UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC

O Processo de Construção de Ontologias Baseado na Modelagem UML.

Giovanni Won Dias B. Victorette

Florianópolis – SC

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

O Processo de Construção de Ontologias Baseado na Modelagem UML.

Giovanni Won Dias B. Victorette

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

FLORIANÓPOLIS – SC

Giovanni Won Dias B. Victorette

O Processo de Construção de Ontologias Baseado na Modelagem UML.

Trabalho de conclusão de curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação

_____

Prof. José Leomar Todesco, Dr.
Universidade Federal de Santa Catarina
Orientador

Banca Examinadora

_____

Prof. Fernando A. Ostuni Gauthier, Dr.
Universidade Federal de Santa Catarina
co-orientador

_____

Ricardo Haus Guembarovski, M. Eng.
Centrais Elétricas de Santa Catarina

RESUMO

Com o avanço da aplicação de ontologias nas diversas áreas de conhecimento, verifica-se uma necessidade de evolução e aprimoramento de técnicas e ferramentas para construção de ontologias. Este trabalho propõe uma abordagem para o desenvolvimento de ontologias com enfoque em processos da engenharia de software. Tal iniciativa pode propiciar uma aproximação entre a comunidade de Engenharia de Ontologias e Engenharia de Software. A proposta utiliza-se da linguagem UML, dentro dos conceitos estabelecidos da arquitetura dirigida a modelos (MDA), focando a parte de análise e projeto na construção de ontologias. Com isto obteve-se um guia, o UML2ONTO, no qual detalha as fases do processo de construção de ontologias com a orientação da engenharia de software.

Palavras-chaves: ontologia, UML, OWL

ABSTRACT

With the ontology progress in the various fields of knowledge, there is a need for systematic unfolding and improvement of techniques and tools for ontology development. This paper presents an approach to the ontology development focusing on software engineering processes. This initiative can provide an approach between the Ontology Engineering and Software Engineering communities. The proposal uses the Unified Modeling Language (UML), within the established concepts of the Model Driven Architecture (MDA), focusing the analysis and project in ontology development. Furthermore, a guide has been achieved, the UML2ONTO, which details the stages of the construction of ontologies with the guidance of software engineering.

Keywords: Ontologies, UML, OWL

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGENS

LISTAS DE ABREVIATURA

IA – Inteligência Artificial

OWL – Ontology Web Language

XML – Extensible Markup Language

RDF – Resource Description Language

OIL - Ontology Inference Layer

DAML - Darpa Agent Markup Language

W3C - World Wide Web Consortium

MOF – Meta Object Facility

UML – Unified Modeling Language

XMI – XML Metadata Interchange

XSLT – Extensible Stylesheet Language Transformations

Sumário

1. Introdução

Com o advento da web semântica e avanço da aplicação de ontologias, verifica-se uma necessidade de evolução e aprimoramento de técnicas e ferramentas existentes para construção e desenvolvimento das mesmas. Ferramentas de metodologia que guiam uma elaboração padronizada ainda estão amadurecendo. Em conseqüência, os métodos de desenvolvimento, validação, verificação, e mesmo de documentação de ontologias continuam evoluindo.

Existe uma crescente preocupação, por parte das empresas de distribuição de energia elétrica no Brasil, com a evolução dos serviços prestados. Motivado pelo aumento de competitividade, juntamente com as resoluções do órgão regulador da energia elétrica no Brasil (ANEEL). Neste contexto é exigida a melhoria da qualidade do fornecimento e conseqüente evolução dos sistemas e processos de distribuição de eletricidade. Assim sendo torna-se evidente a necessidade de desenvolver métodos mais eficientes de tomada de decisão e análises gerenciais para o cumprimento de metas contratuais das empresas do setor.

Dessa forma, pretende-se propor uma abordagem técnica baseada em tarefas para o desenvolvimento de uma ontologia de domínio utilizando-se para tal a modelagem UML e técnicas de engenharia de software, obtendo-se também a formalização e a representação do conhecimento sobre a distribuição de energia elétrica de média tensão para fins de implementações de sistemas de conhecimento que apóiem a gestão da distribuição da rede de média tensão.

1.1 Motivação

Uma das motivações deste trabalho é desenvolver um guia de passos para se construir uma ontologia, de forma que possa ser útil para diminuir a lacuna existente entre a comunidade de engenharia de software e a comunidade de engenharia de ontologias. Em segundo plano, documentar o conhecimento de engenheiros e de especialistas a respeito da rede de média tensão na forma de uma base de conhecimento, de maneira que seja possível caracterizar a rede e propor ações, evitando a evasão do conhecimento com o desligamento de funcionários especializados da empresa e facilitando o treinamento e a aprendizagem de novos colaboradores.

1.2 Objetivos

1.2.1. Objetivo geral

Propor uma abordagem baseada em tarefas para o processo de construção de ontologias baseado na modelagem UML.

1.2.2. Objetivos específicos

Dentre os objetivos específicos podemos citar:

- Estudar e formalizar novos procedimentos acerca das tarefas de construção de ontologias baseados em um meta-modelo.
- Definir uma ontologia de domínio para a formalização e representação do conhecimento no que se refere à distribuição da rede de energia de média tensão;

- Construir um guia para organizar as tarefas referentes à construção de uma ontologia.

1.3. Delimitação do escopo

A proposta não se caracteriza como uma metodologia e não apresentará foco na conversão de formatos de arquivos utilizado entre as ferramentas estudadas.

Para o estudo de caso será utilizada apenas para o domínio da rede elétrica de média tensão.

1.4. Justificativa

Muitos trabalhos na área de tecnologia vêm explorando o uso de ontologias e metodologias para construção das mesmas. Segundo Noy (2001), não há uma forma correta de modelar um domínio, ou seja, não existe uma única maneira de se construir uma ontologia.

A construção e manipulação de ontologias têm sido sistematizadas por metodologias. Baseado nos estudos de Corcho, Fernández-López e Gómez-Pérez (2001), apresentam-se várias metodologias para a construção de ontologias. Como todas essas metodologias apresentam aspectos positivos e negativos, muitos dos desenvolvedores acabam utilizando essas metodologias apenas como base para o desenvolvimento de metodologias próprias. Dentre as metodologias mais conhecidas estão a metodologia 101 [Noy, McGuinness , 2001], On-To-Knowledge [Fensel, 2002] e o Methontology [Fernández-López, 1997].

O gerenciamento de redes de distribuição de energia, por parte dos especialistas, é feito com o tratamento de uma grande diversidade de variáveis, muitas das quais são variáveis lingüísticas, mais adequadas ao tratamento de sistemas fortemente influenciados por experiência, julgamento, percepção e raciocínio humano. Para se obter um tratamento adequado de tais sistemas deve levar em consideração os fatores humanos, os quais são vagos, mal definidos, ambíguos e imprecisos. Segundo Guembarovski (2004), existe uma grande dificuldade de se aplicar um modelo matemático para representar esse sistema e que seja capaz de descrever e capturar todas as peculiaridades do mesmo. Os problemas são resolvidos pelos especialistas, uma vez que estes possuem uma experiência acumulada do tratamento dos fatores que influenciam o problema. Esses conhecimentos são ditos tácitos, decorrem de experiências e geralmente não apresentam formalismos.

Em face da crescente competitividade e dos critérios de qualidade e de formulação de preço impostos pela agência de regulação do setor (ANEEL), as empresas de distribuição de energia elétrica no Brasil são forçadas a buscarem alternativas para a melhoria do processo de gestão dos circuitos de alta e baixa tensão [Guembarovski 2004].

Associado a este cenário é identificada a necessidade de conservar conhecimentos (know-how) já adquiridos, documentado o conhecimento de especialistas em redes de distribuição de média tensão na forma de uma base de conhecimento. O maior esforço a ser apresentado é a representação do conhecimento utilizando um meta-modelo, fruto da integração do estilo camadas, a UML e visões, possibilitando desta forma um maior nível de abstração e uma melhor documentação.

1.5. Metodologia proposta

As principais etapas para o desenvolvimento do trabalho são:

- Pesquisas de tecnologias e ferramentas para a criação e manipulação de ontologias;
- Pesquisa sobre ontologia, gestão do conhecimento, Meta Modelo e gestão de redes de distribuição;
- Definição da metodologia para a criação da ontologia;
- Utilização das tarefas propostas para a criação da ontologia;
- Demonstração de resultados práticos na geração da ontologia;

1.6. Organização dos capítulos

O presente capítulo fez a introdução do trabalho apresentando o problema que deseja se solucionar, a motivação para a realização e a metodologia a ser utilizada para o desenvolvimento.

No capítulo 2 é feita uma abordagem sobre a engenharia do conhecimento, apresentando definições e técnicas de representação.

O Capítulo 3 apresenta uma pesquisa sobre ontologias, as classificações, as metodologias, as linguagens e ferramenta para edição.

No Capítulo 4 é demonstrado os trabalhos existentes sobre a modelagem UML e as aplicações na construção de ontologias.

O Capítulo 5 apresenta a proposta das tarefas para a construção da ontologia usando UML.

No Capítulo 6 é apresentada a aplicação proposta na rede de distribuição de energia elétrica.

No Capítulo 7 é feita a conclusão do trabalho e considerações sobre trabalhos futuros.

2. A Engenharia do Conhecimento

Para desenvolver bases de conhecimento é necessário adquirir conhecimento humano, seja de peritos ou de outras fontes. Utilizam-se para isto técnicas, linguagens e ferramentas para efetuar a validação do conhecimento formalizado e representado.

É importante compreender que as atividades da engenharia do conhecimento nunca ocorrem em seqüência, de forma linear. Pelo contrário, são todas entrelaçadas e requerem uma rigorosa disciplina. Não é incomum que engenheiros do conhecimento retornem muitas vezes para técnicas e ferramentas alternativas se o que foi escolhido não estiver apropriado.

2.1. Conceitos básicos

Segundo Gasevic (2006), na inteligência artificial, o processo de armazenar conhecimento é guardar o conhecimento em um formato apropriado na memória de um computador. O inverso desta operação é identificado como recuperação do conhecimento, encontrando-o quando necessário. Raciocínio significa usar conhecimento e estratégias para solução de problemas de um

programa inteligente para obter conclusões, inferências e explicações. Um importante pré-requisito para este processo é a aquisição de conhecimento, coletando, organizando e estruturando o conhecimento sobre um tópico, um domínio ou um problema.

Com a redução do ciclo de vida dos produtos e tecnologias se tornando cada vez mais fáceis de se copiar, o conhecimento da organização emerge como a maior fonte de vantagem competitiva. Apesar da Gestão do Conhecimento prometer prover grandes benefícios, estudos têm mostrado que muitas iniciativas são deficientes. Umas das principais razões é que as pessoas não possuem uma compreensão clara sobre a gestão do conhecimento e o conhecimento em si.

O conhecimento é frequentemente a base para a utilização efetiva de muitos recursos importantes [Bloodgood & Salisbury, 2001]. A perspectiva de base de conhecimento postula que serviços rendidos de recursos tangíveis dependem de como eles são combinados e aplicados, o que se torna uma função do conhecimento final da organização.

Como um resultado a Gestão do Conhecimento é observada como um acelerador para realizar a sinergia entre as unidades, conseguindo um alto valor agregado para os clientes, acelerando a inovação, impulsionando a receita e trazendo vantagem competitiva.

2.2. Aquisição e técnicas de representação do conhecimento

O conhecimento é modelado a partir da experiência de um determinado especialista. Companhias que seguem a iniciativa da Gestão do Conhecimento

com uma bem definida e clara visão tendem a terem mais sucesso, considerando que companhias que entram na popularidade do modelo, a fim de colher rápidos benefícios, tendem a falhar [Sheng-Hsun Hsu & Huang-Pin Shen, 2005]. Desta forma pode-se afirmar que se deve ter uma completa compreensão sobre o que o conhecimento e a engenharia do conhecimento são antes de embarcar em uma iniciativa da Gestão do Conhecimento.

A seguir serão apresentadas algumas técnicas de representação. Vale salientar que todos os métodos possuem como limitação a incapacidade de representar informações sub-simbólicas. Assim conclui-se que existe uma parcela de conhecimento que não tem como ser explicitado e, portanto, não pode ser capturado por sistemas processadores de informação.

2.2.1. Frames

Introduzidos por Marvin Minsky (1975), os frames são estruturas compostas de “gavetas” (slots ou escaninhos) capazes de conter um valor. Um grupo associado de gavetas forma um frame. O conteúdo de um slot pode estar associado a instâncias hierarquicamente superiores. Através desse modelo, temos como implementar conceitos como herança, tão importante para as modernas linguagens orientadas a objeto [NAVEGA 2005].

As estruturas de frames associados montam complexas relações entre cada conceito, permitindo que os sistemas baseados em frames possam responder determinadas perguntas. Como exemplo, na figura 1 pode-se perguntar se um Alimentador pode atuar.

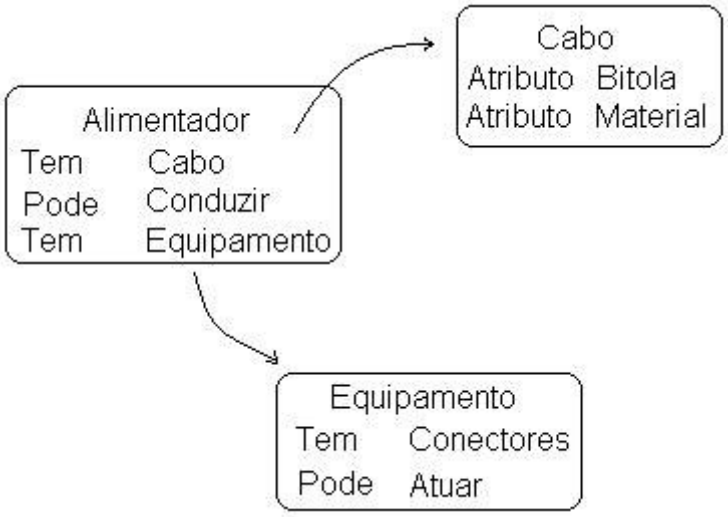

Figura 1 Sistema elétrico representado por frames

2.2.2. Redes Semânticas

As Redes Semânticas (Semantic Networks) são construções que representam conhecimento através de nós conectados por arcos [NAVEGA 2005]. Estas redes são muito antigas e são exploradas por diversas áreas. Assim como os frames a rede semântica também pode responder perguntas. Segundo Navega (2005), essa estrutura parece bastante similar à dos frames. Mas a similaridade termina quando extensões das redes semânticas entram em ação.

