3
A Estratégia

Apesar dos esforços descritos no capítulo anterior, alinhar os termos de diferentes ontologias continua um problema em aberto e que precisa ser resolvido para viabilizar uma série de promessas da Web Semântica. Por exemplo, permanece a necessidade de como garantir a possibilidade de comunicação automática entre os agentes de software de aplicações semânticas permitindo a cooperação, i.e., compartilhamento e reutilização, das informações disponibilizadas nestas aplicações semânticas.

Neste trabalho, uma estratégia para o alinhamento taxonômico de ontologias é proposta. Como descrito em Doan (2003), o componente central em uma ontologia é sua taxonomia. Desta maneira, em um primeiro momento, apenas os conceitos com relacionamentos de especialização entre eles, i.e., relacionamentos do tipo "é-um", de duas ontologias[6] de entrada são investigados. Nesta estratégia proposta, ilustrada na Figura 8, o alinhamento é obtido em três etapas, executadas seqüencialmente.

A primeira etapa da estratégia, explicada em detalhes no tópico 3.2. deste trabalho, faz uso de comparação lexical entre os conceitos das ontologias de entrada e mecanismo de poda estrutural dos conceitos associados como condição de parada. Seus resultados são as ontologias entradas enriquecidas com os alinhamentos conseguidos nesta etapa da estratégia. Estas ontologias são transformadas em arquivos do tipo *XML*, onde apenas suas hierarquias são representadas.

A segunda etapa da estratégia, explicada em detalhes no tópico 3.3. deste trabalho, compara estruturalmente as hierarquias das ontologias resultantes da primeira etapa da estratégia, identificando as similaridades entre suas sub-árvores comuns. Os conceitos destas sub-árvores são classificados como *similares*.

[6] A estratégia proposta tem como entradas pares de ontologias. Caso seja necessário o alinhamento de mais de duas ontologias, este pode ser realizado em passos seqüenciais, sempre alinhando as ontologias duas a duas.

A terceira etapa da estratégia, explicada em detalhes no tópico 3.4. deste trabalho, refina os resultados da etapa anterior classificando aqueles conceitos identificados como *similares* em *bem similares* ou *pouco similares*, de acordo com um percentual de similaridade pré-fixado. Os resultados desta etapa são os conceitos classificados como *bem similares*.

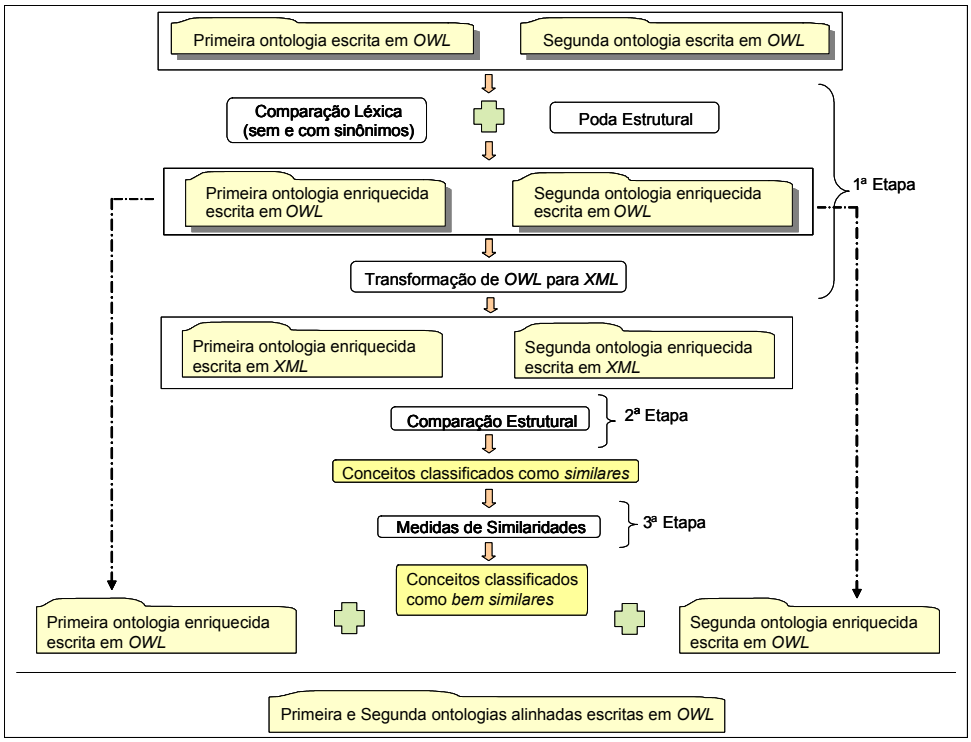


Figura 8 – Estratégia para o alinhamento taxonômico de ontologias

No final da execução das três etapas da estratégia, as informações de equivalência de conceitos *bem similares* são adicionadas nas ontologias resultantes da primeira etapa. Após esta adição, as ontologias alinhadas são unidas em uma ontologia única. Esta ontologia única é o resultado final da estratégia.

O fato do resultado final da estratégia ser uma ontologia única foi uma decisão de implementação. As ontologias alinhadas pela estratégia continuam sendo reconhecidas pela identificação de seus *namespaces* e existe a ligação entre os conceitos equivalentes na ontologia única, permitindo a reutilização e o compartilhamento de suas informações comuns, características do mecanismo de alinhamento de ontologias.

Para a estratégia elaborada ser confiável, deve-se minimizar a possibilidade de erros nos alinhamentos realizados. Para isto, cada uma de suas etapas possui condições que precisam ser satisfeitas para efetivarem os alinhamentos. Estas condições precisam ser bem escolhidas para que não inviabilizem qualquer

solução dada. Deve-se permitir, contudo, a possibilidade de refinamentos e melhorias incrementais dos resultados encontrados.

Na estratégia proposta, as entradas de cada uma de suas etapas são os produtos da etapa anterior com, possivelmente, as informações dos alinhamentos realizados adicionadas. Desta maneira, o resultado é refinado a cada nova etapa. Por exemplo, os conceitos não alinhados na comparação lexical da primeira etapa da estratégia podem ser alinhados com a identificação de sub-árvores comuns na comparação estrutural da segunda etapa e uso de medidas de similaridade da terceira etapa.

Todos os conceitos das ontologias a serem alinhadas são comparados, exceto os possíveis conceitos existentes originais de ontologias importadas. Estes conceitos não são utilizados em nenhuma das etapas da estratégia porque foi verificado que não trazem melhorias significativas que ajudem na decisão do alinhamento. Além disto, tornam qualquer comparação bem mais lenta pela quantidade de informação adicionada que precisa ser analisada.

Quando uma ontologia importa outras ontologias, todos os termos importados são adicionados nela. No entanto, os termos importados são diferenciados dos termos da ontologia que os importa pelo seu *namespace*. O *namespace* de um termo indica a localização de sua ontologia original, i.e., onde este termo foi criado. A análise do *namespace* de um termo revela se este termo é importado ou não. Termos importados em uma ontologia possuem seu *namespace* diferente do *namespace* da ontologia onde se encontram importados.

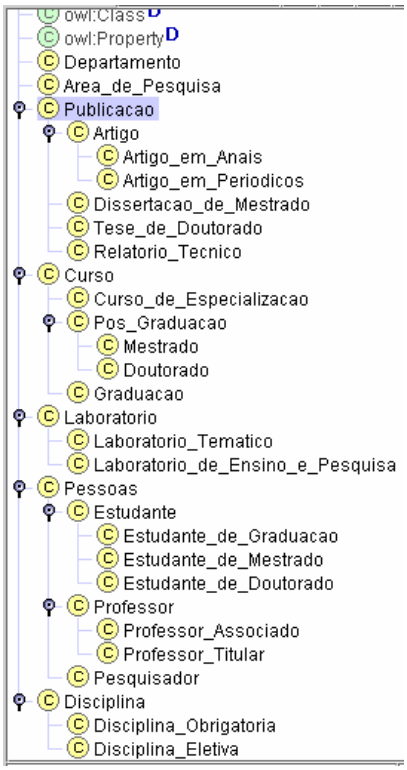
3.1. Um Exemplo Simplificado

Para exemplificar o alinhamento realizado pela estratégia descrita, duas ontologias simplificadas, ilustradas na Figura 9, foram criadas. A primeira ontologia, **O1**[7], encontra-se no lado esquerdo da figura e refere-se a uma simplificação de uma ontologia exemplo do Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro – PUC-RIO. A segunda

[7] A abreviação **O1** é utilizada ao longo do texto para facilitar a referência à primeira ontologia descrita.

ontologia, **O2**[8]**,** encontra-se no lado direito da mesma figura e refere-se a uma ontologia de publicação.

