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

EXEHDA-SS: Um Mecanismo para Sensibilidade ao Contexto com Suporte Semântico

por
Luthiano Rodrigues Venecian

Relatório de Conclusão da Disciplina
Dissertação de Mestrado I

Orientador: Prof. Dr. Adenauer Corrêa Yamin

Pelotas, julho de 2009

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

AGRADECIMENTOS

Agradeço a Deus e aos meus pais pelo carinho e apoio constante em cada momento da minha vida.

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

Aos professores, funcionários e colegas do PPGInf-UCPEL, em especial a Nelsi, que além de ser uma pessoa talentosa, é uma amiga presente 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.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

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
RDF	Resource Description Language
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 menores e com maior poder de computação e comunicação o que potencializa a mobilidade do usuário portando seus equipamentos. Um Ambiente Ubíquo contém diferentes dispositivos, tais como sensores, atuadores e eletroeletrônicos em geral que interagem com a pessoa de forma natural. A diversidade de dispositivos e informações de um Ambiente Ubíquo assim constituído, introduz diferentes desafios para comunicação 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, seu processamento e armazenamento. Na perspectiva de atender as demandas da computação ubíqua, está sendo concebido o EXEHDA-SS que será responsável pela realização de tarefas de manipulação e raciocínio sobre as informações contextuais empregando suporte semântico.

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

TITLE: “EXEHDA SS: A MECHANISM FOR CONTEXT-AWARENESS WITH SEMANTIC SUPPORT”

ABSTRACT

This work has as main objective to propose a mechanism for sensitivity to the context in ubiquitous computing. With technological advances we have smaller devices with greater computing power and communication, that potencialize the user mobility with theirs equipments. An ubiquitous environment so constituted contains differents devices such as sensors, actuators, electronics in general that interact with the person in a natural way. The diversity of devices and information in ubiquitous environment introduces severals challengers for communication among the differents involved parts. So, when you build and run ubiquitous applications context awareness, you have a number of features that should be provided, involving since the acquisition of contextual information from the set of heterogeneous and distributed sources, by the representation of these informations, its processing and storage. In order to meet the demands of ubiquitous computing, is designed to EXEHDA SS, responsible for carrying out tasks of manipulation and reasoning on 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 torna-se invisível para o 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 uma aplicação de perceber características de seu ambiente, e é um requisito chave para permitir a adaptação em resposta às mudanças ambientais. Aplicações sensíveis ao contexto, onde contexto é ”qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, local ou objeto) que é considerada relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação (DEY, 2000).”, são aplicações que 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, utilizam sensores para tomar ciência de modificações que venham a acontecer no ambiente. Tais modificações são 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 prover cada vez mais um maior poder computacional, 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 nos perfis dos usuários (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).

A modelagem de contexto utilizando ontologias permite a definição do comportamento do ambiente ubíquo em tempo de execução, mas não fornece o suporte necessário para lidar com o dinamismo do ambiente. Uma das alternativas para lidar com o dina-

mismo é o uso de Serviços Web Semânticos.

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. Em vez de deixar essas funcionalidades a cargo da aplicação, as incorporando ao código do negócio, são utilizadas como infra-estruturas subjacentes as plataformas ou *middlewares* de provisão de contexto (ABOWD G. D.; RODDEN, 2002).

Essa dissertação de mestrado está inserida nos esforços de pesquisa do Projeto PERTMED (Sistema de TeleMedicina Móvel), 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 Web Semântica 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 definido no projeto ISAM(Infra-estrutura de Suporte às Aplicações Móveis Distribuídas) e promovido pelo *middleware* EXEHDA (*Execution Environment for Highly Distributed Applications*) (YAMIN, 2004).

O EXEHDA-SS está sendo concebido para ser responsável pelo processamento das informações contextuais, realizando tarefas de manipulação e raciocínio sobre o contexto, utilizando ontologias para a representação 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 contexto de 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 construção 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 aplicação, serão desenvolvidas aplicações da área médica.

A computação ubíqua tem como requisito a manipulação de diferentes contextos de execução. Um dos aspectos deste tema, a sensibilidade ao contexto, é considerada um dos grandes desafios desta área de pesquisa.

O desenvolvimento deste trabalho esta compreendendo estudos que visam com-

parar diferentes mecanismos de sensibilidade ao contexto, através do estabelecimento da relação existente entre computação ubíqua, sensibilidade ao contexto e tecnologias de web semântica.

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

1.2 Contexto de pesquisa

O contexto de pesquisa desta dissertação é composto por dois projetos que serão 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, 2009) propõe fazer a ponte entre os sistemas automatizados existentes (registro de pacientes, exames laboratoriais,...) 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.

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.

O *middleware* EXEHDA faz parte dos esforços de pesquisa do Projeto ISAM. O ISAM vem sendo desenvolvido por um consórcio de universidades gaúchas, e foi iniciado na UFRGS sob a coordenação do Prof. Cláudio Geyer (ISAM, 2007).

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 são 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 políticas de adaptação 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 por sensores, dispositivos 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 clientes 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.

1.4 Objetivos

O objetivo geral deste trabalho é explorar a correlação entre computação ubíqua, sensibilidade ao contexto e tecnologias da Web Semântica. 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 está sendo concebido como uma extensão de um mecanismo de sensibilidade ao contexto atualmente existente, contemplando o uso de suporte semântico no suporte a execução de aplicações sensíveis ao contexto, e tem por premissa ser integrado ao *middleware* EXEHDA.

Como objetivos específicos do trabalho em andamento, considerando o já realizado e o por realizar, destacaríamos:

- estudar os fundamentos teóricos sobre computação ubíqua e computação sensível ao contexto;

- revisar as plataformas sensíveis ao contexto em ambientes de execução para computação ubíqua;

- estudar tecnologias para suporte semântico com a utilização da web semântica;
- identificar os principais projetos em medicina que explorem recursos da computação ubíqua (medicina ubíqua);
- compreender a correlação entre computação ubíqua, sensível ao contexto e suporte semântico;
- estudar o *middleware* EXEHDA, revisando seus fundamentos, e reconstruindo as decisões inerentes a concepção dos diversos módulos de sua arquitetura;
- estudar métodos para integração de suporte semântico em um mecanismo de sensibilidade ao contexto a ser integrado ao *middleware* EXEHDA;
- perseguir a integração com grupos que trabalham com Computação Ubíqua no cenário nacional.

1.5 Estrutura do texto

A estrutura do texto deste trabalho contempla seis capítulos, sendo o primeiro esta introdução, e os outros cinco 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: Mecanismos de Sensibilidade ao Contexto, são apresentados a avaliação de onze trabalhos relacionados a sensibilidade ao contexto e suas interligações;
- Capítulo 4: Fundamentos do EXEHDA-SS, apresenta os fundamentos teóricos da web semântica, ontologias, inferências 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: Conclusão, são apresentadas as conclusões pertinentes deste trabalho.

2 SENSIBILIDADE AO CONTEXTO: PRINCIPAIS CONCEITOS

Este capítulo resume conceitos inerentes a área central do trabalho em desenvolvimento. Foi fruto de um estudo em abrangência que contemplou tanto aspectos de fundamentação, como de metodologias associadas.

Nas mais diversas situações do dia a dia as pessoas fazem uso do conhecimento do contexto para delimitar e direcionar ações e comportamentos. As mensagens trocadas para comunicação trazem junto um contexto associado que apóia a compreensão do seu conteúdo. 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 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, enumeraram exemplos de contextos. 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 defendida por (DEY, 2000):

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

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

Podemos ainda entender contexto como "circunstâncias ou situações em que uma tarefa computacional está inserida", definição extraída de (HENRICKSEN K; INDULSKA, 2002).

Freqüentemente usadas como sinônimo de contexto, as informações contextuais são nada mais que as informações que caracterizam um determinado contexto. São elas as informações relevantes para se determinar o estado atual de um contexto em questão. Considere o contexto localização, as informações contextuais referentes a esse contexto são, por exemplo, a latitude e a longitude de uma entidade ou em que sala de um edifício se encontra uma determinada pessoa. Nesse trabalho, os termos contexto e informações contextuais são utilizados de forma semelhante, mas fica clara a diferenciação entre esses dois conceitos.

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 na hora certa; e (vi) o tratamento da qualidade da informação contextual.

As próximas subseções apresentam o estudo 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 delimitar as ações dependentes de contexto nesses sistemas e identificar os elementos contextuais que caracterizam a situação na qual essas ações são executadas. Estudos mostram que a identificação dos elementos de contexto depende fortemente do tipo da tarefa e domínio em questão (OZTURK P., 2001). Por outro lado, o papel que o contexto desempenha pode ser generalizado sobre tarefas e 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 muito difícil, ou mesmo impossível, para o projetista de uma aplicação sensível ao contexto enumerar o conjunto de estados contextuais que podem existir na aplicação, identificar que informação pode determinar com precisão um estado contextual desse conjunto, e definir que ações devem ser executadas em um estado particular (OZTURK P., 2001).

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 (ex. dia, hora, estação do ano), condições ambientais (ex. temperatura, qualidade do ar, luz, som), disponibilidade de recursos (ex. bateria, largura de banda, tamanho da tela), recursos próximos (ex. dispositivos acessíveis, impressoras, hosts), medidas fisiológicas (ex. pressão sanguínea, batimento cardíaco, atividade muscular, tom de voz), entre outros.

Contextos complexos podem ser, por exemplo, a situação do indivíduo (ex. se está falando, lendo, caminhando ou escrevendo), situações sociais (ex. com quem o usuário está, quem são as pessoas próximas), atividades sociais (ex. 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: período de tempo, episódios de uso anteriores conhecidos pela pessoa, estado das interações sociais, mudanças nos objetivos internos, e influências do local onde a pessoa se encontra (OZTURK P., 2001).

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 ciente de 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, 2002), a natureza da informação contextual deve ser levada em conta ao se construir sistemas ubíquos e de computação sensível ao contexto. Eis alguns pontos que cabe destacar:

Características temporais da informação contextual: pode-se caracterizar uma informação contextual como sendo estática ou dinâmica. Informações contextuais estáticas descrevem aspectos invariáveis dos sistemas, como a data de aniversário de uma pessoa. 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 CyberGuide (ABOWD G. D.; RODDEN, 2002) 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) 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 em (ABOWD G. D.; RODDEN, 2002) não sugerem completeza, mas sim, um conjunto básico de diretrizes a ser seguido no processo de construção de uma aplicação sensível a contexto. Nesse interim, (TRUONG K. N.; BROTHERTON, 2001) discutem uma dimensão semântica originada do domínio de aplicações de captura e acesso:

- *How* (como) - no contexto de aplicações de captura e acesso, esta dimensão fornece informações relativas a como 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 úteis. Contudo, os atuais modelos para desenvolvimento de software não oferecem suporte para o projeto e de tais aplicações e, sobretudo, tais técnicas de modelagem não provêem suporte para modelagem das informações contextuais (HENRICKSEN K; INDULSKA, 2002).

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

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

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

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

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

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

Modelos baseados em ontologias: uma ontologia é uma especificação de uma conceituação, isto é, uma descrição de conceitos e relações que existem entre eles, em um domínio de interesse. Nesse modelo o contexto é modelado em ontologias, construindo uma base de conhecimento do contexto.

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

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

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

3. **Riqueza e qualidade da informação (rqi):** a qualidade e a riqueza das informações podem variar de acordo com o tempo e com o tipo de sensor. Daí

Tabela 2.1: Avaliação das Abordagens para Modelagem de Contexto

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

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.

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.

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 checar 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.

