

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CIÊNCIAS DA COMPUTAÇÃO

Fábio dos Santos

**XMLtoOWL: Uma Ferramenta para a Geração de uma
Ontologia a partir de Documentos XMLs**

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

MSc. Rodrigo Gonçalves

Prof. Dr. Ronaldo dos Santos Mello

Florianópolis, Agosto de 2008

XMLtoOWL: Uma Ferramenta para a Geração de uma Ontologia a partir de Documentos XMLs

Fábio dos Santos

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, e aprovado em sua forma final pelo Programa de Graduação em Ciência da Computação.

Prof. Dr. Luis Fernando Friedrich

Banca Examinadora

MSc. Rodrigo Gonçalves

Prof. Dr. Ronaldo dos Santos Mello

Prof. Dr. Mauro Roisenberg

Prof. Dr. Renato Fileto

Ofereço este trabalho aos meus pais, sem eles nada disto
seria possível...

Agradecimentos

Primeiramente gostaria de agradecer a Deus, pelas condições favoráveis à realização deste trabalho. Agradeço também aos meus pais e minha irmã, Eng. João Julio dos Santos, Teresa Cristina dos Santos e Juliana dos Santos pelo apoio, amor, atenção, companheirismo e tudo que uma verdadeira família pode proporcionar.

Aos meus orientadores MSc. Rodrigo Gonçalves e Prof. Dr. Ronaldo dos Santos Mello, pela atenção, revisões, e “horas perdidas”...

Ao meu primo pelos incentivos... e por ter ido embora lá de casa! Brincadeira, agradeço por ser um irmão que eu não tive.

A minha namorada Fernanda pela compreensão e carinho; e a todos os meus amigos, principalmente ao Vinícius dos Santos, vulgo Bozo, e José Raupp, pela ajuda em todos os sentidos na produção deste trabalho.

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CIÊNCIAS DA COMPUTAÇÃO

Fábio dos Santos

**XMLtoOWL: Uma Ferramenta para a Geração de uma
Ontologia a partir de Documentos XMLs**

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

MSc. Rodrigo Gonçalves

Prof. Dr. Ronaldo dos Santos Mello

Florianópolis, Agosto de 2008

Sumário

Sumário	ii
Lista de Tabelas	iv
Lista de Figuras	v
Resumo	vi
Abstract	vii
1 Introdução	1
2 XML	4
2.1 Esquemas	5
2.1.1 DTD	5
2.1.2 XML-SCHEMA	6
3 Ontologia	9
3.1 OWL	10
3.1.1 Elementos Básicos de OWL	11
4 Trabalhos Relacionados	14
4.1 XTRACT	14
4.2 XML2OWL	15
4.3 Mapping XML to Existing OWL ontologies	16
5 XMLtoOWL	18
5.1 Obtenção do XML Schema	20
5.1.1 GUI - Graphical User Interface	20

5.1.2	Lógica	20
5.1.3	Persistência	23
5.2	Obtenção da Ontologia	24
6	Estudo de Caso	27
6.1	Execução	27
7	Conclusão	30
	Referências Bibliográficas	32

Lista de Tabelas

2.1	Tabela de Correspondência entre Estruturas XML e Definições de Conceitos XSD.	8
5.1	Tabela de tratamento dos elementos XSD	26
6.1	Primeiro Documento XML	27
6.2	Segundo Documento XML	29

Lista de Figuras

2.1	Exemplo de Documento XML	5
2.2	Exemplo de DTD	6
2.3	Exemplo de XSD	7
3.1	Linguagens para Web Semântica	10
3.2	Declaração de namespace em OWL	11
3.3	Cabeçalho de uma ontologia	12
3.4	Classes de uma ontologia	12
3.5	Indivíduos de uma ontologia	12
3.6	Propriedades de uma ontologia	13
3.7	Propriedades do tipo <code>datatype</code>	13
5.1	Estrutura da Ferramenta para Geração de uma Ontologia - XMLToOWL	19
5.2	Exemplo de uma Árvore Sumarizada de um Documento XML	22
5.3	Diagrama de Classes da Persistência	24
5.4	Mudança na Classe principal para receber o esquema gerado na etapa anterior	25
6.1	Algumas regras ORM definidas no processo	28

Resumo

Este trabalho apresenta uma ferramenta capaz de construir uma ontologia a partir das informações contidas em documentos XML, com o objetivo de melhorar sistemas de integração de dados existentes, pelo fornecimento de um suporte semântico aos mesmos. O processo adotado pela ferramenta divide-se em dois estágios: o primeiro extrai um *XML Schema* de documentos XML, que é processado pelo segundo estágio e gera, através do uso de regras de conversão ORM, uma ontologia descrita no formato OWL.

Abstract

This work introduces a tool capable of building an ontology from information contained in XML documents, in order to improve the quality of existing data integration systems, by providing semantic support to them. This tool processing is composed by two main stages: the first extracts an XML schema from the XML documents, which is then processed by the second stage, generating, through the use of ORM rules, an ontology described in OWL format.

Capítulo 1

Introdução

A *Web* funciona como um meio para disponibilização de informações de fontes de dados de vários domínios e a XML[ELM 05]¹ tem sido utilizada para estruturar e publicar os dados. Sua adoção deve-se a sua natureza simples, porém flexível e extensível, de organizar informações. Esta flexibilidade permite que fontes de dados XML na *Web* possam ter sua intenção semântica descrita de diversas maneiras através das *tags* definidas por seus autores. Tal fato pode gerar problemas ao comparar dados de duas ou mais fontes durante, por exemplo, consultas sobre elas.

Sistemas de integração de dados provém ao usuário uma visão integrada de duas ou mais fontes de dados. Eles surgiram para viabilizar a consulta a fontes de dados heterogêneas e geralmente disponibilizam, para os usuários, uma visão única e uniforme de um grande número de fontes de dados independentes, dentro de um domínio comum. O objetivo maior destes sistemas é liberar o usuário de ter que localizar as fontes de dados, interagir com cada uma isoladamente e combinar os dados oriundos delas manualmente Hankimpour et.al[HAK 01]. Tais sistemas, porém, têm limitações na qualidade semântica dos seus processos, necessitando do auxílio de usuários especialistas para realizar as suas tarefas. A definição de equivalências semânticas entre os dados das fontes é um exemplo.

Sistemas de integração de dados podem contar com o suporte de ontologias [BRE 05]. Uma ontologia pode dar apoio semântico à estes sistemas de integração Bohring et.Al[BOH 02], provendo uma visão conceitual integrada das informações, mantendo equivalências semânticas para os dados das mesmas.

Uma ontologia pode ser definida como um “catálogo de tipos de coisas”, as quais se assumem existir em um domínio de interesse na perspectiva de uma ou mais pessoas. Com o auxílio de uma

¹eXtensible Markup Language

ontologia, pode-se elaborar consultas mais complexas através da disponibilidade de informações sobre o contexto dos dados. Além disso, uma ontologia fornece uma representação comum para a troca de informações e facilita suporte ao processamento automático de informações fornecendo um padrão a ser adotado pelo processamento [ELM 05]. Diversas linguagens foram desenvolvidas para especificar ontologias, dentre as quais a *OWL Web Ontology Language*, é a recomendação mais recente da W3C².

Este trabalho apresenta uma ferramenta capaz de, a partir de um conjunto de documentos XML, estabelecer uma ontologia contendo informações presentes nestes documentos. Os documentos não precisam ter esquemas associados, o que ocorre na *Web*. O processo dá-se em duas etapas distintas: na primeira, extrai-se o esquema dos documentos XML de uma certa fonte. Na segunda, o esquema extraído é convertido em uma ontologia [GRU 92] especificada em OWL. Tal proposta originou a ferramenta apresentada neste trabalho. Com o auxílio de outra ferramenta também desenvolvida no GBD UFSC³ Garcia et al. [LGG 06]. A ferramenta é dividida em dois estágios, sendo a entrada do primeiro, documentos XML e a saída um XML Schema e a entrada do segundo o esquema gerado na etapa anterior e como saída uma ontologia persistida em formato OWL.

A ferramenta foi concebida no âmbito do projeto Digitex (Processo Institucional CNPq / CTInfo: 550.845/2005-4). O projeto trata da construção de uma Plataforma de Indexação e Busca Personalizada em Bibliotecas Digitais (BibDig), permitindo a consulta sobre bibliotecas digitais de forma centralizada e homogênea. Em várias bibliotecas referentes a um mesmo domínio, como a CiteSeer⁴ e a DBLP⁵ *domínio bibliográfico acadêmico*, a linguagem XML é adotada para a organização e disponibilização das informações. Esta disponibilização, porém, não segue um padrão definido quanto à estrutura dos dados. Autores de publicações, por exemplo, podem ter o dado *nome* representado através de um campo único ou dividido em nome e sobrenome.

Um dos aspectos que o projeto Digitex visa tratar é a análise e compatibilização na forma como BibDigs em um mesmo domínio disponibilizam suas informações. Para tanto, o Grupo de Banco de Dados da UFSC (GBD/UFSC)⁶ têm trabalhado a temática de compatibilização, comparação e integração de documentos XML. Para o processo de compatibilização, diversas heurísticas foram elaboradas como parte de uma dissertação de Mestrado, da qual parte dos resultados foi

²World Wide web Consortium

³Grupo de Banco de Dados

⁴<http://citeseer.ist.psu.edu/>

⁵<http://www.informatik.uni-trier.de/~ley/db/>

⁶<http://www.grupobd.inf.ufsc.br/>

apresentada em dois artigos [GON 07, SAN 07]. Estas heurísticas, porém, sofrem a limitação da necessidade de uma estrutura semântica de suporte. Um exemplo de estrutura capaz de prover este suporte seria uma ontologia.

A construção de uma ontologia demanda tempo e a presença de especialistas no domínio. Deve-se prever todos os possíveis dados que podem estar contidos na mesma, algo difícil quando não se tem um conhecimento prévio dos dados a serem armazenados. Tal realidade é presente no contexto do projeto Digitex. Se for possível automatizar parte do processo de criação da ontologia, têm-se um ganho de tempo no processo, visto que o usuário não precisa definir toda a ontologia. A partir deste problema, surgiu a proposta de elaborar um sistema que, a partir de documentos XML que representam dados de BibDigs, elabore uma ontologia capaz de conter as instâncias de dados presentes nos mesmos.

