UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Departamento de Ciência da Computação

Mapeamento de Modelos e Linguagens para Representação de Conhecimento

Lessandro Mendes Alhadas

Bacharelado em Ciência da Computação
Orientador: Prof. Tarcísio de Souza Lima

CIÊNCIA DA COMPUTAÇÃO

JUIZ DE FORA, MG, BRASIL
Dezembro de 2008

Mapeamento de Modelos e Linguagens para Representação de Conhecimento

Lessandro Mendes Alhadas

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Tarcísio de Souza Lima, MSc. em Informática, PUC/RJ (Orientador)

Raul Fonseca Neto, DSc. em Eng. de Sist. e Computação, COPPE/UFRJ

Fernanda Cláudia Alves Campos, DSc. em Eng. de Sist. e Computação, COPPE/UFRJ

JUIZ DE FORA, MG, BRASIL
Dezembro de 2008

AGRADECIMENTOS

Agradeço a todos os meus familiares, em especial meus tios Theóphilo de Araújo Neto e Sônia Maria Mendes de Araújo, Getúlio Costa e Regina Coeli Mendes Costa ao apoio fornecido durante não somente o período de estudos da graduação, bem como por toda minha vida sendo para mim, exemplo de pessoas de caráter imensurável.

Agradeço a meu orientador, Tarcísio de Souza Lima, pelos ensinamentos proferidos como professor, coordenador e orientador além de fazer parte importante no meu trajeto de formação profissional.

Também, tão merecido é os agradecimentos a todos os professores do Departamento de Ciências da Computação, como Raul Fonseca Neto, Rubens de Oliveira, Regina Braga, secretários, funcionários e a todos aqueles a que tive convívio durante minha graduação na Universidade Federal de Juiz de Fora.

Obrigado José Roberto Tagliati, meu coordenador de bolsa de estudos e que contribuiu fortemente para que este sonho tivesse sendo realizado.

Obrigado aos meus colegas de turma,
Obrigado professores da banca de defesa,

Obrigado UFJF.

DEDICATÓRIA

Este trabalho é dedicado à minha mãe, Maria Divina Mendes Alhadas.

Uma gota de recompensa ao oceano de seus esforços que me forneceram suporte aos estudos e a esperança em mim depositada.

SUMÁRIO

LISTA DE SIGLAS

API	*Application Programming Interface*
CASE	*Computer-aided Software Engineering*
DAML	*Darpa Agent Markup Language*
DAML+OIL	Linguagem que substituiu a linguagem DAML+ONT
E-R	Entidade-Relacionamento
eRDF	*Embedded RDF*
FQ	*Focal Question*
FOAF	*Friend of a Friend*
FOL	*First Order Logic*
GDA	*Global Document Annotation*
GIF	*Graphics Interchange Format*
GPL	*GNU Public License*
hCalendar	*HTML iCalendar*
hCard	*HTML vCard*
HTML	*Hipertext Markup Language*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IES	Instituição de Ensino Superior
JPEG	*Joint Photographic Experts Group*
KBS	*Knowledge-based System*
MDA	*Model-Driven Architecture*
MER	Modelo Entidade-Relacionamento
MIQ	*Machine Intelligence Quotient*
MOF	*Meta-Object Facility*
OCL	*Object Constraint Language*
ODM	*Ontology Definition MetaModel*
OIL	*Ontology Inference Layer*
OMG	*Object Management Group*
OO	*Orientação a Objetos*
OUP	*Ontology UML Profile*

OWL	*Web Ontology Language*
OWL-S	*Semantic Markup for Web Services*
PDF	*Portable Document Format*
PHP	*Personal Home Page*
RDF	*Resource Description Framework*
RDFa	*Resource Description Framework in attribute*
RDFS	*ResourceDescription Framework Schema*
RDQL	*RDF Data Query Language*
RSS	*Really Simple Syndication*
SGBD	Sistema de Gerenciamento de Banco de Dados
SHOE	*Simple HTML Ontology Extension*
SKOS	*Simple Knowledge Organization System*
SPARQL	*Simple Protocol and RDF Query Language*
SWSL	*Semantic Web Service Language*
TPCI	*TIOBE Programming Community Index*
UFJF	Universidade Federal de Juiz de Fora
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
vCard	*Versit Card*
XFN	*XHTML Friends Network*
XHTML	*eXtensible HTML*
XMI	*XML Metadata Interchange*
XML	*Extensible Markup Language*
XOXO	*eXtensible Open XHTML Outlines*
XSLT	*eXtensible Stylesheet Language Transformation*
W3C	*World Wide Web Consortium*
WSDL	*Web Service Definition Language*
WSDL-S	*Web Service Semantics*
WSML	*Web Service Modeling Language*
WSMO	*Web Service Modeling Ontology*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho de monografia visa apresentar os princípios relacionados a representação do conhecimento, ontologias e Web semântica bem como a geração de ambientes especificados e inferidos por estas metodologias, linguagens de representação, uma breve comparação entre ferramentas de programação e uma proposta para mapeamento entre os vários modelos que são definidos com diferentes abordagens. Compilar algumas boas práticas observadas tanto nas representações do conhecimento tanto nas especificações semânticas a fim de obter um mapeamento mais intuitivo, ao mesmo tempo confiável e formalizado semanticamente.

PALAVRAS-CHAVE: Representação do Conhecimento, Web Semântica, Ontologia, Lógica de Descrição, Integração de dados, Mapeamento de Modelos.

ABSTRACT

This monograph work aims introduce the principles related for knowledge representation, ontologies and semantic Web as well as the generation of specified and inferred environments by this methodologies, representation languages, a brief comparison between coding and a proposal to mapping between various models that it's defined in various approaches. Compile some good practices observed in both representations of knowledge both in semantic specifications in order to obtain a more intuitive mapping, while reliable and formalized semantically.

KEY-WORDS: Knowledge Representation, Semantic Web, Ontology, Data Integration, Model's Mapping.

Capítulo 1
INTRODUÇÃO

Este capítulo traz uma breve descrição deste trabalho bem como a motivação para tal, algumas definições do contexto a ser desenvolvido e a organização deste documento ao longo dos capítulos.

1.1 Motivação

Nos dias atuais muito se vê falar de definições e modelos de sistemas computacionais inteligentes, baseados em conhecimento, baseados em ontologias. Mas, ainda longe de ser uma ferramenta para integração e convergência de ideais e convenções para facilitar a construção destes sistemas bem como solução de problemas, reutilização de modelos entre outros aspectos que apóiam se em aspectos metodológicos e organizacionais, estes ambientes ainda sofrem com problemas de base. Diferentes construções e abordagens para uma mesma classe de problema ou definição, a falta de ferramentas confiáveis, funcionais e estáveis para construção, integração e manutenção das ontologias, falta de consenso em vários pontos do planejamento, projeto e implementação dos sistemas e uma enorme variedade de outros problemas que vem causando uma avalanche de imposições para o bom e pleno inter-relacionamento entre sistemas e seus respectivos contextos representados são alguns dos problemas observados nestes sistemas.

Ontologias estão sendo utilizadas majoritariamente para definição e especificação de um domínio de problema específico e, em alguns poucos casos, são fornecidas como subsidio em mecanismos de inferências em sistemas baseados em conhecimento, conhecidos como KBS *(Knowledge-based System),* ao invés de serem mais bem planejadas e projetadas com predições para geração de modelos mais genéricos, de maior amplitude e principalmente adicionarem semântica real às aplicações e suas interações e não somente às definições do mundo sendo modelado. Algumas iniciativas estão utilizando ontologias para descrições superiores, de mais alto nível de abstração e para descrever recursos e serviços na rede, mas ainda são minoria e com concepções imaturas.

Visto estes pontos e que os ambientes e a arquiteturas baseadas em modelos ontológicos, baseados em conhecimento e com visão semântica ainda estão confusos e em processo de amadurecimento, verifica-se a necessidade de um estudo mais profundo e

exaustivo em todos os aspectos e abordagens que norteiam estes sistemas, sendo forte motivação para este trabalho.

1.2 Primeiras Definições

Na visão da Web semântica, à informação é dado um significado bem definido, possibilitando computadores e pessoas trabalharem em cooperação (BERNERS-LEE, HENDLER e LASSILA, 2001). Podemos considerar que a representação do conhecimento define a anotação de informações sobre um determinado termo ou conceito, suas propriedades e relações e que estas informações tenham significado.

A ontologia é a organização explícita da informação, que deve ter sua terminologia compartilhada e disponível a programas, sendo assim, a base para a Web semântica e uma ótima forma de representar o conhecimento. Deve ser construída e mantida por um grupo de interessados no modelo sendo definido, de maneira a expressar suas entidades e relacionamentos em comum, um compromisso ontológico (*ontological commitment*). Para tais tarefas, os integrantes do grupo devem também acordar sobre as linguagens e ferramentas que serão utilizadas bem como os axiomas e as regras para a manutenção dos propósitos previamente definidos.

Este trabalho visa mostrar como o conhecimento está sendo representado, principalmente no âmbito da Web, em suas diferentes abordagens, linguagens, técnicas e metodologias entre os vários propósitos, como por exemplo, conceitos e relações em redes sociais, mapeamento de recursos e dados na rede ou em *grids* computacionais, definição e especificação de serviços Web entre outros. Tem como objeto-alvo o mapeamento entre os modelos de representação do conhecimento e os modelos de representação semântica para a Web.

1.3 Importância do Trabalho

De acordo com BRODIE (2007) “estamos entrando na próxima geração da computação com um modelo de computação e paradigma fundamentalmente diferente caracterizado tecnologicamente por arquiteturas multi-núcleos, virtualização, computação orientada a serviços e Web semântica. A **Ciência da Computação 2.0** irá marcar o **fim da Era da computação com foco em tecnologias** e o **início da Era da Solução de Problemas com seu foco sobre o alto nível de abstração e ferramentas de automação** (isto é, inteligentes) para

domínios do mundo real (isto é, imprecisos) onde são comuns aproximações e mudanças constantes às respostas".

Visto este cenário, este trabalho visa contribuir para a evolução do foco de Web semântica dentre um dos conceitos da **Ciência da Computação 2.0** e objetiva apresentar uma proposta de mapeamento entre os modelos de representação de conhecimento e os modelos de representação semântica bem como um resumo das boas práticas observadas em ambientes computacionais que caminham para uma estruturação mais especificada semanticamente e que utilizam técnicas e metodologias para aumentar a integração e comunicação, dado o ponto de vista contextual das aplicações, das fontes de dados e dos recursos disponíveis.

Existem outros trabalhos que apresentam partes e tópicos aparentemente semelhantes a este, como o de FURGERI (2006) e VICTORETTE (2008), mas diferem na visão e alvo aspirarado. Enquanto outros trabalhos visam uma solução específica onde será utilizada certa arquitetura com certas ferramentas, com tais linguagens, que de certa forma foram classificadas como "as melhores", este trabalho visa obter "o melhor dos mundos" no estado da arte do desenvolvimento semântico, partindo de uma representação de fácil visualização e entendimento e compilar um guia de boas práticas observadas para projetos de sites Web com maior teor significativo, especificados por ontologias e linguagens padronizadas pelo W3C.

1.4 Organização da Monografia

A monografia, além desta introdução e das referências bibliográficas, divide-se em outros quatro capítulos.

No segundo capítulo é apresentado um breve resumo das técnicas de representação do conhecimento, desenvolvimento de sistemas especificados por ontologias e dos modelos e linguagens mais comuns utilizados para representação destas.

O terceiro capítulo aborda as ferramentas que estão dando suporte ao ciclo de vida das ontologias bem como criação, manutenção, descobrimento, reutilização, inferência, estatísticas entre outras tarefas. Uma breve comparação entre as ferramentas mais populares e um estudo de caso realizado na ferramenta mais bem conceituada, de acordo com alguns critérios pré-definidos.

No quarto capítulo serão tratados em mais detalhes os modelos mais utilizados para a representação de ontologias e suas possíveis condições de mapeamento, bem como equivalências, discordâncias, conectividade de recursos, grau de formalidade, capacidade de raciocínio computacional entre outras características necessárias e suficientes para a integração plena das diferentes abordagens.

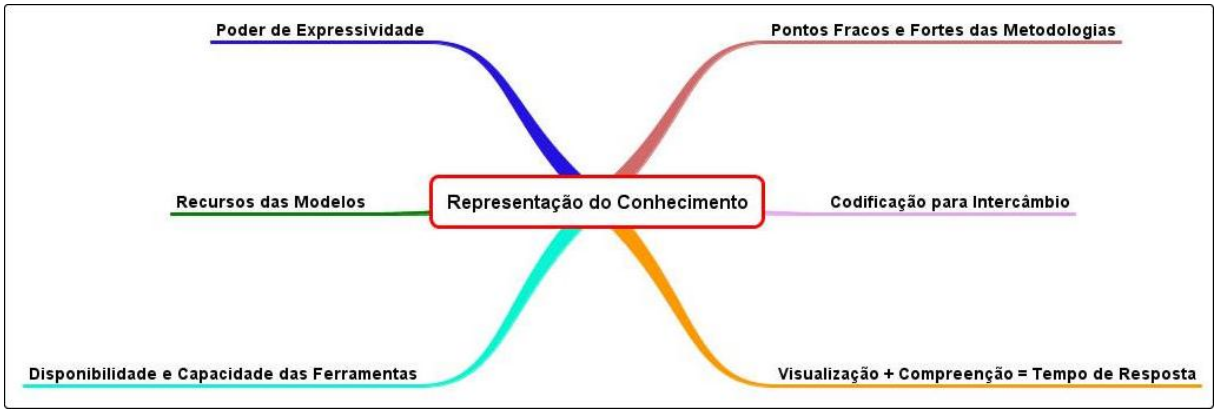
O quinto capítulo apresenta as conclusões alcançadas bem como uma proposta de boas práticas para o desenvolvimento de sistemas Web especificados semanticamente.

Cada capítulo, a partir do Capítulo 2, é iniciado por uma representação em forma de **mapa mental**. Esses mapas são os *Mind Maps*[1]. Estes mapas são utilizados para evidenciar os pontos essenciais de cada capítulo. Cada mapa é representado por um tópico central e suas ramificações, a partir do processo de associações de idéias. A utilização de tais mapas como resumo de capítulos foi inspirada na mesma utilização feita por Thomas Passim em seu livro de semântica e no livro "Web Semântica: A Internet do Futuro" de BREITMAN (2005), à qual este parágrafo, foi retirado e adaptado.

O ANEXO A contém o mapa mental completo desta monografia.

Capítulo 2

REPRESENTAÇÃO DO CONHECIMENTO



Não há um método dado como o melhor ou o mais específico para a representação do conhecimento. Podem ser representados de maneira informal, escritos em frases, diagramas ou desenhos confeccionados manualmente, inclusive com grande apoio de alguns especialistas, devido ao fato de não conter imposições de representação e proporcionar maior liberdade na notação e simbologia. Podem ser construídos com o apoio de ferramentas que fornecem recursos e funcionalidades para que se tenha mais formalismo, apliquem-se regras lógicas, adicionem-se imagens e outras mídias entre várias outras técnicas e facilidades para a aquisição e representação do conhecimento.

[1] Construidos a partir da técnica proposta por Tony Buzam (*The Mind Map Book: How to use radiant thinking to maximize your brain's untapped potential*).

Este trabalho se preocupa com a representação do conhecimento especificado e descrito por humanos sobre qualquer área de aplicação, e que seja fácil de ser construído, manipulado e compreendido, visando principalmente à oportunidade de se explicitar conceitos específicos de domínio e aprendizado adquirido, que serão então mapeados e traduzidos em linguagens para a Web semântica. Observa-se que as ferramentas e os recursos para tal representação diferem-se dos modelos de processo de aquisição do conhecimento de forma automatizada, que pode empregar várias técnicas, em separado ou unidas, sobre uma base de conhecimento, para obter regras e padrões e armazená-las também de diferentes maneiras para que possam ser transmitidas ou utilizadas em novas inferências, comunicação e colaboração.

Resumidamente, pode-se definir a representação do conhecimento no âmbito da ciência da computação, como sendo todo tipo de registro que se tem a respeito de um objeto, um termo, uma entidade, um recurso ou dado na rede, características e relações entre estes. Pode ser um metadado, uma regra aplicável a tal classe de objetos, uma característica de uma instância e assim por diante.

Até pouco tempo atrás, os modelos mais comumente utilizados para representar conhecimento, principalmente no cenário da ciência da computação, eram através de bases lógicas, redes semânticas e *frames* (quadros). Logo, com a necessidade de modelagens com maior acurácia e formalismo para tratamento computacional, outras abordagens, linguagens e técnicas foram sendo desenvolvidas, bem como representação orientada a objetos, *script*s de comportamento e vários outros artefatos para representar o conhecimento de dado domínio, relações entre os conceitos e o conseqüente compartilhamento do conhecimento adquirido.

No desenvolvimento da Web semântica, tem sido incluído o desenvolvimento de linguagens para representação do conhecimento baseadas em XML e padrões tais como RDF, RDFS, mapas de tópicos, SHOE, DAML, OIL, DAML+OIL e OWL, à qual será feito um breve estudo de algumas destas no próximo capítulo.

Nem sempre o conhecimento representado e o sendo adquirido está registrado nestas linguagens. Além disso, linguagens com mais estruturas visuais, como *frames* ou diagramas, facilitam a aproximação da representação à representação do conhecimento humano e possuem interface mais amigável ao usuário. Por isso, se faz útil o estudo do mapeamento entre estes modelos, que facilitam a representação, nem sempre são modelos convencionais, e os modelos mais formalizados utilizados em cômputos e inferências na ciência da computação.

Para tal estudo, será considerado entre outros, redes semânticas, *frames*, MER, diagramas de classes UML, e mapas conceituais como sendo os modelos para a representação

do conhecimento, à qual objetiva-se mapear em um artefato formal e que seja representado em linguagem para a Web semântica. Este artefato deve conter dados, recursos e formalidades necessárias e suficientes para representar o conhecimento, e que este tenha sentido semântico. Para isso, podemos considerar a ontologia e algumas linguagens utilizadas como extensões, como sendo o objeto-alvo do mapeamento.

Antes de tratarmos dos modelos de representação do conhecimento, precisamos ainda, fazer algumas considerações sobre representação do conhecimento e Web semântica para que tenhamos clareza no papel de cada um destes no contexto da Web e na contextualização deste trabalho.

2.1 Representação do Conhecimento e Web Semântica

Um registro sobre uma entidade representa um conhecimento sobre tal entidade. Sentido nas relações entre entidades representa semântica de um contexto à qual estas entidades participam. Ao inserir informações a um arquivo, por exemplo, metadados, estamos registrando, de certa forma, um conhecimento sobre este arquivo bem como estamos inserindo semântica ao contexto onde este está imerso.

A Web semântica, de acordo com (BERNERS-LEE, HENDLER e LASSILA, 2001), é uma extensão da Web primitiva à qual os dados e recursos já disponíveis e os que estão sendo criados e inseridos na rede, vão recebendo informações para compor sua semântica e o conhecimento existente, facilitando futuramente a comunicação, entendimento e raciocínio computacional.

Conclui-se aqui, que a representação do conhecimento e a Web semântica estão intimamente relacionados no intuito de dar sentido a algum conceito existente no domínio sendo abordado, ou seja, inserindo dados e informações em recursos Web, utilizando-se de alguma linguagem da Web semântica, esta se representando conhecimento sobre este recurso/conceito no ambiente Web.

Agora será feito um breve estudo dos modelos para representação do conhecimento, bem como suas características mais relevantes, recursos, equivalências e diferenças com outros modelos, a codificação para o intercâmbio com alguma ferramenta que fará a tradução semântica e a representação em si. Não serão somente estudadas e comparadas a fim de se eleger um modelo ou ferramenta como sendo “o melhor” ou “o mais indicado”, mas sim, extrair o que cada um tem de melhor e as contribuições que concretizaram para o avanço dos sistemas “inteligentes”.

2.2 Modelos Lógicos

Modelos lógicos são os modelos mais primitivos e considerados como base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita, como nos sistemas especialistas baseados na linguagem Prolog, seja de forma implícita na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos, como por exemplo, as listas da forma:

(<atributo>,<objeto>,<valor>,<coeficiente de certeza>)

Este modelo é utilizado como padrão para a representação de conhecimento no sistema MYCIN (um sistema para prover conselhos médicos escrito em Lisp). Mesmo os formalismos não lógicos têm, em geral, seu significado formal descrito através de uma especificação lógica de seu comportamento.

O primeiro formato utilizado é o chamado de Lógica Proposicional (ou Cálculo Proposicional). Nele, usam-se proposições e relações lógicas entre proposições. As proposições podem ser verdadeiras ou falsas, assim como as expressões montadas com proposições e relações lógicas como “e”, “ou”, “implicação”, “equivalência” e “negação”. Contudo, o Cálculo Proposicional é menos expressivo do que a conhecida Lógica de Primeira Ordem (FOL - *First-Order Logic*, ou *Predicate Calculus*). Neste caso, lida-se com uma estrutura de representação mais rica, que inclui elementos como: “objetos”, “relações”, “propriedades”, “funções” e “variáveis”.

Como estas representações são feitas de forma textual, de difícil compreensão em modelagens de grandes domínios e se esta interessado em modelos com maior expressividade visual, não será feito aqui grandes considerações sobre os modelos lógicos, deixando somente estas definições como base.

2.3 Redes Semânticas

É uma forma de representação do conhecimento definida como um grafo direcionado, nos quais os vértices representam conceitos, e as arestas representam relações semânticas entre os conceitos, as ligações. Relações de classificação (*is-a, a-kind-of*) e relações de pertença (*part-of, has*), utilizadas para transmitir herança de propriedades, são as mais comumente utilizadas, mas não são as únicas e obrigatórias, ou seja, as relações entre os conceitos podem ser definidas de acordo com as necessidades do domínio sendo modelado. Observe o exemplo na

Figura 2-1 que, por ser um grafo, transmite o conhecimento representado de forma mais clara, rápida e de fácil compreensão por seres humanos.

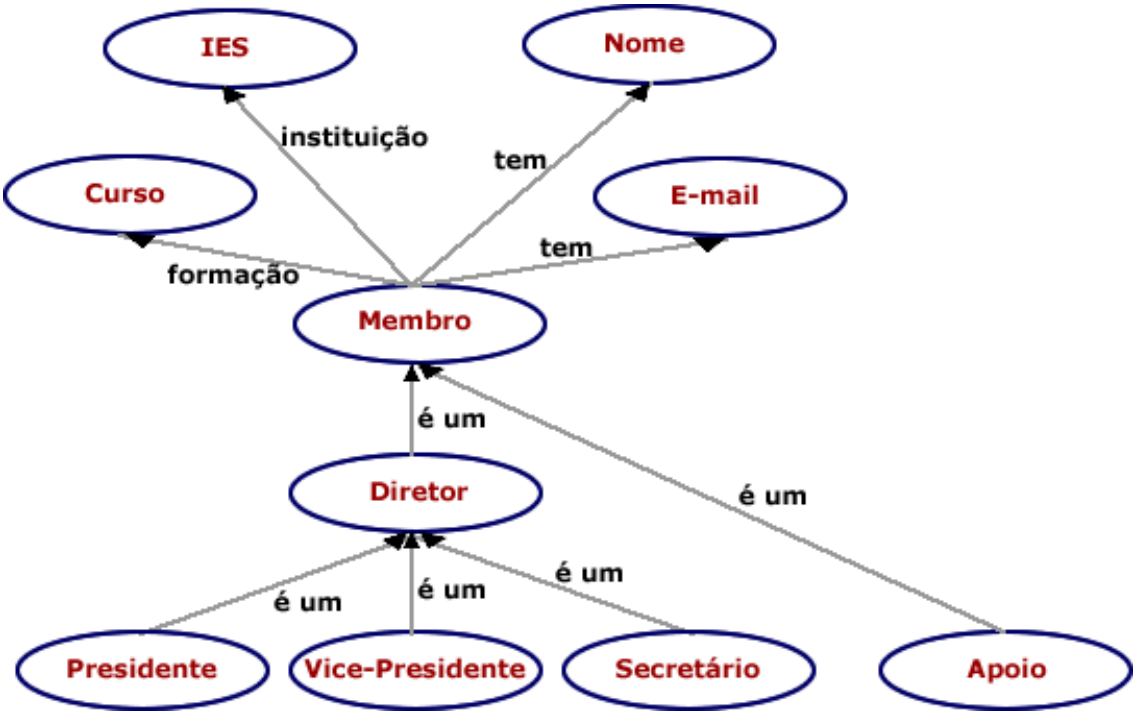


Figura 2-1 Exemplo de representação do conhecimento por rede semântica

Na rede da Figura 2-1, os arcos "formação" e "instituição" podem ser considerados como sendo específicos do domínio em questão e chamados de *traços.* Mas se a análise e o desenvolvimento sobre a rede forem feitos de forma minuciosa e com refinamentos sucessivos pode-se perceber que em grande, parte estas relações podem ser transformadas em relações compostas de "é um" e "tem". E assim tem-se, por exemplo:

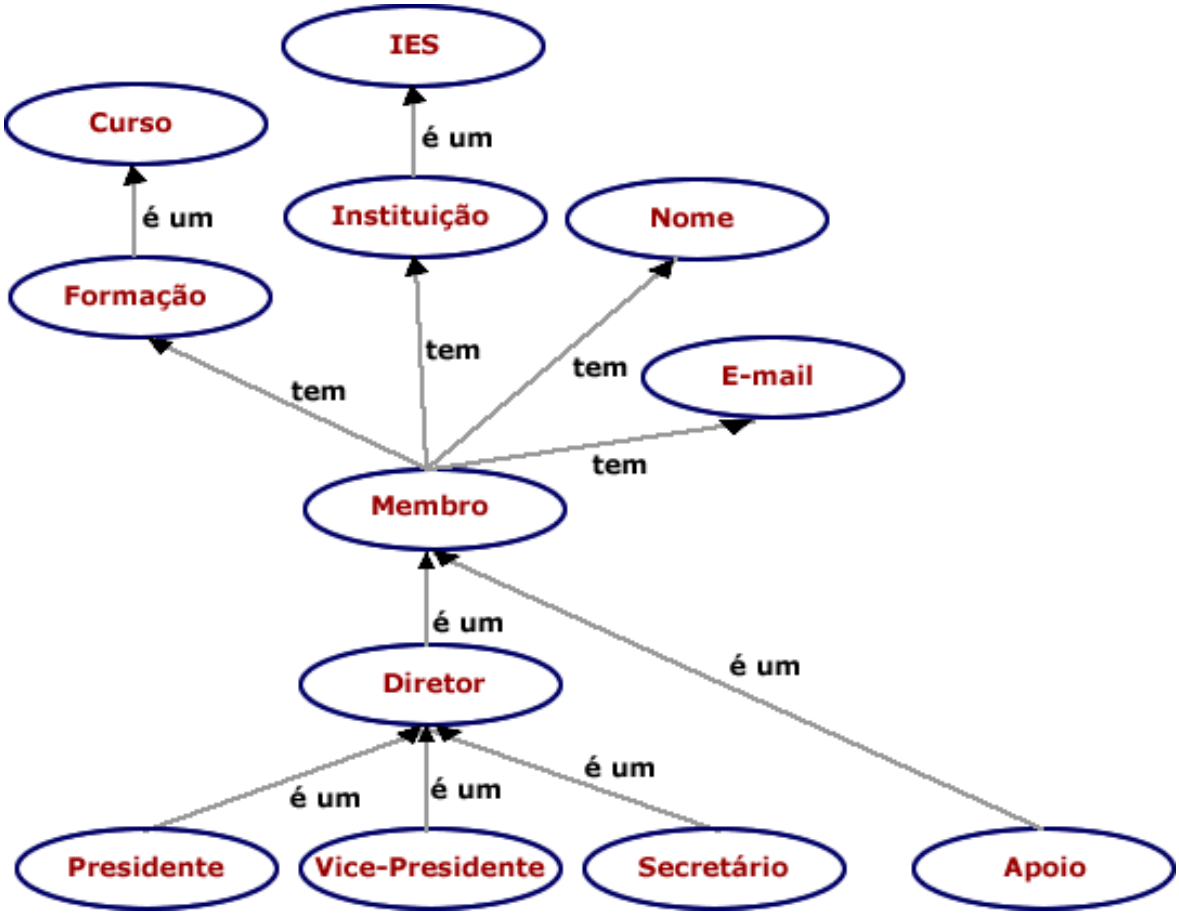


Figura 2-2 Rede semântica refinada

A base de conhecimento da rede então são os nós e as ligações. Uma rede semântica, apesar de menos expressiva, pode ser mapeada em uma representação com mais formalismo, como a lógica de primeira ordem onde os nós representarão termos e os arcos, as relações. São utilizadas em explicações, inferência sobre a herança e busca do completo conhecimento que se tem sobre um nó. A busca por casamento de padrões pode ser feita a partir de um arco ou nó para frente ou para trás através das ligações.

Podemos responder algumas perguntas utilizando inferências simples de classificação e pertença: Por exemplo, O secretário tem e-mail? Seguindo o diagrama, descobrimos que *o* secretário é um diretor, que diretor é um membro e que membro tem e-mail. Em domínios onde se aplicam múltiplas heranças, a herança de propriedades e tratamento de exceções pode ser um grande problema e pode gerar inferências inválidas. Iremos tratar estes e vários outros problemas relacionados ao mapeamento no Capítulo 5.

2.4 Frames

Em geral, um quadro consiste em um conjunto de atributos (*slots)* que, através de seus valores, descrevem as características do objeto representado pelo quadro. Os valores atribuídos a estes atributos podem ser outros quadros, criando uma rede de dependências entre os quadros. Os quadros são também organizados em uma hierarquia de especialização, criando uma outra dimensão de dependência entre eles. Os atributos também apresentam propriedades, que dizem respeito ao tipo de valores e às restrições de número que podem ser associados a cada atributo. Essas propriedades são chamadas *facetas*. Podem estar definindo um domínio possível, um valor único ou um valor padrão, caso não haja informação sobre tal *slot.* Vejamos na Figura 2-3, uma estrutura básica para o *frame* da entidade membro e na Figura 2-4, o conhecimento que anteriormente havia sido representado em redes semânticas.