Ontologia do depto.
de Info. da PUC-RIO (**O1**)



Ontologia de Publicações (**O2**)

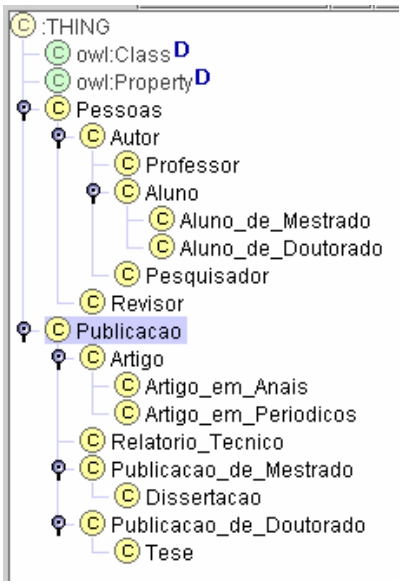


Figura 9 – Exemplo de ontologias a serem alinhadas

Deseja-se alinhar os conceitos equivalentes das duas ontologias criadas, ou seja, que após a execução da estratégia de alinhamento, os seguintes conceitos equivalentes sejam identificados: "Publicação", "Artigo", "Artigo_em_Anais" e "Artigo_em_Periodicos", de ambas as ontologias, "Dissertacao_de_Mestrado" de **O1** e "Dissertacao" de **O2**, "Tese_de_Doutorado" de **O1** e "Tese" de **O2**, entre outros.

Ao fornecer os conceitos equivalentes das ontologias alinhadas, é disponibilizado para os agentes de software de aplicações semânticas um indicativo do caminho para a recuperação das instâncias desses conceitos.

[8] A abreviação **O2** é utilizada ao longo do texto para facilitar a referência à segunda ontologia descrita.

3.2. Primeira Etapa: Comparação Lexical com Uso de Sinônimos e Mecanismo de Poda Estrutural como Condição de Parada

A etapa inicial da estratégia tem como suas entradas duas ontologias a serem alinhadas. Estas ontologias são transformadas[9] em modelos de forma que seus termos sejam manipulados como objetos.

O objetivo desta etapa é realizar a comparação lexical dos conceitos de ontologias de forma "mais inteligente", com o enriquecimento de informações, a procura de conceitos iguais lexicalmente e com o mesmo significado, i.e., conceitos equivalentes.

Quanto mais informações apuradas (e.g. sinônimos, gêneros, números, etc.) estiverem disponíveis para a comparação lexical, mais precisa é a identificação de equivalência entre conceitos. Adições de informações dos conceitos analisados como seus sinônimos, gêneros, concordâncias nominais e verbais, relacionamentos, entre outras, são valiosas.

Para enriquecer os conceitos de ontologias é preciso que exista algum mecanismo de extração de informações de fonte de dados, como dicionários e thesauri. Esta necessidade torna-se um desafio quando realizada de forma automática. Isto porque a maioria das fontes de dados não disponibilizam meios para acesso automático às suas informações e a identificação do significado de um conceito não é uma tarefa trivial, quando realizada sem a intervenção humana.

Um banco de sinônimos com acesso automático disponibilizado foi criado. Nele, novos sinônimos são adicionados, manualmente e sistematicamente, ao longo de seu uso, de forma a diluir seu custo de construção. A Figura 10 ilustra o banco de sinônimos utilizado com algumas informações cadastradas.

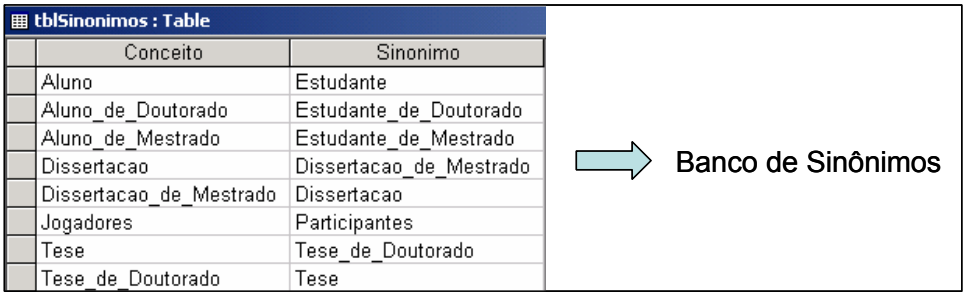


Figura 10 – Informações cadastradas no banco de sinônimos criado

[9] A transformação de ontologias em modelos orientados a objetos é descrita no tópico 3.5.2. deste trabalho.

Em um primeiro momento, há apenas o enriquecimento dos conceitos das ontologias comparadas com as informações de seus sinônimos. Além disso, a qualidade deste enriquecimento é diretamente proporcional à qualidade do banco de sinônimos criado. Quanto mais conceitos conhecidos com seus respectivos sinônimos existirem cadastrados nele, maiores são as chances do enriquecimento de informações entre conceitos e, conseqüentemente, dos alinhamentos.

Na comparação lexical, cada conceito da primeira ontologia é comparado, um a um, com cada conceito da segunda ontologia à procura da igualdade de seus nomes. Também, todos os sinônimos identificados dos conceitos de um modelo são comparados com todos os conceitos do outro modelo.

Encontrada a igualdade lexical entre dois conceitos ou entre um conceito e seu sinônimo nas ontologias comparadas, é preciso também comparar suas sub-árvores como indicativo semântico. Adota-se como critério de investigação a seguinte poda da árvore dos conceitos analisados:

- Generalização: poda até o conceito avô, i.e., conceitos com nomes iguais em seus dois níveis hierárquicos acima;
- Especialização: poda até as instâncias, i.e., instâncias com nomes iguais.

Apesar de uma instância ser identificada pelo par nome e *namespace* da ontologia a que pertence, a condição de poda de especialização só utiliza a igualdade de nomes. Isto porque, como as instâncias comparadas são de ontologias diferentes, então, estas instâncias possuem *namespaces* são diferentes.

Vale ressaltar que apenas as instâncias dos conceitos equivalentes, lexicalmente identificados, são investigadas para responderem à condição de poda de especialização. Estas instâncias devem estar cadastradas nas ontologias. Caso não estejam, não são investigadas.

Quando as instâncias não são armazenadas na própria ontologia, estas podem estar armazenadas em arquivos, em banco de dados, ou em outras estruturas de dados. A escolha do tipo de armazenamento de dados utilizado faz parte do projeto de uma aplicação e, como cada aplicação tem um propósito diferente, não dá para deduzir o tipo escolhido.

A busca pelas instâncias e seus alinhamentos aumentariam, razoavelmente, a complexidade de qualquer estratégia e, conseqüentemente, seu tempo de execução. Na estratégia elaborada, é realizado apenas o alinhamento das instâncias dos conceitos equivalentes identificados nas ontologias comparadas.

Conceitos equivalentes, identificados pela comparação lexical, que satisfazem tanto a condição de poda de generalização quanto a de especialização, são alinhados já na primeira etapa da estratégia. Os resultados desta etapa da estratégia de alinhamento são os modelos das duas ontologias entradas com as ligações estabelecidas entre seus conceitos equivalentes.

