARQL;
- Processamento e manipulação de ontologias descritas em OWL ou DAML;

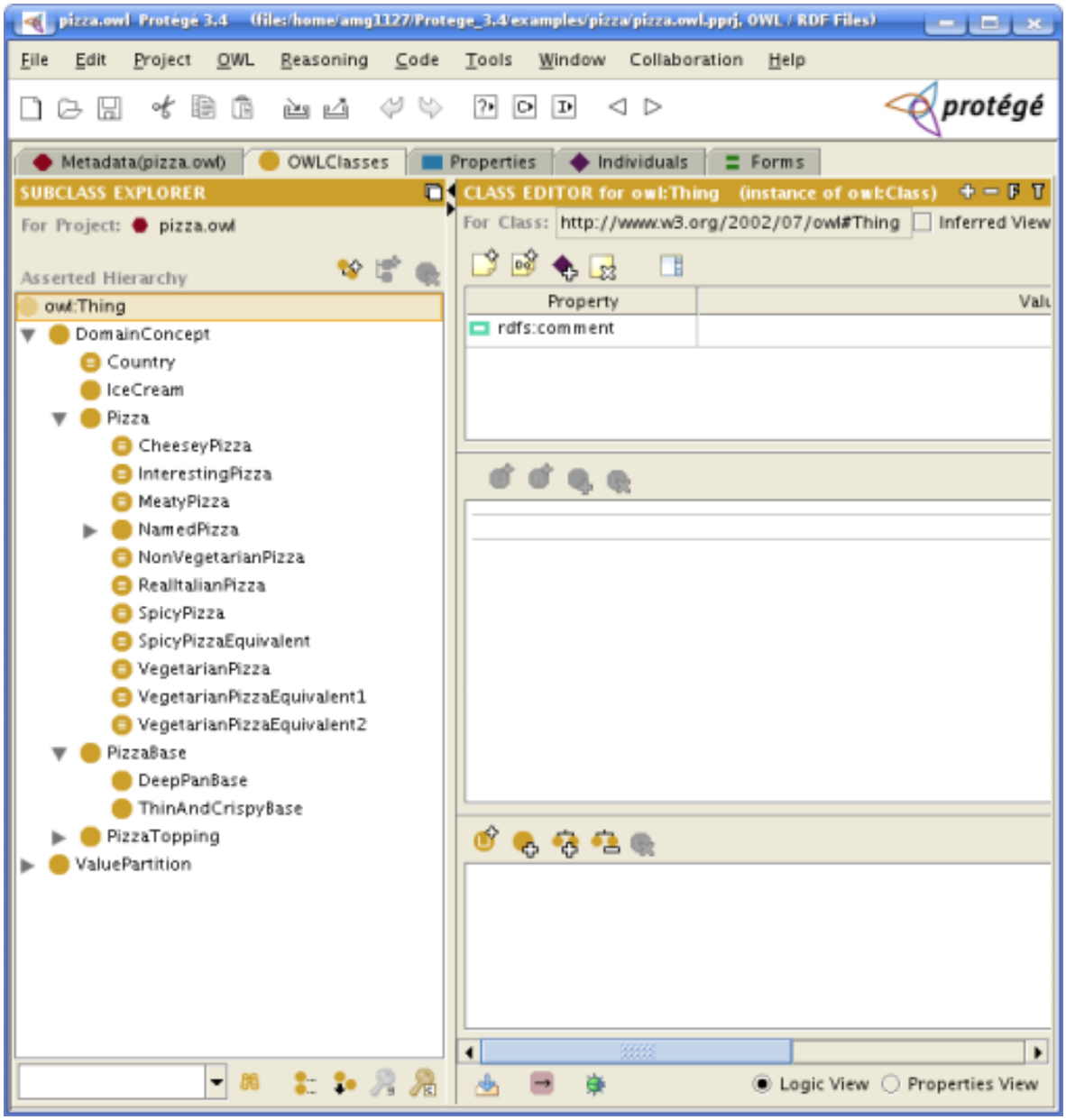

Figura 4.1: Tela do ambiente Protégé, exibindo a hierarquia de classes de uma ontologia

- Inferência sobre OWL e RDFS;
- Persistência de modelos de ontologias sobre bases de dados.

A Jena caracteriza-se por possibilitar a criação e manipulação de modelos RDF, representadas pelos recursos (*resources*), propriedades (*properties*) e *literals*, formando as tuplas que darão origem aos objetos criados pelo Java. Esse conjunto de objetos é usualmente chamado de *model* (WARKEN, 2008).

4.2.3 SPARQL

A SPARQL (***S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage*) é uma linguagem de consulta e protocolo de acesso a dados em RDF, recomendado pelo W3C. A equipe do W3C RDF *Data Access Working Group* divulgou três recomendações para a tecnologia: a SPARQL *Query Language* para RDF; o protocolo SPARQL para RDF; e a SPARQL *Query Results* para o formato XML. A Oracle, HP e IBM participam deste grupo de trabalho (WARKEN, 2008).

A SPARQL abrange três áreas:

1. Linguagem de consulta;
2. Formato dos resultados;
3. Protocolo de acesso.

A figura 4.2 ilustra um exemplo de consulta escrita na linguagem SPARQL.

