Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Curso Bacharelado em Ciência da Computação
Trabalho de Conclusão de Curso

Vagner Nogueira Silva

Modelagem de Base de Conhecimentos Baseada em Ontologia

Estudo de Caso em Recuperação de Informação Bibliográfica

Belém
2010

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Curso Bacharelado em Ciência da Computação
Trabalho de Conclusão de Curso

Vagner Nogueira Silva

Modelagem de Base de Conhecimentos Baseada em Ontologia – Estudo de Caso em Recuperação de Informação Bibliográfica

Trabalho de Conclusão de Curso apresentado para a obtenção do grau de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. Adagenor Lobato Ribeiro

Belém
2010

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Faculdade de Computação
Curso Bacharelado em Ciência da Computação
Trabalho de Conclusão de Curso

Vagner Nogueira Silva

Modelagem de Base de Conhecimentos Baseada em Ontologia – Estudo de Caso em Recuperação de Informação Bibliográfica

Monografia apresentada para a obtenção do grau de Bacharel em Ciência da Computação.

Data da defesa: 17 de Junho de 2010.
Conceito:

Banca Examinadora

Prof. Dr. Adagenor Lobato Ribeiro
Faculdade de Computação/UFPA - Orientador

Prof. Dr. Antonio Morais Silveira
Faculdade de Computação/UFPA - Membro

Prof. Dr. Francisco Edson Lopes da Rocha
Faculdade de Computação/UFPA - Membro

À minha mãe, minha mulher, demais familiares e amigos.

AGRADECIMENTOS

Agradeço à Deus pela força que me concedeu para superar as dificuldades e provações da vida. À minha mãe pelo amor e dedicação. Aos Professores da Faculdade de Computação pelas lições e aprendizado, em especial o Prof. Dr. Francisco Edson pelos conselhos e pela dedicação à Faculdade. Ao meu honorífico orientador pela paciência e tempo dispensados e pela confiança depositada. Ao Professor Morais por nos ter concedido a honra de sua participação em banca de defesa. E, finalmente, aos meus amigos pelo companheirismo.

"Quando vos mandei sem bolsa, alforge ou sandálias, faltou-vos, porventura, alguma coisa? Eles responderam: Nada."

Lucas 22.35-38

Sumário

Lista de Figuras

Lista de Quadros

Resumo

A Web Semântica foi idealizada por Tim Berners-Lee, o inventor da web atual, para estender a web que se conhece atualmente, com o objetivo de tornar o conteúdo da web entendível por agentes computacionais. A partir dessa proposta o consórcio W3C (*World Wide Web Consortium*) trabalha no desenvolvimento dos padrões que são utilizados nesta nova versão da web. O principal pilar da web semântica são as Ontologias, que podem formar bases de conhecimento complexas, sobre as quais agentes de software, possam "raciocinar"e "inferir", programaticamente, novos conhecimentos. A linguagem OWL é o padrão desenvolvido e recomendado pelo W3C para representação de Ontologias na web. Embora os avanços em pesquisas e desenvolvimento nesta área tenham sido significativos, a arquitetura da Web Semântica ainda não foi concluída. Observa-se que ocorre pouca popularização dos conceitos e padrões da Web Semântica entre desenvolvedores de sistemas web, pois apenas a indústria de ponta e academia têm implementado sistemas nesta área. Este trabalho, traz a web semântica, no seu atual estado, do campo teórico para a prática da implementação de sistemas, tratando acerca da modelagem de base de conhecimentos baseado em Ontologia. Como estudo de caso, modelou-se uma base de conhecimentos sobre livros, objetivando a recuperação de informação bibliográfica. Para implementação do protótipo do sistema de estudo de caso, utilizou-se a especificação JEE6 em conjunto com o *framework* Jena. Efetuou-se um estudo acerca da Web Semântica; padrões da mesma; engenharia ontológica com o método 101; a linguagem OWL e o ambiente *Protégé*.

Palavras-chave: Web Semântica, Ontologia, OWL, *framework* Jena.

Abstract

The Semantic Web was envisioned by Tim Berners-Lee, inventor of the current web, for extend the Web that is currently known, with the goal of making Web content understandable by computational agents. On this basis the consortium W3C (World Wide Web Consortium) works to develop standards that are used in this new version of the web. The main pillar of the semantic web are ontologies, which can form the basis of knowledge complex, about which software agents, to "reasoning"and "inferred,"programmatically, foreground. The OWL is the standard developed and recommended by the W3C for representing ontologies on the web. While advances in research and development in this major have been significant, the architecture of the Semantic Web has not yet been completed. It is observed that occurs little popularization of the concepts and standards of the Semantic Web developers from Web systems because only the high-tech industry and academia have implemented systems in this major. This work brings the Semantic Web in its current state, the theoretical to the practical implementation of systems, addressing modeling based on the knowledge-based ontology. As a case study, modeled itself a knowledge base about books aimed at information retrieval literature. To implement the prototype system case study, we used the specification JEE6 together with the framework Jena. We conducted a study on the Semantic Web, the same standards, ontological engineering with the method 101, the Protégé OWL language and environment.

Keywords: Semantic Web, Ontology, OWL, Jena Framework.

1 Introdução

1.1 Contextualização

Os conceitos primordiais para a Web Semântica foram expostos por Tim Bernes Lee, James Hendler e Ora Lassila, em um artigo da revista *Scientifc American*, intitulado "*The Semantic Web*", publicado em Maio de 2001. Com uma linguagem clara e simples, os pesquisadores propuseram uma nova forma de organização das informações contidas na web, de maneira que ela possa se tornar inteligível, não só por humanos, mas também por agentes computacionais. Isso possibilita pesquisa, compartilhamento e integração de informações, por softwares, através da web, com maior eficiência e facilidade. A web semântica será, efetivamente, um meio universal para intercâmbio de dados, informação e conhecimento, possibilitando que seus usuários interajam com os recursos computacionais de maneira muito mais eficaz.

Desta maneira, foi concebida a Web Semântica, também conhecida como Web 3.0 ou Web dos dados. Ela é uma extensão da web atual, a web dos documentos, uma evolução, e não uma substituta para a web que se tem atualmente.

A partir destas ideias iniciais, o W3C[1] , têm trabalhado em cima destes conceitos, no sentido de propor padrões para a Web Semântica. Apesar dos esforços do W3C, a arquitetura da Web Semântica ainda não está completa, mas os avanços têm sido significativos.

O principal pilar da web semântica são as Ontologias, que podem formar bases de conhecimento complexas, sobre as quais agentes de software, possam "raciocinar"e "inferir", programaticamente, novos conhecimentos.

Mesmo com a arquitetura ainda inacabada, tanto a academia quanto a alta indústria, têm produzido diversos sistemas semânticos, Ontologias e bases de conhecimentos baseado em Ontologia. Destaca-se a academia da área das ciências da saúde e das ciências

[1] O *World Wide Web Consortium* (ou *W3C Consortium*), criado em outubro de 1994, tem como principal objetivo liderar o desenvolvimento de protocolos comuns para assegurar a interoperabilidade na Internet. Possui mais de 500 organizações membros que contribuem para o crescimento da Web.

biológicas, que muito têm utilizado os padrões já definidos para a Web Semântica. Entretanto, percebe-se pouco interesse, pela Web Semântica, tanto no sentido de pesquisa como no sentido de produção de sistemas que utilizam o arcabouço da Web Semântica, ou bases de conhecimentos baseadas em Ontologia. Uma simples comparação com outras áreas da ciência da computação, fornece a clara confirmação deste fato.

Este trabalho vem mostrar como se pode, facilmente, implementar sistemas utilizando as tecnologias e padrões, já definidos na arquitetura da web semântica em conjunto com outras tecnologias, e a implementação de um protótipo de sistema de base de conhecimentos para recuperação de informação bibliográfica.

A justificativa para a escolha desse tema para a implementação do estudo de caso reside no fato de haver grande dificuldade em se obter informações sobre o domínio de conhecimento em questão. A web tem muitas informações desorganizadas sobre o domínio de livros, qualquer pessoa que tem interesse em fazer alguma pesquisa sobre, por exemplo, o preço médio de um determinado livro em sua cidade, utilizando apenas a web, terá muito trabalho, descartando dados irrelevantes e fazendo cruzamento dos dados relevantes.

1.2 Objetivos

1.2.1 Gerais

Lista-se, a seguir os **objetivos gerais** deste trabalho.

- Trazer alguns padrões, teoria e filosofia da Web Semântica, do campo teórico, para a prática da implementação de sistemas, no que diz respeito à modelagem de bases de conhecimentos baseadas em Ontologia.
- Desenvolver uma metodologia de implementação, que com pouco esforço, resulta num sistema que processa uma base de conhecimentos baseada em Ontologia.
- Contribuir para a popularização dos conceitos e prática da Web Semântica.
- Implementar, como estudo de caso, um protótipo de sistema de recuperação de informações bibliográficas, objetivando "raciocinar", programaticamente em cima de uma base de conhecimentos sobre livros, permitindo ao usuário obter informação relevante sobre o domínio coberto pela Ontologia utilizada.

1.2.2 Específicos

Este trabalho tem como objetivos específicos, os listados a seguir.

- Formatar o embasamento teórico para entendimento dos objetivos deste trabalho.
- Mostrar a utilização o método 101 e o ambiente *Protégé*[2] para a modelagem da Ontologia do estudo de caso.
- Utilizar a arquitetura JEE6[3], em conjunto com a API Jena[4] para atingir os objetivos de implementação do protótipo acima citado.

1.3 Estrutura do trabalho

No capítulo seguinte (cap. 2) foi feito um passeio pelos conceitos da web semântica e o conceito de Ontologia, objetivando embasamento teórico. No capítulo 3 foi feito um resumo teórico acerca dos conceitos da linguagem OWL. No capítulo 4 falou-se sobre a modelagem da informação, engenharia ontológia com o método 101, uma introdução ao ambiente *Protégé* e a modelagem da Ontologia de estudo de caso. O capítulo 5 trata acerca da implementação da base de conhecimento, onde se fez um resumo sobre as tecnologias utilizadas. Posteriormente, descreve-se a implementação do sistema estudo de caso. Finaliza-se com as considerações finais (cap. 6), onde se avalia os resultados e sugere-se trabalhos futuros.

[2] `http://protege.stanford.edu/`
[3] `http://java.sun.com/javaee/technologies/`
[4] `http://jena.sourceforge.net/`

2 Web Semântica e Ontologia

2.1 Web Semântica

A Web Semântica foi idealizada por Tim Bernes-Lee, criador da web atual e líder do W3C. Segundo W3C (2010b), o termo "Web Semântica"refere-se à visão do W3C de Web dos dados vinculados, ou interligados. Tecnologias Web Semânticas permitem que pessoas criem armazenamento de dados na Web, construam vocabulários, e escrevam as regras para a manipulação de dados. A vinculação de dados são permitidas por diversos padrões de tecnologias desenvolvidos pelo W3C para a Web Semântica.

É importante, neste ponto, deixar claro o significado, para este trabalho relevante, de dado e informação. De acordo com Rosa (2002) a diferença entre dado e informação reside no entendimento de seu conteúdo: "A palavra 'dado' refere-se a todo texto que não tem significado embutido, ou seja, que não pode ser entendido. Já a palavra 'informação' refere-se a todo o texto no qual o leitor é capaz de entender seu significado".

A proposta da Web Semântica é, então possibilitar que agentes de software executem tarefas complexas na web. Tarefas que não são possíveis de serem efetuadas por eles atualmente, como transformar dados em informações relevantes para o usuário, realizar inferências sobre informações e realizar consultas complexas, do tipo "informe que livraria com endereço na cidade de Belém do Pará vende o livro de título Engenharia de Software do autor *Pressman* e depois informe os preços que estas livrarias vendem este livro".

Um dos passos para esta automatização proposta, consiste em "etiquetar"os dados da web atual com metadados. Segundo, Rosa (2002), metadados são dados que descrevem o conteúdo, a estrutura, a representação e o contexto de algum dado ou conjunto de dados. Pode-se citar, como exemplo de metadados, o catálogo de peças de uma oficina mecânica, onde se pode encontrar dados sobre peças de veículos.

2.2 Arquitetura da Web Semântica

Foi projetada uma arquitetura em camadas para que a Web Semântica possa se realizar em plenitude. Cada camada apresenta uma série de tecnologias com seus respectivos padrões propostos pelo W3C. Como se pode observar na Figura 1.

