3. Ontologias

Este capítulo tem a finalidade de apresentar um estudo sobre ontologia. Neste estudo, apresentaremos a definição de ontologia e seus objetivos, bem como alguns dos métodos existentes para construção de ontologias, seus meta-modelos, linguagens e ferramentas utilizadas para apoiar a construção das ontologias.

3.1. Definição e Objetivos

A ontologia é definida por diversos autores na literatura, abaixo apresentamos algumas destas definições:

Para [Noy 01], ontologia é definida como:

> "*a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions)). An ontology together with a set of individual instances of classes constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins*."

Em [Oliveira 99], encontramos o seguinte conjunto de definições para ontologias:

> *"(a) Ontologia é uma especificação explícita de uma conceituação [Gruber 93,95].*
> *(b) Ontologia é uma descrição parcial e explícita de uma conceituação [Guarino 95].*

(c) Ontologia é uma teoria sobre um domínio que especifica um vocabulário de entidades, classes, propriedades, predicados e funções; e um conjunto de relações que, necessariamente, amarram esses vocabulários [Farquhar 99]."

Basicamente, podemos notar que ontologias são especificações formais de Universos de Informações, que tem, segundo [Macedo 03], como características importantes:

- Expressar o consenso do conhecimento de uma comunidade de pessoas em um UdI;
- servir como uma referência de termos definidos;
- fornecer uma linguagem suficientemente expressiva para uma comunicação eficaz entre as pessoas, ou seja, onde o entendimento seja único.

A ontologia, para área de informática, tem como objetivo facilitar o compartilhamento e reutilização das informações [Breitman 04], além de definir uma especificação conceitual para "os conhecimentos" de um determinado UdI.

Para [Noy 01], o resumo dos objetivos da ontologia são expressos nos seguintes itens:

- Compartilhar o entendimento comum da informação entre as pessoas e os agentes de *software*.
- Disponibilizar a reutilização do domínio da informação.
- Deixar a compreensão do UdI explícita.
- Separar o conhecimento do UdI do conhecimento operacional.
- Analisar o conhecimento do UdI.

Como vimos nas definições apresentadas, a ontologia é uma especificação formal de conceitos e termos do universo de informações, e objetiva utilizar esta especificação para compartilhar um entendimento único do UdI.

Para auxiliar na criação desta especificação, existem métodos que norteiam o desenvolvimento da ontologia, meta-modelos que auxiliam como guia das informações que devem estar contidas na ontologia, ferramentas para apoiar o desenvolvimento da ontologia e linguagens para auxiliar na representação da

mesma. Nas seções seguintes, apresentaremos uma pesquisa sobre cada um dos itens citados acima como facilitadores na construção de uma ontologia.

Antes de apresentar os itens facilitadores do desenvolvimento de ontologias, enfatizaremos alguns pontos que devem ser considerados na criação da ontologia [Noy 01]:

- Não existe uma maneira correta para modelar um UdI, sempre existirão várias alternativas. A melhor solução depende da aplicação da modelagem.
- Desenvolvimento de ontologia é um processo iterativo.
- Os conceitos, numa ontologia, têm que ser os mais semelhantes possíveis com os objetos e relacionamentos do UdI modelado.

3.2. Métodos para Construção de Ontologia

Os métodos para construção da ontologia nos mostram o passo-a-passo que deve ser executado para desenvolver uma ontologia de UdI. Abaixo apresentamos alguns deles:

- ***A Simple Knowledge-Engineering Methodology* [Noy 01]**

  A autora propõe o seguinte método para construção de uma ontologia de UdI:

  A. Determinar o UdI e o Escopo da Ontologia

  Para auxiliar nesta tarefa a autora propõe que sejam respondidas as seguintes questões:

  - O que o UdI que ontologia irá cobrir?
  - Para que iremos usar a ontologia?
  - Para que tipo de questões as informações da ontologia deveram prover respostas?
  - Quem irá usar e manter a ontologia?

