Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

JULIO LÍVERO JUNIOR

WEB SEMÂNTICA, ONTOLOGIAS E LINGUAGEM OWL: UM ESTUDO DE CASO SOBRE A FERRAMENTA *PROTÉGÉ*

Assis

2012

JULIO LÍVERO JUNIOR

WEB SEMÂNTICA, ONTOLOGIAS E LINGUAGEM OWL: UM ESTUDO DE CASO SOBRE A FERRAMENTA *PROTÉGÉ*

Projeto de pesquisa apresentado ao Curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA à Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Área de Concentração: Web Semântica; Ontologia; Linguagem OWL; Banco de Dados

Assis

2012

FICHA CATALOGRÁFICA

LÍVERO JUNIOR, Julio
WEB SEMÂNTICA, ONTOLOGIAS E LINGUAGEM OWL: UM ESTUDO DE CASO SOBRE A FERRAMENTA *PROTÉGÉ* / Julio Lívero Junior. Fundação Educacional do Município de Assis – FEMA – Assis, 2012.
99 páginas.

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto.
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.
1.Web Semântica. 2.Ontologias. 3.OWL. 4.Protégé

CDD: 001.6
Biblioteca da FEMA

WEB SEMÂNTICA, ONTOLOGIAS E LINGUAGEM OWL: UM ESTUDO DE CASO SOBRE A FERRAMENTA *PROTÉGÉ*

JULIO LÍVERO JUNIOR

Trabalho de Conclusão de curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação analisado pela seguinte comissão examinadora:

Orientador: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Analisador: Prof. Esp. Fernando Cesar de Lima

ASSIS

2012

RESUMO

Com a constante evolução da *World Wide Web* eclodem novos padrões de desenvolvimento das páginas web. A Web Semântica surge com o objetivo de estruturar os conteúdos das páginas Web, possibilitando às máquinas compreender o sentido dos documentos. Estar atento a essas mudanças é de fundamental importância para os profissionais de Ciência da Computação. Dessa forma, o presente projeto tem por finalidade fazer um estudo de caso sobre a ferramenta de construção e edição de ontologias *Protégé-OWL*, sob a linguagem *OWL (Web Ontology Language)*. Demonstra-se o uso da ferramenta para a elaboração da proposta da Web Semântica.

Palavras-chave: Web Semântica, Ontologia, OWL, Protégé-OWL.

ABSTRACT

With the constant evolution of the World Wide Web hatch new web pages development patterns. The Semantic Web arises with the purpose of structuring the web pages content, enabling machines to understand the documents meaning. Be alert to these changes is fundamental for Computer Science professionals. In this way, this project is intended to do a case study of ontologies building and editing tool Protégé-OWL, under the OWL (Web Ontology Language). Demonstrates tool usage in the Semantic Web proposed.

Keywords: Semantic Web, Ontology, OWL, Protégé-OWL

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento da Web possibilitou o surgimento de um novo mecanismo de aquisição de conhecimento. A quantidade de informação e conhecimento acessíveis pela rede mundial cresceu de forma explosiva. Dessa maneira, a recuperação de informação relevante, de forma precisa, torna-se fundamental para os usuários da rede.

Em 2001 Berners-Lee, Hendler e Lassila publicaram um artigo na revista *Scientific American* intitulado: *The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities.* Neste artigo, os autores descrevem os cenários futuros da Web Semântica, que terá um papel fundamental no dia a dia dos indivíduos.

A Web Semântica surge com o objetivo de estruturar os conteúdos das páginas Web. Segundo Oliveira (2006, p.15), com o advento da Web Semântica “um novo modelo de representação do conhecimento, baseado em ontologias, está buscando formas de interligar os sistemas de informação... baseados em inferências e contextos”.

Isso não significa a extinção da Web atual. A Web Semântica não é separada da Web, mas sim uma extensão da Web atual, em que a informação é dada com um significado bem definido, permitindo que computadores e pessoas trabalhem em cooperação (BERNERS-LEE et al., 2001).

A Web Semântica tem sua base formada em Ontologias. Ontologia “É um documento que define as relações entre termos e conceitos” (BERNERS-LEE et al., 2001).

Segundo Almeida & Bax, (2003, p.1):

> Uma ontologia é criada por especialistas e define as regras que regulam a combinação entre termos e relações em um domínio do conhecimento. Os usuários formulam consultas usando conceitos definidos pela ontologia. O que se busca, em última instância, são melhorias nos processos de recuperação da informação.

Para que as regras de domínio sejam compreendidas pelas máquinas, faz-se necessário o uso de alguma estrutura e linguagem que apliquem ontologias. Uma das soluções possíveis é o uso de OWL – *Web Ontology Language*, dada como padrão pela W3C.

A OWL é uma linguagem de marcação semântica para publicação e compartilhamento de ontologias na *World Wide Web*. A OWL facilita a interpretação da máquina para os conteúdos da Web. Dessa maneira a OWL destina-se a ser utilizada quando a informação contida nos documentos necessita ser processada por aplicações, em oposição às situações em que o conteúdo das páginas web necessita apenas ser apresentada para os seres humanos. OWL pode ser usada, então, para representar explicitamente o significado de termos em vocabulários e as relações entre essas condições. (MCGUINNESS; HARMELEN, 2004).

A OWL fornece três sublinguagens: *OWL-Lite* – oferece suporte àqueles usuários que necessitam, principalmente, de uma hierarquia de classificação e restrições simples. *OWL-DL* – oferece suporte àqueles usuários que querem o máximo de expressividade, mantendo completude computacional (todas as conclusões são garantidas para ser computável) e poder de decisão (todas as computações terminarão em tempo finito). *OWL-Full* - é destinada a usuários que querem máxima expressividade e liberdade sintática do *RDF* (*Resource Description Framework*) sem garantias computacionais. (MCGUINNESS; HARMELEN, 2004)

Tendo posse da OWL faz-se necessário o uso de ferramentas que implementem tal linguagem. *Protégé* é a ferramenta escolhida como foco principal deste trabalho.

Usando os pontos de vista de interface do usuário do *Protégé*, designers de ontologia basicamente criam classes, atribuem propriedades para as classes, e, em seguida, restringem as facetas das propriedades em determinadas classes. Usando as ontologias resultantes, *Protégé* é capaz de gerar automaticamente interfaces de usuário que suportam a criação de indivíduos (instâncias). Para cada classe da ontologia, o sistema cria um formulário com componentes de edição *(widgets)* para cada propriedade da classe. (KNUBLAUCH, et al.)

Para demonstrar o uso da ferramenta foi selecionado o domínio de Classificação de Veículos conforme Tipo/Marca/Espécie seguindo a Portaria Número 1101, de 20 de

dezembro de 2011, alterada pela Portaria Número 309 de 15 de junho de 2012, do DENATRAN (Departamento Nacional de Trânsito).

1.1. OBJETIVOS

O presente projeto tem por objetivo fazer um estudo de caso sobre a ferramenta de construção e edição de ontologias *Protégé*, sob a linguagem *OWL*. Será verificada como a ferramenta implementa a arquitetura de ontologias com base no levantamento bibliográfico no que diz respeito à Web Semântica, Ontologias, e OWL.

1.2. JUSTIFICATIVAS

Carvalho (2009, p.150) ao abordar a Web Semântica afirma que:

> A WS resolve os problemas que impediam a Web se tornar um efetivo sistema de informação, mas ela caminhou pouco diante das propostas. Evidente que existe a necessidade de maturidade de usuários, dos desenvolvedores, dos pesquisadores e ferramentas para que isso aconteça.

Souza e Alvarenga (2004, p.140) afirmam que:

> ...a importância da Web e das demais redes digitais de troca de informações no panorama mundial são amostras de como a atividade de organização da informação é necessária para a evolução dos indivíduos, organizações e da sociedade em geral.

O projeto de pesquisa justifica-se no que tange a busca de conhecimento e tecnologias para a compreensão e aplicação da proposta da Web Semântica baseada em Ontologias. Bem como no que se refere à organização da informação, visando a padronização das páginas Web, impactando em buscas mais eficazes.

1.3. MOTIVAÇÃO

Com a constante evolução da Web eclodem novos padrões de desenvolvimento das páginas. Estar atento a essas mudanças é de fundamental importância para os profissionais de Ciência da Computação.

Analisar se conceitos de Web Semântica e linguagens, nas quais são formadas, são incorporados por ferramentas de desenvolvimento de ontologias faz-se necessário e de extrema importância na escolha de uma ferramenta adequada para padrões de projetos de desenvolvimento.

1.4. ESTRUTURA DO TRABALHO

O presente trabalho terá a seguinte disposição:

O Capítulo 1 terá a apresentação do tema abordado, bem como a estruturação do projeto de pesquisa.

No Capítulo 2 serão apresentados os conceitos de Web Semântica.

No Capítulo 3 será apresentada a definição do termo ontologia

O Capítulo 4 apresenta os conceitos básicos da Linguagem *XML* (*Extended Markup Language*) e da Linguagem *RDF* (*Resource Description Framework*), para que se possa apresentar a linguagem OWL.

No Capítulo 5 serão apresentados os conceitos e fundamentos da Linguagem *OWL*, bem como sua arquitetura.

Será apresentado no Capítulo 6 o estudo de caso sobre a ferramenta *Protégé*, no qual serão mostrados os conceitos, a arquitetura e a análise da ferramenta.

No último capítulo serão retomados os conceitos abordados no trabalho e será analisado se a ferramenta de construção e edição de ontologias *Protégé* atende aos conceitos de web semântica e ontologias.

2. WEB SEMÂNTICA

Este capítulo apresentará uma visão geral sobre a *Web semântica*. Serão apresentados os conceitos que envolvem este termo, as vantagens alcançadas ao utilizar tal estrutura de organização das páginas *Web,* bem como a estrutura recomendada pela W3C ao se implementar um projeto de Web Semântica.

2.1. ASPECTOS GERAIS DA WEB SEMÂNTICA

De forma global, a Web Semântica auxiliará a separar o sentido dos dados das páginas *Web* com o conteúdo de seus documentos, ou código de aplicação.

BERNERS-LEE et al., (2001) afirmam que,

> The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users.

A Web Semântica surge como uma possível solução para a estruturação semântica dos dados na web, viabilizando o processamento da informação por parte das máquinas. Se uma máquina possuir a capacidade de analisar a estrutura semântica de um documento, não interpretará apenas os caracteres que compõem esse documento, compreenderá o sentido deste documento. O que proporcionará a realização de sofisticadas tarefas para os usuários.

Segundo Carvalho (2009, p.115), a meta da Web Semântica é:

> ... criar um meio universal para a troca de dados, interconectando a administração de informações pessoais, integrando aplicações em empresas e compartilhando dados comerciais, científicos e culturais em escala global. A partir de instalações para colocar na Web dados que podem ser compreendidos por máquinas, com a ajuda de organizações, indivíduos e comunidades.

Segundo FERNEDA (2003, p. 110) o desafio da Web Semântica é:

> ...melhorar a recuperação de informação em grandes repositórios como a Web, pesquisas atualmente em curso estão buscando encontrar formas de possibilitar a agregação de um maior nível semântico às páginas Web. Procura-se aumentar a eficiência dos mecanismos de busca e de outros tipos de ferramentas de processamento automático de documentos através da utilização de linguagens que permitam definir dados e regras para o raciocínio sobre esses dados.

A Web Semântica, então, representa a evolução da Web atual e não uma proposição de uma nova Web. Enquanto a Web tradicional, também denominada Web Sintática, tem seu desenvolvimento para ser entendida apenas pelos usuários, a Web Semântica foi projetada para ser compreendida pelas máquinas, na forma de agentes computacionais, que são capazes de operar eficientemente sobre as informações, podendo entender seus significados.

A Web Semântica pretende fornecer estruturas e dar significado semântico ao conteúdo das páginas web, criando um ambiente no qual agentes de software e usuários possam trabalhar de forma cooperativa.

Neste novo contexto, a web será capaz de representar associações entre coisas que, em princípio, poderiam não estar relacionadas. Segundo Berners-Lee (2001, p.3) os computadores necessitam ter acesso a coleções estruturadas de informações (dados e metadados) e de conjuntos de regras de inferência que ajudem no processo de dedução automática para que seja administrado o raciocínio automatizado, ou seja, a representação do conhecimento.

Tais regras são especificadas através de ontologias, que permitem representar explicitamente a semântica dos dados. Através dessas ontologias é possível elaborar uma ampla rede de conhecimento humano, complementando o processamento da máquina e melhorando qualitativamente o nível de serviços na web.

2.2. ARQUITETURA DE WEB SEMÂNTICA

A *W3C – World Wide Web Consortium* – definiu uma estrutura em camadas (Figura 1) que reflete os passos que devem ser dados para que o projeto da Web Semântica seja realizado.

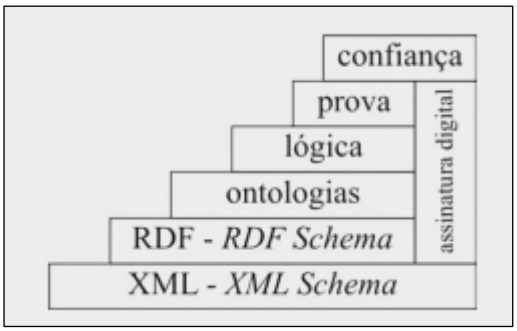

Figura 1 – Arquitetura Web Semântica (In: FERNEDA, 2003, p. 111).

Nesta arquitetura, na primeira camada, a linguagem *XML* permite definir documentos Web com *marcações personalizadas* para garantir um maior nível semântico em relação às páginas *HTML.* Contudo, faz-se necessário formalizar e validar a estrutura de páginas *XML* para garantir que estejam corretamente definidas. Para isso utiliza-se a linguagem *XML Schema*.

Na segunda camada a linguagem *RDF* fornece um meio de agregar semântica a um documento sem se referir à sua estrutura. A linguagem define um modelo para descrever relacionamentos entre recursos através de suas propriedades e valores. Contudo, a linguagem *RDF* não fornece mecanismos para declarar as entidades e também não possibilita definir tais relacionamentos. Para suprir tais necessidades utiliza-se *RDF Schema*, que permite gerar um sistema de classes extensível e genérico que pode ser utilizado como base para a descrição conceitual de um domínio específico.

A camada de ontologias pode ser considerada como a espinha dorsal da Web Semântica. Segundo FERNEDA (2003, p.116)

> Na maioria das vezes uma ontologia toma a forma de uma árvore hierárquica de classes, de maneira que cada classe herda as características de uma ou mais classes superiores. Cada classe representa um conceito do domínio que está sendo modelado, e seu significado é expresso pelas suas propriedades, similaridades e diferenças em relação aos outros conceitos.

Ao ordenarem os termos, as ontologias incorporam à Web a preocupação com a organização da informação e, por consequência, de atribuição de significado a esses termos.

A camada de Lógica é definida por FERNEDA (2003, p.119), como sendo “... um conjunto de regras de inferência que os agentes (computacionais ou humanos) poderão utilizar para relacionar e processar informação.” A partir de tais regras, os agentes podem raciocinar ou inferir informações sobre as estruturas de dados definidas nas camadas *XML* e *RDF* utilizando as relações, definidas na camada de ontologia, entre esses objetos.

A camada de Prova tem a função de executar as regras da camada lógica, funcionando como um mecanismo de inferência da Web Semântica.

A camada de Confiança tem o propósito de avaliar se a prova está correta ou não. Isso é feito através da verificação da autenticidade e resolução de contradições. Segundo FERNEDA (2003, p.120):

> A autenticidade e confiabilidade das fontes adquirem um novo significado quando consideramos que agentes raciocinando sobre os dados podem chegar a conclusões que afetem a ação humana. As assinaturas digitais serão a forma de cada agente verificar a autenticidade das suas fontes. De acordo com a informação que a assinatura digital lhe fornecer, o agente poderá alterar o grau de certeza associado ao resultado do seu raciocínio ou mesmo ignorar a informação.

A partir dessa arquitetura, os softwares chamados agentes inteligentes utilizarão toda a informação semanticamente marcada pelas camadas para solucionar problemas informacionais dos usuários, como também trocar essas informações com outros agentes da Web.

2.3. PROJETOS COM WEB SEMÂNTICA

Com base na proposta da Web Semântica, algumas empresas estão adaptando seus produtos para atingir um melhor desempenho de seus recursos.

O Google, por exemplo, anunciou que implementará recursos de Web Semântica em seu buscador. O recurso permitirá que as máquinas consigam tomar decisões sozinhas baseadas em padrões previamente estabelecidos.

O recurso permitirá que o computador reconheça as características de cada usuário

e faça inferências sobre quais resultados aquela pessoa quer obter ao buscar determinadas palavras na web. O cálculo é feito a partir da análise do comportamento daquele usuário na internet.

Segundo Efrat (2012), o Google não está substituindo seu sistema de busca atual com base em palavras-chave, que determina a importância de um site com base nas palavras que ele contém, em vez disso, a empresa está com o objetivo de fornecer resultados mais relevantes pela tecnologia incorporando a busca semântica, que se refere ao processo de compreensão do real significado das palavras.

Ainda segundo Efrat (2012), alguns especialistas em busca semântica também acreditam que a mudança vai ajudar o Google a manter-se com o Facebook, a rede social que acumulou um banco de dados sobre centenas de milhões de pessoas, lugares e coisas, mas não ofereceu um serviço de busca robusta.

No entanto, será preciso que os dados publicados na internet sejam acompanhados de descrições baseadas nos padrões W3C, para que os agentes do Google sejam capazes de compreender o conteúdo das páginas web. Por meios destes códigos, os computadores vão identificar o que representa um nome, um endereço ou uma cidade para cada tipo de pessoa.

Porém, mesmo com o anúncio do Google de que tais mudanças darão início no ano de 2012, há a necessidade de que os desenvolvedores de conteúdo na web introduzam dados semânticos em seus textos, fotos e vídeos publicados na web.

Outra empresa que está trabalhando com busca semântica é o Wallmart. Segundo Ribeiro (2012), “essa nova busca não só ajuda os usuários a encontrarem os artigos desejados no seu site, como também proporciona resultados com base em prováveis interesses e intenções”.

Desenvolvida pela equipe do WalmartLabs (centro de tecnologia e investigação da varejista), a nova ferramenta de busca foi denominada de Polaris. Baseia-se no projeto Social Genome do laboratório que utiliza dados públicos na Internet, dados proprietários, e mídias sociais, para identificar entidades e relacionamentos interessantes – e adicioná-los para a Social Genome.

Segundo Ribeiro (2012):

> O motor de busca usa algoritmos como a compreensão de consulta e a extração de sinônimos para descobrir intenções do utilizador na obtenção de resultados. Como resultado, se um usuário digita a palavra “denim”, ele disponibiliza resultados para “jeans”, enquanto “pastilhas de cloro” gera resultados relacionados com equipamento de piscina – segundo os exemplos fornecidos pela empresa.

A partir deste novo motor de busca, o site da Walmart onde a nova busca esteve em uso nos últimos meses já registou uma aumento de 10% a 15 % no número de clientes que completam uma compra depois de procurar por um produto usando o Polaris – de acordo com a empresa. A varejista planeja lançar a tecnologia, que também pode ser usada para pesquisas em mobilidade, nos seus sites internacionais de comércio eletrônico nos próximos meses.

A Microsoft também dá seus passos em relação à busca semântica. A tecnologia *Microsoft Powerset* está sendo usada para proporcionar uma melhor experiência de pesquisa no *Wikipedia*.

Segundo Brinkmann (2009), com o uso do *Powerset* não só é possível digitar palavras-chave específicas, como Google Chrome ou Microsoft, mas também questões como "o que é que Benjamin Franklin inventou" ou frases como "pintores impressionistas". Wikipedia vai retornar uma lista de resultados possíveis para essas consultas, deixando o usuário com a tarefa de olhar através deles para encontrar o artigo que contém as informações corretas.

3. ONTOLOGIAS

Este capítulo abordará conceitos relacionados ao termo Ontologias no contexto da representação da informação. Será efetuada a conceituação do termo, quais suas aplicabilidades, bem como as forma como ontologias podem ser representadas.

Originalmente o termo Ontologia provém da filosofia. Aristóteles o utilizava para designar a ciência das coisas de acordo com a sua essência. Hoje o termo refere-se também no contexto da representação da informação e da Inteligência Artificial sendo tratado como uma especificação explícita de um conceito.

Segundo OLIVEIRA (2006, p. 56):

> A tentativa de associar conceitos em um ambiente eletrônico da mesma forma com que é feito no cérebro humano é o grande desafio da representação do conhecimento. Para tanto, as ontologias revelam-se numa solução promissora para fazer com que os computadores representem e encontrem informações de uma forma intuitiva, como fazem os humanos.

Segundo SEGARAN, EVANS & TAYLOR(2009, p.128):

> An ontology provides a precise vocabulary with which knowledge can be represented. This vocabulary allows us to specify which entities will be represented, how they can be grouped, and what relationships connect them together.

Com o uso de ontologias, então, é possível representar o conteúdo das informações, através de um vocabulário que especifica entidades, como são agrupadas e quais relações possuem.

A representação das informações em uma ontologia é organizada de forma hierárquica “como se fossem árvores conceituais cujas folhas são os termos e seus respectivos sinônimos, antônimos ou qualquer outra relação semântica que se queira atribuir.” OLIVEIRA (2006, p. 57).

Nem sempre as ontologias possuem a mesma estrutura, contudo existem componentes comuns que está presente em sua maioria. Tais elementos são apresentados por ALMEIDA & BAX (2003, p. 09):

> Classes (organizadas em uma taxonomia), relações (representam os tipos de interação entre os conceitos de um domínio), axiomas (usados para modelar sentenças sempre verdadeiras) e instâncias (utilizadas para representar elementos específicos, ou seja, os próprios dados).

As Ontologias podem ser visualizadas de forma gráfica (Figura 2), em forma de taxonomias (Figura 3) e em forma de código fonte *XML* estendida pelo o uso de *RDF* e *OWL* (Figura 4).

