UNIVERSIDADE FEDERAL DE PELOTAS

Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

SISTEMA DE CONSULTA DE INFORMAÇÕES DE CONTEXTO PARA APOIO A ADAPTAÇÃO DE APLICAÇÕES PERVASIVAS

LUCAS DUTRA NUNES

PELOTAS, 2008

LUCAS DUTRA NUNES

SISTEMA DE CONSULTA DE INFORMAÇÕES DE CONTEXTO PARA APOIO A ADAPTAÇÃO DE APLICAÇÕES PERVASIVAS

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

Orientadora: Prof[a]. Ana Marilza Pernas Fleischmann, MSc.

Co-orientador: João Ladislau Barbará Lopes, MSc.

PELOTAS – 2008

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia - UFPel

N972s Nunes, Lucas Dutra

Sistema de consulta de informações de contexto para apoio a adaptação de aplicações pervasivas / Lucas Dutra Nunes ; orientador Ana Marilza Pernas Fleischmann ; co-orientador João Ladislau Barbará Lopes. – Pelotas, 2008. – 77f. : Il. - Monografia (Conclusão de curso). Curso de Bacharelado em Ciência da Computação. Departamento de Informática. Instituto de Física e Matemática. Universidade Federal de Pelotas. Pelotas, 2008.

1.Informática. 2.Computação pervasiva. 3. Ontologias. 4.Sensibilidade ao contexto. 5.Adaptação ao contexto. I.Fleischmann, Ana Marilza Pernas. II.Lopes, João Ladislau Barbará. III.Título.

CDD: 004.65

BANCA EXAMINADORA

_____

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

_____

Prof. MSc. João Ladislau Barbará Lopes (CI/UFPel)

_____

Prof. Dr. Adenauer Yamin - (UCPel)

_____

Prof. Dr. Mauricio Lima Pilla - (DINFO/UFPel)

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha família pelo apoio, confiança e compreensão durante toda essa jornada, que me foi dado nos momentos mais difíceis. Eu não teria ido até o fim se não fosse por eles.

Agradeço também aos meus orientadores, a professora Ana Marilza Pernas Fleischmann e diretor do Centro de Informática João Ladislau Barbará Lopes, pelo conhecimento transmitido, disposição em me ajudar, e pela compreensão em todas as minhas ações. Essa vitória é tanto minha como de vocês.

Agradeço também a todos os meus colegas, professores e amigos da UFPel, que marcaram os melhores anos da minha vida, e que de alguma forma ou outra sempre me ajudaram, me incentivaram e me trouxeram muita diversão nesses cinco anos do curso.

A todos vocês o meu muito obrigado.

RESUMO

NUNES, Lucas Dutra. **Sistema de Consulta de Informações de Contexto para apoio a Adaptação de Aplicações Pervasivas**. 2008. 78p. Trabalho acadêmico – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas, Pelotas.

Atualmente, pesquisas estão sendo feitas para viabilizar o paradigma da Computação Pervasiva, em especial o aspecto de sensibilidade ao contexto. O projeto GRADEp provê um *middleware* de suporte para o desenvolvimento e execução de aplicações pervasivas. As pesquisas desenvolvidas atualmente nesse projeto têm sido encaminhadas no sentido de dotar o serviço de contexto do ambiente de execução de mais expressividade no gerenciamento de informações do ambiente pervasivo. Neste sentido, observou-se como solução a utilização de ontologias para descrição dos dados de contexto, uma vez que estas permitem uma representação semântica, interpretável por máquina, do ambiente de execução. No momento, a ontologia presente no servidor de contexto oferece informações atuais, geradas por meio de sensores ou de forma manual, a respeito dos dispositivos presentes em cada célula do ambiente pervasivo. A cada alteração ocorrida em um determinado dispositivo, por exemplo, aumento da temperatura do processador, um novo dado relativo a esta mudança é armazenado no servidor de contexto. Considerando-se estas questões de pesquisa, foi modelado e implementado um sistema de consulta ao servidor de contexto celular baseado em ontologias do *middleware* GRADEp. O sistema desenvolvido tem como objetivo central prover informações ao processo de adaptação dos componentes de software que fazem parte das aplicações, uma vez que cada componente de software pode se adaptar de forma distinta às modificações de contexto.

Palavras-chave: Computação Pervasiva. Ontologias. Sensibilidade ao Contexto. Adaptação ao contexto.

ABSTRACT

NUNES, Lucas Dutra. **Sistema de Consulta de Informações de Contexto para apoio a Adaptação de Aplicações Pervasivas**. 2008. 78p. Trabalho acadêmico – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas, Pelotas.

Nowadays, research is being made to make viable the Pervasive Computing paradigm, the context-awareness aspect in special. The GRADEp project provides a middleware for development and execution support of pervasive applications. The research made on this project has been directed to fit the context service on the executive environment with more expressivity in the management of information about the pervasive environment. On this subject, it has been observed as a solution the use of ontologies to describe the context data, since it allow a semantic representation, interpretable by machine, of the execution environment. Today, the ontology present in the context server offer current information generated by sensors or manually, about the devices in each cell of the pervasive environment. On each change on a device, for example, an increase in the processor temperature, new data about this change is stored in the context server. Considering those topics of research, a query system was modeled and implemented in the cell context server, based on ontologies about the GRADEp middleware. The developed system has as central objective to provide information to the adaptation process of the software components that build the applications, since each component can adapt itself in a different way to the context modifications.

Key-words: Pervasive Computing. Ontologies. Context Awareness. Context Adaptation.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CASS – Context-awareness sub-structure

CML – Context Modeling Language

CoBrA – Context Broker Architecture

COBRA-ONT – CoBrA-Ontology

DAML+OIL – DARPA (Defense Advanced Research Projects Agency) Agent Markup Language + Ontology Interchange Language

DTD – Document Type Definition

EXEHDA – Execution Environment for Highly Distributed Applications

GRADEp – GRADE Pervasiva

ISAM – Infra-estrutura de Suporte às Aplicações Móveis Distribuídas

ISAMpe – ISAM Pervasive Enviroment

J2SE – *Java Standard Edition*

J2ME – *Java Micro Edition*

OWL – Web Ontology Language

PC – Personal Computer

PDA – Personal Digital Assistant

RDF – Resource Description Framework

RDQL – RDF Data Query Language

SAX – *Simple API for XML*

SQL – Structured Query Language

SOCAM – Service-oriented context-aware middleware

SOUPA – Standard Ontology for the Ubiquitous and Pervasive Applications

SPARQL – SPARQL Protocol and RDF Query Language (sigla recursiva)

UCPEL – Universidade Católica de Pelotas

UFPEL – Universidade Federal de Pelotas

UFRGS – Universidade Federal do Rio Grande do Sul

UFSM – Universidade Federal de Santa Maria

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

W3C – World Wide Web Consortium

XML – eXtended Markup Language

SUMÁRIO

1 INTRODUÇÃO

A Computação Pervasiva é um paradigma computacional que permite aos usuários acesso aos seus respectivos ambientes computacionais de todo o lugar, o tempo todo. Possui suas origens no final da década de 80, quando Mark Weiser descreveu que o próximo passo da “ubiqüidade” dos computadores pessoais e dos dispositivos computacionais móveis seria a integração para que os usuários possam acessar seu ambiente computacional a qualquer lugar, a qualquer momento.

Weiser disse que as melhores tecnologias são aquelas que desaparecem no ambiente, tornando-se invisíveis a percepção ativa de seu usuário, tornando-se indistinguíveis dos objetos do dia-a-dia. Ainda segundo ele, nesse paradigma a computação deve estar perfeitamente integrada ao ambiente, melhorando as interfaces humano-humano, e reduzindo as interfaces humano-computador.

Nesse contexto, o ambiente pervasivo é o conjunto de todos os dispositivos computacionais do usuário integrados de forma transparente. As aplicações que executam nesses dispositivos são desenvolvidas para agregar funcionalidade ao ambiente. A funcionalidade pode ir desde o monitoramento de alguma atividade trivial até mesmo reconhecimento do estado atual do usuário. Isso é possível porque essas aplicações possuem como característica essencial a Sensibilidade ao Contexto, isto é, modificam seu funcionamento de acordo com o estado atual do ambiente, se baseando em informações adquiridas por sensores e processadas por serviços no ambiente pervasivo.

O *middleware* GRADEp, desenvolvido em (YAMIN, 2004) sobre o nome “EXEHDA”, propõe a arquitetura de um ambiente pervasivo, organizado em células, sensível ao contexto e adaptável, com distribuição a nível global.

Nos últimos anos, a utilização de ontologias para definição de informações dos sistemas sensíveis ao contexto vem sendo considerado como uma solução viável (CHEN *et al*., 2004a). Ontologias permitem que conceitos de um domínio sejam definidos de acordo com regras estabelecidas sobre esse domínio, assegurando que o conhecimento expresso possua um vocabulário comum a todas as entidades que compartilham interesse sobre esse domínio. Dada a característica heterogênica da Computação Pervasiva, isso é de grande validade para a Sensibilidade ao Contexto.

O *middleware* GRADEp originalmente não suporta a utilização de ontologias. Com o trabalho de Lopes (2008), foi criado o EXEHDA-ON, que é um sistema de gerenciamento de informações de contexto do ambiente pervasivo do GRADEp, ao nível celular, utilizando uma ontologia especialmente definida para o ambiente do *middleware*.

Apesar de apresentar o uso de ontologias para auxílio à adaptação ao contexto, não existe ainda uma forma de acesso a informação de contexto por parte das aplicações GRADEp. Desta forma, o presente trabalho se destina a desenvolver um sistema de consulta a dados de contexto, o qual será utilizado pelas aplicações GRADEp para que estas possam conhecer o contexto atual do ambiente, de forma a facilitar sua execução.

1.1 Motivação

A Sensibilidade ao Contexto é uma subárea da Computação Pervasiva sendo ativamente pesquisada, pois possui importância essencial na questão da adaptação do ambiente às necessidades e tarefas do usuário.

A utilização de ontologias na Sensibilidade ao Contexto amplia as a variedade de se obter informações de contexto ao permitir uma maior expressividade nas informações de contexto

O sistema de consultas desenvolvido possui uma interface de comunicação simples, que é mais fácil de ser utilizada do que se uma aplicação pervasiva fosse desenvolvida para acessar, pesquisar e processar os resultados de uma ontologia.

Além disso, questões de escalabilidade, outro tópico importante na Computação Pervasiva, também precisam ser levadas em conta.

1.2 Objetivos

O objetivo central deste trabalho foi, então, desenvolver um sistema de consultas de informações de contexto para apoio a adaptação de aplicações pervasivas, utilizando as informações presentes na ontologia do EXEHDA-ON.

1.3 Contribuição Esperada

Espera-se confirmar que a utilização do sistema de consultas traz as vantagens da utilização de ontologias para a adaptação ao contexto, no caso do ambiente do middleware GRADEp com o sistema EXEHDA-ON.

Outra contribuição esperada é que o acesso, comunicação, criação de consultas, e manipulação dos resultados fiquem a cargo do sistema de consultas, ao invés das aplicações pervasivas, facilitando o desenvolvimento para o programador. Isso significa, em termos práticos, uma menor quantidade de código nas aplicações e menos bibliotecas para serem transferidas pela rede, o que é importante para aplicações em dispositivos móveis, que são mais restritos quanto a recursos.

1.4 Organização do Trabalho

Este documento foi dividido em sete seções, iniciando por seções introdutórias, de conceituação, até as de desenvolvimento e de conclusão.

A seção 2 faz uma introdução ao tema que é o escopo desse trabalho: o paradigma de Computação Pervasiva, abordando temas como conceitos iniciais, desafios atuais e a sub-área de sensibilidade ao contexto.

A seção 3 faz uma introdução ao tema de ontologias, incluindo conceitos, alguns temas relacionados, a sua ligação com a Computação Pervasiva, e a ferramenta utilizada para manipular a ontologia utilizada nesse trabalho.

A seção 4 fala sobre o projeto GRADEp, tratando sobre o *middleware* de mesmo nome e de sua arquitetura, em especial o subsistema de reconhecimento do contexto, e sobre o EXEHDA-ON, uma adição a esse subsistema para a utilização de uma ontologia especialmente definida.

A seção 5 aborda a modelagem do sistema de consultas proposto, sua implementação e sua ligação com o GRADEp e o EXEHDA-ON.

Na seção 6 são comentados os testes realizados para verificar e validar o funcionamento do sistema de consultas, utilizando a biblioteca de testes JUnit.

E finalmente, na seção 7, são apresentadas as conclusões e trabalhos futuros.

2 FUNDAMENTOS EM COMPUTAÇÃO PERVASIVA

Com o avanço da tecnologia, a computação hoje caminha para além do computador pessoal *desktop*. Avanços significativos em áreas como redes, computação móvel e computação distribuída geram grandes possibilidades para criação de soluções que facilitem a vida do usuário. Tem-se hoje uma grande facilidade de se obter acesso a um dispositivo computacional, móvel ou fixo, virtualmente em qualquer lugar.

Mas, embora essas soluções permitam que o usuário tenha acesso a um dispositivo computacional em qualquer lugar, a dificuldade de se acessar os recursos de cada um dos dispositivos disponíveis ao usuário faz com que essas soluções sejam disjuntas. Uma série de motivos, como por exemplo, padrões incompatíveis entre si criados por fabricantes diferentes ou a natureza de alguns dispositivos dificultam a sua interoperabilidade, e de suas aplicações. Além disso, esses dispositivos requerem que os usuários forneçam informações explícitas sobre seu funcionamento, sem que essa interação seja de fato produtiva, como por exemplo, antes que um usuário possa usar uma aplicação no seu PDA (*Personal Digital Assistant*), é necessário que este copie os arquivos necessários para a aplicação de um PC (*Personal Computer*).

Usuários de dispositivos móveis precisam especificar explicitamente quais recursos desejam utilizar na localidade onde estão, como uma impressora ou um serviço da rede, por exemplo. E não há garantia de que o procedimento que o usuário deva realizar para utilizar o recurso (instalar *drivers* da impressora, endereço da rede do serviço) seja o mesmo em qualquer dispositivo, ou ainda que este seja transparente.

Nesse contexto, surge a necessidade de se integrar os ambientes computacionais do usuário, de forma transparente. Essa é a iniciativa da Computação Pervasiva, que é um paradigma computacional que tem por objetivo satisfazer essa necessidade de integração.

2.1 Conceitos Iniciais

Computação Pervasiva é um paradigma computacional onde sistemas computacionais se integram de forma contínua à vida dos usuários, provendo-os com serviços e informação 'em qualquer lugar, a qualquer hora' (CHEN *et al*., 2004a). Outra definição, feita por Lyytinen & Yoo (2002), diz: "Computação Pervasiva (ou ubíqua) são ambientes físicos saturados com computação e comunicação, embora integrados graciosamente com usuários humanos." Ambientes pervasivos objetivam alterar a forma como se dá a computação: ao invés desta ocorrer em um espaço virtual isolado, onde o usuário deve entrar para realizar suas tarefas e conhecer como manipulá-lo, esta passa a ocorrer em meio ao espaço físico do usuário, permeando-o com computação, comunicação e informação, realizando automaticamente tarefas que visam a aumentar a sua produtividade.

Esse paradigma é relativamente novo, sendo idealizado por Mark Weiser no final da década de 1980. Segundo o autor, as tecnologias de maior sucesso são aquelas que se integram ao ambiente, junto aos objetos, de maneira a se tornarem indistinguíveis deles (WEISER, 1991).

Segundo (SAHA, MUKHERJEE, 2003) os avanços tecnológicos para se construir um ambiente computacional pervasivo caem em quatro áreas abrangentes:

a) **dispositivos computacionais;**
b) **rede pervasiva;**
c) ***middleware*;**
d) **aplicações.**

Os dispositivos computacionais presentes em ambientes pervasivos são dos seguintes tipos:

a) dispositivos tradicionais de entrada e saída de dados, como teclados e monitores;
b) sistemas móveis sem fio, como PDAs, celulares e *pagers*;
c) dispositivos inteligentes, como sensores para recolher dados sobre o ambiente e biosensores.

A utilização de sensores que captam, transferem e tomam ações sobre informações captadas automaticamente também é um ponto importante na Computação Pervasiva. Sensores como receptores GPS que transmitem dados de localizações precisas de pessoas ou objetos, e câmeras que transmitem imagens para identificação dessas pessoas e objetos aumentam a quantidade de informação no ambiente, aumentando também as possibilidades de ações e tarefas a serem realizadas.

A rede pervasiva é de extrema importância, pois é através dela que os dispositivos computacionais se integram com sistemas sociais, ou seja, o ambiente físico dos usuários. A Internet pode ser utilizada para formar essa rede pervasiva em escala global (VIVAN, 2007).

