UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Um processo de integração de instâncias XML baseado em operadores de integração

Carlos Alberto Souza Junior

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau Bacharel em Ciências da Computação

Florianópolis – SC
2008/1

Carlos Alberto Souza Junior

Um processo de integração de instâncias XML baseada em operadores de integração

Trabalho de conclusão de curso apresentada como parte dos requisito para obtenção do grauBacharel em Ciências da Computação

Orientador: Professor Doutor Ronaldo dos Santos Mello

Banca Examinadora

Professor Doutor Renato Fileto

Professor Doutor Fernando Alvaro Ostuni Gauthier

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

A grande quantidade de informações disponíveis na *Web* fez com que houvesse a necessidade de existir sistemas dedicados para a busca dessas informações. XML é o formato mais utilizado para armazenar informações na *Web*, pois os dados são organizados de forma semi-estruturada, não exige homogeneidade na organização de informações de diferentes domínios de dados. A natureza dinâmica do XML torna a integração destes dados complexa. Este trabalho propõe um sistema que provê técnicas para realizar o processo de integração de instâncias XML similares. É definido um conjunto de operadores de integração de instâncias XML para dar apoio ao processo com a finalidade de reduzir diferenças nas estruturas e termos das instâncias. A padronização das estruturas e termos é feita através da utilização de uma ontologia de domínio.

Abstract

The great availability of information in the web created the necessity for dedicated system for searching those information. XLM is the most used format for storing information in the Web, since the data are organized in a semi-structured way, it does not requires homogeneity for structuring information from different data domains. The XML's dynamic nature makes integrating those data a complex task. This work proposes a system that provides techniques for implementing a integration process for similar XML instances. A set of operators for XML instances integration for supporting the process is defined, with the goal of reducing the differences in the structures and terms from the instances. The terms and structures standardization is done by the utilization of a domain ontology.

1 Introdução

Junto com o crescimento da tecnologia e dos meios de comunicação, aumentou-se expressivamente o número de fontes de dados armazenados em meio eletrônico. Este aumento deve-se, principalmente, ao surgimento da *Web,* pois a publicação de informações na mesma tem baixo custo e a forma de acessá-la e pesquisá-la é fácil.

Com a popularização da *Web,* as informações contidas nela podem ter diversas naturezas. Essa diversidade motivou o surgimento de um padrão para o intercâmbio e disponibilização das fontes de dados publicadas. Um dos formatos que se tornou padrão foi a linguagem XML

XML (*eXtensible Markup Language*) [XML 07] é uma linguagem de marcação, padrão da W3C (*World Wide Web Consortium*) [W3C 08] para a representação de dados semi-estruturados.

A forma de representação semi-estruturada em XML torna mais flexível a definição de dados e o suporte à representação de dados da Web. Esta flexibilidade possibilita a heterogeneidade dos documentos que representam dados de um mesmo domínio, ou seja, uma mesma informação pode estar organizada de várias formas distintas entre si tanto na estrutura quanto nos tipos de dados. Desta forma, é comum a existência de informações duplicadas e heterogêneas entre duas ou mais fontes de dados. A recuperação e a integração destas informações, em função disto, é trabalhosa, devido à necessidade de eliminar esta heterogeneidade indesejável [CAR 03].

A heterogeneidade em nível XML, geralmente, consiste em elementos similares, ou seja, que representam a mesma informação, estarem em ordem e/ou níveis distintos na hierarquia dos documentos e possuírem nomes diferentes. Assim sendo, tornou-se necessário o desenvolvimento de sistemas de integração de dados, em especial dados na *Web*, pois o processo de consulta às suas fontes de dados tornou-se complexo. Estes sistemas têm, geralmente, como objetivo o acesso a fontes de dados de um mesmo domínio de aplicação e a apresentação destes dados de forma unificada.

O objetivo da integração de instâncias é, a partir de duas ou mais fontes de dados, que representam a mesma informação, construir uma única que as represente de forma unificada, evitando, assim, duplicidade de informação nos resultados de consultas. Entretanto, alguns sistemas de integração existentes executam tarefas de integração que não tem boa performance.

1.1 Objetivo geral

Visando contribuir para a solução desta problemática, este trabalho propõe a construção de uma ferramenta que dá apoio a um processo de integração de instâncias XML. Este processo é parte de uma dissertação de mestrado em andamento. A ferramenta tem como base operadores de integração. Operadores são métodos que acessam e modificam componentes (elementos ou conteúdos) e estruturas das instâncias

XML. Estes operadores são aplicados objetivando a redução de diferenças estruturais e de nomenclatura existentes entre as instâncias. A redução citada melhora o desempenho do processo de integração.

1.2 Objetivo específico

O processo proposto recebe um conjunto de instâncias XML com seus respectivos graus de similaridade e gera uma única instância representando o conjunto. Ele tem o apoio de uma ontologia de domínio que serve de base para a uniformização de termos e da estrutura da instância resultante da integração. A construção da ontologia está fora do escopo do trabalho.

Ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferência sobre os objetos do domínio. As ontologias compartilham muitas semelhanças estruturais, independente das linguagens em que são expressas. A maioria das ontologias descrevem indivíduos (exemplares), classes (conceitos), atributos e relacionamentos. Uma ontologia de domínio modela um domínio específico, ou parte do mundo. Ela representa os significados dos termos aplicados ao domínio em questão. Algumas das vantagens do uso de ontologias são: fornecer um vocabulário para representação do conhecimento, permitir o compartilhamento de conhecimento, fornecer uma descrição exata do conhecimento e permitir a expansão das mesmas. Do ponto de vista deste trabalho, a ontologia representa uma estrutura conceitual de base para a geração da estrutura e da nomenclatura da instância XML integrada.

O desenvolvimento deste trabalho foi financiado pelo projeto DIGITEX do programa CNPq/CTInfo. DIGITEX é um projeto que foi desenvolvido por pesquisadores da Universidade Federal de Santa Catarina, Universidade Católica de Pelotas e Universidade Federal do Rio Grande do Sul. O DIGITEX consiste na pesquisa do desenvolvimento de um sistema de editoração, indexação, busca e integração de documentos científicos. O Grupo de Banco de Dados da UFSC [GBD 08] atuou neste projeto trabalhando na parte de integração de documentos digitais. Especificamente, o grupo trabalhou no apoio a um processo de integração de instâncias heterogêneas de documentos digitais estruturados semanticamente equivalentes. Um artigo a respeito desse trabalho, descrito em [CAR 07], foi submetido e aceito para apresentação no Workshop de Teses e Dissertações em Banco de Dados do SBBD 2007 (XXII Simpósio Brasileiro de Banco de Dados).

Este trabalho está organizado ainda nos seguintes capítulos:

• **Capítulo 2**: Comenta alguns trabalhos relacionados;

• **Capítulo 3**: Descreve os operadores e o processo de integração;

• **Capítulo 4**: Comenta a implementação do sistema. Neste capítulo são apresentados os pacotes, diagrama de classes e as funções das classes criadas no desenvolvimento do projeto;

• **Capítulo 5**: Apresenta um estudo de caso mostrando a ferramenta em funcionamento, com exemplos de instâncias XML, da ontologia, da árvore da ontologia, das transformações sendo feitas e do resultado final da integração;

• **Capítulo 6**: Reforça o objetivo e relata contribuições e trabalhos futuros.

2 Trabalhos Relacionados

Há trabalhos relacionados na atividade de integração de dados XML que podem ser encontrados na literatura. Alguns se baseiam nas linguagens de consulta *XPath* ou *XQuery* na implementação do processo de integração. Em [MAY 01 a] e [MAY 01 b] é proposta uma extensão à *XPath* chamada *XPathLog*, que utiliza expressões do tipo DataLog e heurísticas para identificar conteúdos duplicados. Entretanto, estas abordagens de integração são custosas e não apresentam boa *performance* pela implementação das sentenças *XPathLog*.

Em [LIU 00], é definido um único operador de integração de dados semi-estruturados, o *integrator*. Ele engloba e estende operadores algébricos relacionais como união e junção. A integração leva em conta a ausência de informação e valores inconsistentes, como por exemplo, nomes incompletos de pessoas. Em [GUH 02] são considerados algoritmos e métricas para a realização de operações de junção entre representações de estruturas heterogêneas de dados XML, a fim de determinar a melhor forma para realizar esta operação. Em [CAR 03], são sugeridas estratégias para a definição de funções de similaridade a serem aplicadas a uma operação de junção de representações heterogêneas, que realiza ou não a junção com base na similaridade indicada pela função.

Em [LIN 04], é definido um operador de casamento (*merge*) que utiliza regras de junção pré-definidas. Em [FON 02], utiliza-se um guia para geração de resultados integrados, chamado *Delta XML*, para realizar as junções entre instâncias XML.

Estes trabalhos citados, diferente desta proposta, não definem um esquema base para definir a estrutura e nomenclatura da instância XML resultante, que no caso deste trabalho, é a ontologia. Portanto, estes trabalhos podem gerar estruturas heterogêneas.

3 Sistema de Integração

Este capítulo irá detalhar os operadores e o processo de integração deste projeto. O capítulo é baseado no trabalho [CAR 07] apresentado no Workshop de Teses e Dissertações em Banco de Dados do SBBD (XXII Simpósio Brasileiro de Banco de Dados) que aborda o processo de integração deste trabalho.

3.1 Operadores de Integração

A qualidade do processo de integração das instâncias de dados XML pode ser prejudicada pelas diferenças estruturais entre as mesmas. Estas diferenças são caracterizadas, principalmente, pela escrita das *tags* (nome dos elementos) e pela estruturação dos dados nas instâncias.

Na escrita das *tags*, pode-se escreve-lás com letras maiúsculas, minúsculas e com nomes compostos. Podem haver possíveis ocorrências de palavras diferentes que são equivalentes, ou seja, representam o mesmo conceito.

Elementos podem ser simples e/ou complexos em instâncias XML. Esta forma de representação pode causar diferenças estruturais entre um conjunto de instâncias. Estas diferenças se caracterizam pela estrutura e hierarquia dos elementos. Uma ocorrência de uma instância pode ter um elemento X como filho de um elemento Y e numa outra instância semelhante o mesmo elemento X ser filho de um elemento Z.

Para resolver os problemas de nomenclatura e estrutura, foi definido um conjunto de operadores que realiza transformações necessárias nas instâncias a serem integradas. Eles são denominados *operadores de integração*.

Os operadores de integração estão definidos em duas categorias: *unários* e *binários*. Eles são aplicados sobre as instâncias XML com base na definição da ontologia visando uniformizá-las e diminuindo, assim, a complexidade da atividade de integração. Portanto, os operadores são os responsáveis pela preparação léxica e compatibilização das instâncias. No escopo deste projeto, a compatibilização consiste na uniformização de termos (*tags*) e reestruturação hierárquica.

