UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO DE SISTEMAS DE COMPUTAÇÃO PARA WEB

ESTUDO SOBRE ONTOLOGIAS APLICADAS À AREA DE COMPUTAÇÃO MÓVEL E PERVASIVA

MONOGRAFIA DE ESPECIALIZAÇÃO

Cleber Zalamena Braun

Santa Maria, RS, Brasil
2007

ESTUDO SOBRE ONTOLOGIAS APLICADAS À AREA DE COMPUTAÇÃO MÓVEL E PERVASIVA

por

Cleber Zalamena Braun

Monografia apresentada ao Curso de Especialização do Programa de Pós-Graduação de Sistemas de Computação para Web, Área de Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Especialista em Ciência da Computação.

Orientadora: Profª. Iara Augustin

Santa Maria, RS, Brasil
2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação de Sistemas de Computação para Web

A Comissão Examinadora, abaixo assinada,
aprova a Monografia de Especialização

ESTUDO SOBRE ONTOLOGIAS APLICADAS À AREA DE COMPUTAÇÃO MÓVEL E PERVASIVA

elaborada por
Cleber Zalamena Braun

como requisito parcial para obtenção do grau de
Especialista em Ciência da Computação

COMISÃO EXAMINADORA:

Iara Augustin, Dra.
(Presidente/Orientadora)

Nara Augustin Gehrke, Dra. (UFSM)

Giovani Rubert Librelotto, **Dr.** (UNIFRA)

Santa Maria, 2 de março de 2007.

RESUMO

Monografia de Especialização
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

ESTUDO SOBRE ONTOLOGIAS APLICADAS À AREA DE COMPUTAÇÃO MÓVEL E PERVASIVA

AUTOR: CLEBER ZALAMENA BRAUN
ORIENTADORA: IARA AUGUSTIN
Data e Local da Defesa: Santa Maria, 28 de janeiro de 2007.

A partir do crescimento de informações disponíveis na Web e também dos inúmeros sistemas de padrões adotados para o desenvolvimento de aplicações, surge a necessidade de uma semântica para facilitar descrições nesses sistemas. Uma forma de uniformizar e facilitar a escrita é adotando uma *ontologia* para a descrição e reutilização de termos e seus relacionamentos. Elas são não só modelos de dados, como também as estruturas de conceitos das bases de conhecimento. As ontologias estão sendo empregadas também em ambientes da *Computação Pervasiva*, justamente com este propósito. A Computação Pervasiva representa um novo paradigma computacional que permite ao usuário o acesso e compartilhamento de informações computacionais a partir de qualquer lugar, todo o tempo. A aplicações móveis são desenvolvidas pelas características implícitas dessa tecnologia, que envolvem: autonomia, mobilidade e proatividade de sistemas. As Ontologias SOUPA e DAML+OIL apresentam *arquiteturas*, como o COBRA e o GAIA respectivamente, para aplicações destinadas a ambientes pervasivos, com o objetivo de facilitar o desenvolvimento de sistemas. O presente trabalho teve por finalidade, estudar essas duas ontologias, como também as arquiteturas anteriormente citadas, descrevendo e comparando suas principais características neste novo ambiente que representa o futuro da computação atual.

Palavras Chave: Computação Pervasiva, Ontologias, arquiteturas.

ABSTRACT

Monografia de Especialização
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

STUDY ABOUT APPLIED ONTOLOGIES TO THE MOBILE AND PERVASIVE COMPUTATION AREA

AUTOR: CLEBER ZALAMENA BRAUN
ORIENTADORA: IARA AUGUSTIN
Data e Local da Defesa: Santa Maria, 28 de janeiro de 2007.

Starting from the increase of available information at the Web and also from the countless standard systems adopted for the development of the applications, arises a necessity of a semantics to facilitate descriptions in these systems. A way to standardize and facilitate the writing is adopting an *ontology* for the description and reuse of terms and their relationship. They are not only data models concept structures of the knowledge basis. The ontologies are also being used in Pervasive Computation background, just for this purpose. The *Pervasive Computation* symbolizes a new computational paradigm that permits to the user the access and share of computational information from any place, all the time. The mobile applications are developed by the implicit characteristics of this technology that involve: autonomy, mobility and systems' proactivity. The SOUPA and DAML+OIL present *architectures*, like the COBRA and the GAIA, respectively, for applications destined to pervasive environment with the objective to facilitate the systems' development. The purpose of this work was to study these two ontologies, as well as the architectures mentioned before, by describing and comparing their main characteristics in this new environment which represents the future of present computation.

Key-words: Pervasive Computation, Ontologies, architectures.

LISTA DE ABREVIATURAS

ACL	Agent Communication Language
CoRE	Context Reasoning Engine Standard
CORBA	Common Object Request Broker Architeture
COBRA	Context Broker Architecture
DAML	Darpa Agent Markup Language
DARPA	Defence Advanced Research Projects Agency
DTD	Document Type Definition
FOAF	Friend-of-a-Friend
FIPA	Foundation for Intelligent Physical Agents
GPS	Global Positioning System
HTML	HyperText Markup Language
IBM	International Business Machines
MoPP	Module for Privacy Protection
OIL	Ontology Inference Layer
OWL	Ontology Web Language
OCLC	Online Computer Library Center
PDA	Personal Digital Assistant
PARC	Palo Alto Research Center
RDF	Resource Description Framework
SOUPA	Ontology for Ubiquitous and Pervasive Aplications
SGML	Standard Generalized Markup Language
URL	Uniform Resouce Locator
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

SUMÁRIO

INTRODUÇÃO

Diante da crescente expansão da informática e de sua evidente necessidade no cotidiano da sociedade contemporânea, seja no lazer, seja no comércio, é necessária uma padronização de termos usados tanto no desenvolvimento de aplicações, como nas buscas por informações na Internet. Essa padronização se torna favorável não somente aos usuários de alto grau de conhecimento, mas principalmente aos usuários iniciantes, que, de certa forma, são inseridos no complexo mundo virtual. A quantidade e a diversidade de dados disponíveis na Web acrescentam dificuldades na busca por uma determinada informação.

Outra dificuldade é a extração de informações automatizadas, pelo fato de elas exigirem a interpretação humana na sua seleção. Por outro lado, através do computador, estas informações podem ser organizadas e processadas em uma hierarquia rígida, deixando aos usuários a ligação de pequenas unidades de informações aleatórias.

Com base nessa constatação, é de fundamental importância a relação de disposição de conceitos e seus relacionamentos dentro de um universo, de maneira onde não somente a *sintaxe* determine uma busca, mas sim toda uma semântica.

Na Internet, as novas informações inseridas rapidamente se tornam outro fator agravante, devido ao vasto conteúdo a ser pesquisado, o que, conseqüentemente, dificulta a tarefa dos motores de busca.

No campo de desenvolvimento de *softwares*, por exemplo, fica evidente a dificuldade de um analista ou programador, na análise e implementação de um projeto respectivamente, sem a adoção de uma semântica no processo.

Porém, somente ter o conteúdo semântico pouco ajuda. É preciso utilizá-lo de maneira sistemática para a manipulação, visualização e transformação das informações (CARNEIRO, 2003).

Uma das formas de padronizar, uniformizar e facilitar a escrita é o uso de *ontologias,* constituindo estas toda uma semântica no conteúdo presente, a fim de facilitar a realização das tarefas por parte dos usuários. Elas fornecem um vocabulário comum a uma área e definem o significado dos termos e dos relacionamentos entre eles.

Nos últimos tempos uma área que vem se destacando em computação e que preconiza a centralização nas tarefas e atividades dos usuários é a Computação Pervasiva (CHEN, 2006). Nesta, por exemplo, muitos sistemas e arquiteturas obtiveram sucesso em alguns

aspectos, porém eles oferecem um fraco suporte no que se refere a compartilhamento dos dados, pelo fato de eles não terem sido desenvolvidos mediante a adoção de uma ontologia comum. Esses sistemas necessitam de várias regras entre os programas para o compartilhamento de informações, pois eles não podem facilitar o conhecimento compartilhado em um ambiente aberto e dinâmico. Com o emprego de ontologias nessas aplicações, problemas de compartilhamento de informações podem ser simplificados.

O presente trabalho pretende de uma forma geral, estudar as ontologias SOUPA e DAML+OIL, como também as arquiteturas COBRA e GAIA, que estão sendo usadas para facilitar o desenvolvimento e manipulação das informações na Computação Móvel e Pervasiva.

Os próximos capítulos organizam-se da seguinte forma: o capítulo 1 aborda os conceitos e características das Ontologias e da Computação Pervasiva de uma forma geral, por fim, o capítulo 2 aborda o relacionamento das Ontologias com a Computação Pervasiva, apresentando projetos e arquiteturas desenvolvidas.

1 ONTOLOGIAS: CONCEITOS E FERRAMENTAS

Este capítulo aborda os conceitos e as características das ontologias aplicadas em diferentes ambientes, como a Web e a Computação Pervasiva.

1.1 Ontologia

Antes de se estudar mais especificadamente as ontologias na Computação Móvel e Pervasiva, é relevante compreender-se o sentido literal da palavra.

> No Dicionário Aurélio (1996, p. 1225), a palavra Ontologia tem o seguinte conceito: "[De ont(o)- + -log (o)- + -ia.] S. f. Filos. Parte da filosofia que trata do ser enquanto ser, i. e., do ser concebido como tendo uma natureza comum que é inerente a todos e a cada um dos seres".

Ontologia é uma palavra grega introduzida na filosofia no século XIX por filósofos alemães. Como disciplina na filosofia, uma determinada ontologia tem por finalidade prover categorias de sistemas que agreguem uma certa visão do mundo (JÚNIOR, 2003).

Segundo Gruber (1993), trata-se de uma descrição da semântica de uma realidade. Basicamente, ontologia é a especificação explícita e formal de uma conceitualização. Uma ontologia de um dado domínio de conhecimento oferece um vocabulário terminológico de referência que pode ser utilizado na resolução de conflitos semânticos entre conceitos e termos utilizados nas fontes de dados distribuídas.

No campo da computação, as ontologias têm sido aplicadas na inteligência artificial para facilitar o reuso e compartilhamento de informações. Atualmente, as ontologias estão se expandindo em outras áreas:

- Integração inteligente de informação;
- Comércio eletrônico;
- Sistemas cooperativos;
- Produtos de software baseados em agentes.

Diante do contexto de que a ontologia deve fazer uma especificação formal de uma área de conhecimento, foram definidos cinco componentes para essa formalização (Gruber, 1995):

- Conceitos: uma tarefa, uma função, uma estratégia, etc. Pode representar qualquer coisa em um domínio;
- Relações: são as interações entre os conceitos no domínio;
- Funções: trata-se de um caso especial de relações;
- Axiomas: são sentenças sempre verdadeiras;
- Instâncias: representam os elementos do domínio.

As ontologias são compostas também por *Taxonomias* e *Thesauros*. *Taxonomias* representam um sistema de classificação que organizam conceitos e entidades. Elas descrevem os relacionamentos hierárquicos entre os conceitos, identificando os membros das classes e subclasses. *Thesauros* são uma rede de termos inter-relacionados através de um conjunto de relações semânticas dentro de um domínio particular. Eles estendem as taxonomias, tornando-as mais complexas para a descrição de um domínio (LIBRELOTTO, 2005).

A Figura 1 ilustra um exemplo de uma ontologia para o planeta Terra.