Um *middleware* pervasivo é necessário para servir de interface entre a a rede e as aplicações pervasivas executando nos dispositivos computacionais. Esse *middleware* precisa consiste principalmente de conjuntos de *software* e *firmware* especializados para manter a comunicação do ambiente pervasivo, funcionando tanto em modo cliente-servidor como também *peer-to-peer*.

Finalmente, as aplicações pervasivas são o *software* que representa o meio pelo qual o usuário realize uma dada tarefa no ambiente pervasivo, diferentemente da visão tradicional de que um *software* é desenvolvido para se aproveitar de recursos de um dispositivo. Isso significa que as aplicações são desenvolvidas levando em conta sua interação com o ambiente, ao invés da interação com a *Web* ou então em mobilidade.

O ambiente pervasivo precisa que os seus dispositivos sejam integrados por rede e por software, para que as aplicações pervasivas possam melhorar o ambiente físico do usuário. Na Figura 1 abaixo é mostrada como estrutura a Computação

Pervasiva. O *middleware* controla as interações entre o usuário e os dispositivos conectados em rede.

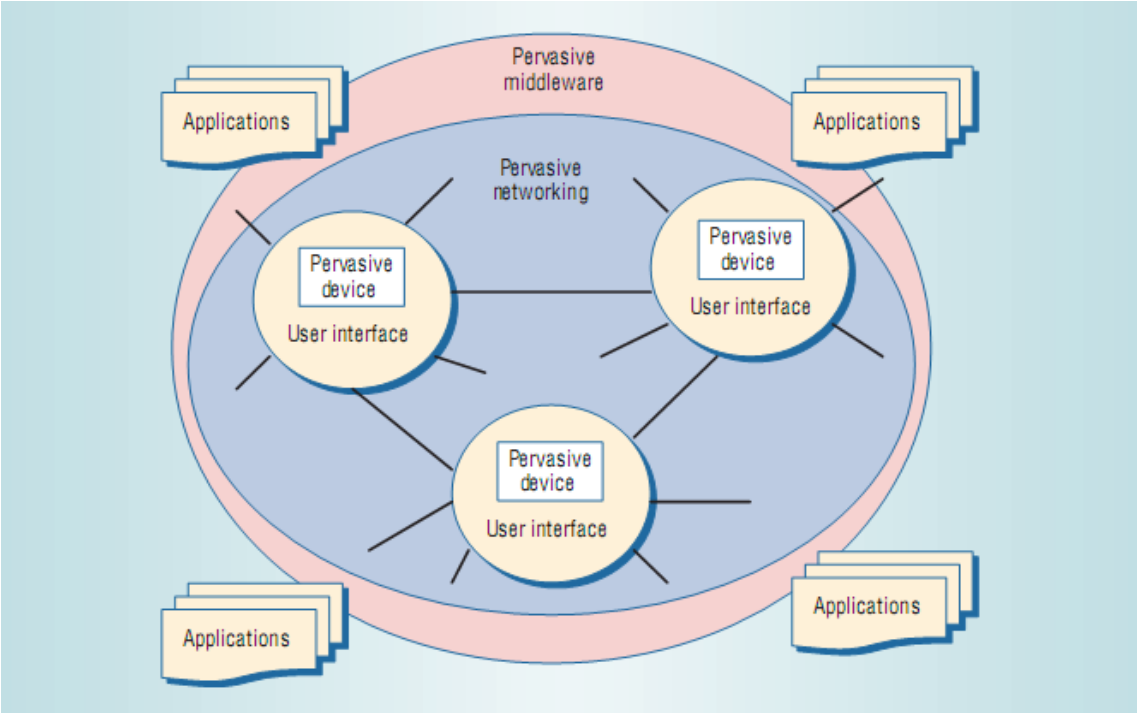

Figura 1 - Estrutura genérica da Computação Pervasiva
Fonte: SAHA, MUKHERJEE, 2003

2.2 Desafios da Computação Pervasiva

A Computação Pervasiva é um super-conjunto da computação móvel, e, portanto, compartilha áreas de pesquisa com essa, ao mesmo tempo em que abre novas áreas únicas. Questões importantes para a Computação Pervasiva incluem as seguintes (SAHA, MUKHERJEE, 2003):

Escalabilidade

Conforme os ambientes pervasivos se popularizam, a quantidade de usuários e dispositivos presentes também aumentam, criando novas interações em uma escala nunca vista. Além disso, aplicações pervasivas precisam executar em plataformas diferentes, e conforme o sistema cresce, e mais aplicações e plataformas diferentes são adicionadas, manter versões diferentes para cada par

[aplicação, plataforma] se tornará impraticável, especialmente se somado a isso a questão do ambiente pervasivo estar distribuído em uma área geográfica ampla.

Heterogeneidade

A conversão de elementos entre dois domínios é essencial para a computação e para a comunicação. Tendo em mente isso, a Computação Pervasiva precisa ser capaz de esconder a diferença que existe entre duas implementações de ambientes pervasivos dos seus usuários. Por exemplo, um laboratório científico e uma loja poderão ter infra-estruturas diferentes para seus ambientes pervasivos, e os ambientes precisam manter a experiência do usuário.

Integração

Para haver uma integração dos dispositivos computacionais no sentido de implementar um ambiente pervasivo, é necessário que estes dispositivos trabalhem em conjunto, tanto no nível de hardware como no nível de software. Um complicador, mencionado anteriormente, ocorre quando os dispositivos ou a rede que os interliga são heterogêneos entre si, ou seja, suas especificações não são iguais, e a quantidade de recursos que esses dispositivos e redes provêem também é diferente. Assim, faz-se necessário desenvolver uma maneira de resolver essas diferenças, mantendo a interoperabilidade.

Nos esforços práticos da Computação Pervasiva, as soluções propostas para esse problema geralmente envolvem a utilização de uma infra-estrutura, que ofereça suporte para o desenvolvimento e para a execução de aplicações pervasivas.

Invisibilidade

Assim como Mark Weiser definiu, as melhores tecnologias são aquelas que se integram ao ambiente, tornando-se indistinguíveis dele, ou então, invisíveis. Para a Computação Pervasiva isso significa que o sistema do ambiente pervasivo requer mínima intervenção humana. O ideal é que a intervenção ocorra quando o sistema falhe em satisfazer as expectativas do usuário automaticamente, e então ajustes ao sistema são feitos. Essa intervenção faz parte de um processo de aprendizado contínuo do próprio ambiente, que pode também realizar ajustes automáticos em

seus processos, sem distrair os usuários na realização de suas tarefas, e diminuindo ainda mais a intervenção humana.

Esses ajustes ocorrem em todos os níveis do ambiente pervasivo, desde auto-configuração de dispositivos de rede até adaptação ao contexto pelas aplicações, o que será visto mais adiante na seção 2.3.

Baseando-se nos conceitos tratados na subseção anterior e nos desafios da Computação Pervasiva mostrados nessa seção, a infra-estrutura desenvolvida para computação pervasiva deve suportar a mobilidade do usuário, das aplicações e dos serviços do ambiente pervasivo, pois esse precisa ter sempre acesso ao seu ambiente computacional. Também se faz necessário que a infra-estrutura seja largamente distribuída, seguindo as premissas da computação em grade.

Como uma das características das aplicações pervasivas é a mobilidade, se faz necessário que as aplicações possuam conhecimento prévio do dispositivo móvel para o qual elas estarão se deslocando. Isso pode ser visto da seguinte forma: a aplicação precisa conhecer, de antemão, quais são as limitações e quais são os recursos oferecidos por um dado dispositivo. A aplicação precisa dessas informações para poder se adaptar ao **contexto de execução** de um dispositivo, por exemplo, um dispositivo com maior largura de banda de rede permite que uma aplicação VoIP possa ter melhor desempenho. Dentre os vários desafios existentes na Computação Pervasiva, o relativo especificamente a sensibilidade ao contexto será tratado neste trabalho.

2.3 Sensibilidade ao Contexto

Segundo (CHEN *et al.*, 2004b), os próximos avanços na Computação Pervasiva envolvem a utilização de sistemas inteligentes que exploram informações sobre o ambiente pervasivo para conhecer o contexto local, usando esse conhecimento para suportar aplicações inteligentes. A Sensibilidade ao Contexto é então definida como sendo a utilização desse conhecimento pelos sistemas presentes no ambiente pervasivo, para que estes possam reagir de forma ativa a alterações nos dados e a novos eventos ocorridos. Eventos dizem respeito a

qualquer tipo de alteração nos dados de contexto, por exemplo: estado do sistema pervasivo, localização do usuário, tempo, temperatura, entre outros.

O termo “contexto” ainda não possui uma definição aceita por todos os autores da área, uma vez que pode ser utilizado e interpretado de formas diferentes por aplicações diferentes. Uma definição abrangente de contexto é dada em (PREKOP; BURNETT, 2003), que define contexto como qualquer informação que possa ser usada para caracterizar a situação de uma entidade, onde entidade é definida como sendo qualquer pessoa, lugar ou objeto considerado relevante na interação entre um usuário e uma aplicação. Já (CHEN; KOTZ, 2000) define contexto de uma forma mais específica, como sendo o conjunto de estados ambientais e configurações que determinam o comportamento de uma aplicação ou um evento de uma aplicação que seja interessante para um usuário. Disso pode-se perceber que o contexto do ambiente pervasivo é dinâmico, se alterando durante a execução de uma aplicação. Por essas definições pode-se concluir também que o estado inicial de execução de uma aplicação sensível ao contexto pode não ser o mesmo esperado durante seu funcionamento.

Na Computação Sensível ao Contexto o ambiente pervasivo precisa que as aplicações sejam capazes de reconhecer e se adaptar a qualquer alteração ocorrida, sendo esta interessante às aplicações, para que a funcionalidade do ambiente como um todo seja mantida. Isso é válido tanto para adaptações necessárias devido a alguma alteração prejudicial ao sistema, como por exemplo, falha em algum ponto da rede, como também para alterações que possam gerar a melhora funcional do sistema, como a adição de mais memória em um nodo computacional.

A utilização de informações de contexto permite que aplicações possam se adaptar melhor as tarefas e necessidades de usuários de ambientes pervasivos. Como uma grande premissa da Computação Pervasiva é o acesso do usuário ao seu ambiente pervasivo a qualquer hora e a qualquer lugar, faz-se necessário o conhecimento do contexto de execução do local e do momento no tempo para que o ambiente pervasivo possa ajudar o usuário da melhor maneira possível. No caso do *middleware* pervasivo estudado e utilizado nesse trabalho, a essa premissa se dá o nome de **Semântica Siga-me**, que será tratada no capítulo 4 deste trabalho.

2.3.1 Tipos de Contexto

São importantes para a adaptação ao contexto não somente alterações de informações sobre comunicação, dispositivos de entrada e saída, estados dos recursos computacionais (uso de processador, memória, espaço em disco etc.), mas qualquer tipo de informação que diga respeito à relação do ambiente pervasivo com o usuário, como por exemplo, preferências pessoais. E o sistema do ambiente pervasivo precisa estar preparado para lidar e até se antecipar a alterações nessas informações sobre o contexto (SATYANARAYANAN, 2001). Satyanarayanan diz também que o ambiente pervasivo precisa inclusive distinguir se o momento é adequado para chamar a atenção do usuário para algum evento, levando em conta até mesmo o estado emocional, o que também pode ser considerado.

De acordo com os tipos de dados relevantes para entendimento do contexto, é possível dividir este em dois tipos. Paul e Mark dizem em (PREKOP; BURNETT, 2003) que os trabalhos iniciais em sensibilidade ao contexto davam mais ênfase a localização atual do usuário, pois restringiam o conjunto de ações a serem executadas pelo usuário, auxiliando-o apenas com informações relativas à localização.

Trabalhos mais atuais têm apresentado maior preocupação com a riqueza dos dados a serem usados para guiar a adaptação ao contexto, sendo explorados dados vindos de sensores para executar a adaptação, como luminosidade, localização de objetos, temperatura, umidade, etc. Outros tipos de sensores menos aparentes também vêm sendo usados, como informações de uso de recursos do ambiente, estado de execução de aplicações, sistema de inferências sobre dados já coletados, entre outros. Em (LOPES, 2008) três tipos de sensores são exibidos:

a) **Sensores físicos**: sensores de hardware capazes de capturar informações físicas, como temperatura, velocidade, pressão, localização, umidade etc.
b) **Sensores virtuais**: sensores para captação de dados vindos de software, como software de calendários e agendas, emails, compreensão de fala, reconhecimento de faces etc.

c) **Sensores lógicos**: sensores que usam informações vindas de outros sensores, físicos ou lógicos, juntamente com informações já disponíveis ao ambiente pervasivo (como um banco de dados) para gerar mais informações de contexto. Ex.: um ambiente pervasivo pode determinar a localização de um usuário usando informações de autenticação, como cartões magnéticos, nomes de usuários e senhas utilizadas em terminais, em um prédio sem sensores físicos presentes.

Percebe-se que muitos dos projetos utilizam sensores físicos para guiar a sensibilidade ao contexto, como é o caso do trabalho de (CHEN; KOTZ, 2002).

Existem também projetos que se utilizam de dados de diferentes naturezas para guiar o contexto, como é o caso do trabalho descrito em (HENRICKSEN; INDULSKA, 2005), que utiliza dados vindos de sensores, de fontes estáticas, providos pelos usuários (através de perfis) e derivados de outros dados já presentes no ambiente. Nesse exemplo, a diferenciação se dá pela utilização de diferentes fontes de dados de contexto para a adaptação do ambiente pervasivo.

3 ONTOLOGIAS

As ontologias vêm sendo utilizadas por várias áreas da Ciência da Computação, principalmente com o intuito de dotar os sistemas de meta-conhecimento (LOPES, 2008). Nessa seção são mostradas as definições para ontologias, seu uso na computação, em especial na Computação Pervasiva, e linguagens associadas aos seus conceitos.

3.1 Definições

Uma das definições de ontologia mais aceita na literatura é a de Gruber (1993), que diz que uma ontologia é uma especificação explícita de uma conceituação compartilhada. Nesta definição, os conceitos representados por uma ontologia, bem como seus objetos e relacionamentos, se forem representados através de um formalismo declarativo, podem ser descritos usando um conjunto de termos representativos que compõem um vocabulário que representa o conhecimento. Esses termos definem entidades do universo representado pela ontologia, como classes, relações, funções e outros objetos.

Outra definição diz que ontologias se destinam a fornecer uma biblioteca para fácil reutilização de classes de objetos para a modelagem de problemas e domínios. O objetivo desta proposta é o desenvolvimento de uma biblioteca de ontologias, a qual poderia ser reusada e adaptada a diferentes classes de problema e ambientes. Além disso, as ontologias são especificadas por um conjunto de axiomas em uma linguagem formal (GRUNINGER, 1996).

Fensel (2000) diz também que uma ontologia provê uma conceituação explícita (meta-informação) que descreve a semântica dos dados. Fensel compara

ontologias com modelos de base de dados, mostrando que diferem significativamente por que:

a) uma linguagem para definir ontologias é sintática e semanticamente mais rica do que as utilizadas para bases de dados;
b) a informação que é descrita por uma ontologia consiste de textos de linguagem natural semi-estruturados, e não de informações tabulares;
c) uma ontologia deve ser uma terminologia compartilhada e consensual pois é utilizada para armazenamento e troca de informações;
d) uma ontologia provê uma teoria de domínio, e não uma estrutura de dados.

Esse trabalho leva em consideração essa última definição, pois esta considera o tratamento de informações segundo uma descrição de um domínio, o qual é bastante relacionado com o conceito de contexto visto na seção anterior.

3.2 Formas de Classificação de Ontologias

Segundo Lopes (2008), as ontologias podem ser classificadas em diferentes tipos, os quais podem variar em função dos autores que as propõem. De modo geral, as proposições feitas por autores como (BORST, 1997) e (STUDER; BENJAMINS; FENSEL, 1998 apud LOPES, 2008), ainda se mantêm, e apontam para a existência de quatro tipos de ontologias:

a) ontologias de domínio: capturam o conhecimento válido para um tipo particular de domínio (como mecânica, medicina, biologia, entre outros). Expressam o vocabulário relativo a um domínio particular, descrevendo situações reais deste domínio;
b) ontologias genéricas: são similares às ontologias de domínio, entretanto os conceitos definidos por elas são considerados genéricos entre diversas áreas. Descrevem conceitos tipicamente gerais, como: estado, espaço, tempo, processo, evento, importância, ação, entre outros, os quais são independentes de um domínio ou problema particular. Conceitos em ontologias de domínio são freqüentemente definidos como especializações de conceitos de ontologias genéricas;

c) ontologias de aplicação: contém todos os conceitos necessários para modelagem do conhecimento requerido por uma aplicação em particular. Esses conceitos correspondem freqüentemente aos papéis desempenhados por entidades do domínio enquanto executam certa atividade;
d) ontologias de representação: não se comprometem com nenhum domínio em particular. Determinam entidades representacionais sem especificar o que deve ser representado. As ontologias de domínio e as ontologias genéricas são descritas por meio das primitivas fornecidas pelas ontologias de representação.