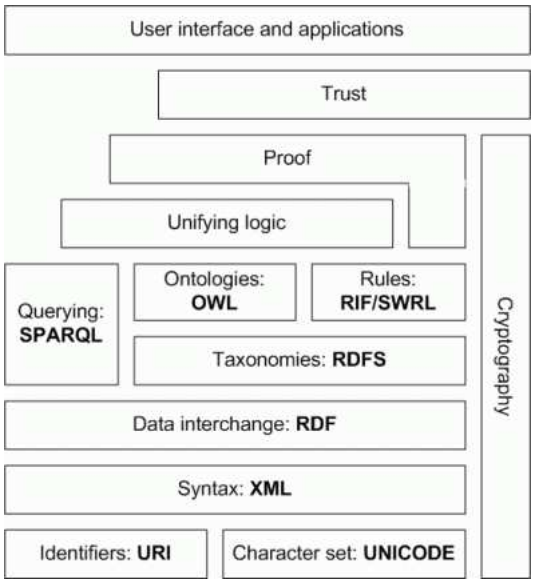


Figura 1: Arquitetura de camadas da Web Semântica e suas respectivas tecnologias.
Fonte: `http://semanticweb.org/images/3/37/Semantic-web-stack.png`

De acordo com Rosa (2002), "cada linguagem é construída baseada nas linguagens definidas em níveis inferiores, sendo os níveis de URI e Unicode os mais fundamentais, pois tratam da representação de caracteres e recursos. A camada XML define a sintaxe de todas as linguagens das camadas superiores".

A seguir será dada uma visão geral acerca de cada camada da arquitetura da Web Semântica.

2.2.1 Unicode e URI

Unicode[1] é um padrão para representação de caracteres. Segundo unicode.org (2009), Unicode fornece um número único para cada caractere, não importa qual a plataforma, o programa, o idioma. Para a Web Semântica o Unicode é o padrão para escrita de caracteres, desta forma parte do requisito de universalidade da informação se torna satisfeito.

URI[2](*Uniform Resource Identifier*) é o padrão para identificar recursos na web. Uma URI é uma cadeia de caracteres que identifica uma página HTML, uma imagem, um

[1]http://www.unicode.org/
[2]`http://www.w3.org/Addressing/`

arquivo, um serviço e qualquer outro recurso, disponibilizado via web.

2.2.2 XML

A camada XML é composta pelos padrões XML, XML *Schema* e XML *NameSpaces*, descritos a seguir.

XML[3] (*Extensible Markup Language*) é uma linguagem de marcação de texto simples e muito flexível, derivada da SGML [4] (ISO 8879).

Segundo W3C (2003) "XML foi originalmente concebida para responder aos desafios de grande escala da publicação eletrônica. XML também desempenha um papel cada vez mais importante na troca de uma ampla variedade de dados na Web e em outros lugares".

XML é a base de todas as linguagens da Web Semântica que expressam significado. O XML definirá as estruturas destas linguagens.

Lista-se, a seguir, alguns benefícios oferecidos pela linguagem XML, citados por Rosa (2002):

- Linguagem independente de fornecedor e de fácil compreensão;
- Redução de custos de treinamento e desenvolvimento devido ao formato simples;
- Linguagem legível por humanos e por agentes de software;
- Aumento da confiança devido à automatização dos agentes de software;
- Trata-se da base de apoio da Web Semântica na qual proporciona interoperabilidade;
- Liberdade na criação das estruturas por usuários;
- Permite a reutilização de dados sem nenhuma ligação com ferramentas proprietárias ou formatos não documentados.

A linguagem XML utiliza marcações, chamadas de *tags*, para definir informações. Cada informação deve ser precedida pela *tag* de incío < *tag*> e sucedida pela *tag* de fim </ *tag*>. Cada *tag* deve ser colocada entre colchetes angulares "<"e ">"com seu identificador (*element*) e pode ter ou não atributos (*attributes*). Exemplo de um documento XML:

[3] `http://www.w3.org/XML/`

[4] SGML (*Standard Generalized Markup Language*) é um padrão internacional de processamento de texto com marcação. O documento ISO da SGML pode ser encontrado em: `http://www.iso.org/iso/cataloguedetail.htm?csnumber=16387`

```
<lista id="banca">
<item>Adagenor</item>
<item>Edson</item>
<item>Antonio</item>
</lista>
```

No exemplo acima, lista e item são elementos, id é um atributo.

XML *Schema* é uma linguagem de definição de classes de documentos XML. Define e descreve a estrutura e os conteúdos de classes de documentos XML, ou seja, a gramática para a validação da estrutura do documento XML.

Rosa (2002) complementa que um documento XML bem formado é aquele no qual está sintaticamente correto na visão de XML. Entretanto, um documento XML válido é aquele que segue as especificações do seu documento XML Schema correspondente.

XML *Namespace*[5], é definido por Bray et al. (2009) como uma coleção de nomes, identificados por uma referência URI, que é utilizada em documentos XML, como tipos de elementos e nomes de atributos, possibilitando combinação de documentos XML com vocabulário controlado. XML *Namespace* possibilita, também, o compartilhamento e reutilização da definição de outros esquemas XML sem que haja conflito de identificadores de elementos.

2.2.3 RDF

A camada RDF[6] é composta pelos padrões RDF e RDF *Schema*, descritos a seguir. RDF(*Resource Description Famework*), segundo W3C (2009), é um modelo padrão para o intercâmbio de dados na web. RDF tem características que facilitam a fusão de dados, mesmo se os esquemas subjacentes diferem, e apoia especificamente a evolução dos esquemas ao longo do tempo sem a necessidade de todos os consumidores de dados serem alterados.

RDF estende a estrutura de interligação da Web para usar URIs para nomear a relação entre coisas, bem como as duas extremidades do link (o que normalmente é referenciado como uma "tripla"). Usando este modelo simples, ela permite que dados estruturados e semi-estruturados possam ser misturados, expostos e compartilhados entre aplicações diferentes.

[5]Bray et al. (2009)
[6]http://www.w3.org/RDF/

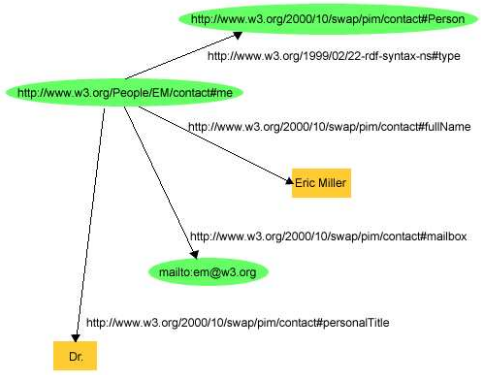


Figura 2: Exemplo de grafo RDF.
Fonte: `http://www.w3.org/TR/rdf-primer/fig1dec16.png`

O link estrutural, em uma tripla RDF, relacionando dois recursos, forma um grafo direcionado, rotulado, onde as arestas representam o link nomeado entre dois recursos, representados pelos nós do grafo. Esta visão de grafo é o modelo mental mais fácil possível para RDF e é frequentemente usado para entendimento fácil e explicação visual, a qual exemplificamos com a Figura 2.

RDF *Schema*[7], similarmente à XML *Schema*, é uma linguagem que define uma estrutura válida para documentos RDF.

2.2.4 Camada de Ontologia

A camada de Ontologia contém as tecnologias de representação de conhecimento com poder de expressividade maior do que a RDF pode proporcionar. As linguagens de Ontologias permitem que se escrevam conceituações formais e explícitas de modelos de domínio, empregando, como requisitos básicos, uma sintaxe bem definida, uma semântica formal e alto nível de expressividade, fornecendo apoio eficiente ao processo de inferência de significado. A camada de Ontologia na arquitetura da Web Semântica está bem estável e amadurecida, assim como as camadas abaixo dela, vide Figura 1. As camadas a cima da camada de Ontologia ainda não apresentam padrões definidos ou recomendações, apenas algumas poucas propostas.

O enfoque maior nesta camada está na seção 2.3.

[7] `http://www.w3.org/TR/rdf-schema/`

2.2.5 Lógica

A camada de lógica é composta, basicamente, por regras de inferência, com as quais os agentes poderão processar e relacionar a informação.

2.2.6 Prova

Camada que objetiva determinar a consistência da informação inferida das camadas anteriores.

2.2.7 Confiança

Camada que objetiva garantir a autenticidade e confiabilidade das fontes de informação pelos agentes.

2.3 Ontologia

Segundo o Novo Dicionário Aurélio versão 5.0, o verbete Ontologia, tem o seguinte significado:

> "[De *ont(o)-* + *-logia.*] Substantivo feminino. 1.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."

O termo deriva do grego "*onto*", ser, e "*logia* ", discurso escrito ou falado. Segundo Lima e Carvalho (2005), este termo tem sido usado, ao longo da história, pela Filosofia Grega (Aristóteles 384-322 a.C.) e é uma teoria sobre a existência da natureza, sobre que tipos de coisas existem ou o que se dizer sobre o mundo. O termo foi emprestado então para a ciência da computação passando a ter outro sentido.

As definições mais relevantes para o contexto deste trabalho são:

> Ontologia é uma especificação formal e explícita de uma conceitualização, o que existe é aquilo que pode ser representado [...]Quando o conhecimento de um domínio é apresentado num formalismo declarado, o conjunto de objetos que podem ser representados são chamados de universo do discurso. Esse conjunto de objetos, e o relacionamento descritivo entre eles, são refletidos num vocabulário representacional com o qual um programa de conhecimento de base representa o conhecimento. Mas, no

> contexto de Inteligência Artificial, nós podemos descrever a Ontologia de um programa pela definição de um conjunto de termos representacionais. Nesse tipo de Ontologia, definições associam os nomes de entidades no universo do discurso (por exemplo, classes, relações, funções ou outros objetos) com textos legíveis descrevendo o que os nomes significam, e axiomas formais que limitam a interpretação e o uso bem formado desses termos. Formalmente, uma Ontologia é uma afirmação da lógica teórica. (GRUBER, 1993).

Berners-Lee, Hendler e Lassila (2001):

"É um documento que define as relações entre termos e conceitos."

Lima e Carvalho (2005), também explica que:

> Uma Ontologia conceitua um modelo abstrato de algum fenômeno do mundo em um conhecimento consensual, isto é, compartilhado por todos. Além disso, os conceitos, as propriedades, as funções, os axiomas devem ser especificados explicitamente e serem manipuláveis por computador. (LIMA; CARVALHO, 2005).

Ainda citando Lima e Carvalho (2005): "a relevância das Ontologias para a Web deve ser tratada sobre três aspectos: identificação de contexto [...]; fornecimento de definições compartilhadas e reuso".

Destaca-se, neste ponto, a definição de ontologia mais clara, na visão do autor deste trabalho, e relevante para o contexto deste trabalho, por ser uma definição bem prática de ontologias e que, de certo modo, resume as acima citadas, é a de Noy e McGuinness (2001), a qual cita-se com algumas paráfrases, mas com a essência conservada:

> "[...] Uma Ontologia é uma descrição formal e explícita dos **conceitos** em um domínio de discurso (classes[...]), **propriedades** de cada conceito que descreve várias características e atributos dos conceitos [...], e **restrições**" dessas propriedades. "Uma Ontologia Define um **vocabulário comum** e um **entendimento compartilhado**" acerca de um domínio de conhecimento.

Para que se possa utilizar Ontologias no contexto da Web Semântica é necessário que se tenha disponível uma linguagem para escrever um documento que conterá a Ontologia. Em vista disso, o W3C desenvolveu a OWL *Web Ontology Language*[8] como padrão para representação de Ontologias.

[8] `http://www.w3.org/2004/OWL/`

> A OWL é uma revisão da linguagem DAML+OIL. Ela possui mais facilidades para expressar significados e semânticas do que XML, RDF e RDF Schema, embora seja baseada em RDF e RDF Schema e utilize a sintaxe XML. A OWL foi projetada para ser usada por aplicações que necessitem processar o conteúdo de informações, ao invés de somente apresentar a visualização destas informações. (LIMA; CARVALHO, 2005).

Trata-se, mais detalhadamente, sobre OWL no capitulo 3.

2.3.1 Componentes de uma Ontologia

Baseando-se em Horridge (2009) lista-se os componentes de uma Ontologia, a seguir:

- **Indivíduos** (também chamados de *individuals*, *instances* ou instâncias de classes): são os objetos no domínio de interesse ou domínio do discurso da Ontologia.
- **Propriedades** (conhecidas também como *properties*, regras, *rules* ou *slots*): são definidas como relações binárias entre indivíduos. Em lógica descritiva as propriedades são chamadas de papéis (*roles*) e em UML de relações (*relationships*).
- **Classes**: São conjuntos que contêm os indivíduos. São descritas formalmente, através de descrições matemáticas, de forma que sejam bem definidos os requisitos para a participação de um indivíduo na classe. As classes podem formar cadeias em hierarquias superclasses-subclasses, as subclasse herdam todas as características de suas superclasses. Chama-se esta hierarquia de **taxonomia**. Subclasses também são conhecidas como especializações de suas superclasses. O termo conceito (*concept*) é as vezes usado no lugar de classe. Porém, as classes são, na verdade, representações concretas de conceitos.

