NEGOCIAÇÃO DE SIGNIFICADO PARA VIABILIZAR INTEROPERABILIDADE SEMÂNTICA

Jairo Francisco de Souza

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Jano Moreira de Souza, Ph.D.

Prof. Geraldo Bonorino Xéxeo, D.Sc.

Profa. Virgínia Verônica Bezerra Brilhante, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2007

SOUZA, JAIRO FRANCISCO DE

Negociação de significado para viabilizar interoperabilidade semântica [Rio de Janeiro] 2007

XIII, 171 p., 29,7 cm (COPPE/UFRJ, M.Sc, Engenharia de Sistemas e Computação, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Interoperabilidade semântica
2. Negociação do significado
3. Gestão do conhecimento

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Ao meu orientador, professor Jano Moreira de Souza, por estimular o meu trabalho e por ser exigente e crítico para que eu sempre obtivesse o melhor resultado.

Deixo aqui registrado minha gratidão à minha co-orientadora Jonice de Oliveira Sampaio. Obrigado, antes de tudo, pela amizade durante esses anos de mestrado! Em segundo lugar, seu incentivo, sua paciência e sua disponibilidade foram muito valiosos para que eu pudesse terminar esse trabalho da melhor maneira possível.

Obrigado também para a doutora e amiga Melise de Paula pelas importantes contribuições feitas neste trabalho, as quais foram decisivas para definir o modelo de negociação e o direcionamento da pesquisa.

Agradeço ao professor Geraldo Xéxeo e à professora Virgínia Brilhante por aceitarem fazer parte da banca, mesmo diante de tantos compromissos e atividades. Ao professor Blaschek por acreditar no meu trabalho e por me orientar na vida profissional.

Agradeço à Pricila por agüentar minhas reclamações e por estar sempre presente, mesmo a horas de distância.

Finalmente, agradeço à maior conquista durante esse longo tempo de mestrado: os amigos que conheci e que com certeza continuarão presentes na minha vida pós-mestrado. Obrigado pelas alegrias, pelos convites, pelo incentivo, pelas idéias e pelos conselhos (em especial a turma da República Pão de Queijo e a turma de Caxias, grandes amigos que fiz).

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

NEGOCIAÇÃO DE SIGNIFICADO PARA VIABILIZAR INTEROPERABILIDADE SEMÂNTICA

Jairo Francisco de Souza

Março / 2007

Orientador: Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Ontologias são utilizadas, entre outros casos, para aumentar a interoperabilidade semântica entre diferentes sistemas e na comunicação entre agentes. Contudo, a interoperabilidade semântica é comprometida num cenário onde mais de uma ontologia é utilizada, como projetos com comunidades multidisciplinares, uma vez que não há meios simples de compatibilizar os conceitos entre essas diferentes estruturas de conhecimento. Para apoiar esse cenário, propomos um modelo de negociação para chegada de consenso sobre interpretações. Nosso modelo visa o mapeamento de ontologias através do uso de técnicas de negociação usadas em ambientes de negócios, levando em consideração a resolução de conflitos gerados por conseqüência das diferentes interpretações. Neste trabalho é utilizado o método de revisão bibliográfica nos conceitos de ontologia, técnicas de integração, negociação e resolução de conflito e um comparativo entre os projetos atuais na área de interoperabilidade semântica, criando a partir desses elementos o referencial teórico, ou concepção teórica. Para o seu desenvolvimento foi criado e implementado o ambiente GNoSIS. A proposta foi avaliada a partir de estudos de observação, as principais conclusões são comparativas, tendo como referencial os trabalhos correlatos e as possíveis melhorias obtidas com o uso deste trabalho no contexto de interoperabilidade semântica.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MEANING NEGOTIATION FOR SEMANTIC INTEROPERABILITY ACHIEVEMENT

Jairo Francisco de Souza

March / 2007

Advisor: Jano Moreira de Souza

Department: Computer and Systems Engineering

Ontologies are used to improve the semantic between different systems and communication among agents. However, the semantic interoperability is harmed in a scenario where more than one ontology is used, as projects with multidisciplinary communities, since there are no simple ways to suite concepts between these different structures of knowledge. To support this scenario, we consider a model of negotiation for reaching consensus on interpretations. Our model aims at the mapping of ontologies through the use of common business negotiation techniques, considering conflicts resulting from these different interpretations. This work had used the theoretical revision about concepts of ontology, integration techniques, negotiation and conflict resolution, and a survey about semantic interoperability research area, creating our theoretical foundation. For its development was created and implemented GNoSIS environment. This work was evaluated using observation studies. Main conclusions are comparatives, analyzing similar approaches and possible improvements obtained using this work in semantic interoperability context.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas

API – *Application Programming Interface*

BATNA – *Best Alternative To a Negotiated Agreement*

DAML+OIL – *Darpa Agent Markup Language + Ontology Inference Layer*

GAV – *Global As View*

GC – *Gestão do Conhecimento*

GQM – *Goal-Question-Metric*

GVV – *Global Virtual View*

HTML – *HyperText Markup Language*

IBIS – *Issue Based Information Systems*

IEEE – *Institute of Electrical and Electronics Engineers*

KIF – *Knowledge Interchange Format*

KM – *Knowledge Management*

KUP – *Knowledge Unified Process*

MOMIS – *Mediator EnvirOnment for Multiple Information Sources*

OBSERVER – *Ontology Based System Enhanced with Relationships for Vocabulary heterogeneity Resolution*

ODE – *Ontology Design Environment*

OTK – *On-To-Knowledge Methodology*

OWL – *Web Ontology Language*

OWL-S – *Ontology Web Language for Services*

P2P – *Peer-to-Peer Computer Network*

RDF – *Resource Description Framework*

RDFS – *Resource Description Framework Schema*

RUP – *Rational Unified Process*

SIMS – *Services and Information Management for decision Systems*

SSN – Sistema de Suporte à Negociação

SUMO – *Suggested Upper Merged Ontology*

TOVE – *Toronto Virtual Enterprise*

URI - *Uniform Resource Identifier*

URL - *Uniform Resource Locator*

XML - *Extensible Markup Language*

W3C – *World Wide Web Consortium*

Web – *World Wide Web*

WS – *Web Semântica*

WWW – *World Wide Web*

Capítulo 1 – Introdução

A *World Wide Web* (WWW ou simplesmente Web) surgiu para garantir, de maneira prática, o suprimento do recurso mais importante da atualidade: a informação. A Web é atualmente o maior repositório de informações existente, com um número de usuários cada vez maior e em franco processo de evolução, podendo vir a se tornar uma enorme base de conhecimento.

Porém, embora a Web seja uma poderosa fonte provedora de informação, a falta de confiabilidade e a pouca estruturação faz com que o montante de informação disponibilizada possua baixa qualidade e, assim, baixo valor informativo. Por esses motivos, os atuais mecanismos de busca existentes na WWW não são capazes de suprir eficientemente as necessidades dos usuários em geral, gerando muito retorno indesejado (conhecido como sobrecarga de informação) e causando insatisfação. Sempre nos deparamos com domínios complexos demais ou pouco conhecidos. Devido ao excesso de informação disponibilizada, oriunda de diversas fontes, é possível absorver e empregar conceitos - oriundos destes domínios - de forma equivocada, pois, em muitos casos, depara-se com a falta de clareza e ambigüidade com que eles são tratados. É nesse contexto que identificamos a necessidade da construção de ontologias, formalizando conceitos e relações pertencentes a esse domínio.

Contudo, a simples construção de uma ontologia não resolve todos os problemas no que se refere a um maior enriquecimento semântico. Mesmo ontologias criadas através de metodologias e técnicas que garantam sua conformidade com os conceitos de compartilhamento e reuso que as regem, não são capazes de garantir uma visão consensual e nem de abranger todos os conceitos necessários por quaisquer aplicações. Tal problema emerge, principalmente, em ambientes multidisciplinares, onde os membros da equipe lidam com domínios diferentes da sua área de conhecimento. Dessa forma, torna-se necessário a criação de novas ontologias para (1) expressar o mesmo domínio com uma diferente visão de mundo, conciliando as diferentes visões e entendimentos sobre um domínio ou (2) descrever um domínio específico.

Neste cenário de múltiplas ontologias, persiste o problema de como compatibilizar as diferentes visões de mundo. Dessa forma, tornam-se necessários mecanismos que permitam a interoperabilidade semântica. Entende-se por

interoperabilidade semântica a capacidade de dois ou mais sistemas heterogêneos e distribuídos trabalharem em conjunto, compartilhando as informações entre eles com entendimento comum de seu significado (BURANARACH, 2001).

Um dos ambientes ricos em ontologias e que, por isso, necessita de mecanismos que permitam a interoperabilidade semântica é o denominado de Web Semântica. Esse ambiente será descrito na seção abaixo.

1.1 – A Web Semântica

Segundo (BERNERS-LEE, HENDLER, LASSILA, 2001), a Web Semântica (WS) é uma extensão da Web atual, porém apresenta uma estrutura que possibilita a compreensão e o gerenciamento dos conteúdos armazenados na Web independente da forma em que estes se apresentem, seja texto, som, imagem ou vídeo. Isto se dará a partir da valoração semântica desses conteúdos e através de agentes, programas coletores de conteúdo advindos de fontes diversas capazes de processar as informações e permutar resultados com outros programas. Toda a informação é dada de uma forma bem-definida, possibilitando que computadores e pessoas possam trabalhar em cooperação de forma mais eficaz.

A WS requer capacidade para representar e gerenciar conteúdo semântico na Web a fim de que um agente possa "aprender" o significado de um novo termo a partir de uma especificação formal. A partir daí, surge a necessidade de formalização, que ocorre com o uso de ontologias e metadados[1]. Requer também integração e interoperabilidade, mostrando que dois agentes estão semanticamente integrados se puderem comunicar-se entre si com sucesso, levando em consideração a distinção das linguagens de representação, a incompatibilidade de conceitos e os diferentes termos e estilos de modelagem.

Basicamente, a WS se compõe de três elementos: representação do conhecimento, ontologias e agentes (BREWSTER *et al.*, 2004). A representação do conhecimento faz com que a WS estruture o conteúdo significativo de páginas Web criando um ambiente onde os agentes podem, através de buscas de uma página a outra, promover certa integração que levará à execução de tarefas mais sofisticadas para os

[1] Etimologicamente, metadado significa "dado sobre dado"; dado que descreve a essência, os atributos e o contexto de emergência de um recurso e caracteriza suas relações, visando o seu acesso e o seu uso potencial (FERREIRA, 1986).

usuários. As ontologias são uma maneira de se definir um conteúdo específico sobre o conhecimento a ser compartilhado e reusado entre diferentes agentes. Os agentes têm a função de coletar conteúdos na Web a partir de diversas fontes, processar a informação e permutar os resultados com outros programas, permitindo, através de uma linguagem que expresse inferências lógicas resultantes do uso de regras, a troca de informação como as especificadas pelas ontologias (OLIVEIRA, 2002).

Tim Berners-Lee (1998) sugeriu uma estrutura de camadas para a Web Semântica, como é ilustrado na Figura 1.

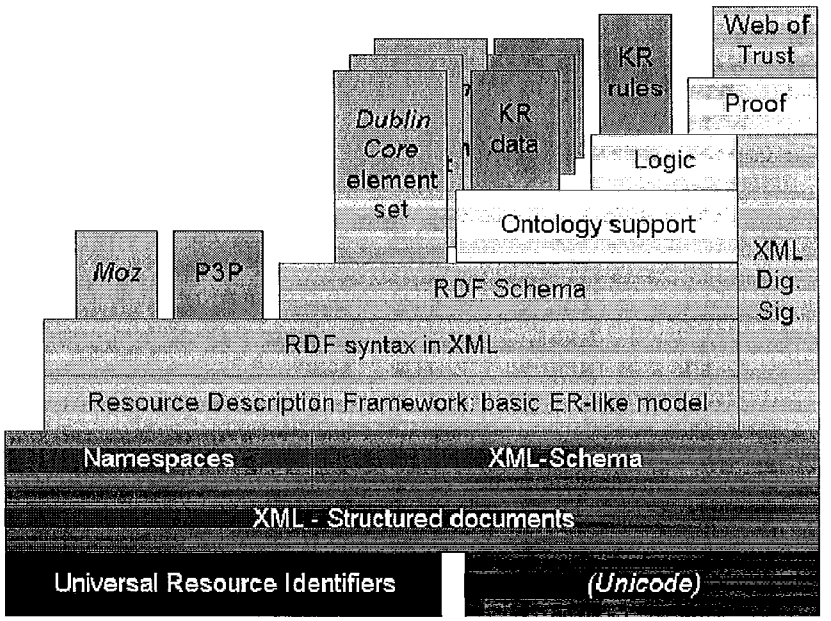


Figura 1 – Arquitetura definida para a Web Semântica em (BERNERS-LEE, 2000a)

Nesta arquitetura, cada camada estende a funcionalidade e expressividade de suas camadas inferiores. Nas primeiras duas camadas é fornecida uma sintaxe comum. O *Uniform Resource Identifier* (URI) provê um modo padrão de referir as entidades[2], enquanto o Unicode é um padrão para troca de símbolos. As camadas referentes à linguagem XML *(Extensible Markup Language)* (XML, 2006) e RDF *(Resource Description Framework)* (MILLER, MANOLA, MCBRIDE, 2006) desempenham papéis fundamentais (DECKER *et al.*, 2000). A camada XML permite que os autores de

[2] URL (Uniform Resource Locator) refere a uma localidade de URI, por exemplo, um endereço http:// ou ftp://. URL e URI são quase sempre usados como sinônimos, porém URLs são subclasses de URIs. Para mais detalhes, consulte http://www.w3.org/Addressing (último acesso em 21/01/2005).

documentos descrevam dados de forma estruturada e que separem esta descrição da interface com o usuário (a apresentação). A camada RDF é responsável pelo modelo dos dados. É nessa camada que realmente começa a Web Semântica, já que nesta camada já existe a declaração de semântica no conteúdo dos documentos. De acordo com as recomendações da W3C (RDF, 1999), a RDF "é uma base para processamento de metadados[3]; ela provê interoperabilidade entre aplicações que trocam informação inteligível por máquinas na Web". No entanto, essas camadas são elementares e, por isso, são necessárias outras camadas de suporte às ontologias e à lógica para fornecer a semântica esperada para a evolução da Web.

Nas próximas camadas estão os vocabulários de ontologia e lógica. Hoje a comunidade da Web semântica considera estas camadas como uma única já que ontologias permitem usar axiomas lógicos. A camada lógica é necessária para prover o formalismo passível de processamento automático (FENSEL *et al.*, 2001) como, por exemplo, permitir serviços de raciocínio por máquinas. Por fim, a última camada trata-se da "Web da Confiança", onde as assinaturas digitais funcionam como um mecanismo de prevenção de inconsistências nessa Web Semântica (BERNERS-LEE, 2000b).

Como pode ser notado na Figura 1, quanto mais superior for uma camada da WS, ela se torna cada vez mais centrada em aspectos sociais. Em outras palavras, mesmo imaginando que todas essas camadas possam ser resolvidas por soluções técnicas, elas vão se relacionando com mais aspectos sociais. Pode-se tomar, por exemplo, a camada de confiança, a qual pode ser resolvida tecnologicamente, como é visto na comunidade de agentes (MULLER, VERCOUTER, BOISSIER, 2005), mas é um fenômeno social por excelência. Não importa quão boa seja a solução tecnológica provida, quão superior for a camada da WS, essas soluções precisam respeitar mais fenômenos sociais, e fenômenos cada vez mais complexos (O'HARA, 2004).

O acordo entre partes é um fenômeno social presente nas camadas da WS, principalmente na camada que diz respeito à estruturação do conhecimento. Esse conhecimento é compartilhado com outros usuários, os quais decidiram que ele deveria ser estruturado de tal forma. Para que esse acordo seja firmado, propomos, nesse trabalho, um método de negociação de significado baseado em técnicas de negociação usadas em negócios.

---

[3] Metadados são utilizados para descrever as características de recursos e seus relacionamentos. São comumente definidos como "dados que descrevem dados".

1.2 – Organização do trabalho

Os estudos e análises realizados no desenvolvimento desta dissertação foram organizados em quatro capítulos adicionais, como se segue.

O capítulo 2 apresenta uma caracterização de ontologias, tais como a definição utilizada neste trabalho, exemplos de sua aplicabilidade, metodologias existentes para sua construção e descrevem, mesmo que subjetivamente, um processo de compatibilização de conceitos. Nos aprofundamos na interoperabilidade entre ontologias descrevendo as divergências que emergem ao utilizarmos mais de uma ontologia, os mecanismos de aproximação usados para compatibilizar ontologias e as fórmulas mais usadas para cálculo de similaridade entre conceitos.

O capítulo 3 descreve nosso processo de negociação de significados para auxiliar a chegada de consenso durante o processo de compatibilização de termos ontológicos visando a interoperabilidade semântica entre ontologias com domínios sobrepostos. Esse capítulo também descreve a arquitetura criada para exemplificar e validar a viabilidade do processo proposto. Por fim, é feito uma comparação entre o processo proposto e outras abordagens encontradas.

O capítulo 4 mostra os benefícios da utilização deste processo e os procedimentos definidos para sua avaliação. Os resultados conseguidos com a utilização do processo também são apresentados neste capítulo.

No capítulo 5 são reportadas as conclusões gerais e as contribuições desta tese, bem como algumas indicações para trabalhos futuros.

Capítulo 2 – Integração de Ontologias

Neste capítulo discutiremos o conceito de ontologia, sua aplicabilidade e metodologias que auxiliam no processo de estruturação do conhecimento, destacando, nessas metodologias, a existência de uma etapa de reutilização de conhecimento já estruturado, criando a necessidade de compatibilização de conceitos. Relacionaremos os principais problemas encontrados no processo de compatibilidade de ontologias, os mecanismos de aproximação usados para esse o processo de compatibilidade e as fórmulas mais usadas para cálculo de similaridade de conceitos.

2.1 – Ontologia

A definição de ontologia que será usada neste trabalho é apresentada na seção 2.1.1 e suas diversas aplicações são apresentadas na seção 2.1.2.

2.1.1 – Definição

Na filosofia, ontologias são o estudo da existência do ser (CHANDRASEKARAN, JOSEPHSON, BENJAMINS, 1999). A ontologia é entendida como um sistema de categorias de uma determinada visão do mundo (GUARINO, 1998). Assim sendo, esta abordagem não depende de uma linguagem particular. Por outro lado, em seu uso na Inteligência Artificial, a ontologia faz referência a um artefato de engenharia, constituído de vocabulário específico usado para descrever certa realidade, acrescida de um conjunto de suposições com respeito ao significado esperado das palavras do vocabulário. Este conjunto de suposições é normalmente representado na forma de lógica de primeira ordem, onde as expressões do vocabulário aparecem como predicados unários ou binários, chamados respectivamente de conceitos e relações (SOWA, 1999). No caso mais simples, uma ontologia descreve uma hierarquia de conceitos relacionados por meio de regras, enquanto que, em casos mais sofisticados, são adicionados axiomas adequados para expressar relacionamentos mais complexos e para restringir a interpretação desejada de seus conceitos (SOWA, 1999).

Dessa forma, não é o vocabulário que é qualificado como sendo uma ontologia e sim as conceitualizações que os termos no vocabulário pretendem capturar. Assim

sendo, ao traduzir os termos de uma ontologia de uma língua para outra, por exemplo, do Português para o Inglês, conceitualmente não haverá mudanças na ontologia.

Na Ciência da Computação, a definição mais comum de ontologia é a proposta por Tom Gruber (1993) e posteriormente refinada por outros pesquisadores (GUARINO, 1996; SOWA, 1999), onde uma ontologia é definida como uma *"especificação explícita e formal de uma conceitualização compartilhada"*. A palavra conceitualização refere-se a uma abstração, visão simplificada do mundo que desejamos representar para algum propósito, construído através da identificação dos conceitos e relações relevantes. O termo explícita indica que os tipos de conceitos e as restrições ao seu uso são explicitamente definidos através de uma estrutura declarativa. Como formal entende-se que a ontologia deve ser compreensível por um computador, não podendo ser somente escrita em linguagem natural, sendo necessário que ela seja representável matematicamente, de forma não-ambígua, ou seja, que ela seja representada através de uma linguagem ontológica. Finalmente, compartilhada implica em que o conhecimento representado é consensual, aceito por um grupo como um todo e não apenas por um único indivíduo ou, pelo menos, que reúna definições que serão compartilhadas com mesma semântica por agentes inteligentes durante sua comunicando.

A estrutura para descrição de ontologias difere entre algumas áreas e abordagens. Contudo, a estrutura proposta em (MAEDCHE, 2002) é atualmente a mais conhecida por estar em conformidade com a linguagem padrão para ontologias, a linguagem OWL – *OWL Web Ontology Language* (DEAN, 2006).

Esta estrutura é representada pela tupla $O := \{C, R, H_C, rel, A_O\}$, onde:

C é um conjunto de conceitos do domínio;

R é um conjunto não-taxonômico de relações descrito por suas restrições de domínio;

C (conceitos) e R (relações) são dois conjuntos disjuntos;

H_C é uma relação direcionada $H_C \subseteq C \times C$ que é chamada hierarquia de conceitos ou taxonomia, podendo existir heranças múltiplas (heterarquia). Por exemplo, $H_C (C_1, C_2)$ significa que C_1 é um subconjunto de C_2;

rel é uma função $rel : R \rightarrow C \times C$ que relaciona não taxonomicamente conceitos;

A_O é um conjunto de axiomas expresso em linguagem lógica apropriada.

Na estrutura acima, conceitos de um domínio com relacionamentos de especialização, i.e., relacionamentos do tipo "é-um" (*is-a*), são descritos em ontologias utilizando uma organização taxonômica. Aspectos de composição, i.e., relacionamentos do tipo "parte-de/todo" (*part-of/whole*), são ortogonais às ontologias e devem ser representados por meio de funções não taxonômicas, por exemplo, propriedades.

2.1.2 – Aplicabilidade

Muitas áreas de aplicações baseadas em computadores - como Gestão do Conhecimento, Comércio Eletrônico, Sistemas Geográficos, *E-Government*, Educação, entre outras - estão usando as vantagens das ontologias. Abaixo, apresentamos algumas soluções baseadas em ontologias que foram sugeridas por pesquisadores e alguns sistemas baseados nessa tecnologia.

Sistemas de processamento de textos que usam ontologias são especialmente úteis para processar dados relacionados a um domínio específico na Web, uma vez que os mecanismos de pesquisa baseados em palavras-chave, mesmo que robustos, provaram ser uma ferramenta imprecisa na busca de informações (FREITAS, STUCKENSCHIMDT, NOY, 2005). Estes mecanismos geralmente retornam um grande montante de documentos irrelevantes para a consulta feita, levando o usuário à insatisfação. O principal problema dessa abordagem é não levar em consideração o contexto da informação descrita nas páginas, uma vez que não utilizam trechos úteis de informação que poderiam ser processadas se os sistemas pudessem contar com alguma forma de conhecimento prévio sobre os assuntos principais referentes às páginas.

Basicamente três abordagens são propostas para resolver esse problema, ao fazerem uso de ontologias. A primeira delas é a expansão da consulta através de uma ontologia, onde a consulta do usuário é enriquecida por sinônimos e hiperônimos (superclasses) relacionados ao conceito a ser pesquisado após ser enviada para a máquina de processamento de consultas. Estes conceitos são organizados numa ontologia de domínio (ROBIN, RAMALHO, 2001; PAZ-TRILLO, WASSERMANN, BRAGA, 2005). A outra abordagem diz respeito aos sistemas semânticos de recuperação de informação, nos quais um corpo de texto é previamente anotado de acordo com uma ontologia de domínio durante o processo de indexação (REZENDE *et al.*, 2006). A etapa de recuperação envolve um procedimento de inferência para conferir as respostas ou enriquecê-las com informação implícita (NOY, 1997). Por último, temos os sistemas baseados em ontologias que reúnem informação, os quais usualmente

realizam mais de uma tarefa relacionada ao processamento de textos, como recuperação, classificação e extração (CRAVEN *et al.*, 1999; FREITAS, BITTENCOURT, 2003). Este tipo de sistema também pode ser usado para realizar geração semi-automática de anotação em páginas da Web (KUPER *et al.*, 2003), definindo-se padrões sintáticas e gramaticais para identificar certos tipos de objetos no texto (por exemplo, lugares, pessoas, animais, etc). Esses padrões são relacionados com conceitos da ontologia e as instâncias desses conceitos (classe) podem ser identificadas nos documentos e anotadas com o seu tipo (classe a que pertence).

O problema de integração de dados tem recebido grande atenção da comunidade de banco de dados, tendo sido estudado desde o início da década de 80 (ZIEGLER, DITTRICH, 2004). São encontrados, na literatura, diversos sistemas e propostas tentando solucionar tal problema. O aparente vasto número de soluções pode, erroneamente, indicar uma área de pesquisa já resolvida. Pelo contrário, é uma área cada vez mais importante e onde não existe uma solução geral que seja adequada ou que se ajuste aos diversos problemas de integração, o que se constata pelo surgimento de novas propostas. Dentre elas, diversos trabalhos utilizam ontologias para a definição de metadados, buscando uma maior semântica para os mesmos, como mostrado nos sistemas descritos nos próximos parágrafos.

O ONTOBROKER (DECKER *et al.*, 1999) é um sistema para integração de páginas Web cujo principal objetivo é extrair, inferir e gerar metadados específicos de domínio para integrar páginas Web, usando uma ontologia de domínio que reflete o consenso de um grupo de usuários Web. O ONTOBROKER utiliza RDF/RDFS para representar seus metadados, anotados nos recursos Web, e *Frame-Logic* (F-Logic) (KIFER, LAUSEN, WU, 1995) é utilizada como base da máquina de inferência, ou seja, F-Logic é usada quando o sistema está respondendo consultas baseadas em uma ontologia.

O sistema OBSERVER (*Ontology Based System Enhanced with Relationships for Vocabulary heterogeneity Resolution*) (MENA *et al.*, 1996) tem como objetivo principal o processamento de consultas em sistemas de informação globais, baseado em interoperabilidade entre ontologias pré-existentes. Ele utiliza metadados para capturar o conteúdo das informações dos repositórios. O usuário efetua consultas sobre o sistema expressando suas necessidades de informação utilizando descrições de metadados representados utilizando *Description Logics* (BRACHMAN, SCHMOLZE, 1985).

O SIMS (*Services and Information Management for Decision Systems*) (ARENS, KNOBLOCK, 1992; ARENS *et al.*, 1993; ARENS, KNOBLOCK, SHEN, 1996) é um sistema para integração de informações cuja abordagem explora um modelo semântico do domínio do problema para integrar informações de várias fontes de informação, tais como bancos de dados, bases de conhecimento, programas etc. Para proceder à integração, o SIMS necessita de uma ontologia que descreva o domínio no qual as informações a serem tratadas se encontram, assim como a estrutura e os conteúdos das fontes de informação.

O MOMIS (*Mediator EnvirOnment for Multiple Information Sources*) é um framework para extração e integração de informações de fontes de informação heterogêneas que implementa uma metodologia semi-automática para integração de dados que segue a abordagem *Global As View* (GAV) (BENEVENTANO, BERGAMASCHI, 2004). O resultado do processo de integração é um esquema global, chamado *Global Virtual View* (GVV), o qual representa a informação contida nas fontes, sendo uma ontologia que representa o domínio das fontes integradas.

O YACOB (KARNSTEDT *et al.*, 2003) é um mediador baseado em ontologias que usa conhecimento de domínio modelado, representado em termos de conceitos, propriedades e relacionamentos, para integrar dados heterogêneos da Web. O YACOB foi originalmente desenvolvido para permitir acesso integrado a bancos de dados, disponíveis na Web, de artefatos culturais perdidos ou roubados durante a segunda guerra mundial. Suas ontologias são definidas em RDFS.

O trabalho de Suwanmanee (SUWANMANEE, BENSLIMANE, THIRAN, 2005) tem o objetivo de integrar fontes de dados heterogêneas no contexto da Web Semântica. Ele utiliza uma abordagem ontológica para integração de dados, utilizando construções nativas da linguagem OWL para representar metadados e mapeamentos.

Além de serem úteis na integração de dados, como mostrado nos exemplos acima, ontologias também consistem numa tecnologia adequada para explicitar as definições precisas, não ambíguas, relações e restrições que irão representar o contexto da rotina diária de empregados, e fazê-los registrar suas experiências, colaborar e aumentar a inteligência coletiva de empresas que se preocupam em gerir seu conhecimento (FREITAS, STUCKENSCHIMDT, NOY, 2005). Além disso, os requisitos dos sistemas de gestão do conhecimento se encaixam bem com os benefícios das ontologias, como, por exemplo, a capacidade de suportar a integração de base de dados já existentes e a habilidade de lidar com informações implícitas. Um exemplo

disso é a descoberta dos concorrentes de um produto da empresa, que é uma dedução feita usando o fato de que os concorrentes são outras empresas que vendem produtos do mesmo tipo daqueles oferecidos pela empresa (VOLZ, ABECKER, 2003).

Ontologias estão sendo usadas também na gestão do conhecimento pessoal tanto para disseminação de conhecimento e aprendizado em ambientes P2P (REZENDE *et al.*, 2005) como em ambientes ubíquos (LEITE *et al.*, 2005). Essas ferramentas propostas são baseadas em teorias construtivistas (SASTRE *et al.*, 1997) e oferecem um complemento para a educação à distância, onde o aluno pode organizar seu conhecimento através de suas experiências ou usar o conhecimento criado por outros. Grandes empresas como Schlumberger, Daimler Chrysler e BTelecom estão investindo pesadamente em gestão do conhecimento e soluções para memória organizacional baseadas em ontologia como diferencial de competências, ou criando seus próprios departamento de GC, provendo fundos para projetos de pesquisa europeus ou em pequenas empresas de desenvolvimento de *software* como Intraspect, Tacit Knowledge and OntoPrise (FREITAS, STUCKENSCHIMDT, NOY, 2005).

Ontologias também são sendo usadas para comunicação entre agentes e composição semântica de serviços (BURSTEIN, 2004; COSTA, PIRES, MATTOSO, 2004; BURSTEIN, MCDERMOTT, 2005). Em particular no ambiente de serviços Web, linguagens de definição e manipulação de ontologias como OWL-S (*Ontology Web Language for Services*) (OWL-S, 2005) proporcionam a integração da semântica aos serviços de forma simples, proporcionando uma maior facilidade na busca, recuperação e reutilização destes componentes, em repositórios espalhados pela Web, para que possa ter uma composição mais inteligente e otimizada. Em (CASARE, SICHMAN, 2005), os autores introduzem uma ontologia para ser usado em sistemas de multi-agentes. A ontologia possui termos sobre os diferentes modelos e suas respectivas terminologias empregadas em cada um desses modelos. Segundo os autores, essa ontologia poderia ter um importante papel de interoperabilidade na interação de agentes, mesmo quando estes agentes usam diferentes modelos de reputação.

Para que ontologias possam ser aplicáveis, um primeiro problema se encontra na forma com que é criada. Para ajudar nesse processo, algumas metodologias tentam especificar etapas para o entendimento e representação de um domínio, as quais serão discutidas na próxima seção.

2.2 – Metodologias para criação de ontologias

Para diminuir os riscos e aumentar o entendimento dos requisitos de um *software*, são necessários alguns passos e metodologias para sua construção (CRISTANI, CUEL, 2005). Uma vez que a construção de ontologias é um processo custoso e, em parte, se assemelha com o levantamento de requisitos de um domínio (FELICÍSSIMO *et al.*, 2003), então também é importante minimizar os riscos de uma construção errônea. Como o tema ontologia ainda é recente, cada grupo de pesquisadores utiliza seu próprio método, processo ou metodologia para a criação e atualizações de ontologias. A meta aqui é apresentar um resumo das diferentes abordagens para o desenvolvimento de ontologias, destacando a existência de um processo de compatibilização (ou integração) de conceitos dentro dessas metodologias.

2.2.1 – Metodologia Uschold e King

Esta metodologia é baseada no contexto do desenvolvimento da *Enterprise Ontology*, uma ontologia para processos de modelagem de empresas (USCHOLD, KING, 1995). Ela fornece quatro principais diretrizes para o desenvolvimento de ontologias (FERNÁNDEZ-LÓPEZ, GÓMEZ-PÉREZ, 2002), que são descritas a seguir:

- *Identificação do propósito e escopo* – onde detalha-se o porquê da construção da ontologia e onde ela será utilizada.
- *Construção da ontologia* - Esta fase é dividida em três passos, que são:
  - Captura da ontologia: onde ocorre a identificação dos conceitos e relacionamentos chaves para o domínio de interesse, focando no sentido concreto destes, para haver produção de definições textuais não ambíguas e a identificação dos termos para referenciar tais conceitos e relacionamentos. Para a identificação dos conceitos chave são propostas três possíveis estratégias: partindo dos conceitos mais concretos para os mais abstratos (bottom-up), dos mais abstratos para os mais concretos (top-down), ou partindo dos conceitos mais relevantes para os mais abstratos e para os mais concretos (middle-out).
  - Codificação: nesta etapa a representação do conhecimento adquirido no passo anterior é explicitamente mostrado em linguagem formal.

- Integração de ontologias existentes: durante os processos de captura e codificação surge a questão de como e se é realmente necessário usar ontologias que já existem.

– *Avaliação* - aqui é necessária a referência à especificação e análise dos requisitos, as questões de competência, e/ou mundo real.

– *Documentação* - recomenda-se que diretrizes sejam estabelecidas para documentar as ontologias, possivelmente diferindo de acordo com o tipo e o propósito da ontologia. Tudo deve ser documentado, desde os conceitos principais até as primitivas usadas para definir a ontologia.

2.2.2 – Metodologia Grüninger & Fox

Esta metodologia é baseada na experiência de desenvolvimento de uma ontologia dentro do domínio de processos de negócios e modelagem de atividades, a ontologia TOVE (*TOronto Virtual Enterprise*) (KIM, FOX, GRUNINGER, 1999; GRUNINGER, ATEFI, FOX, 2000).

Ela se baseia na construção de um modelo lógico do conhecimento que será especificado através de uma ontologia. Este modelo não é construído diretamente, já que é feita uma descrição informal da especificação do que deve constar na ontologia para então ser formalizada. As seguintes etapas são propostas nesta metodologia:

– *Captura dos cenários de motivação* - identificação de problemas ou exemplos que não são adequadamente focados por ontologias existentes e fornecimento de um conjunto de possíveis soluções para os problemas do cenário no qual se está trabalhando.

– *Formulação de questões de competência informais* - essas questões são baseadas nos cenários obtidos na etapa anterior e podem ser consideradas como requisitos de expressividade que estão em forma de questões. A ontologia tem que ser capaz de representar essas questões usando sua terminologia e caracterizar as respostas para estas questões usando axiomas e definições.

– *Especificação da terminologia da ontologia numa linguagem formal* - os termos definidos nesta etapa devem ser suficientes para responder as questões de competência formuladas na etapa anterior. Estes termos servirão como base para especificar a terminologia em uma linguagem formal, tal como lógica de primeira ordem ou na linguagem equivalente KIF

(*Knowledge Interchange Format*) (GINSBERG, 1991; GENESERETH, FIKES, 1992). Estes termos permitirão que definições e restrições usadas sejam expressas por meio de axiomas.

- *Formulação das questões de competência formais* - as questões de competência são reformuladas usando a linguagem lógica de primeira ordem. Uma vez que estas questões foram propostas e a terminologia foi definida, as questões de competência são definidas formalmente.
- *Especificação dos axiomas* - nesta fase a ontologia é formalmente codificada em uma linguagem lógica. Os axiomas especificam as definições dos termos da ontologia e das restrições em sua interpretação. Eles são definidos em sentenças de primeira ordem para definir os termos e restrições na ontologia, definindo a semântica, ou significado, destes termos.
- *Verificação dos teoremas de completude da ontologia* - uma vez que as questões de competência foram formalmente declaradas, é necessário definir as condições sob as quais as soluções para as questões são completas.

2.2.3 – Metodologia 101

Esta metodologia foi desenvolvida por autores envolvidos na criação de ambientes de edição de ontologias como Protege-2000, Ontolingua e Chimaera (NOY, MCGUINNESS, 2001). Eles propõem um guia simples baseado num processo iterativo que ajuda os desenvolvedores, mesmo que não sejam especialistas em Engenharia de Ontologias, a criarem uma ontologia usando estas ferramentas.

A seqüência de passos para desenvolver uma ontologia usando essa metodologia é resumida abaixo:

- *Escolha do domínio e escopo da ontologia* - Determine quais perguntas a ontologia deverá responder. Perguntas de competência são muito importantes nesse domínio, elas permitem que o desenvolvedor entenda quando a ontologia contém informação suficiente e quando ela possui o nível esperado de detalhes ou representação. Também nessa fase o desenvolvedor deve se preocupar com quem irá manter a ontologia, uma vez ela seja posta em uso.
- *Reutilização de ontologias existentes* - Procure por ontologias que definem o domínio. Existem bibliotecas de ontologias reusáveis na Web e na literatura

(por exemplo: biblioteca de ontologias Ontolingua, DAML, UNSPSC, RosettaNet e DMOZ).

- *Enumeração dos termos importantes na ontologia* - Escreva uma lista de todos os termos usados na ontologia e descreva esses termos, seus significados e suas propriedades.
- *Definição das classes e hierarquias de classes* - Existem várias abordagens possíveis para criação de uma hierarquia de classes. O processo desenvolvimento *top-down* inicia com a definição dos conceitos mais gerais no domínio e subseqüente especialização dos conceitos. O processo de desenvolvimento *bottom-up* é o caminho contrário do processo anterior. A combinação desses dois processos é chamada de processo *middle-out*.
- *Definição das propriedades das classes* - Adicione todas as propriedades e informações necessárias para que a ontologia responda as perguntas de competência.
- *Definição das restrições das propriedades* - Defina os valores permitidos para cada propriedade, sua cardinalidade, seu domínio e alcance.
- *Criação de instâncias* - Crie instâncias das classes na hierarquia.

2.2.4 – Método KACTUS

O método KACTUS foi desenvolvido como parte do projeto *Esprit KACTUS*, para construções de ontologias no domínio de redes elétricas (BERNARAS, LARESGOITI, CORERA, 1996). Um dos objetivos do projeto KACTUS é investigar a viabilidade do reuso de conhecimento em sistemas técnicos complexos.

Este método para desenvolver ontologias está condicionado ao desenvolvimento da aplicação. À medida que uma aplicação vai sendo construída, a ontologia que representa seu conhecimento também é construída. Esta ontologia pode ser desenvolvida pela reutilização de outras e pode ser integrada a ontologias futuras. Logo, a cada aplicação desenvolvida, os seguintes passos são tomados:

- *Especificação da aplicação* - fornece o contexto da aplicação e uma visão dos componentes que a aplicação tenta modelar.
- *Projeto preliminar* - os conceitos relevantes de alto nível da ontologia são definidos a partir das listas de termos e de tarefas desenvolvidos na etapa anterior.

Meta-Processo de Conhecimento - Esta é a fase de introdução de uma aplicação baseada em ontologia, onde os documentos e o conhecimento necessários são identificados. Este processo consiste das fases mostradas a seguir:

1ª Fase – Estudo de viabilidade - nesta etapa ocorre a identificação de problemas e suas possíveis soluções, determinando as viabilidades econômicas, técnicas e de projeto, para saber se a ontologia deve ser construída ou não.

2ª Fase – Inicialização - nesta etapa são realizadas atividades tais como identificação dos requisitos, onde o objetivo, o domínio e o escopo da ontologia são definidos. É aqui também que pode ser examinada a possibilidade de haver uma integração com ontologias existentes. Em adição, um número de questões de competência devem ser formuladas para capturar os requisitos necessários para a ontologia.

3ª Fase – Refinamento - nesta etapa ocorre a extração do conhecimento, utilizando o método de extração mais aplicável (*top-down*, *middle-out* e *bottom-up*) à fonte de conhecimento disponível. É realizada a construção de uma ontologia modelo contendo os conceitos relevantes e descrevendo os relacionamentos entre eles, onde esta ontologia será usada para se chegar à ontologia desejada, expressada numa linguagem formal.

4ª Fase – Avaliação - nesta fase a ontologia é checada de acordo com o documento de especificação e as questões de competência formuladas anteriormente, sendo que esta fase funciona de maneira iterativa com a anterior, podendo, inclusive, detectar falhas na representação da ontologia. Assim, vários ciclos podem ser executados até a ontologia ser considerada boa o bastante.

5ª Fase – Aplicação e Evolução - durante esta etapa ocorre a aplicação da ontologia em sistemas baseados em ontologia. É aqui onde o engenheiro recebe a recomendação de práticas para lidar a evolução/atualização da ontologia, a manutenção, que faz com que mudanças no mundo real reflitam em mudanças na ontologia.

Processo de Conhecimento - É um processo iterativo da gestão de conhecimento que é aplicado na organização. Neste processo são realizadas as seguintes atividades:

Atividade 1 – Criação ou importação de dados: o conhecimento deve ser criado ou adaptado para se moldar aos padrões da organização.

Atividade 2 – Captura do conhecimento: inclusão dos dados que se referem aos conceitos da ontologia.

Atividade 3 – Recuperação e acesso ao conhecimento: visões adicionais do conhecimento podem ser derivadas, permitindo a inferência de relacionamentos e descrições através de um mecanismo de inferência apropriado.

Atividade 4 – Utilização do conhecimento: a utilização e a reutilização do conhecimento são facilitadas pela ontologia, sendo que este conhecimento pode ser derivado ou adicionado de alguma outra fonte para que as metas do usuário sejam alcançadas.

2.2.7 – METHONTOLOGY

A METHONTOLOGY é um *framework* que possibilita a construção de ontologias e inclui a identificação do seu processo de desenvolvimento, um ciclo de vida baseado na evolução de protótipos e técnicas particulares para cumprir cada atividade (BLÁZQUEZ *et al.*, 1998).

Esta é uma metodologia que engloba o processo de criação da ontologia desde seu início, passando pelos estágios de especificação, conceitualização, formalização, integração, implementação e manutenção, como mostra a Figura 2. O planejamento deve ser feito antes do início destas etapas e a aquisição do conhecimento, a documentação e a avaliação ao longo de todo o ciclo. As fases são distinguidas como segue:

- *Especificação* - deve conter o propósito da ontologia, os possíveis usuários finais, seu nível de formalidade, seu escopo, incluindo o conjunto de termos a ser representado, e a escolha da linguagem.
- *Aquisição do Conhecimento* - há várias fontes de conhecimento que podem ser usadas, tais como análise de texto, troca de informações entre grupos, entrevistas com peritos, através de outras ontologias etc.
- *Conceitualização* - um completo glossário de termos deve ser construído e em seguida os conceitos e verbos devem ser separados e distribuídos em árvores de classificação de conceitos e diagramas de verbos, havendo também a construção de tabelas de condições e regras.
- *Integração* - para reusar definições em outras ontologias é necessário inspecionar as meta-ontologias (conceitos usados para modelar a ontologia) para selecionar a que melhor se ajusta à conceitualização e tentar encontrar

quais outras ontologias contêm definições de termos coerentes com os da conceitualização e as que usam as definições mais apropriadas.