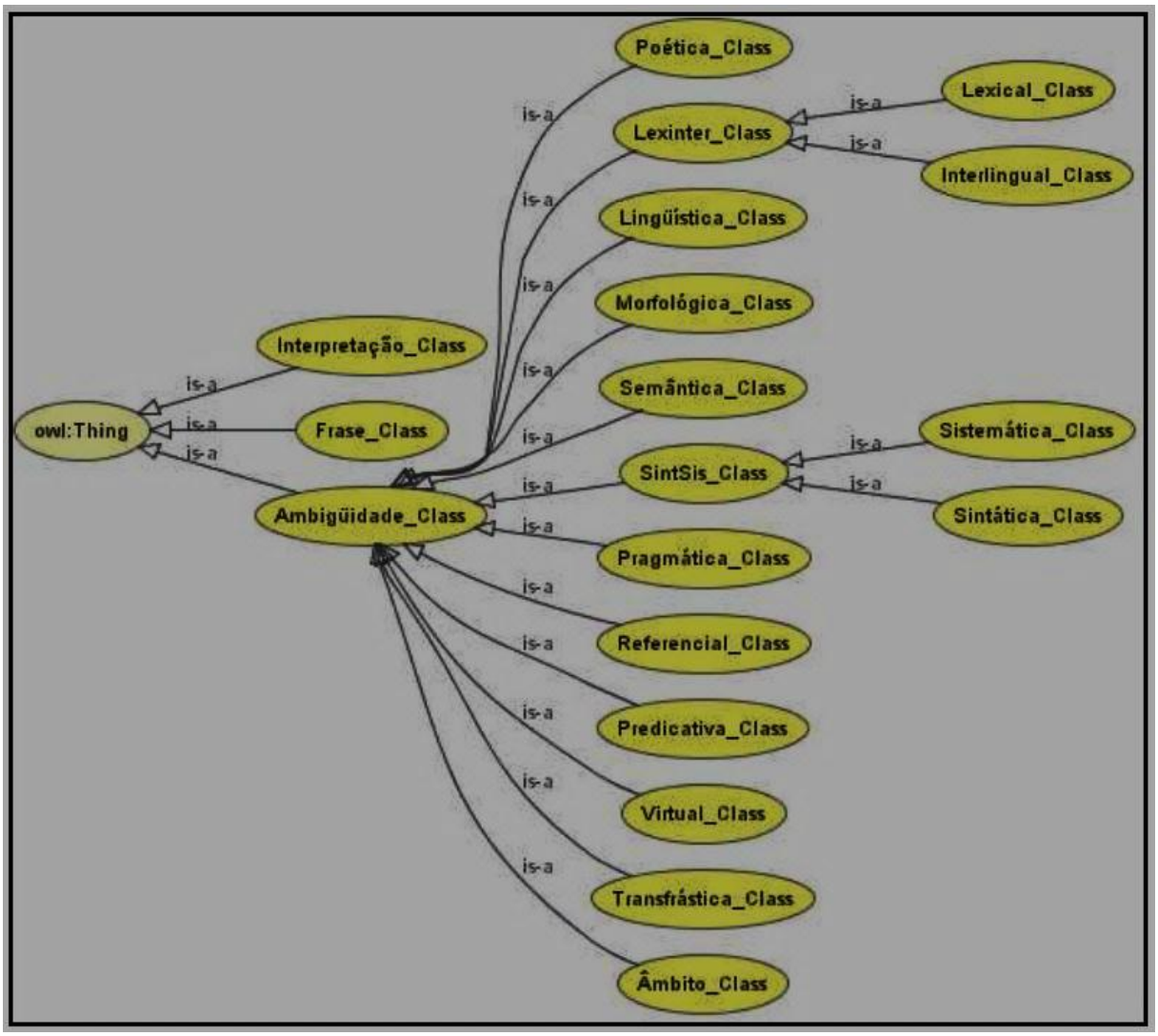

Figura 2 - Exemplo de uma Ontologia (In: OLIVEIRA, 2006, p. 60).

Tabela 1: Taxonomia enumerativa das ambigüidades da Língua Portuguesa

	Classificação	Subclassificação
Ambigüidades	Âmbito	
	Interlingual	
	Lexinter	Lexical
		Interlingual
	Lingüística	
	Morfológica	
	Poética	
	Pragmática	
	Predicativa	
	Referencial	
	Semântica	
	Sintsis	Sistemática
		Sintática
	Transfrástica	
	Virtual	

Figura 3 – Exemplo de Taxonomia (In: OLIVEIRA, 2006, p. 60).

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Lingüística_Class">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >gerada apenas por questões lingüísticas e é detectada quando
determinados enunciados em condições já previstas apresentam problemas de
escolha lingüística ao receptor, gerando uma flutuação entre duas ou mais
condições aceitáveis.
Não se contamina com aspectos não lingüísticos</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Ambigüidade_Class"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Virtual_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Lexinter_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="SintSis_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Âmbito_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Transfrástica_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Poética_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Pragmática_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Semântica_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Referencial_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Predicativa_Class"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Morfológica_Class"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Referencial_Class">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >está relacionada a enunciados potencialmente ou efetivamente
incompletos, possui a característica de ser uma ambigüidade elíptica. O
efeito elíptico determina que um enunciado não pode conter todas as
informações necessárias para o seu completo entendimento, logo é sempre
possível acrescentar um elemento que possa, necessariamente contribuir
para o seu entendimento causando um efeito de precisão,
modificação ou também causando mais contradição ao enunciado em
questão</rdfs:comment>
    <owl:disjointWith>
```

Figura 4 – Exemplo de Código-fonte (In: OLIVEIRA, 2006, p. 61).

Uma ontologia define os termos utilizados para descrever e representar uma área de conhecimento, por isso elas podem ser usadas por pessoas, bases de dados e aplicações que precisam compartilhar informações de domínios. Um domínio pode

ser um assunto ou área específica de conhecimento como medicina, botânica, gastronomia, ou qualquer outra área. As ontologias incluem definições básicas de conceitos, legíveis por computador, e as relações entre eles, ou seja, as ontologias codificam o conhecimento de um domínio, tornando esse conhecimento reutilizável por outros sistemas ou pessoas. (HEFLIN, 2004)

ALMEIDA e BAX (2003, p.10) sintetizam os tipos de ontologias relacionados à sua função, ao grau de formalismo de seu vocabulário, à sua aplicação e à sua estrutura e conteúdo da conceitualização.

Quanto à sua função as ontologias são classificadas em: Ontologias de Domínio – são ontologias que são reutilizáveis no domínio, fornecem vocabulário sobre conceitos, seus relacionamentos, sobre atividades e regras que os governam – Ontologias de tarefa – fornecem um vocabulário sistematizado de termos, especificando tarefas que podem ou não estar no mesmo domínio – Ontologias Gerais – incluem vocabulário relacionado a coisas, eventos, espaço, casualidade, comportamento, funções entre outros.

Quanto ao grau de formalismo as ontologias são classificadas em: Ontologias Altamente Informais – expressas livremente em linguagem natural – Ontologias Semi-informais – expressas em linguagem natural de forma restrita e estruturada – Ontologias Semiformais – expressa em uma linguagem artificial definida formalmente – Ontologia Rigorosamente Formal – os termos são definidos com semântica formal, teoremas e provas.

Quanto às aplicações as ontologias são classificadas em: Ontologias de autoria neutra – um aplicativo é escrito em uma única língua e depois convertido para uso em diversos sistemas, reutilizando-se as informações – Ontologias como especificação – cria-se uma ontologia para um domínio, na qual é usada para documentação e manutenção no desenvolvimento de softwares – Ontologias de acesso comum à informação – quando o vocabulário é inascessível, a ontologia torna a informação inteligível, proporcionando conhecimento compartilhado dos termos.

Quanto à sua estrutura, as ontologias são classificadas em: Ontologias de alto nível – descrevem conceitos gerais relacionados a todos os elementos da ontologia

(espaço, tempo, matéria, objeto, evento, ação, entre outras) nos quais são independentes de problema ou domínio – Ontologias de Domínio – descrevem o vocabulário relacionado a um domínio, como por exemplo, medicina ou automóveis – Ontologias de tarefa – descrevem uma tarefa ou atividade, como por exemplo, diagnósticos ou compras, mediante inserção de termos especializados na ontologia.

Quanto ao conteúdo, as Ontologias são classificadas em: Ontologias terminológicas – especificam termos que serão usados para representar o conhecimento em um domínio – Ontologias de informação – especificam a estrutura de registros de banco de dados – Ontologias de modelagem do conhecimento – especificam conceitualizações do conhecimento, possuem uma estrutura interna semanticamente rica e são refinadas para uso no domínio do conhecimento que descrevem – Ontologias de aplicação – contém as definições necessárias para modelar o conhecimento em uma aplicação – Ontologias de domínio – expressam conceitualizações que são específicas para um determinado domínio de conhecimento – Ontologias genéricas – similares às ontologias de domínio, porém os conceitos que as definem são considerados genéricos e comuns a vários campos – Ontologias de representação – explicam as conceitualizações que estão por trás dos formalismos de representação do conhecimento.

Segundo Oliveira (2006, p.63), com a ajuda das ontologias:

> ... a pesquisa e visualização da informação pode ser feita por meio de especializações ou generalizações de termos usados para identificar a informação, expandindo assim as possibilidades de formulação de queries de pesquisa em bancos de dados. Isso só é permitido por meio de relações lógicas numéricas, atribuindo assim uma linguagem formal que expressa o conhecimento e provê uma semântica formal bem compreendida com a ajuda de agentes inteligentes (robôs) que tornam o conhecimento tácito em explícito. Tal transformação acontece por meio de mecanismos de inferência que são deduzidos das formas de representação do conhecimento, é chamada de raciocínio ontológico de agentes inteligentes.

Por fim, tem-se que o estado da arte revela a tendência de se usarem ontologias como base para a representação do conhecimento, compartilhamento, gestão, modelagem, engenharia, educação, dentre outras. Dessa forma, tem-se que as ontologias são um recurso potencial para a produção de conhecimento para a Web Semântica.

4. XML E RDF

Atualmente, um grande número de páginas Web é constituído sob o formato *HTML* (*HyperText Markup Language*). Dessa forma, grande parte dos documentos acessados na Web está inserida nessas páginas. Porém, esta linguagem, normalmente, não possui nenhuma informação extra, capaz de otimizar a localização das páginas. Limita-se a indexar, em forma de hipertexto, de maneira não ordenada, os dados inseridos na página.

Esta restrição inviabiliza o uso de HTML no contexto da Web Semântica, pois a informação sobre os dados é fundamental na constituição da Web Semântica. Segundo Oliveira (2006, p.52), “representar informações via metadados é um elemento crucial de modo a recuperar informações depois de publicadas”.

A informação sobre os dados, ou “dados sobre dados” é denominada metadados. Metadados podem ser considerados uma documentação onde estaria a descrição sobre as informações e suas respectivas ligações, organizadas de forma estruturada visando facilitar o acesso e manutenção desses dados.

Para tal, a Web Semântica é formada sobre linguagens que possibilitem a utilização de metadados. Dessa forma, ela é constituída com base nas linguagens *XML* e *RDF.*

Os subcapítulos seguintes abordarão os conceitos relacionados às linguagens de marcação *XML* e *RDF.* Serão consideradas suas características e sua importância na construção de Ontologias.

4.1. XML (EXTENDED MARKUP LANGUAGE)

A linguagem *XML* (*Extended Markup Language*) surge em 1996 com a premissa de “resolver” as limitações da Linguagem *HTML* (*HyperText Markup Language*). A linguagem de marcação *XML* possui como principal característica a possibilidade de definir um número ilimitado de *tags*. A Figura 5 demonstra uma comparação básica entre as linguagens *Html* e *XML.*

HTML

```
<html>
  <body>
  Microcomputador Pentium 4, 1.5 GHz,
256MB de RAM, Monitor 17 polegadas,
mouse, teclado, estabilizador.
  </body>
</html>
```

XML

```
<microcomputador>
   <modelo>Pentium 4</modelo>
   <velocidade>1.5 GHz</velocidade>
   <ram>256Mb de memória</ram>
   <monitor>17 polegadas</monitor>
   <teclado>Sim</teclaco>
   <mouse>Sim</mouse>
   <estabilizador>Sim</estabilizador>
   <impressora>Não</impressora>
</microcomputador>
```

Figura 5 – Comparação entre as Linguagens HTML e XML (In: FERNEDA, 2003, p. 105).

Nota-se que em ambas as linguagens há a premissa de se representar as características de um microcomputador, porém, a linguagem *XML* "possibilita discriminar cada um das características e apresentar o dado relacionado à característica" (FERNEDA, 2003, p. 105).

Segundo Oliveira (2006, p.52),

> Hoje a Web está sofrendo uma mudança de linguagem: o HTML deixa de ser a linguagem de autoria mais usada, por ser simples demais para a troca de informação, ou seja, não oferece uma estrutura nem uma semântica da informação. O XML passa a ocupar seu lugar, por dar suporte à representação da informação com base em um vocabulário compartilhado e com base no significado da informação, ou seja, nos metadados.

Também pode ser vinculada à *XML* uma especificação de um esquema. Tal especificação é opcional na criação de um documento *XML*, porém ela se faz importante para manter a consistência do documento *XML* verificando sua validade perante o esquema definido previamente. Os principais tipos de esquemas em *XML* são o *DTD* e *XML Schema,*

Segundo FERNEDA (2003, p. 106 - 107):

> A DTD (*Document Type Definition*) é um arquivo do tipo texto onde são definidas as *tags*, a ordem em que elas devem aparecer no documento XML e sua obrigatoriedade. Essas definições são feitas com a utilização de uma meta-linguagem cuja sintaxe difere significativamente da sintaxe XML... Na maioria das vezes dois documentos, XML e DTD, trabalham em conjunto em uma página Web. Com a ajuda da DTD, o *browser* consegue verificar todos os detalhes do documento XML e informar alguma inconsistência.

> (...)
>
> A linguagem *XML Schema,* apesar de ter a mesma função da DTD, possui muitas características que a torna mais poderosa (e mais complexa) do que a DTD. Com a *XML Schema* é possível não apenas especificar a sintaxe de um documento XML, mas também especificar os tipos de dados de cada elemento desse documento. É possível também reutilizar a definição de elementos de outros esquemas, criar tipos de dados personalizados, especificar o número mínimo e máximo de vezes que um elemento pode ocorrer, criar listas e grupos de atributo.

A Figura 6 apresenta um documento *DTD* relacionado a um documento *XML.* A Figura 7 apresenta um documento *XML Schema* relacionado a um documento *XML.* Já, a Figura 8 apresenta uma comparação entre *DTD* e *XML Schema.*

DTD (arquivo: "livro.dtd")

```
<!ELEMENT livro (titulo,genero?,autor+,editora)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT genero (#PCDATA)>
<!ELEMENT autor (nome, dtnasc)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT dtnasc (#PCDATA)>
<!ELEMENT editora (#PCDATA)>
```

XML

```
<!DOCTYPE livro SYSTEM "livro.dtd">
<livro>
  <titulo>A Rosa do Povo</titulo>
  <genero>poesia</genero>
  <autor>
    <nome>Carlos Drummond de Andrade</nome>
    <dtnasc>1902-10-31</dtnasc>
  </autor>
  <editora>José Olympio</editora>
</livro>
```

Figura 6 – Exemplo de utilização de uma DTD em um documento XML (In: FERNEDA, 2003, p. 106).

XML Schema (http://sites.uol.com.br/ferneda/livro.xsd)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="livro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titulo"  type="xs:string"/>
        <xs:element name="genero"  type="xs:string"/>
        <xs:element name="autor"   type="TAutor" minOccurs="1"/>
        <xs:element name="editora" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="TAutor">
    <xs:sequence>
      <xs:element name="nome"    type="xs:string"/>
      <xs:element name="dtnasc"  type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

XML

```
<livro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://sites.uol.com.br/ferneda/livro.xsd">
  <titulo>A Rosa do Povo</titulo>
  <genero>poesia</genero>
  <autor>
    <nome>Carlos Drummond de Andrade</nome>
    <dtnasc>1902-10-31</dtnasc>
  </autor>
  <editora>Jose Olympio</editora>
</livro>
```

Figura 7 – Exemplo de utilização de um XML Schema em um documento XML (In: FERNEDA, 2003, p. 109).

DTD

