Compatibilização de ontologias: um estudo comparativo de ferramentas.

Isabella Pires Capobiango

JUIZ DE FORA
JULHO, 2011

Compatibilização de ontologias: um estudo comparativo de ferramentas.

ISABELLA PIRES CAPOBIANGO

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Jairo Francisco de Souza

JUIZ DE FORA
JULHO, 2011

COMPATIBILIZAÇÃO DE ONTOLOGIAS: UM ESTUDO COMPARATIVO DE FERRAMENTAS.

Isabella Pires Capobiango

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

Aprovada por:

_____
Jairo Francisco de Souza, orientador.
MSc em Engenharia de Sistemas e Computação COPPE/UFRJ

_____
Regina Maria Maciel Braga Villela
DSc em Engenharia de Sistemas e Computação COPPE/UFRJ

_____
Alessandreia Marta de Oliveira
MSc em Engenharia de Sistemas e Computação COPPE/UFRJ

JUIZ DE FORA, MG - BRASIL
8 de JULHO, 2011

Resumo

Com a evolução da Web para a Web Semântica, o uso de agentes de software vem aumentando, pois surge a necessidade de mecanismos que garantam a interoperabilidade entre aplicações, devido ao fato que as máquinas passam a processar as informações disponibilizadas. Os agentes de *software* precisam de informações que os apoiam para então efetivamente colaborar, eles atuam em ambientes abertos e heterogêneos. A proposta mais comum para essas informações utilizadas pelos agentes de *software* é uso de ontologias. As ontologias objetivam representar conhecimento consensual entre pessoas e aplicações, permitindo o reuso e compartilhamento de informações. O problema passa a ser como compatibilizar diferentes ontologias de modo a garantir a comunicação entre agentes de *software*. Na literatura, esse problema é tratado como compatibilização de ontologias. Neste trabalho, apresentamos diferentes ferramentas para compatibilização de ontologias e realizamos testes em cada ferramenta para alguns pares definidos de ontologias com a finalidade de identificar diferenças e similaridades entre as abordagens de compatibilização adotadas por essas ferramentas.

Palavras-chave: Ontologias, Interoperabilidade, Teste de ferramentas.

Abstract

With the evolution of the Web to the Semantic Web, the use of software agents is increasing as the need arises for mechanisms to ensure interoperability between applications, due to the fact that the machines are processing the information provided. The software agents need information to support that work so effectively, they operate in open, heterogeneous environments. The most common proposal for this information used by software agents is the use of ontologies. Ontologies aim torepresent consensual knowledge among people and applications, enabling the reuseand sharing of information. The problem becomes how to reconcile differentontologies to ensure communication between software agents. In the literature, this problem is treated as compatibility of ontologies. In this paper, we present tools forcompatibility of different ontologies and have tested each tool set for some pairs of ontologies in order to identify differences and similarities between the approachestaken by these tools compatible.

Keywords: Ontologies, Interoperability, Test tools.

Agradecimentos

Primeiramente, agradeço aos meus pais, Rafael e Rosa, pelo amor incondicional e apoio irrestrito, bem como os ensinamentos sobre a importância do estudo na vida de uma pessoa.

Aos meus irmãos Rafael e Pablo, pela enorme amizade, o amor, e sonhos que construímos ao longo desses anos, mesmo não estando presentes em todos os momentos.

Ao Bruno, pelo companheirismo, paciência e cumplicidade em todos os momentos em que passamos juntos nessa caminhada.

Sumário

Lista das Figuras

Lista de Tabelas

1. Introdução

Encontra-se na *Web* um grande volume de informações disponibilizadas sem uma forma adequada para a representação de conhecimento. Desta maneira, o conteúdo das páginas *Web* é passível de ser entendido apenas por humanos; máquinas não obtêm suporte explícito para este tipo de tarefa. Frente a esta dificuldade, pesquisadores acadêmicos e da indústria vêem explorando a possibilidade de criar uma *Web* Semântica, e junto da mesma surgiu a possibilidade do uso de ontologias.

A palavra ontologia tem a sua origem nos pensamentos filosóficos de Aristóteles, que trata do ser enquanto ser.

A área da Computação resgatou este termo, inserindo-o em um novo contexto, apesar de próximo do significado original.

Dentre tantos significados iremos destacar a definição apresentada por (BORST, 1997), onde: Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada.

Na definição supra, *formal* significa legível para computadores; *especificação explícita* diz respeito a conceitos, propriedades, relações, funções, restrições, axiomas, explicitamente definidos; *compartilhado* quer dizer conhecimento consensual; e *conceitualização* diz respeito a um modelo abstrato de algum fenômeno do mundo real que se deseja representar. A ontologia é formada por um vocabulário controlado que é arranjado hierarquicamente e através de relações entre conceitos.

Na *Web* Semântica, onde as informações são heterogêneas e distribuídas, o uso de ontologias, ou especificações explícitas de uma conceituação, oferece uma representação comum para a troca das informações.

Através de ontologias, as informações são estruturadas utilizando-se um vocabulário livre de ambigüidades e com um formalismo passível de processamento automático sem nenhuma padronização, dificultando assim o seu reuso. Desta maneira, elas são criadas com o intuito principal de permitir o processamento automático de informação por agentes de *software,* que seriam capazes de processá-las e identificar o significado preciso dos termos e relações nelas contidas.

No entanto, para que trabalhem em conjunto, trocando automaticamente as informações representadas em ontologias, são necessários mecanismos que garantam a interoperabilidade destas estruturas.

1.1. Motivação

O processo de compatibilização de ontologias pode ser realizado de três formas, a automática, a semiautomática, onde há a necessidade de intervenção humana em algumas etapas para a tomada de decisão, ou até mesmo manual.

Atualmente, a necessidade do menor tempo possível é imprescindível, fazendo com que o processo de compatibilização de ontologia seja rápido e, se possível, automático.

O processo rápido se faz necessário pela própria natureza dinâmica da *Web*.

Entende-se que, ao ser disparado o pedido de compatibilização pelos agentes de *software*, a resposta deva ser, se possível, imediatamente depois da execução do processo, ou seja, em um tempo finito de execução, o que é considerada uma boa resposta desde que se respeite o limite de confiabilidade (mínima margem de erros).

As escolhas sobre as ferramentas a serem utilizadas para o processo são decididas balanceando suas contribuições com seus impactos. Qualquer nova escolha que faça com que um dos requisitos do processo não seja satisfeito será descartada.

Existem diferentes abordagens para a compatibilização de ontologias, e diversas ferramentas estão disponíveis, sendo que cada ferramenta utiliza uma abordagem distinta, dificultando para o engenheiro de ontologias a escolha de qual ferramenta utilizar, devido ao diferente resultado de cada uma delas.

1.2. Objetivos

Como possuímos diversos recursos para a compatibilização de ontologias, existem dificuldades de encontrarmos um que resulte em um resultado ótimo. Diante disso, o estudo desse projeto tem como objetivo contribuir para a escolha de uma ferramenta que atinja um resultado otimizado, ou seja, quando sabemos qual ferramenta utilizarmos para obtermos o melhor resultado para um par de ontologias.

Testar a viabilidade de uso de diferentes ferramentas de comparação de ontologias e avaliar estas ferramentas consiste no objetivo desse trabalho.

1.3. Organização

Esse trabalho será dividido em cinco capítulos.

O presente capítulo trata da introdução e apresenta os conceitos iniciais para um melhor entendimento do trabalho.

No segundo capítulo, os conceitos de compatibilização de ontologias serão esclarecidos separadamente. São eles, combinação de ontologias, alinhamento de ontologias, mapeamento de ontologias e integração de ontologias.

No terceiro capítulo, é apresentada a especificação das ferramentas, PROMPT (Noy e Musen, 2003) e GNoSIS, ferramenta desenvolvida no Núcleo de Pesquisa em Engenharia de Conhecimento (NEnC) da Universidade Federal de Juiz de Fora (UFJF), que foram selecionadas para o trabalho.

Por fim, no quarto e quinto capítulos, encontram-se, respectivamente, o estudo comparativo das mesmas e à conclusão do trabalho, juntamente com as propostas de trabalhos futuros.

2. Compatibilização de Ontologias

A maior dificuldade encontrada no reuso de informações de ontologias é devido à falta de um padrão para sua criação, pois são construídas utilizando diferentes linguagens e cada linguagem tem a sua sintaxe, expressividade e capacidade de raciocínio, bem como, são baseadas em diferentes paradigmas (frames, lógica de primeira ordem, descritores lógicos, etc.), com isso a fase de aquisição do conhecimento para a implementação é praticamente descartada, gerando diversos problemas como (Guimarães, 2002):

- A conceitualização da ontologia não fica muito clara no código da implementação;
- A falta de padronização acaba por dificultar seu reuso, pela dificuldade de compreensão;
- Gera dificuldade para a implementação de ontologias mais complexas, devido à falta do uso de uma metodologia, o que torna a transferência do conhecimento para a implementação mais difícil.

Devido a estes problemas encontrados tem-se a necessidade da adoção de metodologias para a construção de ontologias, reduzindo assim as dificuldades encontradas em suas implementações, e facilitando a reutilização com os mecanismos que garantam a interoperabilidade.

As ontologias têm como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre sistemas de aplicação e pessoas.

Mas para que possam trabalhar em conjunto, trocando automaticamente as informações representadas em ontologias, são necessários mecanismos. Dentre os mecanismos que podem ser usados para compatibilidade de ontologias, destacamos: combinação de ontologias (Noy e Musen, 1999), alinhamento de ontologias (Noy e Musen, 1999), mapeamento de ontologias (Noy e Musen, 2003), integração de ontologias (Pinto, 1999).

2.1. Combinação de Ontologias

Na combinação de ontologias tem-se como resultado a versão das ontologias originais combinadas em uma ontologia única, com todos seus termos juntos e sem a

definição clara de suas origens. Normalmente as ontologias originais descrevem domínios similares ou de sobreposição.

A Figura 1 ilustra um exemplo de combinação de ontologias, onde os dois conceitos compatíveis, por exemplo, carro da ontologia *O1* e o veículo da ontologia *O2*, são combinados ou unidos, na ontologia única *O*.

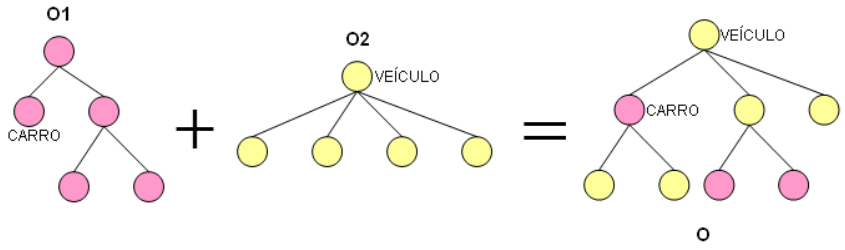


Figura 1: Exemplo do mecanismo de combinação de ontologias

Temos que *O1 + O2 = O3*, o conceito carro da ontologia *O1* é um tipo de veículo, sendo conceitos compatíveis podemos combinar as ontologias *O1* e *O2* na ontologia *O*.

Quando dois agentes com domínios similares ou de sobreposição, interagem combinando as suas informações, é possível efetuar a troca de conhecimentos.

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*.

Buscando uma melhor comunicação entre um agente colaborador *A* com conhecimentos de enfermagem e um agente colaborador *B* com conhecimentos de medicina, cada um utilizando uma ontologia de domínio específico de sua área, no momento em que eles interagirem para trocarem informações, pode-se aplicar uma técnica de combinação de ontologias para ter como resultado uma única ontologia com as informações que as suas áreas possuem em comum, conforme demonstrado na Figura 2.

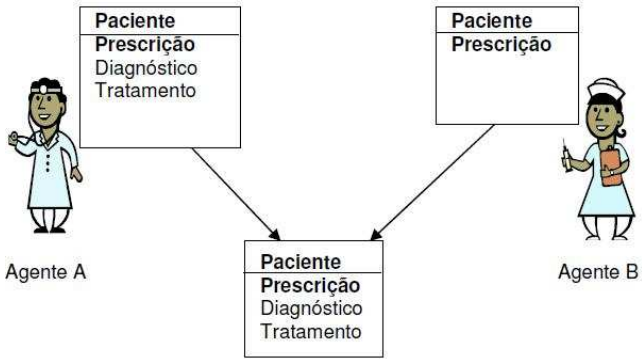


Figura 2: Exemplo de combinação de ontologias (HINZ e PALAZZO, 2007)

No exemplo anterior, a classe paciente possui a propriedade prescrição, que é definida como propriedade objeto, justamente por ser comum nas áreas de medicina e enfermagem, podendo se relacionar e formar uma única ontologia. As demais propriedades são definidas como propriedade de tipo de dados.

2.2. Integração de Ontologias

Na integração de ontologias tem-se como resultado uma ontologia única, criada pela montagem, extensão, especialização ou adaptação de outras ontologias de assuntos diferentes, sendo possível identificar as regiões que foram criadas a partir das ontologias originais.

A Figura 3 ilustra um exemplo de integração de ontologias, onde os conceitos automóveis de *O1*, carro de *O2* e veículo de *O3* são integrados ou unidos, na ontologia única *O*.

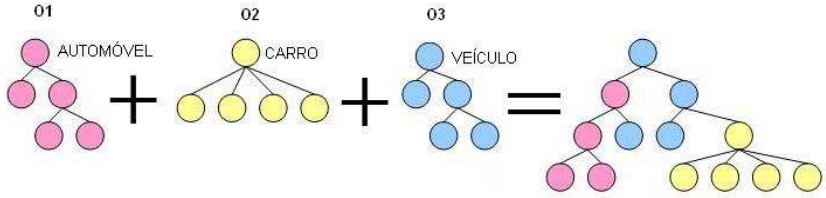


Figura 3: Exemplo do mecanismo de integração de ontologias

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 mecanismo é que no primeiro as ontologias tratam do mesmo assunto, o que não acontece necessariamente no segundo. Sendo assim, a integração

possibilita a expansão da informação de uma ontologia, abrangendo um conhecimento maior de um determinado domínio.

Na Figura 4 é exemplificado o mecanismo de integração de ontologias, onde possui um agente colaborador *A* com conhecimentos de medicina e outro agente colaborador *B* com conhecimentos de música.

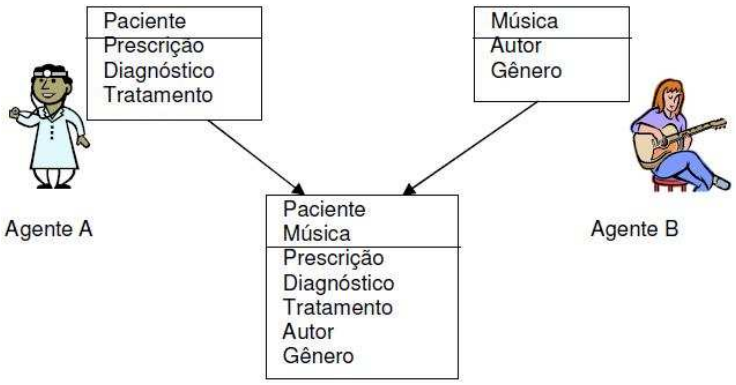


Figura 4: Exemplo do mecanismo de integração de ontologias (HINZ e PALAZZO, 2007)

Quando esses agentes interagirem irá formar uma única ontologia, contendo todos os dados das duas ontologias. Esse resultado poderia ser uma ontologia de musicoterapia.

2.3. Alinhamento de Ontologias

O alinhamento de ontologias pode ser entendido como o processo a que as diversas aplicações de sistemas abertos, com suas diferentes ontologias, terão que se submeter para garantir uma representação intermediária da informação que poderá ser compartilhada entre elas.

O alinhamento difere da integração e da combinação em relação ao seu resultado, pois, mantém as duas ontologias originais separadas, mas com as ligações estabelecidas entre elas, permitindo que as ontologias alinhadas reusem as informações uma das outras. O alinhamento normalmente é realizado quando as ontologias são de domínios complementares (Noy, Musen e Smart, 1999).

A Figura 5 ilustra um exemplo de alinhamento de ontologias, onde a ontologia *O1* se refere a carros, a ontologia *O2* se refere a veículos e a *O3* os meios de transporte.

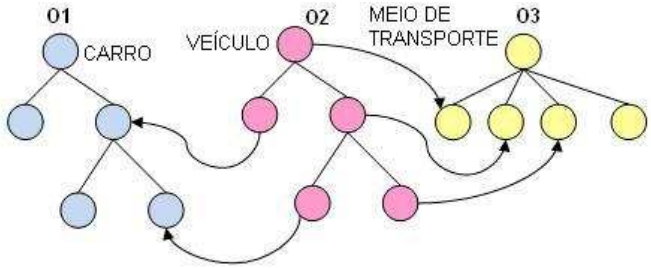


Figura 5: Exemplo do mecanismo de alinhamento de ontologias

No exemplo acima, foram alinhados os termos da ontologia *O1* referente a carros, da ontologia *O2* referente a veículos e da ontologia *O3* referente aos meios de transportes, uma vez que esses domínios são complementares.

As setas indicam que os termos da ontologia *O2* foram alinhados com os termos das ontologias *O1* e *O3*, esses alinhamentos ou ligações podem ser realizados considerando diferentes informações.

As informações contidas nesse processo dependem do tipo de ligação semântica encontrada entre os elementos e do tipo de formalismo utilizado na ontologia para representar a sua semântica durante o alinhamento. Por exemplo, dois elementos podem ser semelhantes (em diferentes graus), ou um pode ser parte do outro, ou então podem ter algum outro tipo de relacionamento que é identificado com o auxílio de um especialista no domínio.