Este trabalho organiza-se da seguinte maneira: o capítulo 2 apresenta uma introdução sobre a linguagem XML e os Schemas oriundos desta. Em seguida, no capítulo 3, levanta-se o estado da arte das tecnologias empregadas na construção de ontologias, concluindo com um estudo sobre OWL. O capítulo 4 apresenta os trabalhos relacionados que contribuíram para a construção da ferramenta proposta. No capítulo 5 é apresentada a ferramenta e seu funcionamento, seguido pelo capítulo 6 que apresenta um caso de uso da ferramenta. A trabalho encerra-se com o capítulo 7 que trata das observações finais deste trabalho e possíveis trabalhos futuros para a melhoria da ferramenta elaborada.

Capítulo 2

XML

XML é uma linguagem baseada em *tags* que advém da SGML ¹. Sua escolha deve-se a insatisfação com os formatos existentes até então, como por exemplo, a SGML, que é uma linguagem muito complexa, e a HTML, que é uma linguagem adequada à formatação da apresentação de dados na *Web*, mas não tem tanta representatividade semântica como SGML [MAU 02].

As *tags* que estão presentes na XML são na verdade elementos escritos pelo usuário que está escrevendo nesta linguagem, com isto a XML apresenta a característica de não possuir elementos pré-definidos, sendo assim o usuário é quem define seu próprio conjunto de elementos que melhor se aplica no contexto para representação de seus dados.[MAU 02].

Um documento XML mantém um conjunto de dados organizados em uma hierarquia de elementos definidas pelo seu autor, como por exemplo `<raiz>texto</raiz>`. Todo documento XML deve ser bem formado, ou seja, atender as seguintes regras definidas pela W3C [CON]:

- O documento deve iniciar com a declaração XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

- Todas os elementos têm que ter um começo, `<elemento>`, que representa o início do elemento, e um fim,

```
</elemento>
```

 que representa o fechamento do elemento;

- Como XML é representado em um formato de árvore, há a necessidade de um elemento raiz.

Exemplo: `<Raiz> <de>...</de>< /Raiz>`.

- O documento pode conter atributos dentro de seus elementos.

¹Standard Generalized Markup Language

Exemplo: `<exemplo atributo="exemplo"/>`

A Figura 2.1 representa um exemplo de um documento XML.

<code><?xml version="1.0" encoding="UTF-8t"?></code>	Declaração XML
<code><Biblioteca></code>	Raiz do documento
<code><Livro></code>	Abertura do elemento
<code><Título>ENTREVISTA COM O VAMPIRO</Título></code>	
<code><Autor>Anne Rice</Autor></code>	
<code></Livro></code>	
<code></Biblioteca></code>	Fechamento do elemento

Figura 2.1: Exemplo de Documento XML

Como foi comentado anteriormente, a estruturação dos elementos presente nos documentos XML é feita pelo usuário, não apresentando um padrão. Com isto, dois ou mais documentos XML são constituídos com o intuito de descrever informações sobre um mesmo contexto, não necessariamente possuem uma mesma estrutura, visto que os elementos não são pré-definidos pela linguagem XML. Sendo assim, quando há necessidade de um padrão para representar um documento XML, define-se um esquema.

2.1 Esquemas

Um esquema, em dados XML define restrições para a organização hierárquica e para o conteúdo dos dados. Seu uso favorece uma descrição do próprio documento, bem como auxilia na troca de informações entre diferentes tipos de dados, visto que a partir de esquemas também é possível fazer uma verificação de equivalência de dados. A W3C sugere para a definição de esquemas XML: XML Schema. Porém, anteriormente ao XML Schema o DTD² era o padrão adotado pela W3C, sendo assim nas próximas seções são abordados estes dois modelos para esquemas XML.

2.1.1 DTD

Um esquema DTD é um conjunto de regras que define quais os elementos que podem ser usados em um documento XML e quais os valores são válidos para o conteúdo dos mesmos. Com ele define-se uma gramática para documentos XML. Foi a primeira recomendação da W3C para esquemas em documentos XML. Um exemplo de DTD está representado na Figura 2.2.

²(ocument Type Definition

<pre><?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE name [<ELEMENT nome (titulo*, primeiro nome inicial, nome do meio?, ultimo nome+)>] <!DOCTYPE primeiro nome [<ELEMENT primeiro nome #PCDATA1>]</pre>	<p>Declaração XML</p> <p>Define que a raiz do documento será name</p> <p>Define que o elemento nome poderá ter quatro elementos</p> <p>Define que o elemento primeiro nome vai apresentar primeiro PCDATA que significa que os conteúdos entre as tags serão analisados por um processador deste documento</p>
<p>Uso do DTD acima</p> <pre><?xml version="1.0" ?> <name> <nome> <titulo>Ms</titulo> <primeiro nome>Anne</primeiro nome> <ultimo nome>Rice</ultimo nome> </nome> </name></pre>	<p>Declaração XML</p> <p>Raiz do documento</p> <p>Fechamento do elemento</p>

Figura 2.2: Exemplo de DTD

2.1.2 XML-SCHEMA

O XML Schema é uma linguagem baseada em XML, que define regras de validação para documentos XML. Pelo fato do XML-Schema ser descrito em XML (diferente do DTD, que segue outro tipo de formatação) é o formato recomendado atualmente pela W3C. Um exemplo de XML Schema é representado na Figura 2.3.

XML-Schema é uma extensão do DTD, ou seja ele incorpora todas as funcionalidades do DTD e provém outras:

- Definição e especialização dos tipos de elementos;
 - semelhança com esquemas orientados a objetos;
- Definição de tipos de dados;
 - simples (*string*, *integer*, *boolean*, ...);
 - complexos (*list*, *union*);
- Facilidades adicionais para definição de restrições;
 - intervalos de valores permitidos e padrões de conteúdo via expressões regulares.

Alguns elementos presentes no XSD e que são tratados pela ferramenta proposta podem ser conferidos na tabela 2.1.

<pre> <?xml version="1.0"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com" elementFormDefault="qualified"> <xs:element name="note"> <xs:complexType> <xs:sequence> <xs:element name="to" type="xs:string"/> <xs:element name="from" type="xs:string"/> <xs:element name="heading" type="xs:string"/> <xs:element name="body" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element> </xs:schema> </pre>	<p>Declaração XML</p> <p>Exemplo de uma declaração de um XSD</p> <p>Tipo do Elemento</p> <p>Sequencia de Filhos do elemento</p>
<p>Uso do XSD acima</p> <pre> <?xml version="1.0"> <note> <to>Tove</to> <from>Jani</from> <heading>Reminder</heading> <body>Don't forget me this weekend!</body> </note> </pre>	<p>Declaração XML</p> <p>Raiz do documento</p>

Figura 2.3: Exemplo de XSD

Definições	Descrição	XML	Estrutura XSD de saída
SimpleType	Especifica um elemento simples	<x>bozo</x>	<element name = ``x`` type ``string`` />
Atribute	Especifica um atributo	<x>bozo</x> <x atrib: ``a``> bozo</x>	<element name = ``x`` type ``string``> <complexType> <attribute name=``a`` type=``string``> </complexType> </element>
Simple-Content	Especifica um elemento simples que contém um atributo e precisa ser declarado como elemento complexo	<x atrib:``a``> bozo</x>	<element name = ``x`` type ``string``> <complexType> <attribute name=``a`` type=``string``> </complexType>\n</elemente>
Empty	Especifica um elemento vazio sem conteúdo que pode conter ou não um atributo	<x/>	<element name=``x``> <complexType> </complexType> </elemente>
Sequence	Indicador de ordem de um elemento complexo	<x> <a>bozo </x>	<element name=``x``> <complexType> <element name=``a`` /> </complexType> </element>
Choice	Indicador de um tipo complexo, definindo que deve ser escolhido somente um dos elementos definidos dentro do choice	<x> <a>bozo </x>	<element name=``x``> <complexType> <choice> <element name=``a`` type``string`` /> <element name=``b`` type``string`` /> </choice> </complexType> </element>
All	Indicador de um tipo complexo, definindo os elementos filhos podem ocorrer em qualquer ordem mas devem aparecer somente uma vez	<x> <a>palhaço bozo </x>	<element name=``x``> <complexType> <all> <element name=``a`` type``string`` /> <element name=``b`` type``string`` /> </all> </complexType> </element>
Mixed	Indicador de um tipo complexo, que apresenta filhos e textos misturados	<x> exemplo <a>palhaço bozo </x>	<element name=``x``> <complexType mixed=``true``> <all> <element name=``a`` type``string`` /> <element name=``b`` type``string`` /> </all> </complexType> </element>
Anytype	Indicador de um tipo de , elemento que não possui uma estrutura definida podendo aparecer complexo ou simples	<x> bozo </x> <x>bozo</x>	<element name=``x`` type=``anytype``> <complexType> <element name=``a`` type``string`` /> </complexType> </element>

Tabela 2.1: Tabela de Correspondência entre Estruturas XML e Definições de Conceitos XSD.

Capítulo 3

Ontologia

Ontologia, palavra que vem do grego *ontos* e *logos*, significa “conhecimento do ser”. Na filosofia, consiste na ciência que estuda as estruturas dos objetos e suas propriedades, os eventos, processos e relacionamentos entre eles em todas as áreas da realidade, auxiliando em sistemas de categorias e identidades para organizar a realidade [BRE 05].

O consórcio W3C define uma ontologia como “*a definição dos termos utilizados na descrição e na representação de uma área do conhecimento*” [BRE 05]. Exemplo de elementos contidos em uma ontologia são:

- Classes - conjuntos, coleções ou tipos de objetos. Exemplo: Classe Carro;
- Atributos - propriedades, características ou parâmetros que os objetos podem ter e compartilhar. Exemplo: Classe carro possui 4 rodas;
- Relacionamentos - as formas como os objetos podem se relacionar com outros objetos. Exemplo: Classe carro possui quatro rodas e é da marca Ford.