B. Considerar a Reutilização de Ontologias Existentes

Ao iniciar o desenvolvimento da ontologia, deve-se considerar a possibilidade de alguém já ter desenvolvido alguma ontologia que possamos utilizar de fonte para refinar e estender seus conceitos no desenvolvimento da ontologia de UdI.

C. Enumerar Termos Importantes da Ontologia

Deve-se montar uma lista com termos que são usualmente utilizados em sentenças dos usuários do UdI. A partir destes termos, devemos formular perguntas sobre os mesmos, que auxiliem na derivação de novos termos. Por exemplo: quais são as propriedades dos termos? O que estes termos expressam no UdI?

Estes termos servirão de base para a definição das classes da ontologia do UdI.

D. Definir as Classes e a Hierarquia das Mesmas

Para a autora, existem três abordagens para definir a hierarquia de classes:

- ***Top-down***: São definidos os termos do UdI, que são mais genéricos, e a partir destes termos são realizadas especializações. Por exemplo: um termo bem genérico do UdI de vinhos é o próprio vinho, que pode ser especializado em vinho branco, tinto e rose.
- ***Bottom-up***: São definidos os termos mais específicos do UdI e, em seguida, são realizados agrupamentos destes termos a partir de características comuns, obtendo-se, desta forma, uma generalização dos mesmos.
- ***Combination***: É uma combinação das duas abordagens acima, onde são definidos termos considerados mais destacados dentro do UdI e, a partir dos mesmos, são realizadas especializações ou generalizações, conforme for mais apropriado.

E. Definir as Propriedades das Classes – "*Slots*"

Deve-se, para cada classe, descrever sua estrutura interna de conceitos, ou seja, suas propriedades. Por exemplo: cor e sabor são propriedades da classe vinho.

As propriedades das classes são classificadas como:

- Propriedades Intrínsecas: São propriedades da estrutura interna da classe. Por exemplo: sabor do vinho.
- Propriedades Extrínsecas: São propriedades da estrutura externa da classe. Por exemplo: nome do vinho.
- Propriedades Partes De: São propriedades relacionadas à organização da classe. Por exemplo: no UdI de automóveis, o volante é parte do carro.
- Relacionamentos com outras classes: São propriedades que exibem os relacionamentos com outras classes. Por exemplo: fabricante de vinhos tem uma relação entre o vinho e a vinicultura.

F. Definir as Facetas das Propriedades das Classes

As propriedades das classes possuem facetas, ou seja, particularidades que definem, de forma mais específica, as propriedades, ou "*slots*". Como exemplos de facetas temos a definição do tipo de valor da propriedade, valores permitidos para propriedade, número de valores permitidos (cardinalidade). Abaixo listaremos facetas mais comuns citadas pela autora:

- Cardinalidade

A cardinalidade permite definir a quantidade de valores que o "*slot*" pode possuir. Podendo ser expressa por valores definidos, ou através dos termos cardinalidade singular ou cardinalidade múltipla, onde não especificamos exatamente a quantidade de valores do "*slot*", apenas informamos que pode ter apenas 1 valor, ou múltiplos valores.

- Tipo de valor

É o tipo de valor que o "*slot*" pode ter. Abaixo alguns destes valores:

- *String*: Utilizado nos "*slots*" do tipo texto. Exemplo: "*slot*" nome (classe Vinho).
- *Number*: Utilizado nos "*slots*" do tipo numérico. Exemplo: "*slot*" preço (classe Vinho).
- *Boolean*: Utilizado nos "*slots*" do tipo "*flag*", onde o tipo de valor deve ser "verdadeiro" ou "falso". Exemplo: "*slot*" gasoso (classe Vinho).
- *Enumerated*: Utilizado nos "*slots*" que permitem ter uma lista de valores. Exemplo: "*slot*" Sabor (classe Vinho), que tem como lista de valores, forte, moderado e delicado.
- *Instance*: Utilizado nos "*slots*" que representam relacionamento com outras classes. Exemplo: "*slot*" produzido (classe Vinho) representando o relacionamento com a classe "Produtor de Vinho".