- *Implementação* - a ontologia é codificada em uma linguagem formal.
- *Avaliação* - verificação da corretude da ontologia e sua validação no sistema.
- *Documentação* - ao fim de cada fase um documento deve ser criado, contendo tudo o que foi feito nesta fase.

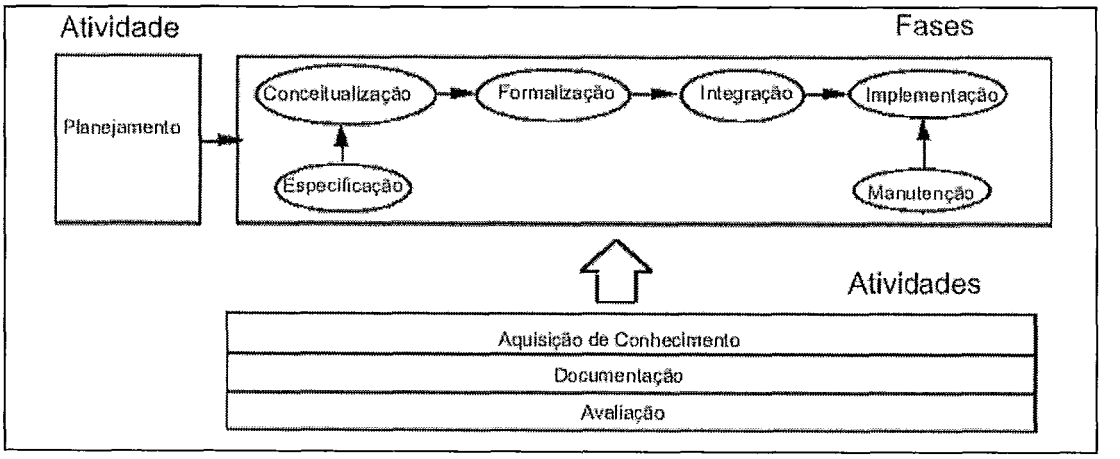


Figura 2 – Fases e atividades do método METHONTOLOGY, adaptado de (FÉRNANDEZ, GÓMEZ-PÉREZ, JURISTO, 1997)

Para o processo de desenvolvimento da ontologia são descritas algumas atividades, que são divididas em:

- *Atividades de Gerência de Projeto* - inclui planejamento, controle e garantia de qualidade.
- *Atividades Orientadas ao Desenvolvimento* - inclui especificação, conceitualização, formalização e implementação.
- *Atividades de Suporte* - incluem uma série de atividades que são executadas paralelamente às atividades de desenvolvimento da ontologia, sem as quais a ontologia não poderia ser construída. Elas incluem aquisição de conhecimento, avaliação, integração, documentação e gerenciamento de configuração, que registra todas as versões da documentação.

O ambiente para construção de ontologias que utiliza esta metodologia é o ODE (*Ontology Design Environment*) (BLÁZQUEZ *et al.*, 1998). Seu objetivo é dar suporte aos engenheiros de ontologia durante o ciclo de vida do processo de desenvolvimento,

automatizando cada atividade e integrando os resultados de cada fase com as entradas da fase seguinte.

A METHONTOLOGY é uma das mais completas e abrangentes metodologias estudadas, já que engloba a maioria (não todos) dos aspectos do processo de desenvolvimento de *software* padronizado pelo IEEE. Ela é baseada na idéia de prototipação e evolução do ciclo de vida da ontologia e é altamente recomendada para o reuso de ontologias existentes.

2.2.8 – Um *framework* para ontologias multilíngües

Os autores (LAUSER *et al.*, 2002) usaram a metodologia METHONTOLOGY e a enriqueceram modificando ações específicas para que o processo de criação e evolução de ontologias de domínio seja semi-automático. A ontologia de domínio é construída usando duas diferentes abordagens de aquisição de conhecimento:

- *Abordagem de aquisição 1* - criação da ontologia núcleo. Uma pequena ontologia núcleo com os conceitos mais importantes e seus relacionamentos é criada. Este estágio é composto basicamente de três passos das atividades de desenvolvimento da METHONTOLOGY: especificação de requisitos, conceitualização do conhecimento de domínio e formalização do modelo conceitual em linguagem formal.
- *Abordagem de aquisição 2* - Deriva uma ontologia de domínio de um *thesaurus*[4].
- *Combinação das ontologias* - Combina a ontologia criada manualmente e a ontologia derivada usando os termos do *thesaurus*.
- *Extensão e refinamentos da ontologia* - Os termos mais freqüentes do domínio são usados como possíveis candidatos a conceitos ou relacionamentos para estender a ontologia. Estes termos devem ser avaliados por especialistas do domínio e conferidos pela relevância na ontologia.

[4] Um "thesaurus" é um instrumento que reúne termos escolhidos a partir de uma estrutura conceitual previamente estabelecida, destinados à indexação e à recuperação de documentos e informações num determinado campo do saber. Não é simplesmente um dicionário, mas um instrumento que garante aos documentalistas e aos pesquisadores o processamento e a busca destas informações (MEDELYAN, WITTEN, 2006).

2.2.9 – Processo KUP

O KUP (Knowledge Unified Process) é um processo unificado proposto em (ORLEAN, 2003) para desenvolvimento de ontologias e bases de conhecimento, que deverão ser utilizadas em aplicações para a Web Semântica. Para garantir a qualidade de desenvolvimento de ontologias, ele foi baseado nos critérios de avaliação de metodologias para desenvolvimento de ontologias propostas por (FERNÁNDEZ-LÓPEZ, GÓMEZ-PÉREZ, 2002) e faz a adaptação e integração das melhores práticas de desenvolvimento de *software* apresentadas pelo RUP (*Rational Unified Process*) (RATIONAL, 1998). Baseia-se também nas atividades do padrão definido pela IEEE para a construção de processos de ciclo de vida e desenvolvimento de *software* (IEEE, 1995) tendo em vista atender aos critérios para avaliação de metodologias e processos de desenvolvimento de ontologias e bases de conhecimento.

Para se resolver o problema do risco que se mantém alto durante todo o processo de desenvolvimento de *software* foram identificadas seis práticas principais, citadas em (ORLEAN, 2003): desenvolvimento iterativo, gerência de requisitos, uso de componentes arquiteturais, modelagem visual, verificação contínua de qualidade e gerência de mudanças. Apesar de estas melhores práticas terem sido propostas para solucionar problemas que ocorrem durante desenvolvimentos de *software*, elas foram aproveitadas no KUP para unificação das atividades propostas no processo.

Diferente da maneira como são executadas as tarefas durante o processo de desenvolvimento de *software* em cascata, o KUP é composto de três fases que são realizadas de forma iterativa ao longo de todo o processo de desenvolvimento e não apenas nas fases finais do projeto. As fases são as seguintes:

- *Fase de Concepção* - foco na análise de viabilidade do projeto, na estratégia de desenvolvimento da ontologia, no escopo e levantamento dos requisitos.
- *Fase de Construção* - foco no projeto, implementação e implantação da ontologia, levando em conta os novos requisitos que possam surgir e mudanças na estratégia.
- *Fase de Evolução* - foco na integração de novos requisitos ao projeto, incluindo novos conceitos, relações e axiomas na ontologia.

Durante as fases do processo artefatos são gerados a partir de atividades executadas nas disciplinas. Uma disciplina agrupa atividades para integrar um processo específico a um determinado tema e um artefato é tudo o que é produzido, consumido

ou modificado por uma atividade, podendo ser um documento, um modelo, um código fonte ou mesmo programas inteiros. O diagrama da Figura 3 descreve as iterações entre as atividades de cada disciplina nas fases propostas pelo KUP.

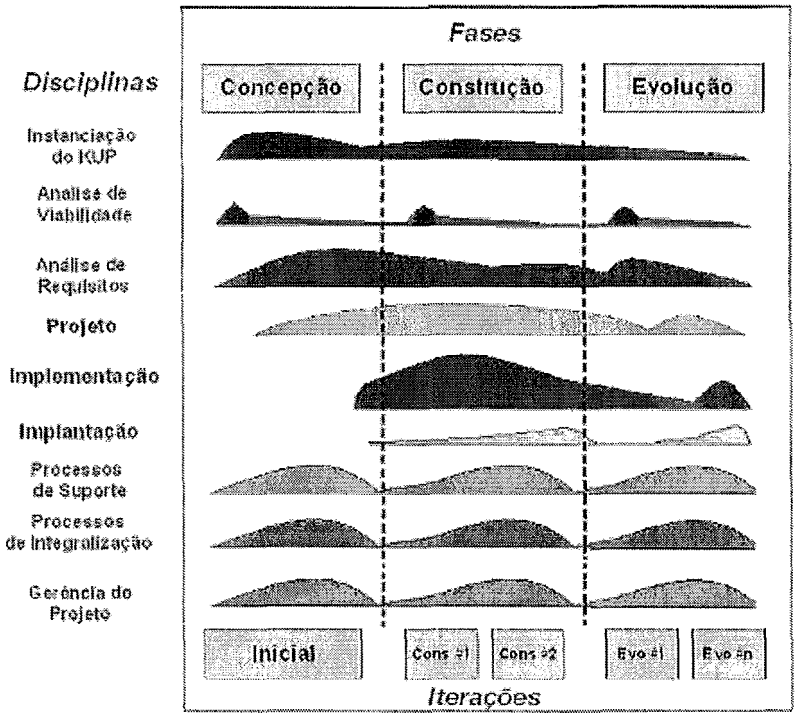


Figura 3 – Diagrama de iterações entre disciplinas e fases do KUP (ORLEAN, 2003)

Para deixar clara a possibilidade de reuso de ontologias nesta metodologia, uma breve descrição das disciplinas e artefatos do KUP se faz necessária. Maiores detalhes podem ser obtidos em (ORLEAN, 2003).

- *Instanciação do KUP* - aqui é proposta a Instanciação do *Framework* de Processos, que inclui a definição das fases, disciplinas, papéis, atividades e artefatos que farão parte do processo concreto que será usado no projeto. Devem ser escolhidos os elementos que garantam a expressividade necessária para a execução do processo e a facilidade de execução; os critérios de monitoramento e controle e de gerência de qualidade do processo devem ser definidos.
- *Análise de viabilidade* - nesta etapa se faz uma identificação prévia de possíveis oportunidades que podem acelerar o processo de desenvolvimento ou de possíveis problemas que podem aumentar o seu risco. Ela deve focar

em aspectos técnicos, econômicos e de projeto, de forma a minimizar os riscos nas fases seguintes.

- *Análise de requisitos* - compreende atividades iterativas que permitem desenvolver a especificação de requisitos da ontologia ou das aplicações que farão uso dela. Esta análise se subdivide em três atividades:
  - Identificação de propósito e escopo: onde será definido o propósito da construção da ontologia e seu uso pretendido, se na comunicação, interoperabilidade ou Engenharia de Sistemas.
  - Captura dos cenários motivacionais: identificação de possíveis situações e problemas que não são cobertos pelas ontologias existentes no domínio em questão. A partir daí serão inferidas possíveis soluções, que fornecerão uma semântica informal aos conceitos e relações que serão posteriormente incluídos na ontologia.
  - Elicitação de requisitos e atores: são levantados os requisitos para o desenvolvimento do projeto, as fontes de informação e os atores dos cenários de uso da ontologia.
  - Projeto: compreende atividades que serão executadas com o objetivo de apresentar uma representação coerente e consistente da ontologia que atenda aos requisitos especificados. É nesta etapa que ocorrerá a geração da ontologia em uma linguagem formal ou um modelo conceitual que permita sua implementação, através de seus conceitos, relações e axiomas.
  - Conceitualização: é proposta a extração dos termos a partir do artefato que descreve as questões de competência informais. Com base no que é extraído são formuladas as questões de competência formais, a partir das quais se gera um Modelo Conceitual para a ontologia, construindo-se sua taxonomia, identificando suas relações de especialização ou generalização entre os conceitos e é feita a adição das relações não taxonômicas para os conceitos.
  - Formalização: é feita a integração das ontologias identificadas na etapa anterior, a linguagem de representação é escolhida, ocorre a especificação formal da terminologia, axiomas e questões de competência.

- Avaliação da ontologia projetada e Refinamento: atividade executada depois da Formalização para que seja possível identificar omissões e erros no artefato gerado. A ontologia deve ser refinada até que todas as omissões ou erros sejam corrigidos e as questões de competência respondidas.
- Implementação: apresenta atividades que irão transformar a representação conceitual da ontologia em sua implementação. Para tal objetivo é preciso escolher a linguagem de representação de conhecimento e o ambiente de engenharia de ontologias mais adequados ao projeto. Após tais escolhas a ontologia é codifcada na linguagem escolhida.
- Implantação: compreende os esforços para integrar a ontologia desenvolvida às diversas aplicações identificadas anteriormente.
- Processos de suporte: são propostas atividades de operação, suporte, manutenção e arquivamento da ontologia e suas aplicações. Estas atividades não estão descritas na versão disponibilizada do KUP.
- Processos de integralização: são propostas atividades de aquisição de conhecimento, verificação e validação, gestão de configuração da ontologia, documentação e treinamento. Estas atividades também não foram descritas.
- Gerência do projeto: realizada paralelamente às outras disciplinas do KUP, para assegurar o gerenciamento no decorrer do ciclo de vida da ontologia, propondo atividades de gerenciamento tais como: definição da equipe, definição de prazos e cronograma.

2.2.10 – Metodologia DILIGENT

A metodologia DILIGENT (TEMPICH *et al.*, 2004) tem por objetivo criar um conjunto de ontologias para que o usuário possa compartilhá-las e, ao mesmo tempo, possa expandir seu uso local seja por seu desejo ou necessidade individual. O objetivo dessa metodologia é superar erros de entendimento das ontologias que são criadas por um pequeno grupo de pessoas (os engenheiros de ontologias e os especialistas do domínio que representam os usuários), mas são utilizadas por um grande número de usuários. Segundo os autores, a engenharia de ontologia precisa lidar com um cenário

onde a evolução das ontologias se dê de forma distribuída e que não seja estabelecido grande controle dessa evolução.

Com essa metodologia, os autores fornecem um modelo de processo para engenharia distribuída das estruturas de conhecimento e planejam torná-la mais completa e fortemente testada.

Ao analisar esta metodologia, podemos dizer que ela é baseada em cinco fases de alto nível que são descritas abaixo:

- *Construção* - A definição de uma ontologia inicial. A equipe envolvida na construção da ontologia inicial deve ser relativamente pequena para que facilite a chega de consenso de uma primeira versão dessa ontologia compartilhada. Não é requerido que a ontologia criada seja possua completude.
- *Adaptação local* - Usuários trabalham em uma ontologia núcleo e a adaptam para seus negócios e necessidades locais. As adaptações locais são registradas e coletadas por um controle central.
- *Análise* - O controle analisa as ontologias locais e as modificações coletadas e tenta identificar similaridades na ontologia dos usuários. Uma das principais atividades do controle é decidir quais mudanças deveriam ser introduzidas na ontologia compartilhada. No final dessa fase, a ontologia compartilhada se encontra numa versão mais refinada.
- *Revisão* - O controle regularmente revisa as ontologias locais e a compartilhada, e avalia as ontologias de ponto do vista técnico e de domínio. Outra tarefa do controle nessa fase é assegurar algumas compatibilidades entre as versões anteriores da ontologia compartilhada.
- *Atualização local* - Com a nova versão da ontologia compartilhada, os usuários atualizam suas ontologias locais para melhor se ajustarem à nova versão da ontologia compartilhada e suas necessidades locais.

A metodologia considera importante que os usuários envolvidos na construção colaborativa da mesma ontologia sejam especialistas com competências diferentes e complementares.

2.2.11 – Considerações

Como foi visto nas seções acima, todas as metodologias para desenvolvimento de ontologias apresentam alguma etapa na qual se torna relevante a compatibilização de conceitos para gerar a ontologia final. A reutilização de conhecimento já estruturado é aconselhada durante o processo de criação ou atualização de ontologias, pois considera-se que seja mais custoso estruturar um novo conhecimento do que adaptar estruturas já existentes (SHUM, 2004). Contudo, em nenhuma dessas metodologias é detalhado como a fase de integração deve ser realizada em termos computacionais. Assim, mesmo que uma ontologia tenha sua construção suportada por metodologias e métodos para criação de ontologias, persiste o problema de como compatibilizar seus termos com os termos de outras diferentes ontologias (FELICÍSSIMO, BREITMAN, 2004).

Essa fase de integração pode ser encarada como um processo de chegada ao consenso, principalmente num cenário onde vários especialistas de domínio trabalham juntos, como abordado na metodologia DILIGENT. Levando-se em consideração que as tarefas de construção e atualização de ontologias estão, cada vez mais, sendo suportadas por ferramentas de edição, essas ferramentas devem também dar suporte à chegada de consenso.

Apresentaremos um método de chegada ao consenso usando técnicas de negociação no capítulo 3. A utilização de tal método pode ser útil em qualquer uma das metodologias apresentadas. Antes, porém, discutiremos na seção 2.3 e 2.4 os principais problemas encontrados durante a compatibilização de conceitos.

2.3 – Cenários de uso para interoperabilidade

A tentativa de prover interoperabilidade sofre de problemas similares aos encontrados na comunicação entre diferentes comunidades de informação. A diferença importante é que os atores não são pessoas capazes de entender e capturar o conhecimento de senso comum sobre o significado dos termos, mas máquinas. Para que máquinas possam entender umas às outras, temos que explicitar o contexto de cada sistema num nível de formalidade muito maior para que este possa ser inteligível por máquinas (USCHOLD, JASPER, 1999).

Um dos principais benefícios do uso de ontologias é a sua capacidade de prover interoperabilidade semântica entre sistemas, pois oferecem um formato comum para troca de dados (USCHOLD, GRUNINGER, 1996). Cada sistema que deseja trocar

informações com outros deve transferir suas informações por esse *framework* comum. Entende-se por interoperabilidade semântica a capacidade de dois ou mais sistemas heterogêneos e distribuídos trabalharem em conjunto, compartilhando as informações entre eles com entendimento comum de seu significado (BURANARACH, 2001).

Em geral, podem ser identificadas três diferentes formas de uso de ontologias: (1) uma abordagem onde uma única ontologia é utilizada, (2) uma abordagem com múltiplas ontologias e (3) uma abordagem híbrida (FREITAS, STUCKENSCHIMDT, NOY, 2005).

2.3.1 – Abordagem com uma única ontologia

Esta abordagem usa uma única ontologia global que provê um vocabulário compartilhado para a especificação das semânticas, conforme é ilustrado na Figura 4. Nesta abordagem, todas as fontes de informação estão relacionadas com a ontologia global.

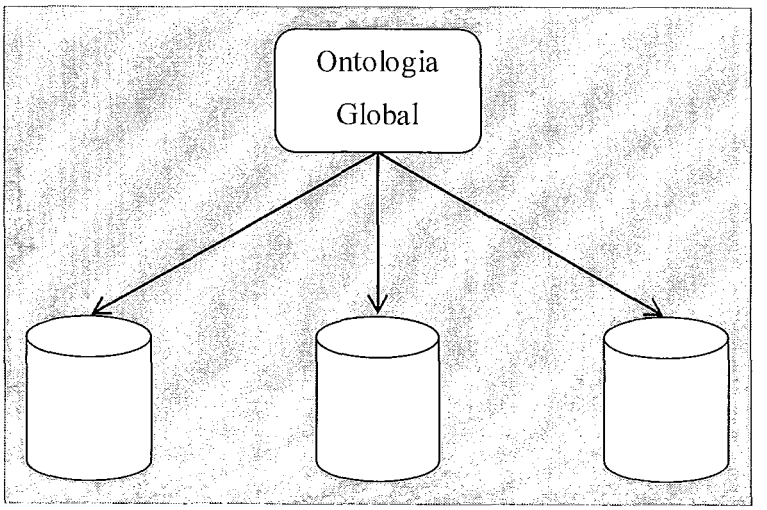


Figura 4 – Abordagem com uma única ontologia

Um exemplo dessa abordagem pode ser visto no sistema SIMS (ARENS, KNOBLOCK, 1992; ARENS *et al.*, 1993; ARENS, KNOBLOCK, SHEN, 1996), descrito anteriormente na seção 2.1.2. O modelo do domínio da aplicação feita no SIMS inclui uma base de conhecimento com uma estrutura hierárquica onde os nós representam objetos, ações e estados. Um modelo independente de fonte de informação precisa ser descrito para o sistema a fim de relacionar os objetos de cada fonte com o

modelo de domínio global. Os relacionamentos indicam a semântica os objetos fonte e ajudam a encontrar objetos semanticamente correspondentes.

Essa abordagem que usa uma única ontologia pode ser aplicada em problemas de integração onde todas as fontes de informação a serem integradas possuem uma visão de domínio muito próxima. Porém, se alguma das fontes de informação possuir uma visão diferente do domínio, por exemplo, ao necessitar de outro nível de granularidade[5], encontrar o comprometimento ontológico mínimo[6] se tornará uma tarefa custosa (GÓMEZ-PÉREZ, CORCHO, FERNÁNDEZ-LÓPEZ, 2004).

Por exemplo, se duas fontes de informação fornecer informações de um produto, mas fizerem referência a um catálogo de produtos heterogêneos, o desenvolvimento de uma ontologia global que combina os diferentes catálogos se torna custoso. Fontes de informação com referência a catálogos de produtos similares são muito mais fáceis de integrar. Além disso, a abordagem de uma única ontologia é suscetível a mudanças das fontes de informação, que pode afetar a conceitualização do domínio representado na ontologia. Dependendo da natureza das mudanças em uma fonte de informação, isto pode implicar em mudanças na ontologia global e nos mapeamentos para as outras fontes. Essas desvantagens levam ao desenvolvimento de uma arquitetura que use múltiplas ontologias.

2.3.2 – Abordagem com múltiplas ontologias

Nesta abordagem, a semântica de cada fonte de informação é descrita por sua própria ontologia, conforme ilustra a Figura 5.

---

[5] A granularidade semântica diz respeito à quantidade de aspectos cognitivos que são representados na ontologia (HORNSBY, EGENHOFER, 2002), quanto mais informação for representada numa ontologia, maior a sua granularidade.

[6] Segundo (GRUBER, 1995), projetos de criação de ontologias devem considerar um (1) comprometimento mínimo com a ontologia e (2) um comprometimento mínimo com a codificação. O primeiro diz que a ontologia deve possuir somente os termos suficientes para dar suporte às atividades de compartilhamento de conhecimento desejadas. O segundo diz que a conceituação deve ser especificada no nível do conhecimento, isto é, sem depender de uma codificação particular no nível simbólico ou de codificação, ou seja, a conceitualização deve apenas considerar o nível de conhecimento; consequentemente as escolhas feitas por conveniências de implementação devem ser evitadas.

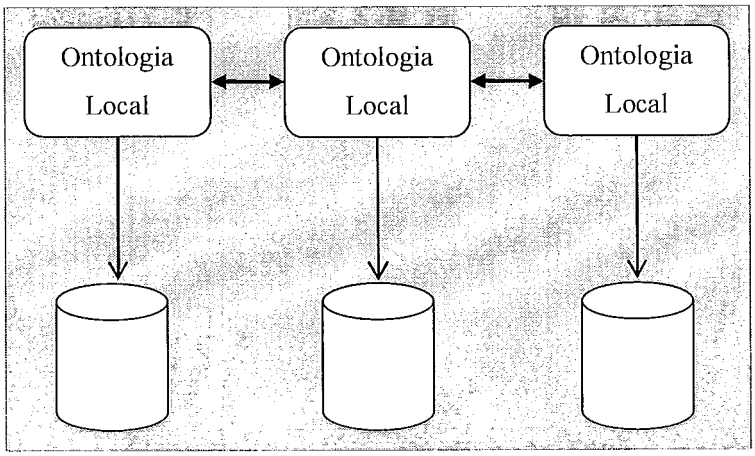


Figura 5 – Abordagem com múltiplas ontologias

Por exemplo, no OBSERVER (MENA *et al.*, 1996) a semântica das fontes de informação é descrita por uma ontologia separada. A princípio, a "fonte de informação" pode ser uma combinação de várias outras ontologias, mas não pode ser assumido que as diferentes "fontes de informação" compartilhem os mesmos vocabulários.

Num primeiro momento, a vantagem dessa abordagem de ontologias múltiplas é que não é necessário uma ontologia comum ou um comprometimento ontológico mínimo com a ontologia global. Cada ontologia fonte poderia ser desenvolvida sem referência às outras fontes ou suas ontologias; então não é necessário que exista uma ontologia comum que seja consenso entre todas as fontes. Essa arquitetura pode simplificar mudanças, por exemplo, modificações em uma fonte de informação ou adição e remoção de fontes.

Contudo, este modelo traz um dos mais difíceis problemas na pesquisa de ontologias: o mapeamento entre diferentes ontologias de domínios similares[7], complementares[8], sobrepostos[9] ou disjuntos[10] para encontrar similaridades e diferenças

[7] Neste trabalho, entende-se por domínios similares aqueles domínios que tratam do mesmo assunto.

[8] Neste trabalho, entende-se que um domínio A qualquer é complementar a um domínio B qualquer, quando A adiciona informações a B. Normalmente, domínios complementares tratam de assuntos diferentes, mas suas partes comuns tratam de um mesmo assunto.

[9] Neste trabalho, entende-se que um domínio A qualquer é de sobreposição a um domínio B qualquer, quando A além de possuir as informações contidas em B também possui informações adicionais sobre o mesmo assunto de B.

[10] Neste trabalho, entende-se por domínios disjuntos aqueles domínios que tratam de assuntos distintos e não possuem partes comuns.

entre elas. Esse também é um dos problemas que são inerentes à Web Semântica, uma vez que a WS é um ambiente com várias ontologias, onde agentes percorrem essas ontologias para coletar conhecimento, transformar dados, responder consultas e outros serviços.

2.3.3 – Abordagem híbrida

Para resolver os problemas das duas abordagens descritas acima, foi criada uma abordagem híbrida (Figura 6). Similar à abordagem com múltiplas ontologias, as semânticas de cada fonte é descrita por sua própria ontologia. Para fazer com que as ontologias fontes sejam comparáveis com as outras, elas são construídas utilizando um vocabulário global compartilhado. O vocabulário compartilhado contém os termos básicos (primitivas) do domínio. Para construir os termos complexos de uma ontologia fonte, as primitivas são combinadas por alguns operadores. Uma vez que os termos de cada ontologia fonte são baseados nas primitivas, os termos se tornam mais fáceis de comparar do que os descritos na abordagem múltipla. Algumas vezes, o vocabulário compartilhado também é uma ontologia.

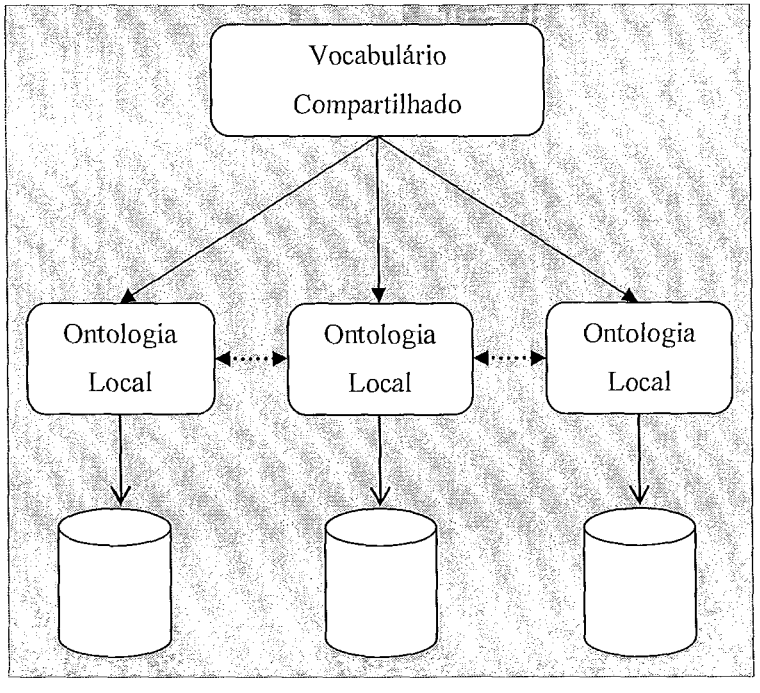


Figura 6 – Abordagem híbrida

A vantagem da abordagem híbrida é que novas fontes podem ser facilmente adicionadas sem a necessidade de modificação nos mapeamentos ou no vocabulário

compartilhado. A abordagem também suporta a aquisição e evolução das ontologias. O uso de um vocabulário compartilhado faz com que as ontologias fontes sejam comparáveis e evita as desvantagens da abordagem múltipla.

Contudo, o problema da abordagem híbrida é que as ontologias existentes não podem ser facilmente reusadas, mas devem ser recriadas, uma vez que todas as ontologias fontes devem referir ao vocabulário compartilhado. Dessa forma, qualquer forma de reutilização de ontologias ou comunicação com outros sistemas fica prejudicada caso as ontologias não sejam recriadas usando o vocabulário compartilhado.

2.4 – Mecanismos para prover interoperabilidade entre ontologias

Uma vez que ontologias possuem a característica de prover interoperabilidade semântica entre fontes de informação, é comum encontrarmos a palavra *integração* sendo usada para descrever tanto a interoperabilidade semântica resultante de seu uso, quanto a compatibilidade entre as possíveis ontologias distintas usadas para gerar essa interoperabilidade semântica (sendo elas de um mesmo domínio ou não).

Contudo, nesse cenário de aplicação onde mais de uma ontologia são utilizadas e, por isso, é necessário compatibilizá-las encontrando suas similaridades e diferenças, a *integração* é somente uma das aproximações para o processo de compatibilidade de ontologias que podem ser aplicadas. Eles são: (1) combinação de ontologias (NOY, MUSEN, 2001), (2) alinhamento de ontologias (NOY, MUSEN, 1999), (3) mapeamento de ontologias (NOY, MUSEN, 2003), (4) integração de ontologias (PINTO, GÓMEZ-PÉREZ, MARTINS, 1999), entre outros. Os principais mecanismos são descritos nas seções abaixo, outros mecanismos podem ser encontrados em (KLEIN, 2001).

2.4.1 – Combinação

A combinação é o processo de construção de uma ontologia de um tema reutilizando duas ou mais ontologias diferentes daquele tema. Neste processo, as ontologias-fontes são unificadas em uma única ontologia e, dessa forma, se torna difícil identificar na ontologia resultante regiões que foram retiradas das ontologias combinadas e que foram deixadas mais ou menos inalteradas. Normalmente as ontologias originais descrevem domínios similares ou de alguma sobreposição.

A Figura 7 ilustra um exemplo de combinação de ontologias, onde os dois conceitos compatíveis, **carro** da ontologia **O1** e **veículo** da ontologia **O2**, são combinados, isto é, unidos, na ontologia única **O**.

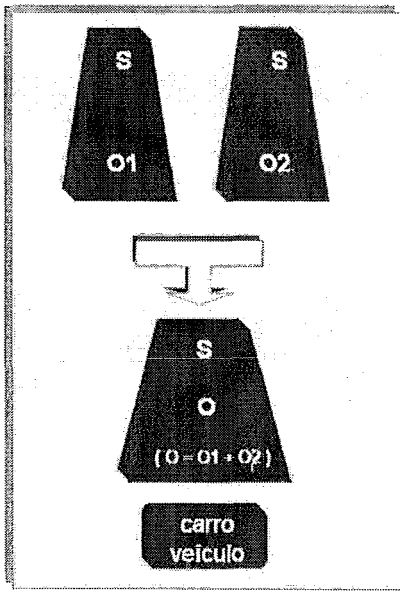


Figura 7 – Combinação de ontologias, adaptado de (FELICÍSSIMO, 2004)

Como pôde ser visto na figura acima, nós temos, como entrada no processo, um conjunto de ontologias (pelo menos duas) que vão ser combinadas (O_1, O_2, ..., O_n) e, como produto, a ontologia resultante O. O objetivo é criar uma ontologia mais geral sobre um assunto ao reunir em uma única estrutura coerente, o conhecimento de várias outras ontologias sobre aquele mesmo assunto. O assunto S das ontologias combinadas e da ontologia resultante é o mesmo, apesar de algumas ontologias serem mais gerais que outras.

Deve-se salientar que no processo de combinação as ontologias-fontes são verdadeiramente ontologias diferentes e não apenas revisões, melhoramentos ou variações da mesma ontologia.

2.4.2 – Alinhamento

O alinhamento de ontologias estabelece ligações entre duas ontologias para permitir que as ontologias alinhadas reusem informação umas das outras. Essa ligação se torna um acordo mútuo entre as ontologias para que se tornem consistentes e coerentes (KLEIN, 2001).

No alinhamento tem-se como entrada duas ontologias que normalmente descrevem domínios complementares e, como resultado, têm-se as duas ontologias originais separadas, mas nestas são adicionadas as ligações entre seus termos equivalentes (FELICÍSSIMO, 2004).

A Figura 8 ilustra um exemplo de alinhamento de ontologias, onde o conceito **carro** de **O1** é alinhado, i.e., ligado, ao conceito **veículo** de **O2**.

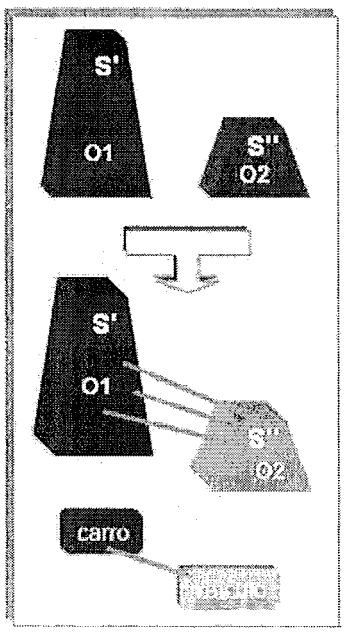


Figura 8 – Alinhamento de ontologias, adaptado de (FELICÍSSIMO, 2004)

2.4.3 – Integração

Neste processo, as ontologias-fonte são agregadas, combinadas para formar a ontologia resultante, possivelmente depois as ontologias reutilizadas terão sofrido alguma mudança, como extensão, especialização ou adaptação. Em um processo de integração, podem-se identificar na ontologia resultante regiões que foram retiradas das ontologias integradas. O conhecimento descrito nestas regiões permanece praticamente inalterado. A Figura 9 ilustra um exemplo de integração de ontologias, onde os conceitos **carro** de **O1**, **veículo** de **O2, automóvel** de **O3** e **meio de transporte terrestre** de **O4** são integrados, i.e., unidos, na ontologia única **O**.

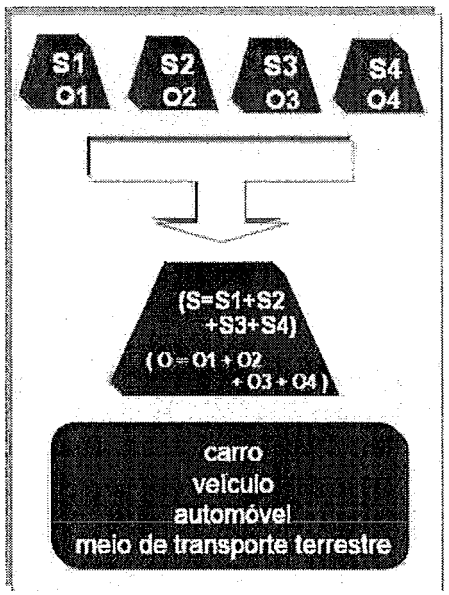


Figura 9 – Integração de ontologias, adaptado de (FELICÍSSIMO, 2004)

Como ilustra a figura acima, nós temos, de um lado, uma (ou mais) ontologias que serão integradas (O_1, O_2, ..., O_n), e na outra, a ontologia resultante do processo de integração. As ontologias integradas são aquelas que serão reutilizadas. Elas são partes da ontologia resultante. A ontologia resultante do processo de integração é a que desejamos construir e, apesar dela ser referenciada como uma ontologia, ela pode ser composta de vários "módulos", que são (sub)ontologias. Isto acontece não só na integração, mas também quando começamos a construir uma ontologia sem reutilizar conhecimento previamente estruturado.

Quando a ontologia integrada é reutilizada pela ontologia resultante, os conceitos integrados podem ser, entre outras coisas (RESNIK *et al.*, 2005): (1) utilizados como estão, (2) adaptados (ou modificados), (3) especializados (levando a uma ontologia mais específica no mesmo domínio) ou (4) acrescidos de novos conceitos (ou por conceitos mais gerais ou por conceitos do mesmo nível).

Os domínios das ontologias integradas usualmente são diferentes entre si, ou seja, cada ontologia que foi integrada à ontologia resultante geralmente diz respeito a um domínio diferente, mas esses domínios são relacionados de alguma forma.

Na integração, a ontologia resultante não deveria se assemelhar a nenhuma outra ontologia previamente existente, caso contrário dever-se-ia simplesmente reutilizar esta ontologia já existente. As ontologias reutilizadas são escolhidas entre as disponíveis nas bibliotecas de ontologias que se encaixam com uma série de requisitos, por exemplo,

domínio, abstração, tipo, generalidade, modularidade, e outros mais (RODRÍGUEZ, EGENHOFER, 2003).

Apesar do resultado final tanto da combinação quanto da integração de ontologias ser uma ontologia única, constituída pela união dos termos das ontologias originais, a principal diferença entre estes dois mecanismos é que, no primeiro, as ontologias tratam do mesmo assunto, o que não acontece necessariamente no segundo.

2.4.4 – Mapeamento

No mapeamento de ontologias tem-se como resultado uma estrutura formal com expressões que relacionam conceitos ou relações similares de uma fonte diferente para a outra através de uma relação de equivalência. Este mapeamento pode ser usado para transferir instâncias de dados, esquemas de integração e de combinação, e outras tarefas similares. A Figura 10 ilustra um exemplo de mapeamento de ontologias, onde os conceitos **carro** de **O1** e **veículo** de **O2** são mapeados em expressões formais.

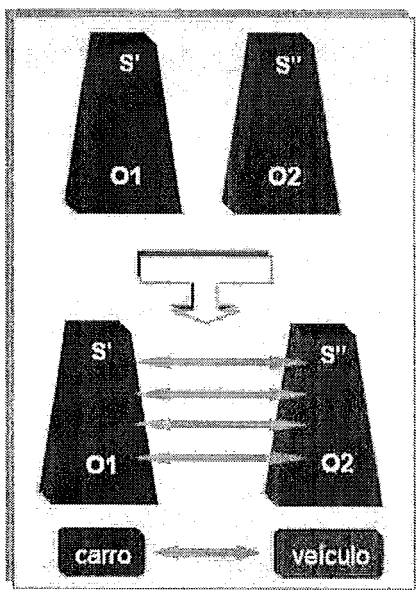


Figura 10 – Mapeamento de ontologias, adaptado de (FELICÍSSIMO, 2004)

Existem pelo menos duas formas diferentes de ser fazer esse mapeamento entre ontologias. Na primeira delas, as duas ontologias a serem mapeadas compartilham uma ontologia de referência comum, como mostrado na Figura 11. Ontologias genéricas[11],

[11] Ontologias genéricas, ou upper ontologies, são aquelas que possuem descrições mais gerais, ou seja, limitam-se a conceitos que são meta, genéricos, abstratos e filosóficos. Conceitos específicos de um dado domínio

como SUMO (NILES, PEASE, 2003), OpenCyc (BUNNINGEN, 2004) e DOLCE (GANGEMI *et al.*, 2003), às vezes são usadas em algumas arquiteturas de ontologias mapeadas pois facilitam o compartilhamento de conhecimento. As duas primeiras ontologias são padrões do grupo de trabalho de ontologias genéricas do *IEEE – Institute of Electrical and Electronics Engineers* (SUOWG, 2006).

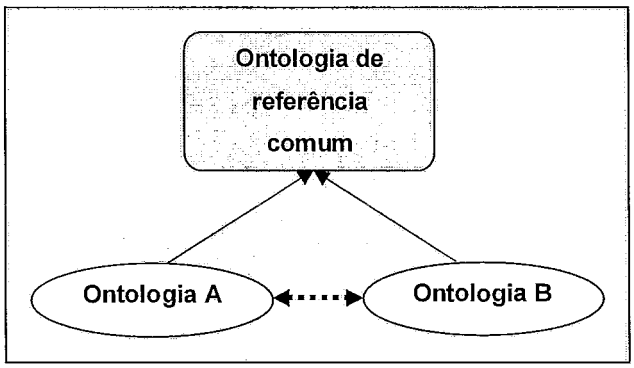


Figura 11 – Mapeamento de ontologias utilizando uma ontologia de referência

Em (GRUNINGER, KOPENA, 2005), os autores propõem um mapeamento de ontologias que é baseado na idéia de uma linguagem compartilhada, na qual não existe mapeamentos diretos entre as ontologias, mas apenas para a linguagem, como mostra a figura abaixo.

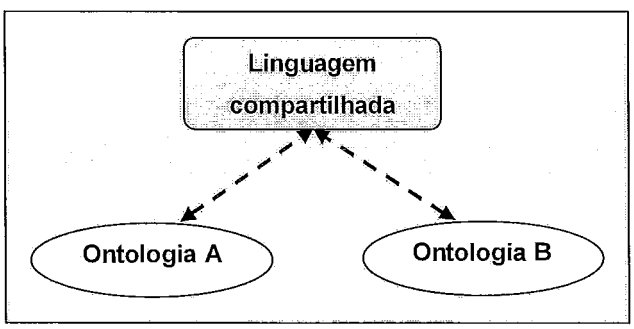


Figura 12 – Mapeamento de ontologias utilizando uma linguagem compartilhada

não são incluídos nas ontologias genéricas. Assim, estas ontologias fornecem estrutura e conceitos genéricos o suficiente para serem utilizados, em um nível elevado, na construção de outras ontologias de várias áreas de domínio (SUOWG, 2006).

Quando não existe uma ontologia compartilhada, podem-se usar ferramentas para realizar um mapeamento diretamente de uma ontologia à outra, como ilustra a Figura 13.

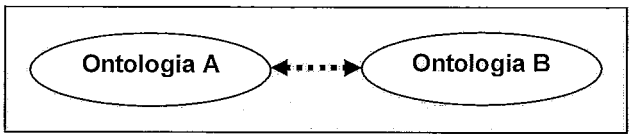


Figura 13 – Mapeamento entre ontologias sem recursos compartilhados

Essas ferramentas utilizam algum tipo de informação estrutural ou uma definição para descobrir novos mapeamentos. Estas informações incluem elementos como relacionamentos de herança, domínio e alcance de propriedades, análises da estrutura de grafo da ontologia, etc (KALFOGLOU, SCHORLEMMER, 2003). Algumas ferramentas utilizadas para criar mapeamentos entre ontologias são QOM (EHRIG, STAAB, 2004), Similarity Flooding (MELNIK, GARCIA-MOLINA, RAHM, 2002) e Prompt (NOY, MUSEN, 2003).