Nas últimas décadas surgiram várias linguagens para descrever ontologias, como OWL, DAML¹, OIL+DAML², etc, especialmente devido à popularização da *Web* e a publicação na mesma de informações. Estas junto com a XML auxilia a semântica validando e estruturando as informações, um modelo de camadas para a arquitetura da *Web* do futuro está sendo desenvolvido [BRE 05]. Este modelo foi proposto para aproveitar o que já está pronto, que no caso seria toda a parte de HTML e de programação, faltando a parte de semântica. Esta abordagem é proposta porque seria custoso remodelar todos os dados existentes na *Web*, sendo assim foi proposto um

¹DARPA Agent Markup Language

²Ontology Inference Layer

modelo que acrescenta à *Web* existente novas camadas sobre as estruturas já existentes, como mostra a Figura 3.1.

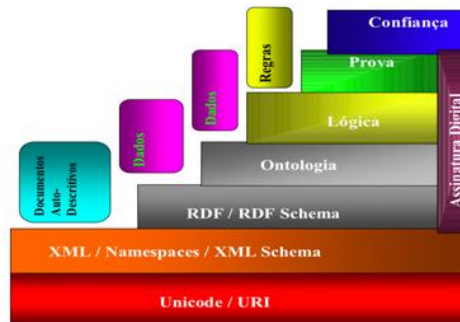


Figura 3.1: Linguagens para Web Semântica

A primeira camada é a HTML, que por ser uma linguagem simples e não dar suporte a semântica, necessita a adoção de linguagens mais sofisticadas, como o caso XML. Na segunda camada têm-se o RDF³ que tem o papel de fornecer metadados e facilitar a troca de informações na *Web*. Conta-se também com o rdfschema, que foi criado para suplementar o RDF, adicionando funcionalidades como a construção de hierarquias, classes, propriedades, etc. A combinação do RDF + *rdfschema* forma o *RDFSResource Description Framework Schema*.

Sobre o RDFS são propostas algumas linguagens de especificação para ontologia: OIL(Ontology Inference Layer), DAML(DARPA(Defense Advanced Research Project Agency) Agent Markup Language), DAML + OIL e OWL(Ontology Web Language).

3.1 OWL

O consórcio W3C fez uma revisão na linguagem DAML+OIL atualizando-a e adicionando novas funções para atender as necessidades das aplicações atuais da *web*, estabelecendo assim a linguagem OWL.

OWL vem com o objetivo de representar conceitos e seus relacionamentos na forma de uma ontologia. Ela possui três variantes, que são classificadas em ordem crescente de expressividade:

- OWL Lite - Oferece suporte à criação de hierarquias simplificadas de classificação e suas restrições, suportando também cardinalidade. Contudo, não fornece suporte às ontologias que necessitam de axiomas e estruturas de relacionamentos sofisticados.

³Resource Description Framework

- OWL DL⁴ - fornece subsídios para mapeamento de linguagens deste tipo de lógica, além de incluir todas as construções da linguagem OWL Lite. Porém estas construções têm restrições (por exemplo: embora uma classe possa ser subclasse de muitas classes, uma classe não pode ser instância de outra classe).
- OWL Full - É direcionada para aqueles usuários que almejam a máxima expressividade e a liberdade sintática do RDF sem nenhuma garantia computacional. OWL Full permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL.

3.1.1 Elementos Básicos de OWL

- *Namespaces* - Representa os vocabulários que estão sendo utilizados na presente ontologia. Os *namespaces* são declarações que formam os identificadores que estarão presentes no documento OWL. A função dos namespaces é ajudar os documentos a serem interpretados sem que haja ambigüidades. Uma ontologia típica em OWL começa com um conjunto de declarações de *namespace*, conforme a Figura 3.2.

<pre><rdf:RDF xmlns = "http://www.mindswap.org/2003/owl/swint/terrorism#" xmlns:base = "http://www.w3c.org/TR/2003/owl/swint/terrorism#" xmlns:terr = "http://www.w3c.org/TR/2003/owl/swint/terrorism#" xmlns:owl = "http://www.w3.org/2002/07/owl#" xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd = "http://www.w3.org/2001/XMLSchema#"></pre>	<p>Nas primeiras linhas consta o namespace da ontologia.</p> <p>Identifica-se um URI base para o documento</p> <p>Identifica-se um namespace para o vocabulário OWL</p> <p>Aqui são definidos namespaces para RDF e XSD</p>
--	---

Figura 3.2: Declaração de namespace em OWL

- Cabeçalhos - Uma vez definidos os *namespaces*, é comum incluir uma coleção de sentenças sobre a ontologia agrupados sob a tag `owl:Ontology`. Pode-se colocar comentários sobre a ontologia, bem como data e a sua versão exemplificado na Figura 3.3.
- Classes - Representam um conjunto ou uma coleção de indivíduos (objetos, pessoas), que estão agrupados segundo característica semelhantes. Utiliza-se esta abstração para descrever conceitos de um determinado domínio, como por exemplo: carros, animais, matérias. Em OWL as classes são representações de conceitos básicos de um domínio, formando os elementos primordiais utilizados na criação de várias taxionomias. Um exemplo de classe em OWL pode ser conferido na tabela 3.4.

⁴(Lógica de Descrição)

<pre><owl:Ontology rdf:about=""> <owl:versionInfo> 01/08/2008 </owl:versionInfo> <owl:priorVersion rdf:resource="http://www.w3c.org/TR/2003/owl/swint/terrorism#" /> <rdfs:comment>This file specify a OWL ontology</rdfs:comment> <owl:imports rdf:resource="http://www.mindswap.org/2003 /owl/swint/person#" /> <rdfs:label> Terrorism Ontology </rdfs:label> ... </owl:Ontology></pre>	<p>Neste segmento estão presentes comentários e a versão da ontologia.</p>
---	--

Figura 3.3: Cabeçalho de uma ontologia

<pre><owl:Class rdf:ID="Automoveis"/> <owl:Class rdf:ID="Lanchas"/> <owl:Class rdf:ID="Tratores"/></pre>	<p>Representa a a existencia destas Classes</p>
--	---

Figura 3.4: Classes de uma ontologia

- **Indivíduos** - São objetos do mundo: pertencem a classes e são relacionados a outros indivíduos (e classes) por propriedades como é mostrado na tabela 3.5.

<pre><owl:Class rdf:ID="Cachorro"> <owl:subClassOf rdf:resurce="#Animal"/> </owl:Class> <owl:Class rdf:ID="PitBull"/></pre>	<p>Demonstra o exemplo da Classe cachorro que é subclasse de animal e pitbull é uma instância</p>
---	---

Figura 3.5: Indivíduos de uma ontologia

- **Propriedades** - Descrevem características de indivíduos ou de uma classe toda, apresentado na tabela 3.6. Em OWL, as propriedades se dividem em duas:
 - Tipo object - Relacionamentos entre duas classes. Exemplificado na tabela 3.7;
 - Tipo datatype - Indicam um relacionamento entre instâncias de classes e literais expressos em RDF e datatypes do *XML Schema*. Exemplificado na Figura 3.7.

<pre><owl:ObjectProperty rdf:ID="Come ração"> <rdf:domain rdf:resource="#Cachorro"/> <rdf:range rdf:resource="#Ração"/> </owl:ObjectProperty></pre>	Exemplo que todo cachorro come ração
---	--------------------------------------

Figura 3.6: Propriedades de uma ontologia

<pre><owl:Class rdf:ID="SerVivo"> <owl:DatatypeProperty rdf:ID="anoDeNascimento"> <rdf:domain rdf:resource="#SerVivo"/> <rdf:range rdf:resource="&xsd:positiveInteger"/> </owl:DatatypeProperty> </owl:Class></pre>	Demonstra através de uma propriedade datatype que relaciona todos os seres vivos com um ano de nascimento
---	---

Figura 3.7: Propriedades do tipo datatype

Capítulo 4

Trabalhos Relacionados

Neste capítulo são abordados trabalhos relacionados referentes ao primeiro estágio da ferramenta, relacionado a extração de esquemas XML e também trabalhos relacionados com ao segundo estágio relacionado a geração de uma ontologia a partir de XML Schema gerado.

4.1 XTRACT

O principal objetivo da abordagem de Garofalakis et al [GAR 03] é a geração de um DTD perfeito, na visão dos autores, a partir de um documento XML. A ferramenta é dividida em três passos:

1. *Generalização*: No primeiro passo são aplicados algoritmos com cunho heurísticos na tentativa de encontrar padrões nas sequências de entradas dos documentos XML com o objetivo de trocar estes padrões por expressões regulares. Faz-se esta substituição porque expressões regulares apresentam-se mais generalista [GAR 03], como mostra o exemplo:

Uma sequência de entrada para este passo pode ser representada por *ababcababc*, este passo faz uma varredura e encontra que o símbolo *ab* é utilizado em dois campos e irá substituir este símbolo pela expressão $A1$. Assim o resultado fica $A1A1cA1A1c$, que no final do passo será substituído por $((ab) * c)^*$.

2. *Fatoração*: Este passo utiliza-se dos resultados obtidos no primeiro passo, que consiste em um conjunto de possíveis candidatos ao DTD. Este passo faz otimização nesses candidatos aplicando técnicas adaptadas da literatura de otimização lógica. Usam-se estes métodos de otimização na tentativa de se obter um DTD menor e conseqüentemente de melhor qualidade [GAR 03], como mostra o exemplo:

Uma sequência de entrada para este passo pode ser representada por $I = ab, abab, ac, ad, bc, bd, bbd, bbbbe$, este passo faz uma *generalização* e obtém como resultado $G = IU(ab)^*, (alb)^*, b * d, b * e$. No final do processo têm-se $F = GU(alb)(cld), b * (dle)$.

3. Aplicação do princípio da “Descrição do Tamanho Mínimo”: Que consiste em escolher o melhor candidato a DTD com base no seu custo. Este custo depende do tamanho de sua gramática e do tamanho dos seus dados, os dois sendo representados em bits, como mostra o exemplo:

Uma sequência de entrada para este passo pode ser representada por $I = ab, abab, ababab$ e o resultado deste passo representado por (1) $ablabablababab$ e (2) $(ab)^*$. O cálculo de (1) resultaria em 17 bits e o do (2) em 8 bits, neste caso, a DTD escolhida é representada por (2), porque este resultado conseguiu abranger toda a sequência de entrada e, ao mesmo tempo, conforma-se mais compacta.

