UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Um Mecanismo de Sensibilidade ao Contexto com Suporte Semântico para Computação Ubíqua

por
Luthiano Rodrigues Venecian

Dissertação apresentada como
requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof. Dr. Adenauer Corrêa Yamin

DMII-2010/1-004

Pelotas, março de 2010

Dedico aos meus pais,
pelo apoio, incentivo e compreensão durante o desenvolvimento deste trabalho.

AGRADECIMENTOS

Agradeço a Deus autor da vida, pela providência divina, força e saúde.

A meu orientador, Prof. Dr. Adenauer Yamin, pela sua amizade, orientação e motivação durante esse período de convivência, que em muitas situações foram essências para minha permanência e conclusão deste curso de mestrado.

Ao meu irmão, Jean Venecian, por seu incentivo na realização deste trabalho.

Aos professores, funcionários e colegas do PPGINF, em especial à Nelsi e ao Dilli, por serem presentes em todos os momentos desta jornada.

Ao Prof. João Ladislau, pela sua amizade e acompanhamento desde os meus primeiros passos na informática.

Ao Prof. Cleber Telles e ao Escritório Administrativo da Renovação Carismática Católica do Brasil pela confiança depositada.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

E a todos os amigos que contribuíram de uma forma direta ou indireta.

SUMÁRIO

LISTA DE FIGURAS 7

LISTA DE TABELAS 9

LISTA DE ABREVIATURAS E SIGLAS 10

RESUMO 12

ABSTRACT 13

1 INTRODUÇÃO 14
1.1 Tema 15
1.2 Escopo da pesquisa 15
1.2.1 Projeto PERTMED 16
1.2.2 *Middleware* EXEHDA 16
1.3 Motivação 16
1.4 Objetivos 17
1.5 Estrutura do texto 18

2 SENSIBILIDADE AO CONTEXTO: PRINCIPAIS CONCEITOS 19
2.1 Definição de Contexto 19
2.2 Conceitos em Computação Sensível ao Contexto 20
2.2.1 Identificação dos Elementos de Contexto 21
2.2.2 Características das Informações Contextuais 22
2.2.3 Dimensões de Informação de Contexto 23
2.2.4 Aquisição de Contexto 25
2.2.5 Modelagem de Contexto 26
2.2.6 Interpretação de Contexto 27
2.2.7 Processamento e Raciocínio sobre o Contexto 28
2.2.8 Armazenamento de Informações Contextuais 28
2.3 Considerações Sobre o Capítulo 29

3 TRABALHOS RELACIONADOS 30
3.1 *Context Management System* 30
3.2 *Context Toolkit* 31
3.3 *Middleware* de Contexto do Gaia 32
3.4 *Social Philanthropic Information Environment* 34
3.5 *Context Aware Mobile Networks and Services* 35

3.6 ***Service-Oriented Context-Aware Middleware*** . . . 36
3.7 ***Context Broker Architecture*** . . . 38
3.8 ***Mobile Collaboration Architecture*** . . . 39
3.9 ***Framework* de Contexto** . . . 40
3.10 ***Semantic Context Kernel*** . . . 41
3.11 **Infraware** . . . 42
3.12 **Considerações Sobre o Capítulo** . . . 44

4 EXEHDA-SS: FUNDAMENTOS . . . 47
4.1 **Ontologias** . . . 47
4.1.1 O Conceito de Ontologia . . . 47
4.1.2 Linguagens para Representação de Ontologia . . . 48
4.2 **Projeto PertMed** . . . 50
4.3 ***Middleware* EXEHDA: Revisão Arquitetural e Funcional** . . . 52
4.3.1 Premissas de Pesquisa do EXEHDA . . . 52
4.3.2 Organização do EXEHDA . . . 54
4.3.3 Subsistemas do EXEHDA . . . 58
4.4 **Considerações Sobre o Capítulo** . . . 59

5 EXEHDA-SS: CONCEPÇÃO E MODELAGEM . . . 60
5.1 **Modelo de Representação de Contexto** . . . 60
5.1.1 OntUbi . . . 61
5.1.2 OntContext . . . 64
5.2 **Modelagem da Arquitetura de Software** . . . 65
5.2.1 Gerente de Aquisição de Contexto . . . 69
5.2.2 Gerente de Interpretação de Contexto . . . 74
5.2.3 Gerente de Notificação . . . 77
5.3 **Considerações Sobre o Capítulo** . . . 78

6 EXEHDA-SS: TECNOLOGIAS UTILIZADAS E ESTUDO DE CASO . . . 79
6.1 **Principais Tecnologias Utilizadas** . . . 79
6.1.1 Java . . . 79
6.1.2 JSF . . . 80
6.1.3 API Jena . . . 80
6.1.4 Linguagem de Consulta SPARQL . . . 81
6.1.5 Protégé . . . 82
6.2 **Estudo de Caso: Cenário Direcionado à Medicina** . . . 82
6.2.1 Objetivos da AUP . . . 82
6.2.2 Configuração do EXEHDA-SS pela AUP . . . 83
6.2.3 Contexto Deduzido para Envio Automático de Mensagens . . . 88
6.2.4 Configuração dos Sensores nos EXEHDAnodos . . . 88
6.2.5 Publicação de Informações Contextuais ao EXEHDA-SS . . . 89
6.2.6 Processamento e Notificação de Contextos Obtidos pelo EXEHDA-SS . . . 90
6.3 **Considerações Sobre o Capítulo** . . . 96

7 CONSIDERAÇÕES FINAIS 97
7.1 Principais Contribuições 98
7.2 Discussão dos Trabalhos Relacionados ao EXEHDA-SS 99
7.3 Trabalhos Futuros 100
7.4 Publicações Realizadas 101

REFERÊNCIAS 102

ANEXO A APLICAÇÃO AUP - TELA EM PDA 107

LISTA DE FIGURAS

Figura 2.1 Definição de (DEY, 2000) 20
Figura 2.2 Definição de (CHEN G., 2002) 20

Figura 3.1 Arquitetura do CXMS (ZIMMERMANN A., 2005a) 31
Figura 3.2 Hierarquia de Componentes e Arquitetura do CTK (DEY, 2000) . . . 32
Figura 3.3 Visão Geral do *Middleware* de Contexto do GAIA (RANGANATHAN A., 2003) 33
Figura 3.4 Modelo de Contexto (BELOTTI, 2004) 34
Figura 3.5 Arquitetura do SOPHIE (BELOTTI, 2004) 35
Figura 3.6 Gerenciamento de Dados no Awareness (WEGDAM, 2005) 36
Figura 3.7 Visão Parcial da Ontologia de Alto Nível do CONON (WANG X. H., 2004) 37
Figura 3.8 Visão Geral da Arquitetura do SOCAM (GU T., 2004) 38
Figura 3.9 Visão Geral da Arquitetura do CoBrA (CHEN H., 2005) 39
Figura 3.10 Serviços da Arquitetura Moca (SACRAMENTO et al., 2004) 40
Figura 3.11 Arquitetura do *Framework* de Contexto (HENRICKSEN K., 2005a) . 41
Figura 3.12 Visão Geral da Arquitetura do SCK (BULCAO NETO R. F., 2005) . . 42
Figura 3.13 Visão Geral do Modelo de Contexto (BULCAO NETO R. F., 2005) . 43
Figura 3.14 Plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006) . . . 44

Figura 4.1 Arquitetura de Software *Middleware* EXEHDA (YAMIN, 2004) . . . 55
Figura 4.2 Ambiente ubíquo provido pelo EXEHDA (YAMIN, 2004) 56
Figura 4.3 Organização dos Subsistemas do EXEHDA (YAMIN, 2004) 57
Figura 4.4 Organização do Núcleo do EXEHDA (YAMIN, 2004) 58

Figura 5.1 Diagrama de Classes, Subclasses e Relacionamentos da OntUbi . . . 63
Figura 5.2 Classes da OntContext 66
Figura 5.3 Integração do EXEHDA-SS ao Subsistema de Adaptação e Reconhecimento de Contexto do *Middleware* EXEHDA [adaptado de (YAMIN, 2004)] 67
Figura 5.4 Visão Geral da Arquitetura de Software do EXEHDA-SS 68
Figura 5.5 Classes da OntContext para Operação dos Sensores 71
Figura 5.6 Exemplo de Configuração do Arquivo SensorConfiguracao 71
Figura 5.7 Exemplo de Configuração do Arquivo SensorPublicacao 72
Figura 5.8 Tratamento dos Dados Sensoreados 73
Figura 5.9 Exemplo de Identificação de Contextos de Interesse nas Publicações Realizadas pelos Sensores 73

Figura 5.10 Exemplo de Definição de Regras 76
Figura 5.11 Fluxo do Motor de Inferência do EXEHDA-SS 77
Figura 5.12 Exemplo de Notificação as Aplicações 78

Figura 6.1 Configuração dos Sensores para Monitoramento de Pacientes e Reconhecimento de Dispositivos na AUP 88
Figura 6.2 Publicação Realizada pelo Sensor de Frequência Cardíaca 89
Figura 6.3 Publicação Realizada pelo Sensor de Temperatura 89
Figura 6.4 Publicação Realizada pelo Sensor de Pressão Alta 89
Figura 6.5 Nível de Alerta 1 - Versão Desktop 91
Figura 6.6 Nível de Alerta 2 - Versão Desktop 92
Figura 6.7 Nível de Alerta 3 - Versão Desktop 92
Figura 6.8 Nível de Alerta 4 - Versão Desktop 93
Figura 6.9 Nível de Alerta 1 e 2 - Versão PDA 93
Figura 6.10 Nível de Alerta 3 e 4 - Versão PDA 94

LISTA DE TABELAS

Tabela 2.1 Avaliação das Abordagens para Modelagem de Contexto (STRANG T; LINNHOFF-POPIEN, 2004) 27

Tabela 3.1 Modelo de Representação Contextual 45
Tabela 3.2 Informações Contextuais 45

Tabela 5.1 Classes Java do Servidor de Contexto do EXEHDA-SS 69

Tabela 6.1 Valor Traduzidos na AUP 85
Tabela 6.2 Faixas Operacionais do Sensor de Frequência Cardíaca para AUP . . 86
Tabela 6.3 Faixas Operacionais do Sensor de Temperatura para AUP 86
Tabela 6.4 Faixas Operacionais do Sensor de Pressão Alta para AUP 87
Tabela 6.5 Sensor de Reconhecimento de Dispositivos para a AUP 87
Tabela 6.6 Regra de Dedução - Risco de Infarto 88
Tabela 6.7 Classe Contexto 90
Tabela 6.8 Classe Contexto_Notificado - Monitoramento de Pacientes 94
Tabela 6.9 Classe ContextoNotificado_Sensor 95
Tabela 6.10 Classe Contexto_Notificado - Reconhecimento de Dispositivo 95
Tabela 6.11 Classe ContextoNotificado_Sensor 95
Tabela 6.12 Classe Contexto_Notificado - Contexto Deduzido 96
Tabela 6.13 Classe ContextoNotificado_Deduzido 96

Tabela 7.1 Comparação do EXEHDA-SS com outros trabalhos relacionados (capítulo 3) 99

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CML	Context Modelling Language
DOM	Document Object Model
EXEHDA	Execution Environment for Highly Distributed Applications
Foaf	Friend of a Friend
GSM	Global System for Mobile Communications
GTSH	Gator Tech Smart House
HCI	Human Computer Interaction
HTML	Hiper Text Markup Language
ISAM	Infra-estrutura de Suporte às Aplicações Móveis Distribuídas
OML	Ontology Markup Language
OPEN	Ontology-Driven Pervasive Environment
ORM	Object-Role Modeling
OSGi	Open Service Gateway Initiative
OWL	Web Ontology Language
PDA	Personal Digital Assistant
RCN	Repositório de Contexto Notificado
RDF	Resource Description Language
RIC	Repositório de Informação Contextual
RFID	Radio Frequency Identification
SQL	Structured Query Language
SO	Sistema Operacional
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SWS	Serviços Web Semânticos
UML	Unified Modeling Language

URI	Universal Resource Identifier
WASP	Web Architectures for Services Platforms
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XOL	Ontology Exchange Language

RESUMO

Este trabalho tem como objetivo central a proposição de um mecanismo para sensibilidade ao contexto na computação ubíqua. Com os avanços tecnológicos temos dispositivos cada vez menores e com maior poder de computação e comunicação o que potencializa a mobilidade do usuário quando portando seus equipamentos. Neste sentido, um Ambiente Ubíquo pressupõe a existência de diferentes dispositivos, tais como sensores, atuadores e eletroeletrônicos em geral que interagem de diferentes maneiras com os usuários. A diversidade de dispositivos e informações de um Ambiente Ubíquo assim constituído, introduz diferentes desafios para interoperabilidade entre as diferentes partes envolvidas. Portanto, ao se construir e executar aplicações ubíquas sensíveis ao contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de informações contextuais, a partir do conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, e seu processamento. Na perspectiva de atender estas demandas da computação ubíqua, foi concebido o EXEHDA-SS, para ser responsável pelo tratamento das informações contextuais, realizando tarefas de manipulação e dedução sobre o contexto, utilizando ontologias para a representação e processamento das informações contextuais empregando suporte semântico.

Palavras-chave: Sensibilidade ao Contexto, Suporte Semântico, Medicina Ubíqua.

TITLE: “A MECHANISM OF CONTEXT-AWARENESS WITH SUPPORTED SEMANTIC FOR UBIQUITOUS COMPUTING”

ABSTRACT

This work was aimed at the proposition of a mechanism for context awareness in ubiquitous computing. With technological advances we have smaller and smaller devices and greater computing power and communication which enhances the mobility of the User when carrying their equipment. In this sense, a ubiquitous environment requires the existence of different devices such as sensors, actuators and electronics in general that interact in different ways with users. The diversity of devices and information in a ubiquitous environment as it is, introduces different challenges for interoperability between the different parties involved. Therefore, when building and running applications ubiquitous context-aware, a number of features that should be provided, involving the acquisition of contextual information from the set of heterogeneous sources and distributed to the representation of the information and its processing . With a view to meet these demands of ubiquitous computing, is designed to EXEHDA-SS, to be responsible for processing the contextual information for tasks of manipulation and deduction from the context using ontologies for the representation and processing of contextual information using semantic support.

Keywords: Context-Aware, Semantic Support, Medical Ubiquitous.

1 INTRODUÇÃO

A Computação Ubíqua (COSTA; YAMIN; GEYER, 2008), é uma forma de computação onde o processamento está espalhado no ambiente através de vários dispositivos, que executam tarefas bem definidas dependendo de sua natureza, interligados de forma que essa estrutura exija pouco gerenciamento por parte do usuário. Aplicações ubíquas executam em ambientes instrumentados com sensores, geralmente dotados de interfaces de redes sem fio, nos quais dispositivos, agentes de software e serviços são integrados de forma transparente e cooperam para atender aos objetivos da aplicação. Essa categoria de aplicações caracteriza-se por constantes mudanças em seu estado de execução, geradas pelos ambientes altamente dinâmicos em que executam.

A Sensibilidade ao Contexto refere-se à capacidade de um sistema computacional perceber características de seu ambiente, e é um requisito chave para permitir a adaptação em resposta às mudanças ambientais. Aplicações sensíveis ao contexto conhecem o ambiente no qual estão sendo utilizadas e tomam decisões de acordo com mudanças no seu próprio ambiente. Ou seja, reagem a ações executadas por outras entidades, podendo essas ser pessoas, objetos ou até mesmo outros sistemas, que modifiquem o ambiente. Essas aplicações, de um modo geral, tomam ciência de modificações que venham a acontecer no ambiente. Tais modificações denomina-se alterações nas informações de contexto.

Com o avanço recente da computação móvel, a computação ubíqua pode fazer uso de dispositivos móveis para que sistemas estejam cada vez mais centrados nos usuários, cientes das freqüentes variações das informações de contexto que são inerentes a esses sistemas. Como exemplo de dispositivos móveis podemos citar os *handhelds* e *smartphones* que, além de dispor um maior poder computacional cada vez maior, utilizam redes sem fio para se comunicarem com outros dispositivos ou com a Internet.

Um ambiente ubíquo tem uma natureza dinâmica, devido à mobilidade do usuário, a variedade de dispositivos e tecnologias existentes, assim como às mudanças constantes no ambiente computacional (ZHOU Y.; CAO, 2007). Para fornecer suporte ao dinamismo do ambiente ubíquo, requer a definição das suas regras de comportamento em tempo de execução (WALTENEGUS, 2006).

Ao se construir e executar aplicações ubíquas sensíveis ao contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de informações contextuais, a partir do conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, seu processamento, armazenamento, e a realização de inferências para seu uso em tomadas de decisão. Registra-se a tendência atual de remover estas funcionalidades, repassando as mesmas para *middlewares* de provisão de contexto (HENRICKSEN K., 2005a).

Essa dissertação de mestrado contempla esforços de pesquisa na área de medicina ubíqua, a principal razão desta inserção é o fato da área médica, estar sendo alvo de avanços das tecnologias móveis e sem fio, como *Bluetooth*, *WiFi*, *GPRS*, os quais somados a popularização dos dispositivos móveis e sem fio, *PDAs*, celulares, *GPS* e pequenos dispositivos médicos, como *holters*, entre outros, facilitam a tarefa de monitoramento de pacientes. Infra-estruturas de software para o gerenciamento dessas informações contextuais necessitam, em geral, coletar uma grande quantidade de informações de diferentes naturezas do ambiente, analisando essas informações como variáveis independentes, ou combiná-las com outras informações do passado ou presente. Além disso, essas aplicações são caracterizadas por apresentarem contextos altamente dinâmicos e variados, com um grande grau de mobilidade dos seus principais atores (médicos, pacientes, paramédicos, etc.).

Na perspectiva de suprir estas funcionalidades, este trabalho tem como objetivo principal propor a integração de tecnologias de suporte semântico em mecanismo de sensibilidade ao contexto, desde a sua aquisição, processamento e distribuição das informações contextuais, direcionado a Computação Ubíqua. O mecanismo proposto utiliza o ambiente ubíquo promovido pelo *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) (YAMIN, 2004).

O EXEHDA-SS está sendo concebido para ser responsável pelo tratamento das informações contextuais, realizando tarefas de manipulação e dedução sobre o contexto, utilizando ontologias para suporte a representação e processamento das informações contextuais. Através do uso de inferências espera-se garantir um refinamento qualificado dessas informações capturas e distribuídas nas células de execução do EXEHDA.

Este capítulo apresenta o tema do trabalho e o escopo da pesquisa, destaca as motivações e objetivos do trabalho, bem como descreve a estrutura do texto como um todo.

1.1 Tema

Este trabalho tem como tema central a concepção de um mecanismo para sensibilidade ao contexto com suporte semântico. Este mecanismo será direcionado ao atendimento das demandas inerentes a computação ubíqua, e no tocante a sua avaliação, foi desenvolvida aplicação da área médica.

O desenvolvimento deste trabalho compreende estudos que visam sistematizar a comparação de diferentes mecanismos de sensibilidade ao contexto, através da exploração da relação existente entre computação ubíqua, sensibilidade ao contexto e tecnologias para suporte semântico.

A previsão é que sejam desenvolvidas para a avaliação do mecanismo proposto aplicações na área de medicina ubíqua.

1.2 Escopo da pesquisa

O escopo da pesquisa desta dissertação tem como referência os dois projetos resumidos a seguir.

1.2.1 Projeto PERTMED

O sistema de saúde do futuro prevê o uso de tecnologias da computação ubíqua formando um espaço inteligente (reativo e pró-ativo), onde dispositivos móveis e fixos estão integrados ao ambiente físico (objetos) visando captar informações do meio e transmitir as alterações detectadas para sistemas de gerenciamento de informações, os quais tomarão decisões e adaptar-se-ão às situações detectadas (computação consciente de contexto).

O projeto PERTMED (PERTMED, 2007) propõe fazer a ponte entre os sistemas automatizados existentes (registro de pacientes, exames laboratoriais, etc.) e o médico no local em que este se encontra , por exemplo em regiões remotas ou em trânsito. Desta forma, elimina-se a exigência de estar-se conectado a uma rede fixa e com um computador pessoal na área do hospital para ter acesso às informações do paciente.

O projeto PERTMED prevê o uso do EXEHDA como *middleware* direcionado a computação ubíqua. Neste sentido, este trabalho busca atender as demandas do projeto PERTMED através do emprego semântico no mecanismo de sensibilidade ao contexto do EXEHDA.

1.2.2 *Middleware* EXEHDA

O EXEHDA é um *middleware* adaptativo ao contexto e baseado em serviços que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução, sob este ambiente, das aplicações que expressam a semântica siga-me. Estas aplicações são distribuídas, móveis e adaptativas ao contexto em que seu processamento ocorre, estando disponíveis a partir de qualquer lugar, todo o tempo.

Para atender a elevada flutuação na disponibilidade dos recursos, inerente à computação ubíqua, o EXEHDA é estruturado em um núcleo mínimo e em serviços carregados sob demanda. Os principais serviços fornecidos estão organizados em subsistemas que gerenciam: a execução distribuída, a comunicação, o reconhecimento do contexto, a adaptação, o acesso ubíquo aos recursos e serviços, a descoberta e o gerenciamento de recursos.

No EXEHDA, as condições de contexto devem ser pró-ativamente monitoradas e o suporte à execução deve permitir que tanto a aplicação como ele próprio utilizem essas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. O mecanismo de adaptação do EXEHDA emprega uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar contextos de interesse das aplicações para reger o comportamento de cada um dos componentes que constituem o software da aplicação (YAMIN, 2004).

1.3 Motivação

Aplicações ubíquas são tipicamente sensíveis ao contexto. Essas aplicações recebem dados produzidos pelo ambiente computacional e ações de usuários e contemplam um alto suporte a mobilidade (COSTA; YAMIN; GEYER, 2008). Nessas aplicações, informações de contexto relativas ao usuário, ambiente e localização tendem a mudar com freqüência e, consequentemente, eventos contextuais emergem de forma concorrente e dinâmica, fazendo-se necessário o uso da arquitetura baseada em eventos.

O serviço de contexto é responsável por entregar mudanças de contexto aos cli-

entes que se subscreveram para as mudanças de contexto relacionadas. A concorrência e dinamicidade de eventos contextuais podem ser exemplificados com o estudo de caso apresentado em (AL., 2006) onde um paciente acometido com uma séria doença cardíaca pode ter suas funções cardíacas monitoradas através de sensores e, caso seja identificado um estado preocupante, pessoas da família, seu médico e até mesmo uma ambulância podem ser notificados. Um exemplo de uma situação preocupante seria caso os sensores identificassem que a pressão sanguínea e a quantidade de batimentos por minuto de seu coração encontram-se em uma faixa perigosa. Desse modo, mecanismos de comunicação baseada em eventos que provêem suporte para composição de eventos concorrentes permitem uma maior expressividade na declaração de interesses.

Pode-se resumir que a principal motivação para esta dissertação é atender as demandas introduzidas pela crescente complexidade dos contextos modernos, aos quais as aplicações ubíquas estão submetidas. Partindo-se da premissa que é possível qualificar o processamento destes contextos com o emprego de suporte semântico. Motivação esta alinhada com as premissas operacionais do *middleware* EXEHDA.

1.4 Objetivos

O objetivo geral deste trabalho é explorar a correlação entre computação ubíqua, sensibilidade ao contexto e tecnologias para Suporte Semântico. A solução que está sendo proposta é denominada EXEHDA-SS, e será validada através do atendimento de demandas da medicina ubíqua.

Deste modo, o EXEHDA-SS foi concebido considerando o histórico de pesquisas do grupo no assunto, contemplando o uso de suporte semântico no suporte a execução de aplicações sensíveis ao contexto.

Como objetivos específicos desta dissertação, destacaríamos:

- resumir os fundamentos teóricos sobre computação ubíqua e computação sensível ao contexto;
- sistematizar a partir da literatura as plataformas sensíveis ao contexto em ambientes de execução para computação ubíqua;
- avaliar o emprego das tecnologias para prover suporte semântico;
- explorar a correlação entre computação ubíqua, sensibilidade ao contexto e tecnologias para suporte semântico;
- avaliar o *middleware* EXEHDA, revisando seus fundamentos e as decisões inerentes a concepção dos diversos módulos de sua arquitetura;
- propor modelo para integração de suporte semântico em um mecanismo de sensibilidade ao contexto a ser integrado ao *middleware* EXEHDA;
- difundir o conhecimento pertinente à área de suporte semântico à execução de aplicações na Computação Ubíqua, sobretudo no que diz respeito aos aspectos de processamento de contextos;
- perseguir a integração com grupos que trabalham com Computação Ubíqua no cenário nacional;

- fornecer subsídios para a elaboração de relatórios, artigos e trabalhos futuros, relacionados com o tema pesquisado.

A discussão destes objetivos, bem como os conceitos, as metodologias e técnicas para sua concepção estão registrados no texto cuja estrutura é apresentada a seguir.

1.5 Estrutura do texto

A estrutura do texto deste trabalho contempla sete capítulos, sendo o primeiro esta introdução, e os outros seis organizados em um crescente de especificidade conforme resumos abaixo:

- Capítulo 2: Sensibilidade ao Contexto: Principais Conceitos, são conceitualizados os principais conceitos da computação sensível ao contexto;
- Capítulo 3: Trabalhos Relacionados, são apresentados a avaliação de onze trabalhos a sensibilidade ao contexto e suas interligações;
- Capítulo 4: Fundamentos do EXEHDA-SS, apresenta os conceitos em Ontologias, Projeto PertMed e uma revisão arquitetural do *middleware* EXEHDA;
- Capítulo 5: Concepção e Modelagem do EXEHDA-SS, são tratados aspectos da modelagem da arquitetura de software do EXEHDA-SS, sendo discutidos aspectos referentes a concepção, bem como o modelo ontológico para representação contextual;
- Capítulo 6: Tecnologias Utilizadas e Estudo de Caso do EXEHDA-SS, são apresentandas as principais tecnologias empregadas no EXEHDA-SS e o estudo de caso para avaliação das funcionalidades concebidas pelo EXEHDA-SS;
- Capítulo 7: Considerações Finais, são apresentadas as principais contribuições pertinentes deste trabalho, discussão dos trabalhos relacionados ao EXEHDA-SS, trabalhos futuros e publicações realizadas.

2 SENSIBILIDADE AO CONTEXTO: PRINCIPAIS CONCEITOS

Este capítulo resume conceitos inerentes a área central do trabalho desenvolvido. Foi fruto de um estudo em abrangência que contemplou tanto aspectos de fundamentação científica, como aqueles pertinentes ao uso da sensibilidade ao contexto.

Nas 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. Contexto ajuda a melhorar a qualidade de conversação e a compreender certas situações, ações ou eventos.

Contexto desempenha um papel importante em qualquer domínio que envolva requisitos como compreensão, raciocínio, resolução de problemas ou aprendizado (SANTORO F. M., 2005). Contexto é uma importante ferramenta para apoiar a comunicação entre pessoas e sistemas computacionais, pois ajuda a diminuir ambigüidade e conflitos, aumenta a expressividade dos diálogos, e possibilita a melhoria dos serviços e informações oferecidos pela aplicação. Com isso, a tendência é que as aplicações se tornem mais amigáveis, flexíveis e fácies 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.

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. Este capítulo, estabelece as bases teóricas do trabalho, apresentando os conceitos fundamentais relacionados ao tema “contexto”.

2.1 Definição de Contexto

A palavra “contexto” no dicionário Houaiss (HOUAISS, 2007) significa a “inter-relação de circunstâncias que acompanham um fato ou uma situação”. Por mais que essa definição forneça uma noção geral do significado de contexto, 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.

Alguns pesquisadores, com o intuito de limitar a abrangência desse conceito, propuseram definições referentes ao mesmo. A seguir, três visões bastante conhecidas referentes a contexto são apresentadas.

Como referência clássica na área, Schilit (SCHILIT, 1995) (SCHILIT B.N., 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 e temperatura.

Além disso, (CHEN G., 2002) defende a inclusão do Tempo (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 citação histórica de (DEY, 2000), vide figura 2.1:

> *"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."*

Figura 2.1: Definição de (DEY, 2000)

(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 G., 2002) na figura 2.2 define 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."*

Figura 2.2: Definição de (CHEN G., 2002)

2.2 Conceitos em Computação Sensível ao Contexto

A construção do suporte à sensibilidade ao contexto para as aplicações apresenta inúmeros desafios, dentre eles: (i) a caracterização dos elementos de contexto para uso na

aplicação; (ii) a aquisição do contexto a partir de fontes heterogêneas, tais como sensores físicos, base de dados, agentes e aplicações; (iii) a representação de um modelo semântico formal de contexto; (iv) o processamento e interpretação das informações de contexto adquiridas; (v) a disseminação do contexto a entidades interessadas de forma distribuída e no momento oportuno; e (vi) o tratamento da qualidade da informação contextual.

As próximas subseções resumem o estudo feito sobre os fundamentos inerentes para o suporte à sensibilidade ao contexto.

2.2.1 Identificação dos Elementos de Contexto

Um grande desafio ao desenvolver um sistema sensível ao contexto é a delimitação das ações dependentes de contexto nesses sistemas, bem como a identificação dos 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. Por outro lado, o papel que o contexto desempenha pode ser generalizado sobre tarefas de domínios específicos.

O contexto é uma construção dinâmica e, embora em algumas situações o contexto seja relativamente estável e previsível, existem muitas outras onde isso não acontece. Na maioria dos casos é bastante complexo para o projetista de uma aplicação sensível ao contexto enumerar o conjunto de todos os estados contextuais que podem existir na aplicação, bem como definir que ações devem ser executadas para os diferentes estados.