Para adicionar ligações entre os conceitos equivalentes identificados em ontologias comparadas, deve-se incluir as informações de equivalência em ambos os modelos dessas ontologias. No entanto, não se pode alterar os modelos dessas ontologias quando estes estão sendo analisados e percorridos. Isto porque, iterações são realizadas nesses modelos e novas inclusões nestes, fazem com que as iterações em execução percam suas numerações e seus limites corretos. A solução para este problema é trabalhar com uma estrutura de dados auxiliar para armazenar as equivalências identificadas dos conceitos e no final das iterações no modelo da ontologia analisada, adicionar as equivalências neste modelo.

3.2.1. Revendo o Exemplo

Para exemplificar a primeira etapa da estratégia, o exemplo descrito no tópico 3.1. é revisto.

Ao executar a primeira etapa da estratégia, todos os sinônimos dos conceitos das duas ontologias de entrada cadastrados no banco de sinônimos são identificados. A Figura 11 ilustra estas identificações. O *namespace* da ontologia **O1** é "file:firstOnto.owl" e o da **O2** é "file:secondOnto.owl". As informações depois da string "#" referem-se aos nomes dos conceitos e seus sinônimos identificados estão entre colchetes. Por exemplo, a informação que o conceito "Aluno" da segunda ontologia ("file:secondOnto.owl#Aluno") possui o termo "Estudante" como sinônimo é identificada.

```
Sinonimos de file:firstOnto.owl#Tese_de_Doutorado: [Tese]
Sinonimos de file:firstOnto.owl#Dissertacao_de_Mestrado: [Dissertacao]
Sinonimos de file:secondOnto.owl#Aluno_de_Mestrado: [Estudante_de_Mestrado]
Sinonimos de file:secondOnto.owl#Aluno: [Estudante]
Sinonimos de file:secondOnto.owl#Dissertacao: [Dissertacao_de_Mestrado]
Sinonimos de file:secondOnto.owl#Tese: [Tese_de_Doutorado]
Sinonimos de file:secondOnto.owl#Aluno_de_Doutorado: [Estudante_de_Doutorado]
```

Figura 11 – Sinônimos identificados dos conceitos das ontologias analisadas

Apesar de a sinonímia ser uma relação reflexiva, a relação entre suas informações cadastradas no banco de sinônimos criado não é reflexiva, ou seja, se lá existe a informação cadastrada que o conceito "Aluno" é sinônimo do conceito

"Estudante", por exemplo, isto não significa, necessariamente, que o conceito "Estudante" é sinônimo do conceito "Aluno". Nenhum sinônimo é deduzido automaticamente. Todos os conceitos e seus respectivos sinônimos precisam estar cadastrados no banco de sinônimos como entradas únicas para poderem ser utilizados.

A Figura 12 ilustra as informações identificadas na primeira etapa da estratégia para o exemplo escolhido. Os conceitos "Artigo_em_Anais", de ambas ontologias, são alinhados pela igualdade lexical e porque satisfazem as condições de poda desta etapa da estratégia. No entanto, os conceitos "Estudante" de **O1** e "Aluno" de **O2** não são alinhados porque apesar do sinônimo "Estudante" do conceito "Aluno" ser identificado, aqueles conceitos não satisfazem à condição de poda porque possuem conceitos pais, i.e., um nível hierárquico acima, diferentes.

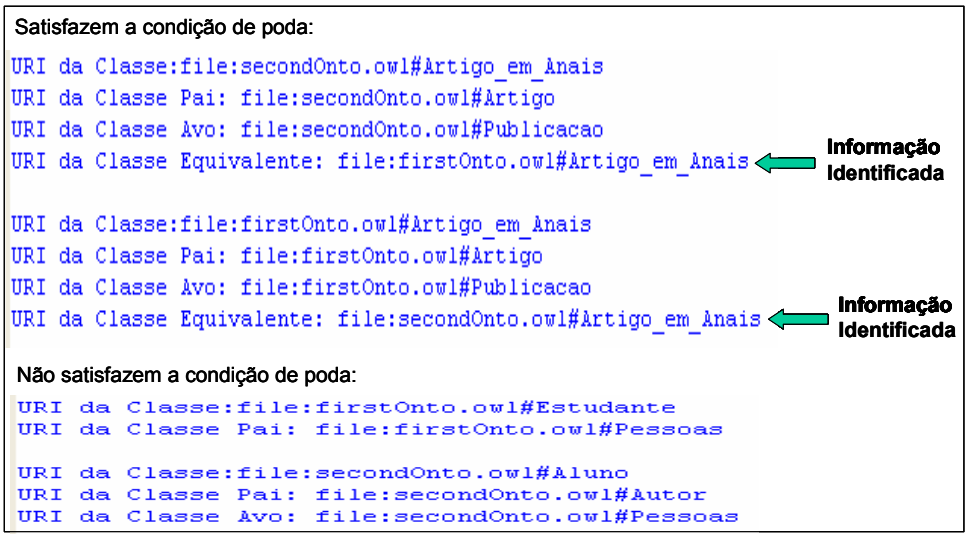


Figura 12 – Informações identificadas na primeira etapa da estratégia

Quando dois conceitos são alinhados, as equivalências são adicionadas nas duas ontologias comparadas. Por exemplo, a informação que o conceito alinhado "Artigo_em_Anais" da primeira ontologia é equivalente ao conceito "Artigo_em_Anais" da segunda ontologia é adicionada em ambas as ontologias.

3.3. Segunda Etapa: Comparação Estrutural Usando uma Implementação do Algoritmo *TreeDiff*

O segundo passo da estratégia compara as estruturas hierárquicas de ontologias com o objetivo de identificar suas sub-árvores comuns. Esta comparação é baseada nos relacionamentos de especialização, i.e., relacionamentos do tipo "é-um", entre os conceitos de uma ontologia. Outros

termos das ontologias, como as propriedades e axiomas, não são comparados atualmente pela estratégia.

Para comparação de árvores, existem algoritmos de busca de similaridades estruturais, tais como: *TreeDiff* (Wang, 1998), *TreeToTree* (Tai, 1979), *TreeMatcher* (TreeMatcher, 2004), entre outros.

O algoritmo do *TreeDiff* (Wang, 1998) descrito em (Bergmann, 2002) é o escolhido por satisfazer os requisitos para a busca de similaridades estruturais e possuir sua implementação disponibilizada.

O objetivo do *TreeDiff* é encontrar a maior subestrutura comum entre duas árvores descritas de acordo com o modelo *Document Object Model – DOM –* (DOM, 2004). A ordem dos filhos de cada nó é levada em consideração. A aplicação do algoritmo resulta em um conjunto de operações de edição, como renomear, remover ou inserir um nó, que deve ser aplicado na primeira árvore de maneira a obter-se a segunda árvore. Cada operação possui um custo associado e o cômputo da subestrutura comum é realizado procurando-se minimizar o custo de edição. A aplicação deste algoritmo pode ser utilizada para identificar tanto as similaridades quanto as diferenças entre as árvores comparadas.

No contexto do alinhamento de ontologias deste trabalho, apenas as informações resultantes de similaridades identificadas são utilizadas. As informações das diferenças entre as árvores são descartadas.

Com poucas modificações na implementação disponibilizada do algoritmo do *TreeDiff* em (Bergmann, 2002), é possível fazer com que todas as operações de edição sejam de renomeação e não mais de inserção, exclusão e renomeação como era originalmente. Estas modificações são necessárias porque uma ontologia não é a evolução da outra na comparação entre ontologias complementares, mas possuem termos comuns que precisam ser identificados e alinhados.

A disponibilidade da implementação dada em (Bergmann, 2002) tornou-se um importante recurso, porém, foi necessária a customização desta solução para o problema de alinhamento a ser tratado por ela. Para uso da implementação com ontologias, uma customização seria ter seus arquivos de entrada escritos nas linguagens para ontologias, como *DAML+Oil* (Connolly et al., 2001) ou *OWL*, e não mais em *XML* como era originalmente.

No entanto, o *parser*, i.e., interpretador, *DOM* que utiliza a visão de árvore a partir do modelo *DOM* de um documento, é implementado apenas para

documentos do tipo *XML*. Um *parser DOM* para documentos escritos nas linguagens de ontologias não foi encontrado e nem existe como padrão da *W3C* (DOM, 2004).