3.3 Elementos que Formam as Ontologias

Ontologias são formadas por elementos que representam uma dada conceituação. A seguir são definidos esses elementos:

a) classes: coleções de objetos do mesmo tipo, que podem possuir subclasses derivadas;
b) indivíduos: instâncias ou objetos de uma classe;
c) atributos: propriedades e/ou características que objetos e classes podem possuir, os quais diferenciam os indivíduos de uma classe;
d) relacionamentos: maneiras que classes e indivíduos podem se relacionar, na forma “é-um”, como por exemplo, o ser humano é um mamífero;
e) restrições: declarações formais que devem ser verdadeiras para que uma afirmação possa ser usada como entrada;
f) regras: declarações na forma de uma sentença “se-então” que descrevem as inferências lógicas que podem ser retiradas de uma afirmação;
g) axiomas: afirmações e regras em uma forma lógica que compõem o conceito geral que a ontologia descreve no seu domínio da aplicação. Nem todas as ontologias precisam ter axiomas, de acordo com o seu domínio;
h) eventos: a mudança de atributos ou relacionamentos.

Esses conceitos são implementados através de linguagens de definição de ontologias. Estas linguagens são tratadas na subseção 3.5 deste trabalho.

3.4 Áreas de Aplicação de Ontologias

Ontologias podem ser aplicadas em diversas áreas, pela necessidade de se classificar conceitos e representar o conhecimento. Essas áreas incluem, mas não se limitando, às seguintes: engenharia do conhecimento, representação do conhecimento, modelagem qualitativa, engenharia de linguagem, projeto de bases de dados, modelagem da informação, integração da informação, análise orientada a objetos, recuperação e extração da informação, manipulação e organização da informação e projeto de sistemas baseados em agentes. As áreas de aplicação são variadas, incluindo integração de empresas, tradução de linguagem natural, medicina, engenharia mecânica, padronização de conhecimento de produtos, comércio eletrônico, sistemas de informações geográficas, sistemas de informações legais, e sistemas de informações biológicas (GUARINO, 1998).

3.4.1 Aplicação de Ontologias na Computação Pervasiva

Ontologias são conceituações compartilhadas sobre um domínio, que podem ser usadas para representar o contexto de um ambiente pervasivo. Segundo (CHEN, FININ, JOSHI, 2004b), ontologias são requerimentos chave para construir sistemas conscientes de contexto pelas seguintes razões: (i) uma ontologia comum habilita o compartilhamento de conhecimento em sistemas abertos e dinamicamente distribuídos, (ii) ontologias com semânticas declarativas bem definidas provêm um meio para agentes inteligentes racionalizar sobre informação contextual e (iii) ontologias explicitamente representadas permitem a dispositivos e agentes interoperarem, mesmo não sendo expressivamente desenhados para trabalhar em conjunto, atingindo ‘interoperabilidade serendipídica’.

Representando o contexto atual do ambiente pervasivo dessa maneira é possível determinar conhecimento útil sobre o próprio ambiente pervasivo, e sobre as suas aplicações que fazem parte desse ambiente, de maneira que todos os entes envolvidos possam se valer deste conhecimento para seu próprio fim. O fato de o

conhecimento sobre o ambiente pervasivo estar representado em forma de uma ontologia também ajuda no sentido de padronizar a maneira que se dá a comunicação entre os entes na ontologia.

Gu, Pung e ZHANG (2004) também afirmam que um modelo para informações de contexto baseados em ontologia permite que contextos sejam descritos semanticamente de maneira independente de linguagem de programação, sistema operacional ou *middleware*, além de permitir também a análise formal do domínio do conhecimento.

3.5 Linguagens de Definição

Como foi visto anteriormente, ontologias devem ser definidas formalmente de acordo com um vocabulário que representa o conhecimento. SU e ILEBREKKE (2002) dividem as linguagens de definição de ontologias em dois grandes grupos, as linguagens ditas tradicionais e as linguagens voltadas a *Web* Semântica. Como a ontologia utilizada neste trabalho foi descrita em uma linguagem que faz parte do segundo grupo, e uma conversão da ontologia utilizada para uma linguagem do grupo tradicional resultaria em esforço adicional, sem fins práticos, somente serão avaliadas linguagens voltadas à *Web* Semântica. A seguir mostra-se a descrição de algumas dessas linguagens.

3.5.1 RDF

O *Resource Description Framework* (RDF) (MANOLA; MILLER, 2004) é um *framework* para representar informação sobre recursos na *Web*. Foi desenvolvido para representar metadados sobre recursos da *Web*, como informações de autoria e *copyright* de páginas da *Web* ou disponibilidade de um recurso compartilhado. Entretanto, atualmente seu uso tem sido aplicado na representação de metadados sobre qualquer item identificável na Web, mesmo que esse item não possa ser acessado na *Web*. O RDF é processável por máquina, e, usando URIs (*Uniform Resource Identifier*), é possível ligar informações espalhadas por toda a *Web*. RDF forma a base para várias linguagens de descrição de ontologias, incluindo a utilizada neste trabalho.

A RDF funciona através de afirmações em forma de triplas, que identificam um recurso, uma propriedade desse recurso e o valor dessa propriedade. Uma tripla RDF indica que existe uma relação entre o sujeito e o objeto da tripla, indicada pelo predicado. Um exemplo seria a seguinte afirmação sobre uma URL (*Uniform Resource Locator*):

"http://www.exemplo.org/index.html foi criado por João da Silva."

A tripla dessa afirmação pode ser decomposta da seguinte forma:

a) O recurso é a URL "http://www.exemplo.org/index.html".
b) A propriedade referenciada é "criador".
c) O valor dessa propriedade é "João da Silva".

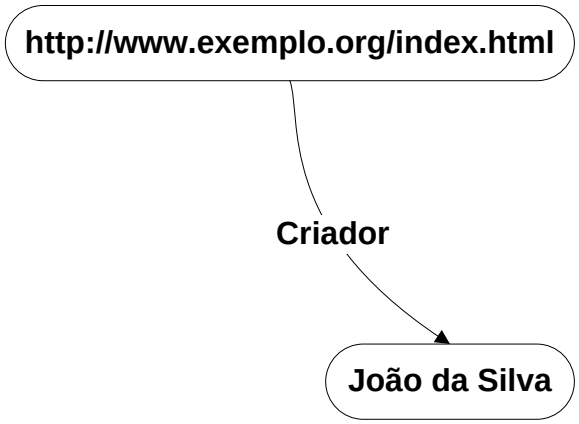

Figura 2 - Exemplo de tripla RDF

O RDF usa uma terminologia particular para identificar as várias partes das afirmações. Os termos, no caso do exemplo, são: *sujeito* para a URL "http://www.exemplo.org/index.html", *predicado* para a propriedade "criador" e *objeto* para a expressão "João da Silva".

Propriedades RDF usam URIs para identificar precisamente os relacionamentos que existem entre os itens ligados. Enquanto uma URL identifica a localização de um recurso em uma rede, um URI é mais geral, não se limitando a identificar coisas que possuam identificação, como por exemplo, números ISBN de livros.

Estruturalmente, o RDF modela as afirmações como nodos e arestas de um grafo. Um nodo representa o sujeito, outro nodo representa o objeto do predicado, e a aresta conectando os dois nodos representa o predicado. Além disso, objetos também podem ser considerados recursos, e podem também ter afirmações a seu respeito representadas no grafo. Além disso, é perfeitamente possível que um dado recurso possa ter mais de uma propriedade, cada uma com o seu próprio valor (MANOLA; MILLER, 2004).

3.5.2 DAML+OIL

DAML+OIL (*DARPA Agent Markup Language* + *Ontology Interchange Language*) (HARMELEN; PATEL-SCHNEIDER; HORROCKS, 2001) é uma linguagem semântica de marcação para recursos da *Web*. Sua estrutura se baseia na RDF, e a estende, provendo primitivas de modelagem mais ricas. A linguagem, como se pode esperar de seu nome, é uma junção de características da linguagem DAML com a linguagem OIL, ambas usadas para expressar ontologias.

DAML+OIL está na versão de março de 2001, mas foi substituída pela linguagem OWL como padrão da W3C (*World Wide Web Consortium*) para expressar ontologias. A linguagem OWL se baseia na DAML-OIL, estendendo suas funcionalidades.

3.5.3 OWL

A linguagem OWL (*Web Ontology Language*) (SMITH; WELTY; MCGUINNESS, 2004) é uma linguagem para definir e instanciar ontologias para Web, reconhecida como padrão pela W3C. A linguagem apresenta estrutura baseada na linguagem declarativa de *Web* Semântica RDF, mas serve para representar conhecimento, ao invés de representar um formato de mensagens (OWL, 2008).

OWL provê três sub-linguagens que servem para serem usadas por comunidades diferentes de desenvolvedores e usuários. São elas:

a) OWL *Lite*, usada para casos onde apenas a classificação e restrições simples bastam para atender as necessidades;
b) OWL DL, usada para casos onde seja necessária expressividade máxima sem perda da completeza computacional (todas as conclusões lógicas são garantidas que serão computadas) e da decidibilidade (todas as computações terminam em um tempo finito) de sistemas de raciocínio;
c) OWL *Full*, usada para casos onde seja necessária expressividade máxima com toda a liberdade possível, mas sem garantias de computação, como por exemplo, tratar instâncias como classes; sobre essa sub-linguagem o W3C informa que talvez nunca exista um software de raciocínio que suporte toda funcionalidade prevista sobre o OWL *Full*.

Cada uma dessas linguagens é uma extensão da anterior, ou seja:

- Cada ontologia OWL *Lite* corretamente definida também é uma ontologia OWL DL legal;
- Cada ontologia OWL DL corretamente definida também é uma ontologia OWL *Full* legal;
- Cada conclusão OWL *Lite* válida também é uma conclusão OWL DL válida;
- Cada conclusão OWL DL válida também é uma conclusão OWL *Full* válida;

A ontologia definida em Lopes (2008) foi desenvolvida utilizando a linguagem OWL DL, por isso será a utilizada neste trabalho. Além disso, a OWL possui o poder de expressividade necessário para descrição de informações de contexto.

3.6 Linguagens de Consulta

As linguagens de definição de ontologias permitem que informações sobre o domínio modelado sejam manipuladas e armazenadas. Entretanto, apenas armazenar as informações não é suficiente, é preciso realizar consulta a essas

informações armazenadas. Para isso foram desenvolvidas linguagens específicas para consulta de ontologias, as quais são descritas nas próximas seções.

3.6.1 RDQL

A linguagem de consultas RDQL (*RDF Data Query Language*) (SEABORNE, 2004) é uma linguagem desenvolvida a partir de idéias utilizadas em outras linguagens. Uma pesquisa RDQL consiste em uma tripla RDF (sujeito-predicado-objeto) com nomes de variáveis, a qual opcionalmente pode conter restrições. A sintaxe é bastante parecida com a linguagem de manipulação de banco de dados SQL (*Structured Query Language*).

Abaixo segue um exemplo de pesquisa RDQL, a qual expressa um exemplo genérico de pesquisa para obter o URI de um recurso de acordo com a propriedade fictícia `<http://example.com/someType>` e do tipo também fictício `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`. Note que essa também é uma pesquisa SPARQL válida, que será vista na próxima subseção:

```
SELECT ?x
WHERE (?x,  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
         <http://example.com/someType>)
```

Figura 3 - Exemplo fictício de pesquisa RDQL.
Fonte: SEABORNE, 2004.

A linguagem RDQL está sendo abandonada em favor da linguagem SPARQL, por esta última conter melhorias consideráveis, além de ser padrão recomendado pela W3C. Portanto, optou-se pela não utilização da RDQL neste trabalho.

3.6.2 SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language*) (PRUD'HOMMEAUX; SEABORNE, 2004), pronunciado *sparkle* em inglês, é uma linguagem de consultas originalmente desenvolvida para acessar dados sobre objetos na Web, representados através do modelo de metadados RDF, que usa declarações sobre recursos na forma de expressões sujeito-predicado-objeto,

chamada de triplas. Como linguagens de representação de conhecimento como OWL foram desenvolvidas em cima desse modelo, SPARQL também pode ser utilizada para acessar informações de ontologias.

A sintaxe SPARQL é semelhante a da linguagem de manipulação de dados de banco de dados SQL. As consultas usam as cláusulas *select* e *where* para determinar, respectivamente, quais variáveis de uma tripla devem ser retornadas, e qual o padrão usado para comparar com a ontologia para recuperar os dados.

SPARQL é um super-conjunto de RDQL, possuindo todas as suas funcionalidades e adicionando novas, sendo a principal das diferenças a definição de tipos de dados, que não está presente na definição da RDQL. SPARQL se tornou padrão recomendado pela W3C em 15 de janeiro de 2008, o que reforça o seu uso para extrair dados que se apresentam no formato RDF. Abaixo segue um exemplo de consulta SPARQL. Aqui a pesquisa retornará uma relação de: títulos de livros e seus preços, se o preço for abaixo de 30, ou então títulos sem preço:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

Figura 4 - Exemplo de consulta SPARQL sobre livros
Fonte: PRUD'HOMMEAUX; SEABORNE, 2008.

3.7 Protégé: Ambiente para Manipulação de Ontologias

O desenvolvimento de uma ontologia pode ser feito inteiramente utilizando um editor de texto simples para definir os conceitos básicos, já que a maioria das linguagens de definição de ontologias utiliza padrões XML, o qual é legível por humanos (HAROLD; MEANS, 2002), como base para sua sintaxe. Porém, esse método é extremamente trabalhoso. Desta forma, foram desenvolvidos ambientes de desenvolvimento que auxiliam a definição de ontologias, geralmente apresentando interfaces gráficas para melhor representar os conceitos expressos na ontologia sendo desenvolvida. Nesse trabalho, embora não envolva o desenvolvimento de uma ontologia, ainda assim se faz necessário a utilização de um ambiente adequado

para estudar a ontologia empregada, e realizar testes sobre as consultas SPARQL desenvolvidas.

O *Protégé* é um editor de ontologias *open-source* desenvolvido pela Universidade de Stanford em colaboração com a Universidade de Manchester. É também um *framework* para o qual vários outros projetos desenvolvem extensões, permitindo expandir as capacidades iniciais do editor, permitindo criar e manipular ontologias sobre perspectivas diferentes do padrão do Protégé.

A escolha desse editor se baseou no fato que esse foi o editor no qual a ontologia utilizada foi desenvolvida, além de possuir toda a funcionalidade necessária para a análise da ontologia utilizada no protótipo desse trabalho. A Figura 5 a seguir ilustra o *Protégé* aberto na edição de classes, exibindo a estrutura da ontologia.

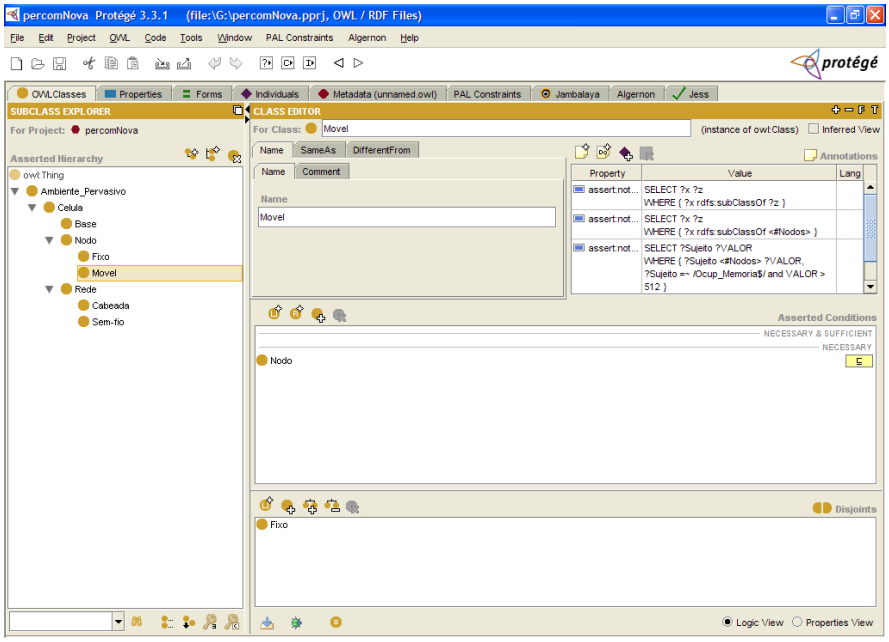

Figura 5 - Editor de classes do Protégé

4 VISÃO GERAL DO PROJETO GRADEp

A discussão nesse capítulo descreve os conceitos de um ambiente pervasivo implementado pelo *middleware* GRADEp, o qual faz parte do projeto ISAM. É também descrita a proposta de utilização de ontologias para a questão de sensibilidade ao contexto, na subseção do EXEHDA-ON.