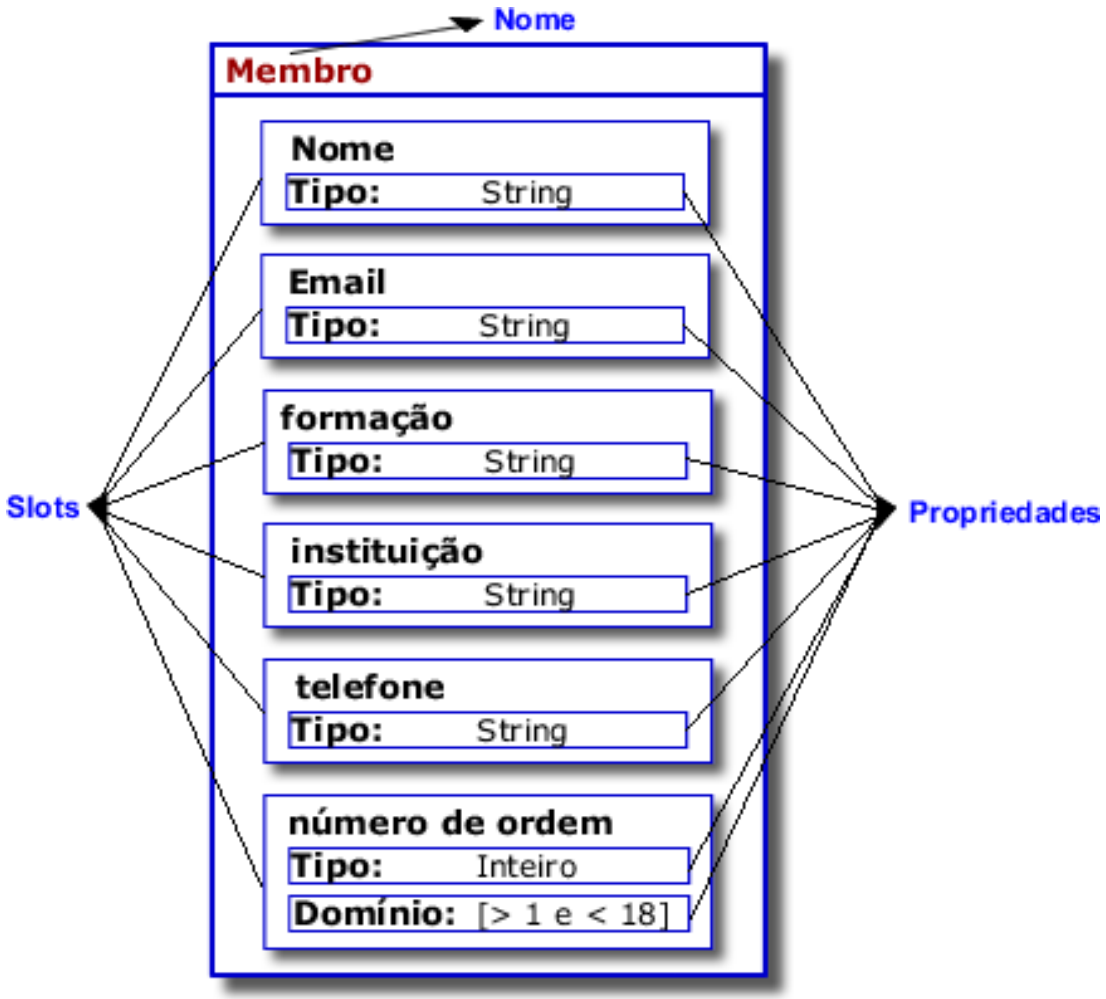


Figura 2-3 *Frame* básico para a entidade Membro (Adaptado de SURGERI, 2006)

Definimos o tipo "String" para algumas "características" como instituição e formação uma vez que os *frame*s exigem que tenhamos estas definições. Outra abordagem poderia

definir o tipo “IES” para instituição como uma coleção de instituições possíveis e o tipo “Curso”, para a característica formação e posteriormente definir esta classe curso com outros detalhes se necessário. Várias maneiras de ver o problema podem tornar confuso o que é detalhe de *design* de programação e o que é detalhe conceitual e de representação de conhecimento.

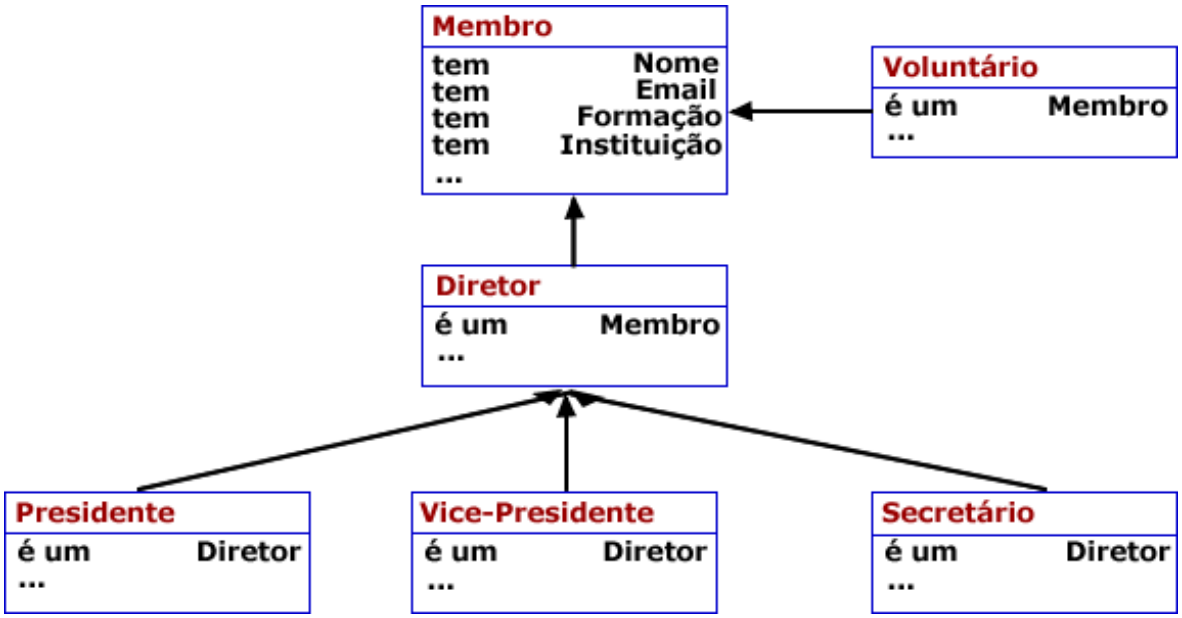


Figura 2-4 Exemplo de representação do conhecimento por *frames*

Observam-se bastantes semelhanças nas estruturas dos *frames* e das redes semânticas, mas diferem no que representam. Quer dizer, enquanto cadeias semânticas representam objetos simples, um sistema de *frames* pode representar objetos complexos. Algumas idéias fundamentais são compartilhadas por estes sistemas. Uma dessas idéias é o conceito de herança de propriedades, o que permite a especificação de propriedades de uma classe de objetos através da declaração de que esta classe é uma subclasse de outra que goza da propriedade em questão. Cada *frame* possui ao menos um *frame* hierarquicamente superior e, portanto, constitui uma base com mecanismo de herança, como dito anteriormente.

Outra característica importante aos *frames* é a possibilidade de se anexar procedimentos ao *frame* ou a *slots*, também chamados de *“gatilhos”* e que são disparados através de consultas ou atualizações a *slots* ou *frames* do sistema como, por exemplo, quando valor é requisitado, lido ou modificado para um dado *slot*, executa-se uma tarefa ou uma verificação.

Algumas limitações como falta de semântica formal e a falta de noção de encapsulamento e componentes também serão tratados no Capítulo 5, mas já se percebe a partir dos *frame*s, uma similaridade com características precursoras da linguagem orientada a objetos.

2.5 Orientação a Objetos

Relacionado à programação orientada a objetos, linguagem orientada a objetos ou paradigma de programação orientado a objetos, encontra-se algumas versões para seu surgimento como o mais popular sendo: “uma extensão quase que natural para a Programação Modular” ou “veio da linguagem SIMULA” (LEITE, RAHAL Jr, 2002), onde a SIMULA-68 foi a primeira ocorrência dos conceitos de programação orientada a objetos. Depois então, apareceu a linguagem Smalltalk da Empresa Xerox que contribuiu fortemente para a popularização do que logo à frente veio a ser chamado de Paradigma de Programação Orientado a Objetos.

Mas essas colocações remontam um outro caminho e outras influências que não estão no foco deste trabalho. Olharemos aqui, para o paradigma de programação orientado a objetos como uma evolução dos conceitos e as idéias que foram inicialmente abordadas com *frame*s, para de alguma forma representar conhecimento e onde slots ou mesmo *frame*s podiam ter procedimentos associados. O paradigma prega que todo o mundo a ser modelado pode ser representado por um conjunto de objetos, entes ativos e que relacionam entre si. Possui uma abordagem muito próxima das iterações observadas no mundo real. É como se de um minuto para o outro, as entidades ou objetos especificadas no modelo Entidade-Relacionamento tomassem vida com os métodos dos objetos em OO, precursoramente abordado pelos *frame*s e os relacionamentos começassem realmente a acontecer devido à comunicação entre objetos.

Várias são as vantagens desse paradigma frente às várias outras metodologias, principalmente no que diz respeito a desenvolvimento de sistemas de informação e sistemas onde as entidades a serem representadas são equivalentes as entidades do mundo real. A partir de agora, tem-se um paradigma para análise, planejamento, projeto e desenvolvimento de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos. Não serão aqui esgotados os assuntos relacionados à orientação a objetos que demonstram e creditam sua alta capacidade de representar conhecimento, muito menos seu trajeto de evolução, mas sim os pontos principais e avanços tomados em direção a uma especificação mais semântica do domínio e poder de integração de aplicações deste tipo.

Lembrando dos *frame*s, as facetas ou valores padrões, valores possíveis, intervalos etc. “evoluíram para o conceito de classe” em OO onde objetos, ou seja, instâncias de uma classe, que compartilham as mesmas características pertencem a uma mesma classe. Atributos são características dos objetos. A um dado instante, o conjunto de atributos de um objeto define seu estado, como objetos do mundo real. Métodos são funções e procedimentos que podem ser executados por um objeto. Através desses métodos os objetos podem relacionar e

comunicar-se entre si, como descrito nos relacionamentos de um diagrama Entidade-Relacionamento e mudar de estado. Mas as inovações mais importantes para o sucesso da OO e de impacto para as definições semânticas foram: a sobrecarga, polimorfismo, encapsulamento, pacotes e interface.

A sobrecarga é empregada na utilização de um mesmo nome para métodos com operações distintas. São geralmente diferenciados por suas assinaturas, ou seja, quantidade, tipo e nome dos parâmetros componentes de entrada e saída do método. Em especificações semânticas não é aconselhável este tipo de prática, pois se deseja sempre uma definição única e exata para um dado termo, evitando ao máximo interpretações dúbias.

Polimorfismo é o princípio pelo quais duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura, mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse. A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do mecanismo de ligação tardia. No caso de polimorfismo, é necessário que os métodos tenham exatamente a mesma identificação, sendo utilizado o mecanismo de redefinição de métodos. Esse mecanismo de redefinição não deve ser confundido com o mecanismo de sobrecarga de métodos. Verifica-se também, que pode ou não ocorrer problemas semânticos uma vez que os métodos executados podem processar atividades semanticamente idênticas com entradas e/ou saídas diferentes ou processar atividades diferentes semanticamente de acordo com o objeto sendo observado na execução.

O encapsulamento é um mecanismo que roga por separar as partes do programa o mais isoladas possível e que provê proteção nos aspectos internos de um objeto. Esta proteção consiste em se usar modificadores de acesso mais restritivos sobre os atributos definidos na classe. Através de interfaces o objeto “publica” os métodos que são de real interesse no relacionamento com outros objetos do sistema. Afeta mais as considerações de implementação dos sistemas do que aspectos semânticos, e a mesma idéia pode ser aplicada nas especificações semânticas, na decisão de declarar ou não conhecimento sobre um determinado aspecto de uma entidade.

Pacotes são aglomerados de classes que geralmente realizam tarefas em prol de atender a uma atividade de mais alto nível de abstração. Facilita a modularização do problema a ser solucionado e proporciona um artefato para reutilização de código. Caso a especificação semântica seja de grande proporção pode-se do mesmo modo definir pacotes de definições de interesses comuns.

Uma interface, em ciência da computação pode ser definida como uma fronteira que define a forma de comunicação entre duas entidades. É um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface. Como o encapsulamento, interfaces possuem maior impacto sobre as tarefas de projeto e programação. Mas especificações semânticas de interfaces podem auxiliar na busca e o casamento de objetos para a composição da comunicação por uma dada interface do mesmo modo que nos dias atuais, especificações através de ontologias de *Webservices* têm ganhado espaço na descrição e descoberta de serviços Web, tema à qual pode ser mais bem observado em GAMA (2008).

A aceitação, magnitude do movimento ao redor do paradigma e a invasão no mercado de uma enxurrada de linguagens, plataformas, *frameworks*, máquinas virtuais, bibliotecas e uma avalanche de novos recursos promovendo a OO fizeram com que, em curto espaço de tempo, todos os aspectos estruturais estivessem sólidos, com padrões de projetos bem definidos, ótimas bibliotecas de suporte, técnicas e metodologias eficientes de desenvolvimento e comunicação entre outros. Por outro lado, volta-se no problema de ser uma linguagem de programação e ter regras formais também rígidas, ou seja, seguindo a evolução que se estava tendo graficamente e visualmente: redes semânticas, *frames*, diagramas E-R, regride-se a uma representação textual, de maior expressividade, mas bem formalizada.

Ainda faltando uma representação mais amigável para o conhecimento sendo descrito, eis que surge a UML que entre seus diagramas, temos o diagrama de classe, à qual pode-se dizer “modela” o respectivo sistema de classes, objetos e relacionamentos do mundo sendo especificado no modelo OO. Hoje, algumas ferramentas de edição de diagramas UML, dão suportes à geração de código digamos iniciais, protótipos em linguagens OO como Java. Às vezes com algumas limitações, mas já adiantam bastante o trabalho de codificação. Essa característica é fator importante para a facilitação da representação visual e tradução em código textual formal. Por isso fez-se necessário esta exposição sobre a orientação a objetos para tratarmos mais intimamente do diagrama de classes UML na próxima seção. Para mostrar a possibilidade de representação por OO vejamos então o exemplo de representação do conhecimento descrito anteriormente, e que foi representado em diagrama E-R agora em uma linguagem de programação orientada a objetos, neste caso, Java. A Figura 2-5, mostra uma parte da tela da IDE Netbeans 6.1 apresentando o código em Java para a entidade “Membro” à qual podemos observar a aba “Projetos”, em destaque verde onde estão os diretórios e arquivos do projeto dispostos hierarquicamente, a aba “Membro.java”, em

destaque vermelho que mostra o código em linguagem Java para a entidade “Membro” e a aba “Membro.java - Navegador”, em destaque de azul onde visualizamos os atributos da entidade e seus respectivos tipos de dados.

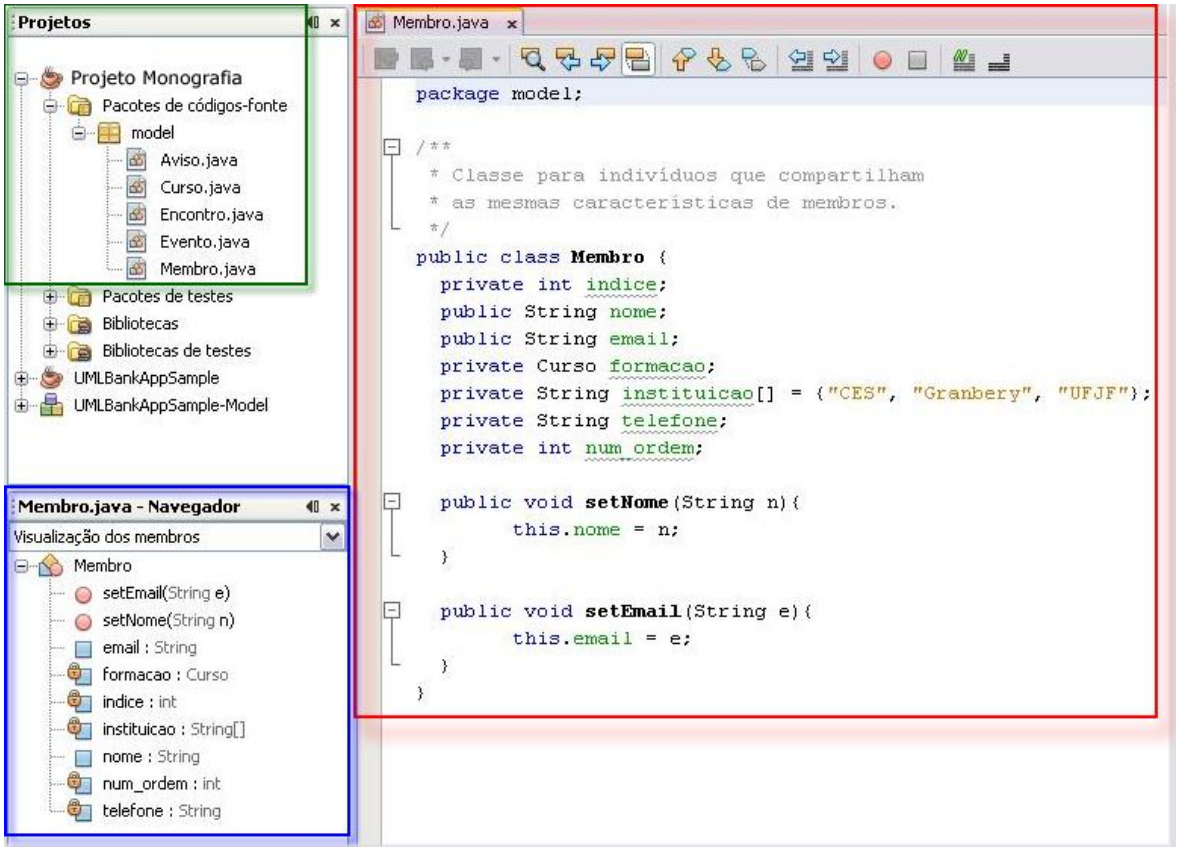


Figura 2-5 Exemplo de representação do conhecimento na linguagem orientada a objetos Java

Observe que foram definidos dois métodos associados à entidade “Membro” e que alteram seu estado: **`setNome(String n)`** e **`setEmail(String e)`**. Semanticamente podemos dizer que **`setNome`** atualiza a propriedade nome de objetos da classe “Membro” com o valor n à qual deve ser um elemento do tipo String, **`setEmail`** atualiza a propriedade e-mail com o valor à qual também deve ser um elemento do tipo String. Estes métodos então podem ser executados a partir de uma instância da classe “Membro”. Nota-se também que propriedades podem ser outras classes criando formação de objetos complexos. Todas as entidades foram definidas dentro do pacote “*model*”. Pacotes podem ser reutilizados posteriormente utilizando mecanismo de importação.

Vários recursos e facilidades de IDEs como Netbeans[2] ou Eclipse[3] aceleram drasticamente as tarefas de codificação de sistemas de grande porte e com relações e compromissos complexos. Mais uma vez deve-se considerar às preocupações da

[2] http://www.netbeans.org
[3] http://www.eclipse.org

representação do conhecimento através de ferramentas com maiores recursos gráficos a fim de traduzi-las posteriormente em linguagens de programação que irão executar as atividades necessárias ao funcionamento destes sistemas e que tem grande poder de expressão semântica.

Até o presente, pode-se dizer que estas linguagens são linguagens orientadas a objetos. Outros aspectos observados em padrões de projetos como em REIS (2007) também estão sendo analisados, soluções sendo construídas e outras metodologias e até paradigmas, como programação orientada a aspectos (LEITE, 2002), por exemplo, surgindo para evoluir na concepção e manutenção de sistemas cada vez mais inteligentes e que tendem a aumentar o índice MIQ *Machine Intelligence Quotient*, mas os fundamentos básicos para a representação do conhecimento, atualmente, se definem nos conceitos da orientação a objetos e estas novas soluções somente acrescentam melhorias na visão de implementação e não na visão conceitual do sistema. Na IDE Netbeans, ao clicar sobre algum arquivo na aba "Projetos", este arquivo é aberto na aba mais à direita para a edição. E daí, pode-se editar os outros arquivos do exemplo: Aviso.java, Curso.java, Encontro.java, Evento.java e IES.java. Vejamos o código: Abaixo de cada faixa azul está o início do código de cada entidade (Figura 2-6).

Na definição de uma classe a palavra *extends* seguida de um nome de uma outra classe, por exemplo, a classe "Encontro" *extends* "Evento", está explicitando que a classe "Encontro" é um evento, mas possui características específicas. Do mesmo modo ocorre com a classe "Aviso". Visto por outro lado, temos que evento é um agrupamento das características semelhantes entre encontros e avisos. A propriedade "instituição" possui um conjunto de valores possíveis que no caso da linguagem Java pode ser especificado de várias tipos de estruturas de dados como vetores no exemplo, listas, pilhas e outros objetos. As palavras "*public*" e "*private*" contribuem nos tratamentos de acesso a um atributo ou recurso do objeto. A classe "Curso" foi criada com intuito de mostrar outra maneira de representar uma característica composta e armazenar informações de tal entidade expandindo o conhecimento representado.

```java
/**
 * Classe que agrega valores iguais para encontros e avisos.
 */
public class Evento {
    private int indice;
    public String texto;
}
```

```java
/**
 * Definições para os eventos do tipo Encontro.
 */
public class Encontro extends Evento {
    public String titulo;
    public String data;
}
```

```java
/**
 * Classe para os Avisos
 */
public class Aviso extends Evento {
    private String data_expiracao;
}
```

```java
/**
 * Classe para as definições da entidade curso.
 */
public class Curso {
     public String nome;
     public String area;
     public int duracao;
}
```

Figura 2-6 Exemplo de código Java para representar as entidades do exemplo

No exemplo descrito no diagrama E-R, ainda tem-se que efetivar o relacionamento "Realiza" que ocorreria, por exemplo: a partir de um terminal, um usuário irá inserir os dados necessários através de uma interface gráfica para o registro de um evento. Pode-se então criar uma classe em OO que terá um método para inserir estes dados na tabela "freqüência", que pode ser observada na seção anterior sobre o MER. Em Java, esta classe e um trecho de código para a inserção poderiam ser declarados desta forma:

```java
package model;

import java.util.Date;

/**
 * Classe para os métodos que manipulam
 * a tabela 'frequencia' na base de dados.
 */
public class Frequencia {

    public Frequencia() {
    }

    public boolean insert(int indMembro, int indEvento, String tipoEvento){
        // Declaração de variáveis
        boolean ok = false;
        Date data = new Date();

        /*
         * Aqui vai o código para efetuar as ações
         * de tratamento e persistência necessárias.
         * Se tudo ocorrer corretamente, ok = true;
         */
        return ok;
    }
}
```

Manipulação da Tabela

```java
// Criamos o objeto frequencia para manipularmos a
// tabela no banco de dados, utilizando seus métodos.
Frequencia freq = new Frequencia();

// Inserimos um evento utilizando o objeto.
if(freq.insert(132,35,"Encontro") == true){
    // Evento inserido com sucesso !
}else{
    // Erro na execussão.
}
```

Figura 2-7 Classe freqüência para manipular a tabela na base de dados e trecho de código de exemplo

Entre várias outras definições para representação do conhecimento essas podem ser descritas com linguagem OO. Muito já se pode fazer com linguagens OO e as possibilidades são intermináveis, mas o exposto aqui foi o necessário e suficiente para compreender a representação do conhecimento via paradigma orientado a objetos. Detalhes de implementações e soluções complexas irão depender das características de cada projeto e capacidade de integração e resultados de cada grupo de *stakeholders*.

Na seção **2.7** será tratada a representação de classes em UML à qual irá efetivar a representação gráfica do domínio, definida de acordo com os conceitos de OO.

2.6 MER – Modelo Entidade-Relacionamento

De acordo com COUGO (1999), o MER, descreve o mundo como: “cheio de coisas que possuem características próprias e que se relacionam entre si”. Essas coisas podem ser objetos concretos ou abstratos, pessoas, conceitos, atividades, eventos, etc., como visto anteriormente, uma definição bem próxima da representação do conhecimento. Por possuir o diagrama entidade-relacionamento boa visualização, como sendo sua forma de representação e de fácil entendimento, este modelo pode ser alvo de muitas pesquisas e aplicações que exploram a representação de fatos que ocorrem no sistema e por possuir tradução direta para a consolidação dos dados em SGBDs.

De acordo com a Figura 2-8 pode-se elucidar os principais elementos deste modelo. Estes símbolos que descrevem os elementos não possuem definição própria, dada como senso comum. Podem variar um pouco suas representações, mas foram suficientes para representar muitas bases de dados em domínios razoavelmente simples. Com o avanço das aplicações, foram necessários mecanismos para representar as relações semânticas de generalização e especialização, muito utilizadas em sistemas grandes e complexos. Daí, no diagrama utiliza-se um triângulo, disposto como que apontando para o conjunto de entidades de nível mais baixo. Há pessoas que utilizam o triângulo em sentido inverso.

Figura 2-8 Elementos básicos do diagrama Entidade-Relacionamento

Para exemplificar, será esboçado um caso em que um membro freqüenta aos encontros semanais realizados e que confirma a leitura dos avisos emitidos pela diretoria. São armazenados tanto a ata do encontro, os avisos emitidos quanto a freqüência dos membros aos eventos.

A Figura 2-9 mostra este exemplo em um diagrama Entidade-Relacionamento.

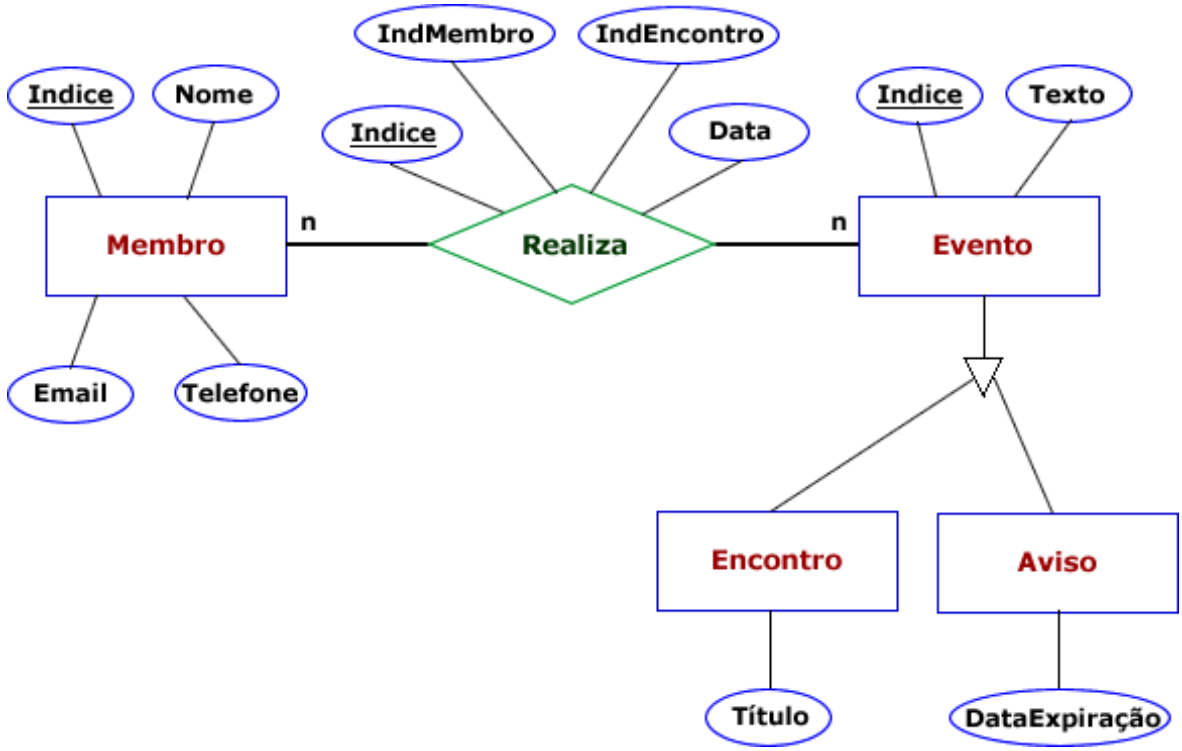


Figura 2-9 Exemplo de diagrama Entidade-Relacionamento

Analisando o diagrama podem-se observar várias informações como: - atributos sobre as entidades “Membro”, “Evento”, “Encontro” e “Aviso” representam conhecimento que se tem sobre essas entidades; - o relacionamento “Realiza” irá se tornar uma tabela no banco de dados, com nome “freqüência”, por exemplo, e armazenará a data do fato além das chaves para o membro e o encontro, a qual é chamado de entidade associativa; - a cardinalidade n pra n nos indica que um membro pode freqüentar vários eventos e que um evento pode ter a presença de vários membros; - O triângulo entre entidades nos informa que a entidade “Evento” agrega valores tanto de encontros quanto de avisos, ou seja, que encontros e avisos possuem características diferentes e específicas, que neste caso, título é uma característica somente de eventos do “tipo” encontro e data de expiração é uma característica somente de eventos do “tipo” aviso.

Outros relacionamentos de destaque são: o relacionamento reflexivo, como: entidade “Funcionário” coordena entidade “Funcionário” e o relacionamento ternário, ou seja, relação entre três entidades.

2.7 UML – *Unified Modeling Language*

A UML de acordo com o site da OMG (2008): "ajuda a especificar, visualizar, e documentar modelos de sistemas de software, incluindo a sua estrutura e concepção, de uma forma que satisfaça todos estes requisitos. (Você pode usar UML para modelagem de negócio e outros sistemas que não sejam software.)". Estas características estão unidas à facilidade de compreensão e modelagem uma vez que UML é uma linguagem gráfica. Na versão atual, a UML 2.0 define treze tipos de diagramas, divididos em três categorias: seis destes tipos de diagramas representam estruturas estáticas das aplicações; três representam tipos gerais de comportamento; e quatro representam diferentes aspectos das interações. É notório que estes diagramas têm objetivo de representar, compartilhar e transmitir conhecimento, comumente em grupos de desenvolvedores de softwares.