```
<!ELEMENT livro (titulo,genero?,autor+,editora)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT genero (#PCDATA)>
<!ELEMENT autor (nome, dtnasc)>
<!ELEMENT editora (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT dtnasc (#PCDATA)>
```

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="livro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="titulo"  type="xs:string"/>
        <xs:element name="genero"  type="xs:string"/>
        <xs:element name="autor"   type="TAutor" minOccurs="1"/>
        <xs:element name="editora" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="TAutor">
    <xs:sequence>
      <xs:element name="nome"    type="xs:string"/>
      <xs:element name="dtnasc"  type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Figura 8 – Comparação entre DTD e XML Schema (In: FERNEDA, 2003, p. 108).

Uma das principais razões para utilização do XML na Web Semântica é devido à possibilidade de auto definição e descrição de documentos, mas apresenta limitação no que diz respeito aos significados das informações.

XML é uma linguagem flexível, pois possibilita que os próprios usuários podem criar entidades para a descrição de seus dados. Do ponto de vista da Web Semântica essa flexibilidade vem a ser um problema, pois os conteúdos desses documentos ainda possuem suas entidades definidas por humanos, e para o reconhecimento de cada entidade ainda haveria a necessidade da intervenção humana. Nesses termos, já que um documento XML foi criado por uma pessoa, pode-se afirmar que somente o próprio autor do arquivo XML é capaz de responder o significado de cada entidade criada por ele em seus arquivos.

Procurando minimizar esse problema, a Web Semântica utiliza o *XML Schema* com a intenção de validar os dados de um arquivo XML conforme seus tipos de dados, contendo ainda um vocabulário dos significados das entidades e ainda a descrição dos relacionamentos dos dados.

4.2. RDF (RESOURCE DESCRIPTION FRAMEWORK)

A linguagem *RDF* (*Resource Description Framework*) tem um papel essencial na criação da Web Semântica, possibilitando o compartilhamento dos dados entre pessoas e entre máquinas. “RDF provides a standard way of expressing graphs of data and sharing them with other people and, perhaps more importantly, with machines” (SEGARAN, et al., 2009, p. 64).

O RDF é uma estrutura de descrição de recurso desenvolvida para representar recursos, modelar metadados e permitir o gerenciamento da informação na WEB Semântica. (BECKETT, 2004).

Segundo OLIVEIRA (2006, p.53),

> O desenvolvimento do RDF é motivado pelos seguintes casos de uso:

> - metadados para a Web – fornecer informações sobre recursos da Web e seus respectivos sistemas (descrição de funcionalidades, conteúdos, preferências, etc.);
>
> - aplicações que requerem modelos de informação abertos, em vez de limitados (fechados), como processos organizacionais, anotação de páginas Web, etc.;
>
> - tratamento de informações processadas automaticamente na Web do mesmo modo com que se faz o hipertexto, permitindo que os dados processados fora de um ambiente específico no qual foram criados possam ser utilizados em toda a Internet;
>
> - interconexão entre diversas aplicações da Web, combinando dados dessas aplicações e formando novas informações;
>
> - processamento automatizado de informações da Web por meio de agentes inteligentes de software, migrando-se de um ambiente legível por humanos para uma rede mundial de processos compatíveis, com base no RDF;

Embora tenha o *XML* como base, há diferença de propósito entre o *RDF* e o *XML*. Diferentemente do *XML*, o modelo abstrato utilizado pelo *RDF* permite que a informação seja dividida em pequenas partes, utilizando regras simples de semântica (sentido). O intuito do RDF, então, é especificar um método geral que seja simples e suficientemente flexível para expressar qualquer fato, permitindo que computadores possam processá-los. “O XML é a sintática, a semântica é RDF” (W3C, 2004).

A estrutura básica de qualquer expressão em RDF é uma coleção de triplas, cada uma consistindo de um sujeito, um predicado e um objeto. Um conjunto de triplas é chamado de grafo *RDF.* Cada tripla representa uma declaração de uma relação entre as coisas denotadas pelos nós que as vincula. Cada tripla possui três partes: um sujeito, um objeto e um predicado (também chamado de propriedade) que denota um relacionamento, conforme Figura 9. A direção do arco é significativa: sempre aponta para o objeto (KLINE & CARROLL, 2004).

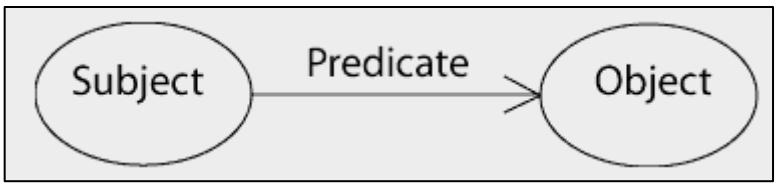

Figura 9 – Tripla RDF (In: KLINE & CARROLL, 2004).

Dessa forma, pode-se fazer afirmações sobre entidades. Por exemplo, se “Automóvel” for tratado como sujeito, “Veículo” como objeto e “é tipo de” como predicado, tem-se a afirmação que “Automóvel é tipo de Veículo”.

Essa possibilidade de relações entre as entidades é de fundamental importância para o desenvolvimento de Ontologias.

Porém, ainda faz-se necessário um mecanismo para definição de recursos, de propriedades e de relacionamentos. Esta é a função exercida pelo *RDF-Schema* que possibilita a criação de classes de tipos de recursos e de propriedades; descrições dessas classes; combinações possíveis de classes; propriedades e valores; e restrições entre relacionamentos, definindo assim esquemas que podem ser utilizados em conjunto com vocabulários descritivos.

5. OWL – WEB ONTOLOGY LANGUAGE

Este capítulo terá como premissa a recomendação da W3C para a construção de Ontologias em *OWL*.

5.1. DEFINIÇÕES

Segundo FERNEDA (2003, p. 110):

> Para a realização da Web Semântica são necessárias linguagens que permitam não apenas a definição de dados através de marcações, mas que possibilitem também descrever formalmente estruturas conceituais que possam ser utilizadas pelos agentes (robôs) de indexação de mecanismos de busca.

A *OWL – Web Ontology Language* – é uma das linguagens que possibilitam a implementação da Web Semântica, contemplando os requisitos acima citados.

Segundo McGuinness & Harmelen (2004), a *OWL* é projetada para que possa ser utilizada por aplicações que precisam processar o conteúdo de informações ao invés de apenas apresentar informações. A *OWL* facilita a interpretação do conteúdo da *Web* pela máquina, fornecendo vocabulário adicional em conjunto com uma semântica formal. A *OWL* possui mais facilidades para expressar significado e semântica do que *XML*, *RDF* e *RDF-S*. Sendo assim, a linguagem *OWL* vai além dessas linguagens em sua habilidade de representar conteúdo interpretável para a máquina na *Web*.

Segundo Bechhofer et al. (2004), *OWL* é uma linguagem de marcação semântica para publicação e compartilhamento de ontologias na *World Wide Web*. Tal linguagem é desenvolvida como uma extensão de vocabulário da linguagem *RDF* (*Resource Description Framework*) e é derivado do *DAML+OIL Web Ontology Language*.

Retomando o conteúdo abordado nos capítulos anteriores tem-se que a *Web Semântica* é uma prospecção para o futuro da *Web*. À informação será dada

significado explícito, tornando mais fácil e ágil para as máquinas processarem automaticamente o conteúdo das páginas *Web* e possibilitar a integração da informação disponível na *Web*.

McGuinness e Harmelen (2004) afirmam que a *Web Semântica* irá disponibilizar à *XML* habilidades para definir os regimes de marcação personalizada e possibilitará à *RDF* uma abordagem flexível para representação de dados. O primeiro nível acima da linguagem *RDF* necessário para a Web Semântica é uma linguagem de ontologia que possa descrever formalmente o significado da terminologia utilizada em documentos da *Web*. Espera-se que as máquinas realizem tarefas de raciocínio úteis sobre estes documentos, a linguagem deve ir além da semântica básica do Esquema *RDF*.

A *OWL* fornece três sublinguagens – *OWL Lite*, *OWL DL* e *OWL Full*. A escolha por uma delas dependerá do grau de expressividade que o usuário necessita para disponibilizar seus documentos na *Web*.

5.2. SUBLINGUAGENS *OWL*

As diferenciações entre as sublinguagens *OWL* descritas são definidas por McGuinness e Harmelen (2004).

OWL Lite suporta aqueles usuários que necessitam, principalmente, de uma hierarquia de classificação e restrições simples. Como por exemplo, o suporte a restrições de cardinalidade simples, pois tal sublinguagem só permite valores de cardinalidade zero ou um. A *OWL Lite* oferece um caminho de migração rápida para tesauros e outras taxonomias. *OWL Lite* também tem uma menor complexidade formal que *OWL DL*.

Em contrapartida *OWL DL* suporta aqueles usuários que querem o máximo de expressividade, mantendo completude computacional e poder de decisão. *OWL DL* inclui todas as construções da linguagem *OWL*, mas, somente podem ser utilizadas sob certas restrições. Por exemplo, enquanto uma classe pode ser uma subclasse de muitas classes, uma classe não pode ser uma instância de outra classe. *OWL DL*

é assim chamado devido à sua correspondência com as lógicas de descrição, um campo de pesquisa que estuda as lógicas que formam a base formal da *OWL*.

Já a *OWL Full* é destinada a usuários que querem o máximo de expressividade e a liberdade sintática do *RDF* sem garantias computacionais. Por exemplo, em *OWL Full* uma classe pode ser tratada, simultaneamente, como uma coleção de indivíduos e como um indivíduo por direito próprio. *OWL Full* permite uma ontologia aumentar o significado do vocabulário pré-definido (*RDF ou OWL*).

Desenvolvedores de ontologias *OWL* devem considerar adotar a sublinguagem que melhor se adapte às suas necessidades. A escolha entre *OWL Lite* e *OWL DL* depende do grau em que os usuários precisam das construções mais expressivas fornecidas pela *OWL DL*. A escolha entre *OWL DL* e *OWL Full* depende, principalmente, na medida em que os usuários exigem as instalações meta-modelagem de *RDF Schema* (por exemplo, a definição de classes de classes, ou anexando propriedades de classes).

OWL Full pode ser visto como uma extensão do *RDF*, enquanto *OWL Lite* e *OWL DL* podem ser vistos como extensões de uma visão restrita de *RDF*. Cada documento *OWL* (*Lite*, *DL* e *Full*) é um documento *RDF* e todo documento *RDF* é um documento *OWL Full*, mas apenas alguns documentos *RDF* serão documentos *OWL Lite* ou *OWL DL*. Devido a isso, alguns cuidados devem ser tomados quando um usuário deseja migrar um documento *RDF* para *OWL*. Quando a expressividade da *OWL DL* ou *OWL Lite* é considerada adequada, algumas precauções devem ser tomadas para garantir que o documento *RDF* original está em conformidade com as restrições adicionais impostas pelo *OWL DL* e *OWL Lite*. Entre outros, cada *URI (Uniform Resource Identifier*) que é usado como um nome de classe deve ser explicitamente declarado como sendo do tipo < *owl: Class* > (e similarmente para propriedades), cada indivíduo deve ser afirmado pertencer a, pelo menos, uma classe (mesmo que apenas < *owl: Thing* >). Os detalhes destas e de outras restrições sobre *OWL DL* e *OWL Lite* serão abordados nos tópicos subsequentes.

5.2.1. Sublinguagem OWL Lite

De acordo com o que foi visto anteriormente, *OWL Lite* usa apenas algumas das características da linguagem *OWL* e tem mais limitações no uso dos recursos do que *OWL DL* ou *Full OWL*.

McGuinness & Harmelen (2004) apontam que as classes *OWL Lite* só podem ser definidas em termos de superclasses nomeadas (superclasses não podem ser expressões arbitrárias). E apenas certos tipos de restrições de classe podem ser usados. Equivalência entre classes e os relacionamentos entre as classes subclasses também são permitidas apenas entre as classes nomeadas, e não entre expressões de classes arbitrárias. Da mesma forma, as restrições em *OWL Lite* usam classes apenas nomeadas. *OWL Lite* também tem uma noção limitada de cardinalidade, as cardinalidades explicitamente declaradas são zero ou um.

A seguir serão apresentadas as características da sublinguagem *OWL Lite* agrupadas de acordo com suas funcionalidades. Tais definições são recomendações W3C – dispostos nos artigos de McGuinness e Harmelen (2004) e Bechhofer et al. (2004).

5.2.1.1. Recursos *OWL Lite* – RDF Schema

Sintaxe <owl: Class>: Uma classe define um grupo de indivíduos que pertencem juntos por possuírem propriedades em comum. Classes podem ser organizadas em uma hierarquia de especialização usando *subClassOf* . Há um *built-in* de classe mais geral chamado *owl: Thing* que é a classe de todos os indivíduos e é uma superclasse de todas as classes OWL. Há também um outro *built-in* de classe mais específico chamado *owl: Nothing* que é a classe que não tem casos e uma subclasse de todas as classes OWL.

Smith et al. (2004) demonstram um exemplo da criação de classes em OWL Lite tomando como base o domínio Vinhos:

Criação das Classes *Winery* (Vinícula), *Region* (Região) e *ConsumableThing* (Coisa

Consumível):

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

Na sintaxe acima se verifica que apenas foram criadas as classes sob a denominação indicada por <*rdf:ID*= “ ”>. A partir de sua criação, as classes agora podem ser referenciadas através de *#Region.* Por exemplo: <*rdf: resource = "# Region"*>. Outras ontologias podem referenciar esta denominação usando sua forma completa, <*http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#Region*>.

Outra forma de referência usa a sintaxe <*rdf: about = "# Region"*> para estender a definição de um recurso. O uso desta sintaxe é um elemento crítico na criação de uma ontologia distribuída. Ela permite a extensão da definição importada de *Region* sem modificar o documento original apoiando a construção incremental de uma maior ontologia.

Com a classe *Winery* criada, pode-se referenciá-la utilizando seu identificador *#Winery*. Outros documentos podem precisar fazer referência a esta classe. A forma mais razoável para fazê-lo é proporcionar definições de *namespace* e *entity* que incluem a definição do documento

```
...
<!ENTITY vin  "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#" >
<!ENTITY food "http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#" >
...
<rdf:RDF xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
      xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#">
```

Dadas estas definições, a classe *Winery* pode ser referenciada utilizando também a tag *XML* <*vin: Winery*> ou o valor do atributo <*&vin; Winery*>. É possível, também, fazer referência a um recurso usando seu *URI* completo, neste caso *http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine # Winery*.

Sintaxe <*rdfs: subClassOf*>:

Hierarquias de classes podem ser criadas através de uma ou mais declarações especificando que uma classe é uma subclasse de outra.

Neste ponto, é possível criar uma definição simples para a classe de *Wine*. Vinho é um *PotableLiquid*.

```
<owl:Class rdf:ID="PotableLiquid">
  <rdfs:subClassOf rdf:resource="#ConsumableThing" />
  ...
</ Owl: Class>


<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="#PotableLiquid"/>
  <rdfs:label xml:lang="en"> wine </ rdfs: label>
  <rdfs:label xml:lang="fr"> vin </ rdfs: label>
  ...
</ Owl: Class>
```

Define-se então, que a classe *PortableLiquid* é uma subclasse de *ConsumableThing* e uma superclasse de *Wine.*

As entradas <*rdfs: label*> fornecem um nome legível aos seres humanos. São requisitos opcionais para as classes. O atributo <*lang*> fornece suporte para vários idiomas – “en” (Língua Inglesa) e “fr” (Língua Francesa). Um rótulo é como um comentário e em nada contribui para a interpretação lógica de uma ontologia.

Individual: Além das classes, é necessário descrever seus membros ou objetos. Os objetos são tratados como *Individuals* (indivíduos) em nosso universo de *things* (coisas). Um indivíduo é minimamente introduzido, declarado para ser um membro de uma classe. Indivíduos são instâncias de classes. As propriedades podem ser utilizadas para o relacionamento entre os indivíduos.

```
<owl:Thing rdf:ID="CentralCoastRegion" />
<owl:Thing rdf:about="#CentralCoastRegion">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

Vale ressaltar que a tag <*rdf: type*> é uma propriedade *RDF* que liga um indivíduo a uma classe da qual é membro.

É fundamental compreender que primeiramente se define *CentralCoastRegion* (uma área específica) como membro de *Region*, a classe que contém todas as regiões geográficas. Para se ter disponíveis algumas classes a mais para as propriedades introduzidas nas próximas seções, deve-se definir um ramo da taxonomia *Grape* (Uva), com um indivíduo que denota o *Cabernet Sauvignon* (tipo específico de uva). As uvas são definidas na ontologia alimentos:

```
<owl:Class rdf:ID="Grape">
  ...
</owl:Class>
```

E então na ontologia de vinhos, tem-se:

```
<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="&food;Grape" />
</owl:Class>

<WineGrape rdf:ID="CabernetSauvignonGrape" />
```

Dessa forma, *CabernetSauvignonGrape* é um indivíduo, porque denota uma única variedade de uva.

Sintaxe <*rdfs: domain*>: Um domínio (*domain*) de uma propriedade limita os indivíduos para que a propriedade possa ser aplicada. Se uma propriedade relaciona um indivíduo com outro, e a propriedade tem uma classe como um dos seus domínios, então o indivíduo deve pertencer à classe. <*rdfs: domain*> é chamado de restrição global

Sintaxe <*rdfs: range*>*:* O intervalo (*range*) de uma propriedade limita os indivíduos que a propriedade pode ter como valor. Se uma propriedade relaciona um indivíduo com outro, e a propriedade tem uma classe como um intervalo, o outro indivíduo

deve pertencer à classe de intervalo. *Range* é também uma restrição global tal como *domain*.

Sintaxe <*rdf: Property*>:

As propriedades podem ser usadas para relacionamentos entre indivíduos ou de indivíduos para os valores de dados. Exemplos de propriedades incluem *hasChild*, *hasRelative*, *hasSibling*, e *hasAge*. Os três primeiros podem ser usados para relacionar uma instância de uma classe para outra instância da mesma classe (ocorrências de *ObjectProperty*). O último pode ser usado para relacionar uma instância de uma classe para uma instância de um tipo de dados (ocorrências de *DatatypeProperty*). Ambos <*owl: ObjectProperty*). > e <*owl: DatatypeProperty*> são subclasses da classe RDF <*RDF: Property*>.

Em *OWL* dois tipos de propriedades são abordados: propriedades de tipo de dados – relacionam as instâncias de classes com *RDF* literal e com os tipos de dados *XML Schema* – e propriedades do objeto – as relações entre instâncias de duas classes.

Quando se define uma propriedade é possível restringir a relação de diversas maneiras. O domínio (*domain*) e o intervalo (*range*) podem ser especificados. A propriedade pode ser definida como uma especialização (subpropriedade) de uma propriedade existente.

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</owl:ObjectProperty>
```

Em *OWL*, uma sequência de elementos sem um operador explícito representa um conjunto implícito. A propriedade *madeFromGrape* tem um *domain* de vinho e um *range* de *WineGrape* . Ou seja, são relacionadas instâncias da classe Wine com instâncias da classe *WineGrape*. Vários *domains* significa que o domínio da

propriedade é a interseção das classes identificadas (e similarmente para intervalo).

Da mesma forma, a propriedade *course* (curso) amarra um *Meal* (Refeição) para um *MealCourse*.

O uso de *range* e de informação de *domain* no *OWL* é diferente de informações do tipo em uma linguagem de programação. Entre outras coisas, os tipos são usados para verificar a consistência de uma linguagem de programação. Em OWL, um intervalo (*range*) pode ser usado para inferir um tipo.

```
<owl:Thing rdf:ID="LindemansBin65Chardonnay">
  <madeFromGrape rdf:resource="#ChardonnayGrape" />
</owl:Thing>¬
```

Pode-se inferir que *LindemansBin65Chardonnay* é um vinho porque o domínio (*domain*) de *madeFromGrape* é vinho (*wine*).

```
<owl:Class rdf:ID="WineDescriptor" />

<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
  ...
</ Owl: Class>

<owl:ObjectProperty rdf:ID="hasWineDescriptor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineDescriptor" />
</ Owl: OBJECTPROPERTY>

<owl:ObjectProperty rdf:ID="hasColor">
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineColor" />
  ...
</ Owl: OBJECTPROPERTY>
```

No exemplo, a classe *WineDescriptor* é uma superclasse de *WineColor* relacionando os componentes do vinho com sua cor. À propriedade *hasWineDescriptor* é

delimitado o *domain* e o *range* interligando-o à classe *WineDescreptor*. *hasColor* é uma subpropriedade de *hasWineDescreptor*, com *range* ainda mais restrito a *WineColor* . A sintaxe *<rdfs: subPropertyOf>*, neste caso, significa que “qualquer coisa” com uma propriedade *hasColor* de valor X também tem uma propriedade *hasWineDescriptor* com o valor X. Nota-se que tanto as classes quanto as propriedades são organizadas hierarquicamente. As hierarquias de propriedades podem ser criadas fazendo uma ou mais declarações de que uma propriedade é uma subpropriedade de uma ou mais propriedades.

5.2.1.2. Características de Propriedades OWL Lite

Em *OWL Lite* existem identificadores que fornecem informações acerca as propriedades e seus valores.

A Sintaxe *<inverseOf>* : Fornece a propriedade inversa de outra propriedade. Se o P1 propriedade é indicado para ser a inversa da propriedade P2, em seguida, se X está relacionado com Y pela propriedade P2, então Y está relacionada com X pela propriedade P1.

```
<owl:ObjectProperty rdf:ID="hasMaker">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="producesWine">
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

Sintaxe *<TransitiveProperty>* : Se uma propriedade, P, é especificado como transitiva então, para qualquer x, y, e z: P (x, y) e P (y, z) implica P (x, z).

```
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="&owl;Thing" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>
```

```
<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>

<Region rdf:ID="CaliforniaRegion">
  <locatedIn rdf:resource="#USRegion" />
</Region>
```

Nota-se, então, uma hierárquica de *Region*. *SantaCruzMountainsRegion* Está localizada em *CaliforniaRegion,* que por sua vez está inserida em *USRegion.* Sendo assim, *SantaCruzMountainsRegion* está em *USRegion.*

Sintaxe <*SymmetricProperty*>: Se uma propriedade, P, é etiquetado como simétrica, então, para qualquer X e Y: P (x, y) se P (y, x)

```
<owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Region" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="MendocinoRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
  <adjacentRegion rdf:resource="#SonomaRegion" />
</Region>
```

Tem-se, então, que *MendocinoRegion* é adjacente a *SonomaRegion* e vice-versa, ambas regiões produtoras de vinho. E que *MendocinoRegion* está localizada (*locatedIn*) em *CaliforniaRegion*, porém *SonomaRegion* não, pois *locatedIn* não se trata de uma propriedade simétrica.

Sintaxe <*FunctionalProperty*>: Se uma propriedade, P, é marcada como funcional para todos os x, y e z: P (x, y) e P (x, z) implica y = z.

Tais propriedades podem ser indicadas para possuir um valor único. Se uma

propriedade é uma *FunctionalProperty*, então não tem mais do que um valor para cada indivíduo. *FunctionalProperty* é uma abreviação para afirmar que a cardinalidade mínima da propriedade é zero e sua cardinalidade máxima é 1.

```
<owl:Class rdf:ID="VintageYear" />

<owl:ObjectProperty rdf:ID="hasVintageYear">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Vintage" />
  <rdfs:range  rdf:resource="#VintageYear" />
</owl:ObjectProperty>
```

No exemplo, verifica-se que *hasVintageYear* é funcional. Um vinho possui um único ano. Isto é, um indivíduo *Vintage* só pode ser associado a um único ano utilizando a propriedade *hasVintageYear.*

Sintaxe <*InverseFunctionalProperty*>: Se uma propriedade P é marcada como *InverseFunctional* para todos os x, y e z: P (y, x) e P (z, x) implica y = z.

Se uma propriedade é funcional inversa, então o inverso da propriedade é funcional. Assim, o inverso da propriedade tem, no máximo, um valor para cada indivíduo. Trata-se de uma propriedade não ambígua.

```
<owl:ObjectProperty rdf:ID="hasMaker" />

<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

Observe que *producesWine,* visto anteriormente, é uma função inversa (*inverseOf*). A razão é que o inverso de uma propriedade funcional deve ser funcional inversa. Tal função poderia ter sido definida como *hasMaker* e *producesWine* (exemplo acima) para alcançar o efeito idêntico a propriedade *inverseOf*.

5.2.1.3. Restrições de Propriedades *OWL Lite*

OWL Lite permite restringir como as propriedades podem ser utilizadas por instâncias de uma classe. Este tipo de restrição é utilizado dentro do contexto de uma <*owl: Restriction*>. O elemento <*owl: onProperty*> indica a propriedade restrita.

Sintaxe <*owl: allValuesFrom*>: esta restrição requer que, para cada instância da classe que possui instâncias da propriedade especificada, os valores da propriedade são todos os membros da classe indicados pela cláusula <*owl: allValuesFrom*>.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:allValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Neste caso, com o uso de *allValuesFrom,* é definido que um vinho <*Wine*> não requer possuir um fabricante <*hasMaker*>, porém se o tiver, deverá necessariamente ser uma vinícola <*Winery*>.

Sintaxe <owl: *someValuesFrom*>: Esta restrição descreve o conjunto de indivíduos que tem pelo menos um tipo específico de relacionamento com indivíduos membros de uma classe.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:someValuesFrom rdf:resource="#Winery" />
    <Coruja /: Restrição>
```

```
 </ Rdfs: subClassOf>
 ...
</ Owl: Class>
```

Neste caso, com o uso de *someValuesFrom é* requerido que haja pelo menos um fabricante de que é uma vinícola, contudo, podem existir fabricantes que não são vinícolas.

5.2.1.4. Restrições de Cardinalidade OWL-Lite

Em *OWL* as restrições de cardinalidade podem se do tipo Cardinalidade Mínima <*owl: mimCardinality*>, Cardinalidade Máxima <*owl: maxCardinality*> e Cardinalidade exata <*owl: cardinality*>.

As restrições de cardinalidade mínima especificam o número mínimo de relacionamentos que um indivíduo pode possuir para uma determinada propriedade.

As restrições de cardinalidade máxima, por sua vez, especificam o número máximo de relacionamentos que um indivíduo pode possuir para uma determinada propriedade.

Por fim, as restrições de cardinalidade exata, especificam o número exato de relacionamentos que um indivíduo deve estabelecer com determinada propriedade.

```
<owl:Class rdf:ID="Vintage">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasVintageYear"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Neste caso, especifica-se que cada época de colheita de vinhos <*Vintage*> está restrito a possuir apenas um ano de colheita <*hasVintageYear*> através da propriedade de cardinalidade exata <*owl: cadinality*>.

5.2.1.5. Recurso de Igualdade e Desigualdade *OWL Lite*

Em *OWL Lite* existem identificadores que são usados para fornecer informações sobre as relações de igualdade e desigualdade.

Sintaxe <*equivalentClass*>: Classes equivalentes possuem as mesmas instâncias, em outras palavras, podem ser utilizadas para criar classes sinônimos. Por exemplo, um carro pode ser indicado para ser <*equivalentClass*> de Automóvel. Neste caso, infere-se que qualquer indivíduo que é uma instância de carro também é uma instância de automóveis e vice-versa.

Sintaxe <*equivalentProperty*>: Propriedades equivalentes referem-se ao mesmo princípio de classes equivalentes, porém são aplicadas em propriedades *OWL.*

Sintaxe <*sameAs*>: Aplicada aos Indivíduos, indica que dois indivíduos são os mesmos. Por exemplo, o indivíduo Carlos Drummond de Andrade pode ser indicado para ser o mesmo indivíduo Drummond.

Sintaxe <*differentFrom*>: Refere-se ao inverso da sintaxe <*sameAs*>. É utilizada quando se tem a necessidade explícita que dois indivíduos são diferentes.

Sintaxe <*allDifferent*>: Utilizada para indicar que os indivíduos relacionados são mutualmente distintos.

5.2.2. Recursos *OWL DL* e *OWL Full*

As Sublinguagens *OWL DL* e *OWL Full* são constituídas através de um mesmo vocabulário para a construção de ontologias, porém *OWL DL* está sujeita a algumas restrições.

Em *OWL DL* há a restrição de que uma classe não pode ser um indivíduo ou uma propriedade simultaneamente, da mesma forma que uma propriedade não pode ser uma classe ou um indivíduo. Caso isto ocorra, trata-se de *OWL Full.*

Outra diferenciação está no fato de que *OWL DL* aceita exclusivamente que uma

propriedade seja exclusivamente *ObjectsProperties* – relações existentes entre instâncias de classes – ou *DatatypeProperties* – relações entre as instâncias de classes com elementos *RDF* e com os tipos de dados *XML Schema.*

Na sequência são apresentados os recursos adicionais fornecidos pelas sublinguagens *OWL DL* e *OWL Full*.

<*oneof*>: Esta sintaxe refere-se a classes enumeradas. Neste caso, a classe é restrita a possuir determinados indivíduos. Os membros da classe são exatamente o conjunto de indivíduos enumerados, nem mais, nem menos.

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <WineColor rdf:about="#White" />
    <WineColor rdf:about="#Rose" />
    <WineColor rdf:about="#Red" />
  </owl:oneOf>
</owl:Class>
```

Neste caso, restringe-se que <*WineColor*>, coloração do vinho só poderá relacionar-se com indivíduos que sejam *White, Rose* ou *Red* (Branco, Rosé, ou Tinto). Tem-se também que sua cardinalidade máxima será de três relacionamentos.

