UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2008.2

Mapeamento Semântico de Ontologias no SPEED

TRABALHO DE GRADUAÇÃO

Aluno: Thiago Pachêco Andrade Pereira (tpap@cin.ufpe.br)
Orientadora: Profa. Ana Carolina Salgado (acs@cin.ufpe.br)

Dezembro 2008

UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

2008.2

THIAGO PACHÊCO ANDRADE PEREIRA

Mapeamento Semântico de Ontologias no SPEED

Trabalho apresentado à graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora – Ana Carolina Salgado (acs@cin.ufpe.br)

Recife, Dezembro de 2008

Dedico,

A minha família, em especial minha mãe.
Aos meus amigos.

Agradecimentos

Agradeço a minha família, principalmente a minha mãe que sempre lutou pra tornar esse momento possível. A minha orientadora Carol pela ajuda tanto técnica no projeto, tanto como amiga em momentos pessoais difíceis durante o projeto. A Damires e Carlos, pela grande ajuda durante o desenvolvimento deste trabalho. E aos meus amigos e parceiros de projeto, que tornaram o desenvolvimento deste trabalho mais prazeroso.

Resumo

Com o aumento na quantidade de dados em fontes heterogêneas e distribuídas, a necessidade de informação relevante aumenta, e faz surgir aplicações que buscam integrar e fornecer informação consistente. Sistemas PDMS (*Peer Data Management System*) são um exemplo desse tipo de sistema, e têm como objetivo integrar dados em um ambiente com fontes autônomas, heterogêneas e dinâmicas.

O SPEED (*Semantic PEEr-to-Peer Data Management System*) é um sistema que propõe adicionar aos tradicionais sistemas PDMS uma análise semântica, de forma a aumentar a qualidade na integração dos dados. Os *peers* se organizam em uma topologia *super peer*, onde são agrupados em *clusters* semânticos de acordo com seu domínio, formando uma hierarquia de *peers*. No topo dessa hierarquia estão os *peers* semânticos, que representam um domínio ou comunidade semântica. Dentro de cada *cluster* existe um *peer* de integração que é responsável pela integração de dados e processamento de consultas dos *peers* de dados que compõem o *cluster*. No SPEED, os esquemas das bases de dados são representados por ontologias.

Este trabalho visa à geração de uma medida global de correspondência entre ontologias, baseado nas correspondências de termos geradas por um *matcher* de ontologias e nos relacionamentos semânticos existentes entre os termos. Esta medida é a base para tarefas importantes no funcionamento do SPEED. Em um primeiro momento, na entrada de um *peer* na rede, onde é feita a identificação de qual comunidade ele fará parte. E, posteriormente, no momento da submissão de uma consulta, onde as correspondências são necessárias para a reformulação das consultas a serem enviadas a outros *peers*.

Sumário

Lista de figuras

Lista de quadros

1. Introdução

Neste primeiro capítulo, será feito uma contextualização do ambiente do sistema SPEED, mostrando as necessidades atuais, e motivação que tornam este trabalho significante.

1.1 Motivação

O volume de dados e o número de fontes de dados vêm crescendo. Ao mesmo tempo, todas essas fontes estão cada vez mais conectadas, principalmente fazendo uso da rede *peer-to-peer*, permitindo fácil comunicação entre elas. O que deveria ser bom torna-se um problema, pela dificuldade de encontrar informação relevante no meio desse mar de informação.

Nesse cenário, várias pesquisas têm sido desenvolvidas, e uma forte corrente de pesquisa são os *Peer Data Management Systems* (PDMS) [1]. Sistemas PDMS permitem acesso transparente a bases de dados heterogêneas e distribuídas, sem a necessidade de um controle centralizado. Cada *peer* pode submeter uma consulta, e a reformulação de consulta para outros *peers* da rede, é feita a partir de mapeamentos entre cada par de *peers*.

Estendendo essa idéia, surge o trabalho de Xiao [2], que propõe o uso de ontologias como forma de representação de esquemas de um PDMS, surgindo o novo conceito *Ontology-based Data Management Systems* (OPDMS). O sistema SPEED é um OPDMS, e faz uso de ontologias na representação de seus esquemas. Com isto, surge a necessidade de efetuar várias operações sobre ontologias.

Basicamente há duas situações onde estas operações são mais importantes. O SPEED agrupa *peers* semelhantes em *clusters*, e no momento da entrada de um *peer* no sistema, é necessário idenfiticar o grau de semelhança entre a ontologia que representa aquele *cluster*, e a ontologia que representa o esquema do *peer* entrante. Para isso é necessário a geração de alguma medida de similaridade entre essas ontologias para decidir se o *peer* faz parte de um determinado *cluster*. Em outro momento, quando o *peer* já se

encontra na rede, uma consulta pode ser submetida, e para se integrar dados de outras bases, faz-se necessário a geração de mapeamentos entre os esquemas (representados por ontologias).

1.2 Objetivo

O trabalho aqui desenvolvido, busca suprir essas necessidades, com o desenvolvimento de um *Ontology matching* semântico, que é um projeto que colabora com as pesquisas feitas em duas teses de doutorado [3,4] e que será usado dentro de outro trabalho de graduação [5], que vem sendo desenvolvido neste mesmo período.

1.3 Estrutura do documento

Os próximos capítulos deste documento seguem a seguinte organização:

- Capítulo 2: Descreve os conceitos básicos para entender um *Ontology matching*, e conceitos básicos relacionados a esse tipo de ferramenta.
- Capítulo 3: Descreve o sistema SPEED, sua arquitetura e funcionamento. Também detalha o uso de ontologias dentro do sistema, desde o seu uso na representação de esquemas, até a necessidade da geração de alinhamentos.
- Capítulo 4: Descreve a solução implementada para o SPEED, incluindo todo o processo de execução e como cada módulo é importante para o sistema.
- Capítulo 5: Descreve as conclusões e dificuldades encontradas durante o desenvolvimento deste trabalho. Também são citados possíveis trabalhos futuros a serem realizados sobre a ferramenta implementada.

2. Ontologias: conceitos e ferramentas

Este capítulo apresentará conceitos importantes para o entendimento deste trabalho, começando por definições necessárias para o entendimento de um *ontology matching*, depois explicando o seu funcionamento, e por fim falando de ferramentas de *matching* estudadas e utilizadas durante o projeto.

Também é feito um estudo sobre outro tipo de ferramenta, os raciocinadores (ou *reasoners*), que inferem novo conhecimento sobre as informações já existentes na ontologia, e também é necessário para o desenvolvimento do *Ontology matching* semântico.

2.1 Definições

Ontologia é um conceito desenvolvido pela comunidade de inteligência artificial com a intenção de facilitar o compartilhamento e reuso [6] de conhecimento. Elas conseguem descrever um domínio específico que se deseja modelar. O resultado dessa modelagem é uma espécie de hierarquia de conceitos, propriedades e instâncias. A linguagem de representação de ontologias que é padrão da W3C é chamada de OWL (*Web Ontology Language*) [7], e é uma extensão de outra linguagem chamada RDF (*Resource Description Framework*) [8]. O OWL tem uma estrutura parecida com XML, porém é usada para representar os relacionamentos entre as entidades dentro de uma ontologia. Esses relacionamentos são expressos através de axiomas, que são uma tripla formada por sujeito, predicado e objeto, onde geralmente o sujeito e objeto são um conceito ou predicado, e o predicado é o tipo de relacionamento entre o sujeito e objeto.

Com a existência de diversas ontologias, principalmente em OWL, sobre os mais variados domínios, e também a impossibilidade de gerar uma ontologia única que descreva todo o conhecimento existente, surge à necessidade de efetuar operações sobre ontologias, de forma a extrair mais conhecimento para algum fim. Como exemplo de operações possíveis de serem feitas sobre uma ontologia deve-se citar: *merging* e *matching* (também chamado de *mapping*).

Matching é uma operação que gera um conjunto de alinhamentos entre duas ontologias ou mais. Um alinhamento consiste de um relacionamento ou valor de similaridade encontrado entre um conceito de uma ontologia, com um conceito em outra ontologia. *Merging* é o processo de gerar uma ontologia única coerente a partir de duas ou mais ontologias diferentes, mas sobre um mesmo assunto [9].

2.2 *Ontology Matching*

Um *ontology matching*, como o próprio nome sugere, efetua um *matching* entre ontologias, que basicamente consiste de uma operação que dado duas ou mais ontologias de entrada, gera como resultado um conjunto de alinhamentos entre elas. Alinhamentos que podem ser úteis para integrar informações e dados das duas ontologias, identificar a semelhança entre elas através da quantidade de conceitos em comum, ou mesmo como entrada para uma operação de *merging*. Para a identificação desses alinhamentos, um *ontology mathing* executa alguns tipos de técnicas sobre os conceitos das ontologias de entradas.