Um dos aspectos do alinhamento é a questão de como achar os candidatos. Esta pode se basear, dentre outros: (i) na semelhança dos nomes dos termos; (ii) na estrutura da ontologia, como, por exemplo, levando em conta o posicionamento dos termos na estrutura hierárquica das ontologias sendo comparadas ou então as suas relações partitivas ou ainda outros tipos de relações que sejam utilizadas de forma semelhante nas ontologias comparadas (Euzenat e Shvaiko, 2007); (iii) na adição de conhecimento adicional, como, por exemplo, nas informações de uma terceira ontologia ou vocabulário que possua uma hierarquia de conceitos, como a Wordnet (Miller, 1990), que pode ser utilizada, por exemplo, para procura de sinônimos ou para confronto da distância do posicionamento dos termos das ontologias sendo alinhadas em relação a essa terceira ontologia (Reynaud e Safar, 2007) (Sabou, 2006).

Um exemplo prático ilustrado na Figura 6, considera um agente colaborador *A* com conhecimentos específicos de comidas e outro agente colaborador *B* com conhecimentos de bebidas, ambos pertencem ao ramo da alimentação.

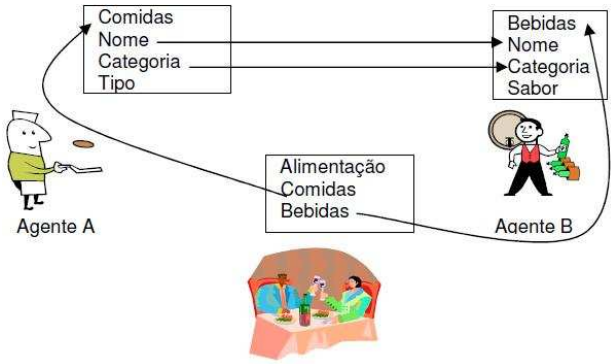


Figura 6: Exemplo do mecanismo de alinhamento de ontologias (HINZ e PALAZZO, 2007)

Quando realizado o alinhamento existirão propriedades que serão complementares ao ramo da alimentação.

2.4. Mapeamento de Ontologias

No mapeamento de ontologias tem-se como resultado uma estrutura formal com expressões que ligam os termos de uma ontologia nos termos de outra ontologia. Este mapeamento pode ser usado para transferir instâncias de dados, esquemas de integração e de combinação, e outras tarefas similares.

A Figura 7 ilustra um exemplo de mapeamento de ontologias, onde os conceitos meio de transporte de *O1* e veículo de *O2* são mapeados em expressões formais.

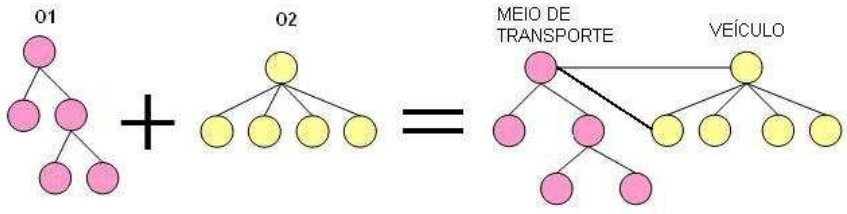


Figura 7: Exemplo do mecanismo de mapeamento de ontologias

Na Figura 7, o termo meio de transporte liga ao termo veículo e a outro termo *X* qualquer, filho de veículo. Em nível de instâncias, podemos executar essas relações semânticas transformando instâncias da ontologia de origem em instâncias da ontologia de destino, ou seja, *meio de transporte = veículo* ou *meio de transporte = X*, com isso podemos concluir que meio de transporte é semanticamente equivalente a veículo e a *X*.

Essas relações semânticas são consideradas como pontes semânticas, pois, elas criam uma integração de informação entre as ontologias.

Além de o mapeamento ajudar no reuso de ontologias, ele serve para expandir e combinar as mesmas, com o intuito de aumentar o conhecimento e a informação de diferentes domínios que são integrados para suportar nova comunicação e uso.

A ilustração da Figura 8 exemplifica o mapeamento de ontologias.

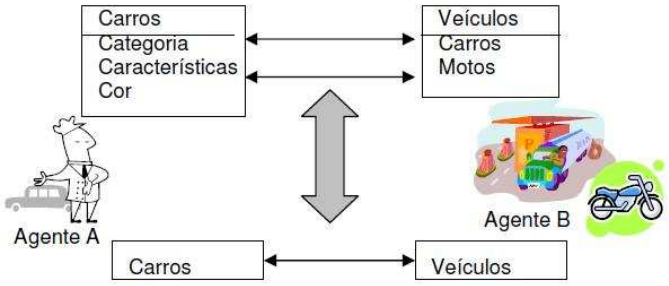


Figura 8: Exemplo do mecanismo de mapeamento de ontologias (HINZ e PALAZZO, 2007)

Nesse exemplo o agente *A* possui conhecimentos de carros e o agente *B* de veículos e tem como resultado uma estrutura formal com expressões que ligam os termos de um aos termos do outro, como categorias de carros e características das motos.

2.5. Conclusão

Esses mecanismos são considerados como um processo para o reuso de ontologias, e apresentam diferentes níveis de comprometimento no que diz respeito à comparação entre ontologias, considerando o alinhamento ao menor nível e, em seguida, mapeamento, integração e o de maior nível combinação.

O alinhamento estabelece apenas correspondências, ligações entre termos de ontologias, enquanto no mapeamento existe uma infraestrutura para transferir instâncias de uma ontologia fonte para uma ontologia alvo (Noy e Musen 2002).

No processo de integração uma nova ontologia é criada a partir do reuso de outras mesmo mantendo as referências para as ontologias fonte. Finalmente, o mecanismo de combinação gera uma nova ontologia sem preservar nenhuma ligação ou referência com as ontologias originais.

Entretanto, todos esses mecanismos são precedidos por uma técnica denominada na literatura como casamento ou *matching* (Euzenat e Shvaiko 2007), embora outros trabalhos

também considerem essa técnica como alinhamento (Euzenat e Valtchev 2004), (Euzenat et al. 2008) (Jean-Mary e Kabuka 2007) (Kiu e Lee 2007), ou mesmo como mapeamento (Nagy et al. 2007).

Independentemente do nome, essa técnica consiste em comparar termos de uma ontologia alvo com os termos de uma ontologia fonte, de modo que medidas de similaridade léxica, estrutural e semântica possam ser definidas entre esses termos.

Assim, quanto mais refinados forem os resultados de similaridades retornados pelo processo de casamento, serão melhores os resultados fornecidos pelo mecanismo de interoperabilidade.

3. Ferramentas para Compatibilização de Ontologias

Ferramentas que executam a tarefa de compatibilização de ontologias são especialmente importantes devido às muitas diferenças que podem existir entre as ontologias, tanto na estrutura de classes como nos nomes de conceitos. Nas seções seguintes, descreveremos as ferramentas, PROMPT (Noy e Musen, 2003) e a GNoSIS, ferramenta desenvolvida no Núcleo de Pesquisa em Engenharia de Conhecimento (NEnC) da Universidade Federal de Juiz de Fora (UFJF).

3.1. PROMPT - Protegé

O PROMPT é uma ferramenta semi-automática que tem como abordagem principal, a combinação, o alinhamento e o mapeamento de ontologias de forma semi-automática.

É dividida em um conjunto de ferramentas como o iPROMPT, uma ferramenta interativa para combinação de ontologias, o AnchorPROMPT, uma ferramenta automática baseada em grafos que visa encontrar similaridade entre ontologias, o PROMPTFactor, uma ferramenta para extração de partes de ontologias e o PROMPTDiff, uma ferramenta para identificação de diferenças estruturais entre duas versões da mesma ontologia.

A ferramenta iPROMPT é interativa, ajuda os usuários na tarefa de combinação das ontologias fornecendo sugestões sobre como os elementos podem ser combinados.

Essa comunicação com o usuário se dá através de identificação de inconsistências e possíveis problemas encontrados nas ontologias pela ferramenta. As sugestões de combinações fornecidas surgem, sendo as possíveis estratégias para resolver esses problemas e inconsistências. Os autores da ferramenta consideraram como um conjunto de possíveis sugestões de combinação, as combinações de classes, de propriedades, de ligações entre uma propriedade e uma classe, entre outras.

Para a identificação de inconsistência foram considerados os possíveis conflitos de nome, referências pendentes, a redundância na hierarquia das classes e as restrições de valor das propriedades que violam a herança da classe.

A ferramenta AnchorPROMPT estende o desempenho da ferramenta iPROMPT, pois não analisa apenas o contexto estrutural, é capaz de gerar como resultado, um alinhamento das ontologias que é obtido automaticamente encontrando termos semanticamente similares. Para isso, essa ferramenta tem como entrada par de termos

relacionados, definidos pelo usuário ou por identificação automática de combinação lexical, e trata uma ontologia como um grafo.

Neste grafo, os conceitos das ontologias são seus nós e suas relações são suas ligações. Os caminhos do sub-grafo limitado pelos pares de termos são analisados e são determinados quais os conceitos que freqüentemente aparecem nas mesmas posições dos caminhos similares. Com estas análises e determinações, além do contexto local, o contexto não-local também é analisado.

O AnchorPROMPT utiliza na detecção de termos similares, a relação estrutural dos termos de ontologias comparadas, medidas de similaridades pré-definidas e grupos de equivalência. Os termos determinados com uma maior freqüência na mesma posição serão apresentados ao usuário para melhorar o conjunto de sugestões possíveis.

Caso as duas ontologias comparadas apresentem grandes diferenças no número de níveis da hierarquia ou do número de relações entre as classes, o algoritmo não funciona corretamente.

A PROMPTFactor é uma ferramenta que permite aos usuários criar uma nova ontologia *fatorada*, ou seja, uma ontologia diferente da original por ter tido algumas partes do seu conteúdo extraída. Neste processo, a ferramenta garante que os termos da ontologia resultante são bem definidos (por exemplo, todos os conceitos da nova ontologia inclui de forma adequada os superconceitos e/ou subconceitos necessários para a sua especificação). Essa ferramenta não será detalhada, pois não se trata do tema do trabalho.

O PROMPTDiff é usado para realizar apenas uma comparação estrutural de duas versões de uma mesma ontologia e identificar as diferenças entre elas , isto é, em classes, em propriedades e/ou em instâncias, identificar alterações apenas em propriedades, e em qualquer parte da sua definição. Mas isso não significa que exista o controle das versões por parte da ferramenta, pois não existe, para o usuário indentificar e gerenciar as versões de uma mesma ontologia precisa encontrar uma forma que melhor se adequa a ele.

Estas ferramentas compartilham alguns componentes como: interface do usuário, heurísticas, e estruturas de dados, que fornecem informação uma para outra.

O Protegé é uma ferramenta que possuí com extensão de todas essas ferramentas para gerenciamento de múltiplas ontologias. A Figura 9 apresenta a infraestrutura do PROMPT como extensão da ferramenta Protegé, e as interações entre suas ferramentas.

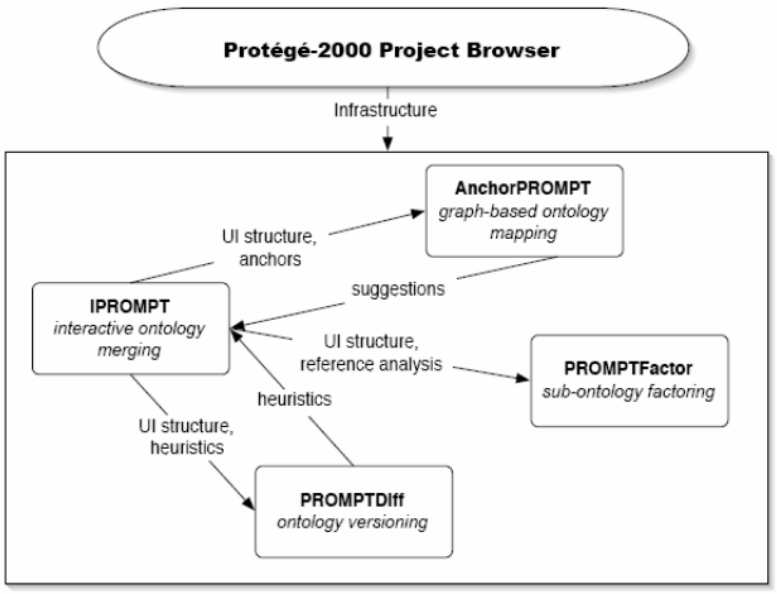


Figura 9: Infraestrutura do PROMPT (NOY E MUSEN, 2003)

3.1.1. iPROMPT

O algoritmo do iPROMPT possui como entrada duas ontologias, e define um conjunto de etapas para o processo interativo de combinação como o ilustrado pela Figura 10.

Figura 10: O usuário seleciona uma operação e o restantes das operações são realizadas pelo iPrompt (NOY E MUSEN, 2003)

Após a seleção da opção *merge* e da configuração do algoritmo, o processo inicia fornecendo uma lista de sugestões de combinações com base na configuração do algoritmo. A configuração pode ser a medida de similaridade lingüística entre os nomes das classes,

combinado com a estrutura interna dos conceitos e sua localização na ontologia, ou apenas uma delas.

A lista de sugestões é resultado do processamento do algoritmo e se resumem em termos que podem ser combinados. O algoritmo também sugere sugestões para que os problemas ou inconsistências encontrados sejam resolvidos. Os termos não comuns são sugeridos para serem copiados para a ontologia resultante.

O algoritmo usa medidas de similaridades simples para encontrar classes com nomes similares. Em seguida o processo passa pelo seguinte ciclo:

a) O usuário seleciona uma das sugestões do iPROMPT a partir da lista ou usando um ambiente de edição de ontologias para especificar os termos que deseja combinar, assim aciona essa operação a ferramenta;

b) O iPROMPT executa a operação, e em seguida executa automaticamente alterações adicionais com base no resultado da operação. Gera uma lista de sugestões para o usuário com base na nova estrutura da ontologia e nas incoerências e problemas encontrados ocasionados pela última operação à ontologia sugerindo as soluções possíveis para esses problemas.

A ferramenta iPROMPT, inicia as combinações por similaridade lingüística e ao decorrer do algoritmo a ferramenta busca encontrar informações baseadas na semântica da ontologia e nas ações do usuário. O resultado desse algoritmo é uma nova ontologia com partes das ontologias fontes, das informações obtidas pelo usuário e das interações resultantes das respostas do usuário.

Embora o algoritmo tenha sido implementado usando medidas simples de similaridade lingüística (Noy & Musen, 2003), a ferramenta iPROMPT foi desenvolvida de tal forma que outros algoritmos de comparação de termos possam ser *plugados* nela. O uso do *WorNet*, por exemplo, para encontrar sinônimos, pode ser uma extensão natural.

As medidas simples de similaridade lingüística são baseadas em *substring* correspondente ao nome do conceito, logo não faz uso de um banco de dados *lexical* para comparar sinônimo, e assim conseqüentemente, sofre problemas com palavras diferentes e do mesmo sentido.

Para exemplificar o algoritmo foi utilizado um exemplo de NOY E MUSEN, na Figura 11 temos a classe *M* que representa a combinação das duas classes, *A* e *B*.

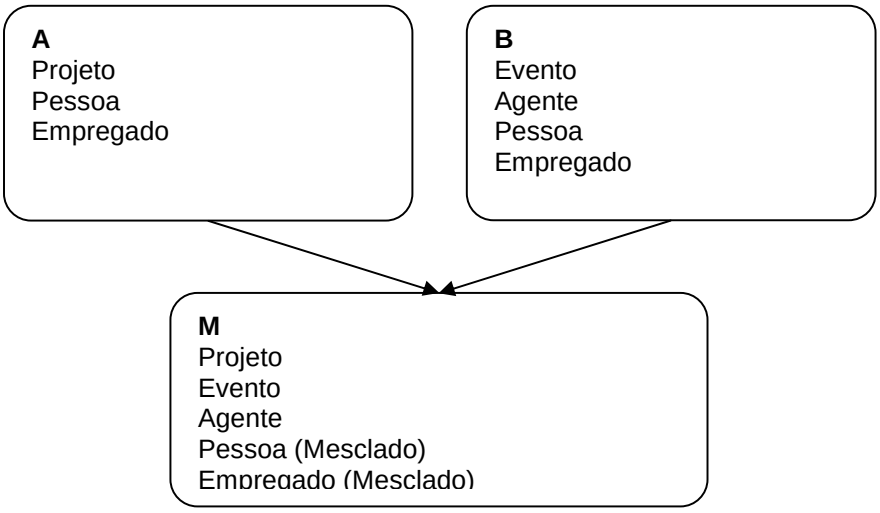


Figura 11: Combinação de Classes (NOY E MUSEN, 2003)

Suponha que um usuário deseja combinar as ontologias *A* e *B*, então ele executa uma operação de combinação de classes para criar uma nova ontologia *M*, conforme demonstrado na figura anterior. Para executar a operação de combinação de classes, o iPROMPT executa as seguintes ações:

a) Cria uma nova classe;

b) Se (nome da classe *A*) = (nome da classe *B*) então atribua esse nome para a nova classe. Senão, solicite ao usuário qual o nome a escolher (a menos que ele tenha designado uma ontologia como preferência de nome, no nosso exemplo o nome é *M*);

c) Para toda superclasse de *A* ou *B* que possua uma imagem na nova ontologia, manter sua relação de superclasse em *M* (restaurando assim a relação original);

d) Para toda subclasse de *A* ou *B* que possua uma imagem na nova ontologia, manter sua relação de subclasse em *M*. No exemplo, a classe Agente em *M* é superclasse de Pessoa (Mesclado);

e) Para todo *slot* que foi anexado a *A* ou *B* manter sua imagem em *M*;

f) Se alguns dos *slots* anexado a *A* ou *B* teve restrições de cardinalidade local então copie as restrições para sua imagem em *M*;

g) Se alguns dos *slots* anexados a *A* ou *B* teve restrições de escala local então copie as restrições para sua imagem em *M*;

h) Para cada par de *slots* que têm nomes lingüisticamente similares na classe *M*, devem ser mesclados. Caso o usuário opte por mesclar outros *slots*, as classes restringem os valores desses *slots* e eles são mesclados;

i) Para cada par de superclasses e subclasses de *M* que possuem nomes lingüisticamente similares, são mesclados;

j) Verifique se existe redundância na hierarquia em *M*. Se houver mais de um caminho para qualquer superclasse de *M* (exceto a raiz), um dos pais devem ser removidos.