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

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

O modelo ontológico CONON processa o contexto usando predicados de primeira ordem, e duas abordagens de raciocínio: baseada em lógica descritiva e baseada em lógica de primeira ordem (WANG X. H., 2004). A estrutura do predicado possui três campos:

Situation	Reasoning Rules
Sleeping	(?u locatedIn Bedroom) $\wedge$ (Bedroom lightLevel LOW) $\wedge$ (Bedroom drapeStatus CLOSED) $\Rightarrow$ (?u situation SLEEPING)
Shower-ing	(?u locatedIn Bathroom) $\wedge$ (WaterHeater locatedIn Bathroom) $\wedge$ (Bathroom doorStatus CLOSED) $\wedge$ (WaterHeater status ON) $\Rightarrow$ (?u situation SHOWERING)
Cooking	(?u locatedIn Kitchen) $\wedge$ (ElectricOven locatedIn Kitchen) $\wedge$ (ElectricOven status ON) $\Rightarrow$ (?u situation COOKING)
Watching-TV	(?u locatedIn LivingRoom) $\wedge$ (TVSet locatedIn LivingRoom) $\wedge$ (TVSet status ON) $\Rightarrow$ (?u situation WATCHINGTV)
Having-Dinner	(?u locatedIn DiningRoom) $\wedge$ (?v locatedIn DiningRoom) $\wedge$ (?u owl:differentFrom ?v) $\Rightarrow$ (?u situation HAVINGDINNER)

Tabela 2.2: Regras de Raciocínio de Contexto em Lógiva de Primeira Ordem (WANG X. H., 2004)

sujeito, objeto e verbo. Por exemplo, o contexto de localização física - Wang está localizado em seu quarto - é descrito como (Wang, locatedIn, Bedroom). A Tabela 2.2 apresenta algumas regras pré-definidas que inferem um contexto de alto nível (situação do usuário) a partir de informações de baixo nível, em uma casa inteligente. As regras inferem se o usuário está dormindo, tomando banho, cozinhando, vendo TV ou jantando.

A figura 2.1 apresenta a formação de um contexto de alta ordem Outdoors a partir de um conjunto de contextos atômicos, usando redes bayesianas (KORPIPAA P., 2003). Os retângulos brancos representam elementos atômicos de contexto, as caixas com cor mais clara (com rótulos como Dark e Normal) representam valores de contexto, e as caixas com cor mais escura (contendo números nos rótulos) contêm o grau de confiança para aquele contexto. Uma rede bayesiana classifica os valores de confiança em uma das classes de entrada *Indoors* e *Outdoors*, para inferir se o usuário está em um ambiente aberto ou em um ambiente fechado. Para isso, são analisados aspectos do ambiente que podem ser medidos por sensores, como a luminosidade, umidade e temperatura.

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

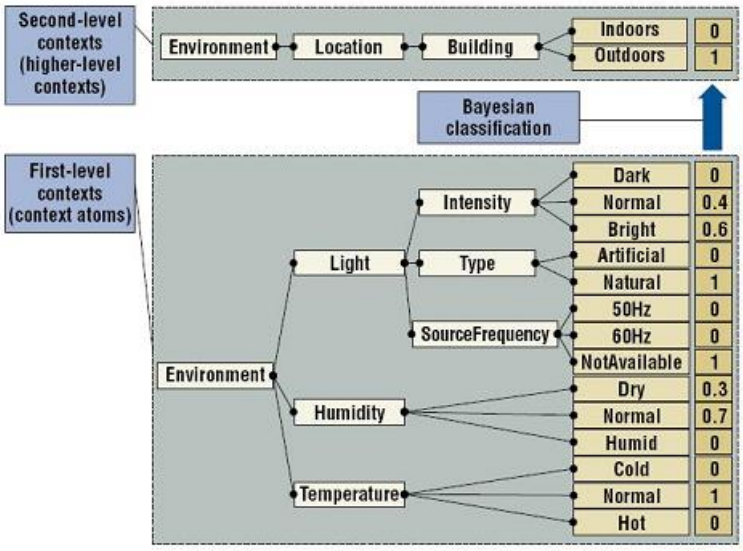

Figura 2.1: Exemplo de Geração de Contexto de Alta Ordem Usando Redes Bayesianas (KORPIPAA P., 2003)

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 MECANISMOS DE SENSIBILIDADE AO CONTEXTO

Este capítulo discute aspectos relacionados aos mecanismos para sensibilidade ao contexto. Particularmente os projetos que tiveram suas características sistematizadas tem como domínio de aplicação a computação ubíqua.

Ao todo, foram avaliadas arquiteturas de onze projetos, cujas características formam um conjunto representativo do que vêm sendo desenvolvido nos últimos anos na direção de infra-estruturas de suporte a sensibilidade ao contexto.

3.0.1 *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 cientes de 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.

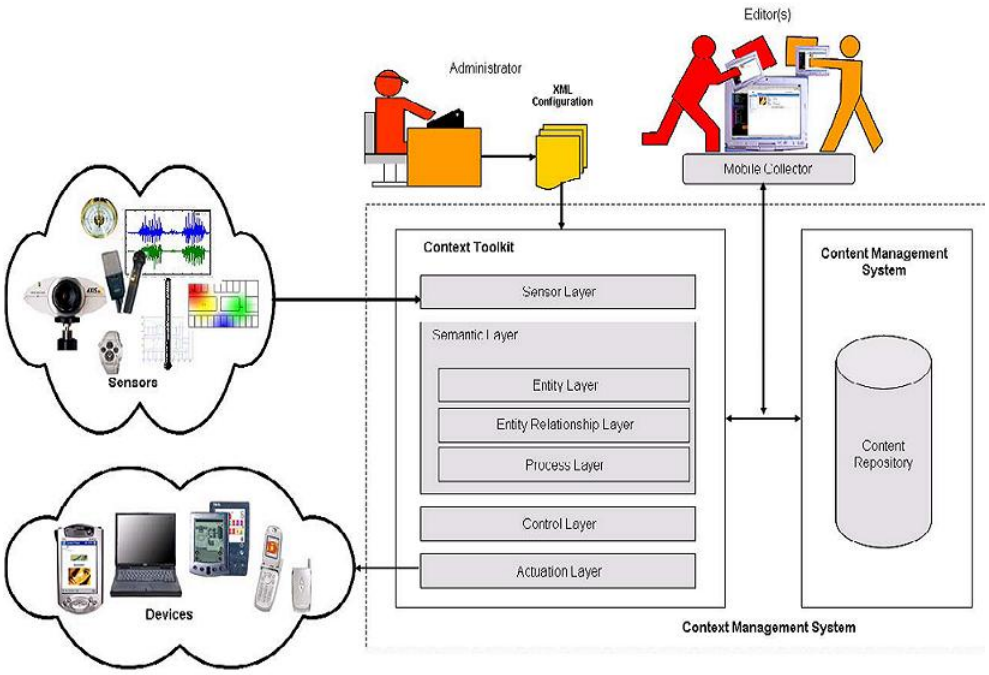

Figura 3.1: Arquitetura do CXMS (ZIMMERMANN A., 2005a)

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.

3.0.2 *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 cientes de contexto interativas, no domínio da computação ubíqua. O CTK compreende um *framework* para aplicações cientes de 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).

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

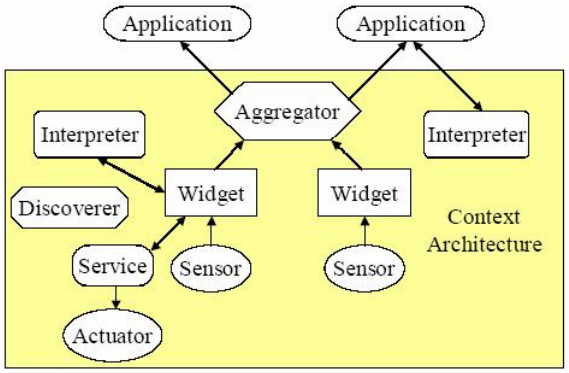

Figura 3.2: Hierarquia de Componentes e Arquitetura do CTK (DEY, 2000)

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 e de formas 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.

3.0.3 *Middleware* de Contexto do Gaia

Gaia é uma infra-estrutura para ambientes inteligentes de computação pervasiva 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.

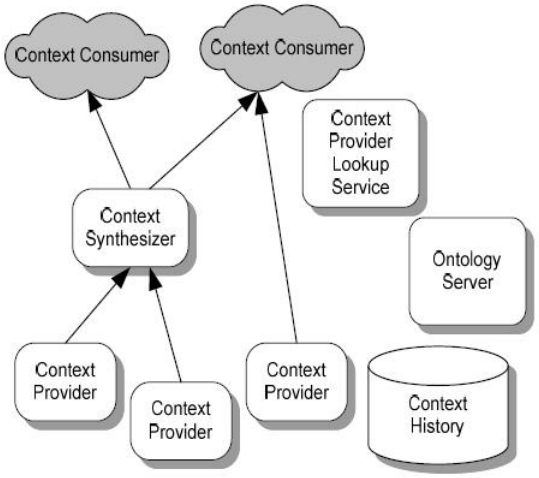

Figura 3.3: Visão Geral do *Middleware* de Contexto do GAIA (RANGANATHAN A., 2003)

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 (ex. localização e tempo); (ii) contextos ambientais (ex. clima, níveis de luz e som); (iii) contextos de informação (ex. notas sobre estoque e placar esportivo); (iv) contextos pessoais (ex. saúde, agenda e atividade); (v) contextos sociais (ex. atividade do grupo, relacionamentos sociais e quem está na sala com quem); (vi) contextos da aplicação (ex. email e páginas web visitadas); e (vii) contextos do sistema (ex. tráfego da rede e status da impressora).

As aplicações cientes de 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).

3.0.4 *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, 2002) 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.

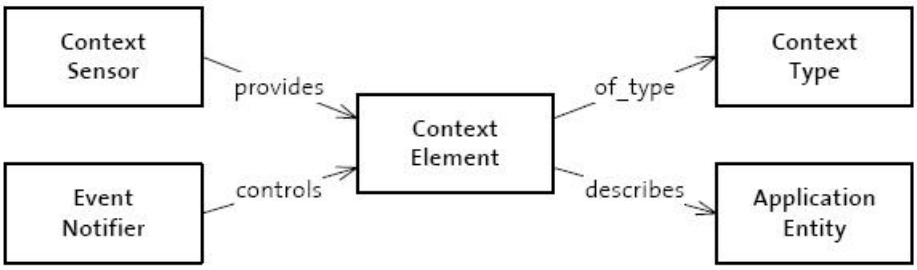

Figura 3.4: Modelo de Contexto (BELOTTI, 2004)

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 contextual 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).

A aquisição do contexto é baseada em sensores. Para instalar um novo sensor, o

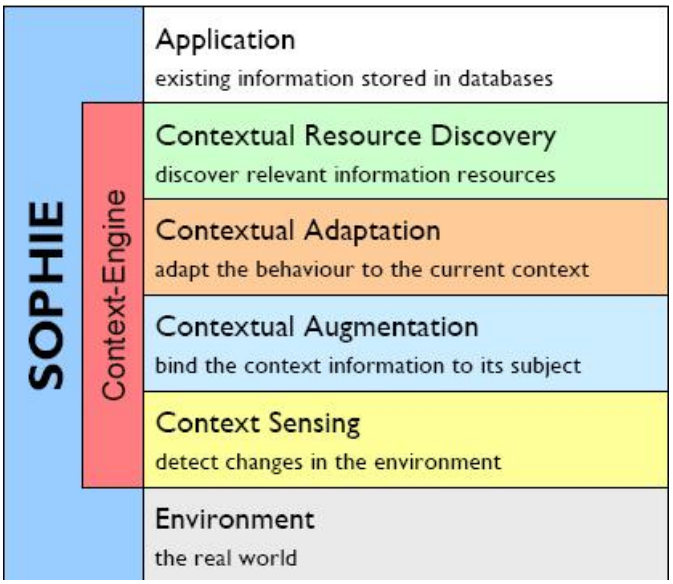

Figura 3.5: Arquitetura do SOPHIE (BELOTTI, 2004)

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).

3.0.5 *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.

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 pos-

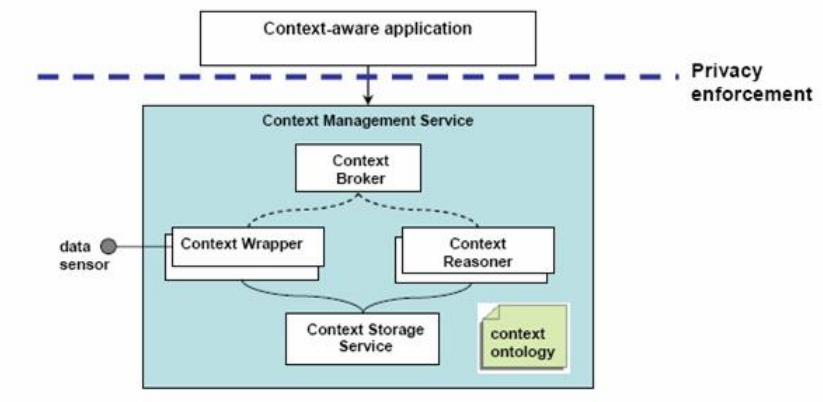

Figura 3.6: Gerenciamento de Dados no Awareness (WEGDAM, 2005)

sibilita 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.