G. Criar Instâncias

Este é o último passo do método, onde são criadas as instâncias das classes definidas nos passos anteriores.

- **Processo de Semi-Automático para Construção de Ontologia [Breitman 04]**

A Figura 8 é um quadro com a descrição passo-a-passo do método, proposta por [Breitman 04] para construção de uma ontologia de UdI. Este método tem como ponto de partida o léxico do UdI [Leite 00], pois é a partir deste artefato que o método é executado, ou seja, o léxico é a entrada deste método.

O léxico, formalmente chamado LAL (Léxico Ampliado da Linguagem), é o conjunto de símbolos usados no contexto do UdI que são descritos na sua forma denotativa e conotativa, ou seja, cada conceito do contexto é representado por um símbolo, que é descrito de forma a retratar dois aspectos: a noção (denotação) e o impacto (conotação). A noção é o

significado do símbolo, enquanto o impacto é a influência do símbolo no contexto [Felicíssimo 04]. Os símbolos do léxico também são classificados em 4 categorias: sujeito, objeto, verbo e estado. [Leite 99]

As descrições dos símbolos seguem o princípio de circularidade, que prega a utilização dos outros símbolos do léxico na descrição dos símbolos e do vocabulário mínimo, que diz que as partes das descrições, que não são outros símbolos do léxico, devem pertencer a um subconjunto reduzido de palavras com significado bem definido. [Leite 99]

1. Listar os termos alfabeticamente de acordo com seu tipo (verbo, objeto, sujeito ou estado)
2. Fazer 3 listas: conceito (classe) (C), propriedade (R) e axiomas (AO). Na lista de classes cada entrada terá um nome, descrição (linguagem natural) e uma lista contendo zero ou mais rel (função que relaciona o conceito em questão a outros, de maneira não taxonômica). As entradas na lista de axiomas terão nomes (labels) somente.
3. Utilizando a lista de símbolos do léxico classificados como sujeito ou objeto, para cada termo:
    3.1Adicione uma nova classe a lista de classes. O nome da classe é o símbolo do léxico propriamente dito. A descrição da classe é a noção do termo.
        3.1.1Para cada impacto,
            3.1.1.1Checar se já faz parte da lista de propriedades da ontologia.
            3.1.1.2Caso não faça parte da lista (a propriedade ainda não existe), adicione uma nova propriedade na lista (de propriedades). O nome da propriedade deve ser baseado no verbo utilizado para descrever o impacto
                3.1.1.2.1Verificar consistência.
            3.1.1.3Na lista de classes adicione uma nova rel para a classe em questão. A rel é formada pela classe + a propriedade (definida em 3.1.1.1) + a classe relacionada (esta classe é o objeto direto/indireto do verbo utilizado no impacto do símbolo doléxico. Usualmente é um termo do próprio léxico e aparecesublinhado).
            3.1.1.4Checar se existem indicativos de negação no vocabulário mínimo que relacionem duas ou mais classes. Verificar se estas classes possuem um relacionamento do tipo disjuntas (exemplo macho e fêmea).
                3.1.1.4.1Se verdadeiro, adicionar o disjoint a lista de axiomas.
    3.2Verificar consistência.
4.Utilizando a lista de símbolos classificados como tipo verbo, para cada termo:
    4.1.Checar se já faz parte da lista de propriedades da ontologia.
        4.1.1.Caso não faça parte da lista (a propriedade não existe), adicione uma nova propriedade na lista (de propriedades). O nome da propriedade é o símbolo do léxico propriamente dito.
            4.1.1.1.1Verificar consistência.
5. Utilizando a lista de símbolos classificados como tipo estado, para cada termo:
    5.1.Para cada impacto
        5.1.1Tentar identificar a importância relativa do termo para a ontologia. Esta estratégia é similar a utilizaçaoo de questões de competência proposta em [Gruninger95]. Estas questões são obtidas através do refraseamento dos impactos de cada símbolo em perguntas inciadas por quando, onde, o quê, quem, porque, e como.
        5.1.2Checar se existem indicativos de negação no vocabulário mínimo que relacionem duas ou mais classes. Verificar se estas classes possuem um relacionamento do tipo disjunto (exemplomacho e fêmea)
            5.1.2.1Se verdadeiro, adicionar o disjoint a lista de axiomas.
        5.1.3 Caso o termo seja central a ontologia, classifique-o como classe (C)).
        5.1.4Caso contrário (o termo não é central para a ontologia) classifique-o como propriedade (R).
        5.1.5Verificar consistência.
6. Quando todos os termos tiverem sido adicionados à ontologia,
    6.1Checar se existe conjuntos de conceitos que compartilham rel idênticos
        6.1.1Para cada conjunto de conceito que compartilha rel, construir uma lista de conceitos separados
        6.1.2Buscar na ontologia conceitos que fazem referência a todos os membros desta lista
            6.1.2.1Se não forem encontrados, busca na noção e no impacto de cada membro da lista de conceitos tentando identificar um termo comum do vocabulário mínimo
        6.1.3Construir uma hierarquia de conceitos onde todos os membros da lista de conceitos é um sub-conceito do conceito encontrado em
        6.1.4Verificar consistência