Algumas classificações para as informações contextuais foram propostas na literatura com o propósito de apoiar a identificação dos elementos de contexto. 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 X. H., 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 (por exemplo, com quem o usuário está, quem são as pessoas próximas), atividades sociais (por exemplo, se o usuário está em reunião ou ministrando aula), entre outras. Para identificar, por exemplo, se um usuário está ministrando uma aula, pode ser criada uma regra lógica que obtenha como contextos básicos a identificação da sala onde ele se encontra, a indicação se existem outras pessoas na sala, a posição do usuário em relação a essas pessoas e se existe um programa de apresentação rodando em um micro instalado na sala.

Greenberg indica que o contexto varia em 5 dimensões: (i) período de tempo, (ii) episódios de uso anteriores conhecidos pela pessoa, (iii) estado das interações sociais, (iv) mudanças nos objetivos internos, e (v) influências do local onde a pessoa se encontra (OZTURK P., 2003).

Uma outra classificação para as informações de contexto separa o contexto em porções estáticas e dinâmicas (GAUVIN M., 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. definiram alguns critérios para a identificação do que escolher como elemento contextual ao construir uma aplicação sensivel ao contexto: (i) habilidade para descrever propriedades úteis do mundo real; (ii) habilidade para inferência de contextos complexos; (iii) facilidade ou viabilidade de ser medido ou reconhecido, automaticamente, de forma o mais precisa e não ambígua possível (KORPIPAA P., 2003). Rosa et al. propõem um *framework* conceitual de contexto para sistemas colaborativos que classifica as informações de contexto em cinco categorias principais: (i) informações sobre as pessoas e os grupos; (ii) informações sobre tarefas agendadas; (iii) informações sobre o relacionamento entre pessoas e tarefas; (iv) informações sobre o ambiente onde ocorre a interação; e (v) informações sobre tarefas e atividades concluídas (ROSA M. G. P., 2003).

De uma maneira genérica, as informações de contextos referentes a uma ação ou tarefa podem ser identificadas a partir da resposta às seis perguntas apresentadas na Seção 2.2.3

2.2.2 Características das Informações Contextuais

As informações contextuais possuem características bem peculiares que devem ser ressaltadas. De acordo com (HENRICKSEN K; INDULSKA, 2003), 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, a identificação de um equipamento, etc. Já as informações dinâmicas, as mais comuns em sistemas ubíquos, variam frequentemente. A persistência de uma informação contextual dinâmica é altamente variável, por exemplo, relações entre colegas e amigos podem durar por meses e anos, enquanto a localização e a atividade de uma pessoa frequentemente se alteram a cada minuto. A característica de persistência 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 de contextos são detectadas através de meios indiretos como sensores.

Informação contextual é imperfeita: 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 ambiente extremamente dinâmico como o da computação ubíqua, a informação contextual rapidamente se torna obsoleta, não refletindo o ambiente que deveria representar. Isso ocorre pelo fato de frequentemente as fontes, os repositórios e os consumidores de contexto estarem distribuídos, gerando muitas vezes um atraso

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. Finalmente, quedas de canais de comunicação, interferências e outras falhas podem ocorrer no caminho entre e o envio e a entrega da informação contextual, perdendo 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. Observe que os requisitos e níveis de abstração que uma informação contextual exige podem variar de uma aplicação para a outra. Portanto, um modelo de contexto deve suportar múltiplas representações do mesmo contexto em diferentes formas e em diferentes 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. Entretanto, outros tipos de relacionamentos entre informações contextuais não são 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.

2.2.3 Dimensões de Informação de Contexto

A partir das definições apresentadas nas seções anteriores, percebe-se que existe 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 a 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 B.N., 1994). Ambos protótipos utilizavam mecanismos emissores de sinais que forneciam a localização de pessoas em um edifício, além de identificarem essas pessoas em mapas eletrônicos periodicamente atualizados. Com tais informações era possível, por exemplo, realizar transferências automáticas de chamadas telefônicas.

Aplicações sensíveis a contexto mais recentes passaram a utilizar as facilidades do sistema de localização outdoor GPS (*Global Positioning System*), bastante utilizado no monitoramento de automóveis em cidades e rodovias. Por exemplo, o sistema Cyber-Guide é utilizado como um guia turístico capaz de escolher conteúdos áudio-visuais para serem exibidos conforme as informações de localização de pessoas. Com os avanços na área de comunicação por redes sem fio, novos sistemas sensíveis a 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.

No entanto, existem outras informações de contexto além de localização e identificação de pessoas e objetos. A maioria dos sistemas sensíveis a 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 a contexto, (ABOWD G. D.; RODDEN, 2002) (COSTA; YAMIN; GEYER, 2008) discutem a utilização de cinco dimensões semânticas de informações de contexto para auxiliar projetistas e desenvolvedores na especificação, na modelagem e na estruturação de informações de contexto de suas aplicações. Essas cinco dimensões semânticas são:

- *Who* (quem) - seres humanos realizam suas atividades e recordam de fatos passados com base na presença de pessoas e/ou objetos. Aplicações sensíveis a 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, email, 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 e/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 quê) - 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 a 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 quê) - 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 a contexto podem obter, via sensores, informações que possam dar uma indicação do estado emocional de um usuário, por exemplo, o foco de atenção e a expressão facial, características de batimento cardíaco e níveis de pressão arterial, entonação vocal e ondas cerebrais do tipo alfa.

Essas cinco dimensões semânticas discutidas não sugerem completeza, mas sim, um conjunto básico de diretrizes a ser seguido no processo de construção de uma aplicação sensível a contexto. Nesse interim, (TRUONG K. N.; BROTHERTON, 2003) introduziram uma nova 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 recursos de um ambiente físico podem ser capturados e acessados. É importante que aplicações sensíveis a contexto te- nham informações não apenas do número e do papel dos dispositivos disponíveis para captura e acesso em um ambiente, mas também que estejam informados acerca das características funcionais de cada dispositivo para captura e acesso. 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.

2.2.4 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 G. K., 2004).

Aquisição sentida: este tipo de informação pode ser adquirido 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. Por exemplo, é possível calcular a idade de uma pessoa baseada na sua data de nascimento.

Aquisição provida: informação que é explicitamente fornecida à aplicação. Por exemplo, os dados cadastrais de um usuário que é diretamente fornecido à aplicação por meio de um formulário.

Esta etapa de aquisição, entretanto, 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.

2.2.5 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 necessárias. Contudo, os atuais modelos para desenvolvimento de software não oferecem suporte para o projeto e de tais aplicações bem como não provêem suporte para modelagem das informações contextuais (HENRICKSEN K; INDULSKA, 2003).

Existem inúmeras abordagens para modelar informações contextuais, dentre as quais pode-se ressaltar (STRANG T; LINNHOFF-POPIEN, 2004):

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 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.

As abordagens citadas acima são avaliadas para modelagem de contexto, considerando os seguintes critérios:

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. Daí

que 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 visa a 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 existente.

Os resultados da análise realizada por (STRANG T; LINNHOFF-POPIEN, 2004) podem ser observados na tabela 2.1. A notação apresentada atribui o sinal -"para o critério não satisfeito pelo modelo, e o sinal "+"para o critério atendido de maneira satisfatória. Sendo "++"para os critérios que são completamente satisfeito.

Tabela 2.1: Avaliação das Abordagens para Modelagem de Contexto (STRANG T; LINNHOFF-POPIEN, 2004)

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

A tabela 2.1, dentre outros aspectos caracteriza que o emprego de ontologias atende os critérios empregados.

2.2.6 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.

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 uma mais primitiva.

O processo de interpretação de contexto pode ser bastante simples como derivar o nome de uma rua a partir de suas coordenadas geográficas ou bastante complexo e oneroso como inferir o humor de um usuário baseado em seu perfil e na 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 (COSTA; YAMIN; GEYER, 2008).

2.2.7 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.

Os termos raciocínio e inferência são geralmente utilizados para indicar qualquer processo pelo qual conclusões são alcançadas (RUSSELL S., 2003). O projeto e implementação de um mecanismo para raciocínio de contextos pode variar bastante a depender 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 R., 2004).

O raciocínio é utilizado para verificar a consistência do contexto e para inferir contexto implícito de alto nível, a partir de contextos explícitos, de baixo nível (WANG X. H., 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.

2.2.8 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 é possível de ser realizado.

2.3 Considerações Sobre o Capítulo

Este capítulo resumiu os aspectos referentes a computação sensível ao contexto, necessário para a concepção da proposta do EXEHDA-SS. O estudo contemplou aspectos como: definições, identificação de elementos, características, dimensões, aquisição, modelagem, interpretação, processamento e raciocínio e armazenamento. Estes aspectos serão utilizados no próximo capítulo, para embasar a análise dos projetos de mecanismos de sensibilidade ao contexto e a comparação realizada entre eles.

3 TRABALHOS RELACIONADOS

Este capítulo discute trabalhos relacionados ao EXEHDA-SS. Os mecanismos de sensibilidade ao contexto avaliados foram selecionados por critério de similaridade, bem como pela relevância na literatura.

Ao todo, foram avaliadas arquiteturas de onze projetos, cujas características constituem um conjunto representativo do que vêm sendo desenvolvido nos últimos anos na direção de infra-estruturas de suporte a sensibilidade ao contexto. Adotou-se manter as figuras com a nomenclatura nativa, boa parte em inglês, devido as mesmas possuírem nomes específicos dos elementos que constituem as arquiteturas.

3.1 *Context Management System*

O CXMS (*Context Management System*) é um *framework* que oferece um conjunto de ferramentas para facilitar o desenvolvimento e manutenção de sistemas e serviços sensíveis ao contexto. Para isso, ele considera as seguintes abstrações principais: (i) aquisição do contexto; (ii) modelagem do contexto; (iii) definição do comportamento da aplicação; e (iv) apresentação da informação de forma correta (ZIMMERMANN A., 2005a) (ZIMMERMANN A., 2005b).

O CXMS é formado pelos seguintes componentes (ver figura 3.1): o *Context Toolkit*, o *Content Management System* (CMS), o Administrator, uma ferramenta de administração para configuração da aplicação, e o *Mobile Collector*, uma ferramenta de edição para a criação de ligações entre os conteúdos e os parâmetros de contexto.

O *Context Toolkit* é o elemento responsável pelo gerenciamento do conhecimento contextual e divide-se em quatro camadas: Sensores, Semântica, Controle e Atuação. A camada dos sensores (*Sensor Layer*) é responsável pela aquisição do contexto por meio de uma rede de sensores físicos que reconhece mudanças no ambiente e recebe todos os eventos de entrada enviados pela aplicação, relacionados à situação atual do usuário. A camada semântica (*Semantic Layer*) atende à modelagem do contexto, fornece a interpretação do contexto enriquecendo semanticamente os dados coletados pela camada dos sensores, e subdivide-se nas camadas: entidade, define as entidades do domínio e gerencia suas propriedades; relacionamento entre entidades, modela as dependências entre as entidades; e processo, observa a evolução dos contextos ao longo do tempo. A camada de controle (*Control Layer*) é responsável por definir o comportamento da aplicação e por decidir que ações devem ser disparadas se condições particulares no modelo forem verdadeiras. Finalmente, a camada de atuação (*Actuation Layer*) lida com a apresentação da informação de forma correta. Para isso, são mapeadas as decisões tomadas pela camada de controle

para ações do mundo real e são modificados os parâmetros de variáveis do domínio de acordo com o comportamento do usuário.

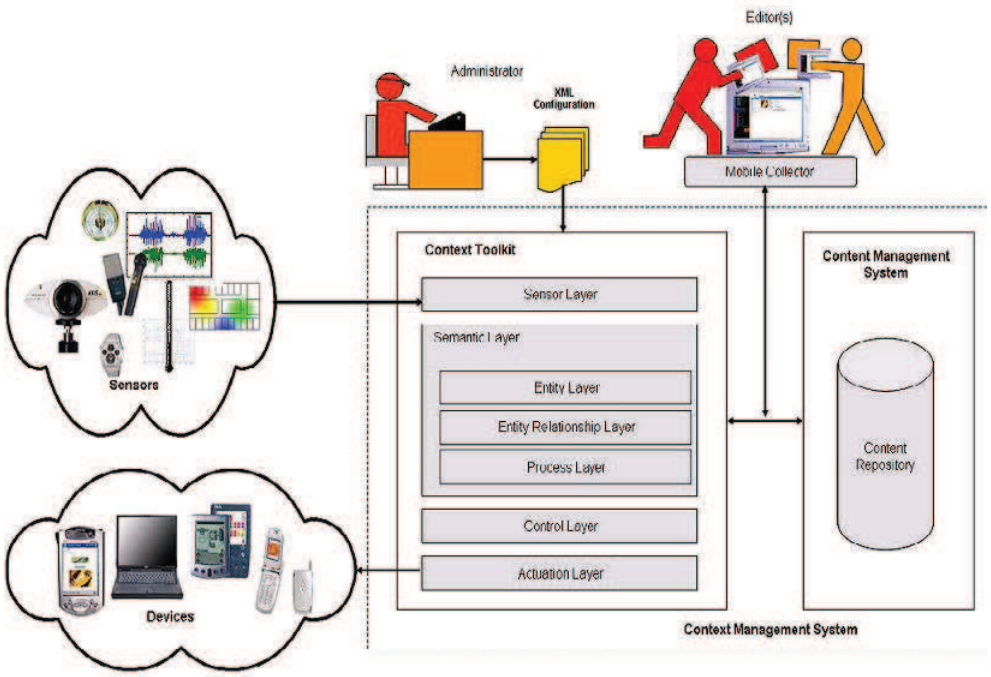

Figura 3.1: Arquitetura do CXMS (ZIMMERMANN A., 2005a)

A abordagem utilizada no CXMS para representação das informações contextuais é baseada no modelo de pares de chave-valor, escolhido pela simplicidade e flexibilidade. Cada contexto é uma enumeração de um ou mais atributos de contexto e cada entidade possui, por definição, um contexto estático e diversos tipos de contextos dinâmicos. É mantido um histórico para cada tipo de contexto. O modelo de contexto cobre quarto dimensões: identidade, localização, tempo e ambiente. Um contexto sempre representa todas as informações atualizadas e disponíveis que descrevem a situação atual de um usuário ou grupo de usuários.

3.2 *Context Toolkit*

O CTK (*Context Toolkit*) é um dos primeiros e mais referenciados projetos de mecanismo para gerenciamento de contexto (DEY, 2000). Seu objetivo é prover uma solução reutilizável para tratamento do contexto que facilite a implementação e desenvolvimento de aplicações sensíveis ao contexto interativas, no domínio da computação ubíqua. O CTK compreende um *framework* para aplicações sensíveis ao contexto baseado em sensores e provê um número de componentes reutilizáveis para construção dessas aplicações.

O CTK incorpora vários serviços relacionados ao gerenciamento de contexto, incluindo aquisição do contexto, acesso a dados de contexto e persistência do contexto. A figura 3.2 mostra a hierarquia de componentes do CTK e uma visão geral da sua arquitetura. O componente raiz é o *BaseObject*, que subdivide-se em *widgets* de contexto (*widgets*), interpretadores de contexto (*interpreters*) e um servidor para descoberta de recursos

(*discoverer*). Os *widgets* são compostos por serviços de contexto (*services*) e agregadores de contexto (*aggregators*) (DEY, 2000).

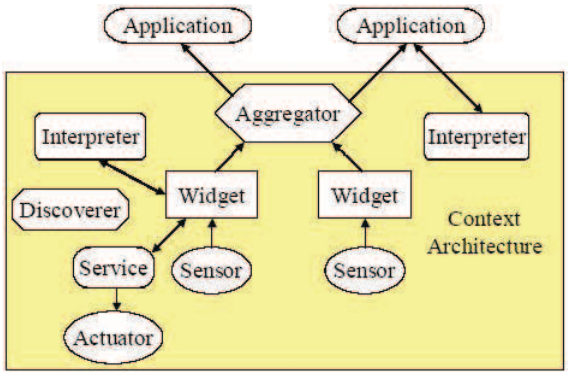

Figura 3.2: Hierarquia de Componentes e Arquitetura do CTK (DEY, 2000)

Os *widgets* de contexto são uma analogia aos *widgets* de interface gráfica e têm por objetivo apoiar a aquisição do contexto, através de sensores, e a disseminação dos contextos, por meio dos atuadores. Os interpretadores ampliam o nível de abstração da informação contextual, para melhor se adequar aos requisitos da aplicação. Os agregadores combinam os diferentes tipos de informações contextuais relacionados a uma entidade. Quando widgets, agregadores e interpretadores são instanciados, eles se registram a um serviço de localização (*discoverer*) e quando uma aplicação é executada, ela contacta esse serviço para localizar componentes que sejam relevantes à sua funcionalidade.

O CTK representa o contexto sob a forma de pares de chave-valor definidos usando a linguagem XML. O principal problema do CTK é a ausência de um modelo formal de contexto com o objetivo de controlar uma aplicação ou mudança de parâmetros dinâmicos dentro da aplicação. O CTK não provê um suporte para raciocínio sobre contextos e inferência de novos contextos de alto nível ou uma estrutura formal para organizar os diversos tipos de contextos. Além disso, a funcionalidade dos interpretadores para derivação de contexto é limitada, uma vez que eles são geralmente empregados apenas para conversões de tipos de dados simples. Como resultado, o suporte a comparações de contexto também é limitado.

3.3 *Middleware* de Contexto do Gaia

Gaia é uma infra-estrutura para ambientes inteligentes de computação ubíqua cujo principal objetivo é tornar inteligente espaços físicos como salas, casas e aeroportos e auxiliar pessoas nesses espaços (ROMAN M., 2002). Gaia possui um *middleware* de contexto que permite que aplicações obtenham e utilizem diferentes tipos de contextos.

O *middleware* de contexto do Gaia visa prover suporte para as seguintes tarefas de gerenciamento de contexto: aquisição do contexto a partir de diferentes sensores; disseminação do contexto a diferentes agentes; inferência de contextos de alto nível a partir de contextos de baixo nível utilizando diferentes tipos de mecanismos de raciocínio e aprendizagem; facilidades para diferentes atuações dos agentes em diferentes contextos; e (v) interoperabilidade semântica e sintática entre diferentes agentes (RANGANATHAN A., 2003).

A figura 3.3 mostra uma visão geral da arquitetura do *middleware* de contexto. A aquisição do contexto é realizada por provedores de contexto (context providers), os quais, juntamente com sintetizadores de contexto (*context synthesizers*), fazem a disseminação da informação contextual a consumidores de contexto (*context consumers*). Os sintetizadores são responsáveis, também, pela inferência dos contextos de alto nível. Os consumidores obtêm diferentes tipos de contextos, raciocinam sobre o contexto atual e adaptam seu comportamento (*atuam*) de acordo com o contexto. Um serviço de busca (*context provider lookup service*) permite que provedores de contexto anunciem seus contextos e que agentes encontrem os provedores adequados a suas necessidades. Um serviço de histórico (*context history service*) mantém os contextos persistidos em um banco de dados para permitir consulta a contextos passados.

A interoperabilidade semântica e sintática entre agentes é provida pelo uso de ontologias para representação do contexto. Um servidor de ontologias (*ontology server*) mantém as ontologias que descrevem diferentes tipos de informação contextual, de modo que outros agentes possam obter descrições dos agentes no ambiente, meta-informações sobre os contextos e definições dos vários termos utilizados no Gaia.

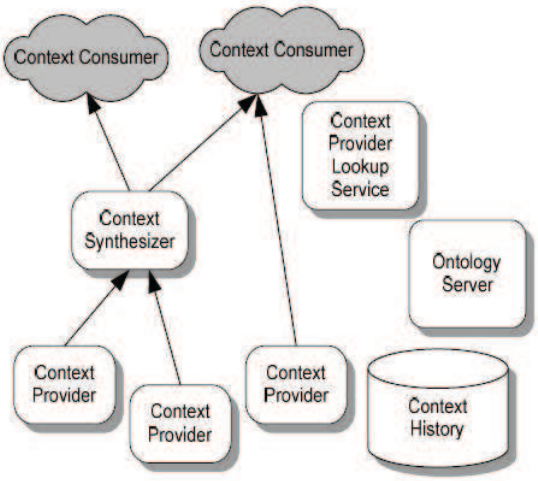

Figura 3.3: Visão Geral do *Middleware* de Contexto do GAIA (RANGANATHAN A., 2003)

A representação de contexto no GAIA era feita, inicialmente, com predicados de lógica de primeira ordem. Posteriormente, em 2003, eles adotaram a abordagem de ontologias, que foi codificada através da linguagem DAML+OIL (DAML, 2009). Os contextos são representados como predicados e as ontologias definem o vocabulário e tipos de argumentos que podem ser utilizados nos predicados.

A ontologia de contexto descreve as entidades, que incluem aplicações, serviços, dispositivos, usuários, fontes de dados, localização, atividades e informações climáticas, suas propriedades e os relacionamentos entre elas, bem como axiomas que restringem as propriedades dessas entidades. As entidades de contexto são classificadas em sete categorias: (i) contextos físicos (localização e tempo); (ii) contextos ambientais (clima, níveis de luz e som); (iii) contextos de informação (notas sobre estoque e placar esportivo); (iv) contextos pessoais (saúde, agenda e atividade); (v) contextos sociais (atividade do grupo, relacionamentos sociais e quem está na sala com quem); (vi) contextos da aplicação (email e páginas web visitadas); e (vii) contextos do sistema (tráfego da rede e

status da impressora).

As aplicações sensíveis ao contexto desenvolvidas em Gaia possuem regras que descrevem que ações devem ser executadas em diferentes contextos. Para escrever essas regras o desenvolvedor deve conhecer os diferentes tipos de contextos disponíveis bem como as possíveis ações que podem ser executadas pela aplicação. As ontologias servem para simplificar a tarefa de escrever essas regras. Gaia inclui uma ferramenta que utiliza a ontologia e auxilia o desenvolvedor na escrita das regras (RANGANATHAN A., 2003).

3.4 *Social Philanthropic Information Environment*

SOPHIE (*Social PHilanthropic Information Environment*) é um ambiente de informação reativo e integrado, que visa rastrear as mudanças constantes que ocorrem em um ambiente e se adaptar a elas, por exemplo, através da disseminação da informação correta aos vários canais de saída (BELOTTI, 2004).

SOPHIE é integrado a um motor de contexto (*context engine*) genérico, com um modelo semântico de contexto (BELOTTI R., 2004a). Esse motor de contexto tem por finalidade gerenciar informações contextuais e pode ser acoplado a aplicações existentes para aumentar a ciência de contexto dessas aplicações (BELOTTI R., 2004b). O modelo de contexto do SOPHIE tem por objetivo consolidar modelos existentes em outras abordagens, movendo os conceitos para um nível mais alto de abstração em termos de descrição semântica do contexto. O modelo baseia-se na extensão ao padrão ORM (*Object-Role Modeling*) proposta por Henricksen et al. (HENRICKSEN K; INDULSKA, 2003) e em semânticas bem definidas que facilitam a reusabilidade e interoperabilidade do contexto em aplicações existentes.

O modelo é centralizado em três conceitos básicos: *context* (contexto de domínio), *providers* (aquisição de contexto) e *notifiers* (comunicação do contexto). Além desses conceitos, eles introduzem um sistema de tipos que define a composição do contexto. Os tipos podem ser definidos para três domínios principais: *ApplTypes*, que indicam os tipos específicos da aplicação, *BaseTypes*, que definem valores primitivos como *string*, inteiro e booleano, e *BulkTypes*, que designam conjuntos de valores de um dado tipo. Os tipos do contexto são definidos como a composição de atributos de um dado tipo (BELOTTI R., 2004a). A figura 3.4 apresenta os principais elementos desse modelo de contexto.

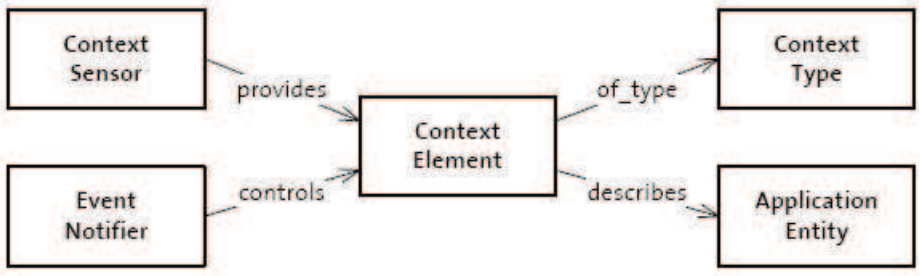

Figura 3.4: Modelo de Contexto (BELOTTI, 2004)

A arquitetura do SOPHIE, e sua integração com o motor de contexto, é ilustrada na figura 3.5. O motor de contexto é dividido em quatro abstrações principais relacionadas ao contexto: aquisição (*context sensing*), que é a obtenção de informações de produtores de contexto; ampliação (*context augmentation*), armazenamento da informação contex-

tual associando-a ao seu assunto; adaptação (*contextual adaptation*), adapta o comportamento a mudanças no contexto corrente; e descoberta de recursos (*contextual resource discovery*), permite descobrir recursos e informações relevantes dependentes do contexto. As informações contextuais processadas pelo motor são obtidas da camada da aplicação (*application*), por meio de informações existentes armazenadas em bases de dados, e da camada de ambiente (*environment*) que representa o mundo real, físico (BELOTTI, 2004).

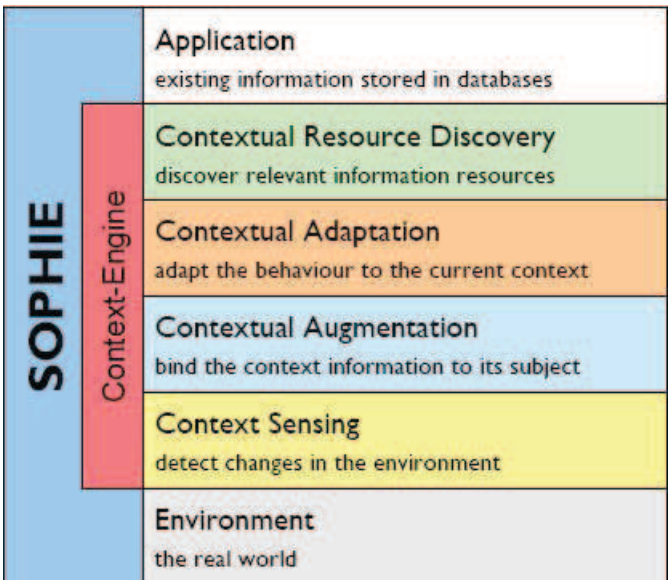

Figura 3.5: Arquitetura do SOPHIE (BELOTTI, 2004)

A aquisição do contexto é baseada em sensores. Para instalar um novo sensor, o desenvolvedor precisa instanciar um sensor e ligá-lo a um dado contexto. O sensor pode ser executado individualmente e atualizar o estado do seu contexto ou pode ser de natureza reativa e responder a uma requisição de atualização do contexto associado (BELOTTI R., 2004a). A disseminação do contexto é feita por meio de notificadores que informam aplicações inscritas como interessadas no contexto sempre que ocorrer um evento sobre o contexto. Um notificador é associado a um contexto e cada vez que o contexto muda, o notificador é invocado. O notificador avalia a mudança no contexto e, caso seja desejado, a aplicação apropriada é notificada (BELOTTI R., 2004a).

3.5 *Context Aware Mobile Networks and Services*

O AWARENESS (*context AWARE mobile Networks and ServiceS*) foi desenvolvido em parceria entre diversas instituições holandesas, dentre elas o CTIT (*Centre for Telematics and Information Technology*) e o Telematica Instituut da University of Twente, o AWARENESS (WEGDAM, 2005) tem como principal objetivo projetar uma infra-estrutura de suporte a serviços e aplicações sensíveis ao contexto. Como domínios de aplicação utilizados para validar a infra-estrutura desenvolvida, são utilizados cenários e aplicações médicas reais. O AWARENESS busca integrar serviços e dispositivos da computação ubíqua com o uso de técnicas e mecanismos de processamento de informações de contexto e suporte à pró-atividade das aplicações, além de ontologias em metodologias de descoberta de serviços. A infra-estrutura ainda em desenvolvimento deverá prover suporte à mobilidade em ambientes sensíveis ao contexto, além de novos

métodos de inferência e uso de contexto em domínios variados (PESSOA R. M., 2006). Seu módulo de gerenciamento de dados é visto na figura 3.6.

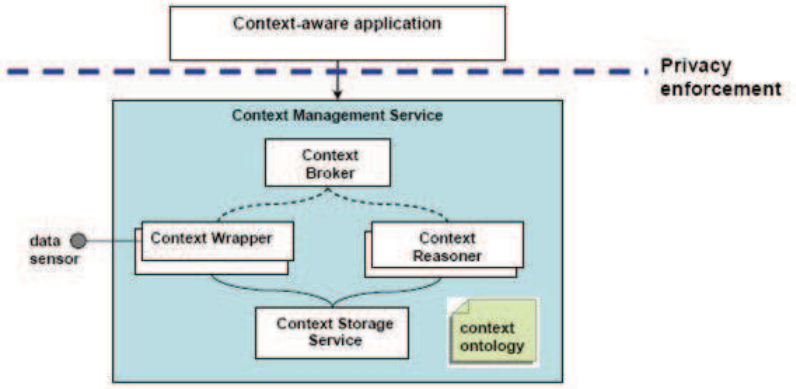

Figura 3.6: Gerenciamento de Dados no Awareness (WEGDAM, 2005)

Os *Context Wrapper*, *Context Storage Service*, e *Context Reasoner* representam fontes de dados contextuais e são implementados com a mesma interface de acesso. O gerenciador sabe quais tipos de contexto estão sendo manipulados por cada um através de uma ontologia. O *Context Broker* é responsável por realizar a descoberta das fontes.