A interação com o usuário é outra fonte importante de informação. Muitos pesquisadores acreditam que um mapeamento completamente automático é impossível de ser feito e, assim, é necessário que haja alguma intervenção do usuário durante o processo. Essa interação pode incluir: alimentar o algoritmo de mapeamento com um conjunto inicial de pares, verificar os pares mapeados que o algoritmo produz ou configurar os cálculos a serem usados (MITRA, WIEDERHOLD, KERSTEN, 2000; MCGUINNESS *et al.*, 2003; NOY, MUSEN, 2003).

Neste trabalho busca-se a identificação dos termos equivalentes de ontologias de domínios com alguma sobreposição e o endereçamento destes termos de forma a permitir a utilização de suas informações. Por estas razões, o mecanismo de mapeamento de ontologias é o escolhido. Como forma de intervenção do usuário, utilizamos técnicas de negociação que serão detalhadas no próximo capítulo. Antes disso, discutiremos, na seção abaixo, alguns métodos que são encontrados na literatura para medir a similaridade de conceitos.

2.5 – Cálculo de Similaridade

Encontrar correspondências entre ontologias é um dos problemas mais difíceis na pesquisa sobre ontologias. Esse problema surge sempre que duas ontologias com alguma sobreposição de termos necessitarem ser usadas numa única aplicação ou por um único agente de *software*. No mundo ideal, existiriam ontologias padrões descrevendo modelos de diferentes domínios: uma ontologia para cada área da medicina, uma para processos de negócios, uma para aplicações de viagens, e assim por diante. Contudo, não apenas este não é o cenário atual, onde temos várias ontologias descrevendo o mesmo cenário, como também é provável que a situação piore no futuro: quanto mais ontologias forem sendo desenvolvidas, mais ontologias com conteúdo similar ou sobreposto existirão. Não é razoável esperar que as pessoas concordarão com um pequeno conjunto de ontologias com pequenas ou nenhuma sobreposição. As razões vão das práticas (diferentes aplicações requerem diferentes visões de um domínio) às institucionais e sociais (uma ontologia desenvolvida em outro lugar pode não ser tão boa quanto à ontologia desenvolvida por nós para nossas finalidades).

Todavia, aplicações que usam diferentes ontologias para descreverem seus domínios ainda necessitam interoperar. E, para isso, necessitamos encontrar correspondência entre diferentes ontologias. Dado duas ontologias, precisamos ser capazes de ver onde estão as similaridades e diferenças entre elas, e expressar essas correspondências (um mapeamento entre as ontologias) de uma maneira processável por máquinas.

2.5.1 – Divergências ontológicas

Quando duas ontologias são comparadas, podem ser identificadas algumas divergências entre elas. É importante identificar quais tipos de divergências ocorrem entre duas ontologias, a fim de resolver estas divergências durante o mapeamento de ontologias. A classificação dessas divergências também é importante para denotar quais tipos de divergências podem ser resolvidas com mapeamento de um formalismo (linguagem de descrição de ontologias) e quais tipos podem ser resolvidos com a ajuda de um algoritmo para cálculo de similaridade.

Neste trabalho, é usado a classificação feita por Klein (2001), que identificou dois níveis de divergências entre ontologias. O primeiro nível é o da linguagem de descrição ou meta-modelo. As divergências inclusas neste nível são as diferenças

sintáticas, diferenças no significado das primitivas nas diferentes linguagens e diferenças na expressividade das linguagens. O segundo nível de divergências é o da ontologia ou modelo, que chamaremos aqui de nível da conceitualização. Essas divergências são detalhadas nas seções seguintes, com maiores detalhes.

2.5.1.1 – Divergências no nível da linguagem

Normalmente, os mapeamentos entre ontologias requerem que as duas ontologias sejam representadas na mesma linguagem. A tradução de uma linguagem para outra pode resolver a maioria dos problemas que podem ocorrer com a diferença de representação. Típicas divergências no nível de linguagem são a sintaxe, a representação lógica, a semântica das primitivas e a expressividade da linguagem (KLEIN, 2001). Como pode ser lido em (PREDOIU *et al.*, 2006), os métodos e ferramentas para mapeamento necessitam que as ontologias sejam representadas no mesmo formalismo e, mesmo que exista uma tradução de qualquer linguagem ontológica para o formalismo desejado, a preservação da semântica pode ainda não ser garantida, pois toda tradução pode gerar perdas semânticas.

Diferenças na sintaxe ocorrem quando linguagens diferentes são usadas na representação de cada ontologia. Por exemplo, para representar o conceito "carro" na linguagem LOOM (LOOM, 2006) usa-se a expressão `(defconcept Carro)`. O mesmo conceito usando a linguagem RDF Schema seria representado com a construção `<rdfs:Class ID="Carro">`. Tal divergência pode ser resolvida através da tradução da ontologia para uma representação qualquer, desde que, no final do processo, ambas as ontologias usem a mesma sintaxe. Essa é uma das divergências mais simples de serem contornadas, porém essa divergência quase nunca ocorre sozinha (JAKONIENE, 2006). Uma diferença que pode ser conseqüência dessa divergência sintática é a encontrada na representação lógica, quando estruturas sintaticamente diferentes, mas logicamente equivalentes, são usadas para representar a mesma coisa. Um exemplo dessa divergência pode ser visto ao representar duas classes disjuntas. Em algumas linguagens é possível representar a disjunção de forma explícita (por exemplo, `disjoint A B`), em outras é necessário usar negações nas subclasses (por exemplo, `A subclass-of (NOT B), B subclass-of (NOT A)`). Nota-se que a divergência não está na capacidade de expressar algo, pois as estruturas são equivalentes, mas quais construtores da linguagem deveriam ser usados para realizar essa explicitação. Além disso, nota-se que essa divergência não diz respeito à representação dos conceitos, mas à representação das

noções lógicas. Este tipo de divergência pode ser resolvido ao usar regras de tradução de uma representação lógica para outra.

Quando a semântica das primitivas é diferente nas diferentes linguagens para descrição de ontologias, isto é, um construtor sintaticamente equivalente tem um significado diferente nas diferentes linguagens, a tradução para uma representação comum precisa levar isto em conta. Por exemplo, existem diferentes interpretações para `A equalTo B` (KLEIN, 2001). Esta divergência também pode ser resolvida com a tradução para uma representação comum, uma vez que, se duas ontologias já utilizam uma representação comum e esta representação não permite estruturas ambíguas, então esta divergência não ocorrerá. A tradução para uma representação comum também é solução para as divergências na expressividade das linguagens, que ocorre quando uma linguagem é capaz de expressar verdades que uma outra linguagem não pode. Por exemplo, algumas linguagens possuem construtores para expressar negação, listas, conjuntos, valores padrão, etc; e outras linguagens podem não possuir alguma dessas capacidades. Essa é a divergência que pode causar maior impacto no processo de tradução, principalmente quando a expressividade da linguagem comum de representação não for um superconjunto da linguagem da ontologia-fonte. Neste caso, alguma semântica pode se perder na tradução (MITRA, WIEDERHOLD, KERSTEN, 2000).

Como pôde ser visto, a tradução de linguagens é uma solução plausível para a maioria dos problemas que surgem com a divergência no nível da linguagem. Neste trabalho, consideramos que as duas ontologias a serem mapeadas estão representadas por uma mesma linguagem de representação (OWL), uma vez que este problema já está sendo bastante explorado pela comunidade acadêmica (VISSER *et al.*, 1997; BUNDY, MCNEILL, SCHORLEMMER, 2003; MCNEILL, BUNDY, WALTON, 2004; MCNEILL, BUNDY, WALTON, 2005).

2.5.1.2 – Divergências no nível da conceitualização

Enquanto as divergências no nível da linguagem incluem diferenças na codificação e significado dos construtores das linguagens, as divergências no nível do modelo incluem diferenças no significado ou codificação dos conceitos nas diferentes ontologias. Essas divergências acontecem quando serão combinadas duas ou mais ontologias que descrevem domínios com alguma sobreposição e podem ocorrer quando

as ontologias estão escritas na mesma linguagem ou em linguagens diferentes (KLEIN, 2001).

Essas divergências podem ser divididas entre (1) divergências de conceitualização, (2) divergências na explicação (ou estruturação) e (3) divergências terminológicas.

Uma **divergência de conceitualização** é uma diferença no modo em que um domínio é interpretado (conceitualizado), o que resulta em diferentes conceitos ou diferentes relações entre esses conceitos. Nas divergências que ocorrem na conceitualização, temos a diferença de escopo que ocorre quando duas classes parecem representar o mesmo conceito, mas não possuem exatamente as mesmas instâncias, apesar de alguma interseção. O exemplo comum para essa divergência é a classe "empregado" (KLEIN, 2001): empresas podem usar conceitos ligeiramente diferentes de "empregado". Por exemplo, como foi levantado em (WIEDERHOLD, 1994), empregados podem ser nomeados como **funcionários** ou **pessoas** nos domínios de **folha de pagamento** e **recursos contratados**, respectivamente. Neste exemplo, a divergência ocorre quando se verifica que no domínio de **recursos contratados** podem-se incluir **pessoas** de outras instituições que não serão inseridas na **folha de pagamento**. Ainda, no domínio de **folha de pagamento**, talvez possa constar auxílios-escola como benefício para filhos de **funcionários**, porém estas crianças, apropriadamente, não estão inseridas no domínio de **recursos contratados**.

Outra divergência se encontra na abrangência do modelo e sua granularidade. Esta divergência é a diferença na parte do domínio que é representado por cada ontologia e no nível de detalhe no qual o domínio é modelado. A Figura 14 ilustra a diferença de granularidade entre a ontologia A, que representa diferentes tipos de computadores pessoais, e a ontologia B, que representa todos os computadores pessoais como uma única classe da ontologia. Chalupsky (2000) exemplifica essa divergência com uma ontologia sobre carros: uma ontologia pode modelar carros, mas não caminhões. Outra pode representar caminhões, mas apenas classificá-los em algumas poucas categorias, enquanto uma terceira ontologia pode explicitar várias distinções entre os tipos de caminhões baseados na sua estrutura física, peso, finalidade, etc.

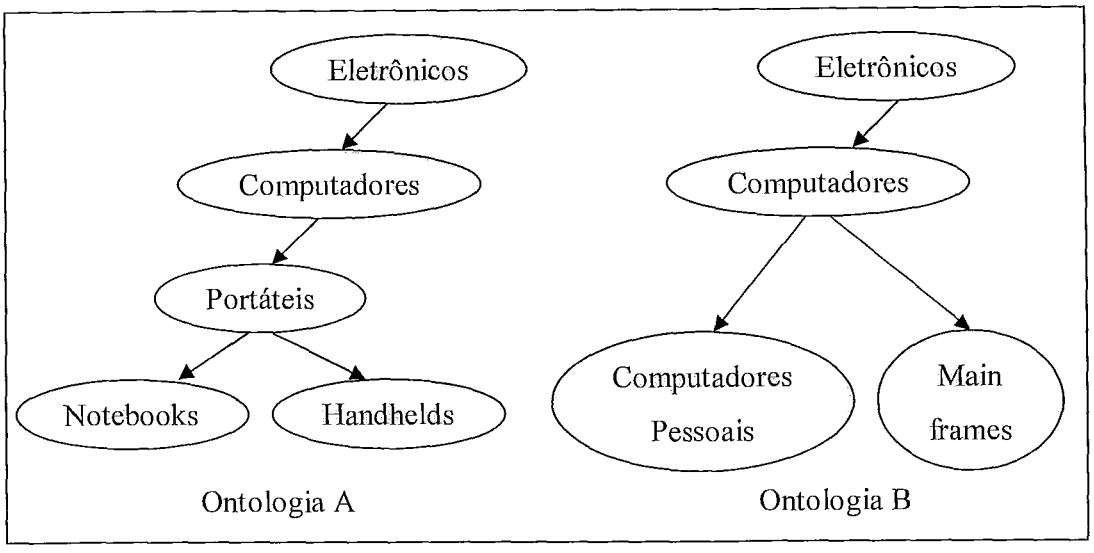


Figura 14 – Diferença de granularidade entre ontologias

Uma **divergência de explicação** é a diferença no modo em que a conceitualização é especificada. Ela pode ocorrer em divergências nas definições, termos ou combinação dos dois, ocorrendo por diferenças no estilo de modelagem usado. Quando diferentes paradigmas são usados para explicar um mesmo conceito, temos uma dessas divergências de explicação. Por exemplo, uma ontologia pode representar tempo usando intervalos, enquanto outra ontologia pode usar pontos. O uso de diferentes ontologias genéricas (*upper ontologies*) também é um exemplo desse tipo de divergência.

Podem existir também divergências no modo em que um conceito é descrito como, por exemplo, as distinções entre duas classes: podem ser modeladas usando uma propriedade que as diferencia ou introduzindo uma classe separada. .A Figura 15 exemplifica essa diferença: a ontologia A possui os conceitos Vinho Branco e Vinho Tinto para diferenciar tipos de Vinho, porém a ontologia B representa essa diferenciação nos tipos de vinhos adicionando o atributo "cor" ao conceito Vinho.

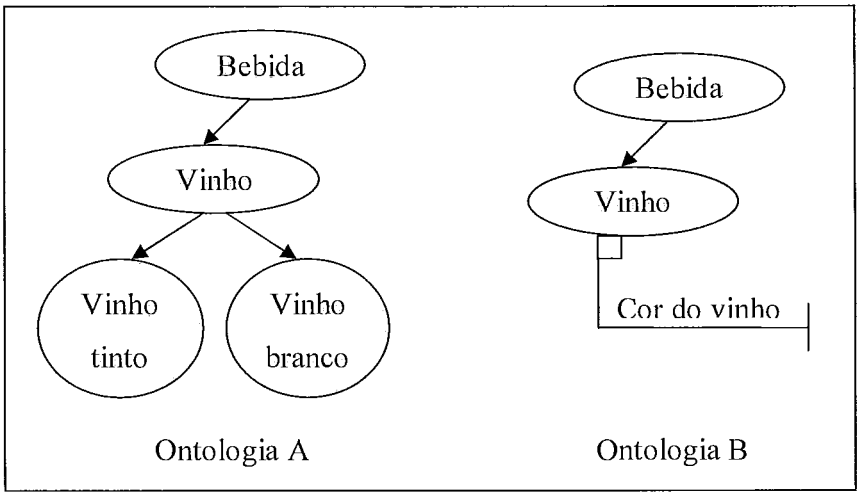


Figura 15 – Divergência no modo no qual o conceito é descrito

Por último, temos as divergências terminológicas, quando aparecem termos sinônimos[12] ou homônimos[13]. Também é considerada uma divergência terminológica as diferenças na codificação. Essa diferença se dá quando valores nas ontologias podem ser codificados em formatos diferentes. Por exemplo, uma data pode ser representada como "dd/mm/aaaa" ou "mm-dd-aa", distâncias podem ser medidas por quilômetros ou milhas, etc.

As diferenças encontradas na conceitualização não podem ser resolvidas automaticamente, segundo Klein (2001). Por isso, requerem conhecimento e decisões de especialistas no domínio. Mesmo que as soluções técnicas para resolver as divergências terminológicas pareçam relativamente simples (uso de *thesauri* para reconhecer sinônimos, transformações para resolver diferenças na codificação de datas e unidades de medidas), a integração de ontologias com sinônimos ou de línguas também requerem esforço humano, pois apresentam vários problemas semânticos. Em especial as inconsistências com homônimos são muito difíceis de resolver computacionalmente e também exigem conhecimento humano para solucionar essa ambigüidade.

[12] Conceitos com mesmo significado, mas com nomes diferentes.

[13] Conceitos com mesmo nome, mas com significados diferentes em outros contextos. Por exemplo, o termo "condutor" quando usado no domínio sobre música tem um significado diferente do usado no domínio sobre engenharia elétrica.

Na seção abaixo, apresentaremos algumas técnicas mais usadas para calcular a similaridade de conceitos em ontologias, que têm como finalidade reduzir as divergências entre ontologias.

2.5.2 – Técnicas para cálculo de similaridade

A tarefa principal ao mapear ontologias consiste em encontrar qual entidade de uma ontologia corresponde à entidade de outra ontologia. Apresentamos, nesta seção, métodos básicos que permitem medir essa correspondência. Existem diferentes técnicas para medir a similaridade entre pares de entidades vindas de ontologias diferentes usando diferentes métodos projetados no contexto da análise de dados, aprendizado de máquinas, engenharia de linguagens, estatística ou representação do conhecimento. Nesta seção, faremos uma breve apresentação dessas técnicas, um estudo mais aprofundado pode ser encontrado em (RAHM, BERNSTEIN, 2001; EUZENAT, VALTCHEV, 2003; GIUNCHIGLIA, SHVAIKO, 2003; SHVAIKO, 2004b).

Os métodos baseados em palavras utilizam da estrutura do termo (sua seqüência de letras) para encontrar a similaridade entre pares de palavras. Existem várias formas de comparar palavras, dependendo do modo em que ela é vista (COHEN, RAVIKUMAR, FIENBERG, 2003). A normalização é uma fase básica usada para ajudar a melhorar os resultados das comparações com palavras. A fase de normalização pode se preocupar em converter letras em caixa alta ou caixa baixa, plurais ou singulares, retirar acentos, espaços, ou *stopwords*[14]. Um método simples para comparação de palavras é a comparação por prefixos e sufixos, onde compara-se se uma palavra começa ou termina com a outra. Porém esse método convive com alguns problemas como, por exemplo, encontrar alta similaridade entre palavras como "bola" e "bolacha" ou "espaço" e "aço".

Após a normalização das palavras, algumas técnicas são usadas para calcular a similaridade de pares. A técnica mais imediata é a distância de Levenshtein (EUZENAT *et al.*, 2004) que conta o número de posições nas quais as duas palavras diferem, ou seja, a distância de Levenshtein, ou distância de edição, calcula o número mínimo de inserções, exclusões e substituições de caracteres que são necessários para transformar

[14] Palavras com alta freqüência na coleção de documentos não são capazes de diferenciar um documento do outro. Essas palavras são chamadas de stopwords e são normalmente filtradas para a indexação por termos, que incluem artigos, preposições, conjunções e outras (BAEZA-YATES, RIBEIRO-NETO, 1999).

uma palavra em outra. Por exemplo, a distância de Levenshtein para as palavras "bolha" e "bola" é igual a 1, uma vez que somente uma operação é necessária para transformar uma palavra na outra: excluir a letra "h" de bolha ou inserir a letra "h" após o "l" em bola. Outro cálculo é a contagem de partes em comuns entre as palavras, chamado de N-gram. Nessa técnica, as duas palavras são divididas em partes de tamanho "n" e quanto maior o número de partes iguais, maior a semelhança entre os termos. Por exemplo, dividindo a palavra "calça" em três, temos "cal", "alç" e "lça".

Um método comum baseado em linguagem é o algoritmo de *stemming*, que reduz as palavras aos seus radicais (PORTER, 1980). Dessa forma, "caminhão" e "caminhoneiro" possuirão alto índice de similaridade, pois possuem um radical comum "caminh".

Algumas técnicas para encontrar conceitos similares utilizam como critério a estrutura das entidades, como o alcance das duas propriedades (atributos e relações), sua cardinalidade, transitividade e/ou simetria das suas propriedades. Esses métodos muitas vezes são descritos na literatura como abordagens baseadas em restrição[15] (RAHM, BERNSTEIN, 2001). Um exemplo dessa abordagem é o algoritmo Cupid (MADHAVAN, BERNSTEIN, RAHM, 2001), usado para descobrir mapeamentos entre elementos de um esquema. O algoritmo se utiliza, entre outras coisas, da compatibilidade entre os tipos de dados dos atributos do esquema. Tipos de dados iguais dão ao par de elementos um valor de semelhança mais alto.

Dado que, em duas ontologias, entidades com estruturas ou propriedades com domínios e alcances similares podem ser bastante numerosas, esses métodos que se utilizam da estrutura dos elementos são normalmente usados combinados com outros métodos como os vistos acima, a fim de reduzir o número de candidatos. Esses métodos ajudam a determinar a correspondência entre os atributos quando nenhuma conclusão pode ser feita simplesmente ao pesquisar um dicionário de sinônimos ou comparar letras dos termos. Eles podem ser usados como um primeiro passo para eliminar a maioria dos atributos claramente incompatíveis.

Dentro os métodos que utilizam dados de estrutura, estão os que analisam a hierarquia de conceitos das ontologias, procurando por pais ou filhos em comum. Assim, a ontologia é vista como um grafo orientado. Usando o conceito de vizinhança nesse grafo, se dois nós de duas ontologias são similares, então os seus vizinhos

[15] Do inglês "constraint".

possuem também alguma similaridade. Em (FELICÍSSIMO, BREITMAN, 2004), a hierarquia de conceitos das duas ontologias é comparada utilizando-se o algoritmo *TreeDiff* (WANG, 1998), que identifica a subestruturas comuns entre as duas árvores. Dessa forma, os conceitos presentes nas subestruturas comuns são tidos como semelhantes.

Capítulo 3 – GNoSIS: Negociação de Significado para Interoperabilidade Semântica

Neste capítulo apresentaremos nossa proposta para facilitar o problema da chegada de consenso sobre termos ontológicos em grupos multidisciplinares que desejam integrar suas bases de conhecimento a fim de manter suas aplicações interoperáveis. Para tal, nos aprofundaremos no processo de negociação proposto na seção 3.2 e explicaremos nossa arquitetura na seção 3.3. Na seção 3.4, faremos uma comparação entre a proposta e as demais abordagens encontradas na literatura.

3.1 – Motivações

Mesmo utilizando-se de uma estrutura de conhecimento para permitir a interoperabilidade entre sistemas - seja na comunicação entre agentes, na integração de banco de dados ou ainda em outros cenários - ainda assim a interoperabilidade é comprometida quando duas estruturas diferentes de conhecimento são utilizadas, pois correlacionar conceitos de domínios com alguma sobreposição torna-se uma atividade pouco computacional, utilizando-se as tecnologias atualmente conhecidas,

Como visto no capítulo anterior, é possível utilizarmos algoritmos para amenizar o problema do mapeamento de conceitos similares. Contudo, segundo Klein (2001), os problemas encontrados na conceitualização necessitam de esforço humano para serem resolvidos, ou seja, podemos facilitar esse processo utilizando-se de meios computacionais, mas provavelmente não conseguiremos eliminar totalmente a necessidade de intervenção do especialista do domínio. Essa necessidade de interventores humanos também se encontra presente nas idéias de O'Hara (2004), que defende o respeito aos fenômenos sociais nas soluções tecnológicas usadas nas camadas mais superiores da Web Semântica.

Uma vez que a estruturação do conhecimento está presente nas camadas superiores da Web Semântica, identificamos que um fenômeno genuinamente social presente nessa camada é a chegada de consenso para criação e reestruturação dessas estruturas de conhecimento (no nosso caso, ontologias). Assim sendo, criamos um processo de negociação com suporte de ferramentas computacionais para auxiliar a tarefa de mapeamento entre conceitos de ontologias distintas.

A negociação é um processo de interação e comunicação social que envolve a distribuição e redistribuição de poder, recursos e compromissos (RAIFFA, 2003), envolvendo duas ou mais pessoas que tomam decisões e se reúnem para trocar informações a fim de firmar um compromisso. Muitas decisões importantes são negociadas, uma vez que pessoas precisam compartilhar e distribuir recursos escassos. O caráter interpessoal, a independência dos participantes como entidades tomadoras de decisões, e sua interdependência na sua habilidade de alcançar unilateralmente os objetivos contribuem para a complexidade da negociação (KERSTEN, CONCILIO, 2002).

Tradicionalmente, dois tipos de negociação existem: a negociação competitiva (distributiva) e a negociação cooperativa (integrativa) (MARTINELLI, ALMEIDA, 1997; HUNG, MAO, 2002; CLARKE, 2006). A negociação competitiva é classificada como uma negociação Ganha/Perde. O negociador com uma postura Ganha/Perde escolhe a competição e relacionamentos de curto prazo priorizando os resultados. Dessa forma, os resultados de uma parte são prejudicados em detrimento dos resultados da outra. Este tipo de negociação é descrita como ganha-perde, soma-zero, conflito puro ou competitiva. Neste processo, um ganho para uma parte representa uma perda para a outra, ou seja, cada parte tenta extrair o máximo de concessões. Assim, o negociador que possui maior poder sempre cede menos com o intuito de obter o máximo de vantagens. Segundo Walton & McKersie, a negociação distributiva é quase sempre uma competição sobre divisão de recursos; quem tem melhores resultados é amplamente dependente das estratégias e táticas empregadas (WALTON, MCKENZIE, 1965). Os negociadores focam nas suas diferenças, ignorando o que possuem em comum, pois possuem uma crença errônea de que os interesses da outra parte são contrários aos seus próprios interesses quando, em muitos casos, eles podem não ser completamente opostos (THOMPSON, 1996).

A negociação cooperativa (também conhecida como negociação colaborativa ou integrativa) é classificada como Ganha/Ganha. Neste processo, as partes envolvidas procuram alternativas para ganhos comuns, ou seja, que supram os interesses de todas as partes (ACUFF, 1993; MARTINELLI, ALMEIDA, 1997; CLARKE, 2006).

Em (LOMUSCIO, WOLLDRIDGE, JENNINGS, 2003), os autores definem negociação como um processo pelo qual um grupo de agentes se comunicam para, de alguma forma, tentar chegar a um acordo mutuamente aceitável. Estes participantes da

negociação não são necessariamente pessoas, podem ser qualquer tipo de ator, como agentes de *softwares*.

Estes atores comunicam de acordo com o protocolo de negociação e agem de acordo com uma estratégia. O protocolo determina o fluxo de mensagens entre os negociadores e dita as regras pelas quais os negociadores precisam seguir caso queiram interagir. A estratégia, por outro lado, é o modo no qual um dado negociador age com essas regras a fim de um esforço de conseguir o melhor resultado da negociação. Enquanto o protocolo da negociação de uma informação pública, a estratégia de cada participante é um dado privado (FISHER, URY, PATTON, 1991; BARTOLINI, PREIST, KUNO, 2006).

Como em todo processo, uma negociação pode ser dividida em fases. Em (KERSTEN, NORONHA, 1999a), os autores sugerem três fases da negociação: pré-negociação, condução da negociação e pós-negociação.

Na fase de pré-negociação, o objetivo é o entendimento do problema da negociação. Esta etapa envolve a análise da situação, do problema, do oponente, idéias, alternativas, preferências, níveis de reserva e a estratégia. Além disso, nesta fase, os negociadores planejam a agenda para as negociações e constroem seu BATNA.

BATNA é o acrônimo para "Best Alternative To a Negotiated Agreement", criado por Roger Fisher e William Ury (1991), e pode ser identificado em qualquer situação através da pergunta: "O que nós iremos fazer caso a negociação não seja bem sucedida?". Em termos mais simples, caso o acordo proposto seja melhor que seu BATNA, então deveria aceitá-lo. Se o acordo não é melhor que o seu BATNA, então deveria reabrir as negociações. Se você não pode melhorar o acordo, então deveria pelo menos considerar cancelar as negociações e utilizar sua alternativa (entretanto, os custos que serão necessários ao escolher a alternativa devem também ser considerados). Uma das principais razões para iniciar uma negociação é conseguir resultados melhores que os possíveis sem a negociação (SPANGLER, 2005). Mais detalhes sobre BATNA podem ser encontrados em (FISHER, URY, PATTON, 1991; MILLS, 1991).

Outro conceito também encontrado na literatura sobre negociação, o valor de reserva (também chamado de base ou preço de reserva), representa o menor ponto favorável que alguém aceita um acordo. O valor de reserva deve ser derivado do BATNA, mas nem sempre a melhor alternativa para o valor de um atributo negociado, caso o acordo não seja alcançado, é idêntica ao valor de reserva definido para o mesmo atributo. O exemplo (PAULA, 2006) a seguir ilustra essa possibilidade:

Imagine que alguém, insatisfeito com o local aonde mora, deseja alugar outra residência em outro bairro. Em uma futura negociação, um dos atributos considerados seria o preço do aluguel. Suponha que, esta pessoa não está disposta a pagar um aluguel superior a quinhentos reais, o que determina o valor de reserva para o atributo preço do aluguel em uma futura negociação. Por outro lado, em sua atual residência, o preço atual do aluguel pago é de trezentos e cinqüenta reais. Na elaboração da BATNA, esta pessoa considera que, caso uma futura negociação não termine com um acordo, a melhor alternativa seria continuar morando na atual residência. Deste modo, tem-se um cenário onde a melhor alternativa para um acordo não alcançado (preço do aluguel igual a trezentos e cinqüenta reais) é diferente do valor de reserva estipulado pelo negociador (quinhentos reais).

O segundo estágio da negociação, a condução da negociação, envolve trocas de mensagens, ofertas e contra-ofertas baseadas em diferentes estratégias e nos tipos de negociação. A pós-negociação é a fase que envolve apenas a avaliação dos resultados da negociação e, mais tarde, da atividade da negociação. Estes resultados incluem a informação sobre o compromisso e a satisfação dos negociadores.

Na negociação colaborativa, existem quatro características que permitem distingui-la da negociação competitiva: a criação de valor, o foco nos interesses e não nas posições, a abertura para novas idéias, e a troca de informação relevante, que é usada para aprendizagem e reestruturação do problema (SEBENIUS, 1992; URY, 1993; FISHER, KOPELMAN, SCHNEIDER, 1994; BAZERMAN, 2001). Diferentes autores apontam a significância dessas características e seus impactos na disposição dos negociadores de colaborar ao invés de competir, para encontrar novas possibilidades ao invés de defender suas próprias posições, para trabalhar em conjunto na resolução de problemas ao invés de demandar mais recursos, porém é a criação de valor que irá mostrar os benefícios da cooperação (PAULA, OLIVEIRA, SOUZA, 2004). Como descrito em (KERSTEN, 2001), são possíveis três interpretações sobre a criação de valor:

- As ofertas são de conhecimento de todas as partes, que escolhem a oferta que melhor lhes convêm. Essa forma de oferta geralmente é associada com leilões, onde existem várias partes negociando um único item.
- As ofertas são desconhecidas dos outros ofertantes, mas são de conhecimento de alguém (por exemplo, um analista). Os negociadores selecionam as ofertas que melhor lhes convêm e um terceiro os guiam

para alcançar um compromisso eficiente, evitando conflitos e convergindo os interesses.

- Ninguém conhece as ofertas das outras partes; durante a negociação os negociadores analisam as possibilidades para conseguir melhores resultados e selecionam as ofertas que dominam as ofertas anteriores.

Uma vez que podemos ter mais de um grupo negociando, a negociação é chamada de multiparte. Negociações multiparte envolvem mais de dois participantes, que podem ser indivíduos ou mesmo grupos. Mesmo envolvendo mais de dois participantes, as negociações multiparte podem ser reduzidas em dois "lados" de interesses (bilaterais) ou tomar a forma de múltiplos lados interagindo (multilaterais). Ao falar de negociações multiparte, pensa-se primeiramente em negociações multilaterais, porém coalizões podem se formar nesse tipo de negociação, a tornando bilateral. Uma negociação multiparte existe quando pessoas ou grupos de pessoas interessadas se juntam para negociar e possuem alguma influência no processo de negociação ou, principalmente, podem ser influenciadas pelo resultado da negociação.

Baseado nestes conceitos de negociação, nas próximas seções nós descreveremos nosso modelo de negociação com foco na chegada de consenso sobre significados.

3.2 – O Processo de negociação

A interoperabilidade semântica é vista como um fenômeno emergente construído incrementalmente, o qual depende da freqüência, qualidade e eficiência com as quais as negociações possam ser conduzidas para encontrar acordos nas interpretações comuns dentro do contexto de uma dada tarefa (ABERER *et al.*, 2004). Assim como um conjunto mútuo de crenças forma o "acordo" ou "consenso" na interação entre agentes, sejam eles humanos ou não, acreditamos que o tipo de negociação mais conveniente e apropriado, quando nós imaginamos o consenso numa equipe multidisciplinar, seja a negociação colaborativa. A razão para acreditarmos na negociação colaborativa está na necessidade de conhecer, aprender e entender o conhecimento especial e relevante das pessoas de diferentes domínios de conhecimento, o qual será base para um trabalho bem realizado e provavelmente o responsável pelo valor competitivo do grupo. Contudo, de acordo com (OUKSEL, 1999), as trocas importantes apenas acontecem com base em proposições mutuamente aceitas.

Assim, nosso modelo, denominado de GNoSIS[16], é divido em três partes, que são: preparação ou pré-negociação, condução da negociação e pós-negociação (SOUZA *et al.*, 2006b). Este modelo é representado na Figura 16.

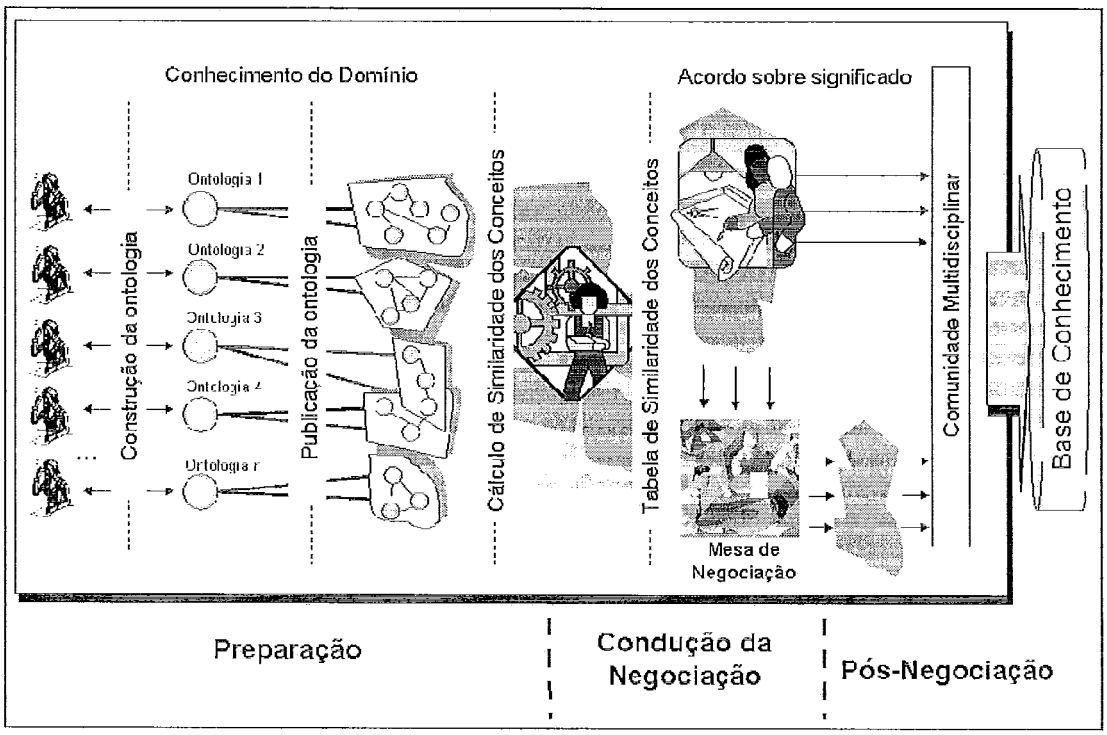


Figura 16 – Processo para negociação do significado, adaptado de (SOUZA *et al.*, 2006b)

3.2.1 – Pré-Negociação

A pré-negociação diz respeito à discussão que precede a negociação formal. No nosso modelo para negociação do significado, esta fase é responsável pela construção e, conseqüentemente, pela escolha da ontologia pessoal ou de domínio. Além disso, essa fase também é responsável pela geração da tabela de similaridade entre os conceitos das ontologias. Esta tabela tem como objetivo indicar aos negociadores os conceitos com maiores possibilidades de mapeamento, direcionando a negociação para os conceitos mais importantes no processo de mapeamento.

[16] Em grego antigo, a palavra "gnosis" é usada para denominar uma forma espiritual de conhecimento que é popularmente conhecida como "sabedoria" ou "iluminação". Para maiores informações: http://en.wikipedia.org/wiki/Gnosis

3.2.1.1 – Publicação da ontologia

A preparação de cada participante da negociação se dá pela escolha da ontologia que será mapeada. Assim, essa fase inicial contempla também a construção ou refinamento da ontologia antes da sua publicação. Qualquer ferramenta pode ser usada manipulação da estrutura de conhecimento pela parte, porém é recomendado que todos os participantes utilizem estruturas de conhecimento no mesmo formalismo para evitar as divergências no nível da linguagem, conforme explicitado na seção 2.5.1.1. São encontradas hoje várias ferramentas para manipulação de ontologias, como Protege-2000 (NOY *et al.*, 2001), OntoEdit (SURE *et al.*, 2002), COE (XEXÉO *et al.*, 2005), OilEd (BECHHOFER *et al.*, 2001), entre outras. A Figura 17 apresenta uma ontologia sendo editada pelo COE.

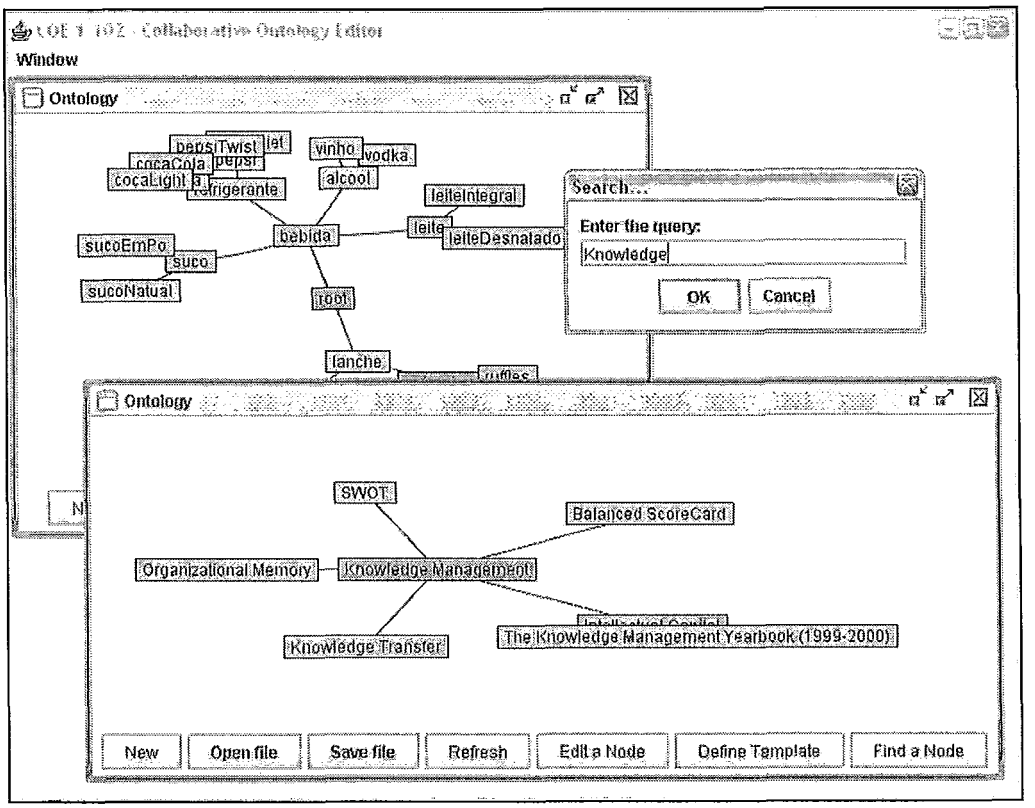


Figura 17 – Ontologia sendo editada na ferramenta COE (REZENDE *et al.*, 2005)

Mesmo sendo a construção da ontologia contemplada nessa etapa do processo de negociação, acreditamos que este processo de negociação será comumente usado por grupos que já possuem uma ontologia definida e em pleno uso. O processo será útil para

auxiliar no mapeamento entre ontologias desses grupos quando surgir a necessidade destes cooperarem e, consequentemente, prover interoperabilidade entre seus sistemas.

Uma vez que cada participante da negociação tenha escolhido uma única ontologia que represente o conhecimento do seu domínio, ela será publicada em um local público, onde os negociadores poderão analisar todas as ontologias postadas. Dessa forma, as partes podem tomar conhecimento de futuras divergências na negociação e preparar seu valor de reserva.

3.2.1.2 – Cálculo de Similaridade

Encontrar similaridades entre conceitos é uma tarefa razoavelmente simples para humanos desde que tenham experiência no domínio em questão. Porém, ao estipular um usuário humano especialista para encontrar relações entre conceitos de ontologias distintas, nos deparamos com uma série de problemas:

- O especialista de domínio instintivamente privilegiará sua visão de mundo em detrimento das outras contidas em ontologias distintas;
- Um único usuário especialista não é necessariamente especialista em todos os domínios possíveis que podem ser necessários conhecer;
- Caso mais de um usuário especialista seja destacado para essa tarefa, conflitos entre eles são passíveis de acontecer, o que antecipa a fase de negociação sem trazer nenhuma vantagem para a próxima mesa de negociação;
- O trabalho de relacionar os conceitos como similares pode vir a se tornar muito custoso em ontologias com um grande número de termos;
- Mesmo que o especialista responsável pela análise de similaridades não tome parte na negociação (tendo o papel de mediador), suas análises podem gerar mais conflitos pelas partes, uma vez que a análise do mediador pode ser entendida como uma decisão arbitrária.

Mediadores não possuem habilidades de decisão e não necessariamente precisam ser especialistas de domínio e, sim, especialistas em resolução de conflitos. O cálculo de similaridade objetiva analisar as estruturas de conhecimento independente de um domínio específico e de uma visão qualquer de mundo. Os termos dos domínios são analisados levando em consideração sua estrutura, ou seja, suas relações, propriedades, restrições, etc, além de ser menos custoso do que a análise manual de estruturas com muitos termos.

Contudo, mesmo o cálculo computacionalmente realizado possui seus custos. O custo principal é o tempo necessário para realizar todas as comparações entre os conceitos. Tendo em vista que alguns algoritmos, para cada conceito de uma ontologia, necessitam fazer comparações com todos os conceitos da outra ontologia, a complexidade de tais algoritmos pode chegar a ser de ordem fatorial[17]. Neste trabalho, estipulamos que o cálculo de similaridade será feito de forma binária ao invés de analisar todas as ontologias de uma só vez, o que pode tornar o trabalho de comparação ainda mais complexo, pois a quantidade de domínios distintos dificulta substancialmente as comparações, como sugere Predoiu *et al* (2006). Além disso, mapeamentos ontológicos são sempre relacionamentos binários, fato que reforça a nossa escolha por calcular (e, posteriormente, negociar) ontologias sempre duas a duas.