O autor afirma que os resultados obtidos são satisfatórios, visto que consegue-se chegar a um DTD *perfeito*, ou quase-perfeito [GAR 03], porém este trabalho gera um DTD somente de um documento XML e o resultado do esquema gerado é o DTD, que nos dias atuais não é o padrão recomendado pela W3C para a representação de esquemas de documentos XML.

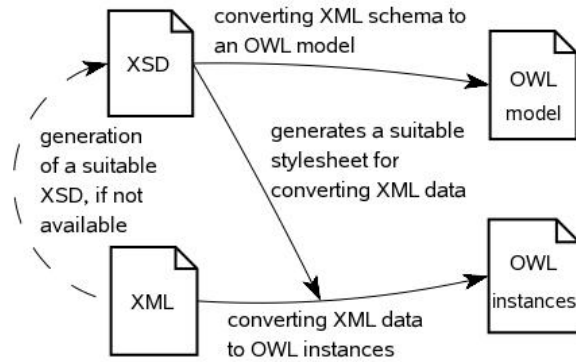
4.2 XML2OWL

Em Bohring et al. [BOH 02] é apresentada uma ferramenta para o mapeamento de um documento XML para um documento OWL. O mapeamento é feito através de um *framework* baseado em XSLT (XSL Transformations) que infere um XML Schema a partir do documento XML e, a partir deste, aplica uma transformação para obter um esquema OWL.

O processo pode ser verificado na Figura 4.1(a), aonde uma transformação XSLT é aplicado num documento XML, a partir desta transformação e das regras mostradas na Figura 4.1(b), a ferramenta gera um OWL.

Por exemplo, o documento XML representado pela Figura 4.1(c) por meio do XSLT implementado na ferramenta é transformado em um XML Schema, como mostra a Figura 4.1(d), a partir deste esquema mais as regras apresentadas na Figura 4.1(b) um OWL é produzido, podemos conferir a relação dos elementos presentes no XSD e transformados em OWL na Figura 4.1(e).

Bohring et al. [BOH 02] não trata alguns elementos do XML Schema, como *SimpleType* e o trabalho não permite a conversão de múltiplos documentos XML.



(a) Sequência de Operação da ferramenta XML2OWL

XSD	OWL
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
named xsd:complexType	owl:Class
named xsd:simpleType	owl:DatatypeProperties
xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sequence, xsd:all	owl:intersectionOf
xsd:choice	combination of owl:intersectionOf, owl:unionOf and owl:complementOf

(b) Mapeamento de um XML Schema para um OWL na ferramenta XML2OWL

```

<!-- ... -->
<record>
  <header>
    <identifier>oai:CiteSeerPSU:1</identifier>
  </header>
  <metadata>
    <oai_citeseer:oai_citeseer>
      <dc:title>A title</dc:title>
    </oai_citeseer:oai_citeseer>
  </metadata>
</record>
<!-- ... -->

<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:oai="http://copper.ist.psu.edu/oai/oai_citeseer/">
  <!-- ... -->
  <element name="metadata">
    <complexType>
      <sequence>
        <element ref="oai:oai_citeseer"
          maxOccurs="1" minOccurs="1"/>
      </sequence>
    </complexType>
  </element>
  <element ref="oai:oai_citeseer">
    <complexType>
      <sequence>
        <element ref="dc:title" maxOccurs="1"
          minOccurs="1" type="xsd:string"/>
      </xsd:sequence>
    </complexType>
  </element>
</schema>
  
```

Name/ID	Type	Constraints
Record	Class	hasHeader: minCard = 1 hasMetadata: minCard = 1
Header	Class	hasIdentifier: minCard = 1
Metadata	Class	hasOai_citeseer: minCard = 1
Oai_citeseer	Class	-

Name/ID	Type	Domain	Range
hasHeader	ObjectProperty	Record	Header
hasMetadata	ObjectProperty	Record	Metadata
hasIdentifier	ObjectProperty	Header	Identifier
hasOai_citeseer	ObjectProperty	Metadata	Oai_citeseer
dc:identifier	DatatypeProperty	Header	xsd:string
dc:title	DatatypeProperty	Oai_citeseer	xsd:string

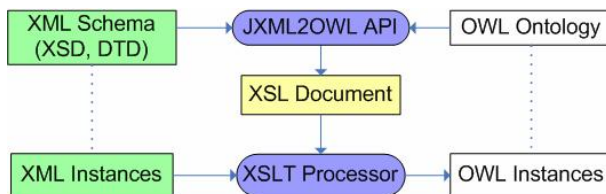
(c) Exemplo de XML usado em XML2OWL

(d) XSD Gerado em XML2OWL

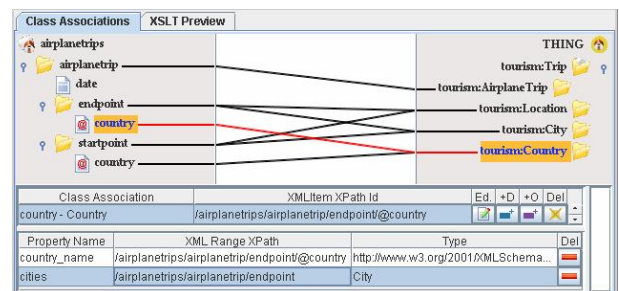
(e) XSD para OWL em XML2OWL

4.3 Mapping XML to Existing OWL ontologies

Cardoso et al [ISA 06] supõe uma ontologia existente e em alinhar os dados de documentos XML com esta ontologia. Foi implementada uma ferramenta, chamada de JXML2OWL, aos moldes da ferramenta de Bohring et al [BOH 02], porém sem se preocupar em formar um OWL, mas sim em alinhar o esquema gerado pela transformação XSLT em uma ontologia pré-formada. A Figura 4.1(f) apresenta o funcionamento da ferramenta e a Figura 4.1(g) apresenta sua interface.



(f) Funcionamento da ferramenta JXML2OWL



(g) Interface da ferramenta JXML2OWL

O primeiro passo é carregar a ontologia e o XML Schema. A partir deles o usuário tem que fazer o mapeamento do XML Schema para a ontologia pré-carregada, como mostra a Figura 4.1(g), depois disso no último passo é utilizada uma transformação XSLT para gerar a nova ontologia.

Os resultados deste trabalho mostram que uma ontologia pré-pronta ajuda na qualidade da conversão de um documento XML em OWL. Entretanto, a custo da criação da ontologia não é considerada.

Capítulo 5

XMLtoOWL

Neste capítulo é apresentada a ferramenta proposta para a geração de uma ontologia a partir de documentos XML. Esta ferramenta é composta por dois estágios, sendo o objetivo do primeiro gerar a partir de diversos documentos XML um XML Schema e o do segundo obter uma ontologia a partir do esquema gerado e persisti-la no formato OWL. Para construir esta ferramenta foi feito um estudo sobre qual seria a melhor estratégia para alcançar a ontologia. A partir de estudos na base de dados dos trabalhos realizados no Grupo de *Banco de dados da UFSC*, foi constatado uma ferramenta que provê uma ontologia a partir de um XML Schema.

Esta ferramenta elaborada por Garcia et al.[LGG 06] é uma implementação em JAVA da primeira partes do sistema BInXS[MEL 02]. A ferramenta foi construída com intuito de gerar uma ontologia a partir de um XML Schema, seguindo as Regras ORM(Object Role Model), estas regras encontram-se definidas no trabalho de conclusão de curso de Frantz et al.[FRA 04]. Contudo, este processo necessita da atuação de um usuário especialista durante o processo, definindo quais regras ORM serão utilizadas sobre o XML Schema para a geração da ontologia.

A partir deste trabalho foi elaborada uma ferramenta capaz de gerar, a partir de documentos XML, um XML Schema para repassado a esta ferramenta existente, com objetivo de gerar uma ontologia a partir de documentos XML. A estrutura da ferramenta desenvolvida pode ser conferida na Figura 5.1, onde o primeiro primeiro estágio corresponde a geração do XML Schema e o segundo a persistência da Ontologia gerada em OWL.

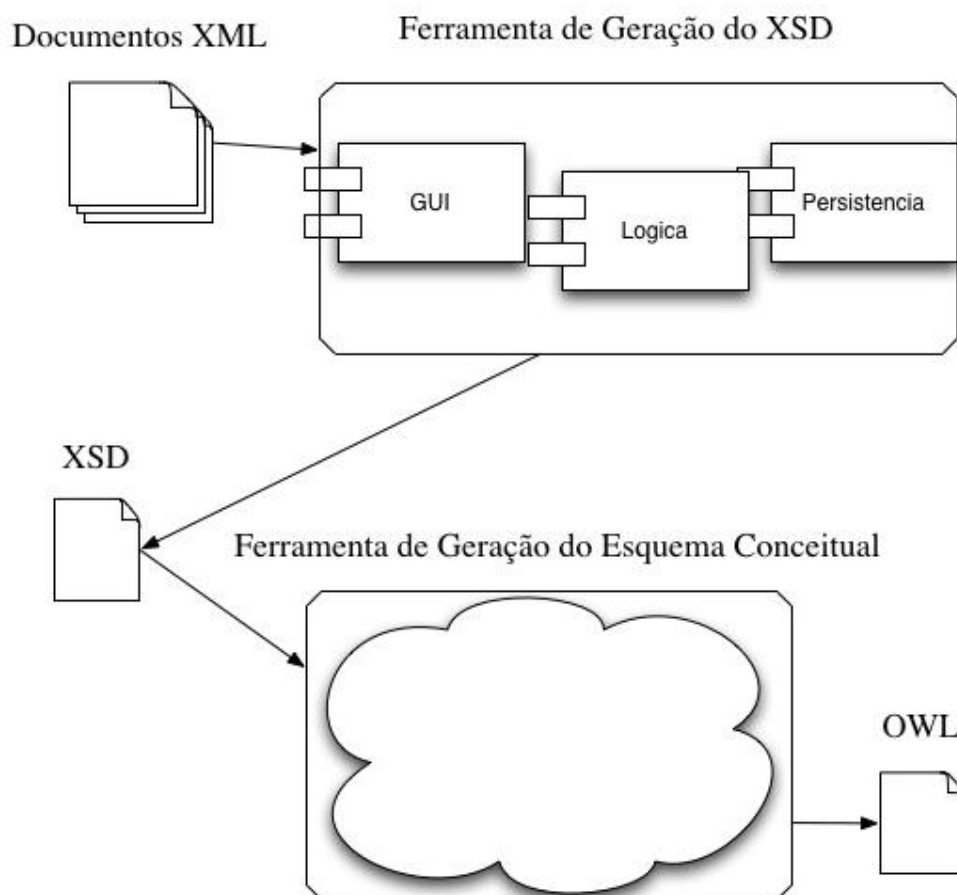


Figura 5.1: Estrutura da Ferramenta para Geração de uma Ontologia - XMLToOWL

5.1 Obtenção do XML Schema

O primeiro estágio do processo para geração da ontologia é a obtenção de um XML Schema a partir dos documentos XML. Para este processo foi construído uma ferramenta que infere, com algumas regras, um XML Schema a partir de diversos documentos XML.