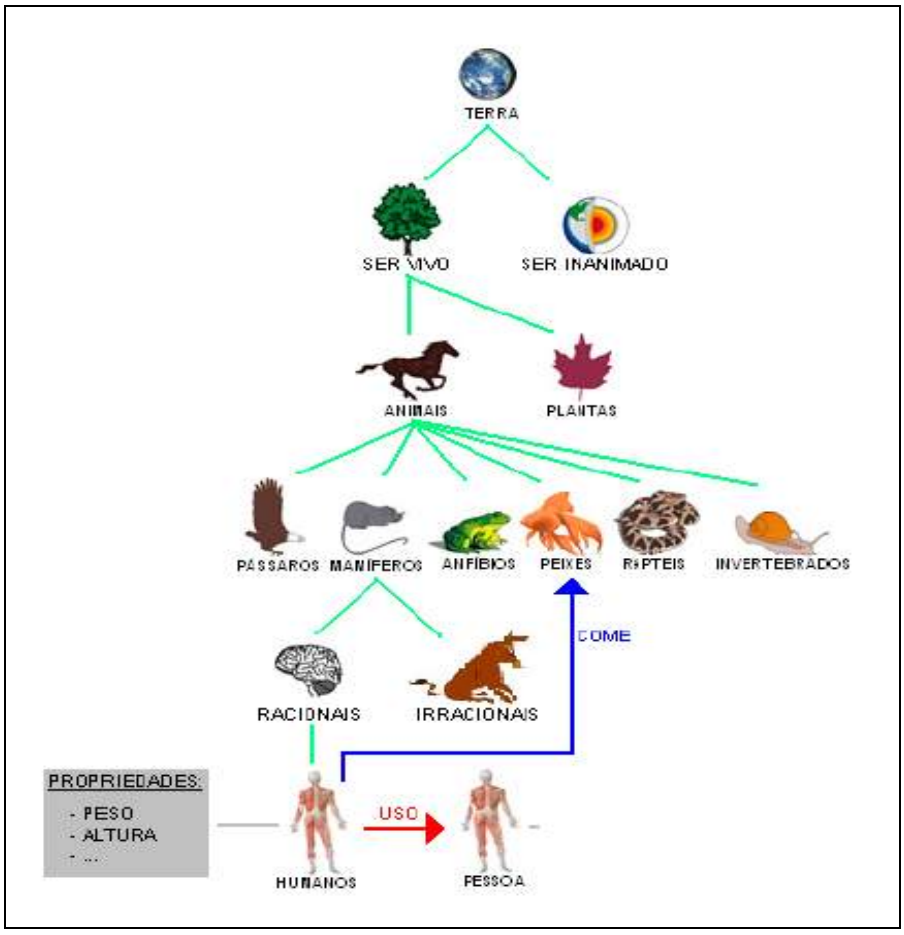

Figura 1: Uma ontologia para o planeta Terra (LIBRELOTTO, 2005)

Na Figura 1, a *taxonomia* representa a hierarquia dos conceitos encontrados no planeta Terra. As relações são as ligações entre os conceitos, definindo as classes e sub-classes da hierarquia. Além das relações definidas, o *thesauri* pode ser identificado na relação do tipo "USO" (apresentando um uso de uma classe).

1.1.1 Tipos de Ontologias

Em uma ontologia, definições estão associadas aos nomes de entidades no universo de discurso (conceitos, relações, funções).

Ontologias comuns descrevem tarefas para um conjunto de agentes de modo que estes possam se comunicar sobre um domínio do discurso sem necessariamente operar em uma teoria global compartilhada (NOVELLO, 2003). Segundo Gruber (1993), um agente pertence a uma ontologia se suas ações observáveis são consistentes com as definições da ontologia.

No desenvolvimento de uma ontologia, devem ser observados os seguintes princípios básicos:

- *Clareza e objetividade*. Os termos devem estar acompanhados por definições objetivas e também por documentação em linguagem natural;
- *Completeza*. Na representação de um termo, deve-se atribuir definições necessárias e suficientes na sua expressão;
- *Coerência*. Para derivar inferências que sejam consistentes com as definições;
- *Extensibilidade monotônica*. Para permitir a inclusão de novos termos sem revisão das definições existentes;
- *Compromisso ontológico*. Para permitir que as especializações e instanciações da ontologia sejam definidas com liberdade;
- *Principio da distinção ontológica*. Não permitir a superposição de conceitos das classes que compõem a ontologia;
- *Diversificação das hierarquias*. Aproveitar os mecanismos de herança múltipla;
- *Modularidade*. Minimizar o acoplamento entre os módulos;
- *Minimização da distância semântica entre conceitos similares*. Agrupar e representar utilizando as mesmas primitivas;
- *Padronização*. Usar sempre que possível, nomes padrões.

Grandes partes das ontologias atuais são desenvolvidas por empresas particulares em benefício próprio e não as disponibilizam. Elas podem ser descritas em diferentes níveis de abstração, dependendo de onde serão aplicadas.

Elas podem ser classificadas em três diferentes níveis (NOVELLO, 2003):

- *Ontologia de Nível de Topo*: descreve os conceitos gerais, independentes do domínio específico;
- *Ontologia de Domínio e de Tarefa*: descreve os conceitos de um determinado domínio, ou de uma tarefa ou atividade por especialização dos termos definidos na ontologia de nível de topo;
- *Ontologia de Aplicação*: descreve na maioria das vezes, os conceitos da relação entre as ontologias de domínio com as ontologias de tarefas particulares.

1.1.2 Vantagens do emprego de Ontologias

Com a utilização de ontologias, é possível definir uma infra-estrutura na integração de sistemas inteligentes no nível do conhecimento; com isso, têm-se grandes vantagens, como (NOVELLO, 2003):

- *Colaboração*: possibilitam o compartilhamento do conhecimento entre os membros interdisciplinares de uma equipe;
- *Interoperação*: facilitam a integração da informação, especialmente em aplicações distribuídas;
- *Modelagem*: as ontologias são representadas por blocos estruturados que podem ser reusáveis na modelagem de sistemas no nível de conhecimento;
- *Informação*: podem ser usadas como fonte de consulta e de referência de domínio.

1.1.3 Ontologia e a Web Semântica

Com o sucesso atual da Web e conseqüentemente, com a variedade e quantidade de informações disponíveis, a busca por determinada informação torna-se cada vez mais

complexa. Isso se deve ao fato de que os motores de busca são muito dependentes ainda da interpretação humana em uma consulta referente a algum conteúdo presente na Web. Esses motores ainda trabalham de forma separada, dificultando o relacionamento entre informações referentes a um mesmo assunto.

A *Web Semântica* representa a evolução da Web atual. Enquanto a Web tradicional foi desenvolvida para ser entendida apenas pelos usuários, a Web Semântica está sendo projetada para ser compreendida pelas máquinas, na forma de agentes computacionais, que são capazes de operar eficientemente sobre as informações, podendo entender seus significados **(**DZIEKANIAK, 2006). Ela fornece uma estrutura semântica ao conteúdo da Web, criando um ambiente onde agentes de software e usuários possam trabalhar de forma cooperativa.

Um processo importante na criação dessa semântica é a adoção de uma linguagem para se definir uma ontologia, com o objetivo de estabelecer um padrão comum entre informações adicionadas a Web.

Existem três fatores de fundamental importância para a escolha de uma ontologia adequada. São eles (CARNEIRO, 2003):

- *Complexidade Computacional*: preocupação freqüente em pesquisas. Pode-se verificar bom comportamento das ferramentas de raciocínio e inferência na prática e em grande parte dos casos reais;
- *Complexidade Técnica*: refere-se à implementação de ferramentas e aplicações. Depois da adoção dos padrões *World Wide Web Consortium* (W3C), teoricamente tornou-se inexistente;
- *Complexidade Conceitual*: está na dificuldade de compreensão e utilização de uma linguagem de ontologia, por parte dos usuários comuns. Uma preocupação dos desenvolvedores é a implementação de uma linguagem completa e simples no seu entendimento para que não haja tal dificuldade.

Na atualidade, algumas comunidades científicas internacionais estão desenvolvendo projetos para a criação de ferramentas capazes de, a partir do computador, interpretar a linguagem humana. Alguns desses projetos são (DZIEKANIAK, 2004):

- *Projeto INDEXA*. Desenvolvido no Brasil pela Escola de Ciência da Informação da Universidade Federal de Minas Gerais, tem por objetivo criar uma ferramenta que analise o documento para auxiliar o desenvolvimento de *websites*;

- *Projeto SCORPION*. Desenvolvido nos Estados Unidos pela *Online Computer Library Center* (OCLC), tem como objetivo a exploração da indexação e a catalogação de recursos eletrônicos.

Como o enfoque do trabalho não é tratar diretamente questões ligadas a Web Semântica, a seguir descreveremos de uma forma sucinta, apenas três de suas principais linguagens. São elas: o XML, o RDF e a OWL.

1.1.3.1 XML

Desenvolvida sob a orientação do W3C e derivada da *Standard Generalized Markup Language* (SGML), o *Extensible Markup Language* (XML) é uma linguagem de anotação para estruturação e marcação de documentos, que permite a organização e armazenamento de informações estruturadas e semi-estruturadas (LIBRELOTTO, 2005).

Para uma melhor compreensão, podemos compará-la com a *HyperText Markup Language* (HTML), também derivada do SGML, pelo simples fato de ser uma linguagem mais conhecida.

O XML supre as deficiências do HTML, pois permite que o usuário crie marcações, com isso proporciona uma maior descrição dos recursos. Sua formatação é organizada separadamente de sua estrutura, um dos principais problemas encontrados no HTML. Enquanto os elementos e atributos da HTML são pré-definidos para a criação de hipertextos, o XML é extensível pelo fato de permitir a criação de diversos elementos e atributos.

XML é a representação textual do dado. Seu componente básico é o *elemento* (element), que nada mais é, o texto entre *tags* (delimitadores). O atributo é definido como um par (nome, atributo) e pode ser associado a elementos.

Um documento XML é uma estrutura lógica sob a forma de uma hierarquia em árvore, onde um elemento é definido como a raiz do documento XML.

Sendo usada para criar linguagens de marcação, o XML necessita de um conjunto de medidas para garantir a interoperabilidade de aplicações. Para isso, torna-se recomendado que o documento possua um *Document Type Definition* (DTD) na definição da estrutura.

Através de validadores, é possível saber se um documento XML foi criado corretamente, associando-o com um DTD apropriado.

A figura 2 apresenta um documento XML que estrutura uma mensagem de *e-mail*:

```
<?xml version="1.0"?>
<!DOCTYPE mail system "http://infowest.com/DTDS/email.dtd">
<email>
        <de>Autor</de>
        <para>Alguém</para>
                <data>Quarta-feira - 2 de dezembro de 2005</data>
                <assunto>Introdução ao XML</assunto>
        <texto>Comentários:
                <p alinhamento="justificado">Obrigado pela leitura desta introdução </p>
                <p>Esperamos que seja proveitosa</p>
        </texto>
</email>
```

Figura 2: Exemplo de um documento XML (VIEIRA, 2005)

Na figura anterior, temos os dados referentes a um *e-mail* em um documento de um arquivo XML. Estão identificados pelos marcadores ou etiquetas (<> e </>) os elementos do arquivo que têm por objetivo encapsular determinada informação. Um elemento pode encapsular outro elemento, e assim por diante.

A este documento está associado um DTD chamado de “email.dtd”, ilustrado na Figura 3.

```
<!ELEMENT Email (De, Para, Cc?, Data?, Assunto, Texto)>
<!ELEMENT De (#PCDATA)>
<!ELEMENT Para (#PCDATA)>
<!ELEMENT Cc (#PCDATA)>
<!ELEMENT Data (#PCDATA)>
<!ELEMENT Assunto (#PCDATA)>
<!ELEMENT Texto (#PCDATA | P | Br)*>
<!ELEMENT P (#PCDATA | Br)*>
<!ATTLIST P alinhamento (esquerda | direita | justificado) "esquerda">
<!ELEMENT Br EMPTY>
```

Figura 3: Documento “email.dtd” (VIEIRA, 2005)

O DTD fica responsável pela definição dos elementos e atributos do documento, assim como sua estrutura.

Uma outra forma de controlar a criação de linguagens a partir do XML é adotando a utilização do *namespace.* Uma nova linguagem é formalizada, a partir da criação de um documento XML. O usuário pode criar uma estrutura, e um segundo usuário, outra diferente. Dessa forma, são duas linguagens distintas que devem ser assinaladas de alguma maneira. A função do *namespace* é definir uma origem para os elementos e resolver conflitos na nomenclatura, processo que garante a interoperabilidade de um documento. A Figura 4 exemplifica a representação de um *namespace:*

```
<abc:mail> ... </abc:mail>
<xyz:mail> ... </xyz:mail>
```

Figura 4: Exemplo representativo de um *namespace* (VIEIRA, 2005)

No exemplo acima, duas linguagens diferentes estruturam e-mails “abc” e “xyz”. O *namespace* situa-se exatamente antes dos dois pontos, antes do nome do elemento. Ele permite também que, em um único documento, diferentes linguagens possam ser usadas, garantindo que as aplicações saibam qual linguagem está sendo executada.