Para a integração de dados, o Awareness também possui entidades responsáveis pela interface superior (com as aplicações) e inferior (com as fontes). O Awareness possibilita a entrega ativa de dados, obedecendo ao padrão de projeto ECA. Este padrão desacopla os três estágios de regra vistos anteriormente, o que fornece grande flexibilidade. As ações executadas podem ser (i) chamadas a Web Services; (ii) resposta para a aplicação ou (iii) execução de serviço interno. A estrutura responsável por sua execução é chamada ECA *Controlling Service*, inspirado em *middleware* existentes com a arquitetura publish-subscribe (SINDEREN, 2006). Em relação a operadores, o Awareness possui expressividade para eventos compostos através de lógica booleana, além de aceitar predicados sob demanda.

3.6 *Service-Oriented Context-Aware Middleware*

SOCAM (*Service-Oriented Context-Aware Middleware*) é um *middleware* para a construção rápida de serviços sensíveis ao contexto em ambientes inteligentes (GU T., 2004). Para permitir a interoperabilidade entre aplicações e apoiar o raciocínio sobre contexto, SOCAM utiliza a ontologia CONON (*Context Ontology*) (WANG X. H., 2004) (GU T., 2004). CONON é um modelo semântico de contexto, serializado em OWL-DL.

Conceitualmente, a CONON é dividida em duas partes distintas (figura 3.7): uma ontologia de alto nível (*upper ontology*) que captura os conceitos genéricos sobre contextos básicos (localização, usuário, atividade e entidade computacional), e uma coleção de ontologias específicas de domínio, que são construídas como especializações da ontologia de alto nível para domínios específicos. Estas definem os detalhes dos conceitos genéricos para cada sub-domínio. A separação em domínios facilita o reuso de conceitos genéricos e provê uma interface flexível para definição de conhecimento específico para a aplicação (WANG X. H., 2004).

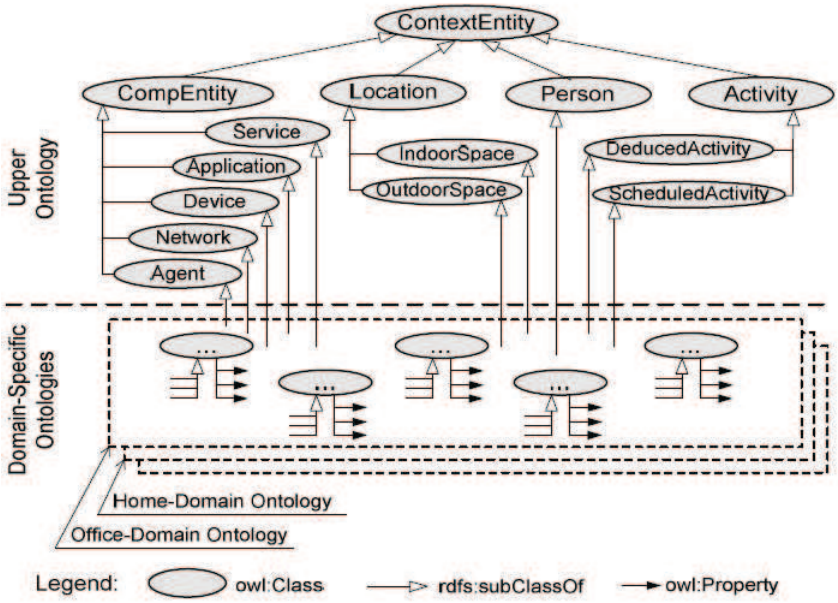

Figura 3.7: Visão Parcial da Ontologia de Alto Nível do CONON (WANG X. H., 2004)

O *middleware* SOCAM (figura 3.8) provê suporte para as seguintes tarefas no gerenciamento de contexto: aquisição, compartilhamento, raciocínio, armazenamento e disseminação do contexto. A aquisição do contexto é feita pelos componentes denominados context providers, que abstraem os contextos a partir de diferentes fontes externas ou internas, e convertem-nos na representação formal da ontologia CONON. O compartilhamento ocorre através da utilização da CONON, que permite que os contextos sejam compartilhados e reutilizados pelos vários componentes do SOCAM.

O raciocínio sobre contexto é executado pelos interpretadores de contexto (*context interpreters*), que consistem de motores de raciocínio de contexto (*context reasoning engine*s) e de uma base de conhecimento contextual (*context* KB). Os motores de raciocínio realizam inferência de contextos, a partir de regras pré-definidas, resolução de conflitos dos contextos, e manutenção da consistência da base de conhecimento contextual.

O armazenamento do contexto é efetuado através da manutenção de um banco de dados de contexto (*context database*) que persiste as ontologias de contexto e os contextos passados. A base de conhecimento contextual provê serviços que permitem que os outros componentes possam consultar, adicionar, remover ou modificar o conhecimento contextual armazenado no banco de dados de contexto.

A disseminação do contexto é realizada pelo serviço de localização de serviços (*service locating service*). Esse serviço provê um mecanismo onde os provedores e interpretadores de contexto possam anunciar suas presenças e os usuários ou aplicações possam localizar e acessar esses serviços.

Para raciocinar sobre a incerteza em relação aos elementos de contexto representados, a SOCAM utiliza redes bayesianas (GU T., 2004). Esse modelo anexa valores de probabilidade aos predicados de contexto definidos na CONON. Para isso foi proposta uma extensão à OWL para inclusão de rótulos de marcação para probabilidades. A escolha pelas redes bayesianas é justificada pela eficiência em lidar com raciocínio probabilístico e por permitir representar relacionamentos causais entre vários contextos. Entre as limitações dessa abordagem está a dificuldade em obter dados para treinar a rede

bayesiana em certas circunstâncias, como aplicação de controle de segurança. A lógica *fuzzy* pode ser utilizada para representar e raciocinar sobre noções imprecisas de contexto, como ”quente”, ”muito baixo”ou ”confiança”.

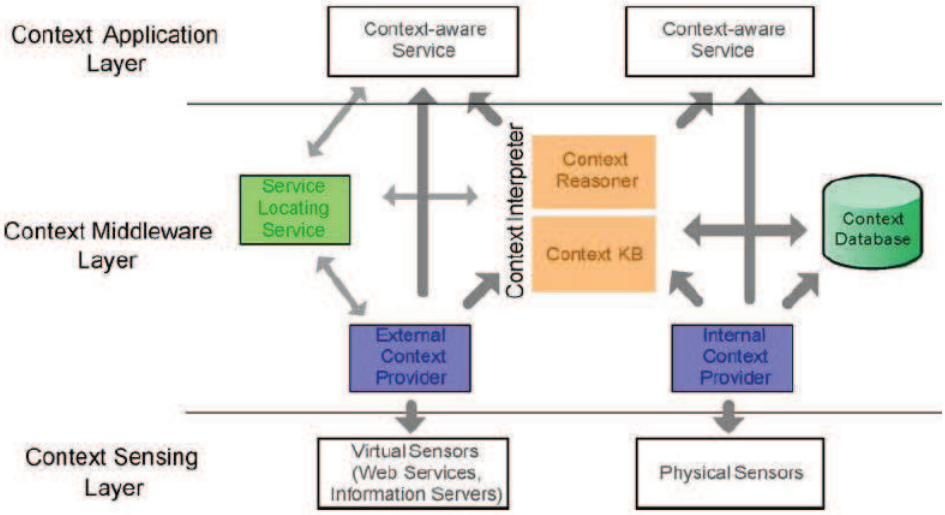

Figura 3.8: Visão Geral da Arquitetura do SOCAM (GU T., 2004)

3.7 *Context Broker Architecture*

CoBrA (*Context Broker Architecture*) é uma arquitetura baseada em agentes cujo objetivo é apoiar sistemas sensíveis ao contexto em espaços inteligentes, em particular salas de reuniões inteligentes em um campus universitário. O elemento principal dessa arquitetura é um agente inteligente chamado context broker que mantém e gerencia um modelo compartilhado do contexto, a ontologia CoBrA-Ont e provê serviços de proteção de privacidade para os usuários (CHEN H., 2005).

CoBrA-Ont é uma ontologia de contexto desenvolvida em OWL e tem por objetivo auxiliar os agentes na aquisição, raciocínio e compartihamento do conhecimento contextual, bem como apoiar a detecção e resolução de conhecimento contextual inconsistente e ambíguo (CHEN G., 2002). A representação gráfica dos conceitos ontológicos da CoBrA-Ont é categorizada em quatro temas distintos e relacionados: (i) conceitos que definem lugares físicos e suas relações espaciais associadas; (ii) conceitos que definem agentes (humanos e de software); (iii) conceitos que descrevem o contexto de localização de um agente em um campus universitário; e (iv) conceitos que descrevem os contextos de atividade de um agente, incluindo papéis, desejos e intenções associadas em um evento de apresentação.

A arquitetura do CoBrA é ilustrada na figura 3.9. Os requisitos para gerenciamento de contexto tratados pelo CoBrA são: (i) aquisição de contexto de fontes heterogêneas, como sensores físicos, serviços web, bancos de dados, dispositivos e agentes (*Context Acquisition Module*); (ii) raciocínio sobre a informação para deduzir conhecimento adicional a partir da informação adquirida, e manter o modelo de contexto consistente (*Context Reasoning Module*); (iii) compartilhamento do conhecimento contextual através do uso de ontologias comuns (*RDF/OWL*), e padrões de comunicação entre agentes como a linguagem *FIPA-ACL* e o protocolo SOAP; (iv) proteção da privacidade dos

usuários através de políticas definidas pelo usuário e de regras de comportamento do broker associadas a essas políticas (*Privacy Management Module*) (CHEN, 2004).

Para realizar o raciocínio sobre contexto, CoBrA utiliza um número de diferentes sistemas baseados em regras, como Jena (JENA, 2009), usado para inferência sobre a ontologia OWL, JESS (*Java Expert System Shell*) (JESS, 2009), usado para interpretação de contexto utilizando regras específicas do domínio, e Theorist (POOLE D., 2006), um raciocinador baseado em sentenças Prolog utilizado para apoiar as inferências lógicas para resolver conhecimento inconsistente (CHEN, 2004).

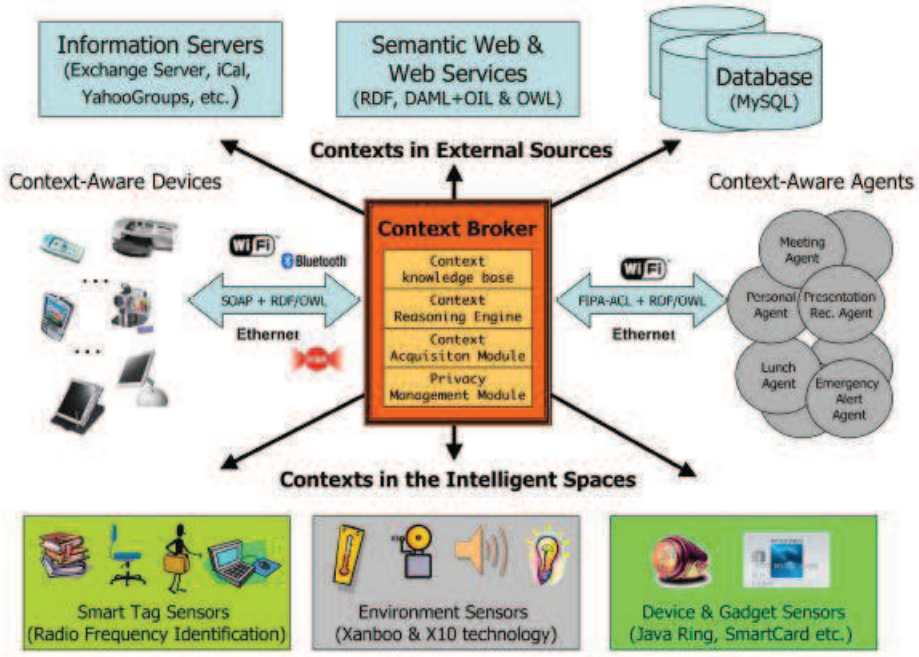

Figura 3.9: Visão Geral da Arquitetura do CoBrA (CHEN H., 2005)

3.8 *Mobile Collaboration Architecture*

A MoCA (*Mobile Collaboration Architecture*) (SACRAMENTO et al., 2004), é uma arquitetura de *middleware* para o desenvolvimento de aplicações colaborativas e sensíveis ao contexto para computação móvel. É constituída por APIs para implementação de clientes e servidores, serviços básicos para aplicações colaborativas e um *framework* para implementação de proxies.

A MoCA define que cada aplicação tem três partes: um servidor, um *proxy* e um ou vários clientes. O servidor e o *proxy* executam em nós fixos, enquanto a parte cliente executa em nós móveis. O *proxy* realiza a intermediação de toda a comunicação entre o servidor e o cliente. É nele que está a lógica de adaptação para a aplicação clienteservidor a qual está vinculado.

Para a construção desses proxies, a MoCA disponibiliza um *framework* denominado *ProxyFramework*. Este *framework* provê mecanismos de acesso às informações de contexto relacionadas à interação cliente-servidor, e também define a programação de adaptações disparadas pelas mudanças de contexto. Ele implementa algumas das características mais comuns, quando se usa uma abordagem baseada em proxy, como meios

para lidar com a mobilidade dos clientes e conectividade intermitente.

Os serviços que formam a arquitetura MoCA destinam-se a dar suporte ao desenvolvimento e a execução de aplicações colaborativas sensíveis ao contexto. O *Monitor* é um processo que executa em segundo plano em cada dispositivo móvel. Ele coleta informações sobre o ambiente e o estado de execução no dispositivo e os envia ao CIS (*Context Information Service*).

O CS (*Configuration Sevice*) é responsável por armazenar e gerenciar as informações de configuração para todos os dispositivos móveis. Cabe ao DS (*Discovery Services*, armazena informações das aplicações e serviços registrados na MoCA.

Outro serviço, CIS (*Context Information Services*) recebe e processa as informações de estado enviadas por cada Monitor. No LIS *Location Inference Service*) é fornecida a localização aproximada dos dispositivos. Na figura 3.10 é ilustrada a arquitetura MoCA e a interação entre seus diversos componentes durante o registro e execução de uma aplicação.

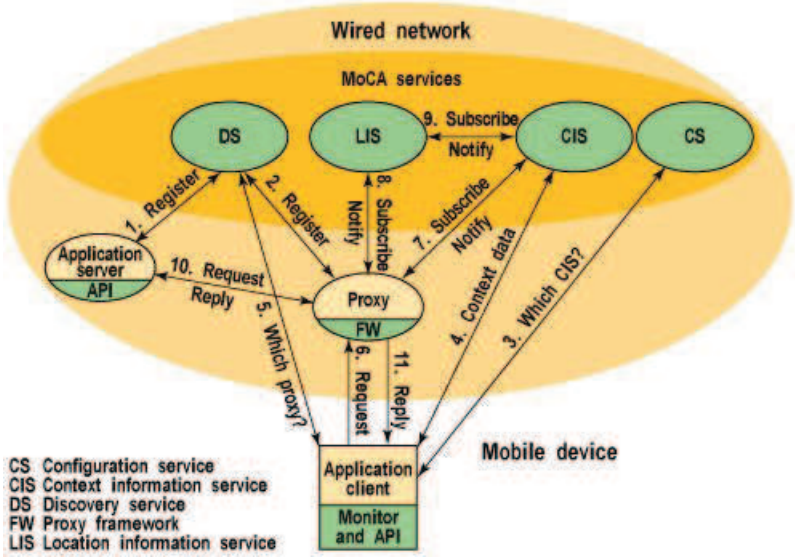

Figura 3.10: Serviços da Arquitetura Moca (SACRAMENTO et al., 2004)

3.9 *Framework* de Contexto

Henricksen e Indulska (HENRICKSEN K., 2005a) propuseram um *framework* de contexto que visa facilitar a construção de aplicações sensíveis ao contexto e prover suporte às tarefas de aquisição, representação, persistência e disseminação de contextos, e adaptação de aplicações ao contexto. Essas funcionalidades são baseadas em um modelo de contexto que identifica a diversidade da informação contextual, sua qualidade, relacionamentos complexos entre dados de contexto e aspectos temporais. O *framework* de contexto (figura 3.11) é organizado em uma hierarquia de seis camadas: (i) camada da aplicação sensível ao contexto; (ii) camada de adaptação; (iii) camada de consulta; (iv) camada de gerenciamento do contexto; (v) camada de recepção do contexto; e (vi) camada de aquisição do contexto.

A camada de aquisição utiliza sensores para adquirir a informação contextual e processa essa informação através de interpretadores e agregadores. A camada de recepção

provê um mapeamento bidirecional entre o contexto adquirido e as camadas de gerenciamento, traduzindo os dados de entrada para o modelo formal definido. A camada de gerenciamento é responsável pela manutenção de um conjunto de modelos de contexto e suas instanciações, onde a aplicação pode definir seu próprio modelo de contexto, segundo a abordagem CML (*Context Modeling Language*), ou compartilhar modelos de aplicações similares. A camada de consulta provê uma interface para que aplicações possam consultar o sistema de gerenciamento de contexto. A camada de adaptação gerencia repositórios comuns de definições de situações e preferências e avalia essas definições usando serviços das camadas mais baixas. A camada da aplicação provê um toolkit de programação com suporte à criação, ativação e desativação de gatilhos.

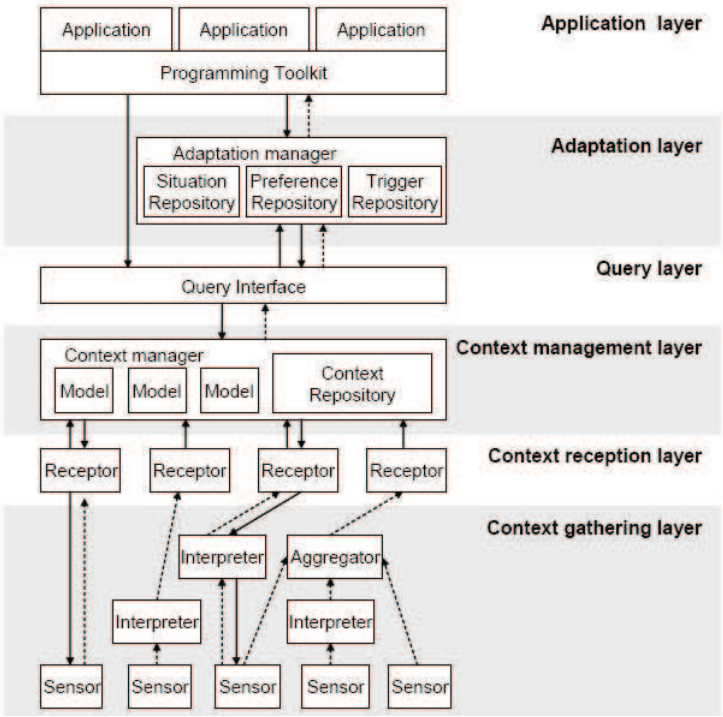

Figura 3.11: Arquitetura do *Framework* de Contexto (HENRICKSEN K., 2005a)

3.10 *Semantic Context Kernel*

O SCK (*Semantic Context Kernel*) é uma infra-estrutura de serviços para gerenciamento de contexto (BULCAO NETO R. F., 2005). Sua arquitetura inclui serviços configuráveis para armazenamento, consulta e inferência sobre contexto, um serviço de descoberta, além de componentes que apóiam as tarefas de aquisição, disseminação e adaptação ao contexto e conversão do contexto em um modelo semântico de representação, baseado em ontologias. O modelo semântico visa prover interoperabilidade semântica e reuso do conhecimento contextual. O principal objetivo do SCK é prover aos desenvolvedores de aplicações sensíveis ao contexto um conjunto de serviços que possam ser configurados para atender aos requisitos da aplicação. Os autores apontam a configurabilidade dos serviços como a principal funcionalidade do Kernel. A arquitetura do SCK é apresentada na figura 3.12.

A informação contextual é adquirida de fontes de contexto heterogêneas (*context*

sources) como aplicações, serviços web e sensores físicos. As informações adquiridas são convertidas em uma representação semântica comum pelos tradutores de contexto (*context transducers*) e utilizadas por consumidores de contexto (*context consumers*) para adaptar seu comportamento ao contexto atual. Um serviço de descoberta (*discovery service*) disponibiliza propagandas de fontes de contexto de modo que consumidores de contexto possam encontrar as informações que necessitam. O serviço de inferência (context inference service) provê um suporte configurável ao raciocínio de contexto, e permite que desenvolvedores definam suas regras de inferência. O serviço de consulta (*context query service*) permite que consumidores de contexto consultem o contexto por meio de uma linguagem declarativa chamada RDQL. Finalmente, o serviço de persistência (*context persistence service*) possibilita o armazenamento persistente do contexto em uma maneira configurável, de forma que os desenvolvedores das aplicações possam escolher os tipos de armazenamento e representação do contexto, como bancos de dados relacionais, arquivos RDF/XML ou arquivos texto em formato N-Triple.

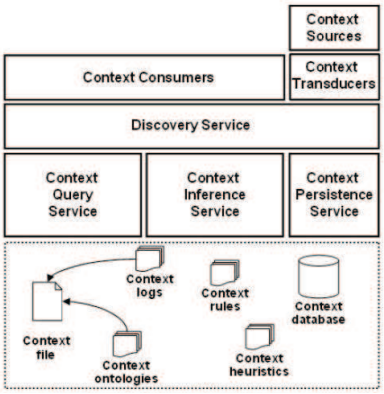

Figura 3.12: Visão Geral da Arquitetura do SCK (BULCAO NETO R. F., 2005)

A representação do contexto no SCK é feita usando uma ontologia (figura 3.13) serializada em OWL. A ontologia divide o contexto em 5 dimensões: identidade dos atores (*who*), localização (*where*), tempo (*when*), atividade (*what*) e perfis dos dispositivos (*how*). A ontologia dos atores usa um conjunto de ontologias externas como *FOAF* (Friend of a Friend) (FOAF, 2009), Dublin Core (DCMI, 2009) e vCard (IANNELLA, 2008) para modelar papéis, projetos, contatos, especialidades, documentos e relacionamentos sociais associados aos atores.

3.11 Infraware

A plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006) é um *middleware* baseado em Web Services com suporte arquitetural para o desenvolvimento, construção e execução de aplicações móveis sensíveis ao contexto. A arquitetura conceitual da Infraware estende, em vários aspectos, a da plataforma WASP (WASP, 2003), um projeto holandês desenvolvido pela University of Twente, Telematica Instituut e Ericsson. A plataforma WASP concentra-se na interface aplicação-plataforma, definindo uma linguagem para especificar como ela deve reagir a uma correlação de eventos. A Infraware, por sua vez, foi definida visando o atendimento a vários requisitos funcionais presentes em ambientes sensíveis ao contexto e a integração desses em uma infra-estrutura única,

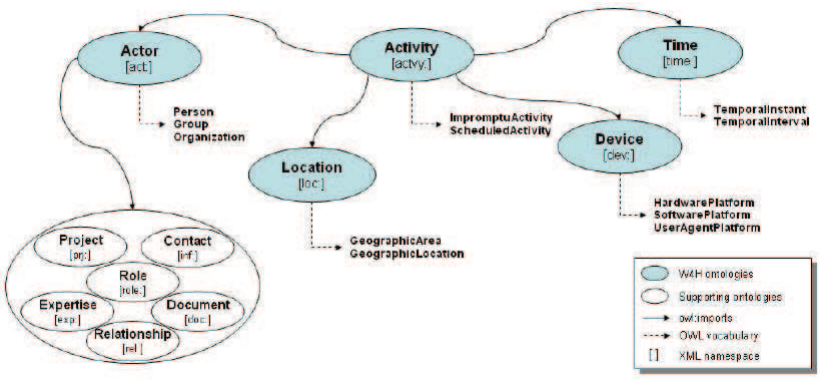

Figura 3.13: Visão Geral do Modelo de Contexto (BULCAO NETO R. F., 2005)

formando uma arquitetura flexível e adequada ao desenvolvimento de aplicações ubíquas reais, em domínios variados. Por exemplo, a Infraware está sendo usada como base para o desenvolvimento de aplicações móveis, especificamente, na área da Saúde.

Uma característica marcante da Infraware é o uso de conceitos da Web Semântica: ontologias especificam modelos formais extensíveis que descrevem não somente o domínio das aplicações, mas também os serviços. Essa abordagem diferenciada provê meios de configurar as interações aplicação-plataforma em run-time. A plataforma também pode ser customizada pela adição de novos serviços e entidades estendendo-se as ontologias. Adicionalmente, a adoção de Web Services como tecnologia de distribuição permite que aplicações acessem os serviços oferecidos através de protocolos da Internet e facilita a inclusão de novos serviços à plataforma por terceiros. Essa flexibilidade torna a Infraware adequada ao desenvolvimento de uma larga gama de aplicações em cenários reais.

A Infraware apresenta uma camada específica para o recebimento e o tratamento das subscrições das aplicações à plataforma e trata do controle de acesso e privacidade de maneira especializada através de um módulo direcionado a tal propósito. A plataforma também resolve o problema do acesso e integração de dados heterogêneos através de uma infra-estrutura dedicada, e é capaz de manipular, derivar e interpretar semanticamente informações de contexto de domínios variados. Além disso, aborda o problema da resolução de conflitos entre aplicações de maneira diferenciada através de um componente coordenador. A figura 3.14 ilustra a arquitetura geral da plataforma.

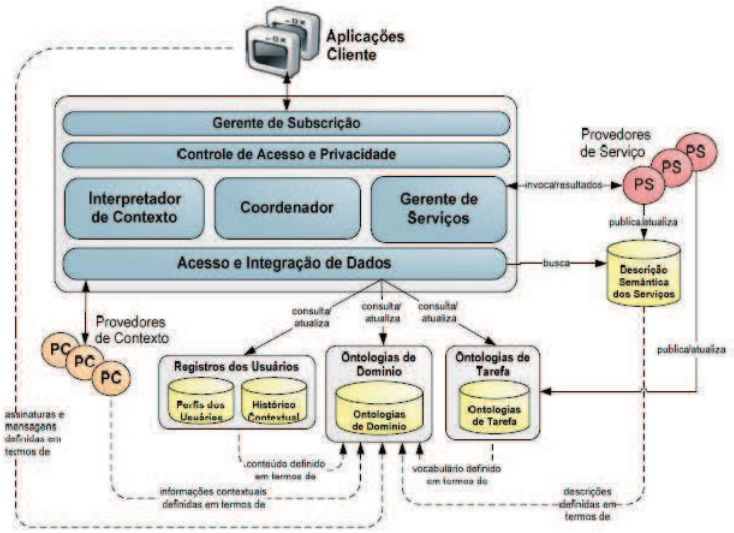

Figura 3.14: Plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006)

3.12 Considerações Sobre o Capítulo

Este capítulo apresentou diversas abordagens de mecanismos de sensibilidade ao contexto presentes na literatura. A tabela 3.1 exibe o modelo de representação contextual utilizado por cada um e na tabela 3.2, apresenta os principais tipos de informações de contexto utilizadas por cada um deles.

Nestas duas tabelas foi empregada a notação a seguir:

- (A) *Context Management System*
- (B) *Context Toolkit*
- (C) *Middleware* de Contexto do Gaia
- (D) *Social Philanthropic Information Environment*
- (E) *Context Aware Mobile Networks and Services*
- (F) *Service-Oriented Context-Aware Middleware*
- (G) *Context Broker Architecture*
- (H) *Mobile Collaboration Architecture*
- (I) Framework de Contexto
- (J) *Semantic Context Kernel*
- (K) Infraware

Tabela 3.1: Modelo de Representação Contextual

Representação	A	B	C	D	E	F	G	H	I	J	K
Par chave-valor	X	X									
Ontologia			X		X	X	X	X	X	X	*X*
Gráfico ORM				X							
Gráfico CLM									X		

Tabela 3.2: Informações Contextuais

Informações Contextuais	A	B	C	D	E	F	G	H	I	J	K
Agentes							X				
Ambiente	X										*X*
Aplicação			X		X					X	*X*
Aplicação define seu contexto				X				X	X		
Atividades		X	X			X	X				*X*
Disponibilidade contínua de componentes de captura								X			
Dispositivos			X		X					X	
Entidade computacional						X					
Estrutura de tipos genérica				X							
Fonte de dados			X								
Identidade	X	X									
Informações climáticas			X								
Intenções							X				
Localização	X	X	X					X		X	*X*
Localização física						X	X				
Papéis							X				
Serviços			X								*X*
Tempo	X	X									
Usuários			X		X	X				X	

Os mecanismos encontrados na literatura são, em sua maioria, voltados para o domínio da computação ubíqua, área em que o estudo e aplicação do contexto estão mais avançados. Por essa razão, eles seguem um modelo e arquitetura similares. Seus componentes modulares executam, em geral, as tarefas de: (i) aquisição do contexto; (ii) representação das informações contextuais; (iii) raciocínio e inferência sobre as informações contextuais capturadas; (iv) persistência do contexto, mantendo um banco de dados histórico; (v) disseminação do contexto; e (vi) notificação das aplicações dos contextos adquiridos.