2.3.2 Tipos de Ontologias

Para o escopo deste trabalho é suficiente distinguir as Ontologias de acordo com seu nível de generalidade, de acordo com Guarino (1998):

- **Ontologias de alto nível**: descrevem conceitos de forma bem geral (como espaço, tempo, material, objeto, evento, ação), os quais são independentes de um problema ou domínio particular.
- **Ontologias de domínio e Ontologias de tarefas**: descrevem, respectivamente, o vocabulário relacionado a um domínio genérico (como medicina ou automóveis) ou uma tarefa genérica (como diagnóstico ou vendas).

- **Ontologias de aplicação**: descrevem conceitos dependendo do domínio e de tarefas particulares. Estes conceitos, frequentemente, correspondem a papéis desempenhados por entidades do domínio, quando na realização e certas tarefas. Por exemplo, ajuda para o diagnóstico de doenças mentais.

3 OWL *Web Ontology Language*

3.1 Definição e conceitos iniciais

OWL Web Ontogy Language é o padrão proposto pelo W3C como uma linguagem para a Web Semântica, projetada para representar Ontologias. Sua última versão, a OWL 2, foi lançada em 27/10/2009 e é uma extensão da primeira versão, adicionando novas características e funcionalidades à primeira versão. Neste trabalho, foca-se a primeira versão da especificação da linguagem OWL, por motivos de praticidade, já que a nova versão está muito recente e existem no mercado poucas ferramentas que suportam completamente a OWL 2. Este capítulo foca os conceitos para se compreender os elementos e regras necessárias à construção de uma Ontologia em OWL, não entrando nos detalhes dos construtos da linguagem. Na seção 4.4 se faz uma introdução ao ambiente *Protégé*, que tem ferramentas necessárias para se construir um documento OWL.

Hitzler et al. (2009) define OWL como uma linguagem computacional baseada em lógica, de tal forma que o conhecimento expresso nela pode ser "raciocinado" por programas de computador, tanto para verificar a consistência desse conhecimento, como para tornar explícito o conhecimento implícito. Ontologias OWL podem ser publicadas na web e referenciadas por outras Ontologias.

OWL é declarativa, não é uma linguagem de programação. Ela descreve um estado de assuntos num caminho lógico. Não é uma linguagem de esquema para conformidade de sintaxe. Diferente de XML, OWL não provê meios para descrever como um documento pode ser estruturado sintaticamente.

OWL não é um *framework* para base de dados. De modo notório, documentos OWL armazenam informação assim como fazem os bancos de dados. Entretanto, conteúdos de bancos de dados, são organizados de acordo com as suposições (ou raciocínio) de mundo fechado (*closed-world assumption/reasoning*). Isto significa que se algum fato não está presente na base de dados ele é dado como falso. Enquanto que documentos OWL seguem as suposições (ou raciocínio) de mundo aberto (*open-world assumption/reasoning*).

OWL na versão 1 apresenta três sub-linguagens, conforme Hitzler et al. (2009) lista:

- **OWL-Lite**: Sub-linguagem sintaticamente mais simples. Destina-se a situações em que apenas são necessárias restrições sobre uma hierarquia de classe simples, como tesauros.
- **OWL-DL**: Mais expressiva que a OWL-Lite e baseia-se em Lógica Descritiva[1], um fragmento de Lógica de Primeira Ordem[2], passível portanto de "raciocínio" computável. Possibilita computar automaticamente a hierarquia de classes e verificar inconsistências na Ontologia;
- **OWL-Full**: Sub-linguagem OWL mais expressiva. Destina-se a situações onde alta expressividade é mais importante do que garantir a decidibilidade ou completeza da linguagem. Não é possível efetuar inferências em Ontologias OWL-Full.

Considera-se algumas heurísticas quanto à escolha da sub-linguagem, ainda citando Hitzler et al. (2009):

- Entre OWL-Lite e OWL-DL, é necessário saber se os construtos da OWL-Lite são suficientes;
- Entre OWL-DL e OWL-Full, é preciso saber se é importante realizar inferências na Ontologia, ou se é importante usar funcionalidades altamente expressivas ou funcionalidades de modelagem, tais como as meta-classes (classes de classes).

Os conceitos seguintes fazem parte do universo OWL:

- Os **MI** (Mecanismos de Inferência ou *reasoners*) são também chamados de classificadores (*classifiers*);
- A tarefa de calcular a hierarquia de classes inferida é também conhecida como classificação da Ontologia (*classifying the ontology*);
- É uma característica do OWL-DL que o relacionamento superclasse-subclasse possa ser computado automaticamente por um MI;

[1] A Lógica Descritiva modela os conceitos, regras e indivíduos e suas relações através de axiomas lógicos.

[2] Lógica de Primeira Ordem(LPO) ou Lógica de Predicados é a parte da Lógica Proposicional que analisa os objetos do universo e as relações entre eles.

- **Classes disjuntas**: Um indivíduo qualquer não pertence à duas classes disjuntas. Considera-se que as classes OWL se sobrepõem. Por isso, não se pode assumir que um indivíduo não é um membro de uma classe específica, simplesmente porque não se declarou que ele é um membro daquela classe. Para desconectar um grupo de classes é preciso torná-las disjuntas. Isto garante que um indivíduo que tenha sido declarado como sendo membro de uma das classes do grupo, não pode ser um membro de nenhuma outra classe naquele mesmo grupo;

- Em uma hierarquia de classes nomeadas, ser uma subclasse significa uma implicação necessária, ou seja, para uma instância de uma subclasse ser dita da referida subclasse, é necessário que essa instância também pertença à superclasse de sua classe;

- OWL não usa o **UNA** (*Unique Name Assumption*). Por isso, deve-se declarar explicitamente que os indivíduos são os mesmos (*SameAs*), ou diferentes (*DifferentFrom*) uns dos outros;

- **Classes enumeradas** (*Enumerated Classes*) são classes definidas pela lista dos indivíduos que são seus membros.

- Complemento (not) de uma classe inclui todos os indivíduos que não são membros da classe.

- OWR - *Open World Reasoning* (Raciocínio de Mundo Aberto): As inferências em OWL (Lógica Descritiva) se baseiam na OWA-*Open World Assumption* (Suposição de Mundo Aberto), também conhecida como OWR-*Open World Reasoning* (Raciocínio de Mundo Aberto). A Suposição de Mundo Aberto significa que não se pode assumir que alguma coisa não existe, até que seja estabelecido explicitamente que ela não existe. Em outras palavras, porque alguma coisa não foi definida como verdade, não significa que ela é falsa. Presume-se que tal conhecimento apenas não foi adicionado à base de conhecimento.

3.2 Propriedades em OWL

As propriedade em OWL representam relacionamentos entre dois indivíduos. Em OWL existem três tipos de propriedades a saber:

- **Propriedades de objeto** (*object properties*) : Conectam um indivíduo a outro indivíduo.

- **Propriedades de tipos de dados** (*datatype properties*): Descrevem relacionamentos entre indivíduos e valores de dados. Ou seja, relacionam um indivíduo a um valor XML Schema Datatype[3] ou um literal RDF[4], ou seja, valores de dados.

- **Propriedades de anotação** (*Annotation Property*): Adicionam metadados às classes, aos indivíduos, às propriedades de objeto e as propriedades de tipos de dados.

3.2.1 Propriedades de anotações (*annotation properties*)

As propriedades de anotações são comentários, usando metadados, sobre classes, propriedades, indivíduos e a própria Ontologia (no cabeçalho do documento OWL).

A sub-linguagem OWL-Full não impõe restrições as essas propriedades. Entretanto, a OWL-DL tem as seguintes restrições:

- O *filler*[5] para as propriedades de anotação deve ter um dado literal, uma URI de referência ou um indivíduo. Um dado literal é um caractere de representação de um datatype value (valor de tipo de dados). P. ex. “Vagner”, "26", "3.11"

- As propriedades de anotação não podem ser usadas em axiomas que atuam sobre propriedades. Por exemplo, não podem ser usados na hierarquia de propriedades, de forma que não podem ter subpropriedades, ou ser subpropriedade de outra propriedade. Também não podem ter domínio e escopo especifico.

Exitem cinco propriedades de anotação pré-definidas em OWL. Elas podem ser usadas para fazer comentários em classes (inclusive classes anônimas, tais como restrições), propriedades e indivíduos:

1. ***owl:versionInfo***: Em geral, o escopo dessa propriedade é um string.

2. ***rdfs:label***: O escopo é um string. Essa propriedade é usada para adicionar nomes significativos (para pessoas) aos elementos da Ontologia, tais como classes, propriedades e indivíduos. Essa propriedade pode ser usada, também, para fornecer nomes multilíngües para elementos da Ontologia.

3. ***rdfs:comment***: O escopo é um string, usada para escrever comentários.

[3]Disponível em http://www.w3.org/TR/xmlschema-2/
[4]Disponível em http://www.w3.org/TR/rdf-primer/
[5]Chama-se *filler* à classe que contém indivíduos os quais atendem uma restrição

4. ***rdfs:seeAlso***: O escopo é uma URI usada para identificar recursos.

5. ***rdfs:isDefinedBy***: O escopo é uma URI usada para referenciar uma outra Ontologia que define elementos da Ontologia (que tem este tipo de comentário) tais como classes, propriedades e indivíduos.

Propriedades de anotação para comentar a Ontologia:

1. ***owl:priorVersion***: Identifica versões anteriores da Ontologia.

2. ***owl:backwardsCompatibleWith***: identifica versão anterior da Ontologia que é compatível com a versão atual. Isso quer dizer que todos os identificadores da versão anterior possuem o mesmo significado na versão atual. Assim, Ontologias e aplicações que fazem referência a versão anterior podem alterar a referencia para a nova versão.

3. ***owl:incompatibleWith***: Identifica a versão anterior de uma Ontologia que não é compatível com a atual.

3.2.2 Características das propriedades OWL

Listam-se, a seguir, as características das propriedades OWL:

- **Propriedade inversa**: Uma propriedade de objeto tem uma propriedade inversa correspondente. Ou seja, se um propriedade liga um indivíduo "a" a um indivíduo "b", então a propriedade inversa correspondente liga o indivíduo "b"ao indivíduo "a".

  – P. ex. *hasParent* (temPais) é inversa de *hasChild (temFilho).*

- **Propriedade funcional** (propriedades de valor único, *Single value properties*, ou características, *features*: Se uma propriedade é funcional, para um determinado indivíduo A, pode existir até no máximo um indivíduo B que está relacionado ao indivíduo A através dessa propriedade.

  – P. ex. *hasBirthMother* (TemMãeBiológica)

- **Propriedade funcional inversa**: Se uma propriedade é uma funcional inversa, isto significa que a sua propriedade inversa é funcional. Para o indivíduo1, pode existir no máximo um indivíduo relacionado ao indivíduo1 através da propriedade.

- **Propriedade transitiva**: Se uma propriedade P transitiva relaciona o indivíduo "a"ao indivíduo "b", e também um indivíduo "b"ao indivíduo "c", infere-se que o indivíduo "a"está relacionado ao indivíduo "c"através da propriedade P.
  - P. ex. *hasAncestor* (temAncestral)
- **Propriedade simétrica**: Se uma propriedade P é simétrica, e relaciona um indivíduo "a"ao indivíduo "b", então o indivíduo "b"também está relacionado ao indivíduo "a"através da propriedade P.
  - P. ex. *hasSibling* (temIrmão)
- **Propriedade anti-simétrica**: Se uma propriedade P é anti-simétrica, e a propriedade relaciona um indivíduo A a um indivíduo B. Então B não pode ser relacionado com um indivíduo A através da propriedade P.
  - P. ex. A *isChildOf* (éFilhoDe) B mas B não é de A.
- **Propriedade reflexiva**: Uma propriedade P diz-se ser reflexiva quando a propriedade deve relacionar cada um para si próprio.
  - P. ex. *knows*(conhece): uma pessoa conhece a si mesmo e a outras pessoas.
- **Propriedade irreflexiva**: Se uma propriedade P é irreflexiva, pode ser descrito como uma propriedade que relaciona um indivíduo A a um indivíduo B, onde cada um dos indivíduos não são os mesmos.
  - P. ex. *isMotherOf*: uma mulher não pode ser mãe dela mesma.

Resume-se algumas observações importantes, a seguir, acerca das propriedades em OWL:

- Se uma propriedade é transitiva, então a propriedade inversa a ela também é transitiva.
- Se uma propriedade é transitiva ela não pode ser funcional, uma vez que a propriedade transitiva, por sua própria natureza, pode formar cadeias de indivíduos.
- OWL-DL não permite que uma propriedade de tipos de dados seja transitiva, simétrica ou tenha uma propriedade inversa.

3.2.3 Domínio e escopo das propriedades

Uma propriedade possui domínio (*domain*) e escopo (*range*). As propriedades conectam indivíduos de um domínio a indivíduos de um escopo. P. ex. A *hasProperty* B, quer dizer A = domínio de *hasProperty* e B = escopo.