3.0.6 *Service-Oriented Context-Aware Middleware*

SOCAM (*Service-Oriented Context-Aware Middleware*) é um *middleware* para a construção rápida de serviços cientes de 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).

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

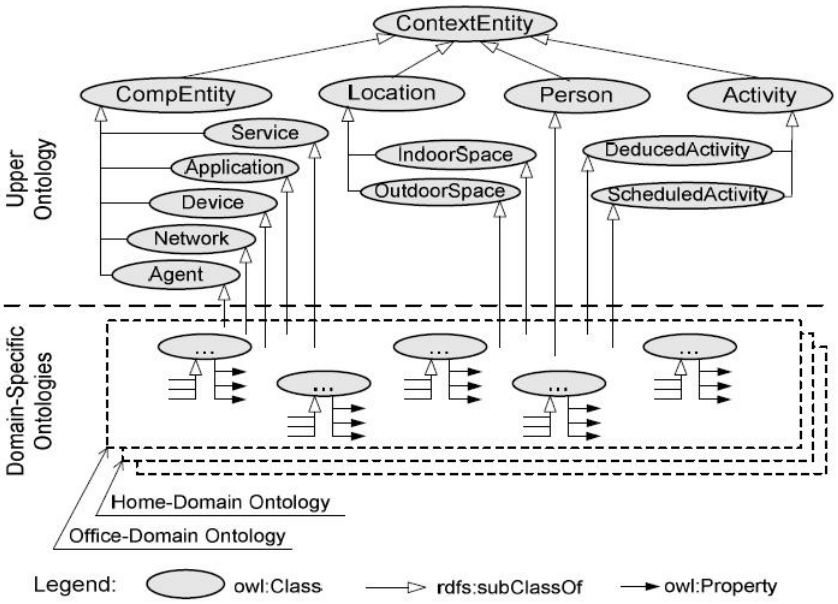

Figura 3.7: Visão Parcial da Ontologia de Alto Nível do CONON (WANG X. H., 2004)

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”.

3.0.7 *Context Broker Architecture*

CoBrA (*Context Broker Architecture*) é uma arquitetura baseada em agentes cujo objetivo é apoiar sistemas cientes de 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

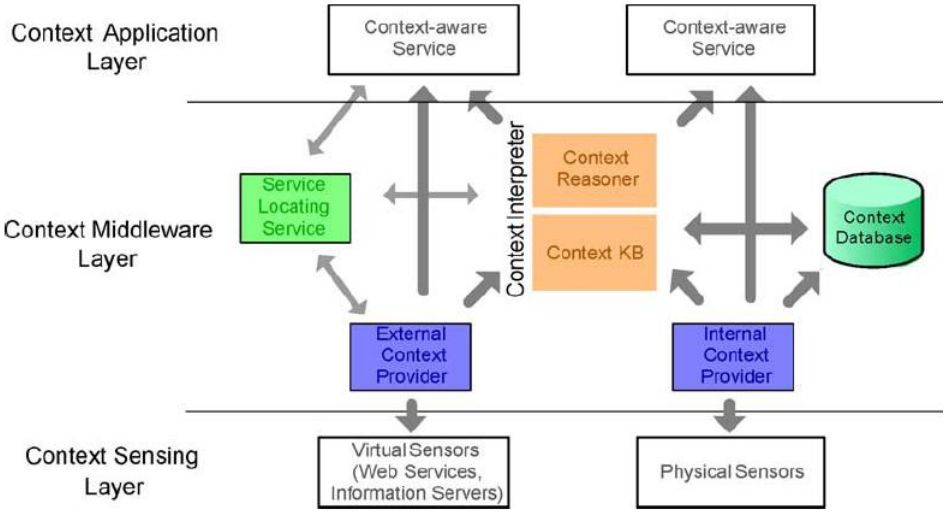

Figura 3.8: Visão Geral da Arquitetura do SOCAM (GU T., 2004)

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 figura 3.9 mostra uma representação gráfica dos conceitos ontológicos da CoBrA-Ont, a qual 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 3.10. 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., 2009), um raciocinador baseado em sentenças Prolog utilizado para apoiar as inferências lógicas para resolver conhecimento inconsistente (CHEN, 2004).

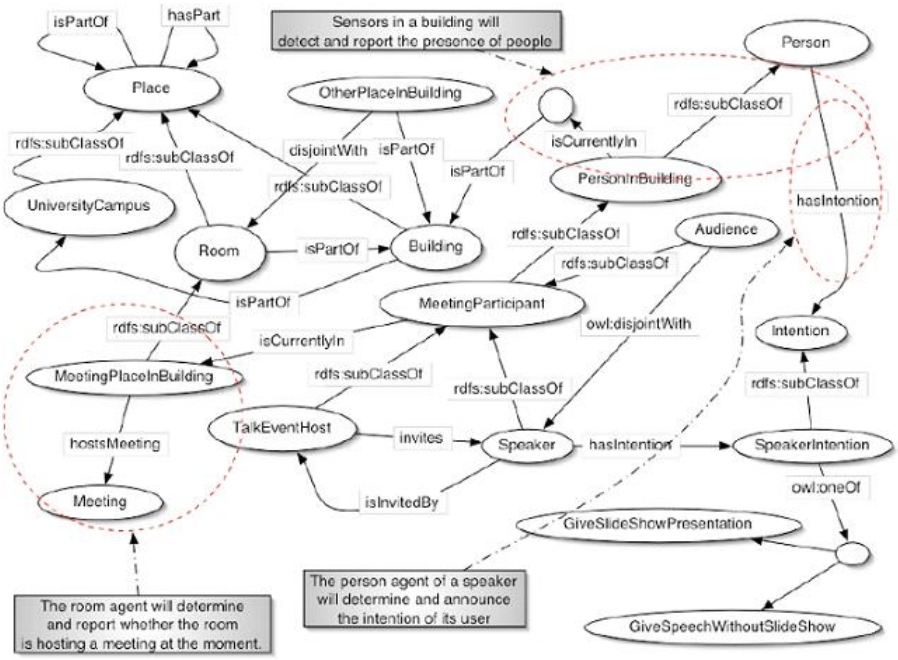

Figura 3.9: Representação Gráfica da Ontologia CoBrA-Ont (CHEN, 2004)

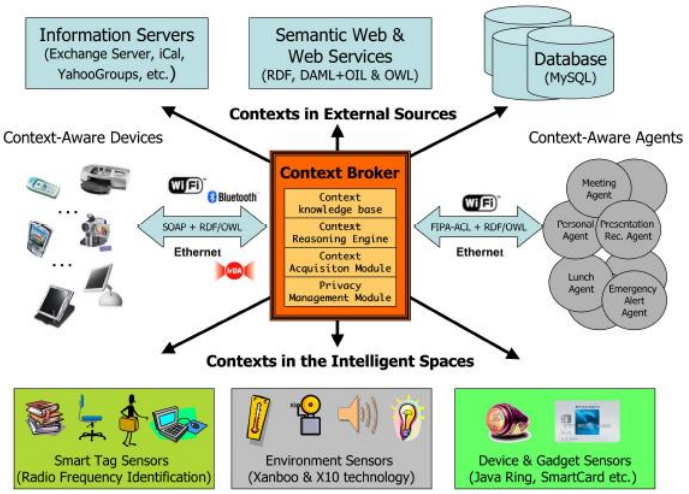

Figura 3.10: Visão Geral da Arquitetura do CoBrA (CHEN H., 2005)

3.0.8 *Mobile Collaboration Architecture*

A MoCA (*Mobile Collaboration Architecture*) (SACRAMENTO et al., 2004), é uma arquitetura de *middleware* para o desenvolvimento de aplicações colaborativas e cientes de 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 cientes de 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 3.11 é ilustrada a arquitetura MoCA e a interação entre seus diversos componentes durante o registro e execução de uma aplicação.

3.0.9 *Framework* de Contexto

Henricksen e Indulska (HENRICKSEN K., 2005a) propuseram um *framework* de contexto que visa facilitar a construção de aplicações cientes de 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 3.12) é organizado em uma hierarquia de seis camadas: (i) camada da aplicação ciente de 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

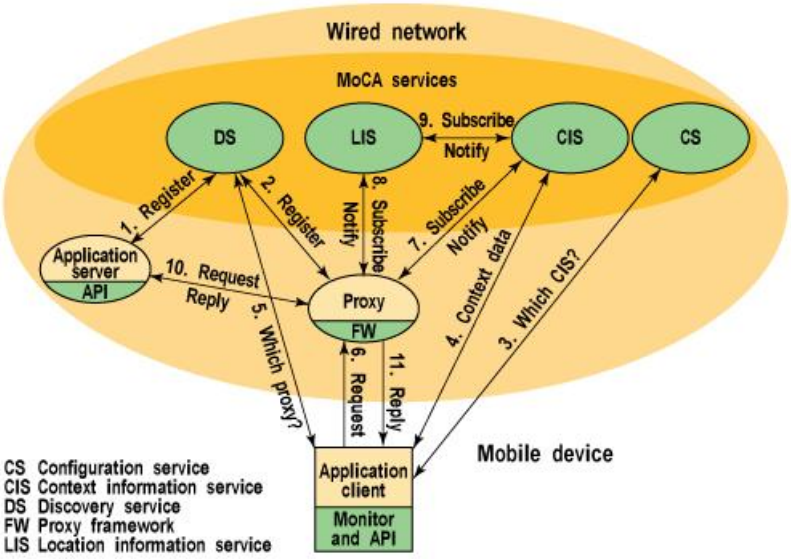

Figura 3.11: Serviços da Arquitetura Moca (SACRAMENTO et al., 2004)

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.

3.0.10 *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 cientes de 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 3.13.

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

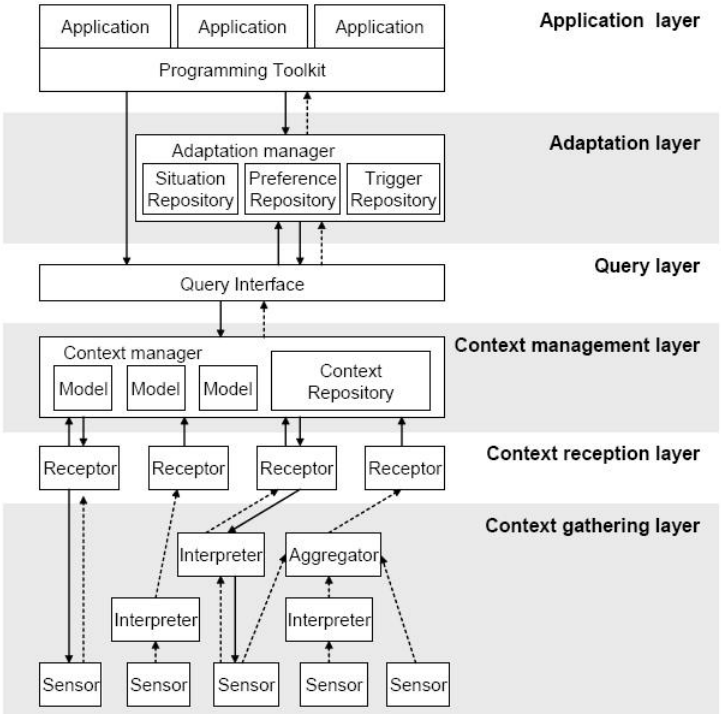

Figura 3.12: Arquitetura do *Framework* de Contexto (HENRICKSEN K., 2005a)

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.

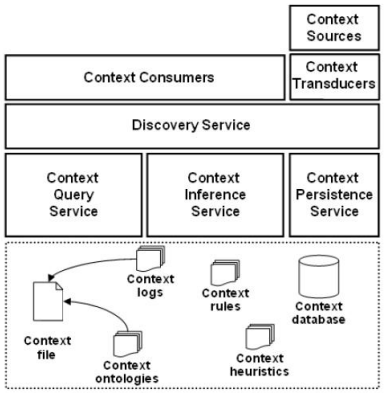

Figura 3.13: Visão Geral da Arquitetura do SCK (BULCAO NETO R. F., 2005)

A representação do contexto no SCK é feita usando uma ontologia (figura 3.14) 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.