Como para a comparação entre árvores é preciso apenas a representação das estruturas taxonômicas de ontologias, a transformação da linguagem de ontologias para *XML*, respeitando as hierarquias das ontologias comparadas, é aceitável. Isto porque é possível representar em *XML* a hierarquia dos conceitos de uma ontologia, sem perder qualquer informação.

A Figura 13 ilustra como o *TreeDiff* realiza o cômputo das diferenças e similaridades entre dois arquivos *XML*. Inicialmente estes arquivos são lidos e transformados nas duas árvores correspondentes, *DOMTree 1* e *DOMTree 2* pelo *parser DOM*. Sobre estas árvores é aplicado o algoritmo *TreeDiff*, obtendo-se como resultado o conjunto das operações a serem aplicadas na primeira árvore de forma a obter a segunda. Por fim, o conjunto de diferenças é processado de maneira a gerar uma lista de diferenças e um conjunto de fatos observados sobre as duas versões. Este conjunto de fatos é utilizado em (Bergmann, 2002) na procura de qual plano caracteriza as ações realizadas pelo desenvolvedor.

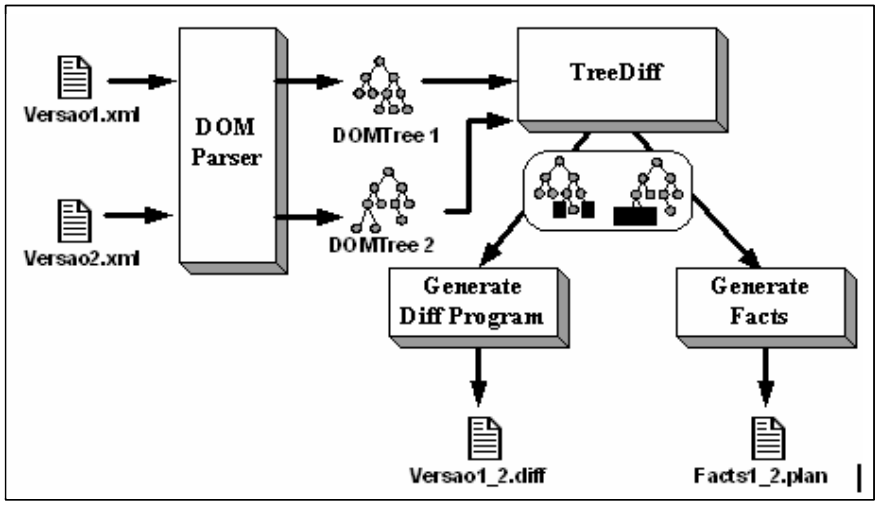


Figura 13 – Uso do algoritmo do *TreeDiff* de (Bergmann, 2002)

A Figura 14 ilustra a transformação da estrutura de ontologias para a estrutura em *XML*. Tais estruturas são bem semelhantes salvo o fato de na última existirem as *tags* (rótulos escondidos com anotações) específicas de Classes (<Classe> nomeDaClasse </Classe>) e Subclasses (<subClasse> nomeDaSubClasse </subClasse>).

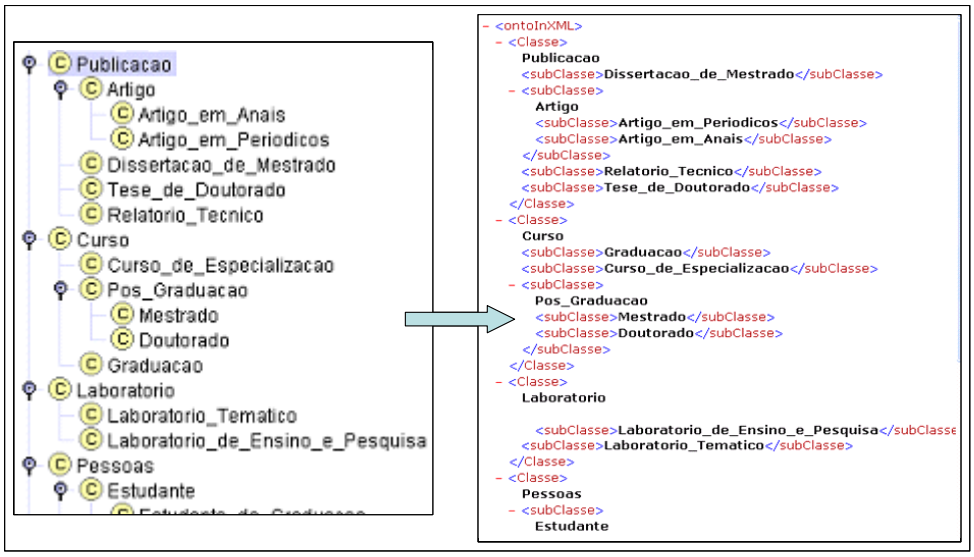


Figura 14 – Transformação da estrutura de ontologias para a estrutura em *XML*

3.3.1. Informações de equivalência

A comparação estrutural da segunda etapa da estratégia utiliza apenas a igualdade de nomes. As equivalências lexicais dos sinônimos, identificadas na primeira etapa da estratégia, não são interpretadas por esta comparação.

Para que os sinônimos identificados como equivalentes aos conceitos sejam utilizados na comparação estrutural, é preciso antes da transformação das ontologias em estruturas hierárquicas, escolher entre o nome do conceito ou o de seu sinônimo identificado a ser utilizado.

A informação de equivalência entre o conceito e seu sinônimo pode ser encontrada nas duas ontologias ou em apenas uma delas. Por exemplo, pode-se ter tanto a informação que o conceito "Aluno" da ontologia **O2** é equivalente ao conceito "Estudante" da ontologia **O1** e o conceito "Estudante" de **O1** é equivalente ao conceito "Aluno" de **O2**, quanto apenas a informação que o conceito "Aluno" de **O2** é equivalente ao conceito "Estudante" de **O1**.

Se a informação de equivalência existir nas duas ontologias analisadas, escolhe-se o nome do conceito ou o de seu sinônimo a ser utilizado na comparação estrutural. No primeiro caso do exemplo acima, tanto o nome do conceito "Aluno" quanto o do conceito "Estudante" podem ser escolhidos.

Caso a informação de equivalência só exista em uma ontologia, então, o nome do conceito que possui a informação de equivalência é substituído. No

segundo caso do exemplo acima, o nome do conceito "Aluno" de **O2** é substituído pelo nome do conceito "Estudante" de **O1**.

No entanto, existe a possibilidade de existência de inconsistências nas substituições de nomes por seus sinônimos. Isto porque, o sinônimo de um conceito de uma ontologia pode estar sendo utilizado como o nome de um conceito da outra ontologia comparada e, talvez, com um significado diferente. Caso isto ocorra, a substituição não será realizada pela estratégia para evitar a existência da inconsistência detectada e a equivalência lexical identificada do nome e de seu sinônimo não será utilizada na comparação estrutural.

3.3.2. Grupos de Equivalência

Como em (Noy e Musen, 2001a), a utilização de grupos de equivalência de conceitos também é utilizada na comparação estrutural implementada. Nesta implementação, os grupos de equivalência são identificados tanto pela comparação lexical quanto pela comparação estrutural. Primeiro, os conceitos comparados com o mesmo nome são identificados pela comparação lexical e, em seguida, as informações da quantidade de filhos (sub-conceitos) e os conceitos equivalentes que estes filhos possuem são analisadas pela comparação estrutural.

Foi verificado na implementação da estratégia que, quando há pouca similaridade estrutural entre as árvores dos conceitos comparados, a organização estrutural dos filhos destes conceitos também influencia na identificação dos grupos de equivalência.

Para investigar a influência da organização dos conceitos nas árvores comparadas, dois módulos de geração de arquivos para a comparação estrutural foram implementados. São eles: um módulo que fornece os conceitos das ontologias estruturados com sua ordem original de criação e um módulo que fornece os conceitos ordenados alfabeticamente. As Figura 15 e Figura 16 ilustram os dois respectivos módulos, com os grupos de equivalência identificados pelos círculos em cada um deles. Estes grupos de equivalência são identificados nas ontologias **O1** e **O2**, que foram apresentadas no tópico 3.1. deste trabalho.