- *Análise linguística:* Nesse primeiro tipo de análise, é considerado o nome dos conceitos e a proximidade entre eles (diferença de apenas alguns caracteres, análise de possíveis radicais, etc). A semântica da palavra pode ser levada em consideração fazendo o uso de um *thesaurus* (dicionário de sinônimos), que pode ajudar a melhorar a qualidade dos *matchings*, porém aqui a semântica é só em relação a sinônimos.
- *Análise estrutural:* A análise estrutural leva em consideração a semelhança estrutural entre dois conceitos, considerando conceitos próximos a eles na hierarquia de suas respectivas ontologias, ou na proximidade a um termo equivalente.

Ainda sobre os alinhamentos, pode-se configurar a cardinalidade de alinhamentos a serem gerados entre as ontologias. Em geral as opções de cardinalidade são:

- *1:1*: Para cada elemento da primeira ontologia, há um alinhamento em relação a apenas outro elemento da segunda ontologia, que pode ser o alinhamento de maior similaridade entre os alinhamentos gerados com cardinalidade 1:N.
- *1:N*: Para cara elemento da primeira ontologia, há um alinhamento com valor de similaridade para outros elementos na segunda ontologia. Neste ponto pode-se fazer o uso de um limiar para eliminar alinhamentos que tenham valores de similaridade muito baixos.

Quando efetuamos um *matching*, o objetivo é encontrar semelhanças entre as ontologias. Porém, muitas vezes, mesmo ontologias com mesmo domínio descrevem o mesmo de formas bem diferentes. Há diversas formas de heterogeneidade que podem ocorrer entre ontologias. Segue algumas descritas em [3,4].

- *Heterogeneidade sintática*: Ocorre quando duas ontologias são descritas com duas linguagens de representação de ontologias diferentes. Por exemplo, uma ontologia descrita em OWL, e outra em F-Logic [10]. Este tipo de heterogeneidade não ocorre no SPEED, pois todas as ontologias são descritas em OWL.
- *Heterogeneidade de termos*: Ocorre quando para se referir a uma mesma entidade, são usados termos diferentes, seja por se usar línguas diferentes, linguagens de domínio diferentes, ou mesmo sinônimo.
- *Heterogeneidade conceitual ou semântica*: Ocorre quando axiomas diferentes são utilizados para descrever um mesmo fato em ontologias diferentes.

2.3 *Matchers* existentes

Durante o desenvolvimento deste trabalho, alguns *ontology matcher*s foram estudados, em um primeiro momento para entendimento do

funcionamento desse tipo de ferramenta, e depois pela necessidade da escolha de uma delas para o uso dentro do nosso projeto.

O COMA++ [11] foi o primeiro a ser estudado, e demonstrou ser uma ferramenta poderosa e com bons recursos, além da existência de muitos artigos descrevendo em detalhes seu funcionamento. Fornece recursos para *matching* e *merging* de ontologias. No caso do *matching*, eram consideradas as análises linguística e estrutural.

O HMatch [12] é um projeto desenvolvido como um framework para o HELIOS (Sistema OPDMS que possui semelhanças com o SPEED). Executa *matchings* usando análises linguística, estrutural e contextual. Outra ferramenta estudada foi o AlignmentAPI [13], que é uma API desenvolvida para realizar *matching* de ontologias.

Para o uso dentro do SPEED, a ferramenta deveria atender a alguns requisitos, dentre eles:

- *Suporte a chamada por console*: Para ser usado dentro do projeto do ontology *matching*, era necessária uma ferramenta que pudesse ser executada através do console ou através de uma API.
- *Geração de alinhamentos 1-N*: A saída dos alinhamentos da etapa (1) devem ser 1-N, então era necessária a geração desse tipo de alinhamento pela ferramenta escolhida.
- *Geração de uma medida de similaridade entre conceitos*: Não só os alinhamentos eram necessários, como também uma medida de quão similar seriam dois conceitos que fazem parte de um alinhamento.
- *Documentação*: Outro fator importante era que a documentação fosse clara em relação ao uso da ferramenta, para facilitar o seu uso.
- *Linguagem da ferramenta*: Um fator não tão importante se a ferramenta possibilitasse chamada em console, mas importante se seu uso fosse feito através de uma API.

As ferramentas foram testadas primeiro na interface que ofereciam (gráfica ou console), analisando os resultados gerados de forma visual ou em arquivo. Depois foi tentado o uso delas dentro de um projeto com código em

Java, que executava a ferramenta e conseguia os resultados. Feito o estudo comparativo, os resultados são mostrados no Quadro 1:

Matcher\Função	Suporte a chamada por console.	Geração de alinhamentos 1-N	Geração de uma medida de similaridade entre conceitos	Documentação	Linguagem da ferramenta.
COMA++	Não	Sim	Sim	Boa	Java
HMatch	Sim	Sim	Sim	Razoável	Java
AlignmentAPI	Sim	-	Sim	Razoável	Java

Quadro 1 - Quadro comparativo entre ferramentas.

Uma das primeiras necessidades para a escolha de uma ferramenta era que ela pudesse ser chamada através de console ou via uma API. E por este motivo, o COMA++, apesar de ser uma ferramenta já mais utilizada e que segundo pesquisas tem resultados bem satisfatórios, não foi a escolhida. Toda a manipulação de ontologias no COMA++ é feita através de uma interface gráfica da ferramenta, impossibilitando o uso desta no nosso trabalho.

A segunda alternativa foi o HMatch, uma ferramenta que também possibilita o *matching* entre ontologias, só que por console, podendo ser chamada dentro do processo do *Ontology matching* semântico. Com ela conseguimos gerar os alinhamentos *1:N* necessários da etapa (1) da especificação.

Outra alternativa que surgiu foi o uso do AlignmentAPI, por também possibilitar a chamada através de console ou também através de uma API. Porém, com essa ferramenta não se conseguiu gerar os alinhamentos *1:N*, pois não há nenhum arquivo de configuração que se possa alterar e, em sua documentação, nada é explicado a respeito. Dessa forma o HMatch foi a ferramenta escolhida para essa versão da implementação por satisfazer os requisitos de forma satisfatória.

2.5 Usos de *matching* de ontologias

O *matching* de ontologias é uma operação vital para alguns tipos de sistema. Um exemplo deles são os sistemas de integração de informação. Inicialmente o *matching* era usado em cenários menores, como a integração de esquemas em um caso onde se necessite juntar dados contidos em dois bancos de dados que possuam informações de domínios semelhantes, mas com esquemas diferentes. A identificação das semelhanças entre entidades e atributos pode ser feita com o auxílio de um *ontology matching*, auxiliando o trabalho de integração de informações.

Em um passo seguinte, *matching* de ontologias começaram a ser usados para fazer a integração dos próprios dados. Nesse cenário eram usadas ontologias locais para representar cada base de dados, e o acesso a informações era feito através de consultas sobre uma ontologia virtual, que funcionava como uma *view* sobre todas as bases. Quando uma consulta era executada sobre a ontologia virtual, operações de *matching* eram realizadas entre a ontologia virtual e as ontologias locais para se identificar correspondências e assim poder se reescrever as consultas de acordo com o esquema local de cada base.

Com o aumento do número de bases de dados e o crescimento das redes *peer-to-peer*, um novo cenário surgiu. A integração e o compartilhamento de informações em uma rede *peer-to-peer* é uma tarefa ainda mais complexa do que os cenários anteriores. O ambiente *peer-to-peer* traz as dificuldades não só pelo aumento de pontos, como também os problemas inerentes a esse tipo de rede como o fato de nem sempre todos os pontos estarem disponíveis e a alta mobilidade de pontos entrando e saindo da rede.

Ao mesmo tempo, sistemas PDMS (*Peer Data Management System*) foram desenvolvidos com um conceito parecido, fazendo uso de esquemas locais e mediadores responsáveis pela reformulação de consultas feitas sobre o esquema mediador. Esses sistemas possuem características em comum com os sistemas baseados em ontologias citados anteriormente.

Unindo esses dois ramos de pesquisa, o SPEED [3,4] é um OPDMS (Ontology-based Peer-to-peer Management System), fazendo uso de ontologias para representar seus esquemas e fazendo o uso de *matchings* para diversas tarefas realizadas dentro do sistema.

2.6 Raciocinadores sobre ontologias

Outro tipo de ferramenta que fazem uso de ontologias são os raciocinadores. Raciocinadores são *softwares* capazes de inferir consequências lógicas a partir de um conjunto de asserções, no caso deste projeto, precisamos deles para inferir os relacionamentos semânticos usando as regras semânticas descritas na especificação do projeto.