3.1.2. AnchorPROMPT

AnchorPROMPT possui como entrada um conjunto de pares de termos relacionados (âncoras) a partir das ontologias fonte, que são obtidos manualmente pelo usuário ou automaticamente pelo sistema. A partir desse conjunto de âncoras previamente identificados, AnchorPROMPT produz um conjunto de novos pares de termos semanticamente próximo através dos seguintes passos:

a) Percorre os caminhos entre as âncoras nas ontologias correspondentes.

b) Então compara os termos desses caminhos para encontrar o valor do grau da similaridade entre eles. O valor do grau da similaridade é encontrado analisando os conceitos que freqüentemente aparecem nas mesmas posições dos caminhos similares (Felicíssimo, 2004). Além disto, utiliza na detecção de termos similares, a relação estrutural dos termos de ontologias comparadas, medidas de similaridades pré-definidas e grupos de equivalência, caminhos diferentes encontrados entre as âncoras.

c) Repete o processo para todos os caminhos existentes que se originam e terminam nos pontos da âncora. O grau de similaridade é acumulado.

No AnchorPROMPT é considerado que se dois pares de termos das ontologias são semelhantes e há caminhos conectando os termos, então os elementos nesses caminhos provavelmente são similares. Portanto, a partir de um pequeno conjunto de termos relacionados, o AnchorPROMPT é capaz de sugerir um grande número de termos que possuem grandes chances de serem semanticamente similares.

Na Figura 12, o retângulo representa as classes, as setas azuis representam as propriedades que se relacionam com as classes, às setas sólidas representam as âncoras e as setas tracejadas representam os termos relacionados.

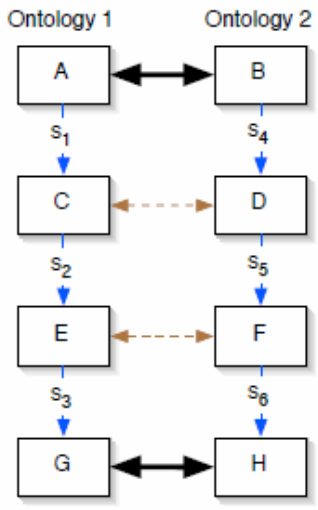


Figura 12: Alinhamento de Ontologia (NOY E MUSEN, 2003)

Supondo que a classe *A* da ontologia 1 seja semelhante a classe *B* na ontologia 2, e a classe *G* da ontologia 1 seja semelhante a classe *H* da ontologia 2 e observando que as ontologias possuem um caminho de *A* a *G*, na primeira ontologia e um caminho de *B* a *H* na segunda ontologia.

Percorrendo os dois caminhos em paralelo, caso o fim do percurso seja a mesma etapa para ambas ontologias a cada duas classes percorridas, é incrementado o grau de similaridade conforme a semelhança entre os nomes das classes e entre os nomes das classes encontradas no caminho.

Por exemplo, depois de percorrermos os caminhos de *A* a *E* e de *B* a *F*, incrementamos o grau de similaridade entre as classes *C* e *D* e entre as classes *E* e *F*. Esse processo é executado para todos os caminhos existentes que se originam e terminam nos pontos de *ancoragem*.

Os autores utilizaram o seguinte processo para calcular o grau de similaridade entre duas classes *G(C1, C2)*, onde *C1* é uma classe de uma ontologia e *C2* é uma classe de outra ontologia:

a) Gerar um conjunto de todos os caminhos de comprimento inferior a um parâmetro *L* que ligam as âncoras de entrada nas ontologias.

b) A partir do conjunto de caminhos gerados no passo anterior, gera um conjunto de todos os pares possíveis de caminhos de comprimento igual onde um caminho seja de uma ontologia e o par desse caminho seja na outra ontologia.

c) Para cada par de caminhos no conjunto gerado no passo anterior e, para cada par de nós *(N1, N2)* localizados nas posições idênticas nos caminhos, incrementar o grau

de similaridade entre cada par de classes respectiva dos nós *(C1, C2)*, por um *X* constante (Lembre-se que *N1* e *N2* podem ser individuais ou grupos de classes de equivalência que incluem várias classes). Portanto, o grau de similaridade *G(C1, C2)* é uma pontuação acumulativa que reflete a freqüência que *C1* e *C2* aparecem em posições idênticas ao longo dos trajetos, considerando todos os caminhos possíveis entre as âncoras (de comprimento inferior a *L*).

Para ilustrar como funciona o AnchorPROMPT, vamos considerar duas ontologias para representar os experimentos clínicos, os seus protocolos, aplicações e resultados, como na Figura 13.

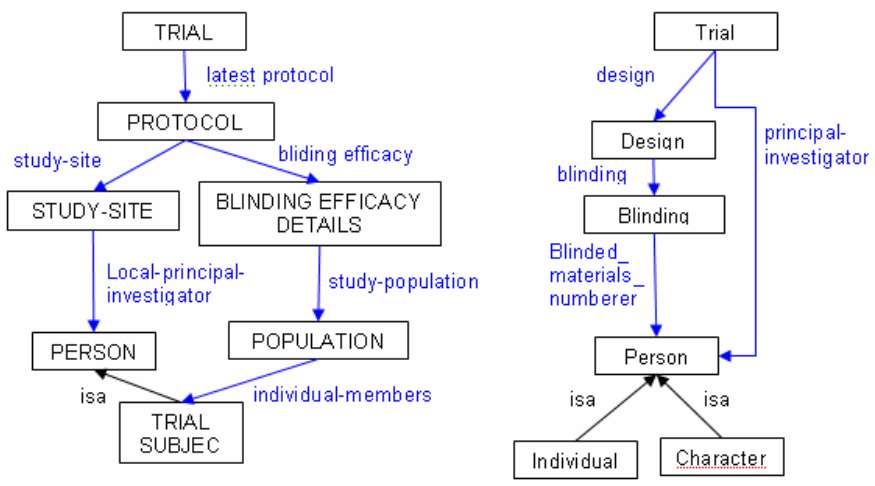


Figura 13: Exemplo da Ferramenta AnchorPROMPT (NOY E MUSEN, 2003)

A primeira ontologia, o *Design-a-Trial* (DAT) ontologia (Modgil, 2000), subjacente a um sistema baseado em conhecimento que ajuda os médicos à produção de protocolos para experimentos clínicos randomizados. Na segunda ontologia (Sim, 1997), os experimentos clínicos randomizados (RCT) são usados na criação de bancos de processo eletrônico que armazena os resultados dos experimentos clínicos, proporcionando assim aos pesquisadores, encontrar, avaliar e aplicar os resultados.

Ambas as ontologias representam experimentos clínicos, mas um deles, DAT, concentra-se na definição de um protocolo de experimento próprio, enquanto que a RCT, concentra-se em que representa o resultado do experimento.

As duas ontologias foram desenvolvidas completamente independentes uma do outro. Portanto, não há correlação intencional entre eles.

Considerando o par de âncoras, *Trial* (Experimento) e *Person* (Pessoa) como entrada, o algoritmo determina pares de outros termos relacionados nas ontologias RCT e DAT. Ele gera um conjunto de todos os caminhos entre a *Person* e *Trial* na ontologia RCT e DAT considerando apenas os caminhos que são mais curtos do que um comprimento de parâmetros pré-definidos.

Considere um par de caminhos deste conjunto que tem o mesmo comprimento:

- DAT: *TRIAL* → *PROTOCOL* → *STUDY-SITE* → *PERSON*
- RCT: *Trial* → *Design* → *Blinding* → *Person*

À medida que atravessa os dois caminhos, AnchorPROMPT aumenta o *score* de similaridade, coeficiente que indica o quão perto os dois termos estão relacionados para os pares de termos nas mesmas posições nos caminhos.

Para os dois caminhos em nosso exemplo, vai aumentar o grau de similaridade para as duas seguintes pares de termos:

- *PROTOCOL* (Protocolo) → *STUDY-SITE* (Sites para estudos)
- *Design* → *Blinding* (Cegueira)

AnchorPROMPT repete o processo para cada par de caminhos do mesmo comprimento. Durante esse processo aumenta a pontuação de similaridade para os pares de termos que encontrar. Ele agrega o grau de similaridade de todas as travessias para gerar o grau de similaridade final.

Conseqüentemente, os termos que aparecem freqüentemente nas mesmas posições sobre os caminhos que vão de um par de âncoras para o outro receberá a maior pontuação.

3.1.3. PROMPTDiff

O algoritmo PROMPTDiff possui duas estruturas, o *diff* estrutural e a tabela PROMPTDiff.

Dadas duas versões de uma mesma ontologia, *V1* e *V2*, a comparação estrutural dessas versões *D(V1,V2)*, *diff* estrutural, é um conjunto de pares de *frames* <*F1*,*F2*> onde:

- *F1* ∈ *V1 or F1 = null*; *F2* ∈ *V2 or F2 = null*;
- *F2* é uma imagem de *F1*, ou seja, *F1* tornou-se *F2*. Se *F1* ou *F2* é *null* então *F2* e *F1* não tem correspondência;
- Cada *frame* de *V1* e *V2* é referenciado pelo menos em um par;

- Para qualquer *F1* existe pelo menos um par de *frames* contendo-o, onde *F2 ≠ null*, então não há par de *F1* com *F2 = null*. (Se encontrado apenas uma combinação de *F1*, não é possível afirmar que ele não possui combinações). O mesmo é verdade para F2.

A definição implica que para qualquer par de *frames* <*F1,F2*> há no máximo uma entrada.

O *diff* estrutural descreve os *frames* que foram alterados a partir de uma versão para outra. No entanto, para que uma comparação seja mais útil para o usuário, deve incluir não só o que mudou, mas também algumas informações sobre a forma como os *frames* mudaram.

Uma tabela PROMPTDiff oferece essas informações detalhadas.

Dadas duas versões de uma mesma ontologia, *V1* e *V2*, a tabela PROMPTDiff é um conjunto de tuplas <*F1, F2, rename-value, operation-value, mapping-level*> onde:

- Existe uma tupla <*F1, F2, rename-value, operation-value, mapping-level*> na tabela PROMPTDiff correspondente a um par <*F1, F2*> na estrutural diff D (V1, V2);
- O *rename-value = true* se os nomes do *frames F1* e *F2* são iguais, caso seja diferentes o *rename-value = false;*
- O operation-value Є OpS, onde *OpS = {add, delete,split, merge, map};*
- *O mapping-level Є MapS*, onde *MapS* = {*unchanged, isomorphic, changed*};

O *OpS* indica ao usuário como um frame mudou de uma versão para a outra: se foi adicionado ou excluído, se ele foi dividido em dois *frames*, ou se os dois *frames* foram fundidos.

A operação de mapeamento é para um par de quadros, caso nenhuma das outros operações se aplique.

O nível de mapeamento indica se os *frames* são diferentes um do outro de um para então justificar a atenção ao usuário. Se o nível de mapeamento é *unchanged*, então o usuário pode ignorar os *frames*, pois nada mudou em suas definições. Se dois quadros são *isomorphic*, então os *slots* correspondentes e tipos são imagens um do outro, mas não necessariamente imagens idênticas. O nível de mapeamento é *changed* se os *frames* possuem *slots* ou tipos que não são imagens do outro.

O algoritmo PROMPTDiff combina um número arbitrário de heurística para verificar as correspondências, onde cada uma verifica uma determinada propriedade dos *frames* e todas seguem um princípio, o princípio da *Monotonicidade*.

Seja *M* um algoritmo para verificar as correspondências, e *T1* e *T2* tabelas PROMPTDiff antes e após a execução de *M*. Então para cada dois *frames F1* e *F2* tal que *F1* $\in$ *V1* e *F2* $\in$ *V2* e se um par <*F1, F2*> esta presente em *T1* então também está presente em *T2*.

Uma heurística conforme o princípio da *Monotonicidade*, pode apagar linhas da tabela PROMPTDiff onde um dos *frames* são nulos e inserir novos pares de *frames*.

As heurísticas podem ser:

- *Frames* do mesmo tipo e nome: Se *F1* $\in$ *V1* e *F2* $\in$ *V2*, e *F1* e *F2* tem os nomes e os tipos iguais, *F1* e *F2* são correspondentes. Esses *frames* podem ser classes, propriedades ou instâncias.
- Únicos diferentes: Se *C1* $\in$ *V1* e *C2* $\in$ *V2*, e *C1* e *C2* são correspondentes, e cada uma das classes possuem uma única subclasse incomparáveis *subC1* e *subC2*, respectivamente, então *subC1 = subC2*.
- Prefixo ou sufixo diferente: Se *C1* $\in$ *V1* e *C2* $\in$ *V2*, e *C1* e *C2* são correspondentes, e todas as subclasses de *C1* tenha o mesmo nome de todas as subclasses de *C2*, exceto por um prefixo ou sufixo, então elas são correspondentes.
- *Slot* incomparável: Se *C1* $\in$ *V1* e *C2* $\in$ *V2*, e *C1* e *C2* são correspondentes, e cada uma das classes tem exatamente um *slot* inigualável, *S1* e *S2*, respectivamente, e *S1* e *S2* tem o mesmo tipo, então *S1* e *S2* são correspondentes.
- *Slots* inversos incomparáveis: Se *S1* $\in$ *V1* e *S2* $\in$ *V2*, *S1* e *S*2 correspondentes, e *invS1* e *invS2* são *slots* inversos de *S1* e *S2* respectivamente, e *invS1* e *invS2* são incomparáveis, então *invS1* e *invS2* correspondentes.
- Divisão de classes: Se *C0* $\in$ *V1* e *C1* $\in$ *V2* e *C2* $\in$ *V2*, e para cada individuo de *C0*, sua imagem é um individuo de *C1* ou *C2*, então *C0* foi divida em *C1* e *C2*. Identifica assim classes que foram combinadas.

Portanto o algoritmo do PROMPTDiff é uma junção de todas essas heurísticas e estruturas definidas anteriormente, com o nome de ponto fixo, pois ele executa todas as heurísticas até que não exista mais alterações na tabela PROMPTDiff, e cada resultado de uma heurística é utilizada na outra executada em seguida.

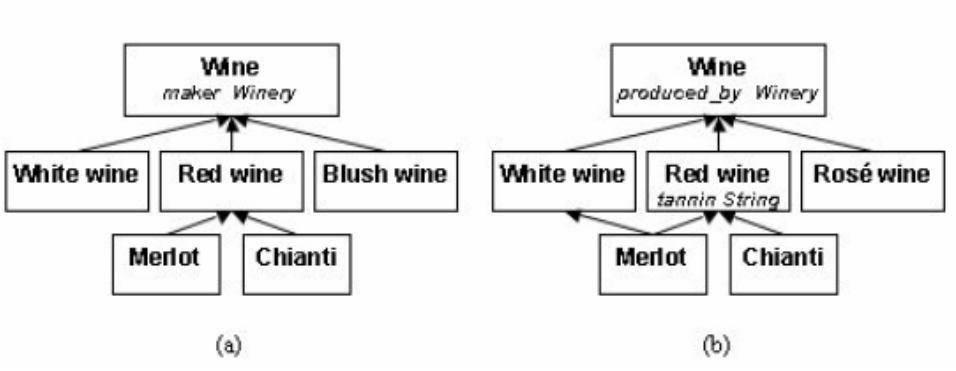
A Tabela 1 a seguir ilustra as dependências entre as heurísticas, o tipo de informação que modifica e utiliza.

Tabela 1: Informações das Heurísticas (NOY e MUSSEN, 2002)

Heurísticas	**Informações Usadas**		**Informações Modificadas**		**Heurísticas Dependentes**
	Classes	*Slots*	Classes	*Slots*	
Frames do mesmo tipo e nome	-	-	+	+	2,3,4,5
Únicos diferentes	+	-	+	-	2,3,4
Prefixo ou sufixo diferente	+	-	+	-	2,3,4
Slot incomparável	+	+	-	+	4,5
Slots inversos incomparáveis	-	+	-	+	4,5
Divisão de classes	+	-	+	-	2,3,4

Depois das alterações da tabela PROMPTDiff, não é necessário executar todas as heurísticas novamente. O PROMPTDiff usa uma tabela para verificar a ordem de dependência em que é necessário executar as heurísticas. Ele mantém uma pilha de verificações que ainda precisam ser executadas. Começa por colocar as verificações que não afeta qualquer outra na parte inferior da pilha e verificações que não são afetados pelas heurísticas de outras na parte superior. Então ele executa a heurística do topo da pilha. Caso produza alterações na tabela PROMPTDiff, o algoritmo adiciona à pilha todas as heurísticas que dependem da heurística executada, removendo as duplicatas. Executa-se até que a pilha esteja vazia.

Para exemplificar o PROMPTDiff foi considerado uma ontologia sobre vinhos como a Figura 14, podemos observar duas versões.



f1	f2	renamed	operation	map level
	S tannin level	No	Add	
C Blush wine	C Rosé wine	Yes	Map	Isomorphic
S maker	S produced_by	Yes	Map	Isomorphic
C Chianti	C Chianti	No	Map	Isomorphic
C Merlot	C Merlot	No	Map	Changed
C Red wine	C Red wine	No	Map	Changed
C White wine	C White wine	No	Map	Changed
C Wine	C Wine	No	Map	Isomorphic
C Winery	C Winery	No	Map	Unchanged

(c)

Figura 14: Exemplo PROMPTDiff (NOY e MUSSEN, 2002)

Na primeira versão da ontologia (Figura 14 a), há uma classe de Vinho com três subclasses, vinho tinto, vinho branco e vinho blush. A classe Vinho tem uma propriedade fabricante cujos valores são instâncias de classe *Winery*. A classe vinho tinto tem duas subclasses, *Chianti* e *Merlot*. Na Figura 14 b mostra uma versão posterior da mesma ontologia. Observe as mudanças: mudamos o nome da propriedade *fabricante* para *produzida por* e no nome da classe vinho blush para Vinho rosê, nós adicionamos uma propriedade *teor de tanino* para a classe de vinho tinto, e nós descobrimos que a *Merlot* pode ser branco então adicionamos outra superclasse a classe *Merlot*.