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
   ?x abc:cityname ?capital ;
      abc:isCapitalOf ?y .
   ?y abc:countryname ?country ;
      abc:isInContinent abc:Africa .
}
```

Figura 4.2: Exemplo de consulta escrita na linguagem SPARQL

4.3 Considerações sobre o capítulo

Este capítulo discutiu sobre conceitos e tecnologias relacionados a ontologias.

Ontologia é uma especificação formal e explícita de uma conceituação compartilhada, constituída por relacionamentos hierárquicos do tipo “é-um”. Opcionalmente, ontologias também possuem relacionamentos não-hierárquicos ou restrições de integridade.

OWL é a linguagem mais adequada para a representação de ontologias nas aplicações, entretanto, não é recomendável o uso do tipo OWL Full, devido à não-garantia de computabilidade. Para o tratamento de ontologias em OWL por humanos, utiliza-se o Protégé, enquanto que, para máquinas, utiliza-se a API Jena.

No capítulo a seguir, serão relacionados alguns projetos adaptativos ao contexto, sob a ótica da Computação Ubíqua.

5 TRABALHOS RELACIONADOS A SENSIBILIDADE AO CONTEXTO

As características do módulo de gerência de contexto das aplicações, bem como o processo de adaptação dinâmica, são questões que ainda não foram resolvidas como um todo na área de Computação Ubíqua. Os trabalhos existentes, referentes à adaptação ao contexto, apresentam aspectos semelhantes (YAMIN, 2004):

1. Atuam gerenciando as larguras de banda utilizadas nas comunicações;
2. Contemplam um domínio específico para as aplicações, tais como: multimídia e informações dependentes do contexto e,
3. Usualmente implementam somente uma estratégia de adaptação: alteração no formato dos dados, replicação de dados ou migração de tarefas.

Nas próximas seções, serão relacionados alguns projetos adaptativos ao contexto, sob a ótica da Computação Ubíqua. São descritas, sucintamente, algumas características de cada modelo.

5.1 SOCAM

SOCAM, de *Service-Oriented Context-Aware Middleware*, é uma arquitetura para a construção de um serviço móvel ciente de contexto. O modelo do contexto é baseado em ontologia, que provê um vocabulário para representar e compartilhar conhecimento de contexto em um domínio da computação onipresente. O projeto da ontologia do contexto é composto de dois níveis hierárquicos. Um nível contém ontologias individuais sobre vários subdomínios, por exemplo, o domínio de uma casa, um escritório, um veículo, etc. Um nível mais alto contém conceitos gerais sobre as outras ontologias, esse nível é chamado de ontologia generalizada ou ontologia de alto nível (GU; PUNG; ZHANG, 2005).

A arquitetura SOCAM segue uma arquitetura semelhante ao padrão *Web Service*, na qual os serviços são registrados em um diretório público e podem ser encontrados e utilizados por outros serviços. Porém, a arquitetura não é independente de linguagem de programação, pois os componentes trocam mensagens usando Java RMI, tornando difícil a integração entre servidores heterogêneos. Além disso, as regras de contexto devem ser carregadas previamente no sistema para que passem a funcionar.

5.2 Flame2008

Flame2008 é um protótipo de uma plataforma de integração para *Web Services* inteligentes personalizados para as olimpíadas de 2008 em Beijing. A principal idéia desse projeto é, através de um dispositivo móvel, realizar *push* de ofertas significantes baseadas na situação atual e no perfil do usuário (WEISSENBERG; VOISARD; GARTMANN, 2004).

No Flame2008, são definidas as seguintes entidades:

- O perfil é composto por interesses, preferências e dados pessoais do usuário.
- Interesses são informações estáticas que não se alteram com a mudança de contexto, por exemplo, se o usuário possui interesse por música clássica e obras de arte.
- As preferências são informações que podem variar, depender da situação. Por exemplo, um usuário em sua cidade pode preferir restaurantes que ofereçam comida italiana, mas quando ele está viajando, pode preferir experimentar a comida local.

Cada usuário pode manter seu perfil através de um conjunto de propriedades que o caracterizam. Além disso, há sensores que obtêm informações do ambiente atual do usuário (localização e tempo). Todas as dimensões do contexto (localização, calendário, perfil) são representadas através de ontologias. Com agregação dessas dimensões, o Flame2008 define situações que são registradas no sistema. Definida a situação em que o usuário se encontra e o seu perfil, o sistema se encarrega de buscar serviços que se encaixam nesses parâmetros. Como resultado, obtém-se instâncias de uma ontologia de alto nível que são usadas para implicitamente construir um perfil de situação que é semanticamente comparada com os perfis de todas as situações conhecidas pelo sistema, e implicitamente comparada com todos os perfis de serviços registrados (WARKEN, 2008).

5.3 JCAF

JCAF, de *Java Context Awareness Framework*, é um *middleware* para Computação Ubíqua baseado em eventos e serviços, desenvolvido na linguagem Java

(BARDRAM, 2005).

Basicamente, ele é constituído de duas partes: *Context-Awareness Programming Framework* e *Context-Awareness Runtime Infrastructure*. O *Context-Awareness Programming Framework* provê um conjunto de classes e interfaces Java para o desenvolvimento de aplicações ubíquas e serviços de contexto, *Context Services*, que constituem a *Context-Awareness Runtime Infrastructure*.

Cada serviço de contexto deve lidar com as informações de um ambiente em particular, por exemplo, uma sala ou um quarto, estando os mesmos dispostos em uma topologia ponto a ponto, para que possam assim se comunicar e trocar informações de contexto entre si. As informações providas por cada serviço de contexto podem ser recuperadas através de requisições e respostas, *request-response*, ou registrando-se como ouvintes de mudanças no contexto.

Um aspecto interessante a ser observado sobre a modelagem de contexto no JCAF é que novas entidades e itens de contexto podem ser adicionados a um serviço de contexto sempre que necessário, mesmo em tempo execução. Porém, toda a modelagem da informação contextual fica a cargo do desenvolvedor da aplicação (WARKEN, 2008).

5.4 LOTUS

LOTUS é uma infra-estrutura projetada com o objetivo de abordar o problema da disponibilização da informação contextual às aplicações em uma abordagem integrada. Foi adotada uma solução que reúne as soluções baseadas em ontologias com as soluções baseadas em *middlewares*. Das soluções baseadas em *middlewares* foram aproveitados mecanismos que permitem que seja feita a descoberta de nós e serviços mesmo em redes heterogêneas. Com relação às ontologias, foi desenvolvido um módulo que permite a representação da informação contextual. Além disso, foram desenvolvidos mecanismos para permitir que a informação presente nas ontologias seja atualizada dinamicamente e um mecanismo para que a informação seja disponibilizada às aplicações. Foi demonstrado ainda, por meio do estudo de caso, que é possível o desenvolvimento de aplicações cientes de contexto usando o Lotus. Neste estudo de caso, foi possível constatar a viabilidade da infra-estrutura, uma vez que a informação contextual é disponibilizada para o desenvolvedor, podendo este focar na lógica de negócio da aplicação. Além disso, a informação contextual é realmente compartilhada entre as aplicações, permitindo que a mesma informação esteja disponível para várias aplicações, independentemente (MOREIRA BUBLITZ, 2007).

5.5 Considerações sobre o capítulo

Este capítulo relacionou alguns projetos que exploram a adaptação ao contexto, sob a ótica da Computação Ubíqua.

O capítulo seguinte caracterizará o *middleware* EXEHDA, plataforma de execução onde o trabalho de Projeto de Graduação será desenvolvido.

6 MIDDLEWARE EXEHDA E SUAS FUNCIONALIDADES