O modelo não estipula análises de similaridade específicas, isto é, o modelo é livre para aceitar quaisquer algoritmos que um usuário desejar utilizar, como uma ou mais das técnicas brevemente citadas na seção 2.5.2, e uma implementação desse módulo de cálculo de similaridade deve levar esse requisito em consideração. Para este trabalho, foi desenvolvido um algoritmo de cálculo de similaridade, o qual será posteriormente explicado na seção 3.3.6.

3.2.1.3 – Tabela de Similaridade

Ao final do cálculo das similaridades feito anteriormente, é gerada uma tabela descrevendo os conceitos mais similares de uma ontologia em relação à outra, como ilustrado na Figura 18.

[17] Algoritmos com complexidade fatorial são também chamados de algoritmos de força bruta: tentam todas as possibilidades para problemas de otimização combinatória. Garantem o objetivo, mas não que a solução será alcançada numa quantidade de tempo razoável

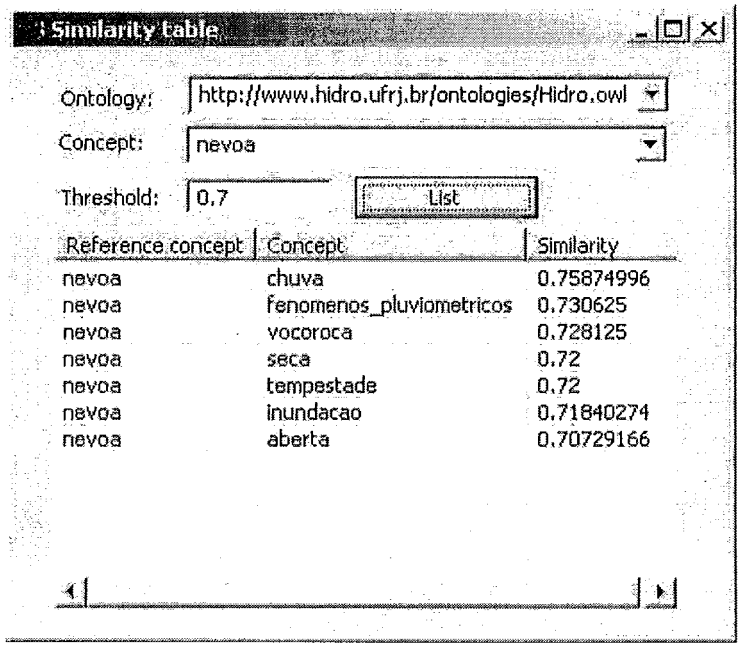


Figura 18 – Exemplo de tabela de similaridade disponibilizada para o mediador

Esta tabela é utilizada pelo mediador para escolher quais mapeamentos serão negociados. A escolha pode ser feita dos conceitos mais similares para os menos similares, ou vice-versa. Quanto menos similar for os conceitos, maior a chance de surgirem conflitos durante a negociação, pois a diferença de estruturação do conceito pode denotar diferenças muito grandes de interpretação do domínio pelos participantes da negociação. Contudo, nem todos os conceitos necessitam ser mapeados, somente aqueles que realmente possuem alguma relação semântica de importância. Tais conceitos, mesmo que selecionados pelo mediador para negociar, poderão ser descartados pelos participantes por estes não concordarem em existir uma relação relevante entre os conceitos. Neste caso, esta rodada da negociação será simplesmente encerrada e outra dupla de conceitos é escolhida pelo mediador.

Na seção 3.2.3.1, descrevemos como o mediador pode ler os valores encontrados nessa tabela e decidir qual mapeamento é o mais adequado.

3.2.2 – Condução da Negociação

A condução da negociação é a fase onde as contrapartes trocam uma série de mensagens e ofertas, criando uma atmosfera adequada pra a negociação, apresentando sua proposta e barganhando até que entrem em um acordo. Os participantes podem conduzir negociações com a assistência de um ou mais participantes neutros, se

utilizando de uma pessoa ou uma equipe como mediadores. Os participantes iniciam as barganhas após tomarem conhecimento do protocolo da negociação definido por um participante neutro. As regras que compõem o protocolo para o modelo de negociação de significados do GNoSIS estão descritas na seção abaixo.

3.2.2.1 – Protocolo da Negociação

O protocolo deve especificar quais são ações válidas durante a negociação, ou seja, quem pode, o que pode e quando pode. A regulamentação definida no protocolo da negociação deve ser obedecida pelas partes envolvidas. Além disso, o protocolo deve ser público e acessível a todos. Esse protocolo é firmado com a ajuda de um mediador da negociação.

Para Moore (1982), o mediador caracteriza-se por ser uma pessoa que ajuda as partes principais a chegarem, de forma voluntária, a um acordo mutuamente aceitável das questões em disputa. O autor acrescenta que a mediação é um processo voluntário em que os participantes devem estar dispostos a aceitar a ajuda do interventor se sua função for ajudá-los a lidar com suas diferenças - ou resolvê-las. O mediador não tem poder normativo, deve possuir uma cultura de debates, de entendimentos, de atualização e de revisão de posições.

O objetivo do mediador é ajudar as partes a negociarem de maneira mais efetiva. Para isto, possui amplos poderes para investigações e perícias de qualquer espécie que julgue necessária, com a finalidade de facilitar a identificação de opções de acordo pelas partes envolvidas. Normalmente, o mediador não sugere uma solução. Contudo, no nosso modelo, o mediador possui a liberdade de sugerir um primeiro mapeamento para iniciar as discussões na mesa de negociação. Essa liberdade dada ao mediador neste primeiro momento só é justificada caso reflita os dados presentes na tabela de similaridade de conceitos previamente calculada computacionalmente, i.e., a sugestão dada pelo mediador neste momento não é feita por conhecimento de domínio do mediador, que podem ser interpretadas como sugestão parcial, mas por análise de dados providos do módulo de cálculo de similaridade, imparcial em sua essência.

São tarefas do mediador (1) definir o protocolo da negociação, (2) controlar a negociação para que o protocolo esteja sendo obedecido, (3) escolher os mapeamentos que serão discutidos com base na tabela de similaridade, (4) interferir na comunicação entre as partes visando organizar a discussão e esclarecer as posições de cada parte, (5) utilizar de recursos externos à discussão para ajudar os negociadores a chegarem em um

consenso como, por exemplo, o histórico de outras negociações, conforme será discutido na seção 3.2.3.2, (6) definir os metadados da negociação como nome das ontologias negociadas, descrição dos domínios que elas descrevem, nome dos negociadores, grupos/empresas a que eles pertencem e a razão pela qual a negociação foi iniciada. Estes metadados serão utilizados como documentação do processo e para facilitar buscas futuras no repositório de negociações (vide seção 3.2.3.2). Os critérios para escolha dos mapeamentos a serem negociados podem seguir a estratégia do mapeamento com menor similaridade entre os conceitos para o menor, ou vice-versa.

A negociação pode ser direta ou com agentes. Na negociação direta, as partes que interagem através do sistema são os interessados diretos e a negociação é conduzida sem agentes humanos intermediários. Em contrapartida, na negociação com agentes as partes que interagem representam os interesses de outro, podendo se configurar como simples intérpretes, atendendo às orientações da parte que defendem ou podendo representá-la de forma pró-ativa.

A negociação é seqüencial, ou seja, deve ser considerada apenas uma relação de mapeamento entre dois conceitos por vez. A cada mapeamento negociado, uma nova negociação é iniciada. Cada negociação possui rodadas. O mediador define qual será a ordem de comunicação de cada participante e o tempo em que cada parte terá para definir sua argumentação. Cada **rodada de negociação** se inicia com a argumentação do primeiro negociador e termina com a argumentação do último negociador definido pelo mediador. Essas informações envolvem a **agenda da negociação**, onde é especificado o tempo máximo de cada rodada de negociação, a ordem em que cada negociador poderá definir sua argumentação e o tempo máximo permitido para o processo. O mediador deve emitir uma mensagem aos negociadores alertando que os prazos estabelecidos para as rodadas de negociação ou para o processo estão prestes a terminar. Este tempo pode variar de projeto para projeto, dependendo da disponibilidade das partes: quando todas as partes estão reunidas no mesmo momento para negociar, então a negociação será feita num ambiente síncrono de mensagens. Caso contrário, a negociação pode ser feita de forma assíncrona e os tempos definidos na agenda possuirão valores maiores.

Durante a negociação, podem ser trocados os seguintes tipos de mensagens: oferta ou sugestão, argumentação, definição e conversação. Toda mensagem deve conter a identificação do negociador responsável pelo envio. Cada mensagem estabelece uma ou mais regras.

Para cada mensagem do tipo **oferta**, o negociador pode associar uma mensagem argumentando a oferta enviada. Essa mensagem é denominada **argumentação** e deve ser composta pelo texto referente à argumentação. Na negociação de significados, ofertas são interpretadas como sugestões de mapeamento entre os dois conceitos.

Para definição dos conceitos, os negociadores deverão trocar mensagens do tipo **definição**. A cada mensagem do tipo definição enviada, a outra parte pode concordar ou enviar outra mensagem do tipo definição com a contra-proposta. Essas mensagens podem ser acompanhadas por mensagens de argumentação.

As mensagens do tipo oferta, argumentação e definição são enviadas, cada uma delas, com um propósito específico e possuem uma estrutura definida em função do seu objetivo. Entretanto, a comunicação entre as partes durante a negociação não pode estar restrita somente aos propósitos considerados nessas mensagens. Por exemplo, um dos negociadores pode necessitar esclarecer uma dúvida quanto à negociação ou, em um determinado momento, uma das partes pode desejar interromper a negociação. Deste modo, prevendo esta possibilidade, será permitida a troca de mensagens do tipo **conversação**. Estas mensagens podem ser consideradas como um recurso para facilitar a comunicação entre as partes e poderão ser enviadas durante toda a negociação.

3.2.2.2 – IBIS

Negociações são compostas por argumentos. A argumentação é a prática de encontrar um senso comum entre os participantes que tomam posições contrárias (MAYFIELD, 2006). No nosso modelo (OLIVEIRA *et al.*, 2006; OLIVEIRA *et al.*, 2007), os membros podem debater, expor suas posições, argumentos e contra-argumentos para a definição do conceito e seus relacionamentos na ontologia. Isto não é útil apenas em descobrir as motivações por trás de cada escolha de uma parte, mas também na criação de uma interação social construtiva.

Para negociar, os usuários podem consultar as ontologias postadas e a tabela de similaridade entre os conceitos. O discurso se desenvolve ao redor de um único item, no nosso caso uma sugestão de mapeamento que pode vir precedida de uma definição de conceito, o qual algum participante da negociação, normalmente o mediador, assumiu que o seu tratamento deveria ser relevante para o mapeamento das ontologias. Os participantes tomam diferentes posições, defendem suas posições, se opõem a outras, e pesam um aspecto contra outro em forma de argumentos.

Esta negociação é realizada em um ambiente eletrônico síncrono e todas as mensagens são categorizadas usando a metodologia IBIS (*Issue Based Information Systems*), introduzido primeira vez em (KUNZ, RITTEL, 1970). A IBIS é uma ferramenta para identificação e resolução de problemas colaborativos. Toda a metodologia é baseada no conceito de argumentação. O sistema é particularmente útil quando, a princípio, um problema é pobremente definido e/ou a concepção inicial da natureza do problema leva os participantes a tomarem posições adversárias (CONKLIN, 2006). Para evitar a polarização que é comum em debates, a IBIS segue um procedimento que envolve a decomposição do problema, conforme é ilustrado na Figura 19.

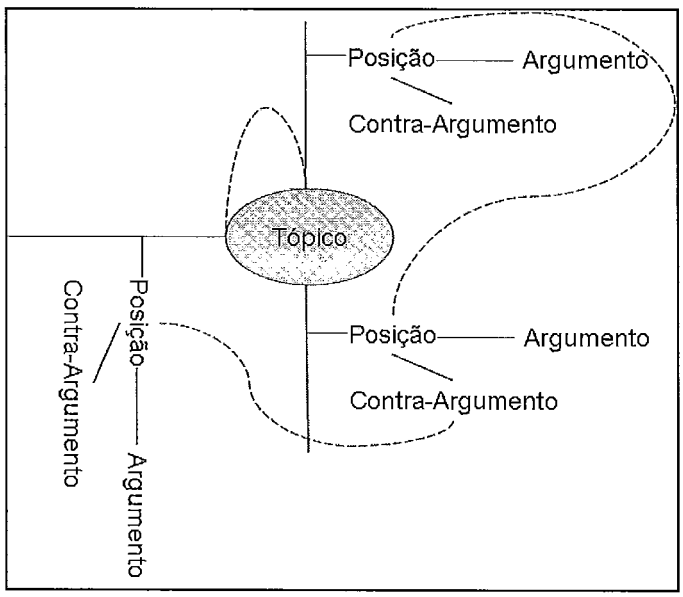


Figura 19 – Categorização das mensagens usando a metodologia IBIS (OLIVEIRA *et al.*, 2006)

A chave do sistema é a idéia de um *tópico*, o qual é por efeito uma *pergunta*. A partir dessa pergunta, artefatos de conhecimento são gerados e armazenados. Todas as mensagens que são trocadas durante uma negociação possuem uma relação entre si. A partir da pergunta inicial, as partes de posicionam positiva ou negativamente, explicitando seus *argumentos*. Todo argumento diz respeito à pelo menos uma posição, toda posição à pelo menos uma idéia, cada idéia à pelo menos um tópico de discussão. Argumentos "apóiam" ou "se opõem a" uma posição; posições "respondem" às idéias.

Outras relações podem ser encaixadas entre os elementos do IBIS. Por exemplo, um argumento pode complementar outro argumento, mesmo que o argumento pertença a uma posição de outra idéia levantada, e diferentes posições também podem se

relacionar (como uma relação de "derivação"). Para facilitar o entendimento, a Figura 20 apresenta uma discussão categorizada pela metodologia IBIS.

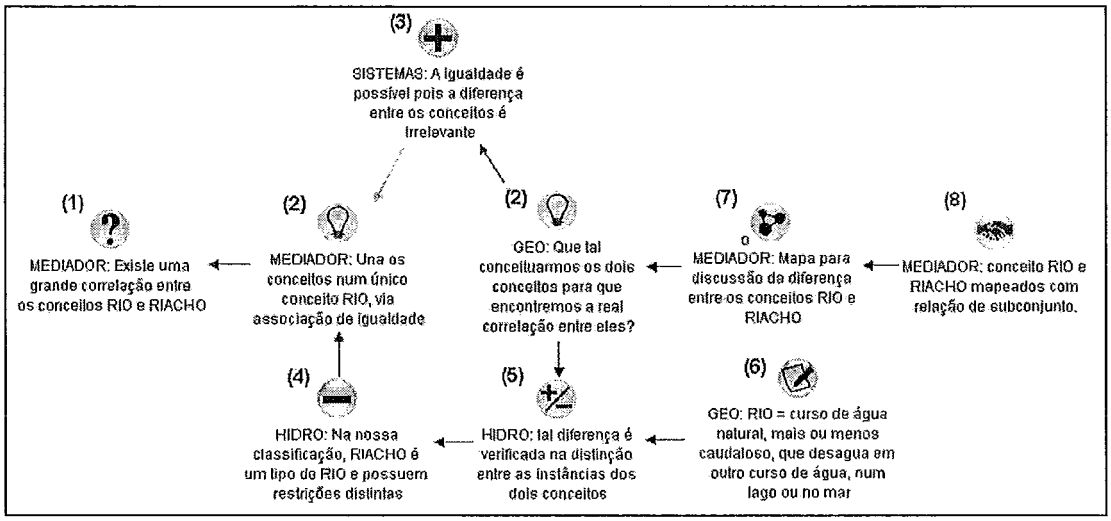


Figura 20 – Exemplo de rede formada utilizando a metodologia IBIS

Na figura acima, os elementos do IBIS são identificados na forma de ícones. O ícone número 1 representa a criação de um tópico, ou seja, uma questão que é enviada para discussão do grupo de especialistas de domínio. Os ícones de número 2 são idéias expressadas por um participante da negociação. Estes ícones denotam mensagens do tipo oferta na negociação de significados. Note que o mediador pode também sugerir resoluções para o conflito, tentando guiar os negociadores a uma conclusão universal benéfica. Essas sugestões são seguidas das posições dos demais negociadores, contras ou a favor. Estas posições possuem também argumentações que justificam a posição do negociador frente à idéia sugerida. Os ícones de números 3 e 4 representam mensagens do tipo argumentação, as quais são a favor e contra a idéia apresentada, respectivamente. Uma argumentação que complementa uma posição é representada usando o ícone de número 5, também uma mensagem do tipo argumentação. Mensagens do tipo definição são representadas como mostra o ícone de número 6. Para não poluir visualmente o mapa da negociação, um novo mapa pode ser criado, como o do ícone de número 7. Por fim, a decisão para o problema inicial é representada pelo ícone de número 8. As mensagens de decisão denotam qual o mapeamento acordado para os dois conceitos. Nos casos em que um acordo não foi firmado ou um mapeamento não foi necessário, a decisão será "Sem acordo" ou "Sem mapeamento", respectivamente.

Mensagens do tipo conversação são enviadas através de um mecanismo de comunicação eletrônica, como *chats* e somente é utilizado, como descrito na seção

anterior, para esclarecer dúvidas quanto ao andamento da negociação ou para cancelar a sua participação no processo, por exemplo.

Embora não exista a garantia de que uma solução apropriada para um dado problema possa ser encontrada, o processo pode revelar respostas nas quais o consenso possa ser encontrado ou descobrir aspectos do problema nos quais outras técnicas de modelagem de raciocínio possam ser usadas pelo mediador.

A metodologia IBIS é utilizada nesta fase para ajudar no entendimento da discussão e no posicionamento de cada negociador, evitando que a negociação perca seu foco. Além disso, a utilização de um ambiente gráfico para organizar a argumentação facilita a visão da negociação, permitindo um fácil entendimento de todo o processo de raciocínio que foi necessário para chegar a certa decisão. Tal facilidade é útil para consulta consultas futuras dessas negociações já concluídas, como será abordado mais à frente, na seção 3.2.3.2.

3.2.3 – Pós-negociação

A pós-negociação é o período após um acordo tenha sido realizado ou, caso não tenha sido possível chegar a um acordo, a negociação tenha sido cancelada ou um processo de renegociação possa ser analisado. Esta fase envolve o compromisso das partes envolvidas na negociação, incluindo o acordo e a satisfação dos negociadores.

No nosso modelo, caso a negociação tenha sido bem sucedida, o acordo entre as partes é firmado com a geração do arquivo de mapeamento e o compromisso de que as partes utilizarão este arquivo da forma com que ele foi elaborado colaborativamente. Assim, o arquivo de mapeamento é disseminado para as equipes no final dessa fase.

Contudo, mesmo com os esforços do mediador e com as facilidades descritas neste modelo, acordos podem não ser firmados dentro do protocolo aceito no início do processo de negociação e esta pode ser abortada pelas partes. Neste caso, o mediador (ou alguma das partes) pode sugerir uma nova negociação ou desistir de tentar novos acordos.

3.2.3.1 – Mapeamento

Esta fase é a responsável por gerar o arquivo de mapeamento entre as duas ontologias e o deixar disponível para as partes. O arquivo de mapeamento é gerado colhendo todas as decisões da negociação.

Um arquivo de mapeamentos é formado por um conjunto de triplas (C_A, C_B, m_i), onde C_A e C_B são, respectivamente, um conceito da ontologia de origem e um conceito da ontologia de destino, e m_i denota a relação semântica entre os dois conceitos, de modo que, sendo M o conjunto finito dos mapeamentos possíveis, então m_i é escolhido onde i=1..k tal que m $\in$ M. Além disso, defini-se que a relação é unidirecional.

A área de estudo dos mapeamentos semânticos ainda se decidiu sobre vários aspectos relacionados sobre esses mapeamentos, como o formalismo que deve ser adotado para descrever essas relações e qual a interpretação universal sobre cada mapeamento. Contudo, o recente artigo de Heiner Stuckenschmidt e Michael Uschold (2005) discute sobre a natureza dos mapeamentos ontológicos e identifica alguns aspectos gerais das abordagens existentes. O principal desses aspectos é a identificação das relações semânticas mais comuns na literatura e que, por isso, tendem a ser as que receberão maior importância numa iminente padronização:

- Equivalência (≡): A relação de equivalência denota que os elementos representam o mesmo objeto do mundo real. Uma forma de equivalência é a igualdade, onde os elementos conectados representam exatamente o mesmo objeto do mundo real. A relação de equivalência é geralmente sugerida pelo mediador para conceitos com alto grau de similaridade.
- Contém ($\supseteq$) e Está contido ($\subseteq$): Essas relações indicam que os elementos de uma ontologia representam um aspecto mais específico do mundo real do que o elemento na outra ontologia. Dependendo de quais elementos são mais específicos, essas relações são definidas uma para cada direção, ou seja, A $\supseteq$ B e B $\subseteq$ A. Geralmente o mediador sugere uma dessas relações quando grupos de conceitos possuem grau de similaridade com valores muito próximos, por exemplo, os conceitos a, b e c de uma ontologia obtiverem valores próximos de 0,8 ao comparados com um dado conceito d da outra ontologia.
- Sobreposição (o): A sobreposição indica que os elementos relacionados representam diferentes aspectos do mundo, mas que possuem alguma característica semelhante. Em particular, esta relação mostra que alguns objetos descritos pelo elemento em uma ontologia pode também ser descrito por outro elemento da outra ontologia. Por exemplo, o indivíduo "Jairo" pode ser descrito pela classe "Bolsista" ou pela classe

"Mestrando", ou seja, são classes distintas, mas que possuem certos elementos em comum que denotam sua sobreposição, porém possuem elementos distintos que não permitem uma relação como "está contido", uma vez que não são todas as instâncias que estão contidas no outro conceito.

Algumas abordagens descrevem também as relações que correspondem às acima negadas. Essas relações podem denotar quando dois elementos não são equivalentes ($\neq$), um não está contido ou não contém o outro ($\not\subset$) ou não possuem nenhuma sobreposição com outro elemento ou, ou seja, são disjuntos ($\varnothing$). Essas oito relações semânticas cobrem todas as relações propostas para linguagens de mapeamento de ontologias (BROCKMANS, HAASE, STUCKENSCHMIDT, 2006). A adição das relações de negação pode, à primeira vista, não ser muito prática, porém essas relações podem ser usadas quando for necessário explicitar uma diferença entre os conceitos, ao invés de uma semelhança, para evitar novas confusões a respeito de sua relação. Assim, consideramos neste trabalho, todos esses oito relacionamentos. Contudo, um novo conjunto de relacionamentos pode ser proposto pelo mediador, caso seja necessário, pois o modelo é livre para agregar novas relações semânticas, uma vez que a negociação em si não difere para cada relação proposta.

3.2.3.2 – Histórico da Negociação

Ao término da negociação, todas as informações que ajudam a representar o contexto da negociação, como as ontologias utilizadas, o mapeamento da ontologia, a tabela de similaridade e o mapa da comunicação categorizada no modelo IBIS, são armazenados no banco de dados. São armazenadas tanto as negociações bem sucedidas quanto as mal sucedidas.

Com isso, o banco de dados se torna uma fonte capaz de prover informações sobre decisões em significados de domínios distintos, podendo auxiliar negociações futuras ao manter registradas quais as razões que levaram um conceito a ser mapeado e todo o raciocínio feito para defini-lo.

Quando negociações futuras envolverem ontologias de domínios já registrados no banco de dados, a negociação e todos os seus artefatos podem ser consultados e referenciados. Tal recurso permite, principalmente, que o mediador possua mais uma ferramenta para resolver conflitos (se baseando em decisões passadas) e que membros da negociação possam aprender sobre conceitos ou domínios que não possuem

conhecimento suficiente. Além disso, a base histórica pode ser utilizada pelas partes como ferramenta de argumentação (quando algum conceito atualmente em discussão já tenha sido objeto de discussão numa negociação passada) ou pelo mediador como ferramenta de renegociação (quando uma negociação esteja se encaminhando para um conflito sem resolução aparente e o mediador identifica uma solução passada que pode ser útil para a negociação atual).

3.3 – Arquitetura Implementada

Mesmo o modelo de negociação proposto não sendo dependente de tecnologias ou ferramentas específicas, foi necessário desenvolver um ambiente que integrasse algumas ferramentas para exemplificar e validar a viabilidade do modelo. A Figura 21 mostra a arquitetura implementada para o modelo proposto.

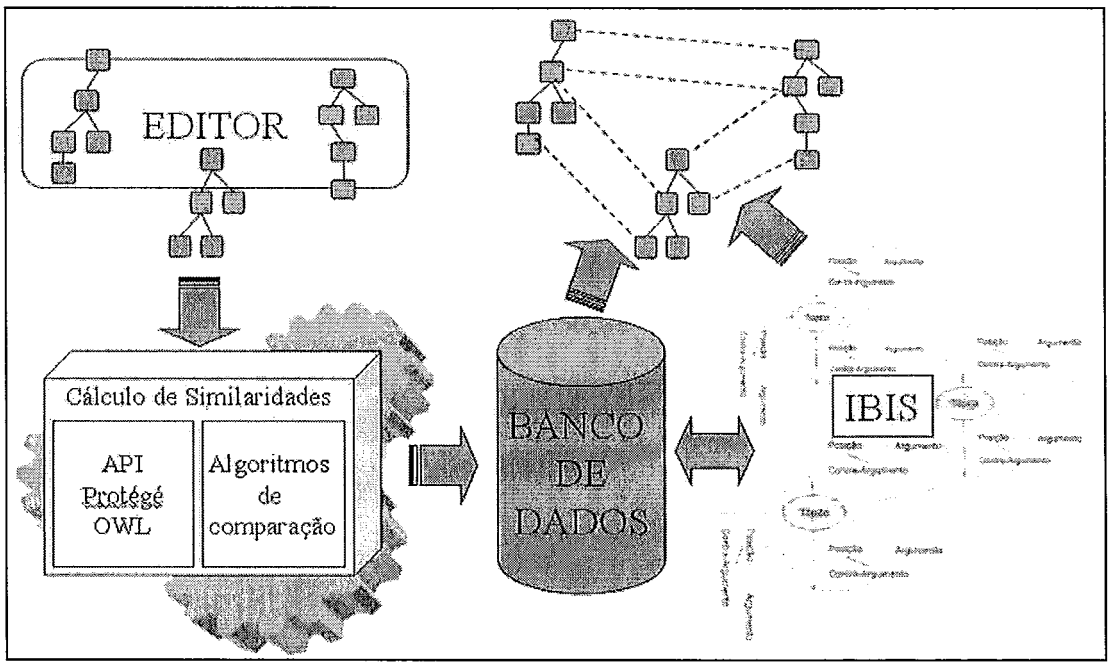


Figura 21 – Arquitetura implementada

3.3.1 – Linguagem de Programação

A linguagem de programação orientada a objetos Java (GOSLING *et al.*, 2005) foi utilizada em toda esta implementação por possuir finalidades gerais e boas características que foram decisivas para sua escolha como a portabilidade, escalabilidade e por ser multi-plataforma. Além disso, uma grande quantidade de bibliotecas programadas para a linguagem Java é disponibilizada na Web. Por esta

razão, o reuso de códigos já implementado é facilitado, o que minimiza o tempo de programação nesta linguagem.

Java pode ser estendida com adições de novas APIs (*Application Programming Interface*) em sua hierarquia. Isto possibilita a utilização de APIs específicas para as soluções programadas nesta linguagem. Por exemplo, a API Protege-OWL (PROTÉGÉ-OWL, 2005), que encapsula e acrescenta facilidades na API Jena (JENA, 2005), específica para o tratamento de ontologias, é utilizada na implementação do cálculo de similaridade.

3.3.2 – Editor de Ontologias

Para que os usuários manipulem as ontologias, é necessário um editor, uma vez que a falta de conhecimento da linguagem de descrição pelos usuários impede a alteração da ontologia diretamente na estrutura XML; além disso, alteração diretamente em código-fonte é altamente passível de erros humanos. Em princípio, qualquer editor pode ser utilizado no modelo. Porém, na nossa implementação, utilizamos o editor COE, ilustrado na Figura 17, que permite a criação colaborativa de ontologias através da utilização de uma rede P2P (XEXÉO *et al.*, 2005). O COE foi aperfeiçoado depois de fazermos uma pesquisa para, entre outras coisas, levantar requisitos de melhoria do editor; as melhorias foram implementadas no trabalho (REZENDE *et al.*, 2006). Com o COE, os especialistas no domínio podem consultar ontologias de outros especialistas para auxiliar o processo de criação. Outra vantagem do COE é que este gera ontologias na linguagem de descrição OWL (*Ontology Web Language*).

A linguagem OWL (DEAN, 2006) é atualmente a linguagem padrão para uma série de linguagens desenvolvidas e evoluídas pelo grupo de trabalho de ontologias da W3C (W3C, 2005b). Esta linguagem é influenciada por formalismos estabelecidos, por paradigmas de representação do conhecimento e pela existência de outras linguagens para ontologias e para a Web Semântica (HORROCKS, PATEL-SCHNEIDER, HARMELEN, 2003). Satisfaz, sobretudo, seus requisitos definidos em (W3C, 2005a). A linguagem respeita a arquitetura da Web Semântica, ilustrada na Figura 1, evoluindo suas linguagens bases: XML, RDF e RDF Schema, além de ser uma revisão da linguagem DAML+OIL, com suas lições aprendidas de projeto e aplicação. A linguagem OWL é descrita em linguagem XML e cada *tag* da linguagem é chamada de **construtor** de um dado elemento ontológico. A linguagem possui construtores para

classes, propriedades, restrições, instâncias e outros elementos herdados das suas linguagens base.

3.3.3 – API Protégé-OWL

A máquina para calcular similaridades foi implementada utilizando-se a API Protégé-OWL. A API é uma biblioteca Java para manipulação de ontologias nas linguagens OWL e RDF. Esta API foi implementada utilizando a API Jena, ou seja, a API Protégé-OWL encapsula a API Jena, acrescentando a ela novas facilidades, ajudando o desenvolvedor na manipulação de ontologias descritas nas linguagens OWL e RDF. Esta API é utilizada em toda a implementação desse trabalho, inclusive no editor COE.

A API Jena transforma uma dada ontologia em um modelo abstrato de dados orientado a objetos e, com isto, seus termos passam a ser manipulados como objetos. Este modelo é baseado na linguagem em que a ontologia é escrita. Por exemplo, existem modelos específicos para OWL, DAML+OIL – *Darpa Agent Markup Language + Ontology Inference Layer* – (CONNOLLY *et al.*, 2001), RDF, entre outros. Isto porque a API em questão recupera as informações das *tags* das ontologias, que são específicas para cada uma das linguagens de ontologias. Nenhuma informação é deduzida, porque a API não é um mecanismo de inferência.

A grande vantagem em transformar ontologias em modelos orientados a objetos é que, com esta transformação, os termos das ontologias são tratados como objetos e a programação orientada a objetos pode ser utilizada. Desta maneira, as manipulações nestes modelos tornam-se transparentes e comuns para os programadores Java, por exemplo.

3.3.4 – IBIS e Banco de Dados

O ambiente para construção da rede de comunicação categorizada pela metodologia IBIS utiliza-se das facilidades do Compendium (CONKLIN, 2005; SHUM *et al.*, 2006), um software livre com código aberto, implementado em Java, que provê uma interface visual para gerenciar conexões entre informações e idéias. O Compendium possui os elementos da metodologia IBIS, entre outros, disponíveis para uso e permite a manipulação desses elementos de forma gráfica.

Criamos um ambiente que incorpora o quadro para desenhos do Compendium e facilidades visuais para utilização do módulo para cálculo de similaridade e do gerador

dos arquivos finais de mapeamento. Além disso, utilizamos um banco de dados mySQL (MYSQL, 2005) que registra continuamente as informações dos objetos inseridos no quadro, assim como suas posições. Assim, podemos simular um ambiente síncrono quando mais de um usuário está utilizando o quadro do Compendium: a tela dos usuários é atualizada com as novas informações registradas no banco numa freqüência pré-determinada por cada usuário, ou seja, quando a tela é atualizada com as informações do banco, tem-se a última versão da rede de comunicações. As atualizações podem ser feitas numa freqüência mínima de cinco segundos. Contudo, quanto menor for o tempo estipulado, menor será a usabilidade do sistema, uma vez que o tempo necessário para o usuário concluir a sua alteração local pode ser maior do que o tempo que o sistema permanecerá em estado de espera aguardando uma nova chamada de atualização de tela. Quando isso ocorre, o sistema fará carga no banco com os dados fornecidos pelos quadros dos outros usuários (posição dos objetos, textos inseridos, relações estabelecidas, etc) até aquele momento e atualizará a tela, sendo necessário o usuário novamente selecionar o objeto IBIS que estava manipulando e continuar o seu trabalho. Recomendamos uma freqüência de atualização de 15 segundos para que esses problemas não ocorram com freqüência indesejada.

O banco de dados também é responsável por armazenar dados históricos. Para facilitar a consulta na base de dados, construímos uma tela simples de pesquisa, conforme mostra a Figura 22. Nesta tela, o usuário pode selecionar os artefatos históricos nos quais deseja pesquisar (descrições de negociações, nomes de ontologias, informações de negociadores ou nas redes de argumentação) e o período de interesse.

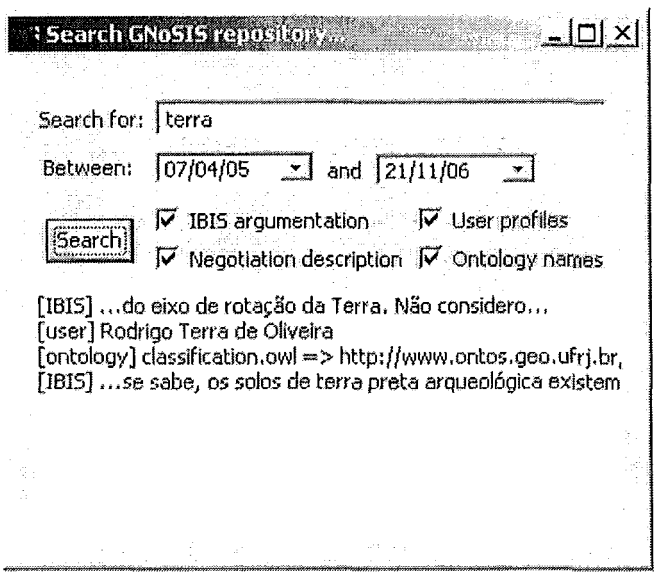


Figura 22 – Tela de pesquisa na base de histórico de negociações

3.3.5 – Mapeamento

Para facilitar a criação dos mapeamentos, criamos uma interface (Figura 23) para que o mediador possa confirmar os mapeamentos decididos no final de cada negociação de significados.

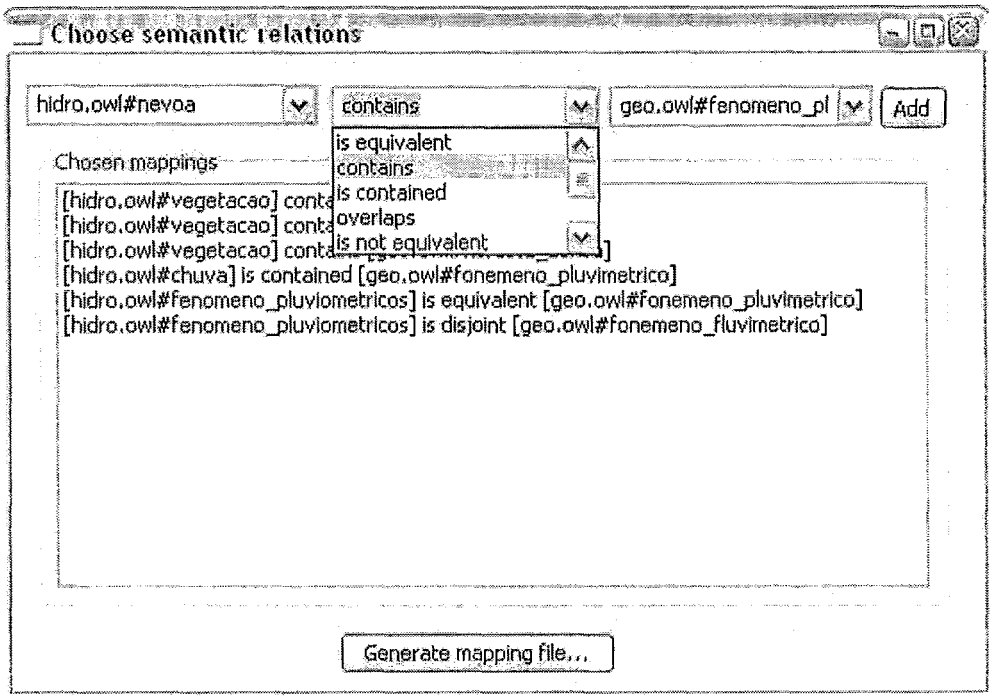


Figura 23 – Janela para geração do arquivo de mapeamento

Ainda não existe uma linguagem padrão para mapeamento de ontologias. A linguagem OWL possui construtores que podem ser usados para denotar mapeamentos, como `owl:disjointWith`, `owl:equivalentClass` e `owl:sameAs`. A diferença entre esses três construtores é que o primeiro é utilizado para indicar classes que são disjuntas, ou seja, representam objetos com nenhuma instância em comum. Ao contrário do primeiro, o segundo construtor é utilizado para indicar que duas classes são equivalentes se, e somente se, possuem precisamente as mesmas instâncias. Por fim, o terceiro construtor é usado para declarar a igualdade de indivíduos, ou seja, somente é utilizado caso haja a necessidade de relacionar instâncias. A falta de construtores nativos da linguagem faz com que a linguagem OWL não seja apropriada para denotar todas as relações que descrevemos na seção 3.2.3.1.

Além disso, também evitamos acrescentar informações de mapeamento nas ontologias originais, uma vez que elas são usadas somente para leitura do algoritmo de cálculo de similaridade e também para consulta dos participantes da negociação, ou

seja, evitamos alterar essas ontologias, pois não podemos avaliar o impacto que tais mudanças podem causar nas aplicações as quais elas estão relacionadas.

Ainda, mapeamentos ontológicos necessitam ser lidos por máquinas de inferências que reconheçam e contenham as regras para cada tipo de relacionamento descrito. Como não existe ainda um consenso nem uma linguagem padrão para descrever essas relações, cada grupo implementa sua própria solução para o conjunto de mapeamentos permitidos e para o formato do arquivo, ambos definidos pelo grupo.

Considerando esses aspectos, decidimos criar um arquivo de mapeamento que contém referências para as duas ontologias as quais ele descreve e adotar um formato próprio para descrever as relações. A Tabela 1 mostra um exemplo de um arquivo de mapeamento gerado pela nossa implementação do GNoSIS.

Tabela 1 - Exemplo de arquivo de mapeamento

```
<mapFile>
   <ontologies>
      <ontology url="http://hidro.ufrj.br/onto/hidro.owl" alias="hidro"/>
      <ontology url="http://labbd.cos.ufrj.br/GNoSIS/ontologias/geo.owl"
alias="geo" />
   </ontologies>
   <mappings>
      <mapping type="equivalency">
         <source>hidro#fenomeno_pluviometricos</source>
         <target>geo#fenomeno_pluviometrico</target>
      </mapping>
      <mapping type="contained">
         <source>hidro#chuva</source>
         <target>geo#fenomeno_pluviometrico</target>
      </mapping>
      <mapping type="disjoint">
         <source>hidro#fenomeno_pluviometricos</source>
         <target>geo#fenomeno_fluviometrico</target>
      </mapping>
        ...
   </mappings>
</mapFile>
```

Como pode ser visto acima, o arquivo de mapeamento guarda o endereço de cada ontologia, a qual é identificada dentro do arquivo pelo valor do atributo "alias". Para cada mapeamento, o arquivo registra o seu tipo e qual a direção na qual o mapeamento é lido (de "source" para "target"). Os valores possíveis para o atributo

"type" são: "equivalency", "contained", "contains", "overlap", "disjoint", "not-equivalency", "not-contained" e "not-contains".

3.3.6 – Cálculo de similaridade

As divergências ontológicas, como discutido ao longo da na seção 2.5, podem ser divididas em (1) divergências no nível da linguagem (diferenças causadas pelo uso de diferentes formalismos) e (2) divergências no nível da conceitualização (diferenças quanto à estruturação dos conceitos na ontologia).

Divergências no nível da linguagem são resolvidas trocando o formalismo de uma ou das duas ontologias. A troca de formalismo também gera novos problemas, como os causados pela diferença de expressividade de um formalismo em relação a outro, mas ainda assim é a solução mais adequada para resolver esse tipo de divergência. Neste trabalho, adotamos a linguagem OWL como padrão de descrição de ontologias e, assim, não foi necessário tratar problemas de divergências desse nível.

Divergências no nível da conceitualização ocorrem, entre outros casos, pela diferença de codificação, uso de sinônimos, uso de ontologias genéricas distintas, diferença de granularidade entre as ontologias, etc. Esses casos exigem uma comparação da estrutura dos conceitos e do contexto, ou seja, uma comparação semântica. Comparações sintáticas podem adicionar bons resultados às comparações semânticas encontrando relações semânticas entre termos, como acontece em muitos algoritmos que mineram corpo de textos (MANNILA, 1996; HOBBS *et al.*, 1997; CHAKRABARTI, 2000; FAATZ, STEINMETZ, 2002). Nesta implementação, fazemos uso tanto de técnicas sintáticas quanto semânticas. A estratégia elaborada para o cálculo de similaridades foi implementada em (SOUZA *et al.*, 2006a).

O cálculo de similaridade é feito utilizando-se funções simétricas de semelhança F, onde F_i, i=1,k; F: A x B → [0,1], A é o conjunto de elementos da primeira ontologia e B é o conjunto de elementos da segunda ontologia. Por exemplo, F_1 pode ser uma função como "a quantidade de propriedades com o mesmo nome de um conceito", onde pode retornar o valor 1 caso as ontologias possuam o mesmo número de propriedades com o mesmo nome, 0 caso não possuam nenhuma propriedade idêntica. Valores intermediários representam o grau de semelhança entre os conceitos, quanto maior o número, maior a semelhança entre os conceitos comparados.

Para cada elemento ontológico (classe, propriedade, restrição, etc), um conjunto de funções de semelhança é aplicado. Em princípio, todos os construtores usados para

criar uma ontologia podem ser úteis no cálculo de similaridade e, assim, ter uma ou mais funções de semelhança associadas.