Figura 14 c mostra as diferenças entre as duas versões em uma tabela produzida automaticamente pelo PROMPTDiff. As duas primeiras colunas são pares de correspondência entre os quadros das duas ontologias. A terceira coluna identifica ou não se algum quadro mudou. As duas últimas colunas especificar quanto o quadro mudou.

3.4. GNoSIS

A ferramenta GNoSIS foi desenvolvida no Núcleo de Pesquisa em Engenharia de Conhecimento (NEnc) e permite o cálculo de similaridade entre conceitos de ontologias através da especificação de diferentes funções de similaridade a serem executadas (Capobiango, 2010).

A ferramenta também permite avaliar as funções de similaridade e realizar uma escolha das funções mais adequadas durante o alinhamento das ontologias.

A Figura 15 apresenta o funcionamento básico da ferramenta é separado por módulos.

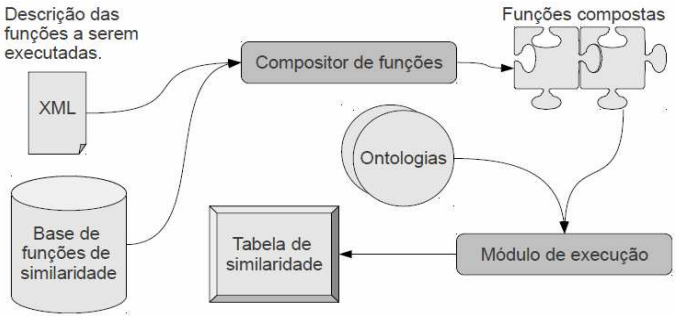


Figura 15: Funcionamento do sistema de execução de funções de similaridade

O módulo de composição de funções é responsável por compor as funções de acordo com a composição especificada em um arquivo XML. O módulo executor é responsável por recuperar os conceitos das ontologias a serem comparadas e aplicar as funções compostas em cada par de conceitos, gerando uma tabela de similaridade.

Para cada elemento ontológico (classe, propriedade, individuo, etc), um conjunto de funções de semelhança e de funções para o cálculo de distância de edição pode ser definido no XML para serem aplicadas nesse elemento. Caso não sejam definidas as funções para o cálculo de distância de edição, os construtores usados para criar as ontologias nas funções de semelhanças possuem uma função para o cálculo já associado a elas, todas essas funções serão detalhadas mais a frente.

O módulo de composição de funções possui as funções para cálculo de distância de edição. Este cálculo de distância de edição recebe duas cadeias de caracteres como entrada e computa a distância entre as cadeias, que é dado pelo número mínimo de inserções, eliminações ou substituições de caracteres que são necessárias para transformar uma cadeia de caracteres em outra.

Cada função de distância de edição permite um conjunto dessas operações. As funções implementadas na ferramenta são:

- *Equal*: Compara se os conceitos são exatamente iguais, ou seja não foi realizado nenhuma inserção, eliminação ou substituição de caractere, assim retorna 1. Caso o retorno seja 0, quer dizer que foi realizado pelo menos uma das operações acima.
- *Hamming distance*: É necessário que os dois conceitos sejam do mesmo comprimento, assim compara o número de posições nas quais elas diferem entre si, ou seja, o menor número de substituições necessárias para transformar um conceito no outro.
- *Levenshtein distance*: É dada pelo número mínimo de operações necessárias para transformar um conceito no outro. Essas operações são a inserção, eliminação ou substituição de um caractere.
- *Damerau–Levenshtein distance*: É dada pelo número mínimo de operações necessárias para transformar um conceito no outro. Essas operações são a inserção, eliminação, substituição de um caractere, mas a transposição de dois caracteres adjacentes, o que diferencia do *Levenshtein distance.*
- *Jaro–Winkler distance*: O cálculo *Jaro-Winkler* é uma melhoria no cálculo *Jaro*. Foram implementados juntos na ferramenta devido à necessidade do cálculo *Jaro* para o *Jaro-Winkler,* eles consideram que quanto maior a distância entre dois conceitos, maior sua semelhança, portanto foi adaptado nessa ferramenta o resultado ao contrário, fazendo com que seja mais intuitivo a visualização dos resultados entre as funções. Para o cálculo da função é utilizada uma métrica, sendo que esse cálculo é mais adequado para seqüências curtas.

Estas funções de semelhança podem ser especificadas para serem utilizadas em conjunto por quaisquer funções de semelhança que analisem os termos, por exemplo, análise de nome de conceitos, nome de propriedades, nome de indivíduos ou range de propriedades.

Além dessas funções, foram implementadas diversas funções de semelhanças para análise de relações, instâncias e hierarquia de conceitos. Essas funções são:

- *ConceptNameSimilarity*: Compara todos os conceitos das ontologias ou apenas os especificados no XML;
- *DirectDataTypePropertybyRangeEqualSimilarity*: Verifica se os ranges (tipos) dos *DataTypeProperty* relacionados diretamente aos conceitos são exatamente

iguais, utiliza apenas a função de similaridade *Equal*, que não precisa ser especificada no XML;

- *DirectDataTypePropertybyRangeSimilarity*: Verifica se os ranges (tipos) dos *DataTypeProperty* relacionados diretamente aos conceitos estão dentro de um mesmo grupo, esses grupos são definidos no algoritmo, os grupos das datas, dos números, das *strings*, dos coringas e das referências. Utiliza também apenas a função de similaridade *Equal*, que não precisa ser especificada no XML;
- *DirectDataTypePropertybyNameSimilarity*: Compara os nomes dos *DataTypeProperty* relacionados diretamente aos conceitos, utiliza como *default* a função de similaridade *DamerauLevenshteinEditDistance*.
- *DirectIndividualbyNameSimilarity*: Compara os nomes dos indivíduos relacionados diretamente aos conceitos, utiliza como *default* a função de similaridade *DamerauLevenshteinEditDistance*;
- *DirectObjectPropertybyNameSimilarity*: Compara os nomes dos *ObjectProperty* relacionados diretamente aos conceitos, utiliza como default a função de similaridade *DamerauLevenshteinEditDistance*;
- *DirectObjectPropertybyRangeSimilarity*: Compara os *ranges* (tipos) dos *ObjectProperty* relacionados diretamente aos conceitos, utiliza como *default* a função de similaridade *DamerauLevenshteinEditDistance*;
- *DirectSubClassSimilarity*: Compara as subclasses relacionadas diretamente aos conceitos especificados, utiliza como *default* a função de similaridade *DamerauLevenshteinEditDistance*;
- DirectSuperClassSimilarity: Compara as super-classes relacionadas diretamente aos conceitos especificados, utiliza como default a função de similaridade *DamerauLevenshteinEditDistance*.

Foi implementado também duas funções para realizar as combinações dos resultados das funções de semelhanças e retornar ao usuário um valor entre 0 e 1.

A função *FirstMatchCombination* é uma combinação simples, ela considera o primeiro resultado que recebe como o melhor da combinação dos conceitos em questão e em seguida retira os outros resultados desses mesmos conceitos, realiza esse processo para todos os outros conceitos. Já a função *DeepCombination*, busca o melhor resultado de cada combinação e desconsidera os piores, realiza essa busca para todos os outros conceitos. A função *DeepCombination* é a função *default* nas funções de semelhanças, sendo que é

possível trocá-la no XML.

O módulo executor recupera as informações do XML e executa as funções conforme especificadas no XML, em seguida retorna o resultado da função caso não exista mais de um elemento relacionado.

Para os casos onde existam mais de um elemento relacionado a ser usado nas funções de semelhança, como, por exemplo, conceitos que possuem várias propriedades ou conceitos com mais de uma subclasse ou superclasse, o resultado da função de semelhança aplicada ao conceito corresponde à média aritmética do resultado da aplicação da função de semelhança em cada um dos elementos, nos casos onde os dois conceitos possuem o mesmo número de elementos (propriedades, subclasses ou superclasses).

Se alguma informação estiver faltando a algum dos conceitos, por exemplo, os conceitos se diferem quanto ao número de propriedades que possui, é utilizada uma técnica para fazer uma média dos pesos, podendo introduzir uma penalidade negativa. É calculado da seguinte forma:

$$\rho(a,b) = somaSimilaridades() / (Lmin + penalidade * (Lmax - Lmin))$$

Onde *Lmin* e *Lmax* representam o número mínimo e o número máximo de propriedades do conceito (ou filhos ou pais do conceito), respectivamente, e *somaSimilaridades*() é uma função que realiza a soma das similaridades inseridas numa tabela *T* calculadas por uma função de similaridade qualquer. A penalidade pode ser definida com valores entre 0 (quando não importa a diferença no número de elementos de um conceito, resultando numa simples média aritmética) até 1 (quando a diferença no número de elementos é importante).

Assim, quanto mais próximo o número de elementos (uma vez que ρ é aplicado ao cálculo da função de semelhança entre propriedades, superclasses ou subclasses), maior será a similaridade entre os conceitos.

Para exemplificar a utilização da ferramenta, foram utilizadas as ontologias conforme a Figura 16.