Os raciocinadores encontrados em geral também trazem com eles uma API para manipulação de ontologias, que também foi de bastante utilidade para o desenvolvimento das regras. Três raciocinadores foram estudados e analisados sobre alguns requisitos necessários. Foram analisados o Jena [14], o Sesame [15] e o Kaon2 [16] com relação aos seguintes critérios:

- *Open source*: Um dos requisitos era a ferramenta ser open source, para permitir o seu uso dentro do projeto.
- *Inferência*: A capacidade de inferir asserções sobre OWL e RDF.
- *Persistência*: Um requisito desejável era a possibilidade de armazenar os dados inferidos usando a própria ferramenta.
- *Documentação*: A maioria dos raciocinadores analisados eram APIs, onde para se fazer bom uso delas é necessário uma boa documentação.
- *Linguagem*: Por serem APIs a serem usadas, aqui é necessário que a linguagem utilizada seja Java.

As ferramentas foram testadas, seguindo a documentação existente. Foram feitos pequenos projetos em Java para testar cada funcionalidade. Uma análise sobre os critérios avaliados é mostrada no Quadro 2:

Raciocinador/Critério	Open source	Inferência	Persistência	Documentaçã	Linguagem

				o	
Jena	Sim	RDF/OWL	Sim	Muito boa	Java
Sesame	Sim	RDF	Sim*	Boa	Java
Kaon2	Sim	RDF/OWL	Sim*	Fraca	Java

Quadro 2 – Quadro comparativo entre raciocinadores.

Dentre os raciocinadores analisados, o Jena é a ferramenta mais usada e que possui um fórum com uma comunidade ativa, além de uma API bem feita. Ele oferece inferência e suporte à manipulação de ontologias em RDF e OWL, e também permite o armazenamento de ontologias inferidas em um banco de dados. O Jena possui um mecanismo próprio para a execução de regras, as chamadas *jena rules*, que permite a inferência de novo conhecimento através da execução das mesmas.

O Sesame foi outra ferramenta testada. Ele, como o Jena, também é open source e possui uma documentação razoável. Oferece suporte à inferência em RDF apenas. O uso de OWL para inferência é apenas possível com o uso de uma biblioteca externa chamada OWLIM. Essa biblioteca possui uma versão *free* e uma versão comercial. A versão *free*, chamada SwiftOWLIM, possui pouca documentação e não foi possível fazer testes com ela.

O Kaon2, último raciocinador analisado, promete muito nos artigos escritos e na descrição do site, porém, não conseguimos fazer muitos testes usando ele. Muitos erros apareciam durante a execução e nenhum resultado foi realmente obtido com o uso desta ferramenta.

Pelos seus pontos positivos, o Jena foi o raciocinador escolhido para ser usado nesta implementação.

2.7 Considerações finais

Neste capítulo tivemos uma visão geral dos conceitos e estudos feitos que são a base para o entendimento deste trabalho. No próximo capítulo veremos como funciona o sistema SPEED para entender melhor a necessidade de um *ontology matching*.

3. SPEED e Ontologias

Como já foi citado, sistemas PDMS (Peer-to-peer Data Management Systems) têm sido estudados como uma forma de lidar com bases de dados distribuídas em uma rede *peer-to-peer*. Cada ponto de dados precisa compartilhar seu esquema, ou parte dele para os outros pontos, essa informação é importante para cada ponto ter conhecimento sobre os demais. Esse modelo de ter um esquema ou parte dele representados de uma forma, para serem compartilhados entre os *peers*, lembra, em proporção reduzida, a proposta do uso de ontologias na Web Semântica [17].

A idéia de juntar esses dois mundos surgiu no trabalho de Xiao [2] que criou a definição de OPDMS (Ontology-based Peer Data Management Systems). O SPEED vem sendo desenvolvido como um OPDMS e, durante a pesquisa, alguns requisitos adicionais [3,4] foram identificados em relação ao trabalho de Xiao:

- *Representação do esquema exportado*: Dados do *peer* devem ser mapeados em uma ontologia para visualização externa.
- *Conceitualização global*: Uma ontologia global pode ser usada para prover uma visão conceitual sobre os *peers* locais.
- *Identificação de mapeamentos*: Uma ontologia pode ser também usada para facilitar a identificação de mapeamentos semânticos entre os *peers* locais.
- *Suporte a processamento de consultas*: Usar uma ontologia global com dois propósitos: (i) como uma visão alto-nível dos *peers*. (ii) como um mediador, facilitando a reescrita de consultas.
- *Índice semântico*: Pode-se usar um índice semântico para localizar de forma eficiente fontes de dados que podem contribuir no resultado de uma consulta.
- *Capacidade de gerar matching semântico*: Os peers devem ter a capacidade de fazer *matching* entre ontologias.

No SPEED, os *peers* são agrupados de acordo com o conteúdo compartilhado [3,4]. Isso foi pensado para facilitar a identificação de pontos de dados que têm informações em comum, diminuindo a complexidade da busca, pois a busca seria feita apenas dentro de um *cluster* (detalhes serão explicados na próxima seção).

Segundo Xiao [2], duas características são apontadas como críticas para o funcionamento de um OPDMS: a geração de alinhamentos entre ontologias e a reformulação de consultas. Neste trabalho, o *ontology matching* desenvolvido colabora para a solução desses problemas dentro do Sistema SPEED. Gerando alinhamentos entre as ontologias considerando tanto a parte linguística e estrutural, como a semântica, com o auxílio de uma ontologia de domínio. Esses alinhamentos são úteis para a identificação do *cluster* onde um *peer* entrante deve ficar, e também para a identificação de relacionamentos semânticos entre ontologias, uma tarefa que serve de base para a reformulação de consultas.

3.1 Arquitetura

A arquitetura do Sistema SPEED (Semantic Peer Data Management System) foi formulada por [3] com base nos requisitos necessários para um OPDMS. Duas topologias de rede são usadas no SPEED. A topologia DHT, no anel superior, é formada pelos *peers* semânticos, que são os pontos mais representativos de uma comunidade semântica. A topologia *Super-peer*, é formada pelos *peers* de integração e de dados, e liga os *clusters* semânticos. Existem diferentes tipos de *peers* de acordo com o nível em que eles se encontram na arquitetura.

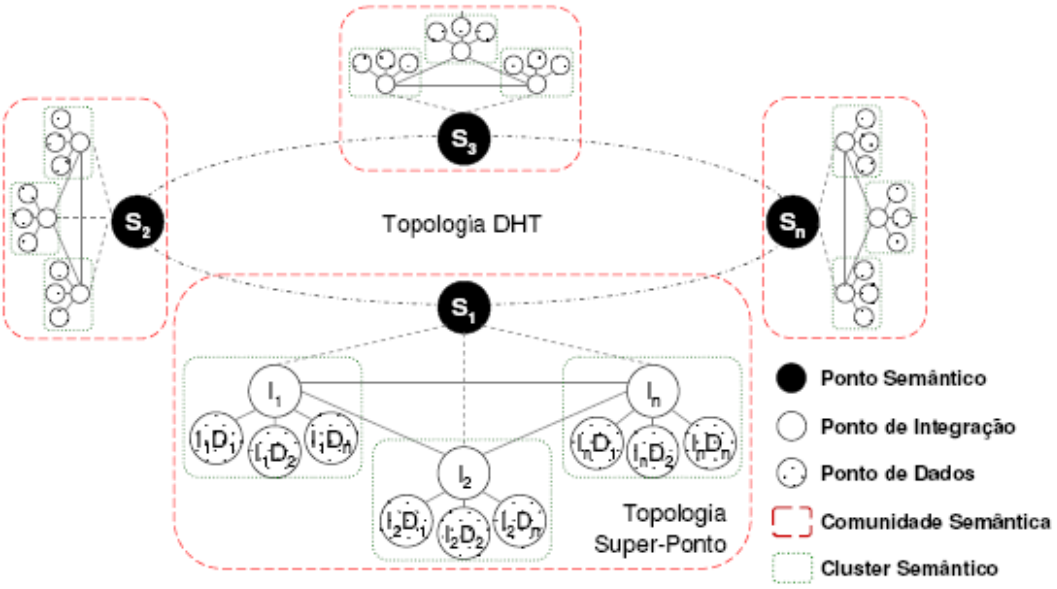


Figura 1 - Arquitetura do SPEED.

Os pontos semânticos aparecem na Figura 1, representados por S_1, S_2, S_3 e S_n. Eles funcionam como pontos de entrada para uma comunidade semântica, cada um deles possui uma ontologia de domínio que representa a comunidade semântica, e que é formada a partir dos esquemas dos pontos de dados e integração. Essa ontologia de domínio (que chamamos de *Community Ontology*, ou CMO) é usada no momento de entrada de um novo ponto, para decidir em que comunidade o ponto ficará, e no momento em que é executada uma consulta em algum dos pontos, ajudando na reformulação de consultas e fazendo o enriquecimento semântico.

Os pontos de integração aparecem na Figura 1, representados por I_1, I_2 e I_n. Dentro de um *cluster* semântico, eles são o ponto de dados com maior representatividade e possuem conhecimento de todos os pontos de dados dentro do cluster. Estes *peers* se comunicam com outros pontos de integração e com os pontos de dados. São eles os responsáveis pelo controle e processamento de consultas. Um *cluster* semântico também possui uma ontologia (chamada de *Cluster* Ontology, ou CLO), que o representa e consiste de uma espécie de junção dos esquemas dos pontos de dados.