Este capítulo caracteriza o *middleware* EXEHDA, plataforma de execução com a qual esse Projeto de Graduação contribui, no que diz respeito ao mecanismos de sensibilidade ao contexto.

O EXEHDA, *Execution Environment for Highly Distributed Applications*, é um *middleware* direcionado às aplicações distribuídas, móveis e conscientes do contexto da Computação Ubíqua (YAMIN, 2004; YAMIN et al., 2005). Este *middleware* é baseado em serviços, que tem por objetivo criar e gerenciar um ambiente ubíquo.

As seguintes premissas foram perseguidas durante a fase de concepção do EXEHDA (YAMIN, 2004):

Controle da adaptação

A gerência da adaptação pode ocorrer no limite de dois extremos: (i) no primeiro, denominado *laissez-faire*, a aplicação é responsável por toda a adaptação que será realizada; por sua vez, no segundo extremo, (ii) denominado *application-transparent*, o sistema é encarregado de gerenciar toda a adaptação que vier a ocorrer. Nenhuma dessas estratégias pode ser considerada a melhor. Uma estratégia diferente pode ser requerida para cada circunstância e/ou aplicação. Há situações, por exemplo, onde o código-fonte da aplicação não está disponível e a estratégia a ser utilizada deve ser a *application-transparent*. Em outros casos, pode ser mais oportuno incluir apenas na aplicação os mecanismos adaptativos, sem envolver o ambiente de execução. Logo, a proposta para o EXEHDA é modelar um *middleware* que faculte uma estratégia colaborativa com a aplicação nos procedimentos de adaptação. Deste modo, considerando a natureza da aplicação, o programador poderá definir a distribuição de responsabilidades entre o *middleware* e a aplicação no processo de adaptação (VENECIAN, 2009).

Suporte às mobilidades lógica e física

Na perspectiva do EXEHDA, entende-se por mobilidade lógica a movimentação entre equipamentos de artefatos de software e seu contexto, e por mobilidade física o deslocamento do usuário, portando ou não seu equipamento.

A Computação Móvel genericamente se refere a um cenário onde todos, ou alguns nodos que tomam parte no processamento, são móveis. Desta definição, diferentes interpretações podem ser derivadas. Em um extremo, a mobilidade leva em conta as necessidades dos usuários nômades, isto é, usuários cuja conexão na rede ocorre de posições arbitrárias e que não ficam permanentemente conectados. Em outro extremo, estão os usuários móveis, os quais retêm a conectividade durante o deslocamento, tipicamente explorando conexões sem fio. Desta forma, a Computação Móvel é caracterizada por três propriedades: mobilidade, portabilidade e conectividade.

Na proposta do EXEHDA, estas propriedades desdobram duas preocupações de pesquisa: (i) os segmentos sem fio da rede global levantam novas condições operacionais, entre as quais se destaca a comunicação intermitente. A ocorrência de desconexões de nodos no ambiente móvel, sejam estas voluntárias ou não, é mais uma regra do que uma exceção; (ii) a natureza dinâmica do deslocamento do hardware e do software na rede global introduz questões relativas tanto à identificação física dos nodos quanto à localização dos componentes de software que migram.

Estas questões apontam para a necessidade de mecanismos dinâmicos que realizem o mapeamento dos componentes móveis, de modo a viabilizar sua localização e permitir a interação com os mesmos. Portanto, o *middleware* para suporte à Computação Ubíqua deve levar em conta essas limitações, de modo que as aplicações não percam sua consistência quando um componente de software migrar entre nodos da rede global, ou quando um nodo temporariamente não estiver disponível por estar desligado ou sem conexão, ou ainda trocar sua célula de execução em função do deslocamento.

Enfim, o *middleware* deverá viabilizar a semântica *siga-me* das aplicações ubíquas. A localização é um aspecto-chave para os sistemas com mobilidade, pois a localidade influi significativamente no contexto disponibilizado para os mecanismos de adaptação. O contexto representa uma abstração peculiar da Computação Ubíqua, e inclui informações sobre recursos, serviços e outros componentes do meio físico de execução. Nesta ótica, o ambiente de execução deve fornecer informações de contexto que extrapolam a localização onde o componente móvel da aplicação se encontra (VENECIAN, 2009).

Suporte a semântica *siga-me*

No EXEHDA, esse suporte é construído pela agregação de funcionalidades relativas ao reconhecimento de contexto, ao acesso pervasivo e à comunicação. Como estratégia para tratamento da complexidade associada ao suporte da semântica *siga-me*, é adotada a decomposição das funcionalidades de mais alto nível, recursivamente, em funcionalidades mais básicas. Nesta perspectiva, o reconhecimento de contexto está relacionado com dois mecanismos: (i) um de monitoração que permite inferir sobre o estado atual dos recursos e das aplicações, e (ii) outro que pode promover adaptações funcionais e não funcionais, tendo em vista o contexto monitorado (YAMIN, 2004).

A adaptação não-funcional consiste na capacidade do sistema atuar sobre a localização física dos componentes das aplicações, seja no momento de uma instanciação do componente, seja, posteriormente, via migração do mesmo. Ambas operações demandam a existência de mecanismo para instalação sob demanda do código, assim como mecanismos para descoberta e alocação dinâmicas de recursos e acompanhamento de seu estado. Por sua vez, a adaptação funcional consiste na capacidade do sistema atuar sobre a seleção da implementação do componente a ser utilizado em um determinado contexto de execução. Novamente, surge a necessidade do suporte à instalação de código sob demanda. A funcionalidade da instalação sob demanda implica que o código a ser instalado esteja disponível em todos os dispositivos nos quais este venha a ser necessário. Considerando as dimensões do ambiente ubíquo, é impraticável manter a cópia de todos os possíveis códigos em todos os eventuais dispositivos. Procede daí a necessidade de um mecanismo que disponibilize acesso ubíquo ao repositório de código, mecanismo este, que deve considerar fortemente o aspecto de escalabilidade. O aspecto de mobilidade, tanto dos componentes das aplicações quanto do usuário, inerente à semântica *siga-me*, faz propícia uma estratégia de comunicação caracterizada pelo desacoplamento espacial e temporal (VENECIAN, 2009).

6.1 Organização do EXEHDA

O *middleware* EXEHDA tem por finalidade definir a arquitetura para um ambiente de execução destinado às aplicações da Computação Ubíqua, no qual as condições de contexto são pró-ativamente monitoradas e o suporte à execução permite que tanto a aplicação como ele próprio utilizem estas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais (VENECIAN, 2009).

O meio físico sobre o qual o EXEHDA é definido constitui-se por uma rede infra-estruturada, cuja composição final pode ser alterada pela agregação dinâmica de nodos móveis. Os recursos da infra-estrutura física são mapeados para três abstrações básicas,

as quais são utilizadas na composição do EXEHDA (YAMIN, 2004):

- EXEHDAnodo: é o elemento organizacional básico de um ambiente EXEHDA. É nele que as aplicações do *middleware* são executadas, portanto, é nele que ocorre a interação entre o sistema e o usuário, bem como a reunião das informações que constituem os contextos de execução. Um subcaso desse tipo de recurso é o *EXEHDAnodo móvel*, caracterizado por ter elevada mobilidade, possuir interface de rede para operação sem fios e, em boa parte dos casos, dispor de capacidade computacional mais restrita.

- EXEHDAbase: é o ponto de contato para os EXEHDAnodos. É responsável por todos os serviços básicos do EXEHDA, além de efetuar a gerência dos EXEHDAnodos geograficamente próximos. Vale notar que, em alguns casos, com o objetivo de garantir escalabilidade, os serviços residentes logicamente em uma *EXEHDAbase* podem residir fisicamente em mais de uma plataforma computacional.