Visto a facilidade de modelagem e advento da UML e todo o impacto causado pela Web Semântica logo iniciativas foram lançadas com foco e propósito de integração entre essas abordagens no desenvolvimento de ontologias. Vários aspectos, que eram delimitadores para esta integração foram solvidos, mas que em quase todos os casos retornam a soluções com utilização de padrões ou outras linguagens de caráter textual, com regras bem definidas e com lógica específica de estruturação em que um desenvolvedor com conhecimentos específicos da linguagem e fundamentos lógicos precisos é requerido, como é o caso da linguagem OCL (*Object Constraint Language*) que foi desenvolvida em forma de expressões para impor restrições a objetos da modelagem UML e XMI (*XML Metadata Interchange*) que é um padrão para troca de metadados via XML que pode ser usado para qualquer meta modelo cujos metadados podem ser expressos em MOF (*Meta-Object Facility*) como UML. A utilização mais comum de XMI é como um formato para intercâmbio justamente de modelos UML, embora possa também ser usado para a serialização de modelos de outras linguagens.

O uso da linguagem OCL se torna de grande importância uma vez que a UML, não provê todos os aspectos relevantes para a especificação de sistemas complexos e a utilização de XMI se torna imprescindível para a tradução da codificação do conhecimento em codificação semântica, como podemos observar no trabalho desenvolvido por VICTORETTE (2008), à qual possui grandes similaridades de propósitos com este estudo. Seu trabalho descreve a criação de uma ontologia utilizando uma ferramenta UML para modelar o domínio e exportar para XMI. Daí usa-se o conversor UMLtoOWL, que na verdade é um processador XSLT, que produz um arquivo OWL que pode ser importado numa ferramenta de manipulação de ontologias como Protégé. Na verdade o processo num todo e principalmente

em sistemas com relações muitos complexas sofre algumas limitações e pontos ainda não solucionáveis, mas este modelo merece consideração especial dado a grande evolução na direção concepção e manutenção das ontologias que irão especificar os sistemas Web inteligentes do futuro.

No Capítulo 5, específico sobre o mapeamento das representações, vai-se utilizar como artefato base e detalhar o método UML2ONTO de VICTORETTE (2008) com os procedimentos[4] para a geração de ontologias OWL a partir da modelagem UML, permeando e incorporando outras boas práticas de especificação e adição semântica observadas durante todo o estudo deste trabalho e para o desenvolvimento de portais Web semânticos. De imediato, retorna-se aqui a atenção às várias maneiras de representação do conhecimento e observando-se a Figura 2-10, tem-se o exemplo anterior em um diagrama de classes UML, onde podem ser observados os elementos e as características, recursos e potencialidades desta abordagem.

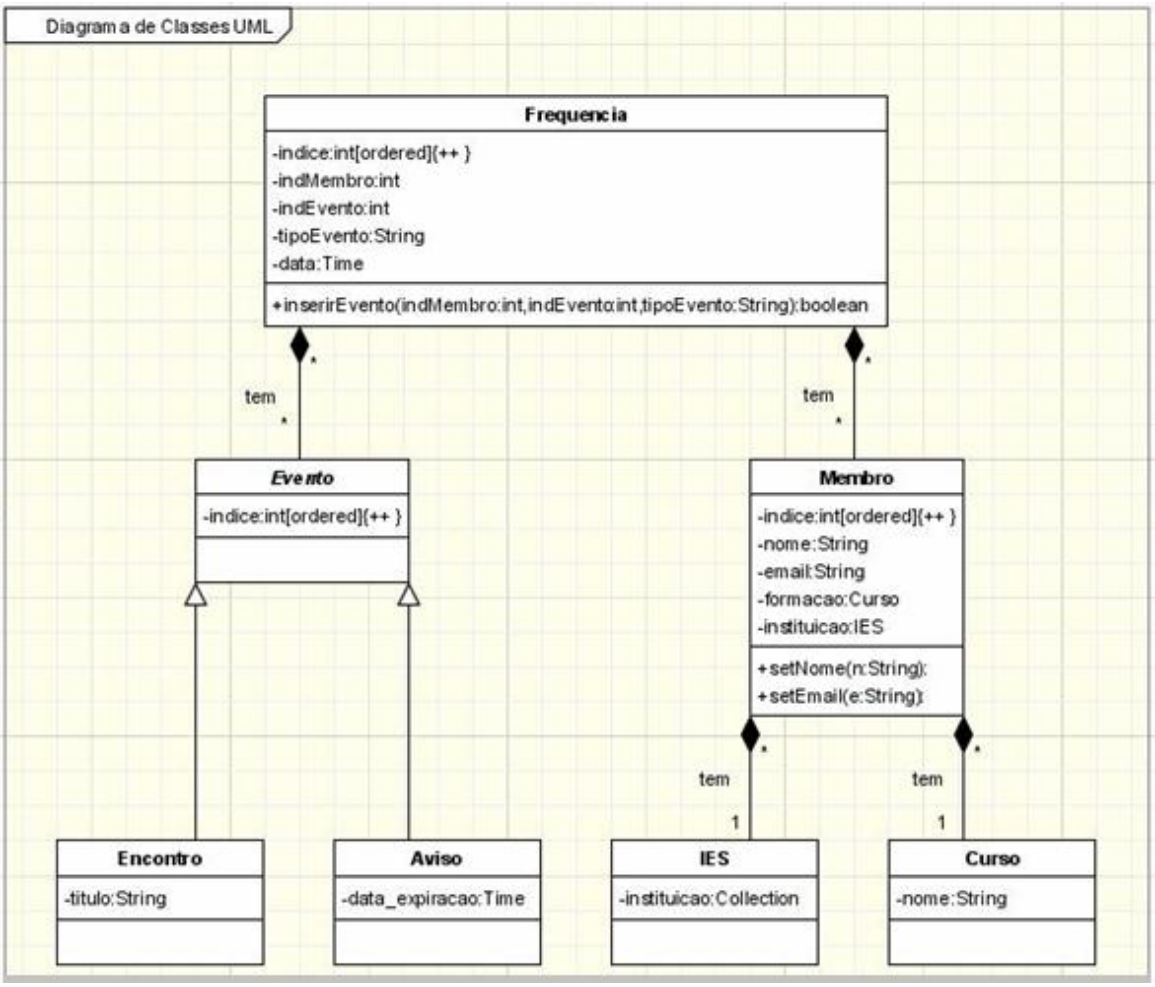


Figura 2-10 Exemplo de representação de conhecimento em diagrama de classe UML

[4] http://wwwsfu.ca/~dgasevic/

Também bastante similar aos modelos com *frame*s, principalmente no aspecto visual, o diagrama UML define tipos de dados e procedimentos aos atributos das classes, que são chamados de facetas de um slot do quadro no modelo de *frame*s. Já as notações para composição e agregação são bastante semelhantes ao MER. Mas UML leva vantagem sobre outras abordagens, pois representa o conhecimento de acordo com o paradigma orientado a objetos a qual se assemelha bastante com o mundo real, define outros mecanismos de associação, especifica claramente a cardinalidade entre as classes, tem ótima aceitação e popularidade além de excelentes iniciativas no desenvolvimento e aperfeiçoamento de ferramentas para produção e suporte. Funcionalidades mais refinadas e complexas como análise do modelo, aplicação de *profiles* e integração entre diagramas além de recursos como exportação para imagens do tipo GIF ou JPEG ou exportação para arquivos de intercâmbio como XMI ainda são privilégio em algumas ferramentas proprietárias, mas ferramentas gratuitas já oferecem muitas outras capacidades que vieram para aperfeiçoar e/ou solucionar os obstáculos encontrados nas abordagens anteriores, ou seja, UML é uma abordagem que está à frente na evolução das representações que ponderam notação gráfica e expressão formal.

Diagramas de classes UML modelam classes e suas dependências. Então se deve sempre tomar cuidados para não confundir a modelagem conceitual da modelagem OO de *design*, projeto e implantação, isto pode ocorrer devido as ferramentas terem sido primeiramente desenvolvidas para paradigma de programação orientada a objetos e não desenvolvimento de ontologias. Seus elementos principais e que podemos ver na figura 10 são:

- **Classes**. Representam termos, conceitos, classes e conjuntos do modelo conceitual. São compostas de seus atributos e métodos.

- **Atributos de uma classe**. Juntos definem as características da entidade.

- **Métodos de uma classe**. Procedimentos a qual as instâncias desta classe executam. Geralmente alteram o estado do próprio objeto, mas podem também modificar o estado de um ou vários objetos, consequentemente em todo o cenário do sistema. Ferramentas modernas oferecem mecanismos de definição de parâmetros de entrada e saída, visibilidade, escopo e tipos baseados em linguagem de programação, como neste caso, à qual os tipos de dados foram especificados para a linguagem Java.

- **Associações**. As mais comuns são composição e agregação, mas de acordo com a ferramenta oferecem um leque de opções como também algumas distinções, por exemplo: associação e associação direta. No exemplo de amostra, temos as associações de "composição" entre Membro, Curso e IES, i.e. membros tem curso e instituição de ensino, e entre Freqüência, Evento e Membro, i.e. Freqüência tem evento e tem membro. Também pode ser descrever seus nomes e anexar regras.
- **Herança de propriedades**. Ao criar a especialização de "Encontro" e "Aviso", especificando também que "Evento" é uma classe abstrata, esta se repassando as propriedades e procedimentos de "Evento" para suas subclasses. Na ferramenta utilizada, a indicação que "Evento" é uma classe abstrata é descrito com o nome da classe em itálico e a especialização pela ligação entre as classes com o triângulo apontando para a super classe.
- **Cardinalidade**. Nas associações descritas no exemplo pode-se observar que a descrição de cardinalidade nos diz que um membro pode ter apenas um curso registrado em seu nome, mas um curso pode estar registrado no cadastro de vários membros do mesmo modo ocorre com as IES. Quanto à freqüência, observa-se que vários eventos podem conter nas freqüências e pode haver várias freqüências para um mesmo evento do mesmo modo que para os membros.

Entre essas e outras propriedades do modelo UML para a representação do conhecimento e o poder das ferramentas que dão suporte a este propósito, deve-se novamente frisar a importância da capacidade de exportar os diagramas para linguagem de programação como Java e linguagem de intercâmbio com XMI.

2.8 Tesauros

Como outros conceitos no âmbito da representação do conhecimento e estudos que se originam em computação ou psicologia, tesauros não possuem definição dada como consenso e pode variar de acordo com a utilização a qual estão sendo abordados. Listas de significados, dicionários de idéias afins, livros de sinônimos e antônimos e linguagem documentária são algumas das inúmeras definições encontradas na literatura. Em Tecnologia da Informação um tesauros representa uma base de dados ou uma lista semanticamente ortogonal de chaves de busca. Em Inteligência Artificial, algumas vezes pode ser referenciado como uma ontologia, mas diferenças significativas podem ser observadas, como no estudo "Semelhanças e diferenças entre Tesauros e Ontologias" de Rodrigo de Sales e Ligia Café (2008).

Importante ressaltar aqui que este modelo visa definir e esclarecer o significado de alguns termos ou conceitos do domínio, a fim de deixá-los sem ambigüidade no contexto considerado, ao passo que ontologias definem os conceitos e suas relações, as funções dos termos, deixando um tanto vago seu significado. Algumas técnicas para construção de tesauros podem ir especificando termos de acordo com que eles vão aparecendo nas definições, outros podem organizá-los por uma ordem alfabética ou hierárquica.

Tesauros estão sendo largamente construídos em sistemas Web onde se devem passar esclarecimentos ou detalhes de definições, regras a serem cumpridas, estatutos entre outros materiais de caráter informacional. O W3C tem demonstrado apoio em prol de publicações de tesauros na Web que utilizam vocabulário do núcleo SKOS (*Simple Knowledge Organization System*), *Dublin Core*, FOAF e OWL além de serem codificados em linguagem RDF, ou seja, é uma iniciativa para criação, adição e aumento de semântica na Web.

Para exemplificar, vê-se na Figura 2-11 a definição do termo "Encontro" para o exemplo considerado anteriormente, utilizando o vocabulário do núcleo SKOS.

Figura 2-11 Exemplo de definição de termo em um tesauro com vocabulário do núcleo SKOS

Observa-se que pode se tornar uma tarefa um tanto quanto extensa e complexa, uma vez que se tem necessidade de definir termos para que se compreenda outro termo, como é o caso de Evento, à qual se não for especificado, não podemos classificar a especificidade de Encontro perante um Evento. Mas ainda com poucas ferramentas para geração, visualização e manutenção de gráficos RDF este modelo também se torna alvo de estudo e com possibilidades animadoras, uma vez que tornaria a tarefa de criação e atualização mais intuitiva e com interfaces mais amigáveis, sendo codificada em linguagem formal e padrão RDF, recomendado pela W3C.

2.9 Mapas Conceituais

De um modo geral, mapas conceituais, ou mapas de conceitos, são apenas diagramas indicando relações entre conceitos, ou entre palavras que usamos para representar conceitos. Apresentam-se como um grafo direcionado com nós que representam conceito e ligações que representam relações. Pode ser entendido como uma representação visual utilizada para partilhar significados, pois explicita como o autor entende as relações entre os conceitos enunciados (TAVARES, 2007). Existe uma grande variedade de tipos de mapas disponíveis, que foram imaginados e construídos pelas mais diversas razões. Alguns são preferidos pela facilidade de elaboração (tipo aranha), pela clareza que explicita processos (tipo fluxograma), pela ênfase no produto que descreve, ou pela hierarquia conceitual que apresenta. Quando se deseja otimizar um determinado processo, a utilização do mapa tipo fluxograma é a representação mais adequada. Esse tipo de mapa deixa claro quais são as confluências e as possíveis opções a serem escolhidas. Ele ainda é extremamente utilizado na elaboração de programas de computador, quando se deseja construir um algoritmo eficiente para determinada função. No entanto, o único tipo de mapa que explicitamente utiliza uma teoria cognitiva em sua elaboração é o mapa hierárquico do tipo proposto por NOVAK e GOWIN (1999).

Embora normalmente tenham uma organização hierárquica e, muitas vezes, incluam setas, tais diagramas não devem ser confundidos com organogramas ou diagramas de fluxo, pois não implicam seqüência, temporalidade ou direcionalidade, nem hierarquias organizacionais ou de poder. Mapas conceituais são diagramas de significados, de relações significativas; de hierarquias conceituais, se for o caso. Isso também os diferencia das redes semânticas que não necessariamente se organizam por níveis hierárquicos e não obrigatoriamente incluem apenas conceitos. Mapas conceituais também não devem ser confundidos com mapas mentais que são associacionistas, não se ocupam de relações entre conceitos, incluem coisas que não são conceitos e não estão organizados hierarquicamente. Não devem, igualmente, ser confundidos com quadros sinópticos que são diagramas classificatórios. Mapas conceituais não buscam classificar conceitos, mas sim relacioná-los e hierarquizá-los.

Possuem grande poder de expressão, transmissão clara de conhecimento a humanos e ferramentas computacionais estáveis para criação e manutenção dos mesmos, mas por serem

formados de ligações (relações) livres entre os conceitos, se tornam praticamente impossíveis de serem mapeados em uma linguagem formal para então aplicar raciocínios. Muito há para se considerar a respeito de mapas conceituais bem como estrutura, questão focal (FQ, *Focal Question*), formação de ciclos entre outros aspectos importantes para a sua representação. Neste trabalho, será mostrado somente um exemplo simples para observação, uma vez que este modelo não atende aos propósitos definidos previamente, bem como sua característica informal das composições e visto que ainda não possui codificação bem consolidada e definida para que possa se traduzir para uma linguagem da Web semântica. Estudos futuros de mapeamento podem abordar este modelo utilizando uma codificação XML gerada a partir do mapa construído em ferramentas como o CmapTools[5], mas ainda terão que fazer um tratamento de quebra e uma análise complexa dos predicados além do *parser* para estes arquivos.

Na Figura 2-12 pode-se ver o cenário que está sendo descrito de exemplo com os três tipos de eventos e as relações dos membros com estes. Algumas relações podem parecer óbvias, mas podem estar representado imposições, delimitações, requisitos e até um artefato de armazenamento para futuras recuperações. Por exemplo, temos os predicados: "Tarefas são executadas por Membros" e "Membros devem concluir Tarefas". O segundo predicado induz-nos a pensar que é uma conseqüência do primeiro, mas tarefas podem ter sido registradas para um membro e ele não a concluiu. Daí então o segundo predicado foi necessário como requisito do sistema. Os problemas mais difíceis são ciclos e possibilidade de interpretações diferentes. No mapa da Figura 2-12, pode-se notar que um membro deve ler um aviso, que é emitido por um diretor que é um membro. Aqui verificamos que uma inferência errônea pode concluir que um membro tem privilégios para emitir um aviso, o que não é correto no sistema.

[5] http://cmap.ihmc.us/conceptmap.html

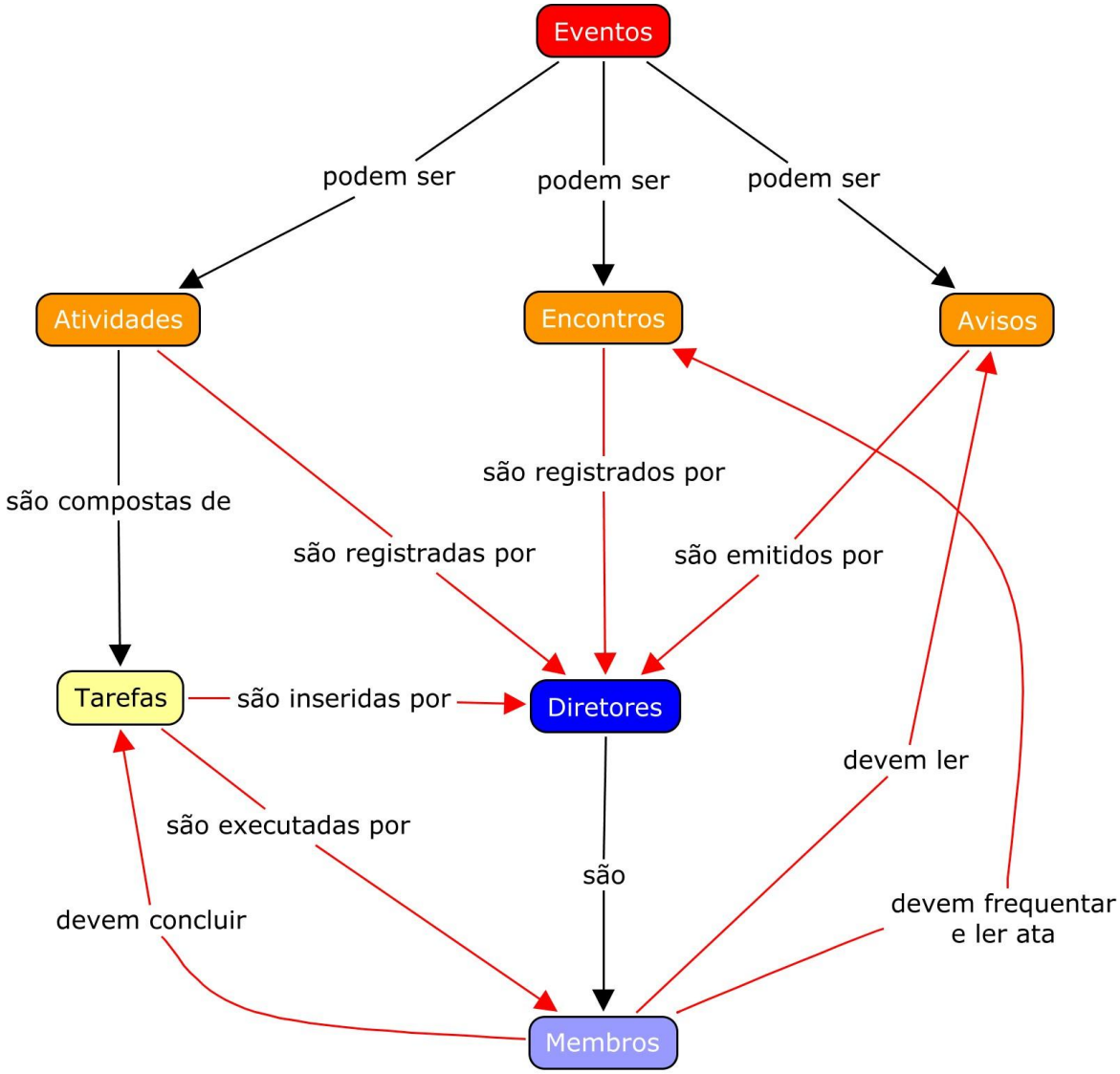


Figura 2-12 Exemplo de representação do conhecimento através de mapa conceitual

Ao contrário do que se espera de uma representação do conhecimento em ambientes corporativos, organizações e grandes instituições estes mapas absorvem e refletem melhor o conhecimento de um indivíduo e não de um grupo de indivíduos que devem entrar em consenso sobre o domínio a ser representado. Estes modelos, principalmente mapas conceituais hierárquicos, se colocam como um instrumento adequado para estruturar o conhecimento que está sendo construído por um aprendiz, assim como uma forma de explicitar o conhecimento de um especialista. Neste trabalho busca-se a representação do conhecimento adquirido e sendo construído, por um grupo que poderá se integrar a outro, que poderá se integrar a outro e assim sucessivamente, e não a explicitação do domínio por um indivíduo somente, uma vez que se busca convergir para a representação semântica que

atualmente se baseia na criação e manutenção de ontologias compartilhadas em grupos e subgrupos.

2.10 Diretório de Arquivos

De acordo com FURGERI (2006) “podemos considerar que a primeira estrutura de representação e organização usada pela Ciência da Computação foram os diretórios. Um diretório corresponde a um local lógico, definido em uma unidade, para armazenar arquivos comuns. Com o advento do sistema operacional *Windows* o diretório passou a ser conhecimento pelo nome de “pasta”, um nome mais sugestivo para usuários comuns”.

Nesta seção vamos tratar do papel dos diretórios para unir características de arquivos, documentos e programas em comum, sua nomeação, renomeação e as implicações para a representação do conhecimento e a representação semântica. Os diretórios têm como objetivo principal o de agrupar e organizar os conteúdos armazenados para facilitar recuperações futuras. Quantas vezes nos deparamos com a situação de necessitar de um arquivo, um texto em PDF, por exemplo, que você ainda não lembra que é com a extensão PDF, e também não lembramos mais onde colocamos. E daí, há que se procure daqui, dali e depois de muito tempo, e de ter executado e lido vários arquivos você o encontra em uma pasta com nome “Nova Pasta” que está dentro do diretório “Arquivos pendeentes”, sim, com erro de ortografia, que está dentro de outros diretórios com nomes também “estranhos”, em seu segundo disco de armazenamento, e com nome “xpto.pdf”. Fazer uma busca no computador seria praticamente inútil uma vez que não lembrávamos do nome do arquivo, e sim lembrávamos de um dia haver lido um trecho de texto do assunto a qual estamos interessados, em um arquivo em nosso computador.

Para esboçar melhores os problemas, pontos importantes e as técnicas para solucionar esse e outros problemas de semântica, principalmente no âmbito da representação do conhecimento em ambientes Web, será visto o exemplo de diretório para a construção desta monografia. O 0 expõe as conclusões de uma forma mais generalizada para Web, uma Web semântica. Foi criado um diretório para armazenar os arquivos e todo tipo de material relacionado a esta monografia. Após alguns dias, lendo alguns artigos, textos na Internet e digitando alguns trechos, o diretório já estava como na Figura 2-13. Deve-se tentar analisar de forma visual este diretório de acordo com a figura e explicitar brevemente o que pode ser concluído do conjunto de arquivos, tentando descrever de maneira mais imparcial possível as relações, ou seja, como se um observador estivesse descrevendo o conteúdo da pasta. Importante ressaltar que esse observador na visão da Web semântica é um agente inteligente

de software, espalhados pela rede, a fim de executar alguma tarefa baseado no contexto do domínio à qual estão imersos.

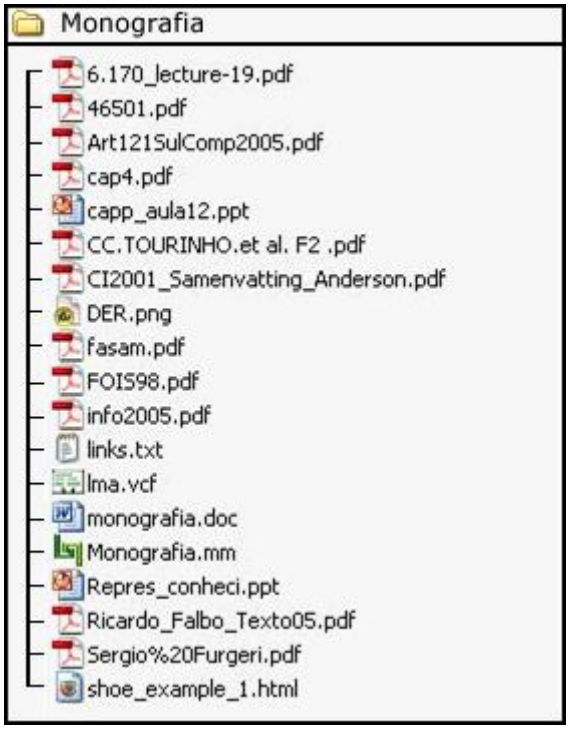


Figura 2-13 Exemplo de associação de conteúdo por diretórios

Após uma ligeira observação no diretório pode-se fazer uma breve descrição do conteúdo do mesmo. Como o nome do diretório é **Monografia** e temos um arquivo tipo "documento de texto", nomeado como **monografia.doc** é provável que este seja o texto editável do trabalho. Um arquivo **links.txt** deve conter *link*s da Web relacionados ao tema proposto. Caso o observador tenha familiaridade e conhecimento sobre extensões de arquivo também poderá concluir que **Monografia.mm** é um arquivo de mapa mental da ferramenta InteliMap[6], **lma.vcf** é um arquivo do tipo vCard para representação de contato virtual onde não sabemos ainda o que significa o **lma** de seu nome e o arquivo "shoe_example_1.html" provavelmente seja um arquivo com um exemplo de código da linguagem SHOE. Temos também a aparição de alguns nomes como Anderson Samenvatting, Ricardo Falbo, e Sérgio Furgeri que devem ser autores que abordam temas relacionados. Mas afinal de contas, quais os temas de maior relevância para esta monografia?

Com nomes de arquivos como "6.170_lecture-19.pdf", "46501.pdf" e "capp_aula12.ppt" é impossível extrairmos qualquer informação por mínima que seja para relacionar à monografia. Estes arquivos foram baixados da Web ou adquiridos no laboratório

[6] http://www.intelimap.com.br/intelimap.html

da universidade como várias pessoas fazem todos os dias. A comodidade de simplesmente armazená-los em um diretório, com os mesmos nomes a qual foram adquiridos, trará futuros problemas na recuperação e até em colaboração e comunicação como vimos um exemplo anteriormente. Ao ler estes arquivos e analisá-los, podemos renomeá-los com um nome a qual nos sugere o assunto abordado pelo mesmo. É notório que neste ponto sofremos forte influência da percepção pessoal, ou seja, a falta de imparcialidade e senso comum na nomeação do arquivo, pois:

“A imensa diferença entre uma coisa e a mesma coisa,
é a maneira à qual olhamos para ela.”

Resumidamente, somos todos diferentes, pensamos diferentes e pela natureza imprecisa do mundo real, ponderamos as coisas de formas diferentes. Estamos à mercê da semântica pessoal. O trabalho de hierarquização dos diretórios e renomeação destes e dos arquivos, pode ser uma tarefa um tanto quanto chata, cansativa, improdutiva e parecer não trazer grandes benefícios. Mas ao contrário do que se pensa, principalmente no ambiente da Web, onde a quantidade de dados e arquivos estão crescendo em velocidades estonteantes, esta prática se torna de grande impacto para a contextualização semântica e integração com ontologias, políticas de acesso e resultados de buscas além de serem feitas, cada vez mais, de forma automatizada e/ou semi-automatizada. A Tabela 2-1 mostra algumas das considerações que foram feitas no diretório da monografia após analise.

Tabela 2-1 Alguns arquivos e seus respectivos temas abordados, segundo a semântica pessoal do autor

Nome do Arquivo	Tema Abordado
46501.pdf	Models for Representing Task Ontologies
6.170_lecture-19.pdf	Modelos Conceituais
Art121SulComp2005.pdf	Aquisição de Conhecimento Automatizada para Sistemas Especialistas Probabilísticos
cap4.pdf	Formalização de Ontologias e Projeto de Domínio
fasam.pdf	O Estado da Arte no Estudo das Ontologias
CI2001_Samenvatting_Anderson.pdf	Cognitive Psychology and it's Implications
Ricardo_Falbo_Texto05.pdf	An Agile Approach for Web Systems Engineering
Sergio%20Furgeri.pdf	Representação de Informação e Conhecimento

Existem sistemas computacionais que executam esta análise de forma automatizada em documentos à qual esta tarefa tem o nome de *scanning* e proporciona uma busca por palavras-chave, padrões de descrição, termos repetidos entre várias outras técnicas para decidir sobre o assunto, adicionar metadados e então nomear o dado arquivo. Mover os arquivos entre os diretórios já se torna uma tarefa mais complicada e custosa para o desempenho do sistema. Algumas soluções então, estão na nomeação e locação do arquivo na hora que ocorre o *upload.* Além disto, o sistema pode armazenar um nome sugestivo e com significado semântico em uma tabela no banco de dados e exibir o nome original para os usuários a fim destes não sofrerem imposições de nomes. Após o refinamento e renomeação dos arquivos tem-se o diretório como na Figura 2-14.