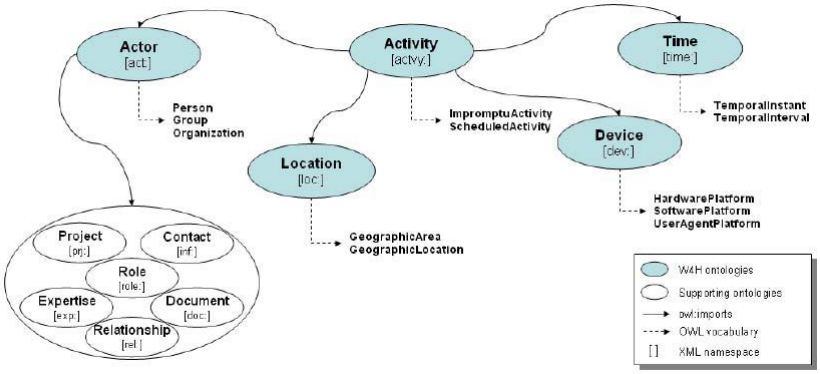

Figura 3.14: Visão Geral do Modelo de Contexto (BULCAO NETO R. F., 2005)

3.0.11 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, 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 3.15 ilustra a arquitetura geral da plataforma.

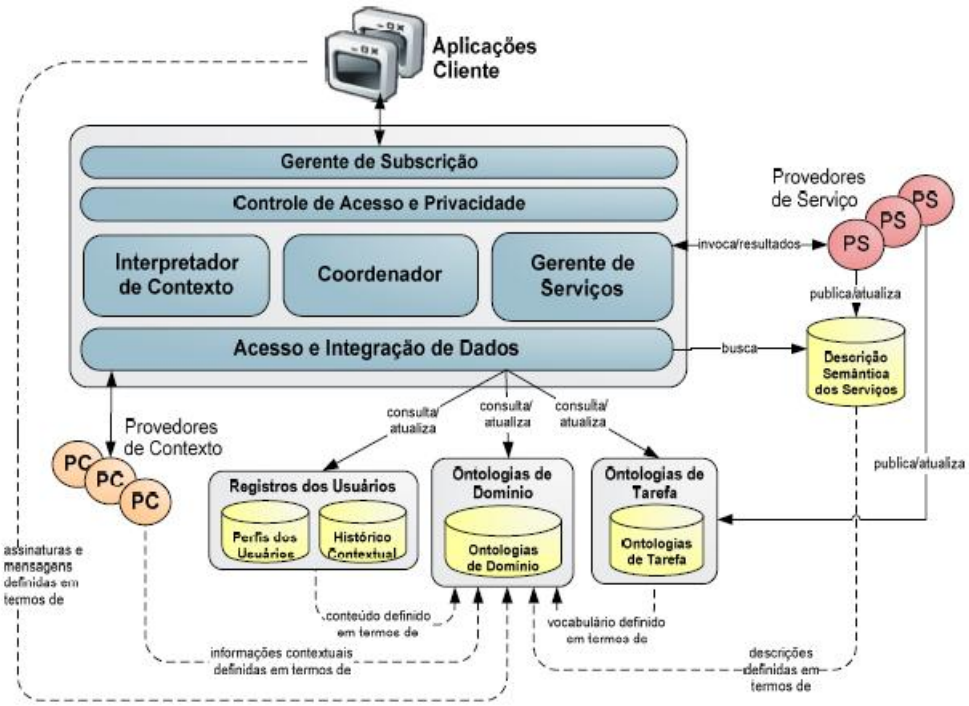

Figura 3.15: Plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006)

3.1 Considerações Sobre o Capítulo

Este capítulo apresentou diversas abordagens de mecanismos de sensibilidade ao contexto presentes na literatura. A Tabela 3.1 exibe um resumo das principais características desses mecanismos, o modelo de representação utilizado por cada um, as tarefas de gerenciamento que implementam, e os tipos de informações contextuais que consideram.

Os mecanismos para sensibilidade ao contexto 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. Nenhum dos sistemas avaliados apresentou soluções para lidar com questões relacionadas a otimização do tratamento do contexto.

O EXEHDA-SS propõe uma abordagem que possibilita aos dados contextuais capturados pelo servidor de contexto, possam ser qualificados pelo mecanismo de suporte semântico, que será detalhado no Capítulo 5. A seguir serão apresentadas os fundamentos para arquitetura do EXEHDA-SS.

Tabela 3.1: Resumo das Abordagens para Gerenciamento de Contextos

Mecanismo	Representação	Funcionalidades	Informações Contextuais
Framework CXMS	Par chave-valor	Aquisição, modelagem, armazenamento, definição do comportamento da aplicação, e apresentação da informação	*Identidade, localização, tempo e ambiente*
Context Toolkit	Par chave-valor	Aquisição, agregação, armazenamento, disseminação, adaptação, localização de recursos	*Identidade, localização, tempo e atividade*
Middleware do Gaia	Ontologia	Aquisição, compartilhamento, agregação, raciocínio, armazenamento, disseminação, adaptação, localização de recursos	*Aplicações, serviços, dispositivos, usuários, fontes de dados, localização, atividades e informações climáticas*
Motor do SOPHIE	Gráfico ORM	Aquisição, armazenamento, adaptação, disseminação e localização de recursos	*Estrutura de tipos genérica. Aplicação define seu contexto*
AWARENESS	Ontologia	Aquisição, armazenamento, interpretação, descoberta de recursos	*Aplicação, dispositivos, usuários*
Middleware SOCAM	Ontologia	Aquisição, compartilhamento, raciocínio, incerteza, qualidade, armazenamento, disseminação, localização de recursos	*Localização física, pessoa, atividade e entidade computacional*
Middleware CoBrA	Ontologia	Aquisição, compartilhamento, raciocínio, armazenamento, segurança e privacidade	*Localização física, agentes, atividade, papéis, intenções*
Moca	Ontologia	Aquisição, interpretação, armazenamento, localização de recursos	*Localização, disponibilidade contínua de componentes de captura*
Framework de Contexto	Gráfico CML	Aquisição, representação, agregação, armazenamento, disseminação e adaptação	*Aplicação define seu contexto*
Infraware	Ontologia	Aquisição, interpretação, acesso e integração de dados, controle de acesso e privacidade	*Aplicação, dispositivos, usuários, localização*

4 FUNDAMENTOS DO EXEHDA-SS

Neste capítulo, são discutidos os principais conceitos relacionados ao EXEHDA-SS. São considerados aspectos das tecnologias de web semântica, do conceito de ontologia e suas principais linguagens para o emprego de suporte semântico, sendo abordado também, a API Jena, seus mecanismos de inferência e linguagens de consulta RDF. 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 Tecnologias Web Semântica

A Web Semântica é definida pelo W3C (BERNERS, 2001) como uma extensão da Web atual em que a informação tem um significado bem definido, permitindo que pessoas e computadores trabalhem melhor em cooperação. A Web Semântica objetiva ter dados na Web bem definidos e ligados de tal modo que possa ser usado mais efetivamente para descoberta, automação, integração e reuso através das várias aplicações. Além disso, dados podem ser compartilhados e processados por ferramentas automatizadas bem como por pessoas.

Para alcançar este objetivo há necessidade de estabelecer padrões de troca de informações que sejam interpretáveis por máquinas. Estes padrões não somente definiriam a sintaxe da informação, mas também o seu significado. A Web Semântica, em 2001, tornou-se uma abordagem mais realista com a proposta do W3C. A proposta definiu várias novas camadas para a Web, sugerindo linguagens e padrões para as camadas, como a figura 4.1 mostra. A seguir serão descritas as camadas propostas pelo W3C:

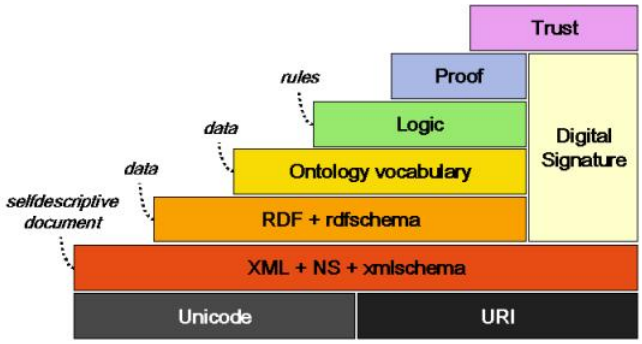

Figura 4.1: As camadas da Web Semântica (BERNERS, 2001)

A primeira camada que serve de base para a Web Semântica repousa sobre o conjunto de caracteres Unicode e a URI (*Uniform Resources Identifier*) que é utilizada para referenciar os recursos na rede. Na segunda camada, a linguagem de marcação XML(*eXtensible Markup Language*) associada aos recursos de NameSpaces e XML Schema servem para descrever documentos que podem ser compreendidos por pessoas e software.

A partir deste ponto, todas as camadas seguintes estão estruturadas sobre a camada XML e tiram proveito dela. Assim, a estrutura seguinte (RDF + RDF Schema - *Resource Description Framework*) é uma camada de metadados que é construída sobre a de XML e que visa a interoperabilidade entre as aplicações, representando os dados sob a forma de triplas através de descrições de recursos, propriedades e valor.

A próxima estrutura corresponde à camada de ontologias, descrita detalhadamente na seção 4.2. Ela define um vocabulário comum para que o conteúdo semântico possa ser compartilhado entre as aplicações. Neste nível, a linguagem OWL (*Ontology Web Language*) usada para representar as ontologias possibilita agregar maior conteúdo semântico aos documentos utilizando recursos do RDF e do XML das camada inferiores.

As ultimas camadas, alicerçadas sobre a estrutura já existente na parte inferior da arquitetura, implementam a definição de regras, e possibilitam a prova e validação de inferências realizadas por agentes de software que fazem uso dos recursos de toda a estrutura existente para a representação do conhecimento.

4.2 Ontologias

Ontologias têm sido largamente utilizadas em áreas como gerenciamento de conteúdo e conhecimento, comércio eletrônico e Web semântica. 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 exibidos os principais conceitos relacionados a este assunto, partindo do conceito de ontologia, passando pelos principais tipos de ontologias, benefícios advindos do uso de ontologias e finalmente descrevendo as principais linguagens para ontologias.

4.2.1 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, 2001). 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.

4.2.2 Linguagens para Ontologias

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.

4.2.2.1 RDF

RDF (*Resource Description Framework*) 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 na Web, como páginas, serviços, documentos, etc. Além dos identificadores Web (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 da Web, 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, 2002), 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.

4.2.2.2 RDF Shema

RDF *Schema* (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”autosuficiente. 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).

4.2.2.3 OWL

A OWL (*Web Ontology Language*) é uma linguagem para definir e instanciar ontologias na Web. Ela foi projetada para disponibilizar uma forma comum para o processamento de conteúdo semântico da informação na Web. 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 da Web 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 na Web.

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 seus parentes mais expressivos, 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.

4.3 API Jena

A API Jena é um *framework* desenvolvido na linguagem de programação Java pela empresa HP (*Hewlett-Packard*) para construção de aplicações Web Semânticas. 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, 2003):

- 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 do sistema 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 virtualmente, como resultado de processos de inferência aplicados a outras fontes de triplas (WILKINSON KEVIN; SAYERS, 2003). 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.

4.3.1 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: um conjunto de raciocinadores úteis para checagem de consistência, porém incompletos da linguagem OWL Lite;

- Raciocinador DAML micro: usado internamente no Jena para fornecer um mínimo de inferência para ontologias descritas em DAML (legado);

- Raciocinador de Regra Genérico: raciocinador baseado em regra que é usado para implementar ambos raciocinadores, RDFS e OWL, mas também está disponível para uso em geral. Suporta inferência baseada em regras sobre grafos RDF que podem ser definidas externamente pelo usuário.

Tabela 4.1: Terminologia DL (HORROCKS, 2003)

Tipo	Significado
ABox - Assertional Box	*Contém um exemplo concreto do domínio de conhecimento e axiomas declarados sobre indivíduos, ou seja, um indivíduo é uma instância de um conceito; ou um indivíduo está relacionado a outro por uma função.*
TBox - Terminological Box	*Define a estrutura do domínio de conhecimento, consistindo de um conjunto de axiomas declarados, ou seja, a definição de um novo conceito em termos de outros conceitos definidos previamente.*
KB - Knowledge Box	*Uma base de conhecimento DL, ou seja, combinação de ABox e TBox*

Além dos raciocinadores já incluídos, 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). A interface DIG do Jena facilita a conexão de qualquer raciocinador que suporte o padrão DIG (*DL Implementors Group*) (BECHHOFER, 2006).

A verificação de consistência na terminologia DL, mostrada na Tabela 4.1, consiste na operação de verificar a consistência de um ABox com respeito a um TBox.