O EXEHDA-SS propõe uma abordagem que possibilita aos dados contextuais capturados pelo servidor de contexto, possam ser tratados pelo mecanismo de suporte semântico. No próximo capítulo serão apresentadas os fundamentos para arquitetura do EXEHDA-SS.

4 EXEHDA-SS: FUNDAMENTOS

Neste capítulo, são discutidos os principais fundamentos relacionados ao EXEHDA-SS. São considerados aspectos relacionados aos conceitos e linguagens para representação de ontologia, Projeto PertMed e por fim, é descrito o *middleware* EXEHDA, sua organização e subsistemas. Os subcapítulos descritos a seguir, constituem os principais fundamentos utilizados na proposição do EXEHDA-SS.

4.1 Ontologias

Ontologias têm sido largamente utilizadas em áreas como gerenciamento de conteúdo e conhecimento e comércio eletrônico. Particularmente, a comunidade científica tem apontado o uso de ontologias para lidar com alguns dos principais desafios relacionados à construção de ambientes ubíquos. De um modo geral, ontologias têm sido usadas para representar ambientes ubíquos, descrevendo, comumente, entidades envolvidas e suas respectivas propriedades. Elas definem principalmente os diferentes tipos de aplicações, serviços, dispositivos, usuários, entre outros. Além disso, estas ontologias definem descrições padrões para localização, atividades, informação sobre temperatura, etc. Neste capítulo, são tratados os principais conceitos relacionados a este assunto, partindo do conceito de ontologia, passando pelos principais tipos de ontologias, os benefícios advindos do uso das mesmas e finalmente concluindo com a descrição das principais linguagens para ontologias.

4.1.1 O Conceito de Ontologia

Embora a palavra "ontologia" denote, em sua origem filosófica, uma teoria sobre a natureza do ser, para a Computação, ela vem sendo usada como um conjunto de entidades com suas relações, restrições, axiomas e vocabulário. Segundo (GRUBER, 2003), "uma especificação de um vocabulário de representação para um domínio de discurso compartilhado - definições de classes, relações, funções e outros objetos - é uma ontologia". O termo ontologia pode também ser definido a partir dos requisitos para possibilitar sua aplicação em informática. Sendo assim, uma ontologia pode ser definida como "uma especificação explícita e formal de uma conceitualização compartilhada" (RUDI STUDER V. RICHARD BENJAMINS, 2003). Esclarecendo os requisitos desta definição, tem-se que (FREITAS, 2003):

- Por "especificação explícita", pode ser entendida como sendo definições de conceitos, instâncias, relações, restrições e axiomas;

- Por “formal”, que é declarativamente definida através de uma linguagem formal, portanto, compreensível para agentes inteligentes e sistemas;
- Por “conceitualização”, que se trata de um modelo abstrato de uma área de conhecimento ou de um universo limitado de discurso;
- Por “compartilhada”, por tratar-se de um conhecimento consensual, seja uma terminologia comum da área modelada ou acordada entre os desenvolvedores dos agentes que se comunicam.

4.1.2 Linguagens para Representação de Ontologia

Ontologias estão intimamente relacionadas com a linguagem usada para representá-las. Atualmente, existem algumas linguagens com esse propósito. A seguir, é apresentada uma visão geral sobre as principais linguagens, assim como das ontologias de cada linguagem.

4.1.2.1 *Resource Description Framework*

RDF é uma linguagem de propósito geral para representar informação na Internet que baseia-se na idéia de identificar coisas através identificadores Web: os URIs (*Uniform Resource Identifier*). URIs são cadeias de caracteres utilizadas para identificar recursos, como páginas, serviços, documentos, etc. Além dos identificadores (URIs), esta linguagem descreve recursos em termos de simples propriedades e valores. Isto permite que RDF represente recursos sob a forma de expressões sujeito-predicado-objeto:

1. O sujeito: é o recurso, ou seja, qualquer coisa que pode conter um URI, incluindo as páginas, assim como elementos de um documento XML.
2. O predicado: é uma característica descritiva ou aspecto do recurso e por vezes expressa uma relação entre o sujeito e o objeto.
3. O objeto: é o objeto da relação ou o valor da característica descritiva RDF é um tipo de rede semântica (SOWA, 2006), sendo parecida, em termos de linguagem, com o Modelo Relacional. Isto implica que RDF é uma forma de representação de conhecimento que possui semântica auto-contida e oferece uma grande liberdade para criação de extensões personalizadas.

4.1.2.2 *Resource Description Framework Shema*

RDFs é uma linguagem para representação de conhecimento que baseia-se na idéia de Frames (BUBLITZ, 2005). Ela tem sido usada para aumentar a expressividade de RDF, dispondo assim de um melhor suporte à definição e classificação. Este modelo organiza o conhecimento através de herança e de construtores de ontologias (frames, slots e facetas). Os frames são organizados em rede, significando que quando qualquer um deles for acessado, ligações com outros quaisquer, potencialmente importantes, estarão disponíveis, podendo ser visto como uma “unidade de conhecimento” auto-suficiente. Um frame é uma descrição de um objeto complexo. Ele é identificado por um nome e consiste de um conjunto de slots. Cada slot possui um nome único ao frame em que está definido, consistindo de um conjunto de facetas (atributos) de valores particulares. Sistemas baseados em frames permitem que os usuários representem o mundo com diferentes

níveis de abstração, com ênfase sobre as entidades. Em adição ao que já é herdado pelo fato de basear-se em frames, RDFs dispõe de construtores de ontologias que tornam as relações menos dependentes de conceitos: usuários podem definir relações como uma instância de rdf:Property, descrever relações de herança como rdfs:subPropertyOf e então associar relações definidas com classes usando rdfs:domain ou rdfs:range (LI DING PRANAM KOLARI, 2005).

4.1.2.3 *Web Ontology Language*

A OWL é uma linguagem para definir e instanciar ontologias. Ela foi projetada para disponibilizar uma forma comum para o processamento de conteúdo semântico da informação. Ela foi desenvolvida para aumentar a facilidade de expressar semântica disponível em XML, RDF e RDFs. Conseqüentemente, pode ser considerada uma evolução destas linguagens em termos de sua habilidade de representar conteúdo semântico interpretável por máquinas. Já que a OWL é baseada em XML, a informação pode ser facilmente trocada entre diferentes tipos de computadores usando diferentes sistemas operacionais e linguagens de programação. 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 que pode ser resolvida utilizando-se de ferramentas adequadas. OWL vem sendo usada para criar padrões que forneçam um arcabouço para gerenciamento de ativos, integração empresarial e compartilhamento de dados.

OWL atualmente tem três sub-linguagens (algumas vezes também chamadas de "espécies"): OWL Lite, OWL DL e OWL Full. Estas três sub-linguagens possuem nível crescente de expressividade, e foram projetadas para uso por comunidades específicas de programadores e usuários.

1. OWL Lite dá suporte 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 mais simples fornecer ferramentas que suportem OWL Lite que linguagens correlatadas mais expressivas, e ela também permite um caminho de migração mais rápido de dicionários e outras taxonomias.

2. 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 DL é assim chamada devido a sua correspondência com as lógicas de descrição, um campo de pesquisa que estudou a lógica que forma a base formal da OWL.

3. OWL Full é direcionada àqueles usuários que querem a máxima expressividade e a liberdade sintática do RDF sem nenhuma garantia computacional. Por exemplo, em OWL Full uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um único indivíduo. OWL Full permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL.

4.2 Projeto PertMed

O projeto PERTMED vem sendo desenvolvido por três universidades do sul do Brasil (UFSM, UCPEL e UFPEL) e conta com a colaboração de equipes médicas ligadas a essas universidades.

O sistema de saúde do futuro prevê o uso de tecnologias da computação ubíqua formando um espaço inteligente (reativo e pró-ativo), onde dispositivos móveis e fixos estão integrados ao ambiente físico (objetos) visando captar informações do meio e transmitir as alterações detectadas para sistemas de gerenciamento de informações, os quais tomarão decisões e adaptar-se-ão as situações detectadas (computação sensível ao contexto) (PERTMED, 2007).

Neste momento, em termos de pesquisa e inovações, a computação ubíqua na saúde (*pervasive healthcare*) está sendo conduzida sob duas perspectivas: (i) uso de tecnologias ubíquas para criar um hospital virtual; (ii) tornando as informações relativas a saúde disponíveis em todo lugar, a qualquer tempo, usando diversos dispositivos de acesso pertencentes ao próprio paciente/médico ou dispositivo no ambiente. No Brasil, é praticamente inexistente trabalhos tendo a saúde e computação ubíqua como tema.

Usando a experiência do grupo de pesquisa em sistemas móveis e ubíquos (GMob), da Universidade Federal de Santa Maria (UFSM), do grupo de processamento paralelo (G3PD) que compõem a Universidade Católica de Pelotas (UCPEL) e a Universidade Federal de Pelotas (UFPEL) e a qualificação da equipe médica dos hospitais universitários dessas universidades, propõem-se uma inovação nos sistemas de comunicação e informação através do uso de estratégias usadas na computação ubíqua para inserir aspectos importantes ainda não presentes nos sistemas de informação da saúde no país: tornar a informação disponível aonde (lugar) ela é necessária de forma contextualizada. Informação é a base da tomada de decisão.

O projeto PERTMED propõe fazer a ponte entre os sistemas automatizados existentes (registro de pacientes, exames laboratoriais, entre outros) e o médico no local em que este se encontra (regiões remotas ou em trânsito, por exemplo). Desta forma, elimina-se a exigência de estar-se conectado a uma rede fixa e com um computador pessoal na área do hospital para ter acesso às informações do paciente (PERTMED, 2007).

O uso das funcionalidades e capacidades fornecidas pelos telefones celulares e *smartphones* que tornam informações do paciente disponível para o médico/enfermeiro com direito de acesso, em qualquer lugar que esteja necessitando dessas informações para tomada de decisão. No momento que este solicita a informação, ela é acessada e enviada para o dispositivo móvel em uso pelo médico/enfermeiro no momento (por exemplo, *smartphone* e adaptada ao dispositivo que o médico/enfermeiro usa. Logo, a distância do médico em relação ao sistema de equipamentos que armazena as informações sobre o paciente é irrelevante, tornando o sistema de informação mais flexível e de acordo com a liberdade de movimentação requerida pelos profissionais da saúde.

A *pervasive healthcare* está sendo considerada a próxima etapa da *Web-based Healthcare Computing* que oferece vantagens competitivas aos provedores de serviço de saúde; em particular, aumenta a eficiência do serviço, a qualidade e melhora o gerenciamento da relação com o paciente (VARSHNEY, 2003). Este novo sistema de saúde também prevê uma visão de hospital virtual, o qual estende-se para a casa dos pacientes ou lugares onde eles se encontram, onde sensores/dispositivos monitoram as condições ambientais e do paciente e comunicam-se, via rede sem fio, com as centrais médicas para tomada de decisões e ações pertinentes. Experiências nesse sentido estão sendo conduzi-

das por alguns projetos de pesquisa europeu, como o do *Centre of Pervasive Healthcare* na Dinamarca que desenvolve o projeto *Hospital of the Future* (BARDRAM J.; BOSSEN, 2005).

Como se vê, a computação ubíqua terá um enorme potencial de aplicabilidade na área da saúde. O projeto PERTMED tem como motivações contribuir para que sejam superados alguns desafios de área de saúde dentre eles destaca-se:

- Falta de acesso a serviços especializados em regiões remotas ou carentes;
- Alto custo de transporte de pacientes, especialmente de áreas pobres e rurais;
- Aumento da fragmentação e falta de sequência do tratamento.

Uma questão que permeia esses três problemas é o acesso a informação de onde ela é gerada para onde ela é necessária, em tempo razoável com a gravidade da situação sendo tratada. A rapidez da decisão médica depende da pronta disponibilidade de informação sendo esta a chave para a qualidade dos serviços prestados. Acesso à informação pode se usado para substituir o transporte, por exemplo, um "paciente virtual" (formado por um conjunto de informações sobre seu estado de saúde) pode ser monitorado por especialistas que estão quilômetros de distância.

Com a introdução da computação ubíqua, tornará-se possível contribuir para que algumas barreiras sejam superadas dentre elas destaca-se: desigualdades regionais e sócio-econômicas, relativas ao acesso às informações dos sistemas de saúde, com o uso de duas tecnologias amplamente disponíveis: telefones celulares/*smarthphones* e Internet (aplicações Web).

Em termos científicos, o objetivo do projeto é avaliar o potencial de aplicabilidade de algumas estratégias e tecnologias usadas na computação ubíqua para os sistemas de saúde: (i) credenciais associadas aos modernos códigos de barras multimensionais (tags) e (ii) mecanismos de disseminação ubíqua de dados no ambiente internet móvel (especialmente, smartphones), às informações sobre o paciente, respeitadas as restrições de segurança e privacidade.

O sistema PERTMED irá fazer a ligação entre os sistemas automatizados existentes e o médico seja lá qual for o local onde ele se encontra. Com isso, atende-se a necessidade e flexibilidade e liberdade de movimentação do médico no atendimento aos pacientes. Também, haverá uma melhoria na agilidade dos serviços uma vez que a informação poderá ser obtida assim que for gerada (resultado de um exame laboratorial, por exemplo). Agilidade tende diminuir as filas de atendimento entre outras coisas. Outra possibilidades que a tecnologia permite e que pode ser avaliada é a de orientação simples (mensagens) serem enviadas aos pacientes via seus celulares, a partir da decisão do médico. O uso da tecnologia para comunicação médico-paciente irá beneficiar o sistema de saúde, em termos de agilidade no atendimento, o qual contribuiu para uma melhor qualidade de serviço.

Redes de alta velocidade e computadores pessoais não são a realidade em muitas regiões e hospitais. Porém, um pequeno computador - o telefone celular - está amplamente difundido e está mais presente nas casas do que computadores pessoais com acesso à Internet, segundo dados oficiais. Logo, pode-se tirar vantagens da disponibilidade de comunicação via telefonia móvel, a qual permite independência de lugar e tempo, e começar a explorar seu uso na saúde, com o desenvolvimento de aplicações práticas, simples e úteis.

O custo de transmissões de dados digitais está ficando mais barato (INFOEXAME, 2007). A tendência é de declínio dos preços a medida que aumenta o uso de aplicações além das conversas telefônicas. As operadoras de telefonia celular podem oferecer planos especiais a baixo custo para o uso na saúde.

Espera-se que, com o início de cooperação entre as equipes de computação e as equipes médicas das instituições, crie-se uma prática de transferência de tecnologia da pesquisa em computação ubíqua para a saúde, fazendo aplicações reias que usam os conhecimentos gerados como resultado da pesquisa como ocorre em países desenvolvidos.

Para o desenvolvimento do sistema está sendo utilizado métodos, técnicas e ferramentas de análise e projetos orientado a objetos. Particularmente, usam-se padrões de projetos e diagramas UML que auxiliam na modelagem do sistema, os quais facilitam futuras alterações/evoluções.

A linguagem utilizada para o desenvolvimento é a plataforma Java. Está foi escolhida pela ampla aceitação, facilidades fornecidas para projetos na área de mobilidade e Web, e pela portabilidade o que facilita a programação de PDAs, telefones celulares e *smartphones*.

O projeto PERTMED prevê o uso do EXEHDA como *middleware* direcionado a computação ubíqua. A seguir é apresentado uma revisão arquitetural e funcional do mesmo.

4.3 *Middleware* EXEHDA: Revisão Arquitetural e Funcional

O EXEHDA (YAMIN, 2004) é um *middleware* que integra o projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas) (ISAM, 2009), sendo direcionado às aplicações distribuídas, móveis e sensíveis ao contexto da Computação Ubíqua. Abaixo são apresentadas premissas perseguidas na concepção do EXEHDA (YAMIN, 2004).

4.3.1 Premissas de Pesquisa do EXEHDA

A seguir serão apresentadas as principais premissas de pesquisa do *middleware* EXEHDA, considerando para isso, a dinamicidade e heterogeneidade do ambiente de processamento, o controle da adaptação, o suporte às mobilidades lógica e física e o suporte da semântica siga-me.

4.3.1.1 Dinamicidade e Heterogeneidade do Ambiente de Processamento

O deslocamento do usuário (mobilidade física), aspecto característico da Computação Ubíqua, determina a execução de aplicações a partir de diferentes equipamentos e/ou pontos da rede global, nos quais a oferta e/ou disponibilidade dos recursos computacionais é variável. Disto decorre:

- Elevada flutuação na banda passante disponível para as comunicações;
- Equipamentos de usuários com acentuadas diferenças nos atributos de *hardware* e sistema operacional;
- Diferentes infra-estruturas para conexão à rede global.

4.3.1.2 Suporte a Sensibilidade ao Contexto

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.

4.3.1.3 Suporte às Mobilidades Lógica e Física

A Computação Móvel genericamente se refere a um cenário onde todos, ou alguns nodos que tomam parte no processamento, são móveis. Desta definição podem derivar diferentes interpretações. 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.

4.3.1.4 Suporte a Semântica Siga-me

Oferecer suporte à semântica siga-me é uma das contribuições centrais do EXEHDA ao Projeto ISAM. No EXEHDA, este 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, no EXEHDA é adotada a decomposição das funcionalidades de mais alto nível, recursivamente, em funcionalidades mais básicas. Nesta perspectiva, no EXEHDA 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 possóveis 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 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.

4.3.2 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 deve permitir que tanto a aplicação como ele próprio utilizem estas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. Entende-se por adaptação funcional aquela que implica a modificação do código sendo executado. Por sua vez, adaptação não-funcional, é aquela que atua sobre a gerência da execução distribuída. Também a premissa siga-me das aplicações ubíquas deverá ser suportada, garantindo a execução da aplicação do usuário em qualquer tempo, lugar e equipamento (YAMIN, 2004).

As aplicações-alvo são distribuídas, adaptativas ao contexto em que executam e compreendem a mobilidade lógica e a 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 (ISAM, 2009).

4.3.2.1 Arquitetura de Software

A figura 4.1 apresenta a arquitetura de software do *middleware* EXEHDA. Os principais requisitos que o EXEHDA deve atender são: (i) gerenciar tanto aspectos não-funcionais como funcionais da aplicação, e de modo independente, (ii) dar suporte à adaptação dinâmica de aplicações; (iii) disponibilizar mecanismos para obter e tratar informações de contexto; (iv) utilizar informações de contexto na tomada de decisões, (iv) decidir as ações adaptativas de forma colaborativa com a aplicação e (v)disponibilizar a semântica siga-me, possibilitando ao usuário o disparo de aplicações e o acesso a dados a partir de qualquer lugar, e a execução contínua da aplicação em face ao seu deslocamento.

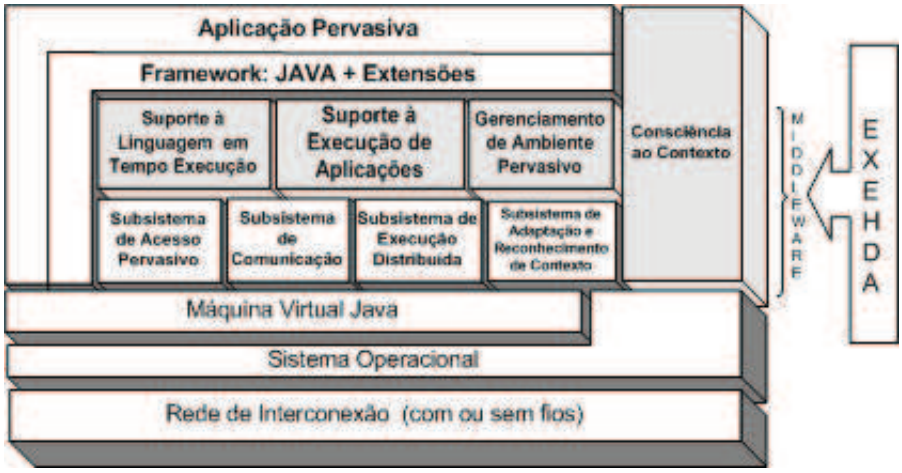

Figura 4.1: Arquitetura de Software *Middleware* EXEHDA (YAMIN, 2004)

4.3.2.2 Ambiente Ubíquo Disponibilizado

O ambiente ubíquo corresponde ao ambiente computacional onde recursos e serviços são gerenciados pelo EXEHDA com o intuito de atender os requisitos da Computação Ubíqua. A composição deste ambiente envolve tanto os dispositivos dos usuários, como os equipamentos da infra-estrutura de suporte, todos instanciados pelo seu respectivo perfil de execução do *middleware*. A integração dos cenários da computação em grade, da computação móvel e da computação sensível ao contexto, é mapeada em uma organização composta pela agregação de células de execução do EXEHDA, conforme pode ser visto na figura 4.2 (ISAM, 2009).

O meio físico sobre o qual o ambiente ubíqua 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 ambiente ubíquo (YAMIN, 2004):

- EXEHDAcels: 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;

- EXEHDAbase: é o ponto de contato para os EXEHDAnodos. É responsável por todos os serviços básicos do ambiente ubíquo e, embora constitua uma referência

lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários equipamentos;

- EXEHDAnodo: são os equipamentos de processamento disponíveis no ambiente ubíquo, sendo responsáveis pela execução das aplicações. Um subcaso deste tipo de recurso é o EXEHDAnodo móvel. São os nodos do sistema com elevada portabilidade, tipicamente dotados de interface de rede para operação sem fio e, neste caso, integram a célula a qual seu ponto-de-acesso está subordinado. São funcionalmente análogos aos EXEHDAnodos, porém eventualmente com uma capacidade mais restrita (por exemplo, PDAs).

O ambiente ubíquo é formado por equipamentos multi-institucionais, o que gera a necessidade de adotar procedimentos de gerência iguais aos utilizados em ambientes de Grade Computacional (Grid Computing) (YAMIN, 2004). O gerenciamento da organização celular do ambiente ubíquo resguarda a autonomia das instituições envolvidas. Apesar de não contemplar mecanismos de gerência específicos para recursos especializados como impressoras, scanners, etc., o EXEHDA permite a catalogação de tais recursos como integrantes de uma determinada célula do ambiente ubíquo, tornando-os, desta forma, passíveis de serem localizados dinamicamente e serem utilizados pelas aplicações ubíquas.

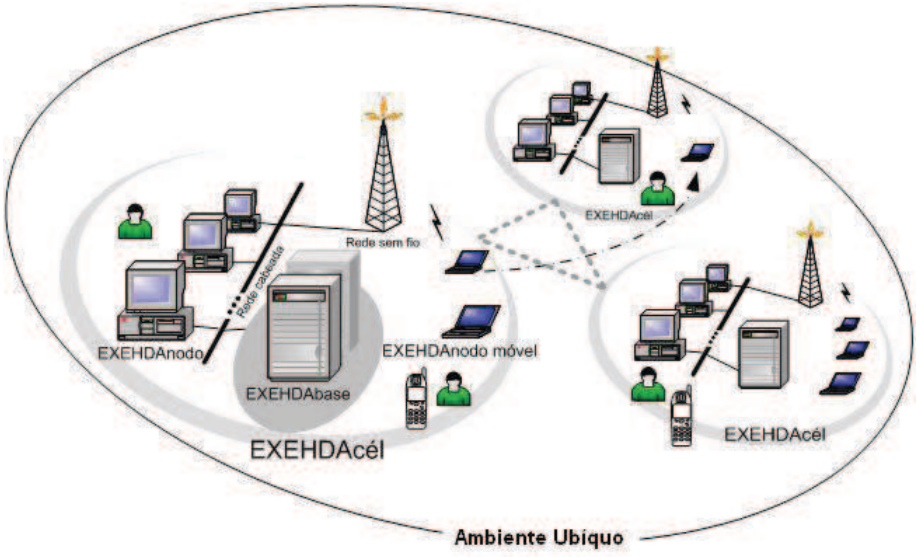

Figura 4.2: Ambiente ubíquo provido pelo EXEHDA (YAMIN, 2004)

4.3.2.3 Composição de Serviços

O requisito de operação em um ambiente altamente heterogêneo, onde não só o hardware exibe capacidades variadas de processamento e memória, mas também as bibliotecas de software disponíveis em cada dispositivo, motivaram a adoção de uma abordagem na qual um núcleo mínimo do *middleware* tem suas funcionalidades estendidas por serviços carregados sob demanda. Esta organização reflete um padrão de projeto referenciado na literatura como micro-kernel. Some-se a isto o fato de que esta carga sob demanda tem perfil adaptativo. Deste modo, poderá ser utilizada versão de um determinado

serviço, melhor sintonizada às características do dispositivo em questão. Isto é possível porque, na modelagem do EXEHDA, os serviços estão definidos por sua interface, e não pela sua implementação propriamente dita.

A contra-proposta à estratégia micro-kernel de um único binário monolítico, cujas funcionalidades cobrissem todas as combinações de necessidades das aplicações e dispositivos, se mostra impraticável na Computação Ubíqua, cujo ambiente computacional apresenta elevada heterogeneidade de recursos de processamento.

Por sua vez, o requisito do *middleware* de manter-se operacional durante os períodos de desconexão planejada motivou, além da concepção de primitivas de comunicação adequadas a esta situação, a separação dos serviços que implementam operações de natureza distribuída em instâncias locais ao EXEHDAnodo (instância nodal), e instâncias locais a EXEHDAbase (instância celular). Neste sentido, o relacionamento entre instância de nodo e celular assemelha-se à estratégia de Proxies, enquanto que o relacionamento entre instâncias celulares assume um caráter P2P. A abordagem P2P nas operações inter-celulares vai ao encontro do requisito de escalabilidade. Uma organização dos subsistemas do EXEHDA pode ser visto na figura 4.3;

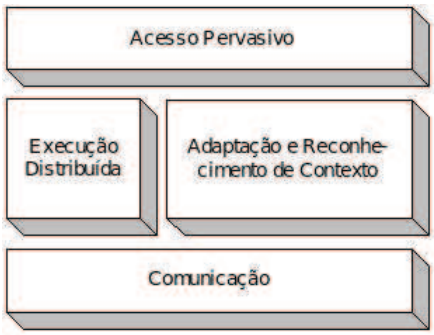

Figura 4.3: Organização dos Subsistemas do EXEHDA (YAMIN, 2004)

Com isso, os componentes da aplicação em execução em determinado dispositivo podem permanecer operacionais, desde que, para satisfação de uma dada requisição pelo *middleware*, o acesso a um recurso externo ao dispositivo seja prescindível. Por outro lado, a instância celular, em execução na base da célula, provê uma referência para os outros recursos, no caso da realização de operações que requeiram coordenação distribuída. Neste sentido, observe-se que a EXEHDAbase é, por definição, uma entidade estável dentro da EXEHDAcel, permitindo que os demais integrantes (recursos) da célula tenham um caráter mais dinâmico no que se refere a sua disponibilidade (presença efetiva) na célula.

4.3.2.4 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 EXEHDAnodo, 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 ubíquo. Este núcleo é formado por dois componentes, conforme figura 4.4:

- 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 demandao código dos serviços do *middleware*, a partir do repositório de serviços que pode ser local ou remoto, dependendo da capacidade de armazenamento do EXEHDAnodo e da natureza do serviço.

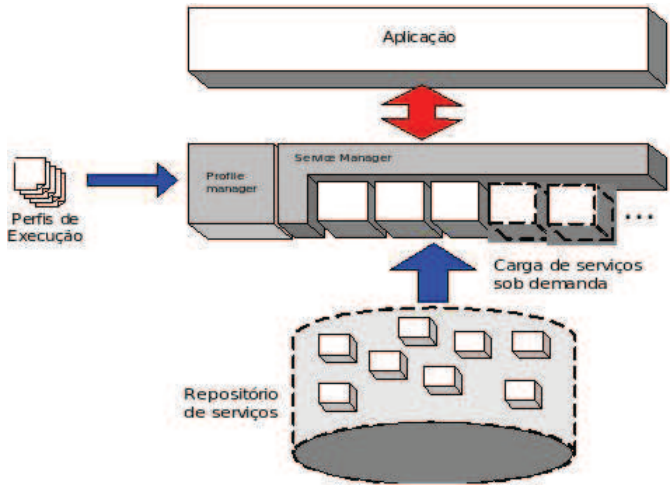

Figura 4.4: Organização do Núcleo do EXEHDA (YAMIN, 2004)

4.3.3 Subsistemas do EXEHDA

O *middleware* EXEHDA é composto por quatro subsistemas: Subsistema de Execução Distribuída, Subsistema de Comunicação, Subsistema de Acesso Ubíquo e Subsistema de Reconhecimento de Contexto e Adaptação. A seguir é apresentado a composição de cada um desses subsistemas.

4.3.3.1 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 ubíqua, 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 da Computação Ubíqua. Este subsistema é constituído pelos serviços: (i) *Executor*, (ii) *Cell Information Base* - CIB, (iii) *OXManager*, (iv) *Discoverer*, (v) *ResourceBroker*, (vi) *Gateway*, (vii) *StdStreams*, (viii) *Logger*, e (ix) *Dynamic Configurator* - DC.

4.3.3.2 Subsistema de Comunicação

A natureza da mobilidade do hardware e, na maioria das vezes, também a do software, não garante a interação contínua entre os componentes da aplicação distribuída. As desconexões são comuns, não somente devido à existência de alguns links sem fio, mas sobretudo como uma estratégia para economia de energia nos dispositivos móveis. O subsistema de comunicação do EXEHDA disponibiliza mecanismos que atendem estes aspectos da Computação Ubíqua. Integram este subsistema os serviços *Dispatcher*, *WORB*, *CCManager* os quais contemplam modelos com níveis diferenciados de abstração para as comunicações.