Os pontos de dados, são representados na figura por I_1D_1, I_1D_2, I_1D_n, I_1D_1, I_1D_2, I_1D_n, I_nD_1, I_nD_2 e I_nD_n. Eles são basicamente qualquer outro ponto

participante da rede, e fazem parte de um cluster semântico, compartilhando seu esquema através de ontologias (*Local ontology*, ou LO). São nesses pontos onde são executadas as consultas.

Um *cluster* semântico é um conceito lógico e associa *peers* de dados a um *peer* de integração que possuem interesses semânticos semelhantes. Cada *cluster* semântico está relacionado a um interesse comum entre *peers* de dados e possui um *peer* de integração que é responsável por tarefas como indexação de metadados, processamento de consultas e integração dos dados. Os clusters servem para prover um ambiente estável onde serão aplicadas as técnicas de associação dos esquemas exportados.

Uma comunidade semântica também é um conceito lógico e agrupa *clusters* semânticos que possuem interesses semânticos em comum, ela é o mais alto nível de um conjunto semântico.

3.2 Ontologia como representação de esquemas

Como citamos anteriormente, o uso de ontologias em sistemas PDMS foi pensado por Xiao em [2]. Porém, segundo [18], a idéia não é nova, e existem outros tipos de sistemas, anteriores ao PDMS que fazem uso de ontologias. Um dos primeiros tipos de aplicação a usar ontologias para representar esquemas foi o campo de integração de informações. Nesse tipo de aplicação, ontologias eram usadas para representar os esquemas locais, e uma ontologia global que continha uma visão global dos esquemas. *Matchings* eram feitos entre a ontologia global e as locais para se reformular as consultas.

Outro tipo de aplicação a usar ontologias eram as que faziam integração de esquemas. Na verdade esse tipo de aplicação gerava alinhamentos entre 2 esquemas representados por ontologias para depois de forma manual, a integração ser feita por algum especialista.

Em [19] são citadas as duas formas que as aplicações correntes usam para criar uma ontologia para representar o esquema. O primeiro tipo são as aplicações que realmente criam uma ontologia a partir de um esquema de banco de dados. Nessa categoria, algumas transformações são diretas como criar uma classe OWL [3,4] para cada entidade do esquema, e também

representar os atributos através de propriedades. Além desses passos mais básicos, outros relacionamentos semânticos são inferidos a partir das ligações entre entidades do banco de dados como chaves estrangeiras e chaves primárias.

A outra forma de se obter uma ontologia para representar o esquema da base de dados, é usar uma ontologia já existente e achar correspondências entre esses esquemas. O mapeamento feito entre os esquemas é complexo, por alguns motivos:

- A heterogêneidade (como as discutidas no Capítulo 2) entre termos da base de dados e a ontologia.
- A natureza das duas representações é diferente, sendo mais complicada a identificação de relacionamentos semânticos semelhantes no esquema e na ontologia.
- A ontologia pode modelar um universo maior ou menor que o representado pelo esquema e não necessariamente conter os mesmos conceitos e propriedades.

O SPEED ainda não possui uma forma escolhida para a geração de ontologias a partir de esquemas. As ontologias locais no SPEED serão normalizadas, ou seja, conterão apenas termos que já existem na ontologia de domínio. Para obter esse resultado, alguma forma híbrida entre os dois tipos citados anteriormente deve ser usada.

3.3 Considerações finais

Neste capítulo foram discutidos o funcionamento do SPEED e sua arquitetura. Também foi discutido o uso de ontologias na representação de esquemas e de que maneira pode ser construída uma ontologia a partir de um esquema.

4. Mapeamento Semântico de Ontologias

Neste capítulo veremos os requisitos necessários para a implementação do *Ontology matching* semântico de forma a atender as necessidades do SPEED. Também será apresentado todo o processo de *matching* e da geração de medida de similaridade. Para finalizar será mostrada a execução de um exemplo na aplicação implementada.

4.1 Especificação

O SPEED faz uso da operação de *matching* de ontologias em dois momentos. Primeiro, no momento da entrada de um *peer* em uma comunidade semântica, para a escolha do *cluster* onde ele ficará localizado. Essa escolha é feita dependendo do grau de similaridade identificado entre as duas ontologias.

Em um segundo momento, quando uma consulta é submetida em um ponto de dados, é necessária a identificação de correspondências entre a base local e as outras bases onde a consulta será reescrita de acordo com essas correspondências.

O objetivo do uso do *Ontology Matching* semântico no SPEED é atender às necessidades citadas anteriormente. Para isso, foi estudada a melhor forma para se obter melhores resultados. Identificados os problemas, foi desenvolvido um documento de casos de uso com as necessidades do sistema SPEED. Uma versão simplificada do diagrama é mostrada na Figura 2.

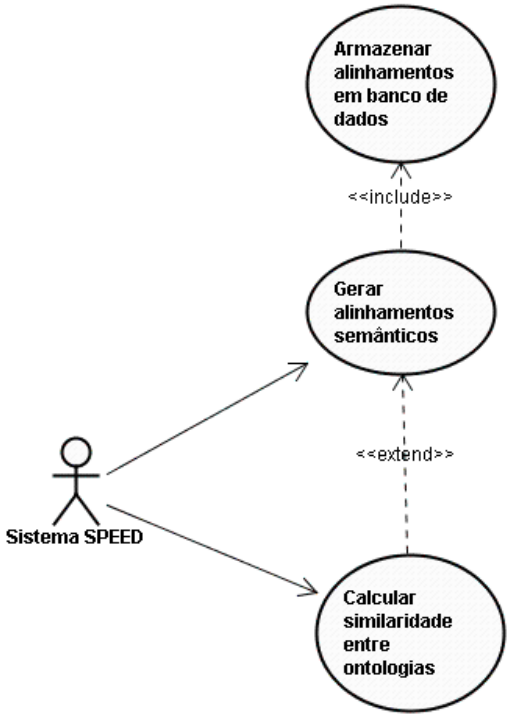


Figura 2 - Casos de uso do *Ontology matching.*

O diagrama apresentado na Figura 2 mostra os dois casos de uso básicos necessários para o funcionamento do sistema SPEED. O caso de uso "gerar alinhamentos semânticos" foi idealizado no trabalho de [4], e trata de gerar as correspondências semânticas para a reformulação de consultas. O caso de uso "calcular similaridade entre ontologias" foi desenvolvido no trabalho de [3], e usa os alinhamentos semânticos, gerados no caso de uso anterior, para melhorar a qualidade dos alinhamentos gerados por um *matcher* linguístico estrutural. Mais detalhes sobre os casos de uso podem ser vistos no Anexo A.

4.2 Correspondências semânticas

Para a geração das correspondências semânticas, equivalências são identificadas entre CLO e CMO, e entre a LO e CMO. A identificação dessas equivalências é feita levando simplesmente o nome do conceito em questão, pois as ontologias estão normalizadas. Identificadas as equivalências, são verificadas as regras semânticas para se encontrar relacionamentos

semânticos. Sete regras são usadas, baseadas no trabalho de [3]. Para especificá-las usaremos a seguinte notação:

1. i:x  j:y, uma correspondência *isEquivalentTo*. (Equivalência)

2. i:x  j:y, uma correspondência *isSubConceptOf*. (Especialização)

3. i:x  j:y, uma correspondência *isSuperConceptOf*. (Generalização)

4. i:x  j:y, uma correspondência *isPartOf*. (Agregação – parte)

5. i:x  j:y, uma correspondência *isWholeOf*. (Agregação – todo)

6. i:x  j:y, uma correspondência *isCloseTo*. (Proximidade)

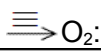
7. i:x  j:y, uma correspondência *isDisjointWith*. (Disjunto de)

Onde x e y são elementos (conceitos ou propriedades) pertencentes a uma ontologia Oi e outra ontologia Oj respectivamente representando duas ontologias do sistema.

- **IsEquivalentTo:**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico ao mesmo conceito *k* em O. Os conceitos *x* e *y* são equivalentes.

if (O_1:x ≡ O:k) and (O_2:y ≡ O:k) then

O_1:x  O_2:y

- **IsSubConceptOf**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O, e em O, *k* é subclasse de *z*. O conceito *x* é SubConceptOf de *y*.

if (O_1:x ≡ O:k) and (O_2:y ≡ O:z) and (O:k μ O:z) then

O_1:x  O_2:y

- **IsSuperConceptOf**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O, e em O, *k* é super classe de *z*. O conceito *x* é SuperConceptOf de *y*.

if (O_1:x ≡ O:k) and (O_2:y ≡ O:z) and (O:k } O:z) then

O_1:x  O_2:y.