- EXEHDAcél: denota a área de atuação de uma EXEHDAbase, e é composta por esta e por EXEHDAnodos. Os principais aspectos considerados na definição da abrangência de uma célula são: o escopo institucional, a proximidade geográfica e o custo de comunicação;

A figura 6.1 ilustra a organização de um ambiente ubíquo EXEHDA.

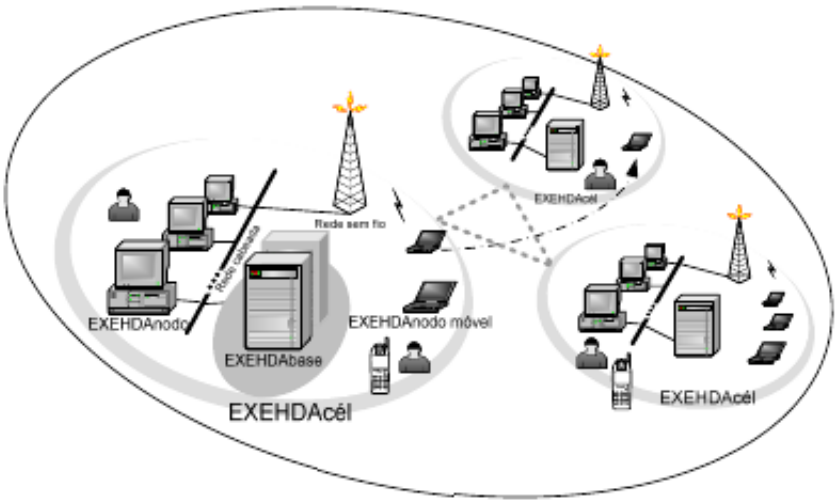

Figura 6.1: Organização do ambiente ubíquo provido pelo EXEHDA (YAMIN, 2004)

6.2 O núcleo do EXEHDA

A funcionalidade provida pelo EXEHDA é personalizável no nível de nodo, sendo determinada pelo conjunto de serviços ativos e controlada por meio de perfis de execução. Um perfil de execução define um conjunto de serviços a ser ativado em um EXEHDA-nodo, associando a cada serviço uma implementação específica dentre as disponíveis, bem como definindo parâmetros para sua execução. Adicionalmente, o perfil de execução também controla a política de carga a ser utilizada para um determinado serviço, a qual se traduz em duas opções: (i) quando da ativação do nodo (*bootstrap* do *middleware*) e (ii) sob demanda.

Desta maneira, a informação definida nos perfis de execução é também consultada quando da carga de serviços sob demanda, assim, a estratégia adaptativa para carga dos serviços acontece tanto na inicialização do nodo, quanto após este já estar em operação e precisar instalar um novo serviço. Esta política para carga dos serviços é disponibilizada por um núcleo mínimo do EXEHDA, o qual é instalado em todo EXEHDAnodo que for integrado ao ambiente. Este núcleo é formado por dois componentes, conforme figura 6.2:

- **ProfileManager**: interpreta a informação disponível nos perfis de execução e a disponibiliza aos outros serviços do *middleware*. Cada EXEHDAnodo tem um perfil de execução individualizado;

- **ServiceManager**: realiza a ativação dos serviços no EXEHDAnodo a partir das informações disponibilizadas pelo ProfileManager. Para isto, carrega sob demanda o código dos serviços do *middleware*, a partir do repositório de serviços que pode ser local ou remoto, dependendo da capacidade de armazenamento do EXEHDA-nodo e da natureza do serviço.

6.3 Serviços do ambiente EXEHDA

Como ilustrado na figura 6.3, os serviços do *middleware* EXEHDA são organizados em quatro grandes subsistemas: execução distribuída, adaptação, comunicação e acesso ubíquo.

Os serviços que compõem cada um desses subsistemas serão listados nas subseções a seguir.

6.3.1 Subsistema de Execução Distribuída

O Subsistema de Execução Distribuída é responsável pelo suporte ao processamento distribuído no EXEHDA. No intuito de promover uma execução efetivamente per-

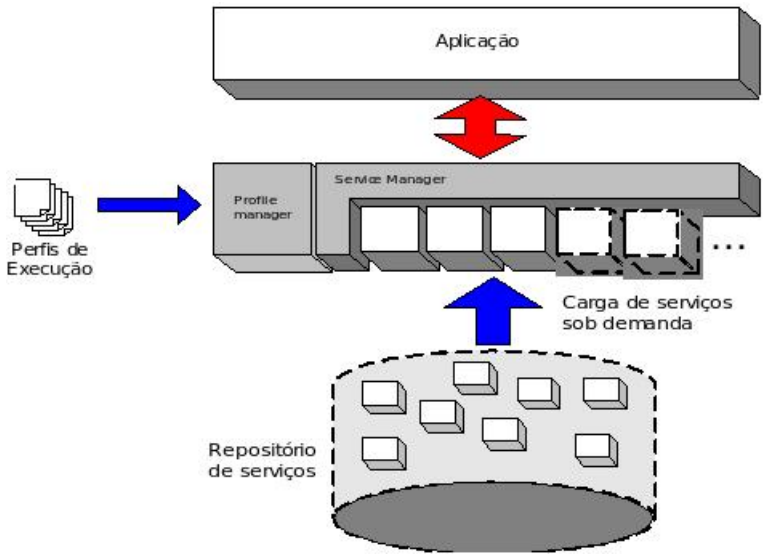

Figura 6.2: Organização do núcleo do EXEHDA (YAMIN, 2004)

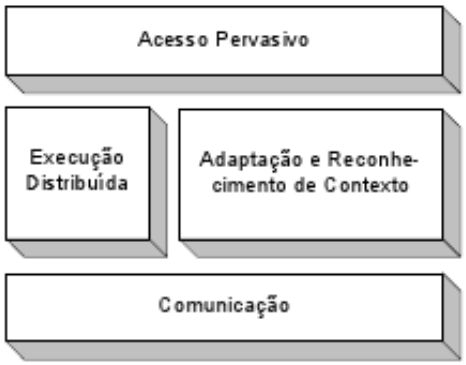

Figura 6.3: Serviços do *middleware* EXEHDA (YAMIN, 2004)

vasiva, este subsistema interage com outros subsistemas do EXEHDA. Em específico, interage com o subsistema de reconhecimento de contexto e adaptação, de forma a prover comportamento distribuído e adaptativo às aplicações EXEHDA (YAMIN, 2004).

Este subsistema compreende os seguintes serviços: (YAMIN, 2004)

- **Executor** - é responsável pelo disparo de aplicações, e de criação e migração de seus objetos.
- **Cell Information Base (CIB)** - implementa a base de informações da célula. Dentre as informações armazenadas, citam-se os dados estruturais da EXEHDAcél, tais como informações sobre os recursos que a compõem, informação de vizinhança, e atributos que descrevem as aplicações em execução, e dados sobre usuários registrados na célula, tais como a sua chave pública codificada em formato x509.
- **OXManager** - é responsável pela gerência e manutenção das meta-informações associadas a ***O**bjectos **EXEHDA***, instanciados pelo serviço *Executor*. O serviço confere às operações de consulta e atualização dos atributos dos OX o necessário caráter pervasivo, permitindo que estas possam ser realizadas a partir de qualquer nodo do EXEHDA.
- **Discoverer** - implementa a descoberta e localização de recursos especializados existentes no EXEHDA, com base em especificações abstratas dos mesmos.
- **ResourceBroker** - gerencia e intermedia o processo de alocação de recursos existentes na EXEHDAcél pelas aplicações.
- **Gateway** - intermedia a comunicação entre nodos externos à EXEHDAcél e os recursos internos a ela.
- **StdStreams** - provê às aplicações o suporte ao redirecionamento dos *streams* padrões de entrada, saída e erro.
- **Logger** - provê às aplicações e aos próprios serviços existentes no EXEHDA uma interface de registro de mensagens de *log*.
- **Dynamic Configurator (DC)** - possibilita realizar configurações de perfis de execução do *middleware* em um determinado EXEHDAnodo de forma automatizada.

6.3.2 Subsistema de Reconhecimento de Contexto e Adaptação

O Subsistema de Reconhecimento de Contexto e Adaptação gerencia o reconhecimento do contexto e a adaptação a ele. O subsistema executa as tarefas de (i) extrair dos

sensores as informações em baixo nível referentes às características dinâmicas e estáticas dos recursos que compõem o EXEHDA, (ii) de identificar essas informações em alto nível dos elementos de contexto e (iii) de disparar ações de adaptação em reação a modificações no estado de tais elementos de contexto (YAMIN, 2004).

Este subsistema compreende os seguintes serviços: (YAMIN, 2004)

- **Collector** - é responsável pela extração da informação bruta (diretamente dos recursos envolvidos) que, posteriormente refinada, dará origem aos elementos de contexto.
- **Deflector** - disponibiliza a abstração de canais de *multicast*, para a disseminação das informações monitoradas.
- **ContextManager** - refina as informações brutas recebidas no processo de monitoração de sensores, produzindo informações em alto nível, referentes aos elementos de contexto. Por exemplo, esse serviço pode receber de um sensor de temperatura um número no intervalo de 0 a 100 e fornecer como resposta um parâmetro de estado, entre 4 valores válidos: *quente*, *morno*, *frio* ou *gelado*.
- **AdaptEngine** - gerencia o controle das adaptações de cunho funcional. O serviço provê facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações, liberando o programador de gerenciar os aspectos de mais baixo nível envolvidos na definição e liberação dos elementos de contexto junto ao *ContextManager*.
- **Scheduler** - é o serviço central na gerência de adaptações de cunho não-funcional. Com base nas informações obtidas a partir do *Collector*, este serviço inicia operações de instanciação e migração de objetos que ele julgar necessárias.

6.3.3 Subsistema de Comunicação

A natureza da mobilidade de hardware e software dificulta o estabelecimento pleno de comunicação entre elementos. As desconexões são comuns, não somente devido à existência de *links* sem fio, mas também porque a desconexão é uma estratégia capaz de economizar significativamente a energia gasta nos dispositivos móveis. O Subsistema de Comunicação, considerando tais aspectos, tem o objetivo de permitir que os componentes e aplicações do EXEHDA possam comunicar-se (YAMIN, 2004).

Este subsistema compreende os seguintes serviços: (YAMIN, 2004)

- **Dispatcher** - disponibiliza o modelo de comunicação mais elementar do EXEHDA: troca de mensagens ponto-a-ponto e transiente, com garantia de entrega e ordenamento de mensagens.

- **Worb** - disponibiliza, com a utilização do serviço *Dispatcher*, uma interface para o suporte a chamada de procedimentos remotos (RPC).

- **CCManager** - fornece o suporte a um mecanismo de comunicação com suporte a desacoplamento temporal e espacial e, também, a outros padrões de comunicação além do ponto-a-ponto.

6.3.4 Subsistema de Acesso Pervasivo