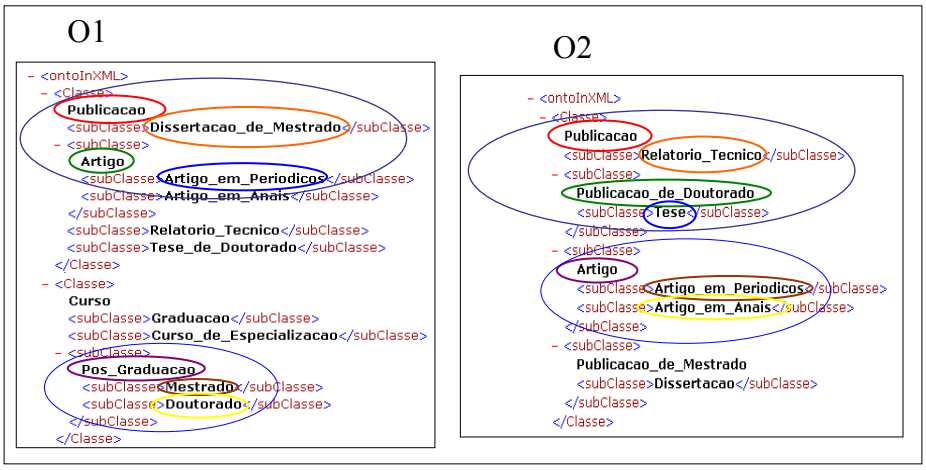


Figura 15 – Grupos de equivalência identificados no módulo sem ordenação alfabética

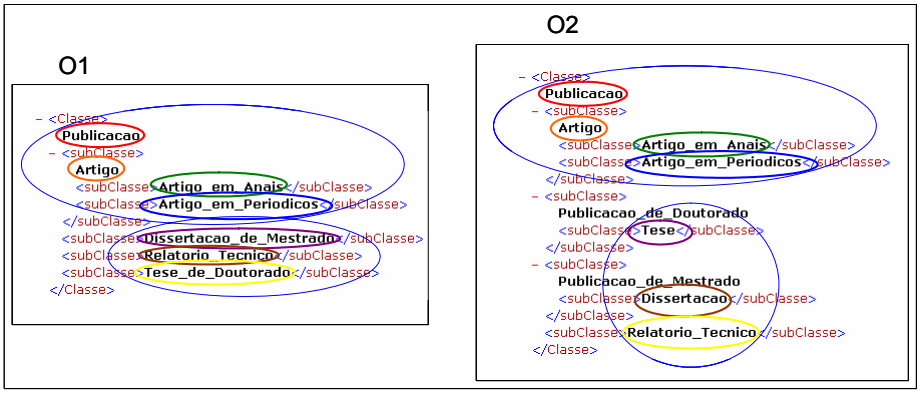


Figura 16 – Grupos de equivalência identificados no módulo com ordenação alfabética

3.3.3. Revendo o Exemplo

Na Figura 15, pela comparação lexical, os conceitos “Publicacao”, de ambas as ontologias, são identificados como *similares* nas duas árvores analisadas. Depois, pela comparação estrutural, seus conceitos filhos são investigados. Como estes três primeiros conceitos, em ambas as ontologias, não possuem filhos então, são identificados como *similares* e os primeiros grupos de equivalência, identificados pelos círculos superiores da Figura 15, são definidos. Em seguida, novamente pela igualdade estrutural, tanto os conceitos filhos “Pos_Graduacao” em **O1** e “Artigo” em **O2** quanto suas sub-árvores são igualados em novos grupos de equivalência, identificados pelos círculos inferiores da Figura 15.

Na Figura 16, também pela comparação lexical, os conceitos “Publicacao”, de ambas as ontologias, são identificados como *similares* nas duas árvores analisadas. Depois, pela comparação estrutural, seus conceitos filhos são investigados. Como estes são idênticos lexicalmente e, pela ordenação alfabética estão no mesmo nível hierárquico, em ambas as ontologias, então, são

identificados como *similares* e os primeiros grupos de equivalência, identificados pelos círculos superiores da Figura 16, são definidos. Como as informações dos sinônimos identificados não são utilizadas porque estes não satisfazem as condições de poda da primeira etapa da estratégia, então, os novos grupos de equivalência, identificados pelos círculos inferiores da Figura 16, são definidos levando-se em consideração apenas as subestruturas hierárquicas dos conceitos comparados.

Percebe-se que para as ontologias comparadas com os conceitos equivalentes descritos com o mesmo nome, a ordenação alfabética traz uma melhoria sensível. Isto porque, após esta ordenação, os conceitos de mesmo nome comparados nos grupos de equivalência estarão próximos estruturalmente. No entanto, para os conceitos equivalentes que usam nomes diferentes ou sinônimos, a ordenação alfabética não traz melhorias e, em alguns casos, pode piorar a solução quando esta distancia estruturalmente os conceitos equivalentes.

O uso do algoritmo do *TreeDiff* fornece as equivalências que não foram descobertas com a comparação lexical da primeira etapa da estratégia porque, possivelmente, os conceitos ali identificados não satisfizeram as condições de poda. No entanto, é na terceira etapa da estratégia, descrita no próximo tópico, que os conceitos identificados como *similares*, pela comparação estrutural, são classificados como *bem similares* ou *pouco similares*. Esta classificação é realizada de acordo com um percentual de similaridade pré-definido. Apenas os conceitos *bem similares* são alinhados no final da estratégia.

Desta maneira, os conceitos “Pos_Graduacao” em **O1** e “Artigo” em **O2**, identificados como *similares* pela comparação estrutural do módulo sem ordenação alfabética, não serão alinhados no final da estratégia porque serão classificados como *pouco similares* pelo uso de medidas de similaridades.

3.4. Terceira Etapa: Uso de Medidas de Similaridades para os Ajustes Finos

A terceira, e última, etapa da estratégia corresponde ao uso de medidas de similaridade para os ajustes finos dos resultados conseguidos na etapa anterior. Tais medidas classificam termos como *bem similares* ou *pouco similares*. A classificação é realizada baseada em um percentual de similaridade previamente escolhido. Apenas os conceitos *similares* da segunda etapa da estratégia e

classificados como *bem similares*, nesta etapa, são alinhados no final da estratégia.

Existe na literatura uma rica bibliografia sobre similaridades e relacionamentos semânticos entre os termos, como: *Conceptual Relations from Text* (Maedche, 2000), *Knowledge maintenance* (Resnik, 1995), *Semantic Similarity between Words* (Richardson, 1994), entre outros.

Em Richardson et al. (1994), por exemplo, é utilizada a combinação da distância conceitual dos termos e a informação de aproximação baseada na estimativa de similaridade semântica. Lá, chegou-se a seguinte expressão para similaridade entre conceitos:

$$Sim(c_1,c_2) = \max_{c_i}[\log \frac{1}{P(C_i)}]$$

onde {Ci} é o conjunto de conceitos entre C1 e C2, P(Ci) é a probabilidade do conceito Ci e log 1/P(Ci) é o índice da informação do conceito Ci.

Para exemplificar esta medida de similaridade, a Figura 17 ilustra alguns de seus valores calculados. Com o uso da expressão de similaridade é descoberto, por exemplo, que os conceitos "carro" (*car*, em inglês) e "garfo" (*fork*, em inglês) são mais similares que os conceitos "carro" e "banana" (*banana*, em inglês). Isto porque "carro" e "garfo" são sub-conceitos de artefato (*artifact*, em inglês), e o conceito "banana" é apenas subconceito de um conceito mais geral chamado "objeto" (*object*, em inglês). Os conceitos "carro" e "garfo" são 1.338 similares, enquanto "carro" e "banana" são 0.763 similares. A maior similaridade entre conceitos é traduzida em um valor calculado mais alto.