Uma dessas extensões são as chamadas redes semânticas proposicionais, onde os nós representam proposições. Os nós podem conter redes aninhadas que especificam mais informações sobre as proposições. Outras extensões são feitas quando os nós representam conceitos ou relações entre conceitos, em uma estrutura conhecida como Grafos Conceituais (Conceptual Graphs, cujo principal teórico é John Sowa). A capacidade expressiva dessas construções é, em muitos casos, superior à conseguida pela lógica de primeira ordem [NAVENGA 2005]. Na figura 2 pode-se observar um pequeno exemplo de

uma rede semântica, também identificada como um Grafo Conceitual. De acordo com a figura pode-se responder a seguinte pergunta: “Um transformador tem isolamento?”.

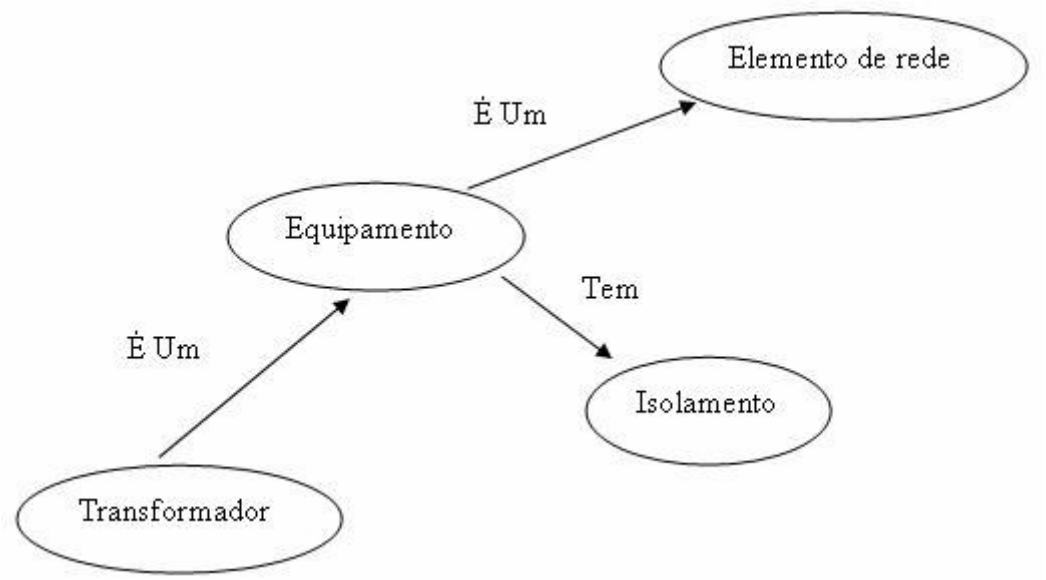

Figura 2 Rede semântica

2.2.3. Regras de produção

Os Sistemas Especialistas tiveram grande destaque na década de 80. Acreditava-se que estavam diante de sistemas realmente inteligentes. Em síntese estes sistemas são constituídos por uma série de regras que analisam informações sobre um assunto específico [Wikipedia, 2007]. Tais regras são conhecidas como Regras de Produção. Nelas, procurava-se representar (explicitar) conhecimentos sobre um domínio. As regras dispunham de antecedentes (condições iniciais) e conseqüentes (condições finais). Na figura 3 pode se observar um exemplo de tais regras.

Conhecimento Explicito	
SE Trecho do Alimentador é separado por chave fusível ENTÃO Trecho é ramal	SE Atuação de religador > 5 e região = rural ENTÃO Efetuar poda e verificar alimentador com termovisor

Figura 3 Codificação de regras de produção

Tipicamente estes sistemas funcionam adicionando em uma memória de trabalho os dados iniciais fornecidos pelo usuário (uma pergunta, por exemplo). Desses dados, o sistema buscava encontrar as regras que poderiam satisfazer as condições iniciais. Tais regras são criadas de acordo com o conhecimento de um especialista quando se analisa um problema.

2.4. Considerações finais

A representação do conhecimento não é algo trivial, o trabalho exige muito, tanto do especialista no domínio quanto do engenheiro do conhecimento. uma das principais razões para tal atividade é permitir o compartilhamento e reuso do conhecimento. Mais adiante será apresentado o conceito de ontologias e sua utilização na engenharia do conhecimento

Segundo Navenga (2005), a representação do conhecimento envolve elaborar uma Conceitualização, ou seja, montar um conjunto abstrato de objetos e outras entidades que são assumidas como existentes nesse domínio específico, além de relações e funções sobre esses elementos. A escolha de um conjunto como esse envolve um compromisso ontológico (ontological commitment), que deve ser aceito por todos os usuários.

3. Ontologias

A palavra ontologia vem do grego, significando o estudo do ser. Em Ciências da Computação e Sistemas de Informação a ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferências sobre os objetos do domínio.

Informalmente, a ontologia de um certo domínio é sua terminologia, todos os conceitos essenciais, sua classificação, sua taxonomia, suas relações e seus axiomas de domínio. Formalmente, para alguém que quer questionar tópicos em um domínio ***D*** usando uma linguagem ***L***, uma ontologia provê um catálogo de tipos de coisas assumindo que existe em ***D***, os tipos na ontologia são representados em termos de conceitos, relacionamentos e predicados de ***L*** [Gasevic, 2006].

De acordo com Gruber, uma ontologia é uma especificação explícita de uma conceitualização, sendo que uma conceitualização é uma visão abstrata do mundo que se deseja representar, consistindo nos objetos, nos conceitos e nas relações entre eles, que existem no mundo representado (GRUBER, 1993).

Segundo Gómez-Pérez (1999) uma ontologia é um conjunto de termos ordenados através de uma hierarquia para fazer a descrição de um domínio, que poderá ser utilizado como um esqueleto por uma base de conhecimento.

3.1. Conceitos básicos

Na IA, o termo “ontologia”, basicamente, vem a significar uma das duas coisas relacionadas [Chandrasekaran et al., 1999]:

- Vocabulário de Representação, muitas vezes especializado em algum domínio;
- Um corpo da descrição do conhecimento de algum domínio em particular usando um vocabulário de representação.

3.2. Classificação das ontologias

Existem várias formas de classificações para as ontologias, sendo que uma das mais populares, por ser citada em diversos trabalhos sobre o assunto é a proposta por Guarino (1998), que utiliza a conceitualização como principal critério (figura 4). Classificando-as como:

- Ontologias de alto-nível
- Ontologias de domínio
- Ontologias de tarefa
- Ontologias de aplicação

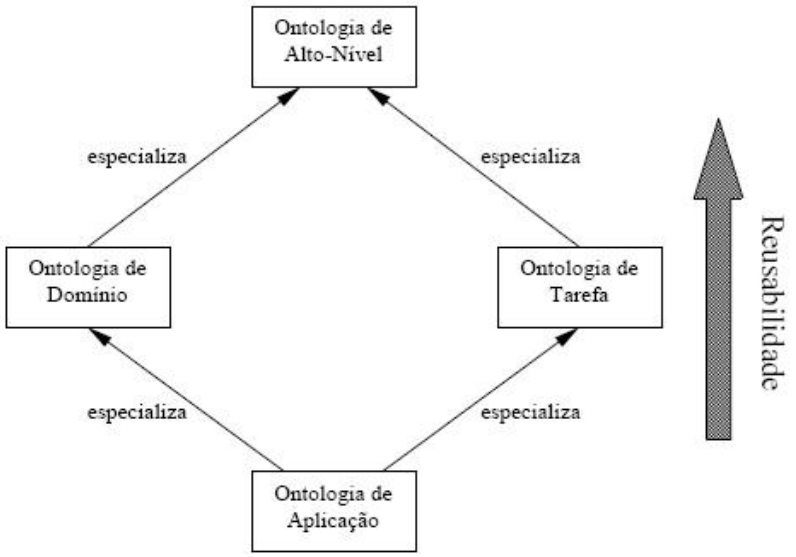

Figura 4 de ontologias e suas relações (GUARINO, 1998).

3.3. Metodologias para construção de ontologias

A construção e manipulação de ontologias têm sido sistematizadas por metodologias. Baseado nos estudos de Corcho, Fernández-López e Gómez-Pérez (2001), apresentam-se várias metodologias para a construção de ontologias. Como todas essas metodologias apresentam aspectos positivos e negativos, muitos dos desenvolvedores acabam utilizando essas metodologias apenas como base para o desenvolvimento de metodologias próprias. Esta iniciativa tem o objetivo de se obter uma metodologia adaptável para diferentes cenários [REZGUI, 2007].

A seguir serão apresentadas algumas metodologias mais difundidas e as quais foram feitos estudos aprofundados para o desenvolvimento do atual trabalho.

3.3.1 Metodologia 101.

Esta metodologia foi desenvolvida por Natalya F. Noy e Deborah L. McGuinness, ambas da Stanford University, nos Estados Unidos, no ano de 2001. Algumas idéias desta metodologia foram baseadas na modelagem orientada a objetos; entretanto, o desenvolvimento de ontologias é diferente do desenvolvimento de classes e relações da modelagem orientadas a objetos.

A programação orientada a objetos foca principalmente nos métodos e classes – um programador toma suas decisões de projeto baseado nas propriedades operacionais de uma classe, enquanto que um projetista de ontologia toma suas decisões baseado nas propriedades estruturais de uma

classe. Como resultado, a estrutura de uma classe e os relacionamentos entre classes em uma ontologia são diferentes da estrutura de um domínio similar em uma modelagem orientada a objetos. (NOY, 2001, p. 2).

Esta metodologia segue três regras principais, que são consideradas em todo o processo de criação da ontologia:

- Não há uma forma correta de modelar um domínio – sempre existem alternativas viáveis. Geralmente a melhor solução dependerá da aplicação que será utilizada;
- Desenvolvimento de ontologia é necessariamente um processo iterativo;
- Conceitos da ontologia devem ser referentes a objetos (físicos ou lógicos) e relacionamentos no domínio de interesse. Provavelmente serão substantivos (objetos) ou verbos (relacionamentos) nas sentenças que descrevem o domínio.

Uma vez que a ontologia foi modelada, é interessante considerá-la como uma versão inicial que, a partir deste estágio, pode ser testada, revisada e melhorada. Este processo iterativo e evolutivo de modelagem provavelmente continuará através de todo o ciclo de vida da ontologia (NOY, 2001, p. 4).

A base da metodologia está representada pela seqüência dos passos conforme figura 5. Deve-se lembrar que pessoas diferentes seguindo os mesmos passos irão gerar ontologias diferentes, uma vez que a criação de uma ontologia trata-se de um processo criativo.

Figura 5 de passos da metodologia 101 (TODESCO, J. L. - 2007)

3.3.2 Metodologia On-To-Knowledge

O projeto On-To-Knowledge (Fensel et al., 2002), (Sure et al., 2002b) tem como visão geral ser uma metodologia para introdução e manutenção de soluções de KM baseadas em ontologias para empresas com foco em:

- Meta Processo do Conhecimento, que consiste em introdução e manutenção de novas soluções de KM em uma organização.(Probst et al.,2001) – identificação do conhecimento.
- Processo do Conhecimento. - manipulação de soluções de KM já definidas (Probst et al., 2001) – criação do conhecimento.

Um ponto muito importante abordado nesta metodologia é a manutenção da especificação de requisitos e das perguntas de competência a cerca do domínio estudado, perguntas as quais auxiliam na identificação de termos e relações.

3.3.3 Methontology

Gómez-Pérez (1999) destaca que para a construção de uma ontologia, cinco tipos de componentes têm que ser levados em conta: conceitos (termos ou classes, e seus domínios de valores), relacionamentos, funções (relações especiais onde o n-ésimo elemento da relação é único para os n-1 elementos precedentes), axiomas (modelam sentenças que são sempre verdadeiras) e instâncias.

Segundo Novello (2002), os relacionamentos mais utilizados na representação de ontologias são a taxonomia (“é um”, “tipo de”), a partonomia (“parte de”), a mereologia (teoria “parte-todo”), a cronológica (precedência entre

os conceitos) e a topologia (teoria de limite e fronteira).

Methontology (Fernández-López et al., 1997) é uma metodologia que define o processo de criação de ontologias desde o seu início, passando pelas etapas de planejamento, especificação, aquisição de conhecimento, conceitualização, formalização, integração, implementação, avaliação, documentação e manutenção (figura 6). É baseada na idéia de prototipação e evolução como a abordagem bem adequada para o ciclo de vida de uma ontologia e dá uma especial ênfase ao reuso das mesmas.

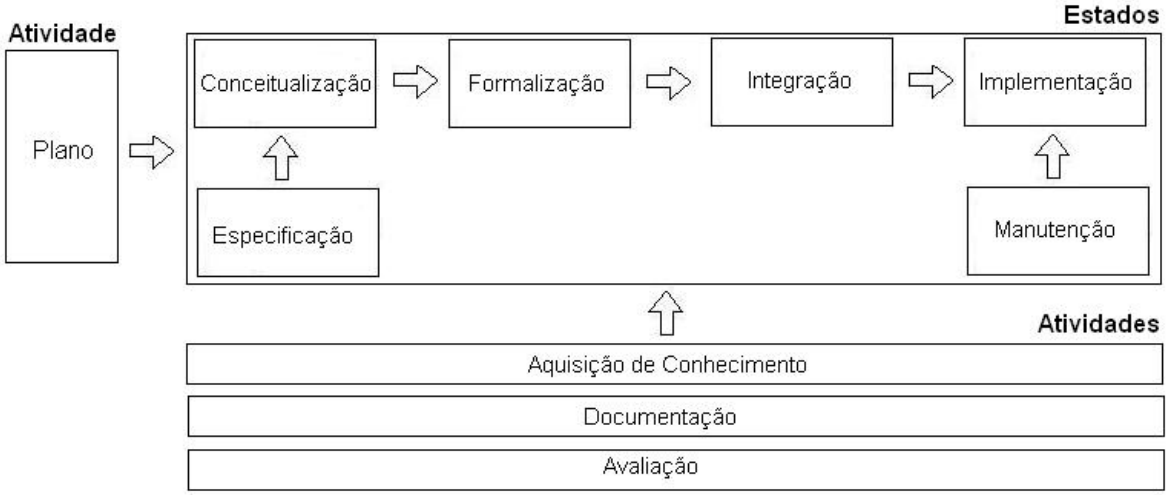

Figura 6 Processo de construção de ontologia segundo METHONTOLOGY

3.4. Linguagens de desenvolvimento

Um grupo padrão de ferramentas para desenvolvimento de uma ontologia inclui a linguagem de representação e o desenvolvimento gráfico da mesma.