Figura 8 - Processo Semi-automático para construção de ontologia. [Breitman 04]

Existem, na literatura, outros métodos utilizados para construção de ontologias. Abaixo apresentamos um quadro que mostra algumas de suas principais características segundo [Perez 04]:

Caracteristicas	Cyc	Uschold & King	Gruninger & Fox	Kactus	Methontology	Sensus	On-To-Knowledge
Ciclo de Vida	Desenvolvimento de Prototipos	Não proposta	Desenvolvimento de Prototipo ou	Desenvolvimento de Prototipos	Desenvolvimento de Prototipos	Não proposta	Incremental e ciclica com desenvolviment
Estrategia de identificação de conceitos	Não especificada	Middle-out	Middle-out	Top-down	Middle-out	Não especificada	Top-Down Botton-up Middle-out
Ferramentas de apoio	Cyc Tools	Não especificada	Não especificada	Não especificada	ODE WedODE OntoEdit Protege-2000	Não especificada	OntoEdit com o plugins
Aceitação por organizações externas	Não conhecida	Não conhecida	Não conhecida	Não conhecida	FIPA (www.fipa.org)	Não conhecida	VU Amsterdam,CognIT,Aidministrator,SwissLife
Ontologias criadas	Cyc Tools	Enterprise Ontology	TOVE	Electrical network ontologies	Chemicals OntoRoadMap Esperonto ontologies	Militar air campaing	Não citada

Tabela 1 - Características dos métodos para construção de ontologia [adaptado de Perez 04].

Analisando os métodos, podemos notar no primeiro método muita subjetividade nas descrições de suas etapas. Encontramos, na descrição, muito mais a explicitação do meta-modelo utilizado (no caso o *frame ontology*), do que o passo-a-passo que deve ser executado para construção da ontologia. Já o segundo método é bem mais específico na descrição do seu passo-a-passo, possibilitando uma execução bem mais objetiva do mesmo. E também conta com a vantagem de ser baseada num processo maduro para a construção de léxicos (entrada do método), que já foi validado em projetos reais [Breitman 05].

3.3. Meta-modelos de Ontologia

O meta-modelo é necessário, na construção da ontologia, porque auxilia a especificação das informações que devem estar presentes no modelo de ontologia. Seu objetivo é unificar a semântica utilizada na representação da ontologia [Perez 04]. Abaixo apresentamos alguns destes meta-modelos:

- ***Frame Ontology* [Gruber 93]:**

Este meta-modelo é constituído de uma coleção de convenções comuns para organização do conhecimento. Na descrição do método "*A Simple Knowledge-Engineering Methodology*" [Noy 01] já foram descritos os itens que compõem este meta-modelo. Abaixo apresentamos um resumo dos principais componentes deste meta-modelo segundo [Perez 04]:

- Classes: Uma coleção de indivíduos.
- Indivíduos ou Instâncias: Uma instância de uma classe.
- "*Slots*" ou Propriedades: São as características das classes
- Facetas: É a especificação dos "*slots*". Exemplo: Tipo da classe (*String*, Numérico).