A premissa de acesso em qualquer lugar, todo o tempo, a dados e código da Computação Pervasiva, requer um suporte do *middleware*. No EXEHDA, o Subsistema de Acesso Pervasivo é responsável pelo provimento desse suporte (YAMIN, 2004).

Este subsistema compreende os seguintes serviços: (YAMIN, 2004)

- **BDA - Base de Dados pervasiva das Aplicações** - para que o EXEHDA possa fornecer às aplicações a facilidade de adaptação funcional, é necessária a existência de um repositório de código que seja capaz de hospedar todas as funcionalidades dessas aplicações, além de uma infra-estrutura para o suporte a instalação de código sob demanda. Essas são as finalidades do serviço *BDA*.

- **AVU - Ambiente Virtual do Usuário** - a premissa *siga-me* reflete-se não só nas aplicações que um usuário atualmente executa, mas no seu ambiente computacional como um todo - as informações de personalização das aplicações definidas, o conjunto de aplicações instaladas e, também, os seus arquivos privados. É atribuição do serviço *AVU* a manutenção do acesso pervasivo a este ambiente virtual, da forma mais eficiente possível.

- **SessionManager** - é responsável pela gerência da sessão de trabalho do usuário, sendo definida pelo conjunto de aplicações concorrentemente em execução para aquele usuário.

- **Gatekeeper** - intermedia acessos entre as entidades externas ao EXEHDA e os serviços do *middleware* de execução, conduzindo os procedimentos de autenticação necessários.

6.4 Considerações sobre o capítulo

Neste capítulo, foram exploradas as características e funcionalidades do *middleware* EXEHDA, descrevendo cada um dos serviços implementados em sua arquitetura de software.

O EXEHDA é um *middleware* direcionado às aplicações distribuídas, móveis e conscientes do contexto da Computação Ubíqua. Este *middleware* é baseado em serviços,

que tem por objetivo criar e gerenciar um ambiente ubíquo. Seu núcleo é formado por dois componentes, que são o *ProfileManager* e o *ServiceManager*. Seus serviços estão organizados em 4 subsistemas, que são o de Execução Distribuída, o de Reconhecimento de Contexto e Adaptação, o de Comunicação e o de Acesso Pervasivo.

O capítulo seguinte apresentará o EXEHDA-SS, serviço do EXEHDA o qual este Projeto de Graduação contribuiu para modelagem e implementação.

7 EXEHDA-SS: MODELAGEM E IMPLEMENTAÇÃO

O objetivo central deste Projeto de Graduação foi colaborar com o desenvolvimento do EXEHDA-SS - *framework* direcionado ao *middleware* EXEHDA para suporte à sensibilidade ao contexto na Computação Ubíqua, em tempo de execução. O trabalho foi desenvolvido dentro do G3PD, Grupo de Pesquisa em Processamento Paralelo e Distribuído, na Universidade Católica de Pelotas.

O EXEHDA-SS é um componente que realiza tarefas relacionadas a captura, processamento e notificação, contemplando o emprego de ontologia na representação de elementos internos. Essas tarefas compreendem aquisição de contexto, representação de informações contextuais, tradução, raciocínio e inferência sobre as informações, persistência de dados e remessa de notificações.

Sua implementação é feita com a linguagem Java e é dividido nos seguintes módulos: o **módulo de sensoriamento** e o **módulo de sensibilidade ao contexto**, que são descritos nas seções a seguir.

7.1 Módulo de sensoriamento

O módulo de sensoriamento é o componente que, quando executado no EXEHDAnodo, é responsável por fornecer uma interface para a ativação, configuração e captura de informações dos sensores do nodo. As tarefas executadas dentro do módulo de sensoriamento são as seguintes:

- Verificar, dentre os sensores existentes no EXEHDAnodo, quais estão plenamente funcionais e podem ser ativados;
- Configurar os sensores de acordo com os parâmetros recebidos a partir de sua interface;
- Adquirir informações sensoriais e enviá-las para o módulo de sensibilidade ao con-

texto em execução na EXEHDAbase. O envio pode ser de forma periódica ou a pedido do módulo de sensibilidade ao contexto.

O módulo de sensoriamento é formado pelas classes Java básicas: (i) *ExehdaSS_sensor*, que tem a finalidade de escutar a porta 10666/tcp do EXEHDAnodo e instanciar objetos *ExehdaSS_atendimento* para cada conexão de entrada aceita nessa porta; (ii) *ExehdaSS_atendimento*, que efetua o atendimento de cada conexão de entrada na porta TCP que é escutada; (iii) *ExehdaSS_sensorCliente*, que conecta-se periodicamente ao módulo de sensibilidade ao contexto em execução no EXEHDAbase para lhe enviar informações produzidas por todos os sensores ativos; (iv) *ExehdaSS_coletordelixo*, que tem o objetivo de remover da memória objetos *ExehdaSS_atendimento* que se encontrarem em estado não operacional e (v) *Sensores.Sensor*, que oferece a abstração básica de um sensor.

Na fase atual da implementação, o módulo de sensoriamento contempla, dentre outros, sensores de uso de processador, uso de memória, temperatura e uso de espaço em disco. Uma das metas de desenvolvimento do módulo de sensoriamento do EXEHDA-SS é a de estender o conjunto de sensores implementados de maneira a levantar todas as características de hardware do EXEHDAnodo tais como rede, áudio, periféricos de entrada e saída e outros. As características dos sensores já existentes no EXEHDA-SS são discutidos a seguir.

7.1.1 Sensor de processamento

O sensor de processamento captura o número de núcleos de processamento, bem como a porcentagem de utilização e a freqüência de *clock* de cada um desses núcleos. Duas implementações diferentes de sensor de processamento foram avaliadas no trabalho: a implementação *Sensores.ProcImpl.CpuSensorProcImpl*, que extrai informações a partir do pseudo-arquivo */proc/stat* (HAT, 2009a) e *Sensores.SigarImpl.CpuSensorSigarImpl*, que utiliza a biblioteca Hyperic's System Information Gatherer (HYPERIC, 2009) para a captura das informações.

7.1.2 Sensor de memória

O sensor de memória captura a quantidade disponível e total de memória física e virtual (swap) do sistema. Duas implementações diferentes de sensor de memória foram avaliadas no trabalho: a implementação *Sensores.ProcImpl.MemorySensorProcImpl*, que extrai informações a partir do pseudo-arquivo */proc/meminfo* (HAT, 2009b) e *Sensores.SigarImpl.MemorySensorSigarImpl*, que utiliza a biblioteca Hyperic's System Information Gatherer (HYPERIC, 2009) para a captura dinâmica das informações.

7.1.3 Sensor de temperatura

O sensor de temperatura captura as informações dos sensores de temperatura instalados na *motherboard* do EXEHDAnodo. Apenas uma implementação de sensor foi avaliada no trabalho: a implementação *Sensores.CmdImpl.TemperatureSensorCmdImpl*, que extrai informações a partir do utilitário *lm-sensors* (LM_SENSORS, 2009).

7.1.4 Sensor de uso de espaço em disco

O sensor de uso de espaço em disco captura a quantidade disponível e total de espaço existente na partição que hospeda o módulo de sensoriamento. Duas implementações diferentes de sensor de memória foram avaliadas no trabalho: a implementação *Sensores.CmdImpl.DiskSensorCmdImpl*, que extrai informações a partir do utilitário *GNU stat* (FOUNDATION, 2009) e *Sensores.SigarImpl.DiskSensorSigarImpl*, que utiliza a biblioteca Hyperic's System Information Gatherer (HYPERIC, 2009) para a captura das informações.

7.2 Módulo de sensibilidade ao contexto

O módulo de sensibilidade ao contexto, quando executado na EXEHDAbase, é o componente responsável pelas seguintes tarefas: (i) instanciar e manipular a ontologia base do EXEHDA-SS, que armazena e indexa o conhecimento referente aos objetos existentes no ambiente ubíquo, tais como EXEHDAnodos, informações contextuais, aplicações, localizações geográficas e perfis de usuário; (ii) estabelecer comunicação com os EXEHDAnodos ativos na EXEHDAcél e extrair as informações sensoriadas disponíveis; (iii) traduzir e deduzir informações contextuais; (iv) gerenciar registros de contextos de interesse de aplicações e; (v) perceber alterações de contexto e remeter notificações de alteração às aplicações sensíveis ao contexto modificado.