Em alternativa ao DTD, que possui uma sintaxe muito especial, tentou-se encontrar uma maneira de escrever em XML a definição de outra linguagem XML. Definiu-se então a linguagem *XML Schema*. Essa linguagem, também é chamada de *XML Schema Definition* (XSD) e seu propósito é definir os blocos de construção permitidos em um documento XML, como um DTD.

A seguir, algumas características do *XML Schema*:

- Definir elementos que podem aparecer em um documento;
- Definir atributos que podem aparecer em um documento;
- Definir se um elemento é vazio ou pode incluir texto;
- Definir tipos de dados para elementos e atributos;
- Definir valores padrão e fixos para elementos e atributos.

1.1.3.2 RDF

O *Resource Description Framework* (RDF) tem como função definir um mecanismo de representação de *metadados* para descrever recursos não vinculados a um domínio específico em uma determinada aplicação. Esses *metadados* são usados para dar significado aos recursos da Web Semântica, permitindo a sua manipulação e compreensão por parte das máquinas. O RDF, em uma área descoberta de recursos, possibilita também a implementação de mecanismos de pesquisa mais eficientes.

Constituem a base do modelo RDF três tipos de objetos. São eles:

- *Recursos:* dado ou informação que se deseja descrever em RDF. Pode ser uma tabela, página da web ou base de dados. Não necessariamente acessível eletronicamente. Pode ser também um objeto como um livro ou revista. Eles são sempre especificados por *Uniform Resource Identifier's* (URIs);
- *Propriedades:* refere-se às características (atributos) de um determinado recurso;
- *Declarações:* é um recurso mais suas propriedades mais o valor de cada uma dessas propriedades. Esse valor final representa um novo recurso.

A Figura 5 é a representação diagramática de uma declaração RDF:

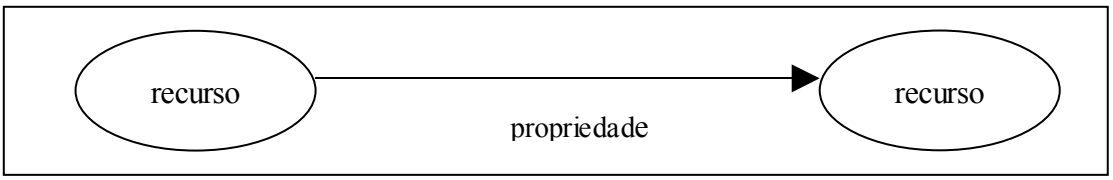

Figura 5: representação diagramática de uma declaração RDF

Essas três partes são denominadas respectivamente de *sujeito*, *predicado* e *objeto*. Portanto, o modelo básico primitivo RDF consiste em triplas formadas por: objeto, predicado e valor.

Duas características tornam o RDF uma das melhores linguagens ou padrões para representação de conhecimento (NARDON, 2002). A primeira característica é o fato de ele

ser muito simples e flexível. A segunda é o fato de que cada conceito representado tem associado a ele uma URI.

No exemplo ilustrado na Figura 6, estão representadas duas declarações. Na primeira, o paciente “ID1234” (sujeito) tem como médico (predicado) uma pessoa chamada “João” (objeto). A segunda determina que o mesmo paciente “ID1234” possui um diagnóstico (predicado) de “câncer hepático” (objeto). Cada sujeito, objeto e predicado possui uma URI que identifica um conceito.

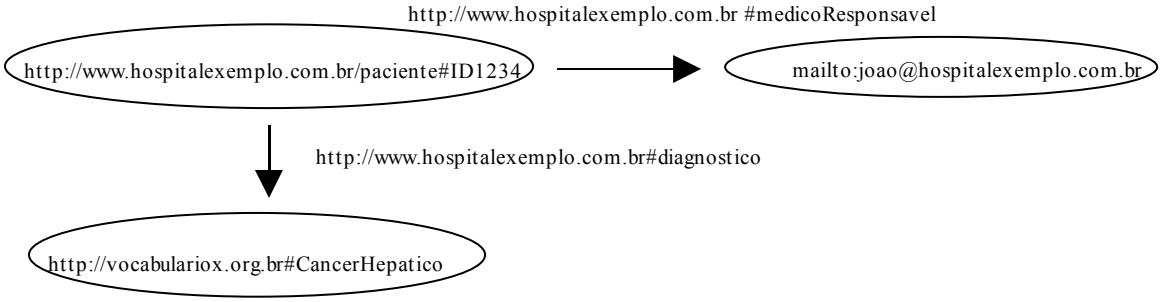

Figura 6: Exemplo de grafo RDF (NARDON, 2002)

A especificação RDF adota o padrão XML para a representação dos grafos. No próximo exemplo, Figura 7, é ilustrada a representação do grafo da Figura 6 em XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:hs="http://www.hospitalexemplo.com.br#">
<rdf:Description rdf : about="http://www.hospitalexemplo.com.br/paciente#ID1234">
        <he:medicoResponsavel>
                mailto:joao@hospitalexemplo.com.br
        </he:medicoResponsavel>
        <he:diagnostico>
                http://www.vocabulariox.org.br#CancerHepatico
        </he:diagnostico>
</rdf:Description>
```

Figura 7: Representação do grafo RDF em XML (NARDON, 2002)

1.1.3.3 OWL

Também uma linguagem proposta pela W3C, a *Ontology Web Language* (OWL) é uma linguagem destinada a definir padrões para a Internet. A OWL apresenta formalmente um vocabulário e definições mais apuradas que as demais linguagens para a descrição de ontologias, no que se refere a relações entre classes, igualdade, cardinalidade, etc. Permite descrever conceitos de um domínio de aplicação e a relação entre conceitos através de vários conjuntos de operadores. Baseia-se em modelo de lógica que torna possível que os conceitos sejam descritos e bem definidos e permite que motores de inferência verifiquem se as declarações e as definições da ontologia são mutuamente consistentes (CARNEIRO, 2003).

A OWL pode ser subdividida em três diferentes linguagens:

- *OWL Lite* - sua finalidade principal é dar suporte para que classificações hierárquicas com restrições simples sejam criadas. Por exemplo, restrições de cardinalidade são permitidas para apenas valores iguais a zero ou um. Assim, *OWL Lite* é própria para a construção de taxionomias e tesauros;
- *OWL Description Logic* - sua finalidade é prover um maior grau de expressividade onde todas as conclusões são computáveis e todas as computações terminam em tempo finito*;*
- *OWL Full* - sua finalidade é prover o máximo de expressividade e liberdade sintática. A partir dela, os usuários podem aumentar o vocabulário pré-definido de RDF ou OWL, porém sem nenhuma garantia computacional. Assim, máquinas de inferência poderão não derivar conclusões ou poderão não ser computáveis em tempo finito.

As classes da ontologia são definidas usando o elemento `owl:Class`. Podemos afirmar que classes em OWL são disjuntas, usando o elemento `owl:disjointwith`, ou seja, um objeto não pode ser instância de mais de uma destas classes. São definidas duas classes mais gerais: `owl:Thing` e `owl:Nothing`, onde toda classe é subclasse de `owl:Thing` e superclasse de `owl:Nothing`. Também podem ser definidas classes equivalentes usando o elemento `owl:equivalenteClass`.

Na Figura 8, tem-se a declaração de duas classes: "`Curso`" e "Pos-`Graduação`". Sendo que, na linha 3, é identificado que "Pos-`Graduação`" é subclasse de "`Curso`".