- **IsPartOf**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O, e em O, *k* é parte de *z*. O conceito *x* é PartOf de *y*.

if (O_1:x ≡ O:k) and (O_2:y ≡ O:z) and (O:k ≅ O:z) then

O_1:x  O_2:y.

- **IsWholeOf**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O, e em O, *k* é todo de *z*. O conceito *x* é WholeOf de *y*.

if (O_1:x ≡ O:k) and (O_1:y ≡ O:z) and (O:k 0 O:z) then

O_1:x  O_2:y.

- **IsCloseOf**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O. E em O, *k* e *z* são subclasses de um conceito *a* em comum, esse conceito *a* não é o root e *k* e *z* estão até no máximo *n* níveis de profundidade abaixo de *a* (onde *n* é o valor do *threshold*). O conceito *x* é CloseOf de *y*.

If (O_1:x ≡ O:k and O_2:y ≡ O:z) and (O:k μ O:a and O:z μ O:a)

and O:a ≠ ♦

and (depth(O:k,O:a) < *threshold* and depth(O:z,O:a) < *threshold*) then

O_1:x $\overset{\approx}{\rightarrow}$ O_2:y.

- **IsDisjointWith**

Se há um conceito *x* em O_1 que é idêntico a um conceito *k* em O, e outro conceito *y* em O_2 é idêntico a um conceito *z* em O. E em O, *k* e *z* são subclasses do conceito *a* que é o root, e *k* e *z* estão até no máximo *n* níveis de profundidade abaixo de *a* (onde *n* é o valor do *threshold*). O conceito *x* é CloseOf de *y*.

If (O_1:x ≡ O:k and O_2:y ≡ O:z) and (O:k μ O:a and O:z μ O:a)

and O:a ≡ ♦ and (depth(O:k, O:a) < threshold

and (depth(O:z, O:a) < threshold))

and there is no other common super concept between O:k and O:z then

O_1:x $\overset{\not\equiv}{\rightarrow}$ O_2:y.

O processo de identificação de correspondências semânticas é visto na Figura 3 e é usado para a reescrita de consultas. A entrada desse processo são duas ontologias, e faz-se uso das regras mostradas para encontrar relacionamentos semânticos, sendo essas correspondências encontradas à saída do processo. Dentro deste processo, nas regras acima descritas, O_1 e O_2 podem ser uma CLO e LO, ou 2 CLOs. Os alinhamentos finais podem ser armazenados em um banco de dados ou salvos em arquivos, o que também era um requisito para o projeto.

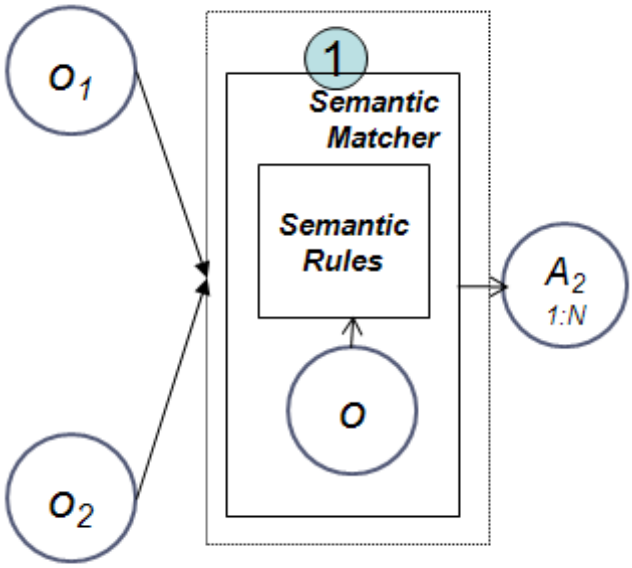


Figura 3 - Parte do *Ontology matching* responsável pela geração de correspondências.

4.3 Processo geral do *Ontology Matching* semântico

Para o processo de geração da medida de similaridade entre as duas ontologias, o primeiro passo é feito em (1), onde é executado um *matching* lingüístico-estrutural, fazendo uso de uma ferramenta externa, gerando correspondências *1-N*. Em paralelo, no módulo (2), é feito o *matcher* semântico. O *macher* semântico foi descrito na seção anterior e sua saída são as correspondências semânticas, que são usadas para adicionar semântica ao resultado do *matcher* lingüístico-estrutural. Uma visão do processo geral é visto na Figura 4.

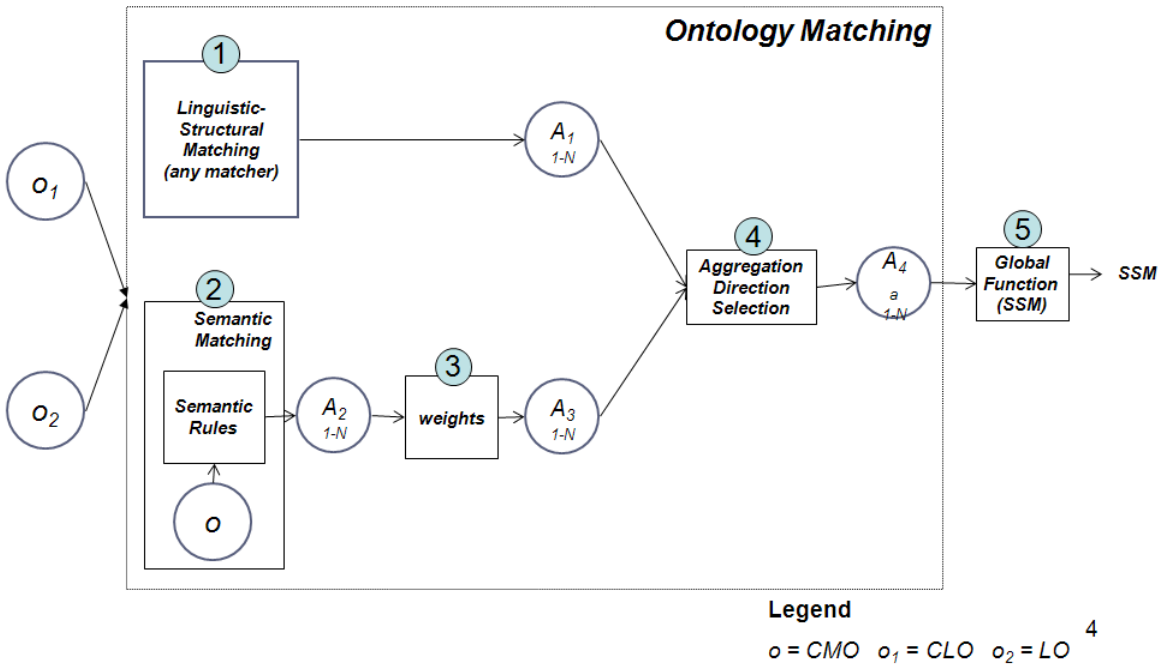


Figura 4 - Processo do *Ontology Matching* semântico.

Obtidos os alinhamentos semânticos, na fase (3), a cada um desses relacionamentos encontrados é atribuído um peso que corresponde a um dos seguintes:

- isEquivalentTo => 1.0
- isSubConceptOf => 0.9
- isSuperConceptOf => 0.8
- isCloseTo => 0.7
- isPartOf/isWholeOf => 0.3
- isDisjointWith => 0.0

4.4 Cálculo da medida de similaridade

Como entrada para o passo (4), temos dois pares de alinhamentos 1-N, um vindo do *matching* lingüístico-estrutural, outro vindo do *matching* semântico. Os alinhamentos semânticos são usados para aumentar a qualidade do *matching* linguístico e estrutural acrescentando conhecimento semântico. A

junção dos alinhamentos é feita calculando uma média ponderada dos alinhamentos linguístico estrutural e semântico, onde o primeiro tem peso 0.4, e o segundo tem peso 0.6. Formamos, então, uma tabela de valores de alinhamentos como no Quadro 1 (valores exemplo):

O1\O2	O2Conceito1	O2Conceito2	O2Conceito3
O1Conceito1	1.00	0.56	0.56
O1Conceito2	0.72	0.68	0.54
O1Conceito3	0.68	0.54	0.76

Quadro 3 – Exemplo de tabela formada por *matchings* (Valores exemplo).

Podemos escolher o melhor alinhamento entre cada classe considerando as linhas e as colunas. Por exemplo, de O_1 para O_2, O conceito O1Conceito2 tem maior alinhamento com O2Conceito1, mas de O_2 para O_1, o conceito O2Conceito2 tem melhor alinhamento com O1Conceito2. Assim, são selecionados os melhores alinhamentos para cada conceito, escolhendo os com maior valor encontrado para a média entre o alinhamento lingüístico-estrutural e o alinhamento semântico, sobrando um conjunto de alinhamentos 1-1 em cada direção (de O_1 para O_2 e de O_2 para O_1). Usamos esse conjunto de alinhamentos, para calcular a medida de similaridade global, temos duas opções de fórmulas:

DICE:

$$\frac{(\text{Número de alinhamentos de } O_1 \text{ para } O_2) + (\text{Número de alinhamentos de } O_2 \text{ para } O_1)}{(\text{Número de conceitos em } O_1) + (\text{Número de conceitos em } O_2)}$$

Average:

$$\frac{(\text{Soma dos valores de alinhamentos de } O_1 \text{ para } O_2) + (\text{Soma dos valores de alinhamentos de } O_2 \text{ para } O_1)}{(\text{Número de conceitos em } O_1) + (\text{Número de conceitos em } O_2)}$$

A função *DICE* tem uma abordagem mais otimista, considerando o número de alinhamentos no numerador, como se cada alinhamento tivesse valor 1. A *Average*, por sua vez, usa a soma dos valores de cada alinhamento.