7.2.1 Arquitetura básica

A figura 7.1 ilustra a arquitetura do módulo de sensibilidade ao contexto do EXEHDA-SS (VENECIAN, 2009). Nela, são representados os sensores, de onde informações contextuais são capturadas, e a aplicação que solicita ser notificada no instante da modificação do contexto de seu interesse.

(VENECIAN, 2009) O servidor de contexto tem por objetivo gerenciar os contextos de interesse das aplicações (CIA), que são o conjunto de parâmetros de publicação que definem que e de maneira informações contextuais serão adquiridas pelos sensores, traduzidas e deduzidas. O servidor também possui a responsabilidade adicional de perceber os contextos que sofreram modificações e remeter às aplicações notificações de alteração

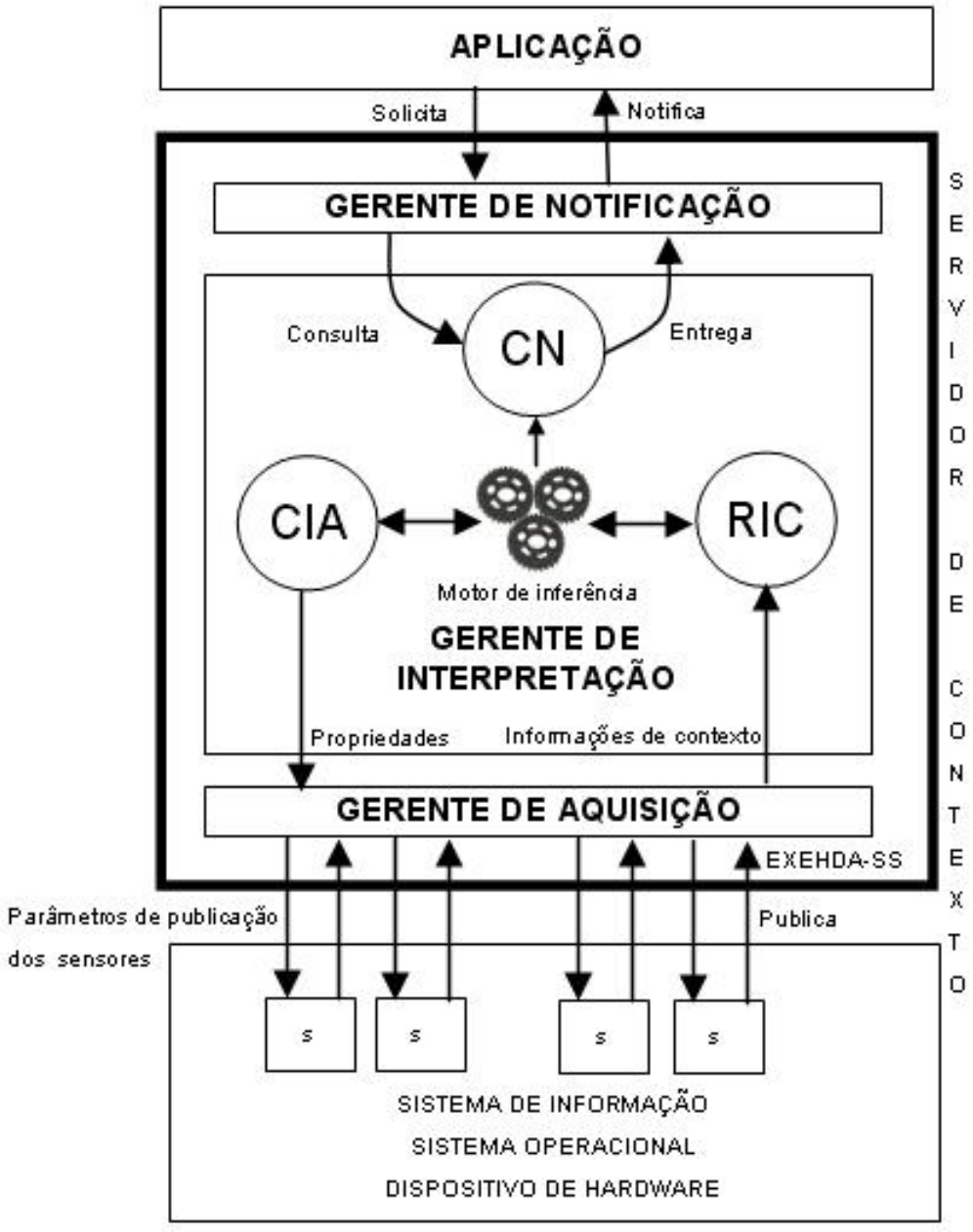

Figura 7.1: Visão geral da arquitetura do módulo de sensibilidade do EXEHDA-SS

de contexto.

No EXEHDA-SS, as informações contextuais são armazenados e acessados no Repositório de Informações Contextuais (RIC) e o histórico de notificações de alteração de contexto é armazenado no Contexto Notificado (CN).

O servidor de contexto é composto por três gerentes:

- **Gerente de Aquisição de Contexto:** exerce um papel fundamental na arquitetura, fornecendo ao Gerente de Interpretação uma abstração sobre como as informações contextuais são capturadas, para, posteriormente, serem processadas e manipuladas. Também é responsabilidade do Gerente de Aquisição a tarefa de ativar e configurar os componentes de sensoriamento, de acordo com as propriedades estabelecidas no contexto de interesse das aplicações.

- **Gerente de Interpretação de Contexto:** tem como principal função avaliar cada contexto em função das informações contextuais capturadas e peceber quando um determinado contexto estiver ativo ou as mudanças que ocorrerem no seu estado. Também é responsável por traduzir as informações contextuais de acordo com regras passadas pela aplicação e produzir informações contextuais adicionais via procedimentos de dedução.

- **Gerente de Notificação:** é responsável por entregar os contextos processados pelo servidor de contexto à aplicação sensível ao contexto. Essa entrega pode ser feita, basicamente, de maneira ativa ou passiva. A entrega ativa é realizada como resposta a uma consulta enviada pela aplicação ao gerente de notificação, solicitando diretamente a informação contextual desejada. A entrega passiva ocorre por iniciativa do servidor de contexto - nesse caso, a aplicação efetua uma inscrição prévia perante o gerente de notificação, posicionando-se como interessada em determinados tipos de informação contextual e, então, sempre que essa informação for modificada, o gerente envia à aplicação uma mensagem de notificação.

7.2.2 Implementação

Na fase atual da implementação do módulo de sensibilidade ao contexto do EXEHDA-SS, as seguintes classes Java foram definidas:

- *ExehdaSS_coletor* - é a classe principal do módulo de sensibilidade ao contexto;

- *ExehdaSS_escutaporta* - tem a finalidade de escutar a porta 20666/tcp da EXEHDAbase e instanciar objetos *ExehdaSS_atende* para cada conexão de entrada aceita nessa porta;

- *ExehdaSS_atende* - efetua o atendimento de cada conexão de entrada na porta TCP que é escutada;

- *ExehdaSS_cliente* - tem a finalidade de conectar-se periodicamente aos módulos de sensoriamento em execução nos EXEHDAnodos para extrair destes as informações produzidas pelos seus sensores ativos;

- *ExehdaSS_coletalixo* - tem o objetivo de remover da memória os objetos *ExehdaSS_atende* que se encontrarem em estado não operacional;

- *ExehdaSS_individuo* - atualiza o conhecimento armazenado na ontologia base do EXEHDA-SS com as informações sensoriadas que são trazidas pelas classes *ExehdaSS_cliente* e *ExehdaSS_atende*;

- *ExehdaSS_gerInterpretacao* - implementação do Gerente de Interpretação; avalia contextos armazenados na ontologia, identificando ativações e alterações, traduz informações contextuais de acordo com regras de tradução específicas e gera informações contextuais adicionais via procedimentos de dedução;

- *ExehdaSS_gerNotificacao* - implementação do Gerente de Notificação; é a classe responsável por enviar as notificações de alteração de contexto às aplicações.

A classe *ExehdaSS_coletor* é responsável por inicializar toda a infra-estrutura necessária para o funcionamento dos mecanismos definidos no módulo de sensibilidade ao contextodo EXEHDA-SS. Ela é responsável por efetuar a leitura do arquivo OWL que descreve a OntUbi (descrita na subseção 7.2.3), bem como do arquivo XML que descreve cada um dos EXEHDAnodos existentes na base (descrito na subseção 7.2.4). Além disso, é sua tarefa instanciar um Gerente de Interpretação, um Gerente de Notificação e, periodicamente, gravar em um arquivo OWL o conhecimento armazenado na OntUbi.

7.2.3 Representação das informações contextuais do ambiente ubíquo

(VENECIAN, 2009) O EXEHDA-SS utiliza na sua arquitetura de software uma representação ontológica definida na OntUbi. Na OntUbi, Ontologia do Ambiente Ubíquo, é modelado o ambiente ubíquo de execução do EXEHDA-SS. As instâncias dos Contextos de Interesses das Aplicações, contextos coletados e notificados estão definas pela OntContext. Desta forma a OntContext está contida na OntUbi.

Para a representação das informações contextuais propostas para o EXEHDA-SS, foram concebidos dois conjuntos de classes e subclasses de caracterização de elementos. A seguir, serão apresentadas a OntUbi e a OntContext.

7.2.3.1 OntUbi

A OntUbi é a ontologia do ambiente ubíquo, nelas estão representadas o ambiente ubíquo de execução do EXEHDA-SS. As classes definidas na OntUbi, bem como o relacionamento entre elas, pode ser visualizado na figura 7.2.

7.2.3.2 OntContext

A OntContext é o subconjunto de classes da OntUbi que tem a responsabilidade de representar os contextos coletados, notificados e os contextos de interesses das aplicações. Compõe as seguintes classes da OntContext (vide figura 7.3):

Classes de Contexto de Interesse das Aplicações

- *Aplicações*: Contem a identificador e descrição.
- *Componentes*: Identificador e descricao.
- *Atributos*: Possui identificador, descrição, intervalor de medicação dos sensores, taxa de flutuação mínima dos dados a serem publicados pelos sensores, valor inferior e superior e a regra para tradução de dados coletados.
- *Sensor*: Identifica o sensor pedido pela aplicação.

Classes para Coleta e Notificação de Contextos

- *Contexto*: Armazena os dados sensorados pelo sensores como: nodo, data/hora, sensor, valor e usuario.
- *Contexto_Notificado*: Contém os contextos notificados, com o identificador, aplicacao, componente, usuario e nodo.
- *Contexto_Notificado_valor*: Possui os sensores e valores respectivos.

7.2.4 XML de descrição de nodos

Na fase atual de implementação do EXEHDA-SS, o módulo de sensibilidade ao contexto identifica a localização dos EXEHDAnodos a partir de um arquivo em formato XML, que é lido na fase de inicialização da classe *ExehdaSS_coletor*. O formato básico do arquivo é ilustrado na figura 7.4.

Dentro do elemento raiz (<**EXEHDA-SS**>) são definidos zero ou mais elementos-filhos <**Nodo**>, que tem a finalidade de descrever os dispositivos e periféricos de *hardware* existentes em cada nodo. Os nomes e os atributos dos elementos que são declarados abaixo de <**Nodo**> seguem os nomes de classes e relacionamentos que são definidos na OntUbi:

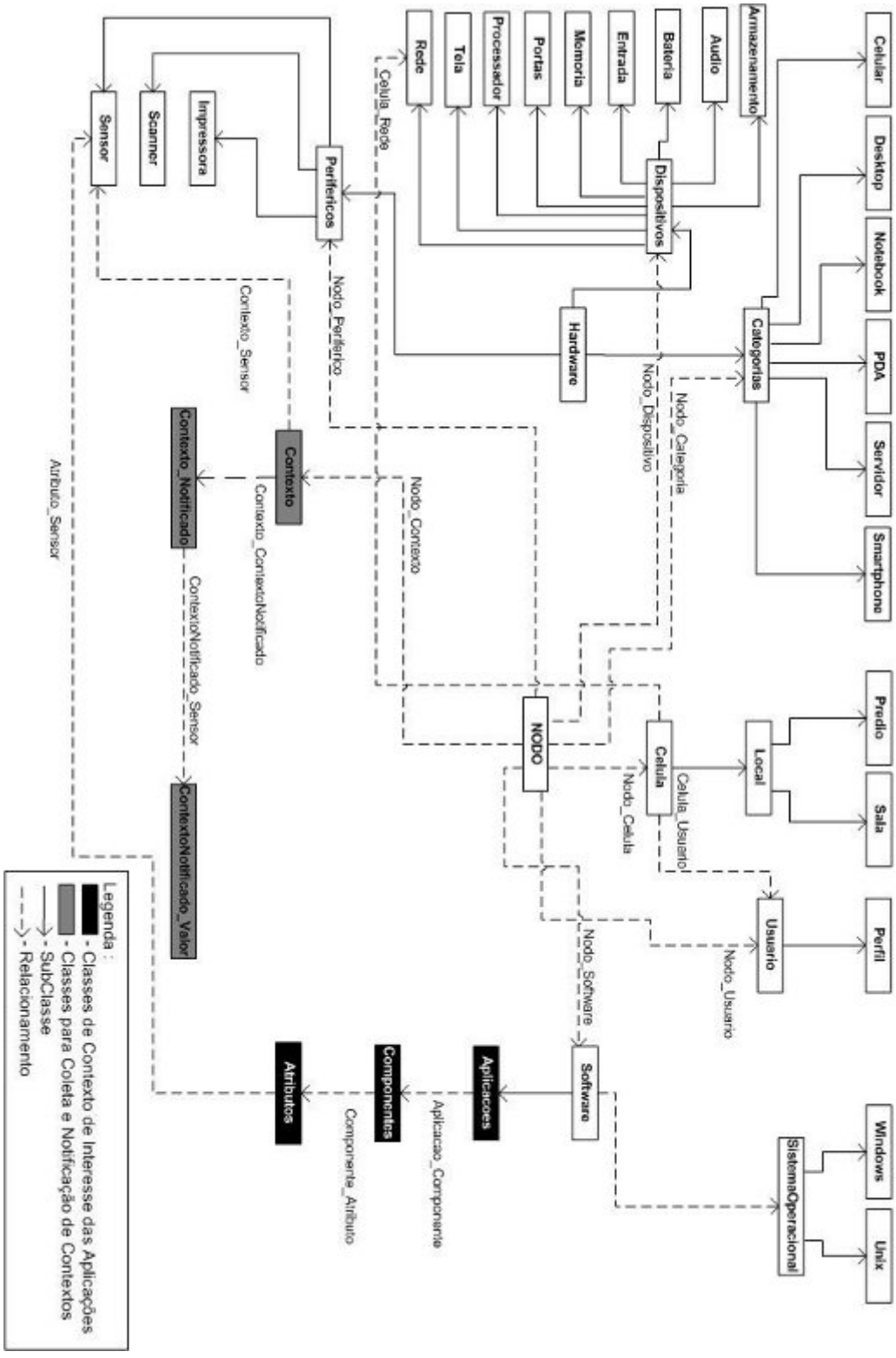

Figura 7.2: Diagrama de classes e relacionamentos da OntUbi

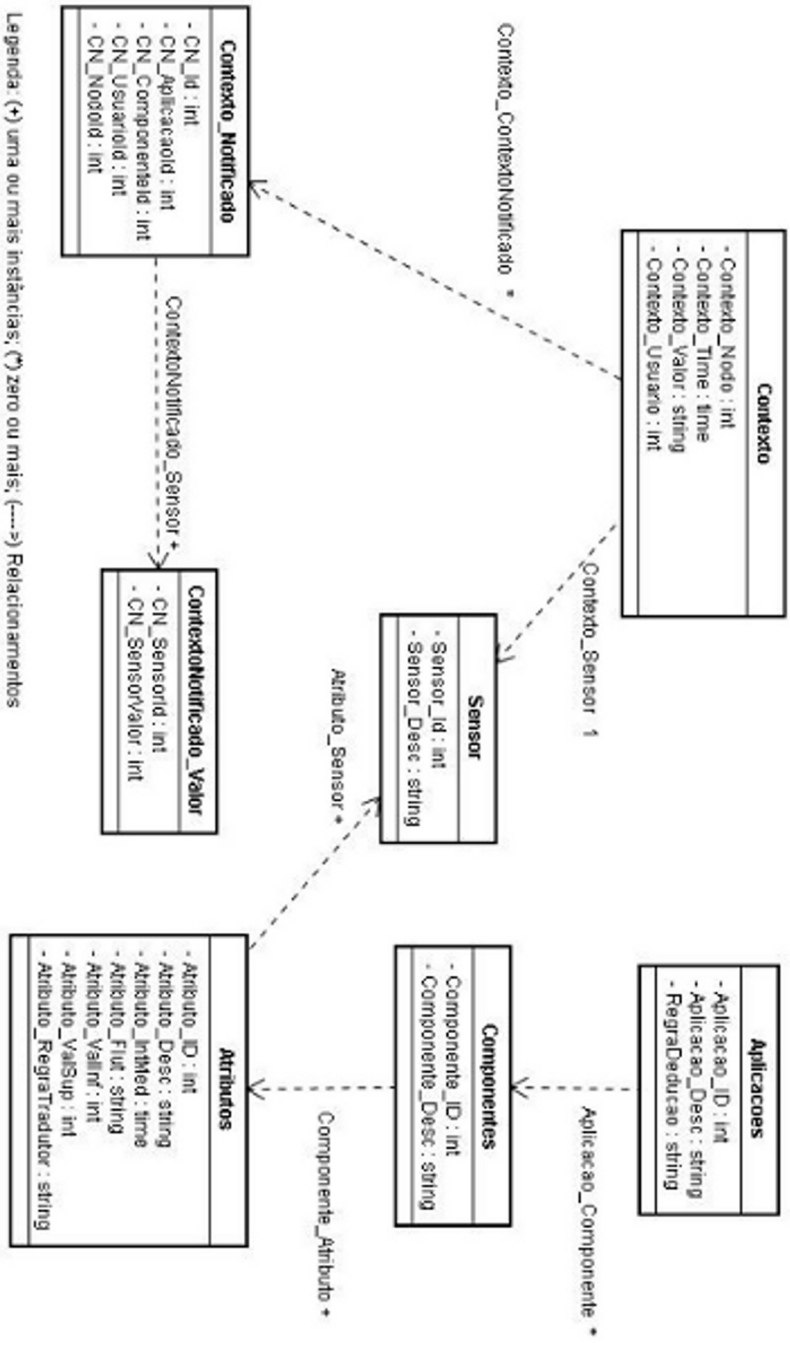

Figura 7.3: Diagrama de classes da OntContext