4.3.3.3 Subsistema de Acesso Ubíquo

A premissa de acesso em qualquer lugar, todo o tempo, a dados e código da Computação Ubíqua, requer um suporte do *middleware*. Os serviços que compõem este subsistema no EXEHDA são: (i) BDA - Base de dados Ubíqua das Aplicações, (ii) AVU - Ambiente Virtual do Usuário, (iii) *SessionManager* e (iv) *Gatekeeper*.

4.3.3.4 Subsistema de Reconhecimento de Contexto e Adaptação

Integram este subsistema os serviços (i) *Collector*, (ii) *Deflector*, (iii) *ContextManager*, (iv) *AdaptEngine* e (v) *Scheduler*. Particularmente, *AdaptEngine* e *Scheduler* são responsáveis, respectivamente, pelo controle das adaptações de cunho funcional e não funcional. Entende-se por adaptação funcional aquela que implica a modificação do código sendo executado. Por sua vez, adaptação não-funcional, é aquela que atua sobre a gerência da execução distribuída, na qual podem ser identificados seus diversos serviços (YAMIN, 2004).

4.4 Considerações Sobre o Capítulo

Neste capítulo foram apresentados os fundamentos do EXEHDA-SS incluindo os conceitos e motivações para o uso de ontologias e suas linguagens.

Ainda neste capítulo foram apresentados o Projeto PertMed, e exploradas as características e funcionalidades do *middleware* EXEHDA, considerando o foco do trabalho, destacou-se o subsistema de reconhecimento de contexto e adaptação, especialmente os serviços e componentes que fazem parte da arquitetura de software. Com base nestes fundamentos apresentados, o capítulo a seguir introduz a proposição da modelagem e arquitutura do EXEHDA-SS.

5 EXEHDA-SS: CONCEPÇÃO E MODELAGEM

Este capítulo apresenta as linhas gerais do novo mecanismo de sensibilidade ao contexto que está sendo proposto para o *middleware* EXEHDA. Este mecanismo será integrado ao Subsistema de Reconhecimento de Contexto e Adaptação do *middleware*, e tem como contribuição central oferecer um suporte semântico no processamento das informações contextuais.

Assim sendo, a concepção do EXEHDA-SS contempla um servidor de contexto que realiza funcionalidades relacionadas a captura, processamento e notificação dos contextos, contemplando para representação dos mesmos o emprego de ontologias.

O EXEHDA-SS trabalha de maneira colaborativa com o EXEHDA-DA (WARKEN, 2009), que é o serviço de controle de adaptação dinâmica das aplicações ubíquas do *middleware* EXEHDA. As mudanças do Contexto de Interesse das aplicações são enviadas ao EXEHDA-DA para serem implementadas.

As funcionalidades propostas para o mecanismo de sensibilidade ao contexto com suporte semântico do EXEHDA-SS foram concebidas considerando a discussão dos mecanismos de sensibilidade ao contexto realizada no Capítulo 3. Esses mecanismos exploram características de: aquisição de contexto, representação das informações contextuais, dedução e inferência sobre as informações, persistência do contexto e notificação das aplicações dos contextos adquiridos. Esses elementos são de fundamental importância para a construção deste trabalho e constituem a modelagem do mecanismo proposto desta dissertação.

As seções subsequentes apresentam, respectivamente, o modelo de representação de contexto do ambiente ubíquo e a modelagem da arquitetura de software para sensibilidade ao contexto personalizável por componentes de software das aplicações com suporte a processamento semântico do EXEHDA-SS.

5.1 Modelo de Representação de Contexto

Na concepção do EXEHDA-SS, foi selecionado o uso de ontologias como mecanismo para representação e processamento de contexto. Tal decisão teve como critério principal o estudo resumido na seção 2.2.5. Neste sentido, foi definida a OntUbi, Ontologia do Ambiente Ubíquo, onde ficam instanciados o ambiente ubíquo de execução do EXEHDA-SS.

As instâncias das Políticas das Adaptações das Aplicações estão definidas pela OntAdapt (WARKEN, 2009) e o Contexto de Interesse das Aplicações pela OntContext. Desta forma a OntContext e a OntAdapt estão contidas na OntUbi. A seguir serão apresentadas os dois modelos contextuais utilizados pelo EXEHDA-SS: a **OntUbi** e a **OntContext**. A primeira concebida para representação do ambiente ubíquo e a segunda para as informações contextuais coletas, processadas e notificadas pelo EXEHDA-SS.

5.1.1 OntUbi

A OntUbi, figura 5.1, é a ontologia responsável pela representação do ambiente de execução ubíquo promovido pelo EXEHDA-SS. Ela foi desenvolvida em conjunto com os outros trabalhos do Grupo de Processamento Paralelo e Distribuído - G3PD e está em constante evoluções pois é a ontologia base para representação dos elementos que compõe o ambiente ubíquo provido pelo EXEHDA.

Na perspectiva da concepção do mecanismo de sensibilidade ao contexto do EXEHDA-SS, a OntUbi foi modelada e validada considerando um conjunto de classes, subclasses e relacionamentos, sendo composta por quatro classes bases, sendo elas: Software, Usuario, Celula e Hardware. Essas classes são relacionadas a classe **nodo**. Desse modo, definiu-se que um nodo é composto por atributos e relacionamentos. As classes, subclasses e relacionamentos da OntUbi são detalhados a seguir.

Classes, Subclasses e Atributos do Nodo

- Nodo: Compõem a classe Nodo os seguintes atributos:
  - Nodo_Base (*true* caso seja um EXEHDAbase ou *false EXEHDAnodo*); - Nodo_Movel (*true* se for um nodo model ou *false* caso não); - Nodo_Status ('A' para ativo e 'I' para inativo).
  A classe Nodo é composta pelos relacionamentos:
  Nodo_Categoria; Nodo_Celula; Nodo_Dispositivo; Nodo_Contexto; Nodo_Periferico; Nodo_Software e Nodo_Usuario.

- Hardware: Formada pelas subclasses Dispositivos, Categorias e Perifericos. Os atributos da classe Hardware herdados pelas suas subclasses são:
  - Hardware_Fabricante (nome do fabricante do respectivo artefato de hardware);
  - Hardware_Modelo (descrição do modelo).
  Dispositivos ficam instanciados suas características para Armazenamento, Audio, Bateria, dispositivos de Entrada (teclado, mouse e outros), Memoria, Portas (USB, Paralela, Serial e outras), Processadores, Tela e Rede.
  Categorias estão definidas as categorias dos dispositivos entre elas: celular, desktop, notebook, PDA, Servidor e Smartphone.
  Por fim, na subclasse Periferico estão definidos os periféricos do *hardware* que serão conectados aos EXEHDAnodos, como: Impressora, Scanner e Sensor.

- Celula: Composta pelo atributo Celula_Id - identificador da EXEHDAcell e a subclasse Local - onde estão instanciados os prédios e salas da respectiva célula. Compõem os seguintes relacionamentos: Celula_Usuario e Celula_Rede.

- Usuario: A classe Usuario é composta pelos atributos nome, login, senha e pelo relacionamento com a subclasse Perfil. Nos perfis ficam instanciados os tipos de

perfis de atuação na EXEHDAbase e nos EXEHDAnodos como administrador, visitante e outros.

- Software: A subclasse SistemaOperacional representa as informações dos sistemas operacionais como: versão, tipo de sistema operacional(windows, unix), modelo. As subclasses Contexto_Interesse, Sensor_Public e Contexto_Deduzido são instancias do Contextos de Interesse das Aplicações do EXEHDA-SS - vide seção 5.1.2. Enquanto que a subclasse Aplicacao, Componente, Adaptador, Param_Valor, Param_Tipo, Adapt, Estado constituem a Política da Adaptação da Aplicação (WARKEN, 2009), onde são relacionadas as aplicações com componentes, adaptadores, tipos de parâmetros e valores; da mesma forma que a subclasse Adapt são instanciadas os componentes e nodos do Executor do EXEHDA e seus respectivos estados especificados na subclasse Estado.

Relacionamentos do Nodo

- Nodo_Software: Relacionamento entre Nodo e Software. A função deste relacionamento é especificar os softwares utilizados nos EXEHDAnodos.
- Nodo_Usuario: Relacionamento entre Nodo e Usuario. Por meio deste relacionamento são obtidos os usuário que operam o nodo.
- Nodo_Celula: Relacionamento do Nodo com Celula onde estão instanciados os locais da EXEHDAcell com seus respectivos prédios e salas.
- Nodo_Categoria: Relacionamento do Nodo com Categorias. A finalidade desse relacionamento é definir no EXEHDAnodo a categoria de dispositivos (celular, desktop, outros) pertencentes ao nodo.
- Nodo_Dispositivo: Relacionamento de Nodo com Dispositivo. Defini os tipos de dispositivos que estão no EXEHDAnodo, permitindo que embora seja de um mesmo modelo do que outro possuir mais características diferentes do outro. Esse relacionamento é uma das características diferenciais do modelo contextual da OntUbi.
- Nodo_Contexto: Relacionamento do Nodo com Contexto. Relaciona as informações dos nodos publicados pelos sensores;
- Nodo_Periferico: Relacionamento do Nodo com Periferico. Onde relaciona-se os perifericos que podem ou não estar conectados ao EXEHDAnodo.

Relacionamentos da Celula

- Celula_Usuario: Relacionamento da Celula com Usuario, onde é possivel saber os usuarios que estão conectos na Celula.
- Celula_Rede: Estabelece a relação de Celula com Rede, sendo possível obter as redes que estão operando na Celula.

As classes Contexto e Contexto_Notificado juntamente com as subclasses ContextoNotificado_Sensor e ContextoNotificado_Deduzido pertencem a OntContext que será detalhada na seção seguinte.

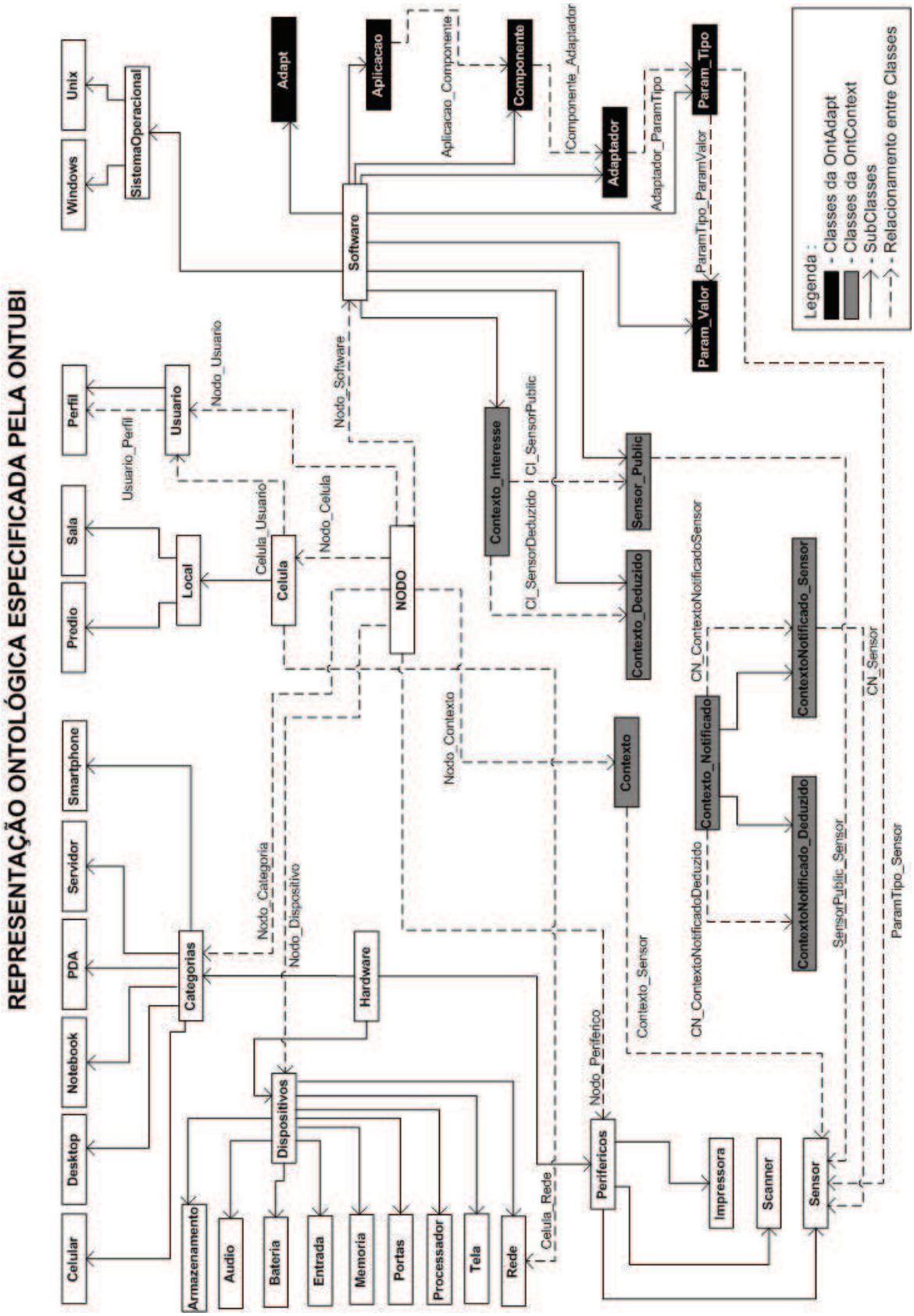

Figura 5.1: Diagrama de Classes, Subclasses e Relacionamentos da OntUbi

5.1.2 OntContext

A OntContext, figura 5.2, é a ontologia definida para ser responsável pela representação dos contextos coletados, notificados e das instâncias dos Contextos de Interesse das Aplicações. A OntContext é utilizada pelo servidor de contexto do EXEHDA-SS para prover suporte semântico ontológico nas tarefas de processamento das informações contextuais bem como a utilização das regras de inferências. Esta ontologia é formada pelas seguintes classes, subclasses e relacionamentos.

Classes, Subclasses e Atributos para Coleta e Notificação de Contextos

- Contexto: A classe contexto armazena os dados coletados pelo sensores como: nodo, data/hora, usuario, identificador do Contexto de Interesse, sensor, valor coletado pelo sensor e valor que ocorreu alguma tradução. Nesta classe ficam instanciados as informações contextuais do Repositório de Informação Contextual do Gerente de Interpretação do EXEHDA-SS.

- Contexto_Notificado: Esta classe contém os contextos notificados com os seguintes atributos: identificador, aplicacao, componente, adaptador e usuario.

- ContextoNotificado_Sensor: Esta subclasse possui identificação e o sensor que gerou uma notificação, além do valor coletado pelo respectivo sensor, nodo e valor que tenha ocorrido processo de tradução.

- ContextoNotificado_Deduzido: São armazenados nesta subclasse os identificadores das regras de dedução e valores deduzidos pertencentes a notificação.

As classes Contexto_Notificado, ContextoNotificado_Sensor e ContextoNotificado_Deduzido armazenam as informações contextuais processadas e deduzidas no Repositório de Contexto Notificado do Gerente de Notificação do EXEHDA-SS.

Relacionamentos para Coleta e Notificação de Contextos

- Nodo_Contexto: Relacionamento entre Nodo e Contexto.

- Contexto_Sensor: Relacionamento entre Contexto e Sensor.

- CN_Usuario: Relacionamento entre a classe Contexto_Notificado e Usuario.

- CN_Sensor: Estabelece a relação da classe ContextoNotificado_Sensor e Sensor.

- CN_ContextoNotificadoSensor: Relacionamento de Contexto_Notificado com ContextoNotificado_Sensor.

- CN_ContextoNotificadoDeduzido: Relacionamento de Contexto_Notificado com ContextoNotificado_Deduzido.

Classes e Atributos do Contexto de Interesse das Aplicações

- Contexto_Interesse: Contém o identificador, aplicação, componente e adaptador.

- Sensor_Public: Possui identificador, descrição, intervalo do tempo de medicação entre os sensores, taxa de flutuação mínima dos dados a serem publicados pelos sensores, valor inferior e superior, valor *default* do sensor, regra para tradução de dados coletados e número máximo de publicações a serem armazenadas no Repositório de Informação Contextual.

- Contexto_Deduzido: Compõem a esta classe dois atributos: identificador e a regra de dedução. Essas regras são processadas pelo Motor de Inferência do Gerente de Interpretação do EXEHDA-SS gerando contextos deduzidos.

Relacionamento do Contexto de Interesse das Aplicações

- CI_SensorPublic: Relacionamento entre a classe Contexto_Interesse e Sensor_Public. Esse relacionamento especifica aos contextos de interesse os respectivos sensores com seus parâmetros operacionais para ativação e instanciação de dados coletados.

- SensorPublic_Sensor: Relacionamento entre a classe Sensor_Public e Sensor. Relaciona-se os parâmetros especificados com os sensores.

- CI_ContextoDeduzido: Relacionamento entre a classe Contexto_Interesse e Contexto_Deduzido. Esse relacionamento especifica para os contextos de interesse as regras para produção de contextos deduzidos.

5.2 Modelagem da Arquitetura de Software

O EXEHDA-SS enquanto um módulo do EXEHDA prevê a inclusão de serviços e componentes nos 'EXEHDAnodos' e no 'EXEHDAbase', conforme apresentado na figura 4.2. A figura 5.3 mostra a visão da proposição da integração do EXEHDA-SS ao Subsistema de Reconhecimento de Contexto e Adaptação do EXEHDA.

Nos 'EXEHDAnodos' são executados os componentes responsáveis por fornecer uma interface de ativação, configuração e capturação de informações dos sensores. Enquanto que no 'EXEHDAbase' são executados os componentes e serviços responsáveis pelas tarefas de:

- Instanciar e manipular a OntUbi, que armazena e indexa o conhecimento referente aos contextos coletados no ambiente ubíquo pelos EXEHDAnodos;

- Estabelecer comunicação com os EXEHDAnodos ativos na EXEHDAcél e extrair as informações sensoreadas disponíveis;

- Traduzir e deduzir informações contextuais;

- Gerenciar registros de Contextos de Interesse das Aplicações;

- Perceber alterações de contexto e notificar aos demais serviços do *middleware* EXEHDA;

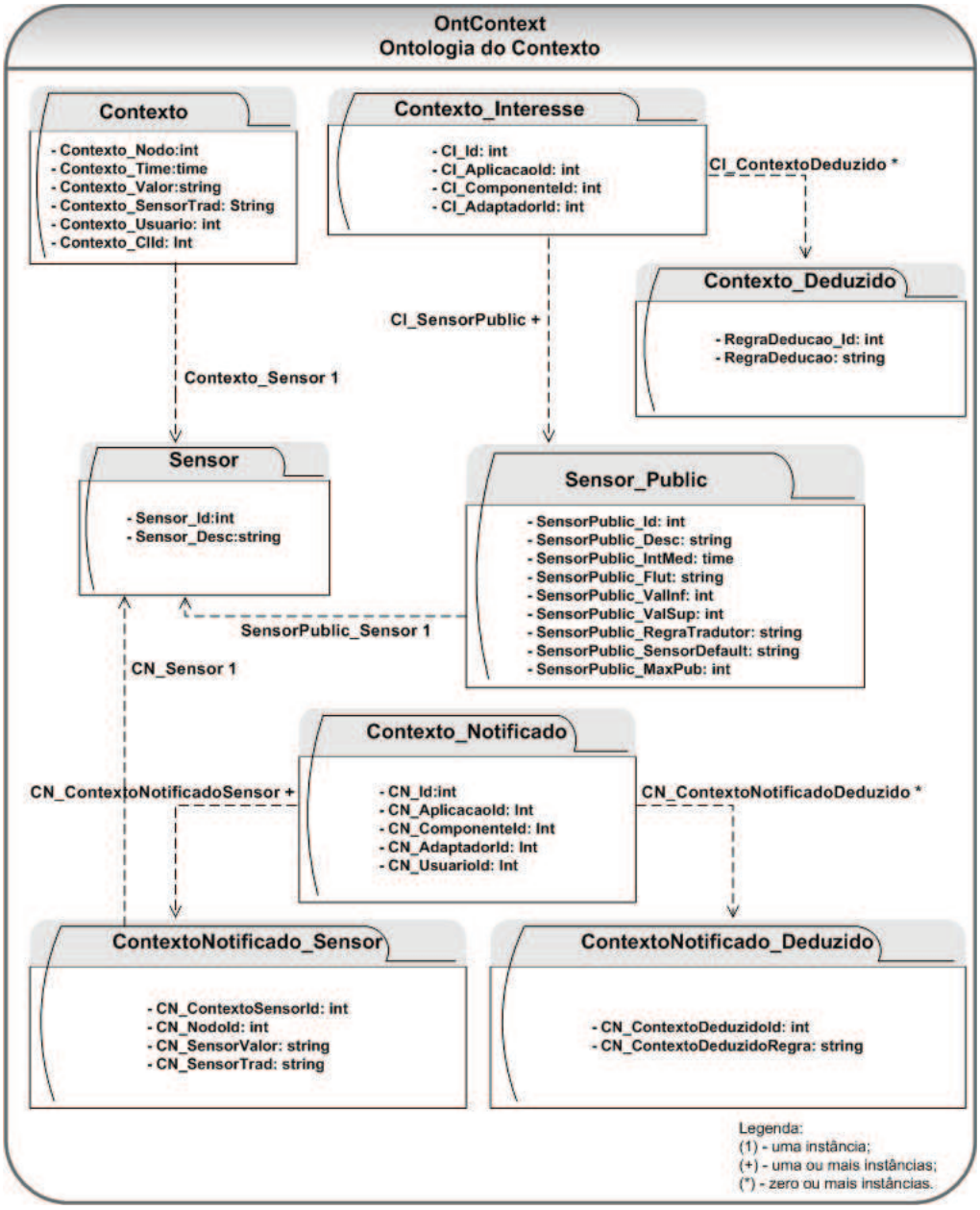

Figura 5.2: Classes da OntContext

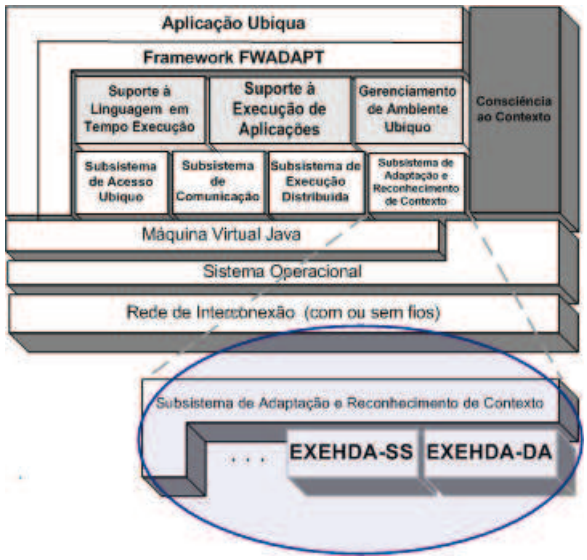

Figura 5.3: Integração do EXEHDA-SS ao Subsistema de Adaptação e Reconhecimento de Contexto do *Middleware* EXEHDA [adaptado de (YAMIN, 2004)]

- Receber subscrições de aplicações e notificar as informações contextuais sensoreadas e deduzidas.

As aplicações são configuradas para a adaptação pelo desenvolvedor no FWADAPT (*Framework para definição da Política de Adaptação Dinâmica dos Componentes da Aplicação*). O objetivo principal do framework é instanciar as Políticas das Adaptações das Aplicações (WARKEN, 2009) e os Contextos de Interesse das Aplicações.

A arquitetura de software do EXEHDA-SS precisa ser alimentada por Contextos de Interesse das Aplicações para que as mesmas funcionem colaborativamente com o EXEHDA-DA. Esses contextos de interesse são responsáveis por caracterizar os aspectos que devem ser considerados nos procedimentos de monitoração do ambiente ubíquo, de interpretação dos dados capturados e das respectivas notificações.

No Contexto de Interesse das Aplicações ficam definidas que informações contextuais serão adquiridas pelos sensores, traduzidas e deduzidas. Essas especificações são relacionadas por aplicação, componente e adaptador, considerando os seguintes parâmetros e regras:

- Instanciação dos sensores que participam das aplicações e adaptações;
- Parâmetros operacionais para ativação e publicação dos sensores;
- Regras para tradução dos dados adquiridos pelos sensores;
- Especificação do número máximo de registros pelos sensores e armazenados no repositório contextual;
- Regras de Inferência para deduções sobre os dados coletados pelos sensores.

Em linhas gerais, o EXEHDA-SS prevê a captura das informações contextuais a partir de sensores de software e/ou hardware. Entende-se como contribuição central deste trabalho a preposição do emprego de suporte semântico na realização de ta-

refas de manipulação e dedução sobre as informações contextuais obtidas, e a decorrente notificação das mesmas ao serviço de adaptação do EXEHDA.

Uma visão geral da arquitetura de software para o EXEHDA-SS é ilustrada na figura 5.4. O servidor de contexto é composto por três serviços: o Gerente de Aquisição de contexto, o Gerente de Interpretação de contexto e o Gerente de Notificação. Cada um destes gerentes é responsável por uma etapa do processamento do contexto, desde sua aquisição até o momento em que: tanto (i) o servidor de adaptação é notificado, (ii) como as aplicações que tenham interesse em manipulação de dados contextuais de forma direta.

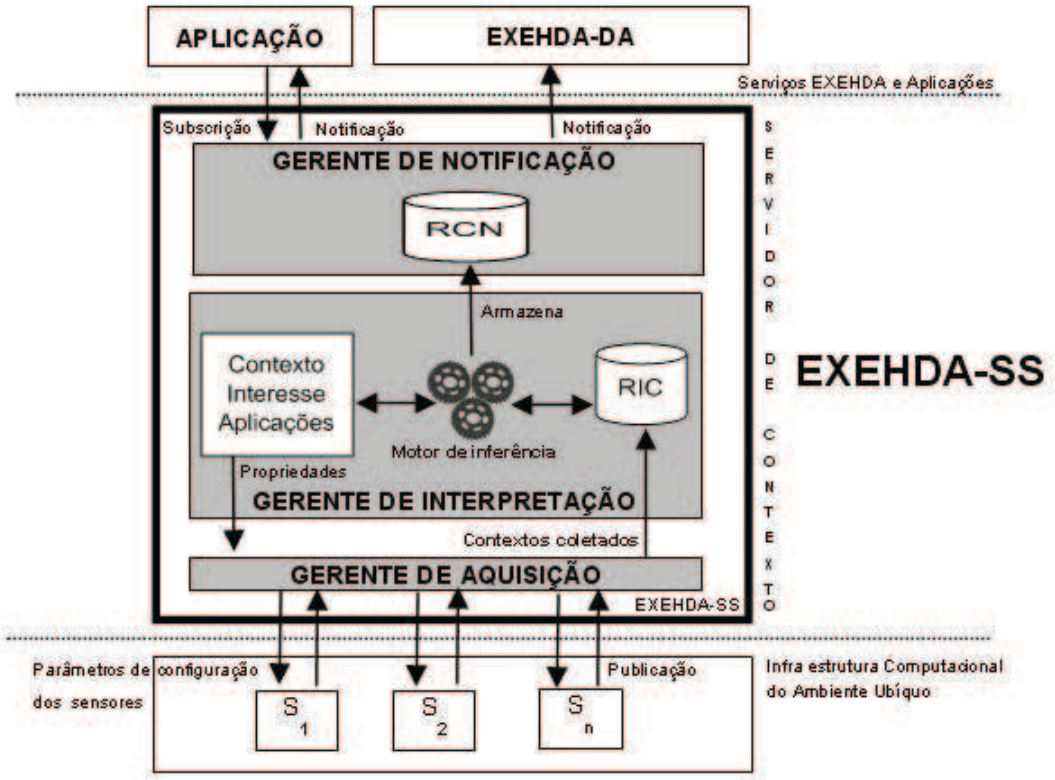

Figura 5.4: Visão Geral da Arquitetura de Software do EXEHDA-SS

No servidor de contexto proposto são utilizados gerentes autônomos e cooperantes para a realização de tarefas de manipulação e dedução sobre o contexto. A tarefa de dedução sobre os dados coletados através dos sensores com o intuito de produzir dados de contexto de mais alto nível é feita pelo o Gerente de Interpretação de contexto. Três são as principais funções que foram definidas para este gerente: (i) manter o Repositório de Informações Contextuais, que armazena os contextos capturados pelo Gerente de Aquisição; (ii) utilizar um Motor de Inferência para processamento e dedução sobre as informações de contexto mantidas no Repositório de Informações Contextuais e nos Contextos de Interesse das Aplicações; (iii) alimentar o Repositório de Contexto Notificado, que armazena os estados dos contextos disponibilizados pelo Gerente de Notificação.

Para a implementação do servidor de contexto do EXEHDA-SS, foram definidas classes na linguagem Java, apresentadas na tabela 5.1. A classe *ExehdaSS_coletor* é responsável por inicializar toda a infra-estrutura necessária para o funcionamento dos mecanismos definidos na arquitetura de software do EXEHDA-SS. Ela é responsável por efetuar a leitura do arquivo OWL onde está definido o modelo de representação ontológica

do EXEHDA-SS (subseção 5.1), bem como de cada publicação realizada pelos EXEHDAnodos descrito pelo arquivo *SensorPublicacao*. Além disso, é sua tarefa instanciar os Repositórios Contextuais presentes nos Gerentes do EXEHDA-SS, traduzir, processar e deduzir contextos, notificar alterações nos estados dos contextos aos demais serviços do EXEHDA e receber subscrição das aplicações e notifica-las.