Tentando ser o mais abrangente possível, consideramos como informação útil no cálculo de similaridade de conceitos os seguintes elementos ontológicos: nome do conceito, sua hierarquia e suas propriedades. Outros elementos poderiam ser usados como restrições e, se for o caso, instâncias.

O problema em se utilizar dados de instâncias está na vasta aplicabilidade das ontologias. Dependendo da finalidade da ontologia, esta pode conter suas instâncias em arquivo OWL, o que seria o caso mais simples, como pode conter suas instâncias como anotações em documentos de vários formatos (HMTL, XML, etc) espalhados em repositórios distintos. Neste último caso, a ontologia em si não conhece quais são as suas instâncias, pois são as instâncias que apontam para o conceito da ontologia e não o contrário. Ainda, ontologias podem simplesmente não possuir instância alguma. Segundo alguns autores (NOY, KLEIN, 2004; SHVAIKO, 2004a), ontologias são encaradas como modelos e uma vez que instâncias necessitam ser armazenadas, o nome dado é **base de conhecimento**. Por tais motivos, o uso de instâncias no cálculo de similaridade se aplica somente a um possível conjunto de ontologias e esses casos não são explorados nessa implementação, tentando ser o mais geral possível. Todavia, acreditamos que um novo trabalho pode ser realizado para utilizar dados de instâncias caso essas existam nas ontologias a serem comparadas e que este pode trazer melhoras nos resultados em comparação ao atual nos casos citados.

Cada propriedade possui funções de semelhança que analisam seu nome e tipo. Em OWL, as propriedades podem ser de dado ou objetos, representadas pelos construtores `owl:DatatypeProperty` e `owl:ObjectProperty`, respectivamente. Propriedades de dados são aquelas que não intrínsecas do conceito, isto é, que não se relacionam com outros conceitos. Por exemplo, podem ser propriedades de dados para o conceito "Pessoa": nome, idade, tamanho, etc. Por sua vez, propriedades de objetos são aquelas que são extrínsecas ao conceito, i.e., que se relacionam com outros objetos. Para o mesmo conceito "Pessoa", são exemplos de propriedades de objetos: "trabalha em" ou "é filho de".

Propriedades de dados possuem tipos de dados primitivos e propriedades de objetos possuem tipo de dados como complexos, ou seja, outros conceitos da ontologia. Assim, por exemplo, a propriedade "nome", citada anteriormente, é uma cadeia de caracteres (ou, em OWL, do tipo `xsd:string`) pois é uma propriedade de dado. Da

mesma forma, as propriedades "é filho de" e "trabalha em" podem ser do tipo "Pessoa" e "Local de Trabalho", respectivamente; ambos os tipos são, obrigatoriamente, conceitos descritos na ontologia.

A primeiras das técnicas (SHVAIKO, 2004a) que usamos como funções de semelhança é a *distância de edição* ou *distância de Levenshtein*. Esta distância recebe duas cadeias de caracteres como entrada e computa a distância entre as cadeias, i.e., que é dado pelo número mínimo de inserções, eliminações e substituições de caracteres que são necessárias para transformar uma cadeia de caracteres em outra. O nome advém do cientista Vladimir Levenshtein, que considerou esta distância em 1965. Esta distância pode ser considerada como uma generalização da distância de Hamming, usada para cadeias de caracteres com o mesmo tamanho. No nosso algoritmo, nós normalizamos a distância de edição, conforme a equação abaixo:

$$D(a,b) = \frac{distanciaEdicao(a,b)}{max(tamanho(a),tamanho(b))}$$

Equação 1 – Distância de edição normalizada

Por exemplo, a distância de edição entre "porteiros" e "portaria" é 0,625, pois, para transformar a palavra "porteiros" na palavra "portaria", são necessárias 5 operações: (1) substituir a letra "e" pela letra "a", (2) eliminar a letra "i", (3) inserir a letra "i" após a letra "r", (4) substituir a última letra "o" pela letra "a" e (5) eliminar a letra "s". Assim, temos a distância de edição igual a 5 dividido pelo tamanho da maior cadeia de caracteres que é 8, resultando no valor de 0,375.

Quanto maior a distância de edição, menor a semelhança entre as cadeias de caracteres. Assim, declaramos uma função de semelhança como:

$$F(a,b) = 1,0 - \frac{distanciaEdicao(a,b)}{max(tamanho(a),tamanho(b))}$$

Equação 2 – Função de semelhança que utiliza distância de edição normalizada

Utilizando a função de semelhança acima, o resultado de F(porteiros, portaria) é igual a 0,375. Ou seja, 1 menos 0,625.

A distância de edição é utilizada para comparação de nomes de conceitos e nomes de propriedades. Também calculamos a semelhança entre os tipos das

propriedades. A função de semelhança para nomes de propriedades é idêntica para propriedades de tipo de objeto e de tipos de dados, porém a função de semelhança para tipos de propriedades é diferente para as duas. Caso as propriedades possuam como valor tipos de dados primitivos (inteiros, pontos flutuantes, etc) idênticos, então elas receberão valor de semelhança 1. Caso contrário, receberão valor de semelhança 0. Caso as propriedades sejam relacionamentos entre conceitos (propriedade de tipo de objeto), então comparamos o grau de semelhança do tipo da propriedade calculando a distância de edição entre os conceitos que essas propriedades se relacionam. Dessa forma, temos que, se duas propriedades se relacionam com conceitos que são semelhantes, então essas propriedades possuem certo grau de semelhança e, por conseqüência, os dois conceitos que possuem essas propriedades também possuem certo grau de semelhança.

Exemplificando as duas funções de semelhança acima, considere os conceitos "Veículo" e "Carro" de ontologias distintas. O conceito "Veículo" possui uma propriedade de tipo de dado chamada "Ano de fabricação" que recebe valores inteiros e uma propriedade de tipo de objeto chamada "possui" que se relaciona com o conceito "Roda". Por sua vez, o conceito "Carro", possui uma propriedade de tipo de dado chamada "Data de fabricação" que recebe valores do tipo data e uma propriedade de tipo de objeto chamada "contém" que também se relaciona com um conceito chamado "Roda", conforme ilustra a Figura 24. Podemos calcular a função de semelhança de nome de propriedades analisando "Ano de fabricação" e "Data de fabricação". Ao aplicarmos a distância de edição no nome dessas propriedades, encontraremos o valor (1 – 4 / 18) = 0,78; vide Equação 2. Aplicando a função de semelhança de tipo de propriedades também em "Ano de fabricação" e "Data de fabricação", encontraremos o valor zero (tipos de dados diferentes). A mesma função de semelhança aplicada às propriedades de tipo de objeto "possui" e "contém" retornará o valor 1 por ser o retorno da distância de edição entre os dois conceitos.

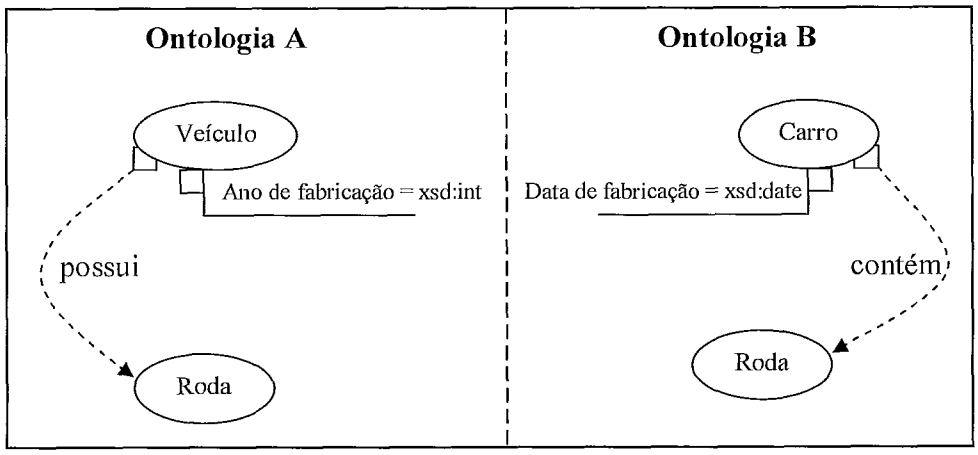


Figura 24 – Trecho de ontologias contendo um conceito e suas propriedades

Levamos também em consideração a hierarquia dos conceitos ao comparamos seus filhos e pais, i.e., dois conceitos não-folhas são estruturalmente semelhantes se o conjunto dos seus filhos imediatos é altamente semelhante. A mesma idéia é também usada para os pais imediatos dos conceitos. Essa função de semelhança de hierarquia analisa o contexto do conceito, ou seja, para calcular o grau de semelhança de dois conceitos A e B, é necessário calcular o grau de semelhança de seus parentes imediatos.

Nesta função, conceitos existentes nas folhas, ou seja, conceitos que não possuem filhos (subclasses), são computados com grau de semelhança total. Da mesma forma, conceitos raízes, ou seja, conceitos que não possuem pais (superclasses), são computados com grau de semelhança total. Essa análise, contudo, é feita somente para fins didáticos. Computacionalmente, não existem nós sem superclasse ou sem subclasse na linguagem OWL. Caso um conceito não possua uma superclasse definida, sua superclasse é o conceito "owl:Thing". Da mesma forma, todos os conceitos existentes nas folhas possuem como subclasse o conceito "owl:Nothing". Esses dois conceitos facilitam nossa análise, pois independente das ontologias a serem comparadas, sempre sabemos de antemão qual o nó mais geral ("Thing") e o mais específico ("Nothing") da hierarquia.

Para um exemplo simples dessa função de semelhança, considere as ontologias da Figura 25.

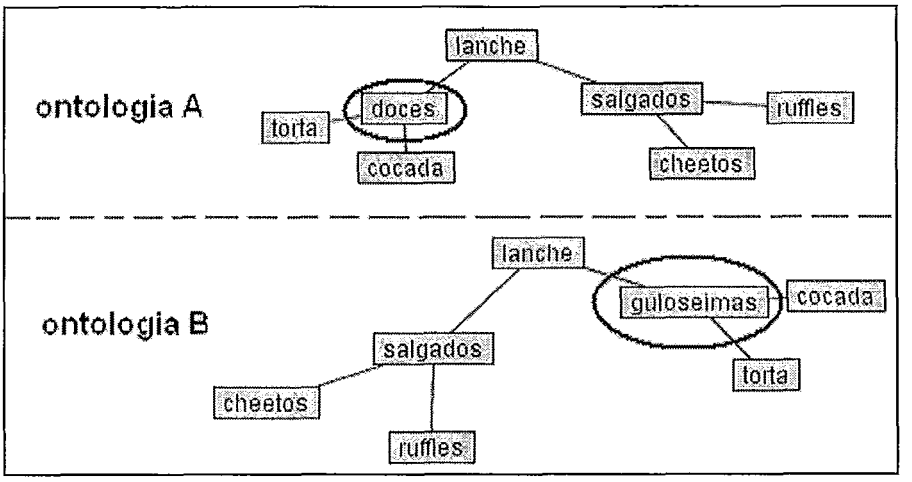


Figura 25 – Ontologias onde os conceitos marcados possuem pais e filhos iguais

Considerando que todos os conceitos possuem as mesmas propriedades, essas ontologias divergem somente quanto ao nome usado em alguns conceitos. Ao compararmos os "guloseimas" e "doces" pela função de semelhança de hierarquia, o grau de semelhança entre os conceitos é 1, dado que eles compartilham do mesmo conceito pai ("lanche") e possuem os mesmos filhos ("cocada" e "torta"). A função de semelhança dos conceitos "lanche", por sua vez, será um valor menor do que 1, uma vez que ambos não possuem superclasse (mostrando, assim, alta semelhança quanto às suas raízes) e as subclasses não são todas iguais: (doces, salgados) x (guloseimas, salgados). Para calcular esta função de semelhança, levamos em consideração o grau total de simlaridade das subclasses e superclasse [18]do conceito.

O grau total de similaridade de dois conceitos é calculado através a equação abaixo, retirada dos estudos de (SOUZA, 1986). A equação é usada para avaliar a função de semelhança F_x aplicada aos pares comparáveis de elementos ontológicos "a" e "b", usando as funções de semelhanças F_q e seus pesos associados W_q, q=1,m, respectivamente.

$$F_x = \sum_{q=1}^{rn} F_q(a,b) * W_q$$

Equação 3 – Somatório das funções de semelhança

[18] Neste trabalho, não consideramos herança múltiplas e utilizamos a subespécie OWL-DL.

A Figura 26 exibe as funções de semelhança (dentro das caixas) e os pesos padrão usados para a comparação entre os conceitos. Os pesos são usados para ajustar o algoritmo e são normalizados para somarem 1. As funções de semelhança retornam 1 para uma semelhança perfeita e um valor positivo menor para pares com menores semelhanças. A equação acima é usada para agregar a informação sobre as sub-semelhanças de baixo pra cima na árvore da figura abaixo, até a raiz. Ou seja, a semelhança final entre dois conceitos de ontologias distintas é dada pela soma das suas funções de semelhança.

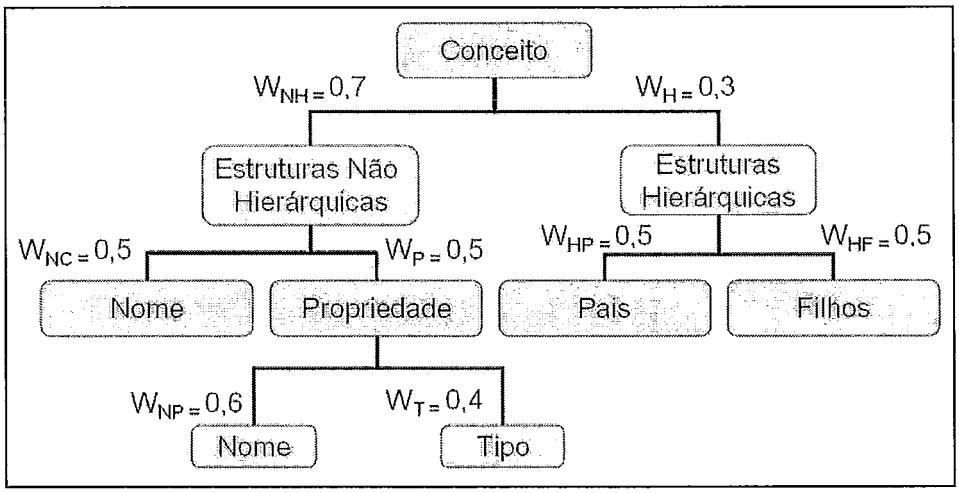


Figura 26 – Funções de semelhança e seus respectivos pesos padrão

Por exemplo, suponha que dois conceitos sendo comparados possuam uma única propriedade, e as funções de semelhança "Nome" e "Tipo" retornaram 0,5 e 1 respectivamente. Esta informação contribuirá para o resultado da função de semelhança "Propriedade" com os pesos 0,6 e 0,4 respectivamente. Assim, a função de semelhança "Propriedade" será calculada como: 0,5*0,6 + 1 * 0,4 = 0,7.

Para os casos onde existam mais de um elemento relacionado a ser usado na função de semelhança, ou seja, os conceitos que possuem várias propriedades ou conceitos com mais de um filho, a média do resultado das funções é usada nos casos onde ambos os lados possuem o mesmo número de elementos. Se alguma informação estiver faltando em uma das ontologias, por exemplo, o número de propriedades que os dois conceitos comparados possuem é diferente, nós utilizamos uma técnica para fazer uma média dos pesos. O objetivo é introduzir uma penalidade (equivalente a um peso negativo) que é calculado como segue, onde L_{min} e L_{max} representam o número mínimo

e o número máximo de propriedades do conceito (ou filhos ou pais do conceito), respectivamente:

$$W_{media} = \frac{\sum_{K=0}^{N} F_k(a,b)}{L_{min} + Penalidade * (L_{max} - L_{min})}$$

Equação 4 – Somatório das funções de semelhança com aplicação de penalidade

A penalidade varia de zero (quando não importa a diferença no número de elementos de um conceito, resultando numa simples média aritmética dos resultados da função de semelhança) até 1 (quando a diferença no número de elementos é muito importante).

Ao analisarmos a semelhança entre dois conceitos onde cada um possui mais de uma propriedade ou que possuem mais de uma subclasse, caímos no problema de escolher quais relações serão escolhidas como relevantes. Por exemplo, imagine o conceito A e B, com respectivas subclasses [a1, a2] e [b1,b2,b3]. A função será aplicada em toda combinação de subclasse gerando, por exemplo, uma matriz como a da Tabela 2, ordenada por grau de semelhança:

Tabela 2 - Matriz com combinações de semelhança entre entidades (SOUZA, 1986)

```
[ a1, b2, 0.8 ]
[ a2, b2, 0.7 ]
[ a1, b3, 0.6 ]
[ a2, b1, 0.4 ]
[ a1, b1, 0.3 ]
[ a2, b3, 0.3 ]
```

O retorno dessa função pode se dar pela escolha da primeira combinação encontrada, sem repetir elementos, onde teríamos [a1, b2, 0.8] e [a2, b1, 0.4] totalizando uma semelhança de 1,20. Um algoritmo que confere todas as combinações para escolher a melhor resposta possível escolheria as combinações [a2, b2, 0.7] e [a1, b3, 0.6] totalizando 1,30. Mesmo em alguns casos retornando um valor inferior, nosso algoritmo se utiliza da primeira abordagem, pois (1) nossa principal preocupação é encontrar as maiores semelhanças entre os conceitos e encontrar um elemento muito semelhante é

mais interessante do que encontrar dois elementos menos semelhantes, mesmo que a soma total das funções seja maior; além disso, (2) encontrar a resposta que retorne o maior valor total é um problema NP completo (SOUZA, 1986) e pode ser impraticável calcular a resposta quando a quantidade de elementos for muito elevada.

Após o cálculo das semelhanças, os conceitos que satisfazem a equação abaixo são listados ao usuário como altamente similares. Nesta aplicação, consideramos o valor de Ω como 0,7, i.e., serão listados para o usuário todos os conceitos que possuem similaridade maior do que 0,7.

$$\sum_{q=1}^{N} F_q(a,b)^* W_q \geq \Omega$$

Equação 5 – Regra para selecionar conceitos altamente similares

Todos os pesos descritos nessa dissertação foram aplicados arbitrariamente. Contudo, a aplicação feita permite que o usuário altere o valor desses pesos e o valor de Ω, ajustando os melhores valores para as ontologias usadas.

No capítulo 4 apresentaremos uma análise mais detalhada dessa solução.

3.4 – Comparação com outras abordagens existentes

Existem vários trabalhos que tratam de questões semelhantes às discutidas nesta dissertação, como interoperabilidade semântica, chegada de consenso e negociação de significados.

A interoperabilidade de ontologias vem sendo estudada por diferentes pesquisadores. Em (BOUQUET, SERAFINI, ZANOBINI, 2003) é proposto um algoritmo para descoberta de mapeamentos semânticos, cruzando classificações hierárquicas, baseado em uma aproximação para a coordenação semântica, i.e., alinhamento. Esta aproximação traz o problema de coordenação semântica do problema da lingüística computacional ou similaridades estruturais para o problema de dedução de relações entre conjuntos de fórmulas lógicas, que representam o significado dos conceitos dos diferentes modelos. São combinadas as informações do nível do conhecimento léxico (conhecimento sobre as palavras utilizadas), do nível do conhecimento do domínio (conhecimento sobre as relações entre os sentidos das palavras) e do nível do conhecimento estrutural (conhecimento vindo de como as

palavras estão organizadas na árvore de representação) para a construção de uma nova representação do problema. Nesta, o significado de cada nó é codificado como uma fórmula lógica e os conhecimentos relevantes do domínio e das relações estruturais entre os nós são adicionados aos nós como conjuntos de axiomas. Assim, o problema de descoberta da relação semântica entre estes nós passa a ser traduzido não como um problema de mapeamento, mas como um problema de dedução lógica.

Ao contrário do GNoSIS, os autores não utilizam técnicas para comparações sintáticas, o que dota a abordagem de certas vantagens como a facilidade em encontrar relações entre conceitos de ontologias descritas em línguas distintas. Contudo, ao se basear somente em comparações não sintáticas, esta abordagem exige que as ontologias comparadas possuam uma granularidade muito alta (ou seja, uma estrutura necessita conter muitas classes hierarquizadas) para que possa ser mapeada. Isso acontece porque a abordagem utiliza somente dados estruturais, descartando comparações nas propriedades dos conceitos para encontrar relações somente com dados léxicos (acessando dados do WordNet). Além disso, ao se basear fortemente na base WordNet (FELLBAUM, 1998), que descreve somente palavras da língua inglesa, a vantagem inicialmente citada dessa abordagem (a não utilização de dados sintáticos) acaba não sendo explorada. A implementação feita do GNoSIS, por sua vez, não é tão dependente de ontologias com alta granularidade, embora quanto maior a granularidade da ontologia, maior será a relevância do grau de comparação calculado. Além disso, utilizamos a conceitualização da ontologia (dados de propriedades) como forma de cálculo semântico.

Outra diferença quanto à abordagem acima são os tipos de mapeamentos abordados. O GNoSIS, conforme visto na seção 3.2.3.1, implementa oito tipos diferentes de mapeamentos que, i.e., cobre todas as relações semânticas propostas para linguagens de mapeamento de ontologias (BROCKMANS, HAASE, STUCKENSCHMIDT, 2006). Na abordagem acima (BOUQUET, SERAFINI, ZANOBINI, 2003), os autores trabalham com cinco possíveis relações, que são: *equivalência* (quando os conceitos são sinônimos), *contém* (quando o conceito que contém o outro é hipônimo[19] ou merônimo[20] do outro conceito), *está contido* (quando o

[19] Um hipônimo é uma palavra cujo significado é hierarquicamente mais específico que o de outra; por exemplo, cenoura ou nabo estão em relação de hiponímia relativamente a legume.

conceito que está contido é hiperônimo[21] ou holônimo[22] do outro conceito), *compatibilidade* (quando os conceitos possuem alguma relação, mas não são sinônimos) e *disjunção* (quando são antônimos).

Já o serviço Articulation Service (ARTICULATION, 2005) realiza o mapeamento de duas ontologias. A primeira ontologia é a chamada de ontologia de assunto e a segunda de ontologia objeto. O mapeamento é realizado de forma assimétrica da ontologia de assunto para a ontologia objeto. Não existe documentação suficiente no sítio do serviço para avaliarmos as diferenças com a nossa proposta. A principal diferença encontrada está na criação do arquivo de mapeamentos que o Articulation se baseia numa ontologia que descreve o domínio dos relacionamentos semânticos, enquanto a nossa implementação do GNoSIS se utiliza de um formato criado por nós. Contudo, é mister notar que o modelo GNoSIS não define um formato específico para o arquivo de mapeamento, como informado na seção 3.2.3.1, i.e., a utilização de um dado formalismo é sustentado pelo modelo e implementações futuras do modelo pode explorar os formalismos que necessitar.

Para o trabalho com múltiplas e extensas ontologias, é proposto a ferramenta iPROMPT (NOY, MUSEN, 2003). O iPROMPT é uma ferramenta semi-automática de combinação de ontologias. Esta ferramenta guia o usuário, apresentando sugestões para os termos das ontologias que devem ser combinados, e identifica inconsistências e problemas potenciais, sugerindo estratégias para resolvê-los. Seu algoritmo faz uso tanto da informação da estrutura dos conceitos na ontologia e relações entre eles quanto da informação obtida do usuário. No entanto, as informações analisadas entre os conceitos são limitadas ao contexto local, ou seja, apenas são analisadas as informações das relações entre os conceitos ligados diretamente. Nossa implementação do GNoSIS compara todos os conceitos de uma ontologia com todos os conceitos da outra

---

[20] Relação de hierarquia semântica entre duas unidades lexicais; uma denotando a parte (merônimo) e criando uma relação de dependência ao implicar a referência a um todo (holônimo), relativo a essa parte. Exemplo: A unidade lexical "dedo" (merônimo) implica a unidade lexical "mão" (holônimo).

[21] Hiperônimo é sinônimo de superordenado, nome que se dá ao termo cujo sentido inclui aquele (ou aqueles) de um ou de vários outros termos, chamados hipônimos. Assim, animal é hiperônimo de cão, gato, burro, etc.

[22] Relação de hierarquia semântica entre duas unidades lexicais; uma denotando a parte (merônimo) e criando uma relação de dependência ao implicar a referência a um todo (holônimo), relativo a essa parte. Exemplo: carro/volante. Carro estabelece uma relação de holonímia com volante, sem porém lhe impor as suas propriedades.

ontologia, analisando suas propriedades, ou seja, seus relacionamentos com outros conceitos e a diferença de contexto entre os conceitos.

A importância do compartilhamento da informação e a cultura de distribuição do conhecimento têm encorajado alguns pesquisadores a explorar o consenso como um indicador de conhecimento. Os métodos de análise de consenso foram apresentados, pela primeira vez, pelos artigos seminais (BATCHELDER, ROMNEY, 1986; ROMNEY, WELLER, BATCHELDER, 1986; BATCHELDER, ROMNEY, 1988). Para introduzir os fundamentos da análise de consenso, esses artigos também citaram exemplos de sua aplicação na modelagem de conhecimento de informações gerais com estudantes de um colégio norte-americano e a classificação de conceitos sobre doenças com cidadãos da Guatemala. Outras aplicações mais recentes da análise de consenso têm focado na medição da diversidade cultural entre organizações (CAULKINS, HYATT, 1999) e diferentes graus de *expertise* nas organizações e criação de comunidades de prática (RODRIGUES, OLIVEIRA, SOUZA, 2005).

A tecnologia tem suportado os processos de negociação. Um exemplo da aplicação da tecnologia no processo de negociação são os Sistemas de Suporte a Negociação (SSN), considerados um ramo dos Sistemas de Suporte a Decisão em Grupo (FJERMESTAD, HILTZ, 1999) e podem ser definidos como mecanismos computacionais elaborados para apoiar os negociadores durante o processo de negociação facilitando a obtenção de acordos eficientes (LIM, GAN, CHANG, 2002). Os Sistemas de Suporte a Negociação têm sido estudados e pesquisados por mais de duas décadas (KERSTEN, 1985; LOMUSCIO, WOLLDRIDGE, JENNINGS, 2003). Entretanto a adoção desse tipo de sistema nos negócios é mais recente, causado pelo aumento do número de usuários com banda larga, a habilidade de comunicação on-line destes usuários, a inclusão de mídias mais ricas como vídeos e áudio, a evolução dos mecanismos de troca de mensagens e os avanços em áreas como a Inteligência Artificial (YUAN, TUREL, 2004).

Por ser uma subárea ainda muito recente, a negociação de significados possui atualmente poucos trabalhos publicados. Contudo, alguns trabalhos já se mostram relevantes. Em (NOVAK *et al.*, 2004), os autores descrevem uma ferramenta para auxiliar na comunicação entre o gerenciamento autônomo de conhecimento local dentro de comunidades e no compartilhamento, negociação e coordenação de conhecimento entre diferentes comunidades (heterogêneas). Este sistema combina métodos para construção de artefatos que refletem os padrões da linguagem na comunidade,

chamados de Mapas de Linguagem, uma junção de Mapas de Conceito[23] e Mapas de Contexto[24]. Para conciliar os conceitos de vários mapas, é usado uma ferramenta chamada Reconciler, que é usada por um grupo seleto de usuários. Os mapas são projetados para em uma tela para os outros participantes acompanharem o processo e intervirem verbalmente, quando necessário. O maior problema encontrado nesta abordagem, e o que a difere do modelo GNoSIS, é que, mesmo existindo a figura de um mediador, não existe a especificação de um protocolo para a negociação. Conforme foi visto na seção 3.2.2.1, o protocolo da negociação define as regras do processo: quem pode, o que pode e quando pode. Sem uma regulamentação definida, uma negociação eletrônica pode ser não controlada. Além disso, embora os autores utilizem um mediador para recolher as sugestões dos participantes, não estão definidas quais as ações que o mediador poderá tomar e quais técnicas para resolução de conflito ele fará uso.

Recentemente, uma abordagem similar à proposta nesta dissertação foi discutida em (TROJAHN *et al.*, 2006). Neste artigo, os autores descrevem um modelo de negociação para ser usado em ambientes com múltiplos agentes que possuem a capacidade de negociar conceitos de ontologias. Contudo, existem diferenças marcantes entre as duas propostas. A primeira delas está na ausência de intervenção do usuário que os autores sugerem. Assim, os agentes são autônomos nas suas decisões, o que, nesta abordagem, causa mapeamentos errôneos provindos de decisões que poderiam ser questionadas caso tivesse uma intervenção humana. Parte desse problema está também na forma com que a similaridade entre os conceitos é calculada: os autores não se preocupam em utilizar comparações semânticas dos conceitos, somente em comparações sintáticas e léxicas (usando a base WordNet). Diferentemente da proposta do GNoSIS, a abordagem proposta em (TROJAHN *et al.*, 2006) somente é útil em ontologias de mesmo domínio, onde existem um grande número de termos semelhantes e somente algumas disparidades. Por sua vez, o GNoSIS visa facilitar o processo de prover interoperabilidade semântica em ontologias de domínios diferentes mas com alguma sobreposição. Nestas ontologias, embora o número de conceitos a serem

[23] Redes de termos que representam grupos de palavras distintas usadas em contextos similares e os relacionamentos entre eles.

[24] Representam relacionamentos entre conceitos e grupos de documentos que aparecem em contextos similares.

mapeados possa ser menor, as divergências quanto à conceitualização (vide seção 2.5.1.2) são mais contundentes, uma vez que especialistas com conhecimentos de áreas diferentes podem conter visões de mundo bem distintas.

Além das diferenças quanto ao cálculo de similaridade, existem também diferenças marcantes no processo de negociação e na forma de mapeamento. No processo de negociação proposto pelos autores, o protocolo é bem definido, mas existe um equívoco teórico quanto o mecanismo para resolução de conflitos na negociação colaborativa. Os autores sugerem uma votação entre os agentes para escolher qual mapeamento será utilizado quando um impasse se formar. Porém, esta forma de resolução de conflito descaracteriza uma negociação, uma vez que, ao invés de encontrar uma solução satisfatória para todas as partes, votações impõem uma solução que, em muitos casos, não satisfaz a todos os participantes. Ao impor uma solução, os autores eliminam a possibilidade da negociação não ser bem sucedida, ou seja, de um acordo não ser firmado entre as partes. Porém, essa imposição não caracteriza uma negociação bem sucedida, ao contrário, invalida o processo, uma vez que os negociadores que se sentirem lesados não se sentirão motivados a negociar novamente e não irão cumprir o contrato (arbitrariamente) firmado. Quanto ao mapeamento, os autores identificam somente um tipo de relacionamento semântico: equivalência, que indica a existência de alguma similaridade entre os conceitos. Assim, não é possível deduzir qual é a relação semântica existente entre esses conceitos mapeados, o que dificulta a utilização de uma máquina de inferência que possa trabalhar com esses mapeamentos.

Alguns outros trabalhos tratam da negociação do significado para criação de contratos comerciais, evitando desentendimentos entre as partes (BAGBY, MULLEN, 2005; YAN, ZHANG, YAN, 2006). Estes trabalhos que também podem ser categorizados na área de negociação de significados, porém não está focada nos problemas de interoperabilidade semântica que discutimos nesta dissertação.

Em resumo, o GNoSIS é um modelo de negociação que se utiliza de técnicas conhecidas de negociação, usualmente empregadas no cenário de negócios, e de meios tecnológicos para facilitar a argumentação e o mapeamento das estruturas de conhecimento. Nos trabalhos encontrados na literatura, a tentativa de facilitar a interoperabilidade entre estruturas de conhecimento se dá principalmente por (1) uma ferramenta para editar estruturas e encontrar similaridades, porém o trabalho é feito por um único usuário, sem intervenção dos outros especialistas de domínio, ou (2) uma

ferramenta é utilizada para manipulação das ontologias e um grupo de especialistas é responsável pelo produto final, porém nenhuma técnica para facilitar o consenso entre as partes é utilizada ou, por fim, (3) técnicas de negociação são usadas para chegada de consenso dos conceitos, porém a negociação não possui suporte de tecnologias mais avançadas para cálculo de similaridades de conceitos ou para auxiliar a troca de mensagens e argumentos entre as partes.

Capítulo 4 – Avaliação do modelo e das ferramentas

Experimentação é um modo sistemático e controlado para avaliação. Novos métodos, técnicas, linguagens e ferramentas devem ser experimentados para obter uma comparação com os já existentes (TRAVASSOS, GUROV, AMARAL, 2002).

O modelo de negociação de significados GNoSIS, proposto neste trabalho, tem como principal objetivo facilitar o acordo entre equipes multidisciplinares que desejam interoperar seus sistemas através de estruturas de conhecimento distintas. O modelo agrega técnicas de negociação utilizada em ambientes de negócios, técnicas de chegada de consenso e visualização de discurso, além de estudos sobre as estruturas de conhecimento.

Neste capítulo, apresentamos um exemplo de uso do mecanismo de cálculo de similaridade, avaliando o seu comportamento em ontologias com diferentes níveis de semelhança. O algoritmo de cálculo de similaridade é o módulo mais complexo da nossa implementação do GNoSIS e as abordagens de similaridades de estruturas ontológicas encontradas na literatura mostram que esse ainda é um campo não completamente explorado e, assim, permite que novas idéias sejam apresentadas. Por isso, damos essa atenção à este módulo e não a outros com menor relevância. Na segunda metade desse capítulo, apresentamos um estudo de observação feito para avaliar nosso modelo de negociação. A avaliação visa, principalmente, verificar qualitativamente o modelo e reunir os pontos de possíveis melhorias.

4.1 – Avaliação do cálculo de similaridade

Antes de avaliarmos o modelo GNoSIS propriamente dito, iremos verificar, nesta seção, a utilização do mecanismo de cálculo de similaridade implementado. Esse estudo não visa mensurar variáveis como tempo de resposta e número de acertos, muito menos comparar a nossa abordagem com algoritmos já existentes. Nosso objetivo aqui é provar que nossa abordagem é viável para comparar conceitos de ontologias e determinar seu grau de semelhança. É mister atentar, contudo, que o estudo realizado não pode ser considerado exaustivo, uma vez que foi realizado com um número reduzido de ontologias, sem considerar estruturas com diferenças significativas de granularidade.

É necessário ressaltar que o cálculo de similaridade implementado não faz parte do modelo GNoSIS, mas é, sim, uma implementação para o módulo de cálculo de similaridade que é especificado no modelo. Ou seja, o modelo, assim como as metodologias descritas na seção 2.2, não define um método específico para calcular similaridade de conceitos. Ao contrário, o modelo GNoSIS define que uma implementação do módulo de cálculo de similaridade deve ser capaz de utilizar quaisquer algoritmos que o mediador desejar utilizar para refinar o processo. Para fins de avaliação do modelo, implementamos um algoritmo para cálculo de similaridades experimentando uma abordagem levemente distinta das encontradas na literatura, conforme pode ser lido na seção 3.4.

No final deste estudo são apontadas possíveis melhorias para esse cálculo de similaridade.

4.1.1 – Instrumentos e procedimento

Para avaliarmos essa abordagem, iremos realizar três comparações distintas.

Na primeira delas, compararemos um trecho da ontologia que descreve o domínio do paradigma orientado a objetos, disponível no anexo D (item 1), com uma cópia dela que teve os nomes dos conceitos arbitrariamente alterados. Assim, temos o intuito de avaliar que o algoritmo funciona para o caso em que as duas ontologias possuem estruturas idênticas e nomes diferentes.

A segunda comparação será feita alterando-se a estrutura da segunda ontologia e com nomes de conceitos quase idênticos.

Por fim, a terceira comparação será feita alterando-se novamente os nomes da segunda ontologia, de forma arbitrária, representando assim diferenças grandes de conceitualização entre as duas ontologias. Este último é o cenário de pior caso em que o algoritmo poderá agir.

4.1.2 – Execução

No nosso primeiro cenário, iremos comparar duas ontologias que possuem a mesma estrutura (propriedades e hierarquia), contendo somente diferenças entre os nomes dos conceitos que foram arbitrariamente alterados. O trecho a ser comparado é ilustrado na Figura 27.

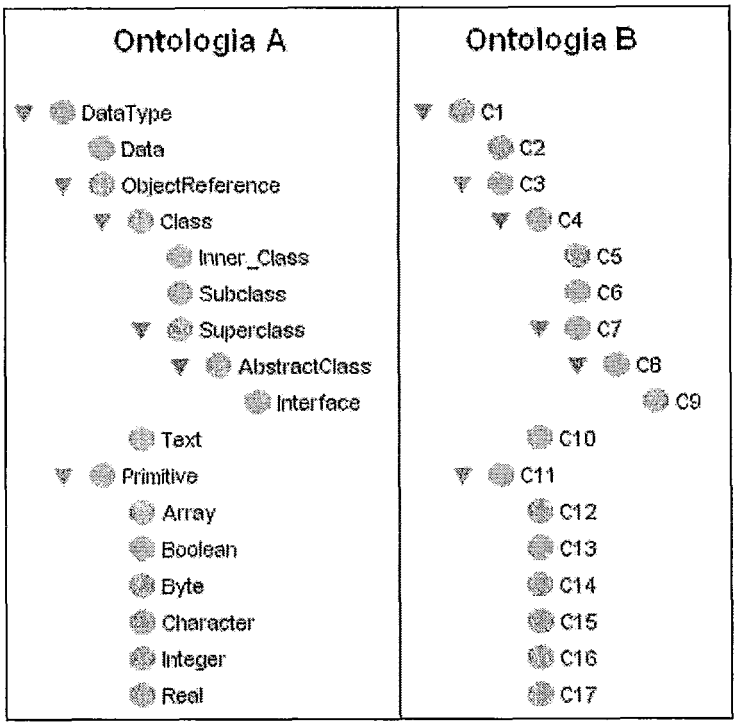


Figura 27 – Ontologias com termos diferentes e estruturas iguais

A figura acima exibe a estrutura hierárquica dos conceitos que serão analisados pelo estudo. Como toda representação gráfica de ontologias, existe perdas semânticas nesta visualização, uma vez que não conseguimos visualizar outros aspectos que não sejam os hierárquicos, como, por exemplo, as propriedades de dados e de objetos desses conceitos. Para visualizar todas as informações das ontologias, disponibilizamos os arquivos OWL no anexo D dessa dissertação, na íntegra.

O primeiro passo para utilização do módulo de cálculo de similaridade é carregar os arquivos OWL (comando "load"), como mostra a Tabela 3. O módulo trabalha com comandos em *prompt*; para informações sobre instalação e utilização do módulo de cálculo de similaridade, leia o Anexo F.

Ao carregar as ontologias, o módulo carrega também os pesos-padrão que serão utilizados nas funções de semelhança, conforme descrito na seção 3.3.6. Os pesos padrões considerados neste estudo são os mesmos descritos na Figura 26.

Tabela 3 – Carregando o algoritmo com as duas ontologias

```
>>>> GNoSIS :: X-Ray Matcher v1.5 beta (by JairoSouza) <<<

Press h for help...

Command: load
First ontology name: comparacao_ontologia1.owl
Second ontology name: comparacao_ontologia2.owl
>>>>Loading ontologies...
[ProtegeOWLParser] Completed triple loading after 922 ms
[TripleChangePostProcessor] Completed lists after 0 ms
[TripleChangePostProcessor] Completed anonymous classes after 31 ms
[TripleChangePostProcessor] Completed deprecated classes after 0 ms
[TripleChangePostProcessor] Completed properties after 16 ms
[TripleChangePostProcessor] Completed named classes after 15 ms
... Loading completed after 1047 ms
[ProtegeOWLParser] Completed triple loading after 391 ms
[TripleChangePostProcessor] Completed lists after 0 ms
[TripleChangePostProcessor] Completed anonymous classes after 0 ms
[TripleChangePostProcessor] Completed deprecated classes after 0 ms
[TripleChangePostProcessor] Completed properties after 15 ms
[TripleChangePostProcessor] Completed named classes after 0 ms
... Loading completed after 469 ms

Command:
```

Após a carga das ontologias, a análise poderá ser feita (comando "analise"), onde todas as funções de semelhança serão calculadas para gerar o grau de similaridade total dos conceitos. A janela abaixo (Figura 28) exibe a lista de todos os conceitos com maiores graus de similaridade calculado para o conceito ObjectReference, acima do limite mínimo de 0,65 (campo *threshold*) e ordenado por grau de similaridade. O conceito com maior grau de similaridade encontrado foi C3, com valor aproximado de 0,805. Ao analisarmos a Figura 27, verificamos que esse conceito corresponde realmente ao conceito ObjectReference. Os outros conceitos listados na lista dos mais similares a ObjectReference são C1 e C11, com valores aproximados de 0,70 e 0,67, respectivamente. Conforme pode ser verificado na Figura 27, C1 é o correspondente ao conceito DataType (pai de ObjectReference) e C11 ao conceito Primitive (irmão de ObjectReference), ou seja, conceitos próximos na sua estrutura (tanto hierárquicas quanto nas propriedades herdadas).

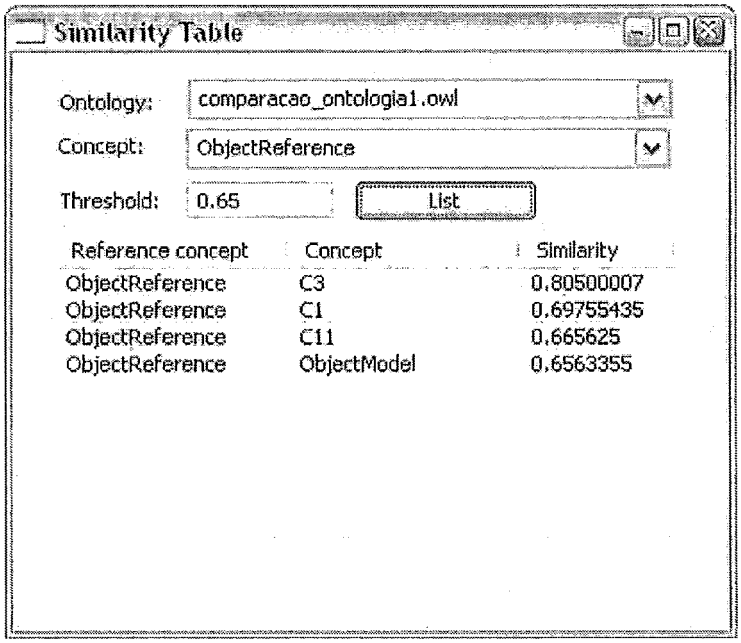


Figura 28 – Conceitos mais similares de ontologiaA#ObjectReference

A Tabela 4, descreve os resultados do algoritmo com os valores encontrados para cada dupla de conceitos. A tabela foi montada com cada conceito da ontologia A e com o conceito mais similar apontado pelo algoritmo. Nota-se que ao recuperarmos os conceitos mais similares a cada conceito da primeira ontologia, o algoritmo acerta em todos os casos.