Os operadores unários são executados sobre uma instância. Eles são executados em uma etapa denominada pré-processamento (ou transformação). O operador binário é executado sobre duas instâncias, depois de aplicadas as transformações nas mesmas através da execução dos operadores unários. Ele é o responsável pela integração propriamente dita das instâncias passadas como argumento, resultando em uma única representação das mesmas. O conjunto dos operadores unários é o seguinte: *SetAttributes*, *Set*, *Rename*, *Move*, *Remove* e *UnNest*. O operador binário é constituído pelo *Merge*.

A definição dos operadores de integração, com respectivos exemplos, é a seguinte:

• ***SetAttributes(inst, attribs)***: muda todos os atributos dos documentos para subelementos de seus respectivos elementos. Isso faz com que reduzam as diferenças estruturais para não haver comprometimento no processo de integração. No exemplo da

Tabela 1, o atributo *type* do elemento *band*, na primeira coluna, torna-se subelemento do próprio elemento na segunda coluna. O valor do atributo, *progressive,* torna-se o conteúdo do subelemento.

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel
Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   <band>
      <type>progressive</type>
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel
Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

Tabela 1: Aplicação do operador *SetAttributes*

• ***Set(inst, caminho, conteúdo):*** altera, na instância *inst,* os conteúdos dos elementos indicados em *caminho* para o novo conteúdo indicado pelo argumento *conteúdo*. Este operador realiza a uniformização dos conteúdos dos elementos equivalentes quando houver ambos nas duas instâncias a serem integradas e esses contiverem pequenas variações nas representações dos conteúdos. Entre estas variações, estão a abreviação em um dos conteúdos e a presença de pequenos erros na escrita do conteúdo. Para detectar essas variações, utiliza-se métricas de similaridades entre *strings*. Após análises de testes, a métrica escolhida para ser aplicada neste trabalho foi a *Levensthein* [LEV 08]. A Tabela 2 mostra o operador *Set* sendo aplicado no conteúdo do elemento *saxophone*. Verifica-se que, antes da transformação, o nome do conteúdo era Mel Collins e após a operação seu novo nome é Mel C. A atribuição deste novo nome é decidida pelo processo de integração.

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   < band type="progressive">
     <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel C.</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

Tabela 2: Aplicação do operador *Set*

• ***Rename (inst, caminho, novoNome):*** altera, na instância inst, o nome do elemento indicado em *caminho* para o novo nome indicado pelo argumento *novoNome*. Na execução deste operador, é feita a detecção das variações que ocorrem nos nomes das *tags*. As variações que ocorrem nos conteúdos dos elementos, comentado na definição do operador *Set,* também ocorrem na nomenclatura das *tags.* Além destas variações, as *tags* podem ser escritas com letras maiúsculas e minúsculas e podem ser constituídas de palavras compostas separadas por um hífem ou sublinhado. Outra variação que é analisada e organizada por este operador é a utilização dos termos na nomenclatura das *tags.* Um exemplo de variação de termos pode ocorrer em relação a times de futebol,

pois em instâncias semelhantes pode haver a ocorrência do termo *treinador* em uma instância e do termo *tecnico* em outra. Neste caso, ambas têm o mesmo conceito. Este operador adequa a nomenclatura das *tags* de acordo com a ontologia. Na Tabela 3, o operador *rename* está aplicado no elemento *band*, primeira coluna, que teve seu nome alterado para *group* na segunda coluna da tabela.

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   <group type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel C.</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </group>
</bands>
```

Tabela 3: Aplicação do operador *Rename*

• ***Move(inst, caminhoO, caminhoD, pos)***: move, na instância *inst*, a sub-árvore T, indicada em *caminhoO,* para o nodo N, indicado em *caminhoD,* na posição *pos* dentre os filhos de N, modificando, assim, a estrutura hierárquica da instância. Este operador busca resolver a diferença hierárquica das informações, visando padronizar a estrutura das instâncias XML. Duas ou mais instâncias podem ter a mesma informação representada de forma diferente, por exemplo, hierarquia invertida ou diferente relacionamento, ou seja, os mesmos elementos têm pais diferentes. O operador *move* toma a árvore da ontologia como referência estrutural para sua execução. Como resultado da execução deste operador, um determinado elemento tem um novo pai, após o seu deslocamento. Na ilustração da Tabela 4, o elemento *saxophone* do segundo elemento *band* foi movido para o primeiro elemento *band*.

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
   <band type="punk">
      <name>X-Ray Spex</name>
      <vocals>Poly Styrene</vocals>
      <saxophone>Laura L.</saxophone>
      <guitar>Someone else</guitar>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <saxophone>Laura L.</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
   <band type="punk">
      <name>X-Ray Spex</name>
      <vocals>Poly Styrene</vocals>
      <guitar>Someone else</guitar>
   </band>
</bands>
```

Tabela 4: Aplicação do operador *Move*

• ***Remove (inst, caminho)***: remove o elemento indicado pelo argumento *caminho* na instância *inst*. O objetivo desta operação é eliminar um elemento que não têm uma conceitualização na ontologia, ou seja, ele é uma propriedade irrelevante para o domínio em questão. No exemplo da Tabela 5, o operador *remove* foi aplicado ao segundo elemento *band*, filho de *bands*. Quando o elemento removido for complexo, os seus respectivos filhos também serão removidos.

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
   <band type="punk">
      <name>X-Ray Spex</name>
      <vocals>Poly Styrene</vocals>
      <saxophone>Laura L.</saxophone>
      <guitar>Someone else</guitar>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   <band type="progressive">
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <saxophone>Mel Collins</saxophone>
      <saxophone>Laura L.</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

Tabela 5: Aplicação do operador *Remove*

• ***UnNest(inst, caminho)***: transforma um elemento complexo da instância *inst*, indicado em *caminho*, em um elemento simples. Nesta operação, todas as sub-árvores de nodos filhos do elemento complexo são substituídas por um conteúdo textual resultante da concatenação dos conteúdos de todos os seus elementos simples. Este operador é aplicado quando for detectado em elemento complexo numa instância e na ontologia seu respectivo conceito é uma entidade simples. Na instância mostrada na Tabela 6, o elemento *instrumentals* sofreu o *UnNest*, ou seja, os conteúdos de seus elementos filhos tornaram-se o conteúdo dele próprio e os seus elementos filhos não existem mais.

```
<?xml version="1.0"?>
<bands>
   <band>
      <type>progressive</type>
      <name>King Crimson</name>
      <instrumentals>
         <guitar>Robert F.</guitar>
         <saxophone>Mel</saxophone>
         <bass>Boz</bass>
      </instrumentals>
   </band>
</bands>
```

```
<?xml version="1.0"?>
<bands>
   <band>
      <type>progressive</type>
      <name>King Crimson</name>
      <instrumentals>Robert F  Mel Boz
      </instrumentals>
   </band>
</bands>
```

Tabela 6: Aplicação do operador *UnNest*

• ***Merge(inst1, inst2)***: unifica as instâncias *inst1* e *inst2*. A aplicação deste operador é feita após a passagem de *inst1* e *inst2* pela etapa de pré-processamento, ou seja, os operadores unários foram aplicados em ambas as instâncias de forma que a nomenclatura e a estrutura já estejam uniformizadas com base na árvore da ontologia. A Tabela 7 mostra duas fontes de dados XML a serem integradas. Na Tabela 8 temos o resultado da operação merge aplicada sobre as duas instâncias.

O operador *Merge* resolve os seguintes conflitos: elementos encontrados apenas em uma das instâncias e sub-elementos com cardinalidades diferentes. Nodos presentes em apenas uma das instâncias são anexados ao resultado. Nodos duplicados em ambas as instâncias são inseridos uma única vez no resultado e nodos idênticos em nomenclatura/estrutura, mas com conteúdos heterogêneos são ambos inseridos no resultado, como é o caso do elemento *vocals* das instâncias da Tabela 7.

```
<?xml version="1.0>
<bands>
   <band
      <type>progressive</type>
      <name>King Crimson</name>
      <guitar>Robert F.</guitar>
      <vocals>Poly Styrene</vocals>
      <vocals>Johnn Neil</vocals>
      <saxophone>Mel C.</saxophone>
      <bass>Boz</bass
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

```
<?xml version="1.0>
<bands>
   <band
      <type>progressive</type>
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <vocals>Johnn Neil</vocals>
      <saxophone>Mel C.</saxophone>
      <bass>Boz</bass
      <drums>Ian Wallace</drums>
   </band>
</bands>
```

Tabela 7: Aplicação do operador *Merge* - instâncias de entrada