4.3.2 Linguagem de Consulta RDF

Para que as consultas possam ser realizadas e as informações possam ser extraídas, a linguagem SPARQL disponibiliza uma sintaxe que, ainda que haja algumas particularidades, funciona de maneira similar à linguagem SQL (*Structured Query Language*). A linguagem SPARQL possibilita ordenação de seqüências baseado em condições, limitação de seqüências, definir 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.

4.4 *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, 2007), sendo direcionado às aplicações distribuídas, móveis e sensíveis ao contexto da Computação Ubíqua. Abaixo são apresentadas premissas perseguidas na concepção do EXEHDA (YAMIN, 2004).

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.

Não comprometimento com uma classe específica de aplicações adaptativas

Mobilidade implica adaptabilidade, o que significa que sistemas devem ter consciência da localização e do contexto, e devem tirar vantagem desta informação, estruturando-se de modo distribuído e reconfigurando-se dinamicamente.

Controle da adaptação

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

aplicação nos procedimentos de adaptação. Deste modo, considerando a natureza da aplicação, o programador poderá definir a distribuição de responsabilidades entre o *middleware* e a aplicação no processo de adaptação.

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.

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.

4.4.1 Organização do EXEHDA

O *middleware* EXEHDA tem por finalidade definir a arquitetura para um ambiente de execução destinado às aplicações da Computação Ubíqua, no qual as condições de contexto são pró-ativamente monitoradas, e o suporte à execução 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, 2007).

4.4.1.1 Arquitetura de Software

A figura 4.2 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.

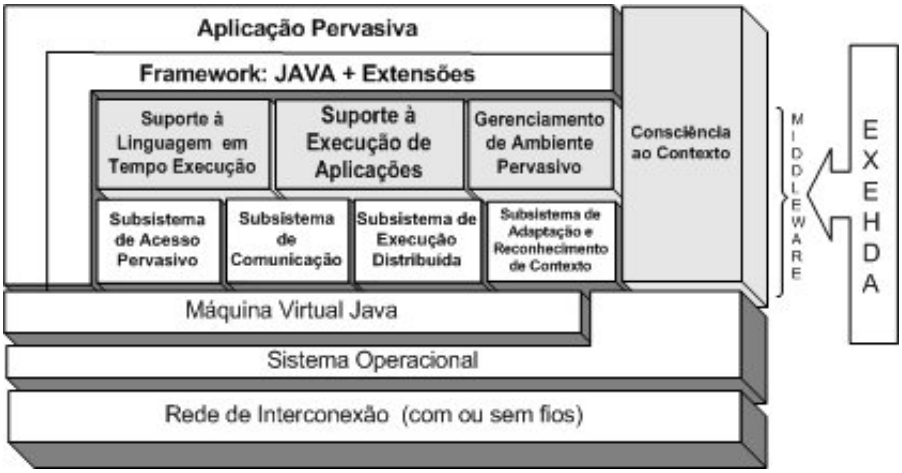

Figura 4.2: Arquitetura de Software *Middleware* EXEHDA (YAMIN, 2004)

4.4.1.2 Ambiente Ubíquo Disponibilizado

O ISAMpe (ISAM pervasive environment) 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 4.3 (ISAM, 2007).

O meio físico sobre o qual o ISAMpe 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 composiçãao do ISAMpe (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 ISAMpe 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 ISAMpe, 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 ISAMpe 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 ISAMpe 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 ISAMpe, tornando-os, desta forma, passíveis de serem localizados dinamicamente e serem utilizados pelas aplicações ubíquas.

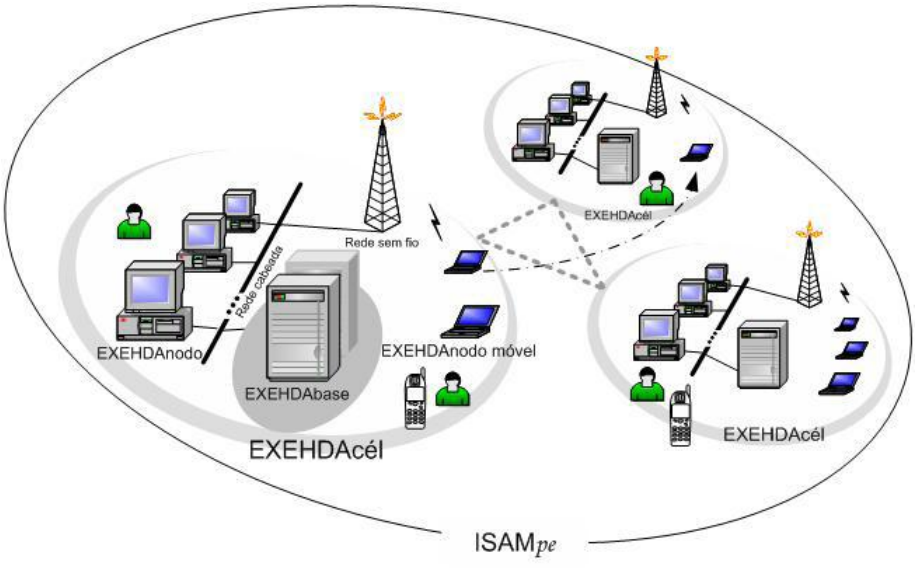

Figura 4.3: ISAM *Pervasive Environment* (YAMIN, 2004)

4.4.1.3 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 4.4;

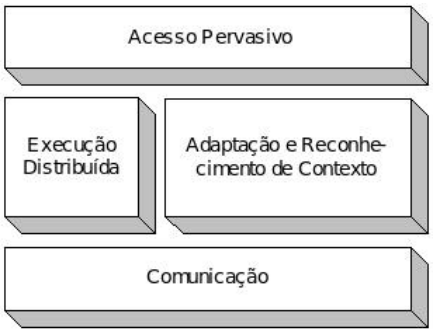

Figura 4.4: Organização dos Subsistemas do EXEHDA (YAMIN, 2004)

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.

4.4.1.4 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 ISAMpe. Este núcleo é formado por dois componentes, conforme figura 4.5:

- 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.

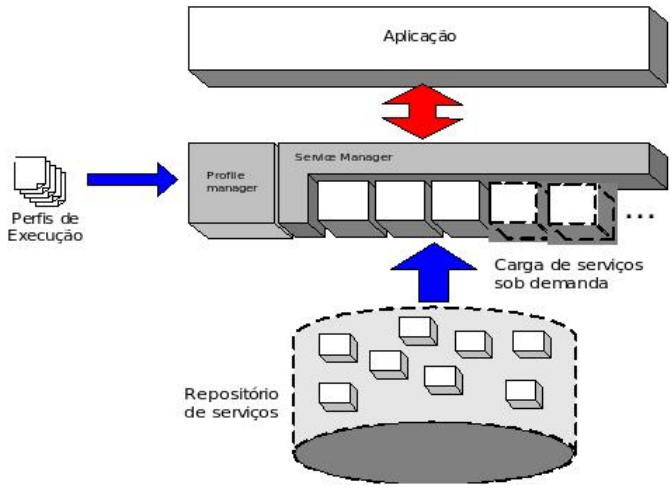

Figura 4.5: Organização do Núcleo do EXEHDA (YAMIN, 2004)

4.4.2 Subsistemas do EXEHDA

4.4.2.1 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 seguintes componentes:

Executor
Serviço que acumula as funções de disparo de aplicações, e de criação e migração de seus objetos. Na implementação destas funções é empregada a instalação de código sob demanda. Para tal, interage com os serviços CIB e BDA, que podem ser vistos no itens seguintes.

Cell Information Base - CIB
Serviço que implementa a base de informações da célula. Sua principal funcionalidade está relacionada à manutenção da infra-estrutura distribuída que forma o ISAMpe.

OXManager
A abstração OX (Objeto eXehda (EXEHDA)), provida pelo *middleware* às aplicações, consiste em uma instância de objeto, criada por intermédio do serviço *Executor*, à qual pode ser associada meta-informação em tempo de execução. No caso da abstração OX, esta meta-informação ocorre na forma de atributos, i.e., pares, nome, valor, sendo que 'nome' é uma cadeia de caracteres ASCII que descreve o atributo, podendo 'valor' ser um conteúdo genérico, inclusive de natureza binária. A gerência e manutenção da

meta-informação associada a um OX é atribuição do serviço *OXManager*. Este serviço confere às operações de consulta e atualização dos atributos do OX o necessário caráter ubíquo, permitindo que estes sejam acessados a partir de qualquer nodo do ISAMpe.

Discoverer
O serviço de descoberta de recursos responsável pela localização de recursos especializados no ISAMpe a partir de especificações abstratas dos mesmos. As especificações caracterizam o recurso a ser descoberto por meio de atributos e seus respectivos valores. Adicionalmente, a requisição de descoberta de recurso incorpora um parâmetro que define a amplitude da pesquisa. Nesse sentido, os seguintes valores estão definidos:

- 0: próprio nodo;
- 1: segmento local;
- 2: célula local;
- 3: vizinhança estática da célula local;
- 4: vizinhança completa da célula local, nível 1;
- 5: vizinhança completa da célula local, nível 2;

ResourceBroker
O controle da alocação de recursos às aplicações no EXEHDA é desempenhado pelo serviço *ResourceBroker*, o qual atende tanto requisições originárias da própria EXEHDAcel quanto oriundas de outras células do ISAMpe.

Gateway
Serviço que faz a intermediação das comunicações entre os nodos externos à célula e os recursos internos a ela. Da sua ação integrada com o *ResourceBroker* decorre o controle de acesso aos recursos de uma EXEHDAcel. Pode-se sintetizar que: *controle de acesso entre células = Gateway + ResourceBroker*.

StdStreams
Serviço que provê o suporte ao redirecionamento dos *streams* padrões de entrada, saída e erro. Sua funcionalidade se dá numa perspectiva por aplicação, sem a necessidade de modificação no código da mesma. Para isto, define que cada aplicação em execução em um determinado EXEHDAnodo possui um componente Console individualizado, o qual agrupa os três *streams* padrões.

Logger
A funcionalidade de registro de rastro de execução (*logging*) é amplamente empregada em sistemas computacionais. Na fase de desenvolvimento, pode ser empregada para depuração de um programa auxiliando a identificar comportamentos errôneos ou imprevistos (gargalos de execução - *bottlenecks*). Por sua vez, considerando a segurança dos sistemas computacionais, esta funcionalidade é frequentemente empregada para registro de operações importantes e/ou críticas realizadas, facilitando a identificação de

situações de intrusão, ou de uso indevido do sistema. Para atendimento destes aspectos, é disponibilizado o serviço *Logger*.

Dynamic Configurator - DC
Serviço que tem como objetivo realizar a configuração do perfil de execução do *middleware* em um determinado EXEHDAnodo de forma automatizada.

4.4.2.2 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.

Dispatcher
Serviço que disponibiliza o modelo de comunicação mais elementar do EXEHDA: troca de mensagens ponto-a-ponto com garantia de entrega e ordenamento das mensagens, o qual é especializado para operação no ambiente ubíquo. As mensagens trafegam entre instâncias do *Dispatcher* localizadas em nodos diferentes através de estruturas denominadas canais. Nesse sentido, quando de sua inicialização, o *Dispatcher* atualiza a informação do EXEHDAnodo na CIB (Cell Information Base), em específico o atributo *contactAddress*, provendo uma lista de protocolos e endereços que podem ser utilizados para alcançar aquele EXEHDAnodo.

WORB
Serviço que tem o objetivo de simplificar a construção de serviços distribuídos, permitindo que os programadores focalizem esforços no refinamento da semântica distribuída associada ao serviço em desenvolvimento, abstraindo aspectos de baixo nível relativos ao tratamento das comunicações em rede. Para tanto, oferece um modelo de comunicação baseado em invocações remotas de método, similar ao RMI, porém sem exigir a manutenção da conexão durante toda a execução da chamada remota.

CCManager
Considerando a premissa da mobilidade lógica dos componentes que constituem as aplicações ubíquas, o suporte a um mecanismo de comunicação com característica de desacoplamento temporal e espacial é particularmente oportuno, à medida que este simplifica a construção de tais aplicações. Este serviço vem atender a esta demanda, disponibilizando um mecanismo baseado na abstração espaço de tuplas, o qual prescinde da coexistência temporal de emissor e receptor. Outro aspecto oportuno desta abstração é a facilidade com que podem ser implementados outros padrões de comunicação, além do ponto-a-ponto.