<*hasValue*>: Esta sintaxe refere-se a valores de propriedades, neste caso, uma propriedade é obrigada a possuir determinado indivíduo como valor.

```
<owl:Class rdf:ID="Burgundy">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSugar" />
      <owl:hasValue rdf:resource="#Dry" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Neste caso, restringe-se que a classe <*Burgundy*>, tipo de vinho, só poderá relacionar-se através da propriedade <*hasSugar*> com indivíduos que possuam o valor <*Dry*> (seco).

<*disjoinWhith*>: Esta restrição aplicada às classes as tornam disjuntas, ou seja, garante que um indivíduo membro de uma classe não pode, simultaneamente, ser uma instância de outra classe.

```
<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing"/>
  <owl:disjointWith rdf:resource="#Meat"/>
  <owl:disjointWith rdf:resource="#Fowl"/>
  <owl:disjointWith rdf:resource="#Seafood"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

Neste caso, tem-se que um indivíduo pertencente à classe <*Pasta*> não poderá ser instância das demais classes precedidas pela restrição <*disjoinWhith*>. Contudo, não é afirmado que as classes *Meat* (Carne) e *Seafood* (Frutos do mar) são disjuntas entre si. Para isso deve-se criar a disjunção mutuamente para cada par declarado como disjunto.

As sublinguagens *OWL DL* e *OWL Full* permitem também o uso de combinações booleanas de classes e restrições.

<*unionOf*>: A classe declarada sob a restrição de união é criada pela combinação de duas ou mais classes fazendo uso do operador *OR*.

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SweetFruit" />
    <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

Neste caso, é declarada que a classe *Fruit* (Fruta) contém os indivíduos que são declarados como *SweetFruit* (Fruta doce) ou *NonSweetFruit* (Frutas não doces).

<*intersectionOf*>: A classe declarada sob a restrição de intersecção é criada pela combinação de duas ou mais classes fazendo uso do operador *AND*.

```
<owl:Class rdf:ID="WhiteWine">
```

```
<owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#Wine" />
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasColor" />
    <owl:hasValue rdf:resource="#White" />
  </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

Esta construção fixa que W*hitewine* (Vinho Branco) é exatamente o cruzamento da classe *Wine* (Vinho) e do conjunto de coisas que são de cor branca, utilizando a propriedade *hasColor* (possui cor) e fixando, através da restrição *hasValue*, a obrigatoriedade de ser *White* (Branco). Isto significa que, se algo é branco e um vinho, então é uma instância de *Whitewine* .

<*complementOf*>: O uso desta sintaxe seleciona todos os indivíduos do domínio do discurso, que não pertencem a uma determinada classe.