```
<?xml version="1.0"?>
<bands>
   <band>
      <type>progressive</type>
      <name>King Crimson</name>
      <guitar>Robert Fripp</guitar>
      <vocals>Poly Styrene</vocals>
      <vocals>Johnn Neil</vocals>
      <saxophone>Mel C.</saxophone>
      <bass>Boz</bass>
      <drums>Ian Wallace</drums>
</band>
</bands>
```

Tabela 8: Aplicação do operador *Merge* – resultado

3.2 O Processo de Integração

O conjunto de instâncias XML similares e a similaridade entre os pares dos documentos são fornecidos como parâmetros de entrada ao processo de integração. Os graus de similaridade são definidos e obtidos de um outro processo que não pertence ao escopo deste trabalho. Este outro processo também é parte integrante do projeto DIGITEX. Instâncias XML similares são instâncias que possuem semântica semelhante.

A Figura 1 mostra um exemplo da execução do processo de integração das instâncias XML. Verifica-se que a integração das instâncias é feita em pares, começando pelas instâncias mais similares. A ordem das instâncias a serem integradas é definida com base nos valores dos graus de similaridade. Na Figura 1, as instâncias 1 e 2 têm o maior grau de similaridade: 97. Portanto, o processo inicia a integração por essas duas instâncias. O processo cria instâncias intermediárias para cada par de instâncias que são integradas. Estas instâncias intermediárias, por sua vez, são integradas com as demais instâncias do conjunto e assim por diante. A intenção dessa abordagem é sempre integrar as instâncias mais similares possíveis, com estruturas e nomenclaturas mais afins, a fim de diminuir a quantidade de transformações a serem realizadas nas mesmas e melhorar o desempenho da integração. O processo de integração é subdividido em alguns módulos responsáveis por tarefas específicas.

Uma ontologia de domínio é colocada no processo para haver padronização de nomes de elementos e de estrutura do resultado gerado. Um padrão de árvore é obtido a partir de uma análise feita entre a ontologia e as instâncias XML. Esta análise consiste em verificar se, em cada instância para cada elemento XML, existe um conceito semanticamente equivalente na ontologia. O elemento é adicionado no padrão de árvore, caso seu conceito seja encontrado em alguma das instâncias. Ao final desta etapa, a árvore contém apenas nodos que representam os elementos do conjunto de instâncias a serem integradas. O objetivo da padronização da árvore é torná-la mínima possível para obter melhor desempenho no processo de integração.

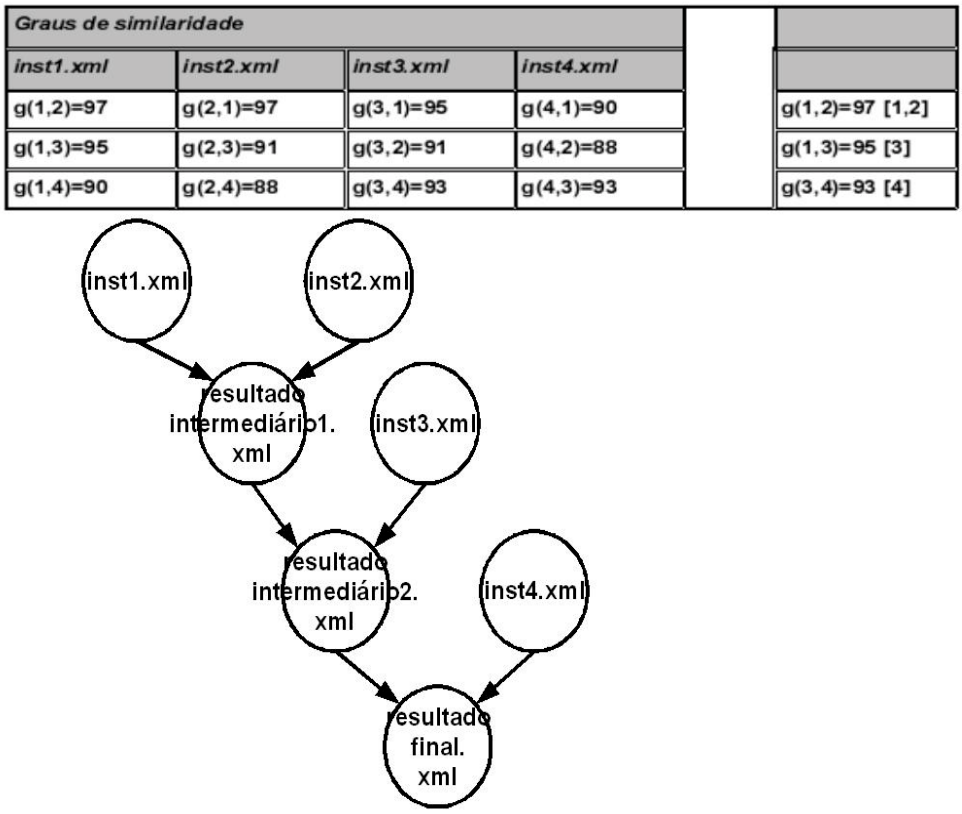

Figura 1: Exemplo de execução do processo de integração de instâncias XML

O padrão de árvore serve como referência de estrutura e nomenclatura para as instâncias XML, garantindo que o resultado seja homogêneo e de acordo com a semântica particular do contexto em que os dados integrados estão sendo gerados [CAR 07].

Conforme mostra a Figura 2, a tarefa de padronização da árvore da ontologia é realizada por um módulo denominado *Wrapper da Ontologia*. O nodo raiz deste padrão de árvore corresponde ao conceito semanticamente equivalente ao elemento raiz das instâncias XML. Assume-se, neste trabalho, que as instâncias XML a serem integradas representam ocorrências de conceitos existentes na ontologia.

O processo de integração é composto por alguns módulos que são descritos a seguir.

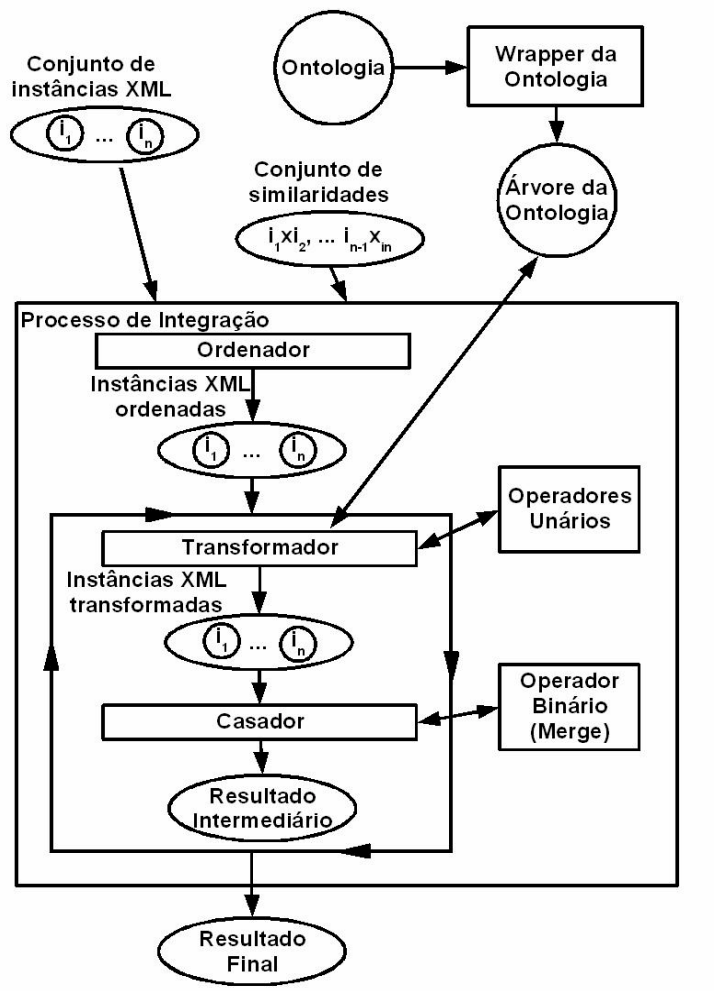

Figura 2: Visão geral do processo de integração de instâncias XML

O módulo *Ordenador* define a precedência de casamento para as instâncias XML e determina a ordem em que as instâncias são enviadas ao módulo *Transformador*, de acordo com o grau de similaridade existente entre elas.

O módulo *Transformador* uniformiza a estrutura de cada uma das instâncias XML de acordo com o padrão de árvore estabelecido. Este módulo é responsável pelo pré-processamento das instâncias antes das mesmas serem integradas.

A primeira tarefa do módulo *Transformador* é aplicar o operador *SetAttributes* para converter todos os atributos contidos na instância sendo transformada em elementos simples. O próximo passo da compatibilização é a renomeação dos elementos para adequar a nomenclatura da estrutura da instância ao padrão de árvore. Esta etapa é realizada pelo operador *Rename* e também conta com o apoio de um dicionário semântico ou técnica de similaridade para encontrar elementos e conceitos semelhantes. O *Rename* é o segundo operador a ser aplicado para tornar a análise entre as instâncias e a ontologia, na execução dos próximos operadores, mais eficiente.

Após a realização das renomeações necessárias, é feita a análise na estrutura hierárquica das instâncias. Caso seja encontrada alguma incompatibilidade com as associações existentes na ontologia, ela é corrigida com a aplicação do operador *Move* que ajusta a posição dos elementos de acordo com o padrão de árvore. Este operador

deve ser aplicado antes do operador *Remove*, pois algum elemento, que é uma propriedade relevante para o domínio em questão, pode ser filho de algum elemento que não tenha um conceito representado na ontologia. O elemento relevante seria removido, pois o elemento pai deste elemento relevante iria ser removido por não está representado na ontologia.