Em OWL, domínio e escopo, não são restrições sujeitas a verificação e são utilizados como axiomas em inferências.

É possível, mas não recomendável, indicar que uma classe e não seus indivíduos são escopo de uma propriedade. É um erro pensar que o escopo de uma propriedade é uma classe, quando um escopo corresponde na verdade aos indivíduos membros da classe. Ao especificar o escopo de uma propriedade como uma classe, trata-se tal classe como um indivíduo. Isto é um tipo de meta-declaração, e pode levar a Ontologia para o OWL-Full.

É possível especificar várias classes como escopo de uma propriedade. Caso isso seja feito o escopo da propriedade é interpretada como uma intersecção das classes.

Em geral, o domínio para uma propriedade é o escopo de seu inverso, e o escopo para uma propriedade é o domínio de sua inversa. P. ex. A *hasProperty* B $\Rightarrow$ B *isPropertyOf* A.

3.3 Descrição e definição de classes OWL

3.3.1 Restrições de propriedades

As propriedades são usadas para criar restrições. Restrições são utilizadas para restringir os indivíduos de uma classe. Todos os tipos de restrições descrevem um conjunto sem nome que pode conter indivíduos. Este conjunto corresponde a uma classe anônima (não nomeada). Quaisquer indivíduos membros da referida classe anônima satisfazem a restrição que descreve a classe. Quando se descreve uma classe nomeada usando restrições, o que se faz realmente na prática é descrever uma superclasse anônima da classe nomeada que contém os indivíduos que satisfazem a restrição.

3.3.2 Restrições de quantificador

(*quantifier restrictions*) São compostas por um quantificador, uma propriedade e uma classe nomeada que contém indivíduos os quais atendem a restrição (denominada *filler*).

- Restrição de quantificador = Propriedade + Quantificador (*some/only*) + *filler* (Classe).
  - P. ex. *hasBase some PizzaBase*

Quantificadores

Em OWL exitem dois tipos de quantificadores, listados a seguir:

- Quantificador existencial (∃): significa "(existe) pelo menos um"(*at least one*), ou algum (*some*), ou ainda, *someValuesFrom* (algunsValoresDe);
- Quantificador universal (∀): significa "apenas"(*only*) ou *allValuesFrom* (todosValoresDe).

3.3.3 Restrições existenciais

As restrições existenciais descrevem o conjunto de indivíduos que tem pelo menos um tipo específico de relacionamento com indivíduos membros de uma classe.

P. ex. "a"prop *some* "b"⇒ cada "a"deve ter pelo menos um relacionamento prop com "b", porém podem existir outras relações prop não especificadas explicitamente, com outros indivíduos que não sejam "b".

Restrições existenciais são o tipo mais comum de restrição. São conhecidas como *Some Restrictions*, ou *some values from restrictions*.

3.3.4 Restrições universais (*universals restictions*)

Descrevem o conjunto de indivíduos os quais, para uma dada propriedade, tem relacionamento apenas com outros indivíduos, membros de uma classe específica. Para uma dada propriedade, o conjunto de indivíduos descritos pela restrição universal vai também conter os indivíduos que não tem qualquer relacionamento com essa propriedade, para qualquer outro indivíduo. Não obriga que os únicos relacionamentos através da propriedade que possa existir sejam obrigatoriamente com indivíduos membros de uma classe específica (*filler*).

Para uma determinada propriedade, as restrições universais não especificam a existência de relacionamento. Apenas indicam que, se existe um relacionamento para a propriedade, ele ocorre para indivíduos membros de uma classe. P. ex. "a"prop only "b"descreve

a classe anônima de indivíduos que tem apenas relacionamentos prop com indivíduos membros da classe "b", ou, indivíduos que definitivamente não participam em qualquer outro relacionamento prop.

3.3.5 Restrições de cardinalidade (*cardinality restrictions*)

Descrevem a classe dos indivíduos que tem pelo menos um, ou no máximo ou exatamente um número especifico de relacionamentos (ou valores de tipos de dados) com outros indivíduos. São usadas para explicitar o número de relacionamentos em que um indivíduo pode participar para uma propriedade.

Para uma dada propriedade P:

- **Restrição de cardinalidade mínima** (*minimum cardinality restriction*) especifica o número mínimo de relacionamentos P dois quais um indivíduo deve participar ($\geqslant$).
- **Restrição cardinalidade máxima** (*maximum cardinality restriction*) especifica o número máximo de relacionamentos P dos quais um indivíduo pode participar ($\leqslant$).
- **Restrição de cardinalidade exata** (*exact cardinality restriction*) especifica o número exato de relacionamentos P dos quais um indivíduo participa ($=$).

Os relacionamentos (por exemplo, entre dois indivíduos) são considerados como relacionamentos separados quando se pode garantir que também são distintos os indivíduos que funcionam como *fillers dos relacionamentos*.

3.3.6 Restrições temValor (*hasValue*)

Restrições temValor descrevem o conjunto de indivíduos que possui pelo menos um relacionamento através da propriedade com indivíduo específico (p. ex. a hasCountryOfOrigin value Italy). Ou seja, descreve uma classe anônima de indivíduos que estão relacionados a outros indivíduos específicos por uma propriedade.

P. ex. "a" prop value "abc". essa restrição descreve a classe anônima (a) de indivíduos que tem pelo menos um relacionamento através da propriedade prop com o indivíduo abc, porém podem existir outros relacionamentos com a propriedade prop.

3.4 Condições declaradas (*asserted conditions*)

As condições declaradas são utilizadas para definir as classes em OWL. Os dois tipos dessas condições são tratados a seguir.

3.4.1 Condições necessárias (*necessary criteria*)

Têm a seguinte definição: se um indivíduo é membro de uma classe nomeada então é obrigatório que satisfaça as condições necessárias para tal. Entretanto, se algum indivíduo satisfaz as condições necessárias, não se pode dizer que seja membro da referida classe nomeada. Ou seja, as condições necessárias não são suficientes para que se possa dizer isso.

Á classe que tem apenas condições necessárias, dá-se o nome de classe primitiva (*primitive class*) ou classe parcial (*partial class*).

Em OWL, na definição de uma classe, superclasses são consideradas aquelas que têm condições necessárias. Ou seja, se a classe A é descrita por condições necessárias, então pode-se dizer que se um indivíduo é membro de A, ele deve satisfazer as condições. Entretanto, não se pode dizer que qualquer indivíduo que satisfaça tais condições é um membro da classe A.

3.4.2 Condições necessárias e suficientes (*necessary & sufficient criteria*)

Têm a seguinte definição: se um indivíduo é membro de classe nomeada então é obrigatório que satisfaça as condições. Se algum indivíduo satisfaz as condições então é obrigatório que seja membro de classe nomeada.

À classe que tem pelo menos um conjunto de condições necessárias e suficientes, chama-se classe definida (*defined class*) ou classe completa (*completed class*).

Em OWL, na definição de uma classe A, classes equivalentes (*equivalent classes*) a ela, são aquelas que têm as restrições necessárias e suficientes que satisfazem a definição da classe A. Ou seja, se a classe A é definida usando condições necessárias e suficientes, pode-se dizer que, se um indivíduo é membro da classe A ele deve satisfazer as condições, e pode-se dizer que qualquer indivíduo que satisfaz essas condições deve ser membro de A. As condições não são apenas necessárias para a associação com A, mas são, também, suficientes de forma a determinar que, se alguma coisa satisfaz essas condições é um

membro de A.

As classes que têm pelo menos um conjunto de condições necessárias e suficientes são conhecidas como classes definidas. Tais classes tem uma definição, e qualquer indivíduo que satisfaça tal definição pertence a classe. Classes que não tem nenhum conjunto de condições necessárias e suficientes (apenas condições necessárias) são conhecidas como classes primitivas.

Verificar a relação classe/superclasse (*subsumption relationship*) de classes é uma tarefa básica de um MI de lógica descritiva, e é possível usá-lo para computar automaticamente a hierarquia das classes.

Em OWL é possível ter conjuntos múltiplos de condições necessárias e suficientes.

3.5 Axiomas comuns

Lista-se, a seguir, três axiomas comuns (mas não obrigatórios) em uma Ontologia OWL:

1. **Axiomas de fechamento (*closure axiom*)**: Consiste em uma restrição universal que atua na propriedade informando que ela pode apenas ser preenchida por *fillers* específicos. A restrição tem um *filler* que é a união dos fillers que ocorrem nas restrições existenciais da propriedade.

2. **Partições de valor (*value partitions*)**: São usadas para refinar as descrições de classes, são padrões de projetos (soluções desenvolvidas por especialistas e reconhecidos como soluções para problemas comuns de modelagem).

3. **Axiomas de Cobertura (*covering axioms*)**: Consiste de duas partes: a classe que está sendo coberta, e as classes que formam a cobertura. Por exemplo, tem-se três classes A, B e C, e as classes B e C são subclasses de A. Tem-se um Axioma de Cobertura que especifica que a classe A é coberta pela classe B e também pela classe C. Isto significa que um membro da classe A deve ser membro da classe B e/ou C. Se as classes B e C são disjuntas, então um membro de A deve ser um membro de B ou de C. Em geral, embora B e C sejam subclasses de A, um indivíduo pode ser um membro de A sem ser membro de uma das classes B ou C. Um axioma de cobertura manifesta-se como uma classe que é a união das classes que estão sendo cobertas, as quais formam a superclasse da classe que está sendo coberta. No caso de A, B e C, a classe A pode ter uma superclasse de B ou C.

3.6 *Namespaces*

Cada Ontologia tem seu próprio *namespace*, que é conhecido com o *namespace* padrão e pode usar outros *namespaces*. Um *namespace* é uma sequência de caracteres que precede os identificadores de classes, de propriedades e de indivíduos em uma Ontologia. É possível a uma Ontologia referenciar classes, propriedades e indivíduos em outra Ontologia, sem ambigüidades e sem causar problemas com nomes, através da manutenção de *namespaces* distintos para todas as Ontologias.

A garantia de que os *namespaces* são único reside em sua representação via URIs-(*Unique Resource Identifiers*), terminados com / ou com #. O uso de URIs se baseia na garantia de unicidade.

4 Modelagem da informação

4.1 Níveis de organização da informação

Lista-se, a seguir, os níveis de organização da informação, no contexto semântico, relevantes à nossa temática:

1. Vocabulário controlado: O conjunto das palavras especializadas em qualquer campo de conhecimento ou atividade; nomenclatura, terminologia[1].

2. Glossário: Uma lista de termos de um determinado domínio de conhecimento com a respectiva definição destes termos[2] (dentro do contexto do domínio de conhecimento no qual estão inseridas) .

3. Taxonomia: Um hierarquia de termos em um vocabulário controlado, usada para classificar termos.

4. Tesauro: Uma compilação de termos, com os respectivos sinônimos e/ou antônimos e termos relacionados dentro de um domínio de conhecimento [3].

5. Redes semânticas: Grafos direcionados e rotulados, com nodos representando objetos físicos ou conceituais e arcos representando relações entre os objetos [4].

6. Ontologia: (ver sessão 2.3).

7. Base de conhecimento: segundo Noy e McGuinness (2001): "Uma ontologia, juntamente com um conjunto de instâncias de indivíduos de classes constitui uma base de conhecimentos. Na realidade, existe uma linha tênue, onde termina a ontologia e a base de conhecimentos se inicia."

[1]Novo Dicionário Aurélio versão 5.0
[2]Definição retirada de `http://pt.wikipedia.org/wiki/Glossario`
[3]Definição baseada em: `http://pt.wikipedia.org/wiki/Tesauro`
[4]Definição baseada em Real (2010) e `http://pt.wikipedia.org/wiki/RedeSemantica`

4.2 Paralelo com paradigma de orientação a objetos

Lista-se, a seguir, um paralelo entre uma Ontologia e o paradigma de orientação a objetos, sob a ótica de uma estrutura de classes:

- Uma ontologia:
  1. reflete a estrutura do **mundo**;
  2. é frequentemente sobre a **estrutura de conceitos**;
  3. representação física **não** é problema.
- Uma estrutura de classes orientadas a objeto:
  1. reflete a estrutura de **dados e de código**;
  2. geralmente descreve **comportamentos** (métodos das classes);
  3. descreve a r**epresentação física dos dados** (char, long, int, ...).

4.3 Engenharia ontológica e o método 101

Engenharia ontológica (*ontology engineering*):

> é a definição dos termos em um domínio de conhecimento e das relações entre eles, definindo os conceitos (classes), hierarquia de classes, atributos e propriedades das classes (e restrições para seus valores) e indivíduos (assim como o preenchimento de suas propriedades).(NOY; MCGUINNESS, 2001).

O método 101, proposto pelas pesquisadoras Natalya Noy e Deborah McGuinness[5], define um processo cíclico de desenvolvimento de Ontologias. Essa metodologia tem seus sete processos descritos a seguir.