Figura 16: Ontologias utilizadas para exemplificar o uso da ferramenta

Em seguida, foi criado o XML indicando as ontologias acima, a forma utilizada para o alinhamento, no nosso caso vamos comparar os conceitos das ontologias, a função do cálculo de edição, *Levenshtein distance* e a penalidade igual 1, conforme a Figura 17.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE similarityCalc (View Source for full doctype...)>
<similarityCalc>
- <container name="Funcoes principais">
  - <function weight="1.0">
      <class>br.ufjf.ontology.gnosis.similarity.structure.ConceptNameSimilarity</class>
      <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.LevenshteinEditDistance</strategy>
      <penalty>1.0</penalty>
    </function>
  </container>
- <ontologies>
    <ontology id="Ontologia1.owl" />
    <ontology id="Ontologia2.owl" />
  </ontologies>
</similarityCalc>
```

Figura 17: XML utilizado na ferramenta

A ferramenta retornou o resultado abaixo utilizando o XML e as ontologias acima.

Tabela 2: Resultados Gerados pela ferramenta GNoSIS

	Money Dolar	PurchaseableItems	Body	BodyWithNon AdjustableShutterSpeed	Camera	Range
Money	0.5	0.05	0.39	0.09	0.16	0.19
Purchaseable Item	0.06	0.94	0.0	0.24	0.12	0.12
Body	0.19	0.0	1.0	0.12	0.0	0.0
BodyWith NonAdjustable ShutterSpeed	0.12	0.24	0.12	1.0	0.09	0.06

Camera	0.19	0.11	0.0	0.09	1.0	0.16
Range	0.1	0.11	0.0	0.06	0.16	1.0

É importante ressaltar que esse simples exemplo é apenas para mostrar o funcionamento básico da ferramenta e o resultado foi devido a escolha da função *Levenshtein distance* e o alinhamento analisando apenas no nome dos conceitos.

Entretanto é possível utilizar as mesmas ontologias com funções de similaridades e formas de alinhamento diferentes, que foram implementadas na ferramenta podendo ser usadas em conjuntos, retornando assim resultados diferentes, podendo ser melhor ou não do exemplificado. Essa forma de utilização da ferramenta será mostrada mais à frente no capítulo 4.

3.5. Conclusão

Estes são apenas alguns dos muitos recursos existentes distribuídos na *Web*.

Para a seleção dos recursos, a utilização da linguagem OWL e os algoritmos diferentes para realizar a compatibilização foram levados em consideração.

4. Estudo Comparativo

Nesse capítulo é descrita a metodologia utilizada para realizar o estudo comparativo das ferramentas. É especificado cada passo para a obtenção dos resultados e em seguida é realizada a análise dos mesmos.

4.1. Metodologia

Para analisar o resultado das ferramentas, existem pelo menos dois tipos de abordagem de experimentação.

Na primeira, poderíamos utilizar a ferramenta para analisar duas ontologias distintas. A escolha das duas ontologias não é trivial, pois teríamos que encontrar duas ontologias com características de distinção que nos interesse. A outra forma seria utilizar uma única ontologia e realizar alterações nesta ontologia para analisar como as ferramentas se comportam diante de tais alterações. Neste estudo, iremos realizar essa segunda abordagem.

A escolha da segunda abordagem possibilita um estudo mais confiável, pois já sabemos quais são as diferenças entre as ontologias, logo podemos deduzir um resultado para a ferramenta. Com a primeira abordagem não poderiamos ter todo o conhecimento das diferenças nas ontologias, o que dificultaria a análise dos resultados.

Para o estudo das ferramentas foi escolhida a ontologia Celo disponível na *Web* (http:\\celo.mmc.ufjf.br\Celo.owl). A Celo foi desenvolvido como parte de uma tese de Mestrado em Modelagem Computacional na Universidade Federal de Juiz de Fora (UFJF). A ontologia captura a estrutura de um modelo celular e as propriedades dos componentes funcionais e foi desenvolvida em OWL-DL.

Na Figura 18 é ilustrada uma parte da ontologia em questão.

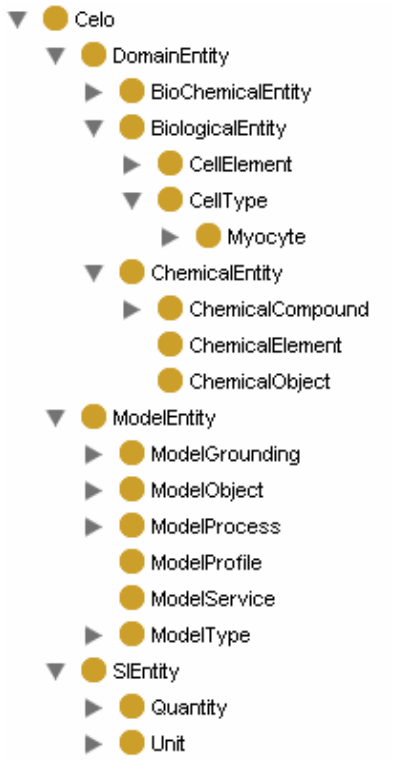


Figura 18: Parte da ontologia Celo

Com a ontologia Celo já selecionada para o estudo, será realizada uma modificação nos nomes dos termos da ontologia. Depois com a ontologia original uma modificação será realizada na estrutura da mesma e por fim as modificações dos termos e da estrutura na ontologia original.

A modificação nos nomes dos termos da ontologia será necessária para avaliar a execução das ferramentas na comparação taxonômica. A modificação da estrutura da ontologia será necessária para avaliar se as ferramentas realizam uma comparação hierárquica das ontologias, ou seja, verifica subclasses, superclasses, propriedades e etc.

Para a mudança dos termos, foi adotada a tradução dos termos, uma forma de padronizar a modificação, mas sem nenhuma justificativa técnica, essa mudança poderia ser realizada de qualquer outra forma.

A Figura 19 ilustra uma parte da ontologia Celo já modificada.

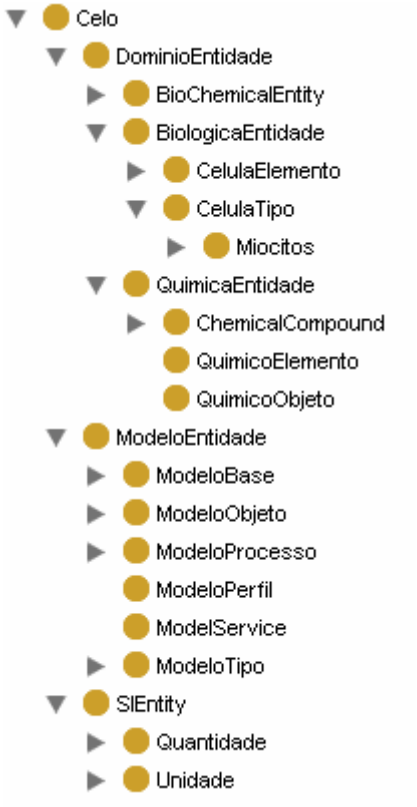


Figura 19: Parte da ontologia Celo modificada o nome dos termos

Para a modificação da estrutura da ontologia, foi alterada a hierarquia dos termos, conforme parte da ontologia na Figura 20.

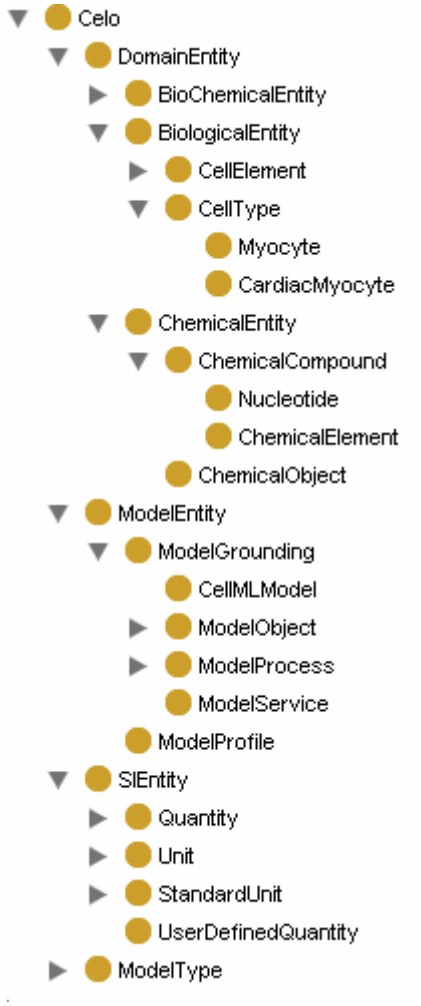


Figura 20: Parte da ontologia Celo modificada hierarquicamente

Por fim, será realizado na ontologia Celo as duas modificações acima para obter um terceiro resultado. A figura 21 ilustra parte da ontologia.

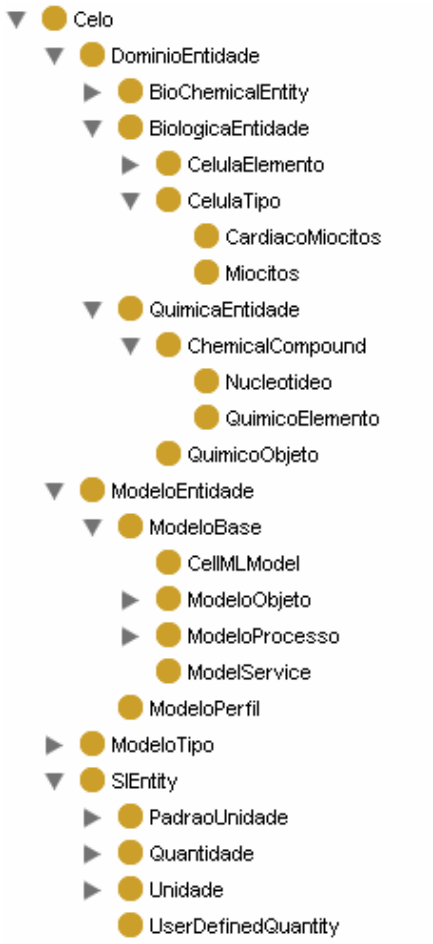


Figura 21: Parte da ontologia Celo modificada os nomes dos termos e hierarquicamente

Com as ontologias já modificadas o próximo passo é a execução das ferramentas, PROMPT e GNoSIS.

Cada ferramenta será executada com a ontologia original e com cada ontologia modificada para a obtenção dos resultados que serão analisados no próximo tópico.

4.2. Análise dos Resultados

Nessa seção será demonstrada a utilização de todas as ferramentas citadas anteriormente com os seus respectivos resultados.

Foi utilizada a ontologia Celo.owl, a CeloModTermo.owl, que possui a ontologia Celo.owl com alterações nos nomes dos termos, a CeloModEstrutura.owl, que possui a ontologia Celo.owl com alterações na sua estrutura, e a CeloModTermoEstrutura.owl, que possui as duas modificações acima.

4.2.1. iPROMPT

Na utilização do iPROMPT selecionamos dois projetos no Protegé.

Primeiro foi utilizado o projeto com as modificações dos nomes dos termos, *CeloModTermo.owl* e o projeto da ontologia *Celo.owl*. Selecionado os projetos, foi considerado na configuração do algoritmo para realizar a combinação, a medida de similaridade lingüística entre os nomes das classes, combinado com a estrutura interna dos conceitos e sua localização na ontologia.

Inicialmente a ferramenta exibe uma lista de sugestões para o usuário aceitar ou rejeitar. A Tabela 3 exibe as sugestões rejeitadas.

Tabela 3: Sugestões Rejeitadas no iPROMPT

Celo.owl	**CeloModTermo.owl**
IndependentVariable	DependentVariable
ModelParameterIn	ModelParameterOut
ModelParameterIn	ModelParameter
ModelParameter	ModelParameterOut
ModelParameter	ModelParameterIn
InternalComponent	ExternalComponent
Component	Compartment
ModelType	Modelo
ModelEntity	SIEntity
ModelParameterOut	ModelParameter
ModelParameterOut	ModelParameterIn
ChemicalEntity	BioChemicalEntity
NonLinearModel	LinearModelo
LinearModel	NonLinearModel
ExternalComponent	InternalComponent

Para cada sugestão aceita, a ferramenta exibe outras sugestões para a combinação e também sugestões para evitar conflitos existentes. Essas sugestões são para o usuário aceitar ou rejeitar. No nosso caso, aceitamos todas.

Por fim chegamos ao seguinte resultado da Figura 22.

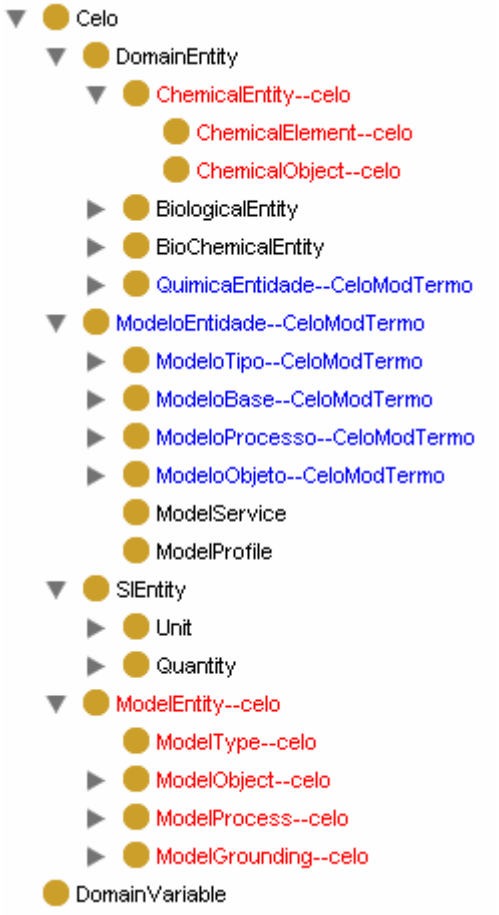


Figura 22: Parte do resultado do iPROMPT com a ontologia CeloModTermo.owl

Na Figura 22, as classes na cor vermelha representam as classes copiadas da ontologia original e as classes na cor azul representam as classes copiadas da ontologia modificada. A ferramenta reconheceu poucas classes com os nomes modificados, a maioria das operações fora adição das classes modificadas e das respectivas originais, o que representa inconsistência nos algoritmos de semelhanças.

O segundo teste foi realizado com a ontologia *CeloModEstrutura.owl*, seguindo os mesmos passos do teste anterior. A Tabela 4 exibe as sugestões rejeitadas.

Tabela 4: Sugestões Rejeitadas no iPROMPT

Celo.owl	**CeloModEstrutura.owl**
SIEntity	ModelEntity
BioChemicalEntity	ChemicalEntity
IndependentVariable	DependentVariable
ModelParameterIn	ModelParameter
ModelParameterIn	ModelParameterOut

ModelParameter	ModelParameterIn
ModelParameter	ModelParameterOut
InternalComponent	ExternalComponent
Component	Compartment
ModelEntity	SIEntity
ModelParameterOut	ModelParameterIn
ModelParameterOut	ModelParameter
ChemicalEntity	BioChemicalEntity
NonLinearModel	LinearModel
LinearModel	NonLinearModel
Compartment	Component
ExternalComponent	InternalComponent
DependentVariable	IndependentVariable

Com as sugestões iniciais analisadas e a aceitação de todas as demais sugestões nas outras etapas, chegamos ao seguinte resultado da Figura 23.

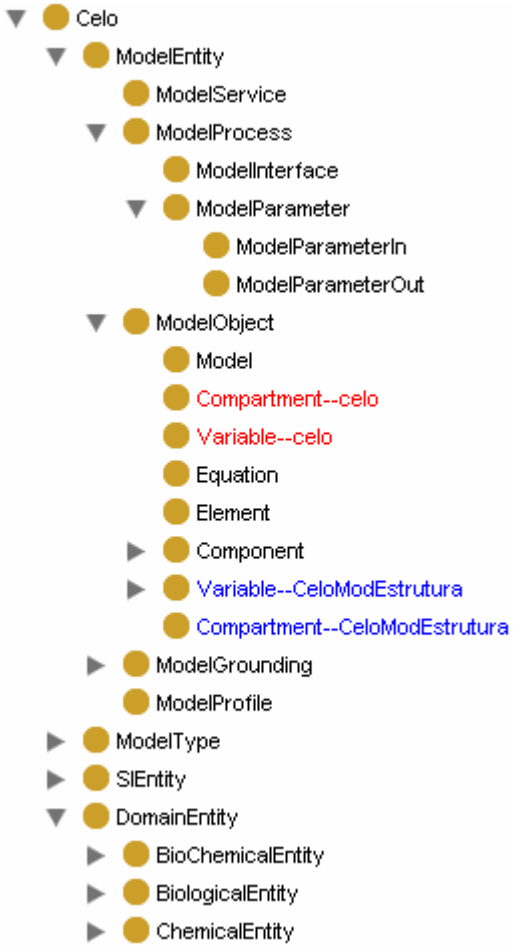


Figura 23: Parte do resultado do iPROMPT com a ontologia CeloModEstrutura.owl

Com a modificação estrutural a ferramenta reconheceu os nomes das classes bem mais do que na modificação dos nomes dos termos. Ocorreu uma inconsistência ou, outra adicionando duas vezes a mesma classe, como é exibido na Figura 23.

Ao meu ver a ferramenta apresentou um resultado mais sólido com a modificação apenas estrutural comparado com a modificação dos nomes dos termos.

O último teste da ferramenta foi realizado com a ontologia *CeloModTermoEstrutura.owl*, que possui as duas modificações acima em um única ontologia. Seguindo os mesmos passos dos testes anteriores, a Tabela 5 exibe as sugestões iniciais rejeitadas.

Tabela 5: Sugestões Rejeitadas no iPROMPT

Celo.owl	**CeloModTermoEstrutura.owl**
ModelParameterIn	ModelParameterOut
ModelParameterIn	ModelParameter
ModelParameter	ModelParameterOut
ModelParameter	ModelParameterIn
InternalComponent	ExternalComponent
Component	Compartment
ModelType	Modelo
ModelEntity	SIEntity
ModelParameterOut	ModelParameter
ModelParameterOut	ModelParameterIn
ChemicalEntity	BioChemicalEntity
NonLinearModel	LinearModelo
LinearModel	NonLinearModel
ExternalComponent	InternalComponent

Depois de todas as demais sugestões aceitas, a ferramenta retornou como parte do resultado a Figura 24.

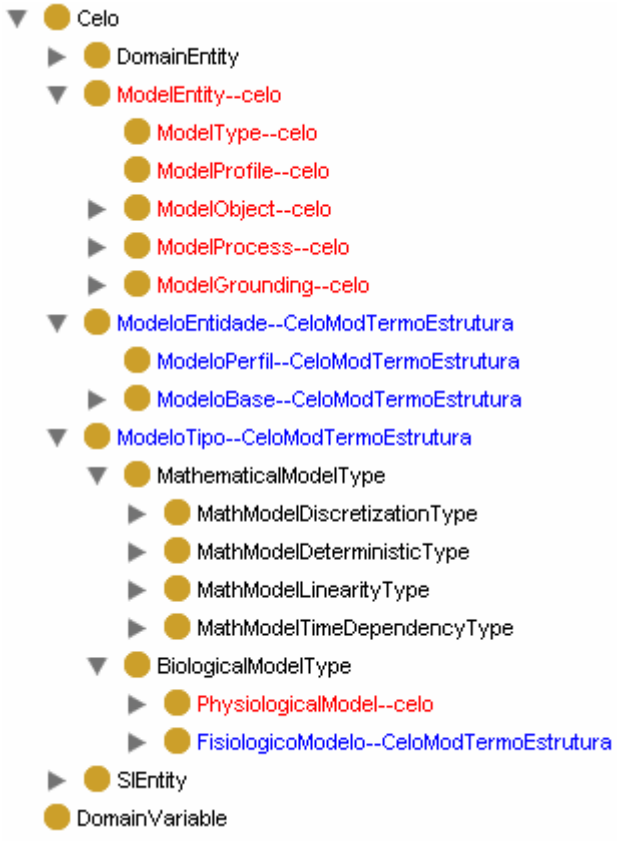


Figura 24: Parte do resultado do iPROMPT com a ontologia CeloModTermoEstrutura.owl

Analisando o resultado é possível perceber que uma pequena modificação nos nomes dos termos faz com que a ferramenta considere que as classes, propriedades ou indivíduos, sejam diferentes, ocorrendo assim adição de um mesmo termo duas vezes no resultado da combinação.

4.2.2. AnchorPROMPT

Na execução da ferramenta AnchorPROMPT, abrimos a ontologia *Celo.owl* no Protegé e em seguida indicamos no AnchorPROMPT a ontologia modificada. Inicialmente foi utilizada a ontologia *CeloModTermo.owl*.

Selecionado os projetos, a escolha pelas âncoras foi definida automaticamente pela ferramenta, esperando assim que a ferramenta nos retorne um melhor resultado.

Em seguida, a ferramenta exibe uma lista de sugestões para realizarmos o mapeamento. Essa lista inicialmente possuía 47 candidatos da ontologia *Celo.owl* e 46 candidatos da ontologia *CeloModTermo.owl* para serem alinhados, sendo que o total de termos das ontologias é de 85.

Foi realizada uma análise na lista e então retiradas algumas sugestões que não necessitava pelo alinhamento devido a diferenças entre elas. No final da análise, o alinhamento ficou com 54 candidatos nas duas ontologias. As sugestões rejeitadas estão listadas na Tabela 6.

Tabela 6: Sugestões Rejeitadas no AnchorPROMPT

Celo.owl	**CeloModTermo.owl**
IndependentVariable	DependentVariable
ModelParameterIn	ModelParameterOut
ModelParameterIn	ModelParameter
ModelParameter	ModelParameterOut
ModelParameter	ModelParameterIn
InternalComponent	ExternalComponent
Component	Compartment
ModelType	Modelo
ModelEntity	SIEntity
ModelParameterOut	ModelParameter
ModelParameterOut	ModelParameterIn
ChemicalEntity	BioChemicalEntity
NonLinearModel	LinearModelo
LinearModel	NonLinearModel
ExternalComponent	InternalComponent

Em seguida, a cada alinhamento realizado, a ferramenta exibe uma nova lista de sugestões, então é necessária a interação do usuário para analisar as sugestões e aceitá-las ou não, nesta etapa foi aceito todas as sugestões.

Por fim, foram alinhados 53 candidatos da ontologia *Celo.owl* e 56 candidatos da ontologia *CeloModTermo.owl* de forma semi-automática. Caso o usuário deseje alinhar mais termos, ele tem a possibilidade de selecionar os termos de cada ontologia e criar o alinhamento.

No nosso caso, isso não é importante, devido estarmos preocupado com o resultado que a ferramenta nos oferece. Visto que a interação do usuário para realizar o alinhamento uma a uma não é vantajosa, ignoramos esta etapa.

A Figura 25 exibe parte das duas ontologias alinhadas, o *m* indica onde ocorreu o alinhamento.

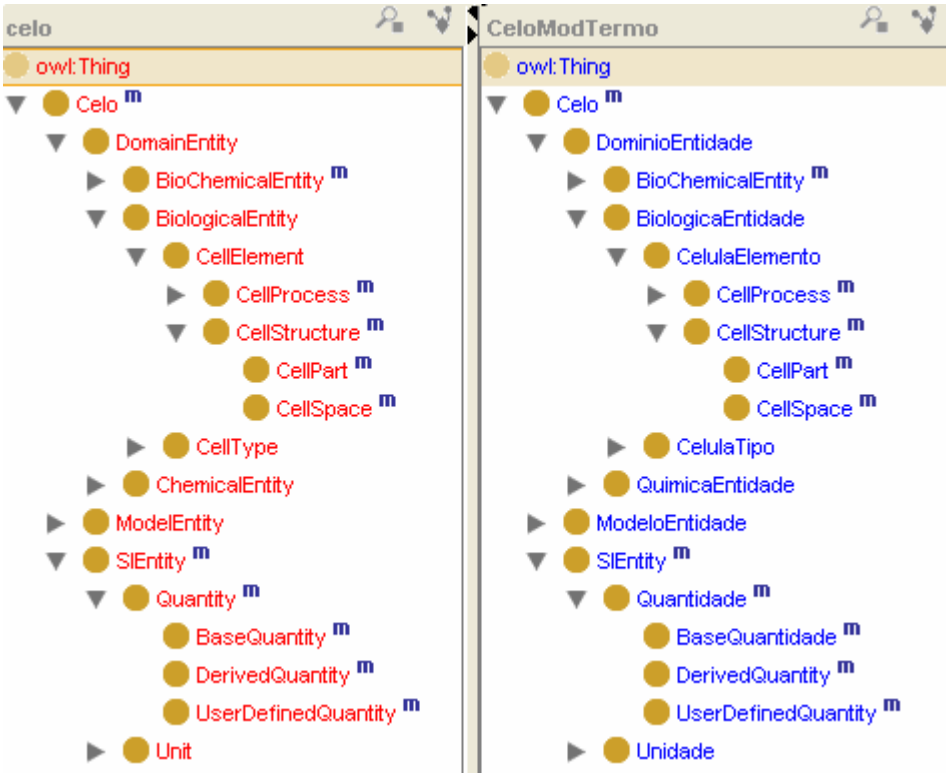


Figura 25: Parte do resultado do AnchorPROMPT com a ontologia CeloModTermo.owl