4.1 Projeto ISAM

YAMIN (2004) descreve em sua tese de doutorado a arquitetura de um *middleware* adaptativo, com orientação a serviços, para suportar a execução de aplicações pervasivas, chamado de **EXEHDA** (*Execution Environment for Highly Distributed Applications*). O *middleware* provê suporte de software para a criação de ambientes pervasivos, dinâmicos quanto à disponibilidade de recursos, e podendo apresentar alta heterogeneidade de equipamentos envolvidos com o ambiente.

Esse *middleware* faz parte de um projeto maior, o **ISAM** (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas), que tem como objetivo prover uma infra-estrutura de suporte para o projeto, implementação e execução de aplicações pervasivas em escala global. Esse projeto teve início na UFRGS (Universidade Federal do Rio Grande do Sul) em 2001, mas já possui várias outras frentes de pesquisa com outras universidades, como na UCPEL (Universidade Católica de Pelotas), UFSM (Universidade Federal de Santa Maria), e na UFPEL (Universidade Federal de Pelotas).

No projeto ISAM o ambiente pervasivo é provido pelo *middleware*, o qual define três áreas relacionadas (YAMIN *et al*., 2004): o ISAMpe (ISAM *Pervasive*

Environment), aplicações ISAMadapt e gerenciamento de serviços que suportam a semântica siga-me das aplicações pervasivas.

Durante a sua existência, o *middleware*, que se chamava EXEHDA, foi rebatizado para GRADEp, e na literatura ambas as nomenclaturas se referem ao mesmo *middleware*. Nessa monografia isto também ocorre.

Por questão de simplicidade e de abstração com o tema do trabalho, serão tratados apenas alguns aspectos da arquitetura do GRADEP, isto é, somente aqueles pertinentes ao presente trabalho. Para maiores detalhes é sugerida pesquisa a bibliografia relacionada.

4.2 Ambiente Pervasivo

O ambiente físico do *middleware* é composto por recursos e serviços, disponibilizados por dispositivos computacionais do sistema, assim como os dispositivos que provêem infra-estrutura para o seu funcionamento. Cada dispositivo é configurado de acordo com seu perfil de execução. Todos esses elementos descrevem o ambiente, o qual é gerenciado pelo *middleware*.

O ambiente pervasivo é organizado por **células** computacionais, interconectadas, cada uma contendo **nodos** representados pelos dispositivos computacionais e equipamentos de infra-estrutura. Cada célula é auto-contida, ou seja, nenhum recurso essencial para a execução dos serviços básicos do ambiente pervasivo está presente em outra célula. Na Figura 6 abaixo é mostrada o ambiente pervasivo ISAM.

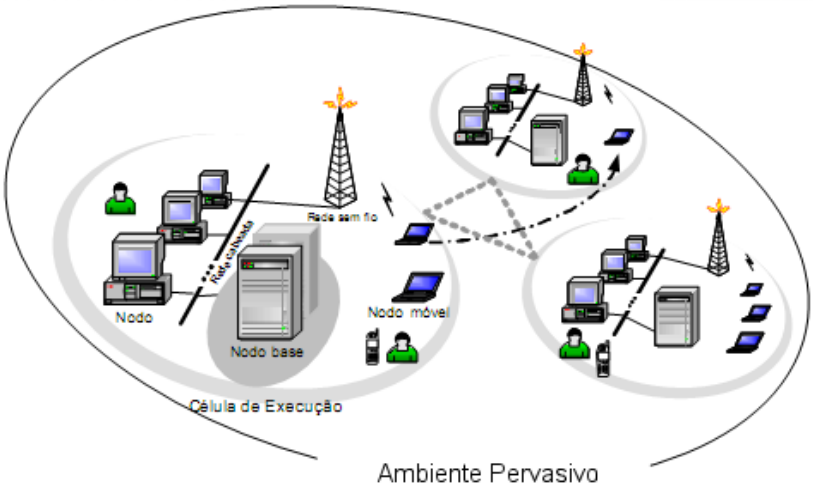

Figura 6 - Ambiente pervasivo do GRADEp
Fonte: YAMIN, 2004

Uma célula é composta por dois tipos de abstrações lógicas específicas:

a) **Nodo base**: nodo lógico único na célula, que contém todos os serviços básicos de gerenciamento do ambiente da célula em qual está presente. Vale notar que o nodo base de uma célula pode ser composto de vários nodos computacionais físicos, interligados entre si. Isso vai de encontro com a solução de problemas de escalabilidade com o eventual crescimento da célula, seja por adição de nodos.

b) **Nodo**: nodo computacional que pode aparecer zero ou mais vezes na célula, os quais oferecem os recursos básicos para a execução das aplicações pervasivas, como processador, memória e conexão de rede com o resto da célula. Outros recursos também podem estar presentes no nodo, mas estes devem ser explicitamente catalogados no nodo base, de acordo com um arquivo XML definido na arquitetura do *middleware*, para que possam ser utilizados. Esses nodos possuem outra denominação se tiverem características de mobilidade física: **nodo móvel**. Os nodos assim denominados são essencialmente iguais aos nodos não-móveis, apenas podem mover-se fisicamente, e por isso possuem conexão de rede sem fios para comunicação com outros nodos da célula.

Como o ambiente pervasivo provido possui escala global, a organização das células foi desenhada para funcionar ao nível institucional. Isso leva a um procedimento de gerência análogo ao praticado em ambientes de Grade Computacional (*Grid Computing*) (GEYER *et al*., 2002 apud YAMIN, 2004). Essa forma de gerenciamento da organização celular é objetivada levando em conta a autonomia das instituições envolvidas no ambiente pervasivo.

4.3 Arquitetura do GRADEp

O middleware GRADEp possui arquitetura orientada a serviços, executando sobre plataformas heterogêneas, onde há grande variação nos recursos disponíveis pelos dispositivos. Na **Erro! Fonte de referência não encontrada.** a seguir está descrita a visão geral da arquitetura do *middleware*.

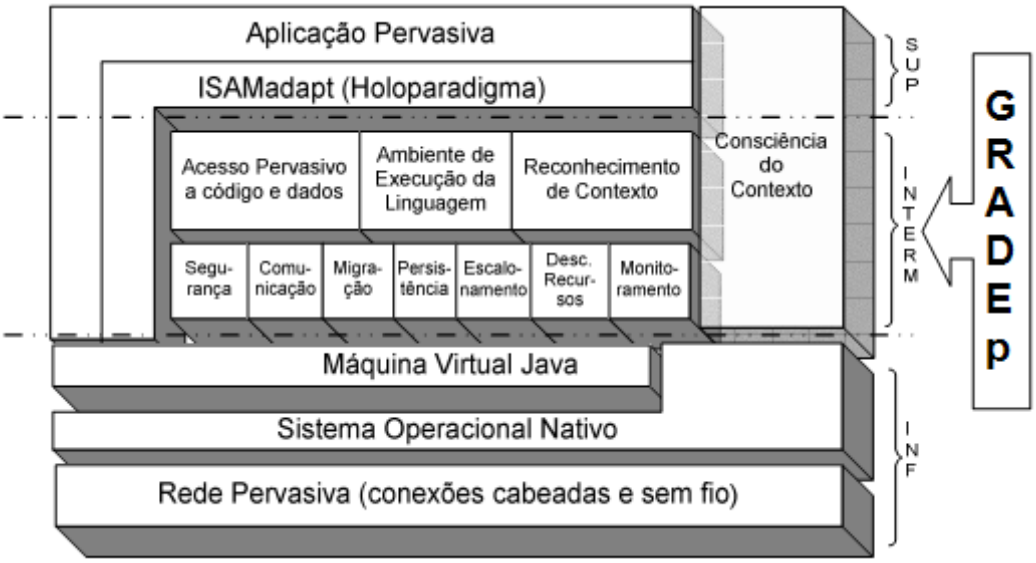

Figura 7 - Visão geral da arquitetura GRADEp
Fonte: YAMIN, 2004.

Na camada superior está localizada a linguagem de programação ISAMadapt, a qual disponibiliza abstrações para a programação de aplicações pervasivas.

A camada intermediária se subdivide em dois níveis. No nível superior estão localizados os subsistemas de **acesso pervasivo a código e dados**, de **ambiente de execução da linguagem** e de **reconhecimento de contexto**.

O subsistema de acesso pervasivo a código e dados disponibiliza o ambiente pervasivo, com os módulos de Ambiente Virtual do Usuário (AVU), Ambiente Virtual da Aplicação (AVA) e Base de Dados pervasiva das Aplicações (BDA). O módulo AVU é o módulo que contém os elementos de interface das aplicações com o usuário, e é responsável por permitir que as aplicações se desloquem conforme necessário para outros nodos sem descontinuidade na sua execução. O módulo AVA identifica as instanciações específicas de cada aplicação. Já o módulo BDA é o repositório de código executável das aplicações pervasivas presentes na célula.

O subsistema de ambiente de execução da linguagem é o encarregado pelo gerenciamento da aplicação durante seu tempo de vida

O subsistema de reconhecimento de contexto é responsável por informar o estado dos elementos de contexto de interesse de um consumidor de contexto e do próprio ambiente de execução. Efetivamente, o suporte a adaptação do GRADEp e de seus consumidores dependem diretamente deste componente. É parte da funcionalidade expressa nesse módulo que o presente trabalho aborda as suas questões, e que será visto em detalhes mais adiante.

Um consumidor de contexto é qualquer código executável que precise de informações de contexto para a sua execução. Essa nomenclatura engloba aplicações comuns que acessam os serviços de contexto do GRADEp, aplicações pervasivas com adaptação ao contexto, ou até serviços que também se adaptem ao contexto de execução do ambiente. Esse capítulo usa a nomenclatura “consumidor de contexto”, já o capítulo sobre o sistema de consultas usa a nomenclatura “aplicação pervasiva”, pois o foco do sistema é permitir o acesso a informações de contexto para essas aplicações.

No nível inferior estão os serviços mais básicos do *middleware*, que provêm a funcionalidade necessária para que os serviços do primeiro nível da camada intermediária possam executar. São eles:

a) **migração**: mecanismos para deslocar uma aplicação fisicamente para outro nodo da célula;
b) **persistência**: mecanismo para aumentar a disponibilidade e o desempenho ao acesso aos dados para as aplicações;
c) **descoberta** de recursos: para permitir o deslocamento de dispositivos computacionais móveis entre as células;
d) **comunicação**: para prover comunicação anônima e assíncrona na célula e entre células;
e) **escalonamento**: para escolher o melhor nodo da célula ao instanciar uma aplicação pervasiva;
f) **monitoramento**: sensores que fornecem informações sobre o ambiente de execução da aplicação.

Finalmente, a camada inferior é composta pelos sistemas, linguagens e infra-estrutura que estão disponíveis no nodo que está executando o *middleware*. Por questões de portabilidade, o *middleware* foi implementado utilizando a linguagem Java, nas plataformas J2SE (*Java Standard Edition*) e J2ME (*Java Micro Edition*). A arquitetura supõe que há uma rede em escala global, com suporte a operação sem fio, interligada a outra rede cabeada, para prover a infra-estrutura de comunicação e serviços do ambiente pervasivo.

4.4 Semântica Siga-me

Uma das características que o *middleware* contribui é que as aplicações "seguem" o usuário onde ele estiver, dentro do ambiente pervasivo, permitindo que ele tenha acesso fácil ao seu ambiente computacional. A essa característica se dá o nome de "semântica **siga-me**", que define que "o usuário executa suas aplicações no local em que se encontra, mesmo em deslocamento" (YAMIN, 2004).

A arquitetura do *middleware* define que as aplicações podem executar em qualquer dispositivo computacional que faça parte do ambiente pervasivo, independente das características desses dispositivos. O reconhecimento de contexto, o acesso pervasivo aos dados e o sistema de comunicação, juntos, dão suporte a semântica siga-me.

Levando em consideração a afirmação anterior, o reconhecimento de contexto é relacionado a dois mecanismos: **monitoração**, que adquire informações sobre os recursos e aplicações do ambiente; **adaptação**, que altera o código de um consumidor de contexto de acordo com alterações nas informações adquiridas pelo primeiro mecanismo.

As adaptações podem ser de duas modalidades: **funcional** e **não-funcional**. A adaptação funcional é aquela definida pela aplicação, que altera a execução da aplicação segundo alterações nas informações monitoradas. A adaptação não-funcional é aquela definida pelo sistema do GRADEp para alterar a localização física dos componentes das aplicações, ao instanciar uma nova aplicação ou ao migrar uma aplicação já instanciada para outro nodo, para acompanhar o usuário em um deslocamento físico, por exemplo.

4.5 Subsistema de Reconhecimento de contexto

O subsistema de reconhecimento de contexto é o componente do GRADEp responsável pela adaptação ao contexto no ambiente pervasivo. O subsistema é formado por vários componentes projetados para adquirir informações brutas sobre o ambiente pervasivo, agregar e transmitir essas informações para seu destino, e processar essas informações para abstrações maiores. Lopes (2008) desenvolveu uma alternativa a esse subsistema, também para prover informações de contexto, mas utilizando ontologias, e será abordado na seção 4.6. Os componentes que realizam cada um desses serviços são descritos a seguir.

4.5.1 Collector

O serviço *Collector* aglutina as informações brutas do ambiente pervasivo, vindas de vários monitores, e as envia para os consumidores registrados no *Collector* com interesse nas informações. Um monitor é um conjunto de sensores parametrizáveis do ambiente pervasivo. Esses serviços existem em uma unidade executando para cada nodo de uma célula, e também para o base.

Vale notar que é perfeitamente possível um *Collector* ser consumidor de outro *Collector* de um nodo diferente, aumentando a quantidade de informações disponíveis para um consumidor.

O *Collector* é responsável por modificar os parâmetros de operação dos sensores, como por exemplo, a freqüência de *pooling* de um dado sensor. Essa alteração se dá por intermédio do monitor associado ao sensor.

4.5.2 Deflector

O serviço *Deflector* é responsável por transmitir informações de contexto para todos os consumidores associados a essas informações, através da implementação da abstração de canais de *multicast*. Pela natureza da abstração *multicast*, a utilização desse serviço vai ao encontro à busca por escalabilidade para a arquitetura do *middleware*.

O *Deflector* é um serviço único que está localizado no nodo base, e possui em sua interface métodos para a criação de canais, inscrição e desinscrição de canais criados, e de disparo de disseminação de informações de um canal.

4.5.3 ContextManager

O serviço *ContextManager* é o serviço responsável pelo tratamento da informação bruta produzida pelos monitores para a geração de informações mais abstratas referentes a elementos de contextos. A definição de um elemento de contexto é feita a partir de um documento XML que descreve todos os possíveis estados desse elemento de contexto através das informações retornadas pelos monitores. Ao ser criado um novo contexto, os interessados neste podem se registrar para receber qualquer modificação a respeito desse contexto.

O *ContextManager* gera as informações de contexto com três componentes internos:

a) *aggregator*, responsável por compor os dados recebidos dos monitores em um valor agregado;

b) *translator*, responsável por converter o valor agregado em um valor abstrato, conforme definido pelo documento XML;
c) *notifier*, responsável por gerar eventos de notificações de alteração de um contexto quando os valores abstratos do *translator* são alterados.

4.5.4 AdaptEngine

O serviço *AdaptEngine* é responsável por gerenciar a adaptação funcional, ou seja, aquela disparada pelo consumidor de contexto. O serviço provê funcionalidade de alto nível para a gerência do comportamento adaptativo das aplicações, ocultando a gerência de elementos de contexto do *ContextManager*.

O serviço *AdaptEngine* trabalha em conjunto com o *ContextManager*, monitorando o estado de cada elemento de contexto registrado. O serviço permite que códigos executáveis diferentes de um consumidor de contexto, cada qual relacionado a um estado de contexto, possam ser carregados e executados no evento de alteração do contexto. É possível também que um consumidor de contexto possa esperar para que um dado estado de contexto ocorra para só então continuar a executar.

4.5.5 Scheduler

O *Scheduler* é o serviço utilizado para a adaptação não-funcional, ou seja, disparada pelo sistema, ao invés da aplicação, e que não implica em alteração de código executável da aplicação. Esse serviço é utilizado para escolher os nodos adequados no caso de uma instanciação ou de uma migração, ou quando um dado recurso não atende mais as exigências da aplicação para a sua execução.

Na subseção seguinte será mostrado o EXEHDA-ON, uma adição ao middleware GRADEp visando a expressão do contexto utilizando uma ontologia. Uma figura que ilustra como os serviços do subsistema de reconhecimento de contexto do GRADEp, descritos aqui, estão relacionados com os elementos do

ambiente pervasivo, e também como que o EXEHDA-ON se integra ao GRADEp, também é mostrada abaixo.