1. **Determinar domínio escopo**: Nesta etapa deve-se definir que domínio a Ontologia irá cobrir, qual será o uso da Ontologia e que tipo de questões as informações contidas na ontologia irão responder (chamadas de questões de competência). É importante enfatizar que durante o ciclo de vida do desenvolvimento da ontologia essas definições podem mudar.

[5]vide Noy e McGuinness (2001)

2. **Considerar reuso**: Deve-se considerar o reuso de outras Ontologias que tratam, de alguma forma, do domínio de conhecimento a ser coberto com os objetivos de se reduzir o esforço no desenvolvimento, interagir com ferramentas que usam outras Ontologias e utilizar ontologias que tenham sido validadas pelo uso em outras aplicações.

3. **Enumerar termos**: Definir os termos que serão cobertos pela Ontologia, as propriedades desses termos e o que se deseja saber sobre estes termos.

4. **Definir classes e hierarquia de classes**:

   - Definir que termos são considerados classes (conceitos) no domínio de conhecimento coberto pela Ontologia, tendo em mente que as classes são conjuntos ou coleções de elementos com propriedades similares, os quais chama-se instâncias das classes ou indivíduos.
   - As classes geralmente constituem uma hierarquia taxonômica (superclasse-subclasse), definindo um relacionamento "éUm"(*is-a*). A instância de uma subclasse é uma instância de sua superclasse. Uma classe é uma conjunto de elementos e uma subclasse é uma subconjunto desses elementos.
   - Os modos de desenvolvimento de hierarquia taxonômica podem ser:
     - **top-down**: definir os conceitos mais gerais e especializá-los.
     - **botton-up**: definir os conceitos mais específicos e organizá-los em classes mais gerais.
     - **combinação**: definir os conceitos mais notáveis primeiro, em seguida, generalizar ou especializar.
   - Deve-se definir a documentação das classes:
     - descrevendo as classes em linguagem natural;
     - listando suposições de domínio relevantes para a definição da classe;
     - e listando sinônimos.

5. **Definir propriedades**: Definir os atributos das instâncias das classes e relações entre as instâncias.

   - Definir os seguintes tipos de propriedades
     - intrínsecas: aquelas que fazem parte da essência do objeto, são inerentes ou peculiares ao objeto. P. ex. cor, sabor, ...

  - extrínsecas: aquelas que não fazem parte da essência do objeto. P. ex. nome, título, preço, ...
  - partes: aquelas que definem a composição do objeto. P. ex. ingredientes, peças, ...
  - relações com outros objetos: aquelas que definem como uma objeto se relaciona com outro. P. ex. autor de livro, publicador de livro (editora), produtor de peças, dentre outras.

- Definir as propriedades simples e complexas:
  - Propriedades simples: são os atributos ou (propriedades de tipos de dados em OWL), elas contém valores primitivos (*strings*, números, ...)
  - Propriedades complexas: são as temValor (*hasValue*) em OWL, ela contêm (ou apontam) pra outros objetos.
- Deve-se ter sempre em mente que as classes herdam todas as propriedades de suas superclasses. Se uma classe tem múltiplas superclasses ela herda todas as propriedades de suas superclasses.

6. **Definir restrições**: Definir as descrições dos limites dos conjuntos de possíveis valores para uma propriedade. As restrições são também conhecidas como *facetas*. P. ex. o número de valores que uma propriedade se relaciona, o tipo de valor que uma propriedade tem ou a faixa de valores de uma propriedade que se relaciona com números.

7. **Criar instâncias**: Definir as instâncias das classes, preencher as propriedades e as restrições de acordo com a instância.

Finaliza-se esta seção com algumas heurísticas enfatizadas pelas autoras do 101, no que concerne ao desenvolvimento de ontologias:

- Não há uma única maneira correta de modelar um domínio, há sempre alternativas viáveis. A melhor solução quase sempre depende da aplicação que se tem em mente;
- O desenvolvimento de uma Ontologia é necessariamente um processo iterativo;
- Os conceitos na Ontologia devem ser próximos de objetos (físicos ou lógicos) e de relacionamentos do domínio de interesse. Os objetos são mais suscetíveis de serem substantivos e os relacionamentos de serem verbos em sentenças que descrevem o domínio de conhecimento.

4.4 Ambiente *Protégé*

De acordo com Stanford University (2010) o software gratuito[6] e livre[7] *Protégé*[8], é uma plataforma que fornece um conjunto de ferramentas para a construção de aplicações de modelos de domínio e de bases de conhecimentos baseadas em Ontologias. Em sua essência, *Protégé* implementa um rico conjunto de estruturas de modelagem de conhecimento e ações de apoio à criação, visualização e manipulação de ontologias em vários formatos de representação. Pode ser customizado para fornecer suporte amigável de domínio para a criação de modelos de conhecimento e inserção de dados. Além disso, pode ser extendido por meio de uma arquitetura baseada em API[9] Java de *plugins* para a construção de ferramentas baseadas em conhecimento e aplicações.

Palavra interessante, o "é" pronuncia com som fechado, parecido com a pronúcia de "ê". De acordo com o *Cambridge dictionaries online*[10], *Protégé* significa:

> "Protegido ou favorito. Uma pessoa jovem que é protegida por alguém mais velho que tem experiência ou influência".

O software Protégé habilita a construção de Ontologias OWL e pode ser baixado em `http://protege.stanford.edu/download/download.html`. A Figura 3, exibe a tela inicial do *Protégé* na construção de uma ontologia OWL.

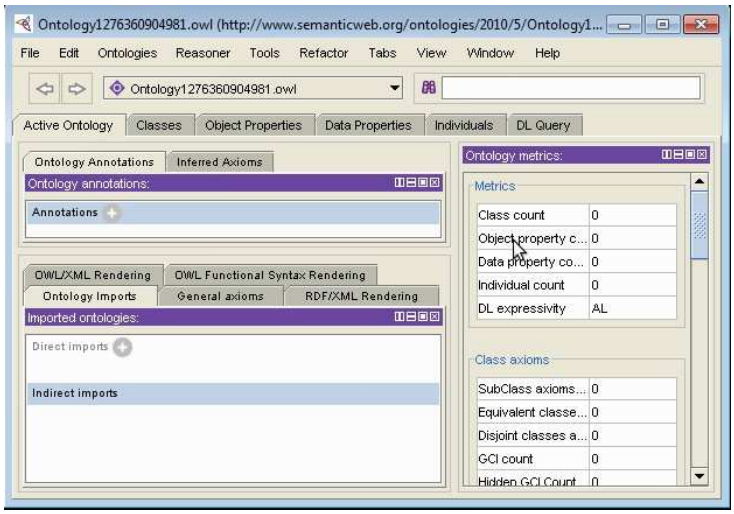


Figura 3: Tela do ambiente Protegé.
Fonte: Do autor.

[6] `http://pt.wikipedia.org/wiki/Software_gratuito`
[7] `http://www.gnu.org/philosophy/free-sw.pt-br.html`
[8] `http://protege.stanford.edu/`
[9] *Application Programming Interface*
[10] `http://dictionary.cambridge.org/dictionary/british/protege`

4.5 Modelagem da ontologia estudo de caso

Faz-se, agora, a modelagem da base de conhecimento do estudo de caso deste trabalho, que consiste em recuperação de informações bibliográficas. Para tal, utiliza-se o método 101 e o ambiente *Protégé*.

4.5.1 Determinar a abrangência (domínio e escopo)

Inicia-se com questões de competência simples e suas respostas:

- Qual o domínio que a Ontologia irá cobrir?
  - A Ontologia irá cobrir o domínio de livros.
- Qual será o uso da Ontologia?
  - Recuperação de informações acerca de livros no contexto da web.
- Quem irá usar a Ontologia?
  - A Ontologia será usada por qualquer cliente *web services* que precise fazer uma pesquisa ou inferência no domínio de conhecimento coberto pela Ontologia, para recuperar informações para seus usuários.
- As informações na ontologia responderão que tipo de questões?
  - Quais livros o autor X escreveu?
  - Em que livrarias se pode comprar o livro Y? Quanto custa?
  - Quanto custa o livro Z na livraria A da cidade B?
  - Em que biblioteca posso ler o livro Z?
  - Qual livraria na cidade A vende o livro X mais barato?

4.5.2 Considerar reuso

Executou-se diversas buscas nos seguintes mecanismos de buscas e repositório de ontologias:

- Swoogle (`http://swoogle.umbc.edu/`)
- SWSE (`http://swse.deri.org/`)

- Sindice (`http://www.sindice.com/`)
- Wat-son (`http://watson.kmi.open.ac.uk/WatsonWUI/`)
- Falcons (`http://iws.seu.edu.cn/services/falcons/objectsearch/index.jsp`)
- TONES *Ontology Repository* (urlhttp://owl.cs.manchester.ac.uk/repository/)

Porém nenhum deles retornou algum resultado satisfatório sobre livros. Não foi possível baixar, também, a ontologia de livros do *amazon.com* ou de outras livrarias. Decidiu-se então, neste ponto, construir uma Ontologia sobre livros do princípio da construção de uma Ontologia, sem reaproveitar elementos já previamente construídos em outras Ontologias. As Ontologias que poderia-se utilizar, no sentido de contribuir para a construção da Ontologia dos livros são listadas a seguir.

- *Dublin Core*[11]: Uma Ontologia construída em linguagem RFD que provê metadados para recursos na web, tais como sites, imagens, textos, e demais recursos que podem ser disponibilizados na web. Como exemplo de metadados providos pela *Dublin Core* tem-se título, criador assunto, descrição, dentre outros.
- *The Friend of a Friend (FOAF) Project*[12]: Uma ontologia RDF que objetiva compartilhar informações entre amigos na web, como nome, e-mail, foto, e diversas outras informações.
- GeoRSS[13]: Um vocabulário de termos que podem ser usados em documentos RDF para representar uma informação geoespacial.
- OWL Time[14] Um conjunto de ontologias que podem ser utilizadas para representar informação temporal em linguagem OWL.

Ao analisar estas ontologias nota-se que elas apresentam uma certa complexidade, que vai além do escopo deste trabalho e podem ser utilizadas no enriquecimento de nossa Ontologia de livros, em trabalhos futuros.

4.5.3 Enumerar termos

Nesta seção lista-se apenas alguns dos termos relevantes ao domínio de conhecimento que a Ontologia de livros deve cobrir, vide Quadro 1.

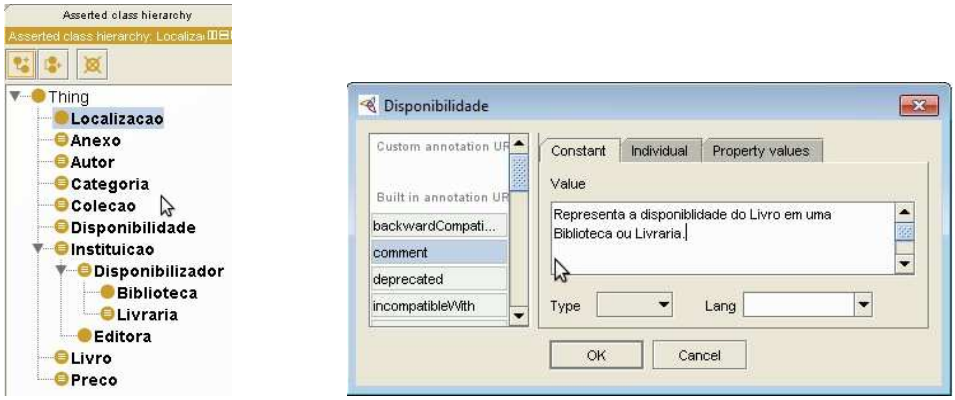
[11] `http://dublincore.org`
[12] http://www.foaf-project.org/
[13] http://georss.org
[14] http://www.w3.org/TR/owl-time/

livro, livraria, editora, biblioteca, localização, preço, autor, anexo, categoria, disponibilidade, título, coleção

Quadro 1: Enumeração de termos.
Fonte: Do autor.

4.5.4 Definir classes e hierarquia de classes

Na definição da hierarquia de classes foi necessário acrescentar mais duas classes: Instituição e Disponibilizador. Vide Figura 4. Por motivos de padronização todos os termos que representam classes foram grafados com inicial maiúscula, já as propriedades foram escritas, posteriormente, com inicial minuscula. As classes da Ontologia, em sua maioria não foram documentadas com comentários por apresentar nomes auto-explicativos. Para exemplificar foi feito apenas uma breve documentação de comentário na classe 'Disponibilidade'.



(a) Hierarquia (b) Documentação

Figura 4: Hierarquia e documentação de classes.
Fonte: Do autor.

4.5.5 Definir propriedades

As Figura 5 exibe as propriedades de atributos para cada uma das classes. Note que para obter boa organização e facilidade de leitura agrupou-se as propriedades de tipos de dados por Classes em hierarquia.

A Figura 6 mostra as propriedades de objeto da Ontologia. O Quadro 2 mostra as propriedades de objeto que foram comentadas com seus respectivos comentários.