Mais recentemente as ferramentas têm aparecido para automatizar este processo de desenvolvimento e que ajudem na evolução, atualização e manutenção das ontologias.

3.4.1. RDF[1]

RDF é uma aplicação (ou dialeto) de XML definido pelo W3C e sua função é fornecer um modelo formal de dados e sintaxe para codificar meta dados que podem ser processados pelos computadores (BREITMAN, 2005, p. 49). Foi proposto pelo W3C no ano de 2000, fornece as primitivas básicas para a criação de ontologias, incluindo relacionamentos de generalização entre as classes.

Esse padrão prevê interoperabilidade entre aplicações que trocam informações processáveis por máquinas na Web. O RDF provê facilidades em permitir o processamento automatizado de recursos da Web (Swick; Lassila, 1999).

Segundo MCFARLANE (2004), três exemplos de categorias de tipos de informação são: conteúdo, dados e fatos; cada uma sendo processada de forma diferente. Informação de conteúdo geralmente é processada como um todo – exibir uma página HTML, reproduzir uma música –, dados são processados em partes, ou blocos: inserir um registro no banco de dados, ordenar uma lista de objetos. Fatos são itens de dados em forma de instruções. Um exemplo de fato é a frase A cerveja está quente. RDF é uma linguagem que permite a representação explícita de fatos.

3.4.2. RDF-Schema[2]

RDF Schema é uma extensão semântica do RDF e seu objetivo é fornecer uma linguagem para descrição de vocabulários a partir do RDF (RDF, 2004).

O RDF-Schema é uma linguagem de descrição de vocabulários que

[1] <http://www.w3.org/RDF/>
[2] <http://www.w3.org/TR/rdf-schema/>

objetiva descrever propriedades e classes para os recursos RDF. O RDF-Schema também fornece primitivas para a organização dos recursos, permitindo a construção de hierarquias de classes e de propriedades (BREITMAN, 2005)

Um ponto fraco no desenvolvimento de ontologias utilizando-se RDFS[3] é a falta de recursos para a modelagem de axiomas e descrição da semântica referente aos conceitos e relações (BECKER, 2005).

3.4.3. DAML+OIL

Segundo GAŠEVIC (2006), DAML+OIL surgiu de outras duas linguagens, DAML-ONT [Hendler & McGuinness, 200] e OIL [Fensel et al., 2001], as quais foram fortemente influenciadas pelo RDF. No contexto da Web Semântica (figura 7), RDF define um modelo simples para representação semântica e primitivas básicas para modelagem de ontologias, enquanto que DAML-ONT, OIL, e DAML+OIL buscavam prover um vocabulário mais expressivo para o desenvolvimento da ontologia. Estes oferecem caminhos ricos para definir conceitos e atributos, e uma escolha mais intuitiva de algumas das primitivas de modelagem.

[3] A utilização de RDF e RDF *Schema* é usualmente chamada de RDFS.

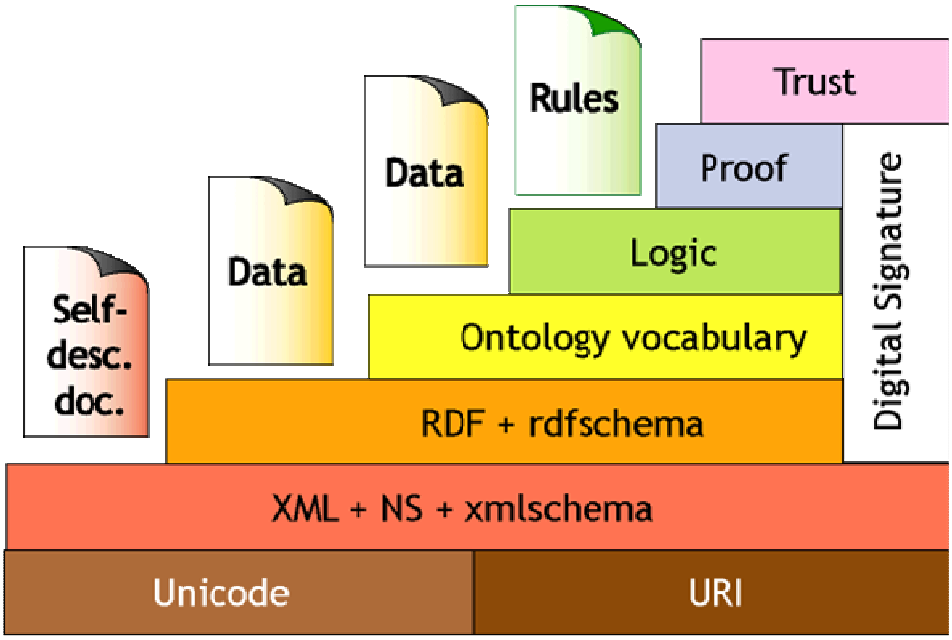

Figura 7. Visão da Web Semântica - Tim Berners-Lee

DAML+OIL [Horrocks & van Harmelen, 2002; Scott Cost et al., 2002] mesclou os princípios e vantagens de DAML e OIL, em um esforço de desenvolver uma linguagem universal para a Web Semântica que poderia possibilitar máquinas a lerem dados interpreta-los e realizar inferências sobre os mesmos. Como um resultado e também provendo regras e definições similares às do RDF DAML+OIL também possibilita maior relacionamento entre os recursos especificados, incluindo cardinalidade, domínio, restrições, uniões, disjunções e regras transitivas e inversas.

3.4.4. OWL[4]

O vocabulário OWL inclui um conjunto de elementos e atributos XML com significados bem definidos. Estes são utilizados para descrever termos de domínio e seus relacionamentos em uma ontologia. Segundo (W3C) o OWL é uma revisão do DAML+OIL, incorporando o conhecimento adquirido com o

[4] http://www.w3.org/2004/OWL/

desenvolvimento e aplicação do DAML+OWL.

A linguagem OWL (listagem 1) foi desenvolvida com base nas necessidades da Web Semântica. Foi desenvolvida pelo consórcio W3C e foi projetada para poder fazer uma especificação formal da Ontologia.

Existem três tipos de sublinguagens – ou níveis – da OWL, variando de forma crescente conforme o grau de complexidade e expressividade:

- OWL Lite: suporta ontologias simplificadas que precisam apenas de uma classificação hierárquica e restrições simples, um exemplo de restrição possível é a cardinalidade. Ela não possui estrutura de relacionamentos sofisticados e axiomas. É voltada para a migração de tesauros e outras taxonomias.
- OWL – DL: apresenta maior expressividade e poder computacional do que a OWL Lite, é baseada em lógica de descrição, que permite estabelecer a classificação automática de hierarquia além da checagem de consistência da ontologia de acordo com o respectivo metamodelo. A abreviatura “DL” é referente à Lógica de Descrição (Description Logic), responsável pela base formal da OWL.
- OWL Full: possui expressividade máxima aliada à liberdade da sintaxe do RDF, mas sem garantias computacionais. usada quando a expressividade semântica é mais importante do que o metamodelo, uma vez que, através dela não é possível fazer a verificação da consistência das ontologias. OWL Full permite a uma ontologia expandir o significado do vocabulário pré-definido (RDF ou OWL).