Tabela 5.1: Classes Java do Servidor de Contexto do EXEHDA-SS

Classe	**Descrição**
ExehdaSS_coletor	Classe principal do Gerente de Aquisição.
ExehdaSS_escutaporta	Finalidade de escutar a porta 20202/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, identificando os respectivos contextos de interesse.
ExehdaSS_coletalixo	Objetivo de remover da memória os objetos *ExehdaSS_atende* que se encontrarem em estado não operacional.
ExehdaSS_individuo	Armazena na OntUbi as informações sensoreadas que são trazidas pela classe *ExehdaSS_atende*, realizando traduções de acordo com regras de tradução específicas.
ExehdaSS_gerInterpretacao	Implementa o Gerente de Interpretação; avalia contextos armazenados na ontologia, identificando ativações e alterações. Gera informações contextuais adicionais via procedimentos de dedução.
ExehdaSS_gerNotificacao	Implementa o Gerente de Notificação. É a classe responsável por enviar as notificações de alteração de contexto às aplicações e aos demais serviços do EXEHDA.
ExehdaSS_Subscricao	Recebe parâmetros para disponibilização de contextos notificados que foram processados e deduzidos.

Uma descrição resumida da arquitetura de software prevista para o EXEHDA-SS é feita a seguir.

5.2.1 Gerente de Aquisição de Contexto

O Gerente de Aquisição de contexto tem como função central na arquitetura proposta para o EXEHDA-SS prover a captura de informações de contexto, disponibilizando as mesmas em um formato adequado para que o Gerente de Interpretação possa implementar suporte semântico utilizando os mesmos.

O Gerente de Aquisição envolve funções cujos fundamentos estão discutidos na seção 2.2.4, a proposição deste gerente a ser concebido pelo EXEHDA-SS é descrita abaixo:

- Processar os contextos de interesse da aplicação, extraindo as informações para sua operação;

- Disparar no ambiente ubíquo os diversos sensores necessários para atender as demandas da aplicação em questão;

- Pré-processar as informações brutas dos sensores em dados normalizados convertendo seus dados de contexto considerando o interesse da aplicação;

- Disponibilizar as informações capturadas ao Gerente de Interpretação de contexto.

As informações de contexto são informações que dizem respeito a uma entidade de contexto, podendo ser estáticas ou dinâmicas. Uma informação de contexto é dita estática se o seu valor não varia com o decorrer do tempo (tamanho e a escala de cores do *display* de um celular). Uma informação de contexto dinâmica pode variar seu valor com o decorrer do tempo (largura de banda, qualidade do sinal, localização, temperatura, entre outros).

As informações contextuais podem ser obtidas de fontes diversas de forma explícita ou implícita. 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. Informações contextuais do ambiente físico incluem a presença física do usuário e de outras pessoas próximas a ele, a localização geográfica dessas pessoas, dos dispositivos e dos recursos disponíveis, entre outras.

A seguir serão apresentados etapas para sensoreamento de dados contextuais, publicação das informações sensoreadas, tradução e instanciação.

5.2.1.1 Sensoreamento de Dados Contextuais

Como mencionado anteriormente, o Gerente de Aquisição é responsável por prover a captura das informações contextuais, a partir de sensores de software e/ou *hardware* adquiridas de diferentes fontes (ex: Sistema de Informação, Sistema Operacional, dispositivo de *hardware*).

Nos EXEHDAnodos, sensores podem ser implantados por meio de arquivos de configuração. Desta forma, um sensor pode ser adicionado, removido ou substituído por outra implementação mais adequada ao dispositivo no qual o serviço de suporte semântico será utilizado.

Para a aquisição dos contextos através de sensores e publicação de dados a partir dos mesmos se faz necessário:

- Especificar intervalos de tempo entre medições;

- Registar flutuação mínima para que aconteça a publicação;

- Definir a faixa na qual os valores dos sensores deverão ser publicados.

Como um dos requisitos para que os sensores sejam ativados e publiquem contextos ao Gerente de Aquisição, foi definido na OntContext um conjunto de classes para representação ontológica que estão especificadas na classe *Contexto_Interesse*. Cada contexto de interesse possue uma relação de sensores com seus parâmetros operacionais, além disso, ficam definidas na classe *Sensor_Public*, regras para tradução das informações

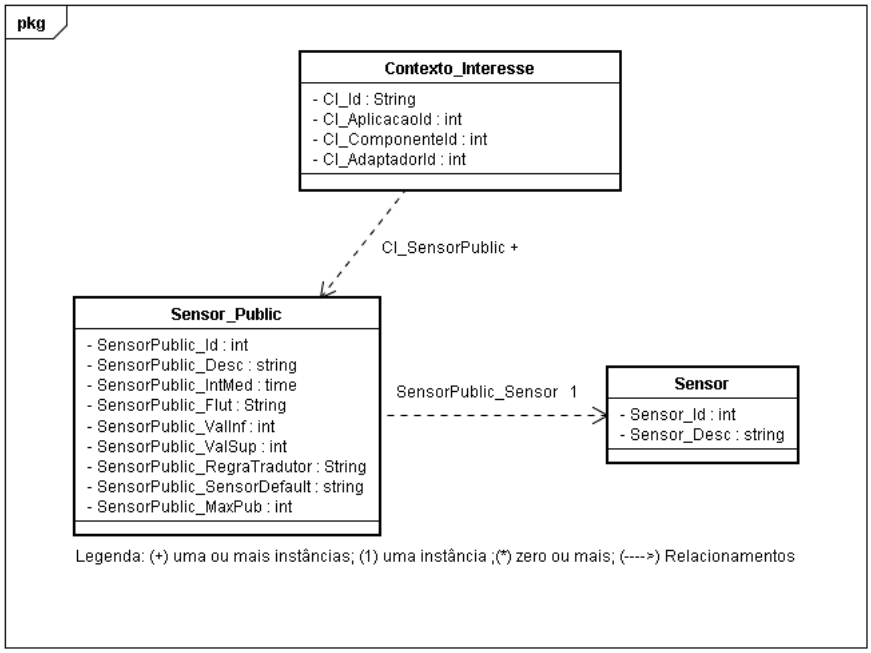

Figura 5.5: Classes da OntContext para Operação dos Sensores

coletadas por estes sensores e o máximo de instâncias armazenadas dos sensores no Repositório de Informação Contextual. A figura 5.5 ilustra as classes de representação ontológica para publicação de dados contextuais pelos sensores.

A aplicação ao ser instalada no EXEHDAnodo, é carregado junto a ela o arquivo SensorConfiguracao - figura 5.6, com parâmetros para inicialização e publicação de contextos pelos sensores ao Gerente de Aquisição.

O arquivo de configuração deverá ser utilizado para configurar os sensores de acordo com seus parâmetros operacionais definidos pelo desenvolvedor no Contexto de Interesse das Aplicações. Esses parâmetros são especificados com informações para publicação, tais como: identificador do sensor, frequência de publicação, flutuação mínima, faixa mínima e máxima.

```
<?xml version="1.0" enconding="UTF-8"?>
<Sensors>
<Sensor identificador="codigo_sensor1" intMed="frequencia_publicacao"
flut="flutuacao_minima" valInf="faixa_minima" ValSup="faixa_maxima"/>
<Sensor identificador="codigo_sensor2" intMed="frequencia_publicacao"
flut="flutuacao_minima" valInf="faixa_minima" valSup="faixa_maxima"/>
</Sensors>
```

Figura 5.6: Exemplo de Configuração do Arquivo SensorConfiguracao

5.2.1.2 Publicação das Informações Sensoreadas

A publicação das informações sensoreadas pelos EXEHDAnodos ocorre através da coleta de contextos pelos sensores. Ao atender os parâmetros especificados pelo SensorConfiguracao, a informação contextual é publicada ao Gerente de Aquisição.

Desta forma, foi definida a classe java *ExehdaSS_sensorCliente*, que conecta-se periodicamente ao Gerente de Aquisição em execução no EXEHDAbase para lhe enviar informações produzidas pelos sensores. Esse procedimento ocorre através da publicação do arquivo SensorPublicacao - figura 5.7.

```
<?xml version="1.0" enconding="UTF-8"?>
<EXEHDA-SS>
<Nodo IP="200.18.78.10" Nodo_Id="codigo" Nodo_Usuario="Usuario01"/>
<Contextos>
<Contexto Contexto_Valor="26°" Contexto_Times="2009-10-18T22:16:14"
Contexto_Sensor="nome_sensor1"/>
</Contextos>
<\EXEHDA-SS>
```

Figura 5.7: Exemplo de Configuração do Arquivo SensorPublicacao

Compõem a estrutura do arquivo SensorPublicacao as informações contextuais descritas a seguir:

- IP: endereço do EXEHDAnodo;
- Nodo_Id: código do EXEHDAnodo;
- Nodo_Usuario: login do usuário;
- Contexto_Valor: o valor sensoreado;
- Contexto_Times: data/hora realizada a coleta;
- Contexto_Sensor: identificador do sensor.

5.2.1.3 Tradutor e Instanciador Contextual

O processo de tradução e instanciação contextual, figura 5.8, é responsável por receber os dados sensoreados especificadas na notação XML adquiridos no processo de sensoreamento, traduzi-los e instancia-los no Repositório de Informação Contextual.

A classe java *ExehdaSS_escutaporta* tem a finalidade de escutar a porta 20202/tcp da EXEHDAbase e instanciar objetos *ExehdaSS_atende* para cada conexão de entrada aceita nesta porta, de onde são providos as informações publicadas pelos sensores ao Gerente de Aquisição.

O Tradutor e Instanciador Contextual é uma subcamada definida no Gerente de Aquisição, responsável pelo pré-processamento das informações contextuais brutas e instancia-las no Repositório de Informação Contextual do Gerente de Interpretação do servidor de contexto do EXEHDA-SS. Três são as principais funcionalidades desenvolvidas:

- Identificar os contextos de interesse envolvidos nas publicações realizadas no sensoreamento;
- Traduzir as informações de acordo com as regras definidas no Contexto de Interesse das Aplicações;

Figura 5.8: Tratamento dos Dados Sensoreados

- Instanciar as informações contextuais coletadas na classe Contexto da OntContext.

O Parser utilizado na arquitetura de software do Gerente de Aquisição do EXEHDA-SS é responsável por instanciar objetos *ExehdaSS_atende* identificando também, os contextos de interesse respectivos de cada sensor publicado. Este processo é realizado através de uma execução em SPARQL - vide figura 5.9, onde é passado o identificador do sensor e são retornados os contextos de interesse pertencentes ao sensor informado.

```
"PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ont: <http://www.owl-ontologies.com/Ontology1251223167.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?cia " +
WHERE { ?cia rdf:type ont:Contexto_Interesse
?cia ont:ContextoInteresse_SensorPublic ?senpub
?senpub ont:SensorPublic_Sensor ont:INSTANCIA_DO_SENSOR}"
```

Figura 5.9: Exemplo de Identificação de Contextos de Interesse nas Publicações Realizadas pelos Sensores

A tradução é um processo definido pelo desenvolvedor onde cada informação coletada por um sensor, deve ser atribuída uma regra específica para que sofra a tradução. Essa especificação fica definida no atributo *SensorPublic_RegraTradutor* da classe Sensor_Public - figura 5.5.

Para a instanciação dos dados de contexto no Repositório de Informações Contextuais foi utilizado a classe java *ExehdaSS_individuo* que converte os dados coletados

em valores semânticos especificados em formato OWL. Entretanto, para que não haja um grande de número de informações desnecessárias, foi definida na arquitetura de software do Gerente de Aquisição presente no EXEHDAbase um Controlador de Armazenamento de Contextos.

A principal função do Controlador é garantir no Repositório de Informação Contextual um número máximo de armazenamento de registros coletados do respectivo contexto de interesse sensoreado. Processadas as traduções, o controlador verifica no atributo *SensorPublic_MaxPub* o número máximo de instanciações e instancia no repositório. Caso, o número de publicação seja superior, o controlador exclui do repositório a instância com maior tempo de publicação e armazena a atual informação no repositório.

No Repositório de Informação Contextual são instanciados as informações coletadas dos sensores, além de outros atributos como: identificador do sensor, valor coletado pelo sensor, valor que tenha ocorrido tradução contextual, contexto de interesse pertencente ao sensor, nome do EXEHDAnodo, usuário e horário da coleta.

Por fim, após obter os contextos de interesse pertencentes a uma determinada publicação, realizar refinamentos, controlar número de instanciações das informações contextuais no Repositório de Informações Contextuais e instanciar os dados coletados, o Gerente de Interpretação é acionado disponibilizando os contextos de interesse para a execução do Motor de Inferência do Gerente de Interpretação presente no servidor de contexto do EXEHDA-SS.

5.2.2 Gerente de Interpretação de Contexto

O Gerente de Interpretação de contexto tem como principal função realizar tarefas de manipulação e dedução das informações contextuais, utilizando para isto informações especificadas nos Contextos de Interesse das Aplicações.

As informações contextuais manipuladas pelo Gerente de Interpretação seguem o vocabulário especificado na OntUbi, ontologia do ambiente ubíquo, detalhada na Seção 5.1.1. O Gerente de Interpretação de contexto mantém um repositório de contextos coletados baseado nesta ontologia, descrito a seguir:

Repositório de Informações Contextuais (RIC): neste repositório são armazenados as informações contextuais obtidas pelo Gerente de Aquisição de contexto, mantendo um histórico dos contextos para que possam ser processados e deduzidos pelo Motor de Inferência deste Gerente de Interpretação.

Os objetivos do Gerente de Interpretação do contexto incluem:

- Manter consistente o Repositório de Informação Contextuais, gerenciar o raciocínio sobre as informações contextuais mantidas neste repositório, inferindo novos contextos a partir de regras lógicas de inferência e de fatos contextuais definidos no Contexto de Interesse das Aplicações;

- Atualizar o repositório contextual, de modo a manter um histórico. Esse histórico servirá como uma base de aprendizado que possibilite melhorar a inferência em interações futuras;

- Raciocinar sobre os fatos mantidos no RIC, produzindo novos fatos a partir de regras lógicas pré-definidas no Contexto de Interesse das Aplicações;

- Processar alterações nos estados dos contextos coletados, armazenado-as no Repositório de Contexto Notificado do Gerente de Notificação.

A seguir são detalhadas as etapas para processamento do Gerente de Interpretação, motor de inferência e detalhamento da interpretação contextual.

5.2.2.1 Motor de Inferência de Contexto

Como descrito anteriormente, o EXEHDA-SS utiliza um Motor de Inferência que fornece um mecanismo de inferência sobre as informações de contexto baseado em ontologias e em regras, para o emprego de suporte semântico ao EXEHDA-SS. A seguir são descritos o motor de inferência baseado em ontologias e em regras, seguido das considerações constatadas durante o processo de prototipação.

Inferência Baseada em Ontologias

Em um processo de inferência baseado em ontologias, inferem-se informações de contexto a partir da combinação semântica definida pelos construtores da linguagem em que uma ontologia é construída e de um conjunto de fatos instanciados na ontologia.

O Motor de Inferência de contexto utilizado pelo Gerente de Interpretação realiza inferência sobre ontologias codificadas na linguagem OWL. A máquina de inferência *OWLReasoner* (JENA, 2009) implementa deduções obtidas de ontologias baseadas em RDF Esquema, bem como diferentes subconjuntos da linguagem OWL.

No caso de ontologias OWL, como foi concebida a OntUbi, diferentes graus de expressividade lógica podem ser aceitos. Desta forma, a configuração apropriada de uma máquina de inferência OWL depende dos construtores OWL envolvidos no processo de inferência.

Inferência Baseada em Regras

O processo de inferência baseado em regras utiliza uma sintaxe de construção de regras adotado pelo subsistema de inferência da API Jena. Essa sintaxe incorpora construtores específicos que podem ser utilizados como predicados. Demais predicados correspondem a termos do vocabulário definido na OntUbi.

O EXEHDA-SS utiliza no motor de inferência do Gerente de Interpretação o raciocinador baseado em regras *Generic rule reasoner* (JENA, 2009), que suporta a criação de regras definidas pelo usuário. Na arquitetura de software do EXEHDA-SS, essas regras ficam definidas pelo programador das aplicações na classe Contexto_Deduzido no Contexto de Interesse das Aplicações.

Para uma aplicação sensível ao contexto utilizar o mecanismo de inferência baseado em regras, o serviço de inferência de contexto deve ser invocado com dois parâmetros de entrada: uma lista de parâmetros de configuração da máquina de inferência e um grafo RDF contendo um conjunto de fatos instanciados a partir do modelo ontológico definido na OntUbi. A figura 5.10 exibe um arquivo com duas regras definido os termos *dualcore* e *quadcore*, no qual é definido que para ser *dualcore* é necessário que o atributo *Num_Proc* possua dois processadores e *quadcore* possua quatro processadores.

```
@prefix ont: <http://localhost/Ontologia.owl#>
@include <RDFS>

[dualcore: (?n ?p ont:Nodo) (?n ont:Num_Proc 2) -> (?n rdf:type ont:dualcore)]
[quadcore: (?n ?p ont:Nodo) (?n ont:Num_Proc 4) -> (?n rdf:type ont:quadcore)]
```

Figura 5.10: Exemplo de Definição de Regras

Considerações sobre as Máquina de Inferência

Durante a etapa de modelagem e prototipação verificou-se que a máquina de inferência baseada em ontologias *OWLReasoner* é necessária para que aconteceçam consultas em SPARQL sobre classes, subclasses e relacionamentos das ontologias. A não utilização de uma máquina de inferência impossibilita inferências nas subclasses e relacionamentos.

Desta forma, o mecanismo de inferência baseado em ontologias do EXEHDA-SS infere sobre a estrutura de classes, subclasses e relacionamentos e o mecanismo de inferência baseado em regras, produz quando definido pelo desenvolvedor das aplicações, contextos deduzidos.

5.2.2.2 Detalhamento da Interpretação Contextual

Após uma publicação de contexto por um sensor o Tradutor e Instanciador Contextual do Gerente de Aquisição identifica e informa o contexto de interesse - *CI_Id*. A classe *ExehdaSS_gerInterpretacao* recebe esse parâmetro e chama o motor de inferência do Gerente de Interpretação.

O Motor de inferência, identifica os sensores e as regras definidas pertencentes ao contexto de interesse repassado - vide figura 5.2 e executa dois possíveis processamentos:

- Ler no Repositório de Informação Contextual da classe Contexto da OntContext os valores coletados pelos sensores envolvidos do respectivo contexto de interesse em processamento - inferência baseada em Ontologias. Caso ainda não tenha ocorrido uma publicação de informação contextual pelos sensores envolvidos do respectivo contexto de interesse, é passado o valor *default*, definido no atributo *SensorPublic_SensorDefault* da classe Sensor_Public;

- Processar as regras definidas pelo desenvolvedor na classe Contexto_Deduzido da OntContext produzindo desta forma contextos deduzidos - inferência baseada em regras.

Finalizando o processo, o Gerente de Interpretação, instância no Repositório de Contextos Notificados do Gerente de Notificação, os valores de contextos coletados pelos sensores e as deduções, pertencentes ao seu respectivo contexto de interesse.

Na classe Contexto_Notificado da OntContext ficam armazenados os atributos pertencentes ao contexto de interesse processado pelo Motor de Inferência do EXEHDA-SS. Os atributos que compõem a notificação são: aplicação, componente e adaptador. Nas subclasses ContextoNotificado_Sensor e ContextoNotificado_Deduzido ficam gravados os

sensores, valores coletados, valores que ocorreram traduções, usuário, nodo e o identificador das regras de deduções com seus respectivos valores deduzidos. A figura 5.11 apresenta o fluxo de interpretação contextual do motor de inferência do EXEHDA-SS.

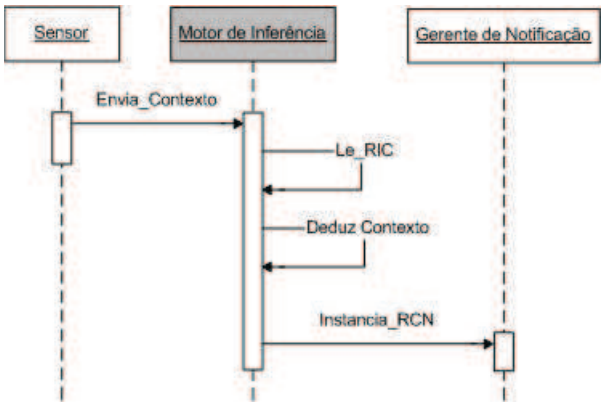

Figura 5.11: Fluxo do Motor de Inferência do EXEHDA-SS

5.2.3 Gerente de Notificação

Esse gerente é responsável por entregar os contextos processados e armazenados pelo Gerente de Interpretação no Repositório de Contextos Notificados (RCN).

No RCN ficam instanciados por aplicação, componente e adaptador, os contextos deduzidos e as informações coletadas pelos sensores pertencentes a uma publicação realizada por um contexto de interesse. Essa informação é notificada ao serviço de adaptação, de acordo que as mesmas forem sofrendo alterações nos seus estados. Quando o atributo *CI_AdaptadorId* da classe *Contexto_Interesse*(vide figura 5.5) possuir valor '0', é que a operação não possui adaptação, logo o processamento semântico não notificará ao serviço de adaptação do *middleware* EXEHDA.

Uma outra funcionalidade do Gerente de Notificação é receber subscrição pelas aplicações e repassa-las ao Gerente de Interpretação do EXEHDA-SS, afim de que possam ser notificadas pelo Gerente de Notificação para as mesmas.

A seguir são detalhadas as etapas para notificação ao serviço de adaptação e subscrição e notificação pelas aplicações.

5.2.3.1 Notificação ao Serviço de Adaptação

A notificação para o serviço de adaptação do *middleware* EXEHDA ocorre através do envio da instância do contexto notificado ao mesmo. Na OntContext estão definidas classes para armazenamento de contextos notificados, nessas classes ficam registrados por identificação, a aplicação, componente, adaptador, usuario, contexto deduzido, nodo e os valores produzidos pelos sensores.

O atributo *CN_Id* da classe Contexto_Notificado contém a instância que deverá ser repassada ao serviço de adaptação no momento em que forem acontecendo alterações nos estados dos contextos através da publicações das informações contextuais sensoreadas e deduzidas no Motor de Inferência do Gerente de Interpretação do EXEHDA-SS.

O serviço de adaptação ao receber a instância que contém alterações contextuais no ambiente ubíquo, faz uma leitura dos demais valores do contexto notificado e inicia as tarefas de execução previstas no mecanismo de adaptação.

5.2.3.2 Subscrição e Notificação das Aplicações

Uma outra funcionalidade prevista para o EXEHDA-SS é que as aplicações possam realizar subscrições no Gerente de Notificação. Para isso é necessário que a aplicação esteja cadastrada pelo desenvolvedor com seus contextos de interesse definidos.

A aplicação deve invocar o método *ExehdaSS_Subscricao(app,userId,ipSubscricao)*, passando três parâmetros: código da aplicação, usuario e o ip do nodo que solicitou a subscrição. O subscritor presente no Gerente de Notificação, recebe esses parâmetros e dispara uma leitura no Repositório de Contexto Notificado que encaminha para a aplicação um arquivo contendo os contextos da respectiva subscrição - vide figura 5.12. Esta forma de execução definida para o **Subscritor** faz com que o serviço de sensibilidade ao contexto do EXEHDA-SS possa ser utilizado diretamente pelas aplicações.

```
<?xml version="1.0" enconding="UTF-8"?>
<EXEHDA-SS>
<Contexto_Notificado UsuarioId="AFD10-4589E">
<Aplicacao AplicacaoId="10">
<Sensor SensorId="21" SensorDesc="Sensor de Temperatura" SensorValor="28°" />
<Sensor SensorId="22" SensorDesc="Sensor de Batimento" SensorValor="180" />
<Sensor SensorId="23" SensorDesc="Sensor Pressão Arterial" SensorValor="19_11" />
<ContextoDeduzido ContextoDeduzidoId="e_cardiaco"
RegraDeduzidoValor="Risco de Parada Cardíaca"/>
</Aplicacao>
</Contexto_Notificado>
</EXEHDA-SS>
```

Figura 5.12: Exemplo de Notificação as Aplicações

5.3 Considerações Sobre o Capítulo

Neste capítulo foram apresentadas a concepção e modelagem do EXEHDA-SS, sendo descrita a modelagem de arquitetura de software, bem como os Gerentes de Aquisição, Interpretação e Notificação.

Também foi discutido nesse capítulo, o modelo de representação de contexto: OntUbi e OntContext. O modelo ontológico definido na arquitetura de Software do EXEHDA-SS possibilita que as informações contextuais sejam coletas, traduzidas, processadas, deduzidas e notificadas às aplicações e aos demais serviços do *middleware* EXEHDA. O capítulo seguinte apresenta estudos de casos voltados para validação do EXEHDA-SS.

6 EXEHDA-SS: TECNOLOGIAS UTILIZADAS E ESTUDO DE CASO

Neste capítulo estão resumidos os principais aspectos das tecnologias utilizadas, bem como o estudo de caso empregado na avaliação das funcionalidades do EXEHDA-SS. O estudo de caso contemplou tarefas referente ao sensoreamento e coleta de informações contextuais, processamento, dedução e notificação dos dados de contexto, aos demais serviços do *middleware*, e também as aplicações que venham a se subscrever no EXEHDA-SS.

Este capítulo contempla duas subseções, a primeira apresentando as principais tecnologias utilizadas e a segunda estudo de caso.

6.1 Principais Tecnologias Utilizadas

Esta subseção apresenta as principais tecnologias empregadas na concepção do EXEHDA-SS, enquanto mecanismo de sensibilidade ao contexto com suporte semântico. Dentre estas tecnologias destacaríamos a (i) linguagem Java, por ser a linguagem definida para o desenvolvimento no *middleware* EXEHDA; (ii) JSF um *framework* voltado ao desenvolvimento de aplicações em JAVA; (iii) a API Jena para manipulação e dedução no modelo de representação ontológico; (iv) a linguagem SPARQL para realização de consultas na ontologia; e o (v) editor de ontologias Protégé. A seguir uma breve caracterização das principais tecnologias empregadas pelo EXEHDA-SS.

6.1.1 Java

Applets, Servlets, JavaBeans e aplicações Java são todos tipos de programas em Java e bem conhecidos dos profissionais da área de Computação. A independência de plataforma que Java oferece, a grande quantidade de bibliotecas disponíveis e a existência de maquinas virtuais embutidas em vários dispositivos tornou a linguagem Java uma tecnologia chave para o desenvolvimento de software na computação ubíqua.

Java originou-se como parte de um projeto de pesquisa que visava a criação de um software avançado que atendesse a uma extensa variedade de maquinário de redes e sistemas embutidos. O objetivo inicial era desenvolver um ambiente operacional pequeno, confiável, portável e distribuído. Inicialmente, a linguagem escolhida foi C++. Porém, a com o passar do tempo, as dificuldades encontradas com C++ aumentaram até o ponto em que os problemas poderiam ser melhor endereçados se fosse criada uma linguagem completamente nova.

Deste modo, Java foi projetada para atender a vários requisitos desejáveis em uma linguagem de programação, como por exemplo, confiabilidade, devido ao seu gerenciamento de memória, o que resulta em um ganho de redigibilidade e reuso de código.

6.1.2 JSF

JavaServer Faces é um *framework* para o desenvolvimento de aplicações Web, que permite o desenvolvimento de aplicações para a internet de forma visual. O JSF é atualmente considerado pela comunidade Java como um ótimo recurso em termos de desenvolvimento de aplicações Web utilizando Java (JSF, 2009).

Algumas características: (i) fornece um conjunto de tags JSP para acessar os componentes; (ii) reutiliza componentes da página; (iii) associa os eventos do lado cliente com os manipuladores dos eventos do lado do servidor, neste sentido os componentes de entrada possuem um valor local representando o estado no lado servidor.

6.1.3 API Jena

A API Jena é um *framework* desenvolvido na linguagem de programação Java pela empresa HP (*Hewlett-Packard*) para construção de aplicações que empreguem suporte semântico. Inicialmente Jena foi desenvolvida nos laboratórios de pesquisa para Web Semântica da HP e posteriormente disponibilizada como projeto de software livre. Jena fornece ambiente de programação para RDF, RDF-Schema, DAML+OIL e OWL, incluindo motores de inferência baseados em regras. Os dois objetivos principais da arquitetura Jena são (WILKINSON KEVIN; SAYERS, 2004):

- Permitir ao programador da aplicação representações flexíveis e múltiplas dos grafos RDF. Dessa forma, facilita a manipulação dos dados nos grafos e o seu acesso possibilitando ao programador da aplicação navegar nas estruturas de triplas;
- Tornar a visão do grafo RDF simples ao programador que deseja expor seus dados como triplas.

Fontes de triplas no Jena podem ser disponibilizadas, por exemplo, em bases de dados ou em memória. Adicionalmente, as triplas podem ser alcançadas, como resultado de processos de inferência aplicados a outras fontes de triplas (WILKINSON KEVIN; SAYERS, 2004). Aplicações que utilizam Jena geralmente interagem com um modelo abstrato, que traduz operações de mais alto nível em operações de baixo nível sobre triplas armazenadas em um tipo de grafo RDF.