O Quadro 3 lista as propriedades de objeto que herdam da propriedade livroTem.

O Quadro 4 mostra as propriedades de objeto que herdam de temLivro.

Figura 5: Propriedades de tipos de dados das classes.
Fonte: Do autor.

Figura 6: Propriedades de objeto da Ontologia.
Fonte: Do autor.

4.5.6 Definir restrições

O Quadro 5 cita as **propriedades de tipos de dados** que são funcionais, com suas respectivas superpropriedades.

O Quado 6 lista algumas das **propriedades de objeto** com suas respectivas características.

Nesta Ontologia, as classes, em mesmo nível de hierarquia foram tornadas disjuntas.

Lista-se, a seguir, os axiomas de equivalência das classes definidas (condições necessárias e suficientes):

- Anexo

Propriedade	Comentário
disponibilidadeEm	Informa que Disponibilizador(res) tem a Disponibilidade (do Livro)
eDisponibilizadorDe	Informa as Disponibilidade(s) do Disponibilizador
livroTem	Superpropriedade dos relacionamentos que outras classes formam com Livro
disponibilizadoPor	Informa que Disponibilizador(res) disponibiliza(m) o Livro
temDisponibilidade	Informa quais as disponibilidade do Livro
relacionado	Propriedade que determina que uma livro é relacionado com outro por motivos de semelhança de tema, dentre outros tipos de semelhanças
temLivro	Superpropriedade dos relacionamentos que Livro forma com outras classes
disponibiliza	Informa que Livro(s) são disponibilizados pelo Disponibilizador
eDisponibilidadeDe	Informa de que Livro é a Disponibilidade

Quadro 2: Propriedades de objeto comentadas.
Fonte: Do autor.

daColecao, disponibilizadoPor, eDaCategoria, escritoPor, publicadoPor, temAnexo, temDisponibilidade, temPreco

Quadro 3: Subpropriedades de livroTem.
Fonte: Do autor.

  - *eAnexoDe some Livro and eAnexoDe only Livro and tipoAnexo some string and tituloAnexo some string and descricaoAnexo only string and tipoAnexo only string and tituloAnexo only string*

- Autor
  - *escreveu only Livro and nomeAutor some string and dataNascimento only string and descricaoAutor only string and emailAutor only string and foto only string and nomeAutor only string and origem only string and siteAutor only string*

- Categoria
  - *categoriaTem only Livro and eSubcategoriaDe only Categoria and eSuperCategoriaDe only Categoria and descricaoCategoria some string and nomeCategoria some string and descricaoCategoria only string and nomeCategoria only string*

- Colecao

categoriaTem, colecaoTem, disponibiliza, eAnexoDe, eDisponibilidadeDe, ePrecoDe, escreveu, publica

Quadro 4: Subpropriedades de temLivro.
Fonte: Do autor.

Superpropriedade	Propriedade
propriedadeDeAnexo	tipoAnexo, tituloAnexo
propriedadeDeAutor	dataNascimento, nomeAutor, origem
propriedadeDeCategoria	nomeCategoria
propriedadeDeColecao	anoPublicacaoColecao, nomeColecao, quantidadeVolumes
propriedadeDeDisponibilidade	disponibilidade, quantidade
propriedadeDeInstituicao	nomeInstituicao
propriedadeDeLivro	acabamento, anoPublicacao, formato, isbn, lingua, numeroEdicao, origemlivro, numeroPaginas, origemLivro, referenciaBibliografica, titulo, subtitulo
propriedadeDeLocalizacao	bairro, cep, cidade, logradouro, numero, uf
propriedadeDePreco	custo, dataPreco, tipoPagamento

Quadro 5: Propriedades de tipos de dados funcionais.
Fonte: Do autor.

  - *colecaoTem some Livro and colecaoTem only Livro and anoPublicacaoColecao some string and descricaoColecao some string and nomeColecao some string and quantidadeVolumes some string and anoPublicacaoColecao only string and descricaoColecao only string and nomeColecao only string and quantidadeVolumes only string*

- Disponibilidade
  - *disponibilidadeEm some Disponibilizador and eDisponibilidadeDe some Livro and disponibilidadeEm only Disponibilizador and eDisponibilidadeDe only Livro and disponibilidade some string and quantidade some string and disponibilidade only string and quantidade only string*

- Disponibilizador
  - *Instituicao and disponibiliza some Livro*

- Livraria
  - *Disponibilizador and ofertaDe some Preco*

- Preco

Característica	Propriedade
funcionais	eAnexoDe, eDisponibilidadeDe, ePrecoDe
funcionais e funcionais inversas	localTemInstituicao, temLocalizacao
transitivas	relacionado, eSubCategoriaDe, eSuperCategoriaDe

Quadro 6: Características das propriedades de objeto.
Fonte: Do autor.

  - *ePrecoDe some Livro and ofertadoPor some Livraria and ePrecoDe only Livro and ofertadoPor only Livraria and custo some string and dataPreco some string and descricaoPagamento some string and tipoPagamento some string and custo only string and dataPreco only string and descricaoPagamento only string and tipoPagamento only string*

- Localizacao

  - *localTemInstituicao only Instituicao and bairro some string and cep some string and cidade some string and logradouro some string and numero some string and uf some string and bairro only string and cep only string and cidade only string and logradouro only string and numero only string and uf only string*

- Instituicao

  - *temLocalizacao some Localizacao and temLocalizacao only Localizacao and nomeInstituicao some string and descricaoInstituicao only string and emailInstituicao only string and nomeInstituicao only string and siteInstituicao only string and telefoneInstituicao only string*

- Livro

  - *disponibilizadoPor some Disponibilizador and eDaCategoria some Categoria and escritoPor some Autor and daColecao only Colecao and disponibilizadoPor only Disponibilizador and eDaCategoria only Categoria and escritoPor only Autor and publicadoPor only Editora and temAnexo only Anexo and temDisponibilidade only Disponibilidade and temPreco only Preco and acabamento some string and anoPublicacao some string and formato some string and isbn some string and lingua some string and numeroEdicao some string and numeroPaginas some string and origemLivro some string and referenciaBibliografica some string and subtitulo some string and titulo some string and acabamento only string and anoPublicacao only string and capa only string and descricao*

only string and formato only string and isbn only string and lingua only string and numeroEdicao only string and numeroPaginas only string and origemLivro only string and referenciaBibliografica only string and siteLivro only string and subtitulo only string and titulo only string

A classe Colecao tem o axioma de condições necessárias: *colecaoTem min 2 Livro*

4.5.7 Criar instâncias

Nesta etapa, foi feito apenas uma ou duas instâncias de prova de cada classe para provar a consistência da ontologia.

Figura 7: Criação de algumas instâncias de classes (indivíduos).
Fonte: Do autor.

5 Implementação da base de conhecimentos

5.1 Java *Enterprise Edition*

De acordo com Corp. Oracle (2009), Java[1] *Enterprise Edition* (JEE)[2] é uma plataforma, de ambiente centrado em Java, para o desenvolvimento, construção e implantação de aplicações *enterprise online* baseadas na web. A plataforma JEE é composta por um conjunto de serviços, APIs, e protocolos que fornecem funcionalidades para o desenvolvimento de várias camadas, para aplicações baseadas em web. JEE simplifica o desenvolvimento de aplicações e diminui as necessidades de programação e treinamento do programador para criar componentes padronizados, reutilizáveis e modulares, permitindo cada camada tratar muitos aspectos de programação automaticamente.

Descreve-se brevemente, nas seções seguintes deste capítulo, as tecnologias JEE utilizadas neste trabalho.

5.1.1 *Enterprise JavaBeans*

Enterprise JavaBeans[3] (EJB) é um modelo de componente padrão do lado do servidor para aplicativos de negócio distribuídos, conforme explicado por Burke e Monson-Haefel (2007). A especificação EJB oferece um modelo padrão para construir componentes do lado do servidor que representam processos do negócio, que podem ser combinados para se criar aplicativos de negócio.

Componentes EJB rodam em um contêiner EJB, um ambiente de execução dentro de um servidor de aplicações *enterprise*. O contêiner EJB provê diversos serviços de nível de sistema, transparentes para o desenvolvedor da aplicações, como transações e segurança

[1] `http://java.sun.com/`
[2] `http://java.sun.com/javaee/`
[3] `http://java.sun.com/products/ejb/`

dos *entreprise beans*.

Sun Microsystems Inc. (2009) lista os requisitos para se usar *enterprise beans*:

- A aplicação precisa ser escalável[4]. Para acomodar um número crescente de usuários, pode ser necessário distribuir os componentes de uma aplicação em várias máquinas, que assim como a localização dos componentes *enterprise beans*, ficará transparente para os clientes da aplicação.
- Transações devem assegurar a integridade dos dados. *Enterprise beans* suportam transações, os mecanismos que gerenciam o acesso simultâneo de objetos compartilhados.
- A aplicação terá uma variedade de clientes. Com apenas algumas linhas de código, clientes remotos podem facilmente localizar *enterprise beans*. Esses clientes podem ser poucos, diversos ou numerosos.

Lista-se, a seguir, os tipos de *entreprise beans*, de acordo com Sun Microsystems Inc. (2009):

- *Session*: Um *bean* de sessão (*session bean*) encapsula a lógica de negócio que pode ser chamada programaticamente por um cliente local, remoto ou via *web services*. Para acessar um aplicativo que está implantado no servidor, o cliente invoca os métodos do *bean* de sessão, que realiza um trabalho para o seu cliente, protegendo o cliente de complexidade e executando tarefas de negócios dentro do servidor. Um *bean* de sessão não é persistente. Existem três tipo de *session beans*.
  - *Statefull*: O estado de um objeto é composto dos valores de suas variáveis de instância. Em um *statefull session bean*, as variáveis de instância representam o estado de uma única sessão cliente-*bean*. Esse estado é chamado de "estado conversasional". Cada instância desse *bean* não é compartilhada dentre as sessões dos clientes.
  - *Stateless*: *Bean* que não mantém estado conversasional com o cliente. Por causa dessa característica, cadas instância desse bean pode ser compartilhada por vários clientes.
  - *Singleton*: É o *bean* que é instanciado apenas uma vez durante a execução da aplicação. Esse tipo de *bean* foi projetado para circunstâncias em que uma

[4]De acordo com Coulouris, Dollimore e Kindberg (2007): "um sistema é descrito como escalável se permanece eficiente quando há um aumento significativo no número de recursos e no número de usuários".

única instância do *bean* deve ser compartilhada e acessada concorrentemente pelos clientes. O*Singleton session bean* mantém o estado conversasional para a aplicação e assim como os *stateless session beans* podem implementar web services endpoints.

- *Message-Driven*: Processa mensagens assincronamente. Atua como um ouvinte para um determinado tipo de mensagens, como o Java Message Service API.

5.1.2 *Web Services*

De acordo com Sun Microsystems Inc. (2009), os *web services* são aplicações cliente e servidor que se comunicam pelo protocolo *HyperText Transfer Protocol* (HTTP) da *World Wide Web* (WWW).

W3C (2010c) sugere uma definição mais completa:

> Serviços da Web fornecem uma forma padrão de interoperação entre diferentes aplicações de software, rodando em uma variedade de plataformas e / ou frameworks. Os serviços Web são caracterizados por sua grande interoperabilidade e extensibilidade, bem como suas descrições máquina processável graças ao uso de XML. Eles podem ser combinados em uma maneira de baixo acoplamento, a fim de alcançar operações complexas. Programas de prestação de serviços simples podem interagir uns com os outros, a fim de fornecer serviços sofisticados de valor acrescentado.(W3C, 2010c).

Coulouris, Dollimore e Kindberg (2007) complementa, dizendo que:

> Um serviço web (*web service*) fornece uma interface de serviço que permite aos clientes interagirem com servidores de uma maneira mais geral do que acontece com os navegadores web. Os clientes acessam as operações na interface de um serviço web por meio de requisições e respostas HTTP [...] permitindo que eles sejam mais facilmente usados em aplicações de internet.(COULOURIS; DOLLIMORE; KINDBERG, 2007).

Protocolo SOAP

O protocolo SOAP[5](*Simple Object Access Protocol*) é projetado para permitir tanto interação cliente-servidor como assíncrona pela internet. Ele define um esquema para uso de XML para representar o conteúdo de mensagens de requisição e resposta, assim

[5] `http://www.w3.org/TR/soap/`

como um esquema para a comunicação de documentos, segundo Coulouris, Dollimore e Kindberg (2007).

O protocolo SOAP é baseado em HTTP, SMTP, TCP ou UDP e é uma recomendação do W3C para serviços web.

JAX-WS

A API JEE para desenvolvimento de serviços web e clientes em SOAP é chamada JAX-WS[6]. Esta API deve estar presente nos servidores de aplicação que seguem a especificação JEE versão 5 ou 6, ou pode ser baixada separadamente como biblioteca e injetada em um servidor mais simples, como um contêiner de *servlets*.