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://owl.protege.stanford.edu#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:UML2="org.omg.xmi.namespace.UML2"
xml:base="http://owl.protege.stanford.edu">
    <owl:Class rdf:ID="Elemento">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty>
                    <owl:DatatypeProperty rdf:about="#temCodigo"/>
                </owl:onProperty>
                <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="temCodigo">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#Elemento"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="Equipamento_Protecao">
        <rdfs:subClassOf>
            <owl:Class rdf:about="#Elemento"/>
        </rdfs:subClassOf>
    </owl:Class>
</rdf:RDF>
```

Listagem 1 – Documento OWL.

3.5. Protégé

O Protégé é uma ferramenta open source de livre distribuição utilizada para a edição de ontologias. Possui uma arquitetura de software complexa, porém, facilmente extensível através de plug-ins. Uma ontologia no Protégé consiste em classes, (Slots) propriedades, (Facets) restrições e axiomas. Classes são conceitos no domínio em questão. Slots descrevem as propriedades de atributos de classes. Facets descrevem propriedades de slots. Axiomas especificam restrições adicionais [Protégé, 2007].

A ferramenta possui um bom nível de maturidade e muitas facilidades para o refinamento de ontologias. A maior complexidade é entender o que é uma

ontologias e como construí-la, tendo isto bem definido a ferramenta será simples de utilizar.

3.5.1. Classes e Instâncias

Classes no Protégé constituem uma taxonomia hierárquica. Se uma classe A é subclasse de uma classe B então cada instância de A também é uma instância de B.

A raiz da hierarquia de classes no Protégé é a classe THING. No Protégé, ambos indivíduos e classes podem ser instâncias de classes. A disposição das classes é feita na forma de árvore que permite uma rápida assimilação do ambiente (Figura 8).

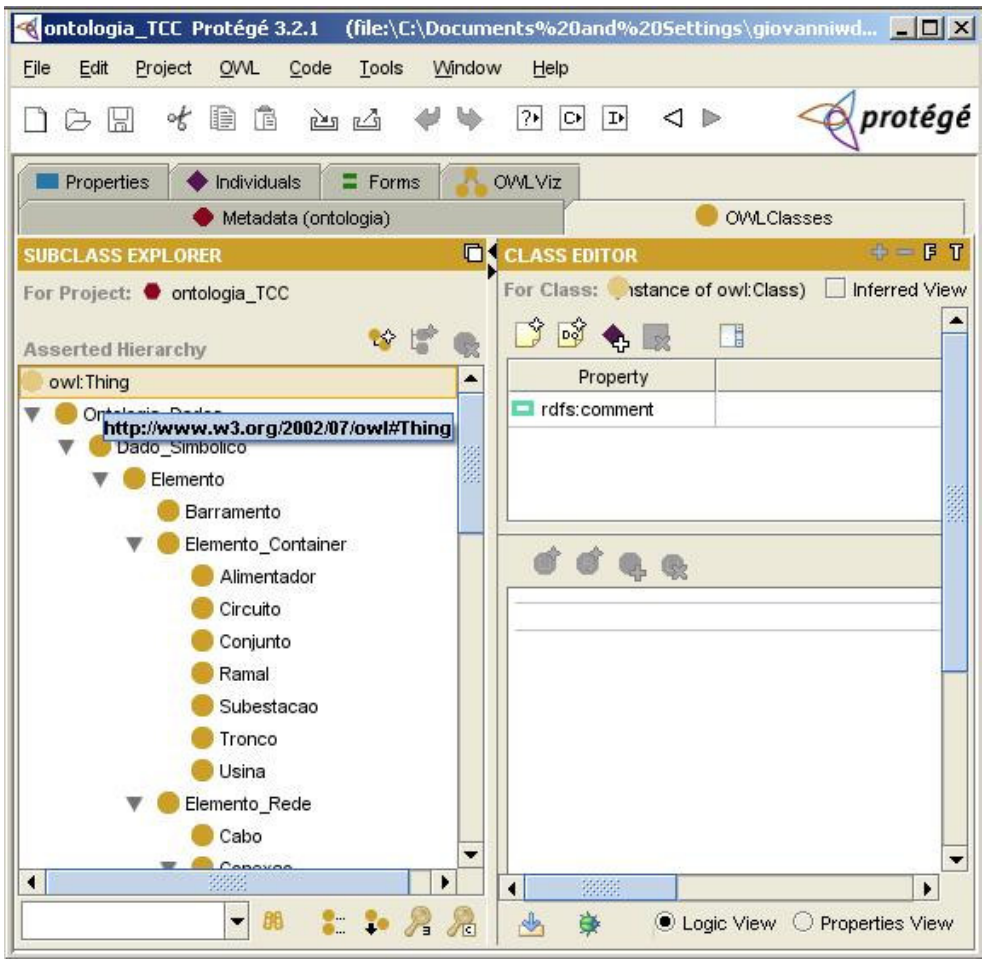

Figura 8. Hierarquia de Classes no Protégé

3.5.2. Propriedades (Slots)

Slots no Protégé-2007 descrevem propriedades de classes e instâncias. Propriedades podem ser definidas independentemente de qualquer classe (Figura 9). Uma propriedade pode ser atribuída a mais de uma classe e também é herdada por outras classes.

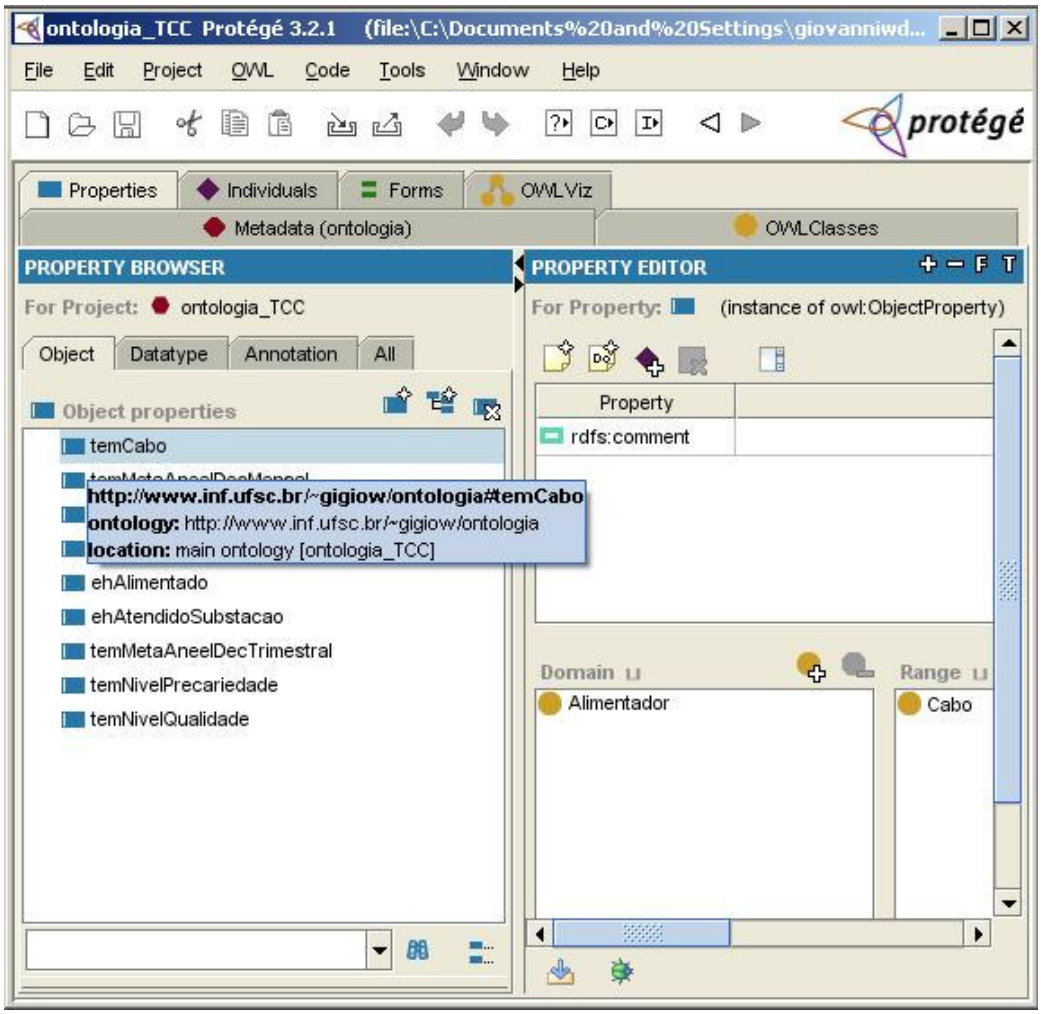

Figura 9. Propriedades de Classes

3.5.3. Restrições de Propriedade (facets)

Uma maneira de especificar restrições em propriedades de dados é através de facets. As restrições de uma propriedade incluem cardinalidade, tipo de dado (int, string...), valor mínimo e máximo para uma propriedade numérica.

3.6. Considerações Finais

Neste capítulo foram abordadas as principais metodologias em uso, procurando apresentar as características mais relevantes para a construção de ontologias. A arte a ciência de construir ontologias estão se desenvolvendo a um ponto que não é mais suficiente apenas modelar e implementar uma ontologia. Faz-se necessário seguir o processo de desenvolvimento avaliando a sua qualidade [SOON AE CHUN, GELLER, J., 2007]. Também foi feita uma apresentação sobre as linguagens de especificação de ontologias, que hoje é definitivamente o OWL.

E por final foram apresentadas algumas características da principal ferramenta de construção de ontologias existe. O Protégé possui um desenvolvimento bem maduro e possui muitas funcionalidades, principalmente no refinamento de uma ontologia.

4. UML e Ontologias

Muitos trabalhos na área de tecnologia vêm explorando o uso de ontologias e metodologias para construção das mesmas, uma vez que não existe uma única maneira de se construir uma ontologia (NOY, 2001). Neste trabalho serão utilizados procedimentos de algumas metodologias e também será proposta uma seqüência de tarefas para a construção da ontologia. Entretanto a realidade é que em UML, OWL e a utilização de metamodelos sofrem com os problemas de semântica incluindo a semântica nas relações de inclusão [AHMAD, M. N., COLOMB, R. M., SADIQ, S., 2008].

4.1. Arquitetura Orientada a Modelos

A relevante iniciativa da comunidade da engenharia de software chamada de *Model Driven Development* (MDD) está sendo desenvolvida em paralelo com a Web Semântica [Mellor et al., 2003ª]. A aproximação da engenharia de software sugere que inicialmente deveria se desenvolver um modelo do sistema em estudo, assim seria transformado em algo real. A iniciativa de pesquisa mais importante nesta área é a Arquitetura Orientada a Modelos (MDA - Model-Driven Architecture), que está sendo desenvolvida sobre o guarda chuva da Object Management Group (OMG).

4.1.1. Modelos e Meta-modelos

Modelos representam a maior regra em MDA. Uma definição geral diz que um modelo é uma visão simplificada da realidade [Selic, 2003] ou, mais formalmente, um modelo é um conjunto de definições de um sistema em estudo [Seidewitz, 2003].

A MDA ou Arquitetura Orientada a Modelos, propõe a criação de modelos em diferentes níveis de abstração separando os interesses de implementação da arquitetura a ser implementada. A MDA utiliza o modelo conceitual de uma aplicação para geração de outro em uma plataforma específica, reduzindo drasticamente o número de implementações de sistemas para o mesmo domínio de aplicação. O objetivo deste trabalho é propor a aplicação do uso da ontologia na construção de um modelo MDA, devido à insuficiência formal com a combinação do uso da UML e MOF (Meta Object Facility). MOF permite a definição formal de linguagens de modelagens por meio de metamodelos. O

metamodelo que descreve UML é uma instância de MOF.

Segundo Freitas (2003), a UML não pode ser considerada um formalismo de representação, devido à ausência de declaratividade, de um motor de inferência e de uma semântica formal, ou seja, não se pode medir a satisfatibilidade dos requisitos funcionais sobre um domínio conceitual. Durante a fase de elicitação de requisitos, os modelos de classes são genéricos, abstraindo boa parte da especificação de um sistema.

Metamodelo é outro conceito chave utilizado em MDA. Um metamodelo é um modelo de especificação para uma classe do sistema em estudo, onde cada classe é um modelo válido expressado em uma certa linguagem. De fato, um metamodelo é um modelo de uma linguagem de modelagem [Seidewitz, 2003]. O Diagrama UML na figura 10 representa a relação entre um sistema e um modelo representado em uma linguagem específica.

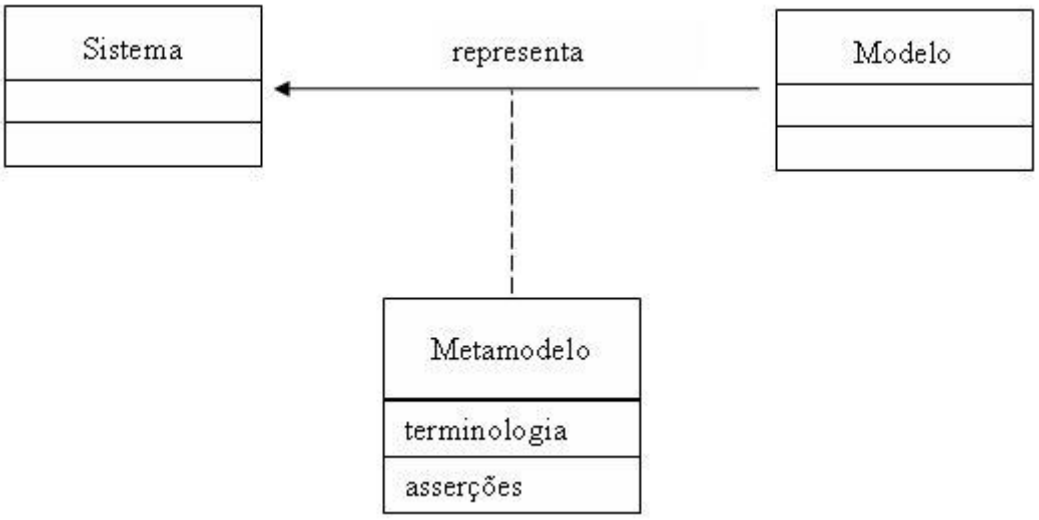

Figura 10 A correspondência entre um modelo e um sistema (Gasevic , 2006, P. 110).

4.1.2. Arquitetura

A MDA é baseada em uma arquitetura de quatro camadas. A figura 11 mostra um exemplo do uso de metaníveis em MDA, umaConta (MO) é um

exemplo da classe “conta” (M1) que implementou a classe “Classe” (M2), por sua vez, sendo uma instância da classe “Classe” em MOF (M3).

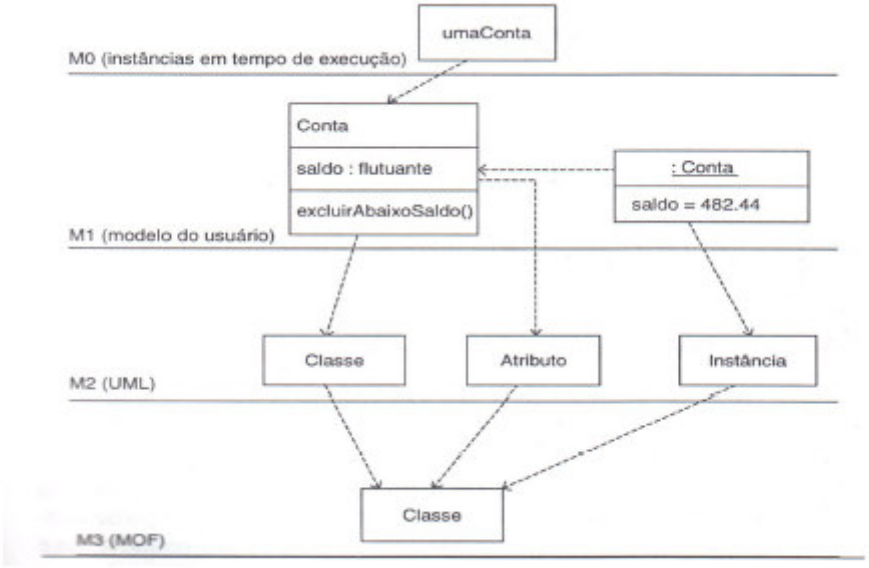

Figura 11 Exemplo de níveis de modelos MDA (Mellor, 2003, P. 48)

4.1.3. Adaptação da UML

Cada metamodelo proposto deve ter no mínimo uma implementação onde sua realização pode ter um ou mais modelos. Um modelo de diagramas UML, por exemplo, é capturado por um metamodelo UML, que descreve como modelos UML podem ser estruturados, os elementos que eles contêm e as propriedades de uma plataforma particular. Os modelos poderão relacionar-se usando mapeamentos, sendo estes feitos automaticamente ou manualmente [Silva, Sampaio, 2006].

Em um mapeamento existem regras que são chamadas regras de mapeamento, estas regras definem basicamente a aceitação de um ou mais modelos que são as origens e um modelo de destino que será gerado. Estas regras são escritas em nível de meta-modelo e são aplicáveis a todos os modelos de origem que obedecem às especificações de seu meta-modelo. Para que seja

possível utilizar mapeamentos em modelos UML é necessário importar o modelo em uma ferramenta MDA através do arquivo XMI (XML Metadata Interchange) que é um artefato gerado a partir das linguagens XML (Extensible Markup Language), UML e MOF que utiliza DTD (Definição de Tipo de Documento), onde são descritas regras de sintaxe dos elementos usados no modelo MDA. A declaração de um DTD pode ser feita interna ou externamente ao documento onde são definidas e ordenadas através de tags [Pezzin, 2006].

4.1.4. UML Profile

Mesmo a linguagem UML tendo sida projetada para atender qualquer tecnologia, a OMG criou um mecanismo para possibilitar o uso da UML de acordo com necessidades específicas. Tal mecanismo é chamado de Profile.

UML Profile é um conceito utilizado para adaptar os construtores básicos de UML para algum propósito específico. Essencialmente, isto significa introduzir novos tipos de elementos estendendo ou adicionando estes para a ferramenta [Gasevic, 2006].

Profiles foram introduzidos na UML 1.3 como uma maneira de estender estereótipos (stereotypes) desta linguagem de modelagem. Na UML 2.0 esta idéia evoluiu utilizando um meta-modelo, que é utilizado para definir a UML, sendo possível adicionar novas regras ao meta-modelo de forma a estender a UML [Kleppe, Warmer, Bast, 2003].

4.2. Trabalhos Relacionados

Um trabalho que tem sido desenvolvido com a utilização de UML para

geração de ontologias é apresentado no livro do GAŠEVIC, D., DJURIC, D., e DEVEDŽIC (Model Driven Architecture and Ontology Development). O principal foco é a compreensão de metamodelos e a utilização de UML profiles para adaptar a UML para modelar ontologias. Este trabalho é uma das principais fontes para a elaboração da proposta de construção de um guia de passos para construir ontologias a partir da modelagem UML.

No livro foram exploradas algumas ferramentas de modelagem UML, mas a maior contribuição utilizada para o atual trabalho é a utilização do arquivo para a conversão de XMI para OWL, no qual o arquivo com extensão XSLT (eXtensible Stylesheet Language Transformation) é o responsável pelo mapeamento.

4.3. Considerações finais

UML possui muitas vantagens para a construção de ontologias, considerando a maturidade do processo e as facilidades existentes para se modelar um domínio. Este capítulo abordou a facilidade da Arquitetura Orientada a Modelos, existente na UML, para se adaptar à modelagem de ontologias. Tal adaptação é possibilitada pela utilização de UML profiles.

Para finalizar o capítulo elaborou-se uma pesquisas sobre trabalhos relacionados e as suas principais contribuições para a área em estudo, que foram de grande valor para o desenvolvimento da proposta aqui estabelecida.

5. Proposta de construção de Ontologias usando UML

Tendo caracterizado um cenário, onde é crescente a preocupação com a engenharia de ontologias e identificado a dificuldade da comunidade de engenharia de software na construção de ontologias, criou-se uma proposta

utilizando tarefas para o processo de construção de ontologias baseado na modelagem UML.

O modelo proposto foi denominado UML2ONTO (UML para Ontologias) (figura 12), uma vez que aborda conceitos de análise e projetos de software aplicados à modelagem da ontologia e em um passo final, onde é gerando um arquivo OWL.

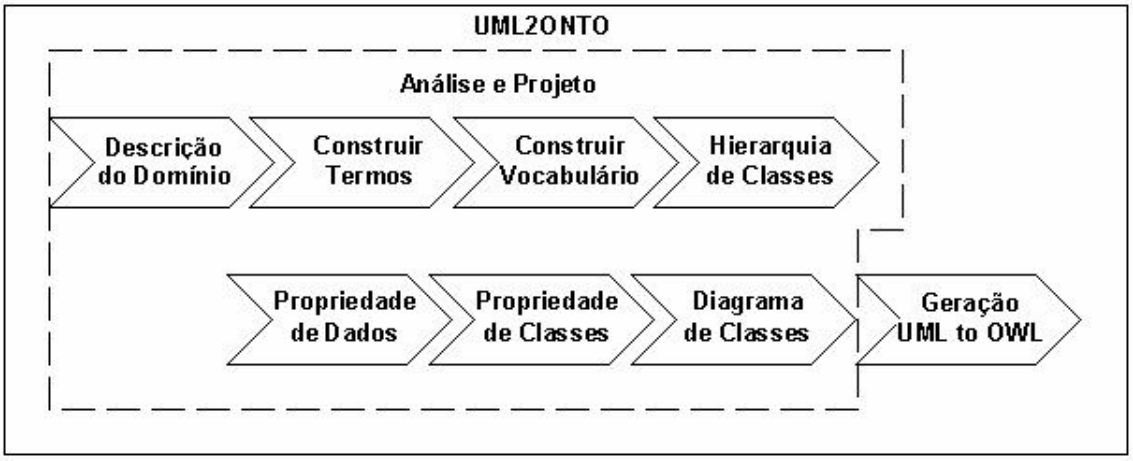

Figura 12. UML2ONTO

No UML2ONTO foi separado um conjunto de tarefas que incorporam uma fase de análise e projeto da construção da ontologia. Este modelo foi concebido tendo como base as fases de análise e projeto na engenharia de software.

5.1. Descrição do Domínio

De acordo com o UML2ONTO a primeira tarefa é definir e descrever o domínio, ou seja, é feita a constituição do cenário. Esta é uma técnica utilizada pela engenharia de software, onde se produz um texto ou um relatório, sobre o domínio estudado, procurando identificar todas as características do domínio.

Esta é uma alternativa apresentada de acordo com a engenharia de software, uma vez que é uma atividade natural para esta área de conhecimento, mas nada impede de se utilizar das perguntas de competência propostas por

algumas metodologias para construção de ontologias, desde que as perguntas representem o domínio estudado.

5.2. Construir Termos

A identificação dos termos é um ponto que merece atenção, mesmo que alguns termos possam ser descartados no futuro. A idéia principal é identificar Termos e Relações no documento gerado no passo anterior. Neste estágio os substantivos encontrados serão candidatos aos termos e os verbos serão candidatos às relações.

Para tal tarefa é recomendado criar uma tabela com os termos candidatos e as respectivas relações identificadas (Tabela 1).

Tabela 1. Termos e Relações

Termos sugeridos	**Relações sugeridas**
Extensao; Alimentador;	temExtensao;
Cabo; Tronco;	temCabo; temTronco;
Ramal; Cabo_predominante;	temRamal; temCaboPredominante;

5.3. Construir Vocabulário

Tendo identificado termos e relações, inicia-se a construção do vocabulário, ou seja, é feita a identificação do que realmente vai ser uma classe. Para isso selecionam-se os termos identificados que são conceitos no domínio. Os erros na definição do que é uma classe da ontologia são bem comuns, para minimizar estas falhas deve-se sempre ter em mente que nem todos os termos serão classes e que uma classe é uma coleção de elementos com propriedades similares.

5.4. Hierarquia de Classes