```
<owl:Class rdf:ID="ConsumableThing" />
  <owl:Class rdf:ID="NonConsumableThing">
    <owl:complementOf rdf:resource="#ConsumableThing" />
  </owl:Class>
```

Neste caso, são declaradas duas classes distintas *ConsumableThing* (Coisa consumível) e *NonConsumableThing* (coisa não consumível). A classe *NonConsumableThing* inclui como membros todos os indivíduos que não pertencem à extensão do *ConsumableThing.*

OWL DL e *OWL Full* também ampliam as funcionalidades das cardinalidades existentes em *OWL Lite.* Enquanto que em *OWL Lite* os valores de cardinalidade são restritos a 0 (zero) ou 1 (um), as outras sublinguagem ampliam o uso destas restrições para que possuam qualquer valor, desde que seja inteiro e não negativos.

6. PROPOSTA DE TRABALHO

Neste capítulo será analisado o editor de ontologias *Protégé na versão 3.4.8* associado ao *Plugin Protégé OWL*. A Plataforma Protégé foi desenvolvida por *Stanford Center for Biomedical Informatics Research* da *Stanford University School of Medicine.*

Serão verificados quais procedimentos devem ser adotados para a implantação dos recursos abordados no capítulo anterior, tais como a criação de classes e Subclasses; a criação de Classes disjuntas; a especificação de Domínio e escopo das classes; a criação de propriedades Funcionais e propriedades Funcionais Inversas; a criação de restrições *allValuesFrom*, *someValuesFrom* e *hasValue*; a criação de restrições de Cardinalidade; e a criação de Indivíduos.

Serão abordadas também questões de testes e importações de ontologias, bem como propriedades de anotação.

Para verificar a implementação dos procedimentos delimitou-se como Domínio da ontologia a Classificação de Veículos conforme Tipo/Marca/Espécie seguindo a Portaria Número 309 de 15 de junho de 2012 (vide Anexo A), do DENATRAN (Departamento Nacional de Trânsito). Optou-se pela restrição de alguns Tipos de veículos conforme Figura 10. No Anexo B tem-se o código *XML* gerado após a criação da ontologia proposta

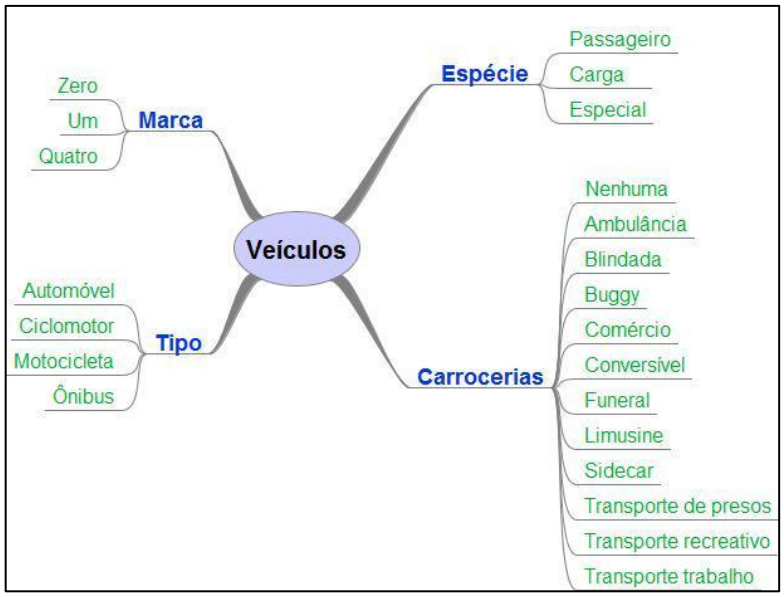

Figura 10 – *Representação do Domínio da Ontologia*

6.1.APRESENTAÇÃO DA PLATAFORMA PROTÉGÉ

Protégé é software livre e open-source. A plataforma fornece um conjunto de ferramentas para a construção de modelos de domínio e conhecimento com aplicações baseadas em ontologias. Na sua essência, *Protégé* implementa um rico conjunto de conhecimentos de modelagem de estruturas e ações de apoio à criação, visualização e manipulação de ontologias em diferentes formatos de representação.

A plataforma pode ser utilizada para carregar, editar e salvar ontologias em diversos formatos. Entre eles destacam-se *RDF*, *XML*, *UML*, *OWL* e bancos de dados relacionais. Na versão analisada o suporte à *OWL* é possível por meio do uso adicional do *OWL Plugin.*

Ao basear o *Plugin OWL* em cima da *Protégé*, também é possível reutilizar a sua base cliente-servidor em modo multiusuário possibilitando que várias pessoas editem a mesma ontologia, ao mesmo tempo. Protégé também fornece um banco de dados *back-end* altamente escalável, permitindo aos usuários criar ontologias com centenas de milhares de classes. (KNUBLAUCH et al., 2004, p. 4)

6.2. ARQUITETURA

A arquitetura da plataforma *Protégé* é separada em *model* (modelo) e em *view* (visão). Segundo KNUBLAUCH et al.(p. 3):

> Protégé's model is based on a simple yet flexible metamodel, which is comparable to object-oriented and frame-based systems. It basically can represent ontologies consisting of classes, properties (slots), property characteristics (facets and constraints), and instances. Protégé provides an open Java API to query and manipulate models. An important strength of Protégé is that the Protégé metamodel itself is a Protégé ontology, with classes that represent classes, properties, and so on.
>
> (…)
>
> Using the views of Protégé's user interface, ontology designers basically create classes, assign properties to the classes, and then restrict the properties' facets at certain classes. Using the resulting ontologies, Protégé is able to automatically generate user interfaces that support the creation of individuals (instances). For each class in the ontology, the system creates one form with editing components (widgets) for each property of the class.

Dessa forma verifica-se que o modelo (*model*) é o mecanismo de representação interna para ontologias e bases de conhecimento. Em contrapartida, a visão (*view*) fornece uma interface de usuário para exibir e manipular o modelo representado.

6.3. CRIAÇÃO DE UM PROJETO

A Figura 11 refere-se à tela inicial da Plataforma *Protégé 3.4.8.* Nela pode-se criar um novo projeto, abrir um projeto existente ou obter ajuda sobre a plataforma. Para criar uma nova ontologia clica-se em *New Project*.

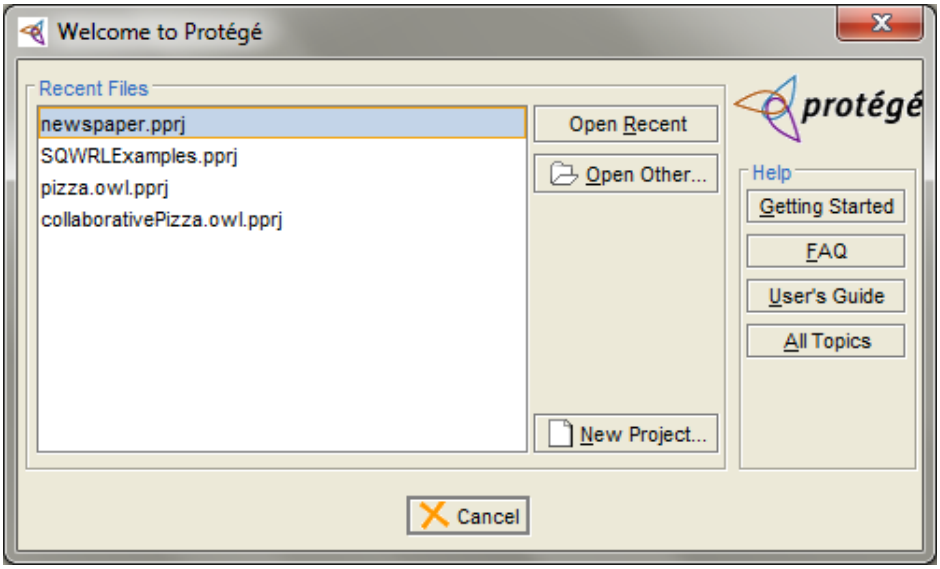

Figura 11 – Tela de Boas-vindas da plataforma Protégé 3.4.8

Em seguida opta-se pelo tipo do projeto a ser criado. Será utilizado o tipo *OWL / RDF Files*, conforme Figura 12.

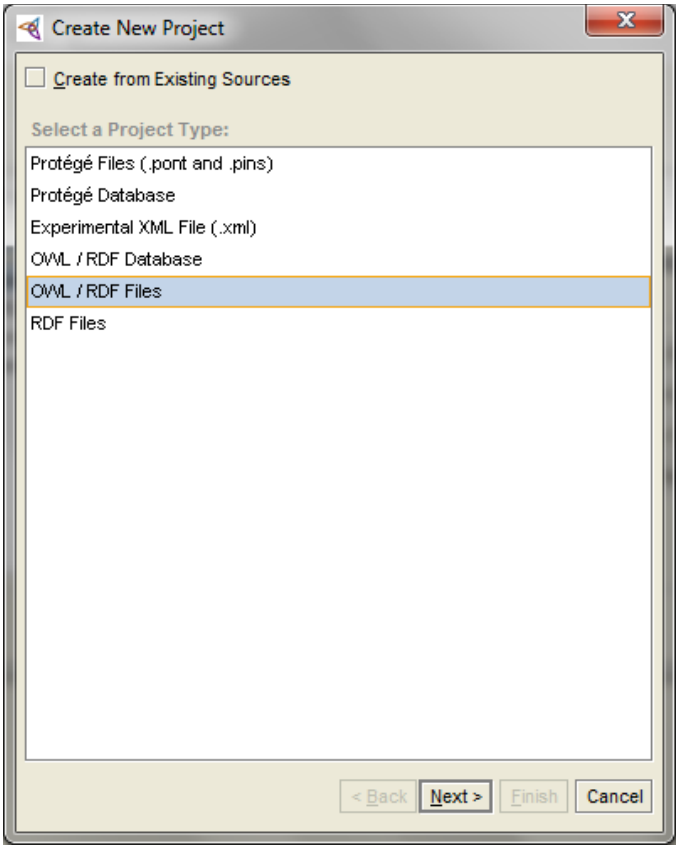

Figura 12 – Escolha do Tipo do Projeto

Na sequência, conforme ilustra a Figura 13, é especificado a *URI* para a ontologia. A partir desta *URI* outras ontologias podem importar o projeto criado. É recomendado que este endereço corresponda à localização da ontologia na *Web.* Neste caso a ontologia foi denominada de *Veículos.*

Em seguida delimita-se a linguagem utilizada na criação da ontologia. Ressalta-se que a troca de linguagem / sublinguagem utilizada pode ser efetuada a qualquer momento, através das *Preferences* (Preferências), conforme Figura 14.

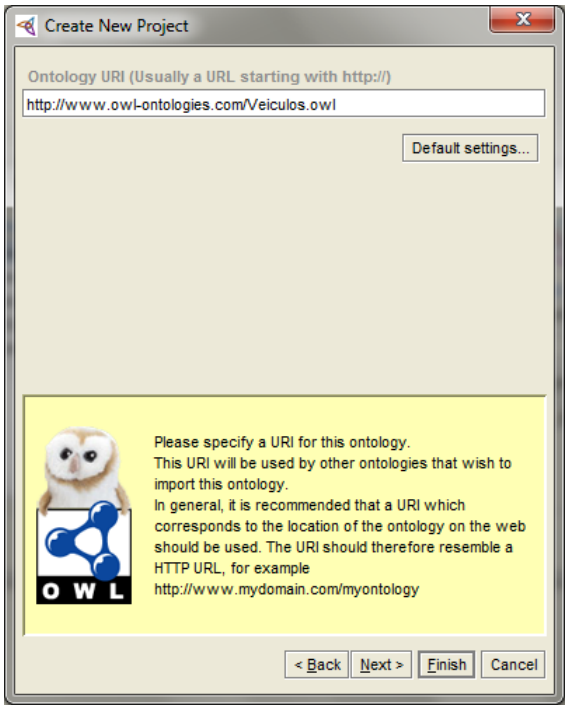

Figura 13 – Delimitação da URI da ontologia

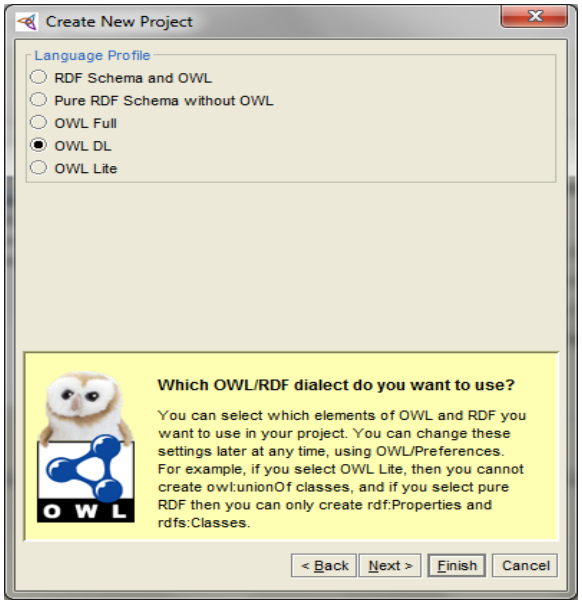

Figura 14 – Delimitação da linguagem / sublinguagem

A última etapa de criação de um novo projeto refere-se à etapa no qual se delimita a interface (Figura 15) que será apresentada para o desenvolvedor. Tal recurso pode ser alterado a qualquer momento na aba *OWL Classes.*

Em *Logic View*, a aba *OWLClasses* é exibida de forma que se visualize a relação existente entre as classes. Já, em *Properties View* é exibido a hierarquia de propriedades a qual a classe é submetida.

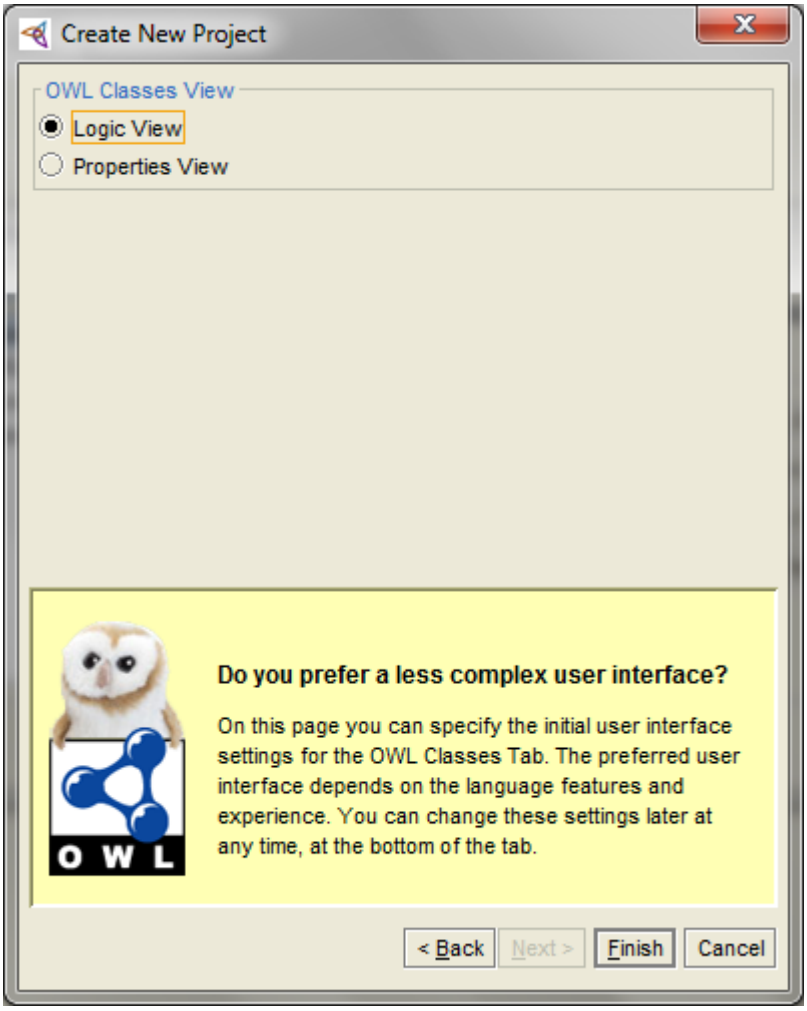

Figura 15 – Delimitação da Interface

Feito isso a janela da plataforma *Protégé* é aberta, possibilitando o acesso às abas principais para a construção de uma ontologia (Figura 16). São elas: *Metadata*, *OWL Classes*, *Properties*, *Individuals* e *forms.* Posteriormente cada aba será detalhada individualmente.

Na aba de Metadados tem-se a *URI* da ontologia criada, bem como a possibilidade de adição de anotações *RDFS.*

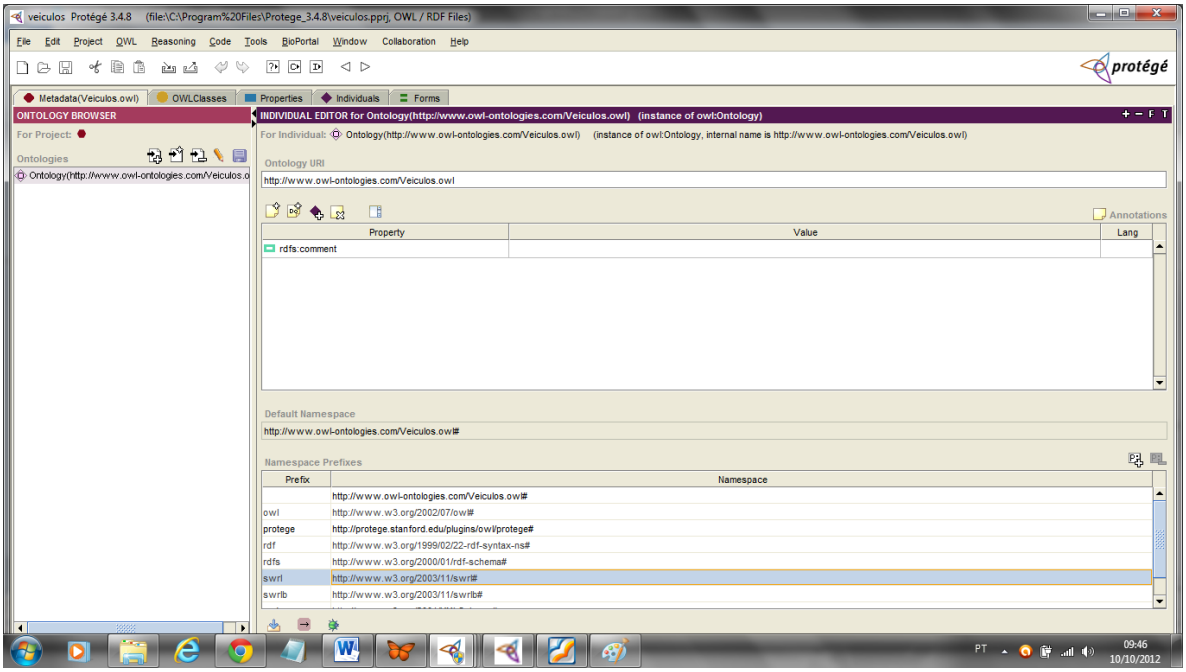

Figura 16 – Janela principal da plataforma Protégé

6.4. CRIAÇÃO DE CLASSES

A próxima aba *OWLClasses* é feita a declaração das classes que compõem a Ontologia, conforme Figura 17.

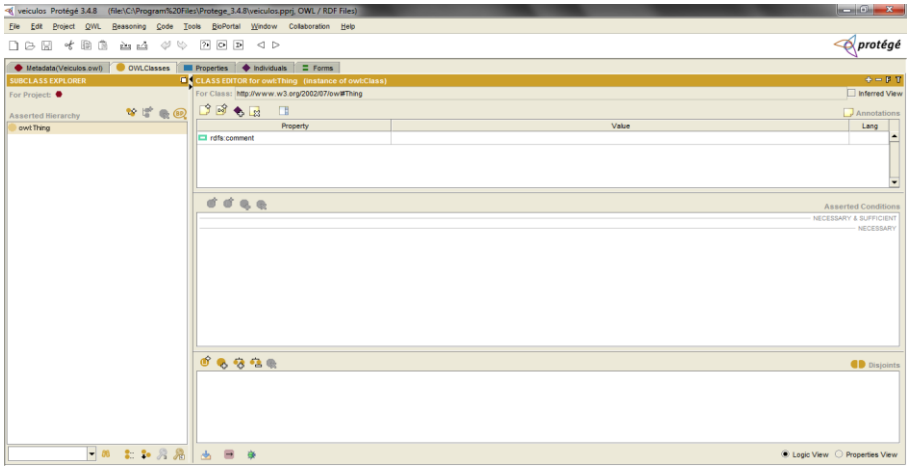

Figura 17 – Aba OWLClasses

Note que existe a presença da classe *owl:Thing.* Todas as Ontologias são derivadas (subclasses) desta superclasse. A classe *owl:Thing* é a classe que representa o conjunto que contém todos os indivíduos, uma vez que todas as classes são subclasses de *owl:Thing*. Isso se dá para possibilitar a comunicação de diferentes ontologias.

Neste ponto serão criadas duas classes denominadas *Veiculos* e *Classificacao*. Protégé permite a criação de classes de duas maneiras distintas. A primeira delas clica-se com o botão direito do mouse na classe escolhida e seleciona-se a opção *Create subclass,* conforme Figura 18. A segunda forma utiliza-se os botões do painel de Hierarquia de Classes (*Asserted Hierarchy*), conforme Tabela 1. Protégé também possibilita a criação de múltiplas classes utilizando um *Wizard.* Neste caso opta-se pela seleção de *Create subclasses.* Opção visualizada na Figura 18.

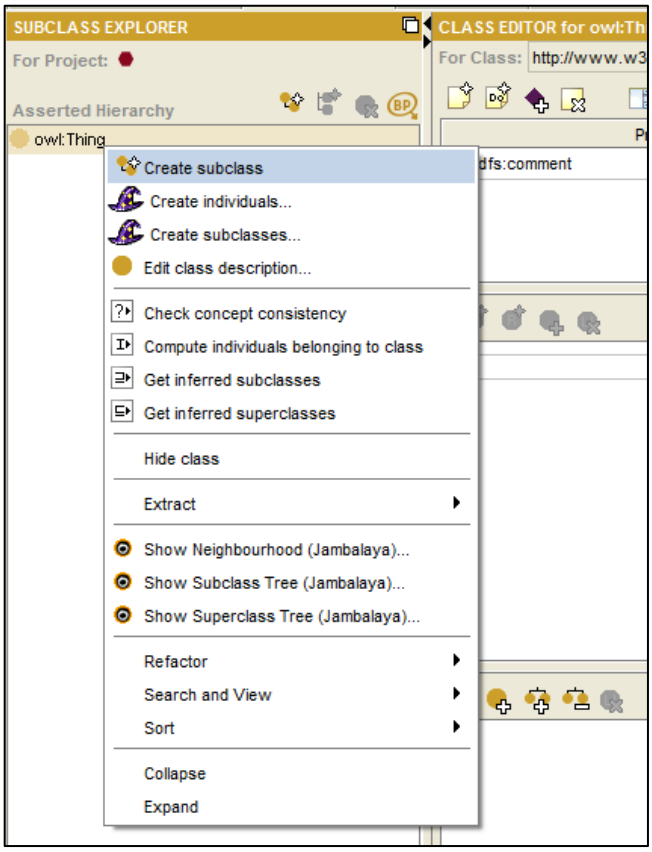

Figura 18 – Visão parcial para criação de classes

Ícone	Descrição
	Create subclasse – Criar subclasse
	Create sibling class – Criar classe irmã
	Delete selected class(es) – Apagar classe

Tabela 1: Legenda do painel de Hierarquia de Classes

Para a criação das classes optou-se pela utilização do painel de Hierarquia de Classes. Ao clicar no botão *Create subclass* é incluída uma subclasse de *owl:Thing*. Para renomeá-la altera-se o campo *For Class* do plainel de edição de classes (*Class Editor*). Conforme verifica-se na Figura 19.

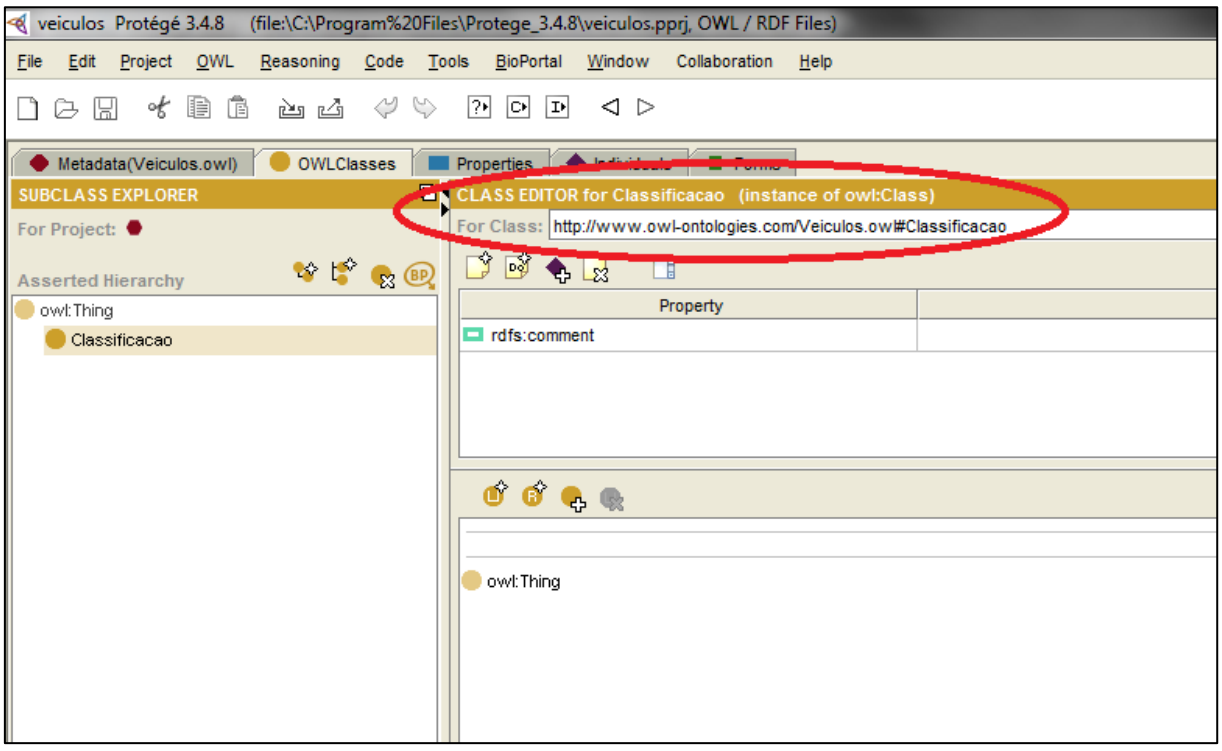

Figura 19 – Criação e renomeação de Classes

Este processo também será feito para as demais classes do projeto. Ao final do

procedimento a hierarquia de classes estará organizada como demonstra a Figura 20.

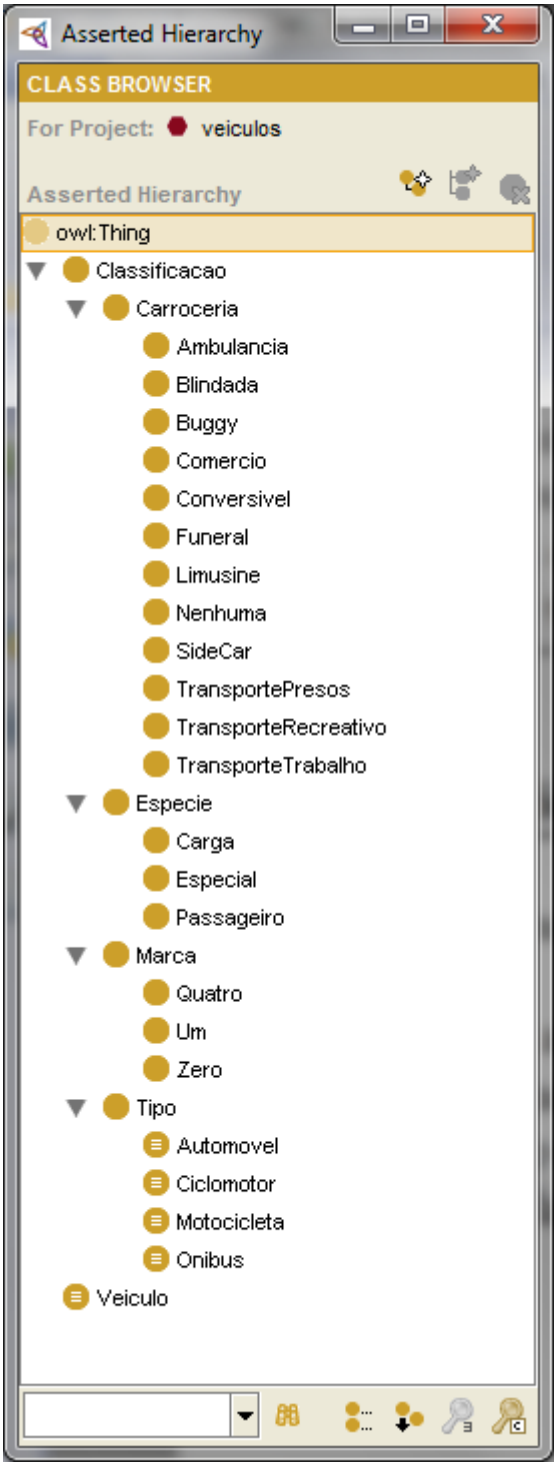

Figura 20 – Hierarquia de classes do Domínio

Feito isso, é necessário que se especifique que as subclasses de *Classificação* (*Carroceria, Tipo, Marca* e *Espécie*) serão disjuntas. Da mesma forma que também serão disjuntas as classes que estarão no mesmo nível de hierarquia, com exceção das classes *Classificação e Veículos.* Para especificar as classes que serão disjuntas da classe selecionada, use a interface gráfica *Disjoints* (Disjunção), localizada no canto inferior direito da aba *OWLClasses*. Conforme ilustra Figura 21.

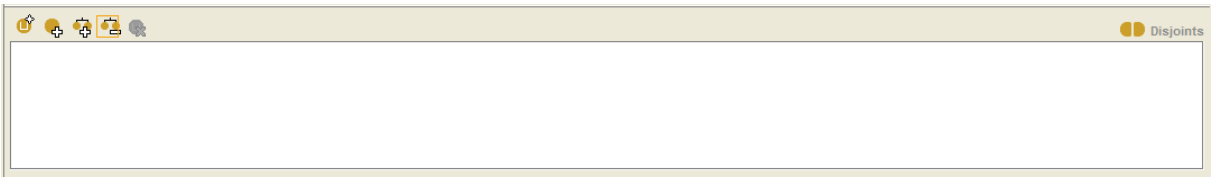

Figura 21 – Painel de criação de classes disjuntas

Para criar classes disjuntas seleciona-se a classe no painel de Hierarquia de Classes e opta-se por um dos botões do painel de Classes disjuntas, conforme Tabela 2. Deve-se estar atento na manipulação da disjunção. Caso se utilize o botão *Add Disjoint class* um painel de navegação será aberto no qual se delimita apenas uma classe que será disjunta, conforme Figura 22. Outra forma de torná-las disjuntas é utilizando o botão *Add all siblings.* Neste caso, todas as classes que estão no mesmo nível da hierarquia serão disjuntas. As selecionar este botão, será aberta uma caixa de diálogo (Figura 23). Se selecionada a opção *Mutually between all siblings* todas as classes serão disjuntas entre si. Em contrapartida, se selecionada a opção *Only between this class and its siblings*, a classe em questão será disjunta a outra, mas não será disjunta entre as demais classes.

Ícone	Descrição
	Create disjoint class from OWL expression - criar classe disjunta para expressão OWL
	Add disjoint class - Adicionar classe disjunta
	Add all siblings - adicionar todas as classes irmãs
	Remove all siblings - Remover todas as classes irmãs
	Delete selected row - Apagar linha selecionada

Tabela 2: Botões do painel de Classes Disjuntas

Figura 22 – Seleção de classe disjunta

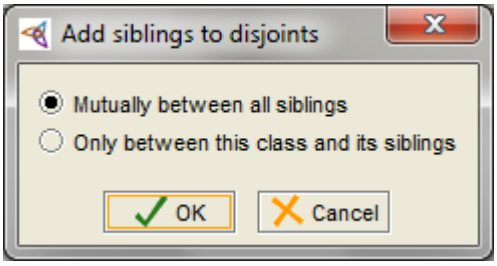

Figura 23 – Caixa de diálogo para seleção de classes disjuntas

Neste caso, todas as classes do projeto, que estão no mesmo nível hierárquico, foram definidas como disjuntas, com exceção das classes *Classificação* e *Veículos.* Veja como exemplo as disjunções contidas na classe *Automóvel*, apresentada na Figura 24.

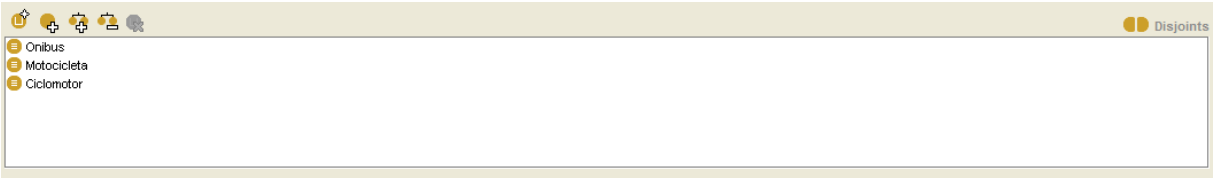

Figura 24 – Classes disjuntas à classe Automóvel

Feito isso, tem-se no projeto todas as classes criadas e as disjunções estabelecidas.

6.5. CRIAÇÃO DE PROPRIEDADES

Após o processo de criação de classe, serão criadas as propriedades que possibilitarão o relacionamento entre as classes. Para criar as propriedades seleciona-se a aba *Properties* (Propriedades), conforme ilustra Figura 25.

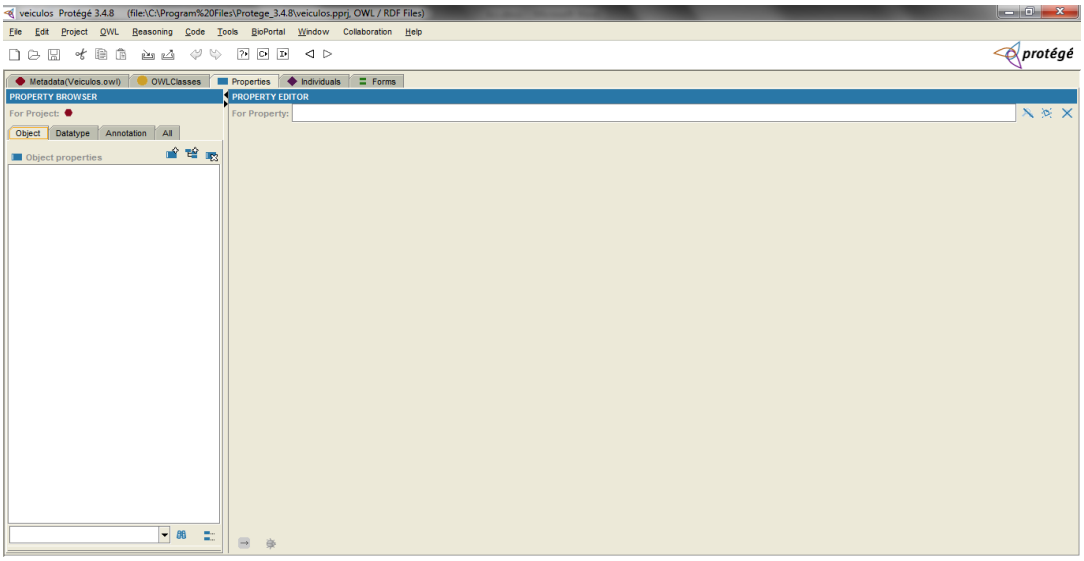

Figura 25 – Aba de Propriedades

Em *Properties Browser* tem-se os tipos de propriedades que podem ser criadas. Tais propriedades são referentes a *Object, Datatype, Annotation* e *All*. A Tabela 3 exibe os botões encontrados na aba *All*, que une todos os botões das demais abas.

Ícone	Descrição
	Create datatype property - Criar propriedade de tipo de dados
	Create oblect property - Criar prapriedade do objeto
	Create Subproperty - Criar subpropriedade
	Create annotation datatype property - Criar propriedade de anotação de tipo de dados
	Create annotation object property - Criar propriedade de anotação de objeto
	Create annotation property - Criar propriedade de anotação
	Delete properties - Apagar propriedades

Tabela 3: Botões para a criação de propriedades

No projeto serão utilizadas as opções referentes à *Objects* (Objetos). Neste ponto serão criadas as propriedades *hasClassificacao*, que possuirá as subpropriedade *hasMarca, hasTipo, hasEspecie* e *hasCarroceria.* E a propriedades *hasVeiculo.* Todas estas propriedades serão do tipo *Functional* (Funcional). Também serão criadas as propriedades funcionais inversas às propriedades funcionais criadas. São elas: *isClassificacaoOf, isMarcaOf, isTipoOf, isEspecieOf, isCarroceriaOf* e *isVeiculoOf.*

Não existe uma regra para a nomenclatura das propriedades, porém, é recomendado pela W3C que as propriedades funcionais possuam o prefixo *has* (tem, possui) e as propriedades funcionais inversas possuam o prefixo *is* e o sufixo *Of* (é ... de).

Para a criação da propriedade *hasClassificacao* seleciona-se o botão *Create object property*. No campo *For property* delimita-se o nome da propriedade. Na interface *domain* é delimitado o domínio da propriedade. Na interface *Range* é delimitado o escopo da propriedade. Também é delimitado o tipo de propriedade no qual pode ser *Functional* (Funcional), *InverseFunctional* (Funcional Inversa), *Symmetric* (Simétrica) e *Transitive* (Transitiva). Caso a propriedade seja Funcional Inversa, deve-se informal a qual propriedade ela é inversa, devendo ser especificada no campo *Inverse*.

Ao final do processo de criação das propriedades a hierarquia de propriedades ficará de acordo com o representado na Figura 26.

Ao final temos que a propriedade *hasClassificacao* possui o domínio *owl:Thing*, possui o *range* Classificacao, é do tipo Funcional e é inversa à *isClassificacaoOf.* A propriedade *hasVeiculo* possui o domínio *owl:Thing*, possui o *range* Veiculo, é do tipo Funcional e é inversa à *isVeiculoOf*. A propriedade *hasTipo* possui o domínio Classificacao, possui o *range* Tipo, é do tipo Funcional e é inversa à *isTipoOf*. A propriedade *hasEspecie* possui o domínio Tipo, possui o *range* Especie, é do tipo Funcional e é inversa à *isEspecieOf*. A propriedade *hasMarca* possui o domínio Classificacao, possui o *range* Marca, é do tipo Funcional e é inversa à *isMarcaOf*. A propriedade *hasCarroceria* possui o domínio Classificacao, possui o *range* Carroceria, é do tipo Funcional e é inversa à *isCarroceriaOf*.

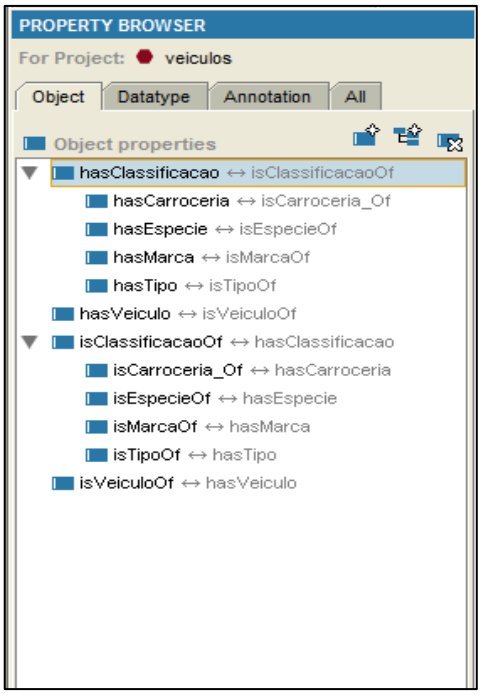

Figura 26 – Hierarquia das propriedades

6.6. CRIAÇÕES DE RESTRIÇÕES DAS CLASSES

Depois de criadas as propriedades, deve-se relacioná-las às classes criadas. Para isso, as restrições de uma classe são exibidas e editadas na interface *Asserted Conditions* (CondiçõesDeclaradas), conforme Figura 37. A interface Asserted Conditions é a parte mais importante da aba *OWLClasses* do Protégé-OWL, uma vez que detém praticamente todas as informações para descrição de uma classe.

Figura 27 – Interface Asserted Conditions

Esta Interface é dividida em duas partes: *Necessary & Sufficient* e *Necessary.* Condições Necessárias indicam que se alguma coisa é um membro da classe então é necessário que preencha as condições ali declaradas. Por sua vez, Condições Necessárias e Suficientes permitem afirmar que se alguma coisa preenche essas condições então, provavelmente, deve ser um membro dessa classe. Caso a superclasse da classe selecionada possua restrições, elas são herdadas pela subclasse, dessa forma será exibida uma nova parte na Interface *Asserted Conditions* com a denominação de *Inherited* contendo as restrições herdadas.