Em seguida, o processo verifica se a utilização do operador *UnNest* é necessária. Este operador é aplicado de modo a adequar heterogeneidades estruturais no que diz respeito ao detalhamento de propriedades das instâncias XML. Como explicado na seção anterior, este operador transforma elementos complexos em elementos simples. As sub-árvores de nodos filhos do elemento complexo são substituídas por um conteúdo textual resultante da concatenação dos conteúdos de todos os seus elementos simples.

Neste momento, a instância sobre a qual o módulo *Transformador* está agindo está com a estrutura hierárquica compatibilizada de acordo com a árvore padrão.

Na seqüência, o processo procura por elementos que não estão representados na ontologia, pois como explicado na seção anterior, estes elementos são irrelevantes para o domínio em questão. O operador *Remove* irá executar esta tarefa através do seguinte procedimento: em cada instância, para cada elemento XML, procura-se na árvore padrão um nodo que represente o tal elemento com o apoio de um dicionário semântico ou técnica de similaridade. Se o nodo não for encontrado na árvore, então o elemento é removido da instância, pois são mantidos apenas os elementos que têm um conceito na ontologia.

Por fim, o módulo *Transformador,* percorre a instância para encontrar diferenças nos conteúdos de elementos equivalentes. Esta tarefa é realizada pelo operador *Set.* Este operador, junto com o *SetAttibutes,* não realiza consultas na árvore obtida da ontologia, pois esta serve apenas para fazer análise da nomenclatura das *tags* e estrutura hierárquica. O operador *Set* avalia os conteúdos de duas instâncias. Os conteúdos dos elementos da primeira instância do módulo *Ordenador* são assumidos como sendo os conteúdos padrão para a nomenclatura dos conteúdos de outras instâncias. Portanto, o operador *Set* será aplicado a partir da segunda instância. Quando o operador for aplicado na instância n (para todo $n > 2$) do *Ordenador,* os conteúdos da instância resultante da integração das instâncias $n - 2$ e $n - 1$ servirão de base para a nomenclatura dos conteúdos da instância n.

Após a realização das transformações em duas instâncias, as mesmas passam para o módulo *Casador*, para que o operador *Merge* seja aplicado nelas, unificando-as. A execução dos módulos *Transformador* e *Casador* continua enquanto houverem instâncias similares a serem integradas (representado pelo ciclo na Figura 2).

4 Implementação do Processo

A implementação do processo de integração foi realizada com a linguagem *Java,* uma linguagem de programação orientada a objeto desenvolvida pela *Sun Microsystems* [SUN 08]. Além de ser orientada a objeto, *Java* é uma linguagem portável (independente de plataforma) e é distribuída com um vasto conjunto de bibliotecas (APIs).

4.1 Pacotes

As classes do programa deste projeto estão guardadas em pacotes para organizá-las em função das tarefas que cada uma executa. Os pacotes criados na implementação do projeto podem ser vistos na Figura 3 que também mostra os relacionamentos entre eles. Os pacotes que são apontados pelas setas contém classes que são membros de outras classes contidas nos pacotes que indicam as respectivas setas. Por exemplo, o pacote *xml* contém classe(s) que é(são) membro(s) de classe(s) contida(s) no pacote *gui.*

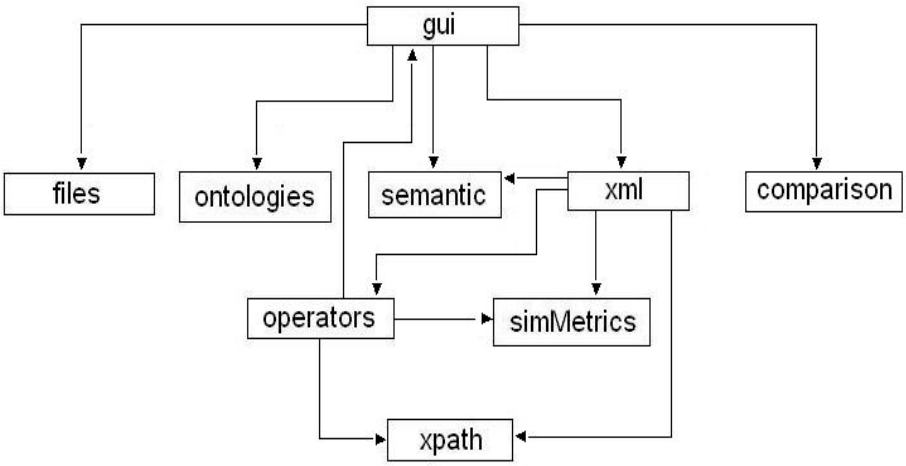

Figura 3: Pacotes da implementação do processo de integração

O pacote *files* obtém a classe que faz a manipulação de arquivos. Qualquer classe do projeto que processar arquivo será armazenada neste pacote.

No pacote *gui* estão as classes que trabalham no desenvolvimento da interface gráfica. As classes que tratam os eventos dos componentes da interface gráfica (botões, menus, campos de texto, etc) também estão armazenadas neste pacote.

O pacote *ontologies* contém as classes que fazem o processamento da ontologia. As classes que trabalham com os operadores para pré-processamaneto e junção das instâncias estão guardadas no pacote *operators.*

semantic é o pacote onde está a classe que é responsável pela análise semântica dos elementos e *simMetrics* contém a classe que faz a tarefa de verificar a similaridade de *strings*.

O pacote *xml* armazena as classes que fazem o processamento dos documentos XML. Neste pacote estão as classes que implementam os módulos que executam o pré-processamento e fazem o casamento das instâncias. Além destas classes, outras, que fazem o processamento das propriedades dos documentos XML (*tags*, conteúdos, atributos, etc), também estão armazenadas neste pacote. No pacote *xpath* está a classe que faz os processamentos *XPath nas* instâncias.

4.2 Classes

Programas em *Java* são formados por um grupo de classes. As classes do programa deste projeto são apresentadas nesta seção. O objetivo aqui é descrever as principais características de cada classe.

As duas classes mostradas na Figura 4 são as classes responsáveis pela criação e o gerenciamento da interface gráfica. Elas pertencem ao pacote *gui. MainWindow* é a classe que inicia a execução do programa. O método *main,* o que inicia um programa *Java* está nesta classe. Todos os componentes de interface gráfica (botões, campo de texto, janelas, etc) são criados nesta classe. Os eventos gerados por esse componentes (clique no botão, clique no menu, etc) são tratados na classe *EventHandler.*