4.5 Implementação

Para a implementação desse projeto, foi usada a linguagem Java, que já vinha sendo usada em outros módulos do projeto SPEED, e foram necessários alguns estudos de ferramentas e APIs externas (mostrados no Capítulo 2).

No passo (1) da Figura 3, por exemplo, apresentada na seção anterior, onde é feito o uso de um *matcher* lingüístico-estrutural, foi preciso um estudo de alguns *matchers* linguístico-estruturais, por não ser o objetivo deste trabalho a implementação de um novo *matcher*. O *matcher* escolhido para esta implementação foi o HMatch (o estudo e comparação sobre outras ferramentas é feito no Capítulo 2).

Na etapa (2) do processo do *Ontology matching*, é necessária a execução de regras semânticas. Para implementar essas regras, foi feito o estudo sobre raciocinadores. Pelos seus pontos positivos, o Jena foi o raciocinador escolhido para ser usado nesta implementação. Para a identificação de relacionamentos semânticos, algumas regras foram implementadas com o uso das *Jena Rules* e outras foram possíveis apenas varrendo a ontologia e verificando a condição. Usando o *Jena Rules*, foram implementadas as regras *IsEquivalentTo*, *IsSubConceptOf*, *IsSuperConceptOf*, *IsPartOf* e *IsWholeOf*.

No caso da regra *IsSuperConceptOf*, houve uma pequena mudança na implementação. Não existe o predicado *superclass* em RDFS, esse predicado é inferido a partir do inverso do predicado. Assim, a regra foi implementada como se segue:

If (O_1:x ≡ O:k) and (O_2:y ≡ O:z) and (O:z rdfs:subclassof O:k)* then

O_1:x IsSubConceptOf O_2:y.

*Como foi dito, e*m RDFS não existe o predicado rdfs:superclassOf, para identificar esse relacionamento se usa o inverso de rdfs:subclassOf*.

As regras *IsCloseTo* e *IsDisjointWith*, por serem mais complexas, só puderam ser implementados usando a API do Jena para varrer a ontologia e verificar se a condição realmente era verdadeira. Apesar de não ser um requisito, foi implementada uma interface gráfica para a melhor visualização dos resultados. Segue uma imagem da tela inicial do sistema.

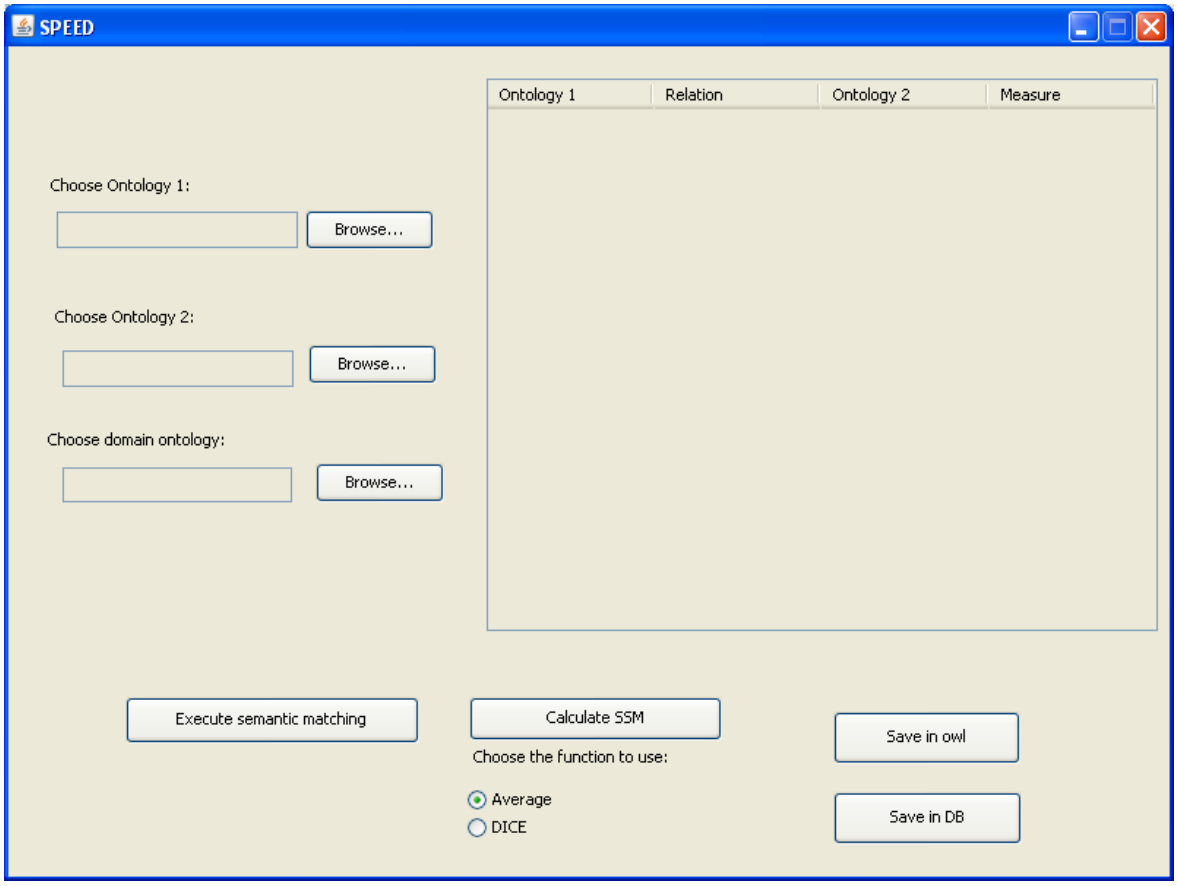


Figura 5 – Tela inicial do *Ontology matching.*

Nesta tela vemos todas as opções de operações possíveis de serem realizadas com o *ontology matching*. Depois de escolhidas as ontologias, é possível:

- *Executar matching semântico*: Executar o matching semântico entre as ontologias de entrada.
- *Calcular a medida de similaridade*: Calcular a medida de similaridade entre as ontologias com a opção de usar a medida *Average* ou *DICE*.
- *Salvar alinhamentos em arquivo*: Salvar alinhamentos semânticos gerados em OWL.

- *Salvar alinhamentos em BD*: Salvar alinhamentos semânticos em um banco de dados MySQL.

Para mostrar as interfaces e resultados, usaremos uma ontologia que representa o domínio de uma conferência. Seguem as imagens com a hierarquia de cada ontologia que foi usada como entrada do *Ontology matching*.

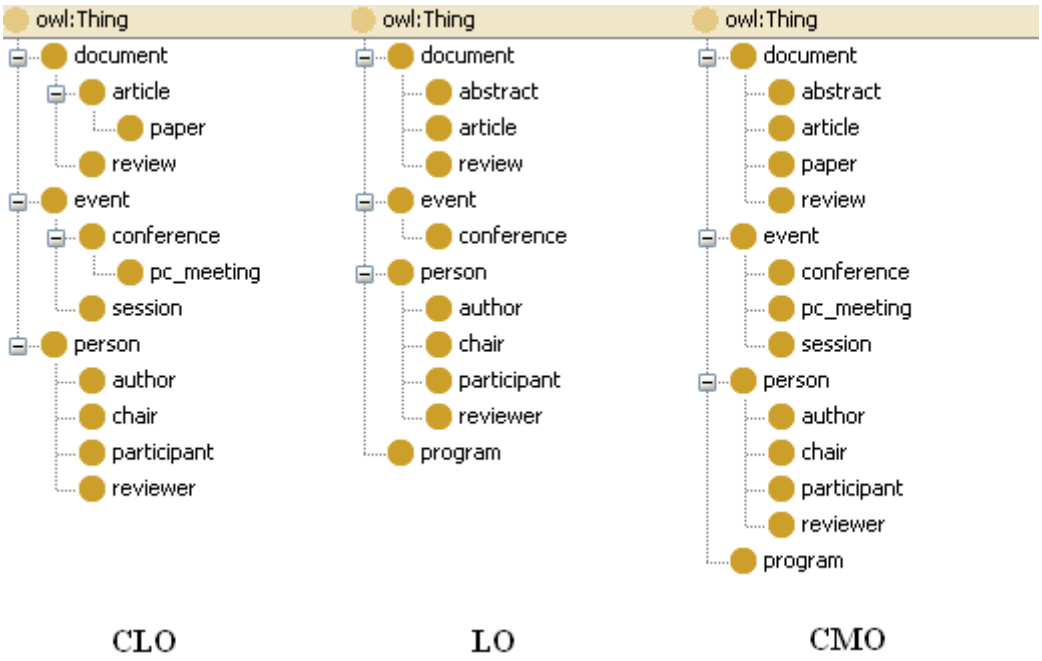


Figura 6 – Ontologias usadas como entrada no *Ontology matching.*

As ontologias são parecidas para mostrar os *matchings* gerados. Podemos notar que elas estão normalizadas como falamos anteriormente, ou seja, todos os conceitos das CLOs e LOs estão contidos na CMO.