Com as classes identificadas é feita a etapa de hierarquia de classes. Neste momento é identificada a taxonomia hierárquica definindo-se superclasse e subclasse. A hierarquia geralmente é caracterizada pela hierarquia IS-A (tabela 2), ou seja, no exemplo temos que um Equipamento_Especial é um Equipamento.

Tabela 2. Hierarquia de Classes

Classe	**Subclasse**
Equipamento;	Equipamento_Especial, Equipamento_Protecao;
Equipamento_Especial;	Gerador, Chave, Transformador, Capacitor, Regulador;
Transformador;	Transformador_AT_MT, Transformador_MT_BT;

A hierarquia constitui o passo que mais pode induzir ao erro no desenvolvimento, devido às sutilezas das hierarquias. A hierarquia deve apresentar clareza e consistência ao se criar subclasses. Deve-se tomar o devido cuidado para que uma classe não tenha subclasses demais, é aconselhável verificar a possibilidade de uso de classes intermediárias.

Para facilitar a visualização do modelo proposto, ou seja, possuir uma maior percepção da hierarquia de classes apresentada na tabela 2, segue figura 13 representando a hierarquia já especificada. É recomendado que se utilize do diagrama, principalmente quando existe um grande número de classes no modelo. A sugestão é que esta estrutura já pode ser iniciada na ferramenta de modelagem UML.

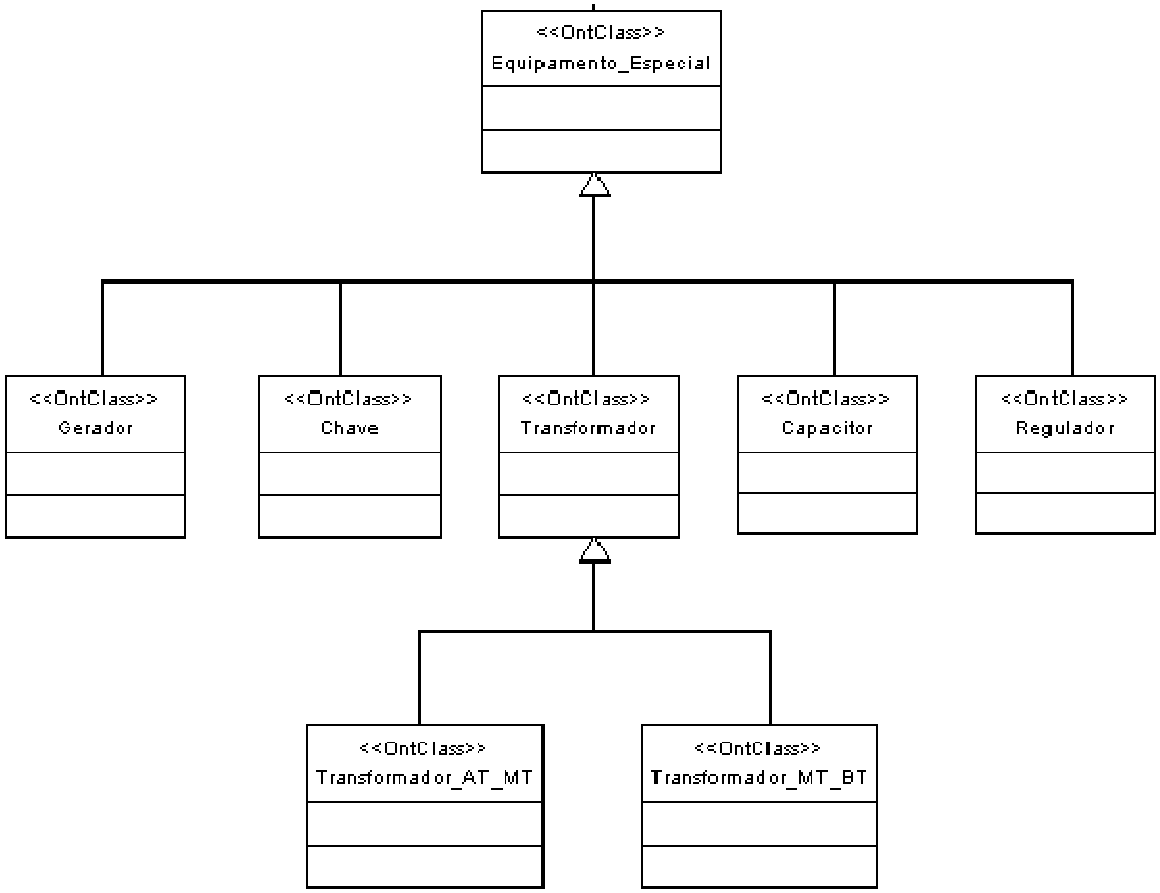

Figura 13. Hierarquia de Classes

5.5. Propriedade de Dados

Atributos de classe ou associações são representados por propriedades. Uma propriedade é uma relação entre um sujeito e um objeto. Na tarefa Propriedade de Dados são identificadas as relações com valores de dados.

Ao se definir uma propriedade, também pode-se estipular a restrição da propriedade que limitam o conjunto de valores para um determinado slot. As restrições podem ser:

- Cardinalidade da propriedade – o número de valores que uma propriedade tem;
- Tipo de valor da propriedade – o tipo de valor que uma propriedade tem;

- Valores mínimo e máximo – um range de valores para uma propriedade numérica.

5.6. Propriedade de Classes

Uma classe sozinha não possui informação suficiente para responder questões que possam ser feitas. Tendo as classes definidas é necessário descrever a estrutura interna dos conceitos (NOY, 2001). Especificar a Propriedade de Classes é definir a propriedade cujo range é uma classe. Para cada propriedade na tabela deve-se determinar qual classe ela descreve, estabelecendo-se assim uma relação entre uma classe domínio e uma classe range (tabela 3).

Tabela 3. Propriedades de Classes

Domínio	**Propriedade**	**Range**
Conjunto	ehAlimentado	Alimentador
Alimentador	temCabo	Cabo
Ramal	temChave	Chave

É interessante observar que propriedades são similares aos conceitos de atributos e associações existentes na Orientação a Objetos. Uma diferença importante é que uma propriedade não depende de alguma classe como os atributos e associações são na UML.

5.7. Diagrama de Classes

Caso já algo já tenha sido representado na ferramenta de modelagem UML, conforme mencionado no passo de definição da hierarquia de classes, já se tem uma estrutura para efetuar um refinamento da ontologia. Se tal tarefa ainda

não foi iniciada, então, na ferramenta UML cria-se o diagrama de classes, procurando representar no primeiro momento a hierarquia previamente definida aplicando os devidos estereótipos. Em um segundo momento representa-se as respectivas relações, sejam elas propriedades de dados ou propriedades de classes (figura 14).

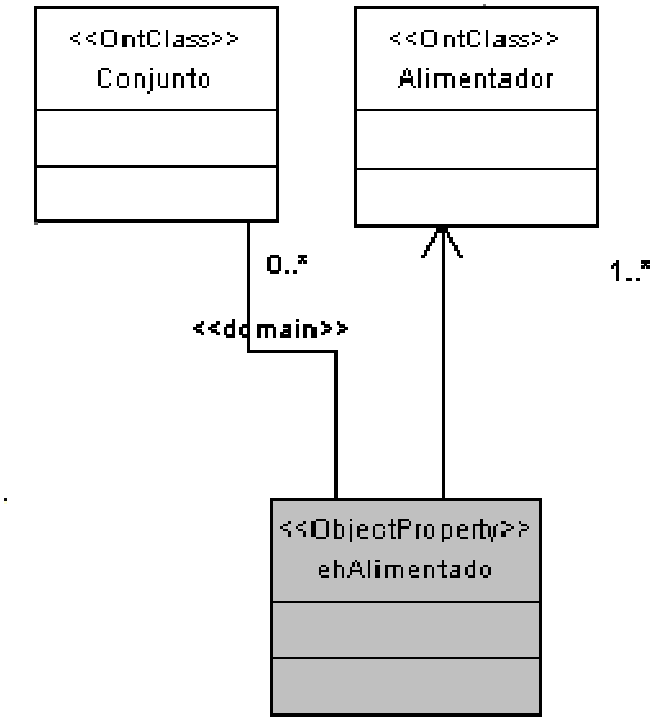

Figura 14. Relação entre Classes

No diagrama, as associações são utilizadas para identificar as propriedades. Para cada associação deve-se indicar o estereotipo correspondente, seja este domain ou range. Também é possível especificar a multiplicidade na associação.

5.8. Geração UML to OWL

Na parte final é feita a exportação do modelo para o formato XMI, que é

um formato de intercâmbio baseado no padrão XML que permite compartilhamento de metadados [OMG XMI, 2002] e utilizando um processador XSLT (eXtensible Stylesheet Language Transformation), que faz o mapeamento do arquivo XMI para um arquivo OWL.

A ferramenta utilizada para a conversão (UMLtoOWL, 2006) foi implementada utilizando XSLT. Com tal ferramenta não existe necessidade de efetuar modificações em uma ferramenta UML. Com uma ferramenta UML que exporte um documento XMI que o processador XSLT (Xalan Java 2, 2007) possa usar como entrada um documento OWL, tal documento pode ser importado por uma ferramenta especializada na construção de ontologias (Protégé), onde pode ser refinado posteriormente.

6. Aplicação da proposta na Rede de Distribuição de Energia Elétrica – Estudo de caso

O Sistema de Distribuição consiste na entrega da energia, em sua forma apropriada de consumo, para o consumidor final. Esta energia deve fluir através dos elementos interconectados na rede. Este é o caminho entre a produção de energia e o consumidor. Estes sistemas de transmissão e distribuição constituem-se de centenas de linhas, subestações, transformadores e outros equipamentos espalhados e interconectados sobre uma grande área geográfica.

O transporte de energia é realizado por diferentes segmentos da rede elétrica que são definidos com base na função que exercem:

- **Transmissão**: redes que interligam a geração aos centros de carga.
- **Interconexão**: interligação entre sistemas independentes.
- **Subtransmissão**: rede para casos onde a distribuição não se conecta a

transmissão, havendo estágio intermediário de repartição da energia entre várias regiões.

• **Distribuição**: rede que interliga a transmissão (ou subtransmissão) aos pontos de consumo sendo subdividida em distribuição primária (nível de média tensão – MT) ou distribuição secundária (nível de uso residencial).

As tensões usuais de transmissão adotadas no Brasil (ANEEL, 2007), em corrente alternada, podem variar de 138 kV até 765 kV incluindo neste intervalo as tensões de 230 kV, 345 kV, 440 kV e 500 kV. Os sistemas ditos de subtransmissão contam com níveis mais baixos de tensão, tais como 34,5 kV, 69 kV ou 88 kV e 138 kV e alimentam subestações de distribuição, cujos alimentadores primários de saída operam usualmente em níveis de 13,8 kV e 23 kV. Junto aos pequenos consumidores existe uma outra redução do nível de tensão para valores entre 110 V e 440 V, na qual operam os alimentadores secundários.

6.1. Subestações de Distribuição

Subestações são conjuntos de equipamentos que comutam, mudam ou regulam a tensão elétrica. Funciona como ponto de controle e transferência em um sistema de transmissão elétrica, direcionando e controlando o fluxo energético, transformando os níveis de tensão e funcionando como pontos de entrega para consumidores industriais [Wikipedia 2007].

As linhas de transmissão e de subtransmissão convergem para as estações de distribuição, onde a tensão é abaixada, usualmente para o nível de 13,8 kV ou 23kV. Destas subestações originam-se alguns alimentadores que se interligam aos transformadores de distribuição da concessionária ou a

consumidores em tensão primária. Os alimentadores primários aéreos operam normalmente de maneira radial e com formação arborescente atendendo aos pontos de carga.

Quando suprida por uma única linha, disporá, na sua alta tensão, de apenas um dispositivo para proteção do transformador. Sua confiabilidade é muito baixa, ocorrendo, para qualquer defeito na subtransmissão, a perda do suprimento da SE. Aumenta-se a confiabilidade dotando-se a SE com uma das duas chaves de entrada aberta ligada a um circuito reserva.

Em regiões com alta densidade de carga utiliza-se um maior número de transformadores [PABLA, 2005], tendo assim um arranjo da SE com maior confiabilidade e maior flexibilidade operacional. Evidentemente cada um dos transformadores deve ter capacidade, na condição de contingência, para suprir toda a demanda da SE. Fato que não ocorre com muita freqüência na prática, uma vez que o custo deste arranjo e expansão do sistema é muito alto. Desta forma visando economia, acabam sacrificando um ponto importante de confiabilidade.

6.2. Alimentadores

As redes de distribuição primária, ou de média tensão, emergem das SEs de distribuição e operam, no caso da rede aérea, radialmente, com possibilidade de transferência de blocos de carga entre circuitos para o atendimento da operação em condições de contingência, devido à manutenção corretiva ou preventiva [KAGAN 2005].

Um alimentador é um circuito elétrico conectado em uma única fonte, ou seja, a subestação. Este opera na tensão de distribuição primária entre 11 e

25kV. Juntos os alimentadores saem de uma SE servindo todas as cargas e cobrindo toda uma área alocada para uma determinada subestação. Para uma empresa de distribuição, uma boa gerência sobre a rede torna-se primordial para manter a confiabilidade do serviço.

A maioria dos sistemas de distribuição são caracterizados por um sistema radial. Este sistema é identificado pelo fluxo de potência em um único sentido. Qualquer interrupção no caminho resulta em uma completa perda na ponta, mas mesmo assim é a mais utilizada por ser a mais barata, simples e fácil de planejar[KAGAN 2005].

Alimentadores são planejados para iniciarem na SE com sua porção principal denominada tronco, caracterizado principalmente pelo condutor de maior bitola. A classificação do mesmo depende da natureza da carga, concentração de carga, taxa de crescimento, qualidade e continuidade de serviço.

Ao se percorrer uma rede, no sentido fonte carga, observa-se um padrão de projeto no que define estruturas físicas e equipamentos da rede. As principais características são:

- Tensão de isolamento;
- Bitola, número e tipo de condutor;
- Seccionamentos;
- Chaves, com corrente nominal e posições normais de operação;
- Bancos de capacitores;
- Reguladores de tensão;
- Religadores;
- Seccionalizadores;
- Consumidores primários;

- Transformadores de distribuição;

Áreas típicas (residencial, industrial, rural, etc.).

6.3. Equipamentos

As estações transformadoras, ETs, são constituídas por transformadores, que reduzem a tensão primária, ou média tensão, para a distribuição secundária, ou baixa tensão. Contam, usualmente, com pára-raios, para a proteção contra sobretensões, e elos fusíveis para a proteção contra sobrecorrentes, instalados no primário. De seu secundário deriva-se, sem proteção alguma, a rede secundária. Nas redes aéreas utilizam-se, usualmente, transformadores trifásicos, instalados diretamente nos postes [KAGAN 2005].

6.4. Ontologia da Rede de distribuição

A estruturação do processo de análise e projeto na construção de uma ontologia permite um maior controle em cada etapa do desenvolvimento. Tendo o domínio caracterizado e bem definido, que no caso é a parte estrutural das redes de distribuição de energia, foram identificados os termos e relações sugeridas (tabela 2).

Tabela 4. Construção dos termos

Termos sugeridos:	**Relações sugeridas:**
Alimentador	temCodigo, temNome
Extensão	temExtensao
Cabo	temCabo
Tronco	temTronco
Ramal	temRamal
Tensao	temTensao
Transformador	temTransformador, temCodigo
Transformador_AT_MT	temCapacidade, temCodigo
Transformador_BT_MT	temTrafoMonofasico, temTrafoBifasico,

	temTrafoTrifasico, temCodigo
Ponto_Conexao	temPontoConexao
Gerador	temGerador
Usina	temCruzeta, temNome
Regulador	temRegulador, temAtuacao, temCodigo
Banco_capacitor	temBC, temCodigo
Banco_regulador	temFaixaRegulacao, temCodigo
Poste	temPoste
Cruzeta	temCruzeta
Isolador	temNivelIsolamento
Chave_Fusivel	temCaveFusivel, temCodigo
Chave_Seccionadora	temCaveSeccionadora, temCodigo
Disjuntor	temLimiteCorrente, temCodigo
Circuito	temCircuito
Conjunto	ehAlimentado, temNome
Equipamento	temCodigo, temEquipamento
Estrutura	temEstrutura
Rede	temRede
Carga	temCarga, temCarregamento
Trecho	temTrecho
Padrao_Rede	temPadraoRede
Equipamento_Especial	temCodigo, temEquipamento
Equipamento_Medicao	temCodigo, temEquipamento
Equipamento_Protecao	temCodigo, temEquipamento

Aplicando-se o modelo proposto na parte de análise e projeto do UML2ONTO e definindo o vocabulário, obteve-se o diagrama de classes (figura 15). No mesmo diagrama foram acrescentadas as devidas relações identificadas em etapas anteriores. Um ponto positivo, que merece destaque, é que o processo de documentação e a geração de produtos ao final de cada etapa auxiliam bastante a organização e o desenvolvimento do trabalho.

Para a modelagem UML foi utilizada a ferramenta ArgoUML[5]. A escolha foi feita por ser uma ferramenta com suporte a XMI e de livre distribuição. A exportação para o arquivo XMI é bem simples e intuitiva, sendo que nenhum problema foi detectado nesta etapa.

[5] http://argouml.tigris.org/

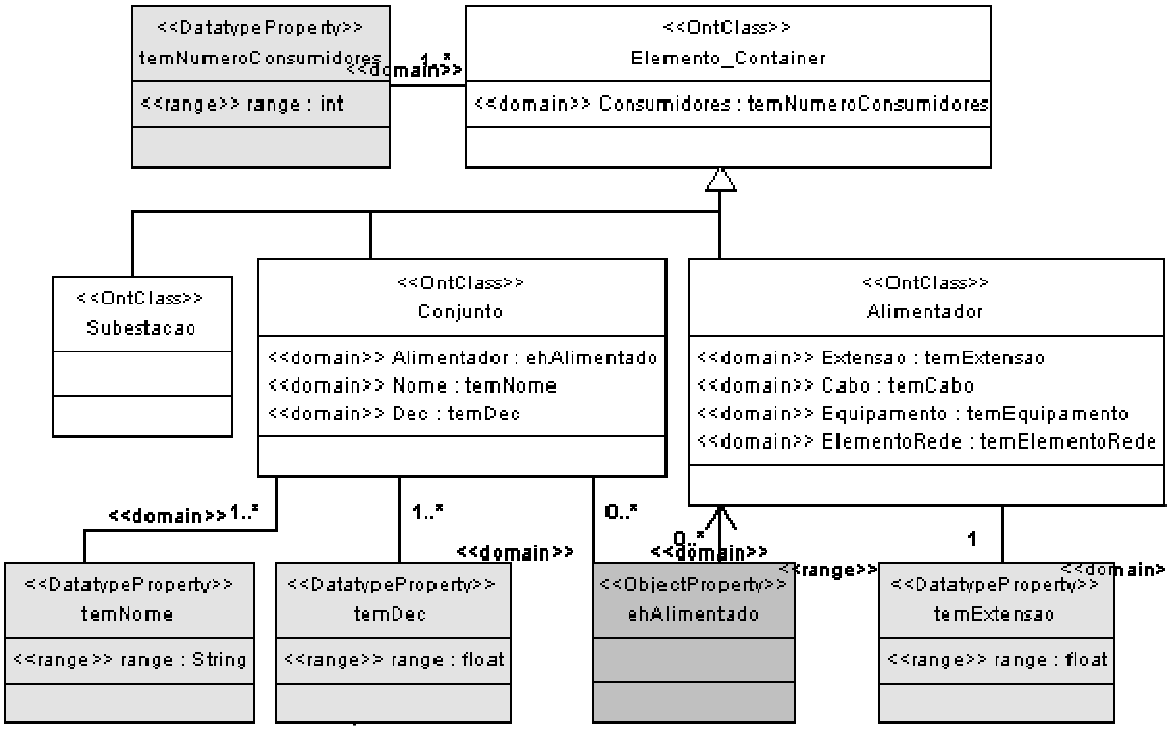

Figura 15. Diagrama de Classes

Para o desenvolvimento do estudo de caso procurou-se explorar várias tecnologias, proprietárias e de livre distribuição, no que diz respeito a ferramentas case, dentre estas a que melhor ofereceu suporte ao desenvolvimento da ontologia foi o ArgoUML. A ferramenta ainda é simples, caso tivesse mais recursos poderia se explorar o refinamento da ontologia criando instâncias e restrições no modelo UML.

Com o modelo de classes definido, utiliza-se a facilidade da ferramenta para exportar o modelo para um arquivo no formato XMI (listagem 2), tal tarefa não apresenta nenhuma restrição. Restrições que foram encontradas em outras ferramentas UML.