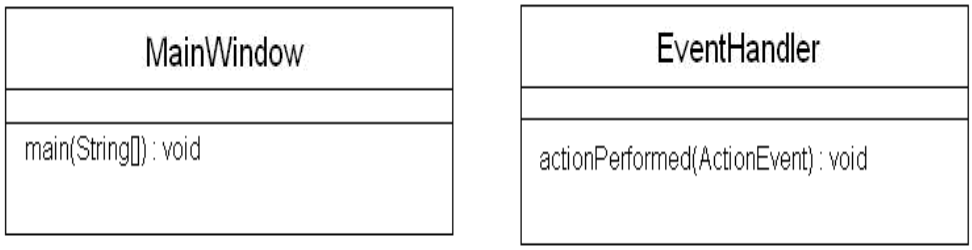

Figura 4: Classes da criação e gerenciamento da interface gráfica

As classes *Ontology* e *OntologyWrapper*, mostradas na Figura 5, pertencem ao pacote *ontologies* e são as responsáveis pelo processamento da ontologia. A classe *Ontology* é responsável pela carga da ontologia. A classe *OntologyWrapper* executa a tarefa de criar a àrvore padrão da ontologia.

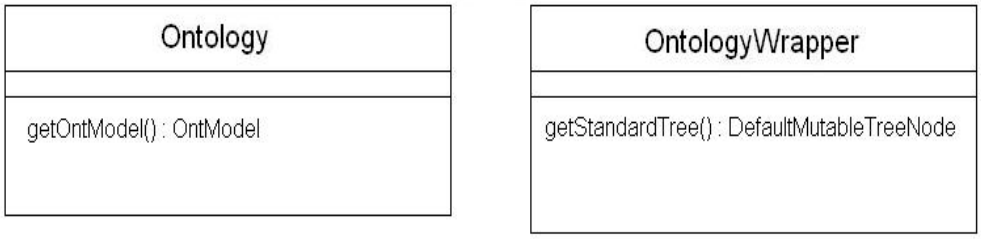

Figura 5: Classes que executam o processamento da ontologia

A Figura 6 mostra o diagrama da classe *Organizer.* Esta classe é responsável pelo trabalho do módulo *Ordenador,* ou seja, ordenar as intsâncias que vão para o módulo *Transformador.* Ela está armazenada no pacote *xml*.

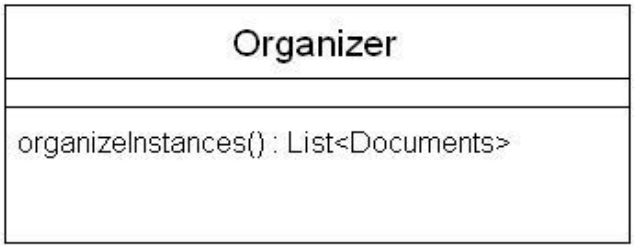

Figura 6: Classe que executa o trabalho do módulo *Ordenador*

As duas classes mostradas na Figura 7 realizam as transformações necessárias nas instâncias antes da integração. A classe *UnaryOperators,* do pacote *operators*, contém os métodos que fazem o pré-processamento das instâncias. A classe *Transformer,* do pacote *xml,* realiza o trabalho do módulo *Transformador* utilizando os métodos da classe *UnaryOperators*.

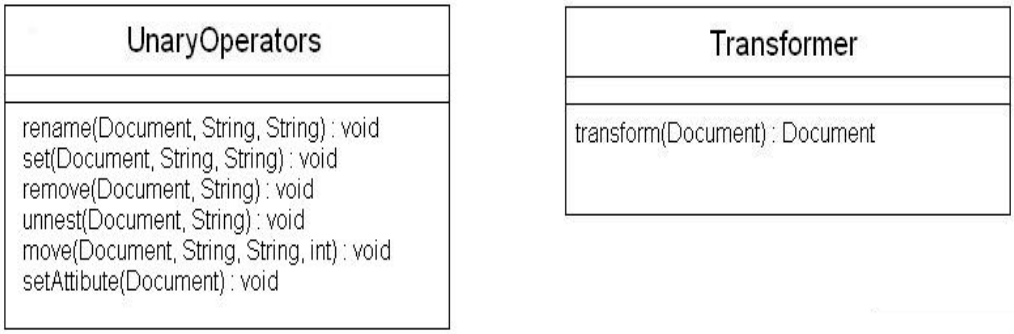

Figura 7: Classes que executam o trabalho do módulo *Transformador*

As classes da Figura 8, *BinaryOperators* e *Matcher*, são responsáveis pela integração das instâncias. *BinaryOperators* contém o método que realiza a integração. *Matcher* executa a tarefa do módulo *Casador* utilizando a classe *BinaryOperators.* A classe *BinaryOperators* está no pacote *operators. Matcher* está no pacote *xml.*

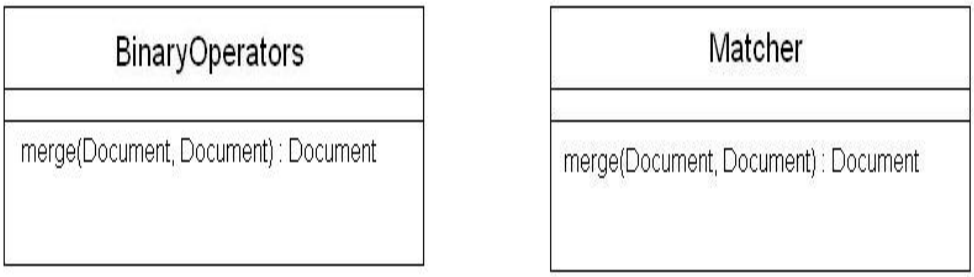

Figura 8: Classes que executam o trabalho do módulo *BinaryOperators*

XMLUtil carrega e cria documentos XML para processamento. A classe *NodeUtil* executa alguns processamentos, requisitados por outras classes, das propriedades dos documentos *XML.* Ambas as classes são do pacote *xml* e o diagrama delas é mostrado na Figura 9.

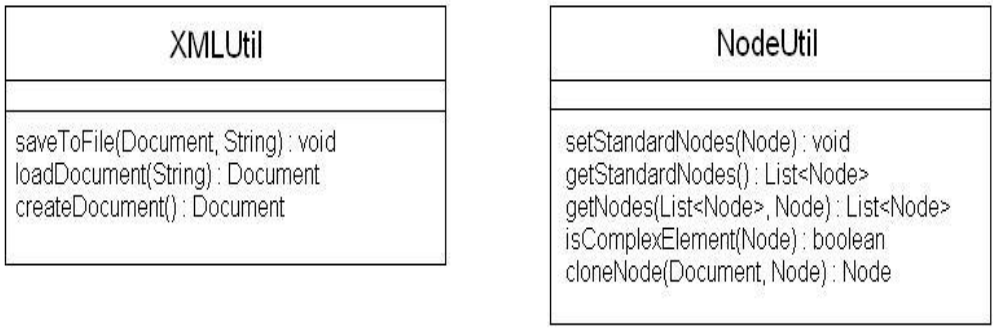

Figura 9: Diagrama das classes *XMLUtil* e *NodeUtil*

SemanticAnalyser e *Similarity,* Figura 10 são responsáveis pela análise semântica e similaridade de *strings*, respectivamente. A classe *SemanticAnalyser* é responsável por relacionar vários termos (encontrados nas instâncias) à um termo específico (encontrado na ontologia). Esta classe representa um dicionário semântico. *SemanticAnalyser* está no pacote *semantic* e *Similarity* é armazenado em *simMetrics*.

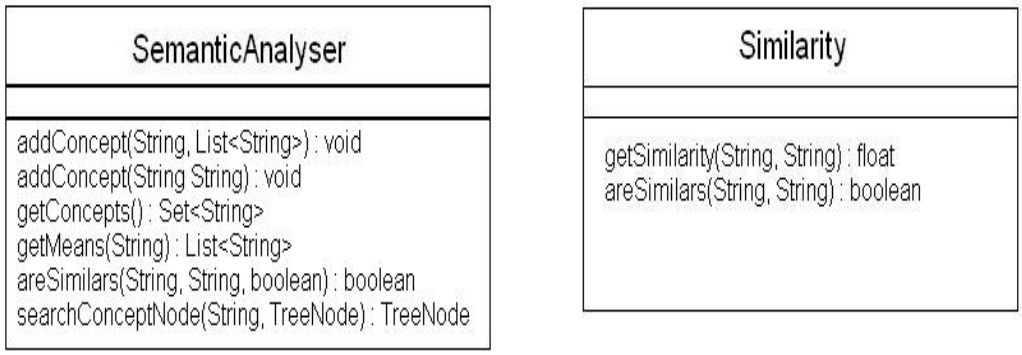

Figura 10: Classes responsáveis pela análise semântica e similaridade de *strings*

As classes da Figura 11, *XPathManager* e *FileManager,* fazem, respectivamente, o processamento das expressões *XPath* e o gerenciamento de arquivos (criar, abrir e salvar). *xpath* e *files* são os respectivos pacotes destas classes.

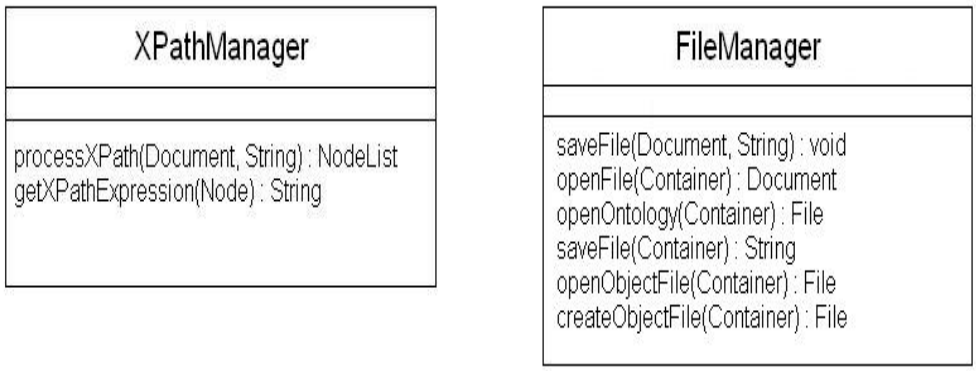

Figura 11: Classes que executam processamento *XPath* e gerenciamento de arquivos

A última classe definida é a *IntegrationProcess*, (Figura 12). Esta classe realiza o ciclo da Figura 2 que contém os módulos *Transformador* e *Casador.* Ela está armazenada no pacote *xml*.

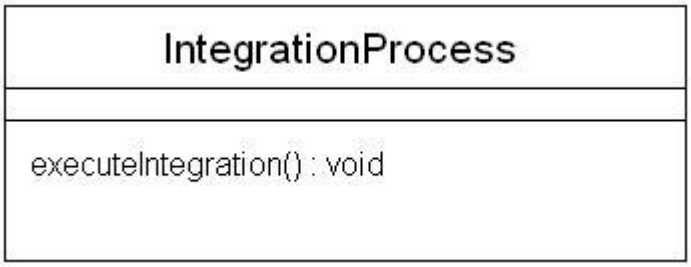

Figura 12: Classe que executa o ciclo da Figura 2

5 Estudo de Caso

O estudo de caso apresentado neste capítulo demonstra o funcionamento da ferramenta que implementa o processo de integração de instâncias XML.

A massa de dados utilizada para realizar o experimento contém dados baseados em um banco de dados da área médica e dizem respeito a dados de exames realizados em pacientes. Os conteúdos dos elementos destas instâncias foram iventados para preservar os dados reais dos pacientes. A Figura 13 mostra um conjunto de instâncias similares que passarão pelo processo de integração neste estudo de caso.

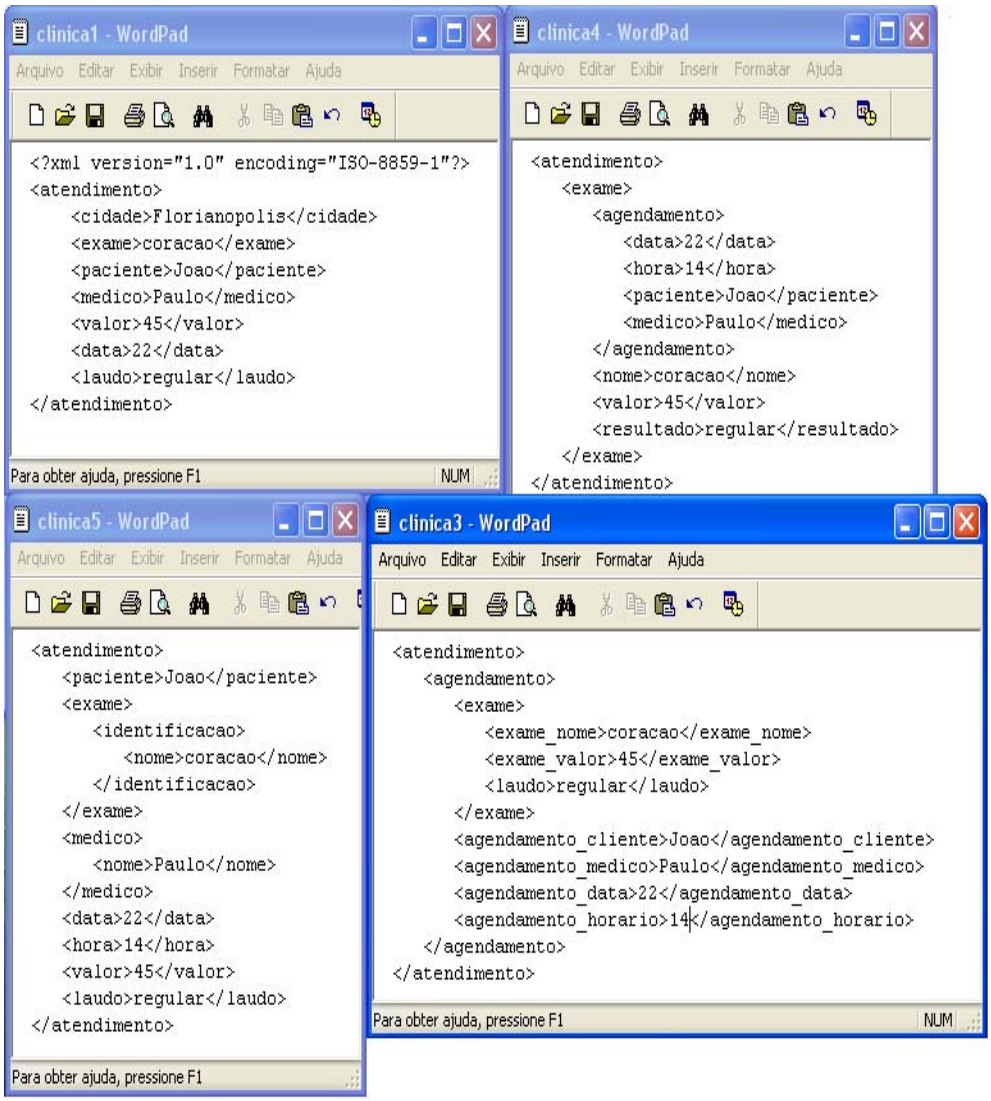

Figura 13: Instâncias utilizadas no estudo de caso

A Figura 14 mostra a ontologia, criada na ferramenta *Oiled* [OIL 08] e utilizada como referência para a execução do processo de integração. *Oiled* é um programa de criação de ontologias criado na *University of Manchester*. Ele foi escolhido por ser software livre e o uso é simples em comparação com outras ferramentas e *Oiled* provê os mecanismos necessários para a construção e uso de ontologias que podem ser

utilizadas neste trabalho. *Oiled* fornece ontologias no formato *daml* apesar de o formato *owl* ser, atualmente, o mais utilizado para a respresentação de ontologias. Isso não significa que o processo não irá funcionar se colocar outros formatos de representação de ontologias, pois este trabalho aceita ontologias representadas em qualquer linguagem, assim como *owl* e *rdf*.

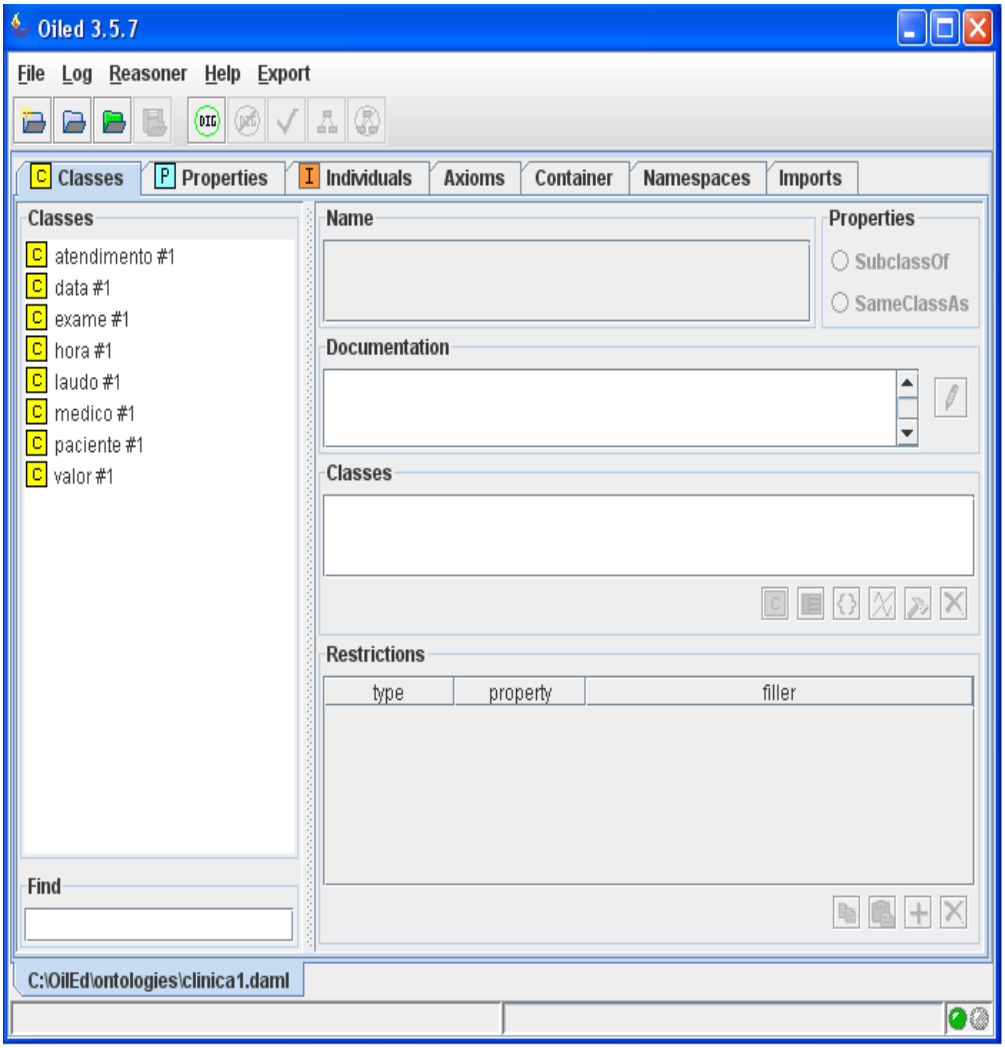

Figura 14: Ontologia utilizada no processo de integração

A Figura 15 mostra a interface com o usuário da ferramenta desenvolvida neste projeto, mostrando os nomes das instâncias que foram selecionadas para serem integradas e os graus de similaridade entre as elas. Como já salientado, esta ferramenta não realiza o processo de comparação de instâncias, pois as similaridades e as instâncias são fornecidas a ela por outro sistema. As similaridades vistas na Figura 15 foram atribuídas por este sistema externo.