As principais taxonomias utilizadas para organizar as classes e instâncias são:

- Subclasse - of: É o relacionamento entre uma classe "filho" que herda as características da classe "pai".
- Superclasse - of: É o relacionamento entre uma classe "pai" que provê características para a classe "filho".
- *Disjoint*: É o relacionamento entre classes que são complemente opostas, ou seja, não possuem nada em comum.
- *Exhaustive*: São classes que não podem ser instanciadas.
- *Partition*: É o relacionamento "Parte de", onde uma classe é parte de outra classe.
- *Instance- of*: É a instanciação de uma classe.

- **OIL *Ontology***

Este meta-modelo tem como principais componentes [Perez 04]:

- **Classes**, que são agrupadas em 6 categorias, descritas abaixo. Este agrupamento de classes tem a finalidade de auxiliar e oferecer mais semântica na definição das classes, visto que as novas classes são definidas por uma, ou um conjunto, destas classes primitivas [Bechhofer01]:

- Classes para definir tipos concretos de expressões: São subclasses da classe *oil: ConcreteTypeExpression* . Estas classes definem expressões numéricas para os números, como, por exemplo: igual, maior que.
- Classes para definir expressões de classes: São subclasses da classe *oil: ClassExpression*. Estas classes podem ser formadas por expressões boleanas, restrições de propriedades e expressões enumeradas. As expressões boleanas são as expressões de conjunção, disjunção e negação entre as classes. As restrições de propriedades qualificam numericamente as restrições através das cardinalidades. As expressões enumeradas representam as classes que podem ter diversos valores.
- Classes para definir características matemáticas: São subclasses das classes *oil: TransititiveProperty, oil: FunctionalProperty e oil: SymmetricProperty*. Estas classes definem expressões matemáticas de transitividade, funcional e simetria.
- Classes para definir axiomas: São as subclasses da classe *oil: axiom*. Estas classes representam a taxonomia de definição de *disjoint* e *exhaustive* para as classes.
- Classes para definir tipos de dados: São subclasses de *oil:string e oil:integer.* São classes para definir os tipos de dados *string* e *integer.*
- Classes Pré-definidas: São subclasses de *oil:Top e oil: Bottom* . A classe *oil:Top* é a mais genérica das classes que são de um mesmo tipo. A *oil:Bottom* é uma classe vazia, e é do tipo de qualquer classe.

- **Propriedades ou Relacionamentos entre Classes:**

  - *subClasseOf* : relaciona uma classe “filha” com a classe “pai”.
  - *hasOperand*: relaciona uma expressão Booleana com os operandos (*And*, *Or* e *Not*).
  - *individual*: representam instâncias das classes.

- *toClass*: utilizada para representar que o *range* de uma classe é uma outra.
- *toConcreteType*: utilizadas para representar classe concretas.
- *individualFiller, integerFiller* e *stringFiller*: utilizadas para representar valores de propriedades nas classes.
- *Number*: expressa cardinalidade.
- *inverseRelationOf*: expressa propriedades inversas entre classes.
- *hasObject, hasSubject e isCoveredBy*: representa as propriedades entre classe de *disjoint* e *exhautive*.

- **OWL *Ontology***

As informações que constituem o meta-modelo OWL *Ontology,* e proporcionam representação semântica no modelo ontológico, foram agrupadas em três conjuntos, formando três meta-modelos derivados que possuem níveis diferentes para representação semântica [Horridge 04]:

- O OWL *Lite*, que possui um conjunto menor de informações;
- o OWL DL, que possui um conjunto intermediário de informações, entre o OWL *Lite* e o OWL *Full*;
- e, finalmente, o OWL *Full*, que possui o conjunto completo de informações deste meta-modelo.