```
<EXEHDA-SS>

 <Nodo ID="localhost" Nodo_Base="true" Nodo_Movel="true" Nodo_Status="Ativo">
  <Nodo_Dispositivos>
   <Intel Processador_Cores="1" Processador_Unidade="MHz" Processador_Vel="2800" />
   <Memoria Memoria_Capacidade="4096" Memoria_Padrao="DDR" Memoria_Unidade="MB" />
  </Nodo_Dispositivos>
  <Nodo_Perifericos>
   <Laser Impressora_Cor="false" Impressora_Velocidade="400" Perifericos_Status="Ativo" />
   <Sensor ID="Sensor_12" Perifericos_Status="Inativo" />
   <Sensor ID="Sensor_13" Perifericos_Status="Ativo" />
   <Sensor ID="Sensor_14" Perifericos_Status="Inativo" />
   <Sensor ID="Sensor_15" Perifericos_Status="Inativo" />
  </Nodo_Perifericos>
 </Nodo>

 <Contexto Contexto_Valor="115" Contexto_Times="2009-10-15T17:37:57" Contexto_Nodo="localhost"
           Contexto_Sensor="Sensor_2" Contexto_Aplicacao="Appl" />

 <Contexto Contexto_Valor="32" Contexto_Times="2009-10-16T22:58:01" Contexto_Nodo="localhost"
           Contexto_Sensor="SensorIdade" Contexto_Aplicacao="Appl" />

 <Contexto Contexto_Valor="90" Contexto_Times="2009-10-20T22:16:14" Contexto_Nodo="localhost"
           Contexto_Sensor="Sensor_10" Contexto_Aplicacao="Appl" />

 <Contexto Contexto_Valor="92" Contexto_Times="2009-10-18T18:37:57" Contexto_Nodo="localhost"
           Contexto_Sensor="Sensor_10" Contexto_Aplicacao="Appl" />

 <Contexto Contexto_Valor="95" Contexto_Times="2009-10-19T22:16:14" Contexto_Nodo="localhost"
           Contexto_Sensor="Sensor_10" Contexto_Aplicacao="Appl" />