4.6 EXEHDA-ON

Lopes (2008) propõe a utilização de uma ontologia especialmente desenvolvida para a arquitetura do ambiente pervasivo, para auxiliar a suprir a necessidade de expressividade de informações de contexto pelo *middleware*.

A modelagem do sistema é composta por dois esforços principais: a modelagem ontológica do ambiente pervasivo e do contexto do ambiente, e a

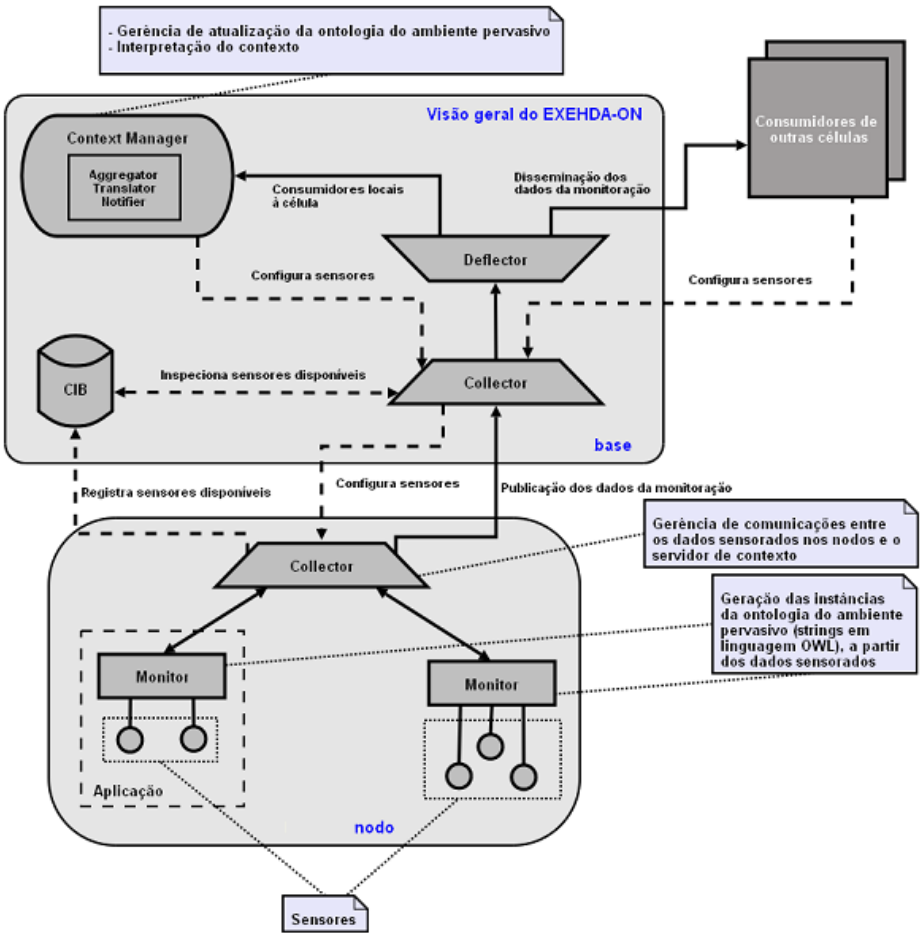

Figura 8 - Arquitetura do subsistema de reconhecimento de contexto em conjunto com o EXEHDA-ON
Fonte: LOPES, 2008

modelagem da arquitetura de software com a especificação dos diferentes serviços que a integram.

As funcionalidades do EXEHDA-ON integram-se ao GRADEp, conforme pode ser observado na Figura 8 acima.

4.6.1 Modelagem Ontológica do EXEHDA-ON

O modelo ontológico foi criado tendo em mente o ambiente pervasivo disponibilizado pelo GRADEp. As informações sobre o ambiente podem ser tanto de natureza estática ou dinâmica. Informações de natureza estática são aquelas que não se alteram conforme o tempo, como por exemplo, dados sobre as especificações de dispositivos e capacidade máxima de uma dada conexão física de rede. Informações da natureza dinâmica são aquelas que variam conforme o tempo, como por exemplo, ocupação do processador ou de memória, ou se um usuário está utilizando um nodo computacional. As informações são obtidas por monitoramento periódico, coletadas por procedimentos disparados por eventos no ambiente, ou ainda de configurações de perfil do sistema (YAMIN, 2004).

Desta forma, a modelagem ontológica baseou-se no entendimento sobre o

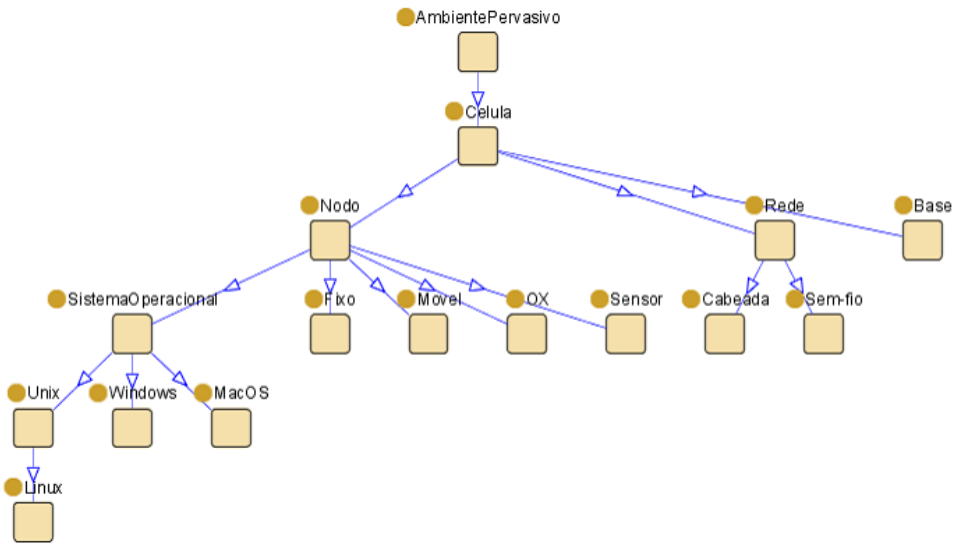

Figura 9 - Árvore de conceitos da ontologia do ambiente pervasivo
Fonte: Lopes (2008)

domínio do ambiente pervasivo, que engloba os tipos de dispositivos, os sensores, os componentes de software e a infra-estrutura das redes de interconexão. Um glossário de termos foi então desenvolvido, e uma estutura em árvore de todos os componentes do domínio foi criada. A Figura 9 acima mostra a hierarquia em árvore das classes do modelo ontológico proposto.

Ao descrever a modelagem da ontologia, Lopes (2008) utilizou a linguagem OWL-DL, mencionada na seção 3.5.3.

Além disso, foi também modelada uma ontologia que define o contexto de interesse das aplicações pervasivas. Essa ontologia representa um subconjunto do ambiente pervasivo, que define informações que o consumidor de contexto precisa para seu funcionamento. Quando o contexto ocorre na ontologia, consumidor de contexto é notificado. Esse modelo também possui um glossário de termos comuns ao domínio, também descrito em OWL-DL. A Figura 10 a seguir mostra a hierarquia em árvore das classes do modelo de contexto de interesse das aplicações.

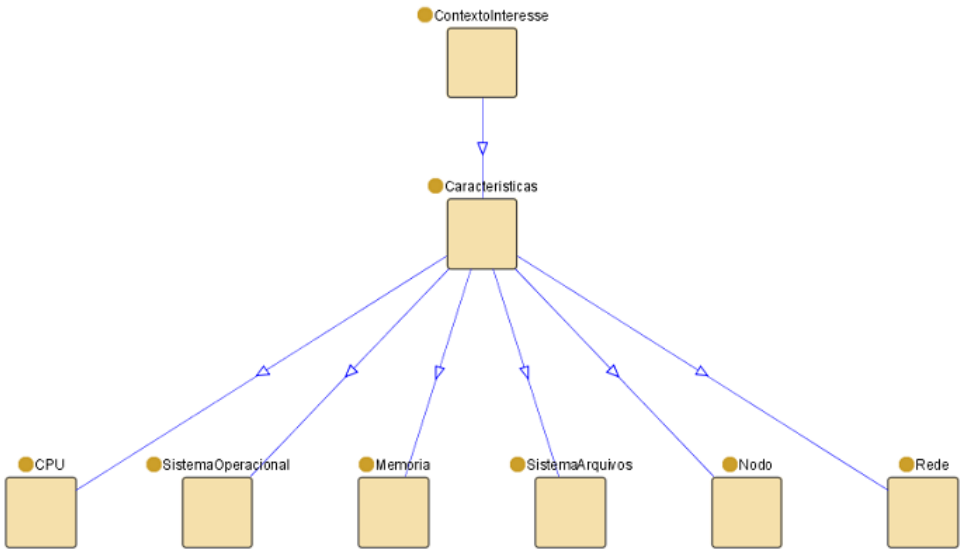

Figura 10 - Árvore de conceitos do contexto de interesse das aplicações
Fonte: Lopes (2008)

4.6.2 Modelagem da Arquitetura de Software do EXEHDA-ON

A modelagem da arquitetura básica adotada foi semelhante a que já existia na modelagem do *middleware* original, mais precisamente à arquitetura do Subsistema de Reconhecimento de Contexto. A arquitetura define que serviços adicionais são adicionados aos nodos e ao nodo base das células do ambiente pervasivo.

Em cada nodo o serviço de “Gerenciamento de monitoramento do Contexto” recebe os dados processados pelo componente “Monitor” do *middleware*. Esses dados são então transformados em instâncias da ontologia do ambiente pervasivo, representadas por documentos OWL, e são incorporados à ontologia da célula.

O EXEHDA-ON usa os serviços *aggregator* e *translator* para compor os dados de um ou mais sensores e para abstrair a informação de contexto, e incorporá-la à ontologia. Essa incorporação se dá com o serviço “Gerenciador de atualização da base ontológica”, que se localiza junto do serviço *ContextManager* do *middleware*, no nodo base. No nodo base também se localiza o serviço responsável pelas inferências sobre a base ontológica do EXEHDA-ON, denominado “Interpretador de contexto”.

O EXEHDA-ON possui diversos serviços encarregados de processar e distribuir as informações de contexto baseando na ontologia modelada. A Figura 11 abaixo mostra uma visão desses serviços e de seus relacionamentos, os quais são descritos a seguir:

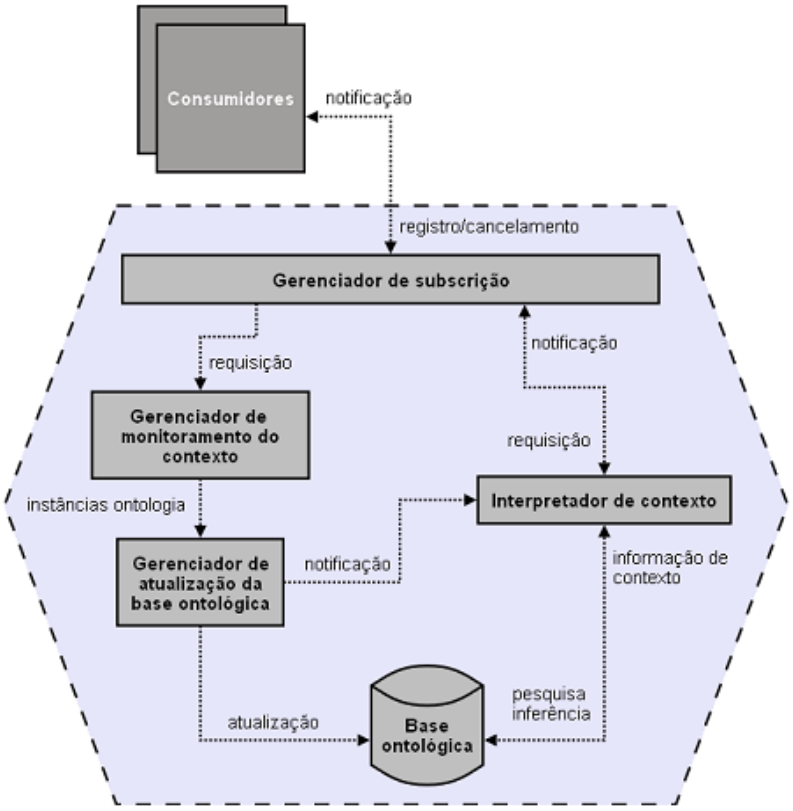

Figura 11 - Visão geral dos serviços do EXEHDA-ON
Fonte: Lopes, 2008

4.6.2.1 Gerenciador de subscrição

O Gerenciador de Subscrição é responsável por interagir com os consumidores de informações de contexto prestadas pelo EXEHDA-ON. Os consumidores informam as suas requisições ao Gerenciador, e este as distribui para todos os serviços do EXEHDA-ON, para identificar o contexto desejado e notificar a sua existência aos consumidores.

4.6.2.2 Interpretador de contexto

O Gerenciador de Subscrição envia para o Interpretador de Contexto as solicitações dos consumidores, para que este realize consultas na base ontológica a fim de encontrar os contextos desejados. Esse serviço realiza consultas e inferências na base ontológica para encontrar os contexto.

4.6.2.3 Gerenciador de monitoramento do contexto

Se o Interpretador de Contexto não for capaz de encontrar o contexto desejado, o Gerenciador de Subscrição encaminha o pedido para o Gerenciador de monitoramento do contexto, que utiliza o subsistema de Reconhecimento de Contexto do *middleware* para ativar sensores, receber dados sensorados e gerar instâncias da ontologia em documentos OWL, para atualizar a base ontológica.

4.6.2.4 Gerenciador de atualização da base ontológica

Esse serviço de atualização recebe as instâncias da ontologia do ambiente pervasivo, no formato de documentos OWL, e atualiza a base ontológica. Também notifica o Interpretador de contexto quando ocorre uma atualização nesta ontologia.

5 MODELAGEM DO SISTEMA DE CONSULTAS

Como foi mostrado na seção 4, a utilização de ontologias vem se mostrando uma solução viável para armazenar informações de contexto em ambientes pervasivos. A discussão nesse capítulo trata de propor uma modelagem de um sistema de consultas, baseado em uma ontologia que descreve os conceitos de um ambiente pervasivo implementado pelo *middleware* GRADEp. É também descrita a implementação de um protótipo sobre a modelagem desse sistema, além de todas as ferramentas utilizadas.

O sistema EXEHDA-ON adiciona a capacidade de representar em uma ontologia o ambiente pervasivo, ao *middleware* GRADEp, ao nível celular do ambiente, monitorando sensores espalhados pela célula, e atualizando a ontologia. Porém, o software desenvolvido baseando-se na arquitetura e organização descritas não possuía mecanismos explícitos para que as aplicações pervasivas pudessem acessar as informações de contexto. A proposta desse trabalho é, então, a modelagem e prototipação de um sistema de consultas, que trabalha em conjunto com o EXEHDA-ON para o acesso dessas informações de contexto pelas aplicações pervasivas. A subseção a seguir expressa como ocorreu a modelagem, implementação e testes desse sistema.

5.1 Modelagem

A modelagem do sistema de consultas se baseou na necessidade de prover informações sobre quais nodos possuem condições de satisfazer as necessidades de contexto das aplicações pervasivas. A aquisição dessas informações é feita através de consultas SPARQL na ontologia expressa pelo EXEHDA-ON. As

consultas são geradas dinamicamente de acordo com a requisição da aplicação, sendo expressas por um documento XML que é passado ao sistema de consultas pela aplicação pervasiva no momento do pedido. A resposta do sistema se dá por meio de outro documento XML, que indica os nodos que cumprem os requisitos dos estados de contexto descritos, ou uma mensagem de erro no caso do processamento do contexto não poder ser concluído.

5.1.1 Requisitos

Para projetar o sistema, vários requisitos básicos foram considerados, sendo os mesmos descritos nos parágrafos a seguir.

O sistema deve ser capaz de receber qualquer entrada, de forma robusta, mesmo que a entrada não esteja formatada corretamente pelos padrões estabelecidos do sistema. Isso é necessário para que o sistema se mantenha em funcionamento mesmo que as aplicações enviem requisições mal-formadas. O sistema deve retornar um erro ao consumidor de contexto em caso de entrada mal-formada.

O sistema deve funcionar como serviço, sem estar profundamente associado ao GRADEp e ao EXEHDA-ON. A arquitetura do sistema deve então estabelecer uma interface para expor o seu serviço prestado, com o mínimo de acoplamento com essas entidades.

O sistema deve ser organizado de forma modular, para permitir a fácil integração ao GRADEp e ao EXEHDA-ON. Isso é particularmente importante no caso do acesso a ontologia, já que a mesma é acessada pelo EXEHDA-ON também para executar modificações.

O sistema deve retornar, opcionalmente, informações de diagnóstico de funcionamento e condições de erro em uma saída padrão, a qual pode ser alterada (redirecionada).