Como cada ontologia possui 85 candidatos para serem alinhados e somente 53 candidatos da ontologia *Celo.owl* e 56 candidatos da ontologia *CeloModTermo.owl* foram alinhados, o aproveitamento da ferramenta é de aproximadamente 64%, o que não representa um valor insignificante devido o alinhamento está correto.

Para o segundo teste, utilizamos a ontologia *CeloModEstrutura.owl* e seguimos os mesmos passos descritos acima. Ao iniciar o alinhamento a ferramenta sugeriu 81 candidatos para o alinhamento de cada ontologia, lembrando que o total de candidatos é de 85 para cada ontologia. Mesmo depois da análise e a retirada de algumas sugestões, o número de candidato para o alinhamento permaneceu para as duas ontologias. A Tabela 7 exibe as sugestões retiradas.

Tabela 7: Sugestões Rejeitadas no AnchorPROMPT

Celo.owl	**CeloModTermo.owl**
SIEntity	ModelEntity
BioChemicalEntity	ChemicalEntity
IndependentVariable	DependentVariable
ModelParameterIn	ModelParameter

ModelParameterIn	ModelParameterOut
ModelParameter	ModelParameterIn
ModelParameter	ModelParameterOut
InternalComponent	ExternalComponent
Component	Compartment
ModelEntity	SIEntity
ModelParameterOut	ModelParameterIn
ModelParameterOut	ModelParameter
ChemicalEntity	BioChemicalEntity
NonLinearModel	LinearModel
LinearModel	NonLinearModel
Compartment	Component
ExternalComponent	InternalComponent
DependentVariable	IndependentVariable

Em seguida, foi aceito todas as sugestões, conforme realizado também no teste anterior. A Figura 26 exibe parte do resultado do alinhamento.

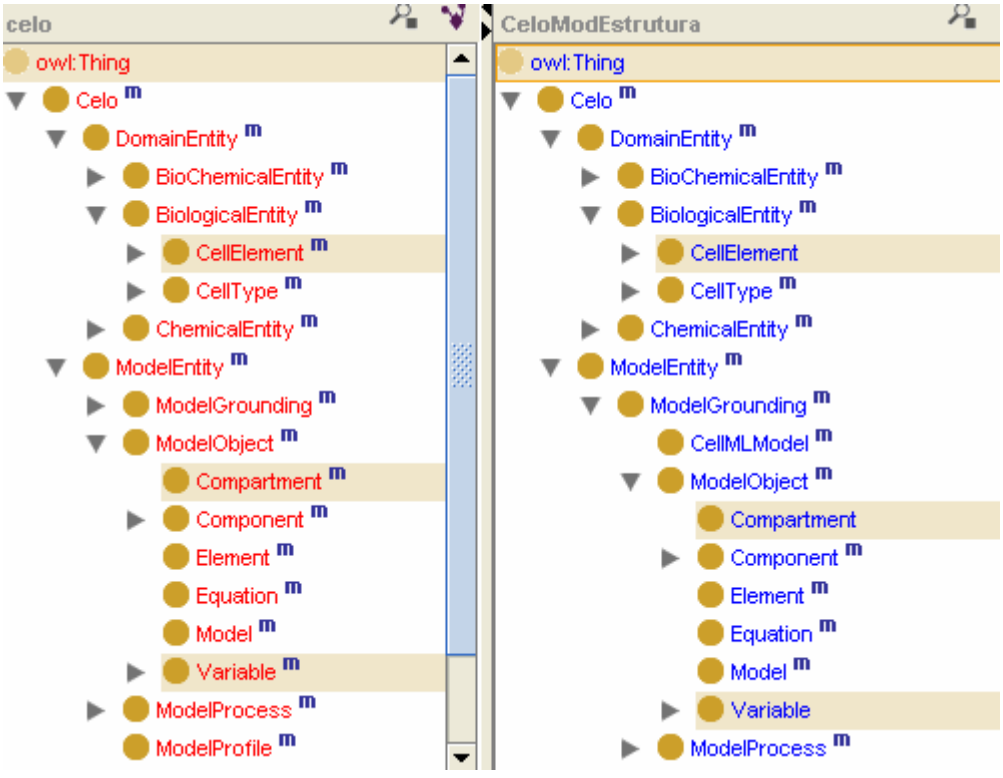


Figura 26: Parte do resultado do AnchorPROMPT com a ontologia CeloModEstrutura.owl

O resultado do alinhamento retornou 78 candidatos alinhados da ontologia *Celo.owl* e 82 candidatos da ontologia *CeloModEstrutura.owl.*

Na Figura 26 as classes destacadas são as que não foram alinhadas na ontologia modificada. A classe *CellElement* na ontologia modificada passou a ser domínio da propriedade *hasChemicalSimbol,* logo o alinhamento não foi possível nessa classe devido esta alteração.

Na classe *Compartment* a única alteração foi na sua superclasse *ModelObject* que passou a ser subclasse de *ModelGrounding* e mesmo o alinhamento sendo realizado nas suas classes irmãs, a ferramenta não conseguiu alinhá-la. O mesmo acontece com a classe *Variable*, sendo que esta classe ocorreu mais uma modificação, ela foi retirada do domínio da propriedade *hasDetail.*

Esse resultado retornou quase 100% de alinhamento mesmo a ontologia sendo bastante modificada em toda sua estrutura, como classes, propriedades e indivíduos.

Para o último teste nesta ferramenta, utilizamos a ontologia *CeloModTermoEstrutura.owl*, e novamente seguimos os passos descritos nas outras duas execuções acima.

Inicialmente a ferramenta retornou 47 candidatos para o mapeamento na ontologia original e 53 candidatos na ontologia modificada. Na análise das sugestões foram rejeitadas as seguintes sugestões listadas na Tabela 8.

Tabela 8: Sugestões Rejeitadas no AnchorPROMPT

Celo.owl	**CeloModTermo.owl**
IndependentVariable	DependentVariable
ModelParameterIn	ModelParameterOut
ModelParameterIn	ModelParameter
ModelParameter	ModelParameterOut
ModelParameter	ModelParameterIn
InternalComponent	ExternalComponent
Component	Compartment
ModelType	Modelo
ModelEntity	SIEntity
ModelParameterOut	ModelParameter
ModelParameterOut	ModelParameterIn
ChemicalEntity	BioChemicalEntity

NonLinearModel	LinearModelo
LinearModel	NonLinearModel
ExternalComponent	InternalComponent

Depois da análise das sugestões, o número de candidatos passou a ser 54 para cada ontologia. Feito o alinhamento aceitando todas as sugestões depois dessa etapa, obtemos o resultado da Figura 27.

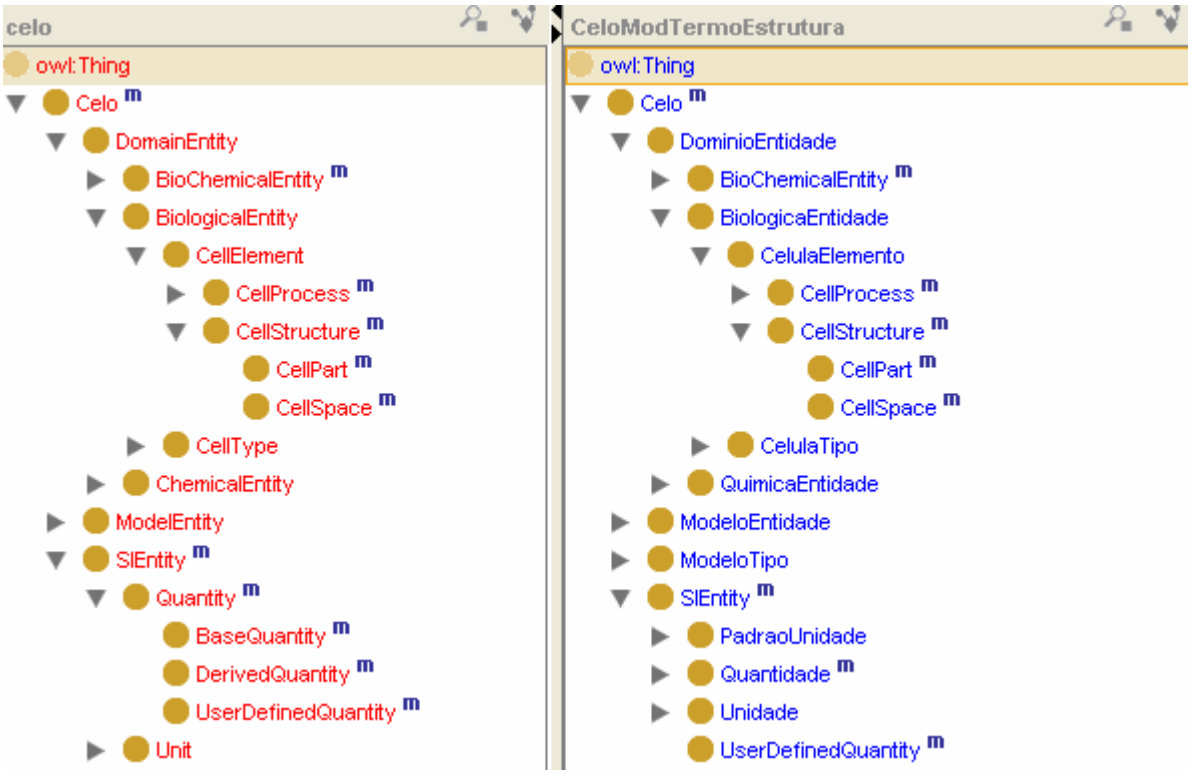


Figura 27: Parte do resultado do AnchorPROMPT com a ontologia CeloModTermoEstrutura.owl

No resultado acima, foram alinhados 53 candidatos da ontologia original e 56 da ontologia modificada, o que significa que a ferramenta retornou aproximadamente 64% do alinhamento. Resultado este que é igual do primeiro teste realizado, na modificação apenas nos nomes dos termos.

Podemos observar que a alteração na estrutura não afeta tanto quanto a alteração nos nomes dos termos, devido à ferramenta realizar métodos de semelhanças de estrutura e lexical.

4.2.3. PROMPTDiff

Para a execução do PROMPTDiff, abrimos a ontologia *Celo.owl* no Protegé e em seguida indicamos no PROMPTDiff a ontologia modificada.

Inicialmente foi utilizado a ontologia *CeloModTermo.owl*. Uma parte do resultado é ilustrada na Figura 28.

f1	f2	renamed
squaremeter	squaremeter	No
uF_per_mm2	uF_per_mm2	No
uA_per_mm2	uA_per_mm2	No
volt	volt	No
weber	weber	No
BioChemicalEntity	BioChemicalEntity	No
BiologicalModelType	BiologicalModelType	No

operation	map level	rename explanation
Map	Directly-changed	frame name and type are the same
Map	Directly-changed	frame name and type are the same
Map	Directly-changed	frame name and type are the same
Map	Directly-changed	frame name and type are the same
Map	Directly-changed	frame name and type are the same
Map	Isomorphic	frame name and type are the same
Map	Isomorphic	frame name and type are the same

(a)

f1	f2	renamed
ActionPotential	AcaoPotencial	Yes
BaseQuantity	BaseQuantidade	Yes
BiologicalEntity	BiologicaEntidade	Yes
CardiacMyocyte	CardiacosMiocitos	Yes

operation	map level	rename explanation
Map	Directly-changed	multiple unmatched siblings with same sl...
Map	Directly-changed	multiple unmatched siblings with similar r...
Map	Directly-changed	Allowed classes for the same slot
Map	Directly-changed	multiple unmatched siblings with same sl...

(b)

Figura 28: Parte do resultado do PROMPTDiff com a ontologia *CeloModTermo.owl*

A ferramenta detectou 41 classes modificadas. Na Figura 28 (a) podemos observar que nos indivíduos e nas classes que não foram alterados os nomes dos termos, a ferramenta conseguiu realizar o mapeamento de forma *changed* e *isomorphic*. O mapeamento *Directly-changed* significa que as classes, propriedades, indivíduos e etc são idênticos, mas não são

imagens uma da outra, isso ocorreu devido à alteração do nome da superclasse da classe que os contêm.

O mapeamento *isomorphic* indica que as classes, propriedades, indivíduos e etc são imagens um do outro, mas não são necessariamente idênticas, neste caso alteramos apenas os nomes das superclasses e subclasses dessas classes. Na Figura 29 é destacada nas duas ontologias a classe do mapeamento exibido na Figura 28 (a).

Figura 29: Classes do mapeamento

A classe *SIDerivedUnit* é a classe dos indivíduos *squaremeter* , *volt* e *weber*, onde foi alterado apenas o nome da superclasse de *StandardUnit* para *PadraoUnidade* da classe *SIDerivedUnit*, e a ferramenta considera que houve troca de classes devido a hierarquia da classe *SIDerivedUnit* ser diferente. A classe *UserDefinedUnit* é a classe dos indivíduos *uF_per_mm2* e *uA_per_mm2* que também só teve alteração no nome da superclasse de *Unit* para *Unidade* da classe *UserDefinedUnit*, e novamente a ferramenta considera que houve troca de classes devido a hierarquia de *UserDefinedUnit.* Nas classes *BioChemicalEntity* e *BiologicalModelType* é possível ver na figura a alteração dos nomes das superclasses e subclasses de cada uma, e mesmo a ferramenta considerando no mapeamento a troca do nome

da classe *DomainEntity* para *DominioEntidade*, ela considera para as classes *BioChemicalEntity e BiologicalModelType* que ocorre movimentação das classes.

Na Figura 28 (b) são exibidas algumas classes que foram alteradas os nomes dos termos e foi possível realizar o mapeamento correto na ferramenta. No entanto, para algumas propriedades, classes e indivíduos, a ferramenta considerou que foram apagadas e depois adicionadas as ontologias, sendo que apenas foram modificados os seus nomes.

No segundo teste, foi utilizado a ontologia *CeloModEstrutura.owl*, e foi realizado os mesmos passos do teste anterior. Como na ontologia alterada a única modificação foi estrutural, a ferramenta detectou apenas 8 classes modificadas e só realizou a operação de mapeamento de forma *changed* e *isomorphic*.

Na Figura 30 são exibidos os resultados.

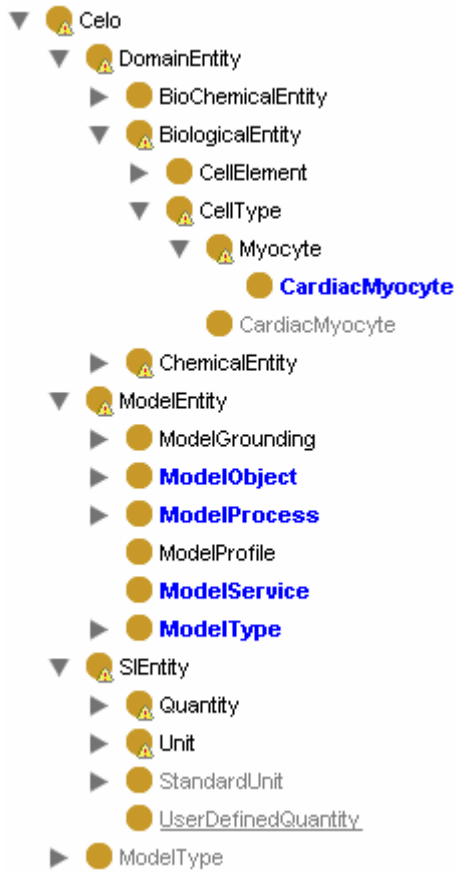


Figura 30: Parte do resultado do PROMPTDiff com a ontologia CeloModEstrutura.owl

Em todas as classes da figura acima que possui um ponto de exclamação, indica que houve alguma alteração nas suas subclasses. As classes em azul indicam que são classes da ontologia original e as classes de cor cinza indicam que são classes da ontologia modificada. Essas classes estão em destaques, pois são semelhantes, mas possui em sua estrutura alguma

diferença, no caso da classe *CardiacMyocyte* a diferença entre as duas classes está em sua superclasse, na classe original sua superclasse é a classe *Myocyte* e na classe modificada, sua superclasse é a classe *CellType*.

Por fim, foi realizado o teste com as duas modificações acima em uma única ontologia, *CeloModTermoEstrutura.owl*, comparada com a ontologia original. A ferramenta retornou o resultado da Figura 31.

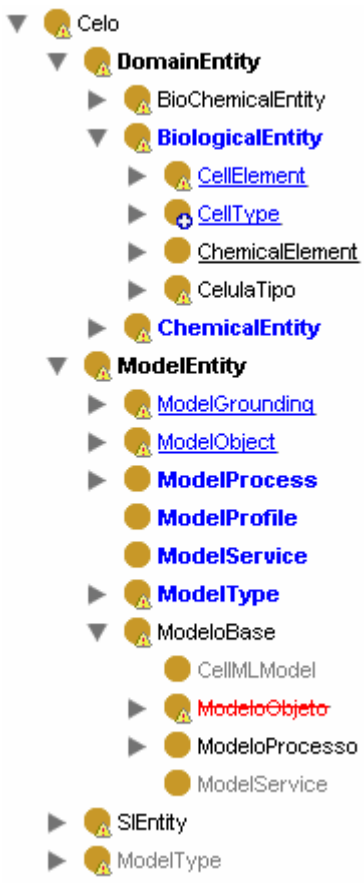


Figura 31: Parte do resultado do PROMPTDiff com a ontologia CeloModTermoEstrutura.owl

Ocorreu troca em 56 subclasses e, analisando o resultado, é possível afirmar que todas as alterações foram detectadas.

Ao que podemos concluir, a única inconsistência dessa ferramenta é a operação de remover e adicionar classes, por exemplo, a classe *ModeloObjeto* poderia ser mapeada com a classe *ModelObject* e encontrar como modificação a alteração de nome e de estrutura nelas, no entanto a ferramenta remove a classe *ModeloObjeto*, como ilustrado acima, e adiciona a classe *ModelObject* da ontologia *Celo.owl*.

Isso ocorre devido à ferramenta realizar inicialmente as operações de adição e de remoção nas classes, antes da operação de mapeamento.

4.2.4. GNoSIS

Para a execução da ferramenta GNoSIS, foram criados três arquivos XML para cada situação: modificação nos nomes dos termos da ontologia original, modificação da estrutura da ontologia original e modificação nos nomes dos termos e da estrutura na ontologia original, conforme descrito anteriormente. A avaliação dos testes foi em apenas 20 termos de cada ontologia.

A diferença nos três arquivos XML's, se encontra no valor do peso de cada função e a forma para realizar o cálculo da distância de edição. Esses arquivos XML's serão ilustrados para cada etapa do processo de testes.

O primeiro teste foi realizado com o XML da Figura 32, como a alteração na ontologia foi realizada apenas nos nomes dos termos, o peso da função de semelhança *ConceptNameSimilarity* foi igual a 30% do valor total devido esta função ser sintática, e o restante distribuído entre as demais funções que são de estrutura, como a função *DirectSuperClassSimilarity*, *DirectSubClassSimilarity*, *DirectDataTypePropertybyRangeSimilarity* e etc.

Para a função do cálculo da distância de edição foi adotado o *DamerauLevenshteinEditDistance*, por possuí um valor muitas vezes melhor que as outras funções quando os nomes são diferentes.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE similarityCalc (View Source for full doctype...)>
- <similarityCalc>
  - <container name="Funcoes principais">
    - <function weight="0.3">
        <class>br.ufjf.ontology.gnosis.similarity.structure.ConceptNameSimilarity</class>
        <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
        <penalty>0.0</penalty>
      </function>
    - <container name="Funcoes Secundarias" weight="0.7">
      - <function weight="0.35">
          <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSuperClassSimilarity</class>
          <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
          <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
          <penalty>0.0</penalty>
        </function>
      - <function weight="0.35">
          <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSubClassSimilarity</class>
          <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
          <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
          <penalty>0.0</penalty>
        </function>
      + <function weight="0.1">
      - <container name="Funcoes Propriedades" weight="0.2">
        - <function weight="0.2">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyRangeSimilarity</class>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.3">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyNameSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.2">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyRangeSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.3">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyNameSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        </container>
      </container>
    </container>
  - <ontologies>
      <ontology id="Celo.owl" />
      <ontology id="CeloModTermo.owl" />
    </ontologies>
</similarityCalc>
```