5.1.3 Servidor GlassFish e IDE Netbeans

Glassfish[7] é um servidor de aplicações open source, projetado para a plataforma JEE. A versão 3 deste servidor já é certificada para a especificação JEE 6, e é utilizada como implementação de referência para a especificação. O projeto GlassFish tem uma ampla comunidade de usuários, desenvolvedores e parceiros.

GlassFish apresenta duas interfaces de operação. A mais notável é a interface de administração, chamada web console. A outra interface é via comandos através do executável "asadmin". Outros subprojetos que se destacam no GlassFish são o *update center*, um gerenciador de atualizações automatizado e amigável; o projeto Grizzly, como servidor web básico responsável pela comunicação HTTp e HTTPS; suporte a Ruby on Rails[8]; e integração facilitada com diversos ambientes de programação.

O ambiente Netbeans[9] é uma IDE (*integrated development environment* - ambiente integrado de desenvolvimento) *open source* e gratuita(*freeware*) com suporte a diversas linguagens de programação (C/C++, PHP, JavaScript, Groovy, Ruby e Java). Netbeans tem suporte nativo à especificação JEE 6 e integração facilitada com o servidor GlassFish.

[6] `https://jax-ws.dev.java.net/`
[7] `https://glassfish.dev.java.net/`
[8] `http://www.rubyonrails.pro.br/`
[9] `http://netbeans.org/`

5.2 API Jena

Jena[10] é um *framework* Java desenvolvido para a implementação de sistemas para a web semântica. Provê ambiente programático para RDF, RDFS e OWL.

Jena possui uma *engine* de inferência baseada em regras, uma engine de query SPARQL, uma API RDF, lê e escreve RDF em RDF/XML, N3 e N-triplas, uma API OWL; e permite persistência em banco de dados ou memória.

Para iniciar o entendimento da programação com a API Jena, se faz necessário entender como suas classes se relacionam com os conceitos da Web Semântica, vide Quadro 7.

Artefato	**Web Semântica**	**Classe Jena**	**Observações**
Sujeito, predicado, objeto	URI	Resource, Property	Um Resource pode ser um sujeito, predicado ou objeto
Declaração (*statement*)	Declaração (*statement*)	Statement	
Dado	Ontologia e instância de dado	Graph e Model	Graphs são blocos de construção básicos para Models. ambos precisam conter a ontologia e instância de dado
Consulta e resultado de consulta	SPARQL e dados da Web Semântica	Query e ResultSet	Análogos a bases de dados relacionais
Raciocinador	Raciocinador	Reasoner	Permite múltiplos raciocinadores, internos ou externos
Regras	SWRL	Reasoner	O suporte a regras é determinado pelo raciocinador específico
Notificação de eventos	Não aplicável	ObjectListener	Habilita processamento de eventos

Quadro 7: Comparando Web Semântica e Jena.
Fonte: Hebeler et al. (2009) (p.270).

Jena possibilita a criação e manipulação de grafos RDF, representadas pelos recursos (Resource), propriedades (Properties) e Literals, formando as tuplas que irá dar origem aos objetos criados pelo java. Assim, esse conjunto de objetos é usualmente chamado de *model*. O Model[11] é o conjunto de declarações (*statements*) que forma o grafo por completo. De acordo com Verzulli (2001), Jena define uma série de interfaces para acessar e manipular declarações RDF (vide Figura 8). Comenta-se estas interfaces abaixo.

[10] `http://jena.sourceforge.net/`

[11] "Resource", "Properties", "Literals" e "Model" estão grafados com inicial maiúscula e não itálico por representarem interfaces Java.

- RDFNode
  - Fornece uma base comum para todos os elementos que podem ser partes de triplas[12].
- Resource
  - É qualquer objeto que possa ser representado por um URI.
- Literal
  - Representa literais como "peixe vermelho"ou 225 que pode ser usado como "objeto" em triplas.
  - A interface Literal fornece métodos de acesso para converter literais Java para diversos tipos como String, int e double.
- Property
  - Objetos java que implementam a interface de Property podem ser "predicado" em triplas.
- Statement
  - Representa uma tripla.
  - Também pode ser usado como "objeto" em uma tripla de uma declaração aninhada.
- Container
  - Objetos java que implementam Container, Alt, Bag, ou interface Seq podem ser o "objeto" triplas.
  - Container são conjuntos de objetos dispostos da maneira adequada (Alt, Bag e Seq).

5.3 SPARQL

SPARQL[13] é o acrônimo recursivo para SPARQL *Protocol and RDF Query Language*. Uma linguagem de consulta e protocolo para RDF, desenvolvido pelo W3C RDF *Data Access Working Group*[14].

[12] considera-se "triplas"como triplas RDF (sujeito, predicado, objeto)
[13] `http://www.w3.org/TR/rdf-sparql-query/`
[14] `http://www.w3.org/2009/sparql/wiki/Main_Page`

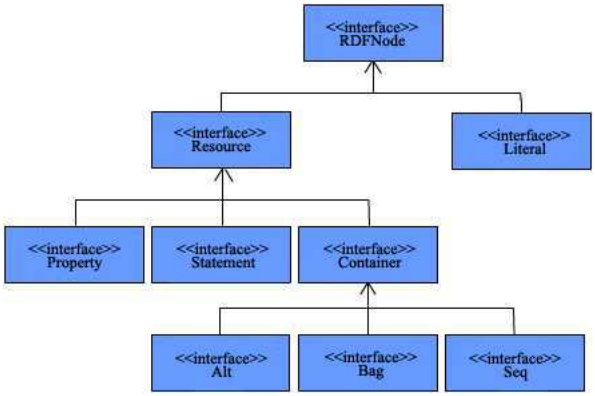


Figura 8: Interfaces Jena.
Fonte: Verzulli (2001).

W3C (2010a) explica que tecnicamente, as consultas SPARQL são baseadas em triplas (padrões). RDF pode ser vista como um conjunto de relacionamentos entre recursos; consultas SPARQL proveem um ou mais padrões diante desses relacionamentos. Estes padrões de triplas são similares a triplas RDF, exceto que um ou mais das suas referências de recursos são variáveis. Uma engine SPARQL retorna os recursos para todas as triplas que atendam a estes padrões.

Pode-se visualizar, e maneira mais didática uma consulta simples em SPARQL como uma tripla RDF (sujeito, predicado, objeto) contendo incógnita(s), por exemplo:

- (sujeito?, predicado, objeto)
- (sujeito, predicado?, objeto)
- (sujeito, predicado, objeto?)
- (sujeito?, predicado, objeto?)
- dentre outras possibilidades, onde a presença do "?" representa a incógnita que se deseja consultar.

Destaca-se, neste ponto, a ferramenta de consulta SPARQL Twinkle[15] que auxilia na construção de consultas SPARQL, carregando, editando e salvando consultas, inserindo prefixos em consultas, configurando *namespaces* customizados, salvando resultados de consultas, acessando diversos tipos de bases RDF, dentre outros recursos interessantes. Essa ferramenta foi construída baseada no ARQ[16], uma parte da API Jena.

[15] `http://www.ldodds.com/projects/twinkle/`
[16] `http://jena.hpl.hp.com/ARQ/`

5.4 Prototipação

5.4.1 Paradigma de obtenção de informação

A Figura 9 ilustra como se pode obter informação, nos dias atuais, a partir de meios computacionais. Geralmente, se têm uma série de aplicativos, dos quais se pode obter algumas informações, e para gerar algum conhecimento ou informação relevante e útil, realiza-se cruzamentos das informações provenientes desses aplicativos.

Esses cruzamentos de informações são feitos, geralmente, sem auxilio computacional. Um exemplo simples e ilustrado por Berners-Lee, Hendler e Lassila (2001) seria o agendamento de uma consulta em um medico, tendo que consultar um aplicativo de agenda e cruzar as informações de datas de compromissos com as disponibilidades de consultas informadas por algum sistema de clínica médica e, a partir de todo este trabalho manual, decidir acerca do agendamento da consulta.

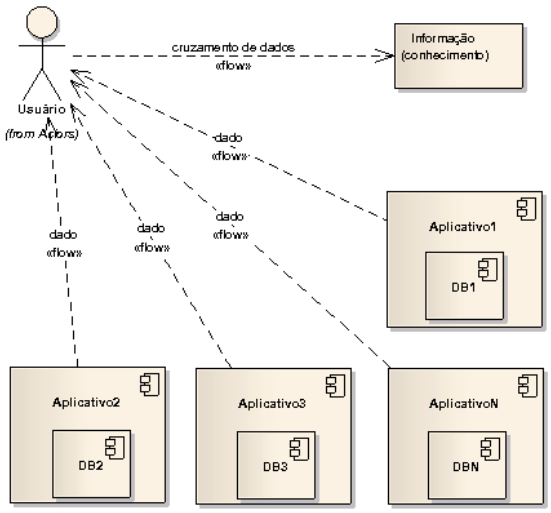


Figura 9: Atual paradigma de obtenção de informação.
Fonte: Do autor.

A proposta de modelagem e implementação de base de conhecimentos deste trabalho vem servir como uma alternativa viável à esse paradigma, onde o usuário consultaria apenas o aplicativo cliente da base de conhecimentos para obter a informação relevante, de maneira que ela seja suficiente ao usuário, liberando este, ao máximo, de cruzamentos de dados manuais.

(BERNERS-LEE; HENDLER; LASSILA, 2001) não explica os detalhes de implementação dos agentes de software que irão rodar na web semântica e no caso da proposta deste trabalho não se preocupou, também, em definir como seriam os aplicativos clientes

da base de conhecimentos, apenas que a base de conhecimentos tem interface *web services*, facilitando a implementação do aplicativos clientes em qualquer tipo de proposta que possa implementar cliente *web services*.

Desta maneira o usuário consulta o aplicativo, este consulta a base de conhecimentos via *web services* e esta retorna a informação que o usuário deseja de maneira direta ou "inferida"computacionalmente. Vide 10.

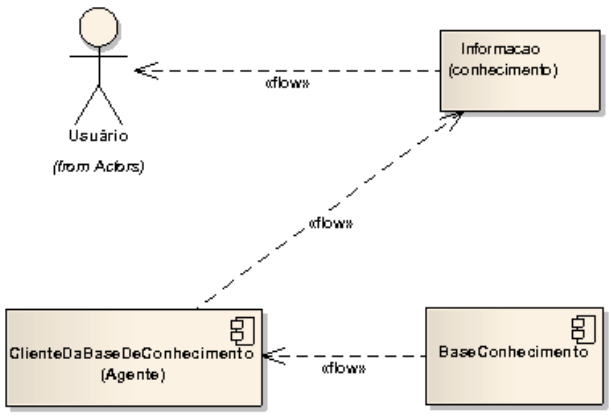


Figura 10: Novo paradigma de obtenção de informação.
Fonte: Do autor.

5.4.2 Requisitos

Definiu-se os seguintes requisitos para proposta de implementação do sistema de base de conhecimentos:

- Gerenciamento da base de conhecimentos
  - Verificar estado da base de conhecimentos (implementado).
    - Quais *namespaces* estão presentes na base, quantos recursos e propriedades a base tem, nela cadastrados, dentre outros.
  - Checar *logging* de eventos sobre a base de conhecimentos (implementado).
    - A base de conhecimentos deve registrar todos os eventos de alteração dos recursos e propriedades RDF dela.
  - Gerenciar concorrência de acesso à base de conhecimentos
  - Customizar a base de conhecimentos
    - A base de conhecimentos deve suportar customizações, como mudança de base de dados, dentre outras.

- Operação sobre a base de conhecimento
  - Manter armazenamento das informações da base de conhecimentos (implementado).
    - em memória e banco de dados
  - Popular a base de conhecimentos (parcialmente implementado).
    - via arquivo local, URL de arquivo remoto, ou edição via declarações semânticas na base de conhecimento
  - Executar combinações na base de conhecimento armazenada em memória e banco de dados (Implementado).
    - como adição, união, diferença, intersecção e verificação de igualdade.
  - Interrogar a base de conhecimentos (implementado).
    - via consultas simples, navegação RDF e consultas SPARQL.
  - Efetuar raciocínio sobre a base de conhecimento (implementado).
    - como validação e inferência normal ou por regras.
  - Exportar os dados da base de conhecimentos (não implementado).
    - em XML ou outros formatos, como *turttle*.
  - Executar tarefas de liberação de recursos quando o sistema de base de conhecimentos for finalizado (implementado).

Dentre estes requisitos, apenas a interrogação da base de conhecimentos fica exposta ao usuário final, como um agente ou cliente *web services* que necessita obter alguma informação contida na base. As demais funcionalidades são expostas apenas aos mantenedores da base de conhecimento.