O módulo *Ordenador* é aplicado após o clique do botão *Ordenar instâncias*. Após o módulo analisar as similaridades, o módulo *Transformador* recebe as instâncias na seguinte ordem, Figura 15: *clinica1.xml, clinica5.xml, clinica3.xml e clinica4.xml.*

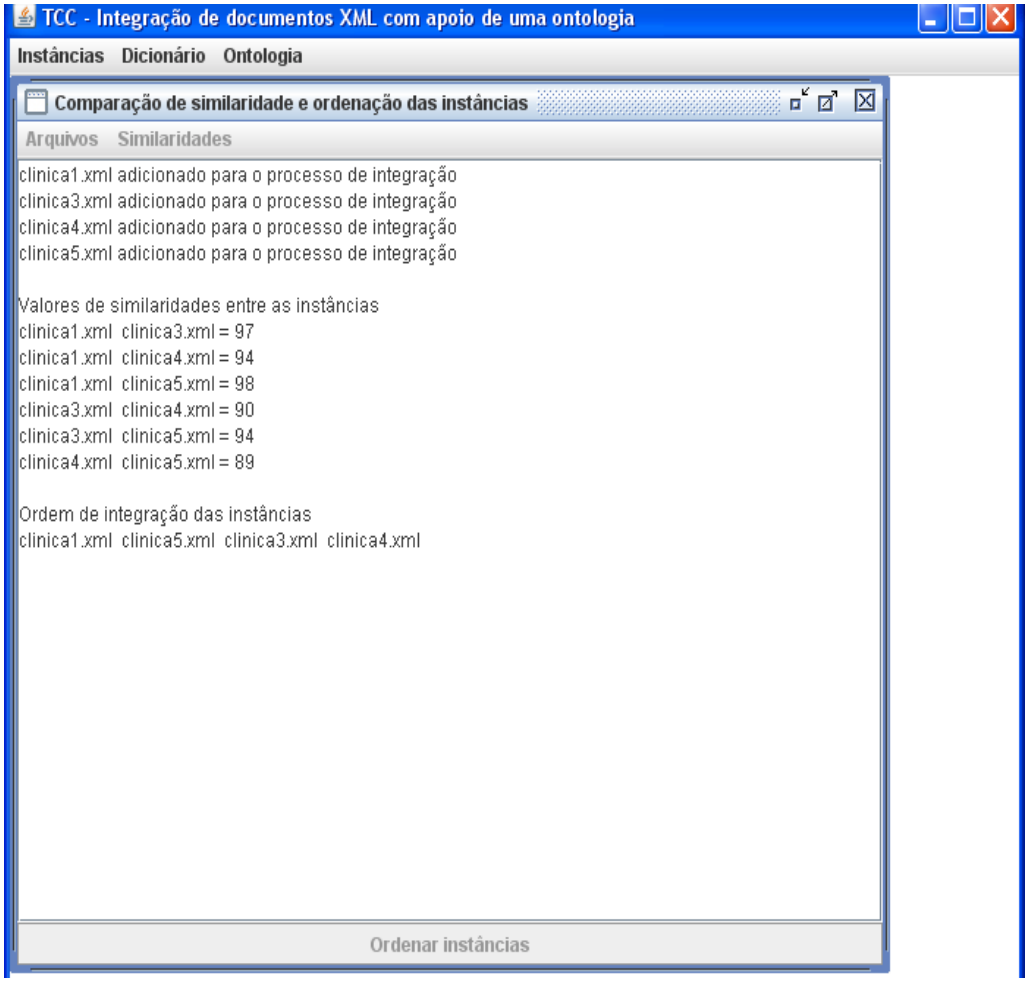

Figura 15: Similaridades entre as instâncias

O usuário pode criar ou alimentar o dicionário semântico (relacionamento de termos). A Figura 16 mostra a definição de alguns conceitos criados por um usuário do sistema. Na figura, a área de texto mostra as palavras "paciente" e "pessoa" como significados do termo "cliente", selecionado no componente *combo box* da figura. A cada termo selecionado no *combo box*, seus respectivos significados são mostrados na área de texto.

O dicionário semântico é um arquivo criado pelo usuário através da seleção do botão *Criar Dicionário*. O usuário pode utilizar outros dicionários que já foram criados em outras execuções do processo de integração, clicando o botão *Abrir dicionário* e adicionar novos conceitos nesses dicionários, clicando *Adicionar novo conceito*. A definição dos conceitos com seus respectivos significados é feita nos campos de textos ao lado dos rótulos *Conceito* e *Significado*. Vários significados (termos) podem ser definidos para um termo específico.

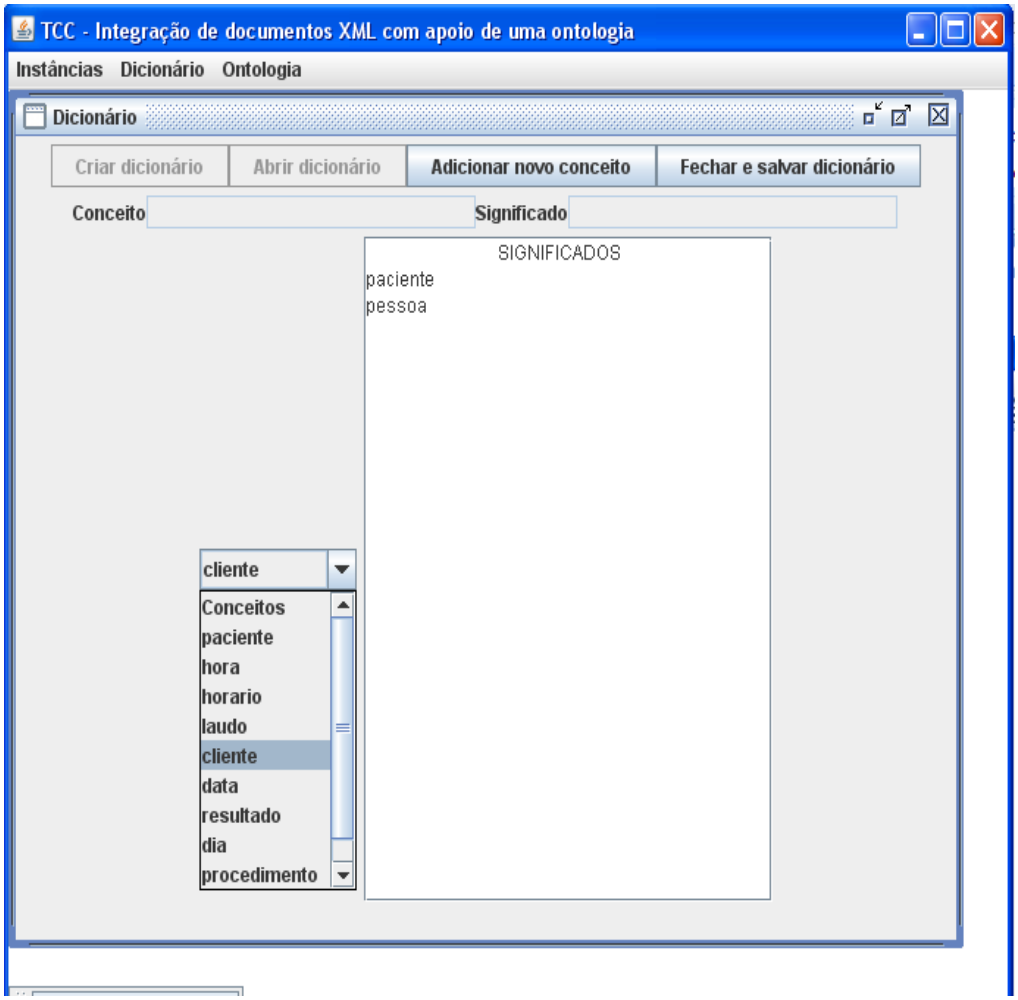

Figura 16: Criação do dicionário semântico

A árvore padrão da ontologia utilizada no processo de integração é mostrada na Figura 17. Esta árvore foi gerada selecionando o arquivo *clínica1.daml* da Figura 14. A ferramenta obtém a ontologia (arquivo selecionado) e cria a árvore pelo relacionamento entre as classes e propriedades da ontologia. Aqui, o módulo *Wrapper da ontologia* é ativado.

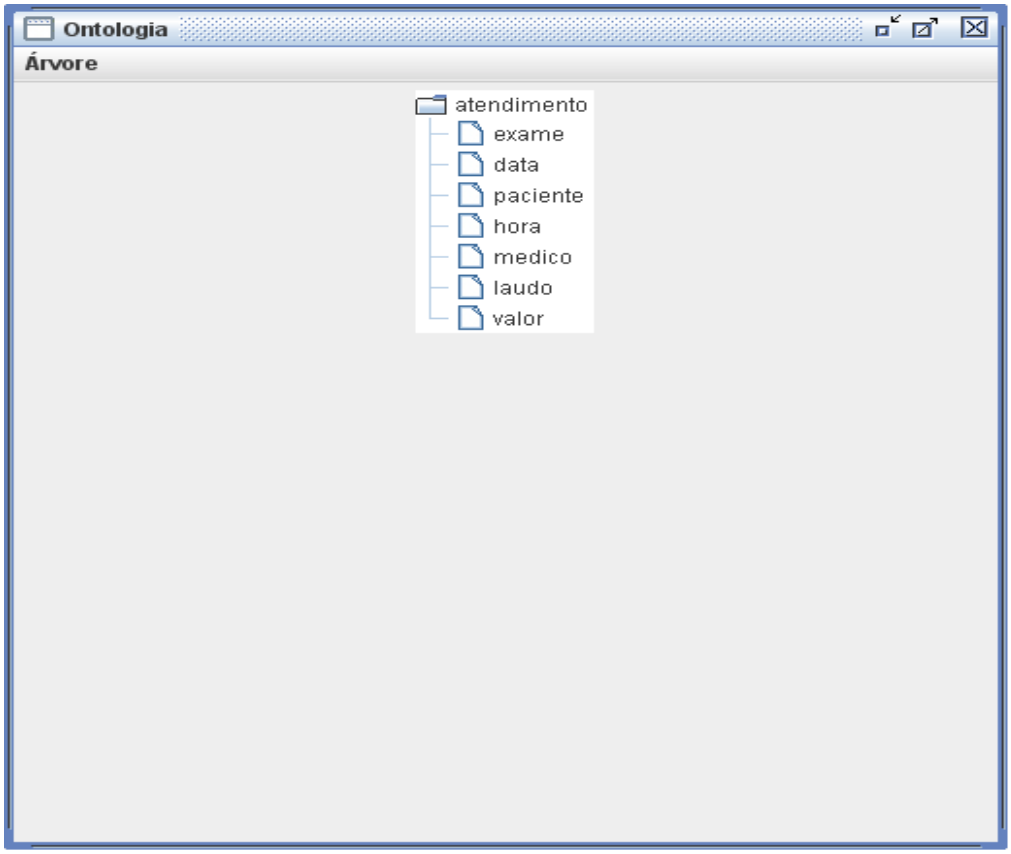

Figura 17: Árvore da ontologia

Com a definição das instâncias, do dicionário semântico e da árvore padrão da ontologia, a ferramenta é capaz de realizar a integração das instâncias. Para iniciar a integração, são acionados os menus *Instâncias >> Integrar*, como mostra a Figura 18.

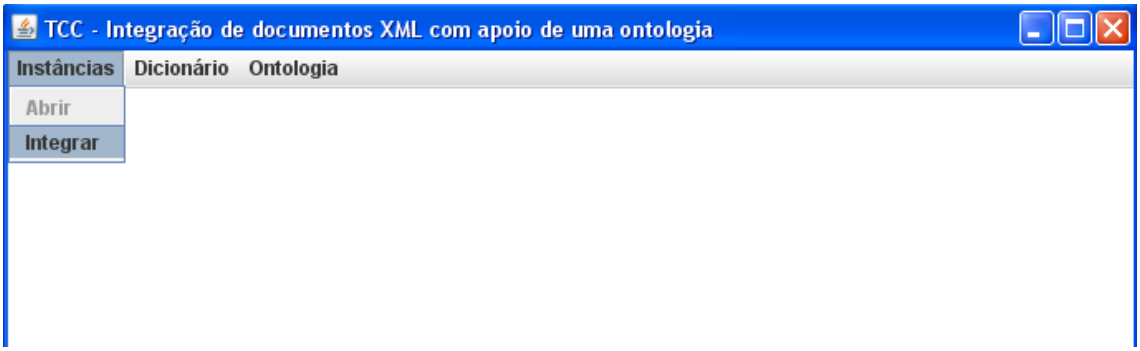

Figura 18: Menu para executar a integração de instâncias XML

O módulo *Transformador* recebe as instâncias ordenadas do módulo *Ordenador* para iniciar o pré-processamento (execução dos operadores unários) das mesmas. O módulo *Transformador*, ao receber a instância *clinica1.xml,* verifica se tal instância deve ter algum nome de *tag* modificado. Ao analisar os nomes das *tags* da instância com a árvore padrão, o módulo não faz alteração alguma, pois todos os nomes de *tags* na instância estão de acordo com a nomenclatura da ontologia. Assim, o operador *Rename* não é aplicado.

O próximo passo é verificar o comportamento estrutural da instância em relação à árvore. Todos os elementos da instância estão com o comportamento estrutural igual ao da árvore. Portanto, o operador *Move* não é aplicado.

O operador *Unnest* também não é aplicado nesta instância, pois não há elemento complexo nela que esteja representado na árvore como um nodo filho simples.

O operador *Remove* é aplicado na instância por causa do elemento *cidade* que não está representado na árvore da ontologia. Após a aplicação deste operador, a instância não contém mais o elemento cidade.

Finalizando o pré-processamento, o operador *Set* não é aplicado na primeira instância, pois os conteúdos dos elementos desta instância servirão de base para as demais. Depois de passar pelo módulo *Transformador,* a instância *clinica1.xml* não sofreu outra modificação além de ter o elemento *cidade* removido.

Na seqüência do processo, agora é a vez da instância *clinica5.xml* passar pelo módulo *Transformador.* Ao se verificar a Figura 13, que mostra as instâncias, e a Figura 17, que mostra a árvore da ontologia, apenas o operador *Unnest* é aplicado na instância. O operador é aplicado nos elementos *exame* e *medico,* pois, na instância, eles são elementos complexos e a árvore mostra que eles devem ser simples. O operador *Unnest* remove os elementos que são filhos do elemento que sofre a ação do operador. Portanto, os elementos *identificação* e *nome*, que não estão representados na árvore, não sofrem a aplicação do operador *Remove*, pois o *Unnest* já tratou desses elementos. Os elementos de *clinica5* que estão em *clinica1* têm os conteúdos exatamente iguais, não sendo necessária a aplicação do operador *Set.*

Com duas instâncias transformadas, o próximo passo é passar ambas para o módulo *Casador* que realiza a integração utilizando o operador *Merge*. O resultado da integração intermediária do processo é mostrado na Figura 19.

Agora, o ciclo da Figura 2, que contém os módulos *Transformador* e *Casador,* volta ao inicio. A próxima instância que irá ao módulo *Transformador* é *clinica3.xml.* Os operadores unários aplicados nesta instância são: *Rename, Move, Unnest e Remove.*

Resultado da integração *clinica1.xml* e *clinica5.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<atendimento>
   <paciente>Joao</paciente>
   <exame>coracao</exame>
   <medico>Paulo</medico>
   <data>22</data>
   <hora>14</hora>
   <valor>45</valor>
   <laudo>regular</laudo>
</atendimento>
```