```
1 <owl:Class rdf:ID="Curso" />
2 <owl:Class rdf:ID="Pos-Graduação">
3 <rdfs:subClassOf rdf:resource="#Curso" />
4 </owl:Class>
```

Figura 8: Declaração de classe e subclasse em OWL

A OWL oferece vários elementos como `owl:Restriction`, `owl:cardinality`, `owl:minCardinality`, `owl:maxCardinality`, `owl:inverseOf`, entre outros. Esses elementos permitem determinar restrições, cardinalidade mínima e máxima, relação inversa, de modo a obter uma ontologia mais completa e melhor modelada.

1.2 Computação Pervasiva

Atualmente com a crescente evolução da informática e dos dispositivos computacionais portáteis, que estão se tornando cada vez mais acessíveis às diferentes classes sociais, a comunicação e troca de informações entre os seres humanos está ocorrendo de forma rápida e eficaz a qualquer hora, em qualquer lugar.

O emprego do termo Computação Pervasiva ficou associado à *International Business Machines* (IBM) quando da edição intitulada *Pervasive Computing* do *IBM System Journal*, onde foi organizada uma digressão sobre os aspectos promissores da Computação Pervasiva, tendo sido também, nessa mesma edição, resgatada por Weiser, no artigo intitulado *"The origins of ubiquitous computing research at PARC in the late 1980s"*, a sua visionária proposta quanto ao futuro da computação, na qual recursos de computação onipresentes se ajustariam, de forma autônoma, para atender os usuários.

Embora a proposta original de Weiser quanto à "Computação Ubíqua" (ou Computação Pervasiva) ainda esteja distante de uma prática cotidiana alicerçada por produtos de mercado, sua proposta vem se materializando, pouco a pouco, através da disponibilização de tecnologias como *PDAs*, *SmartPhones* e a consolidação de padrões para redes sem fio, como, por exemplo, o *Bluetooth* (PINHEIRO, 2006).

Taís aplicações móveis são desenvolvidas pelas características implícitas dessa tecnologia, que envolvem: *autonomia*, *mobilidade* e *proatividade de sistemas*. Isso permite a subdivisão da aplicação em tarefas básicas que facilitam sua distribuição, tornando o processo importante em sistemas destinado à computação móvel, já que ela apresenta características bastante limitadas.

Essa tendência representa o futuro da computação, já que o que antes eram recursos executados apenas por microcomputadores, agora com um pequeno dispositivo é possível aplicar as mesmas funções, em diferentes locais (devidamente adaptados) como restaurantes, aeroportos*, shopping-centers*, etc.

Mas, para que a computação pervasiva se torne uma realidade, ainda existem alguns desafios a serem vencidos. O *espaço pervasivo* (smart space) representa um ambiente altamente dinâmico e heterogêneo, onde a variedade e quantidade de dispositivos e serviços presentes dificultam o desenvolvimento de aplicações para este fim. Nesse sentido, torna-se interessante adquirir novos elementos lógicos (mobilidade de código) em tempo de execução, ganhando assim funcionalidade para atender às demandas do ambiente (BONATTO, 2005).

A questão da adaptação é importante também quando se fala da necessidade de suportar a heterogeneidade nesse tipo de ambiente. Devido a isso, tem sido grande o interesse no desenvolvimento de aplicações *sensíveis a contexto*. Esses sistemas podem ser implementados das mais diversas formas, mas podem ser consideradas três principais abordagens para arquiteturas desses sistemas: arquiteturas ou aplicações que acessam dados diretamente de sensores; sistemas baseados em *middleware,* arquitetura em camadas, onde a interação com os sensores fica escondida da aplicação cliente e por fim, sistemas baseados em *servidores de contexto*, onde, para permitir acesso múltiplo e concorrente às informações de contexto, serviços de notificação são estendidos do *middleware* (FILHO, 2005).

As aplicações *sensíveis a contexto* têm adotado o uso de uma infra-estrutura capaz de gerenciar tais informações de contexto, com o intuito de reduzir a complexidade no desenvolvimento dessas aplicações.

Diante de tal evolução, vários projetos vêm sendo desenvolvidos para a implementação de aplicações sensíveis a contexto, integrando dispositivos móveis e utilizando plataformas *multi-agentes*.

1.2.1 Desafios das aplicações em ambientes pervasivos

A tendência para o futuro, na visão da computação pervasiva, são computadores portáteis que se integraram no cotidiano de todas as pessoas, nos mais variados ambientes abertos e dinâmicos, proporcionando-lhes diversos serviços e compartilhamento de informações.

Seu objetivo principal visa disponibilizar um acesso (aos serviços e informações) computacional de modo invisível e onipresente ao usuário, permitindo que ele utilize essa nova tecnologia com extrema facilidade.

Vários desafios estão presentes no desenvolvimento e integração dessa tecnologia: heterogeneidade, escalabilidade, conectividade transiente, economia de consumo de recursos limitados, gerenciamento do dinamismo do ambiente, interoperabilidade de software (GEYER, 2004). A pervasividade aumenta a complexidade das questões dos sistemas distribuídos e introduz novas questões, como:

- Gerenciamento da mobilidade lógica e física;
- Reconhecimento do contexto de execução;
- Adaptação dinâmica aos recursos disponíveis;
- Descoberta dinâmica de recursos;
- Disseminação de informações.

Entre as questões destacadas acima, a heterogeneidade destaca-se pela presença de inúmeros tipos de dispositivos presentes na atualidade. Devido a essa heterogeneidade, a rede pervasiva necessita de diferentes tipos de protocolos e padrões de comunicação.

Um ambiente pervasivo demanda que as aplicações sejam adaptativas. Assim sendo, a mobilidade de código em tempo de execução, para que possam adaptar seu comportamento de acordo com o ambiente, é de fundamental importância par o sucesso das aplicações.

Outra questão importante é a disponibilidade de dados e serviços, contornando desconexões e falhas em partes da aplicação.

2 USO DE ONTOLOGIAS NA COMPUTAÇÃO PERVASIVA

Atualmente as ontologias têm sido aplicadas com certa freqüência em diversas áreas como: gerência de conhecimento e conteúdo (*knowledge and content management*), comércio eletrônico e na Web Semântica.

No passado, muitas aplicações e arquiteturas tiveram êxito em alguns aspectos na computação pervasiva, porém elas não ofereceram um suporte robusto no que se refere a compartilhamento de informações. Isso se deve ao fato de essas aplicações não terem sido desenvolvidas mediante a adoção de uma ontologia comum (CHEN, 2006).

Atualmente, muitos sistemas já são programados através de uma linguagem orientada a objetos (*Java Class Objects*) para representar o conhecimento sobre o ambiente que os sistemas de computadores possuem. Essas representações requerem que se estabeleça uma série de regras de baixo-nível entre programas para o compartilhamento de informações, pois eles não podem facilitar o conhecimento compartilhado em um ambiente aberto e dinâmico.

Partindo desse contexto, acredita-se que o emprego de ontologias para descrever recursos com o objetivo de compartilhamento de dados (conhecimento) é de fundamental importância dentro da Computação Pervasiva.

Ontologias podem ser usadas para descrever vários conceitos no ambiente de Computação Pervasiva, definindo termos que descrevem diferentes tipos de entidades e suas propriedades. Essas ontologias podem definir diferentes tipos de aplicações, serviços, dispositivos, usuários, dados, fontes, etc. Elas também podem descrever várias relações entre as entidades e estabelecer axiomas nas suas propriedades.

Outra função das ontologias, útil para a Computação Pervasiva, é descrever os diferentes tipos de informações contextuais. Elas podem definir descrições padrões para localizações, atividades, informações relativas ao tempo e outras informações usadas am aplicações de contexto.

Algumas funções das ontologias, dentro de um ambiente pervasivo são (MCGRATH, 2003):

- Verificar se as descrições de diferentes entidades são consistentes com os axiomas definidos na ontologia;
- Permitir que usuários adquiram conhecimento sobre o ambiente e como as diferentes partes que se relacionam entre si;

- Permitir que as buscas sejam executadas em diferentes componentes facilmente, pelos humanos ou agentes automatizados;
- Permitir que humanos e agentes automatizados interajam facilmente com diferentes entidades (através do envio de vários comandos);
- Permitir que novas entidades (que seguem diferentes ontologias) interajam facilmente com o sistema, provendo maneiras para a interoperabilidade da ontologia e permitindo também que os diferentes ambientes pervasivos interajam um com o outro.

Dessa forma, conclui-se que um dos benefícios chaves do uso das ontologias na Computação Pervasiva é que elas ajudam a interação entre usuários e o ambiente, desde que os usuários descrevam concisamente as propriedades e os vários conceitos usados no ambiente.

2.1 Ontologias em Projetos de Ambientes Pervasivos

Na pesquisa bibliográfica realizada, não foram encontrados muitos projetos de Computação Pervasiva que usam ontologias. Por ser ainda um projeto de pesquisa recente, selecionaram-se somente dois (2) projetos para serem descritos, analisados e comparados: projeto COBRA – uso da ontologia *Standard Ontology for Ubiquitous and Pervasive Aplications* (SOUPA) e projeto GAIA – adaptação da ontologia *Darpa Agent Markup Language* associada a *Ontology Inference Layer* (DAML+OIL).

2.1.1 SOUPA

Uma ontologia que preenche os pré-requisitos necessários para atender aplicações destinadas a Ambientes da Computação Pervasiva é chamada de SOUPA. Essa ontologia baseia-se na linguagem OWL e inclui componentes inteligentes na sua estrutura (políticas de segurança e privacidade, perfil de usuários, etc).

O projeto do SOUPA começa em novembro de 2003 e é parte de um esforço contínuo da Web Semântica no grupo *UbiComp Special Interest Group*, um grupo internacional que

reúne academia e indústria, no qual se utiliza a linguagem OWL para Computação Pervasiva. O objetivo do projeto consiste em definir ontologias para suportar aplicações destinadas à pervasividade. O projeto do SOUPA é representado em uma série de *Casos de Uso.*

Os vocabulários SOUPA são retirados de várias outras ontologias diferentes. A estratégia do desenvolvimento do SOUPA está justamente no fato de que ele obtém "emprestado" os termos dessas ontologias, mas não os importa diretamente. Os autores consideram que, embora a importação de ontologias seja bem conhecida, a escolha, por não usar essa abordagem é devido ao *overhead* introduzido pela máquina de raciocínio para importar ontologias, que podem ser irrelevantes para as aplicações pervasivas.

Para permitir uma melhor interoperabilidade entre as aplicações de SOUPA e outras ontologias, é utilizada a linguagem OWL como padrão na sua construção.

A seguir, descreve-se as características de cada uma dessas ontologias que compõem o SOUPA, mostrando sua relevância nas aplicações para Computação Pervasiva.

- ***Friend-of-a-Friend (FOAF).*** Esta ontologia representa um importante bloco para informatização de sistemas que apóiam comunidades *on-line*. Ela permite a expressão pessoal e relacionamentos. Na Computação Pervasiva, *FOAF* pode ser usada para expressar e argumentar sobre perfis de contatos de pessoas e grupos sociais de sua proximidade.

- ***DAML-Time & the Entry Sub-ontology of Time.*** Os vocabulários dessa ontologia são usados para expressar conceitos temporais e propriedades comuns em qualquer formalização de tempo. Na Computação Pervasiva, essa ontologia é utilizada para compartilhar uma representação comum de tempo e argumentar sobre as ordens temporais de diferentes eventos.

- ***OpenCyc Spatial Ontologies & RCC.*** A ontologia *OpenCyc Spatial Ontologies* define um conjunto de vocabulários para representação simbólica do espaço. A ontologia de *RCC* consiste em vocabulários com a finalidade de expressar relações racionais no espaço. Na Computação Pervasiva, essa ontologia pode descrever e argumentar sobre localização e contexto de localização.

- ***COBRA-ONT.*** Ontologia destinada ao suporte de representação do conhecimento e argumentação no ambiente da Computação Pervasiva. *COBRA-ONT* visa modelar contextos em "salas de reunião inteligentes".

- ***MoGATU BDI Ontology.*** Assim como a ontologia COBRA-ONT, a *MoGATU BDI Ontology* também é destinada ao suporte de representação do conhecimento e argumentação no ambiente pervasivo, porém ela visa modelar argumentos como *desejos* e *intenções* dos usuários e agentes de softwares.

- ***Rei Policy Ontology.*** A linguagem *Rei Policy Ontology* define um conjunto de conceitos (direitos, proibições e obrigações) para especificar e argumentar sobre seguranças e controles de acesso. No ambiente da Computação Pervasiva, os usuários podem usar essa ontologia para especificar regras de alto nível por conceder ou revogar acessos de diferentes tipos de serviço.

A ontologia SOUPA é sub-dividida em duas ontologias distintas denominadas *SOUPA Core* e *SOUPA Extention*. A ontologia *SOUPA Core* tenta definir vocabulários genéricos, universais para diferentes aplicações da computação pervasiva. Já a ontologia *SOUPA Extention* (que estende a SOUPA Core), define vocabulários adicionais para suportar tipos específicos de aplicações e provê exemplos para futuras extensões de ontologias. Ela apenas sugere vocabulários adicionais para suportar aplicações pervasivas, não havendo diferença na complexidade computacional com a SOUPA Core. A seguir, descreveremos com mais detalhes as características das duas ontologias.

2.1.1.1 SOUPA Core

A ontologia se agrupa em nove grupos distintos na sua documentação. Contém vocabulários pra expressar conceitos que estão associados a pessoas, agentes, convicção - desejo - intenção (BDI), ação, política, tempo, espaço e evento.

Pessoa

A classe *Pessoa* (Person) define vocabulários típicos para descrever informações referentes a um contato e seu perfil pessoal. A classe OWL `per:Person` é definida para representar as pessoas do domínio SOUPA, e é equivalente a `foaf:Person`, classe da ontologia FOAF. Ela descreve uma serie de propriedades que incluem informações básicas como: (i) nome, idade, data de nascimento, etc; (ii) informações de contato como: *email*, endereço, telefones, *homepage*, etc; (iii) e perfis profissionais ou sociais como: amigos, colegas de trabalho, etc.

Além das propriedades dos vocabulários, que são aplicáveis para descrever uma pessoa na ontologia FOAF, também pode-se descrever um indivíduo da classe `per:Person`. Isto porque todos os indivíduos da classe `per:Person` também são os indivíduos da classe `foaf:Person`.

A Figura 10 descreve uma ontologia para a pessoa chamada “Harry Chen”:

```
<per:Person>
        <per:firstName rdf:datatype="&xsd;string>Harry</per:firstName>
        <per:lastName rdf:datatype="&xsd;string>Chen</per:lastName>
        <per:homepage rdf:resource="http://umbc.edu/people/hchen4"/>
        <per:hasSchoolContact rdf:resource="#SchoolContact"/>
        <per:hasHomeContact rdf:resource="#HomeContact"/>
</per:Person>
<per:ContactProfile rdf:ID="SchoolContact">
         <per:address rdf:datatype="&xsd;string">
                  Dept. of CSEE, UMBC, 1000 Hilltop Circle,
                  Baltimore, MD 21250, USA
          </per:address>
          <per:phone rdf:datatype="&xsd;string> +1-410-455-8648 </per:phone>
          <per:email rdf:resource="mailto:harry.chen@umbc.edu"/>
</per:ContactProfile>
```

Figura 10: Exemplo da classe *Pessoa* (CHEN, 2006)

Política e Ação

Nos ambientes pervasivos, as políticas de segurança e as ações são as grandes preocupações das aplicações. *Política* (Policy) é uma técnica para controle e ajuste de comportamento de sistemas de baixo nível, especificando regras de alto nível. Os

vocabulários desta ontologia são herdados da ontologia *Rei Policy*. Ela define vocabulários para a representação de segurança, políticas de privacidade e uma descrição lógica baseada em mecanismos para argumentar sobre uma política. São definidas no *Documento de Ontologia de Política* (Policy Ontology Document), onde a classe `pol:Policy`, representa um conjunto de todas as políticas.

A Política é uma série de regras especificadas por um usuário ou uma entidade computacional para restringir ou guiar a execução de ações. A representação ontológica de uma *Ação* (Action) é definida no *Documento de Ontologia de Ação* (Acion Document Ontology). A classe que representa o conjunto das possíveis ações chama-se `act:Action`. Objetos dessa classe possuem as seguintes propriedades:

- `act:actor` – entidade que executa a ação;
- `act:recipient` – entidade que recebe o efeito depois da ação executada;
- `act:target` – objeto que a ação aplica para;
- `act:location` – localização onde a ação é executada;
- `act:time` – tempo em que a ação foi executada;
- `act:instrument` – objeto que o ator usa para executar a ação.

A Figura 11 descreve uma ontologia que define uma classe especial de conhecimento que compartilha ação, onde o ator é o agente "ctb@cobra1.cs.umbc.edu", o receptor é qualquer sócio do grupo "eBiquity" e a informação a ser compartilhada é a localização de "Harry".