5.4.3 Implementação

Neste trabalho foi implementado a arquitetura ilustrada pela Figura 11

O núcleo da API Jena para gerenciamento de Ontologia foi inserido no BaseBean, por este motivo ele é um *bean singleton*. As demais funcionalidades satélites do Jena foram inseridas nas outras funcionalidades do sistema localizadas no *bean* OntologiaServico, um bean stateless. Para comunicação com o meio externo foi criada uma fachada *web services* implementada no bean OntologiaFacade. Como banco de dados foi utilizado o MySql[17],

[17] `http://dev.mysql.com/`

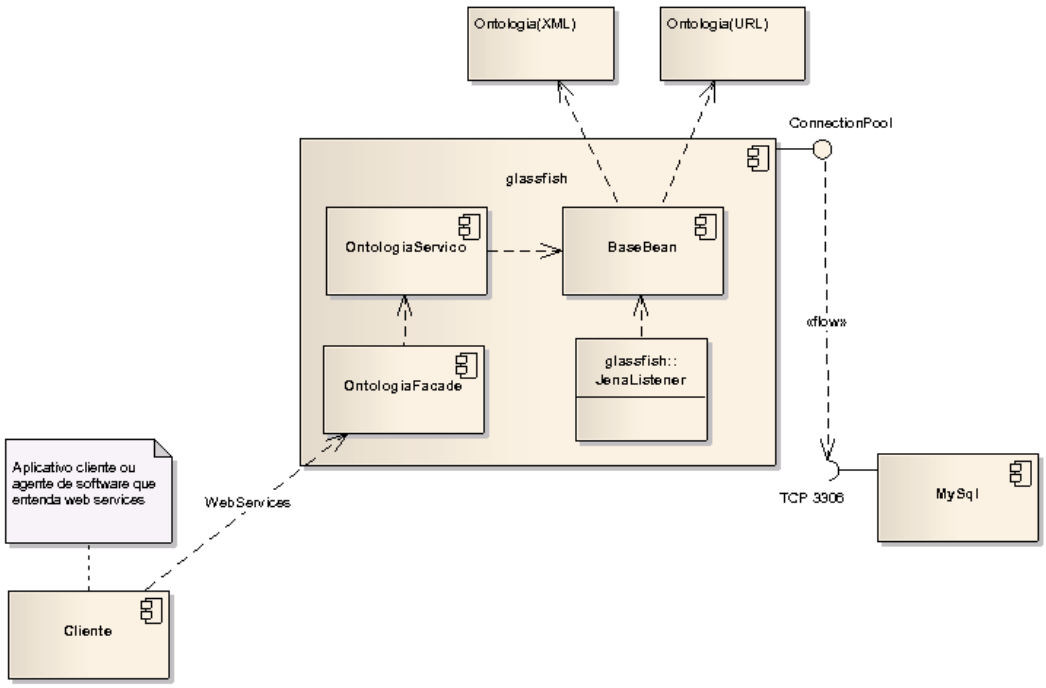


Figura 11: Arquitetura do sistema de base de conhecimentos.
Fonte: Do autor.

que é acessado pelo servidor de aplicação (*glassfish*) via *pool* de conexões. Pode-se verificar a implementação das classes do protótipo através do diagrama de classes na figura 12.

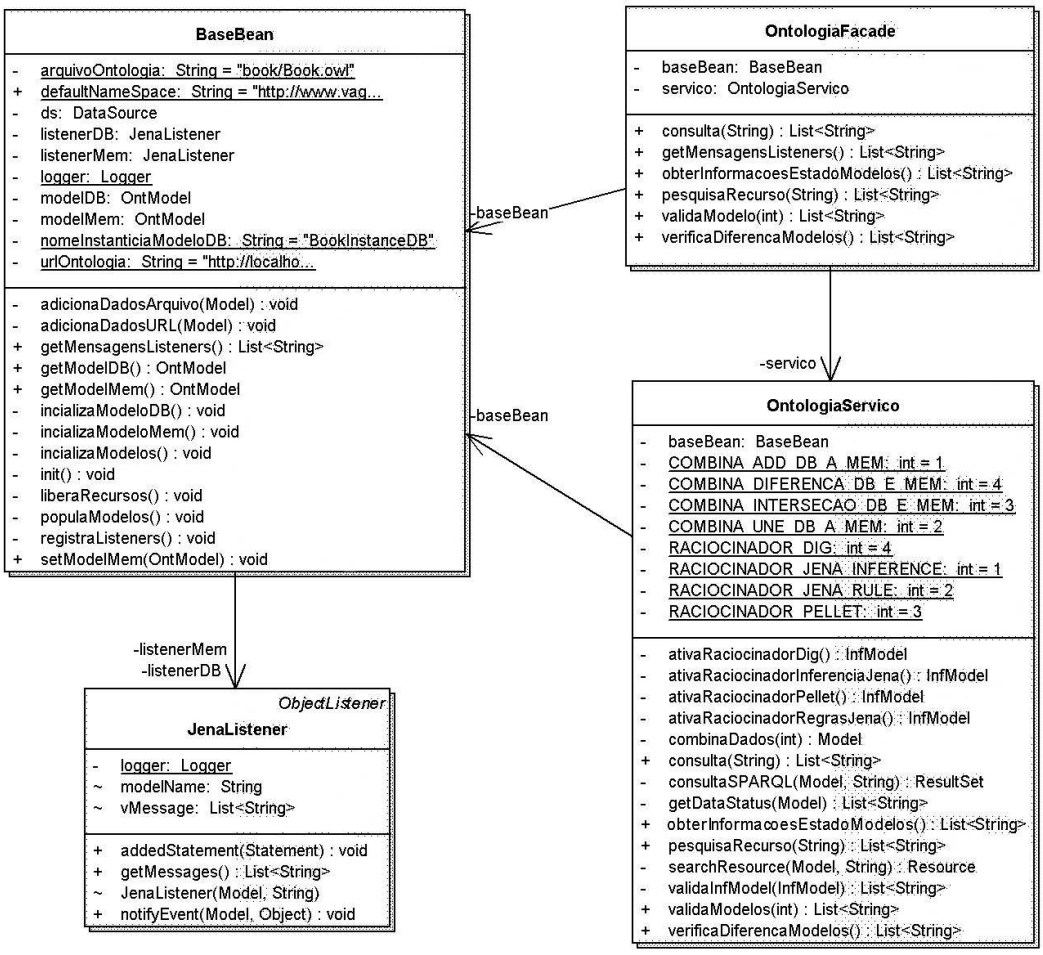


Figura 12: Diagrama de classes.
Fonte: Do autor.

6 Considerações Finais

6.1 Análise dos resultados

Neste trabalho pôde-se atingir os objetivos propostos, modelando uma base de conhecimento e mostrando a implementação da mesma. Embora nem todos os casos de uso tenham sido implementados, foi construído um protótipo razoável com as principais funcionalidades propostas.

Ficou, então, registrada, a contribuição deste trabalho para o desenvolvimento da web semântica no sentido de trazer alguns de seus padrões e sua filosofia para o mundo real da implementação de sistemas. Ficou comprovado, também, que é possível implementar várias soluções para diversos problemas atuais, utilizando nossa proposta de implementação, bastando, para isso, modelar a Ontologia do tema do problema e realizar poucas customizações nos *beans* do projeto de implementação.

6.2 Trabalhos futuros

Dentre as propostas avaliadas como trabalhos futuros, lista-se as seguintes:

- Contribuir com a continuidade do desenvolvimento da API Jena, no sentido de melhorar sua camada de persistência, pois nota-se que ela pode ser melhorada por meio da API JPA, que é o padrão de persistência do JEE. O Jena poderia ser melhorado, também, no sentido de suportar completamente a versão 2 da linguagem OWL.
- Utilizar a OWL-API[1] que é uma API semelhante ao Jena, porém com suporte completo a nova versão da linguagem OWL, fazendo com que a base de conhecimento se torne também totalmente compatível com a nova especificação da linguagem OWL.

[1] `http://owlapi.sourceforge.net/`

- Efetuar testes de carga na infraestrutura proposta com ferramentas como o JMeter[2].
- Projetar um mecanismo automatizado para popular a ontologia, com um indexador automático (*web crawler*) para obter URLs de sites relevantes ao domínio de conhecimento e expor os dados desses sites em RDF para popular a ontologia. Um exemplo de indexador automático de sites é o Nutch[3] e sua utilização como ferramenta de suporte à web semântica pode ser encontrado em Braga e Gomes (2007).
- Efetuar mais documentações acerca das tecnologias da Web Semântica utilizadas neste trabalho. Um bom local para se coletar informações acerca da Web Semântica em geral é em Ribeiro (2010).

[2] `http://jakarta.apache.org/jmeter/`
[3] `http://nutch.apache.org/`

Referências Bibliográficas

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2001.

BRAGA Ângelo de B.; GOMES, M. N. *Web Semântica: Uso de Ontologias para o Desenvolvimento de Buscaa Inteligentes*. Monografia (Graduação) — Faculdade de Computação, Universidade Federal do Pará, Belém, 2007.

BRAY, T. et al. Namespaces in xml 1.0 (third edition). W3C, Cambridge, 2009. Disponível em: <http://www.w3.org/TR/REC-xml-names/>. Acesso em: 22 mai. 2010.

BURKE, B.; MONSON-HAEFEL, R. *Enterprise JavaBeans 3.0*. 5. ed. São Paulo: Pearson Prentice Hall, 2007.

CORP. ORACLE. What is java enterprise edition(j2ee)? Corp. Oracle, 2009. Disponível em: <http://www.java.com/en/download/faq/j2ee.xml>. Acesso em: 8 mai. 2010.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Sistemas Distribuídos Conceitos e Projeto*. 4. ed. Porto Alegre: Bookman, 2007.

GRUBER, T. R. Toward principies for the design of ontologies used for knowledge sharing. In: FORMAL ONTOLOGY IN CONCEPTUAL ANALYSIS AND KNOWLEDGE REPRESENTATION, 1993, Padova. *Technical Report KSL 93-04*. Stanford: Stanford University, 1993.

GUARINO, N. Formal ontology and information systems. In: PROCEEDING OF FOIS'98, 1998, Ternto. Amsterdam: lOS Press, 1998. p. 3–15.

HEBELER, J. et al. *Semantic Web Programming*. Indianapolis: Wiley Publishing, 2009.

HITZLER, P. et al. Owl 2 web ontology language primer. W3C, Cambridge, 2009. Disponível em: <http://www.w3.org/TR/2009/REC-owl2-primer-20091027/>. Acesso em: 22 mai. 2010.

HORRIDGE, M. A practical guide to building owl ontologies using protégé 4 and co-ode tools. The University Of Manchester, Manchester, March 2009.

LIMA, J. C. de; CARVALHO, C. L. de. Ontologias – owl (web ontology language). Universidade Federal de Goiás, Goiânia, junho 2005.

NOY, N. F.; MCGUINNESS, D. L. Ontology development 101: A guide to creating your first ontology. Stanford University, Stanford, 2001.

REAL, R. Redes semânticas. UFRGS, Porto Alegre, 2010. Disponível em: <http://www-.inf.ufrgs.br/gppd/disc/cmp135/trabs/rodrigo/T1/html/redes_ semanticas.html>. Acesso em: 8 mai. 2010.

RIBEIRO, A. L. Belém, 2010. Disponível em: <http://adagenor.blogspot.com/>. Acesso em: 15 jun. 2010.

ROSA, P. A. Web semântica. Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, dez. 2002.

STANFORD UNIVERSITY. What is protégé? Stanford University, Stanford, 2010. Disponível em: <http://protege.stanford.edu/overview/index.html>. Acesso em: 23 mai. 2010.

SUN MICROSYSTEMS INC. The java ee 6 tutorial. Sun Microsystems Inc., 2009.

UNICODE.ORG. What is unicode? unicode.org, 2009. Disponível em: <http://www-.unicode.org/standard/WhatIsUnicode.html>. Acesso em: 22 mai. 2010.

VERZULLI, J. Using the jena api to process rdf. O'Reilly, mai. 2001. Disponível em: <http://www.xml.com/pub/a/2001/05/23% -/jena.html>. Acesso em: 22 mai. 2010.

W3C. Extensible markup language (xml). W3C, Cambridge, 2003. Disponível em: <http://www.w3.org/XML/>. Acesso em: 22 mai. 2010.

W3C. Resource description framework (rdf). W3C, Cambridge, 2009. Disponível em: <http://www.w3.org/RDF/>. Acesso em: 22 mai. 2010.

W3C. Query. W3C, Cambridge, 2010. Disponível em: <http://www.w3.org/standards-/semanticweb/query>. Acesso em: 15 jun. 2010.

W3C. Semantic web. W3C, Cambridge, 2010. Disponível em: <http://www.w3.org-/standards/semanticweb/>. Acesso em: 22 mai. 2010.

W3C. Web services activity statement. W3C, Cambridge, 2010. Disponível em: <http://www.w3.org/2002/ws/Activity>. Acesso em: 10 mai. 2010.