6.1.3.1 Mecanismo de Inferência

O sistema de inferência do Jena é projetado para permitir que um grande número de motores de inferência seja utilizado. Alguns motores permitem criar novas informações derivadas das informações já existentes nas ontologias. Outros permitem realizar checagens de consistências sobre as ontologias, verificando se todos os axiomas definidos nas próprias ontologias são obedecidos. A distribuição Jena já inclui alguns raciocinadores pré-definidos, os principais são (TEAM, 2006):

- Raciocinadores OWL, OWL Mini e OWL Micro: constituem um conjunto de raciocinadores úteis para verificação de consistência, porém incompletos da linguagem OWL Lite;

- Raciocinador DAML micro: usado internamente no Jena para fornecer suporte a inferência em ontologias descritas em DAML;

- Raciocinador de Regra Genérico: raciocinador baseado em regras que é usado para implementar raciocinadores, para dados descritos em RDFS e OWL. O programador tem a possibilidade de definir suas próprias regras em grafos RDF.

Além dos raciocinadores já descritos, Jena permite checagem de consistência sobre ontologias OWL DL através do uso de raciocinadores DL externos, tais como *Pellet* (SIRIN EVREN; PARSIA, 2004), *Racer* (HAARSLEV V.; MOLLER, 2001) ou FaCT (HORROCKS, 2003) (BECHHOFER, 2006).

6.1.4 Linguagem de Consulta SPARQL

Para que as consultas possam ser realizadas e as informações possam ser extraídas, a linguagem SPARQL disponibiliza uma sintaxe que, funciona de maneira similar à linguagem SQL (*Structured Query Language*). A linguagem SPARQL possibilita ordenação de seqüências baseado em condições, limitação de seqüências e definições do tipo de dados RDF entre outras características. A seguir são apresentadas as principais cláusulas que compõem a linguagem SPARQL.

- SELECT: essa cláusula permite selecionar quais informações serão retornadas como resultado da consulta. As informações são armazenadas em variáveis que são identificadas pelo sinal de interrogação (?);

- WHERE: permite especificar as restrições para a realização das consultas. Essas restrições seguem o formato de tripla ¡sujeito, predicado, objeto¿, que podem ser formadas tanto por um objeto quanto por um valor literal;

- FILTER: restringe o conjunto de soluções de acordo com uma ou mais expressões. As expressões podem ser funções e operações construídas sintaticamente. Os operandos dessas funções e operadores são um subconjunto dos tipos de dados do XML Schema (xsd:string, xsd:decimal, xsd:double, xsd:dateTime) e tipos derivados de xsd:decimal;

- ORDER BY: captura uma seqüência de solução e aplica sobre ela condições de ordenação. Uma condição de ordenação pode ser uma variável ou a chamada a uma função. A direção de ordenação é ascendente por padrão. Pode-se explicitamente informar a direção de ordenação em ascendente e decrescente, através de ASC e DESC;

- LIMIT: limita o número de soluções retornadas. Se o número de soluções reais é maior do que o limite, então no máximo o número limite de soluções será retornado.

6.1.5 Protégé

A complexidade envolvida na manipulação de ontologias introduz a necessidade da utilização de ferramentas focadas para criação do modelo de representação ontológico concebido para o EXEHDA-SS. Essas ferramentas objetivam o armazenamento, manutenção e consulta em ontologias. Desta forma, para a criação da OntUbi e a OntContext (vide seção 5.1), utilizou-se o Protégé nas tarefas de criação de classes, subclasses, relacionamentos e atributos.

O Protégé é um ambiente interativo para projeto de ontologias, de código aberto, que oferece uma interface gráfica para edição de ontologias e uma arquitetura para a criação de ferramentas baseadas em conhecimento. Desenvolvido no Stanford Medical Informatics, departamento de informática da área de medicina da Universidade de Stanford tinha como principal objetivo ser utilizado em aplicações na área médica. Fatores como a sua facilidade de utilização e independência de domínio fez com que fosse também utilizado em aplicações de muitas outras áreas.

O Protégé tem uma arquitetura a qual podem ser integradas diversas funcionalidades através de plugins. Com essa capacidade o Protégé teve vários componentes auxiliares desenvolvidos e adicionados ao sistema, grande parte desses componentes foram construídos por grupos de usuários do ambiente, dentre os mais utilizados está o Jambalaya, um utilitário com animação e recursos para visualização de dados instanciados em ontologias (PROTEGE, 2008).

6.2 Estudo de Caso: Cenário Direcionado à Medicina

Este estudo de caso consiste em uma aplicação sintética direcionada à área médica denominada Acompanhamento Ubíquo de Pacientes - AUP, cujas funcionalidades destacadas nesta seção foram concebidos com o objetivo de explorar o mecanismo de sensibilidade ao contexto com suporte semântico provido pelo EXEHDA-SS.

O EXEHDA-SS está inserido no Projeto PertMed, que também é, atualmente, um dos focos do G3PD. Entende-se como uma das contribuições da AUP a possibilidade da mesma vir a ser utilizada pelo PertMed, uma vez que os seus objetivos contemplam o mesmo foco do projeto.

As publicações de dados dos sensores foram produzidas por softwares que instanciaram valores pré-definidos no servidor de contexto do EXEHDA-SS. Tanto os valores praticados, quanto os níveis de alerta, bem como as regras são aproximações médias sem compromisso de traduzir uma realidade médica precisa.

6.2.1 Objetivos da AUP

A premissa buscada é qualificar o acompanhamento de pacientes, que não estejam internados em Unidades de Tratamento Intensivo (UTI). Particularmente, com o intuito de aumentar o período de tempo que o paciente permaneça, sob cuidado dos agentes de saúde, é prevista a operação da AUP também em dispositivos móveis. Nesta perspectiva, os objetivos contemplados na aplicação AUP são:

- Exibir dados de pacientes adquiridos dinamicamente por mecanismo de sensoreamento de sinais;
- Emitir, de forma automatizada, diferentes níveis de alertas, em função dos dados

sensoreados, para os agentes de saúde;

- Integrar o serviço de alertas da aplicação a rede aberta de comunicação *Google Talk*;
- Prover possibilidade de uso, tanto a partir de dispositivo móveis, como de mesa;
- Permitir acesso ubíquo ao histórico dos dados sensoreados dos pacientes por agentes de saúde.

A aplicação AUP está organizada para ser integrada ao Sistema de Informações do Hospital. Nesta etapa de teste foi utilizada uma base de dados privada, sobre a qual foi possível ter liberdade no que diz respeito à manipulação dos dados utilizados.

As informações cadastrais são de três naturezas: (i) dados cadastrais de pacientes; (ii) dados de agentes de saúde; (iii) dados aquisitados dinamicamente pela rede de sensores de sinais vitais de pacientes.

Na versão da AUP, utilizada para avaliação do EXEHDA-SS, os agentes de saúde podem ser tanto enfermeiros quanto médicos. Quando necessário, os mesmos podem disparar o Módulo de captura dinâmica dos dados correspondentes aos sinais vitais de pacientes. Os sinais considerados nesta versão são frequência cardíaca, pressão arterial e temperatura. Em função dos valores dos sinais coletados, são produzidos diferentes níveis de alertas aos agentes de saúde, por sua vez, dependendo do nível de alerta é disponibilizada a opção de envio de mensagem ao agente de saúde ou, ainda, no caso de alerta máximo, o envio incondicional da mensagem.

6.2.2 Configuração do EXEHDA-SS pela AUP

Para que a AUP esteja configurada no ambiente de execução do EXEHDA-SS é necessário que o desenvolvedor habilite a aplicação e seus contextos de interesse no modelo de representação ontológico provido pelo serviço de sensibilidade ao contexto.

Para isso, se faz necessário instanciar no modelo ontológico da OntContext (i) as informações da AUP na classe Aplicação; (ii) os contextos de interesse, classe Contexto_Interesse; (iii) os parâmetros operacionais e as regras de tradução dos sensores, classe Sensor_Public; e (iv) as regras de dedução, classe Contexto_Deduzido.

A AUP foi prototipada para ser executada com o suporte do serviço de adaptação do *middleware*. Nesta organização, o EXEHDA-SS através de seus gerentes, recebe os dados publicados pelos sensores instalados no EXEHDAnodos, realiza o processamento e notifica aos demais serviços do *middleware* para que possam ser realizadas as adaptações e execução da aplicação.

O código da AUP na classe Aplicação no modelo ontológico da OntContext, é especificado a seguir.

Classe Aplicação
Aplicacao_Id = 100
Aplicacao_Desc = Acompanhamento Ubíquo de Pacientes

Os contextos de interesse, no modelo ontológico da OntContext, são definidos pelo código da aplicação, componente e adaptador. O EXEHDA é baseado em componentes

para execução das aplicações, nesta perpectiva, o EXEHDA-SS trata dois dos componentes definidos na estrutura de componentes da AUP: (i) componente adaptativo 500, com dois adaptadores 001 e 002; e o (ii) componente não adaptativo 300. Os adaptadores são definidos, para que o serviço de adaptação do EXEHDA, possa ser notificado quando de uma mudança de contexto de interesse.

A seguir, serão detalhados os contextos de interesse, associados ao seu componente e adaptador.

6.2.2.1 Contextos de Interesse da AUP que promovem Adaptação

O componente 500 é responsável pelo Monitoramento de Pacientes. O adaptador 001 foi especificado para a geração de adaptação de Alertas Automáticos, enquanto que, o adaptador 002 para Reconhecimento de Dispositivos. A seguir são especificados os contextos de interesse definidos para promover adaptação.

Classe Contexto de Interesse
CI_Id = 001
CI_AplicacaoId = 100
CI_ComponenteId = 500
CI_AdaptadorId = 001

Classe Contexto de Interesse
CI_Id = 002
CI_AplicacaoId = 100
CI_ComponenteId = 500
CI_AdaptadorId = 002

6.2.2.2 Contexto de Interesse da AUP para Envio Automático de Mensagens

O envio Automático de Mensagens pelo EXEHDA-SS não requer uma adaptação, neste sentido, foi criado o contexto de interesse para o componente 300, responsável pelo envio de Mensagens. Como este componente não contempla procedimentos adaptativos é atribuído valor 000, conforme descrito abaixo.

Classe Contexto de Interesse
CI_Id = 003
CI_AplicacaoId = 100
CI_ComponenteId = 300
CI_AdaptadorId = 000

Na subseção a seguir será apresentada a configuração dos sensores utilizados na AUP, bem como, o relacionamento dos mesmos conforme seu respectivo contexto de interesse.

6.2.2.3 Configuração dos Sensores para a AUP

Os tipo de sensores utilizados para a aplicação Acompanhamento Ubíquo de Pacientes foram definidos para atender as demandas de Monitoramento de Pacientes e Reco-

nhecimento de Dispositivos. Desta forma, o protótipo foi concebido com os dados sensoreados por 3 tipos de sensores para prover o sensoreamento de dados vitais de pacientes: Frequência Cardíaca, Temperatura e Pressão Alta e um sensor para Reconhecimento de Dispositivos, conforme detalhado a seguir.

6.2.2.4 Sensores para Monitoramento de Pacientes

Os sensores utilizados pela aplicação para prover o monitoramento de pacientes são especificados na classe Sensor_Public da OntContext através do relacionamento com seu respectivo contexto de interesse e a definição das suas faixas operacionais.

É responsabilidade do desenvolvedor da AUP definir os parâmetros de intervalo de medição do sensor, flutuação, regra para tradução, valor *default* e máximo de instânciações a serem armazenadas no Repositório de Informação Contextual do Gerente de Interpretação do EXEHDA-SS.

As regras de tradução definidas para a AUP são responsáveis por caracterizar que a informação sensoreada apresenta sinais normais ou não. Foi utilizado a seguinte notação: (0) sinais normais e (1) fora da faixa de normalidade, conforme a tabela 6.1.

Tabela 6.1: Valor Traduzidos na AUP

Sensor	Sinais normais	Valor traduzido	Fora da faixa normalidade	Valor traduzido
Frequência Cardíaca	<= 100	0	>= 101	1
Temperatura	<= 37	0	>= 38	1
Pressão Alta	<= 180090	0	> 180090	1

A seguir são apresentadas as faixas operacionais definidas para o sensor de Frequência Cardíaca, Temperatura e Pressão Alta, tabelas 6.2, 6.3 e 6.4 respectivamente.

6.2.2.5 Sensores para Reconhecimento de Dispostivos

Como estratégia para acelerar os testes da aplicação AUP, particularmente no que diz respeito ao desenvolvimento de software para dispositivos móveis, foi utilizada a estratégia de disponibilizar os componentes de software através da tecnologia de Java Server Faces.

A aplicação contempla dois tipos de dispositivos: (i) como dispositivo móvel está previsto o emprego de um PDA com resolução de 320x240. Para os testes foi empregado o dispositivo Sharp Zaurus 5600 disponível pelo G3PD; (ii) equipamento do tipo Desktop com resolução mínima 800x600.

A tabela 6.5 a seguir, apresenta os parâmetros definidos pelo desenvolvedor da AUP no Contexto de Interesse das aplicações para prover Reconhecimento de Dispositi-

Tabela 6.2: Faixas Operacionais do Sensor de Frequência Cardíaca para AUP

CI_Id	001
SensorPublic_Id	001
SensorPublic_Desc	SP Frequência Cardíaca
SensorPublic_Sensor	100
SensorPublic_IntMed	00:05:00
SensorPublic_Flut	05
SensorPublic_RegraTradutor	[FC_ForaFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 100) (?c ont:Contexto_Valor ?cv) (greaterThan (?cv,100)) -> (?n rdf:type ont 1)] [FC_DentroFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 100) (?c ont:Contexto_Valor ?cv) (lessThan (?cv,101)) -> (?n rdf:type ont 0)]
SensorPublic_SensorDefault	0
SensorPublic_MaxPub	0050

Tabela 6.3: Faixas Operacionais do Sensor de Temperatura para AUP

CI_Id	001
SensorPublic_Id	002
SensorPublic_Desc	SP Temperatura
SensorPublic_Sensor	101
SensorPublic_IntMed	00:30:00
SensorPublic_Flut	02
SensorPublic_RegraTradutor	[TEMP_ForaFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 101) (?c ont:Contexto_Valor ?cv) (greaterThan (?cv,38)) -> (?n rdf:type ont 1)] [TEMP_DentroFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 101) (?c ont:Contexto_Valor ?cv) (lessThan (?cv,37)) -> (?n rdf:type ont 0)]
SensorPublic_SensorDefault	0
SensorPublic_MaxPub	0050

Tabela 6.4: Faixas Operacionais do Sensor de Pressão Alta para AUP

CI_Id	001
SensorPublic_Id	003
SensorPublic_Desc	SP Pressão Alta
SensorPublic_Sensor	102
SensorPublic_IntMed	02:00:00
SensorPublic_RegraTradutor	[PA_ForaFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 102) (?c ont:Contexto_Valor ?cv) (greater (?cv,180090)) -> (?n rdf:type ont 1)] [PA_DentroFaixa: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 102) (?c ont:Contexto_Valor ?cv) (lessThan (?cv,180090)) -> (?n rdf:type ont 0)]
SensorPublic_SensorDefault	0
SensorPublic_MaxPub	0050

vos na classe Sensor_Public da OntContext através do relacionamento com seu respectivo contexto de interesse.

Como a intenção deste sensor é reconhecer o tipo de dispositivo, foi prevista uma regra para Reconhecimento de Dispositivos, esta regra ao ser processada ela faz uma tradução identificando o tipo de dispositivo de acordo com sua resolução. Será atribuído (0) para desktop e (10) para PDA. Além disso, o programador definirá somente uma única informação sensoreada armazenada para este sensor. O valor *default* atribuído será 0 caso não aconteça alguma publicação por este sensor.

Tabela 6.5: Sensor de Reconhecimento de Dispositivos para a AUP

CI_Id	002
SensorPublic_Id	004
SensorPublic_Desc	SP Reconhecimento de Dispositivos
SensorPublic_Sensor	103
SensorPublic_RegraTradutor	[Disp_Desktop: (?n rdf:type ont: Nodo) (?n ont:Nodo_Dispositivo ?vtela) (?vtela rdf:type ont:Tela) (?vtela ont:Tela_Altura 600) (?vtela ont:Tela_Largura 800) -> (?n rdf:type ont 0)] [Disp_PDA: (?n rdf:type ont: Nodo) (?n ont:Nodo_Dispositivo ?vtela) (?vtela rdf:type ont:Tela) (?vtela ont:Tela_Altura 240) ?vtela ont:Tela_Largura 320) -> (?n rdf:type ont 10)]
SensorPublic_SensorDefault	0
SensorPublic_MaxPub	0001

6.2.3 Contexto Deduzido para Envio Automático de Mensagens

Esta subseção apresenta uma funcionalidade definida pelo desenvolvedor da AUP para produção de informações deduzidas pelo serviço de sensibilidade ao contexto, mais especificamente, o Gerente de Interpretação do EXEHDA-SS.

Anteriormente, foram especificados os parâmetros operacionais dos sensores da AUP, o que possibilida ao desenvolvedor, a aplicação de uma regra de inferência com dedução lógica. Essa regra foi definida na sintaxe do subsistema de inferência da API Jena. Desta forma, a regra fica armazenada na classe Contexto_Deduzido da OntContext relacionado ao contexto de interesse 003, para Envio Automático de Mensagens.

A tabela 6.6 apresenta a regra para dedução da AUP, no qual, caso o sensor de Frequência Cardíaca tenha batimentos superiores a 180 e Pressão Arterial superior a 240100 é inferido risco de infarto - RI.

Tabela 6.6: Regra de Dedução - Risco de Infarto

CI_Id	003
RegraDeducao_Id	001
RegraDeducao	[FC: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 100) (?c ont:Contexto_Valor ?cv) (greaterThan (?cv,180)) -> (?c rdf:type ont FCsup)] [PA: (?c rdf:type ont: Contexto) (?c ont:Contexto_Sensor 102) (?c ont:Contexto_Valor ?cv) (greaterThan (?cv,240100)) -> (?c rdf:type ont PAsup)] [RI: (?c rdf:type ont:FCsup) (?c rdf:type ont:PAsup) -> (?c rdf:type ont Risco_Infarto)]

6.2.4 Configuração dos Sensores nos EXEHDAnodos

Os sensores ao serem executados nos EXEHDAnodos, processam um arquivo de configuração, que contém os sensores e os parâmetros operacionais para ativação e publicação de informações, definidos pelo desenvolvedor da aplicação conforme detalhado nesta seção. A seguir são apresentados o arquivo com os parâmetros dos sensores de frequência cardíaca, temperatura, pressão alta e reconhecimento de dispositivos.

```
<?xml version="1.0" enconding="UTF-8"?>
<Sensors>
<Sensor identificador="100" intMed="00:05:00" flut="05"/>
<Sensor identificador="101" intMed="00:30:00" flut="02"/>
<Sensor identificador="102" intMed="02:00:00"/>
<Sensor identificador="103"/>
</Sensors>
```

Figura 6.1: Configuração dos Sensores para Monitoramento de Pacientes e Reconhecimento de Dispositivos na AUP

6.2.5 Publicação de Informações Contextuais ao EXEHDA-SS

A publicação das informações sensoreadas no Gerente de Aquisição do EXEHDA-SS (conforme seção 5.2.1.2) da aplicação Acompanhamento Ubíquo de Pacientes foram especificadas de acordo com os sensores definidos para a aplicação. A seguir segue as publicações realizadas pelo sensor de Frequência Cardíaca (figura 6.2), Temperatura (figura 6.3) e Pressão Arterial (figura 6.4), com seus respectivos parâmetros publicados: ID e IP do EXEHDAnodo, ID do usuário, ID e valor coletado pelo sensor e horário da coleta.

```
<?xml version="1.0" enconding="UTF-8"?>
<EXEHDA-SS>
<Nodo IP="200.18.68.50" Nodo_Id="1" Nodo_Usuario="300"/>
<Contextos>
<Contexto Contexto_Valor="200" Contexto_Times="2010-05-12T21:15:18"
Contexto_Sensor="100"/>
</Contextos>
<\EXEHDA-SS>
```

Figura 6.2: Publicação Realizada pelo Sensor de Frequência Cardíaca

```
<?xml version="1.0" enconding="UTF-8"?>
<EXEHDA-SS>
<Nodo IP="200.18.68.50" Nodo_Id="1" Nodo_Usuario="300"/>
<Contextos>
<Contexto Contexto_Valor="37" Contexto_Times="2010-05-12T21:16:00"
Contexto_Sensor="101"/></Contextos>
</Contextos>
<\EXEHDA-SS>
```

Figura 6.3: Publicação Realizada pelo Sensor de Temperatura

```
<?xml version="1.0" enconding="UTF-8"?>
<EXEHDA-SS>
<Nodo IP="200.18.68.50" Nodo_Id="1" Nodo_Usuario="300"/>
<Contextos>
<Contexto Contexto_Valor="180100" Contexto_Times="2010-05-12T21:16:14"
Contexto_Sensor="102"/><\EXEHDA-SS>
</Contextos>
<\EXEHDA-SS>
```

Figura 6.4: Publicação Realizada pelo Sensor de Pressão Alta

6.2.6 Processamento e Notificação de Contextos Obtidos pelo EXEHDA-SS

Esta subseção descreve como as etapas de aquisição, processamento e notificação das informações contextuais publicadas pelos sensores são definidas para a AUP.

Os gerentes do servidor de contexto do EXEHDA-SS realizam o processamento destes dados sensoreados, resumidos abaixo:

- Gerente de Aquisição: recebe os dados sensoreados pelos sensores, identifica os Contextos de Interesse da AUP pertencentes a estes sensores, realiza traduções, controla o número de instanciações no Repositório de Informações Contextuais, armazena no RIC e repassa ao Gerente de Interpretação o Contexto de Interesse da AUP publicado pelos sensores;

- Gerente de Interpretação: o Motor de Inferência recebe o Contexto de Interesse, identifica os valores coletados pelos sensores envolvidos e armazenados no RIC. Deduz a regra definida para Envio Automático de Mensagens e armazena no Repositório de Contextos Notificados;

- Gerente de Notificação: notifica ao serviço de adaptação e ao componente para Envio Automático de Mensagens da AUP.

A seguir são sintetizados a tradução dos dados sensoreados, processamento e notificação do EXEHDA-SS ao serviço de adaptação e ao processamento e notificação de contextos deduzidos.

6.2.6.1 Tradução dos Dados Vitais publicados pelos Sensores

A tabela 6.7 apresenta o valor das publicações dos sensores de Monitoramento de Pacientes e Reconhecimento de Dispositivos traduzidos e armazenados no Repositório de Informação Contextual da classe Contexto da OntContext, através da aplicação das regras de tradução definidas na subseção 6.2.2.3 processadas pelo Gerente de Aquisição do EXEHDA-SS.

Tabela 6.7: Classe Contexto

Propriedades	Frequência Cardíaca	Temperatura	Pressão Alta	Reconhecimento de Dispositivo
Contexto_CIId	001	002	003	004
Contexto_Sensor	100	101	102	103
Contexto_Valor	200	37	180100	320x240
Contexto_SensorTrad	1	0	1	10
Contexto_Nodo	1	1	1	1
Contexto_Time	2010-05-12T21:15:18	2010-05-12T21:16:00	2010-05-12T21:16:14	2010-05-12T21:15:18
Contexto_Usuario	300	300	300	300

6.2.6.2 Processamento e Notificação ao Serviço de Adaptação

O Motor de Inferência do Gerente de Interpretação recebe o contexto de interesse da informação contextual publicada ao EXEHDA-SS da AUP.

Para que possam ser disparadas as notificações ao serviço de adaptação, o Motor de Inferência baseado em ontologias, tem a função inicial de identificar os demais sensores envolvidos e realizar uma leitura no Repositório de Informação Contextual dos valores coletados pertencentes ao respectivo contexto de interesse.

A associação entre os dados vitais sensoreados e os Níveis de Alertas Automáticos notificados ao serviço de adaptação pode ser resumida conforme a seguir:

- Nível Alerta = 1: sinais normais. Interface de nível 1 (verde) aplicação com identificação e sinais do paciente: nome, especialidade, frequência, pressão e temperatura. Sinais normais - figura 6.5 (versão Desktop) e figura 6.9 (versão PDA).
- Nível Alerta = 2: início de problema. Frequência Cardíaca ou Temperatura fora do normal. Interface de nível 2 (amarelo) com aviso de cuidado - figura 6.6 (versão Desktop) e figura 6.9 (versão PDA).
- Nível Alerta = 3: problema médio. Apenas Pressão Arterial fora do normal, ou Frequência Cardíaca e Temperatura fora do normal. Interface de nível 3 (laranja) com mensagem de cuidado maior que nível 2, mensagem gtalk para enfermeira - 6.7 (versão Desktop) e figura 6.10 (versão PDA).
- Nível Alerta = 4: alerta máximo. Pressão Arterial fora do normal e Temperatura e/ou Frequência cardíaca fora normal. Interface nível 4 (vermelho) com mensagem de emergência, mensagem gtalk para enfermeira e médico - 6.8 (versão Desktop) e figura 6.10 (versão PDA).

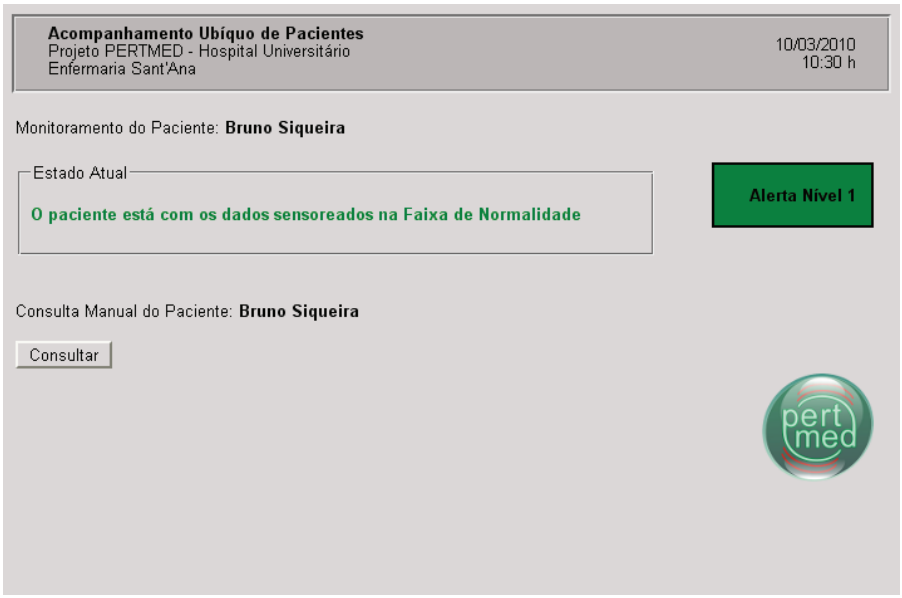

Figura 6.5: Nível de Alerta 1 - Versão Desktop

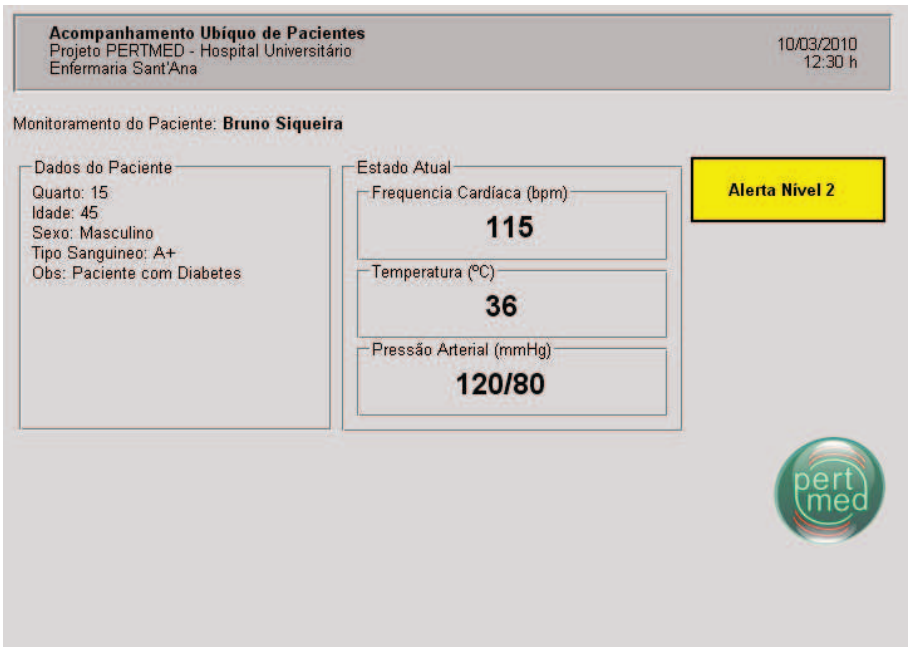

Figura 6.6: Nível de Alerta 2 - Versão Desktop

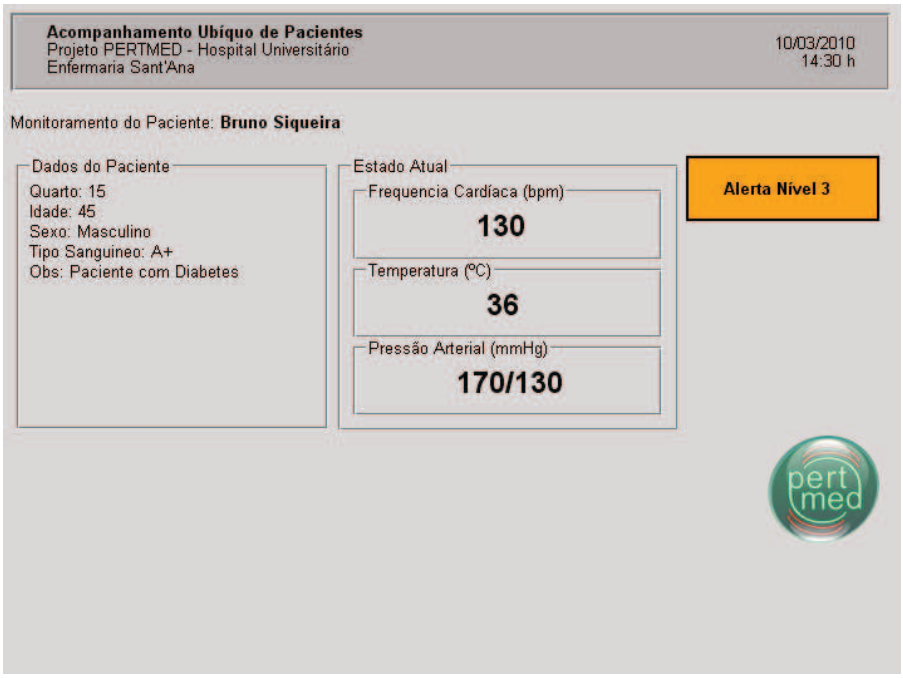

Figura 6.7: Nível de Alerta 3 - Versão Desktop

Figura 6.8: Nível de Alerta 4 - Versão Desktop

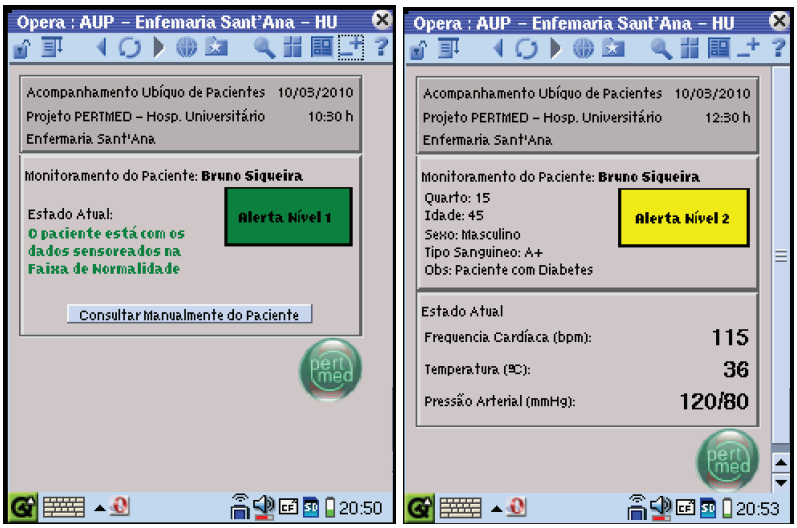

Figura 6.9: Nível de Alerta 1 e 2 - Versão PDA

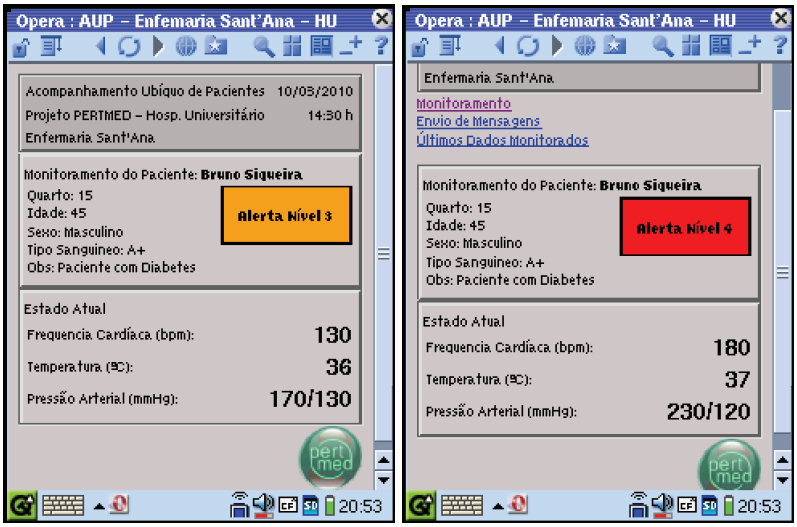

Figura 6.10: Nível de Alerta 3 e 4 - Versão PDA

No anexo A, é ilustrada a figura A.1 com a visão geral do dispositivo PDA Zaurus.