```
<owl:Class rdf:ID="ShareHarryLocInfoWithEBMembers">
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="&act;Action"/>
                <owl:Restriction>
                        <owl :onProperty rdf:resource="&act;actor">
                        <owl l:hasValue>
                                <agt:Agent rdf:about="ctb@cobra1.cs.umbc.edu"/>
                        </owl:hasValue>
                </owl:Restriction>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="&act;target"/>
                        </owl:allvaluesFrom rdf:resource="#LocationContextOfHarry"/>
                </owl:Restriction>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="&act;recipient"/>
                        <owl:allValuesFrom rdf:resource="&eb;EbiquityMembers"/>
                </owl:Restriction>
          </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:ID="LocationContextOfHarry">
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="&loc;LocationContext"/>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="&loc:locationContextOf"/>
                        <owl:hasValue>
                                <per:Person rdf:about="http://umbc.edu/people/hchen4"/>
                        </owl:hasValue>
                </Restriction>
        </owl:intersectionOf>
</owl:Class>
```

Figura 11: Exemplo da classe *Política* (CHEN, 2006)

Para um determinado objeto, podem estar associados mais de um acesso (`pol:permits`) ou proibição (`pol:forbids`). Essas duas propriedades estendem-se respectivamente às classes `pol:PermittedAction` e `pol:ForbiddenAction`.

A Figura 12 ilustra uma política que atribui ao agente "ctb@cobra1.cs.umbc.edu" a permissão para compartilhar a informação que indica a localização de "Harry", com todos os membros da "eBiquity".

```
<pol:Policy rdf:about="&cobra;harrychen-policy">
        <pol:policyOf>
                <per:Person rdf:about="http://umbc.edu/people/hchen4">
                        <per:name rdf:datatype="&xsd;string">Harry Chen</per:name>
                </per:Person>
        </pol:policyOf>
        <pol:defaultPolicyMode rdf:resource="&pol;RequiresExplicitPermission"/>
        <pol:permits rdf:resource="#ShareHarryLocInfoWithEBMembers"/>
</pol:Policy>
```

Figura 12: Exemplo da classe *Ação* (CHEN, 2006)

A política de ontologia também define vocabulários para descrever metainformação sobre políticas individuais:

- `pol:creator`: informa a autora de uma política;
- `pol:enforcer`: entidade que obriga uma política;
- `pol:createdOn`: o tempo de criação de uma política;
- `pol:defaultPolicyMode`: é a falta de argumento de uma política.

O design da política do SOUPA utiliza a classificação como um meio para argumentar sobre políticas. Um típico fluxo do processo da implementação do sistema é o seguinte:

- um usuário ou um sistema administrador define uma política;
- a política é transmitida para um ente apropriado (*policy enforcer*), por exemplo, uma segurança ou um protetor de privacidade do agente;
- antes do *policy enforcer* poder permitir que outros agentes executem uma ação, ele cria uma representação explícita da ação usando o *SOUPA action ontology*;
- esta ação representada é então carregada em um interpretador de lógica de descrição, junto com a ontologia associada;
- o *policy enforcer* permitirá a execução da ação, se ela é classificada como: `pol:PermitedAction`, e proibirá a mesma se ela for classificada como `pol:ForbiddenAction`. No caso de uma ação de introdução ser classificada como ambas, então o *policy enforcer* informará se há uma inconsistência na política e pode proibir sua execução da ação.

Se determinada ação não pode ser classificada em uma classe qualquer, a decisão de permissão ou não de sua execução fica a cargo do *policy enforcer*, baseado-se no *default policy mode* (descrito no exemplo anterior). A ação será proibida se a política for `pol:RequiresExplicitPermission` e permitida se for `pol:RequiresNoExplicitPermission`.

Agente e BDI

Na ontologia *Agente e BDI*, entidades e usuários podem ser modelados como agentes. A representação explícita dos *estados mentais* (*mental states*) dos humanos pode ajudar na

computação de agentes para argumentar sobre as necessidades específicas dos usuários em um ambiente pervasivo.

Esta ontologia contém duas documentações denominadas *Agente* (Agent) e *BDI*. A classe `agt:Agent` descreve um conjunto de agentes existentes na ontologia e está associada a três propriedades para caracterizar os *estados mentais* dos próprios agentes. São eles: `agt:believes, agt:desires` e `agt:intends`. Os valores respectivos que estas propriedades podem assumir são: `bdi:Fact, bdi:Desire` e `bdi:Intention`.

A `bdi:Fact` é uma sub-classe da classe `rdf:Statement`, que representa um conjunto de declarações de reificação RDF. A classe `bdi:Desire` define um conjunto geral de estados que agentes podem provocar. A classe de `bdi:Intention` representa um conjunto de planos para que os agentes executem. Esses planos são definidos em termos de ações, pré-condições e efeitos.

Tempo

Em SOUPA existem também classes que definem relações de tempo. Essa representação consiste basicamente de duas classes: `tme:Instant` e `tme:TimeInterval`; onde a união delas forma a classe `tme:TemporalEntity` (classe da linguagem OWL).

O conjunto de todos os objetos temporais estão divididos em duas classes: `tme:InstantThing`, objetos com descrições temporais do tipo *instantes de tempo* e a classe `tme:IntervalThing`, objetos com descrições temporais relativas a *intervalos de tempo*. A união dessas classes dá origem à classe `tme:TemporalThing`. Para associar intervalos de tempo que possuem na sua descrição data e hora, existe a propriedade `tme:at`, definida para associar uma instância do `tme:InstantThing` com um valor (do tipo data) XML `xsd:dateTime`, e as propriedades `tme:from` e `tme:to` são definidas para associar uma instância do IntervalThing com dois diferentes `tme:TimeInstant`.

A Figura 13 mostra a representação de um intervalo de tempo com uma descrição temporal associada:

```
<tme:TimeInterval>
        <tme:from>
                <tme:TimeInstant>
                        <tme:at rdf:datatype="xsd;dateTime">
                                2004-02-01T12:01:01
                        </tme:at>
                </tme:TimeInstant>
        </tme:from>
        <tme:to>
                <tme:TimeInstant>
                        <tme:at rdf:datatype="xsd;dateTime">
                                2004-02-11T13:41:21
                        </tme:at>
                </tme:TimeInstant>
        </tme:to>
</tme:TimeInterval>
```

Figura 13: Exemplo da classe *Tempo* (CHEN, 2006)

A ontologia define ainda, propriedades para descrever os tipos de relações entre dois instantes de tempo diferentes. São elas: `tme:before`, `tme:after`, `tme:beforeOrAt`, `tme:afterOrAt` e `tme:sameTimeAs`. As propriedades, `tme:before` e `tme:after`, são definidas do modelo `owl:TransitiveProperty`. A propriedade `tme:sameTimeAs` expressa esses dois diferentes instantes de tempo associados, com valores do tipo data e hora equivalentes. Essa propriedade é definida do modelo `owl:SymmetricProperty`.

Existem ainda várias outras propriedades que descrevem diferentes comparações entre dois tempos como: `tme:startsSoonerThan`, `tme:startsLaterThan`, `tme:startsSameTimeAs`, `tme:endsSoonerThan`, `tme:endsLaterThan`, `tme:endsSameTimeAs`, `tme:startsAfterEndOf` e `tme:endsBeforeStartOf`.

Espaço

Ontologia que herda vocabulários da *OpenCyc Spatial Ontologies*, para descrever sobre as relações espaciais entre diferentes regiões geográficas, traçando coordenadas geo-espaciais para a representação simbólica do espaço, como também a representação das medidas geográficas espacias.

Essa ontologia possui dois documentos de ontologia denominados *espaço* (space) e *geomedidas* (geo-measurement). A documentação *espaço* é responsável pela representação simbólica do espaço e as relações espaciais. A documentação *geomedidas* define vocabulários espaciais do tipo: latitude, longitude, distância e área de superfície.

A classe `spc:SpatialThing`, representa um conjunto de todas os objetos que têm extensões espacias no domínio SOUPA. A `spc:GeographicalSpace`, é normalmente composta por objetos de espaço encontrados em mapas ou fotos do gênero. As classes `spc:GeographicalRegion`, `spc:FixedStructure` e `spc:SpaceInAFixedStructure` juntas, compõem a classe `spc:GeographicalSpace`.

No modelo de representação espacial, o indivíduo membro da classe `spc:SpatialThing` é descrito pelas coordenadas de localização. Essas coordenadas são expressas pela propriedade `spc:hasCoordinates`, escala da classe `geo:LocationCoordinates`.

Nesse modelo, múltiplas coordenadas de localização podem ser traçadas para uma única região geográfica, um processo importante para definir cartografias de espaço de diferentes localizações geográficas e coordenadas GPS.

Evento

Eventos (Events) são atividades de extensões espaciais ou temporais que também podem ser descritos pelas ontologias. Esses eventos podem ser: a ocorrência de atividades, horários, etc. A classe `eve:Event` representa um conjunto de todos os possíveis eventos no domínio. A união das classes `tme:TemporalThing e spc:SpatialThing` originam a classe `eve:SpatialTemporalThing`. Esta tem a função de representar as extensões temporais e espaciais. A intersecção das classes `eve:SpatialTemporalThing` e `eve:Event` forma a classe `eve:SpatialTemporalEvent`. Esta, por sua vez, define a descrição de eventos de ambas extensões, temporal e espacial.

Na Figura 14, mostra-se como a ontologia pode ser usada para descrever um evento no qual um dispositivo *Bluetooth* foi detectado em uma determinada data e hora, bem como a localização (através da latitude e longitude) do dispositivo.