```
<?xml version = '1.0' encoding = 'UTF-8' ?>
<XMI xmi.version = '1.2' xmlns:UML = 'org.omg.xmi.namespace.UML' timestamp = 'Thu Apr 10 13:18:02
ACT 2008'>
  <XMI.header>    <XMI.documentation>
     <XMI.exporter>ArgoUML (using Netbeans XMI Writer version 1.0)</XMI.exporter>
     <XMI.exporterVersion>0.24(5) revised on $Date: 2006-11-06 19:55:22 +0100 (Mon, 06 Nov 2006) $
</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.4"/></XMI.header>
  <XMI.content>
    <UML:Model xmi.id = '68cd79:f872d13da7:-7f30' name = 'OUP Project' isSpecification = 'false'
      isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
      <UML:ModelElement.taggedValue>
        <UML:TaggedValue xmi.id = '68cd79:f872d13da7:-7f59' isSpecification = 'false'>
          <UML:TaggedValue.dataValue>tag</UML:TaggedValue.dataValue>
        </UML:TaggedValue>
      </UML:ModelElement.taggedValue>
      <UML:Namespace.ownedElement>
    </UML:Model>
  </XMI.content>
</XMI>
```

Listagem 2 – Estrutura de documento XMI

Algumas ferramentas apresentaram problemas na exportação do documento XMI, impedindo a conversão para o formato OWL. Pelo fato de restrições em ferramentas a proposta do UML2ONTO ficou simplificada no ponto que deixa o refinamento da ontologia para uma ferramenta especializada. Mas isto não é crítico na proposta cujo principal enfoque é a análise e projeto no processo de construção de ontologias.

Para efetuar a conversão do arquivo XMI para OWL foi utilizado um processador XSLT[6]. XSLT é uma linguagem que permite transformar um documento XML em outro documento XML.

XSLT é modelado para uso como parte do XSL, que é uma linguagem de estilos para XML. Desta forma para o XSLT, XSL inclui um vocabulário XML para formatos específicos. XSL especifica o estilo de um documento XML, XSLT descreve como o documento é transformado em outro documento XML usando o vocabulário de formatação [W3C, 1999].

---

6 http://www.w3.org/TR/xslt

No processo de conversão foi utilizado o Xalan-Java2[7], que é um processador XSLT que pode ser utilizado na linha de comando, em uma applet ou um servlet, ou como módulo em outro programa. O arquivo .xslt utilizado foi o UMLtoOWL[8], que faz o mapeamento do arquivo XMI para um arquivo OWL. Ao se utilizar tal arquivo não tem necessidade de efetuar mudanças na ferramenta de modelagem UML para que se possa construir a ontologia.

O passo final na geração do arquivo é executar na linha de comando o trecho que está na listagem 3, substituindo os nomes dos documentos pelos que foram atribuídos previamente.