Tabela 4 – Grau de similaridade entre os conceitos mais similares no primeiro cenário

Ontologia A	Ontologia B	Grau de similaridade
Datatype	C1	0,85
Data	C2	0,85
ObjectReference	C3	0,805
Class	C4	0,711
Inner_Class	C5	0,869
Subclass	C6	0,823
Superclass	C7	0,864
AbstractClass	C8	0,761
Interface	C9	0,871
Text	C10	0,85
Primitive	C11	0,801
Array	C12	0,85
Boolean	C13	0,85
Byte	C14	0,85
Character	C15	0,85
Integer	C16	0,85
Real	C17	0,85

É possível discriminarmos os valores das principais funções de semelhanças aplicadas (comando "showAllCellSims"). A Tabela 5 mostra os valores calculados para as funções de semelhanças aplicados aos conceitos ObjectReference e C3. Nota-se que a similaridade entre os dois conceitos não foi total (valor 1) por conta da sua diferença entre os termos (conceptName = 0). Tal diferença acaba por influenciar, em certo grau, as funções de semelhança aplicadas sobre a hierarquia (HierarchySim), sobre dados não-hierárquicos (ContextFreeVariableSim) e, é claro, o grau de similaridade total (ConceptSim). Por outro lado, estes valores de semelhança possuem valores superiores ao considerado aceitável (limite inferior de 0,65) por causa da alta semelhança nas suas estruturas, o que pode ser notado de forma mais clara no valor 1 retornado pela função de semelhança aplicada às propriedades dos conceitos (PropertySim).

Tabela 5 – Similaridades de [ObjectReference, C3] separadas por função de semelhança

```
Command: h
-> Analise: proccess sintatic and semantic analysis.
-> Exit: terminate the analysis.
-> Help: show help.
-> List: list the concepts most similars to a chosen concept.
-> Load: load ontologies for semantic analysis.
-> Quit: terminate the analysis.
-> Repository: show ontologies saved in repository.
-> ShowAllCellSims: show all similarities from two chosen concepts.
-> ShowConceptSims: show all concepts similarities.
-> ShowCFVSims: show context free variables similarities.
-> ShowPropertySims: show properties similarities.
-> ShowStructure: show tree structure of a chosen ontology.
-> LaunchTable: shows similarity table


Command: ShowAllCellSims
First concept name: ObjectReference
Second concept name: C3

+++Showing similarities
PropertySim: 1.0
ConceptName: 0.0
ContextFreeVariableSim: 0.8
HierarchySim: 0.81000006
ConceptSim: 0.80500007

Command:
```

Este caso de diferença clara entre os nomes das duas ontologias pode ser resolvido ao alterarmos o valor dos pesos aplicados pelo algoritmo nas funções de semelhança. Ao aplicarmos o valor 0 (zero) para as funções que se utilizam do cálculo da distância de edição (seção 3.3.6), como as funções de semelhança aplicadas ao nome dos conceitos e ao nome das propriedades, teremos, neste caso, valores mais reais retornados pelo módulo. A Tabela 6 exibe os dados atualizados, recuperados ao aplicarmos tal alteração nos pesos das funções de semelhança.

Tabela 6 – Grau de similaridade após anular funções de semelhança de nomes

Ontologia A	Ontologia B	Grau de similaridade
Datatype	C1	1,00
Data	C2	1,00
ObjectReference	C3	1,00
Class	C4	1,00
Inner_Class	C5	1,00
Subclass	C6	1,00
Superclass	C7	1,00
AbstractClass	C8	1,00
Interface	C9	1,00
Text	C10	1,00
Primitive	C11	1,00
Array	C12	1,00
Boolean	C13	1,00
Byte	C14	1,00
Character	C15	1,00
Integer	C16	1,00
Real	C17	1,00

No segundo cenário, utilizaremos a mesma ontologia A do exemplo anterior e uma segunda ontologia, baseada na ontologia A, que chamaremos de ontologia C. Esta último ontologia possui seus nomes quase idênticos aos da ontologia A e sua estrutura foi levemente alterada ao modificarmos arbitrariamente algumas propriedades. A Figura 29 exibe as duas ontologias lado a lado, explicitando sua hierarquia (inalterada) e as pequenas alterações nos nomes dos conceitos. A figura, porém, omite a principal diferença entre essas ontologias: suas propriedades. Tais diferenças podem ser verificadas no Anexo D (item 3), onde a ontologia encontra-se descrita.

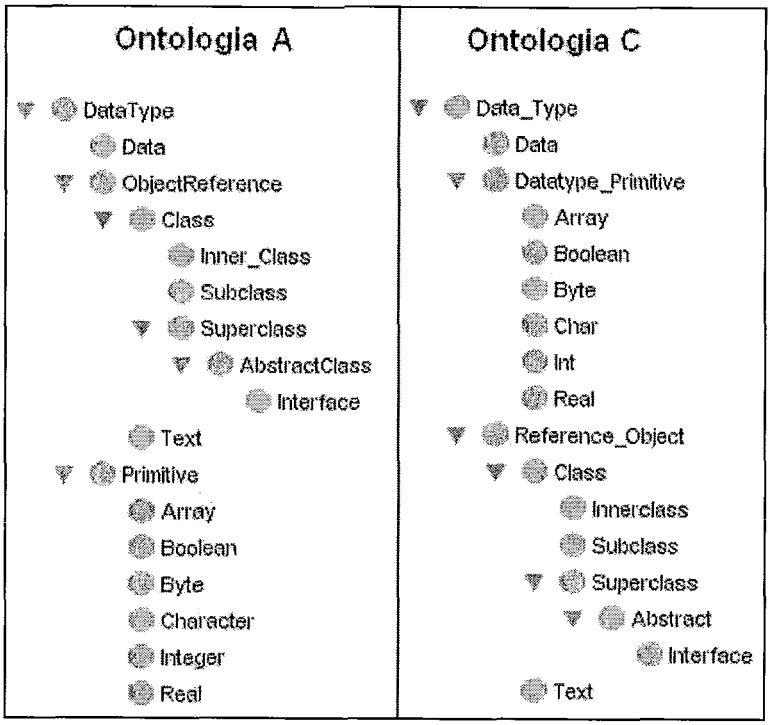


Figura 29 – Ontologias com nomes levemente alterados

Para uma visão mais clara das diferenças entre as duas ontologias, analisemos a Figura 30. A figura ilustra as propriedades entre os conceitos "Class" presentes nas duas ontologias. Considerando somente as linhas que ligam o conceito "Class" aos outros conceitos da ontologia, verificamos que existem diferenças consideráveis entre as duas estruturas. Na ontologia A, o conceito "Class" possui três propriedades: instancia, implementa e contém. A propriedade "instancia" se relaciona com o conceito "Object", a propriedade "implementa" se relaciona com o conceito "Inteface" e a propriedade "contém" se relaciona com os conceitos "Comment", "Field", "Method", "Datatype", "Command", "Heritage", "Declaration" e "Package". Na ontologia C, por sua vez, o conceito "Class" possui somente duas propriedades: implementa e contém. Assim como na ontologia A, a propriedade "implementa" da ontologia C se relaciona com o conceito "Interface". Porém, a propriedade "contém" se relaciona somente com conceitos "Declaration", "Package", "Field" e "Data_Type".

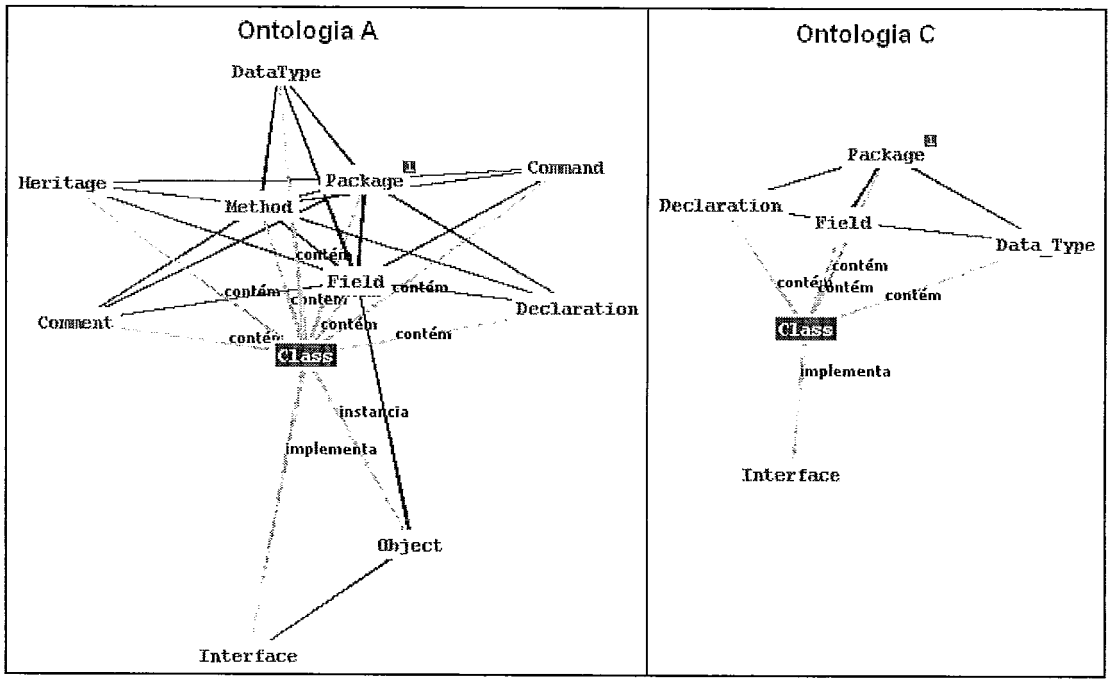


Figura 30 – Diferença nas propriedades do conceito "Class" das duas ontologias

A Tabela 7 representa os maiores valores obtidos após o cálculo de similaridade para cada dupla de conceitos, considerando-se os pesos-padrão originais (Figura 26). Nota-se que, em algumas duplas, os valores calculados são menores que os da Tabela 4, mas, ainda assim, o algoritmo acerta em todos os casos.

Tabela 7 – Grau de similaridade entre os conceitos mais similares no segundo cenário

Ontologia A	Ontologia C	Grau de similaridade
Datatype	Data_type	0,782
Data	Data	0,747
ObjectReference	Object_Reference	0,784
Class	Class	0,651
Inner_Class	InnerClass	0,61
Subclass	Subclass	0,664
Superclass	Superclass	0,66
AbstractClass	Abstract	0,602
Interface	Interface	0,631
Text	Text	0,85
Primitive	Datatype_Primitive	0,805
Array	Array	0,773
Boolean	Boolean	0,776
Byte	Byte	0,773
Character	Char	0,63
Integer	Int	0,63
Real	Real	0,773

A diferença entre os nomes dos conceitos fez com que o grau de similaridade calculados fosse menor nestes conceitos do que nos que possuem nomes iguais. Contudo, o determinante para os baixos graus de similaridade encontrados foram os valores retornados pelas funções de semelhança que levam em consideração a estrutura hierárquica e as propriedades dos conceitos, tendo essa última a pior atuação. A Tabela 8 exibe os valores retornados de algumas funções de semelhança ao comparar os conceitos Class das duas ontologias. Nesta tabela, o valor de semelhança entre as propriedades dos conceitos (PropertySim) se mostrou consideravelmente baixa (0,2). Conforme foi explicitado com mais detalhes na seção 3.3.6, a esta função de similaridade leva em consideração, entre outras coisas, o nome dos conceitos que se relacionam através da propriedade e aplica uma penalidade nos casos em que um conceito possui número de propriedades diferentes do conceito comparado (Equação 4). A penalidade padrão aplicada neste estudo foi de 0,3, vide Anexo F.

Tabela 8 – Similaridades de [Class, Class] separadas por funções de semelhança

```
Command: showAllCellSims
First concept name: Class
Second concept name: Class

+++Showing similarities
PropertySim: 0.2
ConceptName: 1.0
ContextFreeVariableSim: 0.65701
HierarchySim: 0.6284756
ConceptSim: 0.6512378

Command:
```

Sabendo desse impacto causado pela aplicação das funções de semelhança de propriedade aplicadas neste cenário, iremos alterar o peso aplicado a essas funções para zero. Desta forma, a diferença entre as propriedades dos conceitos não será mais determinante para o grau de similaridade total dos conceitos. A Tabela 9 contém os graus de similaridade encontrados ao alterar o peso desta função de semelhança. Nota-se que os valores encontrados são muito próximos dos contidos na Tabela 6, conforme esperado. Tal diferença de valores se dá pela pequena diferença de nomes de conceitos que ainda persiste nas duas ontologias.

Tabela 9 – Graus de similaridade após anular funções de semelhança de propriedades

Ontologia A	Ontologia C	Grau de similaridade
Datatype	Data_type	0,982
Data	Data	0,991
ObjectReference	Object_Reference	0,982
Class	Class	0,991
Inner_Class	InnerClass	0,982
Subclass	Subclass	0,991
Superclass	Superclass	0,991
AbstractClass	Abstract	0,963
Interface	Interface	0,977
Text	Text	0,991
Primitive	Datatype_Primitive	0,918
Array	Array	0,977
Boolean	Boolean	0,975
Byte	Byte	0,977
Character	Char	0,946
Integer	Int	0,957
Real	Real	0,977

Para o terceiro cenário deste estudo, foi feita uma comparação entre a ontologia A e uma ontologia D. A ontologia D foi obtida através de variações mais drásticas feitas na ontologia C, onde seus termos foram bastante alterados. Esse é o caso mais difícil em que o algoritmo para cálculo de similaridades tem que trabalhar. Além das diferenças de nomes dos conceitos, a ontologia D difere também da ontologia A quanto à sua estrutura (contém diferenças nas propriedades e, desta vez também, na árvore hierárquica). Este cenário representa duas ontologias com alta divergência de conceitualização (vide seção 2.5.1.2), i.e., diferenças nas suas visões de mundo.

A Figura 31 exibe as duas ontologias que serão comparadas. Nota-se que as diferenças quanto ao nome dos conceitos e a diferença na hierarquia (conceito "Interface" não é mais filho de "AbstractClass" como era na ontologia A, etc). A figura novamente omite as diferenças quanto às propriedades dos conceitos. Contudo, essas diferenças foram aplicadas da mesma forma arbitrária que no segundo cenário.

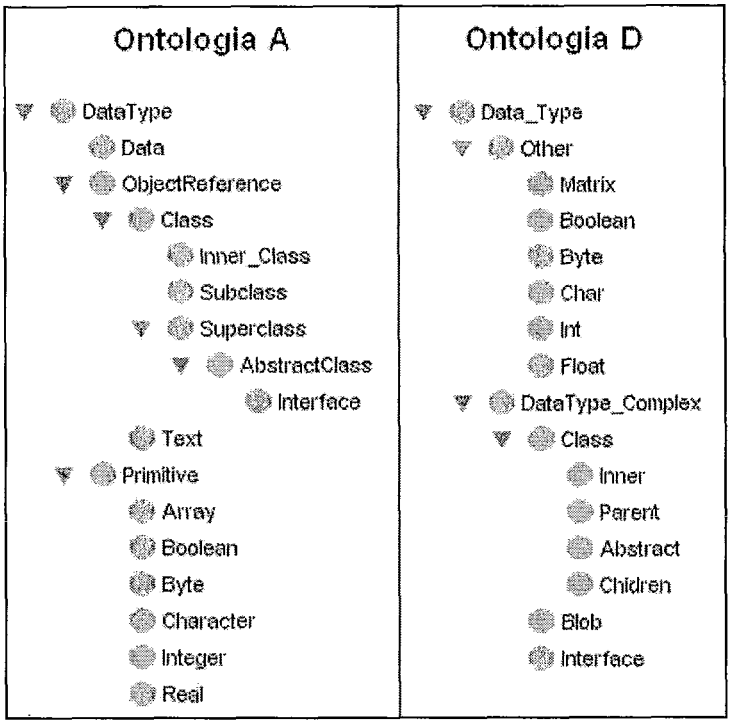


Figura 31 – Ontologias com diferenças tanto nos nomes dos conceitos quanto na estrutura

A Tabela 10 descreve o grau de similaridade de cada conceito, calculados com as ontologias A e D e os pesos-padrões. Desta vez, o ajuste dos pesos não é uma tarefa simplória, uma vez que existem divergências tanto sintáticas quanto semânticas. A maioria dos conceitos mais similares não ultrapassou o limite inferior considerado razoável (*threshold* = 0,6). Contudo, reduzindo-se o limite inferior para 0,3, verificamos que o algoritmo acertou na maioria dos casos, embora tenha retornado duplas indesejadas em alguns conceitos. Os graus de similaridade destacados com um número entre parênteses denota qual a posição que o conceito correto apareceu. Por exemplo, [Datatype, Data_type, 0,47 (2)] quer dizer que o grau de similaridade entre os conceitos corretos "Datatype" e "Data_type" pareceu na segunda posição, ou seja, o algoritmo errou ao encontrar erroneamente um conceito mais similar.

Tabela 10 – Graus de similaridade entre os conceitos mais similares do terceiro cenário

Ontologia A	Ontologia D	Grau de similaridade
Datatype	Data_type	0,924
ObjectReference	DataType_Complex	0,768
Class	Class	0,320
Inner_Class	Inner	0,345 (2)
Subclass	Children	0,569
Superclass	Parent	0,562
AbstractClass	Abstract	0,651
Interface	Interface	0,353 (3)
Text	Blob	0,856 (3)
Primitive	Other	0,879
Array	Matrix	0,472
Boolean	Boolean	0,651
Byte	Byte	0,802
Character	Char	0,683
Integer	Int	0,699 (2)
Real	Float	0,576 (5)

Dos 16 conceitos separados para análise no último cenário, o algoritmo acertou em 11 casos. Nos outros casos deste cenário, o algoritmo colocou o conceito esperado em até quinto lugar na lista dos mais similares. Os conceitos que não foram devidamente encontrados pelo algoritmo correspondem aos que possuem maior divergência quanto a sua conceitualização. Tais conceitos são mesmo muito complexos de identificar e nestes casos, mais comuns conforme mais distintas forem as áreas de atuação dos especialistas que montaram as ontologias, a intervenção humana se torna imprescindível.

4.1.3 – Avaliação

Para os casos testados acima, o algoritmo teve um resultado considerado satisfatório. Para os casos em que o resultado não é o esperado, necessita-se de um passo anterior para escolher os melhores pesos a serem aplicados. Tal diferença acontece, principalmente, quando as ontologias diferem muito na sua conceitualização, i.e., na forma com que os conceitos foram estruturados, na quantidade de propriedades que cada conceito possui e na diferença de granularidade entre suas árvores hierárquicas.

Consideramos os bons resultados obtidos nos experimentos a dois aspectos principais: às funções de semelhança aplicadas em artefatos ontológicos separados e,

como conseqüência, aos pesos aplicados a essas funções. Aplicando-se funções de semelhança em cada artefato que descreve um conceito, evitamos que a falta de uma evidência em um conceito de ontologia prejudique a comparação com um conceito de outra ontologia. Ou seja, ontologias podem possuir diferenças muito alta em sua estrutura hierárquica, mas tal diferença pode ser compensada com uma semelhança considerável nas propriedades dos conceitos ou em seus termos.

O ajuste dos pesos utilizados para o cálculo não é uma tarefa trivial e necessita, basicamente, de algumas tentativas para que se encontrem os pesos que geram os melhores resultados para as ontologias a serem comparadas. Tal ajuste pode ser feito utilizando-se também de consultas à base histórica, recolhendo pesos que foram usados em ontologias com estruturas semelhantes ou calibrando os pesos conforme os graus de similaridade encontrados vão sendo condizentes com o mapeamento final escolhido pela mesa de negociação.

O algoritmo acima não faz uso de todos os artefatos ontológicos passíveis de análise, mas pode ser facilmente estendido para considerar instâncias de ontologias, por exemplo. Contudo, é importante relembrar a discussão feita na seção 3.3.6, i.e., ontologias são modelos de conhecimento e não necessariamente estão populadas com instâncias. Todavia, uma vez que é conhecida a forma de acesso às instâncias (podendo estas estar na forma de tuplas em banco de dados, anotações em documentos locais ou remotos, ou, é claro, construtores dentro do próprio arquivo OWL do modelo ontológico, entre outras formas), o seu uso pode somar no cálculo de similaridade principalmente nos casos em que os artefatos que utilizamos não se mostrem suficientes.

4.2 – Estudo de observação do processo de negociação

A seção anterior consiste em um estudo de viabilidade, realizado com o intuito de prover uma avaliação inicial sobre a principal ferramenta implementada para o modelo GNoSIS. Após o estudo de viabilidade do cálculo de similaridade, um estudo de observação foi realizado para coletar avaliações de outras pessoas quanto ao modelo de negociação. O plano desse experimento seguiu o modelo definido por (BARROS, WERNER, TRAVASSOS, 2005).

A realização de um estudo experimental geralmente pode ser dividida em cinco fases: a definição, o planejamento, a execução, a análise e o empacotamento do estudo. A definição do estudo consiste em resumir seus objetivos, seu foco de qualidade e os

objetos que serão analisados. O planejamento envolve a descrição do perfil dos participantes, dos instrumentos, do processo de execução e uma avaliação crítica dos problemas que podem ser encontrados ao longo desta execução. A execução consiste na realização do estudo experimental pelos participantes, utilizando os instrumentos e o processo definidos no planejamento. A análise consiste na organização dos resultados gerados pelos participantes durante a execução e a realização de inferências sobre estes resultados. Finalmente, o empacotamento consiste na organização e armazenamento dos documentos construídos nas etapas anteriores, com o intuito de facilitar a repetição do estudo experimental no futuro (BARROS, WERNER, TRAVASSOS, 2005).

Nesta seção, apresentamos um estudo de observação que avalia a viabilidade da utilização do modelo GNoSIS. Inicialmente, definimos as etapas envolvidas no estudo e apresentamos como estas etapas foram realizadas. Em seguida, apresentamos as observações obtidas durante o estudo.

4.2.1 – Definição

O contexto global considerado nesse trabalho é que as abordagens de mapeamento de ontologias até o momento apresentam cálculos para reconhecer conceitos com grau de similaridade, mas não consideram o lado social que é atingido por esse processo: a participação dos grupos que serão atingidos pelo processo e a discussão humana que aprova ou recusa certos mapeamentos.

O contexto local desse estudo tem como objetivo avaliar a viabilidade da utilização do modelo de negociação de significados no mapeamento de ontologias, focando nas dificuldades encontradas pelos usuários durante o processo de negociação e a utilização da implementação, além da aderência aos objetivos do modelo (o suporte para encontrar um arquivo e a geração do mapeamento consensual). O estudo foi desenvolvido tendo em vista a continuidade do desenvolvimento de pesquisas relacionadas com esta abordagem.

Neste estudo, não estamos diretamente interessados em mensurar o ganho de tempo ou número de negociações bem sucedidas derivado da utilização do modelo. Mas sim, consideramos que melhorias podem ser identificadas através de análises e estudos futuros, além de mostrar que é viável a utilização do modelo.

Seguindo a notação *Goal-Question-Metric* (GQM) (SOLINGEN, BERGHOUT, 1999), a definição do estudo é:

Analisar *a utilização do modelo GNoSIS durante a negociação de significados*

Com o propósito de *recolher requisitos de melhorias e avaliar a viabilidade de negociação e chegada de consenso em significado*

Referente *aos ganhos obtidos por sua utilização e as dificuldades encontradas*

No contexto de *mapeamento de ontologias*

4.2.2 – Participantes

Por se tratar de um ambiente de negociação eletrônica e com o objetivo de não limitar a realização dos experimentos, os participantes podem estar localizados em locais diferentes e deverão instalar em suas máquinas a implementação do GNoSIS e possuir uma conexão Web estável para se conectar ao banco de dados da aplicação. Deste modo, é necessário que os participantes tenham conhecimentos básicos sobre utilização do computador e uma mínima experiência em acesso à WEB.

Além disso, é necessário que os participantes conheçam o processo de negociação a ser utilizado. Caso contrário pode ser necessário que alguns aspectos do processo sejam esclarecidos para que esses participantes usem de forma apropriada as funcionalidades do sistema.

O estudo de observação contou com a participação de profissionais da área de computação (doutores e mestres) e alunos de mestrado e doutorado do PESC (Programa de Engenharia de Sistemas e Computação). A idéia é manter certa heterogeneidade entre os participantes da amostra em questão.

Como apontado na literatura (NIELSEN, 1994), estudantes podem não ser representativos o bastante como atores reais. Contudo, em (HÖST, REGNELL, WOHLIN, 2000), os autores não observam diferenças significantes entre estudantes e atores reais para tarefas de pequeno julgamento. De acordo com Tichy (TICHY, 2001), usar estudantes como participantes de estudos de avaliação é aceitável caso eles sejam apropriadamente treinados e os dados usados para o experimento sejam direcionados para que possam realizar tarefas específicas. Estas condições foram seguidas nesse trabalho.

O treinamento dos participantes que utilizaram a implementação do modelo GNoSIS foi dividido em duas sessões. O treinamento foi dividido em duas aulas que puderam ser interrompidas a qualquer momento pelos participantes para perguntas. Na primeira aula, realizamos uma exposição sobre a teoria de ontologias, utilizando transparências que também foram distribuídas para os participantes. Estas transparências se encontram no Anexo C. Na segunda aula, apresentamos o modelo de

negociação e a implementação do GNoSIS para que os participantes pudessem se familiar com o processo que propomos.

4.2.3 – Instrumentos

Para que o processo de avaliação fosse adequadamente realizado, disponibilizamos cinco computadores em um laboratório, todos com a implementação do GNoSIS devidamente instalada e uma única máquina com o servidor do banco de dados, a qual possui a função de armazenar os dados das negociações e ser a responsável pela sincronia da interface de argumentação, conforme foi detalhado na seção 3.3.4.

Selecionamos ontologias compatíveis com o domínio dos participantes para que os conceitos fossem familiares aos negociadores, permitindo que a discussão gerada durante a negociação fosse legítima e com argumentações consistentes. As duas ontologias (vide Anexo D, itens 1 e 2) foram criadas para apoiar os trabalhos (PEREIRA *et al.*, 2006; VARELLA, 2007) e posteriormente estendidas para este estudo. Elas descrevem (a) o domínio do paradigma de programação orientada a objetos e (b) o domínio de linguagens de programação, respectivamente. A sobreposição entre as duas ontologias se dá nos conceitos que descrevem linguagens de programação que se utilizam de conceitos de orientação de objetos.

O foco de qualidade do estudo exige critérios que avaliem os ganhos obtidos e as dificuldades encontradas na utilização da linguagem pelos usuários. Tanto os ganhos quanto as dificuldades foram avaliados qualitativamente, através dos questionários. Esta análise tem o objetivo de avaliar a dificuldade de uso do modelo e identificar melhorias para o modelo e sua implementação para estudos futuros.

Nesta abordagem, propõe-se que o GNoSIS seja avaliado qualitativamente pelos usuários a partir de questionários. Antes do início das simulações, os participantes são avaliados por um questionário (vide Anexo A). Após o encerramento da negociação, é solicitado que os negociadores respondam o questionário elaborado para avaliação do modelo e da implementação (vide Anexo B). As questões foram definidas baseadas nas abordagens de avaliação qualitativa realizada em outros trabalhos (KERSTEN, NORONHA, 1999b; SCHOOP, LIST, 2003; PAULA, 2006; SMARTSETTLE, 2006) e considerando as características específicas do modelo proposto.

4.2.4 – Procedimentos

Os participantes foram treinados com apresentações sobre ontologias e explicações orais sobre o modelo GNoSIS e as ferramentas implementadas. Foi pedido para que o grupo elegesse o mediador da negociação, que foi a pessoa com mais experiência em negociação, independente do seu conhecimento do domínio, para que seja a responsável por conduzir as rodadas de negociação e fiscalizar o cumprimento do protocolo firmado na próxima etapa.

Após a escolha do mediador, os demais participantes se dividiram em dois grupos representando equipes com interesses cada um em uma ontologia. Assim, cada grupo teve que aprender sobre a sua ontologia e entender os conceitos que por acaso não possuíam conhecimento, se utilizando de pesquisas na Web ou livros sobre o assunto, disponibilizados no laboratório.

Como todos os participantes utilizaram a implementação ao mesmo tempo, a negociação se tornou síncrona e, assim, o protocolo firmado possui limites de tempo reduzidos. Foi definido na agenda da negociação:

- Tempo máximo para cada rodada: Cada participante possui 10 minutos para expor sua argumentação seguindo o modelo IBIS, o qual foi previamente detalhado para todos os participantes.
- Ordem para argumentação/oferta: O mediador adotou um critério próprio para escolha da ordem dos negociadores, se preocupando em alternar um membro de cada grupo, ou seja, não permitindo que membros do mesmo grupo argumentem consecutivamente.
- Tempo máximo permitido para a negociação: A negociação termina ao exceder 2 horas de trabalho.

Conforme foi dito, os participantes se comunicavam através da implementação utilizada do modelo IBIS e seguindo as regras de envio de mensagens expostas na seção 3.2.2.1. Quando foi necessário enviar mensagens do tipo "conversação", foi utilizado um serviço Jabber e as mensagens foram armazenadas localmente. Ao fim do processo, o mediador disponibilizou o arquivo de mapeamento final, gerado a partir das decisões do grupo. O arquivo pode ser consultado no Anexo E.

4.2.5 – Resultados

A amostra contou com alunos de mestrado e doutorado, além de profissionais da área de ciência da computação (mestres e doutores). A principal área de atuação dos participantes é Banco de Dados (71,4%), seguidas de Redes e Inteligência Artificial, representando 14,3% cada área. Somente 28,6% dos participantes já tiveram experiência com ontologias e 50% destes tiveram contato com sistemas de negociação eletrônica. Quanto à experiência com negociação, 57,1% possuem conhecimento sobre técnicas de negociação por estudarem o assunto ou já participaram de negociação controlada num ambiente empresarial. Todos os participantes conhecem o domínio que disponibilizamos para esse experimento (linguagens de programação e orientação a objetos). Porém, ainda assim, foi disponibilizado um tempo adicional para os participantes pesquisar sobre algum conceito em sites de buscas.

Alguns resultados obtidos estão destacados na Tabela 11, outros serão discutidos ao longo dessa seção. Algumas soluções para os problemas encontrados serão discutidas no capítulo 5.

No que se refere à facilidade de utilização do modelo GNoSIS, basicamente, os participantes não apresentaram muitas dificuldades. Observou-se que as principais dificuldades encontradas estão relacionadas com o conhecimento do tempo que resta ao participante concluir o seu turno de argumentação, uma vez que não há nenhum mecanismo que funcione como cronômetro na mesa de negociação, ficando a cargo do mediador avisar a cada participante do tempo restante. Assim, alguns participantes se sentiram insatisfeitos quando o mediador avisava do término do seu tempo faltando pouco menos de um minuto para finalizar.

Em relação ao modelo de negociação, tendo em vista o domínio semântico negociado e o protocolo firmado no início do processo, além do treinamento realizado, a maioria dos participantes considerou o modelo como de extrema utilidade, apontando que o GNoSIS é uma alternativa viável para tentativa de efetivação de um acordo entre as relações semânticas necessárias para o mapeamento.

Tabela 11 – Resultado do questionário para avaliação do modelo e das ferramentas

Consigo expressar meus argumentos eficientemente e entender os argumentos dos outros participantes através do método IBIS utilizado na negociação?				
Extremamente Difícil			**28,6%**	Extremamente Fácil **71,4%**
As ferramentas implementadas permitem a negociação com os demais colegas, mesmo que de forma assíncrona?				
Não		Razoável **14,3%**		Sim **85,7%**
Os objetivos da negociação (mapeamento) foram atingidos?				
Não		Razoável		Sim **100%**
Estou totalmente satisfeito com a tarefa?				
Extremamente insatisfeito		**14,3%**	**14,3%**	Extremamente satisfeito **71,4%**
Utilizaria o modelo para negociar mapeamentos na prática?				
Não		Possivelmente **28,6%**		Sim **71,4%**
O quão fácil ou difícil você achou usar o GNoSIS?				
Extremamente Difícil		**14,3%**	**28,6%**	Extremamente Fácil **57,1%**
Sua experiência de trabalho em grupo de negociação poderia ser considerada				
Péssima	Ruim	Razoável **14,3%**	Boa **14,3%**	Otima **71,4%**
Durante a negociação, você e sua contraparte fizeram algum tipo de contato fora do sistema?				
Sim **0%**			Não **100%**	

Foi possível perceber que a facilidade proporcionada pela socialização do conhecimento (realizada após explicitação e consulta das argumentações de negociações passadas) trata-se de um recurso considerado importante pelos negociadores. Esta constatação foi observada a partir da análise dos questionários, onde a maioria dos participantes destacou essa funcionalidade como uma das vantagens do modelo. Em contrapartida, foram destacadas funcionalidades que faltam ao modelo como a possibilidade de uma pré-negociação mais formal do protocolo previamente sugerido pelo mediador, além de uma falta de informações sobre o mediador escolhido, se ele foi escolhido por votação, suas experiências, etc. Por fim, a falta de informação sobre o perfil dos outros negociadores também foi apontada como uma falha na mesa de negociação, uma vez que esta é colaborativa e esses dados não necessitam ser sigilosos.

Todos os participantes informaram não possuir experiência na utilização de sistemas de suporte à negociação, embora a maioria dos participantes (57,1%) possui ou

algum conhecimento sobre o processo de negociação a partir da análise de alguns trabalhos da literatura ou experiência prática com negociação por já terem participado alguma vez de mesas de negociação comercialmente. Todos os participantes informaram não terem feito nenhum tipo de contato com as contrapartes, a não ser os previstos pelo mediador, a maioria declarou estar satisfeito com a experiência de negociação inclusive se mostrando interessados em utilizar o modelo para negociar conceitos com outros grupos na prática.

Capítulo 5 – Conclusões

Este estudo buscou investigar os problemas encontrados para prover interoperabilidade semântica entre estruturas de conhecimento e as dificuldades de grupos de usuários em chegar a um acordo sobre o significado de seus conceitos. O problema da interoperabilidade semântica e do trabalho cooperativo é explorado desde seu estudo até a definição de uma modelo de negociação para sua resolução parcial.

O modelo proposto, intitulado de GNoSIS, foi baseado em técnicas de negociação usualmente utilizadas em negociações comerciais e adaptadas para o contexto da negociação de significados. O modelo visa à cooperação entre as equipes para que o resultado da negociação possa ser considerado uma vitória de cada participante e, assim, o compromisso firmado ao fim do processo seja cumprido pelas partes. Foi adicionado ao modelo um ator com funções de mediação e objetivo de fiscalização das regras aceitas para a negociação, de facilitar a comunicação entre as partes e perseguir um consenso entre elas, de forma que o consenso seja firmado por escolha das partes e sem qualquer imposição de membros, o que faria com que, na pós-negociação, o compromisso não fosse perseguido por alguns membros.

O GNoSIS prevê a utilização de métodos computacionais para calcular semelhanças entre estruturas de conhecimento, define a forma de comunicação entre os participantes da negociação e as relações semânticas que podem ser alcançadas. Porém, o modelo se mantém aberto para receber novas relações ou cálculos que venham a ser necessários.

Para estudar a viabilidade do modelo, foi realizado uma implementação de ferramentas. Essas ferramentas foram utilizadas por profissionais da área de computação e alunos de mestrado e doutorado para avaliarmos a utilização do modelo e recolhermos críticas para melhorias do processo e, consequentemente, das implementações.

5.1 – Contribuições

As principais contribuições deste trabalho se referem à elaboração do modelo de negociação no contexto da negociação de significados. O modelo foi elaborado de forma a garantir que a colaboração entre os negociadores seja de forma construtiva

levando ao bem comum. Dentro desta proposta, a principal contribuição está relacionada com a comunicação: o modelo utiliza-se da metodologia IBIS que, mesmo já existente na literatura há algum tempo, não tinha sido explorada para auxiliar na resolução de conflitos num SSN (Sistema de Suporte à Negociação), uma vez que estes tipos de sistemas começaram a ganhar uma atenção maior somente não há muito tempo, conforme pode ser lido em (PAULA, 2006). Neste ínterim, o grande diferencial em relação aos demais modelos é o suporte à socialização do conhecimento a partir da colaboração entre os negociadores e dessa explicitação estruturada de argumentos e idéias que é possível através da metodologia IBIS. Além disso, esse conhecimento externalizado, juntamente com os outros artefatos da negociação, é armazenado para que possa contribuir em negociações futuras.

Ainda, o modelo permite que a tarefa de mapeamento ontológico seja acompanhada por vários especialistas de domínios e não mais sendo parte decisória de somente uma pessoa. Assim, inserimos as pessoas que serão afetadas pelas relações semânticas declaradas no arquivo de mapeamento, uma vez que este processo, conforme disse Klein (2001), é complexo o bastante para impedir que seja completamente realizado somente por meios computacionais. Essa afirmação de Klein nos leva diretamente ao início deste trabalho (seção 1.1), onde apontamos as preocupações de Kieron O'Hara (2004), o qual alerta que as soluções de camadas superiores da Web Semântica, ou seja, as camadas mais relacionadas à explicitação e manipulação de conhecimento devam respeitar fenômenos sociais cada vez mais complexos.

Mesmo não sendo o ponto principal deste trabalho, a implementação do modelo GNoSIS contribui para a área de interoperabilidade semântica, principalmente em relação ao algoritmo para cálculo de similaridade entre conceitos de diferentes ontologias. A idéia primeira do algoritmo foi aplicada e testada em esquemas de banco de dados e apresentamos aqui uma variação dessa abordagem aplicada em estruturas de conhecimento, dando a flexibilidade de ajustes de pesos para se adequar em diferentes tipos de estruturas.

5.2 – Trabalhos Futuros

Como continuação do trabalho, é necessário um estudo de caso mais detalhado, envolvendo uma negociação de maior complexidade e em contextos diferentes, com mais participantes e métricas mais objetivas e quantitativas, dos ganhos obtidos quando da utilização do modelo.

As limitações da implementação, descritas no capítulo 4, também podem ser consideradas limitações atuais do GNoSIS que deverão ser solucionadas a partir dos trabalhos em andamento.

Muitas melhorias podem ser implementadas para dar suporte ao modelo. Entre elas, está a extensão do editor usado para criação de ontologías, o COE, para ler o arquivo de mapeamento e manipular as relações semânticas graficamente, facilitando a sua visualização e atualização. Assim, conforme o mediador for atualizando o arquivo de mapeamentos com as decisões de cada negociação, os participantes poderão acompanhar visualmente as relações entre os conceitos das duas ontologias após o final de cada negociação, tendo conhecimento rápido de todas as decisões formadas.

A mesa de negociação pode ser melhorada, conforme crítica dos usuários, ao deixar disponível para os participantes o tempo que lhes restam no seu turno de argumentação, definido no protocolo firmado. Para tal, pode-se ser implementado um cronômetro, diminuindo assim as tarefas do mediador e focando a sua intervenção mais para a discussão dos participantes e menos para as regras do protocolo.

Quanto ao cálculo de similaridades, pode ser testada uma abordagem para automaticamente calibrar os pesos do algoritmo conforme os graus de semelhança encontrados vão sendo condizentes com o mapeamento posteriormente escolhido. Pode também utilizar a base histórica como entrada do cálculo de similaridade, assim este poderia se basear em conceitos já previamente mapeados e ajustar o grau de similaridade em função desse histórico. O cálculo de similaridade pode também fazer uso de outros elementos ontológicos para comparação entre conceitos como instâncias de classes ou regras lógicas inseridas no arquivo OWL.

Quanto ao arquivo de mapeamento gerado, caso seja criada uma linguagem padrão para especificação de arquivos de mapeamento, este pode se adequar ao novo padrão. De qualquer forma, é necessário que seja criada uma máquina de inferências para ler esse arquivo a fim de que ele possa ser usado para troca de informações e instâncias entre as ontologias.

Além disso, ainda como trabalhos futuros pode-se considerar a adaptação do GNoSIS para utilização em ambientes de agentes de software, onde esses agentes poderiam ser responsáveis por negociar conceitos com uma intervenção menor do usuário e ser capazes de chegar a um acordo sobre a relação de seus conceitos. Tal modelo, contudo, deve ser mais bem estudado para que a negociação construtiva

continue válida, ou seja, um acordo seja formado através de um consenso e com um resultado considerado satisfatório para todas as partes.

Capítulo 6 – Referências Bibliográficas

ABERER, K., CATARCI, T., CUDRE-MAUROUX, P., *et al.*, 2004, "Emergent Semantics Systems". In: *Proceeding of the International Conference on Semantics of a Networked World*, pp. 14-43, França, Junho de 2004.

ACUFF, F.L., 1993, *How to negotiate anything with anyone anywhere around the world*, New York, American Management Association.

ARENS, Y., CHEE, C.Y., HSU, C.-N., *et al.*, 1993, "Retrieving and integrating data from multiple information sources", *International Journal of Cooperative Information Systems*, v. 2, n. 2 (Junho de 1993), pp. 127-158.

ARENS, Y., KNOBLOCK, C.A., 1992, "Planning and reformulating queries for semantically modeled multidatabase systems". In: *1st International Conference on Information and Knowledge Management*, pp. 92-101, Baltimore, MD, Outubro de 1992.

ARENS, Y., KNOBLOCK, C.A., SHEN, W.M., 1996, "Query reformulation for dynamic information integration", *Journal of Intelligent Information Systems*, v. 6, n. 2-3 (1996), pp. 99-130.

ARTICULATION, 2005, "Articulation Service Version 0.2". In: http://codip.grci.com/Tools/ArtiServicePage.html, acessado em dezembro de 2005.

BAEZA-YATES, R., RIBEIRO-NETO, B., 1999, *Modern Information Retrieval*, New York, ACM Press.

BAGBY, J.W., MULLEN, T., 2005, "Legal ontology of contract formation: application to eCommerce". In: *AAAI Workshop on Contexts and Ontologies*, Pittsburgh, PA, Julho de 2005.

BARROS, M.O., WERNER, C.M.L., TRAVASSOS, G.H., 2005, "Um Estudo Experimental sobre a Utilização de Modelagem e Simulação no Apoio à Gerência de Projetos de Software". In: *XIX Simpósio Brasileiro de Engenharia de Software (SBES)*, Uberlândia, MG, Brasil, Outubro de 2005.

BARTOLINI, C., PREIST, C., KUNO, H., 2006, "Requirements for Automated Negotiation". In: http://www.w3.org/2001/03/WSWS-popa/paper19, acessado em janeiro de 2006.

BATCHELDER, W.H., ROMNEY, A.K., 1986, "The statistical analysis of a general Condorcet model for dichotomous choice situations". In: GROFMAN, G., OWEN, G. (eds), *Information Pooling and Group Decision Making*, Greenwich, CT, JAI Press.

BATCHELDER, W.H., ROMNEY, A.K., 1988, "Test theory without an answer key", *Psychometrika*, v. 53, n. 1 (Março de 1988), pp. 71-92.