Esta ferramenta é composta por três camadas, sendo elas: A GUI(Graphical User Interface), Lógica e a Persistência. Foi proposto esta estrutura de camadas em função de um posterior reuso que poderá ser feito aproveitando somente algumas partes da mesma e também para facilitar a manutenção que eventualmente poderá ser feita na ferramenta.

São detalhadas nas próximas seções cada uma das três camadas pertinentes a geração do XML Schema.

5.1.1 GUI - Graphical User Interface

Esta camada interage com o usuário, é nela que o usuário escolhe quais documentos XML serão utilizados para obter uma ontologia e também que o usuário define as regras ORM que serão necessárias para a formação do OWL.

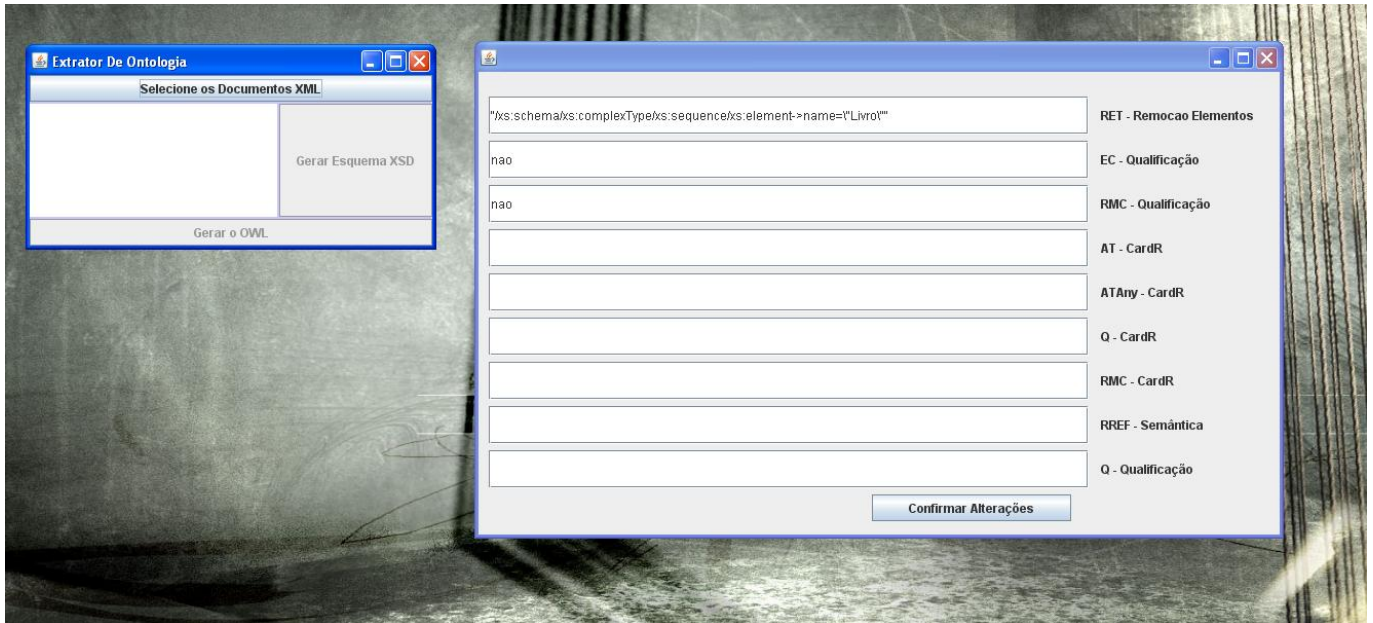
A classe principal desta camada é a *Gui*, mesmo nome da camada que a representa, e é nesta que estão contidas o método *main* do JAVA, invocado para iniciar a aplicação, e também aonde é programado a parte gráfica da ferramenta. A interface foi elaborada utilizando-se a *API SWING*[ECK 98]. A Figura 5.2(a) mostra a interface gráfica construída para interação com o usuário.

Depois do usuário selecionar os documentos XML que serão utilizados para a formação do esquema, ela os repassa para a camada Lógica.

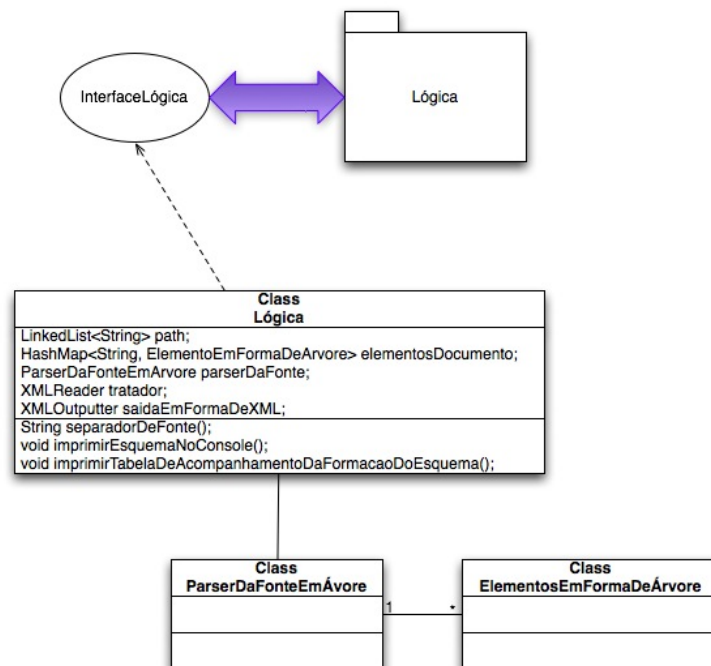
5.1.2 Lógica

Um XML Schema é composto por diversos elementos, alguns dos quais, são gerados pela ferramenta, e podem ser conferidos na Tabela 2.1. Estes elementos são extraídos diretamente dos documentos XML. Para obter estes elementos presentes é executado um algoritmo que percorre os documentos XML e infere os elementos. A camada Lógica realiza este trabalho de percorrer os documentos XML e inferir o XML Schema.

Dentro desta camada que está contido o cerne da ferramenta. Nela está programada toda a especificação da geração do XSD. A estrutura desta camada é representado pelo diagrama de



(a) Interface Gráfica com o Usuário



(b) Diagrama de Classes da Lógica

classes, ilustrado pela Figura 5.2(b) e o relacionamento com outros pacotes e classes do programa é feito por meio da classe Interface.

A classe Lógica repassa serialmente para a classe Parser todos os documentos XML que foram selecionados pelo usuário, sendo que a classe Parser nada mais é que uma implementação da API SAX (Simple API for XML) do JAVA[ECK 98]. A API SAX define uma forma serial de

acesso a documentos XML, isto é, os elementos presentes em documentos XML são tratados um a um.

Apesar de ser uma API rápida, visto que ela acessa de forma serial os elementos contidos em um documento XML, ela limita a conformação de árvore do XML mostrando somente os elementos serialmente ao programador. Como a ferramenta necessita desta conformação para inferir os elementos presentes no XSD, apresentados na Tabela 2.1, utilizou-se uma estrutura de dados para guardar os elementos de interesse obtidos pela execução do parser SAX.

Esta estrutura representa uma árvore sumarizada dos elementos contidos nos documentos XML e cada um dos nodos desta árvore é representado pela instância da classe *ElementosEmFormaDeÁrvore*. Esta classe contém todos os atributos e métodos necessários para a formação do XSD. A geração desta árvore é feita utilizando-se da API SAX, esta pega cada elemento do documento XML e insere-se estes elementos na árvore. A partir do momento que existir uma diferença entre estes elementos, como por exemplo um elemento passou de simples para complexo (*Anytype*) ou se o elemento é complexo e apresenta entre suas *tag* (*Mixed*) texto sua estrutura é tratada e modificada na árvore sumarizada, a Tabela 5.1 mostra como os elementos são tratados e modificados pela ferramenta. Um exemplo da árvore sumarizada é apresentado na figura 5.2.

XML	Árvore Sumarizada	Explicação
<biblioteca>	<i>biblioteca</i>	primeiro elemento da árvore sumarizada
<livro>	<i>livro</i>	segundo elemento da árvore sumarizada
<autor>Jose</autor>	<i>autor</i>	primeiro elemento filho de <i>livro</i>
</livro>		
<livro>		este elemento ja encontra-se na árvore sumarizada
<autor>		este elemento ja encontra-se na árvore sumarizada
<nome>Jose</nome>		primeiro elemento filho de <i>autor</i> , agora autor passou de simples para complexo
</autor>		
</livro>		
<midia tipo="DVD">	<i>midia</i>	terceiro elemento da árvore sumarizada
Nirvana</midia>		

Figura 5.2: Exemplo de uma Árvore Sumarizada de um Documento XML

Para obter os elementos *SimpleType*, *SimpleContent* e *Empty* primeiramente faz-se uma verificação na lista de filhos se ela não possuir elemento isto significa que o elemento é um elemento simples, diante desta confirmação é consultado a lista de atributos se existem atributos define-se o elemento como o complexo do tipo *SimpleContent* a lista de atributos se e elemento simples não apresentar nem atributos nem texto entre suas *tag* ele é setado para *Empty*. Os tipos de dados

presentes no documento são setados nesta etapa fazendo uma tentativa de conversão (*casting*) dos textos presente entre suas *tags*. A partir do momento que o Parser termina a varredura dos elementos dos documentos XML, a árvore sumarizada está pronta para que de forma recursiva construa-se o XML Schema.