```
java –jar xalan.jar -in DocumentoEntrada.xmi -xsl UMLtoOWL.xslt -out
DocumentoSaida.owl
```

Listagem 3 – Linha de comando para gerar OWL

Gerado o documento OWL (apêndice A), é criado um projeto no protégé utilizando-se o wizard que a ferramenta oferece. Em seguida pode-se realizar a importação do arquivo OWL (figura 16).

[7] http://xml.apache.org/xalan-j/
[8] http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/

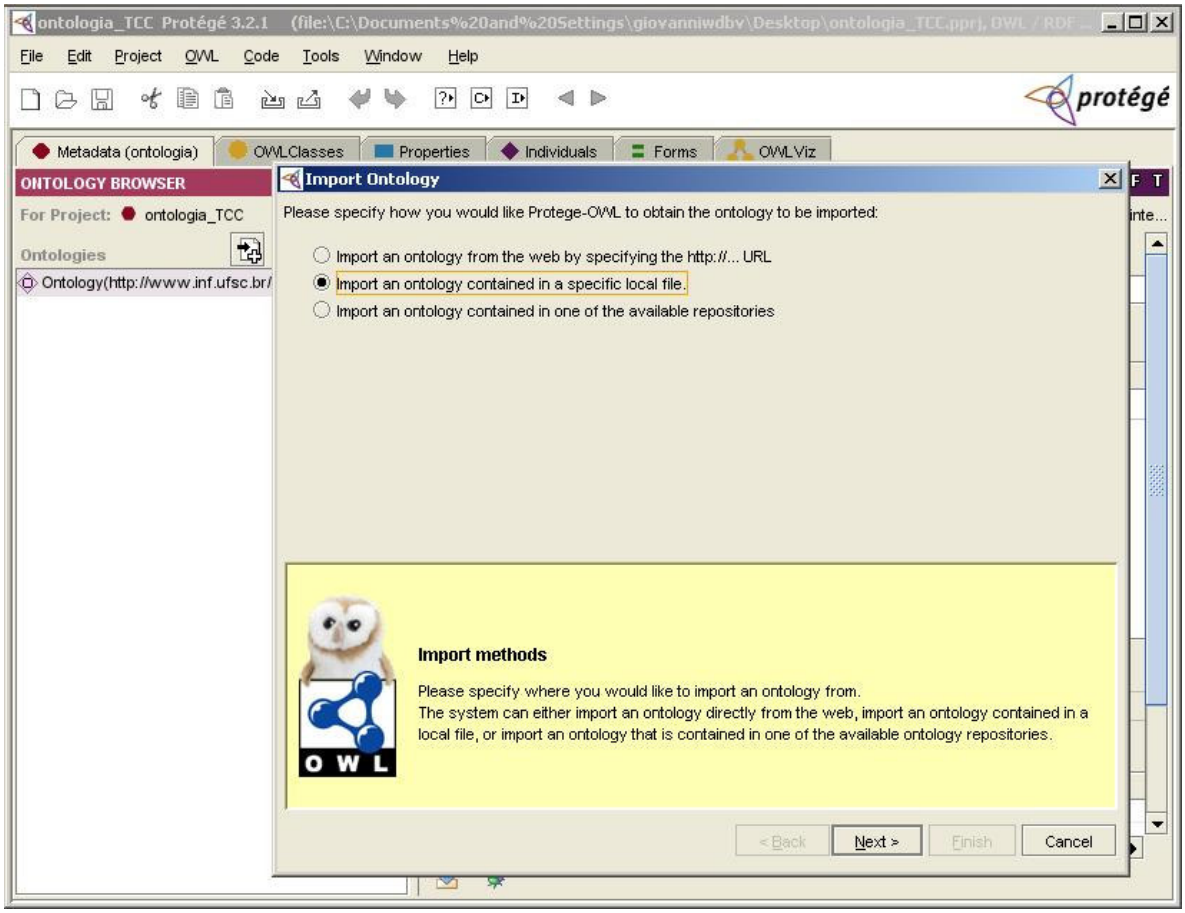

Figura 16. Importação de OWL no Protégé

Finalizada a importação é possível visualizar a estrutura de classes e relações criadas na modelagem UML (figura 17). A partir deste ponto é possível efetuar um refinamento do modelo, seja criando instâncias, criando restrições, definindo novas propriedades, validar a estrutura do documento dentre outras atividades inerente do processo de construção de ontologias.

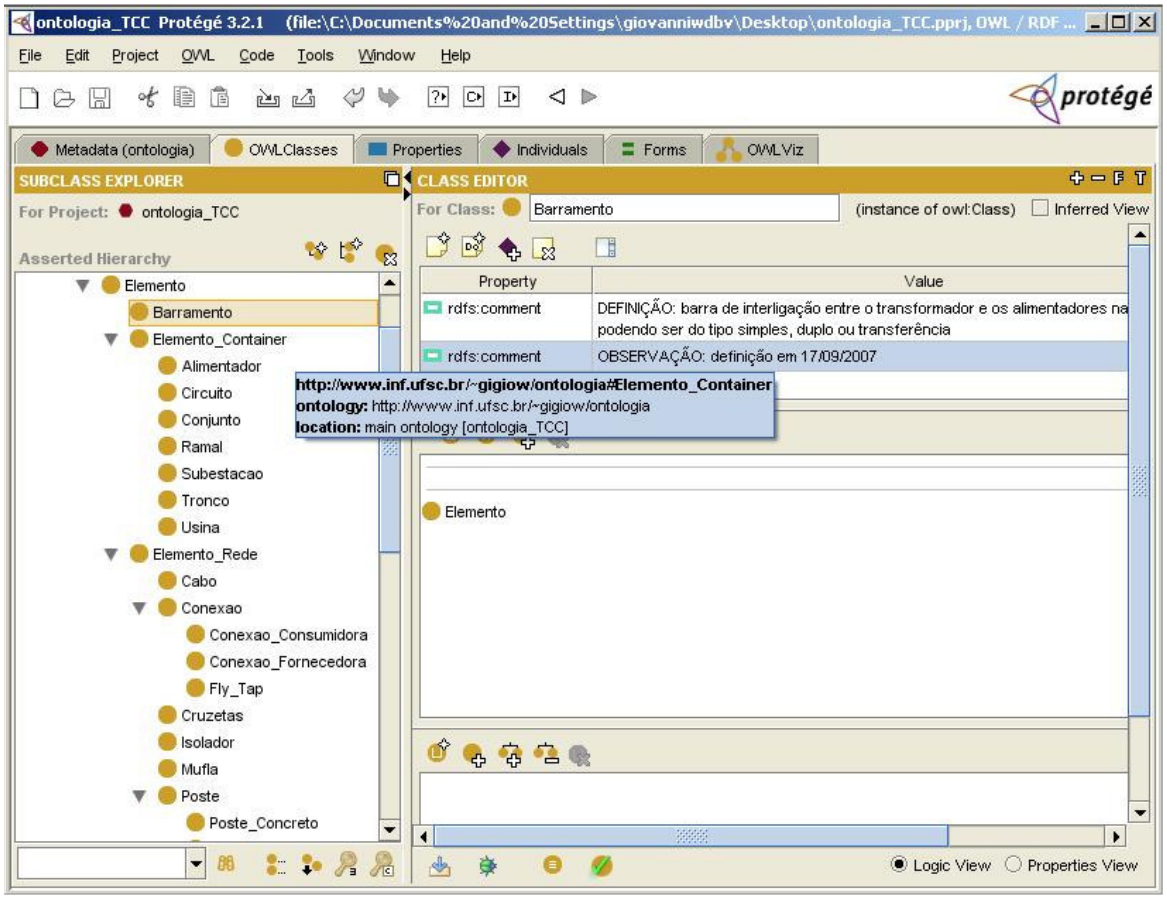

Figura 17. OWL importado

7. Conclusões e Trabalhos Futuros

Na seqüência serão apresentadas as conclusões sobre o trabalho desenvolvido e também serão apresentadas algumas propostas para desenvolvimento de trabalhos futuros.

7.1. Conclusões

A maior dificuldade no processo de análise e projeto na construção de ontologias é entender o domínio e traduzir este para uma formalização, fato que muitas vezes pode causar confusão e atrasos quando tenta se construir um modelo a respeito de um determinado domínio.

A proposta apresenta a ontologia de uma forma sucinta, buscando identificar relações entre as áreas de engenharia de ontologias e engenharia de software. Esta aproximação ocorre propondo tarefas que se utilizam das facilidades da modelagem UML no processo de construção de ontologias, uma vez que, possibilita que conceitos sedimentados na engenharia de software possam ser reutilizados.

As tarefas descritas buscam formalizar os procedimentos identificados para o desenvolvimento de uma ontologia. Foram especificadas de uma forma simplificada, mas possuem a objetividade e a facilidade necessária para a compreensão do modelo.

Usando a proposta apresentada, modelou-se uma ontologia do domínio de redes de distribuição de energia como estudo de caso. Os resultados foram muito satisfatórios, todo o processo esteve bem documentado e teve uma boa

clareza na interpretação. Desta forma além de ser feita a validação obteve-se uma pequena ontologia de domínio que pode ser reutilizada e refinada para futuras aplicações.

Ao se utilizar o UML2ONTO, projetos bem estruturados e bem documentados podem ser construídos, tais como na engenharia de software. Definitivamente, os conhecimentos da área de engenharia de software podem ser reaproveitados na engenharia de ontologias. Assim pode-se afirmar que o guia proporciona uma facilidade para a construção de ontologias em escala.

7.2. Trabalhos Futuros

O UML Profile foi especificado na versão 1.4 da UML, por este detalhe algumas ferramentas de modelagem apresentaram problemas para construção da ontologia. Uma proposta como trabalho futuro é a adaptação do UML profile para a versão 2.0 da UML para a exploração de outras ferramentas que possam permitir uma maior evolução na construção da ontologia ainda na ferramenta case.

Outro ponto possível de estudo é a criação de estereótipos UML específicos para a construção de ontologias, ou seja, a UML incorporar a parte de modelagem de ontologias, assim como na UML existe o padrão MVC (Model View Controler).

Por fim, pode-se aprimorar o modelo proposto do UML2ONTO e avaliar a possibilidade da criação de uma metodologia de análise e projeto de ontologias.

8. Referências Bibliográficas

AHMAD, M. N., **COLOMB**, R. M., **SADIQ**, S., **A Relevant Portion of an Ontology: Defining a System of ED Rules Using a Part-Whole Relationship,** Second Asia International Conference on Modelling & Simulation, 2008

ALMEIDA, M.B.; **BAX**. M.P. **Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção**. Ci. Inf., Brasília, v. 32, n. 3, p. 7-20, set./dez. 2003. Disponível em: < www.ibict.br/cienciadainformacao.> Acessado em novembro, 2006.

CALIUSCO, L., **MAIDANA**, C, **GALLI**, M.R., **CHIOTTI**, O., **Contextual Ontology Modeling Language to Facilitate the Use of Enabling Semantic Web Technologies.** Universidad Tecnológica Nacional - FRSF, Argentina, 2006

FAGUNDES, L. D., **ALMEIDA**, R. A. **Aplicação da gestão do conhecimento para análise de falhas no setor de energia elétrica.** in anais do VI SINCONEE, 2005, Recife

GAŠEVIC, D., **DJURIC**, D., **DEVEDŽIC**, V. **Model Driven Architecture and Ontology Development.** Germany, Springer, 2006

GRUBER, Thomas. **A Translation Approach to Portable Ontologies. Knowledge Acquisition**, V.5, n;2, p.199-200, 1993

GUEMBAROVSKI, R. H., **PIMENTEL**, F. J. S., **SELL**, D., **ANCIUTTI**, I., **RODRIGUES**, M. C., **ALMEIDA**, R., **PACHECO**, R. C. S. **Uma Plataforma de Gestão para Redes de Distribuição de Baixa Tensão.** 2004, Florianópolis

KAGAN, N., **OLIVEIRA**, C. C. B., **ROBBA**, E. J. **Introdução aos Sistemas de Distribuição de Energia Elétrica.** Brasil, Editora Edgard Blücher, 2005

NAVEGA, S. **Técnicas Para Representação Computacional de Conhecimento** publicado nos Anais do Infoimagem (Cenadem, Setembro 2005)

NOY, N.; **MCGUINESS**, D. **Ontology Development 101 – A guide to creating your first ontology** – KSL Technical Report, Standford University, 2001 http://protege.stanford.edu/publications/ontology_development/ontology101.html

PABLA, A. S. **Electric Power Distribution.** USA, Mc Graw Hill, 2005

PATHAK, J., **LI,** Y., **HONAVAR,** V., **MCCALLEY,** J. **A Service-Oriented Architecture for Electric Power Transmission System Asset Management.** Iowa State University, USA, 2006. Disponível em: < www.cs.iastate.edu/~jpathak/WESOA-2006.pdf>. Acessado em Março, 2007.

PROTÉGÉ (2007), Protégé, Stanford University [Online]. Disponível em: < http://protege.stanford.edu/> Acessado em outubro, 2007.

REZGUI, Y. **Text-based domain ontology building using Tf-Idf and metric clusters techniques. The Knowledge Engineering Review**, Vol. 22:4, 379–403. Cambridge University Press - United Kingdom, 2007.

SHENG-HSUN HSU & **HUANG-PIN SHEN**. Total Quality Management, Vol. 16, No. 3, 351–361, Maio 2005.

SILVA, J. B., **SAMPAIO**, M., **PEZZIN**, J. **Usando ontologias na construção de modelos mda (*model-driven architecture*)**. UNIFACS, Salvador, BA, 2006 Disponível em: comp.uniformg.edu.br/plone/artigos2006/heitor/Ercomp_Artigo02.pdf

SOON Ae CHUN, **GELLER**, J. **Evaluating Ontologies Based on the Naturalness of Their Preferred Terms**, Univ. of New York, New York, 2008.

UMLtoOWL, (2006), Conversor XSLT. Disponível em: < http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/> Acessado em Julho, 2007.

WIKIPEDIA (2007), Sistema Especialista, Disponível em: <http://pt.wikipedia.org/wiki/Sistema_especialista/> Acessado em setembro, 2007.

Xalan Java 2, Processador XSLT. Disponível em: < http://xml.apache.org/xalan-j/> Acessado em Julho, 2007.

9. Apêndice A – Código OWL

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://owl.protege.stanford.edu#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#" xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:UML2="org.omg.xmi.namespace.UML2"
xml:base="http://owl.protege.stanford.edu">
<owl:Class rdf:ID="Elemento">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty rdf:about="#temCodigo"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:DatatypeProperty rdf:ID="temCodigo">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Elemento"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Equipamento_Protecao">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Equipamento_Especial">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Transformador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Especial"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Elemento_Container">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
```

```
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty
rdf:about="#temNumeroConsumidores"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Tronco">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Subestacao">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Conjunto">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty rdf:about="#temNome"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty rdf:about="#temDec"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#ehAlimentado"/>
```

```
                    </owl:onProperty>
                    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Alimentador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temCabo"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temElementoRede"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temEquipamento"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty rdf:about="#temExtensao"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
```

```
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Usina">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Circuito">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Ramal">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Container"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temCabo"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Gerador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Especial"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Chave">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Especial"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Capacitor">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Especial"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Regulador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Especial"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede">
```

```
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede_Radial">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Padrao_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Transformador_AT_MT">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Transformador"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Transformador_MT_BT">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Transformador"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede_Aereo">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Padrao_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede_Espinha_Peixe">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Padrao_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede_Subterranea">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Padrao_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Padrao_Rede_Compacto">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Padrao_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Barramento">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Equipamento_Medicao">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento"/>
        </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Transformador_Corrente">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Medicao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Transformador_Tensao">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Medicao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Para_Raios">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Protecao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Seccionador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Protecao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Disjuntor">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Protecao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Chave_Fusivel">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Protecao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Religador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Equipamento_Protecao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Elemento_Rede">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Mufla">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Cruzeta">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
```

```
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temIsolador"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Conexao">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Isolador">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Cabo">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:DatatypeProperty rdf:about="#temCorrente"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Trecho">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Poste">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento_Rede"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temCruzeta"/>
```

```
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temEquipamento"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:about="#temIsolador"/>
                        </owl:onProperty>
                        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Conexao_Fornecedora">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Conexao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Conexao_Consumidora">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Conexao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Fly_Tap">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Conexao"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Poste_Concreto">
        <rdfs:subClassOf>
                <owl:Class rdf:about="#Poste"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Poste_Madeira">
        <rdfs:subClassOf>
```

```
                <owl:Class rdf:about="#Poste"/>
        </rdfs:subClassOf>
</owl:Class>
<owl:DatatypeProperty rdf:ID="temNome">
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Conjunto"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="temDec">
        <rdfs:range rdf:resource="#float"/>
        <rdfs:range rdf:resource="#float"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Conjunto"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="temNumeroConsumidores">
        <rdfs:range rdf:resource="#int"/>
        <rdfs:range rdf:resource="#int"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Elemento_Container"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="ehAlimentado">
        <rdfs:range>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Alimentador"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:range>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Conjunto"/>
                        </owl:unionOf>
```

```
                </owl:Class>
        </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="temCabo">
        <rdfs:range>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Cabo"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:range>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Alimentador"/>
                                <owl:Class rdf:about="#Ramal"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="temCruzeta">
        <rdfs:range>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Cruzeta"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:range>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Poste"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="temIsolador">
        <rdfs:range>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Isolador"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:range>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Cruzeta"/>
                                <owl:Class rdf:about="#Poste"/>
```

```
                    </owl:unionOf>
                </owl:Class>
            </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="temElementoRede">
            <rdfs:range>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Elemento_Rede"/>
                    </owl:unionOf>
                </owl:Class>
            </rdfs:range>
            <rdfs:domain>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Alimentador"/>
                    </owl:unionOf>
                </owl:Class>
            </rdfs:domain>
</owl:ObjectProperty>
<owl:Class rdf:ID="Equipamento">
            <rdfs:subClassOf>
                <owl:Class rdf:about="#Elemento"/>
            </rdfs:subClassOf>
            <rdfs:subClassOf>
                <owl:Restriction>
                    <owl:onProperty>
                        <owl:DatatypeProperty rdf:about="#temTensaoNominal"/>
                    </owl:onProperty>
                    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:cardin
ality>
                </owl:Restriction>
            </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="temEquipamento">
            <rdfs:range>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Equipamento"/>
                    </owl:unionOf>
                </owl:Class>
            </rdfs:range>
            <rdfs:domain>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#Alimentador"/>
                        <owl:Class rdf:about="#Poste"/>
                    </owl:unionOf>
```

```
                </owl:Class>
        </rdfs:domain>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="temCorrente">
        <rdfs:range rdf:resource="#float"/>
        <rdfs:range rdf:resource="#float"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Cabo"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="temTensaoNominal">
        <rdfs:range rdf:resource="#float"/>
        <rdfs:range/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Equipamento"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="temExtensao">
        <rdfs:range rdf:resource="#int"/>
        <rdfs:range rdf:resource="#int"/>
        <rdfs:domain>
                <owl:Class>
                        <owl:unionOf rdf:parseType="Collection">
                                <owl:Class rdf:about="#Alimentador"/>
                        </owl:unionOf>
                </owl:Class>
        </rdfs:domain>
</owl:DatatypeProperty>
</rdf:RDF>
```