O sistema deve possuir acesso a ontologia no momento de sua inicialização, e só pode continuar a execução normal (isto é, consultar a ontologia) se esse acesso

a ontologia foi executado com sucesso. Caso contrário, o sistema deve enviar uma mensagem de erro à saída de erros, e recusar qualquer pedido de contexto.

Os detalhes de como ocorrerá a comunicação do sistema com as aplicações pervasivas devem ser abstraídos ao sistema, para diminuir o acoplamento. O sistema deve possuir um módulo que define a comunicação.

A comunicação entre o sistema de consultas e as aplicações deve ser semelhante a forma como o serviço *ContextManager* do GRADEp se comunica com os seus consumidores. Portanto, um documento XML, chamado de XML de contexto, é usado para as requisições, e que é semelhante na sua estrutura ao documento XML descrita para o serviço ContextManager, do middleware GRADEp. Esse XML deve ser validado de acordo com um DTD (*Document Type Definition*), descrito no apêndice A da monografia, que define como o desenvolvedor de uma aplicação pervasiva pode expressar o contexto.

Para permitir uma abstração maior, o documento XML deve permitir que o contexto seja expresso utilizando os chamados **estados de contexto**, cujo conceito é semelhante ao apresentado por YAMIN (2004), no serviço *ContextManager*. Cada estado de contexto possui relações entre propriedades das classes da ontologia modelada no EXEHDA-ON, e valores associados a essas propriedades. A quantidade de estados de contexto que existem no documento não é limitada, assim como a quantidade de propriedades descritas em um estado de contexto também não é limitada.

As propriedades descritas no documento XML podem ter diferentes associações com os dados. De acordo com a estrutura do XML descrito em YAMIN (2004), um valor de uma propriedade pode:

a) ser o próprio valor único;
b) estar em uma faixa entre dois valores;
c) estar dentro de um limite máximo ou mínimo.

O sistema precisa então oferecer suporte a essas características. Por exemplo, pode ser interessante para o contexto de uma aplicação conhecer quais

nodos possuem a quantidade total de memória (propriedade) maior que um dado valor.

5.2 Estado de contexto

O estado de contexto é uma abstração central para o sistema de consultas. A escolha de se usar tal abstração vem do fato que o módulo *ContextManager* do *middleware* possui uma abstração semelhante, que se baseia no uso de informações sobre os nodos combinadas conforme o desenvolvedor da aplicação pervasiva precisa. Como o modelo de contexto utilizado aqui pode ser definido como “um conjunto de atributos que descrevem o estado das entidades” (YAMIN, 2004), ao agregar determinadas informações da ontologia (os “atributos”), é possível representar o contexto requerido por uma aplicação.

Na elaboração de como se dará a requisição de um estado de contexto foram escolhidos três operadores para expressar os valores dos estados:

a) ***single value***, ou **valor único**: a propriedade no estado de contexto só pode assumir um valor único;
b) ***range***, ou **faixa**: a propriedade no estado de contexto pode assumir qualquer um dos valores compreendidos entre dois valores informados através dos atributos *from* e *to* (“de” e “até”);
c) ***limit***, ou **limite**: a propriedade no estado de contexto pode assumir qualquer um dos valores abaixo ou acima do valor, através dos atributos *lessthan* (menor que) e *greaterthan* (maior que).

A combinação de várias propriedades, cada uma com seus valores e operadores, expressa o estado do contexto. As declarações de vários estados de contexto similares expressam, então, a requisição de um contexto para a aplicação pervasiva.

5.2.1 Documento XML de contexto

O documento XML que a aplicação pervasiva utiliza representa a abstração do seu modelo de contexto, conforme foi mostrado anteriormente. Esse documento,

ao ser recebido pelo sistema de consultas, é validado por um documento DTD especialmente gerado para o formato de entrada de documento XML do sistema de consultas. O documento DTD pode ser visto no Apêndice A deste trabalho. Abaixo, na Figura 12, é apresentado um exemplo de documento XML que expressa uma abstração de como se pode considerar uma dada quantidade de memória de computadores.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CONTEXTXML SYSTEM "ContextDTD.dtd">
<CONTEXTXML>
    <VERSION>1.0</VERSION>
    <CONTEXT>
        <NAME>MemQuant</NAME>
        <STATE>
            <NAME>pouca</NAME>
            <PROPERTY>
                <NAME>TotMemoria</NAME>
                <LESSTHAN>256</LESSTHAN>
            </PROPERTY>
        </STATE>
        <STATE>
            <NAME>media</NAME>
            <PROPERTY>
                <NAME>TotMemoria</NAME>
                <RANGE>
                    <FROM>256</FROM>
                    <TO>1024</TO>
                </RANGE>
          </PROPERTY>
        </STATE>
        <STATE>
            <NAME>bastante</NAME>
            <PROPERTY>
                <NAME>TotMemoria</NAME>
                <GREATERTHAN>1024</GREATERTHAN>
            </PROPERTY>
      </STATE>
    </CONTEXT>
</CONTEXTXML>
```

Figura 12 - Exemplo de documento XML de contexto

5.2.2 Arquitetura

Baseando-se nos requisitos definidos, o sistema de consultas foi arquitetado de forma modular, procurando manter a semelhança com o componente *ContextManager* do *middleware* e com o Interpretador de Contexto do EXEHDA-ON. Os módulos possuem suas tarefas bem definidas e possuem interfaces reduzidas, mas robustas, diminuindo o acoplamento e aumentando a coesão do sistema e de

seus módulos. Na Figura 13 abaixo é mostrada a arquitetura geral do sistema de consultas.

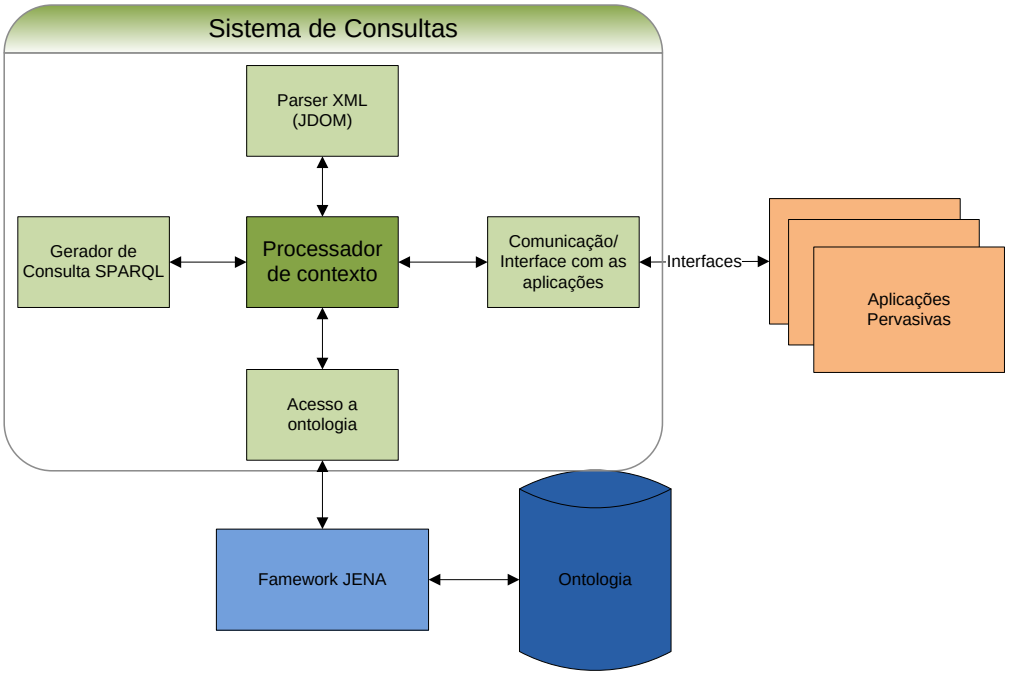

Figura 13 - Arquitetura do sistema de consultas

O sistema é composto por cinco módulos principais: **comunicador**, **processador de contexto**, ***Parser* XML**, **Gerador de consultas SPARQL** e **Acesso a Ontologia**. Cada um desses módulos será explicado a seguir.

5.2.2.1 Comunicador

O módulo comunicador é responsável por representar a interface de comunicação do sistema de consulta com qualquer aplicação interessada em obter informações de contexto da ontologia. Esse módulo possui sua *thread* independente de execução, para permitir que requisições de contexto sejam feitas sem precisar interromper o processamento de outras requisições.

Ao receber uma nova requisição, o módulo guarda internamente a referência de chamada de volta para a aplicação e o documento XML contendo a descrição do contexto, ao mesmo tempo em que passa o documento para a fila de processamento do processador de contexto.

Mesmo que um processamento de contexto esteja sendo realizado, o Comunicador continua esperando por novas requisições de contexto. Conforme novos pedidos chegam para ser processados, o Comunicador continua postando essas novas requisições no Processador de contexto para sua execução.

O Processador de contexto, ao terminar, posta o XML de retorno para o Comunicador, que usa a referência da aplicação para acionar uma chamada de método para enviar o XML para a aplicação. A chamada de método é definida por uma interface que a aplicação deve implementar para utilizar o Sistema de Consultas.

5.2.2.2 Processador de Contexto

O processador de contexto é o módulo de gerenciamento do sistema de consultas, que controla a aquisição das informações de contexto chamando os métodos de outros módulos e adicionando as informações em cada objeto de requisição conforme são processadas por outros módulos. A sua arquitetura prevê a presença de uma *thread* de execução, a qual é utilizada para processar serialmente cada um dos pedidos. O processador possui em sua interface métodos para ativação e desativação de si próprio, e postagem de novos pedidos de contexto.

O processador de contexto verifica a ocorrência de algum estado de erro a cada estágio do processamento de uma requisição. Caso positivo, o sistema ativa a recuperação de mensagem de erro, e cria uma mensagem de retorno para a aplicação, usando o módulo *Parser* XML, informando o ponto do processamento em que ocorreu o erro e o motivo. O processador também verifica se o acesso a ontologia está estabelecido durante a inicialização do sistema de consultas, e recusará qualquer pedido de contexto até que a ontologia esteja acessível e pronta para ser consultada.

5.2.2.3 *Parser* XML

O módulo *Parser* XML é responsável por tratar e montar documentos XML utilizados no sistema de consultas. Seu funcionamento está associado à biblioteca

JDOM, que é utilizada para abstrair documentos XML como objetos Java, facilitando sua manipulação.

O módulo presta três serviços básicos, acessados através de métodos estáticos do módulo: instanciação e retorno de objetos de estado de contexto representados em um documento XML, criação de documento XML de retorno para a aplicação se baseando na pesquisa da ontologia, e criação de uma mensagem de erro para a aplicação, caso o processamento normal não seja concluído com sucesso. A chamada dos métodos é sempre feita pelo Processador de Contexto.

O serviço de instanciação dos objetos de estado de contexto é feito chamando um método estático que recebe por parâmetro uma *string* contendo o documento XML a ser processado. A *string* é então processada pelo *parser* SAX (*Simple API for XML*) presente na biblioteca JDOM, que verifica se o documento é bem-formado, o que é exigência para qualquer representação de um documento XML (HAROLD; MEANS, 2002), e verifica se o documento é válido, isto é, se a estrutura do documento está de acordo com o documento DTD elaborado para a entrada de dados no sistema de consultas. Verificada essas condições, o documento XML é então processado, instanciando objetos que representam os estados de contexto, e retorna uma lista contendo todos os objetos instanciados.

O serviço de criação de documentos XML de retorno recebe como parâmetro as identificações dos nodos encontrados no resultado da pesquisa para cada estado de contexto, e monta o documento XML para refletir esses resultados. A Figura 14 abaixo mostra um exemplo de documento XML de retorno.

```
<CONTEXT>
  <NAME>LimpezaDisco</NAME>
  <STATE>
    <NAME>necessaria</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
  <STATE>
    <NAME>naoNecessaria</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
</CONTEXT>
```

Figura 14 - Exemplo de documento XML de retorno

Finalmente, o serviço de criação de mensagem de erro gera um XML com uma mensagem de erro que pode ser utilizada para informar à aplicação que aconteceu um erro que impossibilita a execução normal do sistema de consultas. Um exemplo de uma mensagem de erro é a que segue abaixo, na Figura 15:

```
<?xml version="1.0" encoding="UTF-8"?>
      <CONTEXT>
         <NAME>"MemQuant"</NAME>
         <ERROR>
         "State in property "Tot_Memoria", on context
state "media" is missing a "from" or a "to" attribute"
      </ERROR>
</CONTEXT>
```

Figura 15 - Exemplo de XML com mensagem de erro

5.2.2.4 Gerador de Consultas SPARQL

O Gerador de Consultas SPARQL é responsável por criar consultas para cada estado de contexto que existe no pedido da aplicação pervasiva. Sua interface consiste em um único método estático, que recebe como parâmetro uma lista de estados de contexto, cada um com propriedades e valores associados, e adiciona a cada objeto de estado de contexto uma string contendo a consulta SPARQL gerada, pronta para ser utilizada na pesquisa.

A consulta contém filtros relativos às propriedades e seus valores e operadores. Como um estado de contexto pode conter várias propriedades, as consultas podem ser tão complexas quanto for necessário.

5.2.2.5 Acesso a Ontologia

O módulo de acesso a ontologia usa as consultas geradas pelo Gerador de Consultas para encontrar os nodos que atendem os estados de contexto. O serviço prestado por esse módulo recebe como parâmetro cada estado de contexto expresso em uma requisição de uma aplicação, depois de terem sido geradas as consultas para esses estados.

Depois de realizada uma consulta, o módulo coloca a identificação (nome) de todos os nodos que atendem ao estado de contexto sendo pesquisado. Isso é feito para cada estado de contexto.

O funcionamento geral do sistema de consultas fica evidente no diagrama da Figura 16 abaixo.

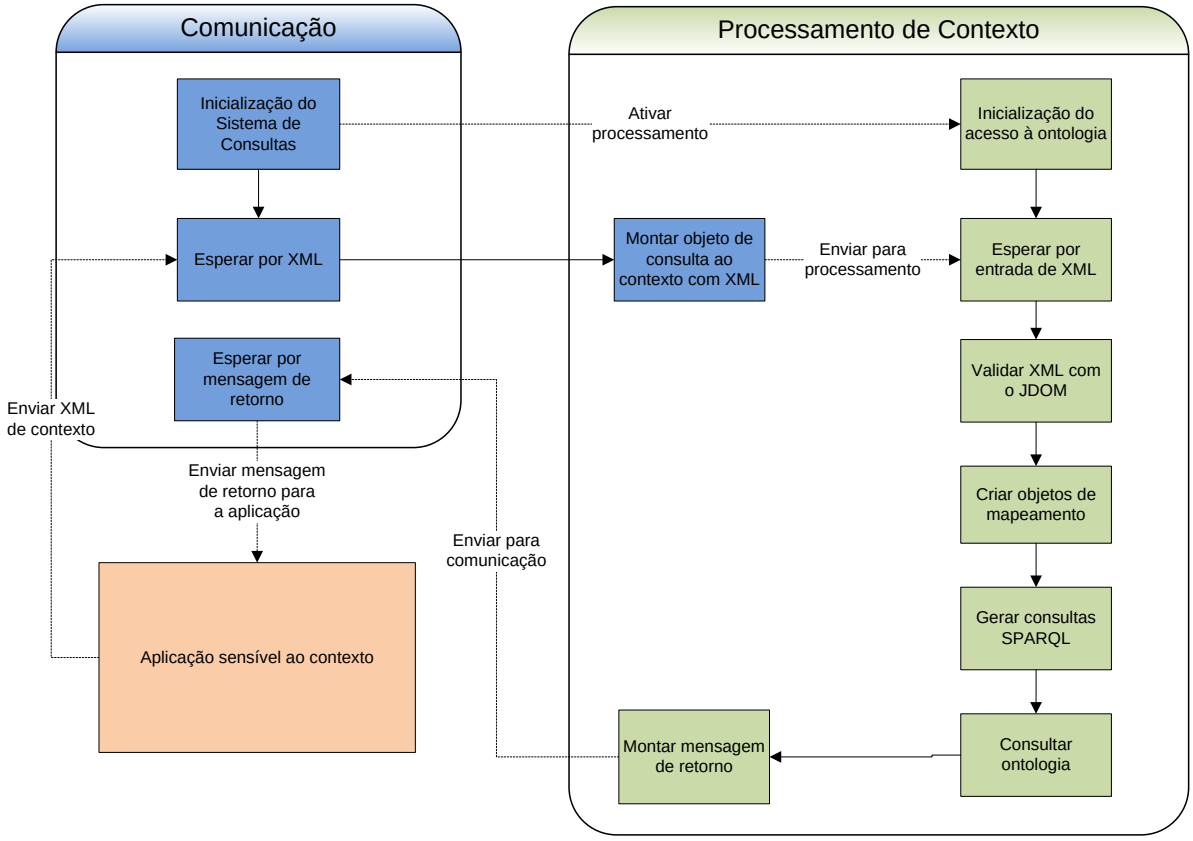

Figura 16 - Diagrama de fluxo de execução do sistema de consultas

5.3 Consultas SPARQL

A arquitetura do sistema de consultas prevê a pesquisa da ontologia através da linguagem SPARQL mencionada na seção 3.6.2. Essas consultas são geradas dinamicamente para cada pedido de uma aplicação por um contexto, para pesquisar a ontologia do EXEHDA-ON em busca de um nodo que apresente o contexto pedido. A Figura 17 a seguir mostra um exemplo de consulta gerado pelo sistema de consultas, que procura nodos aonde a quantidade total de memória presente é maior que 256 MB:

```
PREFIX ont:  <http://www.owl-ontologies.com/unnamed.owl#>
SELECT ?node
WHERE { ?node ont:TotMemoria ?TotMemoria
FILTER ( ?TotMemoria < 256 ) }
```

Figura 17 - Exemplo de consulta SPARQL

Uma consulta é gerada para cada estado de contexto recebido da aplicação. Os valores passados pelos estados são utilizados para montar as cláusulas de filtro da consulta.

Uma consulta pode retornar zero ou mais soluções que passam pelos filtros dessa consulta. Cada solução possui o nome de referência de um nodo da célula que, no momento da consulta à ontologia, está no estado de contexto estabelecido pela consulta.

5.4 Implementação do Protótipo do Sistema de Consultas

Um protótipo baseado na modelagem acima descrita do sistema de consultas foi implementado para validar a modelagem do sistema com testes. A metodologia de desenvolvimento seguida foi o desenvolvimento orientado a testes, aonde uma seqüência de testes foi implementada em código para cada funcionalidade adicionada ao sistema (RAINSBERGER; 2005).

5.4.1 Tecnologias Usadas na Implementação

Para a implementação do protótipo do sistema de consultas foram usadas várias ferramentas. Todas elas são *open*-source, e tem seu uso não comercial irrestrito. Segue a seguir uma descrição de cada uma delas:

Linguagem de desenvolvimento: foi utilizada a linguagem Java SDK (*Software Development Kit*) versão 6, *update* 6 (JAVA, 2008) para o desenvolvimento do sistema. A escolha por esta linguagem foi feita porque tanto o *middleware* como o Servidor de Contexto EXEHDA-ON foram implementados nela,

e, portanto, ambos oferecem suporte natural e interoperabilidade com o sistema de consultas.

Ambiente de desenvolvimento: Foi utilizado o IDE (*Integrated Development Environment*) *NetBeans* 6.1 (NETBEANS, 2008) instalado com Java versão 6, *update* 6 para escrever o código relativo ao programa. A escolha desse ambiente de desenvolvimento se deve ao fato que o *NetBeans* possui integração direta com a linguagem Java, além de oferecer suporte a várias adições (*plug-ins*) que foram úteis ao desenvolvimento.

Controle de versões: O controle de versões da implementação foi feito utilizando um servidor *Subversion* (SUBVERSION, 2008), conectado na Internet, juntamente com o *plug-in Subversion* do *NetBeans* para realizar as atualizações das versões de código com o servidor.

Documentação: A documentação do código foi gerada através de comentários no formato da ferramenta JavaDoc (JAVADOC, 2008), sendo que o IDE *NetBeans* possui suporte nativo para a geração dessa documentação. Diagramas de classes foram gerados através do *plug-in* UML do *NetBeans* com o gerador automático embutido.

Realização de testes: o sistema de consultas foi testado exaustivamente utilizando-se o framework de testes automatizados *JUnit* (JUNIT, 2008), versão 4.0, integrado ao ambiente *NetBeans*. A prototipação do sistema de consultas foi realizada com a metodologia de desenvolvimento orientado a testes. Os testes realizados levaram em consideração o funcionamento dos métodos das classes desenvolvidas, a integração das classes e o funcionamento do sistema de consultas como um todo. Na seção 6 estão descritos os testes realizados com o sistema.

Manipulação da ontologia: A ontologia foi estudada e manipulada por meio do editor *Protégé* (PROTÉGÉ, 2008) durante a modelagem do sistema de consultas. Os acessos do próprio sistema, porém, se dão através do *framework* de *Web* Semântica *Jena*.

API *Jena*: O *Jena* é um *framework* desenvolvido para construir aplicações de *Web* Semântica, provendo um ambiente de programação para várias linguagens

associadas à *Web* Semântica, entre elas OWL e SPARQL, além de incluir um motor de inferência baseado em regras. Esse *framework* é utilizado tanto pelo EXEHDA-ON para manipular e atualizar a ontologia de acordo com os dados passados dos diversos sensores espalhados pelo ambiente pervasivo, como também pelo próprio sistema de consultas para consultar a ontologia com a linguagem SPARQL.

JDOM: Para *parsing*, manipulação e geração dos documentos XML foi utilizado o JDOM (JDOM, 2008), que é um modelo de representação de documentos XML em objetos Java. A facilidade de manipular documentos e suas estruturas como objetos Java, ocultando praticamente toda a necessidade de se realizar *parsing* com os documentos XML, foi o principal motivador da utilização deste modelo de representação.

5.4.2 Detalhamento da Implementação

O sistema de consultas realiza o *parsing* do documento XML, criando uma representação dos estados de contexto, descritos no documento, em objetos Java. Esses objetos contêm as informações sobre os valores que fazem respeito ao estado de contexto de maneira a facilitar que o subsistema de geração de consultas faça uso desses objetos para geração das consultas, uma para cada estado de contexto. Cada objeto contém referências às propriedades utilizadas no estado de contexto, além de seus valores e operadores respectivos. A classe que define as propriedades é abstrata, e três classes implementam essa classe abstrata, para padronizar o acesso às propriedades e seus valores em um único método. Um diagrama de classes UML foi gerado para a representação do estado de contexto, e a Figura 18 abaixo mostra esse diagrama.

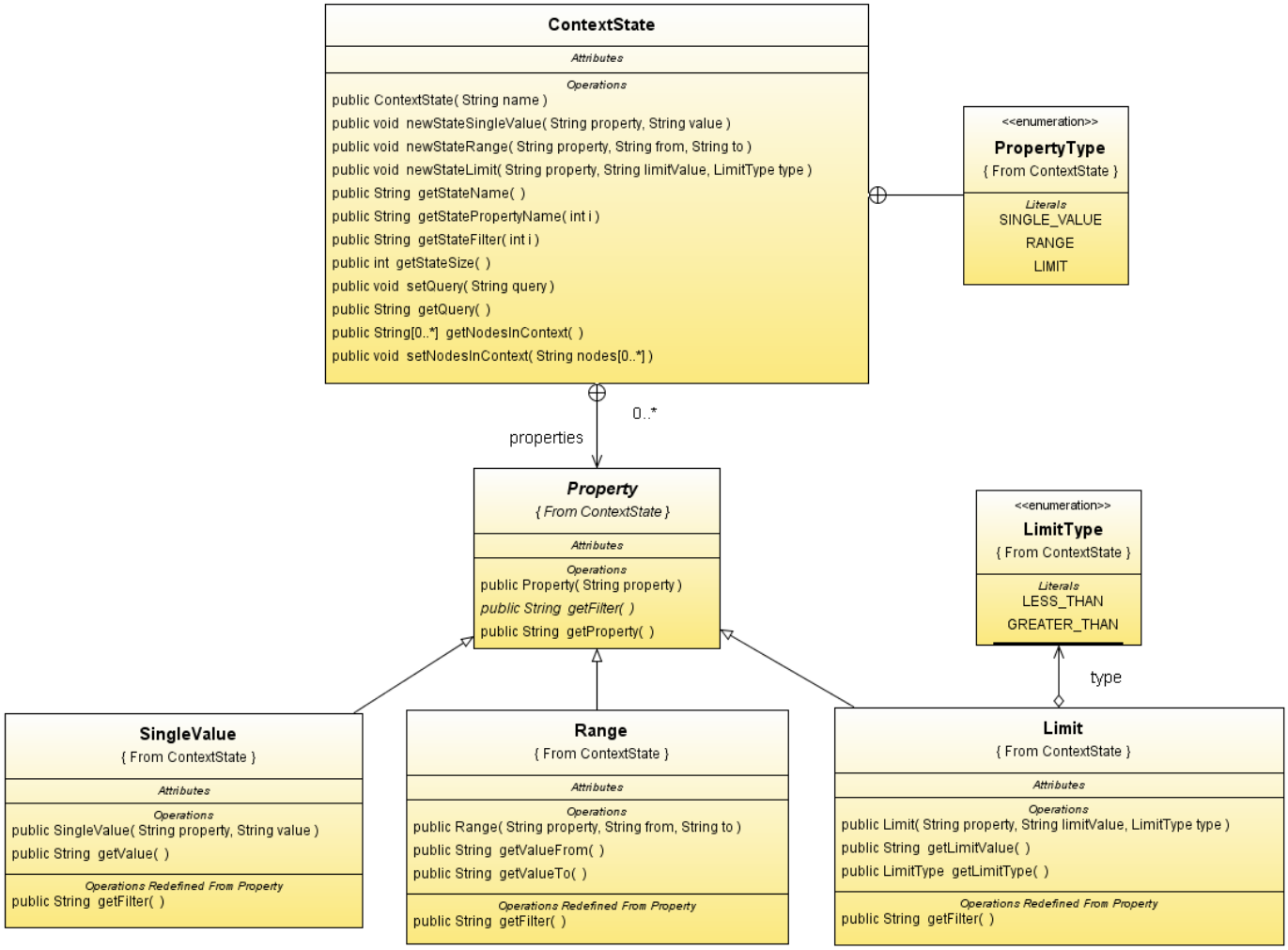

Figura 18 - Diagrama de classes de estados de contexto

O sistema de consultas recebe as requisições de contexto através de uma interface, publicada no GRADEp com o nome *SistCons*. A interface exige que a aplicação também envie como parâmetro uma referência de objeto que será utilizado para receber o retorno do sistema de consultas. Abaixo, a Figura 19 mostra a interface do sistema de consultas, enquanto que a Figura 20 mostra a interface que a aplicação pervasiva deve implementar.

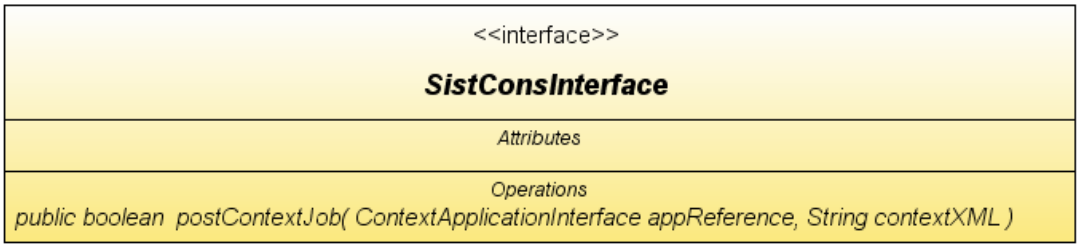

Figura 19 - Interface do sistema de consultas

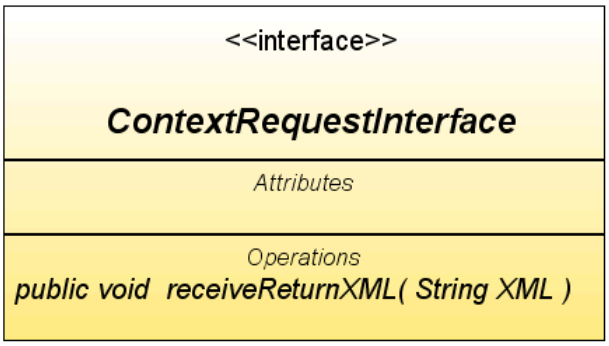

Figura 20 - Interface da aplicação pervasiva

Ao iniciar a execução, o sistema de consultas instancia as *threads* de comunicação e de processamento, e verifica se possui acesso a ontologia, através de chamadas da API *Jena*. Confirmado o acesso, o sistema então exporta a sua interface de acesso, para que esta se torne acessível pelas aplicações pervasivas. A partir desse instante o sistema já pode processar requisições das aplicações. Caso o acesso à ontologia não seja possível, o sistema imprime uma mensagem de erro na saída padrão do GRADEp, explicando o motivo que impossibilitou o acesso, e se desativa.

6 TESTES REALIZADOS

Nessa seção serão mostrados os testes para a verificação da implementação do sistema de consultas. Os testes foram feitos utilizando a biblioteca *JUnit*, integrada a IDE *NetBeans*, e o código de teste foi desenvolvido juntamente com o código do sistema de consultas.

A utilização da biblioteca *JUnit* foi essencial para esses testes, pois essa permite que todos os testes criados para uma determinada unidade de código sejam realizados com apenas um comando da interface da IDE. Assim, quando houve necessidade de alterar a programação de alguma unidade de código devido a algum erro detectado, todos os testes relativos a esse método foram realizados automaticamente, garantindo que a alteração não afetasse o funcionamento da unidade.

Cada módulo do sistema teve suas interfaces (métodos) testadas utilizando dados variados, desde dados corretamente produzidos, referências nulas a objetos como também valores errôneos. Os testes foram realizados em conjunto com o desenvolvimento de cada módulo, permitindo que erros de programação fossem detectados precocemente.

Testes de integração também foram criados, para testar a funcionalidade da chamada de métodos entre os módulos. Novamente, todos os testes envolviam dados corretamente produzidos e dados errôneos.

Por último, também foram realizados cinco testes de funcionamento de todo o sistema de consultas, enviando um pedido para o sistema, aguardando seu processamento, e recebendo o retorno. O teste 1 define três termos simplistas para quantidades de memória física dos nodos. O teste 2 define a necessidade ou não de

limpeza de disco dos nodos. O teste 3 procura por nodos que sejam de um tipo específico de nodo. O teste 4 pesquisa nodos que possuam pouca capacidade de memória ou de disco. E o teste 5 classifica os nodos do ambiente de acordo com a carga de seu processador.

6.1 Preparação dos Testes

O sistema de consultas foi testado utilizando uma ontologia baseada no modelo do EXEHDA-ON. Porém, por fins ilustrativos, as instâncias contidas nela são fictícias, produzidas manualmente utilizando-se o editor *Protégé* mencionado na seção 3.7.

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

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

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

N.I. indica que a informação sobre a propriedade não foi inserida na ontologia.

Por questão de organização, os arquivos XML descritos a seguir não possuem informações de cabeçalho e versão. Além disso, as escolhas dos valores das propriedades em cada teste são arbitrárias, pois o interesse é mostrar que o sistema de consultas está funcionando.

6.2 Teste 1 – Quantidade de Memória

O teste 1 define como estados de contexto três termos para a quantidade total de memória dos nodos. Os nodos são classificados como tento “pouca” memória se possuírem menos de 256MB memória, quantidade “média” se possuírem de 256 a 1024MB, e “bastante” memória se possuírem uma quantidade acima de 1024MB. Abaixo, na Figura 21, são mostrados os documentos XML de pedido de contexto utilizado e de retorno.

```
<CONTEXT>
  <NAME>MemQuant</NAME>
  <STATE>
    <NAME>pouca</NAME>
    <PROPERTY>
      <NAME>TotMemoria</NAME>
      <LESSTHAN>256</LESSTHAN>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>media</NAME>
      <PROPERTY>
        <NAME>TotMemoria</NAME>
        <RANGE>
          <FROM>256</FROM>
          <TO>1024</TO>
        </RANGE>
      </PROPERTY>
    </STATE>
    <STATE>
      <NAME>bastante</NAME>
      <PROPERTY>
        <NAME>TotMemoria</NAME>
          <GREATERTHAN>1024</GREATERTHAN>
        </PROPERTY>
    </STATE>
</CONTEXT>
```

```
<CONTEXT>
  <NAME>MemQuant</NAME>
  <STATE>
    <NAME>pouca</NAME>
    <NODE>Movel_2</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
  <STATE>
    <NAME>media</NAME>
    <NODE>Movel_3</NODE>
    <NODE>Fixo_4</NODE>
    <NODE>Fixo_1</NODE>
  </STATE>
  <STATE>
    <NAME>bastante</NAME>
    <NODE>Fixo_5</NODE>
    <NODE>Fixo_2</NODE>
    <NODE>Fixo_3</NODE>
  </STATE>
</CONTEXT>
```

Figura 21 - Teste 1 - Quantidade de Memória

6.3 Teste 2 – Limpeza de Disco

O teste define como estados de contexto a necessidade ou não de limpeza de disco dos nodos, baseando-se na quantidade de espaço em disco disponível, no caso, menos que 1GB de espaço em disco. Os nodos são classificados como tendo sua limpeza “necessária” se possuírem um valor menor que o definido, ou então “naoNecessaria” se for maior.

Abaixo, na Figura 22, são mostrados os documentos XML de pedido de contexto utilizado e de retorno.