A construção do esquema é feita percorrendo recursivamente a árvore sumarizada e invocando métodos e atributos presentea na classe *ElementosEmFormaDeÁrvore* que representa os nodos da árvore sumarizada. Alguns atributos determinam diretamente características dos elementos desta árvore, por exemplo, o atributo *mixed* que determina se um elementos é do tipo *Mixed*, esse atributo retorna um booleano que representa esta característica. Contudo, algumas características precisam ser inferidas invocando métodos desta classe, um exemplo destes métodos, é o que verifica se um determinado elemento é do tipo *All*, este faz uma verificação dos seus filhos deste elemento, analisando os atributos booleanos *obligatorio* e *temMuitos*, se o atributo *obligatorio* retornar true e o atributo *temMuitos* retornar false para todos os elementos das lista de filhos, tem-se um indicador de ordem *All*. Outro exemplo, é o método que identifica se um elemento é do tipo *Choice*, esta identificação é feita percorrendo os filhos deste elemento, se o atributo *obligatorio* de todos os filhos retornar false, este elemento é setado para *Choice*.

Algumas regras foram programadas para o tratamento de elementos presentes nos documentos XML e que porventura possam gerar conflitos na formação do XSD, como mostra a tabela 5.1. Para contruir o XML Schema no padrão adotado pela W3C foi utilizada a API JDOM[HUN].

Além de tratar a estrutura dos documentos XML e confrontá-la com os elementos do XSD a ferramenta também trata os tipos de dados presentes nos elementos dos Documentos XML. Os tipos de dados que aparecem nos os elementos XML são identificados e armazenados em um atributo da classe *ElementosEmFormaDeÁrvore*. Estes tipos de dados são importantes na formação do XML Schema, colaborando para uma correta correlação do esquema gerado com os documentos XML. Os tipos de dados identificados nos documentos XML são: literais, flutuantes e datas.

A partir do momento que o XSD está pronto ele é repassado para a Gui.

5.1.3 Persistência

Este pacote contém a classe Persistência, que persiste o XML Schema, gerado na camada Lógica, em um arquivo no sistema operacional para que posteriormente possa ser utilizado no próximo estágio da ferramenta. A classe principal usa a API de Java NIO gravando o arquivo no lugar selecionado pelo usuário. O desacoplamento da Camada de Persistência, garante que no

futuro outros tipos de persistência possam ser feito, como por exemplo, persistir o esquema em um banco de dados relacional, a Figura 5.3 representa o diagrama de classes da Persistência.

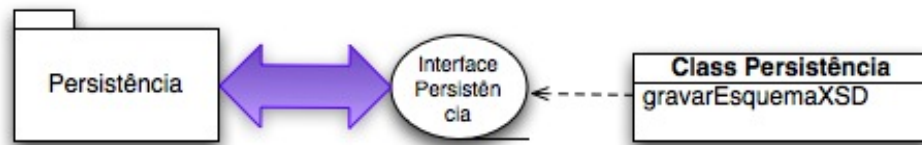


Figura 5.3: Diagrama de Classes da Persistência

5.2 Obtenção da Ontologia

A obtenção da ontologia fica a cargo da ferramenta proposta por Garcia et al. [LGG 06], ela implementa regras de conversão de esquemas descritos em *XML Schema Definition* (XSD) para esquemas conceituais ORM e o seu armazenamento em documentos OWL. Para contextualizar sobre as regras ORM o trabalho de Frantz et al. [FRA 04] fornece teoricamente todas as regras inferidas no trabalho de Garcia et al. [LGG 06].

A ferramenta proposta por Garcia et al. [LGG 06] basicamente é dividida em três passos: *pré-processamento*, *conversão* e *reestruturação*.

O *pré-processamento* realiza uma “limpeza” no XML-Schema, de forma que obtenha-se esquemas mais simplificados e mais bem estruturados. Já o passo de *conversão* realiza uma conversão propriamente dita, para um modelo canônico definido em [MEL 02]. O último passo de *reestruturação* realiza ajustes de forma que otimize a definição do esquema conceitual e gere-se um OWL como resultado. É importante destacar que esta ferramenta apresenta regras que o usuário define em todo processo com o objetivo de obter uma ontologia mais precisa. Contudo estas a explicação destas regras ORM foge ao escopo deste trabalho.

Sendo assim, este trabalho foi visto de forma análoga a uma caixa preta, como mostra a Figura 5.1. Com isto, foi alterado somente a interface da ferramenta, para que com isto ela possa usar o esquema gerado na etapa anterior, como mostra a Figura 5.4.

```
public ProcessoMain(String endereco, String nome) {  
    interfaz = new Interface(endereco, nome);  
  
    String nomeArq;  
    LinkedList<String> nomeArqProcessado = interfaz  
        .intervencaoUsuarioEspecialista("", "Nome Arquivo", "");  
    construtor = new ConstrutorOWL(interfaz, nomeArqProcessado.get(0)  
        .substring(0, nomeArqProcessado.get(0).length() - 4));  
  
    Set<AbstractObserver> observadores = new HashSet<AbstractObserver>();  
    observadores.add(construtor);  
}
```

Figura 5.4: Mudança na Classe principal para receber o esquema gerado na etapa anterior

XML1	Estrutura XSD intermediária	XML2	Estrutura XSD de saída
<x>bozo </x>	<element name = ``x`` type ``string`` />	<x/>	<element name = ``x`` type ``string`` />
<x>bozo </x>	<element name = ``x`` type ``string`` />	<x atrib:``a``> bozo</x>	<element name = ``x`` type ``string``> <complexType> <simpleContent> <attribute name=``a`` type=``string``> </simpleContent> </complexType> </elemente>
<x> <a>teste </x>	<element name = ``x``> <complexType> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element>	<x> teste <c>teste</c> </x>	<element name = ``x`` type ``string``> <complexType> <choice> <element name = ``a`` type ``string`` /> <element name = ``b`` type ``string`` /> <element name = ``c`` type ``string`` /> </choice> </complexType> </element>
<x>bozo teste </x>	<element name = ``x`` type ``string`` />	<x> teste <c>teste</c> </x>	<element name = ``x`` type ``anytype``> <complexType> <sequence> <element name = ``b`` type ``string`` /> <element name = ``c`` type ``string`` /> </sequence> </complexType> </element>
<x> <a>teste </x>	<element name = ``x`` type ``string``> <complexType> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element>	<x> teste<a>teste </x>	<element name = ``x`` type ``string``> <complexType mixed=``true``> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element>

Tabela 5.1: Tabela de tratamento dos elementos XSD

Capítulo 6

Estudo de Caso

Neste capítulo é apresentado um breve estudo de caso da ferramenta. Elaborou-se dois exemplos de documentos XML, apresentados na Figura 6.1(a). Estes apresentam elementos com características díspares, de forma que seja possível demonstrar como a ferramenta trata estas diferenças, no caso destes exemplos eles foram feitos na área de bibliografia de documentos.

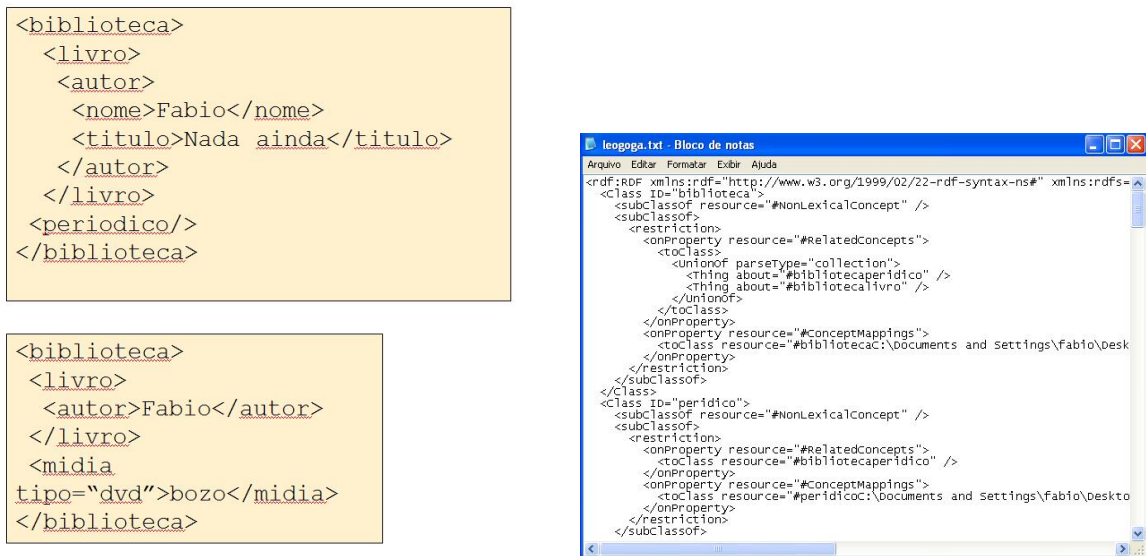
6.1 Execução

A partir do momento que o usuário definiu na camada GUI quais os documentos dos quais deseja-se obter uma ontologia, estes são passados para a camada Lógica. Para este estudo de caso, definiu-se também de forma genérica as regras ORM, porém caso o usuário deseje uma ontologia no formato OWL mais precisa, ele pode inferir nas regras ORM mais especificidades, conforme propõe o trabalho de Garcia et al [LGG 06]. A Figura 6.1 apresenta um exemplo de como estas regras se encontram no código fonte atualmente, de como elas poderão ser modificadas pelo usuário e como elas são definidas neste estudo de caso.