Anexo I – Artigo

O processo de construção de ontologias baseado na modelagem UML

Giovanni W. D. B. Victorette[1]

[1]Sistemas de Informação – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

The Ontology Development Process Based in UML Modeling

Giovanni Won Dias B. Victorette (INE/UFSC, gigiow@gmail.com)

Resumo: Artefatos e técnicas da engenharia de software estão consolidados no ambiente empresarial de desenvolvimento e podem ser utilizados no processo de construção de ontologias. Através de uma abordagem simplificada, porém não menos eficaz, propõem-se nesse artigo a utilização da linguagem UML e utilização estruturada das técnicas de análise e projeto para o desenvolvimento de ontologias. Utilizando-se de tais recursos criou-se um guia para auxiliar, principalmente, o desenvolvimento de ontologias em escala. O fator motivador é o nível crescente de demandas por ontologias, e o fator condicionante é a necessidade de disponibilizar uma metodologia simplificada para facilitar implementações de ontologias.

Palavras-chave: Ontologia. Engenharia de software. UML.

Abstract: *Artifacts and techniques from software engineering are consolidated in enterprise development environment and could be used in ontology development process. Through a simplified approach, however not less effective, this article proposes the utilization of UML Language and structured utilization from analysis and project techniques to ontology development. Using those resources a guide was built to help large-scale ontology development. The motivating factor is the increasing level of demands for ontologies and the constraint factor is the need to provide a simplified methodology to facilitate ontology implementation.*

Keywords: *Ontology. Software engineering. UML.*

1. Introdução

Com o advento da web semântica e o avanço da aplicação de ontologias, verifica-se uma necessidade de evolução e aprimoramento de técnicas e ferramentas existentes para construção e desenvolvimento das mesmas. Ferramentas de metodologia que guiam uma elaboração padronizada ainda estão amadurecendo. Em conseqüência, os métodos de desenvolvimento, validação, verificação, e mesmo de documentação de ontologias continuam evoluindo.

Segundo Desousa [1], "a engenharia de software é um domínio altamente orientado ao conhecimento, no qual os fatores de sucesso estão relacionados com a experiência das pessoas envolvidas nas seguintes fases: projeto, construção, teste e implantação". Mas ainda existe uma necessidade de apoiar a comunidade de engenharia de software para entender a engenharia de ontologias.

Faz-se necessário observar as relações e diminuir a lacuna existente entre a engenharia de software e a gestão do conhecimento. Este artigo procura apresentar procedimentos que sejam úteis para aproximar as duas áreas em questão, utilizando a modelagem UML e procedimentos de engenharia de software.

2. Ontologias

A palavra ontologia vem do grego, significando o estudo do ser. De acordo com Gruber [5], para Ciências da Computação e Sistemas de Informação, uma ontologia é uma especificação explícita de uma conceitualização, sendo que uma conceitualização é uma visão abstrata do mundo que se deseja representar, consistindo nos objetos, nos conceitos e nas relações entre eles, que existem no mundo representado.

Muitos trabalhos na área de tecnologia vêm explorando o uso de ontologias e metodologias para construção das mesmas. Segundo Noy [9], não há uma forma correta de modelar um domínio, ou seja, não existe uma única maneira de se construir uma ontologia.

A construção e manipulação de ontologias têm sido sistematizadas por metodologias. Como todas essas metodologias apresentam aspectos positivos e negativos, muitos dos desenvolvedores acabam utilizando estas apenas como base para o desenvolvimento de metodologias próprias. Dentre as mais conhecidas estão a metodologia 101 [9], On-To-Knowledge [2] e o Methontology [3].

2.1 OWL

A linguagem OWL foi desenvolvida com base nas necessidades da Web Semântica. Foi desenvolvida pelo consórcio W3C e projetada para poder fazer uma especificação formal da Ontologia. O vocabulário OWL inclui um conjunto de elementos e atributos XML com significados bem definidos, sendo dividida em três tipos de sublinguagens:

- OWL Lite: suporta ontologias simplificadas que precisam apenas de uma classificação hierárquica e restrições simples;
- OWL – DL: É baseada em lógica de descrição, que permite estabelecer a classificação automática de hierarquia além da checagem de consistência da ontologia de acordo com o respectivo metamodelo.
- OWL Full: possui expressividade máxima aliada à liberdade da sintaxe do RDF[9] (Resource Description Language), mas sem garantias computacionais.

3. Metamodelos e UML

A relevante iniciativa da comunidade da engenharia de software chamada de Model Driven Development (MDD) está sendo desenvolvida em paralelo com a Web Semântica [7]. A iniciativa de pesquisa mais importante nesta área é a Arquitetura Orientada a Modelos (MDA) [8], que está sendo desenvolvida sob o guarda chuva da Object Management Group (OMG).

A MDA propõe a criação de modelos em diferentes níveis de abstração separando os interesses de implementação da arquitetura a ser implementada. Os modelos poderão relacionar-se usando mapeamentos, sendo estes feitos automaticamente ou manualmente [11].

Mesmo a linguagem UML tendo sido projetada para atender qualquer tecnologia, a OMG criou um mecanismo para possibilitar o uso da UML de acordo com necessidades específicas. Tal mecanismo é chamado de *Profile.*

UML *Profile* é um conceito utilizado para adaptar os construtores básicos de UML para algum propósito específico. Essencialmente, isto significa introduzir novos tipos de elementos estendendo ou adicionando estes para a ferramenta [4]. Na UML 2.0 esta idéia

[9] <http://www.w3.org/RDF/>

evoluiu utilizando-se um meta-modelo, que define a UML, sendo possível adicionar novas regras ao meta-modelo de forma a estender a UML [6].

4. Proposta do UML2ONTO

Identificada a dificuldade da comunidade de engenharia de software quanto à construção de ontologias, e considerando ainda os princípios preconizados pela Engenharia de Conhecimento cujo propósito principal é entender toda a organização, identificando gargalos e as oportunidades, aquisição e representação de conhecimento através de métodos científicos criou-se uma proposta utilizando tarefas para o processo de construção de ontologias baseado na modelagem UML.

O modelo proposto foi denominado UML2ONTO (UML para Ontologias) (Figura 1), uma vez que aborda conceitos de análise e projetos de software aplicados à modelagem da ontologia.

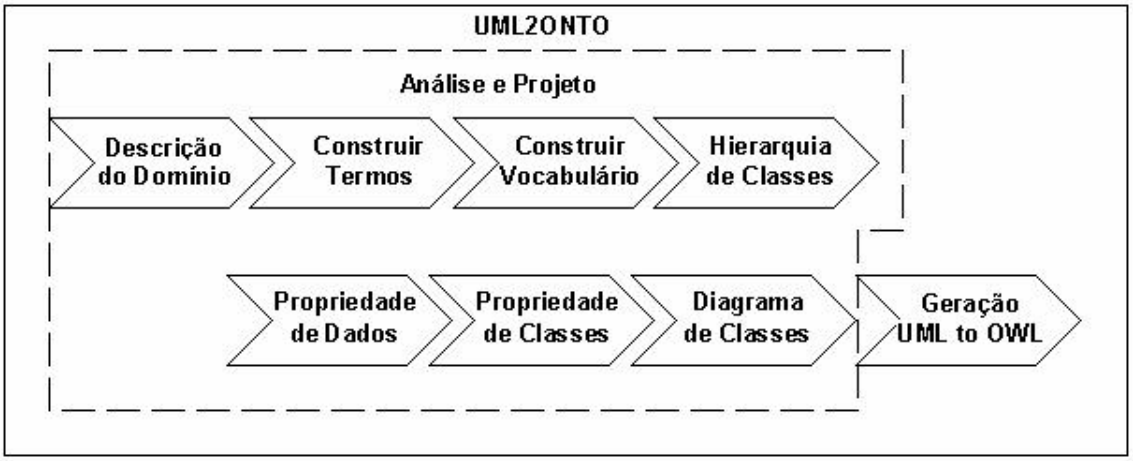

Figura 18. UML2ONTO

De acordo com o UML2ONTO a primeira tarefa visa definir e descrever o domínio. Esta é uma técnica utilizada pela engenharia de software, onde se produz um relatório sobre o domínio estudado, procurando identificar todas as características do domínio.

Avançando nas tarefas é o momento de se construir os termos, ou seja, identificar Termos e Relações no documento gerado no passo anterior. Neste estágio os substantivos encontrados serão candidatos aos termos e os verbos serão candidatos às relações.

Tendo identificado termos e relações, inicia-se a construção do vocabulário, para isso selecionam-se os termos identificados que são conceitos no domínio. Deve-se sempre ter em mente que uma classe é uma coleção de elementos com propriedades similares.

Com as classes identificadas é feita a etapa de hierarquia de classes. Neste momento é identificada a taxonomia hierárquica definindo-se superclasses e subclasses.

Atributos de classe ou associações são representados por propriedades. Uma propriedade é uma relação entre um sujeito e um objeto. Na tarefa Propriedade de Dados são identificadas as relações com valores de dados.

Definindo Propriedade de Classes é definida a propriedade cujo *range* é uma classe.

Na ferramenta UML cria-se o diagrama de classes, aplicando os devidos estereótipos, montando toda a estrutura identificada com as respectivas relações.

Na parte final é feita a exportação do modelo para o formato XMI (XML Metadata Interchange) e utilizando um processador XSLT (eXtensible Stylesheet Language Transformation) gera-se o arquivo OWL [4].

5. Estudo de Caso – Sistema Elétrico de Distribuição

Para a validação do modelo proposto, utilizou-se o domínio de redes de distribuição. A motivação para a escolha de tal domínio está no fato de que existe uma crescente preocupação com a evolução dos serviços prestados, motivado pela modicidade tarifária estabelecida pela ANEEL[10], (Agência Nacional de Energia Elétrica), cujo propósito principal é a eficiência do Sistema Elétrico.

Aplicando-se UML2ONTO, verificou-se que o processo de documentação e a geração de produtos ao final de cada etapa auxiliam bastante a organização e o desenvolvimento do trabalho.

Para a modelagem UML foi utilizada a ferramenta ArgoUML[11], a escolha foi feita por ser uma ferramenta com suporte a XMI e de livre distribuição. A exportação para o arquivo XMI é bem simples e intuitiva, nenhum problema foi detectado nesta etapa.

Para a conversão de XMI para OWL foi utilizado um processador XSLT (Xalan-Java2)[12] e o arquivo UMLtoOWL.xslt [4]. No final obteve-se o arquivo OWL que pode ser importado para uma ferramenta especializada (Protege) [10], possibilitando assim maiores refinamentos.

---

10 <http://www.aneel.gov.br/>

11 <http://argouml.tigris.org/>

12 <http://xml.apache.org/xalan-j/>

6. Conclusões

O artigo apresenta uma abordagem sucinta orientada a tarefas para construção de Ontologias, buscando identificar relações entre as técnicas estruturadas na Engenharia de Software e a Engenharia de Ontologias. Esta aproximação se dá através da proposição de tarefas que se utilizam das facilidades da modelagem UML no processo de construção de ontologias, possibilitando que conceitos sedimentados na engenharia de software sejam reutilizados.

Ao se utilizar o UML2ONTO, produtos decorrentes de práticas consolidadas na engenharia de software, a exemplo de projetos estruturados, adequada documentação e controle do desenvolvimento são obtidos. Definitivamente, os conhecimentos da área de engenharia de software podem ser reutilizados no desenvolvimento de ontologias. Pode-se, assim, afirmar que o guia facilita significativamente a construção de ontologias em escala.

A abordagem proposta deve ser aprimorada visando implementar uma metodologia para o desenvolvimento de ontologias.

Referencias

1. DESOUZA, Kevin C. (2003) Barriers to Effective Use of Knowledge Management Systems in Software Engineering. Communications of the ACM, vol. 46, n. 1, p. 99-101.
2. FENSEL, Dieter. Ontology-Based Knowledge Management. IEEE Computer. November, 2002.
3. FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURINO, N. Methontology: From ontological art towards ontological engineering. In: Spring Symposium Series, Stanford, p. 33-40, 1997
4. GAŠEVIC, D., DJURIC, D., DEVEDŽIC, V. Model Driven Architecture and Ontology Development. Germany, Springer, 2006
5. GRUBER, Thomas. A Translation Approach to Portable Ontologies. Knowledge Acquisition, V.5, n;2, p.199-200, 1993
6. KLEPPE, A., Warmer, J., BAST, W. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley, 2003
7. MELLOR, S.J., CLARK, A.N., & FUTAGAMI, T. (2003a), "Guest editors' introduction: Model-driven development," IEEE Software, vol. 20, no. 5, pp.14-18.
8. MILLER, J., MUKERJI, J. (eds.), "MDA Guide Version 1.0", OMG Document: omg/2003-05-01, http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf, May 2003.

9. NOY, N.; MCGUINESS, D. Ontology Development 101 – A guide to creating your first ontology – KSL Technical Report, Standford University, 2001 http://protege.stanford.edu/publications/ontology_development/ontology101.html

10. PROTÉGÉ (2007), Protégé, Stanford University [Online]. Disponível em: < http://protege.stanford.edu/> Acessado em outubro, 2007.

11. SILVA, J. B., SAMPAIO, M., PEZZIN, J. Usando ontologias na construção de modelos mda (model-driven architecture). UNIFACS, Salvador, BA, 2006