Este divisão proporciona flexibilidade para selecionar o nível de semântica que melhor se adapte às necessidades do usuário no contexto do UdI que será modelado.

Abaixo apresentamos os principais componentes deste meta-modelo (os componentes apresentados estão presentes nos 3 meta-modelos do OWL *Ontology*) [Perez 04]:

- **Classes,** que são agrupadas em 7 categorias. Esta categorização é utilizada na definição de novas classes, assim como no meta-modelo OIL *Ontology*:

- Classes para definir Classes e Restrições: São subclasses de *owl:Class e owl:Restriction.* São utilizadas para definir classes e as restrições das propriedades das classes.
- Classes para definir propriedades: São subclasses de *owl:ObjectProperty,owl:DatatypeProperty,owl:TransitiveProperty,owl:SymmetricProperty,owl:FuncitionalProperty, owl:InverseFunctionalProperty e owl:AnnotationProperty*. São utilizadas para definir propriedades que conectam classes a outras classes (*owl: ObjectProperty*), classes a tipos de dados (*owl:DatatypeProperty*). As classes *owl:TransitiveProperty,owl:SymmetricProperty* servem para definir características lógicas às propriedades. Já as *owl:FuncitionalProperty,owl:InverseFunctionalProperty* são utilizadas definir restrições de cardinalidade.
- Classes para definir diferenças entre instâncias: São subclasses de *owl:AllDifferent*. São utilizadas para definir que duas instâncias são diferentes. Pois, em owl, duas instâncias com identificadores diferentes podem se referir a uma mesma instância.
- Classes para definir tipos de dados enumerados: São subclasses de *owl:dataRange*. São utilizadas para criar tipos de dados enumerados, que têm um conjunto de valores pré-definidos.
- Classes Pré-definidas: São subclasses de *owl:Thing e owl: Nothing*. Representam a mais e a menos genérica classe, respectivamente.
- Classes para descrever ontologias: São subclasses de owl:*Ontology*. É utilizada como classe raiz da OWL *ontology* contendo todas as suas definições.
- Classes para descrever o versionamento da ontologia: São subclasses de *owl:DeprecatedClass e owl:DeprecatedProperty*. São utilizadas com propósito de versionamento de ontologias. São classes ou propriedades que estão depreciadas em relação à versão corrente da ontologia.

- **Propriedades ou Relacionamentos entre Classes:**

Propriedades	Descrição	Meta modelo
owl:intersectionOf	Propriedade que define expressões de classe que representa conjunção.	OWL Lite
owl:unionOf	Propriedade que define expressões de classe que representa disjunção.	OWL DL
owl:complementOf	Propriedade que define expressões de classe que representa negação.	OWL DL
owl:oneOf	Propriedade que define expressões de classe que representa coleção de indivíduos.	OWL DL
owl:onProperty	Propriedade que define expressões de classe para definir restrições de propriedade.	OWL Lite
owl:allValuesFrom	Propriedade que define expressões de classe para definir valores de restrições.	OWL Lite
owl:hasValue	Propriedade que define expressões de classe para definir facetas.	OWL DL
owl:someValuesFrom	Propriedade que define expressões de classe para definir restrições existenciais.	OWL Lite
owl:mincardinality	Propriedade que define expressões de classe para definir restrições numéricas.	OWL Lite
owl:maxcardinality	Propriedade que define expressões de classe para definir restrições numéricas.	OWL Lite
owl:inverseOf	Define o inverso da propriedade.	OWL DL
owl:sameAs	Define equivalência entre recursos.	OWL DL
owl:equivalentClass	Define equivalência entre classes.	OWL DL
owl:equivalentProperty	Define equivalência entre propriedades.	OWL DL