</EXEHDA-SS>
```

Figura 7.4: Arquivo XML de descrição de nodos

- Seja uma propriedade *P* que relaciona um objeto *A* a um objeto *B*. No arquivo XML, são definidos um elemento <**A**>, contendo <**P**> como elemento filho e um elemento <**B**> que terá <**P**> como seu pai. Exemplo: na figura 7.4, é ilustrada a propriedade **P="Nodo_Perifericos"**, que relaciona o objeto **A="localhost"** da classe *Nodo* a um objeto **B="Sensor_12"** da classe *Sensor*.

- Seja uma propriedade *P* que relaciona um objeto *A* a um literal *C*. No arquivo XML, é definido o elemento <**A P="C"**>, isto é, o elemento <**A**> com o atributo *P* definido para o valor *C*. Exemplo: na figura 7.4, é ilustrada a propriedade **P="Perifericos_Status"**, que relaciona o objeto **A="Sensor_12"** da classe *Sensor* ao valor literal com o valor **C="Inativo"**.

Com o objetivo de verificar o fluxo de execução do EXEHDA-SS no interior do Gerente de Interpretação, a especificação do arquivo XML de descrição de nodos é estendida de maneira a permitir a definição de contextos. Para isso, são definidos dentro do elemento raiz zero ou mais elementos-filhos <**Contexto**>. Esses elementos serão instanciados na OntUbi assim que os dados do nodo que é referenciado no atributo **Contexto_Nodo** são lidos.

7.3 Fluxo de execução do EXEHDA-SS

Uma seqüência de etapas é percorrida a partir do momento da instanciação de uma informação contextual até o envio da notificação de alteração de contexto a ser remetido a uma aplicação.

1. Em um primeiro momento, os dados vindos dos sensores deverão alcançar o objeto da classe *ExehdaSS_individuo*. Isso é feito de duas maneiras diferentes:

   - **Por iniciativa do módulo de sensoriamento**: depois de vencido o período de inatividade, o objeto da classe *ExehdaSS_sensorCliente* captura e empacota os dados dos sensores ativos e, a seguir, estabelece uma conexão ao módulo de sensibilidade. Este, ao receber a conexão, instancia um objeto da classe *ExehdaSS_atende* para receber as informações dos sensores e, a seguir, remetê-las ao objeto da classe *ExehdaSS_individuo*.
   - **Por iniciativa do módulo de sensibilidade**: depois de vencido o período de inatividade, o objeto da classe *ExehdaSS_cliente* conecta-se ao módulo de sensoriamento e solicita informações contextuais. Este, ao receber a conexão, instanciará um objeto da classe *ExehdaSS_atendimento*, que capturará e empacotará os dados vindos dos sensores ativos. Depois que os dados empacotados são transferidos de *ExehdaSS_atendimento* para *ExehdaSS_cliente*, este último finaliza a conexão e remete os dados que foram recebidos ao objeto da classe *ExehdaSS_individuo*.

2. Assim que o objeto da classe *ExehdaSS_individuo* identifica a presença de informações sensoriadas em sua entrada, ele grava os dados no Repositório de Informações Contextuais e notifica o Gerente de Interpretação, avisando que o modelo ontológico foi modificado.

3. A seguir, o Gerente de Interpretação consulta o banco de Contexto de Interesse das Aplicações e identifica se as informações contextuais que acabaram de ser gravadas na OntUi fazem parte do contexto de interesse de alguma aplicação. Em caso afirmativo, o gerente produzirá notificações de alteração de contexto, gravará tais notificações no repositório de Contexto Notificado e notificará o Gerente de Notificação, avisando que o repositório foi alterado.

4. Por fim, o Gerente de Notificação obtém a missão de notificar as aplicações.

A fase atual da implementação oferece suporte para registro em arquivo de *log* do procedimento de notificação, bem como o registro na OntContext das informações de subscrição da aplicação. Outrossim, a OntUbi é instanciada com dados gerados pelos

sensores nstalados no ambiente ubíquo. As informações de subscrição das aplicações são utilizadas para atuação do sensoriamento e disparo de notificações.

7.4 Considerações sobre o capítulo

Este capítulo apresentou o trabalho que foi desenvolvido no Projeto de Graduação.

O trabalho consistiu na modelagem e no desenvolvimento inicial do EXEHDA-SS, *framework* de sensibilidade ao contexto para o EXEHDA. A arquitetura do *framework* contempla o módulo de sensoriamento e o módulo de sensibilidade. Este último contempla o Gerente de Aquisição de Contexto, o Gerente de Interpretação de Contexto e o Gerente de Notificação, que cooperarão entre si para prover notificações de alteração de contexto às aplicações em execução no *middleware* EXEHDA que estiverem interessadas em recebê-las.

O capítulo seguinte contempla algumas considerações finais, bem como caracteriza metas futuras de desenvolvimento.

8 CONSIDERAÇÕES FINAIS

Neste capítulo, são resumidas as principais conclusões obtidas no desenvolvimento deste Projeto de Graduacao, bem como são caracterizadas possíveis oportunidades para desdobramentos deste trabalho.

8.1 Principais conclusões

O texto contemplou considerações conclusivas ao longo dos diversos capítulos. Esta seção tem por objetivos registrar os aspectos mais gerais sobre o trabalho que foi realizado no Projeto de Graduação.

A Computação Ubíqua é o terceiro grande paradigma computacional, precedido pelo império dos *mainframes* e pela onda da computação pessoal. Nesse paradigma, a computação é onipresente e imperceptível; sistemas computacionais estarão presentes em todos os lugares e a todo momento, integrados ao ambiente do usuário e interagindo naturalmente com estes. Porém, a infra-estrutura para o desenvolvimento e os mecanismos de auxílio em tempo de execução são, neste momento, focos a serem pesquisados dentro da área da Computação Ubíqua. Soluções genéricas para o desenvolvimento e uso de aplicações ubíquas ainda não são disponíveis.

O reconhecimento da importância do contexto motivou pesquisadores de diversas áreas da computação, como Inteligência Artificial, Interface Homem-Máquina, Computação Ubíqua, Engenharia de Software, Banco de dados e Sistemas Colaborativos, a estudar conceitos de sensibilidade ao contexto e entender como a mesma pode ser formalizada e utilizada nos sistemas computacionais. Na Computação Sensível ao Contexto, é investigado o emprego de informações que caracterizam a situação de uma interação usuário-computador no sentido de fornecer serviços adaptados a usuários e aplicações. Dois grandes desafios lidados ao se desenvolver um sistema sensível ao contexto são o de delimitar as ações dependentes de contexto nesses sistemas e o de identificar os elementos contextuais que caracterizam a situação na qual essas ações são executadas. Estudos mostram que a identificação dos elementos de contexto depende fortemente do tipo da

tarefa e domínio em questão.

As características do módulo de gerência de contexto das aplicações, bem como o processo de adaptação dinâmica, são questões que ainda não foram resolvidas como um todo na área de Computação Ubíqua. Os trabalhos existentes, referentes à adaptação ao contexto, apresentam aspectos semelhantes, tais como gerenciamento da largura de banda, do formato do conteúdo e de mapeamento físico (replicação ou migração).

O *middleware* EXEHDA tem por finalidade definir a arquitetura para um ambiente de execução destinado às aplicações da Computação Ubíqua, no qual as condições de contexto são pró-ativamente monitoradas e o suporte à execução permite que tanto a aplicação como ele próprio utilizem estas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. O meio físico sobre o qual o EXEHDA é definido constitui-se por uma rede infra-estruturada, cuja composição final pode ser alterada pela agregação dinâmica de nodos móveis.

O objetivo deste trabalho de Projeto de Graduação foi a concepção do EXEHDA-SS, *framework* direcionado ao *middleware* EXEHDA para suporte à sensibilidade ao contexto na Computação Ubíqua, em tempo de execução e que realiza tarefas relacionadas a captura, processamento e notificação, contemplando o emprego de ontologia na representação de elementos internos. O trabalho foi desenvolvido dentro do G3PD, Grupo de Pesquisa em Processamento Paralelo e Distribuído, na Universidade Católica de Pelotas.

8.2 Trabalhos futuros

Identificam-se como trabalhos a serem realizados futuramente no EXEHDA-SS os seguintes items:

- Concluir a incorporação do EXEHDA-SS à arquitetura do EXEHDA.
- Implementar sensores adicionais no módulo de sensoriamento, que permitam sensoriar todo o *hardware* acoplado a um EXEHDAnodo.
- Modelar o mecanismo de notificação de alteração de contextos.
- Modelar uma API para a gerência dos Contextos de Interesse das Aplicações.

REFERÊNCIAS

ABOWD, G. D.; MYNATT, E. D.; RODDEN, T. **IEEE Pervasive Computing**. [S.l.]: The human experience, 2002.

AFONSO, F. A. **GRADEp-SC - Sensibilidade ao Contexto no Middleware GRADEp**. 2007. Monografia PGII — Universidade Católica de Pelotas, Pelotas/RS.

ARAUJO, R. B. de. **Computação Ubíqua**: Princípios, Tecnologias e Desafios. [S.l.]: Simposio Brasileiro de Redes de Computadores, 2003. 1-71p.

AUGUSTIN, I. **Abstrações para uma Linguagem de Programação Visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing**. 2004. Tese de Doutorado em Ciência da Computação — Instituto de Informática/UFRGS, Porto Alegre,RS.

BARDRAM, J. E. **The Java Context Awareness Framework (JCAF) A Service Infrastructure and Programming Framework for Context-Aware Applications**. [S.l.]: Springer, 2005. 98–115p. (Lecture Notes in Computer Science, v.3468).

BELOTTI, R.; DECURTINS, C.; GROSSNIKLAUS, M.; NORRIE, M. C.; PALINGINIS, A. **Modelling Context for Information Environments**. [S.l.]: In: Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), 2004.

BRICKLEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0**: RDF Schema. World Wide Web Consortium, Recommendation REC-rdf-schema-20040210.

CHEN, G.; KOTZ, D. **A Survey of Context-Aware Mobile Computing Research**. Dartmouth College: Department of Computer Science, 2002.

DEY, A.; ABOWD, G.; SALBER, D. **A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications**. [S.l.]: Human-Computer Interaction, 2001. 97-166p.

DEY, A. K. **Providing Architectural Support for Building Context-Aware Applications.** [S.l.]: Georgia Institute of Technology, 2000.

DOMINGUES, F. L. **Computação Ubíqua**. Disponível em: <http://www.guiadohardware.net/artigos/computacao-ubiqua/>. Acesso em Abril de 2009.

FENSEL, D. **Ontologies-Silver Bullet for Knowledge Management and Electronic Commerce**. Berlin: [s.n.], 2000.

FOUNDATION, F. S. **14.3 stat**: Report file or file system status. Disponível em: <http://www.gnu.org/software/coreutils/manual/html_node/stat-invocation.html>. Acesso em Novembro de 2009.

GAUVIN, M.; BOURRY-BRISSET, A. C.; AUGER, A. **Context, Ontology and Portfolio**: Key Concepts for a Situational Awareness Knowledge Portal. [S.l.]: In: Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

GRIMM, R.; BERSHAD, B. N. **System Support for Pervasive Applications**. [S.l.]: Springer, 2003. 212–217p. (Lecture Notes in Computer Science, v.2584).

GU, T.; PUNG, H. K.; ZHANG, D. A service-oriented middleware for building context-aware services. **J. Network and Computer Applications**, [S.l.], v.28, n.1, p.1–18, 2005.

HAT, R. **5.2.27. /proc/stat**. Disponível em: <http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/Reference_Guide/s2-proc-stat.html>. Acesso em Novembro de 2009.

HAT, R. **5.2.19. /proc/meminfo**. Disponível em: <http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/Reference_Guide/s2-proc-meminfo.html>. Acesso em Novembro de 2009.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. **Modeling context information in pervasive computing systems**. Zurich, Switzerland: PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, 2002. 167-180p.

HYPERIC. **Hyperic's System Information Gatherer**. Disponível em: <http://www.hyperic.com/products/sigar.html>. Acesso em Novembro de 2009.

JARDIM, A. D. **Resource Description Framework (RDF)**. Disponível em: <http://ia.ucpel.tche.br/ lpalazzo/Aulas/IWS/m03/IntroRDF.pdf>. Acesso em Maio de 2009.

KORPIPAA, P.; MÄNTYJÄRVI, J.; KELA, J.; KERÄNEN, H.; MALM, E. **Managing Context Information in Mobile Devices**. [S.l.]: IEEE Pervasive Computing, 2003.

LM_SENSORS. Disponível em: <http://www.lm-sensors.org/>. Acesso em Novembro de 2009.

MACIEL, R. S. P.; ASSIS, S. R. **Middleware**: Uma solução para o desenvolvimento de aplicações distribuídas. [S.l.]: In: Científico - Ano IV, 2004.

MCBRIDE, B. **An Introduction to RDF and the Jena RDF API**. Disponível em: <http://jena.sourceforge.net/tutorial/RDF_API/index.html>. Acesso em Novembro de 2008.

MOREIRA BUBLITZ frederico. **Infra-estrutura para o Desenvolvimento de Aplicações Cientes de Contexto em Ambientes Pervasivos**. 2007. Tese de Mestrado em Informática — Centro de Engenharia Elétrica e Informática / Universidade Federal de Campina Grande, Campina Grande,PB.

MOSTEFAOUI, G. K.; ROCHA, J. P.; BREZILLON, P. **Context-Aware Computing**: A Guide for the Pervasive Computing Community. Beirute, Libano: Proceedings of the 2004 IEEE/ACS International COnference on Pervasive Services, 2004.

OZTURK, P.; AAMODT, A. **Towards a Model of Context for Case-Based Diagnostic Problem Solving**. [S.l.]: In: Proceedings of the interdisciplinary conference on modeling and using context (Context- 97), 1997. 198-208p.

OZTURK, P.; AAMODT, A. **Context as a Dynamic Construct**. [S.l.]: Human Computer Interaction, 2001. 257-268p.

RANGANATHAN, A.; CAMPBELL, R. H. **A Middleware for Context-Aware Agents in Ubiquitous Computing Environments**. [S.l.]: In: ACM/IFIP/USENIX International Middleware Conference, 2003.

ROSA, M. G. P.; BORGES, M. R. S.; SANTORO, F. M. **A Conceptual Framework for Analyzing the Use of Context in Groupware**. [S.l.]: In: Proc. of CRIWG'03, v. LNCS 2806, pp. 300-313, Springer Verlag Berlin, 2003.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence**. [S.l.]: A Modern Approach, 2003.

SCHILIT, B. **A Context-Aware Systems Architecture for Mobile Distributed Computing**. Columbia University: Ph.D. Thesis, 1995.

SCHILIT, B.; ADAMS, N.; WANT, R. **Context-aware computing applications**. Santa Cruz, California: In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, 1994. 85-90p.

STRANG, T.; LINNHOFF-POPIEN, C. **A context modeling survey**. Nottingham, England: PROCEEDINGS OF THE I INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2004. 34-41p.

TRUONG, K. N.; ABOWD, G. D.; BROTHERTON, J. A. **Who, What, When, Where, How**: Design issues of capture access applications. [S.l.]: Proceedings of the International Conference on Ubiquitous Computing, 2001.

VENECIAN, L. R. **EXEHDA-SS**: Serviço para Sensibilidade ao Contexto com Suporte Semântico. 2009. Dissertação Final — Universidade Católica de Pelotas, Pelotas/RS.

WANG, X. H.; GU, T.; ZHANG, D. Q.; PUNG, H. K. **Ontology based context modeling and reasoning using OWL**. [S.l.]: In: Workshop on Context Modeling and Reasoningat II IEEE International Conference on Pervasive Computing and Communication, 2004.

WANT, R.; PERING, T. **System challenges for ubiquitous & pervasive computing**. [S.l.]: ACM, 2005. 9–14p.

WARKEN, N. **Uma Contribuição ao Controle da Adaptação Na Computação Ubíqua**. 2008. Trabalho Individual — Universidade Católica de Pelotas, Pelotas/RS.

WEISER, M. The computer for the 21st century. **Scientific American**, [S.l.], v.265, n.3, p.94–104, 1991.

WEISSENBERG, N.; VOISARD, A.; GARTMANN, R. **Using ontologies in personalized mobile applications**. [S.l.]: ACM, 2004. 2–11p.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionada às Aplicações Distribuídas, Móveis e Conscientes de Contexto da Computação Pervasiva**. 2004. Tese de Doutorado em Ciência da Computação — Instituto de Informática/UFRGS, Porto Alegre-RS.

YAMIN, A. C.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. **EXEHDA-Adaptative Middleware for Building a Pervasive Grid Environment**. Amsterdam: [s.n.], 2005. 203-219p. v.135.