4.4.2.3 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:

BDA-Base de Dados ubíqua das Aplicações
O ambiente de Computação Ubíqua tem por premissas (i) a possibilidade de o usuário disparar aplicações a partir de qualquer nodo integrante do sistema e, após o disparo, (ii) a mobilidade parcial ou integral de tais aplicações em resposta a modificações em seu contexto de execução, como, por exemplo, a movimentação do usuário ou alteração na condição de carga dos dispositivos atualmente em uso pela aplicação. Para tanto, a capacidade de instalação de código sob demanda é uma necessidade inerente à execução de aplicações na Computação Ubíqua. Seria impraticável manter todo o universo de software disponível instalado e atualizado (com coerência de versões) em todos os dispositivos do sistema. Por sua vez, a implementação do mecanismo de instalação sob demanda requer a existência de um repositório de código que forneça a mesma visão do software disponibilizado a partir de qualquer dispositivo do ISAMpe, mesmo após migrações. Outras funcionalidades para este repositório ubíquo também são desejáveis. Considerando a perspectiva de que as diversas aplicações disponibilizadas sigam linhas de evolução independentes, o suporte a controle de versões é oportuno na direção da manutenção da operacionalidade das diferentes aplicações.

AVU - Ambiente Virtual do Usuário
A premissa siga-me definida no ISAM reflete-se não só nas aplicações que um usuário atualmente executa, mas no seu ambiente computacional como um todo. Este engloba, além das aplicações em execução, as informações de personalização das aplicações definidas pelo usuário, o conjunto de aplicações instaladas (i.e. passíveis de serem disparadas por aquele usuário), como também seus arquivos privados. É atribuição do serviço AVU (Ambiente Virtual do Usuário) do EXEHDA a manutenção do acesso ubíquo a este ambiente virtual, da forma mais eficiente possível.

SessionManager
Serviço responsável pela gerência da sessão de trabalho do usuário, sendo definida pelo conjunto de aplicações correntemente em execução para aquele usuário. A informação que descreve o estado da sessão de trabalho é armazenada no AVU, estando portando disponível de forma ubíqua.

Gatekeeper
Serviço responsável por intermediar acessos entre as entidades externas à plataforma ISAM e os serviços do *middleware* de execução, conduzindo os procedimentos de autenticação necessários.

4.4.2.4 Subsistema de Reconhecimento de Contexto e Adaptação

Integram este subsistema os serviços (i) *Collector*, (ii) *Deflector*, (iii) *Context-Manager*, (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, os quais serão detalhados a seguir (YAMIN, 2004):

Collector
Responsável pela extração da informação bruta (diretamente dos recursos envolvidos) que, posteriormente refinada, dará origem aos elementos de contexto. Para isto, o serviço *Collector* aglutina a informação oriunda de vários componentes monitores Monitor e as repassa aos consumidores registrados *MonitoringConsumer*. Um componente *Monitor* gerencia um conjunto de sensores parametrizáveis. Por outro lado, entre os consumidores da informação extraída pela monitoração estão o serviços *ContextManager* e *Scheduler*.

Vale salientar que em decorrência da organização distribuída do EXEHDA o coletor de um EXEHDAnodo pode também, se necessário, registrar-se como consumidor perante o coletor de outro EXEHDAnodo.

Na arquitetura de monitoração, cada sensor contribui com a extração de um valor que descreve um aspecto específico, estático ou dinâmico, do recurso sendo monitorado. O conjunto dos sensores existentes em um dado EXEHDAnodo, assim como os parâmetros suportados por cada um destes sensores, integra a informação de descrição daquele nodo, disponibilizada no serviço CIB.

O serviço *Collector*, é responsável pela modificação dos parâmetros de operação dos sensores. A atuação do *Collector* ocorre por intermédio do Monitor associado ao sensor sendo re-parametrizado. É responsabilidade do *Collector* estabelecer parâmetros de operação que satisfaçam o caso mais exigente, considerando as necessidades dos consumidores registrados daquele sensor.

O controle da condição de publicação entre o *Collector* e os consumidores registrados é feito por meio do parâmetro *threashold*, que configura o limiar de variação do dado, de forma individualizada para cada sensor, acima do qual uma publicação do novo valor registrado pelo sensor é realizada. A extração da informação junto aos monitores não ocorre em momentos arbitrários, mas apenas em instantes discretos, múltiplos de um determinado quantum de tempo. O quantum é um dos parâmetros de configuração do serviço *Collector* em cada EXEHDAnodo, sendo utilizado para controlar o grau de intrusão do mecanismo de monitoração.

Transcorrido um quantum de tempo, o *Collector* notifica os monitores através do método *quantumExpired*, os quais então, executam uma operação de *polling* em cada um dos sensores ativos naquele momento no seu nodo. O critério de publicação (*treashold*) especificado para o sensor é aplicado, determinando a geração, ou não, de um evento para aquele sensor. Dessa forma, todos os eventos gerados dentro de um quantum são agrupados em uma única mensagem, reduzindo o tráfego de dados até os consumidores registrados.

Deflector
Disponibiliza abstração de canais de multicast para uso na disseminação das informações monitoradas. A presença do serviço Deflector é decorrência da busca de escalabilidade para a arquitetura de monitoração do EXEHDA.

ContextManager
Responsável pelo refinamento (tratamento) da informação bruta produzida pela

monitoração para produção de informações abstratas referentes aos elementos de contexto (valores contextualizados). É recebida como parâmetro uma descrição (XML) de como o dado referente àquele elemento de contexto deve ser produzido a partir da informação proveniente da monitoração.

AdaptEngine
Responsável pelo controle das adaptações de cunho funcional, este serviço provê facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações. Deste modo, libera o programador de gerenciar os aspectos de mais baixo nível envolvidos na definição e liberação dos elementos de contexto junto ao *ContextManager*.

Outra função do serviço *AdaptEngine* é prover um mecanismo de carga de código contextualizado. Desta forma, com base na informação do estado do elemento de contexto fornecido pelo serviço *ContextManager*, seleciona e carrega o código correspondente dentre alternativas de código definidas na política de adaptação funcional vigente. Este serviço disponibiliza ainda métodos para a ativação de ações quando determinado estado de um elemento de contexto tornar-se ativo.

Scheduler
Serviço central na gerência das adaptações de cunho não-funcional no EXEHDA, isto é, que não implicam alteração de código. Nesse sentido, o *Scheduler* emprega a informação de monitoração, obtida junto ao serviço *Collector*, para orientar operações de mapeamento. Essas operações decorrem de instanciações remotas ou migrações realizadas pelo serviço *Executor*, ou quando de chamadas de re-escalonamento, originadas do estado atual de um recurso não satisfazer mais as necessidades de um objeto anteriormente a ele alocado.

4.5 Considerações Sobre o Capítulo

Neste capítulo foram apresentados as tecnologias de web semântica. Também foram discutidas as principais motivações para o uso de ontologias e suas linguagens. Foi detalhadado a API Jena, com seus mecanismos de inferência e linguagem de consulta RDF.

Ainda neste capítulo foram 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 CONCEPÇÃO E MODELAGEM DO EXEHDA-SS

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 tarefas 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 de contexto de interesse das aplicações são enviadas ou notificadas 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, raciocínio 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, a modelagem da arquitetura de software, modelo de representação de contexto e a concepção do Motor de Inferência do EXEHDA-SS.

5.1 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 4.3. A figura 5.1 mostra a visão da proposição da integração do EXEHDA-SS ao Subsistema de Reconhecimento de Contexto e Adaptação do EXEHDA.

Esta sendo previso que a arquitetura de software do EXEHDA-SS seja alimentada com as políticas de adaptação da aplicação para que a mesma funcione colaborativamente com o EXEHDA-DA. Essas políticas serão responsáveis por caracterizar os aspec-

tos que devem ser considerados nos procedimentos de monitoração do ambiente ubíquo, de interpretação destes dados capturados e das respectivas notificações.

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 no na realização de tarefas de manipulação e raciocínio sobre as informações contextuais obtidas e a notificação das mesmas ao EXEHDA-DA.

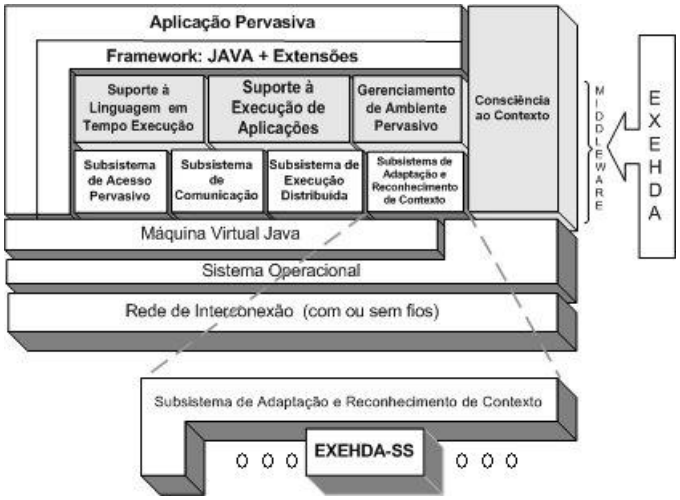

Figura 5.1: Integração do EXEHDA-SS ao Subsistema de Adaptação e Reconhecimento de Contexto do *Middleware* EXEHDA [adaptado de (YAMIN, 2004)]

Uma visão da arquitetura proposta para o EXEHDA-SS é ilustrada na figura 5.2. O servidor de contexto é composto por três gerentes: 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 o servidor de adaptação é notificado pelo EXEHDA-SS.

No servidor de contexto proposto são utilizados gerentes autônomos e cooperantes para a realização de tarefas de manipulação e raciocínio sobre o contexto. A tarefa de raciocínio 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 estão sendo previstas para este gerente: (i) manter o repositório de informações contextuais, que armazena os contextos capturados pelo gerente de aquisição; (ii) manter o repositório de contexto notificado, que armazena os estados dos contextos empregados pelo Gerente de Notificação; (iii) utilizar um Motor de Inferência para processar e raciocinar sobre as informações de contexto mantidas nos repositórios de conhecimento e nas política de adaptação da aplicação.

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

5.1.1 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.

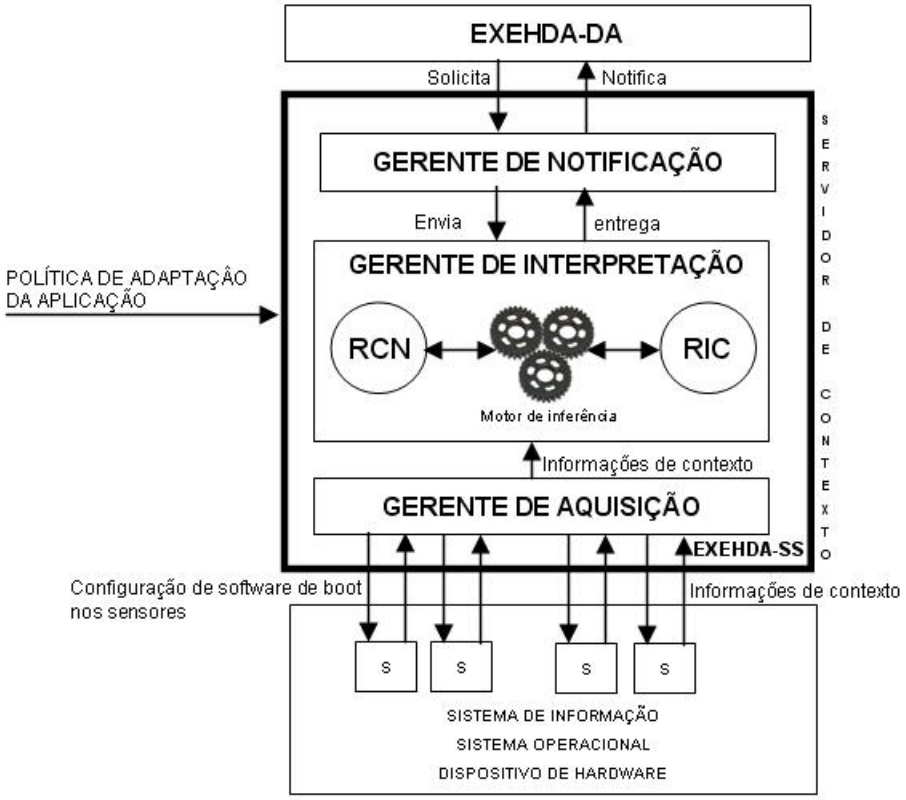

Figura 5.2: Arquitetura do EXEHDA-SS

O Gerente de Aquisição envolve funções que foram caracterizadas na seção 2.2.4, a proposição deste gerente a ser concebido pelo EXEHDA-SS é descrita abaixo:

- processar as políticas de adaptação 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 (ex. o 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 (ex. 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.