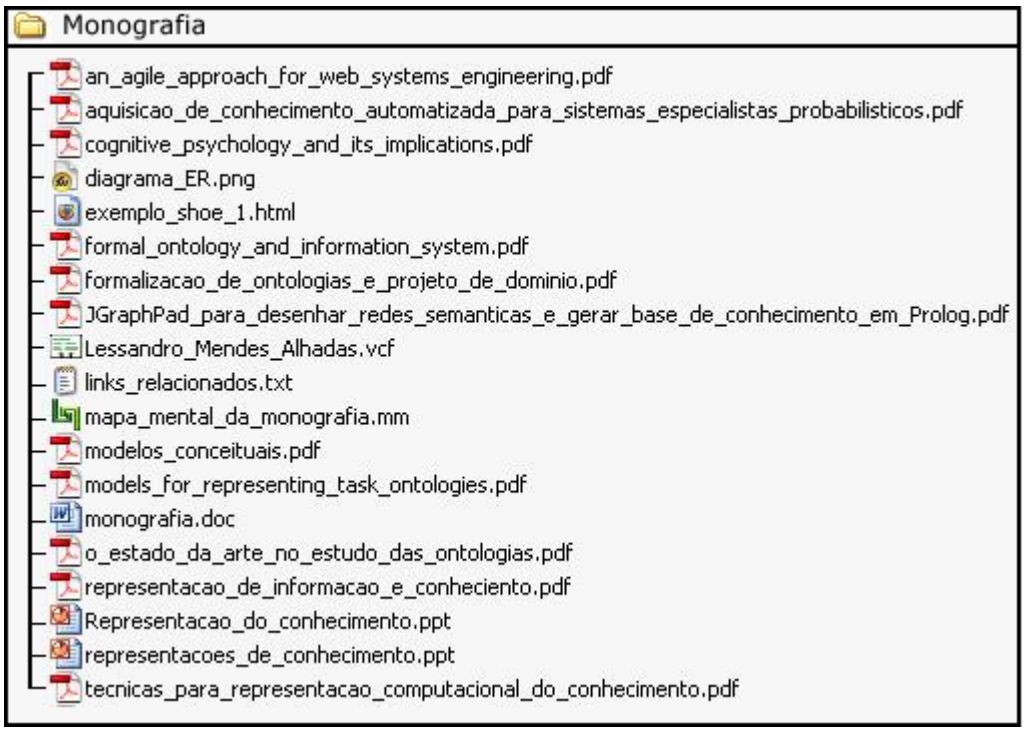


Figura 2-14 Diretório da monografia após algumas renomeações de arquivos

Nitidamente vemos que agora mais informações podem ser extraídas e relacionadas ao tema e assuntos abordados na monografia, apesar de os arquivos ainda estarem misturados quanto a seus fins e alvo de aplicação, linguagem de notação entre outras características que foram decididas antes, e aplicadas na tarefa de *scanning*. Estas decisões são muito importantes, pois irão influenciar de maneira direta sobre mecanismos de busca, mecanismos de inferência, agentes pela rede e no compartilhamento de dados e recursos. Entre as mais relevantes decisões de renomeação, podemos elucidar:

- o nome do arquivo será sempre o título do documento? Ou uma concatenação de duas ou três palavras-chave? Ou com prefixo e um sufixo pré-definidos?

- Se for nomeado pelo título: E se o título for exageradamente grande?

- Sempre passar todo o nome para caixa baixa? Ou com iniciais em caixa alta?

- Espaços em branco são substituídos por sub-escrito, sinal de menos ou outro caractere?

- Acentos e cedilha serão mapeados em que símbolo?

- Preserva-se a língua de origem, por exemplo, inglês, português ou tenta se fazer uma tradução?

- Ao inserir um novo arquivo no diretório com mesmo nome e tipo de extensão como serão feitas as anexações de numeração de índice e versão. Por exemplo, “arquivo.txt”, o próximo será “arquivo2.txt”, “arquivo(2).txt” ou “arquivo_2.0.0.txt”?

Visto então a necessidade de separação dos arquivos e refinamentos, tem-se (Figura 2-15):

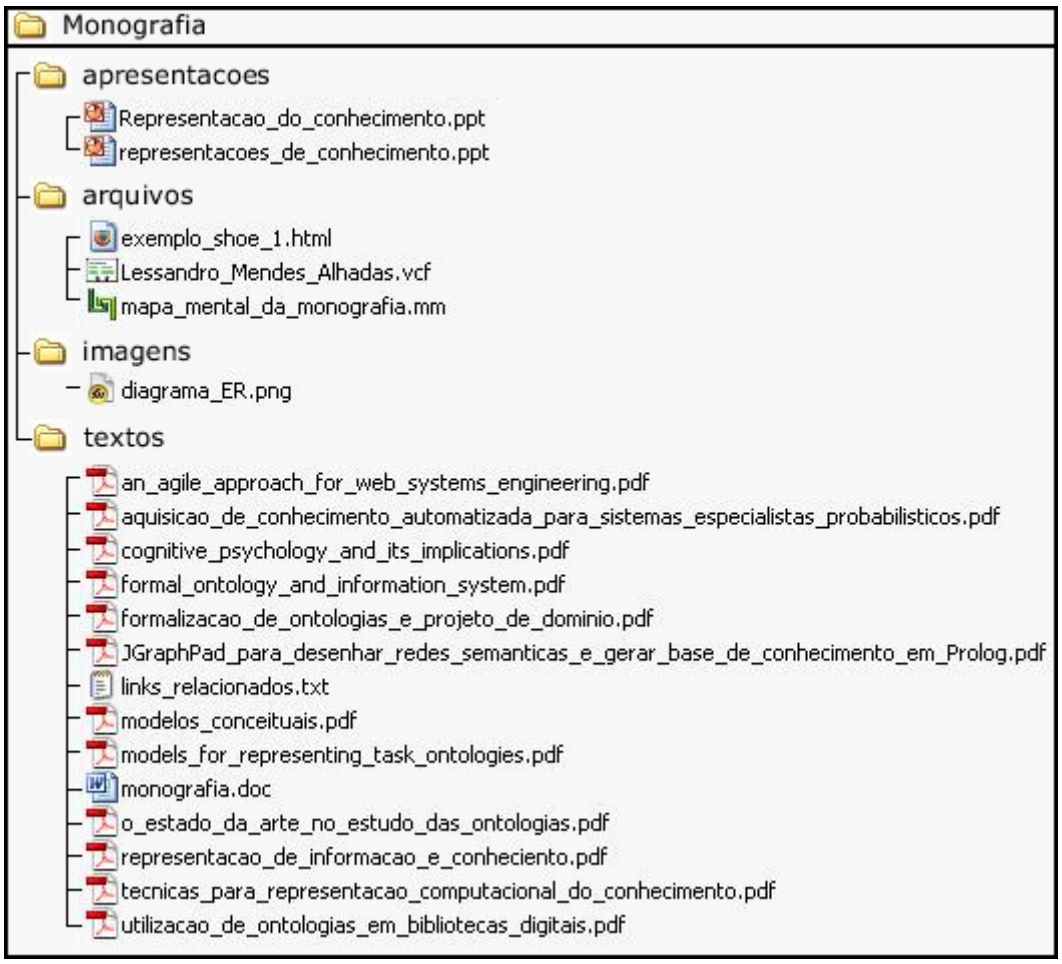


Figura 2-15 Diretório da monografia após uma melhor hierarquização

A hierarquia completa e com mais refinamentos do diretório desta monografia pode ser encontrada no anexo B.

Agora já podemos absorver muitas outras informações do diretório “monografia” como: foi descrita baseada em apresentações, e textos que relatam temas como sistemas Web, modelos conceituais, representação do conhecimento e ontologias. Arquivos de exemplo foram analisados ou desenvolvidos além de uma imagem que aparenta demonstrar um diagrama E-R. Já se consegue identificar o contexto da monografia. A partir de agora, o diretório tem um pouco de significado, uma contextualização para cada elemento que compõe o conhecimento para a confecção desta monografia. No ambiente da Web, estas considerações irão tomar proporções ainda maiores, uma vez que todo documento publicado está disponibilizado através de seu URI. Mecanismos de busca, entre os vários critérios para efetuar a recuperação dos conteúdos relacionados aos termos da pesquisa, por exemplo, utilizam o URI indexado em suas bases de dados para comparação. Encontrar arquivos com nomes sugestivos é extremamente mais fácil e rápido. A Figura 2-16 mostra os dois primeiros resultados obtidos para a pesquisa ao termo “ieee ufjf membros” a fim de encontrar alguma página que contenha informações dobre os membros do Ramo Estudantil IEEE da UFJF. A pesquisa foi efetuada nos sites: Google e Live Search por serem populares, possuem motores de busca (*engines*) diferentes e mostrarem em destaque os termos pesquisados e encontrados no título da página, em *tags* Meta, no texto e no URI caso ele ocorra.

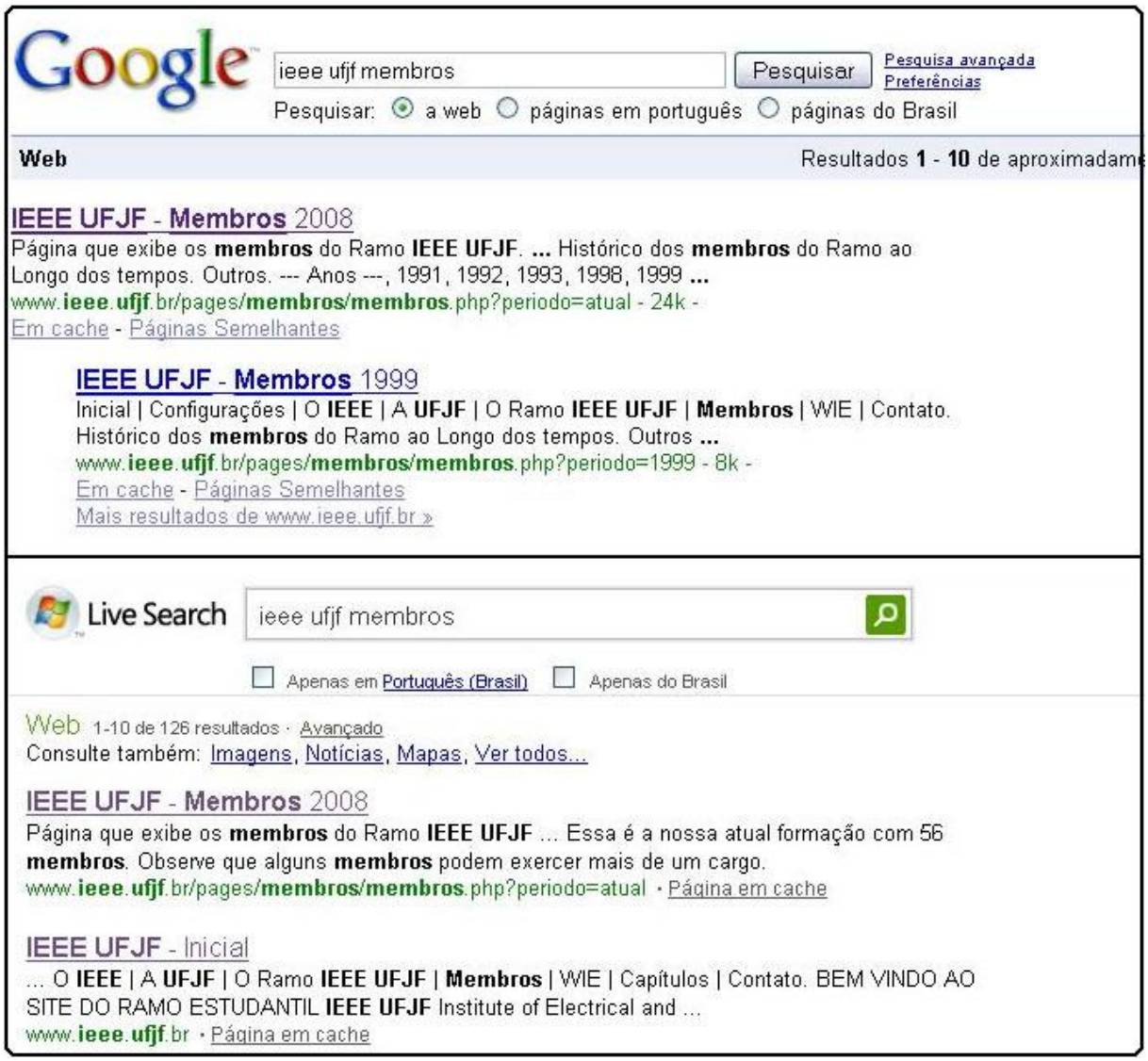


Figura 2-16 Resultados da pesquisa "ieee ufjf membros" no Google e no Live Search em 11/2008

Para este caso, de acordo com os destaques nos resultados, pode-se observar que os termos foram encontrados no título, no texto e no URI, contribuindo para a classificação e ordenação entre os primeiros resultados.

Vê-se aqui, que a organização na hierarquia dos diretórios e a nomeação de arquivos podem afetar aspectos como disponibilização de recursos e políticas de acesso de um Web site. Por exemplo, é de interesse que o portal tenha suas páginas "públicas" descritas preferencialmente em textos estáticos como (X)HTML, com nomes sugestivos e que estejam indexadas aos mecanismos de busca facilitando sua recuperação. Por outro lado, páginas, recursos e documentos "privados" que executam cálculos, acessam informações confidenciais do negócio e alteram informações de caráter administrativo, sendo majoritariamente dinâmicas, não devem ter acesso livre a robôs de *scanning* e devem ter maior segurança de armazenamento. Para contribuir na cobertura deste furo de segurança pode-se, por exemplo,

separar os arquivos públicos dos privados e adicionar um arquivo ROBOTS.TXT, que bloqueia a ação de robôs de indexação, na raiz do diretório à qual deseja-se proibir o acesso. A Figura 2-17 esboça este exemplo:

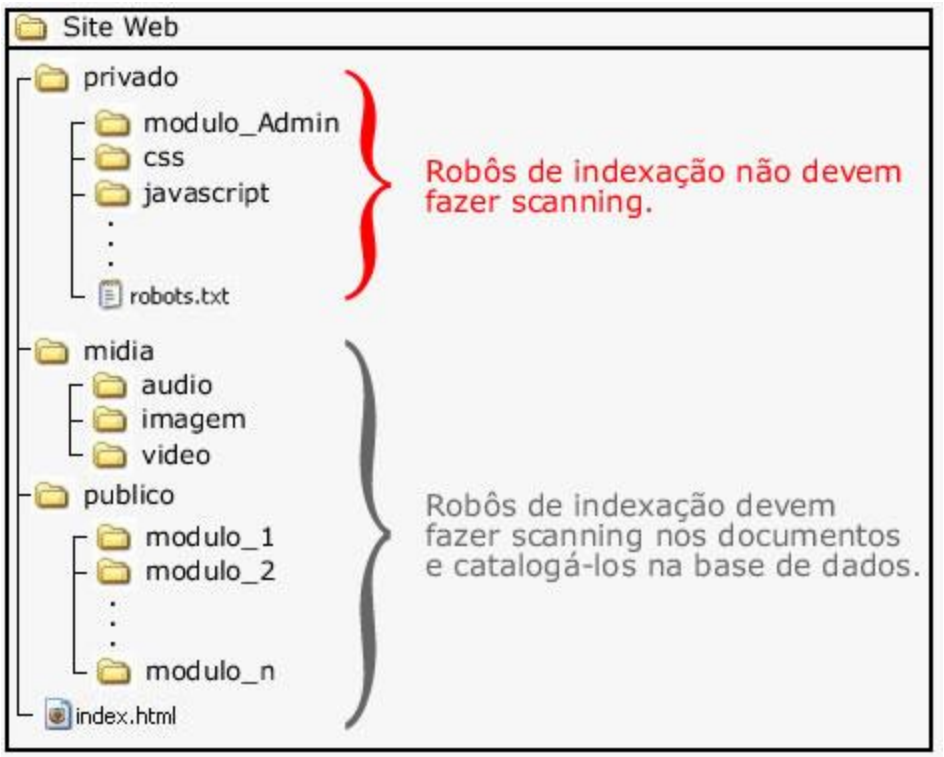


Figura 2-17 Hierarquia de diretórios com bloqueio a robôs de indexação

Sendo assim, tem-se certa segurança sobre os arquivos críticos além de aumentar o desempenho dos robôs de indexação reduzindo seu trabalho. Deve-se ressaltar, que a parte "pública", na Figura 2-17 os diretórios: "midia", "publico" e o arquivo "index.html" devem então receber quanto maior quantidade de informação semântica possível para que se tornem realmente imersos em um contexto com significado na Web, ou seja, devemos utilizar de todos os métodos possíveis e de interesse para adicionar semântica nestes recursos. Alguns desses métodos serão tratados no próximo capítulo convergindo-os nas linguagens para a Web semântica. A parte "privada" pode também receber semântica e ser administrada por regras de sistemas especialistas e/ou como por ontologias, mas devem ser de total domínio dos proprietários e desenvolvedores, e não livres a terceiros.

Em BAYKAN, HENZINGER e WEBER (2008) verifica-se um estudo de reconhecimento da linguagem presente no texto para agrupar resultados na Web, relacioná-los à ícones, disponibilizar informações e mensagens customizadas aos usuários bem como personalização de navegadores Web.

Visto estes pontos entre outros benefícios do modelo de diretórios que estão sendo explorados, considera-se então relevante a prática de organização, nomeação e manutenção de diretórios e arquivos do sistema.

2.11 Scripts

Scripts são mecanimos que descrevem e raciocinam sobre uma seqüência de eventos, sobre um comportamento. Especificam uma trajetória estereotipada de acontecimentos que normalmente se realizam e que se seguem para representar conhecimento procedimental. Contêm um conjunto de "*slots*" dos seguintes tipos:

- Objetos – representando objetos envolvidos nos eventos do *script*;

- Agentes – representando entidades que estão envolvidas nos eventos do *script*;

- Condições de entrada – que devem ser atendidas para que os eventos descritos no *script* possam ocorrer;

- Resultados – que irão ser verdadeiros após a ocorrência dos eventos descritos no *script*;

- Seqüência de Cenas – seqüências de eventos que ocorrem;

São adequados para:

- Os casos em que temos seqüências tipificadas (por exemplo, as etapas para escrever um artigo ou as fases da análise de um incidente);

- Descrever planos que devem ser seguidos (por exemplo, procedimentos para se produzir um artefato ou os tratamentos a seguir para a cura de uma doença);

Exemplo: Tem-se uma reunião semanal, onde todos os membros discutem e renovam assuntos relacionados, as atividades e os eventos que serão ou estão sendo realizados.

Script "Encontro Semanal"

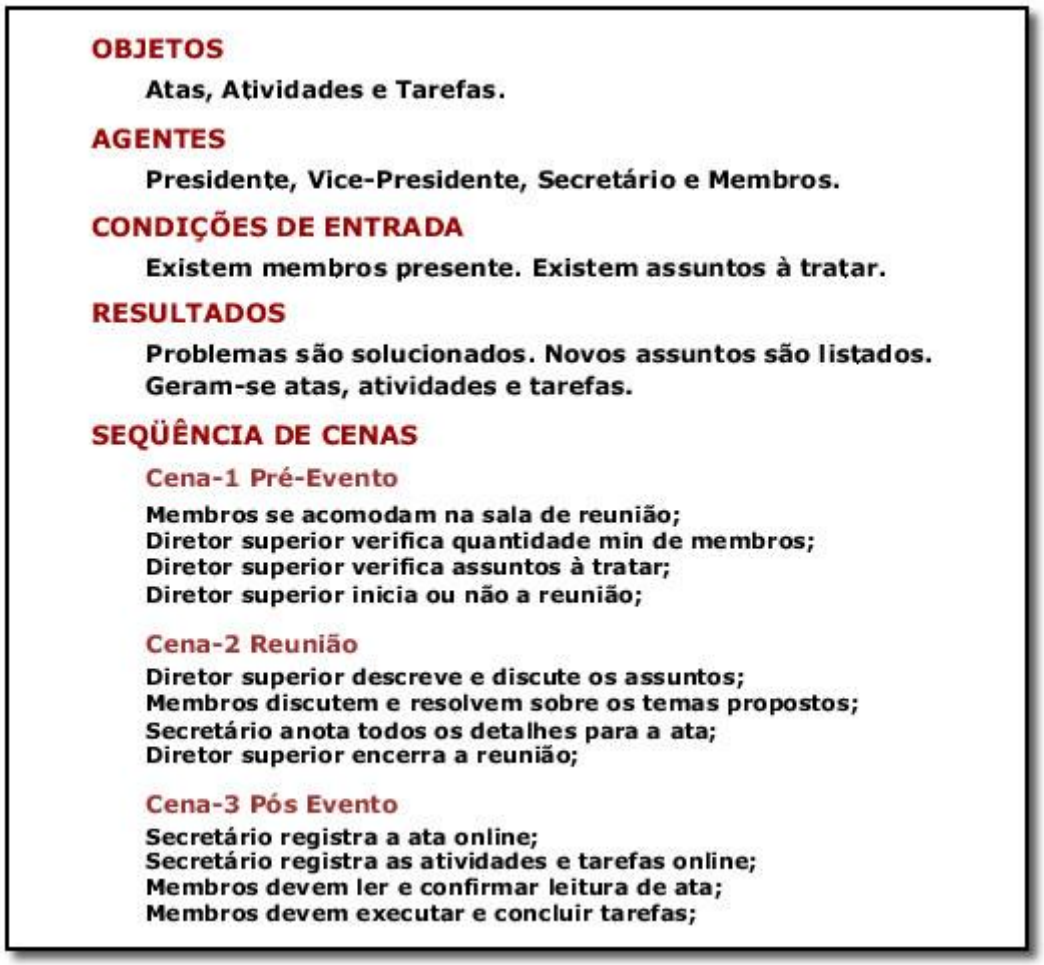


Figura 2-18 Exemplo de representação de conhecimento através de *script*

Deve-se observar aqui o grande poder de expressividade e transmissão de "uma imagem que se tem" do estado do ambiente e algumas cenas, com descrições bem claras e concisas, ou seja, com poucas palavras e uma linguagem bem próxima da linguagem natural, descreve-se todo um cenário onde se tem: quem está nestas ações, quais as coisas envolvidas, o que são requisitos para as ações e o que são consequências dessas ações. Observa-se forte impacto da semântica pessoal neste modelo uma vez que estado de ambiente e visão de cenas são extremamente diferentes de abordagem pra abordagem, de grupo pra grupo, pessoa pra pessoa, além de não haver um vocabulário rigoroso para as descrições, o que pode torná-lo complexo para inferências.

Em eventos com muitas ações e com bastante atores e objetos envolvidos este modelo pode auxiliar como guia para entendimento do fluxo de execução das ações e a identificar alguns requisitos essenciais ao sistema. Por exemplo, de acordo com a Figura 2-18, vê-se que a reunião ocorre caso haja membros e assuntos à tratar. Caso sim, a reunião inicia-se com o diretor declarando e discutindo os tópicos com os membros. Segue a reunião e, após, deve-se

registrar a ata da mesma e as atividades acordadas, que neste caso, são requisitos do sistema além de mecanismos para os mebros lerem a ata e concluirem a reserva de suas tarefas. Deve-se, então, ser desenvolvido módulos para a manutenção destes componentes.

Como os *script*s são representações descritivas e ainda não possuem ferramentas estabelecidas para construção, manutenção e inferência, deve-se aqui concluir que este modelo pode ser levado em consideração em situações específicas, em que ocorra uma seqüência de ações na composição de um evento e que este se apresente de forma complexa como um todo, mas retoma a problemática de codificação textual, liberdade de expressão, e forte influência da semântica pessoal.

2.12 Conclusões

Representar o conhecimento talvez seja a tarefa mais antiga e nobre do ser humano com início atribuído as pinturas rupestres no período paleolítico, considerado como o nascimento da história da arte e a qual é um dos aspectos mais relevantes para a diferença entre os Homens e os Animais. O homem adquire, registra, armazena, recupera, transmite, compartilha e raciocina sobre o conhecimento tanto para evoluir no próprio conhecimento quanto para comunicar-se e transmitir cultura. Para tal, o homem desenvolveu várias maneiras de se representar o conhecimento adquirido e essas maneiras também evoluíram juntamente com a evolução do pensamento humano. No escopo deste trabalho, mostraram-se alguns métodos de representação do conhecimento no âmbito da Ciência da Computação com intuito de extrair o que de melhor cada modelo pode oferecer e então unirmos essas técnicas em prol do desenvolvimento e avanço da Web semântica.

A Figura 2-19 nos mostra algumas características mais importantes para a representação de conhecimento que foram observadas e dispostas como uma linha evolutiva dos modelos anteriormente descritos.

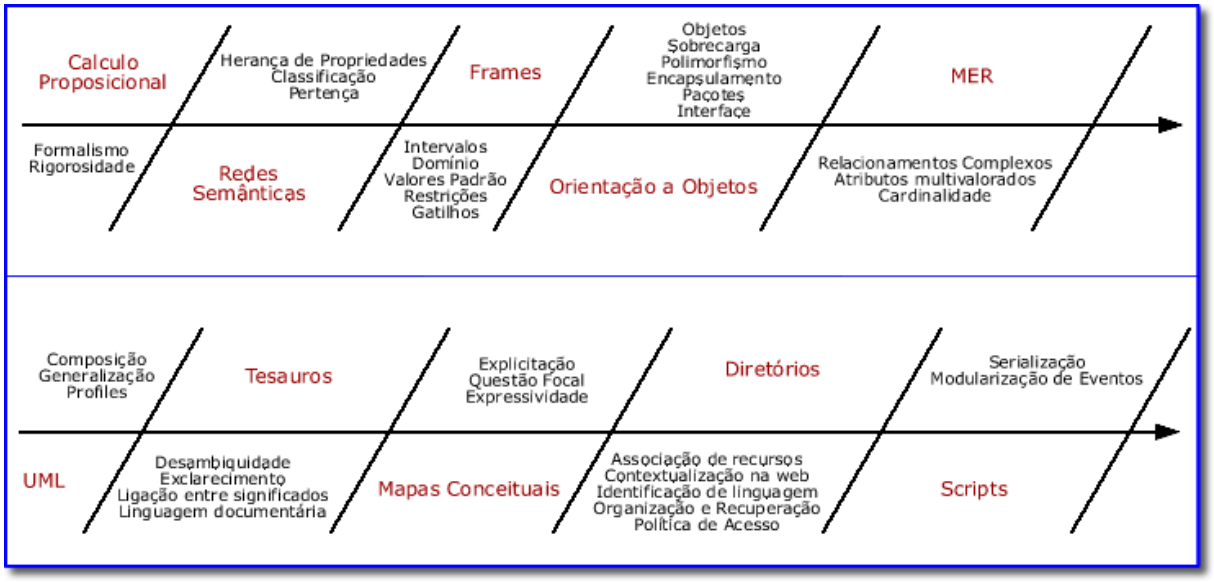


Figura 2-19 Características observadas nos modelos de representação do conhecimento

Outras informações e análises de comparação que foram levadas em consideração neste estudo estão resumidas na Tabela 2-2.

Tabela 2-2 Alguns pontos importantes dos modelos abordados

Modelo	Expressividade	Visualização Gráfica	Codificação para Mapeamento	Ferramentas
Lógico	Ótima	--	Regular	Ótimas
Redes Semânticas	Ótima	Boa	--	Boas
Frames	Boa	Boa	Boa	Boas
Orientação a Objetos	Regular	--	Boa	Ótimas
MER	Regular	Boa	Regular	Ótimas
Tesauros	Boa	Boa[7]	Boa	Boas
Mapas Conceituais	Ótima	Ótima	Ótima[8]	Ótimas
Diretórios	Boa	Boa	--	--
Scripts	Boa	Boa[9]	--	--
UML	Regular	Ótima	Ótima	Ótimas

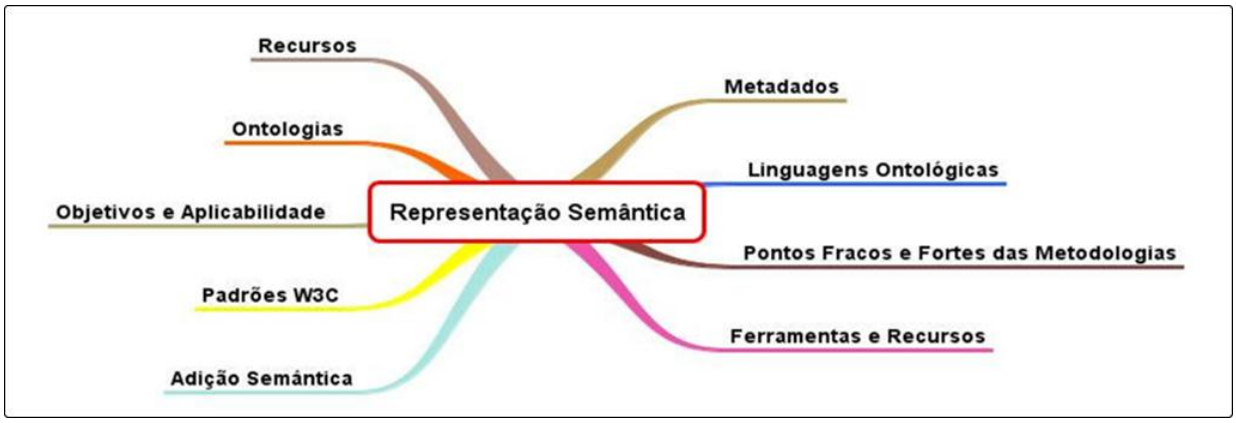
[7] Se representados através de gráficos RDF
[8] Se exportados para XML
[9] Se representados semelhante à *frames*

Conclui-se aqui que a representação UML oferece os melhores benefícios para a representação do conhecimento levando-se em consideração os fatores preponderantes previamente definidos para este trabalho, mas ressaltando que todo modelo tem suas características peculiares e que podem findar em melhores soluções de acordo com a aplicação em questão a serem utilizados. Importante lembrar também da organização e manutenção de diretórios em sistemas Web e utilização de tesauros e *script*s para descrever e esclarecer conceitos e eventos respectivamente. Por fim, podem-se destacar as inúmeras possibilidades de avanço para futuros estudos e pesquisas no mapeamento semântico com mapas conceituais uma vez que possuem ótimas ferramentas para construção, visualização gráfica clara, codificação XML e alto poder de expressividade.