Primeiro mostraremos a execução do *macthing* semântico. Depois de executado o processo, os alinhamentos são mostrados na própria aplicação como na Figura 7:

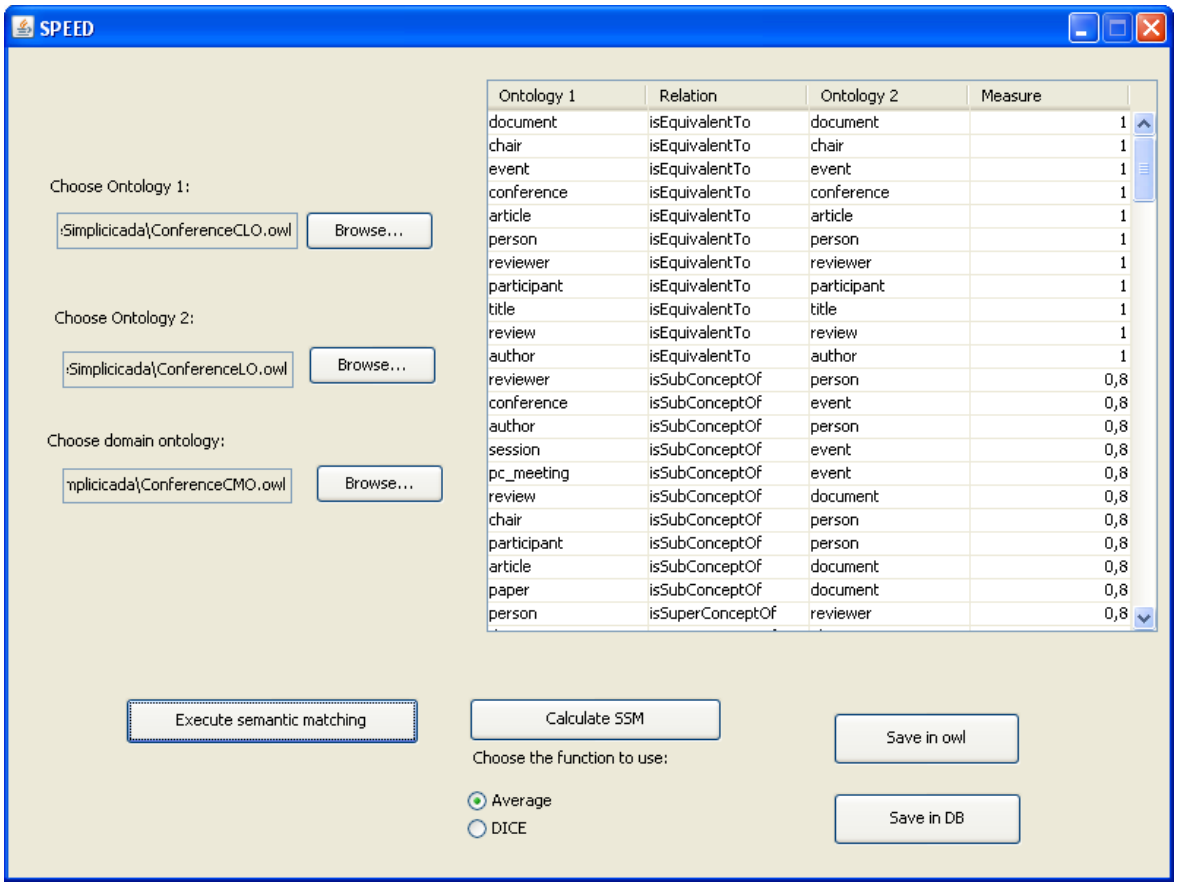
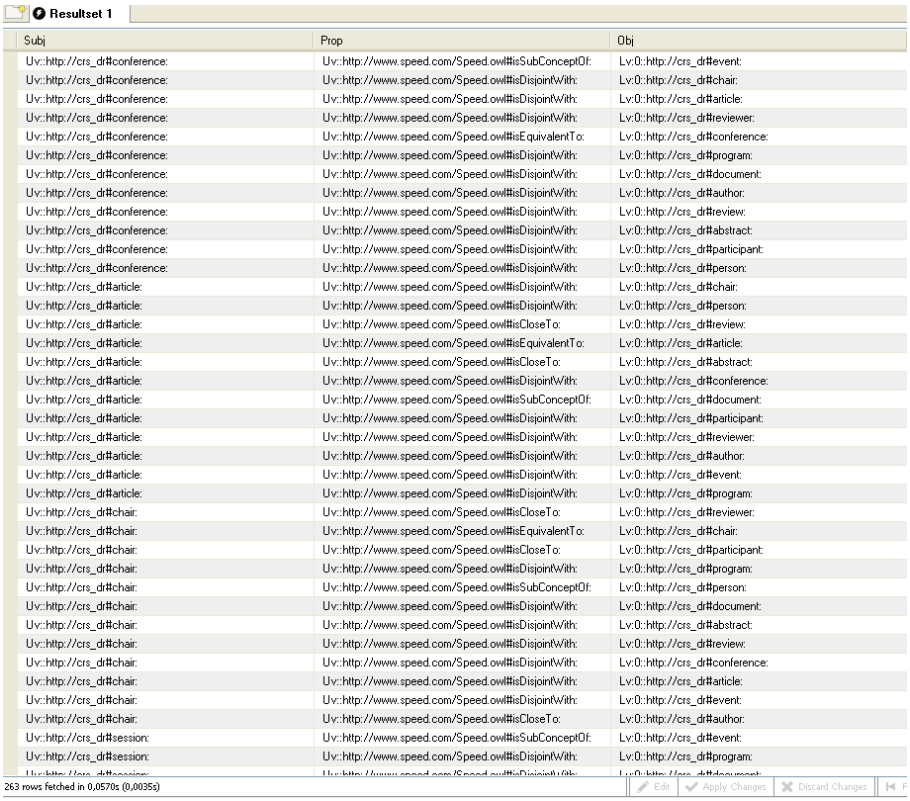


Figura 7 – Resultado do *matching* semântico.