5.1.2 Gerente de Interpretação de Contexto

O Gerente de Interpretação de contexto tem como principal função realizar tarefas de manipulação e raciocínio das informações contextuais, utilizando para isto informações especificadas nas políticas de adaptação da aplicação.

As informações contextuais manipuladas pelo Gerente de Interpretação seguem o vocabulário especificado na OntUbi, ontologia do ambiente ubíquo, detalhada na Seção 5.2.2. O Gerente de Interpretação de contexto mantem dois repositórios de conhecimento baseados nesta ontologia. Esses repositórios mantém um histórico dos contextos para que o Motor de Inferência do Gerente de Interpretação possa inferir nos contextos armazenados e disponibiliza-los ao Gerente de Notificação.

A proposição do Motor de Inferência a ser concebido pelo EXEHDA-SS é detalhado na seção 5.3. Os repositórios do Gerente de Interpretação são descritos a seguir:

Repositório de Informações Contextuais (RIC): neste repositório são armazenados as informações contextuais obtidas pelo Gerente de Aquisição de contexto, mantendo as regras e fatos relativos as informações contextuais a serem manipuladas pelo Motor de Inferência do Gerente de Interpretação de contexto.

Repositório de Contexto Notificado (RCN): mantém o histórico dos contextos notificados ao Gerente de Notificação do EXEHDA-SS. Essas informações armazenadas neste repositório servirá ao Motor de Inferência em interações que possam ser solicitadas e/ou disponibilizadas ao Gerente de Notificação.

Os objetivos do Gerente de Interpretação do contexto incluem:

- manter consistentes os repositórios, gerenciar o raciocínio sobre as informações contextuais mantidas nesses repositórios, inferindo novos contextos a partir de regras lógicas de inferência e de fatos contextuais existentes;

- 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;

- verificar contextos que foram solicitados pelo serviço de adaptação dinâmica do EXEHDA-DA, notificando alterações em seus estados ao Gerente de Notificação;

- notificar alterações nos estados dos contexto coletados e processados para o EXEHDA-DA;

- raciocinar sobre os fatos mantidos nos repositórios, produzindo novos fatos a partir de regras lógicas pré-definidas;

- verificar consistência no servidor de contexto, identificando fatos que não estejam de acordo com as especificações descritas através da descrições ontológicas;

5.1.3 Gerente de Notificação

Esse gerente é responsável por entregar os contextos processados pelo Gerente de Interpretação ao serviço de adaptação dinâmica do EXEHDA-DA. O Gerente de Notificação também recebe solicitações de informações vindas do EXEHDA-DA e comunica ao Gerente de Interpretação do EXEHDA-SS.

O Gerente de Notificação tem por objetivo comunicar as mudanças ocorridas no estado do contexto, informando diferenças em relação ao estado anterior ao serviço de adaptação dinâmica. Quando as informações de interesse, vindas das políticas da adaptação da aplicação e capturadas pelo Gerente de Aquisição sofrerem alterações, o Gerente de Interpretação atualiza e armazena as informações contextuais nos repositórios de contexto. Com essas novas informações, o Gerente de Interpretação de contexto, faz o processamento e, após gerar um novo estado válido de contexto, notifica pelo Gerente de Notificação as mudanças ocorridas para o EXEHDA-DA.

Na próxima seção será descrito o modelo de representação de contexto a ser concebido para emprego no EXEHDA-SS.

5.2 Modelo de Representação de Contexto

A representação das informações contextuais proposta para o ambiente contextual do EXEHDA-SS apresenta diversas associações que podem ser classificadas da seguinte forma:

Quanto à temporalidade:

- Estáticas: são as associações que são fixas e imutáveis durante todo o ciclo de vida da entidade como, por exemplo, a associação entre a entidade Dispositivo e o atributo Modelo;

- Dinâmicas: são todas aquelas que não são estáticas podendo mudar durante o ciclo de vida da entidade como, por exemplo, Largura de Banda;

Quanto à origem dos dados:

- Definidas: trata-se de informações fornecidas. Em geral, são estáticas ou apresentam dinamicidade relativamente baixa, ou seja, que mudam com pouca freqüência como, por exemplo, Login;

- Monitoradas: são aquelas obtidas através de sensores de *hardware* ou de software. Em geral possuem um baixo nível de abstração e o formato varia de sensor para sensor. Em sua maioria, necessitam de um processamento para que sejam padronizadas e para que tenham seu nível de abstração elevado como, por exemplo, Recursos;

- Derivadas: essas são obtidas através de uma função de derivação, a partir de uma ou mais associações como, por exemplo, a associação entre as entidades Usuário e Dispositivo, denominada ”Está próximo de”. Esse tipo de associação é apresentado juntamente com a indicação das outras associações das quais ela depende.

- Depende de: quando a associação é derivada, ela se associa a outra com este tipo, significando que seu valor depende do valor da outra associação.

Para a representação das informações contextuais propostas para o EXEHDA-SS estão sendo concebidos dois modelos contextuais. A seguir serão apresentadas a **OntContext** e a **OntUbi**.

5.2.1 OntContext

A OntContext, figura 5.3, é a ontologia definida para ser responsável pelo registro da situação do contexto, como: estado, identificação do dispositivo, identificação da aplicação/serviço/componente, identificação do usuário, localização, tempo. Ela será integrada ao servidor de contexto do EXEHDA-SS, para a realização do processamento das informações de contexto bem como a utilização de inferências sobre o contexto. A OntContext deverá ser detalhada e integrada ao mecanismo de suporte semântico do EXEHDA-SS.

A OntUbi é a ontologia responsável pela representação do ambiente de execução ubíquo promovido pelo EXEHDA. Suas classes, atributos e associações, estão detalhados na próxima seção.

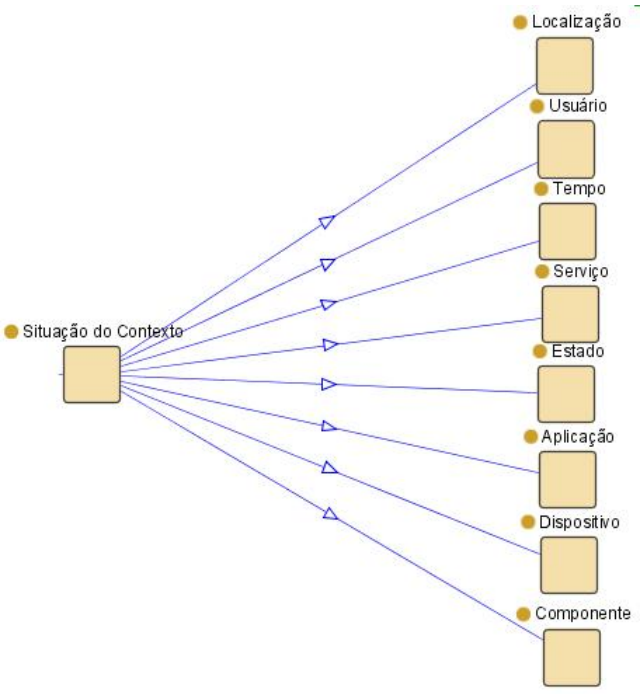

Figura 5.3: Modelo de Representação Contextual na OntContext

5.2.2 OntUbi

A OntUbi é a ontologia básica para a representação de contexto possíveis de interesse para as aplicações, apresentada na figura 5.4. A OntUbi é comporta por três classes básicas, sendo elas: Recursos do Usuário, Recursos da Arquitetura e Recursos de

Software. Essas classes estão atreladas a sua célula. A célula representa o ambiente de execução ubíquo promovido pelo EXEHDA-SS.

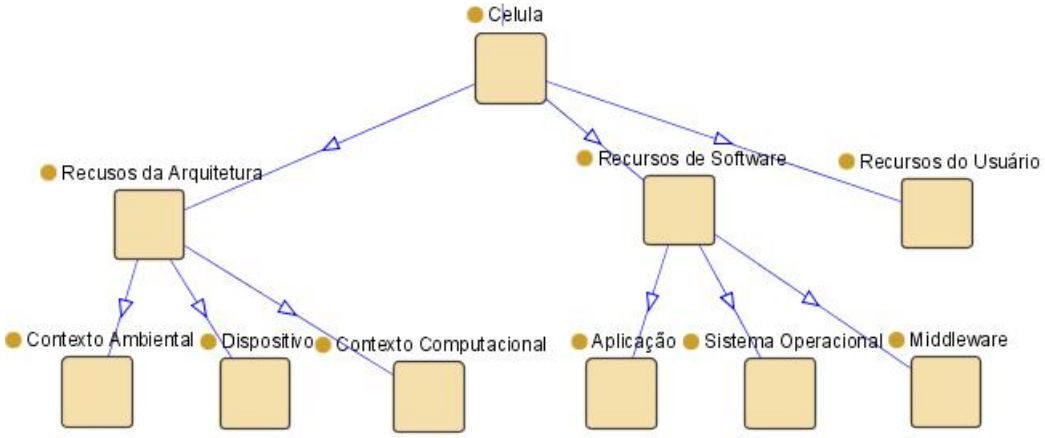

Figura 5.4: Modelo de Representação Contextual na OntUbi

A seguir serão detalhadas os atributos e associações das classes que constituem o modelo de representação de contexto.

1.1 - Recursos do Usuário: representa os recursos dos usuários devidamente registrados e com permissões para acesso ao ambiente contextual. Seus atributos e associações são:

- **Localização**: nome da célula onde o usuário se encontra.
  Associação:
  Monitorado: é obtida por meio de um sensor de localização.

- **Login**: nome de login do usuário.
  Associação:
  Definido: é associado no momento do cadastro do usuário..

- **Perfil**: defini o nível do usuário. Um usuário pode ter um perfil administrador, tendo acesso a todas as atividades na célula. Foram definidos outros perfis, como coordenador e visitante. No coordenador são atribuídas algumas atividades específica, enquanto que no perfil visitante, o usuário tem atividades muito restritas e definidas através do coordenador bem como do administrador.
  Associação:
  Definido: é associado no momento do cadastro do usuário.

- **Atividades**: são as atividades que estão sendo executadas pelo usuário em um dado instante ou foram agendadas por ele. Cada registro possui os seguintes campos: atividade, hora aproximada de início e hora aproximada de término.
  Associação:
  Definido: o usuário pode agendar atividades.
  Temporal: em um dado instante, o usuário está executando uma única tarefa.
  Monitorado: pode-se obter as tarefas do usuário monitorando sua agenda ou as aplicações que estão sendo usadas por ele.

- **Preferências**: são listadas as preferências do usuário. Como qualidade mínima para exibição de imagens, nível de bateria abaixo do qual os componentes e as ferramentas que executam no dispositivo, devem ser finalizadas ou passarem para um estado de hibernação.
  Associação:
  Definido: o usuário escolhe para cada preferência a opção que melhor lhe atende.

- **Necessidades**: são descritas as necessidades para que o usuário execute as aplicações. Por exemplo, uma determinada aplicação necessita que o usuário seja de perfil coordenador, que a aplicação execute com 50MB de espaço em disco com 256MB de Memória RAM.
  Associação:
  Monitorado: pode-se obter as necessidades para a execução de uma aplicação, monitorando o perfil do usuário, suas preferências ou as aplicações que estão sendo usadas por ele.

1.2 - Recursos da Arquitetura: os recursos da arquitetura estão representados em três sub-classes. A seguir serão detalhadas as sub-classes Dispositivo, Contexto Computacional e Contexto Ambiente.

1.2.1 - Dispositivo: esta entidade reflete os dispositivos de *hardware* e seus atributos pouco variáveis que compõem a OntUbi. São, por exemplo, os computadores pessoais de uma rede já instalada, dispositivos móveis, impressoras. Seus atributos e associações são:

- **Características**: para cada modelo de dispositivo existem inúmeras características associadas. Neste etapa da representação contextual, as características foram consideradas, como: tamanho da tela, arquitetura do processador, freqüência máxima do processador, quantidade de cores suportadas e tipo de dispositivo para entrada de dados (teclado convencional, mouse ou teclado de telefone).
  Associação:
  Derivado: o valor de cada característica depende do modelo do dispositivo. Os valores são carregados de um banco de dados contendo os modelos e suas características.
  Depende de: Tem-Modelo

- **Modelo**: a cada dispositivo está associado um modelo
  Associação:
  Estático: é uma característica constante durante toda existência do dispositivo.