Figura 32: XML para a execução do GNoSIS com a ontologia CeloModTermo.owl

Para facilitar o entendimento de como será calculado o valor final, a Figura 33, exemplifica que para obter o valor final é preciso calcular os atômicos.

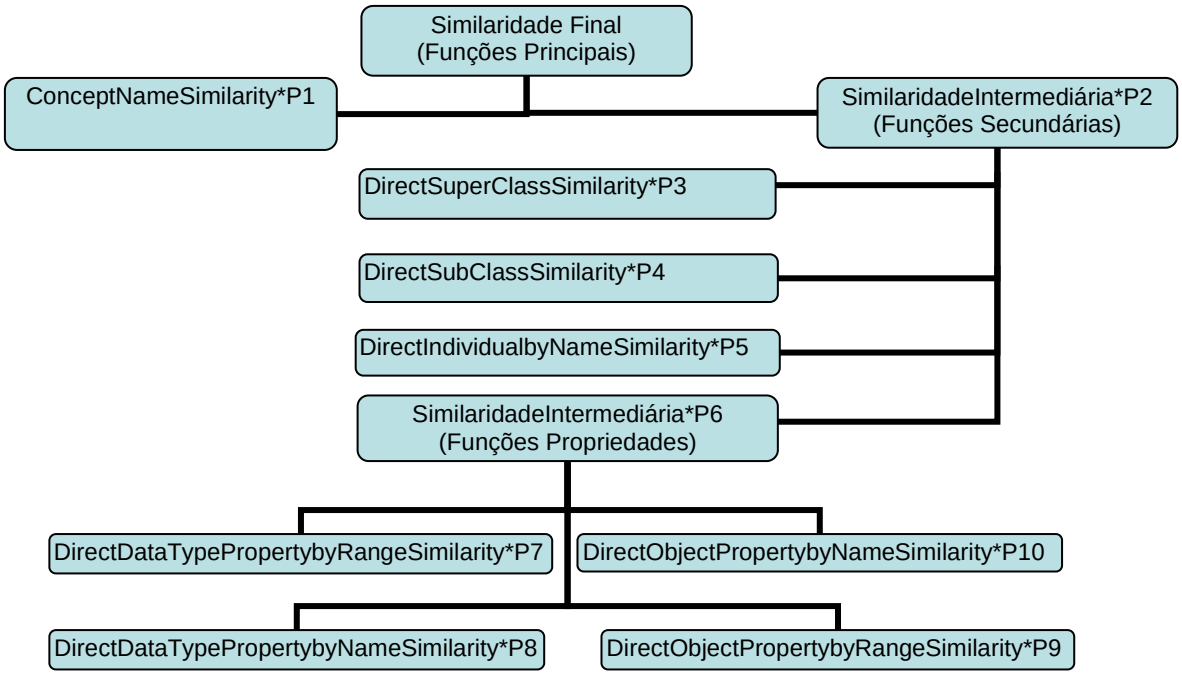


Figura 33: Árvore para o cálculo de similaridade

Para cada execução é necessário indicar na classe *Analyser* o caminho do arquivo XML. Em seguida é só executar a ferramenta para ela exibir uma tabela com as similaridades dos termos das ontologias.

Parte do primeiro resultado é exibida na Tabela 9.

Tabela 9 (a): Parte do resultado do GNoSIS

	Proteina	PadraoUnidade	ModeloTipo	MathematicalModelType	ModeloObjeto
Celo	0,19	0,19	0,28	0,23	0,27
CellElement	0,45	0,26	0,32	0,30	0,31
DomainVariable	0,30	0,31	0,29	0,28	0,27
BioChemicalEntity	0,40	0,30	0,39	0,35	0,35
ModelEntity	0,31	0,26	0,43	0,34	0,41
SIEntity	0,29	0,29	0,32	0,33	0,29
ModelService	0,56	0,19	0,45	0,35	0,42
BiologicalEntity	0,38	0,28	0,41	0,32	0,34
Quantity	0,26	0,19	0,24	0,21	0,23
MathematicalModelType	0,30	0,31	0,40	**0,68**	0,37
ModelType	0,37	0,28	**0,73**	0,47	0,52
StandardUnit	0,28	**0,54**	0,29	0,29	0,25
CellType	0,40	0,28	0,43	0,36	0,33

Unit	0,19	0,14	0,17	0,11	0,16
ChemicalCompound	0,45	0,31	0,35	0,36	0,28
Protein	**0,96**	0,33	0,33	0,30	0,32
DomainEntity	0,30	0,23	0,31	0,34	0,31
ChemicalObject	0,61	0,17	0,20	0,29	0,26
ModelObject	0,37	0,23	0,50	0,40	**0,59**
DefinedUnit	0,48	0,29	0,20	0,21	0,20

Tabela 9 (b): Parte do resultado do GNoSIS

	Chemical Compound	Biológica Entidade	Domain Variable	Químico Objeto	Célula Elemento
Celo	0,23	0,21	0,16	0,13	0,27
CellElement	0,40	0,39	0,24	0,24	**0,62**
DomainVariable	0,28	0,31	**0,69**	0,21	0,25
BioChemicalEntity	0,44	0,51	0,27	0,29	0,40
ModelEntity	0,23	0,31	0,28	0,16	0,30
SIEntity	0,23	0,34	0,28	0,16	0,28
ModelService	0,22	0,27	0,20	0,38	0,25
BiologicalEntity	0,35	**0,64**	0,27	0,28	0,45
Quantity	0,22	0,28	0,20	0,20	0,26
MathematicalModelType	0,34	0,30	0,28	0,21	0,28
ModelType	0,37	0,37	0,25	0,23	0,36
StandardUnit	0,28	0,33	0,33	0,20	0,30
CellType	0,41	0,32	0,26	0,24	0,53
Unit	0,15	0,17	0,11	0,11	0,15
ChemicalCompound	**0,85**	0,35	0,26	0,33	0,43
Protein	0,33	0,33	0,28	0,52	0,37
DomainEntity	0,25	0,33	0,39	0,19	0,28
ChemicalObject	0,40	0,27	0,22	**0,68**	0,31
ModelObject	0,29	0,33	0,23	0,30	0,35
DefinedUnit	0,21	0,22	0,23	0,46	0,23

Tabela 9 (c): Parte do resultado do GNoSIS

	SIEntity	BioChemicalEntity	Unidade	ModelService	CelulaTipo
Celo	0,20	0,23	0,16	0,17	0,30
CellElement	0,30	0,41	0,19	0,21	0,54
DomainVariable	0,28	0,28	0,13	0,24	0,30
BioChemicalEntity	0,34	**0,86**	0,23	0,28	0,40
ModelEntity	0,62	0,39	0,07	0,34	0,29
SIEntity	**0,88**	0,36	0,07	0,22	0,33
ModelService	0,21	0,27	0,23	**0,88**	0,29
BiologicalEntity	0,35	0,61	0,23	0,26	0,35
Quantity	0,33	0,31	0,42	0,23	0,26
MathematicalModelType	0,31	0,34	0,13	0,32	0,30
ModelType	0,30	0,38	0,24	0,45	0,41
StandardUnit	0,29	0,37	0,24	0,17	0,28
CellType	0,28	0,35	0,20	0,26	**0,69**
Unit	0,18	0,19	**0,65**	0,16	0,16
ChemicalCompound	0,31	0,45	0,21	0,19	0,41
Protein	0,29	0,35	0,16	0,49	0,36
DomainEntity	0,61	0,37	0,14	0,24	0,26
ChemicalObject	0,19	0,36	0,17	0,37	0,29

ModelObject	0,25	0,34	0,23	0,45	0,31
DefinedUnit	0,19	0,28	0,15	0,37	0,20

Tabela 9 (d): Parte do resultado do GNoSIS

	DefinedUnit	DominioEntidade	ModeloEntidade	Quantidade	Celo
Celo	0,13	0,30	0,27	0,15	**0,86**
CellElement	0,25	0,26	0,30	0,23	0,25
DomainVariable	0,27	0,36	0,30	0,23	0,19
BioChemicalEntity	0,25	0,30	0,28	0,30	0,19
ModelEntity	0,19	0,56	**0,65**	0,17	0,26
SIEntity	0,19	0,53	0,54	0,23	0,20
ModelService	0,34	0,24	0,33	0,30	0,14
BiologicalEntity	0,24	0,29	0,27	0,30	0,18
Quantity	0,12	0,26	0,20	**0,78**	0,13
MathematicalModelType	0,21	0,32	0,34	0,21	0,21
ModelType	0,17	0,29	0,38	0,30	0,23
StandardUnit	0,37	0,23	0,23	0,31	0,14
CellType	0,20	0,26	0,30	0,26	0,28
Unit	0,11	0,17	0,11	0,51	0,16
ChemicalCompound	0,21	0,30	0,28	0,26	0,23
Protein	0,47	0,29	0,30	0,26	0,20
DomainEntity	0,24	**0,77**	0,56	0,27	0,26
ChemicalObject	0,48	0,21	0,19	0,22	0,15
ModelObject	0,17	0,27	0,36	0,27	0,22
DefinedUnit	**0,86**	0,24	0,18	0,19	0,10

No resultado acima é exibida uma tabela com todos os valores de similaridade calculados entre os 20 termos descritos das ontologias analisadas. Os valores em negrito representam o maior valor de cada linha, sendo que em todas as tabelas as linhas são continuação de uma tabela para outra.

Podemos observar que o valor em negrito se encontra na linha e na coluna dos termos semelhantes, logo a ferramenta retornou o resultado correto do alinhamento dos termos, pois o maior valor, representa a maior semelhança.

A ferramenta poderia ter retornando um valor melhor, mais aproximado de um quando os termos são semelhantes, isso não ocorreu devido à análise de todas as subclasses, superclasses, propriedades e indivíduos, pois os nomes dos termos dos mesmos também foram modificados, trazendo assim um resultado com o valor mais baixo do que esperado, mas com um valor mais exato devido todas as modificações realizadas.

Para o segundo teste, foi alterado a função para o cálculo de edição para *EqualsEditDistance* e o valor do peso da função *ConceptNameSimilarity* para 50% do valor total, devido os nomes dos termos serem iguais. Outros valores do peso das demais funções também foram alterados conforme Figura 34.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE similarityCalc (View Source for full doctype...)>
<similarityCalc>
 - <container name="Funcoes principais">
   - <function weight="0.5">
       <class>br.ufjf.ontology.gnosis.similarity.structure.ConceptNameSimilarity</class>
       <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
       <penalty>0.0</penalty>
     </function>
   - <container name="Funcoes Secundarias" weight="0.5">
     - <function weight="0.3">
         <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSuperClassSimilarity</class>
         <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
         <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
         <penalty>0.0</penalty>
       </function>
     - <function weight="0.3">
         <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSubClassSimilarity</class>
         <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
         <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
         <penalty>0.0</penalty>
       </function>
     - <function weight="0.1">
         <class>br.ufjf.ontology.gnosis.similarity.structure.DirectIndividualbyNameSimilarity</class>
         <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
         <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
         <penalty>0.0</penalty>
       </function>
     - <container name="Funcoes Propriedades" weight="0.3">
       - <function weight="0.2">
           <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyRangeSimilarity</class>
           <penalty>0.0</penalty>
         </function>
       - <function weight="0.3">
           <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyNameSimilarity</class>
           <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
           <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
           <penalty>0.0</penalty>
         </function>
       - <function weight="0.2">
           <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyRangeSimilarity</class>
           <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
           <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
           <penalty>0.0</penalty>
         </function>
       - <function weight="0.3">
           <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyNameSimilarity</class>
           <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
           <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.EqualsEditDistance</strategy>
           <penalty>0.0</penalty>
         </function>
       </container>
     </container>
   </container>
 - <ontologies>
     <ontology id="Celo.owl" />
     <ontology id="CeloModEstrutura.owl" />
   </ontologies>
</similarityCalc>
```

Figura 34: XML para a execução do GNoSIS com a ontologia CeloModEstrutura.owl

Parte do segundo resultado é exibida na Tabela 10.

Tabela 10 (a): Parte do resultado do GNoSIS

	Variable	Standard Unit	Celo	ModelType	DomainEntity
Celo	0,12	0,12	**0,84**	0,12	0,12
CellElement	0,05	0,20	0,12	0,20	0,20
DomainVariable	0,05	0,20	0,12	0,20	0,20
BioChemicalEntity	0,05	0,20	0,12	0,20	0,20

ModelEntity	0,05	0,20	0,12	0,35	0,35
SIEntity	0,05	0,20	0,12	0,35	0,35
ModelService	0,05	0,12	0,05	0,12	0,12
BiologicalEntity	0,05	0,20	0,12	0,20	0,20
Quantity	0,08	0,28	0,05	0,12	0,12
ModelType	0,05	0,20	0,12	**0,85**	0,20
StandardUnit	0,05	**0,85**	0,12	0,20	0,20
CellType	0,05	0,20	0,12	0,20	0,20
Unit	0,12	0,20	0,13	0,05	0,05
ChemicalCompound	0,05	0,20	0,12	0,20	0,20
Variable	**0,93**	0,05	0,12	0,05	0,05
Protein	0,05	0,20	0,12	0,20	0,20
DomainEntity	0,05	0,20	0,12	0,35	**1,00**
ChemicalObject	0,00	0,15	0,08	0,15	0,15
ChemicalEntity	0,05	0,20	0,12	0,20	0,20
DefinedUnit	0,00	0,15	0,08	0,15	0,15

Tabela 10 (b): Parte do resultado do GNoSIS

	CellType	BioChemical Entity	Biological Entity	DefinedUnit	Quantity
Celo	0,12	0,12	0,12	0,08	0,00
CellElement	0,35	0,20	0,20	0,15	0,08
DomainVariable	0,20	0,20	0,20	0,15	0,08
BioChemicalEntity	0,20	**1,00**	0,35	0,15	0,08
ModelEntity	0,20	0,20	0,20	0,15	0,08
SIEntity	0,20	0,20	0,20	0,15	0,08
ModelService	0,12	0,12	0,12	0,23	0,08
BiologicalEntity	0,20	0,35	**1,00**	0,15	0,08
Quantity	0,12	0,12	0,12	0,08	**0,80**
ModelType	0,20	0,20	0,20	0,15	0,08
StandardUnit	0,20	0,20	0,20	0,30	0,08
CellType	**1,00**	0,20	0,20	0,15	0,08
Unit	0,05	0,05	0,05	0,00	0,15
ChemicalCompound	0,20	0,20	0,20	0,15	0,08
Variable	0,05	0,05	0,05	0,00	0,03
Protein	0,20	0,20	0,20	0,30	0,08
DomainEntity	0,20	0,20	0,20	0,15	0,08
ChemicalObject	0,15	0,15	0,15	0,30	0,08
ChemicalEntity	0,20	0,35	0,35	0,15	0,08
DefinedUnit	0,15	0,15	0,15	**1,00**	0,08

Tabela 10 (c): Parte do resultado do GNoSIS

	SIEntity	Chemical Entity	Chemical Compound	Chemical Object	Model Service
Celo	0,12	0,12	0,12	0,08	0,05
CellElement	0,20	0,20	0,20	0,15	0,12
DomainVariable	0,20	0,20	0,20	0,15	0,12
BioChemicalEntity	0,20	0,35	0,20	0,15	0,12
ModelEntity	0,35	0,20	0,20	0,15	0,12
SIEntity	**0,85**	0,20	0,20	0,15	0,12
ModelService	0,12	0,12	0,12	0,23	**0,77**
BiologicalEntity	0,20	0,35	0,20	0,15	0,12
Quantity	0,12	0,12	0,12	0,08	0,12
ModelType	0,20	0,20	0,20	0,15	0,12
StandardUnit	0,20	0,20	0,20	0,15	0,12

CellType	0,20	0,20	0,20	0,15	0,12
Unit	0,05	0,05	0,05	0,00	0,05
ChemicalCompound	0,20	0,20	**1,00**	0,30	0,12
Variable	0,05	0,05	0,05	0,00	0,05
Protein	0,20	0,20	0,20	0,30	0,28
DomainEntity	0,35	0,20	0,20	0,15	0,12
ChemicalObject	0,15	0,15	0,30	**1,00**	0,23
ChemicalEntity	0,20	**0,85**	0,20	0,15	0,12
DefinedUnit	0,15	0,15	0,15	0,30	0,23

Tabela 10 (d): Parte do resultado do GNoSIS

	CellElement	Protein	Unit	DomainVariable	ModelEntity
Celo	0,12	0,12	0,07	0,12	0,12
CellElement	**0,90**	0,20	0,00	0,20	0,20
DomainVariable	0,20	0,20	0,00	**0,85**	0,20
BioChemicalEntity	0,20	0,20	0,00	0,20	0,20
ModelEntity	0,20	0,20	0,00	0,20	**0,85**
SIEntity	0,20	0,20	0,00	0,20	0,35
ModelService	0,12	0,28	0,00	0,12	0,12
BiologicalEntity	0,20	0,20	0,00	0,20	0,20
Quantity	0,12	0,12	0,15	0,12	0,12
ModelType	0,20	0,20	0,00	0,20	0,20
StandardUnit	0,20	0,20	0,00	0,20	0,20
CellType	0,35	0,20	0,00	0,20	0,20
Unit	0,05	0,05	**0,80**	0,05	0,05
ChemicalCompound	0,20	0,20	0,00	0,20	0,20
Variable	0,05	0,05	0,04	0,05	0,05
Protein	0,20	**1,00**	0,00	0,20	0,20
DomainEntity	0,20	0,20	0,00	0,20	0,35
ChemicalObject	0,15	0,30	0,00	0,15	0,15
ChemicalEntity	0,20	0,20	0,00	0,20	0,20
DefinedUnit	0,15	0,30	0,00	0,15	0,15

O resultado acima exibe como o resultado anterior, uma Tabela com 20 conceitos de cada ontologia e os respectivos valores do cálculo de similaridade entre esses conceitos.

A ferramenta retornou um resultado ótimo para essa situação, a diferença nos valores dos termos similares, valores em negrito, e dos demais é muito grande, o que reafirma a confiabilidade do resultado.

O fato de alterar a função da distância de edição para *EqualsEditDistance* fez com que os valores não oscilassem muito, isso ocorreu devido o algoritmo simples da função que retorna 1 para semelhante e 0 para diferente. Apenas em uma função de semelhança, a *DirectDataTypePropertybyNameSimilarity*, não foi alterado a função da distância de edição, esse fato fez com que os valores dos termos que não são semelhantes fossem parecidos.

Podemos analisar os resultados juntamente com as modificações e perceber que a ferramenta retornou um resultado preciso, reconhecendo as modificações nos termos

semelhantes, esses resultados são, por exemplo, valor da similaridade dos termos *Variable*, *Unit* e nos demais que possuem valores aproximados de um.

Para o último teste, adotamos a mesma função de cálculo de edição do primeiro teste e alteramos alguns valores do peso das funções, conforme Figura 35.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE similarityCalc (View Source for full doctype...)>
- <similarityCalc>
  - <container name="Funcoes principais">
    - <function weight="0.3">
        <class>br.ufjf.ontology.gnosis.similarity.structure.ConceptNameSimilarity</class>
        <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
        <penalty>0.0</penalty>
      </function>
    - <container name="Funcoes Secundarias" weight="0.7">
      - <function weight="0.3">
          <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSuperClassSimilarity</class>
          <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
          <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
          <penalty>0.0</penalty>
        </function>
      - <function weight="0.3">
          <class>br.ufjf.ontology.gnosis.similarity.structure.DirectSubClassSimilarity</class>
          <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
          <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
          <penalty>0.0</penalty>
        </function>
      - <function weight="0.2">
          <class>br.ufjf.ontology.gnosis.similarity.structure.DirectIndividualbyNameSimilarity</class>
          <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
          <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
          <penalty>0.0</penalty>
        </function>
      - <container name="Funcoes Propriedades" weight="0.2">
        - <function weight="0.2">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyRangeSimilarity</class>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.3">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectDataTypePropertybyNameSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.2">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyRangeSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        - <function weight="0.3">
            <class>br.ufjf.ontology.gnosis.similarity.structure.DirectObjectPropertybyNameSimilarity</class>
            <combination>br.ufjf.ontology.gnosis.similarity.combination.DeepCombination</combination>
            <strategy>br.ufjf.ontology.gnosis.similarity.editdistance.DamerauLevenshteinEditDistance</strategy>
            <penalty>0.0</penalty>
          </function>
        </container>
      </container>
    </container>
  - <ontologies>
      <ontology id="Celo.owl" />
      <ontology id="CeloModTermoEstrutura.owl" />
    </ontologies>
</similarityCalc>
```