Para cada alinhamento identificado, são mostradas as classes de cada ontologia que fazem parte do alinhamento, o relacionamento encontrado entre elas, e o valor de proximidade entre esses conceitos. No caso do *matching* semântico, esse valor corresponde ao valor que foi descrito na Seção 4.1, na parte do processo onde são atribuídos os pesos ao conjunto de alinhamentos do *matching* semântico. Depois de executado o *matching* semântico, podemos salvar os alinhamentos encontrados de duas formas, em arquivo, ou em banco de dados.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:j.0="http://www.speed.com/Speed.owl#" >
  <rdf:Description rdf:about="http://crs_dr#session">
    <j.0:isSubConceptOf>http://crs_dr#event</j.0:isSubConceptOf>
    <j.0:isDisjointWith>http://crs_dr#program</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#document</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#author</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#person</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#article</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#abstract</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#chair</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#review</j.0:isDisjointWith>
    <j.0:isCloseTo>http://crs_dr#conference</j.0:isCloseTo>
    <j.0:isDisjointWith>http://crs_dr#reviewer</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#participant</j.0:isDisjointWith>
  </rdf:Description>
  <rdf:Description rdf:about="http://crs_dr#reviewer">
    <j.0:isSubConceptOf>http://crs_dr#person</j.0:isSubConceptOf>
    <j.0:isDisjointWith>http://crs_dr#document</j.0:isDisjointWith>
    <j.0:isCloseTo>http://crs_dr#participant</j.0:isCloseTo>
    <j.0:isDisjointWith>http://crs_dr#review</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#event</j.0:isDisjointWith>
    <j.0:isEquivalentTo>http://crs_dr#reviewer</j.0:isEquivalentTo>
    <j.0:isDisjointWith>http://crs_dr#article</j.0:isDisjointWith>
    <j.0:isCloseTo>http://crs_dr#author</j.0:isCloseTo>
    <j.0:isDisjointWith>http://crs_dr#program</j.0:isDisjointWith>
    <j.0:isCloseTo>http://crs_dr#chair</j.0:isCloseTo>
    <j.0:isDisjointWith>http://crs_dr#abstract</j.0:isDisjointWith>
    <j.0:isDisjointWith>http://crs_dr#conference</j.0:isDisjointWith>
  </rdf:Description>
```

Figura 8 – Arquivo OWL com relacionamentos identificados.



Resultset 1

Subj	Prop	Obj
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isSubConceptOf:	Lv:0::http://crs_dr#event:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#chair:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#article:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#reviewer:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isEquivalentTo:	Lv:0::http://crs_dr#conference:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#program:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#document:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#author:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#review:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#abstract:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#participant:
Uv::http://crs_dr#conference:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#person:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#chair:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#person:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isCloseTo:	Lv:0::http://crs_dr#review:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isEquivalentTo:	Lv:0::http://crs_dr#article:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isCloseTo:	Lv:0::http://crs_dr#abstract:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#conference:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isSubConceptOf:	Lv:0::http://crs_dr#document:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#participant:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#reviewer:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#author:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#event:
Uv::http://crs_dr#article:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#program:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isCloseTo:	Lv:0::http://crs_dr#reviewer:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isEquivalentTo:	Lv:0::http://crs_dr#chair:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isCloseTo:	Lv:0::http://crs_dr#participant:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#program:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isSubConceptOf:	Lv:0::http://crs_dr#person:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#document:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#abstract:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#review:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#conference:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#article:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#event:
Uv::http://crs_dr#chair:	Uv::http://www.speed.com/Speed.owl#isCloseTo:	Lv:0::http://crs_dr#author:
Uv::http://crs_dr#session:	Uv::http://www.speed.com/Speed.owl#isSubConceptOf:	Lv:0::http://crs_dr#event:
Uv::http://crs_dr#session:	Uv::http://www.speed.com/Speed.owl#isDisjointWith:	Lv:0::http://crs_dr#program:

263 rows fetched in 0,0570s (0,0035s)

Edit | Apply Changes | Discard Changes

Figura 9 – Alinhamentos salvos no banco de dados.

Outra operação possível de se realizar no *Ontology matching* é o calculo de similaridade entre ontologias. O cálculo pode ser feito considerando umas das duas funções: *Average* ou *DICE*.

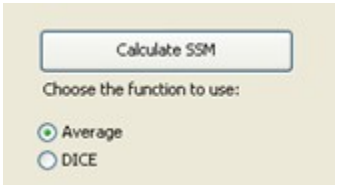


Figura 10 – Local da tela onde é possível escolher entre as duas funções.

Executamos o cálculo de similaridade com cada uma das fórmulas e o resultado é mostrado da seguinte forma:

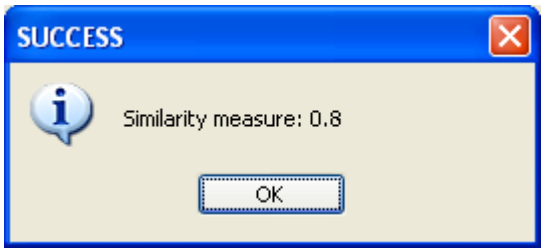


Figura 11 – Resultado do cálculo de similaridade.

Nesse caso, por coincidência, o valor para as duas funções foi 0.8. O valor gerado pela função *DICE* é sempre igual ou maior que o da *Average*, pelo fato de *DICE* considerar o valor de cada alinhamento encontrado como 1.0, podemos notar isso analisando a fórmula na Seção 4.1.

5. Conclusões e trabalhos futuros

Neste trabalho, foi feito um estudo sobre o funcionamento e as características de um o*ntology matching*, com o objetivo de fazer uso de uma ferramenta deste tipo dentro do sistema SPEED. Além de ter sido gerada uma versão de um *Ontology matching* atendendo às necessidades do projeto.

5.1 Contribuições

A ferramenta desenvolvida neste trabalho é importante para tarefas importantes do SPEED, como já citado em seções anteriores. Inicialmente, o foco da implementação foi o *matcher* semântico. Toda a parte de geração de alinhamentos semânticos através da execução das regras. Esta parte do trabalho será fundamental para a reescrita de consultas quando ela é repassada para outros *peers*.

Então, a partir do *matcher* semântico, o *Ontology matching* em si foi desenvolvido, de modo a adicionar a questão semântica aos alinhamentos linguístico-estruturais gerados por uma ferramenta de *matching* externa. Isto contribui na hora de identificar o *cluster* semântico onde um *peer* entrante deverá ser alocado.

Depois da implementação, uma documentação em forma de *javadoc* foi gerada visando facilitar o desenvolvimento de mais funcionalidades sobre a ferramenta. Também com esse propósito, o código foi escrito de forma bem modularizada.

5.2 Dificuldades encontradas

Como foi mostrado em outras seções deste documento, a implementação deste projeto fez uso de algumas ferramentas externas. A análise dessas ferramentas se tornou complicada devido ao fato de nem todas possuírem documentação adequada, ou por não apresentarem todos os

requisitos necessários. Em alguns casos, como o raciocinador Kaon2, mesmo seguindo a documentação não se conseguiu obter os resultados mostrados nos sites e artigos.

Outra dificuldade encontrada durante o desenvolvimento foi a mudança que houve na arquitetura do *Ontology matching*, devido ao tempo de execução longo. Isto era inaceitável devido à quantidade de vezes que este processo precisava ser executado pelo sistema SPEED.

5.3 Trabalhos futuros

Esta versão implementada do *Ontology matching* satisfaz necessidades básicas do SPEED. Porém outras funcionalidades podem ser adicionadas no futuro:

- Gerar correspondências entre propriedades.
- Possibilitar o uso automático de outros *matchers* lingüístico-estruturais além do Hmatch.

Referências

[1] Sung, L. G. A., Ahmed, N., Blanco, R., Li, H., Soliman, M. A., Hadaller,D. 2005. “A Survey of Data Management in Peer-to-peer Systems”. School of Computer Science, University of Waterloo. Tech. Report CS-2006-18, 2006.

[2] Xiao, H. 2006. “Query Processing for Heterogeneous Data Integration using Ontologies”. PhD Thesis. University of Illinois at Chicago, Chicago, USA.

[3] Pires, C. E. 2007. “Um sistema de Gerenciamento Dados com Conectividade Baseada em Semântica”. Exame de qualificação e proposta de tese, FPE, Brasil, abril 2007

[4] Souza, D. Y. 2007. “Reformulação de Consultas Baseada em Semântica para PDMS”. Exame de qualificacao e proposta de tese, UFPE, Brasil, abril de 2007.

[5] Neves, T. A., 2008 “Desenvolvimento do Módulo de Reformulação de Consultas no Sistema SPEED”. Monografia e Proposta de Trabalho de Graduação. CIn, UFPE.

[6] Guarino, N., 1998. “Formal Ontology and Information Systems”. In Proc. the First International Conference on Formal Ontologies in Information Systems, Trento, Italy, 1998, pp.3-15.

[7] “OWL Web Ontology Language Reference”. Acessado em 17/11/2008 <http://www.w3.org/TR/owl-ref/>

[8] “Resource Description Framework (RDF)”. Acessado em 24/11/2008. <http://www.w3.org/RDF/>

[9] H. Sofia Pinto, A. Gomez-Perez, J. P. Martins, “Some Issues on Ontology Integration”, In Proc. of IJCAI99’s Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends, 1999.

[10] “F-Logic”. Acessado em 24/11/2008.

<http://www.w3.org/2005/rules/wg/wiki/F-logic>

[11] Aumueller, D., Do, H., Massmann, S., Rahm, E., 2005. "Schema and Ontology Matching with COMA++". 2005. Schema and ontology matching with COMA++, Proceedings of the 2005 ACM SIGMOD international conference on Management of data, June 14-16, 2005, Baltimore, Maryland.

[12] Castano, S., Ferrara, A., and Montanelli, S., 2003. "H-match: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems Peer-based Systems". In Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003), Berlin, Germany, September 2003.

[13] Euzenat J., 2004. "An API for ontology alignment". In Proceedings of the International Semantic Web Conference (ISWC), pages 698–712, 2004.

[14] "Jena – A Semantic Web Framework for Java". Último acesso em 07/11/2008. <http://jena.sourceforge.net/>

[15] "OpenRDF.org... home of Sesame". Último acesso em 07/11/2008. <http://www.openrdf.org/>

[16] "Kaon2". Acessado em 07/11/2008. <http://kaon2.semanticweb.org/>

[17] Berners-Lee T, Hendler J, Lassila O. 2001 “The Semantic Web”. Scientific American, 284(5):34-43.

[18] Euzenat J., Shvaiko P. “Ontology Matching”. Springer, 2007.

[19] Ghawi R., Cullot N. 2007. "Database-to-Ontology Mapping Generation for Semantic Interoperability". Third International Workshop on Database Interoperability (InterDB 2007), held in conjunction with VLDB 2007.

Assinaturas

_____

Thiago Pachêco Andrade Pereira

Orientando

_____

Ana Carolina Salgado

Orientadora