Figura 19: Resultado da integração *clinica1.xml* e *clinica2.xml*

As *tags* que têm seus nomes alterados são aquelas em que os nomes são formados por palavras compostas. No caso desta instância, ocorrem palavras separadas por "_". Os elementos *exame_nome, exame_valor, agendamento_cliente, agendamento_medico, agendamento_data, agendamento_hora* têm os seus respectivos

nomes alterados para *nome, valor, paciente, medico, data* e *horario.* O processo verificou que os nomes dos elementos continham os nomes dos seus respectivos pais e então eliminou estas partes que continham os nomes dos pais. A Figura 20 mostra a instância após a aplicação do operador *Rename*. Observe que o elemento *agendamento_cliente* teve seu nome alterado para *paciente* ao invés de *cliente*, pois o dicionário semântico indica que estes nomes têm o mesmo significado e a árvore padrão contém o nome *paciente.*

Clinica3.xml após aplicação do operador *Rename*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<atendimento>
   <agendamento>
      <exame>
         <nome>coracao</nome>
         <valor>45</valor>
         <laudo>regular</laudo>
      </exame>
      <paciente>Joao</paciente>
      <medico>Paulo</medico>
      <data>22</data>
      <hora>14</hora>
   </agendamento>
</atendimento>
```

Figura 20: Resultado da operação *Rename* em *clinica3.xml*

O operador *Move* é aplicado nos elementos *exame, valor, laudo, cliente, medico, data e horario,* pois de acordo com a árvore padrão esses elementos devem ser filhos do elemento *atendimento.* A Figura 21 mostra a operação de *Move*.

Clinica3.xml após aplicação do operador *Move*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<atendimento>
   <exame>
      <nome>coracao</nome>
   </exame>
   <valor>45</valor>
   <laudo>regular</laudo>
   <cliente>Joao</cliente>
   <medico>Paulo</medico>
   <data>22</data>
   <hora>14</hora>
   <agendamento/>
</atendimento>
```

Figura 21: Resultado da operação *Move* em *clinica3.xml*

O operador *Unnest* é aplicado no elemento *exame,* eliminando o elemento *nome.* O conteúdo de *nome* torna-se conteúdo de *exame.* A instância contém o elemento *agendamento* que não está representado na árvore padrão. Portanto, o operador *Remove* é aplicado neste elemento. A Figura 22 mostra a instância após todas as transformações feitas nela.

Clinica3.xml após ser pré-processada

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<atendimento>
   <exame>coracao</exame>
   <valor>45</valor>
   <laudo>regular</laudo>
   <paciente>Joao</paciente>
   <medico>Paulo</medico>
   <data>22</data>
   <hora>14</hora>
</atendimento>
```

Figura 22: Resultado final do pré-processamento em *clinica3.xml*

Uma vez realizadas as transformações na instância *clinica3.xml*, ela é enviada para o módulo *Casador* com a instância gerada da integração anterior. A nova instância gerada é igual à instância da Figura 19, pois as instâncias sendo integradas são iguais.

Após a realização de mais uma integração, o módulo *Transformador* recebe a última instância do conjunto de instâncias: *clinica4.xml*. Os operadores *Rename*, *Move* e *Unnest* são aplicados nesta instância.

O operador *Rename* é aplicado no elemento *resultado,* pois o dicionário semântico verificou que este elemento está representado na árvore da ontologia como *laudo.* O operador *Move* é aplicado nos elementos *data, hora, paciente, medico, valor e laudo (*que antes era *resultado).* A Figura 23 mostra *clinica4.xml* após a execução do operador *Move.*

clinica4.xml após a execução do operador *Move*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<atendimento>
   <data>22</data>
   <hora>14</hora>
   <paciente>Joao</paciente>
   <medico>Paulo</medico>
   <valor>45</valor>
   <laudo>regular</laudo>
   <exame>
      <agendamento/>
      <nome>coracao</nome>
   </exame>
</atendimento>
```

Figura 23: *clinica4.xml* após a execução do operador *Move*

O operador *Unnest* é aplicado no elemento *exame.* Nesta operação, os elementos *agendamento* e *nome* são removidos, não sendo necessária a aplicação do operador *Remove.*

A instância da integração anterior é enviada para o módulo *Casador* com a instância *clinica4.xml* para a execução do operador *Merge.* Após a execução deste operador, o processo de integração do conjunto de instâncias da Figura 13 é finalizado,

mostrando ao usuário o resultado final, Figura 24. A instância tem o mesmo comportamento hierárquico da árvore padrão da ontologia da Figura 17.