```
<CONTEXT>
  <NAME>LimpezaDisco</NAME>
  <STATE>
    <NAME>necessario</NAME>
    <PROPERTY>
      <NAME>DiscoDisponivel</NAME>
        <LESSTHAN>1</LESSTHAN>
        </PROPERTY>
  </STATE>
  <STATE>
    <NAME>naoNecessario</NAME>
    <PROPERTY>
      <NAME>DiscoDisponivel</NAME>
      <GREATERTHAN>1</GREATERTHAN>
    </PROPERTY>
  </STATE>
</CONTEXT>
```

```
<CONTEXT>
  <NAME>LimpezaDisco</NAME>
  <STATE>
    <NAME>necessaria</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
  <STATE>
    <NAME>naoNecessaria</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
</CONTEXT>
```

Figura 22 - Teste 2 - Limpeza de disco

6.4 Teste 3 – Tipos de Nodos

O teste define como estados de contexto alguns tipos de nodos a serem pesquisados. Cada estado corresponde a um tipo. Os tipos definidos são PC, PDA e Notebook. Abaixo são mostrados, na Figura 23, os documentos XML de pedido de contexto e de retorno.

```
<CONTEXT>
  <NAME>TipoNodos</NAME>
  <STATE>
    <NAME>PC</NAME>
    <PROPERTY>
      <NAME>TipoNodo</NAME>
      <SINGLEVALUE>PC</SINGLEVALUE>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>PDA</NAME>
    <PROPERTY>
      <NAME>TipoNodo</NAME>
       <SINGLEVALUE>PDA</SINGLEVALUE>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>Notebook</NAME>
    <PROPERTY>
      <NAME>TipoNodo</NAME>
      <SINGLEVALUE>Notebook</SINGLEVALUE>
    </PROPERTY>
  </STATE>
</CONTEXT>
```

```
<CONTEXT>
  <NAME>TipoNodos</NAME>
  <STATE>
    <NAME>PC</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Fixo_2</NODE>
    <NODE>Fixo_1</NODE>
  </STATE>
  <STATE>
    <NAME>PDA</NAME>
    <NODE>Movel_1</NODE>
  </STATE>
  <STATE>
    <NAME>Notebook</NAME>
    <NODE>Movel_3</NODE>
    <NODE>Movel_2</NODE>
  </STATE>
</CONTEXT>
```

Figura 23 - Teste 3 – Tipos de nodos

6.5 Teste 4 – Atualização dos Nodos

O teste define uma situação hipotética, na qual alguns nodos da célula apresentam configurações de *hardware* antigas. Como estados de contexto são definidos os nomes de alguns componentes de nodos que precisam ser atualizados. Os nodos são classificados como tento pouca memória se possuírem menos de 256MB memória, ou como tento uma capacidade de disco pequena se possuírem menos de 40GB de tamanho total de disco. Abaixo são mostrados na Figura 24 os documentos XML de pedido de contexto utilizado e de retorno.

```
<CONTEXT>
  <NAME>Atualizacao</NAME>
  <STATE>
    <NAME>Memoria</NAME>
    <PROPERTY>
      <NAME>TotMemoria</NAME>
       <LESSTHAN>256</LESSTHAN>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>Disco</NAME>
    <PROPERTY>
      <NAME>TotDisco</NAME>
      <LESSTHAN>40</LESSTHAN>
    </PROPERTY>
  </STATE>
</CONTEXT>
```

```
<CONTEXT>
  <NAME>Atualizacao</NAME>
  <STATE>
    <NAME>Memoria</NAME>
    <NODE>Movel_2</NODE>
    <NODE>Movel_1</NODE>
  <STATE>
    <NAME>Disco</NAME>
    <NODE>Fixo_2</NODE>
    <NODE>Fixo_3</NODE>
    <NODE>Fixo_1</NODE>
    <NODE>Movel_1</NODE>
  </STATE>
</CONTEXT>
```

Figura 24 - Teste 4 – Atualização dos Nodos

6.6 Teste 5 – Carga de processamento dos nodos

O teste define como estados de contexto a carga de processamento dos nodos, levando em consideração a quantidade de memória ocupada e a carga de processamento dos nodos. Os nodos são classificados como “livres” se a ocupação do processador estiver a 30%, “médio” se possuírem carga de 30% a 70%, e “ocupado” se possuírem carga acima de 70%. Abaixo são mostrados os documentos XML de pedido de contexto utilizado e de retorno.

```
<CONTEXT>
  <NAME>Ocupacao</NAME>
  <STATE>
    <NAME>livre</NAME>
      <PROPERTY>
      <NAME>CargaCPU</NAME>
      <LESSTHAN>30</LESSTHAN>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>medio</NAME>
    </PROPERTY>
    <PROPERTY>
      <NAME>CargaCPU</NAME>
      <RANGE>
        <FROM>30</FROM>
        <TO>70</TO>
      </RANGE>
    </PROPERTY>
  </STATE>
  <STATE>
    <NAME>ocupado</NAME>
    <PROPERTY>
      <NAME>CargaCPU</NAME>
      <GREATERTHAN>70</GREATERTHAN>
    </PROPERTY>
  </STATE>
</CONTEXT>
```

```
<CONTEXT>
  <NAME>Ocupacao</NAME>
  <STATE>
    <NAME>livre</NAME>
    <NODE>Fixo_4</NODE>
    <NODE>Fixo_5</NODE>
    <NODE>Movel_3</NODE>
  </STATE>
  <STATE>
    <NAME>medio</NAME>
    <NODE>Fixo_1</NODE>
  </STATE>
  <STATE>
    <NAME>ocupado</NAME>
    <NODE>Fixo_2</NODE>
    <NODE>Fixo_3</NODE>
  </STATE>
```

Figura 25 - Teste 5 – Carga de processamento dos nodos

Embora o conjunto de propriedades seja limitado para permitir consultas a informações de contexto mais complexas, os testes mostram que o sistema de consultas funciona como esperado.

7 CONCLUSÃO

Esse trabalho propôs a criação de um sistema de consultas a informações de contexto em uma ontologia, objetivando apoiar a adaptação de aplicações pervasivas, dentro do ambiente pervasivo provido pelo *middleware* GRADEp.

A Computação Pervasiva possui conceitos interessantes para a pesquisa na computação. A possibilidade de dispositivos e aplicações pervasivas serem utilizados como ferramentas para atendimento as requisições dos usuários, integrando-se ao ambiente físico deste de forma transparente, gera um campo de pesquisa fértil, com dificuldades que instigam sua superação. No futuro da computação, com certeza, a Computação Pervasiva estará presente, assim como no dia-a-dia dos usuários.

Sobre a Sensibilidade ao Contexto, a utilização de ontologias permite que as informações de contexto sejam expressas de forma mais flexível e que sua capacidade de exploração seja aumentada, pois são utilizadas todas as características de expressão de conhecimento possíveis.

O sistema de consultas permite o acesso a essas informações de uma maneira programática, facilitando o desenvolvimento das aplicações, ao esconder os aspectos de programação envolvidos no acesso à ontologia. Embora o sistema consista de um protótipo, não estando, portanto, totalmente pronto para aplicação em um ambiente pervasivo real, todos os seus conceitos e modelagem desenvolvida podem ser usados em versões futuras, aptas para uso geral.

Os testes realizados com a biblioteca *JUnit*, como foram todos concluídos com sucesso, mostram que a possibilidade de ocorrência de erros na implementação é reduzida. Os testes mostram também que o contexto requisitado

pelas aplicações pode ser obtido da ontologia, e ser utilizado para qualquer fim necessário à aplicação.

A utilização do GRADEp como *middleware* para ambientes pervasivos é interessante, pois este possui uma base de funcionamento estável. Futuramente, outros trabalhos podem estudar e melhorar aspectos do ambiente, avançando o conhecimento sobre a Computação Pervasiva.

O sistema EXEHDA-ON mostrou que a utilização de ontologias dentro do ambiente é viável, e que pesquisas nesta área podem render mais conhecimento sobre como superar o desafio da Sensibilidade ao Contexto na Computação Pervasiva.

Finalmente, acredita-se que os objetivos desse trabalho foram atingidos, uma vez que a modelagem foi desenvolvida e seu o protótipo apresentou funcionamento satisfatório com relação aos requisitos definidos.

7.1 Trabalhos Futuros

Como proposta de trabalhos futuros, é sugerida a adição de uma funcionalidade de monitoramento de contexto ao sistema de consultas, para que as aplicações pervasivas possam monitorar a alteração dos estados de contexto no transcorrer do tempo.

Outra proposta sugerida é a expansão da ontologia e, por conseqüência, do sistema de consultas, para permitir que outros tipos de contexto sejam expressos, como perfis de usuário e seus interesses pessoais, ou estados físicos do ambiente. Isso permitirá que uma variedade maior de contextos possa ser expresso, aumentando também o seu uso pelas aplicações.

REFERÊNCIAS

CALVI, Camilo Zardo; PESSOA, Rodrigo Mantovaneli; PEREIRA FILHO, José Gonçalves. Um Interpretador de Contexto para Plataformas de Serviço Context-Aware. In **Anais do XXV Congresso da Sociedade Brasileira de Computação (CSBC'05)** e **XXXII Seminário Integrado de Software e Hardware (SEMISH'05)**, São Leopoldo, Brasil, Julho 2005.

CHEN, Guanling; KOTZ, David. **A Survey of Context-Aware Mobile Computing Research. Technical Report**. Department of Computer, Dartmouth College. 2000.

CHEN, Guanling; KOTZ, David. **Solar: a pervasive-computing infrastructure for context-aware mobile applications**. Technical Report, Dept. of Computer Science, Dartmouth College, 2002.

CHEN, Harry; PERICH, Filip; FININ, Tim; JOSHI, Anupam. **SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications**. Department of Computer Science & Electrical Engineering. University of Maryland, Baltimore County. 2004.

CHEN, Harry; FININ, Tim; JOSHI, Anupam. **An Ontology for Context-Aware Pervasive Computing Environments**. Department of Computer Science & Electrical Engineering. University of Maryland, Baltimore County. 2004.

CHEN, Harry. **An Intelligent Broker Architecture for Pervasive Context-Aware Systems**. Dissertação de Doutorado, University of Maryland, Baltimore, 2004.

FAHY, Patrick; CLARKE, Siobhan. CASS – Middleware for Mobile Context-Aware Applications. **International Conference on Mobile Systems, Applications and Services - MobiSys**, Boston, Massachusetts, EUA, 6-9 Junho 2004.

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

GRUBER, Thomas R. A Translation Approach to Portable Ontology Specifications, **Knowledge Acquisition**, v5, p.199-220, 1993.

GRUNINGER, M. Designing and Evaluating Generic Ontologies. In: EUROPEAN CONFERENCE OF ARTIFICIAL INTELLIGENCE, 12., 1996. **Anais do...** 1995. (IJCAI Proceedings 1995).

GU, Tao; PUNG, Hung Keng, ZHANG, Da Qing. A Middleware for Building Context-Aware Mobile Services. **Proceedings of IEEE Vehicular Technology Conference**, Milan, Italy, Maio 2004.

GUARINO, N. Formal Ontology and Information Systems. In: FOIS 98, 1998, Trento, Italy. **Anais do...** FOIS 98, 1998, Trento, Italy. p.3–15.

HARMELEN, Frank van; PATEL-SCHNEIDER, Peter F.; HORROCKS, Ian. **Reference description of the DAML+OIL (March 2001) ontology markup language.** Disponível em <www.daml.org/2001/03/reference.html>. Acesso em 30 de abril de 2008.

HAROLD, Elliotte Rusty ; MEANS, W. Scott. **XML in a Nutshell**, 2ª ed. Sebastopol, Califórnia, EUA: O'Reilly. 2002. 830 pg.

HENRICKSEN, Karen, INDULSKA, Jadwiga. Developing Context-Aware Pervasive Computing Applications: Models and Approach. **Pervasive and Mobile Computing**, In Press, Elsevier, 2005.

Java Resources for Developers. Disponível em <http://java.sun.com>. Acesso em 28 de abril de 2008.

JavaDoc Tool Homepage. Disponível em <http://java.sun.com/j2se/javadoc/>. Acesso em 30 de abril de 2008.

JDOM Project, The. Disponível em <http:// www.jdom.org>. Acesso em 2 de maio de 2008.

JUnit Resources for Test Driven Development. Disponível em <http://www.junit.org/>. Acesso em 15 de maio de 2008.

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

LYYTINEN, Kalle; YOO, Youngjin. Issues and Challenges in Ubiquitous Computing. **Communications of the ACM**, vol. 45, no. 12, p. 62-65.

MANOLA, Frank; MILLER, Eric. **RDF Primer**. W3C Recommendation 10 February 2004. Disponível em <http://www.w3.org/TR/rdf-primer/#intro>. Acesso em: 4 de maio de 2008.

NetBeans IDE. Disponível em < www.netbeans.com>. Acesso em: 28 de abril de 2008.

PREKOP, Paul; BURNETT, Mark. Activities, Context and Ubiquitous Computing. **Computer Communications,** Special Issue on Ubiquitous Computing, 2003.

PRUD'HOMMEAUX, Eric; SEABORNE, Andy. SPARQL Query Language for RDF. W3C Recommendation em 15 de janeiro de 2008. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/> Acesso em: 8 de abril de 2008.

Protégé; Ontology Editor and Knowledge Acquisition System, The. Disponível em <http://protege.stanford.edu/>.

RANGANATHAN, Anand; MCGRATH, Robert E.; CAMPBELL, Roy; MICKUNAS, Dennis M.. **Ontologies in a Pervasive Computing Environment**. Workshop on Ontologies in Distributed Systems, IJCAI 2003.

SATYANARAYANAN, M. **Pervasive Computing: Vision and Challenges**. IEEE Personal Communications, New York, v.8, n.4, p.10–17, 2001.

SEABORNE, Andy.RDQL - A Query Language for RDF. W3C Member Submission em 9 de janeiro de 2004. Disponível em <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>. Acesso em 29 de abril de 2008.

SMITH, Michael K.; WELTY, Chris; MCGUINNESS, Deborah L. OWL Web Ontology Language Guide. W3C Recommendation em 10 de fevereiro de 2004. Disponível em: < http://www.w3.org/TR/owl-guide/> Acesso em: 9 de abril de 2008.

SU, Xiaomeng; ILEBREKKE, Lars. A Comparative Study of Ontology Languages and Tools. In: Advanced Information Systems Engineering, 14th International Conference, 2002. **Anais do...** Springer-Verlag, 2002. p.761–765.

Subversion Project, The. Disponível em <http://subversion.tigris.org/>. Acesso em: 3 de junho de 2008.

UML Projet, The NetBeans. Disponível em <http://uml.netbeans.org/>. Acesso em: 2 de maio de 2008.

VIVAN, Giulian Gonçalves. **Uso de Padrões de Projeto no Desenvolvimento de Aplicações Baseadas em Mobilidade de Código na Computação Pervasiva**. 2007. 102f. Trabalho acadêmico – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas, Pelotas.

WEISER, Mark; GOLD, Rich; BROWN, John. **The origins of ubiquitous computing research at PARC in the late 1980s**. IBM System Journal: Pervasive Computing, v.38, n.4, 1999, p.693-696. Disponível em: <http://www.research.ibm.com/journal/sj/384/weiser.html>.

WASP. Web-services: the cement for mobile, context-aware systems. Projeto WASP – Web Architectures for Service Platform, University of Twente, Holanda, 2004. Disponível em <http://www.freeband.nl/kennisimpuls/projecten/wasp/ENindex.html>.

YAMIN, Adenauer. **Arquitetura para um Ambiente de Grade Computacional Direcionado as Aplicações Distribuídas Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. 194f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

YAMIN, Adenauer Corrêa; AUGUSTIN, Iara; SILVA, Luciano Cavalheiro da; REAL, Rodrigo Araújo; BARBOSA, Jorge Luis Victória; GEYER, Cláudio Fernando Resin. EXEHDA - an adaptive middleware for the pervasive computing scenery. **8th Brazilian Symposium on Programming Languages**. Rio de Janeiro. 2004.

APÊNDICE A

O presente apêndice contém o documento DTD usado para validar os documentos XML recebidos pelo sistema de consultas.

```
<!ELEMENT CONTEXTXML (VERSION, CONTEXT)>
<!ELEMENT VERSION (#PCDATA)>
<!ELEMENT CONTEXT (NAME, STATE+)>

<!ELEMENT STATE (NAME, PROPERTY+)>

<!ELEMENT PROPERTY (NAME, (SINGLEVALUE | RANGE | GREATERTHAN | LESSTHAN))>

<!ELEMENT NAME (#PCDATA)>

<!ELEMENT SINGLEVALUE (#PCDATA)>

<!ELEMENT RANGE (FROM, TO)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT TO (#PCDATA)>

<!ELEMENT GREATERTHAN (#PCDATA)>
<!ELEMENT LESSTHAN (#PCDATA)>
```