Elemento	temMuitos	Obrigatório	categoria	tipo	mixed	anytype
biblioteca	false	true	all	null	false	false
periodico	false	true	empty	null	false	false
livro	all	false	true	null	false	false
autor	all	false	true	null	false	false
nome	simpleType	false	true	string	false	false
título	simpleType	false	true	string	false	false

Tabela 6.1: Primeiro Documento XML

O sistema então, a partir do primeiro documento, gera uma árvore sumarizada contendo em



```

if (regra.equals("EC") && oQue.equals("Qualificação"))
    resposta.addLast("nao");
if (regra.equals("RMC") && oQue.equals("Qualificação"))
    resposta.addLast("nao");
if (regra.equals("AT") && oQue.equals("CardR"))
    resposta.addLast("default");
if (regra.equals("ATAny") && oQue.equals("CardR"))
    resposta.addLast("default");
if (regra.equals("Q") && oQue.equals("CardR"))
    resposta.addLast("default");

```

Figura 6.1: Algumas regras ORM definidas no processo

cada nodo um elemento da mesma com atributos necessários para inferir os elementos do XSD Schema, como mostra a Tabela 6.1. Por exemplo, o elemento *autor*, este aparece somente uma vez no primeiro documento e apresenta dois filhos, sendo assim pelo seus atributos e filhos, infere-se que ele é do tipo *All*, conforme foi explicado no Capítulo 5.

No momento que o segundo documento entra no processo, ele é comparado com os outros elementos presente nos nodos da árvore sumarizada, mantendo sempre a forma mais abrangente de classificação dos elementos e seus atributos, como mostrado na Tabela 6.2. Neste momento o elemento *autor* é definido como *sequence*, por não ter apresentado todos os elementos presentes no primeiro documento, sendo assim seu tipo é modificado para *Anytype* por ter apresentado uma mudança de complexo para simples.

Após a conclusão da varredura nos documentos, é gerado o Schema XSD e este é passado de forma automatizada para o segundo processo. A ontologia OWL gerada pela segunda etapa é

Elemento	temMuitos	Obrigatório	categoria	tipo	mixed	anytype
biblioteca	true	true	all	null	false	false
periodico	false	false	empty	null	false	false
livro	sequence	true	true	null	false	false
autor	sequence	true	true	anytype	false	true
nome	simpleType	false	false	string	false	false
titulo	simpleType	false	false	string	false	false
midia	simpleContent	false	false	string	false	false

Tabela 6.2: Segundo Documento XML

demonstrada na Figura 6.1(b).

Capítulo 7

Conclusão

Este trabalho apresentou uma revisão bibliográfica sobre os conceitos de XML e seus esquemas, ontologia e OWL. Apresentou-se também uma revisão dos trabalhos relacionados que tentam de alguma forma gerar uma ontologia a partir de documentos XML e a partir desta foi elaborada uma ferramenta capaz de gerar de uma ontologia a partir de documentos XML. Para realizar este processo primeiramente foi criada uma ferramenta para a obtenção de XML Schema a partir de documentos XML. Esta ferramenta foi elaborada em cima de regras que tratam de algumas peculiaridades, apresentadas na tabela 2.1, presentes na estrutura de documentos XML e que são representadas no XML Schema de forma que tenha-se uma XML Schema destes documentos.

A partir do esquema gerado na etapa anterior que é a entrada da ferramenta programada por Garcia et al. [LGG 06], gera-se uma ontologia. Esta ferramenta proposta de geração de esquemas conceituais necessita como entrada um XML-Schema, a partir deste e com regras ORM pré-dispostas por um usuário especialista gera-se um esquema conceitual em OWL, com isto temos a formação da ontologia proposta.

O presente trabalho apresenta um diferencial no que diz respeito em analisar um grupo de documentos XML e retornar um único XML Schema e a partir deste prover uma ontologia comum a todos os documentos XML e esta ontologia ser composta por um usuário especialista. Isto gera a possibilidade de usar documentos de diversos autores e tentar chegar em um esquema único. Apesar de oferecer um OWL a partir de documentos XML, alguns pontos-chaves ainda faltam para a completude do presente trabalho. Algumas características dos XML-Schema não foram abordadas, como por exemplo:

- A cardinalidade dos elementos presentes nos documentos XML, prevendo o número máximo e mínimo que um elemento pode aparecer nos documentos XML;

- Apresentar mais tipos de dados presentes nos documento XML, como por exemplo: byte, boolean, etc...

Na parte relacionada com a geração do modelo conceitual, onde há a necessita de intervenção de um usuário especialista, que este não precise alterar o código fonte para aplicar esta intervenção. Propõe-se que esta intervenção fuja do modelo atual, onde é feita no código do programa e esta apareça na interface GUI com o usuário.

Iniciou-se também a conversão da ferramenta em um *WebService* que possa ser integrado a outros sistemas existentes.

Referências Bibliográficas

- [BOH 02] BOHRING, H.; AUER, S. Mapping xml to owl ontologies. [S.l.], 2002.
- [BRE 05] BREITMAN, K. **Web Semântica A internet do Futuro**. LTC, 2005.
- [CON] CONSORTIUM, W. W. W. **XML Schema. World Wide Web Consortium (W3C)**.
- [ECK 98] ECKEL, B. **Thinking in Java**. A Simon & Schuster Company, 1998.
- [ELM 05] ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. Person Education do Brasil, 2005.
- [FRA 04] FRANTZ, A. P.; DOS SANTOS MELLO, R. Um processo de conversão de xml schemas para esquemas conceituais. 2004. [s.n.], 2004.
- [GAR 03] GAROFALAKIS, M. N. et al. Xtract: Learning document type descriptors from xml document collections. **Data Min. Knowl. Discov.**, [S.l.], v.7, n.1, p.23–56, 2003.
- [GON 07] GONÇALVES, R.; DOS SANTOS MELLO, R. Improving XML instances comparison with preprocessing algorithms. In: Wagner, R.; Revell, N.; Pernul, G., editors, DEXA, 2007. Springer, 2007. v.4653 of **Lecture Notes in Computer Science**, p.13–22.
- [GRU 92] GRUBER, T. R. **Ontolingua: A Mechanism to Support Portable Ontologies**.
- [HAK 01] HAKIMPOUR, F.; GEPPERT, A. **Resolving semantic heterogeneity in schema integration: An ontology base approach**.
- [HUN] HUNTER, J. **The JDOM Project**.
- [ISA 06] Isaías, P.; Nunes, M. B.; Martínez, I. J., editors. **Mapping XML to Existing OWL ontologies**. International Conference WWW/Internet 2006, (Eds), 2006.

- [LGG 06] LEONARDO G. GARCIA, R. S. M. Definição e implementação de regras de conversão de um esquema xml schema para um esquema conceitual. [S.l.], p.6, 2006.
- [MAU 02] MAURICIO, C. R. M.; F.PERES, F. F.; DOS SANTOS MELLO, R. Xml schema Ú recursos, características e definição de um modelo de representação gráfica. Universidade Federal de Santa Catarina - UFSC, 2002. Relatório técnico.
- [MEL 02] MELLO, R. D. S. **Uma Abordagem Botton-Up para a Integração Semântica de esquemas XML**. UFRGS, 2002. Tese de Doutorado.
- [SAN 07] SANTOS, F. et al. Um framework para suporte à preparação e comparação de similaridade de documentos xml. In: III ESCOLA REGIONAL DE BANCO DE DADOS, 2007. [s.n.], 2007.

XML2OWL: Uma Ferramenta para a Geração de uma Ontologia a partir de Documentos XML

Fábio dos Santos, Rodrigo Gonçalves, Ronaldo dos Santos Mello

¹Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) - Caixa Postal - 476 - CEP - 88049-900 – Florianópolis - SC - Brasil

{haqim, rodrigog, ronaldo}@inf.ufsc.br

Abstract. *This work aims to build a conceptual schema from XML documents to improve the quality of existing integration systems by providing semantic support to them. It proposes a tool capable of building an ontology from information contained in XML documents. It is composed of two main stages: the first extracts an XSD schema from the documents, which is then processed by the second stage, generating, through the use of ORM rules, a conceptual schema in OWL.*

Resumo. *Este trabalho apresenta uma ferramenta capaz de construir um esquema de uma ontologia a partir das informações contidas em documentos XML, com o objetivo de melhorar sistema de integração existentes, pelo fornecimento de um suporte semântico aos mesmos. O processo adotado pela ferramenta divide-se em dois estágios: o primeiro extrai um esquema XML Schema de documentos XML, que é processado pelo segundo estágio e gera, através do uso de regras de conversão, uma ontologia descrita no formato OWL.*

1. Introdução

A Web funciona como um meio de disponibilização de informações de fontes de dados de vários domínios e a XML tem sido utilizada para publicar estes dados. Sua adoção deve-se a sua natureza simples, porém flexível e extensível de organizar informações. Esta flexibilidade permite que fontes de dados XML na Web possam ter sua intenção semântica descrita de diversas maneiras através das tags definidas por seus autores. Tal fato pode gerar problemas ao comparar dados de duas ou mais fontes durante, por exemplo, consultas sobre elas.

Sistemas de integração de dados procuram resolver este problema geralmente provendo ao usuário uma visão integrada e completa dos esquemas destas fontes de dados. Tais sistemas, porém, têm limitações na qualidade semântica dos seus processos, necessitando do auxílio de usuários especialistas para realizar as suas tarefas. A definição de equivalências semânticas entre os dados das fontes é um exemplo.

Sistemas de integração de dados podem contar com o suporte de ontologias [BREITMAN 2005]. Uma ontologia pode dar apoio a com semântica às fontes de dados [Bohring and Auer 2002], prover assim como uma visão conceitual integrada das informações ou mesmo manter equivalências semânticas para dados das mesmas. Diversas linguagem foram desenvolvidas para especificar ontologias, dentre as quais a OWL, que é a recomendação mais recente da W3C.

Este artigo apresenta uma ferramenta capaz de, a partir de um conjunto de documentos XML, estabelecer uma ontologia contendo informações presentes nestes documentos. Os documentos não precisam ter esquemas associados, como ocorre em muitas fontes na *Web*. O processo dá-se em duas etapas distintas: na primeira, extrai-se o esquema dos documentos XML de uma certa fonte. Na segunda, o esquema extraído é convertido em uma ontologia especificada em OWL.