Figura 35: XML para a execução do GNoSIS com a ontologia CeloModTermoEstrutura.owl

Parte do terceiro resultado é exibida na tabela 11.

Tabela 11 (a): Parte do resultado do GNoSIS

	Variavel	Unidade	Proteina	BiologicaEntidade	PadraoUnidade
Celo	0,33	0,06	0,26	0,27	0,23
CellElement	0,18	0,13	0,50	0,44	0,37
DomainVariable	0,30	0,06	0,37	0,38	0,32
BioChemicalEntity	0,21	0,18	0,45	0,55	0,44
ModelEntity	0,19	0,00	0,37	0,38	0,33
SIEntity	0,19	0,05	0,35	0,40	0,35
ModelService	0,27	0,14	0,58	0,33	0,37
BiologicalEntity	0,21	0,18	0,44	**0,68**	0,43
Quantity	0,20	0,24	0,32	0,34	0,47
Variable	**0,72**	0,12	0,18	0,21	0,23
ModelType	0,25	0,17	0,42	0,42	0,46
StandardUnit	0,21	0,26	0,34	0,39	**0,56**
CellType	0,17	0,13	0,44	0,38	0,40
Unit	0,29	**0,45**	0,25	0,23	0,42
ChemicalCompound	0,19	0,16	0,49	0,41	0,43
Protein	0,21	0,07	**0,96**	0,39	0,45
DomainEntity	0,24	0,02	0,37	0,40	0,30
ChemicalObject	0,08	0,15	0,54	0,26	0,23
ModelObject	0,27	0,14	0,42	0,39	0,42
DefinedUnit	0,04	0,13	0,44	0,22	0,26

Tabela 11 (b): Parte do resultado do GNoSIS

	ModeloObjeto	Quantidade	Domain Variable	DefinedUnit	Celula Elemento
Celo	0,35	0,00	0,23	0,12	0,33
CellElement	0,36	0,16	0,31	0,25	**0,66**
DomainVariable	0,32	0,16	**0,74**	0,26	0,32
BioChemicalEntity	0,35	0,25	0,33	0,25	0,46
ModelEntity	0,49	0,10	0,35	0,19	0,37
SIEntity	0,37	0,23	0,35	0,19	0,35
ModelService	0,43	0,21	0,27	0,31	0,31
BiologicalEntity	0,34	0,25	0,34	0,23	0,49
Quantity	0,23	**0,51**	0,27	0,12	0,32
Variable	0,29	0,11	0,33	0,05	0,22
ModelType	0,53	0,25	0,32	0,17	0,42
StandardUnit	0,28	0,31	0,39	0,36	0,36
CellType	0,38	0,19	0,33	0,19	0,57
Unit	0,16	0,29	0,18	0,11	0,21
ChemicalCompound	0,33	0,22	0,33	0,21	0,48
Protein	0,37	0,17	0,35	0,43	0,42
DomainEntity	0,39	0,12	0,46	0,24	0,35
ChemicalObject	0,24	0,20	0,21	0,44	0,30
ModelObject	**0,60**	0,18	0,30	0,17	0,41
DefinedUnit	0,16	0,18	0,23	**0,88**	0,23

Tabela 11 (c): Parte do resultado do GNoSIS

	ModeloEntidade	ModelService	Celula Tipo	Quimico Objeto	ModeloTipo
Celo	0,33	0,25	0,37	0,13	0,35
CellElement	0,40	0,26	0,57	0,23	0,37
DomainVariable	0,37	0,30	0,36	0,20	0,35

BioChemicalEntity	0,37	0,28	0,45	0,28	0,37
ModelEntity	**0,68**	0,42	0,36	0,16	0,65
SIEntity	0,58	0,30	0,38	0,16	0,54
ModelService	0,40	**0,82**	0,35	0,34	0,40
BiologicalEntity	0,36	0,26	0,41	0,27	0,39
Quantity	0,27	0,25	0,32	0,20	0,24
Variable	0,22	0,28	0,19	0,07	0,18
ModelType	0,49	0,46	0,46	0,22	**0,66**
StandardUnit	0,33	0,21	0,36	0,19	0,33
CellType	0,43	0,31	**0,71**	0,23	0,47
Unit	0,18	0,18	0,22	0,10	0,17
ChemicalCompound	0,41	0,24	0,47	0,31	0,43
Protein	0,36	0,51	0,41	0,47	0,38
DomainEntity	0,60	0,32	0,33	0,19	0,54
ChemicalObject	0,19	0,32	0,28	**0,74**	0,21
ModelObject	0,43	0,46	0,37	0,29	0,45
DefinedUnit	0,18	0,31	0,19	0,42	0,17

Tabela 11 (d): Parte do resultado do GNoSIS

	Celo	BioChemicalEntity	ChemicalCompound	SIEntity	Dominio Entidade
Celo	**0,80**	0,29	0,30	0,26	0,36
CellElement	0,32	0,46	0,47	0,33	0,33
DomainVariable	0,26	0,34	0,34	0,35	0,43
BioChemicalEntity	0,26	**0,88**	0,50	0,39	0,37
ModelEntity	0,32	0,45	0,30	0,65	0,59
SIEntity	0,26	0,43	0,32	**0,79**	0,57
ModelService	0,21	0,33	0,28	0,27	0,31
BiologicalEntity	0,25	0,66	0,46	0,39	0,36
Quantity	0,14	0,37	0,28	0,40	0,33
Variable	0,27	0,19	0,18	0,18	0,26
ModelType	0,30	0,43	0,42	0,32	0,36
StandardUnit	0,21	0,42	0,35	0,36	0,30
CellType	0,35	0,41	0,45	0,31	0,33
Unit	0,20	0,25	0,21	0,25	0,24
ChemicalCompound	0,30	0,50	**0,73**	0,33	0,36
Protein	0,27	0,41	0,39	0,35	0,36
DomainEntity	0,26	0,44	0,32	0,64	**0,80**
ChemicalObject	0,15	0,35	0,39	0,19	0,21
ModelObject	0,28	0,40	0,35	0,32	0,34
DefinedUnit	0,10	0,27	0,21	0,19	0,24

No resultado da tabela acima, podemos observar que em mais uma situação a ferramenta retornou um resultado com alta confiabilidade, pois conseguiu encontrar as termos semelhantes, apesar das suas modificações. O valor em negrito, que representam o maior valor de semelhança do termo da linha com os dos termos da coluna, mostra que a semelhança foi encontrada.

Para o conceito *StandardUnit* na ontologia modificada, alteramos o seu nome para *PadraoUnidade* e alteramos sua estrutura, trocando sua superclasse, mesmo com essas modificações a ferramenta foi capaz de alinhar os conceitos.

4.3. Avaliação dos resultados

No conjunto de ferramentas da ferramenta Protegé, a ferramenta iPROMPT possui um algoritmo bastante simples para reconhecer a similaridade, pois, uma simples modificação faz com que a ferramenta reconheça que os termos são diferentes, fazendo com que a combinação das ontologias possua termos duplicados.

Já na ferramenta AnchorPROMPT, podemos observar que a alteração na estrutura não afeta tanto quanto a alteração nos nomes dos termos, devido a ferramenta realizar métodos de semelhanças de estrutura. Porém podemos perceber que o algoritmo do método de semelhança lexical é simples, pois quando a alteração se encontra nos nomes dos termos a ferramenta retorna um resultado inferior, mas com consistência.

A ferramenta PROMPTDiff reconheceu quase 100% das modificações, possuindo um ótimo resultado para a realização do mapeamento, tanto com a modificação dos nomes quanto da modificação da estrutura. a inconsistência da ferramenta se encontra na operação de remoção e adição caso a ferramenta não consiga encontrar semelhanças entre os termos.

O conjunto de ferramenta do Protegé possui uma baixa qualidade quando se trata de modificação de nomes. Por menor que seja a modificação, podemos observar que as ferramentas têm uma certa dificuldade de detectar as semelhanças. Quando se trata de estrutura o AnchorPROMPT e o PROMPTDiff retornam um ótimo resultado, e aparentemente é bastante confiável.

A ferramenta GNoSIS demonstrou um resultado confiável em todas as três situações, pois em todas elas foi possível alinhar todos os conceitos de forma correta e o valor do similaridade retornado é consistente com as modificações realizadas em todas as situações. Esse valor pode variar conforme o arquivo XML é formatado, mas a variação é pequena e não deixa de ser consistentes com os termos alinhados.

5. Considerações Finais

Atualmente, muito se tem falado em ontologias e os benefícios que as mesmas trazem na organização e compartilhamento de informações, no entanto a falta de padronização entre essas estruturas dificulta o reuso de informações, pois, de forma geral, as ontologias têm como objetivo promover um entendimento comum e compartilhado sobre um domínio, que pode ser comunicado entre pessoas e sistemas de aplicação.

Mas para que possam trabalhar em conjunto, trocando automaticamente as informações representadas em ontologias, são necessários mecanismos que garantam a sua interoperabilidade. Tais mecanismos são: a combinação, o alinhamento e o mapeamento.

Para o estudo da execução desses mecanismos, utilizamos as ferramentas PROMPT e GNoSIS. Buscamos conhecer a viabilidade do uso dessas ferramentas e também mostrar como as ferramentas se comportam para cada tipo de situação e com isso é apresentado um estudo de casos para as mesmas, avaliando o comportamento e analisando os resultados.

Para a ferramenta PROMPT, os estudos mostraram que o resultado obtido algumas vezes não é consistente.

Já a ferramenta iPROMPT, o algoritmo para reconhecer a similaridade entre os termos é bastante simples, o que deixa a desejar no resultado devido a duplicação de termos na ontologia final.

A ferramenta AnchorPROMPT e PROMPTDiff, os resultados foram razoáveis devido a possibilidade da verificação da estrutura da ontologia.

Destarte, a ferramenta GNoSIS retornou um resultado ótimo para todas as situações, pois foi possível alinhar todos os termos, mesmo os modificados, e o valor da similaridade retornada demonstra as modificações existentes. Como limitações da ferramenta, podemos citar o baixo desempenho de execução, o que fez com que se tornasse inviável a comparação da ontologia Celo.owl completa, mesmo com um número de classes não tão expressivos. Para utilizarmos o XML formatado exibido no capítulo 4, fizemos uma análise apenas de vinte conceitos de cada ontologia, dessa forma a ferramenta retornou os resultados exibidos no capítulo 4, mesmo mostrando um desempenho de processamento baixo.

Como trabalho futuro, propõem-se a realização da otimização do desempenho da ferramenta GnoSIS e testes mais aprofundados para avaliação de algoritmos de compatibilização de ontologias. Acrescentar funções semânticas para testar restrições OWL nas ontologias.

Referências

Agent Transaction Language for Advertising Services (ATLAS). Disponível em: <http://www.daml.ri.cmu.edu>. Acesso em: Junho de 2011.
Universidade Carnegie Mellon. Disponível em: <http://www.cmu.edu/>. Acesso em: Junho de 2011.

BERGMANN, U**. Evolução de Cenários Através de um Mecanismo de Rastreamento Baseado em Transformações**, Tese de Doutorado, PUC-Rio, 2002.

BORST, W. N.. **Construction of engineering ontologies.** 1997. Tese (Doutorado). Disponível em: <http://www.ub.utwente.nl/webdocs /inf/1/t0000004.pdf>. Acesso em: 21 junho 2011.

CAPOBIANGO, Isabella. **Cálculo de similaridade entre conceitos de ontologias através da composição de funções de similaridade.** Projeto para Treinamento Profissional. Universidade Federal de Juiz de Fora. 2010.

CMU RI Publications. Disponível em <http://www.daml.ri.cmu.edu/ont/homework/cmu-ri-publications-ont.daml>. Acesso em: Junho de 2011.

EUZENAT, Jerome; SHVAIKO, Pavel. **Ontology Matching. Springer-Verlag,** Heidelberg, Germany, 2007.

FELICÍSSIMO, C. H**. Interoperabilidade Semântica na Web: Uma Estratégia para o Alinhamento Taxonômico de Ontologias.** Dissertação de Mestrado. Rio de Janeiro: PUC, Departamento de Informática, 2004.

FELLBAUM, C. ***WordNet: An electronic Lexical Database***. Cambridge, MA, MIT Press, 1998. Versão para *download* do WordNet disponível em: <http://www.cogsci.princeton.edu/~wn/>. Acesso em Junho de 2011.

General University Ontology. Disponível em: <http://www.mondeca.com/owl/moses/univ.owl>. Acesso em: Junho de 2011.

GRUBER, T.R.. **A Translation Approach to Portable Ontologies Specifications.** Knowledge Acquisition, v. 4, pp 199-220, 1993.

HAENDCHEN, F. A.; STAA, A. v.; LUCENA, C. P. A **Component-Based Model for Building Reliable Multi-Agent Systems**. In: SEW - NASA/IEEE SOFTWARE ENGINEERING WORKSHOP, 28. 2003, Los Alamitos. Anais do SEW - NASA/IEEE Software Engineering Workshop. Los Alamitos, 2003.

HINZ, Verlani; PALAZZO, Luiz Antônio. **Colaboração em Sistemas Multiagentes Modelados por Ontologias.** Workshop – Escola de Sistemas de Agentes para Ambientes Colaborativos, 2007, p 1-13

MCGUINNESS, D.L., Fikes, R., Rice, J. e Wilder, S. (2000) **An Environment for Merging and Testing Large Ontologies,** In: *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado.

MONDECA S.A. **A Semantic Knowledge company**. Disponível em: <http://www.mondeca.com>. Acesso em: Junho de 2011.

NOY, N. F.; MUSEN, M. A.. **Creating semantic web contents with Protégé.** IEEE Intelligent Systems, 2002.

NOY, N. F.; MUSEN, M. A..**The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping.** International Journal of Human-Computer Studies, 2003.

PROTEGE. ***Protégé Project***. Disponível em: <http://protege.stanford.edu/>. Acesso em Junho de 2011.

SMITH, M. K.; WELTY, C.; MCGUINNESS, D. L. **OWL Web Ontology Language Guide**. Recomendação da W3C a partir de 10 de Fevereiro de 2004. Disponível em: <http://www.w3.org/TR/owl-guide/>. Acesso em Abril de 2011.

WORKSHOP ON KNOWLEDGE ACQUISITION, MODELING AND MANAGEMENT, 12, 1999, Banff, Canadá. NOY, N.F.; MUSEN, M.A. **SMART: Automated Support for Ontology Merging and Alignment.** Banff, Canada. v. 4, p. 1-20.