BAZERMAN, M.H., 2001, *Judgment in Managerial Decision Making*, 5 ed., New York, Wiley.

BECHHOFER, S., HORROCKS, I., GOBLE, C., *et al.*, 2001, "OilEd: a Reason-able Ontology Editor for the Semantic Web", *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence*, v. 2174, pp. 396–408.

BENEVENTANO, D., BERGAMASCHI, S., 2004, "The momis methodology for integrating heterogeneous data sources". In: *IFIP World Computer Congress*, pp. 19-24, Tolouse, França, Agosto de 2004.

BERNARAS, A., LARESGOITI, I., CORERA, J., 1996, "Building and Reusing Ontologies for Electrical Network Applications". In: *12th European Conference on Artificial Intelligence*, pp. 298-302.

BERNERS-LEE, T., 2005, "Semantic Web Road Map". In: http://www.w3.org/DesignIssues/Semantic.html, acessado em janeiro de 2005.

BERNERS-LEE, T., 2006, "Building the future". In: http://www.w3.org/2000/Talks/0906-xmlweb-tbl/slide9-6.html, acessado em janeiro de 2006.

BERNERS-LEE, T., 2006, "Rules and Facts: Inference engines vs Web. " In: http://www.w3.org/DesignIssues/Rules.html, acessado em junho de 2006.

BERNERS-LEE, T., HENDLER, J., LASSILA, O., 2001, "The Semantic Web", *Scientific American* (Maio de 2001).

BLÁZQUEZ, M., FERNÁNDEZ-LÓPEZ, M., GARCÍA-PINAR, J.M., *et al.*, 1998, "Building Ontologies at the Knowledge Level using the Ontology Design Environment". In: *Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Canada, Outubro de 1998.

BOUQUET, P., SERAFINI, L., ZANOBINI, S., 2003, "Semantic Coordination: A New Approach and an Application". In: *International Semantic Web Conference*, pp. 130-143, Flórida, EUA.

BRACHMAN, R., SCHMOLZE, J., 1985, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, v. 9, n. 2 (Abril-Junho de 1985), pp. 171-216.

BRAUNER, D.F., BRANDÃO, A.A.F., CUNHA, L.M., *et al.*, 2003, *Um Estudo de Caso para Avaliação do Knowledge Unified Process para o Desenvolvimento de Ontologias*, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, Relatório Técnico MCC50/03.

BREWSTER, C., O'HARA, K., FULLER, S., *et al.*, 2004, "Knowledge Representation with Ontologies: The Present and Future", *IEEE Intelligent Systems*, v. 19, n. 1 (Janeiro/Fevereiro de 2004), pp. 72-81.

BROCKMANS, S., HAASE, P., STUCKENSCHMIDT, H., 2006, "Formalism-Independent Specification of Ontology Mappings-A Metamodeling Approach". In: *OTM 2006 Conferences*, Montpellier, França, Outubro de 2006.

BUNDY, A., MCNEILL, F., SCHORLEMMER, M., 2003, "Dynamic ontology refinement". In: *ICAPS'03 Workshop on Plan Execution*, Trento, Itália, Junho de 2003.

BUNNINGEN, A.H.V., 2004, *Augmented Trading: Predicting stocks with OpenCyc*, Tese de Mestrado, Department of Computer Science, University of Twente, Enschede, Holanda.

BURANARACH, M., 2001, *The Foundation for Semantic Interoperability on the World Wide Web*, Tese de Phd, Department of Information Science and Telecommunications, School of Information Sciences, University of Pittsburgh, Pittsburgh, USA.

BURSTEIN, M.H., 2004, "Dynamic Invocation of Semantic Web Services That Use Unfamiliar Ontologies", *IEEE Intelligent Systems*, v. 19, n. 4 (Julho de 2004), pp. 67-73.

BURSTEIN, M.H., MCDERMOTT, D.V., 2005, "Ontology translation for interoperability among semantic Web services", *Artificial Intelligence Magazine*, v. 26, n. 1 (Março de 2005), pp. 71-82.

CASARE, S., SICHMAN, J.S., 2005, "Using a functional ontology of reputation to interoperate different agent reputation models", *Journal of the Brazilian Computer Society*, v. 11, n. 2 (Novembro de 2005), pp. 81-94.

CAULKINS, D., HYATT, S.B., 1999, "Using consensus analysis to measure cultural diversity in organizations and social movements", *Field Methods*, v. 11, n. 1 (1999), pp. 5-26.

CHAKRABARTI, S., 2000, "Data mining for hypertext: a tutorial survey", *ACM SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, v. 1, n. 2 (2000), pp. 1-11.

CHALUPSKY, H., 2000, "OntoMorph: A translation system for symbolic logic". In: *Seventh International Conference on Principles of Knowledge Representation and Reasoning*, pp. 471-482, San Francisco, CA, USA, 2000.

CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, V.R., 1999, "What Are Ontologies, and Why Do We Need Them?" *IEEE Intelligent Systems*, v. 14, n. 1 (Janeiro de 1999), pp. 20-26.

CLARKE, R., 2006, "Fundamentals of Negotiation". In: http://www.anu.edu.au/people/Roger.Clarke/SOS/FundasNeg.html, acessado em março de 2006.

COHEN, W., RAVIKUMAR, P., FIENBERG, S., 2003, "A comparison of string metrics for matching names and records". In: *KDD-2003 Workshop on Data Cleaning and Object Consolidation*, Washington DC, EUA, Agosto de 2003.

CONKLIN, J., 2005, *Dialogue Mapping: Building Shared Understanding of Wicked Problems*, John Wiley & Sons.

CONKLIN, J., 2006, "The IBIS Manual: A Short Course in IBIS Methodology". In: http://www.touchstone.com/tr/wp/IBIS.html, acessado em fevereiro de 2006.

CONNOLLY, D., HARMELEN, F.V., HORROCKS, I., *et al.*, 2005, "DAML+OIL Reference Description". In: http://www.w3.org/TR/daml+oilreference, acessado em janeiro de 2005.

COSTA, L., PIRES, P., MATTOSO, M., 2004, "WebComposer: a Tool for the Composition and Execution of Web Service-based Workflows". In: *Joint Conference 10th Brazilian Symposium on Multimedia and the Web 2nd Latin American Web Congress*, Ribeirão Preto, SP, Brasil, Outubro de 2004.

CRAVEN, M., MCCALLUM, A., DIPASQUO, D., *et al.*, 1999, *Learning to extract symbolic knowledge from the World Wide Web*, School of Computer Science, Carniege Mellon University, Relatório Técnico CMU-CS-98-122.

CRISTANI, M., CUEL, R., 2005, "A survey on ontology creation methodologies", *International Journal on Semantic Web & Information Systems*, v. 1, n. 2 (Junho de 2005), pp. 49-69.

DEAN, M.S., G.; BECHHOFER, S.; HARMELEN, F. V.; ENDLER, J.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A., 2006, "OWL: Web Ontology Language Reference". In: http://www.w3.org/TR/owl-ref/, acessado em janeiro de 2006.

DECKER, S., ERDMANN, M., FENSEL, D., *et al.*, 1999, "Ontobroker: Ontology based access to distributed and semi-structured information". In: MEERSMAN, R., TARI, Z., STEVENS, S. (eds), *DS-8: Semantic Issues in Multimedia Systems*, Boston, Kluwer Academic Publishers.

DECKER, S., MELNIK, S., HARMELEN, F.V., FENSEL, D., *et al.*, 2000, "The Semantic Web: The roles of XML and RDF. " *IEEE Internet Computing*, v. 4, pp. 63-73.

DING, Y., 2001, "Ontology: The enabler for the Semantic Web. A review of ontologies with the Semantic Web in view", *Journal of Information Science*, v. 27, n. 6 (2001), pp. 377-384.

EHRIG, M., STAAB, S., 2004, "QOM - quick ontology mapping". In: *Third International Semantic Web Conference (ISWC2004)*, pp. 683--696, Hiroshima, Japão, Novembro de 2004.

EUZENAT, J., BACH, T.L., BARRASA, J., *et al.*, 2004, *D2.2.3: State of the Art on Ontology Alignment*, Knowledge Web, Relatório Técnico KWEB/2004/D2.2.3/v1.2.

EUZENAT, J., VALTCHEV, P., 2003, "An integrative proximity measure for ontology alignment". In: *ISWC-2003 Workshop on Semantic Information Integration*, pp. 33–38, Sanibel Island, EUA, 2003.

FAATZ, A., STEINMETZ, R., 2002, "Ontology Enrichment with Texts from the WWW", *Semantic Web Mining 2nd Workshop at ECML/PKDD-2002* (Agosto de 2002).

FELICÍSSIMO, C.H., 2004, *Interoperabilidade Semântica na Web: Uma Estratégia para o Alinhamento Taxonômico de Ontologias*, Tese de Mestrado, Pograma de Pós-Graduação em Informática do Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

FELICÍSSIMO, C.H., BREITMAN, K.K., 2004, "Uma Estratégia para o Alinhamento Taxonômico de Ontologias". In: *I Workshop de Web Semântica do XVIII Simpósio Brasileiro de Engenharia de Software (WWS'2004)*, Brasília, Brasil, Outubro de 2004.

FELICÍSSIMO, C.H., LEITE, J.C.P., BREITMAN, K.K., *et al.*, 2003, "Geração de Ontologias subsidiada pela Engenharia de Requisitos". In: *VI Workshop em Engenharia de Requisitos (WER-03)*, Piracicaba, São Paulo, Brasil, Novembro de 2003.

FELLBAUM, C., 1998, *WordNet: An Electronic Lexical Database*, Cambridge, USA, The MIT Press.

FENSEL, D., HORROCKS, I., HARMELEN, F.V., *et al.*, 2001, "OIL: An ontology infrastructure for the semantic Web", *IEEE Intelligent Systems* (Abril de 2001).

FERNÁNDEZ-LÓPEZ, M., GÓMEZ-PÉREZ, A., 2002, "Overview and Analysis of methodologies for building ontologies", *Knowledge Engineering Review (KER)*, v. 17, n. 2 (2002), pp. 129-156.

FÉRNANDEZ, M., GÓMEZ-PÉREZ, A., JURISTO, N., 1997, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering". In: *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pp. 33-40, Stanford, EUA, Março de 1997.

FERREIRA, A.B.D.H., 1986, *Novo dicionário de língua portuguesa*, 2 ed., Rio de Janeiro, Nova Fronteira.

FISHER, R., KOPELMAN, E., SCHNEIDER, A.K., 1994, *Beyond Machiavelli: Tools for Coping with Conflict*, Harvard University Press.

FISHER, R., URY, W.L., PATTON, B., 1991, *Getting to Yes: negotiating agreement without giving in*, 2 ed., USA, Penguin Books.

FJERMESTAD, J., HILTZ, S.R., 1999, "An assessment of group support systems experimental research: methodology and results", *Journal of Management Information Systems*, v. 15, n. 3 (1999), pp. 7-149.

FREITAS, F., BITTENCOURT, G., 2003, "An ontology-based architecture for cooperative information agents". In: *18th International Joint Conference of Artificial Intelligence*, pp. 37-42, Acapulco, México, Agosto de 2003.

FREITAS, F., STUCKENSCHIMDT, H., NOY, N., 2005, "Ontology Issues and Applications", *Journal of the Brazilian Computer Society*, v. 11, n. 2 (Novembro de 2005), pp. 5-16.

GANGEMI, A., GUARINO, N., MASOLO, C., *et al.*, 2003, "Sweetening wordnet with DOLCE", *Artificial Intelligence Magazine*, v. 24, n. 3 (Setembro de 2003), pp. 13-24.

GENESERETH, M.R., FIKES, R.E., 1992, *Knowledge Interchange Format, Version 3.0 Reference Manual*, Computer Science Department, Stanford University, Relatório Técnico Logic-92-1.

GINSBERG, M.L., 1991, "Knowledge Interchange Format: the KIF of Death", *AI Magazine*, v. 12, n. 3 (1991), pp. 57-63.

GIUNCHIGLIA, F., SHVAIKO, P., 2003, "Semantic matching". In: *IJCAI 2003 Workshop on ontologies and distributed systems*, pp. 139-146, Acapulco, México, 2003.

GÓMEZ-PÉREZ, A., CORCHO, O., FERNÁNDEZ-LÓPEZ, M., 2004, *Ontological Engineering*, 1 ed., Alemanha, Springer Verlag.

GOSLING, J., JOY, B., STEELE, G., *et al.*, 2005, *The Java Language Specification*, 3 ed., Prentice Hall PTR.

GRUBER, T.R., 1993, "A translation approach to portable ontology specifications", *Knowledge Acquisition.*

GRUBER, T.R., 1995, "Towards Principles for the Design of Ontologies for Knowledge Sharing", *International Journal of Human-Computer Studies*, v. 43, n. 5/6 (1995), pp. 907-928.

GRUNINGER, M., ATEFI, K., FOX, M.S., 2000, "Ontologies to support process integration in enterprise engineering", *Computational and Mathematical Organization Theory*, Kluwer Academic Publishers.

GRUNINGER, M., KOPENA, J., 2005, "Semantic Integration Through Invariations", *Artificial Intelligence Magazine*, v. 26, n. 1 (2005), pp. 11-20.

GUARINO, N., 1996, "Understanding, Building and Using Ontologies". In: *10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

GUARINO, N., 1998, "Formal Ontology and Information Systems". In: *1st International Conference on Formal Ontologies in Information Systems*, pp. 3-15, Trento, Italy, Junho de 1998.

HOBBS, J.R., APPELT, D., BEAR, J., *et al.*, 1997, "FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text", *Finite-State Language Processing* (1997), pp. 383-406.

HORNSBY, K., EGENHOFER, M., 2002, "Modeling moving objects over multiple granularities", *Special issue on Spatial and Temporal Granularity, Annals of Mathematics and Artificial Intelligence*, v. 36 (2002), pp. 177-194.

HORROCKS, I., PATEL-SCHNEIDER, P.F., HARMELEN, F.V., 2003, "From SHIQ and RDF to OWL: The Making of a Web Ontology Language", *Journal of Web Semantics*, v. 1, n. 1 (2003), pp. 7-26.

HÖST, M., REGNELL, B., WOHLIN, C., 2000, "Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment", *Empirical Software Engineering*, v. 5, n. 3 (Novembro de 2000), pp. 201-214.

HUNG, P.C.K., MAO, J.-Y., 2002, "Modeling e-Negotiation Activities with Petri Nets". In: *35th Hawaii International Conference on System Sciences*, pp. 379-388, Hawaii, Janeiro de 2002.

IEEE, 1995, *Guide for Software Quality Assurance Planning*, IEEE Computer Society, Relatório Técnico Std. 730.1-1995.

JAKONIENE, V., 2006, "Linköpings Universitet's Course on Logics for the Web: Ontology Integration". In: http://www.ida.liu.se/labs/iislab/courses/LW/slides/ontologyIntegration.pdf, acessado em Julho de 2006.

JENA, 2005, "Jena 2 Ontology API". In: http://jena.sourceforge.net/ontology/, acessado em agosto de 2005.

KALFOGLOU, Y., SCHORLEMMER, M., 2003, "Ontology Mapping: the state of art", *The Knowledge Engineering Review*, v. 18, n. 1 (Janeiro de 2003), pp. 1-31.

KARNSTEDT, M., SATTLER, K.-U., GEIST, I., *et al.*, 2003, "Semantic caching in ontology-based mediator systems". In: *3rd International Workshop of the GI Working Group "'Web und Datenbanken"*, pp. 155-169, Berlin, Outubro de 2003.

KERSTEN, G., NORONHA, S., 1999a, "Negotiations via the Word Wide Web: A Cross-cultural Study of Decision Making", *Group Decision and Negotiations*, v. 8 (1999), pp. 251-279.

KERSTEN, G.E., 1985, "NEGO - Group decision support system", *Information & Management*, v. 8, n. 5 (Maio de 1985), pp. 237-246.

KERSTEN, G.E., 2001, "Modeling Distributive and Integrative Negotiations. Review and Revised Characterization", *Group Decision and Negotiations*, v. 10, n. 6 (2001), pp. 493-514.

KERSTEN, G.E., CONCILIO, G., 2002, "The Science and Engineering of E-negotiation: Review of the Emerging Field", *InterNeg*, v. 4, n. 2 (2002).

KERSTEN, G.E., NORONHA, S.J., 1999b, "WWW-based Negotiation Support: Design, Implementation and Use", *Decision Support Systems*, v. 25, n. 2 (1999), pp. 135-154.

KIFER, M., LAUSEN, G., WU, J., 1995, "Logical Foundations of Object-Oriented and Frame-Based Languages", *Journal of the ACM*, v. 42, n. 4 (Julho de 1995), pp. 741-843.

KIM, H.M., FOX, M.S., GRUNINGER, M., 1999, "An ontology for quality management - Enabling quality problem identification and tracing", *BT Technology Journal*, v. 17, n. 4 (Outubro de 1999), pp. 131-140.

KLEIN, M., 2001, "Combining and Relating Ontologies: An Analysis of Problems and Solutions". In: *Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence*, pp. 53-62, Seattle, USA, Agosto de 2001.

KUNZ, W., RITTEL, H.W.J., 1970, *Issues as Elements of Information Systems*, Institute of Urban & Regional Development, University of California.

KUPER, J., SAGGION, H., CUNNINGHAM, H., *et al.*, 2003, "Intelligent Multimedia Indexing and Retrieval through Multi-source Information Extraction and Merging". In: *Acapulco, México*, pp. 409-414, International Joint Conference of Artificial Intelligence, Agosto de 2003.

LAUSER, B., WILDEMANN, T., POULOS, A., *et al.*, 2002, "A comprehensive framework for building multilingual domain ontologies: Creating a prototype biosecurity ontology". In: *International Conference on Dublin Core and Metadata for e-Communities*, v. 1, pp. 113-123, Florence, Italy, Outubro de 2002.

LEITE, F.G., REZENDE, J., SOUZA, J.F., *et al.*, 2005, "Mobile System To Support Learning Communities Through The Exchange Of Knowledge Chains". In: *IADIS - International Conference on Mobile Learning*, Qawra, Malta, Junho de 2005.

LIM, J., GAN, B., CHANG, T.-T., 2002, "A survey on NSS adoption intention". In: *35th Annual Hawaii International Conference on System Sciences (HICSS 35)*, v. 1, pp. 399-408, Jan, Hawaii, Janeiro de 2002.

LOMUSCIO, A.R., WOLLDRIDGE, M., JENNINGS, N.R., 2003, "A classification scheme for negotiation in electronic commerce", *International Journal of Group Decision and Negotiation*, v. 12, n. 1 (2003), pp. 31-56.

LOOM, 2006, "Loom Project Home Page". In: http://www.isi.edu/isd/LOOM/LOOM-HOME.html, acessado em Junho de 2006.

MADHAVAN, J., BERNSTEIN, P.A., RAHM, E., 2001, "Generic schema matching with cupid". In: *27th International Conference on Very Large Data Bases*, pp. 49-58, 2001.

MAEDCHE, A., 2002, *Ontology Learning for the Sematic Web*, 1a ed., Kluwer Academic Publisher.

MANNILA, H., 1996, "Data mining: machine learning, statistics, and databases". In: *8th International Conference on Scientific and Statistical Database Management*, pp. 2–9, Stockholm, 1996.

MARTINELLI, D.P., ALMEIDA, A.P., 1997, *Negociação: Como transformar confronto em cooperação*, São Paulo, Atlas.

MAYFIELD, R., 2006, "Negotiation and Social Software". In: http://ross.typepad.com/blog/2003/08/negotiation_and.html, acessado em Outubro de 2006.

MCGUINNESS, D.L., FIKES, R., RICE, J., *et al.*, 2003, "An environment for merging and testing large ontologies". In: *7th International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pp. 483-493, Breckenridge, Colorado, USA, 2003.

MCNEILL, F., BUNDY, A., WALTON, C., 2004, "Diagnosing and Repairing Ontological Mismatches". In: *Starting AI Researchers' Symposium*, Valencia, Julho de 2004.

MCNEILL, F., BUNDY, A., WALTON, C., 2005, "Planning from rich ontologies through translation between representations". In: *ICAPS Workshop on The role of ontologies on planning and scheduling*, Monterey, CA, Junho de 2005.

MEDELYAN, O., WITTEN, I.H., 2006, "Measuring inter-indexer consistency using a thesaurus". In: *6th ACM/IEEE-CS Joint Conference on Digital libraries*, v. 1, pp. 274-275, Chapel Hill, NC, USA, Junho de 2006.

MELNIK, S., GARCIA-MOLINA, H., RAHM, E., 2002, "Similarity Flooding: A versatile graph matching algorithm and its application to schema matching". In: *International Conference on Data Engineering (ICDE)*, pp. 117--128, San Jose, CA, USA, Fevereiro de 2002.

MENA, E., KASHYAP, V., SHETH, A., *et al.*, 1996, "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies". In: *1st IFCIS International Conference on Cooperative Information Systems (CoopIS '96)*, pp. 14-25, Brussels, Bélgica, Junho de 1996.

MILLER, E., MANOLA, F., MCBRIDE, B., 2006, "Resource Description Framework (RDF) Primer". In: http://www.w3.org/TR/rdf-primer/, acessado em junho de 2006.

MILLS, H., 1991, *Negotiation: The art of Winning*, EUA, Gower Publishing Company Limited.

MITRA, P., WIEDERHOLD, G., KERSTEN, M., 2000, "A Graph-Oriented Model for Articulation of Ontology Interdependencies". In: *Conference on Extending*

Database Technology (EDBT'00), v. 1777, pp. 86-100, Konstanz, Alemanha, 2000.

MOORE, C.W., 1982, *Natural Resources Conflict Management*, Colorado, Accord Associates.

MULLER, G., VERCOUTER, L., BOISSIER, O., 2005, "A trust model for the reliability of agent communications". In: *8th Workshop on Trust, Privacy, Deception and Fraud In Agent Societies*, Utrecht, Holanda, Maio de 2006.

MYSQL, 2005, "Manual de Referência do MySQL 4.1". In: http://dev.mysql.com/doc/refman/4.1/pt/, acessado em 2005.

NIELSEN, J., 1994, "Heuristic Evaluation". In: NIELSEN, J., MACK, R.L. (eds), *Usability Inspection Methods*, New York, John Wiley and Sons.

NILES, I., PEASE, A., 2003, "Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology". In: *International Conference on Information and Knowledge Engineering (IKE '03)*, Las Vegas, Nevada, Junho de 2003.

NOVAK, J., CUEL, R., SARINI, M., *et al.*, 2004, "A Tool for Supporting Knowledge Creation and Exchange in Knowledge Intensive Organisations". In: *I-KNOW '04 - 4th International Conference on Knowledge Management*, v. 4, Áustria, Junho-Julho de 2004.

NOY, N., 1997, *Knowledge Representation for Intelligent Information Retrieval in Experimental Sciences*, Tese de PhD, Computer Science Faculty, Notheastern University of Boston, Boston, USA.

NOY, N., MUSEN, M.A., 2001, "Anchor-PROMPT: Using Non-Local Context for Semantic Matching". In: *17th Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence*, v. 1, pp. 63-70, Seattle, EUA, Agosto de 2001.

NOY, N., MUSEN, M.A., 2003, "The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping", *International Journal of Human-Computer Studies*, v. 59, n. 6 (2003), pp. 983-1024.

NOY, N.F., KLEIN, M., 2004, "Ontology Evolution: Not the Same as Schema Evolution", *Knowledge and Information Systems*, v. 6, n. 4 (2004), pp. 428-440.

NOY, N.F., MCGUINNESS, D.L., 2001, *Ontology development 101: A guide to creating your first ontology*, Stanford Knowledge Systems Laboratory, Relatório Técnico KSL-01-05.

NOY, N.F., MUSEN, M.A., 1999, "SMART: Automated Support for Ontology Merging and Alignment". In: *12th Workshop on Knowledge acquisition, modeling and management*, v. 4, pp. 1-20, Banff, Canadá, Outubro de 1999.

NOY, N.F., SINTEK, M., DECKER, S., *et al.*, 2001, "Creating Semantic Web contents with Protege-2000", *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, v. 16, n. 2, pp. 60-71.

O'HARA, K., 2004, "Ontologies and Technologies: Knowledge Representation or Misrepresentation". In: *Semantic Web Workshop from Special Interest Group on Information Retrieval Forum*, v. 38, Sheffield, Dezembro de 2004.

OLIVEIRA, J., SOUZA, J.F., PAULA, M., *et al.*, 2006, "Meaning Negotiation for Consensus Formation in Ontology Construction". In: *10th International Conference on Computer Supported Cooperative Work in Design*, v. 1, pp. 1-6, Nanjing, China, Maio de 2006.

OLIVEIRA, J., SOUZA, J.F., PAULA, M., *et al.*, 2007, "A Business-Based Negotiation Process for Reaching Consensus of Meanings". In: SHEN, W., CHAO, K.-M., LIN, Z., *et al.* (eds), *CSCW in Design III*, China, Springer, a publicar.

OLIVEIRA, R.M.V.B., 2002, *Web Semântica: Novo Desafio para os Profissionais da Informação*, Departamento de Informática da Pontifícia Universidade Católica de Campinas, Relatório Técnico.

ORLEAN, D., 2003, *Um Processo Unificado para Engenharia de Ontologias*, Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

OUKSEL, A., 1999, "A Framework for a Scalable Agent Architecture of Cooperating Heterogeneous Knowledge Sources", *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet* (1999), pp. 100-121.

OWL-S, 2005, "Ontology Web Language for Services". In: http://www.daml.org/services/owl-s/1.0, acessado em Dezembro de 2005.

PAULA, M.M.V., OLIVEIRA, J., SOUZA, J.M., 2004, "Knowledge Management in the Business Process Negotiation", *Web Information Systems - WISE 2004, 5th International Conference on Web Information Systems Engineering, Lecture Notes in Computer Science*, v. 3306 (novembro de 2004), pp. 503-509.

PAULA, M.M.V.D., 2006, *Negosys: um ambiente para gestão do conhecimento na negociação*, Tese de Doutorado, Programa de Pós-Graduação em Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

PAZ-TRILLO, C., WASSERMANN, R., BRAGA, P., 2005, "An Information Retrieval Application using Ontologies", *Journal of the Brazilian Computer Society*, v. 11, n. 2 (Novembro de 2005), pp. 17-32.

PEREIRA, V.B., REZENDE, J.L., XÉXEO, G., *et al.*, 2006, "Building a Personal Knowledge Recommendation System using Agents, Learning Ontologies and Web Mining". In: *10th International Conference on CSCW in Design*, Nanjing, China, Maio de 2006.

PINTO, S.H., GÓMEZ-PÉREZ, A., MARTINS, J.P., 1999, "Some Issues on Ontology Integration". In: *Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends (IJCAI99's)*, v. 18, pp. 7-12, Stockholm, Suécia, Agosto de 1999.

PORTER, M.F., 1980, "An algorithm for suffix stripping", *Program*, v. 14, n. 3 (1980), pp. 130-137.

PREDOIU, L., FEIER, C., SCHARFFE, F., *et al.*, 2006, *State-of-the-art Survey on Ontology Merging and Aligning V2*, SEKT Project, Relatório Técnico D4.2.1 (WP4), IST-2003-506826.

PROTÉGÉ-OWL, 2005, "The Protégé-OWL API". In: http://protege.stanford.edu/plugins/owl/api/, acessado em janeiro de 2005.

RAHM, E., BERNSTEIN, P., 2001, "A survey of approaches to automatic schema matching", *VLDB Journal*, v. 10, n. 4 (2001), pp. 334–350.

RAIFFA, H., 2003, *The Art and Science of Negotiation*, 7 ed., London, England, Harvard University Press.

RATIONAL, 1998, *Rational Unified Process: Best Practices for Software development Teams*, Rational Team, Relatório Técnico TP026B, Rev 11/01.

RDF, 2005, "Resource description framework (RDF): model and syntax specification". In: http://www.w3.org/TR/REC-rdf-syntax-19990222/, acessado em fevereiro de 2005.

RESNIK, P., ELKISS, A., LAU, E., *et al.*, 2005, "The Web in Theoretical Linguistics Research: Two Case Studies Using the Linguist's Search Engine". In: *31st Meeting of the Berkeley Linguistics Society*, Berkeley, Califórnia, USA, Fevereiro de 2005.

REZENDE, J.L., SOUZA, J.F., BONFIM, E.L., *et al.*, 2006, "An Empirical Study on Groupware Support for Water Resources Ontology Integration". In: *8th Asia-Pacific Web Conference - Frontiers of WWW Research and Development - APWeb*, v. 3841, pp. 1010-1021, Harbin, China, Agosto de 2006.

REZENDE, J.L., SOUZA, J.F., XEXEO, G., *et al.*, 2005, "Peer-to-Peer Collaborative Integration of Dynamic Ontologies". In: *9th International Conference on Computer Supported Cooperative Work in Design*, v. 2, pp. 1158-1163, Conventry, Inglaterra, Maio de 2005.

ROBIN, J., RAMALHO, F., 2001, "Empirically evaluating WordNet-based query expansion in a web search engine setting". In: *International Workshop on Information Retrieval* pp. 80-89, Oulu, Finlândia, Setembro de 2001.

RODRIGUES, S., OLIVEIRA, J., SOUZA, J., 2005, "Competence Mining for Team Formation and Virtual Community Recommendation". In: *Proceedings of Intemationat Conference on Computer Supported Cooperative Work in Design*, v. 1, pp. 44-49, Conventry, Inglaterra, Maio de 2005.

RODRÍGUEZ, A., EGENHOFER, M., 2003, "Determining Semantic Similarity Among Entity Classes from Different Ontologies", *IEEE Transactions on Knowledge and Data Engineering*, v. 15, n. 2 (2003), pp. 442-456.

ROMNEY, A.K., WELLER, S.C., BATCHELDER, W.H., 1986, "Culture as consensus: A theory of culture and informant accuracy", *American Anthropologist*, v. 88, n. 2 (1986), pp. 313-338.

SASTRE, G., MORENO, M., BUSQUETS, M.D., *et al.*, 1997, *Temas transversais em educação – bases para uma formação integral*, São Paulo, Brazil, Ed. Ática.

SCHOOP, M., LIST, T., 2003, "Negoist: a negotiation support system for electronic business-to-business negotiations in e-commerce", *Data & Knowledge Engineering*, v. 47, n. 3 (2003), pp. 371-402.

SEBENIUS, J.K., 1992, "Negotiation Analysis: A Characterization and Review", *Management Science*, v. 38, n. 1 (1992), pp. 18-38.

SHUM, S.B., 2004, "Contentious, dynamic, multimodal domains... and ontologies?" *IEEE Intelligent Systems*, v. 19, n. 1 (2004), pp. 72-73.

SHUM, S.B., SELVIN, A.M., SIERHUIS, M., *et al.*, 2006, "From gIBIS to MEMETIC: Evolving a Research Vision into a Practical Tool ". In: *Design Rationale Workshop: Design, Computing & Cognition Conference*, Eindhoven, Holanda, Julho de 2006.

SHVAIKO, P., 2004a, "A classification of schema-based matching approaches". In: *Meaning Coordination and Negotiation workshop at International Semantic Web Conference (ISWC)*, Hiroshima, Japão, Novembro de 2004.

SHVAIKO, P., 2004b, *Iterative schema-based semantic matching*, University of Trento (IT), Relatório Técnico DIT-04-020.

SMARTSETTLE, 2006, "SmartSettle". In: www.smartsettle.com, acessado em setembro de 2006.

SOLINGEN, R.V., BERGHOUT, E., 1999, *The Goal / Question / Metric Method: A Practical Guide for Quality Improvement of Software Development*, McGrawn Hill.

SOUZA, J.F., OLIVEIRA, J., PAULA, M., *et al.*, 2006a, "A New Approach for Ontology Integration based on Collaborative Negotiation Model and Similarity Among Schemas". In: *23rd British National Conference on Databases*, Belfast, Northern Ireland, Julho de 2006.

SOUZA, J.F., PAULA, M., OLIVEIRA, J., *et al.*, 2006b, "Meaning Negotiation: Applying Negotiation Models to Reach Semantic Consensus in Multidisciplinary Teams". In: *Group Decision and Negotiation*, v. 1, pp. 297-300, Karlsruhe, Alemanha, Junho de 2006.

SOUZA, J.M., 1986, *Software Tools for Conceptual Schema Integration*, Tese de Doutorado, University of East Anglia.

SOWA, J., 1999, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Pacific Grove, CA, Brooks Cole Publishing Co.

SPANGLER, B., 2005, "Best Alternative to a Negotiated Agreement (BATNA)". In: http://www.beyondintractability.org/essay/batna/, acessado em abril de 2005.

STUCKENSCHMIDT, H., USCHOLD, M., 2005, "Representation of Semantic Mappings", *Semantic Interoperability and Integration*, v. 04391 (2005).

SUOWG, 2006, "Standard Upper Ontology Working Group". In: http://suo.ieee.org/, acessado em Setembro de 2006.

SURE, Y., 2003, *Methodology, tools and case studies for ontology based knowledge management*, Tese de Doutorado, Department of Economics and Business Engineering, University of Karlsruhe, Karlsruhe.

SURE, Y., ERDMANN, M., ANGELE, J., *et al.*, 2002, "OntoEdit: Collaborative Ontology Development for the Semantic Web", *First International Semantic Web Conference*, v. 2342, pp. 221–235.

SURE, Y., STUDER, R., 2002, *On-To-Knowledge Methodology*, Institute AIFB, University of Karlsruhe, Relatório Técnico EU IST-1999-10132.

SUWANMANEE, S., BENSLIMANE, D., THIRAN, P., 2005, "Owl-based approach for semantic interoperability". In: *19th International Conference on Advanced Information Networking and Applications*, v. 1, pp. 145-150, Taipei, Taiwan, Março de 2005.

SWARTOUT, B., PATIL, R., KNIGHT, K., *et al.*, 1996, "Toward Distributed Use of Large-Scale Ontologies". In: *Tenth Knowledge Acquisition for Knowledge-based Systems Workshop*, Alberta, Canada, Novembro de 1996.

TEMPICH, C., PINTO, S., STAAB, S., *et al.*, 2004, "A case study in supporting distributed, loosely-controlled and evolving engingeering of ontologies (DILIGENT)". In: *4th International Conference on Knowledge Management (I-Know'04)*, Graz, Austria.

THOMPSON, L., 1996, "Lose-lose Agreements in Interdependent Decision-Making", *Psychological Bulletin*, v. 120, n. 3 (1996), pp. 396-409.

TICHY, W., 2001, "Hints for Reviewing Empirical Work in Software Engineering", *Empirical Software Engineering: An International Journal*, v. 5 (2001), pp. 309-312.

TRAVASSOS, G.H., GUROV, D., AMARAL, E.A.G., 2002, *Introdução à Engenharia de Software Experimental*, Programa de Engenharia de Sistemas e Computação, Relatório Técnico RT-ES-590-02.

TROJAHN, C., MORAES, M.C., QUARESMA, P., *et al.*, 2006, "A Negotiation Model for Ontology Mapping". In: *International Conference on Intelligent Agent Technology*, pp. 762-768, Hong Kong, China, 22 de Dezembro de 2006.

URY, W., 1993, *Getting Past No: Negotiating your Way from Confrontation to Cooperation*, New York, Bantam Books.

USCHOLD, M., GRUNINGER, M., 1996, "Ontologies: Principles, methods and applications", *Knowledge Engineering Review (KER)*, v. 11, n. 2 (Fevereiro de 1996), pp. 93-155.

USCHOLD, M., JASPER, R., 1999, "A Framework for Understanding and Classifying Ontology Applications". In: *IJCAI-99 Workshop on Ontologies and Problem-Solving Methods*, Stockholm, Sweden, 2 de Agosto de 1999.

USCHOLD, M., KING, M., 1995, "Towards a Methodology for Building Ontologies". In: *International Joint Conference on Artificial Intelligence*, Montréal, Québec, Canada, Agosto de 1995.

VARELLA, A., 2007, *COOPRACTICE – Comunidades de prática virtuais apoiadas por ontologias*, Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

VISSER, P.R.S., JONES, D.M., BENCH-CAPON, T.J.M., *et al.*, 1997, "An analysis of ontological mismatches: Heterogeneity versus interoperability". In: *AAAI-97 Spring Symposium on Ontological Engineering*, pp. 164-172, Califórnia , EUA, 1997.

VOLZ, R., ABECKER, A., 2003, "Ontologies: representation, engineering, learning & applications". In: *3rd IFIP Conference on E-commerce, E-business and E-government (I3E'03)*, São Paulo, Brasil, Setembro de 2003.

W3C, 2005, "OWL Web Ontology Language Use Cases and Requirements". In: http://www.w3.org/TR/webont-req/, acessado em fevereiro de 2005.

W3C, 2005, "W3C Semantic Web Activity". In: http://www.w3.org/2001/sw/, acessado em janeiro de 2005.

WALTON, R.E., MCKENZIE, R.B., 1965, *A Behavioral Theory of Labor Negotiations*, New York, McGraw-Hill.

WANG, J., 1998, "An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 8 (Agosto de 1998), pp. 889-895.

WIEDERHOLD, G., 1994, "An algebra for ontology composition". In: *Proceedings of 1994 Monterey Workshop on Formal Methods*, v. 5661, pp. 56-61, Monterey, CA, Setembro de 1994.

XEXÉO, G., VIVACQUA, A., DE SOUZA, J.M., *et al.*, 2005, "COE: A collaborative ontology editor based on a peer-to-peer framework", *Advanced Engineering Informatics*, v. 19, n. 2, pp. 113-121.

XML, 2006, "Extensible Markup Language (XML)". In: http://www.w3.org/XML/, acessado em junho de 2006.

YAN, Y., ZHANG, J., YAN, M., 2006, "Ontology Modeling for Contract: Using OWL to Express Semantic Relations". In: *Enterprise Distributed Object Computing Conference, EDOC '06*, pp. 409-412, Hong Kong, China, Outubro de 2006.

YUAN, Y., TUREL, O., 2004, *A Business Model for e-Negotiation in Eletronic Commerce*, InterNeg Research Papers, Relatório Técnico INR 02/04.

ZIEGLER, P., DITTRICH, K.R., 2004, "Three decades of data integration - all problems solved?" In: *18th IFIP World Computer Congress*, v. 12, pp. 3-12, Toulouse, França, Agosto de 2004.

Anexo A

QUESTIONÁRIO PARA AVALIAÇÃO DO PERFIL DO USUÁRIO

1) **Dados Pessoais**

Nome: _____

Data de nascimento: ____/____/____ Sexo: ____

Correio eletrônico: _____

2) **Escolaridade**

Grau de Escolaridade:

( ) Não-Graduado ( ) Mestre

( ) Graduado ( ) Doutor

Área de estudo/atuação: _____

Nas perguntas abaixo, quando duas ou mais alternativas forem válidas, marque a alternativa que mais se aplica ao seu caso.

3) **Conhecimento do domínio**

Você já trabalhou com ontologias?

[ ] Construí ontologias

[ ] Fui usuário de sistema que utilizava ontologias

[ ] Implementei sistema que utiliza ontologias

[ ] Homologuei conteúdo de ontologias

Determine seu conhecimento do domínio abordado pelas ontologias

( ) nulo ( ) pouco ( ) parcial ( ) experiente ( ) especialista

Determine a forma com que você adquiriu tal conhecimento

[ ] Disciplinas da faculdade [ ] Cursos de extensão

[ ] No trabalho [ ] Auto-aprendizado

4) Experiência de negociação

Se você está envolvido em alguma atividade que necessite de conhecimentos de negociação ou decisões colaborativas, descreva brevemente seu papel e suas atividades.

Você já utilizou algum outro sistema de suporte à negociação?

( ) sim ( ) não

Qual (is):

Você considera que a utilização de um sistema de suporte à negociação pode facilitar a obtenção de um acordo?

( ) sim ( ) não

Você já leu alguma publicação especifica sobre negociação?

( ) sim ( ) não

Se sim, poderia citar o nome de um dos autores ou da publicação?

Como você se considera no que se refere a sua experiência em negociação?

( )pouco experiente ( )experiência razoável ( )experiente ( )muito experiente

Anexo B

QUESTIONÁRIO PARA AVALIAÇÃO DO MODELO E DA IMPLEMENTAÇÃO

Responda as perguntas abaixo com base na sua experiência com a implementação do modelo GNoSIS considerando tanto as ferramentas implementadas quanto o processo de negociação que foi utilizado.

1) Consigo expressar meus argumentos eficientemente e entender os argumentos dos outros participantes através do mecanismo de argumentação pertencente à ferramenta?

Extremamente difícil O O O O O O Extremamente fácil

2) Os elementos disponíveis no ambiente implementado do GNoSIS permitem a negociação com os demais colegas, mesmo que não estejam ao mesmo tempo na mesa de negociação?

( ) sim ( ) não ( ) razoável ( ) não se aplica

3) As ferramentas implementadas possuem todas as funcionalidades para a negociação colaborativa por mim esperadas?

( ) não ( ) sim ( ) razoável ( ) não se aplica

Se não, quais as funcionalidades que sentiu falta?

_____

4) Fez uso da pesquisa de negociações passadas? Se sim, a pesquisa lhe ajudou de alguma forma? Como?

5) Sentiu falta de algum tipo de informação para a realização da negociação? Quais?

6) Durante a negociação, você e sua contraparte fizeram algum tipo de contato fora do permitido no protocolo?

( ) sim ( ) não

7) Os objetivos da negociação (mapeamento da ontologia) foram atingidos?

( ) sim ( ) não ( ) razoável ( ) não se aplica

8) Estou totalmente satisfeito com a tarefa?

Extremamente insatisfeito O O O O O O Extremamente satisfeito

9) Utilizaria o modelo para negociar mapeamentos de ontologia na prática?

( ) sim ( ) não ( ) possivelmente ( ) não se aplica

10) O quão fácil ou difícil você achou usar o GNoSIS?

Extremamente difícil O O O O O O Extremamente fácil

11) Quais foram as maiores dificuldades e/ou problemas encontrados ao trabalhar em negociar eletronicamente?

12) Quais foram as vantagens encontradas ao trabalhar com uma ferramenta de negociação eletrônica?

13) Quanto tempo durou o processo de negociação?