Para criar uma restrição utilizam-se os ícones da interface, conforme Tabela 4.

Ícone	Descrição
	Create new expression - Cria restrições através de expressões
	Create restriction - Cria restrições através do modo Wizard
	Add named class - Adiciona restrição de classe
	Delete selected row - Apaga uma restrição existente

Tabela 4: Ícones da Interface *Asserted Conditions*

Ao selecionar *Create new expression* surge o painel *Expression Builder* com as opções de restrições a serem criadas, conforme Figura 32. A descrição dos ícones é apresentada na Tabela 5.

Figura 28 – Painel Expression builder

Ícone	Descrição
	Insert allValuesFrom (only) - Inserir restrição allValuesFrom
	Insert someValuesFrom (some) - Inserir restrição someValuesFrom
	Insert hasValue (has) - Inserir restrição hasValue
	Insert cardinality (exactaly) - Inserir cardinalidade exata
	Insert minCardinality (min) - Inserir cardinalidade mínima
	Insert maxCardinality (max) - Inserir cardinalidade máxima
	Insert intersectionOf (and) - Inserir condicional e
	Insert unionOf (or) - Inserir condicional ou
	Insert complementOf (not) - Inserir condicional de Negação
	Insert class - Inserir classe
	Insert property - Inserir propriedade
	Insert Individual - Inserir Indivíduo
	Insert datatype - Inserir Tipo de dados
	Insert true - Inseri condição verdadeira
	Insert false - Inserir condição false
	Insert curly brackets for enumerations: {} - Inserir chaves para enumeração
	Insert round brackets: () - Inserir parênteses
	Backspace - Retrocesso
	Show error message - Mostrar mensagem de erro
	Cancel editing - Cancelar edição

Tabela 5: Descrição dos ícones do painel Expression Builder

As restrições também podem ser criadas através do modo Wizard. Para isso deve-se clicar no ícone *Create restriction* na Interface *Asserted Conditions.* Feito isso é aberta a caixa de diálogo de criação de restrição (Figura 29).

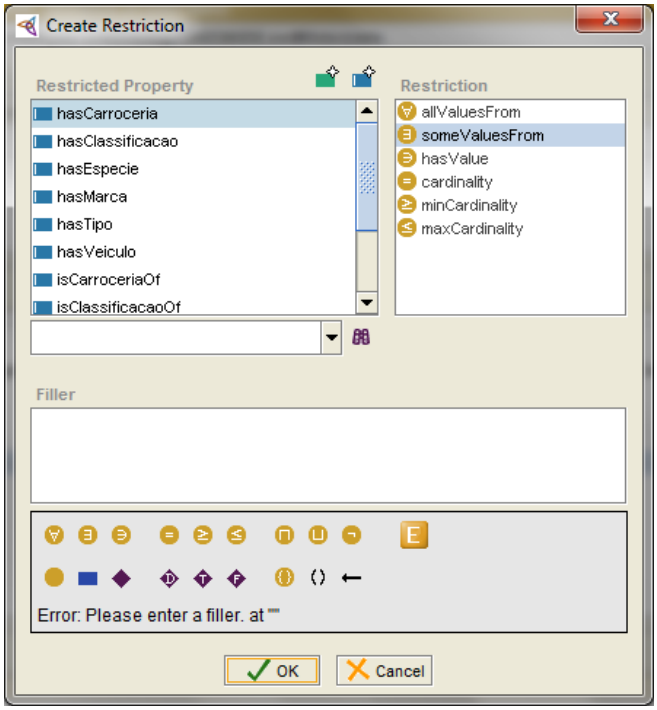

Figura 29 – Caixa de diálogo de criação de restrições

A Caixa de diálogo *Create Restriction* possui quatro partes principais. Em *Restricted Property* é exibida a lista de propriedades previamente declaradas na aba *Properties.* Já em *Restrictions* é apresentada a lista de restrições. Em *Filler* são relacionadas a(s) classe(s) que receberão a restrição. Por fim, tem-se também o painel de construção de expressões.

Para a criação da restrição são necessários três procedimentos: Seleção da propriedade no qual se deseja aplicar a restrição na lista de *Restricted Property*; Seleção do tipo de restrição contida na lista de *Restrictions*; e Especificação de um *Filler* para a restrição que poderá ser formulada com o auxílio do painel de construção de expressões.

Será demostrada a criação de restrições da classe *Automovel*. Primeiramente será feita a restrição à classe *Especie*. Será criada, então, uma restrição que afirmará que um *Automóvel* só poderá possuir a espécie *Passageiro* ou *Especial.* Caso possua a espécie *Passageiro,* só poderá conter *Carroceria* que seja *Nenhuma*, *Buggy*, *Conversível* ou *Limusine*. Casso possua a espécie *Especial* só poderá conter *Carroceria* que seja *Nenhuma*, *Ambulancia*, *Funeral* ou *Comercio*. Para consolidar

tal restrição foi escolhida em *Restricted Property* a propriedade *hasEspecie*, em *Restriction* delimitou-se o uso do relacionamento *allValuesFrom* e em *Filler* foi acrescentada a expressão *((Passageiro and (hasCarroceria only (Nenhuma or Buggy or Conversivel or Limusine))) or (Especial and (hasCarroceria only (Nenhuma or Ambulancia or Funeral or Comercio))))*, conforme ilustra Figura 30. Se a expressão estiver formada corretamente, o ícone *Show error message* é substituído pelo ícone *Assign (Ok)*, isto porque *Protégé* contém um compilador que age em tempo real durante a construção de restrições.

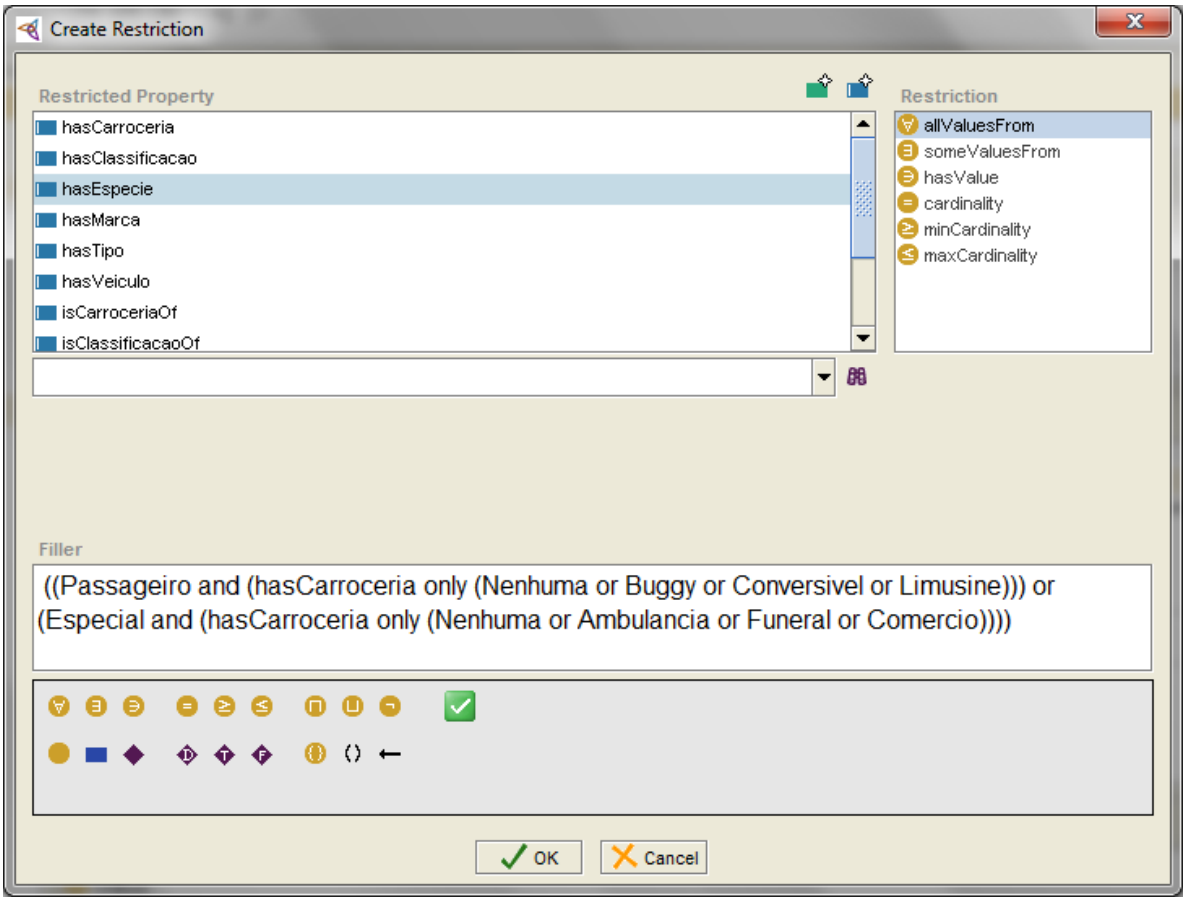

Figura 30 – Criação de Restrições

Um bom recurso disponibilizado pela plataforma é o uso de *autocomplete*. Na caixa *Filler* ao pressionar <CTRL>+<TAB> é disponibilizada uma lista das classes que podem contemplar a *string* informada, conforme apresentado na Figura 31.

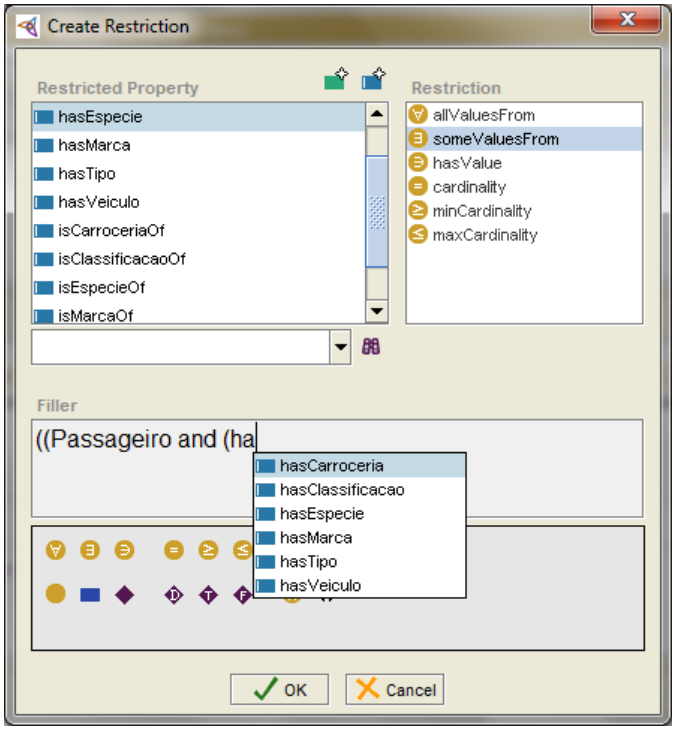

Figura 31 – Exemplificação do uso de Autocomplete

Criada a restrição pressiona-se o botão *OK* e a restrição será adicionada em *Asserted Conditions* (Figura 32)

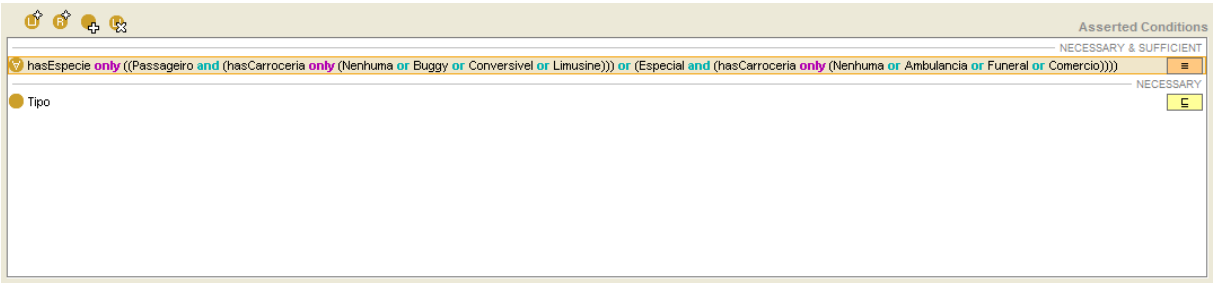

Figura 32 – Exemplificação de restrição adicionada

Ao se inserir esta restrição, não se garante que o relacionamento entre a Espécie (através de *hasEspecie*) é necessário, Informa-se apenas que se houver o relacionamento deverá seguir tal restrição. Para gerar a obrigatoriedade do relacionamento, deve-se inserir restrições de cardinalidade. O processo para se inserir tal restrição é o mesmo do descrito acima. Após inseridas todas as restrições

necessárias, a classe *Automoveis* possuirá restrições conforme Figura 33.

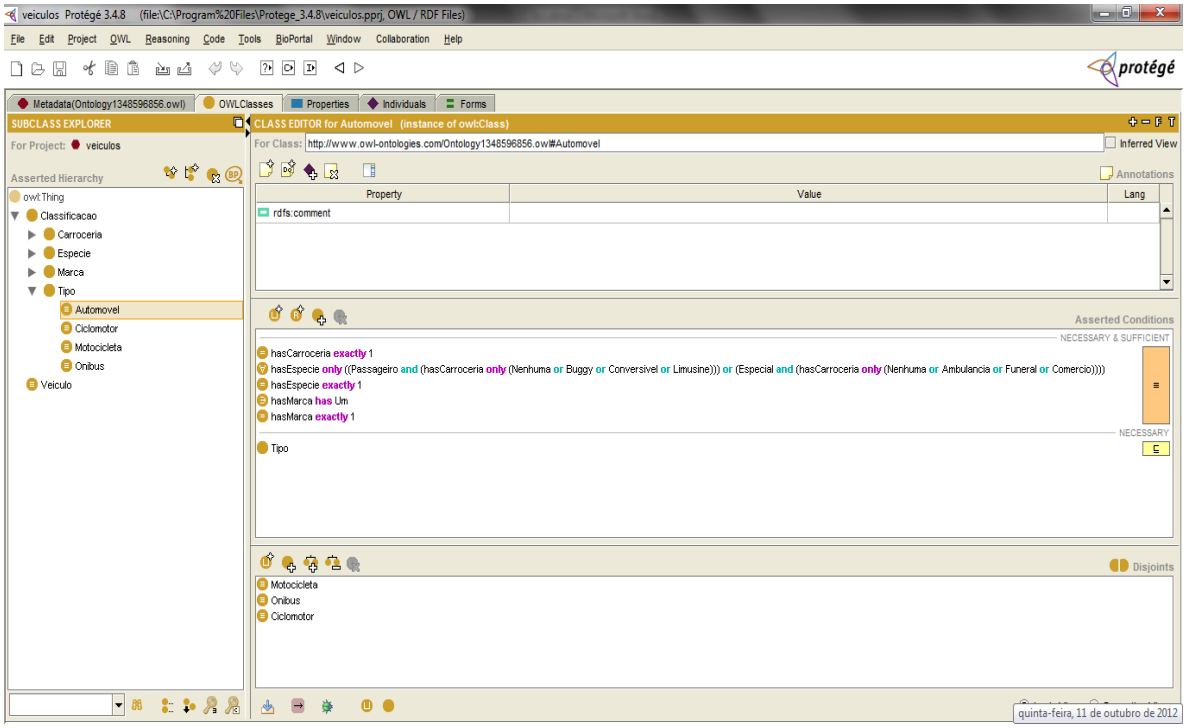

Figura 33 – Restrições da classe Automoveis

As restrições de cardinalidade *hasCarroceria exactly 1*, *hasEspecie exactly 1* e *hasMarca exactly 1* são restrições de cardinalidade exata, isso significa que para um Indivíduo que se declare *Automovel* deve possuir obrigatoriamente uma carroceria, uma espécie e uma marca. A espécie e carroceria suportadas são as declaradas em *hasEspecie only ((Passageiro and (hasCarroceria only (Nenhuma or Buggy or Conversivel or Limusine))) or (Especial and (hasCarroceria only (Nenhuma or Ambulancia or Funeral or Comercio))))* . Já a Marca deve possuir exatamente um membro da classe *Um*.

Outras classes na construção desta ontologia possuem restrições. As restrições da classe *Ciclomotor* são apresentadas na Figura 34. As restrições da classe *Motocicleta* são apresentadas na Figura 35. As restrições da classe *Onibus* são apresentadas na Figura 36. Por fim, as restrições da classe *Veículos* são apresentadas na Figura 37.

Figura 34 – Restrições da classe Ciclomotor

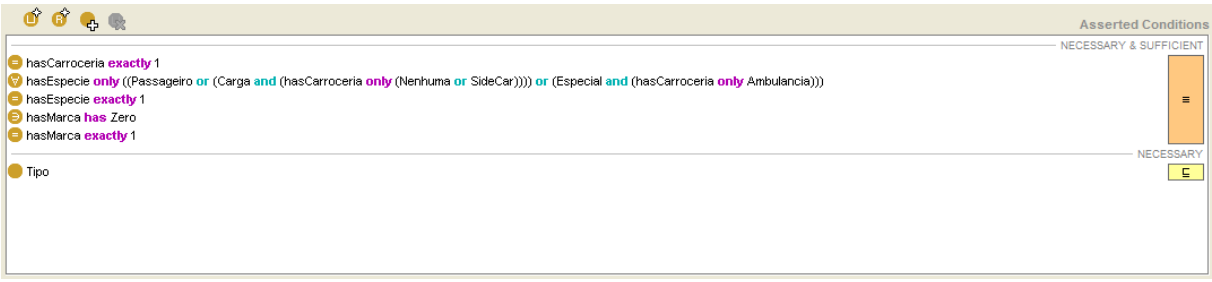

Figura 35 – Restrições da classe Motocicleta

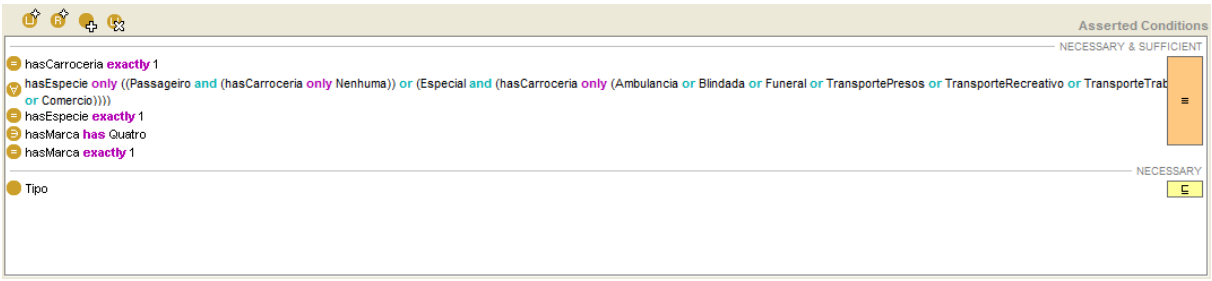

Figura 36 – Restrições da classe Onibus

Figura 37 – Restrições da classe Veiculo

6.7. CRIAÇÃO DE INDIVÍDUOS

O OWL permite definir indivíduos e declarar propriedades sobre eles. Os indivíduos podem também ser usados em descrições de classe, a saber, em restrições *hasValue* e Classes Enumeradas. Para criar indivíduos no *Protege-OWL* usa-se a aba *Individuals* (Indivíduos), conforme Figura 38.

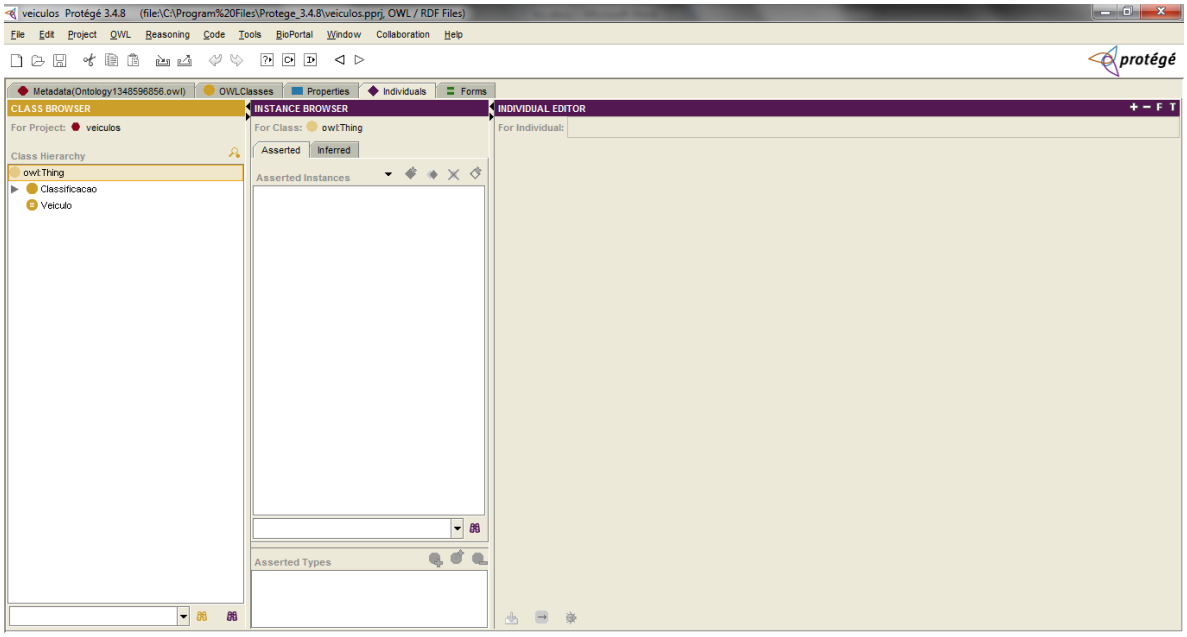

Figura 38 – Aba Individuals

Para criar Indivíduos, que são instâncias das classes, seleciona-se em *Class Browser* a classe na qual será criado o indivíduo. Em seguida, em *Instance Browser* seleciona-se o ícone *Create instance* (Tabela 6). Feito isso, na interface *Individual editor* tem-se a possibilidade de inserir propriedades de anotações para os indivíduos, e, caso a classe, cujo indivíduo foi instanciado, possua restrições, é apresentado um formulário de preenchimento para relacionar o indivíduo criado com outros indivíduos.

Ícone	Descrição
	Create Instance – Criar indivíduo
	Copy Instance – Criar cópia de indivíduo existente
	Delete instance – Apagar indivíduo
	Create anonymous instance – Criar indivíduo anônimo

Tabela 6: Ícones da Interface Instance Browser

6.8. TESTES EM ONTOLOGIAS

Protege-OWL possibilita a aplicação de vários testes na ontologia em edição. Tais testes variam desde testes de sanidade, como por exemplo, verificar se uma característica da propriedade corresponde a sua correspondente na propriedade inversa.

A estrutura de testes é baseada em uma arquitetura de plug-ins que permite a adição de novos testes por terceiros. Os testes podem ser configurados através da aba *Test* (Figura 39), acessados através do comando *Preferences* do menu *OWL.* Após delimitados quais testes serão efetuados utiliza-se o comando *Run Ontology tests...* do mesmo menu.

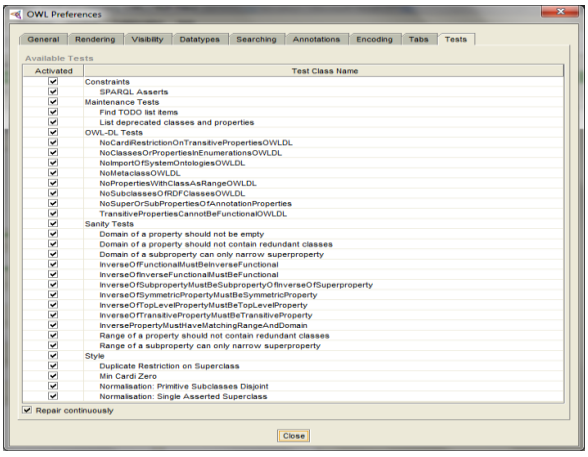

Figura 39 – Delimitação dos testes

6.9. IMPORTAÇÃO DE ONTOLOGIAS

Ontologias OWL podem importar uma ou várias outras ontologias. Quando uma ontologia importa outra, não é feita uma simples referência às classes, às propriedades e aos indivíduos: os axiomas e fatos contidos na ontologia importada são incluídos na ontologia destino.

No *Protege-OWL* a importação de ontologias é normalmente coordenada pelo uso de namespaces. Configuram-se o namespace e o prefixo de namespace da ontologia a importar e em seguida ocorre a importação. Para importar uma ontologia no *Protege-OWL* é preciso primeiro localizá-la e determinar sua URL.

6.10. PROPRIEDADES DE ANOTAÇÃO

O OWL permite que classes, propriedades, indivíduos e o próprio cabeçalho da ontologia sejam comentados usando metadados.

A OWL possui cinco propriedades de anotação pré-definidas que podem ser usadas para fazer comentários em classes propriedades e indivíduos:

<*owl:versionInfo*>: sintaxe utilizada para armazenar informações da versão da ontologia. O dado armazenado é uma *string.*

<*rdfs:label*>: esta propriedade é usada para adicionar nomes significativos aos elementos da ontologia, tais como classes, propriedades e indivíduos. Também é utilizada para fornecer nomes multilíngues para elementos da ontologia, no qual são especificados os idiomas a que se referem. Esta sintaxe possui como escopo uma *string.*

<*rdfs:comment*>: propriedade utilizada para armazenar comentários sobre as classes, também é armazenada sob o formato de *string.*

<*rdfs:seeAlso*>: armazena uma URI usada para identificar recursos da ontologia.

<rdfs:isDefinedBy>: armazena uma URI usada para referenciar uma ontologia que define elementos da ontologia tais como classes, propriedades e indivíduos.

Em OWL também são utilizadas propriedades de anotações que agem diretamente sobre a ontologia, e não em suas classes, propriedades ou indivíduos. Tais propriedades armazenam URIs que possibilitam referencias outras ontologias.

<*owl:priorVersion*>: identifica versões anteriores da ontologia.

<*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.

<*owl:incompatibleWith*>: identifica a versão anterior de uma ontologia que não é compatível com a atual.

No editor de ontologias *Protégé* a interface de propriedades de anotações (Figura 40) está presentes nas abas *Metadata*, *OWL-Classes*, *Properties* e *Individuals.*

Para criar uma nova propriedade de anotação utiliza-se o botão *Create new annotation value*. Ao pressioná-lo é aberta uma caixa de diálogo (Figura 41) no qual se delimita qual propriedade será utilizada. Após a escolha, a propriedade é inserida na interface *Annotations* podendo se alterada seu valor (*value*) e seu respectivo idioma (*Lang*).

Figura 40 – Interface das propriedades de anotação

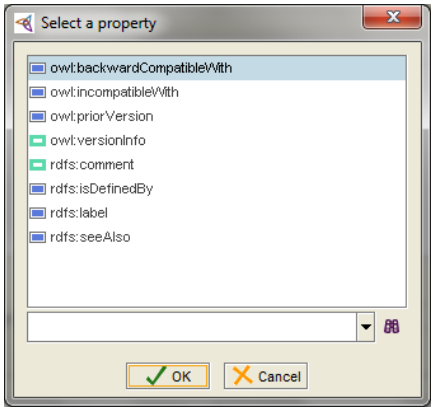

Figura 41 – Caixa de diálogo para seleção de propriedades de anotação

7. CONSIDERAÇÕES FINAIS

Tendo como base todo o panorama discutido e apresentado nessa pesquisa, conclui-se que a Web Semântica não é apenas uma ferramenta para conduzir e auxiliar a execução de tarefas individuais e de pesquisas mais eficientes na Web, mas também uma ferramenta para assistir no desenvolvimento do conhecimento. A Web Semântica oferece um novo e promissor paradigma de produção de conhecimento baseado em ontologias.

Tem-se também que as Ontologias são o alicerce da proposta da Web Semântica. Com o uso de ontologias é possível especificar formalmente as relações existentes entre entidades e suas propriedades possibilitando gerar conhecimento semântico sobre um domínio específico.

Verificou-se também quais recursos, propriedades e funções a *OWL* possui para o desenvolvimento de ontologias. Também foram analisados os aspectos que diferenciam suas sublinguagens. E que a escolha por uma das sublinguagens OWL dependerá do grau de aprofundamento no domínio esperado pelos desenvolvedores de ontologias.

Tem-se, ainda, que a ferramenta de edição de Ontologias Protégé contempla os requisitos abordados pela W3C para a construção da Web Semântica apoiados na linguagem OWL.

Com base no domínio da Classificação de Veículos, foram verificados quais passos são necessários para a criação de classes, subclasses, propriedades e as relações existentes entre elas. Foram implementados os conceitos de classes disjuntas; propriedades funcionais e propriedades funcionais inversas; delimitação de domínio (*domain*) e escopo (*range*) das classes; criação de restrições do tipo *allValuesFrom*, *someValuesFrom* e *hasValue*; criação de restrições de cardinalidade; e criação de indivíduos.

Também foi apresentada a possibilidade de efetuar testes nas ontologias, de realização de importação de ontologias e as propriedades de anotação.

Todos estes recursos são de simples manuseio na ferramenta *Protégé-OWL*.

Verificou-se também que a ferramenta possui facilidades para o desenvolvedor de ontologias: possibilidade de utilização de métodos *Wizard* e recursos de autocomplemento (*autocomplete*).

No contexto deste projeto é possível destacar alguns assuntos que merecem aprofundamento em futuras pesquisas. Pode-se efetuar a análise da ferramenta *Protégé-OWL* na construção de ontologias no que diz respeito ao mapeamento de uma base de dados relacional. Outra possibilidade é a criação de um Agente de Software que utilize o domínio de conhecimento gerado neste projeto.

Pode-se considerar que as ontologias, certamente, se tornarão o que o HTML representa nos dias atuais. E que a escolha da ferramenta *Protégé-OWL* por desenvolvedores, inevitavelmente, trará benefícios na construção de ontologias pautadas em OWL.

Por fim, o surgimento acelerado de novas tecnologias requer dos profissionais de Ciência da Computação uma pesquisa contínua, lançando sobre tais tecnologias um olhar crítico a fim de avaliar a adequação das ferramentas que implentam tais tecnologias.

REFERÊNCIAS

ALJAWARNEH, Shadi; ALKHATEEB, Faisal; AL MAGHAYREH, Eslam. A Semantic Data Validation Service for Web Applications. **J. theor. appl. electron. commer. res.**, v. 5, n. 1, 2010, p. 39 – 55. Disponível em: <http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-18762010000100005 &lng=es&nrm=iso>. Acesso em: 20 mar. 2012.

ALLEMANG, Dean; HENDLER, Jim. **Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL**. Burlington: Elsevier, 2008.

ALMEIDA, Maurício B.; BAX, Marcelo P. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção**. Ciência da Informação**, Brasília, v.32, n.3, p.7-20, set./dez. 2003. Disponível em: <http://www.scielo.br/pdf/ci/v32n3/19019.pdf>. Acesso em: 02 abr. 2012.

BECKETT, Dave. **RDF/XML Syntax Specification (Revised)**. The World Wide Web Consortium (W3C). 2004. Disponível em: < http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ >. Acesso em: 15 jun. 2012.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. **The Semantic Web: a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities.** Scientific American. 2001. Disponível em: <http://www.jeckle.de/files/tblSW.pdf>. Acesso em: 25 mar. 2012.

BRAY, Tim, et al. **Extensible markup language XML 1.0**. 5. ed.The World Wide Web Consortium (W3C). 2004. Disponível em: <http://www.w3.org/TR/REC-xml/>. Acesso em 20 abr. 2012.

BRINKMANN, Martin. Microsoft Powerset Adds Semantic Search To Wikipedia. **Ghacks Technology News**. 2 dez. 2009. Disponível em: <http:// www.ghacks.net/2009/12/02/microsoft-powerset-adds-semantic-search-to-ikipedia/>. Acesso em 20 set. 2012.