Sim(car,fork)		Sim(car,banana)	
<instrumentality>	1.338	< object >	0.763
< artifact >	0.980	< entity >	0.565
< object >	0.763		
< entity >	0.565		

Figura 17 – Valores calculados de similaridade entre os termos

Apesar do material levantado com a pesquisa bibliográfica realizada sobre medidas de similaridade, a implementação de tais medidas escolhida também foi a utilizada em (Bergmann, 2002). Esta implementação é escolhida porque fornece os resultados esperados para a tomada de decisão do alinhamento e leva em conta tanto a estrutura hierárquica dos nós das árvores comparadas, distanciamento dos conceitos comparados às raízes de suas árvores, quanto a quantidade de nós filhos,

i.e., sub-conceitos, similares. Ou seja, implementa o uso de medidas de similaridade aplicadas nos grupos de equivalência identificados na etapa de comparação estrutural da estratégia elaborada. Tais preocupações são indicativos essenciais para o alinhamento de ontologias e, conseqüentemente, foram decisivas para a escolha das medidas de similaridade implementadas em (Bergmann, 2002).

Para a definição do percentual de similaridade, o mais razoável possível, é preciso realizar alguns experimentos. Nestes, as medidas de similaridade são aplicadas no problema com ontologias e, posteriormente, são refinadas.

Atualmente, o percentual de similaridade escolhido é igual a setenta e cinco por cento (75%). Este valor foi obtido empiricamente, através da análise e refinamento de resultados obtidos em experimentos prévios de alinhamento de ontologias. Conceitos com percentual de similaridade igual ou maior que 75% são classificados como *bem similares* e conceitos com percentual de similaridade menor que 75% são classificados como *pouco similares*.

Apenas os conceitos *bem similares* são alinhados no final da estratégia. As novas informações de equivalência são adicionadas nos modelos das ontologias resultantes da primeira etapa da estratégia. Estes modelos com as novas informações são persistidos em uma ontologia única formada pelas ontologias originais, agora, alinhadas.

3.4.1. Revendo o Exemplo

Para exemplificar a terceira etapa da estratégia, o exemplo descrito no tópico 3.1. é revisto.

Os conceitos nomeados “Artigo”, em as ambas as ontologias comparadas, não foram alinhados na primeira etapa porque não satisfizeram à condição de poda de generalização, i.e., não há as informações dos conceitos avôs (dois níveis hierárquicos acima) cadastrados, em ambas as ontologias, para a verificação da condição de poda. No entanto, estes conceitos são identificados como *similares* pela comparação estrutural, porque possuem igualdade lexical entre eles e igualdade estrutural de seus sub-conceitos, e são identificados como *bem similares* pelo uso de medidas de similaridades. Por estas razões, são alinhados no final da estratégia.

A Figura 18 ilustra as informações identificadas na terceira etapa para o exemplo escolhido.

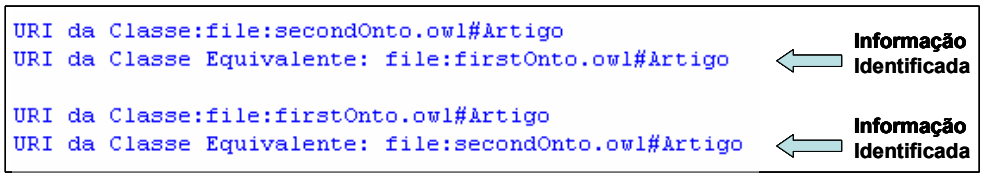


Figura 18 – Informações identificadas na terceira etapa da estratégia

3.5. A Implementação

A estratégia elaborada para o alinhamento taxonômico de ontologias foi implementada em (Felicíssimo, 2003b). A linguagem de programação orientada a objetos Java (Java, 2000) é utilizada em toda esta implementação por possuir finalidades gerais e boas características que foram decisivas para sua escolha. Algumas de tais características são descritas no próximo tópico. A *API – Application Programming Interface* – específica para manipulação de ontologias, a *API Jena*, a linguagem de ontologias *OWL* e o resultado da implementação realizada são apresentados nos tópicos seguintes.

3.5.1. Características da linguagem Java

Um programa escrito em Java é multi-plataforma, portável e escalável. Multi-plataforma e portável porque é compilado para um único *bytecode* Java, independente do sistema operacional onde é executado. O *bytecode* Java, como explicado em (Sikora, 2003), é um código intermediário entre o código-fonte do programa Java e o código de máquina necessário para sua compilação. Este *bytecode* pode ser executado por qualquer interpretador Java, sendo este específico de sistema operacional, que esteja de acordo com as especificações da Máquina Virtual Java (Java, 1999). Escalável, quando executado em um servidor Web, porque Java é uma linguagem *multithread*.

Uma grande quantidade de bibliotecas programadas para a linguagem Java é disponibilizada na Web. Por esta razão, o reuso de códigos já implementados é facilitado, o que minimiza o tempo de programação nesta linguagem.

Java pode ser estendida com adições de novas *APIs* em sua hierarquia. Isto possibilita a utilização de *APIs* específicas para as soluções programadas nesta linguagem. Por exemplo, a *API* Jena (Jena, 2004b) específica para o tratamento de

ontologias, descrita a seguir, é utilizada na implementação realizada para a estratégia elaborada de alinhamento.

3.5.2. A API Jena

Jena é um *framework* em Java para construção de aplicativos para a Web Semântica que provê um ambiente de programação para as linguagens de ontologias *RDF* (Miller, 2004), *RDF Schema – RDF Vocabulary Description Language* – (Brickley, 2004) e *OWL* (Dean et al., 2004a), incluindo um mecanismo ou motor de inferências baseado em regras (Jena, 2004a). Trata-se de um projeto iniciado nos laboratórios de pesquisa para a Web Semântica da empresa HP (HP, 2004), mas que hoje é parte integrante dos projetos da comunidade de software livre (Open Source, 2004).

Jena possui uma *API* específica para o tratamento de ontologias (Jena, 2004b). Esta *API* transforma uma dada ontologia em um modelo abstrato de dados orientado a objetos e, com isto, seus termos passam a ser manipulados como objetos. Este modelo é baseado na linguagem em que a ontologia é escrita. Por exemplo, existem modelos específicos para *OWL*, *DAML+OIL – Darpa Agent Markup Language + Ontology Inference Layer* – (Connolly et al., 2001), *RDF*, entre outros. Isto porque, a *API* em questão recupera as informações das *tags* das ontologias, que são específicas para cada uma das linguagens de ontologias. Nenhuma informação é deduzida, porque a *API* não é um mecanismo de inferência.

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

3.5.3. A linguagem *OWL*

O grupo de trabalho de Ontologias no contexto da *W3C* (W3CSemanticWeb, 2004) vem desenvolvendo e evoluindo uma série de linguagens para ontologias tendo como padrão, atualmente, a linguagem *OWL* (Dean et al., 2004a). Esta linguagem é influenciada por formalismos

estabelecidos, por paradigmas de representação do conhecimento e pela existência de outras linguagens para ontologias e para a Web Semântica (Horrocks et al., 2003). Satisfaz, sobretudo, seus requisitos definidos em (Dean, 2004b).

A linguagem *OWL* é uma nova linguagem para ontologias. Porém, esta linguagem respeita a arquitetura da Web Semântica, ilustrada na Figura 1 deste trabalho, evoluindo suas linguagens bases: *XML* (XML, 2004), *RDF* e *RDF Schema*, além de ser uma revisão da linguagem *DAML+OIL*, com suas lições aprendidas de projeto e aplicação.

A sintaxe da linguagem *OWL* é fornecida pelo *XML*, com o esquema de *tags*, i.e., rótulos escondidos com anotações; o *framework* para a representação de informação na Web através da modelagem de seus meta-dados[10] e das relações entre eles é fornecido pelo *RDF*; o vocabulário para descrição dos conceitos e relações dos recursos, com semântica para as hierarquias de especialização das relações e dos conceitos, é fornecido pelo *RDF Schema*.