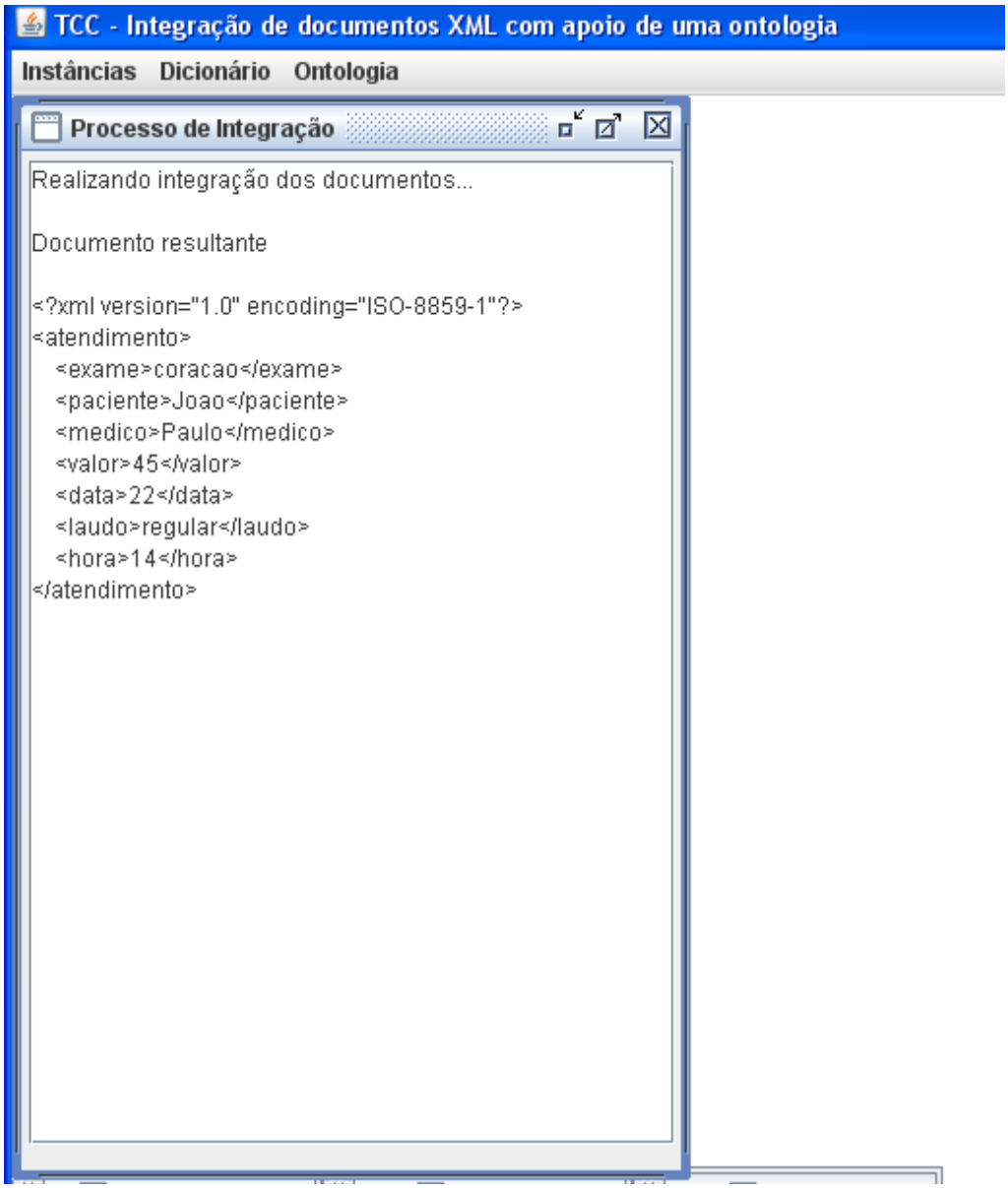

Figura 24: Finalização do processo de integração

O sistema proposto e desenvolvido requer a interação do usuário apenas para escolher as instâncias que irão ao processo de integração, criar ou abrir o dicionário e abrir a ontologia. O pré-processamento das instâncias, análises de dicionário e ontologia entre as instâncias e a integração são tarefas executadas pelo sistema sem a intervenção do usuário.

6 Considerações Finais

A *Web,* atualmente, é um importante meio para a troca de informações devido a facilidade de acesso e publicação. O formato padrão para a publicação de informações na *Web* que ganhou destaque foi o formato XML. Os dados de documentos XML são organizados de forma semi-estruturada, permitindo uma maior flexibilidade na definição da intenção semântica dos mesmos.

A imensa quantidade de informações disponíveis na *Web* causa redundância em resultados de pesquisas realizadas sobre ela. Para resolver este problema, há a necessidade da criação de sistemas que realizam a integração de dados. Esta integração eliminaria a redundância de dados e os resultados das pesquisas viriam de forma unificada.

Este trabalho propôs o desenvolvimento de uma ferramenta para realizar a integração de instâncias XML similares. Para dar apoio e melhorar o processo de integração, foi criado um conjunto de operadores que reduzem as diferenças estruturais entre as instâncias e faz a integração das mesmas. Os operadores estão divididos em duas categorias: unários e binários. Os operadores unários são responsáveis por compatibilizar, de forma automatizada, as instâncias, e o binário é responsável por unificá-las.

Uma ontologia de domínio foi utilizada neste trabalho para tornar a compatibilização das instâncias mais consistentes. A ontologia fornece uma padronização da estrutura hierárquica e nomenclatura das instâncias para o processo de compatibilização. Este trabalho possibilita a construção de um dicionário semântico por um usuário que é guardado em forma de arquivo. Ele busca semelhança entre os termos afins que são escritos de formas distintas. Este dicionário, junto com a ontologia, proporcionou uma melhora expressiva ao processo de integração. Outras estruturas externas podem ser utilizadas neste projeto para melhorar a qualidade da integração dos dados, como por exemplo, *thesauri*.

Melhorias nos pré-processamentos existentes são objetos de estudos futuros. Um exemplo dessa melhoria seria a inclusão de mais um operador no conjunto de operadores unários, definição de um *Nest,* por exemplo. Outro objeto de estudo futuro seria a construção de uma interface *Web* para o acesso a essa ferramenta.

Referências Bibliográficas

[CAR 03] CARVALHO, J.; SILVA, A. da. Finding similar identities among objects from multiple web sources. Proceedings of the 5th ACM international workshop on Web information and data management, ACM Press New York, NY, USA, p. 90–93, 2003.

[CAR 07] CARNIEL F. L.; MELLO, R. S. Um Processo de Integração de Instâncias XML apoiado por uma Ontologia de Domínio. In: WTDBD - XXII Simpósio Brasileiro de Banco de Dados (SBBD), ACM Press New York, NY, USA, 2007.

[DAR 07] THE DARPA Agent Markup Language Homepage. Disponível em: <http://www.daml.org/>. Acesso em dezembro de 2007.

[FON 02] FONTAINE, R. L. Merging XML files: a new approach providing intelligent merge of XML data sets. Proceedings of XML Europe, 2002.

[GBD 08] GBD – Grupo de Banco de Dados Disponível em: <http://www.grupobd.inf.ufsc.br/>. Acesso em abril de 2008.

[GUH 02] GUHA, S. et al. Approximate XML joins. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, ACM Press New York, NY, USA, p. 287–298, 2002.

[JEN 08] JENA A Semantic Web Framework for Java. Disponível em: <http://jena.sourceforge.net/>. Acesso em maio de 2008.

[JUN 07] WELCOME to The New JUnit.org! Disponível em: <http://www.junit.org/>. Acesso em dezembro de 2007.

[LEV 08] LEVENSHTEIN Distance, in Three Flavors. Disponível em: <http://www.merriampark.com/ld.htm>. Acesso em maio de 2008.

[LIN 04] LINDHOLM, T. A three-way merge for XML documents. Proceedings of the 2004 ACM symposium on Document engineering, ACM Press New York, NY, USA, p. 1–10, 2004.

[LIU 00] LIU, M.; LING, T. A data model for semistructured data with partial and inconsistent information. Proceedings of the International Conference on Advances in Database Technology (EDBT 2000), Springer, p. 317–331.

[MAY 01 a] MAY, W. Integration of XML Data in XPathLog. In: CAiSE Workshop on Data Integration over the Web (DIWeb'01), 2001, p. 2–16.

[MAY 01 b] MAY, W. A Framework for Generic Integration of XML Sources. In: 8th International Workshop on Knowledge Representation meets Databases (KRDB), 2001.

[OIL 07] WELCOME to OIL. Disponível em: <http://www.ontoknowledge.org/oil/>. Acesso em dezembro de 2007.

[OIL 08] Announcement: OilEd Ontology Editor. Disponível em: <http://xml.coverpages.org/oilEdANn20001204.html>. Acesso em fevereiro de 2008.

[ONT 07 a] Autor. Ontologia. Disponível em: <http://www2.dbd.pucrio.br/pergamum/tees abertas/0024134o2cap04.pd f >. Acesso em dezembro de 2007.

[ONT 07 b] Autor. ONTOLOGIA - Definições. Disponível em: <http://www.eci.ufmg.br/mba/p12.html>. Acesso em dezembro de 2007.

[OWL 07] OWL Web Ontology Language. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em dezembro de 2007.

[RES 07] RESOURCE Description Framework (RDF). Disponível em: <http://www.w3.org/RDF/>. Acesso em dezembro de 2007.

[SAM 08] SAM'S String Metrics. Disponível em: <http://www.dcs.shef.ac.uk/ sam/stringmetrics.html>. Acesso em maio de 2008.

[SUN 08] Sun MicroSystems. Disponível em: <http://www.sun.com/>. Acesso em maio de 2008.

[W3C 08] W3C. Disponível em: <http://www.grupobd.inf.ufsc.br/>. Acesso em abril de 2008.

[XML 07] W3C. Extensible Markup Language (XML). 2007. Disponível em: <http://www.w3.org/XML/>.

[XPA 08] XML Path Language (XPath). Disponível em: <www.w3.org/TR/xpath>. Acesso em maio de 2008.