Propriedades	Descrição	Meta modelo
owl:sameIndividualAs	Define equivalência entre instâncias	OWL DL
owl:differentFrom	Define duas instâncias como diferentes.	OWL DL
owl:disjointWith	Representa o *disjoint* entre classes.	OWL DL
owl:distinctMembers	Utilizado para definir a lista de instâncias que são diferentes de outras.	OWL DL
owl:versionInfo	Propriedade que fornece informações da versão corrente da ontologia.	OWL DL
owl:priorVersion	Propriedade que define qual versão corrente da ontologia teve uma versão anterior.	OWL DL
owl:incompatibleWith	Propriedade que define que a ontologia é incompatível com outra ontologia.	OWL DL
owl:backwardCompatibleWiyh	Propriedade que define que a ontologia é compatível com outra ontologia.	OWL DL
owl:imports	Propriedade que referencia outras ontologias Owl que foram importadas nesta ontologia.	OWL DL

Tabela 2 – Lista de propriedades da OWL *Ontology* [retirado de Perez 04].

- É importante salientar que as propriedades utilizadas pelo OWL *Lite*, também poderão ser utilizadas pelo OWL DL e OWL *Full*, e as propriedades utilizadas pelo OWL DL são utilizadas tanto pelo mesmo, quanto pelo OWL *Full* [Perez 04].

Analisando os meta-modelos, percebemos que as diferenças entre eles é na composição das informações que a ontologia deve possuir, ou seja, na quantidade de representações semânticas que disponibiliza para o UdI.

A seleção de um meta-modelo deve, segundo [MacGuinness 04], depender do critério de melhor adaptação as suas necessidades, ou seja, selecionar um meta-modelo mais completo, que possibilite uma gama de representações semânticas, não significa estar escolhendo o melhor meta-modelo.

Acreditamos que, para este trabalho, o "*frame ontology*" é o mais adequado por conseguir fornecer, com seu conjunto de informações, um bom conhecimento do UdI com maior facilidade para criação dos modelos. O "*frame ontology*" possui classes e propriedades mais simples que os outros meta-modelos. Isto facilita o desenvolvimento do modelo por torná-lo menos complexo, mesmo acarretando perda de representação semântica. Mesmo com isso, este meta-modelo nos fornece um bom conhecimento de UdI, ou melhor, o conhecimento considerado suficiente para esta primeira versão do método, onde se busca mostrar a capacidade do método em atingir seus objetivos de maneira mais simplificada. Contudo, não descartamos a possibilidade de evolução deste para contemplar meta-modelos que disponibilizem maior representação semântica do UdI.

3.4. Linguagens

A linguagem é utilizada para unificar a formas de representação da ontologia, de forma que o conhecimento da ontologia possa ser interpretado de maneira única [Perez 04]. Abaixo são apresentadas algumas linguagens de ontologias:

- **OIL**

  A linguagem OIL (*ontology inference layer*) foi patrocinada através de um consórcio da Comunidade Européia, através do projeto *On-to-Knowledge*. Esta linguagem foi criada pela necessidade de uma linguagem expressiva que permitisse semântica necessária, e formalismo suficiente, de modo a permitir suporte a mecanismos de inferência. Os serviços de inferência oferecidos incluem detecção de inconsistências e a determinação de relacionamentos do tipo sub-classe de. [Breitman 04]

  Atualmente, os editores para a linguagem OIL permitem tradução para outras linguagens com o RDF, por exemplo, mesmo que na tradução ocorra perda de expressividade, como é o caso do editor OILEd. [Breitman 04]

- **OWL**

Esta linguagem foi criada pelo consórcio W3C, e foi projetada de modo a realizar uma especificação formal do UdI. [Breitman 04]

OWL possui três linguagens, assim como o meta-modelo OWL *ontology*:

- OWL *Lite*

OWL *Lite* suporta a criação de hierarquias de classificação dos termos, bem como algumas restrições mais simples, como cardinalidades. Não possuem axiomas nem estruturas de relacionamentos sofisticadas. [Breitman 04]