No próximo capítulo serão tratados os aspectos relevantes para a representação semântica e alguns conceitos que norteiam todo o ciclo de vida das ontologias e os sistemas definidos por estas.

Capítulo 3

REPRESENTAÇÃO SEMÂNTICA



Semântica é o estudo do significado (PALMER, 1976). Já na definição nos deparamos com o problema mais difícil, senão impossível, no contexto da representação semântica de ser resolvido: a semântica pessoal à qual foi brevemente decorrido no capítulo anterior, na seção **2.10**. As coisas possuem significados diferentes para pessoas que possuem pontos de vistas diferentes. A Web semântica deverá ser uma extensão que adicionará significado aos dados e recursos da Web primitiva contribuindo para comunicação e colaboração dos sistemas computacionais de acordo com o domínio e contexto que estão inseridos. Serão tratados neste

capítulo, os modelos e linguagens que dão suporte a adição, manutenção e evolução de conteúdo semântico em ambientes computacionais, não levando em consideração os problemas e implicações da semântica pessoal.

3.1 Metadados

A definição mais comum na literatura para metadados é que são "dados sobre dados". Informações úteis ou que descrevem um dado de maneira sucinta. Devem dizer claramente o que aquele dado é entre outras informações que julgar necessário. Adicionando metadados a arquivos e recursos na Web se esta, de certa forma, inserindo semântica aos mesmos e inserindo-os a um contexto com significado. Ao longo da evolução da tecnologia da informação o conceito e utilização de metadados foram também sofrendo mudanças e evoluíram paralelamente à sua importância em sistemas computacionais como se pode observar na Tabela 3-1.

Tabela 3-1 Evolução da definição de metadados (Adaptado de IKEMATU, 2005)

ESTÁGIO	RECURSO A SER ADMINISTRADO	DEFINIÇÃO DE METADADOS
DADOS	Valores dos dados	Informação necessária para administrar o recurso dos dados
INFORMAÇÃO	Valores dos dados e o contexto da informação	Informação necessária para administrar o recurso das informações
CONHECIMENTO	Valores dos dados, o contexto da informação e instruções das regras de negócio	Informação necessária para administrar as regras e políticas de negócio da organização
SABEDORIA	Valores dos dados, o contexto da informação, regras de negócio executáveis, monitoração das regras de negócio e regras e métricas de avaliação	Informação necessária para administrar o comportamento da organização de acordo com suas regras e políticas de negócio

A finalidade principal dos metadados é documentar e organizar de forma estruturada os dados das organizações, com o objetivo de minimizar duplicação de esforços e facilitar a manutenção dos dados (IKEMATU, 2005). A prática de adicionar metadados a arquivos, até pouco tempo atrás, era exclusividade de especialistas e profissionais de grandes corporações e instituições que trabalhavam com tecnologia da informação, com grandes volumes de dados, necessitavam de colaboração e faziam ainda de forma restrita, à maneira de cada organização.

Com o tempo foram aparecendo padrões mais genéricos, mais populares e colaborativos. Mas paralelamente, a Web expandia rapida e desordenadamente, sem ter muita preocupação com metadados e separação de informação do conteúdo e a informação de exibição. Com os conceitos e metodologias da Web 2.0 ficou ainda mais nítida a necessidade de separação do conteúdo contido nas páginas da Web e os códigos para a formatação e leiaute do documento em si. Estes problemas foram então estudados e tratados por várias técnicas e metodologias como, por exemplo, *Tableless* (REIS, 2007), trazendo à tona e destacando ainda mais a importância do significado do conteúdo. Para tal, a Web semântica direciona seus esforços em prol da contextualização dos dados e recursos disponíveis.

De uma maneira geral, vários padrões que se consolidaram no mercado tinham finalidades diferentes e ainda voltados para um ambiente restrito da organização como se pode verificar na Tabela 3-2.

Tabela 3-2 Finalidade de alguns padrões de metadados (Adaptado de IKEMATU, 2005)

PADRÃO	FINALIDADE
Directory Interchange Format (**DIF**)	Entradas de diretórios que descrevem um grupo de dados
Government Information Locator Service (**GILS**)	Informações governamentais
Federal Data Geographic Committee (**FDGC**)	Descrição de dados geoespaciais
Machine Readable Card (**MARC**)	Catalogação bibliográfica
Consortium for Interchange of Museum Information (**CIMI**)	Informações sobre museus
Meta Data Interchange Specification (**MDIS**)	Padrão para troca de metadados entre ferramentas de tecnologia da informação
Open Information Model (**OIM**)	Conjunto de especificações para facilitar o compartilhamento e reuso no desenvolvimento de aplicações e *datawarehouses*
Common Warehouse Meta Model (**CWMM**)	Padrão para troca de informações entre esquemas de banco de dados e *datawarehouses*
Dublin Core (**DC**)	Dados sobre páginas da Web

Entre estes, o padrão *Dublin Core*[10] se destaca por ter sido desenvolvido direcionado a Web, bastante introduzido nas especificações de ontologias e possuir uma comunidade bastante ativa e atualizada de mantenedores. Possui um conjunto de 15 elementos para descrição de

[10] http://dublincore.org/documents/dces/

recursos que são bastante genéricos e que fundamentaram e popularizaram sua adoção (Tabela 3-3).

Tabela 3-3 Definição dos elementos de metadados *Dublin Core*

ELEMENTO	DEFINIÇÃO
Contributor	Uma entidade responsável por fazer contribuições ao recurso.
Coverage	Tema espacial ou temporal do recurso, a aplicabilidade espacial do recurso ou jurisdição à qual o recurso é pertinente.
Creator	A entidade primeiramente responsável pela criação do recurso.
Date	Um ponto ou período de temo associado a um evento no ciclo de vida do recurso.
Description	Uma descrição do recurso.
Format	O formato do arquivo, meio físico ou as dimensões do recurso.
Identifier	Uma referência inequívoca do recurso dentro de um determinado contexto.
Language	A linguagem do recurso.
Publisher	Uma entidade responsável por fazer o recurso disponível.
Relation	Um recurso relacionado.
Rights	Informações sobre direitos detidos e sobre o recurso.
Sources	Um recurso relacionado à qual o recurso descrito é derivado.
Subject	O tema do recurso.
Title	Um nome para o recurso.
Type	A natureza ou gênero do recurso.

No contexto da Web a adição de metadados vem sendo aplicada a algum tempo, majoritariamente através de *tags* <meta> nos cabeçalhos das páginas (X)HTML. Importante aqui assinalar, que esta atividade ocorre de forma totalmente dependente do conhecimento, interesse e a consciência do desenvolvedor, ou seja, - Programadores iniciantes podem não conhecer essas *tags*, muito menos o seu fim. - Suas páginas podem conter material que não tenha interesse em serem rastreadas e indexadas a mecanismos de busca. – O desenvolvedor não tem consciência da importância de metadados para comunicação, colaboração e Web semântica como um todo além de ainda poder assinar seus trabalhos, transmitir informações temporais e vários outros aspectos que podem ser explorados quando os metadados estão preenchidos.

A *tag* <meta> registra meta informações sobre o documento. Podem contribuir também na política de acesso às paginas como visto na seção “Diretório de Arquivos” do capítulo

anterior. A Figura 3-1 exibe o código fonte da página Web "http://www.ieee.ufjf.br/pages/membros/membros.php?periodo=atual" à qual foi exibida como primeiro resultado encontrado para a busca do termo "ieee ufjf membros" nos dois sites pesquisados, como apresentado na Figura 2-16. Neste exemplo, pode-se observar na linha 18 da Figura 3-1 que a política de acesso define que todos os robôs de indexação façam o *scanning* e conseqüente anexação à suas bases de dados.

Deve-se observar também na Figura 2-16 que o primeiro resultado exibido, também nos dois sites pesquisados, aparece a frase "Página que exibe os membros do Ramo IEEE UFJF" e em destaque os termos a qual foram requisitados na pesquisa. Essa frase não é exibida na página em si, mas é uma descrição do conteúdo da página que está especificada através da meta *tag* à qual podemos observar na linha 9 da Figura 3-1.

```
01-<?xml version="1.0" encoding="UTF-8"?>
02-<!DOCTYPE html
03- PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
04- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
05-<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt" lang="pt">
06-<head>
07-<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
08-<meta http-equiv="content-language" content="pt" />
09-<meta name="author" content="Lessandro Mendes Alhadas" />
10-<meta name="reply-to" content="lessandro@ieee.org" />
11-<meta name="description" content="Página que exibe os membros do Ramo IEEE UFJF" />
12-<meta name="keywords" content="ieee, ufjf, membros, histórico dos membros" />
13-<meta name="copyright" content="Copyright© 2008 IEEE UFJF, All Rights Reserved" />
14-<meta name="rating" content="general" />
15-<meta name="distribution" content="global" />
16-<meta name="generator" content="DreamWeaver 8" />
17-<meta name="resource-type" content="document" />
18-<meta name="robots" content="all" />
19-<meta name="revisit-after" content="1 Year" />
20-<title>IEEE UFJF - Membros 2008</title>
21-</head><body>
 .
 .
 .
N- </body></html>
```

Figura 3-1 Código fonte de documento XHTML exibindo *tags* Meta

Apesar de haver grandes possibilidades de explorar informações de documentos com *tags* meta, sua finalidade principal no momento é contribuir para pesquisas na Web pela maioria dos motores de busca, explorando para isso somente a *tag description* e em alguns casos a *tag keywords*.

De acordo com a Tabela 3-2 tem-se que o padrão *Dublin Core* fundamenta-se em metadados para a Web, que é objeto de interesse deste trabalho. Visto que este padrão pode ser

expresso utilizando-se as *tags* HTML <meta> e <*link*> temos o exemplo anterior então redefinido de acordo com a Figura 3-2. Utiliza-se a *tag link* para declarar o *namespace* do padrão *Dublin Core* e em seguida utilizando *tags* meta especificam-se os elementos que descrevem as informações do documento. Optou-se por adicionar as definições do padrão *Dublin Core* ao invés de substituir as definições do exemplo anterior, ou seja, algumas informações estão descritas somente através de *tags* meta, outras somente através do padrão *Dublin Core* e algumas especificadas nas duas maneiras como é o caso do elemento *description*.

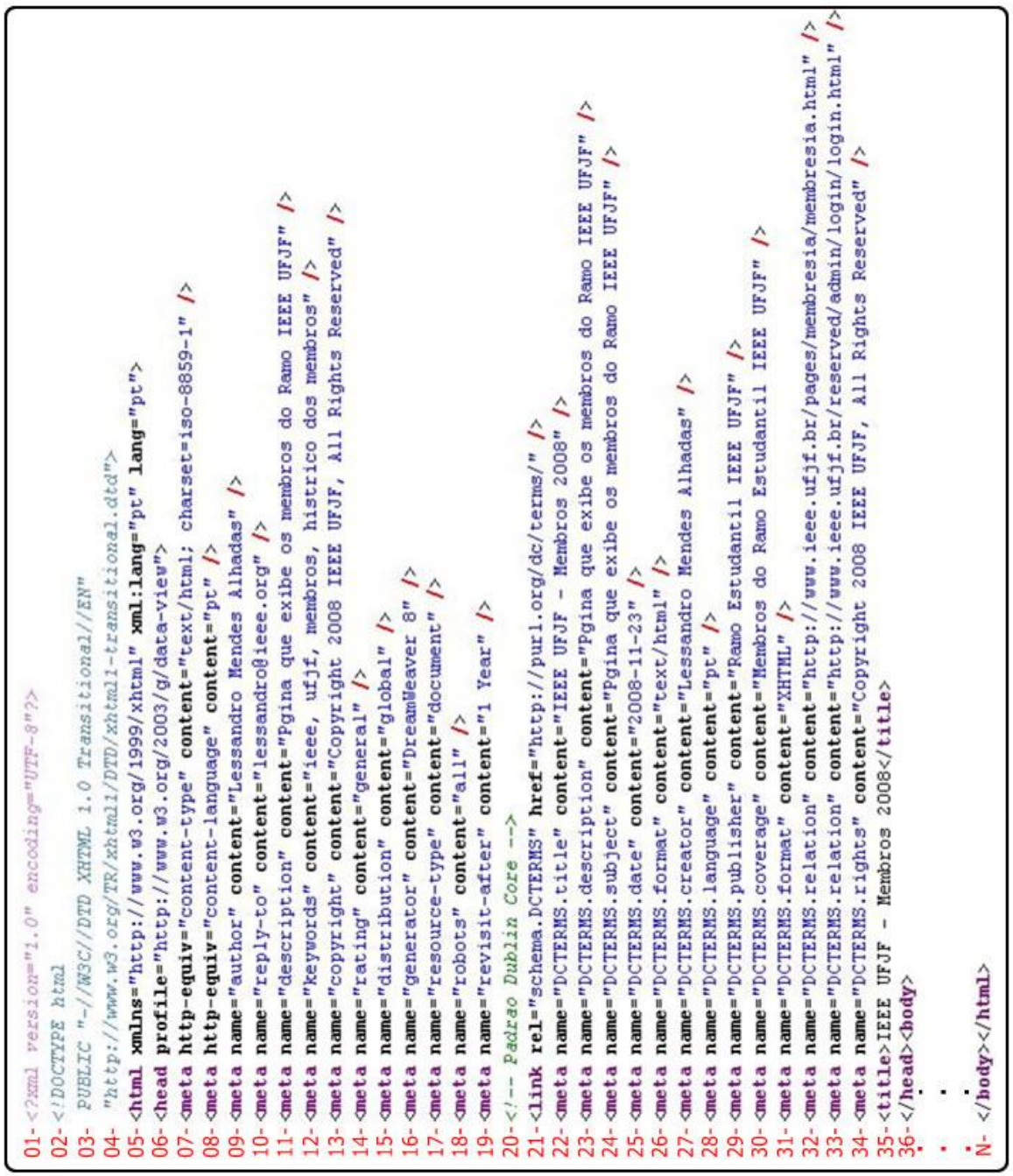


Figura 3-2 Código fonte de documento XHTML com metadados em *tags* meta e padrão *Dublin Core*

Outras iniciativas também exploram a adição de metadados e semântica através de outras tags ou atributos em documentos XML para os mais variados fins como é o caso de *microformats, RDFa*[11] *e eRDF*[12]. Definir e esgotar todos os tópicos relacionados a essas metodologias é uma tarefa um tanto quanto extensa e que deve ser executada em outra fase do projeto de definição semântica ou mapeamento, à qual excede aos objetivos deste trabalho. Neste estudo, será apresentada de forma sucinta uma descrição de *microformats* e um pequeno código de exemplo para que se possa ter uma visão da essência destas técnicas, à qual RDFa e eRDF podem ser consideradas semelhantes no âmbito de adicionar metadados aos documentos da Web, utilizando para tal, as *tags* e atributos universalmente conhecidos de XML e HTML.

Os *Microformats*[13] são itens de marcação semântica que definem um vocabulário comum para que elementos XML tenham significado e possam ser entendidos tanto por humanos quanto por máquinas. De certa forma são especificações de metadados, mas se referem a descrever trechos e conteúdos específicos, como dados de contato, datas de evento, descrição de *links* e localização geográfica, ou seja, coisas que estão fora do escopo das meta *tags*. *Microformats* vieram para suprir a necessidade de metadados para os vários tipos de informação que são inseridos na Web a fim de refinar o uso de *tags* de marcação que eram utilizadas somente para processar e apresentar conteúdo. Resumidamente as primeiras especificações definiam padrões para os nomes de classes de dados sendo codificadas no atributo *class* HTML e valores comuns para os atributos *rel* e *rev*. Entre as especificações já estabelecidas e de grande adoção pela comunidade de tecnologia da informação, destaca-se os formatos: hCard, hCalendar, hAton, hReview, hResume, rel-directory, rel-enclosure, rel-license, rel-nofollow, rel-tag, XOXO e XFN.

Pode-se dizer que o formato hCard é um mapeamento direto da especificação vCard, proposto em 1995 pela Apple, AT&T, IBM e Siemens, para a descrição de metadados de contato em arquivos HTML, XHTML, Aton, RSS e XML. O formato hCalendar define um vocabulário para descrições de eventos e XFN para relacionamentos pessoais descritos em *links* que direcionam à páginas de terceiros, sendo bastante adotado em *blogs* e sites de relacionamento. Várias outras abordagens estão sendo estudadas e aperfeiçoadas para que se tornem de fato especificações do modelo *microformats*. Para melhor esclarecimento, a Figura 3-3 exibe uma descrição de dados de contato utilizando especificação hCard e XFN. Tem-se nesta figura, o vocabulário hCard definido no atributo class, destacado de cor vermelha, seus

11 http://www.w3.org/TR/xhtml-rdfa-primer/
12 http://research.talis.com/2005/erdf/wiki
13 http://microformats.org/

respectivos valores destacados de cor azul e a especificação XFN destacada de cor laranja. Utilizando este padrão, mecanismos de busca e agentes de softwares espalhados pela rede podem reter e inferir conhecimento sobre dados contidos no documento e que antes eram somente “material a ser exibido”.

```
<div id="hcard-Lessandro-Mendes-Alhadas" class="vcard">
 <a class="url fn n" href="http://www.ieee.ufjf.br/pages/creditos/creditos.html">
  <span class="given-name">Lessandro</span>
  <span class="additional-name">Mendes</span>
  <span class="family-name">Alhadas</span>
 </a>
 <div class="org">Ramo Estudantil IEEE UFJF</div>
  <a class="email" href="mailto:lessando@ieee.org" rel="me">lessando@ieee.org</a>
  <div class="adr">
   <div class="street-address">Rua Olegário Maciel 190</div>
   <span class="locality">Bicas</span>,
   <span class="region">Minas Gerais</span>,
   <span class="postal-code">36600-000</span>
   <span class="country-name">Brasil</span>
  </div>
  <div class="tel">+55-32-9953-5089</div>
</div>
```

Figura 3-3 Exemplo de adição semântica utilizando especificação hCard

Neste ponto, vê-se que o leque de possibilidades para a adição semântica está crescendo e tomando uma posição de destaque para a comunicação e colaboração na Web do futuro, como idealizada por Tim Berners-Lee. A escolha de qual linguagem ou padrão utilizar para a adição semântica dependerá de um somatório de fatores onde se deverá ponderar e levar em consideração:

- Práticas e convenções utilizadas pelo time de desenvolvedores;

- Tecnologias e ferramentas a serem utilizadas;

- Propósito geral da aplicação;

- Abrangência e especificidade do conteúdo;

- Tratamento de segurança;

- Conformidade de padrões com outras comunidades de interesse;

- Ambiente de desenvolvimento e implantação;

- Recursos necessários e suficientes;

- Conhecimento sobre linguagens e metodologias;

- entre outros...

Conclui-se então que, cada vez mais a adição de metadados a documentos, a dados e a recursos se torna importante devido ao crescimento de aplicações que dependem e que têm melhor desempenho diante destas informações, como pode ser observado em GAMA (2008), onde o autor descreve um estudo de caso de adição semântica em aplicações de *Webservices*, em OLIVEIRA (2005) alguns detalhes de linguagens e ontologias utilizadas em *grids* computacionais e vários outros sistemas como manutenção e intercâmbio de dados entre *datawarehouses*, recuperação de informação, *workflows*, além do avanço global da semântica na Web.

3.2 Ontologias

A Web semântica visa disponibilizar informações compreensíveis a máquinas e humanos com base no seu significado. Para tal, todo documento, arquivo, mídia e recurso disponível na Web deve conter anotações sobre sua semântica. Verificou-se então, a falta de organização e estrutura de maneira ontológica para que estas informações possam ser processadas e inferidas de acordo com seu contexto.

A ontologia é uma forma de representar o conhecimento de um domínio de interesse com seus termos, conceitos e relacionamentos além de oferecer mecanismos para inferências e obtenção de conhecimento. Por isso, fazem-se necessário um estudo das definições, técnicas e metodologias para a construção e manutenção de ontologias, visto que são a base para a Web semântica e a comunicação formal entre sistemas computacionais.

3.2.1 Definição na Filosofia

O termo ontologia tem suas origens atribuídas à Aristóteles ainda na Grécia antiga em alguns estudos de metafísica. Do grego “onto” que significa ser, o ser, mais a palavra “logos” que significa estudo, ciência, pode-se concluir que ontologia é o estudo do ser. De acordo com a teoria filosófica, a ontologia *design*a o estudo do ser “como ser”. Concessões da realidade. A natureza, existência e organização do ser. Filósofos tentam responder questões como: “o que é o ser?” e “quais são as características comuns de todos os seres?”.

Como este estudo visa o mapeamento entre modelos e linguagens para representação de conhecimento em meios digitais, não há necessidade de aprofundar nos fundamentos com base nos pensamentos filosóficos, e sim no âmbito da ciência da computação.

3.2.2 Definição na Ciência da Computação

Há muitas definições a respeito do que vem a ser uma ontologia (GUARINO, 1997), porém resumindo os principais pontos dessas definições pode-se dizer que uma ontologia é "uma conceitualização compartilhada de um determinado domínio", conforme muito bem definido por GRUBER (1995). É composta de um conjunto de conceitos dentro desse domínio, sendo esses organizados como uma taxonomia, e de relações entre esses conceitos. A ontologia possui também axiomas, ou seja, regras pertinentes ao domínio em questão.

Também em Ciência da Computação, FENSEL (2000) apresenta uma definição mais elaborada para ontologia como "uma especificação formal explicita de uma conceitualização compartilhada". Mais resumidamente podemos dizer que é "a teoria de objetos e seus vínculos". Representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Provê critérios para distinguir vários tipos de objetos (concreto e abstrato, existente e não-existente, real e ideal, dependente e independente) e seus vínculos (relacionamentos, dependências e predicados).

3.2.3 Utilização

Na Ciência da Computação foi inicialmente utilizada em Inteligência Artificial, logo foi observada e incorporada às atividades de Engenharia de Software e Arquitetura da Informação como representação do conhecimento sobre o mundo que está sendo modelado ou parte dele. Geralmente descreve: indivíduos, objetos, classes, conjuntos, tipos de objetos, atributos, propriedades, características, relacionamentos etc. Ontologias são mecanismos de especificação.

No início, construir ontologias era quase uma arte, serviço feito manual e dispendioso de tempo e custos. Com o consenso de que as ontologias estavam sendo de crucial importância para a comunicação e colaboração entre sistemas, ferramentas e novas aplicações foram surgindo e estão sendo aprimoradas para dar suporte ao ciclo de vida das ontologias bem como para integrá-las ao sistema sendo especificado.

Hoje, as ontologias estão presentes nas definições de *Webservices*, recursos em *grids*, sistemas de conhecimento jurídico, modelos de sistemas biológicos, sistemas gestores, *e-commerce*, educação à distância, composição e validação de *workflows*, mediação entre esquemas, recuperação de informação, processamento de linguagem natural, gestão do conhecimento, Web semântica e inúmeras outras áreas de aplicações.

3.2.4 Linguagens de Representação

A necessidade de semântica e melhor organização do conhecimento nos sistemas computacionais geraram um ambiente propício para a especificação e advento de várias linguagens que relacionam recursos, adicionam metadados e colocam significado nas aplicações descrevendo domínios principalmente através de ontologias. Esta seção descreve brevemente as linguagens de maior impacto e poder de expressividade para a codificação do modelo conceitual, deixando de lado detalhamentos e comparações. Para aprofundar nos estudos destas e outras linguagens maiores informações podem ser obtidas, por exemplo, em HORROCKS (2001), RIBEIRO (2003), PACHECO (2004), BREITMAN (2005) e MORAIS (2006).

XML é a metalinguagem base para especificação das linguagens para a Web semântica uma vez que possui formalismos suficientes para ser tratada por máquinas e mecanismo natural de extensibilidade. A partir de XML tem-se:

- **RDF**: linguagem para fazer declarações sobre recursos, adição e especificação de metadados;
- **RDFS**: linguagem de esquema que provê os elementos básicos, domínios e restrições para a linguagem RDF.
- **DAML**: linguagem de marcação que objetiva maior expressividade na descrição de esquemas do que RDFS;
- **OIL**: Linguagem e infra-estrutura com objetivos semelhantes à DAML;
- **DAML+OIL**: linguagem sucessora das linguagens DAML e OIL à qual combinou características de ambas e foi antecessora da OWL.
- **OWL**[14]: linguagem para definição de ontologias na Web.

A OWL veio para suprir a falta de alguns dos formalismos nas linguagens da Web semântica bem como descrições lógicas a qual lhe dá o poder de ser traduzida em lógica de predicados de primeira ordem. Atualmente OWL é a linguagem mais utilizada para definição e colaboração de ontologias na Web além de contar com grande comunidade de desenvolvedores, ferramentas robustas para criação e manutenção e ser uma recomendação do W3C. Possui níveis diferentes de formalidade e abstração, definição de tipos básicos de dados, recursos para anotações, tratamento de versão e mecanismos para especificar propriedades como igualdade, desigualdade, restrição, interseção, inversão e cardinalidade.

[14] http://www.w3.org/TR/owl-features/

Por esses e outros motivos, vamos considerar a OWL como a linguagem alvo do mapeamento a que se busca neste estudo. No Capítulo 5, onde serão tratadas as tarefas para a tradução e codificação em OWL, serão detalhados alguns dos recursos e propriedades desta linguagem.

3.2.5 Linguagens de Extensão

Esta seção tem objetivo de brevemente ressaltar a importância de outras iniciativas que contribuíram ou estão juntamente a evoluir para a modelagem conceitual e adição de semântica à Web. Novamente aqui se deve ratificar que este trabalho tem objeto alvo de mapear um modelo de representação do conhecimento, que tenha uma **"melhor disponibilização gráfica possível**", com capacidades necessárias e suficientes para ser traduzido em uma linguagem para a Web semântica, que tenha o "**maior poder de expressividade possível**".

As linguagens citadas nesta seção serão consideradas "complementares" e de "domínios específicos". Complementares, pois servem para adicionar metadados e semântica diretamente aos próprios recursos do sistema além de poderem estar também ligadas ao modelo conceitual em si, como é o caso das ontologias. De domínios específicos, pois especificam e descrevem semânticas objetivando um dado propósito como relacionamento pessoal em redes sociais, descrição de *Webservices*, recursos em *grids computacionais*, dados abstratos em *datawarehouses*, conteúdo de mídias digitais entre outros.

3.2.5.1 SHOE[15]

Simple HTML Ontology Extension foi uma das primeiras iniciativas de adicionar semântica diretamente ao código (X)HTML e também tem parte nos esforços que basearam as descrições de DAML+OIL e OWL. O código que define a ontologia é inserido entre o código HTML, misturando-se excessivamente código do modelo conceitual com código de formatação e leiaute do documento. Atualmente está caindo em desuso e seus idealizadores estão contribuindo no desenvolvimento das linguagens sucessoras.

3.2.5.2 GDA[16]

Global Document Annotation tem por objetivo prover um conjunto de *tags* XML que habilita máquinas reconhecer automaticamente a semântica e a pragmática das estruturas dos documentos além de reduzir a ambigüidade no mapeamento à gráficos de entidade-relacionamento ou redes semânticas. Estas *tags* definem estruturas sintáticas e semânticas intra-sentenciais, identificadores de operações, identificadores de relacionamento, dependência,

15 http://www.cs.umd.edu/projects/plus/SHOE/index.html
16 http://www.i-content.org/GDA/

elementos frasais entre outros importantes para a compreensão do contexto e significado semântico. Como SHOE seu código deve ser incorporado ao código (X)HTML.

3.2.5.3 WSDL-S[17]

WSDL-S é uma submissão candidata W3C à qual visa prover semântica às descrições de *Webservices*. A W3CWSA[18] arquitetura de *Webservices* do W3C define dois aspectos para uma descrição completa de um *Webservice*: sintático e semântico. O aspecto sintático é representado pela linguagem WSDL[19]. O segundo aspecto é descrito como a semântica do serviço, mas não e coberto pela especificação. Dois serviços podem ter a mesma definição sintática, ou seja, podem conter a mesma quantidade e tipos de parâmetros de entrada e saída de outro, mas concluírem tarefas semanticamente diferentes. Com isso faz-se necessário a adição semântica nas descrições para remoção de dúvidas e ambigüidades. Outras propostas de exemplo são iniciativas, projetos, ontologias e linguagens como **WSMO**[20], **WSML**[21], **METEOR-S**[22], **OWL-S**[23] e **SWSL**[24].

3.2.6 Outros Aspectos

As ontologias estão sendo amplamente utilizadas para modelar domínios conceituais bem como para adicionar semântica a recursos, definir vocabulário comum entre organizações, formalizar objetivos em serviços da Web, mapear conceitos, padronizar esquemas em base de dados, representar conhecimento, servir de entrada para programas de raciocínio e inferência e inúmeras outras aplicações. Para uma aplicabilidade efetiva, os desenvolvedores de ontologias devem levar em consideração vários aspectos de suma importância, além dos já citados no final da seção 3.1 que eram específicos para a escolha da linguagem a ser utilizada. Entre os vários aspectos mais relevantes e que devem ser analisados, preferencialmente antes do início do projeto de criação e manutenção de ontologias, destacam-se:

- A **finalidade** das ontologias. Sem propósito o projeto pode já iniciar dando prejuízos além de se perder o foco durante o ciclo de vida.

- O **vocabulário, termos e conceitos** dados como básicos para o modelo. Servirão de ponto de partida e consenso entre os desenvolvedores.

---

17 http://www.w3.org/Submission/WSDL-S/
18 http://www.w3.org/Submission/WSDL-S/#W3CWSA
19 http://www.w3.org/TR/wsdl
20 http://www.wsmo.org/
21 http://www.wsmo.org/wsml/
22 http://lsdis.cs.uga.edu/projects/meteor-s/
23 http://www.daml.org/services/owl-s/
24 http://www.daml.org/services/swsl/

- O **time de desenvolvedores**. Como ontologias estão intimamente relacionadas ao significado das entidades do domínio, estes devem ser definidos por um grupo de pessoas que tem convívio e familiaridade com estas entidades, a fim de que possam analisar, discutir e acordarem sobre suas definições, sendo importante evitar as definições individuais.