CARVALHO, Rodrigo Aquino de. **Perspectivas na web semântica para a Ciência da informação**. 2009, 186p. Dissertação (Mestrado) - Centro de Ciências Humanas e Sociais Aplicadas - Programa de Pós-Graduação em Ciência da Informação, Pontifícia Universidade Católica de Campinas, Campinas, 2009.

EFRATI, Amir. Google Gives Search a Refresh. **The Wall Street Journal**. 15 mar. 2012. Technology. Disponível em: <http://online.wsj.com/article_email/SB10001424052702304459804577281842851136290-lMyQjAxMTAyMDEwNDExNDQyWj.html>. Acesso em 20 set. 2012.

FERNEDA, Edberto. **Recuperação de Informação: Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação.** 2003, 137 p.Tese (Doutorado) – Escola de Documentação e Artes – Universidade de São Paulo, São Paulo, 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/27/27143/tde-15032004-130230/pt-br.php>. Acesso em: 13 mar. 2012.

HEFLIN, Jeff. **OWL Web Ontology Language: use cases and requirements**. The World Wide Web Consortium (W3C). 2004. Disponível em: <http://www.w3.org/TR/webont-req/>. Acesso em: 12 mai. 2012.

HORRIDGE, Mathew, et al. **A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools**, The University of Manchester: Manchester, 2004.

KLINE, Graham; CARROLL, Jeremy J. **Resource Description Framework (RDF):Concepts and Abstract Syntax.** The World Wide Web Consortium (W3C). 2004. Disponível em: <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-rdf-triple>. Acesso em 10 abr. 2012

KNUBLAUCH, Holger; MUSEN, Mark A. **Editing Description Logic Ontologies with the Protégé OWL Plugin**. Stanford University, CA. Disponível em: < http://protege.stanford.edu/doc/users.html>. Acesso em 08 jun. 2012.

KNUBLAUCH, Holger; et al. **The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications**. ISWC, 2004. Disponível em: <http://protege.stanford.edu/plugins/owl/publications/ISWC2004-protege-owl.pdf>. Acesso em: 30 mar. 2012.

MARTINS, Euclides Edson; SALDIAS, Claudio Ponce. **PROTÉGÉ.** Universidade Federal de Santa Catarina (UFSC), Florianópolis. Disponível em <http://www.das.ufsc.br/~gb/pg-ia/Protege08/claudio_euclides_artigo_protege_rev5.pdf>. Acesso em 17 mai 2012.

MCGUINNESS, Deborah L.; HARMELEN, Frank Van. **OWL Web Ontology Language Overview**. The World Wide Web Consortium (W3C). 2004. Disponível em: < http://www.w3.org/TR/2004/REC-owl-features-20040210/ >. Acesso em: 05 abr. 2012.

OLIVEIRA, Edgard Costa. **Autoria de documentos para a Web Semantica: um ambiente de produção de conhecimento baseado em ontologias**. 2006, 260p.Tese (Doutorado) – Departamento de Ciência da Informação e Documentação – Universidade de Brasília, Brasília, 2006. Disponível em: <http://repositorio.bce.unb.br/bitstream/10482/4794/1/2006_Edgard%20Costa%20Oliveira.pdf>. Acesso em: 02 abr. 2012.

RIBEIRO, John. Wallmart aumenta venda com busca emântica. **IDGNOW!**. 03 set. 2012. Technology. Disponível em: <http://idgnow.uol.com.br/internet/2012/09/03/walmart-aumenta-vendas-com-busca-semantica/>. Acesso em 20 set. 2012.

SEGARAN, Toby; EVANS, Colin; TAYLOR, Jamie. **Programming the Semantic Web.** Sebastopol: O'Reilly Media, 2009.

SOUZA, Renato Rocha; ALVARENGA, Lídia. A Web Semântica e suas contribuições para a ciência da informação. **Ciência da Informação**, v. 33, n. 1, 2004. Disponível em: <http://revista.ibict.br/ciinf/index.php/ciinf/article/view/50>. Acesso em: 26 mar. 2012.

W3C**. World Wide Web Consortium Issues RDF and OWL Recommendations**. 2004. Disponível em: <http://www.w3.org/2004/01/sws-pressrelease>. Acesso em: 30 mar. 2012

ANEXO A – PORTARIA DENATRAN

PORTARIA N º 309, DE 15 DE JUNHO DE 2012.

O DIRETOR DO DEPARTAMENTO NACIONAL DE TRÂNSITO - DENATRAN, no uso da atribuição que lhe foi conferida pelo artigo 19, inciso XXVI do Código de Trânsito Brasileiro,

Considerando o que consta no Processo nº 80000.011762/2012-12;

RESOLVE:

Art. 1º Alterar a Tabela 1 – Classificação de Veículos conforme Tipo/Marca/Espécie do ANEXO I da Portaria DENATRAN nº 1101, de 20 de dezembro de 2011.

Art. 2º Esta Portaria entra em vigor 30 dias após a data de sua publicação.

JULIO FERRAZ ARCOVERDE

ANEXO I

Tabela 1 - Classificação de Veículos Conforme Tipo/Marca/Espécie

Classificação de Veículos Conforme Tipo/Marca/Espécie						
Tipo	**Marca**	**Espécie**	**Carrocerias Possíveis**			
2-Ciclomotor	0	1-Passageiro	999-Nenhuma			
3-Motoneta	0	1-Passageiro	999-Nenhuma			
		2-Carga	999-Nenhuma			
4-Motocicleta	0	1-Passageiro	999-Nenhuma	119-SideCar		
		2-Carga	999-Nenhuma	119-SideCar		
		6-Especial	101-Ambulância			
5-Triciclo	0	1-Passageiro	999-Nenhuma	108-Carroc Fechada		
		2-Carga	999-Nenhuma	107-Carroc Aberta	108-Carroc Fechada	
6-Automóvel	1	1-Passageiro	999-Nenhuma	105-Buggy	110-Conversível	115-Limusine
		6-Especial	999-Nenhuma	101-Ambulância	111-Funeral	178-Comércio
7-Microônibus	4	1-Passageiro	999-Nenhuma			
		6-Especial	101-Ambulância	103-Blindada	111-Funeral	124-Transp Presos
			125 - Transp Recreat	126-Transp Trabalh	178-Comércio	
8-Ônibus	4	1-Passageiro	999-Nenhuma			
		6-Especial	101-Ambulância	103-Blindada	111-Funeral	124-Transp Presos
			125 - Transp Recreat	126-Transp Trabalh	178-Comércio	
10-Reboque	6,7	1-Passageiro	123-Transp Militar	124-Transp Presos	125 - Transp Recreat	126-Transp Trabalh

Classificação de Veículos Conforme Tipo/Marca/Espécie						
Tipo	**Marca**	**Espécie**	**Carrocerias Possíveis**			
		2-Carga	102-Basculante	107-Carroc Aberta	108-Carroc Fechada	109-Chassi Conteiner
			116-Mec Operacional	118-Prancha	120-Silo	121-Tanque
			127-Conteiner/C Ab	128-Prancha Contein	132-Intercambiável	133-Roll-on Roll-off
			143-Transp Toras	145-Ab/Mec Operac	146-Fech/Mec Operac	179-Transp Granito
			180-Silo/Basculante	181- Basc/Mec Operac		
		6-Especial	101-Ambulância	111-Funeral	122-Trailler	130-Trio Eletrico
			131-Dolly			
11-Semi-Reboque	6,7	1-Passageiro	123-Transp Militar	124-Transp Presos	125 - Transp Recreat	126-Transp Trabalh
		2-Carga	102-Basculante	107-Carroc Aberta	108-Carroc Fechada	109-Chassi Conteiner
			116-Mec Operacional	118-Prancha	120-Silo	121-Tanque
			127-Conteiner/C Ab	128-Prancha Contein	132-Intercambiável	133-Roll-on Roll-off
			143-Transp Toras	145-Ab/Mec Operac	146-Fech/Mec Operac	179-Transp Granito
			179-Transp Granito	180-Silo/Basculante	181-Basc/Mec Operac	
		6-Especial	101-Ambulância	111-Funeral	122-Trailler	130-Trio Eletrico
			131-Dolly			
13-Camioneta	2	3-Misto	999-Nenhuma			
		6-Especial	101-Ambulância	111-Funeral	115-Limusine	124-Transp Presos
			178-Comércio			
14-Caminhão	3	2-Carga	102-Basculante	107-Carroc Aberta	108-Carroc Fechada	109-Chassi Conteiner
			112-Furgão	116-Mec Operacional	118-Prancha	120-Silo
			121-Tanque	127-Conteiner/C Ab	128-Pr Contein	133-Rollon Rolloff
			135-Aberta/C Estend	138-Fech/C Estend	140-Ab/Intercamb	143-Transp Toras
			144-Inac/C Est	145-Ab/Mec Operac	146-Fech/Mec Operac	147-Tanq/M Operac
			148-Pranc/M Operac	150-Ab/M Op/C Est	153-Fec/M Op/C Est	156-Tanque/C Estend
			159-Tanq/M Op/C Est	162-Rollon/C Estend	165-Basc/C Estend	168-Pracha/C Estend
			171-Pr/M Op/C Est	174-Ab/Interc/C Est	180-Silo/Basculante	179-Transp Granito

Classificação de Veículos Conforme Tipo/Marca/Espécie						
Tipo	**Marca**	**Espécie**	**Carrocerias Possíveis**			
			181- Basc/Mec Operac	182- Ch Cteiner/C Estend	183- Mec Operac/C Estend	184-Silo/C Estend
			185-Conteiner/C Ab/C Estend	186- Pr Contein/C Estend	187- Tran. Toras/C Estend	188- Silo/Basc/C Estend
		6-Especial	101- Ambulância	103-Blindada	104-Bombeiro	111-Funeral
			115-Limusine	116-Mec Operacional	123-Transp Militar	124-Transp Presos
			125-Transp Recreat	126-Transp Trabalh	130-Trio Eletrico	134-Aberta/C Dupla
			136-Aberta/C Supl	137-Fech/C Dupla	139-Fech/C Suplem	141-C Dup/Inac.
			142 – Mec Op/C Dup	149-Ab/M Op/C Dupl	151-Ab/ M Op/C Supl	152-Fec/M Op/C Dup
			154-Fec/M Op/C Sup	155-Tanque/C Dupla	157-Tanque/C Suplem	158-Tanq/M Op/C Dup
			160-Tanq/M Op/C Sup	161-Rollon/C Dupla	163-Rollon/C Suplem	164-Basc/C Dupla
			166-Basc/C Suplem	167-Prancha/C Dupla	169-Prancha/C Supl	170-Pr/M Op/C Dup
			172-Pr/M Op/C Supl	173-Ab/Interc/C Dup	175- Ab/Interc/C Supl	176-Aberta/C Tripla
			177-Fechada/C Tripla	178-Comércio		
17-Caminhão Trator	3	5-Tração	999-Nenhuma	116-Mec Operacional	129-C Estend	
18-Tr Rodas	5	5-Tração	999-Nenhuma			
19-Tr Esteiras	5	5-Tração	999-Nenhuma			
20-Tr Misto	5	5-Tração	999-Nenhuma			
21-Quadriciclo	0	1-Passageiro	999-Nenhuma			
		2-Carga	999-Nenhuma			
22-Chassi Plataforma	9	1-Passageiro	Não se aplica			
		6-Especial	Não se aplica			
23- Caminhonete	2	2-Carga	102-Basculante	107-Carroc Aberta	108-Carroc Fechada	112-Furgão
			116-Mec Operacional	121-Tanque	135-Aberta/Cab Est	138-Fech/C Estend
			140- Ab/Intercamb	144-Inac/C Est	150-Ab/M Op/C Est	174- Ab/Interc/C Est
			181- Basc/Mec Operac	183- Mec Operac/C Estend		
		6-Especial	101- Ambulância	111-Funeral	115-Limusine	123-Transp Militar

Classificação de Veículos Conforme Tipo/Marca/Espécie						
Tipo	**Marca**	**Espécie**	**Carrocerias Possíveis**			
			124-Transp Presos	126-Transp Trabalh	134-Aberta/Cab Dupla	137-Fechada/C Dupla
			141-C Dup/Inac.	145-Ab/Mec Operac	146-Fech/Mec Operac	149-Ab/M Op/C Dupl
			173-Ab/Interc/C Dupl	175-Ab/Interc/C Supl	178-Comércio	
25 - Utilitário	2	3-Misto	999-Nenhuma	107-Carroc Aberta	108-Carroc Fechada	113-Jipe
		6-Especial	101-Ambulância	111-Funeral	115-Limusine	124-Transp Presos
			178-Comércio			
26 - Motor-casa	8	6-Especial	108-Carroc Fechada			

As espécies 4-Competição e 7-Coleção devem ser registradas com o tipo e carrocerias originais do veículo.

ANEXO B – CÓDIGO XML GERADO

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
    <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>


<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1348596856.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1348596856.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="Ambulancia">
        <rdfs:subClassOf rdf:resource="#Carroceria"/>
        <owl:disjointWith rdf:resource="#Blindada"/>
        <owl:disjointWith rdf:resource="#Buggy"/>
        <owl:disjointWith rdf:resource="#Comercio"/>
        <owl:disjointWith rdf:resource="#Conversivel"/>
        <owl:disjointWith rdf:resource="#Funeral"/>
        <owl:disjointWith rdf:resource="#Limusine"/>
        <owl:disjointWith rdf:resource="#Nenhuma"/>
        <owl:disjointWith rdf:resource="#SideCar"/>
        <owl:disjointWith rdf:resource="#TransportePresos"/>
        <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
        <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
    </owl:Class>
    <owl:Class rdf:ID="Automovel">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasCarroceria"/>
                        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasEspecie"/>
                        <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasEspecie"/>
```

```
            <owl:allValuesFrom>
               <owl:Class>
                  <owl:unionOf rdf:parseType="Collection">
                     <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                           <owl:Class rdf:about="#Especial"/>
                           <owl:Restriction>
                              <owl:onProperty rdf:resource="#hasCarroceria"/>
                              <owl:allValuesFrom>
                                 <owl:Class>
                                    <owl:unionOf rdf:parseType="Collection">
                                       <owl:Class rdf:about="#Ambulancia"/>
                                       <owl:Class rdf:about="#Comercio"/>
                                       <owl:Class rdf:about="#Funeral"/>
                                       <owl:Class rdf:about="#Nenhuma"/>
                                    </owl:unionOf>
                                 </owl:Class>
                              </owl:allValuesFrom>
                           </owl:Restriction>
                        </owl:intersectionOf>
                     </owl:Class>
                     <owl:Class>
                        <owl:intersectionOf rdf:parseType="Collection">
                           <owl:Restriction>
                              <owl:onProperty rdf:resource="#hasCarroceria"/>
                              <owl:allValuesFrom>
                                 <owl:Class>
                                    <owl:unionOf rdf:parseType="Collection">
                                       <owl:Class rdf:about="#Buggy"/>
                                       <owl:Class rdf:about="#Conversivel"/>
                                       <owl:Class rdf:about="#Limusine"/>
                                       <owl:Class rdf:about="#Nenhuma"/>
                                    </owl:unionOf>
                                 </owl:Class>
                              </owl:allValuesFrom>
                           </owl:Restriction>
                           <owl:Class rdf:about="#Passageiro"/>
                        </owl:intersectionOf>
                     </owl:Class>
                  </owl:unionOf>
               </owl:Class>
            </owl:allValuesFrom>
         </owl:Restriction>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#hasMarca"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
         </owl:Restriction>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#hasMarca"/>
            <owl:hasValue rdf:resource="#Um"/>
         </owl:Restriction>
      </owl:intersectionOf>
   </owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="#Tipo"/>
<owl:disjointWith rdf:resource="#Ciclomotor"/>
<owl:disjointWith rdf:resource="#Motocicleta"/>
<owl:disjointWith rdf:resource="#Onibus"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Blindada">
   <rdfs:subClassOf rdf:resource="#Carroceria"/>
   <owl:disjointWith rdf:resource="#Ambulancia"/>
   <owl:disjointWith rdf:resource="#Buggy"/>
   <owl:disjointWith rdf:resource="#Comercio"/>
   <owl:disjointWith rdf:resource="#Conversivel"/>
   <owl:disjointWith rdf:resource="#Funeral"/>
   <owl:disjointWith rdf:resource="#Limusine"/>
   <owl:disjointWith rdf:resource="#Nenhuma"/>
   <owl:disjointWith rdf:resource="#SideCar"/>
   <owl:disjointWith rdf:resource="#TransportePresos"/>
   <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
   <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Buggy">
   <rdfs:subClassOf rdf:resource="#Carroceria"/>
   <owl:disjointWith rdf:resource="#Ambulancia"/>
   <owl:disjointWith rdf:resource="#Blindada"/>
   <owl:disjointWith rdf:resource="#Comercio"/>
   <owl:disjointWith rdf:resource="#Conversivel"/>
   <owl:disjointWith rdf:resource="#Funeral"/>
   <owl:disjointWith rdf:resource="#Limusine"/>
   <owl:disjointWith rdf:resource="#Nenhuma"/>
   <owl:disjointWith rdf:resource="#SideCar"/>
   <owl:disjointWith rdf:resource="#TransportePresos"/>
   <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
   <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Carga">
   <rdfs:subClassOf rdf:resource="#Especie"/>
   <owl:disjointWith rdf:resource="#Especial"/>
   <owl:disjointWith rdf:resource="#Passageiro"/>
</owl:Class>
<owl:Class rdf:ID="Carroceria">
   <rdfs:subClassOf rdf:resource="#Classificacao"/>
</owl:Class>
<owl:Class rdf:ID="Ciclomotor">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasCarroceria"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasCarroceria"/>
               <owl:hasValue rdf:resource="#Nenhuma"/>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasEspecie"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasEspecie"/>
               <owl:hasValue rdf:resource="#Passageiro"/>
            </owl:Restriction>
            <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#hasMarca"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasMarca"/>
                <owl:hasValue rdf:resource="#Zero"/>
            </owl:Restriction>