- **Localização**: nome da célula onde está o dispositivo.
  Associação:
  Monitorado: é obtida por meio de um sensor de localização.

1.2.2 - Contexto Ambiental: corresponde aos sensores de software/hardware responsáveis por monitorar constantemente os recursos e atualizar este atributo.

- **Sensor**: corresponde aos sensores existentes nos nodos. Especifica o número de processadores de um nodo, a capacidade total de armazenamento dos discos rígidos, o tipo de nodo, identifica se existe usuários logados, a temperatura do ambiente, ruídos e a presença de luz. Os sensores também capturam as informações em um determinado instante de tempo, por exemplo: a temperatura do processador, o total de espaço em disco disponível, carga média que está sendo imposta à CPU, o total de memória que está sendo utilizada no nodo.
  Associação:
  Monitorado: monitora o ambiente de contexto constantemente.

1.2.3 - Contexto Computacional: representa o contexto computacional presente nos dispositivos. Seus atributos e associações são:

- **Serviços**: lista de todos os serviços fornecidos pelo ambiente computacional de um dispositivo (servido http, banco de dados ou aceita tarefas para executar).
  Associação:
  Monitorado: são obtidos pelo monitoramento constante de tudo que está em execução em um dado momento.

- **Largura de Banda**: largura de banda disponível em um dado momento.
  Associação:
  Monitorado

- **Protocolo**: protocolos de comunicação aceitos. São explorados TCP e UDP.
  Associação:
  Temporal: só pode, em um dado instante, usar um dos protocolos citados.
  Derivado: depende do tipo de conexão atual.
  Depende de: Possui-Tipo de conexão.

- **Conexão**: tipos de conexão aceitos pelo dispositivo. Como: *Wireless* 802.11x, *Bluetooth* e *Ethernet*/*Fast Ethernet*.
  Associação:
  Temporal: em um dado instante somente um dos tipos está em uso.
  Derivado: Obtido, com base no modelo do dispositivo.
  Depende de: Tem-Modelo

1.3 - Recursos da Software: os recursos de software definidos na OntUbi estão representados em três sub-classes. A sub-classe Aplicação, *Middleware* e Sistema Operacional.

1.3.1 - Aplicação: representa as aplicações que estão em execução ou cuja execução foi solicitada. Entende-se por execução todo o ciclo de vida da aplicação, a partir do momento em que o usuário a requisita. Os atributos e associações dessa entidade são:

- **ID**: cada aplicação é registrada com um ID único.
  Associação:
  Definido: atribuído no momento do registro da aplicação.

- **Ambiente de Tempo de Execução**: ambiente de tempo de execução exigido pela aplicação. Cada entrada possui os seguintes dados: referência para o código binário da aplicação, ambiente, versão e lista de bibliotecas exigidas. Os ambientes aceitos por exemplo são: JVM, Python.
  Associação:
  Definido: No momento do registro da aplicação, devem ser informados todos os ambientes suportados pela aplicação.

- **Requisitos**: cada aplicação possui um conjunto de requisitos mínimos necessários e opcionais. Os requisitos podem ser definidos para cada binário de forma independente. Cada registro contém: referência para o binário, requisito, valor, tipo (necessário ou opcional).
  Associação:
  Definido: durante o registro da aplicação, os requisitos dever ser preenchidos. Eles podem ser alterados temporariamente no momento da solicitação de execução da aplicação.

- **Status**: armazena o status global da aplicação. São aceitos os seguintes valores: executando, finalizada, suspensa e falhou.
  Associação:
  Temporal: dentre os vários estados possíveis, só assume um a cada momento. Monitorado: pode-se obter o estado atual através do monitoramento de cada tarefa em execução.

- **Tarefas**: quando uma aplicação é executada, cada instância que efetivamente executa é chamada de tarefa. Este atributo lista todas as tarefas da aplicação em execução. Cada registro contém: referência para o binário, nodo em que executa, parâmetros de execução e status.
  Associação:
  Monitorado: podem ser obtidos pelo monitoramento de cada solicitação de execução e do monitoramento dos nodos.

1.3.2 - ***Middleware***: são os componentes de software do *middleware*.

- **ID**: cada componente é registrado com um ID único.
  Associação:
  Definido: atribuído no momento da inicialização do componente.

- **Componente**: é o nome do componente utilizado pelo *middleware*.
  Associação:
  Definido: atribuído no momento da inicialização do componente.

1.3.3 - Sistema Operacional: detalha as características do sistema operacional e sua versão.
Associação:
Definido: é definido no momento da inicialização do ambiente computacional.

A seguir é detalhado a preposição do Motor de Inferência a ser concebido pelo EXEHDA-SS.

5.3 Motor de Inferência de Contexto do EXEHDA-SS

Como descrito no Gerente de Interpretação de contexto anteriormente, está sendo proposto que o EXEHDA-SS utilize de um Motor de Inferência que forneça as aplicações um mecanismo configurável de inferência sobre as informações de contexto baseado em ontologias e em regras, para o emprego de suporte semântico ao EXEHDA-SS. A seguir é descrito o uso de inferências baseada em ontologias:

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 desta ontologia. Assim, pode-se inferir informações de contexto com base em relações de generalização, especialização, transitividade, simetria, inversão, etc.

O Motor de Inferência de contexto pode ser configurado para realizar inferência sobre ontologias codificadas nas linguagens RDF Esquema e OWL. Três tipos de máquinas de inferência têm sido integrados ao Motor de Inferência:

- a máquina de inferência transitiva (*TransitiveReasoner*), que infere relações de especialização e generalização entre classes e propriedades por meio dos construtores da linguagem RDF Esquema para definição de subclasses (rdfs:subClassOf) e subpropriedades(rdfs:subPropertyOf);

- a máquina de inferência RDFS (*RDFSReasoner*), que implementa um subconjunto configurável dos construtores da linguagem RDF Esquema, tais como subclasses, subpropriedades e domínio (rdfs:domain) e imagem (rdfs:range) de propriedades;

- máquina de inferência OWL (*OWLReasoner*), que implementam todas as deduções obtidas de ontologias baseadas em RDF Esquema, bem como diferentes subconjuntos da linguagem OWL. No caso das ontologias OWL, 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. Além das máquinas de inferência OWL disponibilizadas pelo *Jena*, inclui-se a máquina de inferência *Pellet*, ideal para processar ontologias que exploram a máxima expressividade de construtores fornecidos pela linguagem OWL.

Para que uma aplicação sensível ao contexto se beneficie de processos de inferência baseados em ontologia, o serviço de inferência de contexto deve ser invocado com quatro parâmetros de entrada: o tipo de máquina de inferência(*TransitiveReasoner*,*RDFSReasoner*, *OWLReasoner*), uma lista de parâmetros de configuração da máquina de inferência, o conjunto de ontologias cuja semântica será utilizada para direcionar o processo de inferência e um conjunto de fatos instanciados a partir desse conjunto de ontologias.

Vale observar que máquinas de inferência OWL conseguem inferir informações de contexto adicionais em relação à outras dada a semântica da linguagen de ontologia OWL, superior à da linguagem RDF esquema, por exemplo. Mais detalhes a respeitos da

utilização de máquinas de inferência serão encontradas na continuidade da dissertação, o qual descreverá um estudo de caso em que o serviço de inferência de contexto será utilizado em uma aplicação voltada para a área da medicina.

5.4 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 Subscrição.

Também foi discutido nesse capítulo, o modelo de representação de contexto: OntUbi e OntContext. A OntUbi foi detalhada enquanto a OntContext encontra-se em desenvolvimento assim como o Motor de Inferência de contexto a ser integrado a arquitetura do EXEHDA-SS ao *middleware* EXEHDA. O capítulo seguinte, apresenta as considerações finais, as publicações realizadas e o cronograma de atividades.

6 CONSIDERAÇÕES FINAIS

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 de uma situação.

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, 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, buscando otimizar seu processamento.

Um sistema é sensível ao contexto, se ele usa 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 mostram ao usuário informações de contexto.

A utilização de tecnologias de Web Semântica para representação e processamento de informações de contexto traz como vantagens: (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 definem principalmente os diferentes tipos de aplicações, serviços, dispositivos, usuários, entre outros. Além disso o uso de Ontologias servem para modelar o reconhecimento e processamento das informações contextuais. As 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.

Nos esforços inerentes a esta dissertação destaca-se a continuidade da pesquisa do Mestrado em Ciência da Computação, que ocorre dentro do mesmo escopo geral desta dissertação, ou seja, a Sensibilidade ao Contexto na Computação Ubíqua com Suporte

Semântico, através do emprego de tecnologias de Web Semântica.

6.1 Publicações Realizadas

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

6.2 Cronograma de Atividades

Atividades/Meses	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1	x	x	x	x								
2	x	x	x	x								
3			x	x	x							
4			x	x	x	x						
5						x	x	x	x	x	x	
6							x	x	x	x	x	
7									x	x	x	x
8				x	x	x	x	x	x	x	x	x
9							x					
10												x

Atividades:

1. Revisão bibliográfica sobre o escopo do trabalho: computação Ubíqua, sensibilidade ao contexto e ontologias (CONCLUÍDO)
2. Estudo de projetos em medicina ubíqua (CONCLUÍDO).
3. Estudo do *middleware* EXEHDA (CONCLUÍDO).
4. Estudo das tecnologias de web semântica para o emprego de suporte semântico (EM ANDAMENTO).
5. Modelagem do mecanismo para sensibilidade ao contexto proposto (EM ANDAMENTO).
6. Implementação e testes (NÃO REALIZADO).
7. Escrita de artigos sobre o tema da dissertação (EM ANDAMENTO).
8. Escrita da dissertação (EM ANDAMENTO).
9. Seminário de andamento (A SER REALIZADO).

10. Defesa da dissertação (A SER REALIZADO).

De modo resumido os esforços da dissertação II estão focados nos seguintes aspectos:

- Integração do mecanismo de suporte semântico do EXEHDA-SS ao *middleware* EXEHDA;
- Modelagem de aplicações direcionadas a área médica.
- Prototipação e testes.

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.

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.

BERNERS, T. **World Wide Web Consortium (W3C)**. [S.l.]: In: http://www.w3.org/Consortium/ - Acesso em 12/08, 2001.

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, 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**. [S.l.]: In: http://www.daml.org, Acesso em 02/2009, 2009.

DCMI. **Dublin Core Metadata Initiative - DCMI Metadata Terms**. [S.l.]: In: http://dublincore.org/documents/dcmi-terms/, Acesso em 03/2009, 2009.

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

FOAF. **FOAF Vocabulary Specification**. [S.l.]: In: http://xmlns.com/foaf/spec/, Acesso em 02/2009, 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, 2002. 167-180p.

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

IANNELLA, R. **Representing vCard Objects in RDF/XML**. [S.l.]: In: http://www.w3.org/TR/vcardrdf, Acesso em 04/2008, 2008.

ISAM. **Infra-estrutura de Suporte às Aplicações Móveis**. http://www.inf.ufrgs.br/ isam/.

JENA. **Jena - A Semantic Web Framework for Java**. [S.l.]: In: http://jena.sourceforge.net/, Acesso em 02/2009, 2009.

JESS. **Jess - The Expert System Shell for the Java Platform**. [S.l.]: In: http://herzberg.ca.sandia.gov/jess/, Acesso em 03/2009, 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, 2001. 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. **Projeto PertMed**. [S.l.]: In: http://pertmed.wkit.com.br/, Acesso em 03/2009, 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, Accesso em 02/2009, 2009.

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.], 2001. 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**. [S.l.]: In: ttp://www.freeband.nl/FreebandKC/documents?keyword_id=2432, Acesso em 02/2009, 2006.

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.], 2002.

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**. [S.l.]: In http://jena.sourceforge.net/inference/index.html, Acesso em 01/2009, 2006.

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, 2001.

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/lib/exe/fetch.php?id=start&cache=cache&media=nwti.pdf, 2009.

WASP. **WASP Project**: http://www.freeband.nl/projecten/wasp/ENindex.html. [S.l.: s.n.], 2003.

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**. [S.l.]: In Proceedings of VLDB Workshop on Semantic Web and Databases. Disponível em http://www.hpl.hp.com/techreports/2003/HPL-2003-266.pdf, Acesso em 12/2008, 2003.

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.