- O **nível de abstração e generalização** dos modelos. Propósitos diferentes podem requerer níveis de detalhamento diferentes.

- A **metodologia** para a construção e manutenção. Metodologias diferem na ponderação de preferências e privilégios além de afetarem nos processos decisórios por todo o projeto.

- A **reutilização de modelos e conceitos**. Deve-se decidir por criar toda a ontologia ou utilizar ontologias superiores e/ou de domínio à qual possuem conceitos semelhantes ao contexto que será modelado. Neste ponto, fatores de integração devem ser analisados como a política de negócio das organizações envolvidas, processos de colaboração, convenções entre outros.

- As **aplicações e agentes de softwares** que utilizarão as ontologias em processos sistêmicos. Aplicações específicas tendem a desempenhar melhores atividades com certos tipos de ontologias ou linguagem de codificação.

3.2.7 Conclusões

Ontologias possuem os melhores formalismos, características e mecanismos para a modelagem, definição e manutenção da representação do conhecimento. Estão sendo amplamente estudadas e aperfeiçoadas para aplicações na Web tanto para a descrição dos modelos conceituais quanto para adição semântica em abordagens como *Webservices* e *grids* computacionais. Entre as várias linguagens de codificação iremos convencionar neste trabalho o uso de OWL para a codificação do modelo traduzido, a qual foi previamente modelado em uma representação gráfica.

No próximo capítulo, serão abordados brevemente outros aspectos que permeiam o desenvolvimento e manutenção dos modelos e que formarão o conjunto de recursos para o mapeamento desde o processo de modelagem gráfica até as inferências na linguagem ontológica.

Capítulo 4

AMBIENTE DE DESENVOLVIMENTO

Resumidamente, a abordagem que será utilizada no mapeamento detalhado no Capítulo 5, segue os seguintes passos: Define-se o modelo conceitual através da linguagem UML, logo após exporta-se o modelo para a linguagem de intercâmbio XMI, posteriormente se faz uma tradução para a linguagem OWL e, por fim, pode-se utilizar desta para refinar o modelo e aplicar raciocínio e inferências. Os recursos descritos neste capítulo dão suporte a estas tarefas.

4.1 Ferramentas

As ferramentas de software que dão apoio aos processos de desenvolvimento e manutenção do sistema bem como o modelo conceitual em si tem papel fundamental para acelerar as tarefas de codificação, alteração e inferência além de ser fator preponderante nas atividades de comunicação, colaboração e consistência de formalismo.

4.1.1 Modelagem UML

Para esta tarefa existem ótimos softwares disponíveis no mercado. Alguns possuem aspectos com vantagens em relação a formalismos, outros possuem melhor visualização gráfica, outros possuem recursos diferenciados para descrever restrições e assim por diante. Entre os softwares mais populares, pode-se citar ArgoUML[25], JudeUML[26], Visual-Paradigm[27], Apollo for Eclipse[28], Poseidon For UML[29], MagicDraw[30] e NetBeans IDE[31]. De maneira geral, estas

25 http://argouml.tigris.org/
26 http://jude.change-vision.com/jude-Web/index.html
27 http://www.visual-paradigm.com/product/vpuml/
28 http://www.gentleware.com/apollo.html

ferramentas não exportam o modelo para um arquivo XMI em suas versões gratuitas. Para o caso específico deste trabalho, foi utilizada a ferramenta Poseidon For UML versão *Standard Edition* 6.0 à qual permite exportar o modelo através de uma distribuição de teste e foi a ferramenta utilizada por Dragan Gasevic na descrição de seu projeto UMLtoOWL[32], à qual será o artefato base para o mapeamento buscado por este trabalho.

4.1.2 Tradução XMI para OWL

De acordo com o projeto UMLtoOWL devemos utilizar um processador XSLT. A função de um arquivo XSLT é prover sintaxe e semântica para que um processador transforme um 'tipo' de arquivo XML em outro 'tipo' de arquivo também XML. Neste caso, XMI irá se transformar em OWL. Para os testes e experimentos realizados no mapeamento utilizou-se o processador Xalan-Java 2.7.1[33] que também faz parte dos requisitos do projeto UMLtoOWL. Deve-se atentar para a instalação e adição de variáveis de ambiente no sistema para que este funcione corretamente.

4.1.3 Manipulação dos Arquivos OWL

As atividades posteriores ao mapeamento excedem aos estudos deste trabalho, mas são justamente a justificativa para o desenvolvimento do mesmo, ou seja, busca se aqui uma técnica ou metodologia à qual facilite a modelagem conceitual com mais recursos gráficos e seja traduzida numa linguagem da Web semântica para que então possa se manipular o modelo ontológico. Para essas tarefas também existem boas ferramentas no mercado que criam, visualizam, manipulam, e inferem sobre ontologias, codificam em linguagens como RDF, RDFS, DAML+OIL e OWL, são baseadas em Java ou não, podem persistir os modelos em banco de dados, serem gratuitas ou pagas. Entre as mais populares e citadas na literatura correlata, destaca-se a ferramenta Protégé[34].

Protégé é uma ferramenta gratuita, baseada em Java para criação, visualização e manipulação de ontologias em modelos de *frames* ou descrições OWL, suporta especificação em XML, RDF, RDFS e OWL e também foi utilizada na importação do arquivo OWL após a tradução UMLtoOWL. Maiores detalhes da Protégé, outras ferramentas como OntoStudio[35],

---

29 http://www.gentleware.com/products.html
30 http://www.magicdraw.com/
31 http://www.netbeans.org/features/uml/index.html
32 http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/
33 http://xml.apache.org/xalan-j/
34 Foi desenvolvida e está sendo mantida pela equipe *Stanford Center for Biomedical Informatics Research* da Universidade de Stanford. Disponível em http://protege.stanford.edu/
35 http://www.ontoprise.de/de/en/home/products/ontostudio.html

OntoBroker[36], OILEd e Chimaera[37] além de um breve resumo comparativo pode ser visto no estudo de COSTA (2006).

4.2 Frameworks

Vale também registrar a contribuição e evolução de iniciativas que visam a dar suporte ao ciclo de vida das ontologias bem como raciocínio e inferências para uma gestão mais intuitiva dos sistemas computacionais. Entre estas vamos destacar Jena e pOWL à qual estão sendo desenvolvidas em Java e PHP respectivamente e que são linguagens amplamente utilizadas no desenvolvimento Web.

4.2.1 Jena

Sendo a linguagem mais utilizada na Web de acordo com o índice TPCI[38] de novembro de 2008, Java foi precocemente alvo de estudos para o desenvolvimento de sistemas para as tarefas do processo ontológico. Jena[39] é um *framework* para construção de aplicações semânticas que provê uma API para criar e manipular modelos conceituais nas principais linguagens ontológicas, possui suporte a armazenagem de modelos em memória ou persistente em banco de dados e um motor de inferência para SPARQL que é uma linguagem de consulta em documentos RDF. *Frameworks* como este, podem manipular os arquivos da representação conceitual bem como controlar os recursos do sistema baseado nas inferências e estratégias desenvolvidas, específicas do domínio modelado. Jena é uma ferramenta poderosa e gratuita. Também em COSTA (2006) pode-se encontrar mais detalhes de sua arquitetura e implementação.

4.2.2 pOWL

Também entre as linguagens mais utilizadas na Web e de fácil compreensão e implementação, a linguagem PHP é a base para a plataforma pOWL[40] à qual oferece uma API orientada a objetos para manipulação de arquivos RDF e OWL, de fácil implantação, possui sofisticado mecanismo de edição de dados e persistência em banco de dados, escalável através de seu conceito de *plug-ins*, permite consultas a base de dados através de linguagem RDQL além de ser liberada sob licença GPL.

---

36 Projeto migrou-se para o OntoStudio

37 http://www.ksl.stanford.edu/software/chimaera/

38 TIOBE Programming Community Index. Índice que faz o ranking das linguagens de programação mais utilizadas. Disponível em http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

39 http://jena.sourceforge.net/

40 http://powl.sourceforge.net/overview.php

4.3 Outros Recursos

Vários outros softwares, *frameworks*, métodos de desenvolvimento, motores de inferência, sites especializados em ontologias e conteúdo sobre Web semântica entre inúmeros recursos estão se multiplicando na Web com intuito de solucionar e aperfeiçoar abordagens para modelagem e manutenção de sistemas inteligentes, providos de metadados e contextualização bem definida. Projetistas e desenvolvedores de sistemas similares a estes podem se beneficiar de uma análise prévia a estes materiais e obter um modelo mais conciso, robusto e adequado às necessidades do domínio a ser tratado.

Além das técnicas, ferramentas e modelos abordados e apresentados até aqui, serão listados ainda, alguns recursos que de alguma forma contribuem para o desenvolvimento da Web semântica e que podem ser levados em consideração no projeto destes sistemas.

4.3.1 KSL Stanford[41]

Site Web do *Knowledge System Artificial Intelligence Laboratory of Stanford Universiry* a qual conduz pesquisas nas áreas de representação do conhecimento e raciocínio automatizado. Entre seus trabalhos atuais podem-se destacar tecnologias para possibilitar a Web semântica, raciocínio dedutivo, engenharia de ontologias entre outras tecnologias baseada em conhecimento.

4.3.2 Swoogle[42]

Swoogle é um motor de busca que descobre, analisa e indexa conhecimentos codificados em documentos na Web. Swoogle age sobre estes documentos e suas partes constituintes (como por exemplo, termos, indivíduos, triplas etc.) e registros de metadados significativos sobre estes. Também provê um serviço de acesso aos dados que pode ser utilizado por humanos ou de forma automática por sistemas de software para encontrar os termos e/ou documentos relevantes. Seu projeto havia sido dado como encerrado em 2006, mas em 2007 uma nova versão foi lançada e atualmente conta com mais de dez mil ontologias indexadas, sendo uma ferramenta útil no processo de analise de ontologias superiores e de domínio para incorporação no modelo sendo descrito.

41 http://www.ksl.stanford.edu/
42 http://swoogle.umbc.edu/

4.3.3 Chimaera[43] e OntoLingua[44]

São softwares desenvolvidos pelo time do KSL Stanford com propósito de criar, editar, navegar e utilizar ontologias.

4.3.4 W3C[45]

O *World Wide Web Consortium* é uma organização internacional formada por um consórcio de empresas que visam a padronização dos conteúdos na Web. A definição e normalização como padrão de técnicas, metodologias e linguagens para a Web semântica como para a Web como um todo deve ser acordada por este consórcio e então difundida para o mundo.

4.3.5 SUO WG IEEE[46]

O *Standard Upper Ontology Work Group* é grupo de trabalhos e estudos do IEEE para o desenvolvimento de um padrão que irá especificar uma ontologia superior para suportar aplicações computacionais como interoperabilidade de dados, aquisição e consulta de informação, inferência automatizada e processamento de linguagem natural.

4.3.6 OAEI[47]

OAEI do inglês *Ontology Alignment Evaluate Initiative* descreve uma API Java para apoio as tarefas de alinhamento taxonômico de ontologias.

4.3.7 FaCT[48]

O *Fast Classification of Terminologies* é um classificador baseado em lógica de descrição que pode ser usado para testar satisfatoriamente lógicas modais e prover raciocínio sobre bases de conhecimento.

4.3.8 Racer[49]

Renamed Abox and Concepts Expression Reasoner é um servidor robusto que trata indivíduos conceituais (ABox) em combinação com amplos e expressivos termos do modelo (TBox). Provê suporte a raciocínio e declarações de restrições além de definir uma linguagem própria de consulta com operadores e declarações especiais.

---

43 http://www.ksl.stanford.edu/software/chimaera/
44 http://www.ksl.stanford.edu/software/ontolingua/
45 http://www.w3.org/
46 http://suo.ieee.org/
47 http://oaei.ontologymatching.org/2008/align.html
48 http://www.cs.man.ac.uk/~horrocks/FaCT/
49 http://www.racer-systems.com/index.phtml

4.4 Conclusões

O Capítulo 4 descreve as ferramentas que foram utilizadas no processo de mapeamento UMLtoOWL (como pode ser melhor visualizado na Figura 4-1), além de prover uma pequena lista de alguns recursos, Web sites e ferramentas que contribuem nas tarefas de manutenção de ontologias e sistemas baseado em conhecimento.

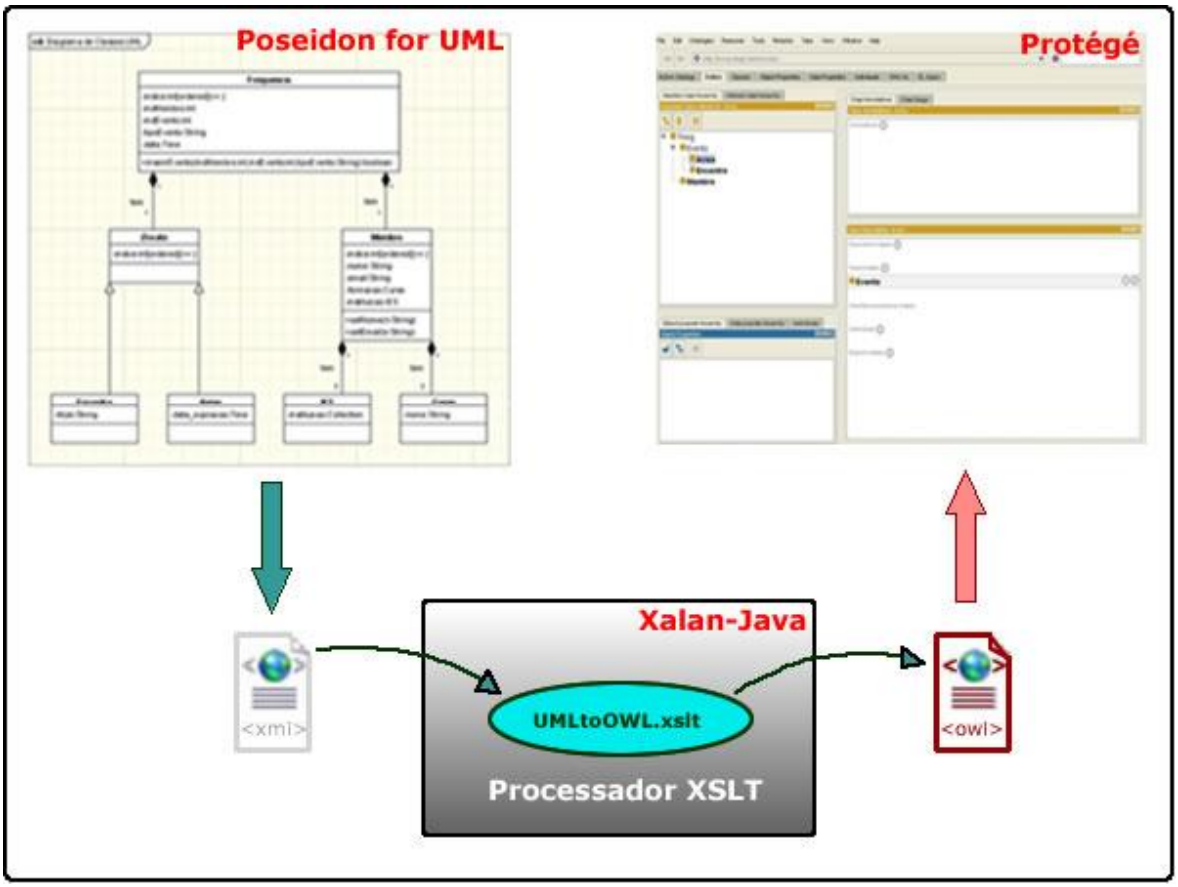


Figura 4-1 Resumo da tradução UMLtoOWL de GASEVIC (2004)

Um estudo mais exaustivo sobre todos os aspectos que envolvem o desenvolvimento de ferramentas, *frameworks* e plataformas para as chamadas SWApps (*Semantic Web Applications*) podem ser encontradas em CUNHA (2006). Neste trabalho, o autor apresenta uma figura na introdução a qual define como: "uma interpretação da lista de ferramentas ou tecnologias da Web Semântica de (FENSEL et al., 2003)" e que merece ser relatada aqui, pois contribui para a compreensão do papel do mapeamento UMLtoOWL dentro do contexto da Web semântica e as ferramentas citadas anteriormente, além de apresentar um esquema simples e resumido. A Figura 4-2 exibe esta interpretação.

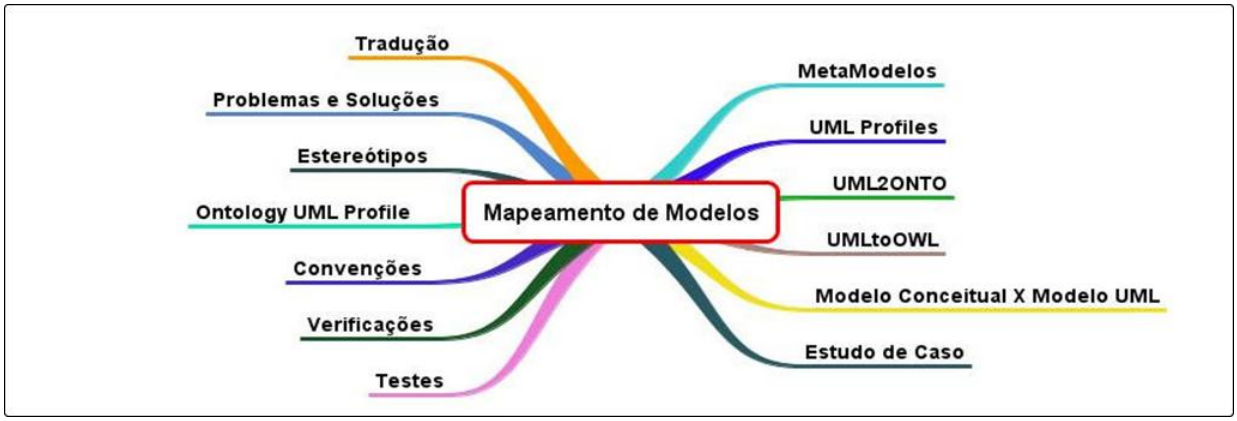
Figura 4-2 Interpretação das ferramentas Web Semânticas de Fensel (Adaptada de CUNHA, 2006)

Observa-se, de acordo com a Figura 4- e o mapeamento UMLtoOWL, que as ferramentas para a edição dos modelos em UML e os refinamentos nos documentos OWL se enquadram no módulo **Editores** e o processo de tradução XMI para OWL se caracteriza por ser um **Serviço de Tradução e Integração**. Como visto nos capítulos anteriores, outros modelos de representação do conhecimento e representação semântica podem ser categorizados em outros módulos dentro desta perspectiva como, por exemplo, a nomeação automatizada de diretórios e adição de metadados caracterizados pelas **Ferramentas de Anotação**, os *Reasoners* que compõe os **Serviços de Raciocínio**, os portais semânticos que provêem mecanismos para o **Acesso à informação** além de OWL ser a linguagem franca para **Representação de Ontologias** e a conseqüente semântica na Web.

No capítulo seguinte serão esboçados mais detalhadamente os aspectos relacionados ao mapeamento entre a representação do modelo conceitual e a representação semântica seguindo o método UML2ONTO (VICTORETTE, 2008).

Capítulo 5

MAPEAMENTO DE MODELOS



O mapeamento do modelo conceitual em uma representação para a Web semântica pode ser efetuado de várias maneiras como em um produto cartesiano dos modelos de representação do conhecimento pelos modelos de representação semântica, ou seja, pode-se modelar os conceitos em redes semânticas e mapeá-los em representação com linguagem RDF, ou *frames* em DAML+OIL e assim por diante. Entre essas inúmeras possibilidades, viu-se nos capítulos anteriores que algumas abordagens de representação possuem vantagens sobre outras em certos aspectos, mas podem não atender às necessidades para uma dada aplicação em outros aspectos devido aos diferentes níveis de expressividade entre as abordagens. Por isso, a escolha certa dos processos de representação, tradução e manutenção bem como o método de tradução entre linguagens devem ser diferentes para cada caso, podendo uns privilegiarem-se de linguagens, padrões de projeto e ferramentas consolidadas no mercado, enquanto outros podem necessitar de implementações e recursos específicos, impostos por políticas de negócio ou até mesmo por integração com sistemas legados.

De acordo com o que foi visto nos dois capítulos precedentes, a UML se concretiza como sendo o método escolhido para a representação do conhecimento devido aos avanços na direção de expandir os recursos que possibilitam a adição de regras aos elementos do modelo bem como semelhança entre itens de especificação do conhecimento e especificação semântica, possui ampla comunidade de desenvolvedores, exporta o modelo para um arquivo em linguagem baseada em XML além de conter ótimas ferramentas para codificação no mercado. OWL é a linguagem padrão para a Web semântica. Definidos estes pontos, será visto neste capítulo, um estudo de caso descrevendo e aplicando o método UML2ONTO para construção

de ontologias baseadas em modelagem UML, que serão então traduzidas em arquivos na linguagem OWL.

5.1 Contextualização

VICTORETTE (2008) utiliza procedimentos de algumas metodologias já conhecidas e descreve algumas tarefas em sua proposta de construção de ontologias usando UML, à qual chamou de UML2ONTO. Ele ressalta que existem problemas de semântica relacionados a esta atividade à qual engloba UML, OWL e metamodelos.

5.1.1 Metamodelos

Um metamodelo MOF (*Meta-Object Facility*) representa o elemento superior na arquitetura orientada a modelos (MDA). A MDA propõe a criação de modelos em diferentes níveis de abstração separando os interesses de implementação da arquitetura a ser desenvolvida. Um metamodelo é um modelo de especificação para uma classe do sistema em estudo, onde cada classe é um modelo válido expresso em certa linguagem. Um 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 e PEZZIN, 2006). Um metamodelo de definição de ontologias (ODM) deve ser concebido para compreender conceitos comuns às ontologias. A fim de fazer uso das capacidades gráficas da modelagem UML, um metamodelo ODM deve ter um UML *profile* correspondente. Este *profile* permite edição gráfica de ontologias usando diagramas UML bem como outros benefícios de se usar ferramentas CASE consolidadas. Como os modelos UML, os modelos ODM e a OWL são descritas em formato XML e um par de transformações XSL pode prover um mapeamento entre um ODM e a OWL.

Todos os detalhes, implicações e definições essenciais para a especificação destes metamodelos bem como toda a infra-estrutura necessária para formulação e utilização dos *profile*s para desenvolvimento de ontologias baseado em modelos UML podem ser encontrados em DJURIC, GASEVIC e DEVEDZIC (2006), base da proposta do UML2ONTO. Serão aqui descritas somente suas funcionalidades e seus papéis no processo de mapeamento e tradução das linguagens, já que não é foco deste trabalho as minúcias de arquitetura e especificação de metamodelos e *profile*.

5.1.2 UML Profiles

UML *profile*s são declarações genéricas de algumas características mais específicas de um domínio do modelo UML. Estes "perfis" especificam elementos padronizados, comuns ao modelo (OMG, 2008)[50]. São definidos usando estereótipos, valores etiquetados e restrições que são aplicados sobre alguns elementos, como classes, atributos, operações e atividades. É uma maneira de customizar e acordar o uso coletivo de elementos comuns do modelo UML em forma de extensões, e assim podem se definir melhores domínios específicos (exemplo, desenvolvimento de software, governamentais, educação, agro negócio).

Definido por DJURIC, GASEV IC e DEVEDZIC (2006), o OUP (*Ontology* UML *Profile*) é um *profile* para descrição de ontologias utilizando UML. Este *profile* foi utilizado para especificar a ontologia em UML no processo de criação de ontologias UML2ONTO. O anexo C contém uma tabela com uma breve descrição comparativa dos mais importantes metamodelos construtores em MOF, RDF, RDFS e equivalência no OUP.

5.2 A Proposta UML2ONTO

O método UML2ONTO de construção de ontologias baseado na modelagem UML proposto por VICTORETTE (2008) é descrito resumidamente através do esquema da Figura 5-1.

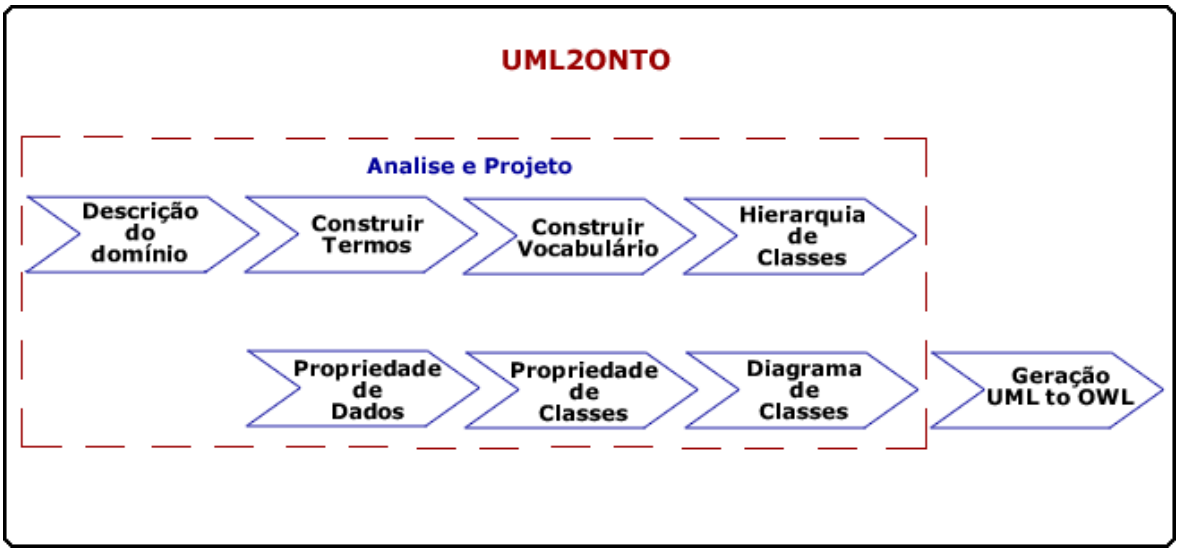


Figura 5-1 Descrição esquemática do método UML2ONTO (adaptado de VICTORETTE, 2008)

Este esquema será seguido e brevemente detalhado na construção de um estudo de caso.

[50] http://www.omg.org/technology/documents/profile_catalog.htm

5.2.1 Descrição do Domínio

Na descrição do domínio deve se fazer um texto ou um relatório utilizando, se possível, técnicas de engenharia de software, mais especificamente engenharia de requisitos, para elucidar os pontos relevantes do sistema a ser modelado. Para este estudo de caso não será descrito todo o texto e relatório disposto para o sistema modelado como um todo por não haver necessidade para a demonstração do mapeamento que se propõe. Utiliza-se a breve descrição de cenário a qual os membros do Ramo Estudantil IEEE UFJF freqüentam as reuniões semanais, suas presenças são registradas na ata do encontro contando como freqüência às atividades do Ramo e avisos são emitidos pela diretoria, que devem ser confirmados a leitura pelo membro, também como forma de contagem na freqüência.

5.2.2 Construir Termos

Devem se identificar os termos e relações nos documentos gerados no passo anterior, utilizando quase sempre a regra que substantivos são candidatos a se tornarem termos e verbos são candidatos a se tornarem relações. Devem se anotar todos os termos e relações identificadas mesmo que estes possam mudar de nome ou serem descartados futuramente. A Tabela 5-1 mostra um exemplo de descrição de termos e relações identificadas e sugeridas no estudo de caso.

Tabela 5-1 Alguns termos e relações sugeridos no estudo de caso

Termos Sugeridos	Relações Sugeridas
Membros	deveFreqEncontro
Evento	deveLerAta
Encontro	deveLerAviso
Aviso	temFrequencia

5.2.3 Construir Vocabulário

Devem-se selecionar os termos que são realmente conceitos no domínio sendo especificado. VICTORETTE (2008) neste ponto atenta para a possibilidade de erro na definição do que é uma classe na ontologia, podendo assim gerar classes errôneas e desnecessárias. Deve-se ter em mente que uma classe é uma entidade que representa uma coleção de elementos com propriedades similares. Nesta tarefa também se deve atentar para a problemática de semântica

pessoal e o comprometimento ontológico (*ontological commitment*) quando a ontologia está sendo desenvolvida em grupo.

5.2.4 Construir Termos

Identifica-se neste passo a hierarquia taxonômica dos conceitos que geralmente é descrito por categorização do tipo é_um (is_a).

Tabela 5-2 Hierarquia de Classes do estudo de caso

Classes	Subclasses
Evento	Encontro, Aviso
Membro	Presidente, Vice-Presidente, Secretário, Apoio

Neste passo pode ser aconselhável a utilização da ferramenta de modelagem UML para facilitar a visualização de modelos grandes e complexos, utilizando o diagrama de classes. A Figura 5-2 mostra a descrição da Tabela 5-2 codificada na ferramenta Poseidon for UML construída sobre as bases do *profile* OUP.