- OWL – DL

OWL-DL é mais expressivo que o OWL *Lite*. É baseado em *Description Logics* e, por isso, possibilita classificação automática de hierarquia e checagem de consistência na ontologia em conformidade com o respectivo meta-modelo. [Horridge 04]

- OWL *Full*

OWL *Full* é a mais expressiva das sub-linguagens de OWL. Deve ser utilizada em situações onde a expressividade semântica é mais importante do que garantir a completeza do meta-modelo, pois não é possível verificar consistência da ontologias representadas em OWL *Full*. [Horridge 04]

A maior parte das ferramentas para edição de ontologias já oferece suporte (ou pelo menos tradução) para a linguagem OWL. Este é o caso do OILEd e do Protege-2000. [Breitman 04]

Baseando-se no fato da maioria das ferramentas para edição de ontologias oferecerem tradução para diversas das linguagens de ontologias, não selecionaremos especificamente nenhuma linguagem para ser utilizada neste trabalho. O importante é a seleção do meta-modelo, pois com o modelo desenvolvido, a tradução para linguagens distintas torna-se trivial ao utilizar as ferramentas.

3.5. Ferramentas

Neste item mostraremos as ferramentas que apóiam a construção de ontologias, com algumas de suas características. Abaixo são exibidas algumas destas ferramentas:

- **Protege – 2000**

  É uma ferramenta "*open source*", "*stand-alone*", de distribuição gratuita com arquitetura extensível, que permite a transformação da ontologia construída para outras linguagens como, XML, OWL, entre outras. O Protege permite a construção de ontologias de UdI, entradas de dados para criação de instâncias, bem como permite que outros aplicativos acessem sua base de dados. [Perez 04] [Breitman 04]

  Atualmente, existem duas versões da ferramenta: *Frames* e OWL[Protege 00], que são versões para adaptar-se aos respectivos meta modelos.

- **Onto Edit**

  É uma ferramenta com um ambiente extensível e flexível através da utilização de *plug-ins*, possibilitando transformar a ontologia desenvolvida para outras linguagens como RDF, XML, entre outras. Esta ferramenta está disponível em duas versões: uma *free* e uma profissional, com funcionalidades diferenciadas. [Perez 04] [Breitman 04]

- **OilEd**

  Esta ferramenta foi desenvolvida em 2001 como um editor para OIL ontologias. Este editor não tem nenhuma metodologia de desenvolvimento associada e também não oferece suporte à integração, nem versionamento. Porém, é de fácil utilização pelo usuário, permitido a escrita e a checagem da ontologia (através do verificador FaCT), ou seja, checar se a ontologia desenvolvida está de acordo com o meta-modelo. Esta ferramenta é *free* e, para obtê-la, basta registrar-se no *site* e realizar o *download*. [Perez 04] [Breitman 04]

Para fins de utilização no nosso trabalho, selecionamos a ferramenta Protege em função de ser umas das ferramentas mais referenciadas na literatura, além de ter distribuição gratuita e possibilitar a tradução do modelo desenvolvido para algumas das principais linguagens de ontologias.

3.6. Conclusão

Neste capitulo apresentamos as principais definições e objetivos de ontologia, bem como os principais elementos que auxiliam para o desenvolvimento da mesma. São eles: os métodos, guiando a construção de ontologias de UdI, os meta-modelos, definindo as principais informações que a ontologia deve conter para modelar um UdI mais completo e as linguagens e ferramentas, auxiliando para melhor compreensão e facilitação de construção, respectivamente.

Podemos notar a importância da ontologia quando vemos a crescente utilização das mesmas como forma de estabelecer uma linguagem comum para compartilhar e reutilizar o conhecimento sobre o UdI do negócio, um dos objetivos da ontologia.[Cota 04]

Nos últimos dois capítulos, apresentamos as duas fontes de conhecimento (modelagem de processo de negócio e ontologia) que são utilizadas pelo método que será proposto no capitulo seguinte. Este método visa integrar o conhecimento obtido pelo modelo de processo do negócio e da ontologia do UdI.