No que diz respeito a adaptação que irá produzir os diferentes níveis de alertas automáticos na AUP, o EXEHDA-SS, enquanto servidor de contexto produz um contexto notificado ao serviço de adaptação, armazenado no Repositório de Contextos Notificados do Gerente de Notificação, o qual representa a mudança de estado dos sensores de sinais vitais.

As instâncias do contexto notificado, atributo CI_Id, são repassadas ao serviço de adaptação. As classes, propriedades e valores do contexto notificado das classes Contexto_Notificado e ContextoNotificado_Sensor para Monitoramento de Pacientes estão descritas nas tabelas 6.8 e 6.9 e para Reconhecimento de Dispositivos na tabelas 6.10 e 6.11.

Tabela 6.8: Classe Contexto_Notificado - Monitoramento de Pacientes

Propriedade	**Valor**
CN_Id	001
CN_AplicacaoId	100
CN_ComponenteId	000500
CN_AdaptadorId	001
CN_UsuarioId	300

Tabela 6.9: Classe ContextoNotificado_Sensor

Propriedades	**Frequência Cardíaca**	**Temperatura**	**Pressão Arterial**
CN_ContextoSensorId	001	002	003
CN_Sensor	100	101	102
CN_SensorValor	200	37	180,100
CN_SensorTrad	1	0	1
CN_NodoId	1	1	1

Tabela 6.10: Classe Contexto_Notificado - Reconhecimento de Dispositivo

Propriedade	**Valor**
CN_Id	002
CN_AplicacaoId	100
CN_ComponenteId	000500
CN_AdaptadorId	002
CN_UsuarioId	300

Tabela 6.11: Classe ContextoNotificado_Sensor

Propriedades	**Reconhecimento de Dispositivos**
CN_ContextoSensorId	004
CN_Sensor	103
CN_SensorTrad	10
CN_NodoId	1

6.2.6.3 Processamento e Notificação de Contexto Deduzido para Envio Automático de Mensagens

Foi previsto que o EXEHDA-SS dispare de forma automatizada envio Automático de Mensagens aos agendes de saúde da AUP em função dos dados sensoreados de elevado risco.

Neste sentido, a AUP deve invocar o método da classe java *ExehdaSS_Subscricao* com os seguintes parâmetros: 100 - id da AUP na OntContext; 300 - id do usuário e IP do nodo onde está sendo executada a AUP. Desta forma, a AUP está a habilitada a ser notificada por regras de deduções (vide subseção 6.2.3) processadas pelo Motor de Inferência baseado em regras do Gerente de Interpretação e notificadas para AUP.

Para que a regra de dedução definida pelo desenvolvedor da AUP seja executada, o Motor de Inferência ao receber uma publicação de algum dos sensores (Frequência Cardíaca e/ou Pressão Alta) especificados na regra, dispara o processamento, deduz e armazena a regra de dedução no Repositório de Contexto Notificado disparando Envio Automático de Mensagens.

As tabelas 6.12 e 6.13 detalham as classes Contexto_Notificado e ContextoNotificado_Deduzido com suas propriedades deduzidos pelo Motor de Inferência baseado em regras do Gerente de Interpretação do EXEHDA-SS.

Tabela 6.12: Classe Contexto_Notificado - Contexto Deduzido

Propriedade	**Valor**
CN_Id	003
CN_AplicacaoId	100
CN_ComponenteId	00300

Tabela 6.13: Classe ContextoNotificado_Deduzido

Propriedades	**Envio Automático de Mensagens**
CN_ContextoDeduzidoId	001
CN_ContextoDeduzidoRegra	Risco_Infarto

6.3 Considerações Sobre o Capítulo

Este capítulo apresentou as principais tecnologias empregadas na concepção do mecanismo de sensibilidade ao contexto com suporte semântico e o estudo de caso de uma aplicação voltada para a área médica utilizada para validação do modelo de representação contextual e da modelagem da arquitetura de software do EXEHDA-SS.

Foram avaliados na Aplicação Ubíqua de Pacientes aspectos relacionadas a configuração da AUP no EXEHDA-SS, configuração dos sensores nos EXEHDAnodos, Publicação de Informações Contextuais ao EXEHDA-SS, Processamento e Notificação ao serviço de adaptação e ao Envio Automático de Mensagens através do processamento semântico de regras de dedução.

O capítulo seguinte, apresenta as considerações finais, trabalhos futuros e as publicações realizadas que culminaram no desenvolvimento desta dissertação.

7 CONSIDERAÇÕES FINAIS

Neste capítulo são resumidas as principais contribuições decorrentes do desenvolvimento deste trabalho. São discutidos os trabalhos relacionados, envolvendo sensibilidade ao contexto e suporte semântico, as principais contribuições do trabalho, trabalhos futuros e também são relacionadas as publicações realizadas.

Desde que Mark Weiser concebeu sua visão de ubiqüidade, importantes evoluções no *hardware* tem sido obtidas, permitindo a criação de dispositivos menores e mais portáteis, sensores e dispositivos de controle com crescente poder de processamento e padronização das tecnologias para comunicação sem fio. Com isso, dia-a-dia estão sendo criadas as condições para permitir a premissa básica da computação ubíqua, ou seja, o acesso do usuário ao seu ambiente computacional a qualquer hora, em qualquer lugar, independente de dispositivo.

Na Computação Ubíqua um aspecto fundamental relaciona-se ao monitoramento e a manipulação das informações contextuais. Neste sentido, a Computação Sensível ao Contexto é um paradigma computacional que se propõe a permitir que as aplicações tenham acesso e tirem proveito de informações que digam respeito às computações que realizam, na perspectiva de direcionar e/ou otimizar seu processamento.

Neste sentido, um sistema é sensível ao contexto, se ele usa o contexto para prover informações ou serviços ao usuário. Suas aplicações são capazes de modificar seu comportamento baseado nas informações de contexto ou são aplicações que disponibilizam ao usuário as próprias informações de contexto.

De modo geral, em Computação Sensível ao Contexto, várias pesquisas têm culminado em propostas de infra-estruturas para suportar o desenvolvimento de aplicações que antecipem as necessidades dos usuários e reajam automaticamente de forma pouco intrusiva diante das situações de contexto.

A utilização de tecnologias para processamento semântico no tratamento de informações de contexto traz como características: (a) a descrição formal, padrão e estruturada de cada dimensão semântica de informação de contexto; (b) o suporte à interoperabilidade sintática, estrutural e, principalmente, semântica entre aplicações sensíveis ao contexto; e (c) a capacidade de interpretar e inferir inter-relacionamentos com base nos conteúdos e descrições semânticas das entidades envolvidas.

As ontologias no EXEHDA-SS são utilizadas para representar os diferentes tipos de aplicações, serviços, dispositivos, usuários, entre outros. Além disso o uso de ontologias serve para modelar o reconhecimento e processamento das informações contextuais. O emprego de ontologias como técnica de modelagem é justificado pelas suas características de formalidade, semântica explícita e abstração de implementação, que são

necessárias para a modelagem das informações contextuais.

Considerando estas premissas, o trabalho realizado contemplou um modelo de representação ontológico expansível das informaçõs de contexto, que trata duas frentes específicas desta proposta: (i) uma relacionada a modelagem das informações referentes ao ambíente ubíquo e outra, (ii) no que diz respeito ao suporte de processamento para os gerentes que constituem o EXEHDA-SS.

7.1 Principais Contribuições

As atividade desenvolvidas ao longo deste trabalho, tendo por base os estudos realizados, a construção de um modelo de representação ontológica, a concepção de um mecanismo sensível ao contexto, permitiram a obtenção das seguintes contribuições:

- A arquitetura do EXEHDA-SS contempla o tratamento de eventos produzidos por contextos, tendo sido modelada para ser expansível, tanto no que diz respeito a captura de dados do ambiente ubíquo, bem como, quanto aos possíveis consumidores de contextos de interesse. Os consumidores podem ser as aplicações que se subscrevem no servidor de contexto, bem como outros serviços do *middleware*, dentre os quais destacaríamos o serviço de adaptação ao contexto;

- Concepção de um servidor de contexto, que contempla três serviços para atender as tarefas de aquisição de informações contextuais, sua interpretação e notificação de contextos de interesse;

- Modelo ontológico para representação contextual dos aspectos do ambiente ubíquo associados as aplicações. Este modelo está representado por uma ontologia denominada OntUbi;

- Modelo ontológico das informações coletas, processadas e notificadas pelo EXEHDA-SS, denominada OntContext. A OntContext é a ontologia responsável para estes processamentos concebidos pelo EXEHDA-SS;

- Especificação dos Contextos de Interesse das Aplicações representadas ontologicamente na OntContext;

- Definição de parâmetros operacionais para ativação e publicação de informações pertencentes as aplicações;

- Expansibilidade do modelo ontológico a outros domínios;

- Manipulação e dedução sobre dados contextuais através do Motor de Inferência baseado em regras.

Assim, o EXEHDA-SS mostra-se alinhado com o objetivo deste trabalho de prover suporte semântico nas tarefas de aquisição, interpretação e notificação das informações contextuais aos demais serviços do EXEHDA e/ou as aplicações. O emprego de ontologias se mostrou oportuno para prover o suporte semântico necessário a proposta do EXEHDA-SS.

7.2 Discussão dos Trabalhos Relacionados ao EXEHDA-SS

Na tabela 7.1, é feita uma comparação entre os principais projetos apresentados no capítulo 3, os quais foram utilizados como referência para concepção do EXEHDA-SS.

Tabela 7.1: Comparação do EXEHDA-SS com outros trabalhos relacionados (capítulo 3)

Funcionalides	A	B	C	D	E	F	G	H	I	J	K	EXEHDA-SS
Aquisição	X	X	X	X	X	X	X	X	X	X	*X*	*X*
Acesso e integração de dados											*X*	*X*
Apresentação da informação	X											*X*
Compartilhamento			X			X	X					*X*
Definição do comportamento da aplicação	X											*X*
Disseminação		X	X	X	X	X			X			*X*
Interpretação								X		X	*X*	*X*
Identificação de recursos		X	X	X		X		X				*X*
Raciocínio			X			X	X			X		*X*

Outros trabalhos, além do EXEHDA-SS, também fazem o uso do emprego de ontologias para representação das informações contextuais, entre eles: *Middleware* de Contexto do Gaia (C), *Context Aware Mobile Networks and Services* (E), *Service-Oriented Context-Aware Middleware* (F), *Context Broker Architecture* (G), *Mobile Collaboration Architecture* (H), *Framework* de Contexto (I), *Semantic Context Kernel* (J) e Infraware (K).

O emprego de ontologias permitiu ao EXEHDA-SS, enquanto servidor de contexto, realizar o processamento de regra de dedução dos dados contextuais, oito dos trabalhos estudados utilizam ontologia, sendo que quatro (*Middleware* de Contexto do Gaia (C), *Service-Oriented Context-Aware Middleware* (F), *Context Broker Architecture* (G) e *Semantic Context Kernel* (J)) possuem um mecanismo de dedução. Neste sentido, a adoção da linguagem de consulta SPARQL no EXEHDA-SS se mostrou fundamental para a que aconteça a dedução de contexto, sendo peça central na proposição do mecanismo para processamento das informações contextuais coletadas, armazenadas nos repositório de contexto do EXEHDA-SS .

O mecanismo de dedução baseado em regras e em ontologias para o processamento contextual proposto para o EXEHDA-SS é baseado na API JENA. A seleção de JENA é decorrente de sua significativa presença na literatura especializada, sendo utilizada em três (*Middleware* de Contexto do Gaia (C), *Context Broker Architecture* (G) e *Semantic Context Kernel* (J)) dos trabalhos pesquisados e principalmente por ter gerado resultados satisfatórios.

Em linhas gerais, o estudos dos trabalhos relacionandos além de prover uma

aproximação com as tecnologias empregadas na concepção do EXEHDA-SS, possibilitou uma sistematização das características a serem consideradas quanto da concepção dos componentes da arquitetura, em particular seus serviços, pelos seguintes aspectos descritos a seguir.

- Aquisição - realizada pelo Gerente de Aquisição, através do Tradutor e Instanciador Contextual;

- Acesso e integração de dados e Apresentação da informação - realizado pelo Gerente de Interpretação, empregando informações disponíveis no Repositório de Informação Contextual;

- Compartilhamento e Disseminação - Gerente de Notificação acessa as informações contextuais no Repositório de Contexto Notificado e notifica os interessados;

- Definição do comportamento da aplicação - definida no Contexto de Interesse das Aplicações na OntContext;

- Interpretação e Raciocínio - realizado no Gerente de Interpretação pelo Motor de Inferência;

- Identificação de recursos - os recursos são definidos na OntUbi.

Considerando que o EXEHDA-SS foi concebido como extensão do *middleware* EXEHDA, além destas características destacadas, ele mantém os aspectos arquiteturais do mesmo. Assim, sua arquitetura é baseada em serviços, cuja integração visa fornecer a infra-estrutura necessária para suporte à sensibilidade ao contexto com suporte semântico em um ambiente ubíquo.

7.3 Trabalhos Futuros

Dentre os aspectos levantados para continuidade do trabalho destacam-se:

- Proposição de um mecanismo baseado em banco de dados para persistência de dados factíveis de serem coletados;

- Inclusão de outros parâmetros para controle operacional dos sensores levando em conta o poder computacional dos mesmos, e suas restrições quanto ao consumo de energia;

- Expandir o mecanismo para construção de contextos que englobem várias células de execução do ambiente ubíquo provido pelo EXEHDA;

- Analisar o desempenho do EXEHDA-SS considerando diferentes domínios na representação ontológica especificada pela OntUbi.

7.4 Publicações Realizadas

- 10ª Escola Regional de Alto Desempenho - ERAD 2010. Luthiano Venecian, João Lopes, Adenauer Yamin, Luiz Palazzo, Iara Augustin. EXEHDA-SS: Uma Contribuição a Sensibilidade ao Contexto na Medicina Ubíqua.

- 9ª Escola Regional de Alto Desempenho - ERAD 2009. Luthiano R. Venecian, João L. B. Lopes, Adenauer C. Yamin, Luiz A. M. Palazzo. Uma Proposta Baseada em Web Semântica para Sensibilidade ao Contexto na Computação Ubíqua.

- 8ª Mostra de Pós-Graduação da Universidade Católica de Pelotas. Luthiano R. Venecian, Adenauer C. Yamin. EXEHDA-SS: Uma Contribuição a Sensibilidade ao Contexto na Medicina Ubíqua.

- 7ª Mostra de Pós-Graduação da Universidade Católica de Pelotas. Luthiano R. Venecian, Luis A. M. Palazzo, Adenauer C. Yamin. Sensibilidade ao Contexto na Computação Ubíqua utilizando a Web Semântica.

- Seminfo 2009. Christian P. Brackmann, Luthiano R. Venecian, Paulo R. G. Luzzardi, Adenauer C. Yamin. GingaSC: Uma Proposta de Sensibilidade ao Contexto para TV Digital Brasileira.

REFERÊNCIAS

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

AL., C. N. et. **Handling exceptional conditions in mobile collaborative applications**: As exploratory case study. [S.l.]: In: 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Entreprises, 2006. 347-142p.

BARDRAM J.; BOSSEN, C. Mobile Work - The Spatial Dimension of Collaboration at a Hospital. **Computer Supported Cooperative Work**, [S.l.], v.14, n.2, p.131–140, 2005.

BECHHOFER. **DL Implementors Group**. [S.l.]: Sean. Interface DIG, 2006.

BELOTTI, R. **Sophie - Context Modelling and Control**. [S.l.]: Diploma thesis, Swiss Federal Institute of Technology Zurich, 2004.

BELOTTI R., D. C. G. M. N. M. C. P. A. **Modelling Context for Information Environments**. [S.l.]: In: Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), 2004.

BELOTTI R., D. C. G. M. N. M. C. P. A. **Modelling Context for Information Environments**. [S.l.]: In: Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), 2004a.

BELOTTI R., D. C. G. M. N. M. C. P. A. **Interplay of Content and Context**. [S.l.]: In: International Conference on Web Engineering (ICWE 2004), 2004b.

BUBLITZ, F. M. **Front-Frame-based Ontology System**: Uma Ferramenta para Criação e Edição de Ontologias. Universidade Federal de Alagoas: [s.n.], 2005.

BULCAO NETO R. F., P. M. G. C. **Toward a Domain-Independent Semantic Model for Context-Aware Computing**. [S.l.]: In: Proceedings of the 3rdIW3C2 Latin American Web Congress, IEEE Computer Society, 2005. 61-70p.

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

CHEN, H. **An Intelligent Broker Architecture for Pervasive Context-Aware Systems**. [S.l.]: PhD Thesis, Faculty of the Graduate School of the University of Maryland, 2004.

CHEN H., F. T. J. A. P. Y. **UMBC eBiquity Project**: Context Broker Architecture (CoBrA). [S.l.]: http://ebiquity.umbc.edu/project/html/id/1/?EBS=0a25f6d5d3c8bd4f33fdb719933e0e03 - Acesso em 11/2008, 2005.

COSTA, C. A. da; YAMIN, A. C.; GEYER, C. F. R. Toward a General Software Infrastructure for Ubiquitous Computing. **IEEE Pervasive Computing**, Los Alamitos, CA, USA, v.7, n.1, p.64–73, 2008.

DAML. **The DARPA Agent Markup Language Homepage**. Disponível em: < http://www.daml.org >. Acesso em 02/2009.

DCMI. **Dublin Core Metadata Initiative - DCMI Metadata Terms**. Disponível em: < http://dublincore.org/documents/dcmi-terms/ >. Acesso em 03/2009.

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

FOAF. **FOAF Vocabulary Specification**. Disponível em: < http://xmlns.com/foaf/spec/ >. Acesso em 02/2009.

FREITAS, F. **Ontologias e a Web Semântica**. Anais do XXIII Congresso da Sociedade Brasileira de Computação: [s.n.], 2003. 1-52p.

GAUVIN M., B.-B. A. C. A. 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.

GRUBER, T. R. **A Translation Approach to Portable Ontologies**. Knowledge Acquisition: [s.n.], 2003. 199-220p.

GU T., W.-X. H. P. H. K. Z. D. Q. **An Ontology-based Context Model in Intelligent Environments**. [S.l.]: In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, 2004.

HAARSLEV V.; MOLLER, H. **Racer system description**. [S.l.]: International Joint Conference on Automated Reasoning, 2001. 701-705p.

HENRICKSEN K., I. J. **Developing Context-Aware Pervasive Computing Applications**: Models and Approach. [S.l.]: In: Pervasive and Mobile Computing Journal, 2005a.

HENRICKSEN K; INDULSKA, J. R. A. **Modeling context information in pervasive computing systems**. Zurich, Switzerland: PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, 2003. 167-180p.

HORROCKS. **Ian. Fact and ifact.** [S.l.]: Proceedings of the International Workshopon Description Logics, 2003. 133-135p.

HOUAISS, A. **Dicionário Houaiss da Língua Portuguesa**. [S.l.]: AB Editora, Rio de Janeiro, Ed. Objetiva, 2007.

IANNELLA, R. **Representing vCard Objects in RDF/XML**. Disponível em: < http://www.w3.org/TR/vcardrdf >. Acesso em 04/2008.

INFOEXAME. Smartphones, porque é hora de comprar um e aposentar seu celular. , [S.l.], n.257, Agosto 2007.

ISAM. **Infra-estrutura de Suporte às Aplicações Móveis**. Disponível em: < http://www.inf.ufrgs.br/ isam/ >. Acesso em 12/2008.

JENA. **Jena - A Semantic Web Framework for Java**. Disponível em: < http://jena.sourceforge.net/ >. Acesso em 02/2009.

JESS. **Jess - The Expert System Shell for the Java Platform**. Disponível em: < http://herzberg.ca.sandia.gov/jess/ >. Acesso em 03/2009.

JSF. **JSF - JavaServer Faces**. Disponível em: < http://pt.wikipedia.org/wiki/JavaServer_Faces >. Acesso em 10/2009.

KORPIPAA P., M. J. K. J. K. H. M. E. **Managing Context Information in Mobile Devices**. [S.l.]: IIEEE Pervasive Computing, 2003.

LI DING PRANAM KOLARI, Z. D. S. A. T. F. J. **Using Ontologies in the SemanticWeb**: A Survey. UMBC: [s.n.], 2005.

MOSTEFAOUI G. K., R. J. P. B. 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., A. A. **Context as a Dynamic Construct**. [S.l.]: Human Computer Interaction, 2003. 257-268p.

PEREIRA FILHO J. G.; PESSOA, R. M. C. C. Z. O. N. Q. C. R. R. M. B. A. C. P. F. C. R. G. L. M. M. **Infraware**: um Middleware de Suporte a Aplicações Móveis Sensíveis ao Contexto. [S.l.]: In: SBRC - SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 2006.

PERTMED, . **PERTMED - Sistema de TeleMedicina Móvel, disponibilizando a informação onde ela é necessária**. Disponível em: < http://pertmed.wkit.com.br/pertmed/doku.php >. Acesso em novembro de 2009.

PESSOA R. M., C. C. P. F. J. A. R. **Aplicação de um Middleware Sensível ao Contexto em um Sistema de Telemonitoramento de Pacientes Cardíacos**. [S.l.]: In: SEMISH - Seminário Integrado de Software e Hardware, 2006.

POOLE D., G. R. A. R. **Theorist**. [S.l.]: In: http://www.cs.ubc.ca/ poole/theorist.html, Acesso em 02/2009, 2006.

PROTEGE. **Protege**. Disponível em: < http://protege.stanford.edu/ >. Acesso em 05/2008.

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

ROMAN M., H. C. K. C. R. R. A. C. R. H. N. K. **Gaia**: A middleware infrastructure to enable active spaces. [S.l.]: IEEE Pervasive Computing, 2002.

ROSA M. G. P., B. M. R. S. S. 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.

RUDI STUDER V. RICHARD BENJAMINS, D. F. **Knowledge Engineering**: Principles and Methods. [S.l.: s.n.], 2003. 161p.

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

SACRAMENTO, V.; ENDLER, M.; RUBINSZTEJN, H. K.; LIMA, L. S.; GONCALVES, K.; NASCIMENTO, F. N.; BUENO, G. A. MoCA: A Middleware for Developing Collaborative Applications for Mobile Users. **IEEE Distributed Systems Online**, Los Alamitos, CA, USA, v.5, n.10, 2004.

SANTORO F. M., B. P. A. R. M. **Context Dynamics in Software Engineering Process**. [S.l.]: International Journal of Advanced Engineering Informatics, 2005.

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

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

SINDEREN, M. e. a. **Overall architecture of the AWARENESS infrastructure**. Disponível em: < http://www.freeband.nl/FreebandKC/documents?keyword_id=2432 >. Acesso em 02/2009.

SIRIN EVREN; PARSIA, B. **Pellet**: An owl dl reasoner. [S.l.]: In: In Proceedings of the 2004 International Workshop on Description Logics, http://dblp.uni-trier.de, Acesso em 01/2009, 2004.

SOWA, J. **Semantic Networks**. In Encyclopedia of Artificial Intelligence: [s.n.], 2006.

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

TEAM. **The Jena Development. Inference Engine User Manual**. Disponível em: < http://jena.sourceforge.net/inference/index.html >. Acesso em 01/2009.

TRUONG K. N., A. 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, 2003.

VARSHNEY, U. Pervasive Healthcare. **IEEE Computer**, [S.l.], v.36, n.12, p.138–140, 2003.

WALTENEGUS, D. **Dynamic Generation of Context Rules**. [S.l.]: Lecture Notes in Computer Science, 2006. 102-115p.

WANG X. H., G. T. Z. D. Q. P. 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.

WARKEN, N. **Uma Contribuição ao Controle da Adaptação na Computação Ubíqua**. [S.l.]: In: http://olaria.ucpel.tche.br/nelsiw/ - Acesso em 07/09, 2009.

WASP. **WASP Project**. Disponível em: < http://www.freeband.nl/kennisimpuls/projecten/wasp/ >. Acesso em 08/2008.

WEGDAM, M. **AWARENESS**: a project on Context AWARE NEtworks and ServiceS. [S.l.]: Proceedings of the 14th Mobile & Wireless Communications Summit 2005, 19-23, 2005.

WILKINSON KEVIN; SAYERS, C. K. H. A. R. D. **Efficient rdf storage and retrieval in jena2**. In Proceedings of VLDB Workshop on Semantic Web and Databases: [s.n.], 2004. Disponível em: < http://www.hpl.hp.com/techreports/2003/HPL-2003-266.pdf >. Acesso em 12/2008.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

ZHOU Y.; CAO, J. R. V. S. J. L. J. **A Middleware Support for Agent-Based Application Mobility in Pervasive Environments**. Washington, DC, USA: In: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, 2007. 9p.

ZIMMERMANN A., L. A. S. M. **Applications of a Context-Management System**. [S.l.]: In: Proceedings of the CONTEXT-2005555, 2005a. 569p.

ZIMMERMANN A., S. M. L. A. **Personalization and Context Management**. [S.l.]: User Modeling and User-Adapted Interaction, 2005b. 275-302p.

ANEXO A APLICAÇÃO AUP - TELA EM PDA

Figura A.1: Nível de Alerta 1 - Visão Geral do Dispositivo Zaurus