```
<owl:Class rdf:ID="DetectedBluetoothDev">
        <rdfs:subClassOf
                rdf:resource="&eve;TemporalSpatialEvent"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="foundDevice">
        <rdfs:domain
        rdf:resource="#DetectedBluetoothDev"/>
</owl:ObjectProperty>
<DetectedBluetoothDev>
        <spc:hasCoordinates>
                <geo:LocationCoordinates>
                        <geo:longitude rdf:datatype...>
                                -76.7113
                        </geo:longitude>
                        <geom:latitude rdf:datatype...>
                                39.2524
                        </geom:latitude>
                </geo:LocationCoordinates>
        </spc:hasCoordinates>
        <foundDevice rdf:resource="url-x-some-device"/>
        <tme:at>
                <tme:TimeInstant>
                        <tme:at rdf:datatype="xsd;dateTime">
                                2004-02-01T12:01:01
                        </tme:at>
                </tme:TimeInstant>
        </tme:at>
<DetectedBluetoothDev>
```

Figura 14: Exemplo da classe *Evento* (CHEN, 2006)

2.1.1.2 SOUPA Extension

Outra divisão da ontologia SOUPA é chamada de *SOUPA Extension*. Essa ontologia tem por objetivos definir um conjunto estendido de vocabulários para suportar tipos específicos de domínios em aplicações pervasivas e demonstrar como definir novas ontologias, estendendo a ontologia SOUPA Core.

A seguir, serão descritas as ontologias que compõem o SOUPA Extension.

Meeting & Schedule

Ontologia responsável pelas informações referentes a reuniões, programações e seus participantes.

Document & Digital Document

Ontologias que descrevem metainformações de diversos documentos e documentos digitais (por exemplo, a criação da data e autor de um documento; a fonte URL de um documento digital, tamanho e tipo de arquivo).

Image Capture

Ontologia que define vocabulários para descrever imagens capturadas pelos eventos (onde e quando uma foto é tirada, qual dispositivo que tirou a foto).

Region Connection Calculus

Ontologia que define vocabulários para expressar relações de espaço para o raciocínio do espaço qualitativo. Ela complementa a Ontologia *Core Space*.

Location

Ontologia responsável pelo contexto de localização de uma determinada pessoa. Esse contexto descreve a localização de uma pessoa ou objeto, incluindo propriedades temporais e de espaço.

A Figura 9 ilustra o diagrama da ontologia SOUPA e seus relacionamentos.

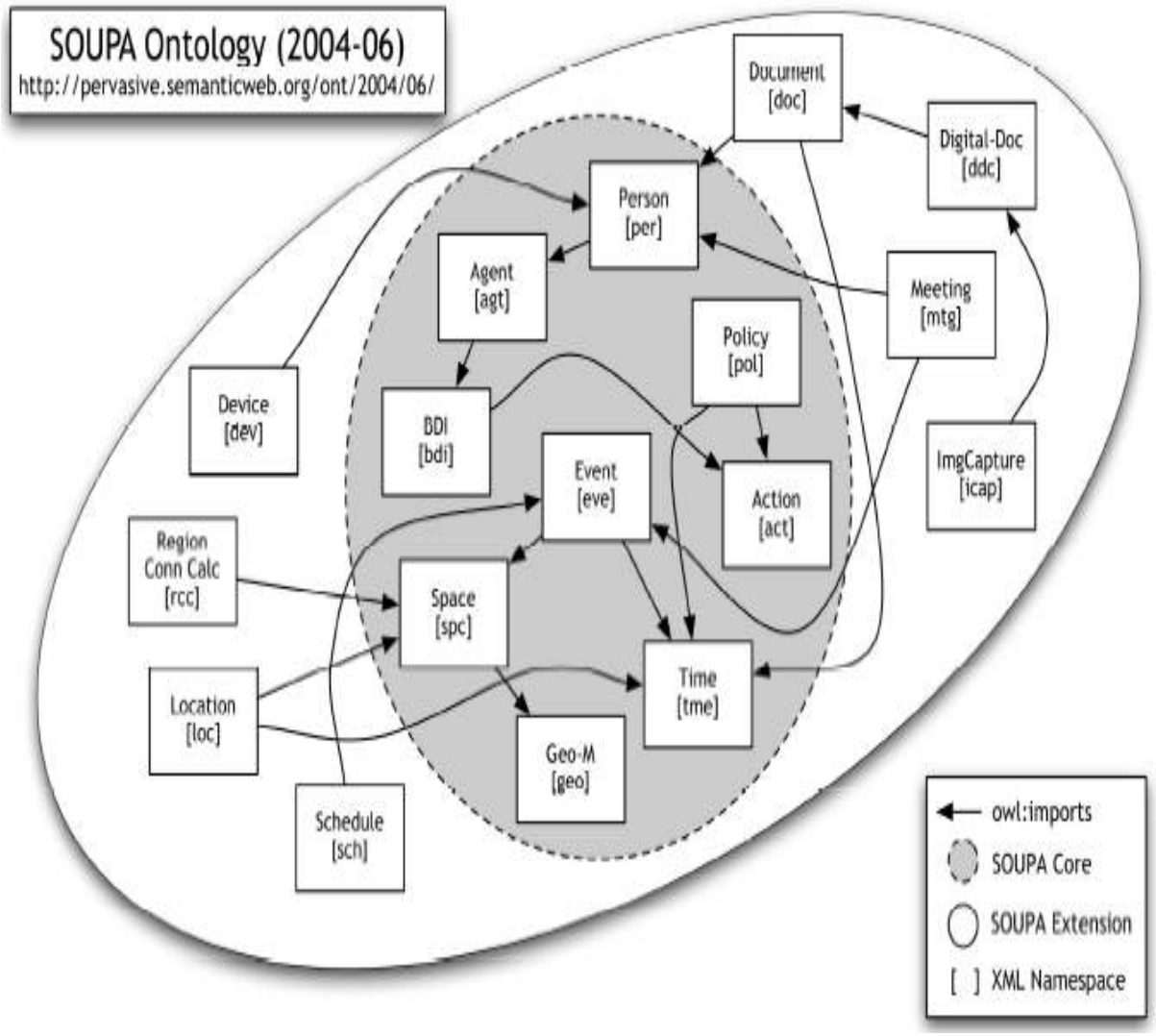

Figura 9: Diagrama da ontologia SOUPA e seus relacionamentos (CHEN, 2006)

2.1.1.3 Aplicação SOUPA - COBRA

O *Context Broker Architecture* (COBRA) é uma arquitetura que permite a implementação de agentes, serviços e dispositivos que exploram informações de contexto em espaços ativos. Foi projetada para atender quatro aspectos principais:

- Forma de representação das informações de contexto;
- Como possibilitar o compartilhamento das informações;
- Como permitir a inserção de novos dados (mesmo aqueles que não podem ser recebidos diretamente de sensores);

- Como proteger e estabelecer políticas para a privacidade do usuário no compartilhamento das informações.

Seu principal componente é denominado *context broker*, um agente inteligente responsável por uma série de responsabilidades em um ambiente inteligente.

A aplicação de um *context broker* foi projetada para ambientes pequenos e fechados, como escritórios ou salas de reuniões, onde espaços diferentes podem ser atendidos por diferentes *brokers* ao passo que um usuário pode receber diferentes informações dos diferentes *brokers* e uni-las para uma visão geral do ambiente como um todo.

Além dos aspectos pelos quais foi projetado, é também sua responsabilidade no ambiente: receber informações de contexto de fontes que não estão acessíveis a partir de dispositivos que possuem recursos limitados e detectar e corrigir inconsistências nos dados de contexto recebidos.

O *context broker* se divide em quatro componentes principais:

- Uma base de dados que armazena informações de contexto representadas em forma triplas RDF;
- O *Context Reasoning Engine* (CoRE), uma máquina de inferência de texto que permite determinar novos dados de contexto a partir de dedução lógica baseada na semântica do OWL e regras de dedução;
- Um módulo de captação de contexto, que consiste em uma biblioteca de procedimentos que permitem criar uma abstração de *middleware* para esconder a complexidade envolvida na coleta de contexto;
- Um módulo responsável pelo gerenciamento da política de privacidade, o *Module for Privacy Protection* (MoPP), que determina quando as informações de contexto de um usuário/agente podem ser compartilhadas.

No ambiente COBRA, existe um protocolo denominado *Agent Communication Language* (ACL), padrão da *Foundation for Intelligent Physical Agents* (FIPA), cuja função é efetuar a ligação de todos os agentes com o *broker,* utilizando as ontologias pre-definidas para representação de contexto. Esses agentes que fornecem ou consomem contexto, podem se comunicar diretamente com o *context broker* para executar suas tarefas.

Todas as entidades devem estar cientes da presença do context broker. Para conhecer sua presença, os agentes participantes podem recorrer ao serviço de descoberta padrão da plataforma de agentes utilizada na implementação, chamado de *FIPA directory facilitator.*

Através da arquitetura COBRA, foram implementadas diferentes aplicações. Dentre as principais, destaca-se a *EasyMeeting* como uma infra-estrutura capaz de prover serviços sensíveis a contexto em uma conferência, auxiliando palestrantes e ouvintes. Essa aplicação utiliza ontologias e regras de lógica de inferência para argumentar sobre a localização dos participantes de uma reunião, baseada na localização de seus celulares pessoais. A partir dessa localização, provê outros serviços que auxiliam reuniões em salas de conferência.

São utilizadas ontologias na arquitetura COBRA por duas razões:

- Prover uma representação explícita do tipo de informação de contexto que o *context broker* seria capaz de compartilhar e processar;
- Eliminar ambigüidades com relação a informações de contexto que poderiam ter diferentes significados em implementações distintas.

Para expressar ontologias de contexto e para realizar inferências, foram escolhidas as linguagens da Web Semântica. Uma delas foi a OWL, pelo fato de ser uma linguagem que permite a construção de uma base maior de conhecimento dentro da ontologia. A OWL foi projetada como um padrão e tem o apoio de uma reconhecida organização (W3C).

A ontologia COBRA importa várias ontologias superiores do padrão SOUPA, desenvolvido especialmente para dar suporte a aplicações pervasivas que utilizam ontologias. Para seu desenvolvimento, foram adotados vocabulários comuns de outras ontologias, mesmo método utilizado pela ontologia SOUPA.

2.1.2 DAML+OIL

No processo de pesquisa de uma linguagem de ontologia para a Web Semântica, o RDF se tornou padrão de *sintaxe*, já que estava sendo bem aceito e usado por outras atividades.

Ao mesmo tempo em que a *Ontology Inference Layer* (OIL) se desenvolvia e se encaixava na sintaxe do RDF, a pesquisa do *Darpa Agent Markup Language* (DAML) também se baseava nessa sintaxe e adicionava conceitos semelhantes ao OIL, como: classe, restrições locais e outras construções inerentes às lógicas de descrições.

A OIL, linguagem desenvolvida pelo *OIL Consortium,* é uma proposta para uma representação baseada na Web e em nível de inferência para ontologias, que combina

primitivas de modelagens de linguagens baseadas em *frame* com semântica formal e serviços de raciocínio providos por descrições lógicas que descrevem precisamente os significados dos termos. DAML, linguagem criada em agosto de 2002 na agência de pesquisa *Defence Advanced Research Projects Agency* (DARPA), tem por objetivo descrever relacionamentos entre objetos.

Embora diferentes em alguns aspectos, as linguagens OIL e DAML deram origem à linguagem DAML+OIL, uma nova proposta do consórcio W3C, para servir como ponto de partida para as atividades da Web Semântica.

A linguagem OWL utiliza a estrutura DAML+OIL como base para a representação de ontologias e metadados. Ambas ontologias são semelhantes, porém a OWL por se tratar de uma linguagem mais recente, inclui novas primitivas para a Web Semântica.

A linguagem DAML+OIL também permite o uso do tipo de dados *XML Shema* para descrever ou definir parte do domínio de tipo de dados. Esses dados não são objetos individuais e são usados com suas URIs na ontologia.

A ontologia DAML+OIL é composta de diversos componentes que podem ser opcionais ou até mesmo repetidos. Eles podem ser objetos do tipo: *headers*, *elementos de classe* (class elements), *elementos da propriedade* (property elements) e *instâncias*. A ontologia consiste em zero ou mais *headers*, seguidos por zero ou mais *elementos de classe*, *elementos de propriedade* e *instâncias*.

2.1.2.1 Elementos da Ontologia

A seguir, são descritas as características de cada um desses objetos existentes na ontologia DAML+OIL.

Headers

Um elemento `daml:Ontology` contém zero ou mais informações de versão (*versionInfo*) e elementos de importação (*imports*), conforme ilustra a Figura 15.

- `daml:versionInfo` - elemento que contem uma *string* que informa a versão.

- *daml:imports* - elemento que referencia (através de uma URI) uma outra declaração de ontologia, contendo definições que se aplicam ao recurso sendo escrito.

```
<daml:Ontology rdf:about="">
  <daml:versionInfo>
        $Id: NOTE-daml+oil-reference-20011218.html,v 1.6 2001/1
  </versionInfo>
  <rdfs:comment>
        An example ontology
   </rdfs:comment>
  <daml:imports rdf:resource="http://www.w3.org/2001/10/daml+oil"/>
</daml:Ontology>
```

Figura 15: Exemplo do elemento `daml:Ontology` (CONNOLLY, 2001)

Elementos de Classe

Um elemento `daml:Class`, sub-classe de `rdfs:Class`, contém a definição de uma ontologia de objeto. Ele referencia o nome de uma classe através de uma URI.

Os principais elementos de classes da ontologia estão descritos a seguir:

- *rdfs:subClassOf*: define que a classe que contém este elemento é uma subclasse da expressão contida nele;
- *daml:disjointWith*: define que a classe não deve ter instâncias em comum com a expressão contida nesse elemento;
- *daml:disjointUnionOf*: define que a classe tem as mesmas instâncias que a união disjunta dos elementos da expressão desse elemento. Assim, todas as classes definidas por expressões de um elemento *disjointUnionOf* devem ser disjuntas, não podendo existir uma instância que pertença a mais do que uma das expressões da lista;
- *daml:sameClassAs*: define que a classe é equivalente à expressão contida nesse elemento. A classe e todas as expressões desse elemento devem ter as mesmas instâncias;
- *daml:equivalentTo*: elemento de mesma semântica do *sameClassAs*. Ele é declarado como um *subProperty* de *subClassOf*, enquanto o *equivalentTo* não. Isso torna o significado de *sameClassAs* parcialmente disponível a um agente RDF Schema, enquanto o significado de *equivalentTo* não é estendido pelo mesmo agente;

- *daml:oneOf:* define que a classe contém exatamente o número de elementos descrito.

Elementos da Propriedade

Um elemento `rdf:Property` refere-se ao nome da propriedade, definido por uma URI. Um elemento pode conter as seguintes propriedades:

- `rdfs:SubPropertyOf`: define que a propriedade é uma sub-propriedade da descrita no elemento;
- `rdfs:domain`: define que a propriedade se aplica somente às instâncias da expressão da classe do elemento;
- `rdfs:range`: define que a propriedade assume somente valores que são instâncias da expressão desse elemento;
- `daml:samePropertyAs`: define que a propriedade é equivalente à propriedade descrita;
- `daml:inverseOf`: define que a propriedade é a relação inversa da propriedade descrita;
- `daml:UniqueProperty`: define que a propriedade pode ter somente um único valor para cada instância.

Instâncias

As instâncias das classes e das propriedades são escritas na sintaxe RDF e RDF Schema, como exemplificado na Figura 16:

```
<continent rdf:ID="Asia"/>
<rdf:Description rdf:ID="Asia">
  <rdf:type>
   <rdfs:Class rdf:about="#continent"/>
  </rdf:type>
</rdf:Description>
<rdf:Description rdf:ID="India">
  <is_part_of rdf:resource="#Asia"/>
</rdf:Description>
```

Figura 16: Instâncias em RDF e RDFS (CONNOLLY, 2001)

Objetos e Datatypes

DAML+OIL divide o universo em duas partes. Uma dessas partes consiste de valores que pertencem aos *datatypes* do XML Schema e chama-se *domínio tipo de dado*. A outra parte consiste de objetos que são considerados membros das classes descritas no DAML+OIL (ou RDF) e chama-se *domínio objeto*.

DAML+OIL está principalmente direcionado a criação de classes que descrevem (ou definem) partes do *domínio objeto*. Tais classes são chamadas *classes de objeto* e são elementos do `daml:Class`, uma subclasse da `rdfs:Class`. DAML+OIL também permite o uso de XML Schema para descrever (ou definir) partes do *domínio datatype*. Estes *datatypes* não são objetos individuais e são usados na DAML+OIL para incluir suas URIs dentro da ontologia. Eles são elementos (implícitos) da `daml:Datatype`.

Expressões de Classe

Uma ***expressão de classe* é** o nome atribuído para:

- um nome de classe (uma URI);
- uma enumeração, incluída na `daml:Class`;
- uma restrição de propriedade (*property restriction*);
- uma combinação booleana, incluída na `rdfs:Class`.

Cada *expressão de classe* refere-se ao nome de uma classe, isto é, à classe que é identificada pela URI ou define implicitamente uma classe anônima, respectivamente a classe que contém exatamente os elementos enumerados, ou a classe de todas as instâncias que satisfazem a *restrição de propriedade,* ou a classe que satisfaz a combinação booleana de tais expressões.

Dois nomes da classe são predefinidos: as classes `daml:Thing` e `daml:Nothing`. Cada objeto é um membro da `daml:Thing` e nenhum objeto é membro da `daml:Nothing`. Conseqüentemente, cada classe é uma sub-classe da `daml:Thing` e a classe `daml:Nothing` é uma sub-classe de todas as classes.

2.1.2.2 Arquitetura de Aplicação: projeto GAIA

GAIA é uma infra-estrutura destinada a espaços pervasivos, que são ambientes da Computação Pervasiva (MCGRATH, 2003). A função desta arquitetura é converter esses espaços físicos, juntamente com os dispositivos que eles contêm, em um sistema programável, oferecendo serviços para administrar espaços e seus estados.

Com o GAIA, especificando interfaces bem-definidas para serviços, podem ser construídas aplicações de uma maneira genérica, ou seja, aplicações que podem ser executadas em diversos *espaços inteligentes*. Os serviços centrais são iniciados por um *protocolo de bootstrap* que, por sua vez, inicia a infra-estrutura de GAIA.

GAIA pode ser comparado a sistemas operacionais tradicionais que administram tarefas comuns a todas aplicações construídas para espaços físicos.

Com ele, é possível fazer com que *salas inteligentes*, com os mais variados tipos de dispositivos, atendam as necessidades de diferentes usuários. Esses dispositivos podem ser:

- Dispositivos de autenticação como sensores de impressão digital e leitores de cartões inteligentes (*smart card readers*);
- Dispositivos de exibição como grande telas de plasma, paredes de vídeos;
- Dispositivos usáveis como relógios e anéis inteligentes;
- Dispositivos *input* como “telas de toques” e microfones.

GAIA provê serviços centrais (*core services*), incluindo eventos, presença de entidade (dispositivos, usuários e serviços), descoberta de recursos e serviço de nomes.

Cada espaço em GAIA é auto-suficiente, porém eles podem interagir com outros espaços. GAIA usa *Common Object Request Broker Architeture* (CORBA), como middleware de comunicação distribuída, para permitir que entidades distribuídas comuniquem-se umas com as outras.

Ontologias em GAIA

As ontologias são empregadas em todo framework GAIA e são armazenadas em um *Servidor de Ontologia* (Ontology Server). Ele é um serviço de sistema padrão que provê uma interface genérica para administrar: ontologias DAML+OIL; uma *Base de Conhecimento*

(Knowledge Base) e questões de lógica. As ontologias e a Base do Conhecimento são firmemente integradas em todo sistema.

GAIA utiliza ontologias DAML+OIL para classificar e descrever muitos conceitos de Ambientes da Computação Pervasiva. Elas possuem *metadados* sobre os diferentes tipos de entidades no ambiente. As ontologias incluem entidades do sistema (não limitada ao software) e informações de contexto. A função das ontologias e do Servidor de Ontologia é de argumentar serviços de sistema, incluindo: administrador de configuração; descoberta de recursos (*Discovery and Matchmaking*); interfaces humanas; interoperações de componentes; um comportamento *sensível ao contexto* (Context Sensitive).

Ontologias argumentam ou substituem taxonomias informais e implícitas dos diferentes tipos de entidades no sistema e descrevem o relacionamento de uma com a outra.

GAIA usa os conceitos descritos na ontologia *CORBA FaCT Reasoning Engine* para certificar-se que esses conceitos são logicamente consistentes e para responder a questões lógicas. Outras entidades em GAIA se conectam ao Servidor de Ontologia para corrigir descrições das entidades no ambiente, metainformações sobre contextos ou definições de vários termos usados em GAIA. Questões referentes à semântica (para instâncias, classificações de indivíduos) também podem ser resolvidas usando o *FaCT Reasoning Engine.* O Servidor de Ontologia registra com o *CORBA Naming Service* de que forma isso pode ser descoberto pelas outras entidades do ambiente.

Outra ferramenta importante da arquitetura chama-se *Ontology Explorer.* Ele é uma interface gráfica com o usuário que permite procurar ontologias no espaço e também, a interação dos mesmos usuários com outras entidades. Essas interações com outras entidades são governadas pelas propriedades dos usuários definidas na ontologia.

Pelo fato de o ambiente pervasivo ser muito dinâmico, novos tipos de entidades podem entrar no ambiente e sair dele a todo momento. Com o Servidor de Ontologia, é possível somar novas classes e propriedades para uma ontologia existente a qualquer momento, pela fusão de novos conceitos na ontologia do sistema. Para esse processo, primeiramente foi desenvolvida uma ontologia que descreve novas entidades. A nova ontologia é somada então às ontologias de compartilhamento que usam os chamados *conceitos de ponte* que relacionam classes e propriedades da nova ontologia às classes e propriedades das ontologias já existentes. Por exemplo, se um novo tipo de impressão digital é adicionado ao sistema, os conceitos de ponte podem declarar que ele é uma sub-classe dos "AuthenticationDevices".

Ontologias para informações de contexto

GAIA tem uma infra-estrutura de contexto que possibilita às aplicações obter e usar diferentes tipos de contexto. A infra-estrutura contém sensores que sentem vários contextos, *interpretadores* (reasoners) que deduzem novas informações de contexto sentidas e aplicações que fazem uso do contexto para adaptar as maneiras que eles se comportam.

Como as informações de contexto são descritas pelas ontologias, diferentes entidades que usam contextos têm uma compreensão comum da semântica dessas informações. Em GAIA, eles são usados como predicados, onde o nome do predicado é o tipo de contexto que está sendo descrito (como localização, temperatura ou tempo). Ontologias essencialmente definem os vocabulários e os tipos de argumentos que podem ser usados nos predicados. Elas são úteis para checar a validade das informações de contexto. Assim, torna mais fácil especificar o comportamento de aplicações de contexto, desde que se saiba os tipos de contextos que estão disponíveis e suas estruturas. Dessa forma, pode-se facilmente construir regras de comportamento da aplicação usando esses predicados de contextos bem definidos.

Existem diferentes tipos de contexto que podem ser utilizados pelas aplicações. Eles podem ser:

- Contextos físicos (localização e tempo);
- Contextos ambientais (tempo, luz e níveis de som);
- Contextos de informação (citações, placar de jogos);
- Contextos pessoais (saúde, humor, programa, atividade);
- Contextos sociais (atividade de grupo, relações sociais);
- Contextos de aplicação (e-mail, websites);
- Contextos de sistema (tráfico da Internet, status de impressoras).

O Servidor de Ontologia pode ser usado para qualquer aplicação, componente ou serviço no ambiente GAIA. As ontologias descrevem entidades e informações de contexto são usadas para permitir que diferentes partes do ambiente pervasivo interajam uma com a outra, facilmente.

Uso das Ontologias no Ambiente Pervasivo GAIA

O *Administrador de Configuração* (Configuration Management) tem a função de executar difíceis tarefas, pois, no dinâmico ambiente pervasivo, algumas situações podem ocorrer: (i) novas entidades nunca vistas podem entrar; (ii) componentes precisam se

descobrir e se colaborar automaticamente; (iii) entidades e componentes são heterogêneos e autônomos.

Ontologias formais aumentam a capacidade de usar descrições de diferentes fontes. As ontologias DAML+OIL podem ser publicadas para possibilitar progressão autônoma e provedores de serviço para seus produtos com o vocabulário correto. Entidades autônomas podem especificar o vocabulário formal correto a ser usado para interpretar as suas descrições se referindo para a ontologia DAML+OIL pertinente.

Outra vantagem do uso de ontologias é a melhoria na interoperabilidade entre entidades. A descrição das propriedades de diferentes classes de entidades permite que usuários e outros agentes automatizados interajam com elas mais facilmente, executando procuras ou emitindo a elas vários comandos. Isso provou ser uma das principais vantagens do uso de ontologias no ambiente da computação pervasiva, desde que isso simplifique ajudas aos usuários e a interação de agentes com tais sistemas complexos. Entidades diferentes permitem diferentes tipos de ações serem executadas nelas. Nesse *framework*, entidades especificam os comandos que eles suportam e os parâmetros desses comandos em uma ontologia.

O Servidor de Ontologia executa as tarefas de *descobridor de semânticas* (Semantic Discovery) e recursos. Ele propõe questões lógicas que envolvem suposição e classificação de conceitos para o *FaCT Server,* que tem conhecimento de todos os conceitos usados no ambiente. Tais questões são úteis para encontrar combinações apropriadas. Outras entidades no ambiente examinam o Servidor de Ontologia para descobrir classes de componentes que satisfazem suas exigências.

Uma ontologia bem aplicada melhora a robustez e portabilidade de aplicações conscientes do contexto. Não é possível adaptar em uma ontologia todos os contextos possíveis. Ontologias de informação de contexto são um importante mecanismo para que dispositivos se adaptem aos ambientes. A aplicação especifica regras para o comportamento sensível ao contexto usando um conjunto específico de conceitos de contexto e eventos (um vocabulário).

Quando uma aplicação se move de um espaço a outro, o contexto pode ser diferente, devido a diferentes sensores, diferentes versões de serviços, ou localizações. Se as diferenças são terminológicas, uma ontologia pode permitir que as regras sejam “traduzidas” e então trabalha corretamente no novo ambiente em questão. Aplicações sensíveis ao contexto em GAIA têm regras que descrevem que ações deveriam ser tomadas nos diferentes contextos. Para escrever uma regra, o desenvolvedor ou colaborador da aplicação deve saber os tipos