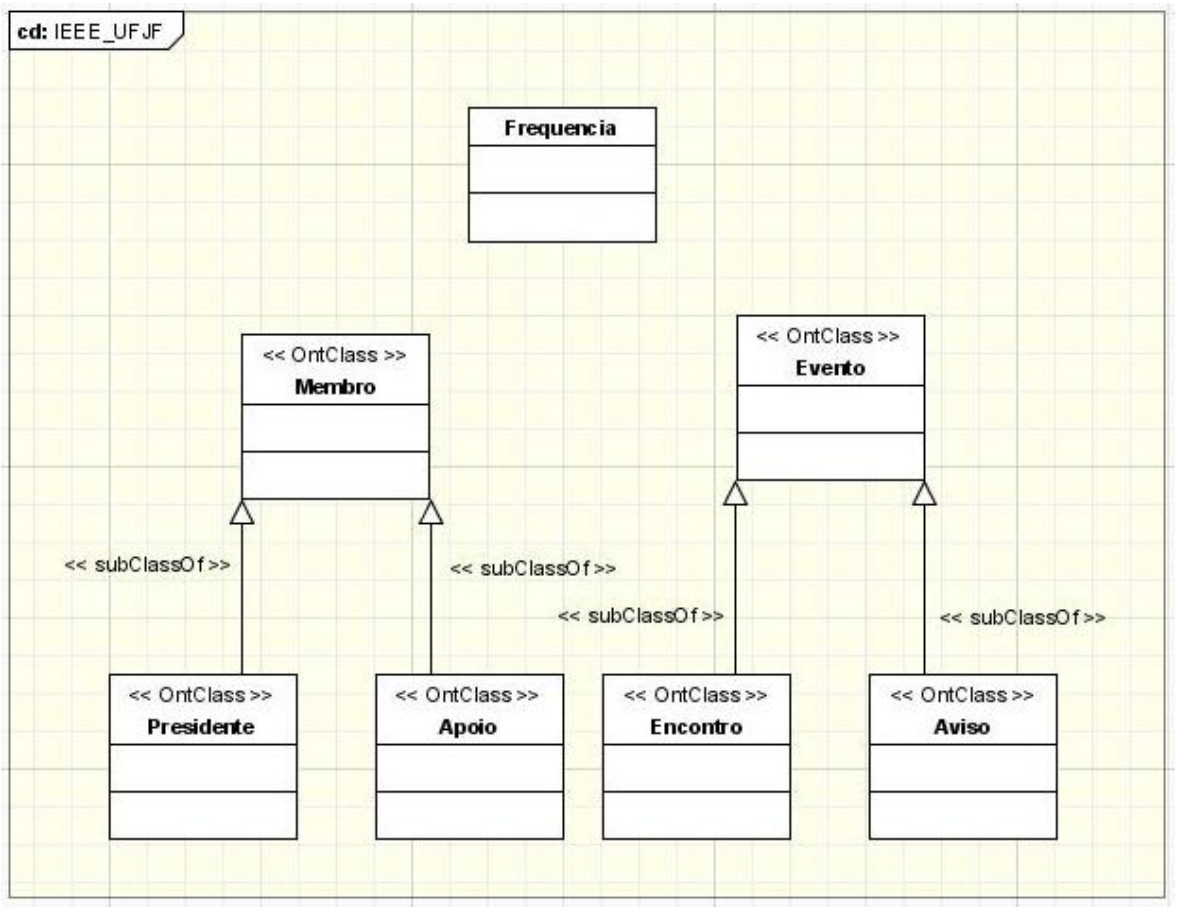


Figura 5-2 Classes do modelo conceitual do estudo de caso baseado no *profile* OUP

5.2.5 Propriedade de Dados

Neste passo se descreve os atributos de classe, propriedades e associações. Uma propriedade é uma relação entre um sujeito e um objeto. Ao se definir uma propriedade podem-se estipular algumas facetas como cardinalidade, tipos de valores, valores padrão, valores mínimos e máximos entre outros. A Figura 5-3 mostra alguns atributos que foram definidos no exemplo.

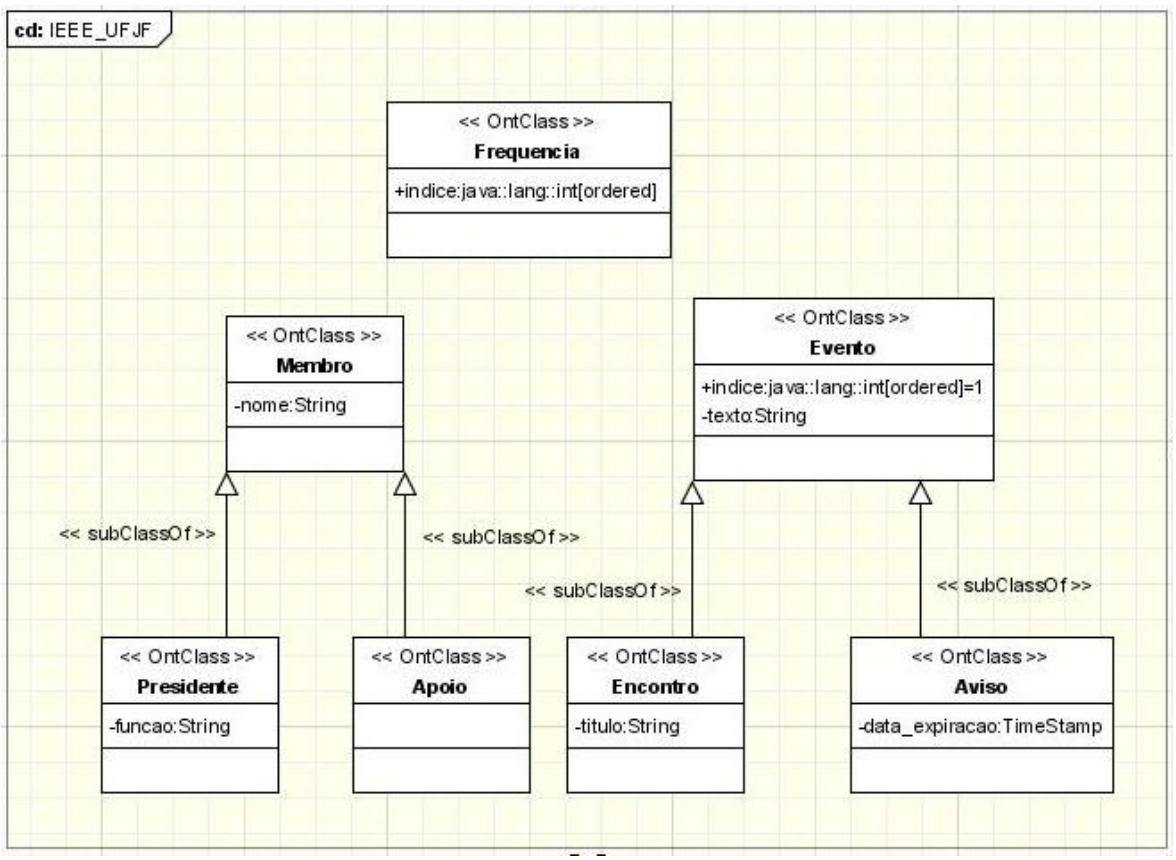


Figura 5-3 Classes do modelo conceitual com alguns atributos

5.2.6 Propriedade de Classes

De acordo com VICTORETTE (2008), especificar propriedade de classes é definir a propriedade a qual se tem o range (intervalo) como sendo uma classe. Deve se estabelecer uma relação entre uma classe domínio e uma classe range.

Tabela 5-3 Definição de algumas propriedades de classes no estudo de caso

Domínio	Propriedade	Range
Freqüência	temMembro	Membro
Freqüência	temEvento	Evento

O autor ainda destaca a similaridade de propriedade com os conceitos de atributos e associações em OO. Na modelagem conceitual as propriedades não dependem de uma classe específica e fazem parte do modelo como um todo.

5.2.7 Diagrama de Classes

Caso os diagramas não tenham sido criados durante o passo de definir a hierarquia de classes deve-se utilizar a ferramenta de modelagem para criar o diagrama de classes a partir dos estereótipos definidos no OUP. Depois se devem definir as relações através das propriedades de dados e propriedades de classes. Neste momento é importante lembrar das diferenças entre a modelagem conceitual em UML e a modelagem UML tradicional, a qual tem uma visão orientada a objetos, evitando assim que não haja confusões nas descrições das relações.

A Figura 5-4 mostra a especificação conceitual a qual pode ser observado algumas relações que se originaram de atributos apresentados na Figura 5-3, sendo que essa não é uma regra geral. Devem-se notar outras definições que foram necessárias para que se tenha os elementos necessários para a tradução como foi o caso da definição de tipo de dados date, string, int e boolean com a utilização do estereótipo <<DataType>> para se definir os intervalos de algumas propriedades de dados do modelo conceitual.

Alguns outros elementos foram criados para se testar e avaliar o mapeamento e tradução de algumas definições de modelos ontológicos como, por exemplo, elemento com propriedade funcional, simétrica e cardinalidade. Como a utilização do OUP na criação de ontologias é um método recente e ainda pouco explorado, foram necessários vários refinamentos no diagrama para que a tradução gerasse um arquivo OWL condizente com o cenário descrito, com especificações ontológicas e definido com os estereótipos do OUP.

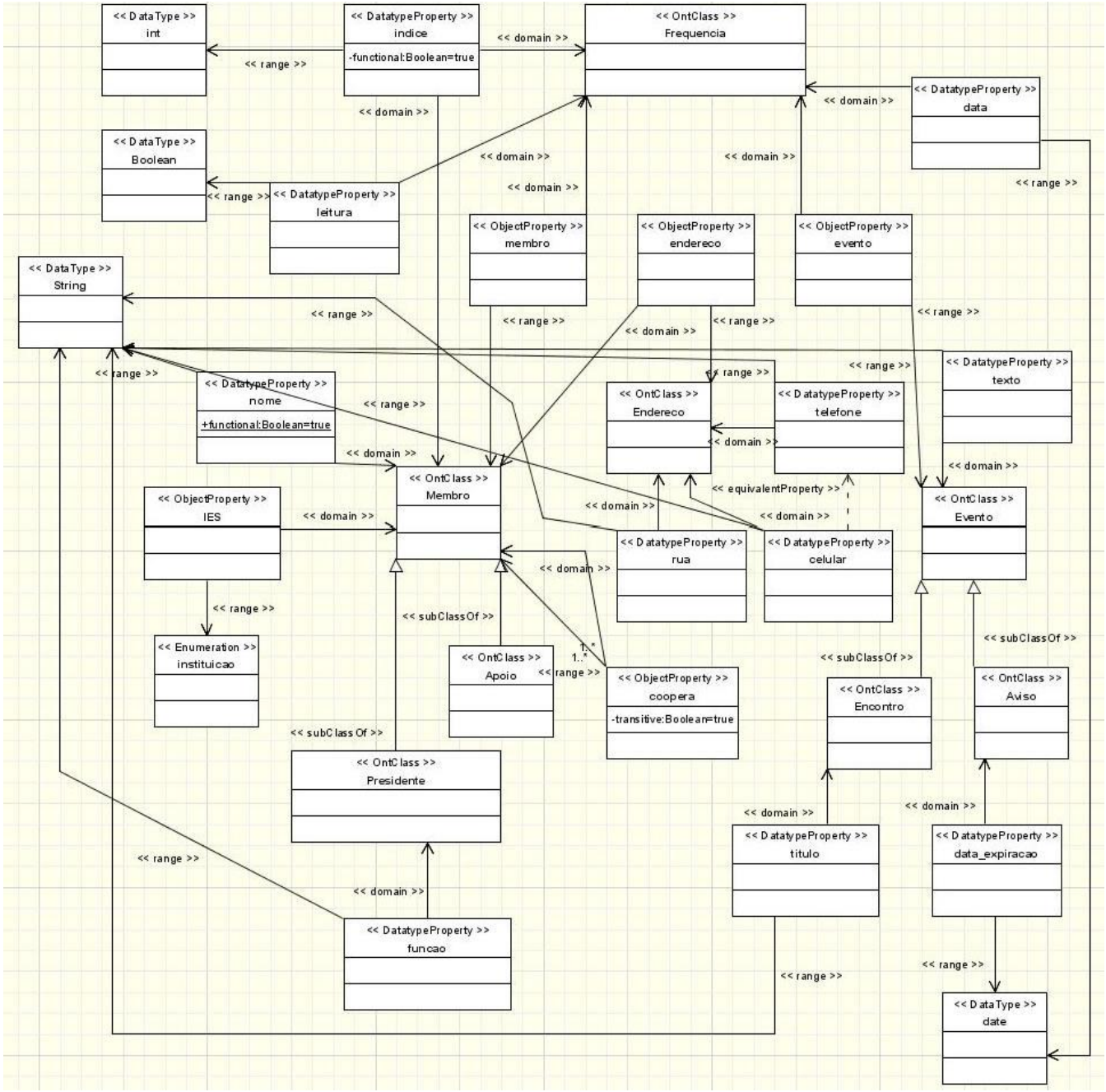


Figura 5-4 Diagrama de classes conceituais e algumas relações do estudo de caso UML to OWL

5.2.8 UML to OWL

Neste passo é feita a tradução do arquivo XMI em um arquivo OWL a qual futuramente pode ser importada por uma ferramenta de edição de ontologias, neste caso Protégé. Deve-se observar a inserção e descrição de alguns elementos que foram necessários para que a tradução possa expor um arquivo com descrições lógicas, utilizando para isto os estereótipos definidos, como é o caso de definições para os tipos de dados **int**, **string**, **date** e **boolean** com o estereótipo <<DataType>>. Estes e outros elementos foram descritos para

serem testados e verificados quanto a tradução, avaliação e comparação com resultados esperados. A Figura 5-5 mostra o arquivo OWL gerado após a tradução e importado pela ferramenta Protégé onde se tem na aba marcada com o número 1, a hierarquia das entidades do modelo.

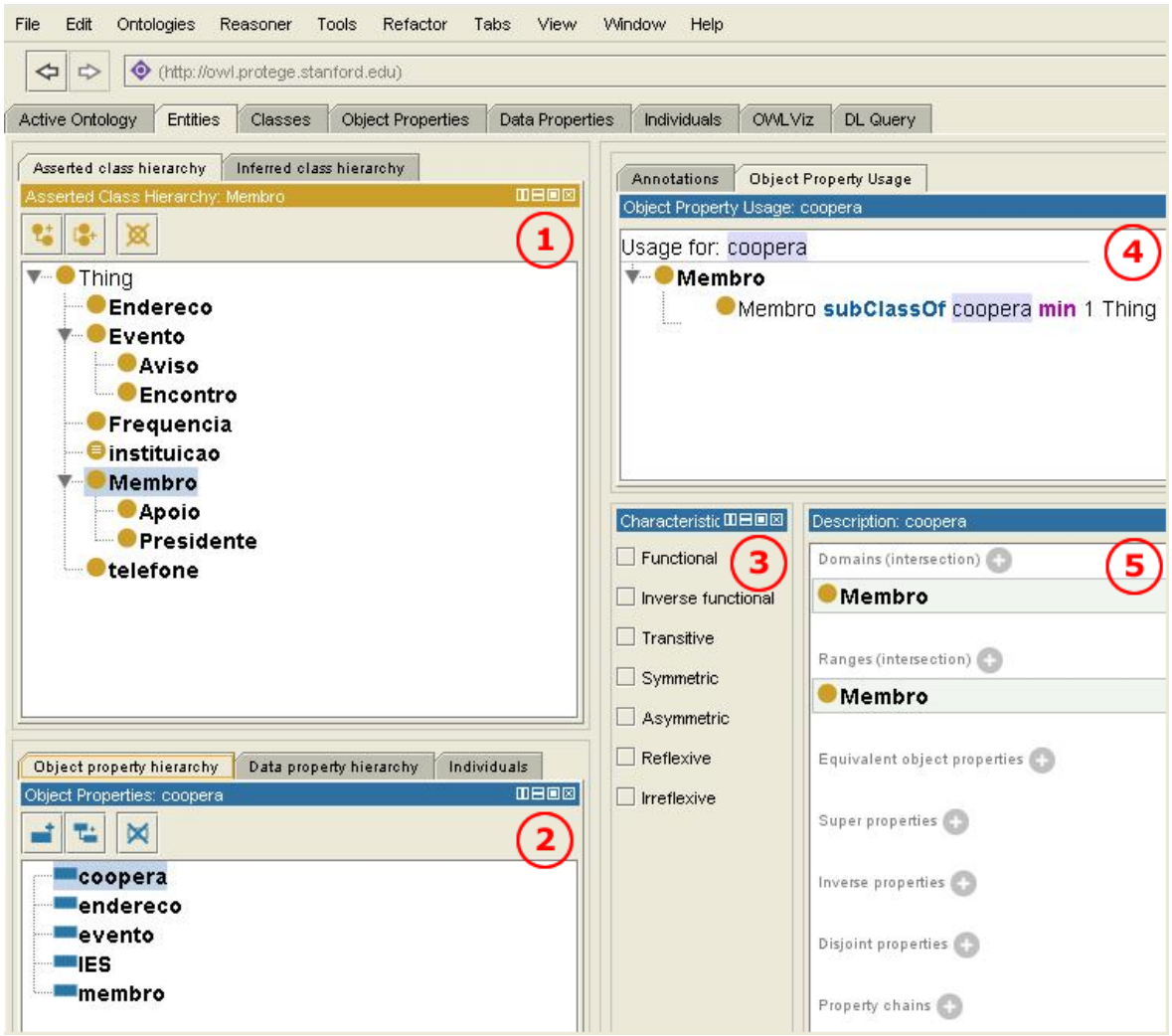


Figura 5-5 Arquivo OWL traduzido e importado pela ferramenta Protégé

Na aba marcada com o número 2, se tem a lista das propriedades de objetos <<ObjectProperty>> que foram descritas no modelo. A propriedade **coopera** foi adicionada com intuito de verificar o mapeamento para o caso de descrições do tipo transitivas, utilizando atributos estereotipados com <<transitive>>. Observa-se na aba número 3 que a característica funcional de coopera não foi devidamente definida. Na aba de número 4, tem-se a lista de usos do termo **coopera** na ontologia. A declaração diz que a classe **Membro** é uma subclasse de **coopera** tendo cardinalidade mínima de um objeto da classe **Thing**. De certa forma essa não é uma declaração correta para o domínio, uma vez que a classe de domínio é um Membro e não

um Thing, ou seja, algumas inferências sobre a ontologia podem retornar fatos e premissas infactíveis ao domínio.

Outro problema de semântica como este pode ser observado na Figura 5-6 onde se tem na aba número 1 a lista das propriedades de dados <<DatatypeProperty>>. Na aba número 2 tem-se os usos do termo **indice** onde diz que a classe **Frequencia** e a classe Membro são subclasses de **indice** que tem cardinalidade exata de um objeto do tipo literal. Logo na aba número 3 tem-se a declaração dos domínios que se aplicam o termo **indice**. Observou-se neste ponto também que a lista de domínios foi formada utilizando o operador **or**, ao invés do operador **and**, o que causa outra inconsistência no modelo. O termo índice é aplicável na classe Freqüência **e** na classe Membro, e não numa classe **ou** em outra. Mais abaixo uma declaração de **int** como sendo o intervalo (*range*) da propriedade.

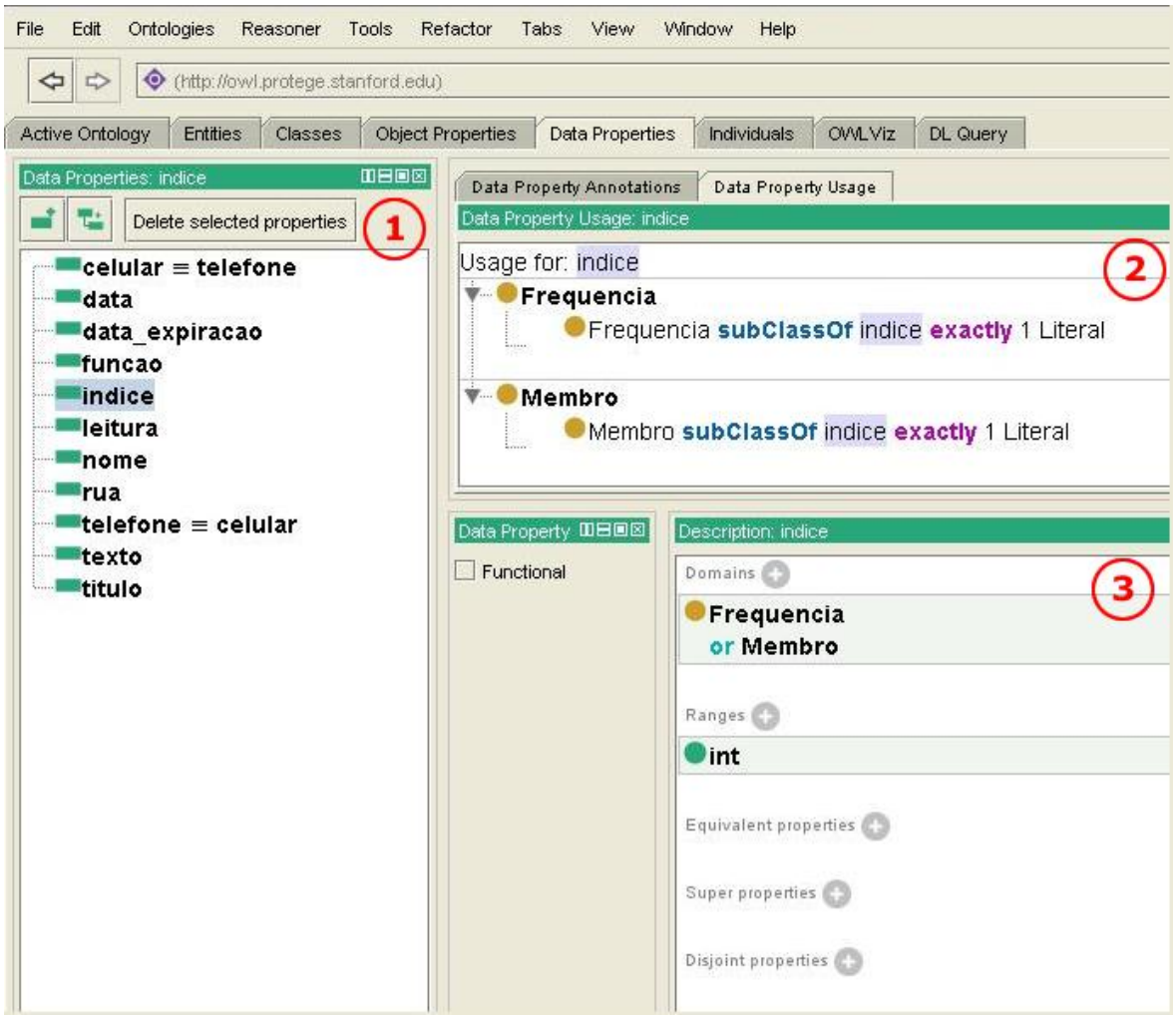


Figura 5-6 Outra visão do código OWL gerado pela tradução UMLtoOWL

Como dito no final da seção anterior, vários refinamentos tiveram que ser efetivados com intuito de familiarizar com o processo de descrição baseado na UML e gerar um arquivo OWL

que melhor representa o cenário descrito. Para esta tarefa então, deve-se repetir o processo: altera-se o modelo na ferramenta UML e exportá-lo para um arquivo XMI. Logo após, faz-se a tradução utilizando o processador XSLT para um arquivo OWL, à qual pode ser manipulado pela ferramenta ontológica. Este processo é repetido até que o modelo esteja de acordo com as necessidades predefinidas. O código para execução do processador é bem simples e definido como na Figura 5-7.

```
java org.apache.xalan.xslt.Process -in arq_de_entrada.xmi -xls
UMLtoOWL.xslt -out arq_de_saida.owl
```

Figura 5-7 Linha de comando para execução do conversor UMLtoOWL

Estas iterações também devem ser efetuadas caso haja refinamentos no arquivo de mapeamento (UMLtoOWL.xslt) e atualizações dos estereótipos OUP.

5.3 Conclusões

O método UML2ONTO de construção de ontologias baseado na modelagem UML provido dos mecanismos de tradução do método UMLtoOWL descreve um guia simplificado para o desenvolvimento de modelos conceituais utilizando OUP à qual estes modelos podem ser traduzidos em linguagem da web semântica. O *profile* OUP definido por GASEVIC (2004) possui especificação de 38 estereótipos (**ANEXO C**) que caracterizam os principais conceitos, definições e relações ontológicas. Estes estereótipos devem ser usados como construtores para definir os elementos do domínio utilizando modelagem UML e servirão de *input* para a tradução. Para alguns poucos aspectos do mapeamento o conversor UMLtoOWL proporciona o tratamento adequado mas com resultados ainda pouco confiáveis uma vez que mudando-se pouco na estrutura do diagrama UML percebeu-se grandes alterações na ontologia gerada. Alguns recursos descritos no modelo UML ainda não estão sendo capturados e tratados pela tradução.

Outro ponto importante e de destaque na construção de modelos conceituais são os indivíduos ou mecanismos de instanciação de classes. Eles permitem descrever entidades mais especificas e de caráter diferenciado do modelo. Na descrição do método UMLtoOWL DJURIC, GASEVIC e DEVEDZIC (2006) ressaltam este aspecto e relatam que OUP foi definido para que seja especificado em ferramentas UML, as quais permitem definir elementos Objetos UML em conjunto com elementos Classes UML em diagramas de classes. Como a ferramenta *Poseidon for UML*, a qual foi utilizada para o mapeamento deste estudo de caso, não permite a criação de elementos Objeto UML juntamente com elementos Classe UML no

mesmo diagrama de classes, outro mecanismo foi proposto, aplicado e está sendo refinado para sanar o problema de especificação de instâncias de classes conceituais.

O método consiste de introduzir o estereótipo <<Individual>> disponível a Classes UML e através da ligação <<instanceOf>> com a classe conceitual à qual pertence o indivíduo, define-se a instanciação, ou seja, para se criar uma instância de uma classe conceitual deve se criar uma Classe UML com o nome da instância e estereótipo <<Individual>> ligada à sua classe de domínio pela relação <<instanceOf>>. Para a identificação da instância e relação à sua classe no processo de tradução, refinamentos foram feitos no arquivo UMLtoOWL.xslt gerando a versão 2.0 para o mesmo. De imediato, a tradução reconhece o novo elemento de instância e seu nome, mas ainda não reconhece a classe de domínio para que a instância seja atrelada a ela. Assim foi definido um nome convencional (AClasse) para a classe conceitual com a finalidade de não parar a execução da tradução. Em refinamentos posteriores deve-se atribuir a classe reconhecida na indicação da relação de tipo <<instanceOf>> no lugar deste nome convencionado. A mostra um exemplo da especificação proposta para definição de instâncias conceituais utilizando Classe UML e estereótipos OUP.

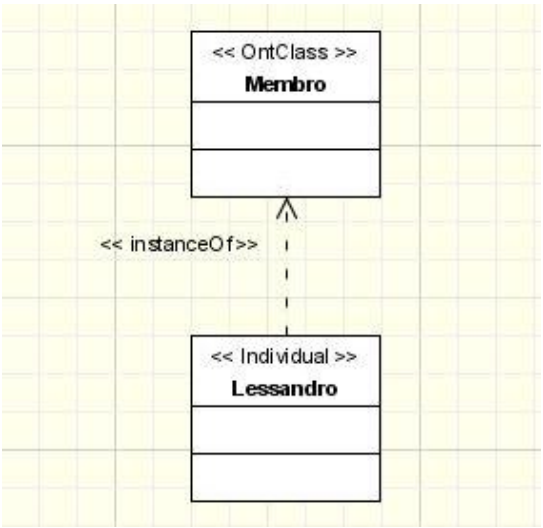


Figura 5-8 Definição de instâncias conceituais utilizando Classes UML expandido estereótipos OUP

A Figura 5-8 mostra o indivíduo gerado a partir da definição do elemento com estereótipo <<Individual>> e mapeado em OWL efetuado pelo arquivo UMLtoOWL_2_0.xslt.

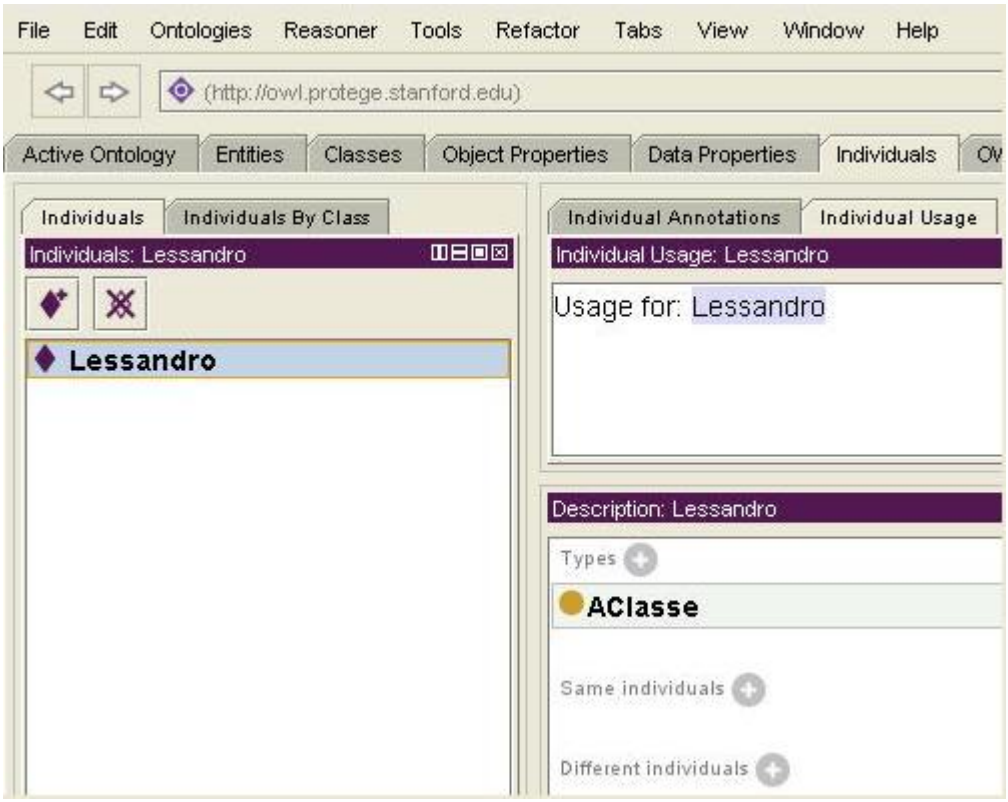


Figura 5-9 Instância de classe conceitual definida pelo estereótipo <<Individual>> no modelo UML

De acordo com o mapeamento proposto e a Figura 5-9 tem-se que a tradução deve reconhecer que a classe **Membro** é a classe conceitual do indivíduo **Lessandro** e atrelar suas características a este.

Capítulo 6

RESULTADOS FINAIS

Este capítulo resume os pontos mais relevantes que foram observados durante o estudo dos modelos de representação do conhecimento, modelos de representação semântica, a tradução e mapeamento do modelo conceitual em linguagem da Web semântica bem como uma breve relação de artefatos que podem motivar estudos e trabalhos futuros.