        </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Tipo"/>
  <owl:disjointWith rdf:resource="#Automovel"/>
  <owl:disjointWith rdf:resource="#Motocicleta"/>
  <owl:disjointWith rdf:resource="#Onibus"/>
</owl:Class>
<owl:Class rdf:ID="Classificacao">
  <rdfs:comment xml:lang="pt"></rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Comercio">
  <rdfs:subClassOf rdf:resource="#Carroceria"/>
  <owl:disjointWith rdf:resource="#Ambulancia"/>
  <owl:disjointWith rdf:resource="#Blindada"/>
  <owl:disjointWith rdf:resource="#Buggy"/>
  <owl:disjointWith rdf:resource="#Conversivel"/>
  <owl:disjointWith rdf:resource="#Funeral"/>
  <owl:disjointWith rdf:resource="#Limusine"/>
  <owl:disjointWith rdf:resource="#Nenhuma"/>
  <owl:disjointWith rdf:resource="#SideCar"/>
  <owl:disjointWith rdf:resource="#TransportePresos"/>
  <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
  <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Conversivel">
  <rdfs:subClassOf rdf:resource="#Carroceria"/>
  <owl:disjointWith rdf:resource="#Ambulancia"/>
  <owl:disjointWith rdf:resource="#Blindada"/>
  <owl:disjointWith rdf:resource="#Buggy"/>
  <owl:disjointWith rdf:resource="#Comercio"/>
  <owl:disjointWith rdf:resource="#Funeral"/>
  <owl:disjointWith rdf:resource="#Limusine"/>
  <owl:disjointWith rdf:resource="#Nenhuma"/>
  <owl:disjointWith rdf:resource="#SideCar"/>
  <owl:disjointWith rdf:resource="#TransportePresos"/>
  <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
  <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Especial">
  <rdfs:subClassOf rdf:resource="#Especie"/>
  <owl:disjointWith rdf:resource="#Carga"/>
  <owl:disjointWith rdf:resource="#Passageiro"/>
</owl:Class>
<owl:Class rdf:ID="Especie">
  <rdfs:subClassOf rdf:resource="#Classificacao"/>
</owl:Class>
<owl:Class rdf:ID="Funeral">
  <rdfs:subClassOf rdf:resource="#Carroceria"/>
  <owl:disjointWith rdf:resource="#Ambulancia"/>
  <owl:disjointWith rdf:resource="#Blindada"/>
```

```
    <owl:disjointWith rdf:resource="#Buggy"/>
    <owl:disjointWith rdf:resource="#Comercio"/>
    <owl:disjointWith rdf:resource="#Conversivel"/>
    <owl:disjointWith rdf:resource="#Limusine"/>
    <owl:disjointWith rdf:resource="#Nenhuma"/>
    <owl:disjointWith rdf:resource="#SideCar"/>
    <owl:disjointWith rdf:resource="#TransportePresos"/>
    <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
    <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:FunctionalProperty rdf:ID="hasCarroceria">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:domain rdf:resource="#Classificacao"/>
    <owl:inverseOf rdf:resource="#isCarroceriaOf"/>
    <rdfs:range rdf:resource="#Carroceria"/>
    <rdfs:subPropertyOf rdf:resource="#hasClassificacao"/>
</owl:FunctionalProperty>
<owl:ObjectProperty rdf:ID="hasClassificacao">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <owl:inverseOf rdf:resource="#isClassificacaoOf"/>
    <rdfs:range rdf:resource="#Classificacao"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:ID="hasEspecie">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:domain rdf:resource="#Tipo"/>
    <owl:inverseOf rdf:resource="#isEspecieOf"/>
    <rdfs:range rdf:resource="#Especie"/>
    <rdfs:subPropertyOf rdf:resource="#hasClassificacao"/>
</owl:FunctionalProperty>
<owl:ObjectProperty rdf:ID="hasMarca">
    <rdfs:domain rdf:resource="#Classificacao"/>
    <owl:inverseOf rdf:resource="#isMarcaOf"/>
    <rdfs:range rdf:resource="#Marca"/>
    <rdfs:subPropertyOf rdf:resource="#hasClassificacao"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTipo">
    <rdfs:domain rdf:resource="#Classificacao"/>
    <owl:inverseOf rdf:resource="#isTipoOf"/>
    <rdfs:range rdf:resource="#Tipo"/>
    <rdfs:subPropertyOf rdf:resource="#hasClassificacao"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasVeiculo">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <owl:inverseOf rdf:resource="#isVeiculoOf"/>
    <rdfs:range rdf:resource="#Veiculo"/>
</owl:ObjectProperty>
<owl:InverseFunctionalProperty rdf:ID="isCarroceriaOf">
    <rdf:type rdf:resource="&owl;ObjectProperty"/>
    <rdfs:domain rdf:resource="#Classificacao"/>
    <owl:inverseOf rdf:resource="#hasCarroceria"/>
    <rdfs:range rdf:resource="#Carroceria"/>
    <rdfs:subPropertyOf rdf:resource="#isClassificacaoOf"/>
</owl:InverseFunctionalProperty>
<owl:ObjectProperty rdf:ID="isClassificacaoOf">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <owl:inverseOf rdf:resource="#hasClassificacao"/>
    <rdfs:range rdf:resource="#Classificacao"/>
</owl:ObjectProperty>
```

```
<owl:InverseFunctionalProperty rdf:ID="isEspecieOf">
   <rdf:type rdf:resource="&owl;ObjectProperty"/>
   <rdfs:domain rdf:resource="#Tipo"/>
   <owl:inverseOf rdf:resource="#hasEspecie"/>
   <rdfs:range rdf:resource="#Especie"/>
   <rdfs:subPropertyOf rdf:resource="#isClassificacaoOf"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:ID="isMarcaOf">
   <rdf:type rdf:resource="&owl;ObjectProperty"/>
   <rdfs:domain rdf:resource="#Classificacao"/>
   <owl:inverseOf rdf:resource="#hasMarca"/>
   <rdfs:range rdf:resource="#Marca"/>
   <rdfs:subPropertyOf rdf:resource="#isClassificacaoOf"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:ID="isTipoOf">
   <rdf:type rdf:resource="&owl;ObjectProperty"/>
   <rdfs:domain rdf:resource="#Classificacao"/>
   <owl:inverseOf rdf:resource="#hasTipo"/>
   <rdfs:range rdf:resource="#Tipo"/>
   <rdfs:subPropertyOf rdf:resource="#isClassificacaoOf"/>
</owl:InverseFunctionalProperty>
<owl:ObjectProperty rdf:ID="isVeiculoOf">
   <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
   <owl:inverseOf rdf:resource="#hasVeiculo"/>
   <rdfs:range rdf:resource="#Veiculo"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Limusine">
   <rdfs:subClassOf rdf:resource="#Carroceria"/>
   <owl:disjointWith rdf:resource="#Ambulancia"/>
   <owl:disjointWith rdf:resource="#Blindada"/>
   <owl:disjointWith rdf:resource="#Buggy"/>
   <owl:disjointWith rdf:resource="#Comercio"/>
   <owl:disjointWith rdf:resource="#Conversivel"/>
   <owl:disjointWith rdf:resource="#Funeral"/>
   <owl:disjointWith rdf:resource="#Nenhuma"/>
   <owl:disjointWith rdf:resource="#SideCar"/>
   <owl:disjointWith rdf:resource="#TransportePresos"/>
   <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
   <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Marca">
   <rdfs:subClassOf rdf:resource="#Classificacao"/>
</owl:Class>
<owl:Class rdf:ID="Motocicleta">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasCarroceria"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasEspecie"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasEspecie"/>
               <owl:allValuesFrom>
```

```
                <owl:Class>
                  <owl:unionOf rdf:parseType="Collection">
                    <owl:Class>
                      <owl:intersectionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Especial"/>
                        <owl:Restriction>
                          <owl:onProperty rdf:resource="#hasCarroceria"/>
                          <owl:allValuesFrom rdf:resource="#Ambulancia"/>
                        </owl:Restriction>
                      </owl:intersectionOf>
                    </owl:Class>
                    <owl:Class>
                      <owl:unionOf rdf:parseType="Collection">
                        <owl:Class>
                          <owl:intersectionOf rdf:parseType="Collection">
                            <owl:Class rdf:about="#Carga"/>
                            <owl:Restriction>
                              <owl:onProperty rdf:resource="#hasCarroceria"/>
                              <owl:allValuesFrom>
                                <owl:Class>
                                  <owl:unionOf rdf:parseType="Collection">
                                    <owl:Class rdf:about="#Nenhuma"/>
                                    <owl:Class rdf:about="#SideCar"/>
                                  </owl:unionOf>
                                </owl:Class>
                              </owl:allValuesFrom>
                            </owl:Restriction>
                          </owl:intersectionOf>
                        </owl:Class>
                        <owl:Class rdf:about="#Passageiro"/>
                      </owl:unionOf>
                    </owl:Class>
                  </owl:unionOf>
                </owl:Class>
              </owl:allValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasMarca"/>
              <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasMarca"/>
              <owl:hasValue rdf:resource="#Zero"/>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Tipo"/>
    <owl:disjointWith rdf:resource="#Automovel"/>
    <owl:disjointWith rdf:resource="#Ciclomotor"/>
    <owl:disjointWith rdf:resource="#Onibus"/>
  </owl:Class>
  <owl:Class rdf:ID="Nenhuma">
    <rdfs:subClassOf rdf:resource="#Carroceria"/>
    <owl:disjointWith rdf:resource="#Ambulancia"/>
    <owl:disjointWith rdf:resource="#Blindada"/>
    <owl:disjointWith rdf:resource="#Buggy"/>
    <owl:disjointWith rdf:resource="#Comercio"/>
```

```
    <owl:disjointWith rdf:resource="#Conversivel"/>
    <owl:disjointWith rdf:resource="#Funeral"/>
    <owl:disjointWith rdf:resource="#Limusine"/>
    <owl:disjointWith rdf:resource="#SideCar"/>
    <owl:disjointWith rdf:resource="#TransportePresos"/>
    <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
    <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
  </owl:Class>
  <owl:Class rdf:ID="Onibus">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasCarroceria"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasEspecie"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasEspecie"/>
            <owl:allValuesFrom>
              <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                  <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                      <owl:Class rdf:about="#Especial"/>
                      <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasCarroceria"/>
                        <owl:allValuesFrom>
                          <owl:Class>
                            <owl:unionOf rdf:parseType="Collection">
                              <owl:Class rdf:about="#Ambulancia"/>
                              <owl:Class rdf:about="#Blindada"/>
                              <owl:Class rdf:about="#Comercio"/>
                              <owl:Class rdf:about="#Funeral"/>
                              <owl:Class rdf:about="#TransportePresos"/>
                              <owl:Class rdf:about="#TransporteRecreativo"/>
                              <owl:Class rdf:about="#TransporteTrabalho"/>
                            </owl:unionOf>
                          </owl:Class>
                        </owl:allValuesFrom>
                      </owl:Restriction>
                    </owl:intersectionOf>
                  </owl:Class>
                  <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                      <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasCarroceria"/>
                        <owl:allValuesFrom rdf:resource="#Nenhuma"/>
                      </owl:Restriction>
                      <owl:Class rdf:about="#Passageiro"/>
                    </owl:intersectionOf>
                  </owl:Class>
                </owl:unionOf>
              </owl:Class>
            </owl:allValuesFrom>
```

```
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasMarca"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasMarca"/>
               <owl:hasValue rdf:resource="#Quatro"/>
            </owl:Restriction>
         </owl:intersectionOf>
      </owl:Class>
   </owl:equivalentClass>
   <rdfs:subClassOf rdf:resource="#Tipo"/>
   <owl:disjointWith rdf:resource="#Automovel"/>
   <owl:disjointWith rdf:resource="#Ciclomotor"/>
   <owl:disjointWith rdf:resource="#Motocicleta"/>
</owl:Class>
<owl:Class rdf:ID="Passageiro">
   <rdfs:subClassOf rdf:resource="#Especie"/>
   <owl:disjointWith rdf:resource="#Carga"/>
   <owl:disjointWith rdf:resource="#Especial"/>
</owl:Class>
<owl:Class rdf:ID="Quatro">
   <rdfs:subClassOf rdf:resource="#Marca"/>
   <owl:disjointWith rdf:resource="#Um"/>
   <owl:disjointWith rdf:resource="#Zero"/>
</owl:Class>
<owl:Class rdf:ID="SideCar">
   <rdfs:subClassOf rdf:resource="#Carroceria"/>
   <owl:disjointWith rdf:resource="#Ambulancia"/>
   <owl:disjointWith rdf:resource="#Blindada"/>
   <owl:disjointWith rdf:resource="#Buggy"/>
   <owl:disjointWith rdf:resource="#Comercio"/>
   <owl:disjointWith rdf:resource="#Conversivel"/>
   <owl:disjointWith rdf:resource="#Funeral"/>
   <owl:disjointWith rdf:resource="#Limusine"/>
   <owl:disjointWith rdf:resource="#Nenhuma"/>
   <owl:disjointWith rdf:resource="#TransportePresos"/>
   <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
   <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
<owl:Class rdf:ID="Tipo">
   <rdfs:subClassOf rdf:resource="#Classificacao"/>
</owl:Class>
<owl:Class rdf:ID="TransportePresos">
   <rdfs:subClassOf rdf:resource="#Carroceria"/>
   <owl:disjointWith rdf:resource="#Ambulancia"/>
   <owl:disjointWith rdf:resource="#Blindada"/>
   <owl:disjointWith rdf:resource="#Buggy"/>
   <owl:disjointWith rdf:resource="#Comercio"/>
   <owl:disjointWith rdf:resource="#Conversivel"/>
   <owl:disjointWith rdf:resource="#Funeral"/>
   <owl:disjointWith rdf:resource="#Limusine"/>
   <owl:disjointWith rdf:resource="#Nenhuma"/>
   <owl:disjointWith rdf:resource="#SideCar"/>
   <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
   <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
</owl:Class>
```

```
    <owl:Class rdf:ID="TransporteRecreativo">
      <rdfs:subClassOf rdf:resource="#Carroceria"/>
      <owl:disjointWith rdf:resource="#Ambulancia"/>
      <owl:disjointWith rdf:resource="#Blindada"/>
      <owl:disjointWith rdf:resource="#Buggy"/>
      <owl:disjointWith rdf:resource="#Comercio"/>
      <owl:disjointWith rdf:resource="#Conversivel"/>
      <owl:disjointWith rdf:resource="#Funeral"/>
      <owl:disjointWith rdf:resource="#Limusine"/>
      <owl:disjointWith rdf:resource="#Nenhuma"/>
      <owl:disjointWith rdf:resource="#SideCar"/>
      <owl:disjointWith rdf:resource="#TransportePresos"/>
      <owl:disjointWith rdf:resource="#TransporteTrabalho"/>
    </owl:Class>
    <owl:Class rdf:ID="TransporteTrabalho">
      <rdfs:subClassOf rdf:resource="#Carroceria"/>
      <owl:disjointWith rdf:resource="#Ambulancia"/>
      <owl:disjointWith rdf:resource="#Blindada"/>
      <owl:disjointWith rdf:resource="#Buggy"/>
      <owl:disjointWith rdf:resource="#Comercio"/>
      <owl:disjointWith rdf:resource="#Conversivel"/>
      <owl:disjointWith rdf:resource="#Funeral"/>
      <owl:disjointWith rdf:resource="#Limusine"/>
      <owl:disjointWith rdf:resource="#Nenhuma"/>
      <owl:disjointWith rdf:resource="#SideCar"/>
      <owl:disjointWith rdf:resource="#TransportePresos"/>
      <owl:disjointWith rdf:resource="#TransporteRecreativo"/>
    </owl:Class>
    <owl:Class rdf:ID="Um">
      <rdfs:subClassOf rdf:resource="#Marca"/>
      <owl:disjointWith rdf:resource="#Quatro"/>
      <owl:disjointWith rdf:resource="#Zero"/>
    </owl:Class>
    <owl:Class rdf:ID="Veiculo">
      <owl:equivalentClass>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasTipo"/>
          <owl:someValuesFrom rdf:resource="#Tipo"/>
        </owl:Restriction>
      </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="Zero">
      <rdfs:subClassOf rdf:resource="#Marca"/>
      <owl:disjointWith rdf:resource="#Quatro"/>
      <owl:disjointWith rdf:resource="#Um"/>
    </owl:Class>
  </rdf:RDF>
```