diferentes de contextos disponíveis como também as possíveis ações que podem ser tomadas pela aplicação. Em GAIA, existem ontologias capazes de descrever os diferentes tipos de informações de contexto (como localização, tempo, temperatura) e também diferentes aplicações e comandos que podem ser enviados a eles.

A ontologia simplifica bastante a tarefa de regras de escrita. Através dela, GAIA se torna uma ferramenta que permite um desenvolvedor escrever regras com facilidade. A ferramenta permite-lhe também, construir condições fora dos vários tipos possíveis de contextos disponíveis. Permite então escolher a ação a ser executada nesses contextos e a lista dos possíveis comandos que podem ser enviados para essa aplicação como descritas na ontologia. Assim sendo, os colaboradores podem atribuir o comportamento de aplicações sensíveis ao contexto (*context-aware*).

2.1.3 Comparativo entre as ontologias SOUPA e DAML+OIL

Neste trabalho, identificaram-se três tópicos principais que, atualmente, dificultam o desenvolvimento de sistemas voltados para Computação Pervasiva. São eles:

- *Descoberta de Recursos* (Discovery e Matchmaking);
- Interoperabilidade entre diferentes entidades;
- *Consciência de Contexto* (Context-awareness).

Esses tópicos foram citados também no Quadro 1, onde são descritas as principais características das ontologias SOUPA e DAML+OIL.

Aspectos comparativos	**SOUPA**	**DAML+OIL**
Significado	*Standard Ontology for Ubiquitous and Pervasive Aplications*	*Darpa Agent Markup Language - Ontology Inference Layer*
Objetivo	Padronizar para dar suporte a aplicações pervasivas que utilizam ontologias	Descrever relacionamentos entre objetos

Ontologias-base	- Adaptação da OWL - Uso de várias outras ontologias: FOAF, Rei Policy Ontology, COBRA-ONT, MoGATU BDI Ontology, OpenCyc Spatial Ontologies & RCC e DAML-Time & the Entry Sub-ontology of Time	Uma extensão da ontologia RDF para a representação de ontologias
Padronização	Grupo internacional *UbiComp Special Interest Group*	*DARPA* e *OIL Consortium*
Arquitetura da ontologia	- SOUPA Core - SOUPA Extention	- Dominio tipo de dado, dominio objeto - Expressão de classe: nome da classe em URI
Importação de ontologias	Não, uso direto formando nova ontologia	Sim, uso das linguagens RDF e RDF Shema
Interoperabilidade com outras ontologias	Sim, uso do padrão OWL.	Não, uso de propriedades e classes da própria ontologia
Conceitos/classes da ontologia	Classes que descrevem características de pessoas, comportamentos, estados, etc (pessoa, agentes, BDI, ação, eventos, política, tempo e espaço)	Eles podem ter objetos do tipo: *headers*, *elementos de classe* (class elements), *elementos da propriedade* (property elements) e *instâncias*
Informações de Contexto descritas por ontologias	Sim, definidos pelas classes, onde suas instâncias descrevem as informações de contexto da ontologia	- Sim, uso de predicados, onde o nome do predicado é o tipo de contexto - Contextos físicos e ambientais, contextos de informação, contextos pessoais e sociais, contextos de aplicação, contextos de sistema

Entidades modeladas	- perfis de contatos de pessoas e grupos sociais - tempo - localização e contexto de localização - conhecimento e argumentação - segurança e controle de acesso	- entidades do sistema - informações de contexto
Arquitetura de aplicação da ontologia	Projeto COBRA (Context Broker Architecture), implementação de agentes, serviços e dispositivos que exploram informações de contexto em espaços ativos	Projeto GAIA, uma infra-estrutura destinada a *Espaços Inteligentes* (Smart Spaces)
Descoberta de recursos	Uso de ontologias com o *FIPA directory facilitator*	- Servidor de ontologias - *FaCT Reasoning Engine* - *CORBA Naming Service*
Consciência do contexto	Regras que descrevem que ações deveriam ser tomadas nos diferentes contextos	Assim como o SOUPA possui regras que descrevem que ações deveriam ser tomadas nos diferentes contextos
Extensível	Sim, o SOUPA Extention tem a finalidade de agregar novos vocabulários para suportar tipos específicos de domínios em aplicações pervasivas	Não, os possíveis vocabulários estão definidos na própria ontologia
Aplicações	*EasyMeeting,* infra-estrutura capaz de prover serviços sensíveis a contexto em uma conferência, auxiliando palestrantes e ouvintes	*Ontology Explorer,* interface gráfica que permite procurar ontologias no espaço *GAIA* e também, a interação dos usuários com outras entidades

Fusão de ontologias	Sim, através da ontologia OWL adotada como padrão	Sim, com o servidor de ontologias
Outras características	*Documento de Ontologia de Ação* e *Documento de Ontologia de Política*	*Ontology Explorer*

Quadro 1: Comparativo entre as ontologias SOUPA e DAML+OIL

Tendo em vista que a ontologia SOUPA foi projetada para atender espaços pervasivos, sua aplicação na área torna-se mais completa e eficaz comparada à ontologia DAML+OIL, que tem como objetivo inicial atender especificadamente a Web Semântica.

Como visto na comparação realizada (Quadro 1), em vários aspectos ambas ontologias se equiparam. Porém, os tópicos referentes à *interoperabilidade com outras ontologias* e *extensibilidade* reforçam a idéia de que SOUPA oferece uma maior variedade e qualidade de recursos.

Ambas arquiteturas, COBRA e GAIA, oferecem suporte as aplicações com *consciência de contexto*. Apresentam também ferramentas capazes de suportar um serviço fundamental em espaços pervasivos: a *descoberta de recursos*. Diante de tais características, pode-se concluir a importância dessas arquiteturas no processo de desenvolvimento de aplicações destinadas *a pervasidade.*

CONCLUSÃO

Este trabalho propôs apresentar uma visão geral sobre os diversos aspectos das ontologias relacionadas assim como arquiteturas para sua aplicação. Ao término, pode-se concluir que o uso de ontologias é de fundamental importância em aplicações para ambientes pervasivos. Adotando uma ontologia como padrão, facilita-se o desenvolvimento das aplicações e permite-se que o programador dê ênfase à implementação do sistema.

Tratando-se de ambientes pervasivos, ontologias e serviços semânticos representarão um papel chave no desenvolvimento de ferramentas mais sofisticadas para construção e administração desses ambientes.

O projeto de SOUPA visa à padronização de ontologias compartilhadas para a Computação Pervasiva. Ele apresenta uma variedade de classes e componentes inteligentes na sua estrutura, que descrevem contextos de agentes de forma satisfatória. Trata-se de uma linguagem robusta que subtrai de outras ontologias o que elas têm de melhor.

A ontologia DAML+OIL inicialmente dirigiu-se às questões referentes à Web Semântica. No entanto, como visto no trabalho, já existem arquiteturas que também aplicam essa ontologia na Computação Pervasiva. Ela provou ser bastante útil, especialmente em combinação com uma interface de programação. Porém, ontologias como o padrão DAML+OIL podem ser aplicadas com sucesso, mas não são suficientes para aplicações pervasivas, pelo fato de apresentar poucos recursos e componentes específicos para essa finalidade.

Propõem-se a partir desse trabalho um estudo mais aprofundado sobre ontologias e arquiteturas que suportam aplicações com *consciência de contexto*, tento em vista que atualmente o conteúdo referente ao assunto é limitado.

A Computação Pervasiva aplicada à nossa vida diária pode simplificar e muito nossas tarefas, por meio de um ambiente de acesso e troca de informações que envolvem os usuários. Ela representa o futuro da computação, porém para que este se torne realidade, são necessárias evoluções em alguns aspectos da computação atual. Dentre estes aspectos, fazem parte as ontologias, que devem oferecer um maior suporte aos componentes (como dispositivos, serviços, eventos) que compõem as aplicações destinadas a Ambientes Pervasivos.

REFERÊNCIAS BIBLIOGRÁFICAS

BONATTO, D. T.; BARBOSA, J. L. V.; CAVALHEIRO, G. G. H. **PHolo: Uma Arquitetura para a Computação Pervasiva Utilizando o Holoparadigma, 2005.** Disponível em: <www.inf.unisinos.br/~barbosa/textos/WSCAD_2005.pdf> Acesso em: 3 setembro. 2006.

CARNEIRO, M. R. F. **Ontologias, Web Semântica e Aplicações, 2003.** Disponível em: <http://www.ime.usp.br/~yoshi/2003i/mac5701/> Acesso em: 20 agosto. 2006.

CHEN, H.; PERICH, F.; FININ, T.; JOSHI, A. **SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, 2006.** Disponível em: <http://ebiquity.umbc.edu/_file_directory_/papers/105.pdf> Acesso em: 15 outubro. 2006.

CONNOLLY, D.; HARMELEN, F. V.; HORROCKS, I.; MACGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A. **DAML+OIL (March 2001) Reference Description, 2001.** Disponível em: <http://www.w3.org/TR/daml+oil-reference#Mixing> Acesso em: 30 outubro. 2006.

DZIEKANIAK, G. V.; KIRINUS, J. B. **WEB SEMÂNTICA, 2004.** Disponível em: <http://www.encontros-bibli.ufsc.br/Edicao_18/2_Web_Semantica.pdf> Acesso em: 20 agosto. 2006.

FERREIRA, A. B. H. **Novo Dicionário da Língua Portuguesa.** Rio de Janeiro: Nova Fronteira, 1996.

FILHO, J.V. **Implementação de Aplicações Sensíveis a Contexto Utilizando Sistemas Multi-Agentes, 2005.** Disponível em: <http://www-di.inf.puc-rio.br/~endler/courses/Mobile/Monografias/05/Viterbo-Mono.pdf> Acesso em: 10 setembro. 2006.

GEYER, C. F. R. **GT 04: Computação em Grade Pervasiva - GRADEp, 2004.** Disponível em: <www.inf.ufrgs.br/~lucc/gradep/rel-P21-gt-gradep-nov04-v7.doc> Acesso em: 15 setembro. 2006.

GRUBER, T. **A Translation Approach to Portable Ontologies.** Knowledge Acquisip.199-200, 1993.

GRUBER, T. **Toward principles for the design of ontologies.** Knowledge Sharing. 43:907-928, 1995.

JUNIOR, R. A. M. **Uma Ontologia para Engenharia de Requisitos de Software, 2003.** Disponível em: <bibliotecadigital.sbc.org.br/download.php?paper=298>. Acesso em: 1º junho. 2006.

LIBRELOTTO, G. R.; RAMALHO, J. C.; HENRIQUES, P. R. **Representação de Conhecimento na Semantic Web, 2005.** Disponível em: <http://www.sbc.org.br/bibliotecadigital/?module=Public&action=SearchResult&author=252> Acesso em: 15 setembro. 2006.

MCGRATH, R. E.; RANGANATHAN, A.; MICKUNAS, M. D.; CAMPBELL, R. H. **Investigations of Semantic Interoperability in Ubiquitous Computing Environments, 2003.** Disponível em: <http://www.actapress.com/PaperInfo.aspx?PaperID=13884> Acesso em: 2 novembro. 2006.

NARDON, F. B. **Compartilhamento de Conhecimento em Saúde Utilizando Ontologias e Banco de Dados Dedutivos, 2002.** Disponível em: <http://www.tridedalo.com.br/fabiane/publications/i2ts.pdf> Acesso em: 22 agosto. 2006.

NOVELLO, T. C. **Ontologias, Sistemas baseados em Conhecimento e Modelos de Banco de Dados**, **2003**. Disponível em: <http://www.inf.ufrgs.br/~clesio/cmp151/cmp15120021/artigo_taisa.pdf> Acesso em: 12 junho. 2006.

PINHEIRO, M. J. **GRADEp-RP: Emprego das Tecnologias do Projeto GRADEp na Agrometeorologia, 2006.** Disponível em: <http://pg.ucpel.tche.br/doku.php#documentos >Acesso em: 6 agosto. 2006.

SILVA, G. H. **Construção de Agentes Inteligentes para a Web Semântica, 2003.** Disponível em: <http://www.linux.ime.usp.br/~cef/mac499-04/monografias/ghsilva/> Acesso em: 20 Junho. 2006.

VIEIRA, R.; SANTOS, A.; SILVA, D. M.; SANTANA, M. R. **Web Semântica: ontologias, lógicas de descrição e inferência, 2005.** Disponível em: <http://www.inf.unisinos.br/~renata/laboratorio/publicacoes/webmedia-webs.pdf> Acesso em: 22 agosto. 2006.