A linguagem *OWL* é mais sofisticada que suas linguagens bases porque, por exemplo, seus conceitos podem ser especificados por combinações lógicas como interseção, união, ou complementos de outros conceitos, ou por enumerações de objetos especiais. *OWL* pode indicar que uma propriedade é transitiva, simétrica, funcional ou inversa, em relação a uma outra propriedade, quais indivíduos, i.e., instâncias, pertencem à quais conceitos, que conceitos e propriedades podem usar indicações de equivalência e disjunção, que indicações de igualdade e diferença podem ser utilizadas entre indivíduos, entre outras informações fundamentais para fornecer o suporte semântico necessário para os primeiros passos da Web Semântica.

3.5.4. O CATO

O Componente para Alinhamento Taxonômico de Ontologias (CATO) é o resultado da implementação realizada em (Felicíssimo, 2003b) da estratégia elaborada para o alinhamento automático de ontologias.

[10] Os meta-dados expressam informações sobre os dados que representam. Por exemplo, assunto, autores, editora e data de publicação são informações adicionais ou os meta-dados de um livro. Essas informações podem ser utilizadas, por exemplo, na procura de livros em uma biblioteca.

Este componente pode ser utilizado em aplicações que precisam que suas informações compartilhadas sejam interpretadas automaticamente como, por exemplo, no *framework* proposto em (Haendchen et al., 2003). Neste *framework*, o alinhamento permitiria uma representação comum das informações dos serviços de agentes de software, o que facilitaria a interpretação automática dessas informações por estes dispositivos de software.

O CATO recebe duas ontologias como suas entradas. Estas ontologias devem estar descritas na linguagem padrão atual para ontologias adotada pela *W3C*, a linguagem *OWL* (Dean et al., 2004a), e construídas de acordo com as regras de construção desta linguagem (Smith et al., 2004).

Devido a uma decisão de implementação, a saída do CATO é uma ontologia única, também escrita em *OWL*, constituída por todos os termos originais das duas ontologias entradas mais as informações das equivalências identificadas com o alinhamento adicionadas. Os termos nesta ontologia final continuam com seu *namespace* original, o que facilita a identificação de suas origens e, conseqüentemente, sua rastreabilidade.

O fato do resultado do CATO ser uma ontologia única foi exclusivamente devido a uma decisão de implementação. No entanto, as ontologias originais são reconhecidas pela identificação de seus *namespaces* e existe a ligação entre os conceitos equivalentes nesta ontologia única, permitindo a reutilização e o compartilhamento de suas informações. Por estas razões, o mecanismo de alinhamento continua sendo caracterizado.

A implementação do CATO é modular. Alguns de seus módulos principais com seus métodos são representados pelas classes ilustradas na Figura 19. A descrição, os parâmetros de entrada e os valores retornados dos métodos destes módulos são descritos no Anexo D deste documento. Os códigos-fonte da implementação do CATO estão disponíveis no Anexo E.

O componente ilustrado nomeado “CATO.bat” é o responsável pelo disparo da execução dos três módulos principais referentes às três etapas da estratégia implementadas pelo CATO.

A classe ilustrada nomeada “SolCombSinonimos” é a responsável pelo primeiro módulo de execução do CATO. Esta classe recebe como entradas as duas ontologias a serem alinhadas e tem como saídas estas ontologias enriquecidas com os alinhamentos realizados. Realizados os possíveis alinhamentos, as hierarquias

das ontologias resultantes são representadas por essa classe em arquivos do tipo *XML*.

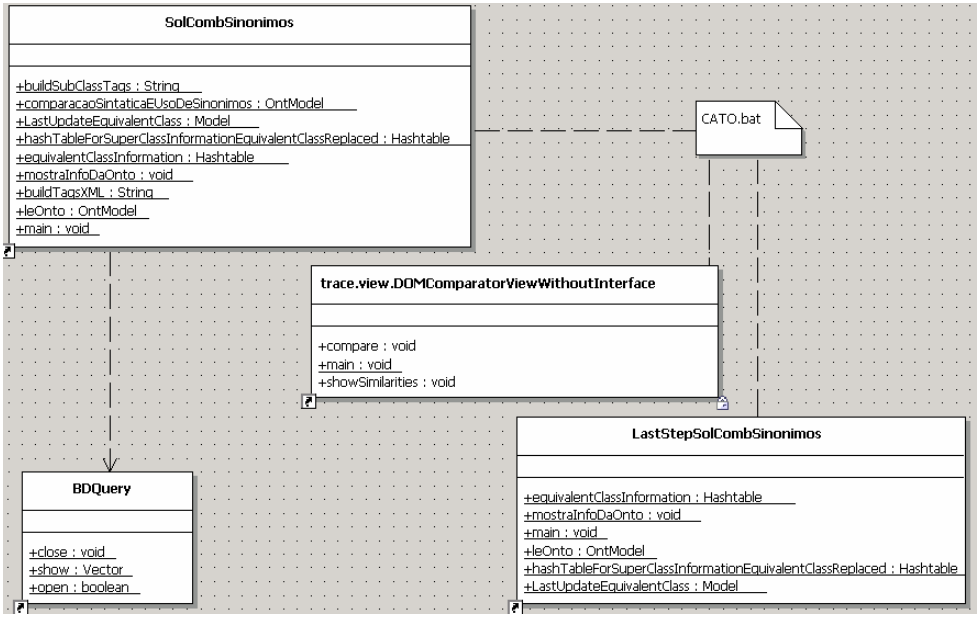


Figura 19 – Classes dos módulos principais do CATO, com seus métodos

A classe "SolCombSinonimos" possui alguns métodos. O nomeado "leOnto" lê uma ontologia e a transforma em um modelo orientado a objetos. O nomeado "comparacaoSintaticaEUsoDeSinonimos" realiza a comparação lexical de cada conceito do primeiro modelo com os conceitos do segundo modelo. Os sinônimos são investigados pela ligação desta classe com a classe nomeada "BDQuery", que estabelece a conexão com o banco de dados utilizado. O método "equivalentClassInformation" fornece as informações do alinhamento desta etapa para que o nomeado "LastUpdateEquivalentClass" persista estas informações nos modelos originais das ontologias. Os métodos "buildTagsXML" e "buildSubClassTags" são os responsáveis pela representação das hierarquias das ontologias em arquivos *XML*.

O método "buildTagsXML" é o único método que difere nas duas versões implementadas do CATO, sem e com ordenação alfabética, devido a função de ordenação chamada *sort* da linguagem Java.

Antes da síntese das ontologias em suas hierarquias, o método nomeado "hashTableForSuperClassInformationEquivalentClassReplaced" substitui o conceito ou o sinônimo identificado como equivalente na comparação lexical da primeira etapa da estratégia para que esta informação de alinhamento possa ser utilizada nas próximas etapas de execução.

A classe ilustrada “trace.view.DOMComparatorViewWithoutInterface” é a classe principal responsável pelos segundo e terceiro módulos de execução do CATO. Esta classe dispara a execução de outras classes específicas da implementação realizada em (Bergmann, 2002) para a comparação estrutural e uso de medidas de similaridades. O seu método nomeado “showSimilarities” é o responsável por passar os alinhamentos destas etapas de execução para serem escritos na ontologia final.

A classe ilustrada nomeada “LastStepSolCombSinonimos” é a responsável pelo módulo que persiste as informações do alinhamento dos três módulos do CATO na ontologia final. Seu método “leOnto”, lê as ontologias resultantes do primeiro módulo e as transforma em modelos orientados a objetos. Nestes modelos são adicionados os alinhamentos realizados nos outros dois módulos. Os dois modelos são unidos e, em seguida, são persistidos em um único arquivo do tipo *OWL* pela função de escrita chamada *write* da linguagem Java. Este arquivo *OWL* é a ontologia que representa o resultado final do CATO.

A arquitetura do CATO é em camadas, como ilustrado na Figura 20. A camada da linguagem Java é a que dá suporte a programação orientada a objetos. A camada da *API Jena* transforma uma ontologia em um modelo orientado a objetos. A camada do CATO é a responsável pelo alinhamento de ontologias.