6.1 Conclusões

Representar conhecimento e traduzi-lo em linguagem da Web semântica é um campo vasto, com grandes desafios e benefícios para uma gestão mais intuitiva de sistemas inteligentes. Entre as abordagens mais populares e abrangentes para representação de conhecimento, a modelagem UML vem tomando sua posição frente a outros métodos, principalmente por possuir ótima apresentação visual, grande comunidade de desenvolvedores e excelentes ferramentas de edição disponíveis no mercado além de possibilitar a aplicação de restrições e outros tipos de descrições lógicas ao modelo. Estas ferramentas também se destacam por possuírem capacidade de exportar o modelo para um arquivo baseado em XML, à qual pode ser traduzido em linguagens ontológicas, que também são baseadas em XML, à qual é a metalinguagem franca da Web.

Deve-se analisar e considerar também outros modelos de representação do conhecimento em fases diferentes do ciclo de vida das ontologias e do sistema como um todo, por exemplo, utilizando o método UML2ONTO para criação de ontologias temos que no passo 1 (seção 5.2.1) deve-se construir textos, relatórios e todo tipo de documento que ajude aos desenvolvedores a identificar os principais termos, conceitos e relações do domínio. Para tal, podem se confeccionar seqüências de declarações lógicas em definições que requerem alto teor de formalismo, redes semânticas que propiciam boa visão de hierarquia de conceitos, *frames* que facilitam a identificação de classes que contém muitas propriedades e procedimentos que devem ser executados como resultado de cálculos, mapas conceituais entre outros. Entre estes métodos alguns podem contribuir tanto na organização do conhecimento como na adição e representação semântica como é o caso de tesauros e nomeação e organização hierarquizada de diretórios.

Para uma representação conceitual mais formalizada, possibilitando raciocínio e inferências lógicas, as ontologias se apresentam como o meio mais adequado, de senso comum e sendo amplamente aplicadas em sistemas que tem compromisso semântico como parte relevante de seus requisitos de desenvolvimento e qualidade. Para este fim temos a linguagem OWL, baseada em XML e que é formada pela composição de um conjunto próprio de construtores, além de elementos das linguagens RDF, RDFS e DAML+OIL.

O método UML2ONTO define um conjunto de passos a ser seguidos na construção de ontologias baseada na modelagem UML, utilizando-se dos estereótipos do OUP para a definição dos conceitos e relacionamentos, a qual serão traduzidos no último passo do método para um arquivo OWL de acordo com as regras de mapeamento definidas no método conversor UMLtoOWL. Muitos pontos da tradução e convenções de mapeamento ainda estão indefinidos ou necessitando de refinamentos para uma conversão com maior grau de acurácia e desvinculação de declarações específicas de algumas ferramentas sendo forte motivo para estudos e experimentos futuros como o problema de instanciação de classes conceituais.

Por fim deve-se ressaltar a importância de boas práticas na adição e manutenção de metadados e extensões semânticas em dados e recursos do sistema procurando sempre utilizar de metodologias e linguagens que sejam padrões recomendados pelo W3C.

6.2 Trabalhos Futuros

Visto que os maiores problemas e obstáculos encontrados para o mapeamento estão relacionados ao processo de tradução, as seguintes tarefas podem ser listadas como ponto de partida e fonte de motivação:

- Concluir o processo de tradução de instâncias de classes conceituais declaradas em elementos Classe UML com estereótipo <<Individual>>;
- Estudar os detalhes da especificação OUP, as aplicações e implicações no desenvolvimento de ontologias;
- Analisar, alterar, testar, estender e validar as regras de mapeamento do arquivo UMLtoOWL_2_0.xslt;
- Desenvolver casos de teste para validação, definição e redefinição de estereótipos do OUP;
- Aplicar de vários casos de inferências para descoberta de conhecimento e inconsistências geradas no modelo após a tradução;

REFERÊNCIAS BIBLIOGRÁFICAS

BAYKAN, E.; HENZINGER, M.; WEBER, I. *Web Page Language Identification based on URLs*. 34Th International Conference on Very Large Data Base, 2008.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. *The Semantic Web, 2001*. Em: Scientific American (edição 50), maio de 2001. Disponível em http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

BREITMAN, K. K. *Web Semântica: A internet do Futuro*. Editora LTC, Rio de Janeiro, 2005.

BRODIE, M. L. *Computer Science 2.0: a new world of data management*. Very Large Data Bases. Proceedings of the 33rd international conference on Very large data bases. 2007.

CALVANESE, D.; LEMBO, D. *Ontology-based Data Access.* Em http://www.inf.unibz.it/~calvanese/teaching/ISWC-2007-tutorial-obda/

COSTA J. C. S. *O Uso de Ferramentas no Suporte à Web Semântica.* Monografia de Final de Curso. Universidade Federal de Juiz de Fora. Juiz de Fora, MG. Março, 2006.

COUGO, P. *Modelagem Conceitual e Projeto de Banco de Dados*. Editora Campus, Rio de Janeiro, 1999.

CUNHA, L. M. *Um Framework de Aplicações para a Web Semântica*. Tese de Doutorado. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, RJ. 2006.

DJURIC, D; GASEVIC, D e DEVEDZIC, V. *Model Driven Architecture and Ontolgy Development*. Germany, Springer. 2006.

FENSEL, D. *The semantic Web and its languages*. IEEE Intelligent Systems. v. 15, n. 6, p. 67-73, nov./dec. 2000.

FURGERI, S. *Representação de Informação e Conhecimento*. Pontifícia Universidade de Campinas. 2006.

GAMA, T. A. *Estudo de Caso: Um Repositório de Serviços Web Semânticos para Simulações em Eletrofisiologia Celular*. Trabalho de conclusão de curso. Universidade Federal de Juiz de Fora. Juiz de Fora. Juiz de Fora, MG. Julho, 2008.

GASEVIC, D. *UMLtoOWL: Converter from UML to OWL*. 2004. Disponível em http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/ Acessado em 30/11/2008

GUARINO, N. *Understanding, Building, and Using Ontologies*. International Journal of Human and Computer Studies. v. 46, n. 2-3, p. 293-310, Fevereiro, 1997.

GUARINO, N. *Formal Ontology in Information Systems*, Proceedings of FOIS'98, Trento, Italy. Amsterdan, IOS Press, pp. 3-15, 6-8 Junho, 1998.

GRUBER, T. R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. International Journal Human-Computer Studies, 43(5-6):907-928, 1995.

HORROCKS, I.; et al. OIL: *An Ontology Infrastructure for the Semantic Web*. Special Inssue, IEEE Intelligent System. September, 2001.

IKEMATU, R. S. Gestão de Metadados: Sua Evolução na Tecnologia da Informação. Revista de Ciência da Informação vol. 2, nº. 6, dezembro de 2001. Disponível em http://dici.ibict.br/archive/00000308/01/Gest%C3%A3o_de_metadados.pdf. Acessado em 21 de novembro de 2008.

LEITE, M. e RAHAL Jr., N. A. S. *Programação Orientada ao objeto: uma abordagem didática*. Revista da Informação e Tecnologia - CCUEC, Unicamp. Revista da Informação e Tecnologia. Agosto, 2002. Disponível em http://www.ccuec.unicamp.br/revista/infotec/index2.html

MORAIS, E. A. M. *O Estado da Arte no Estudo das Ontologias*. Simpósio de Estudos e Pesquisas: Educação, Cultura e Produção do Conhecimento da FASAM. Goiânia, GO. 2006.

NOVAK, J. D. e GOWIN, D. B. *Aprender a aprender*. Lisboa: Plátano Edições Técnicas, 1999.

OLIVEIRA, N. Q. *Ontologias para Aplicações em Grids Computacionais Semânticos.* Trabalho de conclusão de curso.Universidade Federal de Juiz de Fora. Juiz de Fora, MG. Janeiro, 2005.

PACHECO, A. G. *Subsídios para a Engenharia de Ontologias na Web Semântica*. Trabalho de conclusão de curso. Universidade Federal de Juiz de Fora. Juiz de Fora, MG. Fevereiro, 2004.

PALMER, F. R. *A semântica*. Edições 70, Lisboa, 1976.

REIS, T. T. *Desenvolvimento Web com Uso de Padrões: tecnologias e tendências*. Trabalho de conclusão de curso. Universidade Federal de Juiz de Fora. Juiz de Fora, MG. 2007.

RIBEIRO, M. H. F. *Disponibilizando Raciocínio para Consultas na Web Semântica*. Trabalho de conclusão de curso. Universidade Federal de Juiz de Fora. Juiz de Fora. Fevereiro, 2003.

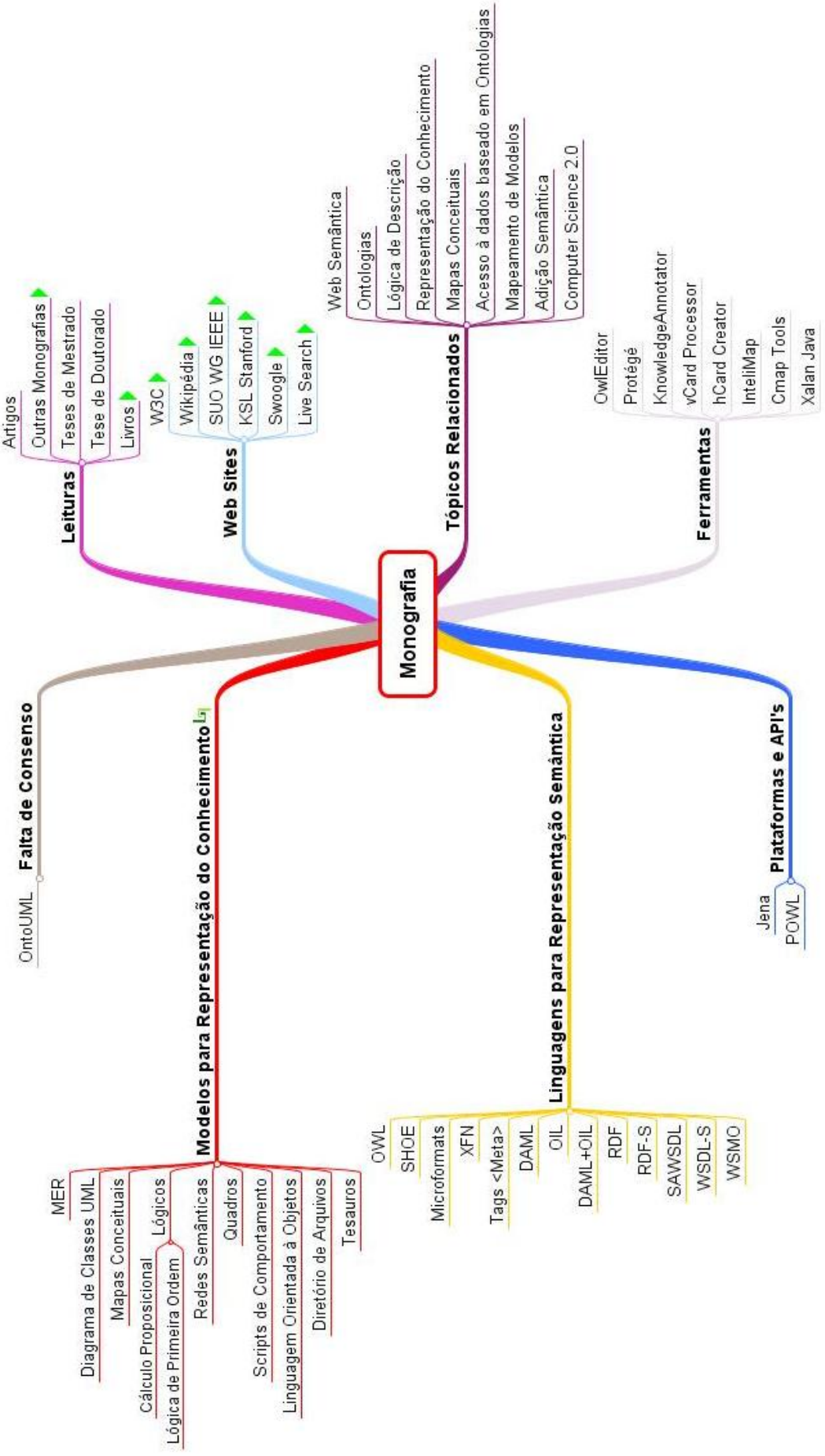
SALES, R. e CAFÉ, L. *Semelhanças e diferenças entre Tesauros e Ontologias.* Trabalho de conclusão de curso. Universidade Federal de Santa Catarina, 2008. Disponível em http://www.dgz.org.br/ago08/Art_02.htm. Acessado em 21 de novembro de 2008.

VICTORETTE, G. W. D. B. *O Processo de Construção de Ontologias Baseado na Modelagem UML.* Trabalho de conclusão de curso. Universidade Federal de Santa Catarina. Florianópolis, 2008.

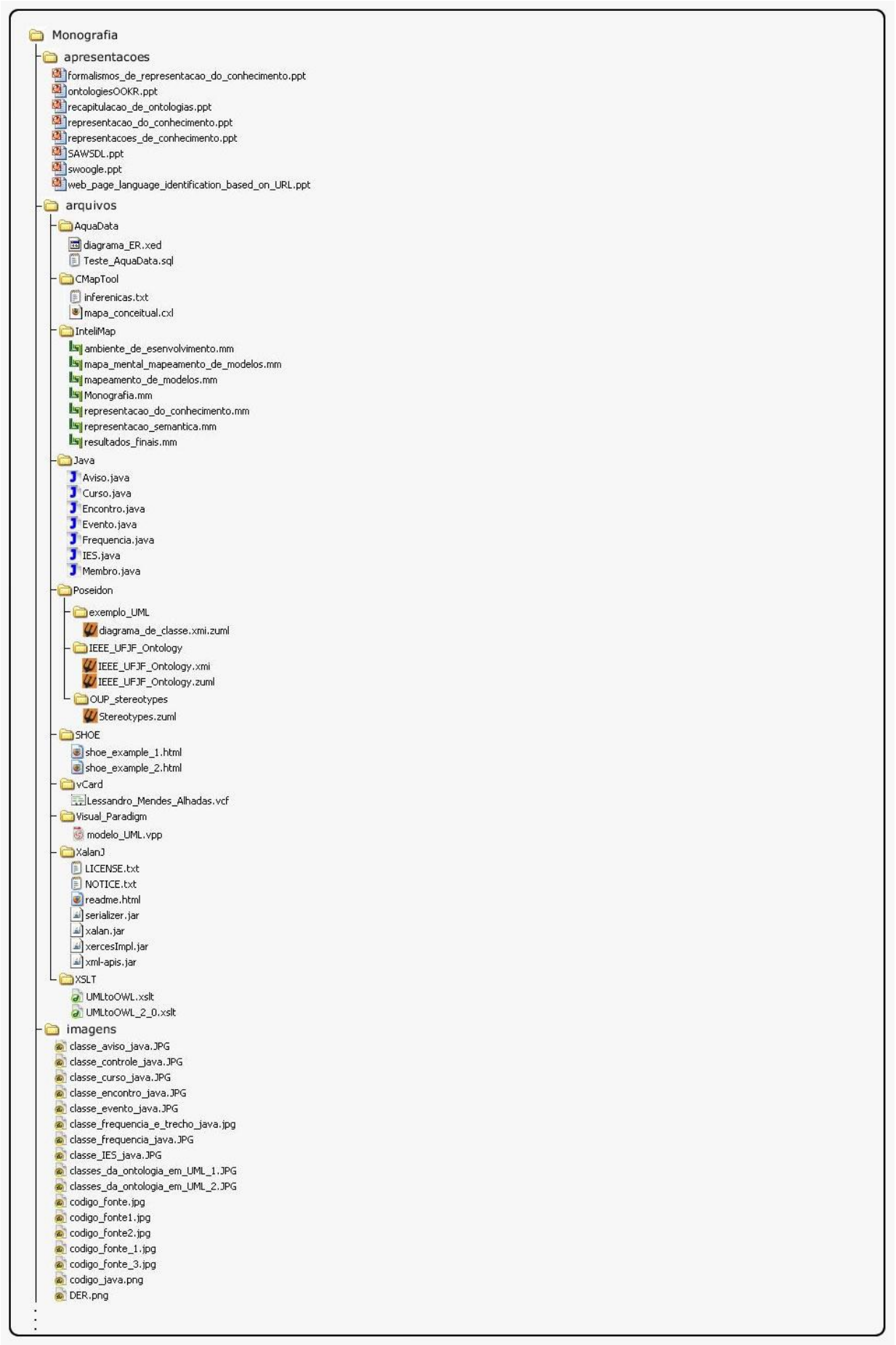
TAVARES, R. *Construindo Mapas Conceituais.* Departamento de Física, Trabalho de conclusão de curso. Universidade Federal da Paraíba, 2007.

SILVA, J. B.; SAMPAIO, M.; PEZZIN, J. *Usando Ontologias na Construção de modelos MDA.* UNIFACS. Salvador, BA. 2006.

ANEXO A – Mapa Mental da Monografia

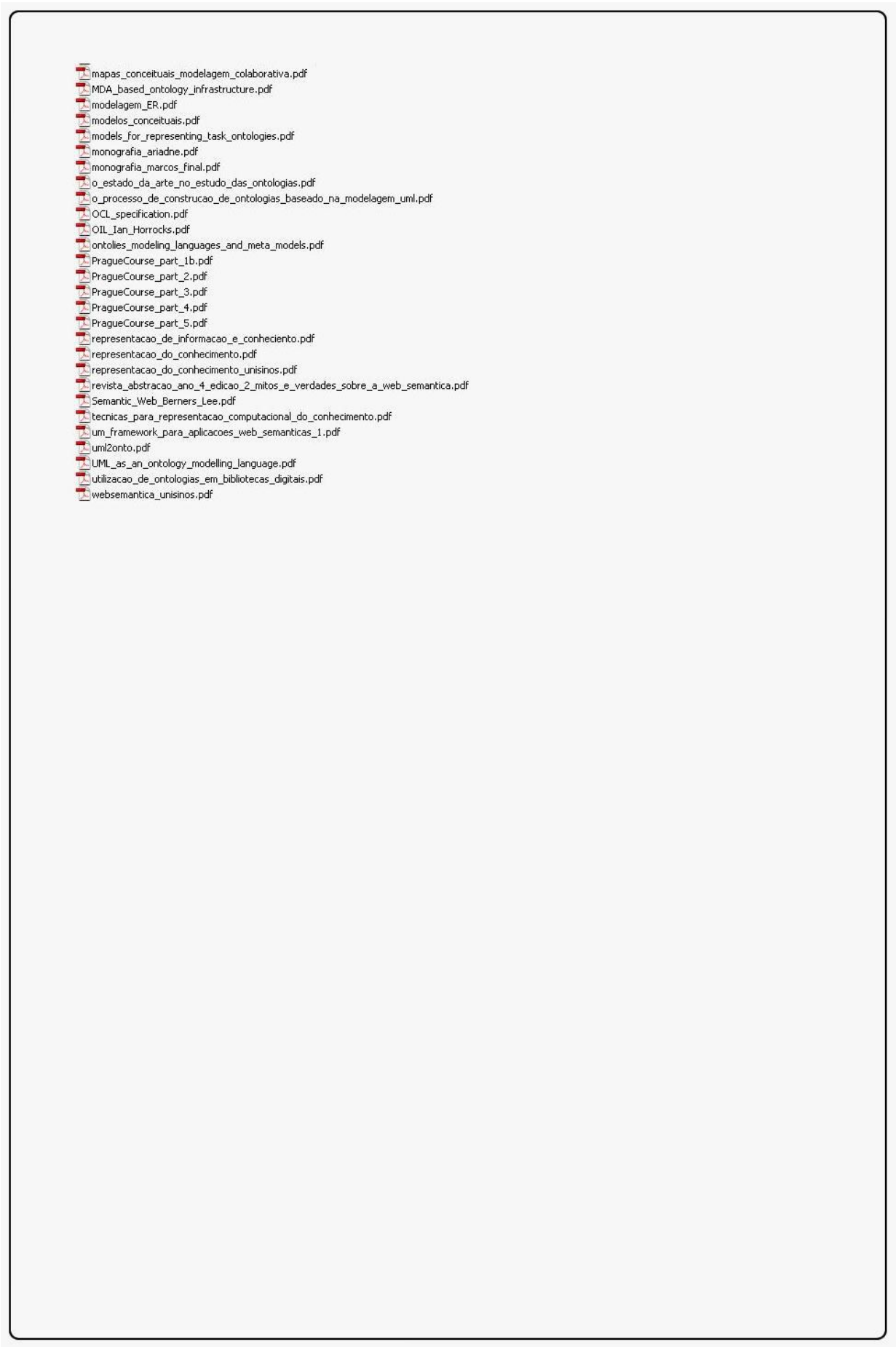


ANEXO B – Hierarquia de Diretórios da Monografia




elementos_DER.png
estrutura_do_frame.png
evolucao_cap2.jpg
figura_de_cunha.jpg
frames.jpg
hCard_e_XFN.jpg
hierarquia_dos_arquivos_da_monografia_2.jpg
hierarquia_dos_arquivos_da_monografia_2_1.jpg
hierarquia_dos_arquivos_inicias_da_monografia.jpg
hierarquia_dos_arquivos_web_site.jpg
IEEE_UFJF.jpg
IEEE_UFJF_protege_1.jpg
IEEE_UFJF_protege_2.jpg
linha_de_comando.png
mapa_conceitual.jpg
mapa_mental_ambiente_de_esenvolvimento.jpg
mapa_mental_da_monografia.jpg
mapa_mental_inteligencias_multiplas.gif
mapa_mental_mapeamento_de_modelos.jpg
mapa_mental_representacao_do_conhecimento.jpg
mapa_mental_representacao_semantica.jpg
mapa_mental_resultados_finais.jpg
modelo_uml.png
netbeans_projeto_IEEE_UFJF.jpg
niveis_ontologicos.jpg
ontologia_IEEE_UJF.JPG
ontologia_IEEE_UJF_1_0.jpg
ontologia_IEEE_UJF_1_0_1.JPG
ontologia_IEEE_UJF_1_0_2.JPG
pesquisa_google.jpg
pesquisa_ieee_ufjf_membros.jpg
pesquisa_live.jpg
protege.jpg
rede_semantica.png
representacao_semantica.jpg
script.jpg
termo_em_tesauros.jpg
UML.jpg
UML2.jpg
UML2ONTO.png
UMLtoOWL.jpg
XML_icon.JPG
textos
estruturais
dicas_monografias_word.pdf
manual_de_monografia_procedimentos_de_formatacao_word.pdf
monografia_luciene.doc
meus
links_relacionados.txt
monografia2.doc
monografia3.doc
monografia4.doc
monografia4_1.doc
monografia4_2.doc
monografia4_3.doc
monografia.doc
parte_da_monografia3.doc
renomeacao_de_arquivos.txt
resumo_a_framework_for_web_science.doc
resumo_do_edital_do_mct.doc
resumo_ontology_based_data_access.doc
topicos_relacionados
an_agile_approach_for_web_systems_engineering.pdf
aquisicao_de_conhecimento_automatizada_para_sistemas_especialistas_probabilisticos.pdf
automatic_text_summarization_based_on_the_global_document_annotation.pdf
cognitive_psychology_and_its_implications.pdf
computer_science_2.0.pdf
construindo_mapas_conceituais.pdf
diagrama_entidade_relacionamento.pdf
engenharia_de_ontologias_em_ECG.pdf
experiences_in_using_a_method_for_building_domain_ontologies.pdf
formal_ontology_and_information_system.pdf
formalizacao_de_ontologias_e_projeto_de_dominio.pdf
gestao_de_metadados.pdf
integrating_knowledge_on_the_web.pdf
JGraphPad_para_desenhar_redes_semanticas_e_gerar_base_de_conhecimento_em_Prolog.pdf
Jose_Maria_Parente.pdf
mapas_conceituais.pdf
mapas_conceituais_e_aprendizagem_siginificativa.pdf

C


mapas_conceituais_modelagem_colaborativa.pdf
MDA_based_ontology_infrastructure.pdf
modelagem_ER.pdf
modelos_conceituais.pdf
models_for_representing_task_ontologies.pdf
monografia_ariadne.pdf
monografia_marcos_final.pdf
o_estado_da_arte_no_estudo_das_ontologias.pdf
o_processo_de_construcao_de_ontologias_baseado_na_modelagem_uml.pdf
OCL_specification.pdf
OIL_Ian_Horrocks.pdf
ontolies_modeling_languages_and_meta_models.pdf
PragueCourse_part_1b.pdf
PragueCourse_part_2.pdf
PragueCourse_part_3.pdf
PragueCourse_part_4.pdf
PragueCourse_part_5.pdf
representacao_de_informacao_e_conheciento.pdf
representacao_do_conhecimento.pdf
representacao_do_conhecimento_unisinos.pdf
revista_abstracao_ano_4_edicao_2_mitos_e_verdades_sobre_a_web_semantica.pdf
Semantic_Web_Berners_Lee.pdf
tecnicas_para_representacao_computacional_do_conhecimento.pdf
um_framework_para_aplicacoes_web_semanticas_1.pdf
uml2onto.pdf
UML_as_an_ontology_modelling_language.pdf
utilizacao_de_ontologias_em_bibliotecas_digitais.pdf
websemantica_unisinos.pdf

D

ANEXO C – Estereótipos OUP

RDFS Concept	Ontology Definition Metamodel Concept	Base UML Class	UML Stereotype (inside « ») or Tag
rdfs:Resource	abstract class Resource		
rdfs:Datatype	class Datatype	DataType	
rdfs:range	association range	Association or Attribute	«range»
rdfs:domain	association domain	Association or Attribute	«domain»
rdfs:type	association type	Dependency	«instanceOf»
rdfs:subClassOf	association subclassOf	Generalization	«subClassOf»
rdfs:subPropertyOf	association subPropertyOf	Generalization	«subPropertyOf»
rdfs:label	attribute label		
rdfs:seeAlso	association seeAlso	Association	«seeAlso»
rdf:Property	abstract class Property		
rdf:Statement	class Statement	Object	«ObjectProperty» or «DatatypeProperty»
rdf:subject	association subject	Link or AttributeLink	«subject»
rdf:object	association object	Link or AttributeLink	«object»
rdf:predicate	association predicate	Dependency	«instanceOf»
rdf:ID	attribute ID	Element Name	

Adaptado de DJURIC, GASEVIC e DEVEDZIC (2006)

OWL Ontology Concept	Ontology Definition Metamodel Concept	Base UML Class	UML Stereotype (inside «») or Tag
owl:Ontology	class Ontology	Package	«ontology»
owl:Class	class Class	Class	«OntClass»
Enumeration	class Enumeration	Class	«Enumeration» or enumeration
owl:Restriction	abstract class Restriction		
owl:onProperty	association onProperty	Association	«onProperty»
ValueConstraint	abstract class ValueConstraint		
owl:allValuesFrom	association allValuesFrom and class AllValuesFrom	Association and Class	«allValuesFrom» (Assoc.) and «AllValuesFrom»
owl:someValuesFrom	association someValuesFrom and class SomeValuesFrom	Association and Class	«someValuesFrom» (Assoc.) andv «SomeValuesFrom»
owl:hasValue	association hasValue and class HasValue	Dependency and Class	«hasValue» (Assoc.) and «HasValue»
CardinalityConstraint	abstract class CardinalityConstraint		
owl:minCardinality	class MinCardinality	AssociationEnd multiplicity	
owl:maxCardinality	class MaxCardinality	AssociationEnd multiplicity	
owl:cardinality	class Cardinality	AssociationEnd multiplicity	
owl:intersectionOf	association intersectionOf and class Intersection	Dependency and TaggedValue	«intersectionOf» (Dep.), intersection tag or «Intersection» for Class
owl:unionOf	association unionOf and class Union	Dependency and TaggedValue	«unionOf» (Dep.), union tag or «Union» for Class
owl:complementOf	association complementOf andClass ComplementOf	Dependency and TaggedValue	«complementOf» (for Dependency), complement tag or «Complement» for Class
owl:equivalentClass	association equivalentClass	Dependency	«equivalentClass»
owl:disjointWith	association disjointWith	Dependency	«disjointWith»
owl:ObjectProperty	class Objectproperty	Class	«ObjectProperty»
owl:DatatypeProperty	class DatatypeProperty	Class	«DatatypeProperty»
owl:equivalentProperty	association equivalentProperty	Dependency	«equivalentProperty»
owl:inverseOf	association inverseOf	Dependency	«inverseOf»
owl:FunctionalProperty	class FunctionalProperty	TaggedValue	functional
owl:InverseFunctionaProperty	class InverseFunctionalProperty	TaggedValue	inverseFunctional

Adaptado de DJURIC, GASEVIC e DEVEDZIC (2006)

F

OWL Ontology Concept	Ontology Definition Metamodel Concept	Base UML Class	UML Stereotype (inside «») or Tag
owl:TransitiveProperty	class TransitiveProperty	TaggedValue	transitive
owl:SymmetricProperty	class SymmetricProperty	TaggedValue	symmetric
Individual	class Individual	Object	«ontClass»
owl:Thing	instance of class Individual		
owl:sameAs and owl:sameIndividualAs	association sameAs	Dependency	«sameAs»
owl:differentFrom	association differentFrom	Dependency	«differentFrom»
owl:allDifferent	association allDifferent	Dependency	«allDifferent»
owl:oneOf	association type	Dependency	«instanceOf»
owl:AllDiferent	class AllDifferent	Class	«AllDifferent»
owl:distinctMembers	association distinctMembers	Dependency	«distinctMembers»
owl:equivalentProperty	association equivalentProperty	Dependency	«equivalentProperty»
owl:backwardCompaibleWith	owl.backwardCompatibleWith	Dependency	«backwardCompatible With»
owl:imports	owl.imports	Dependency	«imports»
owl:incompatibleWith	owl.incompatibleWith	Dependency	«incompatibleWith»
owl:inverseOf	owl.inverseOf	Dependency	«inverseOf»
owl:priorVersion	owl.priorVersion	Dependency	«priorVersion»

Adaptado de DJURIC, GASEVIC e DEVEDZIC (2006)