14) Sua experiência de trabalho em grupo de negociação poderia ser considerada:

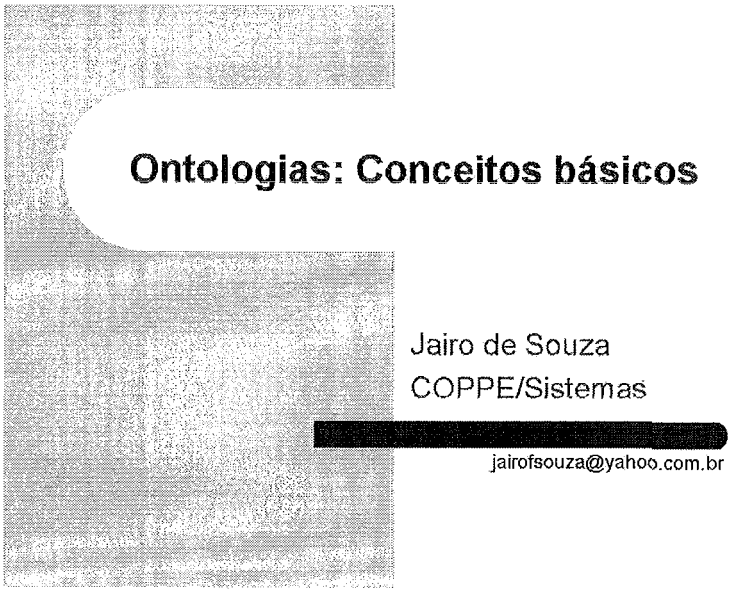
( ) péssima ( ) ruim ( ) razoável ( ) boa ( ) ótima

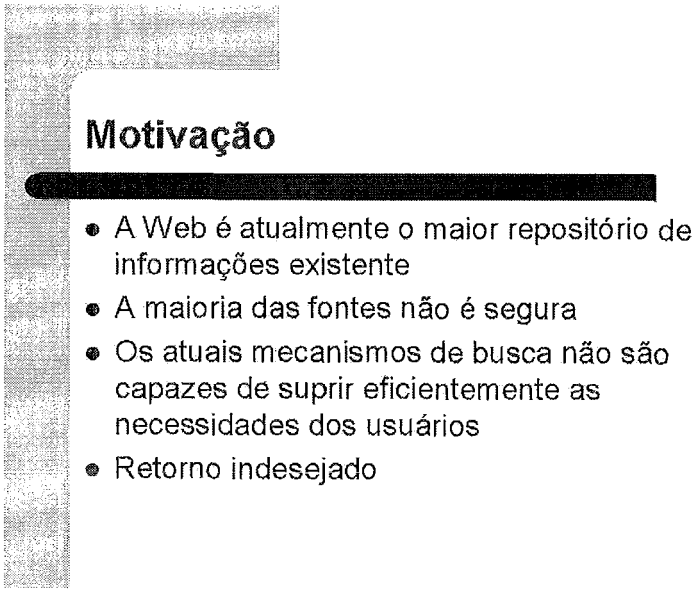
15) Após a negociação, você e sua contraparte se comprometeram com o acordo firmado, ou seja, os mapeamentos, sejam em forma de arquivo de mapeamento ou não, serão realmente utilizados?

( ) sim ( ) não

Anexo C

TREINAMENTO SOBRE ONTOLOGIAS COM OS PARTICIPANTES DO ESTUDO DE OBSERVAÇÃO





Motivação

- Domínios Complexos
- Falta de clareza e ambigüidade nas informações disponíveis
- Sintaxe *versus* semântica

- Tim Berners-Lee → Web Semântica

Web Semântica

- Divide-se em:
  - Representação do conhecimento: estrutura o conteúdo significativo das páginas Web
  - Agentes: coletam o conteúdo na Web, o processam e permutam o resultado com outros programas
  - Ontologias: definem o conteúdo específico sobre o conhecimento a ser compartilhado e reusado entre diferentes agentes

Ontologias - Definição

- No ramo da meta-física é definida como o estudo da natureza do ser.
- Na filosofia, é o estudo da existência do ser.
- A ontologia é entendida como um sistema de categorias de uma determinada visão do mundo.

Ontologias - Definição

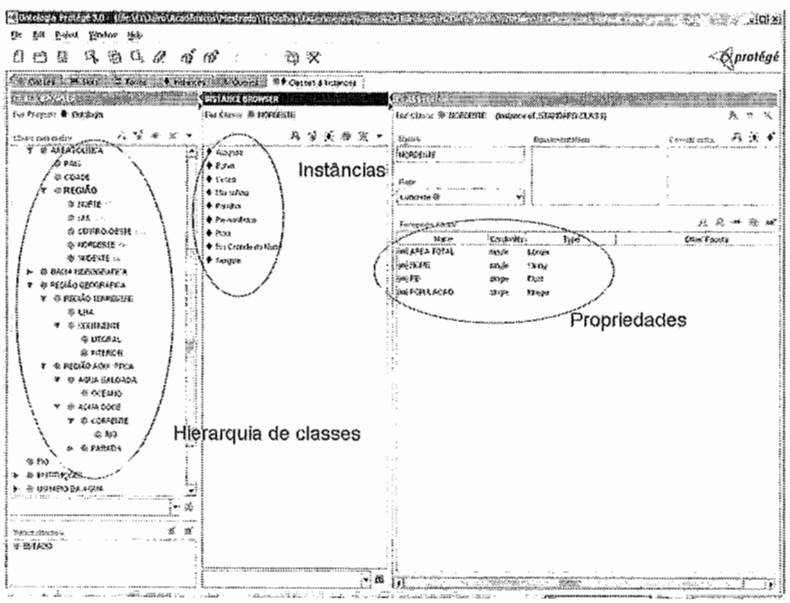
- Na ciência da computação:
  - *"Uma especificação explícita e formal de uma conceitualização compartilhada"* (Gruber)

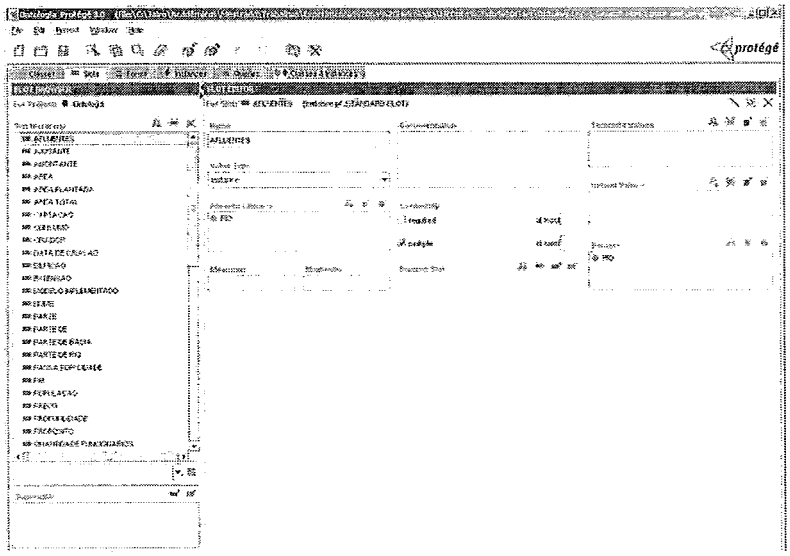
Deve ser compreensível por um computador, não podendo ser somente escrita em linguagem natural, sendo necessário que ela seja representável matematicamente, de forma não-ambígua

Uma abstração, visão simplificada do mundo que desejamos representar para algum propósito, construído através da identificação dos conceitos e relações relevantes

Ontologias - Estrutura

- Hierarquia de classes
- Propriedades de classes
  - Data Type x Object Type
  - **Restrições**
  - Domínio x alcance
- Instâncias
- Axiomas





Ontologias - Tipos

- Segundo Guarino:
  - Ontologias Genéricas (Upper Ontologies)
  - Ontologias de domínio
  - Ontologias de tarefa
  - Ontologias de aplicação

Por que desenvolver uma ontologia? I/II

- Para compartilhar um entendimento comum de uma estrutura de informação
  - entre pessoas
  - entre agentes de software
- Para possibilitar o reuso de conhecimento do domínio
  - introduzindo padrões para permitir interoperabilidade

Por que desenvolver uma ontologia? II/II

- Para explicitar afirmações a respeito de um domínio
  - mais fácil de modificar as "domain assumptions" (por ex., base de conhecimento sobre genética)
  - mais fácil de entender e atualizar "dados legados" (legacy data)
- Para separar conhecimento de domínio de conhecimento operacional
  - reuso de conhecimento de domínio e operacional separadamente

Uso de ontologias – Aplicações

- Modelagem de empresas:
  - Capturam estruturas da organização de uma empresa com ênfase em atividades e processos
- Modelagem de processo de negócio:
  - Formato para troca de descrição de processo automático entre vários modelos de negócio
- Aplicações médicas:
  - Metodologia para integrar terminologias médicas

Uso de ontologias – Aplicações

- *Ontology-based brokering*:
  - Funciona como *broker* entre sistemas heterogêneos.
  - Ontologias são utilizadas para facilitar a intermediação entre tarefas
- Armazenamento de conhecimento
  - Suporte a publicações em documentos "ontology-driven"
- Gerenciamento de conhecimento (KM)

Estar conectado não significa comunicação e o usuário não é a única interface importante!

- Na comunicação entre computadores, como saber o significado ou intenção do outro?

"Algumas vezes o mundo me parece fora de contexto"

Considerações sobre integração de ontologias

Jairo de Souza
COPPE/Sistemas

jairofsouza@yahoo.com.br

Introdução

- A criação de ontologias é uma tarefa custosa
- Quanto maior o domínio a ser modelado, mais custoso o processo de criação
- A reutilização de ontologias é uma tarefa presente em várias metodologias para construção
- Ontologias representam uma visão do mundo
  - Como agentes podem utilizar duas ontologias diferentes sobre o mesmo domínio?

Introdução

- Normalmente, ontologias representam domínios estáticos
- Alterações nessas ontologias são feitas com a ajuda de um engenheiro de ontologias
- A interoperabilidade entre duas ontologias pode ser realizado através de mapeamentos dos conceitos, porém essa tarefa também é custosa caso seja feita sem suporte computacional

O que é chamado de integração de ontologias?

- Na literatura, a palavra 'integração de ontologias' pode ser encontrada em três situações diferentes.

Integração de ontologias durante a construção de uma nova ontologia reusando outras ontologias disponíveis

- Existem ontologias disponíveis que são partes da ontologia a ser criada
- Estas ontologias satisfazem requisitos como nível de detalhes e granularidade, e são descritas numa linguagem adequada
- Estas ontologias podem ser estendidas, especializadas ou adaptadas

Integração de ontologias ao combinar ontologias diferentes sobre o mesmo assunto em uma única que unifica todas elas

- Queremos criar uma ontologia combinando idéias, conceitos, distinções, axiomas, etc, ou seja, unificar o conhecimento de ontologias diferentes de um mesmo domínio
- Existem diferenças entre essas ontologias, não só nas distinções básicas (hierarquia) mas também no modo como os termos são definidos.

Integração de ontologias em aplicações

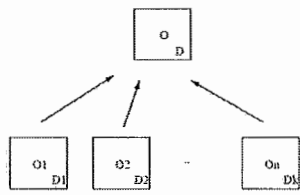
- Introdução de uma ou mais ontologias em uma aplicação que são compartilhadas entre vários softwares que usam uma ou mais ontologias para especificar ou implementar um sistema baseado em conhecimento (KBS)
- As ontologias são usadas ou reusadas para construir uma aplicação

O que é chamado de integração de ontologias?

- Para cada situação citada anteriormente, foi proposto os nomes, respectivamente:
  - Combinação de ontologias
  - Integração de ontologias e
  - Aplicação de ontologias

Integração de ontologias

- Temos, em uma mão, ontologias que são integradas (O1,O2,O3,O4) e, em outra mão, a ontologia resultante do processo de integração
- As ontologias integradas são aquelas que estão sendo reusadas. São partes da ontologia resultante e podem ser encaradas como módulos, ou "subontologias"

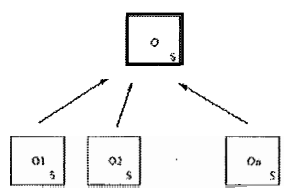


Integração de ontologias

- As ontologias integradas possuem domínios diferentes, mas podem haver sobreposições
- Quando a ontologia é reusada, os conceitos podem:
  - Permanecer intactos
  - Ser adaptados (ou modificados)
  - Ser especializados
  - Ser trocados por outros conceitos

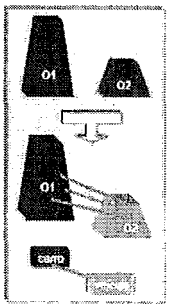
Combinação de ontologias

- Temos, em uma mão, um conjunto de ontologias (O1,O2,O3,O4) que vão ser combinadas, em outra mão, a ontologia resultante.
- O objetivo é criar um ontologia mais geral sobre o domínio ao reunir o conhecimento de várias ontologias sobre o mesmo assunto
- Na combinação, é difícil identificar, na ontologia resultante, regiões que foram preenchidas por uma certa ontologia.



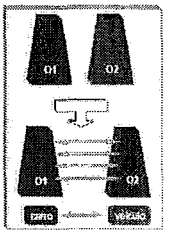
Outros mecanismos

- Alinhamento de ontologias
  - Tem-se como resultado as duas ontologias originais separadas, mas nestas são adicionadas às ligações entre seus termos equivalentes.
  - Estas ligações permitem que as ontologias alinhadas reusem as informações umas das outras.
  - O alinhamento normalmente é realizado quando as ontologias são de domínios complementares.



Outros mecanismos

- Mapeamento de ontologias
  - Tem-se como resultado uma estrutura formal com expressões que ligam os termos das duas ontologias
  - Este mapeamento pode ser usado para transferir instâncias de dados, esquemas de integração e de combinação, e outras tarefas similares



Erros – Nível de linguagem

- Ontologias escritas em linguagens diferentes
  - Sintaxe
    - RDF Schema <rdfs:Class ID='Cadeira'> [vs] Loom (defconcept Cadeira)
    - Solução: reescrever
  - Representação Lógica
    - disjoint A B [vs] B subclass-of (not A)
    - Solução: regras de tradução
  - Expressividade da linguagem
    - Ontologia em OWL e em RDF

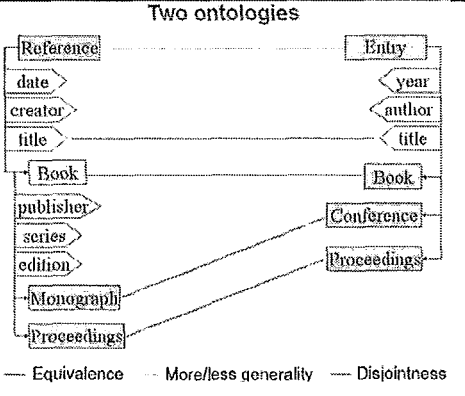
Erros – Nível ontológico

- Erros de conceitualização
  - Diferenças no modo em que o domínio é interpretado
  - Escopo: não exatamente as mesmas instâncias
  - Cobertura do domínio e granularidade
- Explicação
  - Diferenças no modo em que a conceitualização é especificada (estilos de modelagem diferentes)
  - Erros terminológicos:
    - Diferentes linguagens naturais
    - Termos homônimos (conceitos diferentes com o mesmo nome)

Erros – Nível ontológico

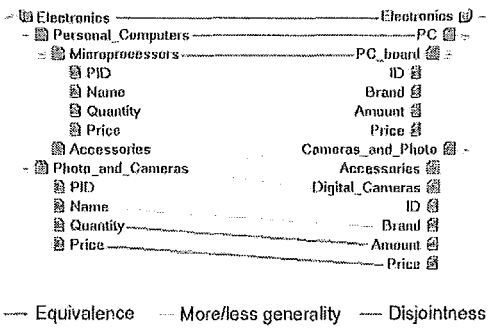
- Codificação
  - dd/mm/yyyy vs mm-dd-yy
  - Quilômetros vs milhas
  - Solução: Implementar transformações

Exemplos de Mapeamento



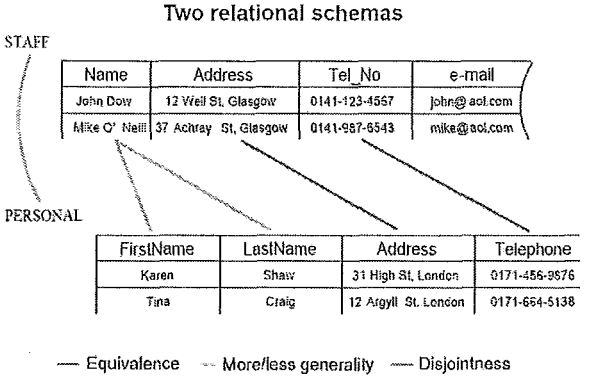
Exemplos de Mapeamento

Two XML schemas



Exemplos de Mapeamento

Two relational schemas



Considerações finais

- Os mecanismos para integração ainda são realizados manualmente
- Contudo, há uma grande necessidade de mecanismos semi-automáticos que dêem suporte a domínios emergentes: P2P, comunicação entre agentes, integração de serviços Web.
- Boa parte do processo ainda será manual e gerência de grupos (negociações, resolução de conflitos, etc) é essencial para ontologias compartilhadas por grupos distintos.

Anexo D

ONTOLOGIAS UTILIZADAS NOS ESTUDOS DE OBSERVAÇÃO

Abaixo se encontram as duas ontologias que foram entregues para os participantes do estudo de observação. A primeira ontologia descreve o domínio do paradigma de orientação a objetos e a segunda descreve o domínio de linguagens de programação.

(1) Domínio: orientação a objetos

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="OOLanguage">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Concept"/>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >oo language</rdfs:label>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="contém"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Restriction>
            <owl:onProperty>
              <owl:TransitiveProperty rdf:about="#contém"/>
            </owl:onProperty>
            <owl:allValuesFrom>
              <owl:Class rdf:ID="Heritage"/>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >linguagem orientada a objeto</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Boolean">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >boolean</rdfs:label>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >booleano</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Primitive"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Primitive">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DataType"/>
    </rdfs:subClassOf>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >primitivo</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Declaration">
```

```
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >statement</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >declaration</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >declaração</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Constructor">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >constructor</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >construtor</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Method"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TypeCast">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >cast</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >type cast</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Comment">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comment</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comentário</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="LogicalOperator">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Operator"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >logical operator</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador lógico</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Class">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Object"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="instancia"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Interface"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="implementa"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Inner_Class"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >classe</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >class</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ObjectReference"/>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Method"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Field"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Field">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >campo</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Data"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >property</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >atributo</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >propriedade</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#DataType"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="API">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >api</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Interface"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >application program interface</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Method">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="manipula"/>
      </owl:onProperty>
      <owl:allValuesFrom>
```

```
        <owl:Class rdf:ID="Variable"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Comment"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Declaration"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Data"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#manipula"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ObjectModel"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#DataType"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >função</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Command"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >método</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >method</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Inner_Class">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >inner class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >innerclass</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:ID="Iteration">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >loop</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >interação</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >iteration</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Command"/>
  </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="Qualifier">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >qualifier</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >qualificador</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >pra colocar static por exemplo</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Real">
  <rdfs:subClassOf rdf:resource="#Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >real</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="AbstractClass">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Superclass"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >classe abstrata</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >abstract class</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Operator">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operator</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="ExceptionTreatment">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exception treatment</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >tratamento de exceção</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#DataType">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >data type</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >tipo de dado</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="ArithmeticExpression">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >aritmethic expression</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >expressão aritmética</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:about="#Command">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >statement</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >command</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comando</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="RelationalOperator">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador relacional</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >relational operator</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Operator"/>
</owl:Class>
<owl:Class rdf:ID="Character">
  <rdfs:subClassOf rdf:resource="#Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >character</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >caractere</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Constant">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >constante</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >constant</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:about="#Variable">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >variable</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#DataType"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >variável</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Data"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ObjectReference">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >object reference</rdfs:label>
  <rdfs:subClassOf rdf:resource="#DataType"/>
</owl:Class>
<owl:Class rdf:ID="Scope">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Pode ser instanciado em Público, protegido e privado</rdfs:comment>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >scope</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >escopo</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Jump">
  <rdfs:subClassOf rdf:resource="#Command"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Para colocar: break, return...</rdfs:comment>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >jump</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >salto</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Superclass">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Subclass"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >super classe</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Class"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >superclass</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >super class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >superclasse</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Array">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >matriz</rdfs:label>
```

```
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >array</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >matrix</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >vector</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >vetor</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Primitive"/>
</owl:Class>
<owl:Class rdf:ID="Annotation">
  <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:about="#Heritage">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >heritage</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >herança</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >inherit</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#ObjectModel">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >object model</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Object">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >instância</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >object</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="eCriadoPor"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Constructor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Field"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >instance</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="eInstanciaDe"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Class"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Interface"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >objeto</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Text">
  <rdfs:subClassOf rdf:resource="#ObjectReference"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >text</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >texto</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Interface">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >interface</rdfs:label>
  <rdfs:subClassOf rdf:resource="#AbstractClass"/>
</owl:Class>
<owl:Class rdf:ID="Byte">
  <rdfs:subClassOf rdf:resource="#Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >byte</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Package">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Interface"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >pacote</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >package</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Class"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#API"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Data">
  <rdfs:subClassOf rdf:resource="#DataType"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >data</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >dado</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Polimorfism">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >polimorfismo</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >polimorfism</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Conditional">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >conditional</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >condicional</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Command"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >devio condicional</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="File">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >arquivo</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >file</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Subclass">
  <rdfs:subClassOf rdf:resource="#Class"/>
  <rdfs:subClassOf>
    <owl:Restriction>
```

- *Refinamento e estruturação da ontologia* - é nesta etapa que se objetiva chegar ao projeto definitivo, de modo que ele seja o mais coerente possível.

2.2.5 – Método SENSUS

Esta metodologia é baseada na ontologia SENSUS, que foi criada para fornecer uma estrutura conceitual ampla para desenvolver tradutores automatizados (SWARTOUT *et al.*, 1996). Ela é de ampla cobertura, possuindo tanto conceitos de alto nível quanto específicos, o que justifica sua denominação de ontologia de ampla cobertura, e tem mais de 70 mil conceitos organizados em hierarquias de acordo com seu nível de abstração. Ao invés de ter sido construída do zero, a SENSUS foi desenvolvida pela extração e união de informações de diferentes recursos eletrônicos, fazendo integração de ontologias por meio da fusão de diferentes ontologias.

A representação da ontologia da SENSUS pode ser descrita na forma de uma árvore, onde a raiz representa o termo mais abstrato e as folhas os mais específicos (BRAUNER *et al.*, 2003). A partir daí, quando uma ontologia vai ser construída para um determinado domínio, os seguintes passos devem ser seguidos:

- São extraídos os termos específicos do domínio e colocados como folhas.
- São incluídos todos os conceitos que vão desde as folhas até a raiz da SENSUS na nova ontologia.
- São adicionados os termos relevantes para o domínio.
- São incluídos como sub-árvore os nós que possuem diversos caminhos a partir deles.

O *software* utilizado para construir ontologias seguindo este método é o Ontosaurus (SWARTOUT *et al.*, 1996). Ontosaurus é um servidor que pode ser acessado através da Web e representa o conhecimento na linguagem de programação LOOM (DING, 2001), permitindo tradução para Ontolingua, KIF e C++.

2.2.6 – Metodologia On-To-Knowledge

A metodologia On-To-Knowledge (OTK) (SURE, STUDER, 2002; SURE, 2003) tem como principal interesse a construção de grandes sistemas de gestão do conhecimento. O processo de desenvolvimento da ontologia nesta metodologia consiste das fases seguintes.

```
      <owl:allValuesFrom rdf:resource="#Method"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="define"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >sub classe</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Superclass"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="herdaDe"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="herda"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Field"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Superclass"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="podeSer"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >subclass</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Method"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="redefinir"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#define"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Field"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#herda"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Method"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >sub class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >subclasse</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Exception">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >except</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exceção</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exception</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:ID="ClassLibrary">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >library</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Package"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >biblioteca</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >class library</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#API"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Integer">
  <rdfs:subClassOf rdf:resource="#Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >integer</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >inteiro</rdfs:label>
</owl:Class>
<owl:ObjectProperty rdf:about="#herda">
  <rdfs:domain rdf:resource="#Subclass"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Method"/>
        <owl:Class rdf:about="#Field"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#eInstanciaDe">
  <rdfs:range rdf:resource="#Class"/>
  <rdfs:domain rdf:resource="#Object"/>
  <owl:equivalentProperty>
    <owl:InverseFunctionalProperty rdf:about="#instancia"/>
  </owl:equivalentProperty>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#instancia"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#define">
  <rdfs:domain rdf:resource="#Subclass"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Method"/>
        <owl:Class rdf:about="#Field"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#eCriadoPor">
  <rdfs:domain rdf:resource="#Object"/>
  <rdfs:range rdf:resource="#Constructor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#implementa">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ClassLibrary"/>
        <owl:Class rdf:about="#Package"/>
        <owl:Class rdf:about="#Class"/>
        <owl:Class rdf:about="#Object"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class>
```

```
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#API"/>
          <owl:Class rdf:about="#Interface"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#herdaDe">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Superclass"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#manipula">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Data"/>
          <owl:Class rdf:about="#Variable"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain rdf:resource="#Method"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#redefinir">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Method"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#podeSer">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Superclass"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="eSerilizavel">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:domain rdf:resource="#Object"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#contém">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Field"/>
          <owl:Class rdf:about="#Heritage"/>
          <owl:Class rdf:about="#Method"/>
          <owl:Class rdf:about="#Package"/>
          <owl:Class rdf:about="#Command"/>
          <owl:Class rdf:about="#Declaration"/>
          <owl:Class rdf:about="#Comment"/>
          <owl:Class rdf:about="#DataType"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Class"/>
          <owl:Class rdf:about="#OOLanguage"/>
          <owl:Class rdf:about="#ClassLibrary"/>
          <owl:Class rdf:about="#Package"/>
          <owl:Class rdf:about="#API"/>
          <owl:Class rdf:about="#Object"/>
          <owl:Class rdf:about="#Method"/>
          <owl:Class rdf:about="#Variable"/>
          <owl:Class rdf:about="#Field"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:TransitiveProperty>
  <owl:InverseFunctionalProperty rdf:about="#instancia">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:equivalentProperty rdf:resource="#eInstanciaDe"/>
    <rdfs:range rdf:resource="#Object"/>
    <owl:inverseOf rdf:resource="#eInstanciaDe"/>
    <owl:sameAs rdf:resource="#eInstanciaDe"/>
    <rdfs:domain rdf:resource="#Class"/>
  </owl:InverseFunctionalProperty>
</rdf:RDF>
```

(2) Domínio: Linguagens de programação

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rss="http://purl.org/rss/1.0/"
    xmlns="http://monet.nag.co.uk/owl#"
    xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://monet.nag.co.uk/owl">
  <owl:Class rdf:ID="Software">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >General class of all computer software</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="C">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ProgrammingLanguage"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="OO-Paradigm"/>
  <owl:Class rdf:ID="Subclass">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Class"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Fortran95">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Fortran"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Aldor_Libraries">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Libraries developed as Aldor Projects</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:ID="ImplementationLanguage"/>
            </owl:onProperty>
            <owl:someValuesFrom>
              <owl:Class rdf:ID="Aldor"/>
            </owl:someValuesFrom>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty rdf:about="#ImplementationLanguage"/>
            </owl:onProperty>
            <owl:allValuesFrom>
              <owl:Class rdf:about="#Aldor"/>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Software"/>
  </owl:Class>
  <owl:Class rdf:ID="Maple">
    <rdfs:subClassOf rdf:resource="#Software"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Maple</owl:hasValue>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="name"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="creator"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Maplesoft</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CPlusPlus">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ProgrammingLanguage"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Interface">
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Superclass">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Class"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Mathematica4">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >v4.2</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Mathematica"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Aldor_Sumit">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithms for manipulating and solving linear ordinary differential and difference
     equations and systems.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Aldor_Libraries"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#creator"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Manuel Bronstein</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Sumit</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Class">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="implementa"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Interface"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Datatypes">
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Complex">
  <rdfs:subClassOf rdf:resource="#Datatypes"/>
</owl:Class>
<owl:Class rdf:ID="Aldor_Algebra">
```

```
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >v1.0.1</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Algebra</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Aldor_Libraries"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >General-purpose computer algebra library in Aldor</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Maple9">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Command the brilliance of a thousand mathematicians</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >v9.0</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Maple"/>
</owl:Class>
<owl:Class rdf:ID="Package">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:ID="contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Interface"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Class"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:about="#ProgrammingLanguage">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >General class of all programming languages</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Comment">
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Maple8">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >v8</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Maple"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Maple Computer Algebra System</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Variable">
```

```
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Datatypes"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Fortran90">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Fortran"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Heritage">
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
</owl:Class>
<owl:Class rdf:ID="Fortran77">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Fortran"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="NAG_C_Library">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >NAG C Library</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#creator"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Numerical Algorithms Group</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom rdf:resource="#C"/>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#ImplementationLanguage"/>
          </owl:onProperty>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#ImplementationLanguage"/>
          </owl:onProperty>
          <owl:allValuesFrom rdf:resource="#C"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Numerical Algorithms implemented in C</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Software"/>
</owl:Class>
<owl:Class rdf:ID="NAG_C_Library_7">
  <rdfs:subClassOf rdf:resource="#NAG_C_Library"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >v7</owl:hasValue>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#version"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Fortran">
```

```
  <rdfs:subClassOf rdf:resource="#ProgrammingLanguage"/>
</owl:Class>
<owl:Class rdf:ID="FinalClass">
  <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:ID="AnonymousClass">
  <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:ID="Java">
  <rdfs:subClassOf rdf:resource="#ProgrammingLanguage"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Restriction>
          <owl:onProperty>
            <owl:TransitiveProperty rdf:about="#contém"/>
          </owl:onProperty>
          <owl:allValuesFrom rdf:resource="#Heritage"/>
        </owl:Restriction>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#OO-Paradigm"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Aldor">
  <rdfs:subClassOf rdf:resource="#ProgrammingLanguage"/>
</owl:Class>
<owl:Class rdf:ID="Library">
  <rdfs:subClassOf rdf:resource="#OO-Paradigm"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Package"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Annotation">
  <rdfs:subClassOf rdf:resource="#Comment"/>
</owl:Class>
<owl:Class rdf:about="#Mathematica">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#name"/>
      </owl:onProperty>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Mathematica</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The way the world calculates</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Software"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Wolfram Research</owl:hasValue>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#creator"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Constant">
  <rdfs:subClassOf rdf:resource="#Variable"/>
</owl:Class>
```

```
  <owl:Class rdf:ID="Primitive">
    <rdfs:subClassOf rdf:resource="#Datatypes"/>
  </owl:Class>
  <owl:Class rdf:ID="Lisp">
    <rdfs:subClassOf rdf:resource="#ProgrammingLanguage"/>
  </owl:Class>
  <owl:Class rdf:ID="Mathematica5">
    <rdfs:subClassOf rdf:resource="#Mathematica"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:about="#version"/>
        </owl:onProperty>
        <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >v5</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#implementa">
    <rdfs:range rdf:resource="#Interface"/>
    <rdfs:domain rdf:resource="#Class"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ImplementationLanguage">
    <rdfs:range rdf:resource="#ProgrammingLanguage"/>
    <rdfs:domain rdf:resource="#Software"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="#name">
    <rdfs:domain rdf:resource="#Software"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#creator">
    <rdfs:domain rdf:resource="#Software"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#contém">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Heritage"/>
          <owl:Class rdf:about="#Datatypes"/>
          <owl:Class rdf:about="#Interface"/>
          <owl:Class rdf:about="#Package"/>
          <owl:Class rdf:about="#Class"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Variable"/>
          <owl:Class rdf:about="#Library"/>
          <owl:Class rdf:about="#Package"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:TransitiveProperty>
  <owl:FunctionalProperty rdf:about="#version">
    <rdfs:domain rdf:resource="#Software"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

(3) Domínio: orientação a objetos (com estrutura e nomes levemente alterados arbitrariamente)

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
```

```
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="OOLanguage">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Concept"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >oo language</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Restriction>
          <owl:onProperty>
            <owl:TransitiveProperty rdf:ID="contém"/>
          </owl:onProperty>
          <owl:allValuesFrom>
            <owl:Class rdf:ID="Heritage"/>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >linguagem orientada a objeto</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Int">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >integer</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >inteiro</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Datatype_Primitive"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Boolean">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >boolean</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >booleano</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Datatype_Primitive"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Data_Type">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >tipo de dado</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >data type</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Constructor">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >constructor</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >construtor</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Method"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="TypeCast">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >cast</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >type cast</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Comment">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comment</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comentário</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Abstract">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >abstract class</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Superclass"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >classe abstrata</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="LogicalOperator">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Operator"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >logical operator</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador lógico</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Class">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Method"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >classe</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Reference_Object"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Field"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="implementa"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Interface"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Innerclass"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Field">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >campo</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >property</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >atributo</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >propriedade</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Data"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Data_Type"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="API">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >api</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Interface"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >application program interface</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Method">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="manipula"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Variable"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Data"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#manipula"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Declare"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Command"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Comment"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ObjectModel"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >função</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >método</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >method</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Data_Type"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Iteration">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >interação</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >loop</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >iteration</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Command"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Qualifier">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >qualifier</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >qualificador</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >pra colocar static por exemplo</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#Datatype_Primitive">
  <rdfs:subClassOf rdf:resource="#Data_Type"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >primitivo</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Real">
  <rdfs:subClassOf rdf:resource="#Datatype_Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >real</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Operator">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operator</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Treatment">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >tratamento de exceção</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exception treatment</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="ArithmeticExpression">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >aritmethic expression</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >expressão aritmética</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:about="#Command">
  <rdfs:subClassOf rdf:resource="#Concept"/>
```

```
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >statement</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >command</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >comando</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="RelationalOperator">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >operador relacional</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >relational operator</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Operator"/>
</owl:Class>
<owl:Class rdf:ID="Constant">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >constante</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >constant</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:about="#Variable">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >variable</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Data"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Data_Type"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >variável</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Scope">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Pode ser instanciado em Público, protegido e privado</rdfs:comment>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >scope</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >escopo</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:ID="Jump">
  <rdfs:subClassOf rdf:resource="#Command"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Para colocar: break, return...</rdfs:comment>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >jump</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >salto</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Superclass">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >super classe</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Class"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >superclass</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >super class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >superclasse</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Innerclass">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >inner class</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >innerclass</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Class"/>
</owl:Class>
<owl:Class rdf:ID="Array">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >matriz</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >array</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >matrix</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >vector</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >vetor</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Datatype_Primitive"/>
</owl:Class>
<owl:Class rdf:ID="Exceptions">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exception</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >except</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >exceção</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Char">
  <rdfs:subClassOf rdf:resource="#Datatype_Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >caractere</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >character</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Heritage">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >heritage</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >herança</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >inherit</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#ObjectModel">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >object model</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Declare">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >declaration</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >statement</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >declaração</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Text">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Reference_Object"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >text</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >texto</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Interface">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >interface</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Abstract"/>
</owl:Class>
<owl:Class rdf:ID="Byte">
  <rdfs:subClassOf rdf:resource="#Datatype_Primitive"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >byte</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Package">
```

```
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Class"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#API"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Interface"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >pacote</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >package</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
</owl:Class>
<owl:Class rdf:about="#Data">
  <rdfs:subClassOf rdf:resource="#Data_Type"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >data</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >dado</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Reference_Object">
  <rdfs:subClassOf rdf:resource="#Data_Type"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >object reference</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Conditional">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >conditional</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >condicional</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Command"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >devio condicional</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="File">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >arquivo</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >file</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Polimorfism">
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >polimorfismo</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >polimorfism</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Subclass">
  <rdfs:subClassOf rdf:resource="#Class"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="podeSer"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Superclass"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >sub class</rdfs:label>
```

```
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >sub classe</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Superclass"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="herdaDe"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Method"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="define"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >subclasse</rdfs:label>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >subclass</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="redefinir"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Method"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ClassLibrary">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >library</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#API"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#implementa"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >biblioteca</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Concept"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Package"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#contém"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >class library</rdfs:label>
</owl:Class>
<owl:ObjectProperty rdf:ID="herda">
  <rdfs:domain rdf:resource="#Subclass"/>
  <rdfs:range rdf:resource="#Method"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="eInstanciaDe">
  <rdfs:range rdf:resource="#Class"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#define">
  <rdfs:domain rdf:resource="#Subclass"/>
  <rdfs:range rdf:resource="#Method"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="eCriadoPor">
  <rdfs:range rdf:resource="#Constructor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#implementa">
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#API"/>
        <owl:Class rdf:about="#Interface"/>
```

```
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#ClassLibrary"/>
          <owl:Class rdf:about="#Package"/>
          <owl:Class rdf:about="#Class"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#herdaDe">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Superclass"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#manipula">
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Data"/>
          <owl:Class rdf:about="#Variable"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain rdf:resource="#Method"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#redefinir">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Method"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#podeSer">
    <rdfs:domain rdf:resource="#Subclass"/>
    <rdfs:range rdf:resource="#Superclass"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="eSerilizavel">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#contém">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Class"/>
          <owl:Class rdf:about="#ClassLibrary"/>
          <owl:Class rdf:about="#Package"/>
          <owl:Class rdf:about="#Variable"/>
          <owl:Class rdf:about="#Field"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Field"/>
          <owl:Class rdf:about="#Data_Type"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:TransitiveProperty>
</rdf:RDF>
```

Anexo E

ARQUIVO DE MAPEAMENTO GERADO NO ESTUDO DE OBSERVAÇÃO

Abaixo se encontra o arquivo de mapeamento gerado pelo mediador ao final do estudo de observação, tendo como fonte as decisões tomadas pelos participantes em cada rodada de negociação, conforme sugere o modelo GNoSIS.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapFile>
  <ontologies>
    <ontology
      url="http://labbd.cos.ufrj.br/gnosis/ontos/objorientontology.owl
      " alias="objorientontology" />
    <ontology
      url="http://labbd.cos.ufrj.br/gnosis/ontos/programminglangs.owl"
      alias="programminglangs" />
  </ontologies>
  <mappings>
    <mapping type="equivalency">
      <source>objorientontology#ClassLibrary</source>
      <target>programminglangs#Library</target>
    </mapping>
    <mapping type="disjoint">
      <source>objorientontology#Constant</source>
      <target>programminglangs#Variable</target>
    </mapping>
    <mapping type="equivalency">
      <source>objorientontology#Interface</source>
      <target>programminglangs#Interface</target>
    </mapping>
    <mapping type="contains">
      <source>objorientontology#Class</source>
      <target>programminglangs#FinalClass</target>
    </mapping>
    <mapping type="contains">
      <source>objorientontology#Class</source>
      <target>programminglangs#AnonymousClass</target>
    </mapping>
    <mapping type="disjoint">
      <source>objorientontology#Comment</source>
      <target>programminglangs#Annotation</target>
    </mapping>
    <mapping type="disjoint">
      <source>objorientontology#Annotation</source>
      <target>programminglangs#Comment</target>
    </mapping>
    <mapping type="contained">
      <source>objorientontology#Annotation</source>
      <target>programminglangs#Class</target>
    </mapping>
    <mapping type="contained">
```

```
  <source>objorientontology#API</source>
  <target>programminglangs#Library</target>
</mapping>
<mapping type="contained">
  <source>objorientontology#Package</source>
  <target>programminglangs#Library</target>
</mapping>
<mapping type="contains">
  <source>objorientontology#Concept</source>
  <target>programminglangs#Heritage</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Subclass</source>
  <target>programminglangs#Subclass</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Superclass</source>
  <target>programminglangs#Superclass</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#DataType</source>
  <target>programminglangs#Datatypes</target>
</mapping>
<mapping type="contained">
  <source>objorientontology#Inner_Class</source>
  <target>programminglangs#Class</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Class</source>
  <target>programminglangs#Class</target>
</mapping>
<mapping type="contained">
  <source>objorientontology#Data</source>
  <target>programminglangs#Datatypes</target>
</mapping>
<mapping type="contained">
  <source>objorientontology#Text</source>
  <target>programminglangs#Datatypes</target>
</mapping>
<mapping type="overlap">
  <source>objorientontology#Concept</source>
  <target>programminglangs#OO-Paradigm</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Package</source>
  <target>programminglangs#Package</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Comment</source>
  <target>programminglangs#Comment</target>
</mapping>
<mapping type="equivalency">
  <source>objorientontology#Primitive</source>
  <target>programminglangs#Primitive</target>
</mapping>
<mapping type="no equivalency">
  <source>objorientontology#Primitive</source>
  <target>programminglangs#Complex</target>
</mapping>
<mapping type="contains">
  <source>objorientontology#API</source>
```

```
      <target>programminglangs#NAG_C_Library</target>
    </mapping>
    <mapping type="contains">
      <source>objorientontology#API</source>
      <target>programminglangs#NAG_C_Library_7</target>
    </mapping>
  </mappings>
</mapFile>
```

Anexo F

PROCEDIMENTO PARA INSTALAÇÃO DOS MÓDULOS DO GNoSIS E REPRODUÇÃO DOS RESULTADOS OBTIDOS

Para utilização dessa implementação e replicação dos resultados do cálculo de similaridade apresentados nesta dissertação, deve-se recuperar o códigos-fonte da implementação do GNoSIS que estão disponíveis no CVS do LabBD (Laboratório de Banco de Dados) do PESC. Para acessar o CVS, utilize a string de conexão :pserver:gnosis@labbd.cos.ufrj.br:/COE. Para se autenticar, utilize a senha "gnosis123". O módulo MN corresponde ao código-fonte implementado nessa dissertação.

Para executar o módulo de cálculo de similaridade, deve-se executar o método "main" da classe MN.java. As ontologias a serem comparadas devem ser copiadas para o diretório-repositório que é configurado na variável PATH_ONTOLOGIES do arquivo gnosis.properties, localizado no pacote br.ufrj.meaning. Os pesos-padrões são também configurados neste arquivo. O conteúdo do arquivo está descrito abaixo:

```
## Weights
W_CFV_NAME = 0.5
W_CFV_PROPERTY = 0.5

W_PROPERTY_VALUE = 0.4
W_PROPERTY_NAME = 0.6

W_CONTEXT_FREE_VARIABLES = 0.7
W_HIERARCHY = 0.3

W_PARENT = 0.5
W_CHILDREN = 0.5

## Penalties
PROPERTY_PENALTY = 0.3
CHILDREN_PENALTY = 0.3

# Bounds
THRESHOLD = 0.6

# Paths
PATH_ONTOLOGIES = D:\\Java Development\\Projects\\MN\\ontologies
```

O projeto foi desenvolvido utilizando-se as facilidades do *plugin* Maven versão 2.0.4. Para detalhes sobre instalação e utilização do Maven, visite o endereço eletrônico http://maven.apache.org/.

A instalação e os códigos-fonte do Compendium estão disponíveis no endereço eletrônico http://www.compendiuminstitute.org/download/download.htm.

É necessário a instalação do banco de dados MySQL para registrar os dados das negociações. A instalação é simples e pode ser obtida no endereço eletrônico http://dev.mysql.com/downloads/.

A interface do módulo de cálculo de similaridades é basicamente em modo de linha de comando. Contudo, ao digitar o comando "help" um menu de ajuda é exibido com todos os comandos disponíveis, assim não há grandes dificuldades em utilizar o módulo, uma vez que este esteja devidamente instalado.