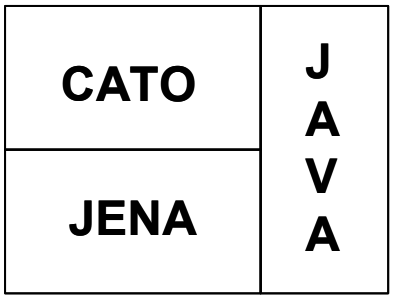


Figura 20 – Arquitetura do CATO

3.5.4.1. Estratégia de Aplicação

Toda a execução do CATO é automática, ou seja, sem intervenção alguma de usuário, precisando apenas ser iniciada com as duas ontologias a serem alinhadas como suas entradas. Estas ontologias, escritas em *OWL*, são ilustradas na Figura 21 pelos arquivos nomeados “firstOnto.owl” e “secondOnto.owl”.

As possíveis informações dos alinhamentos da primeira etapa de execução do CATO (comparação lexical com o uso de sinônimos e mecanismo de poda

estrutural como condição de parada) são adicionadas nas ontologias originais ("firstOnto.owl" e "secondOnto.owl") após a conclusão desta etapa. As ontologias resultantes são ilustradas na Figura 21 pelos arquivos nomeados "newFirstOnto.owl" e "newSecondOnto.owl".

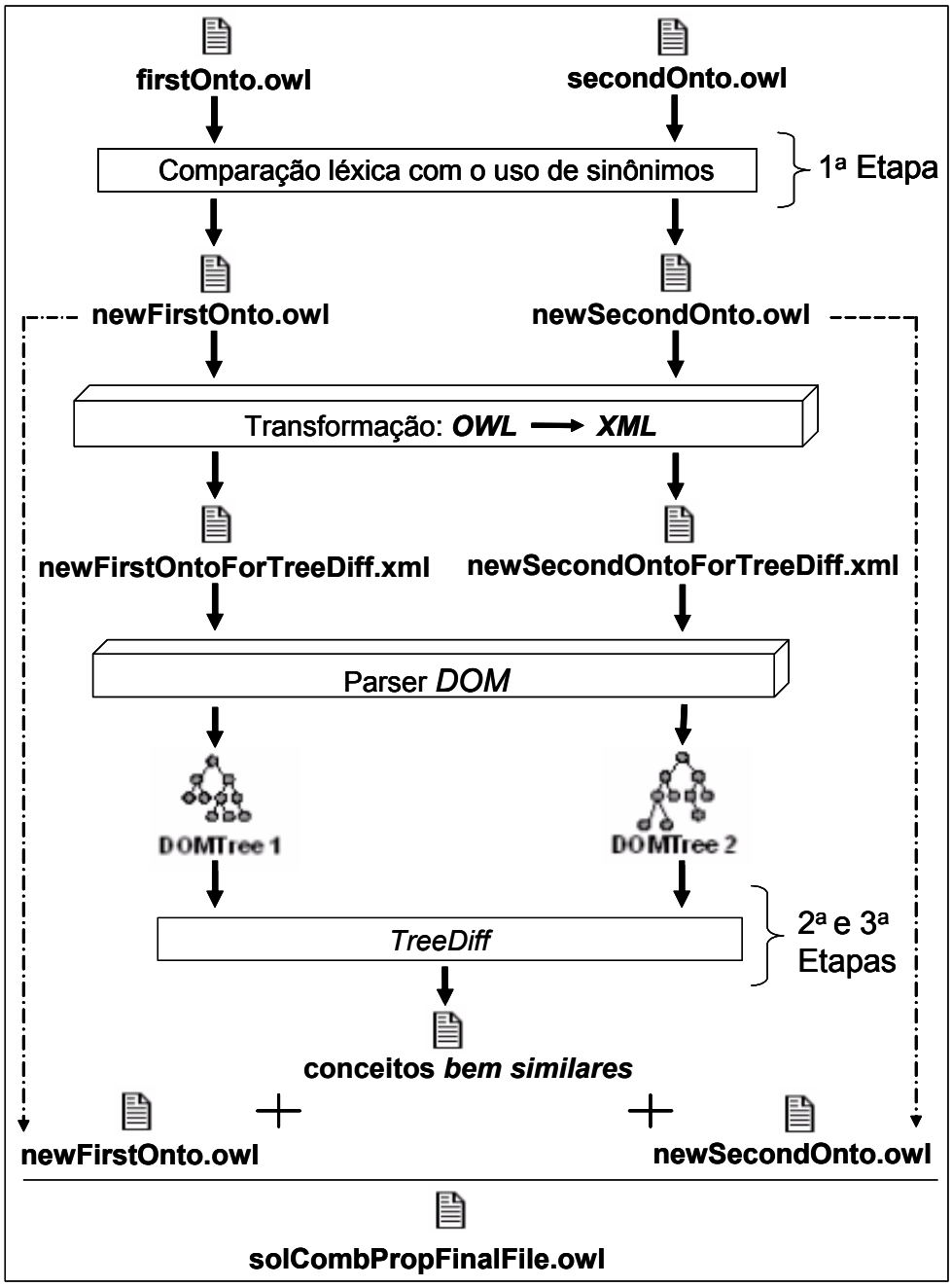


Figura 21 – Entradas e saídas das etapas de alinhamento do CATO

As ontologias "newFirstOnto.owl" e "newSecondOnto.owl" são transformadas em arquivos do tipo *XML*, onde há apenas as representações de suas estruturas taxonômicas (hierarquia de conceitos e sub-conceitos). Os arquivos "newFirstOntoForTreeDiff.xml" e "newSecondOntoForTreeDiff.xml" ilustrados na Figura 21, são os resultados desta transformação de *OWL* para *XML*.

Os arquivos em *XML* são interpretados pelo *parser DOM* que cria as visões de árvores a partir das informações conseguidas com a leitura destes arquivos. Estas visões de árvores, ilustradas na Figura 21 pelas mini-árvores nomeadas "DOMTree1" e "DOMTree2", são as entradas para a execução do algoritmo do *TreeDiff*.

A execução do algoritmo do *TreeDiff*, além de comparar estruturalmente as árvores (segunda etapa de execução do CATO), faz o cálculo das similaridades das sub-árvores identificadas como *similares* (terceira etapa de execução). O arquivo de saída de sua execução, ilustrado na Figura 21, possui os conceitos identificados como *bem similares*.

Nas ontologias resultantes da primeira etapa do CATO são adicionadas as informações dos conceitos classificados como *bem similares*. Após esta adição, as ontologias são unidas em uma ontologia única, ilustrada na Figura 21 pelo arquivo "solCombPropFinalFile.owl". Nesta ontologia estão os resultados conseguidos com o alinhamento, representados pelas ligações entre os conceitos equivalentes identificados das ontologias comparadas.

A ligação entre os conceitos equivalentes é representada na linguagem de ontologias *OWL* pela *tag equivalentClass* e não pela *tag sameAs*. No entanto, esta dúvida é razoável e clarificada a seguir.

A *tag equivalentClass* é utilizada para indicar que dois conceitos são equivalentes se, e somente se, possuem, precisamente, as mesmas instâncias (Smith et al., 2004). Nas *tags* exemplo abaixo, o conceito *Wine* utilizado em uma ontologia local é equivalente, i.e., possui as mesmas instâncias que o conceito *Wine* da ontologia importada referenciada por "vin".

```
<owl:Class rdf:ID="Wine">
   <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```

A *tag sameAs* é utilizada em *OWL* para declarar a igualdade de indivíduos. Apenas em *OWL Full* esta *tag* também pode ser utilizada para representar a igualdade de conceitos (Dean et al., 2004a). Nas *tags* exemplo abaixo, a *tag sameAs* indica que o indivíduo vinho denominado *MikesFavoriteWine* é o mesmo que o conceito denominado *StGenevieveTexasWhite*.

```
<Wine rdf:ID="MikesFavoriteWine">
   <owl:sameAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

Como o alinhamento realizado no CATO é dos conceitos equivalentes das ontologias comparadas, escritas em qualquer um dos tipos de *OWL (OWL Full, DL* ou *Lite)*, e não de suas instâncias, então, a *tag equivalentClass* é a *tag* escolhida. Esta *tag* estabelece as ligações entre os conceitos equivalentes na ontologia final, resultado do alinhamento do CATO.