A seção 2 deste trabalho contextualiza o problema e a solução proposta. A seção 3 descreve a estrutura da ferramenta, que tem seu funcionamento explicado na seção 4. Na seção 5 analisam-se os principais trabalhos relacionados. Por fim, na seção 6 analisam-se as contribuições e possíveis melhorias ao trabalho.

2. Contextualização

A presente ferramenta foi concebida no âmbito do projeto Digitex (Processo Institucional CNPq/CTInfo: 550.845/2005-4). Este projeto trata da construção de uma Plataforma de Indexação e Busca Personalizada em Bibliotecas Digitais (BibDig), permitindo a consulta sobre bibliotecas digitais de forma centralizada e homogênea. Em várias bibliotecas referentes a um mesmo domínio, como a CiteSeer¹ e a DBLP² para o domínio bibliográfico acadêmico, a linguagem XML é adotada para a organização e disponibilização das informações. Esta disponibilização, porém, não possui um padrão definido quanto à estrutura dos dados. Autores de publicações, por exemplo, podem ter o dado *nome* representado através de um campo único ou dividido em nome e sobrenome.

Um dos aspectos que o projeto Digitex visa tratar é a análise e compatibilização na forma como BibDigs em um mesmo domínio disponibilizam suas informações. Para tanto, o Grupo de Banco de Dados da UFSC (GBD/UFSC)³ têm trabalhado a temática de compatibilização, comparação e integração de documentos XML. Para o processo de compatibilização, diversas heurísticas foram elaboradas como parte de uma dissertação de Mestrado, da qual parte dos resultados foi apresentada em dois artigos [Gonçalves and dos Santos Mello 2007, Santos et al. 2007]. Estas heurísticas, porém, sofrem a limitação da necessidade de uma estrutura semântica de suporte. Um exemplo de estrutura capaz de prover este suporte seria uma ontologia.

A construção de uma ontologia demanda tempo e a presença de especialistas no domínio. Deve-se prever todos os possíveis dados que podem estar contidos na mesma, algo difícil se não têm-se um conhecimento prévio preciso dos dados a serem armazenados. Tal realidade é presente no contexto do projeto Digitex, onde várias fontes de dados podem ser integradas. Se for possível automatizar parte do processo de criação da ontologia, têm-se um ganho de tempo no processo.

A partir deste problema, surgiu a proposta de elaborar um sistema que, a partir de documentos XML que representam dados de BibDigs, elabore uma ontologia capaz de conter as instâncias de dados presentes nos mesmos. Tal proposta originou a ferramenta apresentada neste trabalho. Com o auxílio de outra ferramenta também desenvolvida no GBD UFSC [Leonardo G. Garcia 2006], gera-se um esquema conceitual de um conjunto de documentos XML, ou seja, a base estrutural de uma ontologia, na sintaxe OWL.

¹<http://citeseer.ist.psu.edu/>

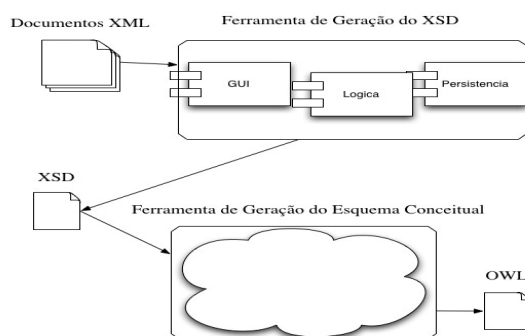
²<http://www.informatik.uni-trier.de/~ley/db/>

³<http://www.grupobd.inf.ufsc.br/>

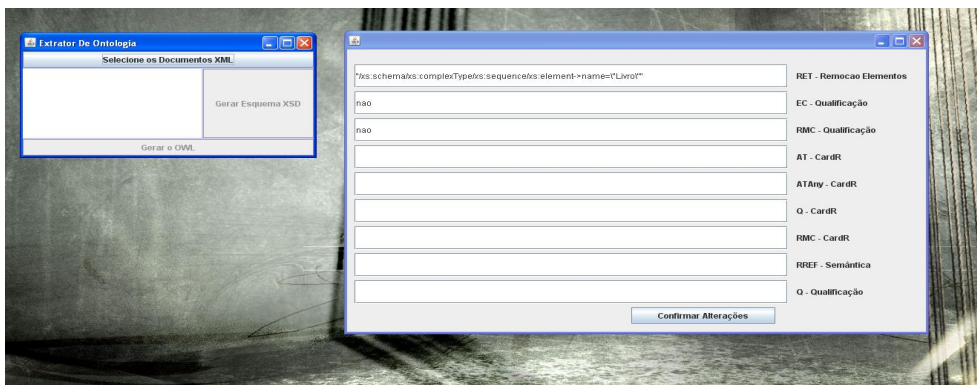
3. Estrutura Geral e Implementação da Ferramenta

A ferramenta foi desenvolvida em Java e usa a biblioteca JDOM para manipular e gerar os documentos XML envolvidos no processo. Ela é composta por dois módulos, como mostra a Figura 1(a). O primeiro é responsável por gerar um esquema em *XML Schema* (XSD) e está estruturado em uma arquitetura de três camadas. O segundo módulo, que gera o esquema conceitual OWL, utiliza uma ferramenta desenvolvida no GBD/UFSC [Leonardo G. Garcia 2006]. As únicas alterações feitas nesta ferramenta foram relativas a sua interface, para integrá-la com a ferramenta de geração de esquemas elaborada.

Diagrama do Funcionamento da Ferramenta



(a) Funcionamento da Ferramenta.



(b) Interface da Ferramenta.

O módulo *GUI* representa a interface com o usuário. Nesta camada ele decide em que lugar estão localizados os documentos XML. Ela é a responsável por invocar as outras camadas, *Lógica* e *Persistência*. A partir do momento que o usuário define os documentos a serem analisados, esta camada repassa a lista de documentos para a camada *Lógica*. Esta realiza então o processo de análise e geração do esquema XSD. Uma vez que a *Lógica* termina sua execução, a *Persistência* é invocada para persistir o esquema XSD gerado, em um local definido na *GUI* pelo usuário. A *Persistência* atualmente faz uso da API Java NIO.

O segundo módulo tem como entrada o esquema XSD gerado. Neste processo, uma interface é apresentada ao usuário, que parametriza o processo. A ferramenta de conversão XSD-OWL é então invocada. Esta ferramenta é vista como uma

caixa preta para este trabalho e detalhes sobre o seu funcionamento são apresentados em [Leonardo G. Garcia 2006].

4. Implementação

Na obtenção do esquema XSD, os elementos definidos no mesmo são gerados a partir de um processo iterativo de varredura sobre a estrutura dos documentos XML. O processo é totalmente automatizado. A partir da análise da estrutura dos documentos, o módulo responsável é capaz de gerar e adaptar o esquema XSD de forma que, ao final do processo, ele seja capaz de representar e validar todos os documentos. A Tabela 1 apresenta alguns exemplos de regras de converção de XML em XSD.

XML1	Estrutura XSD intermediária	XML2	Estrutura XSD de saída
<code><x>bozo</x></code>	<code><element name = ``x`` type ``string`` /></code>	<code><x/></code>	<code><element name = ``x`` type ``string`` /></code>
<code><x>bozo</x></code>	<code><element name = ``x`` type ``string`` /></code>	<code><x atrib: ``a``> bozo</x></code>	<code><element name = ``x`` type ``string``> <complexType> <simpleContent> <attribute name = ``a`` type = ``string``> </simpleContent> </complexType> </element></code>
<code><x> <a>teste </x></code>	<code><element name = ``x``> <complexType> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element></code>	<code><x> teste <c>teste</c> </x></code>	<code><element name = ``x`` type ``string``> <complexType> <choice> <element name = ``a`` type ``string`` /> <element name = ``b`` type ``string`` /> <element name = ``c`` type ``string`` /> </choice> </complexType> </element></code>
<code><x>bozo teste </x></code>	<code><element name = ``x`` type ``string`` /></code>	<code><x> teste <c>teste</c> </x></code>	<code><element name = ``x`` type ``anytype``> <complexType> <sequence> <element name = ``b`` type ``string`` /> <element name = ``c`` type ``string`` /> </sequence> </complexType> </element></code>
<code><x> <a>teste </x></code>	<code><element name = ``x`` type ``string``> <complexType> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element></code>	<code><x> teste<a>teste </x></code>	<code><element name = ``x`` type ``string``> <complexType mixed = ``true``> <sequence> <element name = ``a`` type ``string`` /> </sequence> </complexType> </element></code>

Tabela 1. Tabela de Exemplos de regras para geração do XSD.

Um exemplo deste processo iterativo ocorre quando um elemento muda de característica. Por exemplo, um elemento que apareceu inicialmente como simples em um

documento, apresenta uma versão estruturada em outro documento. Neste caso a ferramenta trata esta mudança, sempre deixando a característica mais genérica do elemento. Este é um diferencial desta ferramenta em relação a outras ferramentas disponíveis, que geram o esquema XSD para apenas um (1) documento XML. Os conteúdos presentes nos elementos também são tratados pela ferramenta, de forma a especificar os tipos de dados dos elementos e atributos no esquema.

No segundo módulo da ferramenta gera-se a ontologia em OWL. Apresenta-se uma interface em que o usuário especialista decide as transformações necessárias para que o OWL seja construído corretamente. A GUI fornece a possibilidade de interagir com a montagem do OWL através de um conjunto de regras conforme estabelecido por Garcia et al. [Leonardo G. Garcia 2006].

4.1. Exemplo de Execução da Ferramenta

```
<biblioteca>
  <livro>
    <autor>
      <nome>Fabio</nome>
      <titulo>Nada ainda</titulo>
    </autor>
  </livro>
</periodico/>
</biblioteca>
```

```
<biblioteca>
  <livro>
    <autor>Fabio</autor>
  </livro>
  <midia
tipo="dvd">bozo</midia>
</biblioteca>
```

(c) Exemplo de dois Documentos XML.

```
leopoga.txt - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <class id="biblioteca">
    <subclassof resource="#NonLexicalConcept" />
    <subclassof>
      <restriction>
        <onProperty resource="#RelatedConcepts">
          <toClass>
            <unionof parseType="collection">
              <thing about="#bibliotecaperiodico" />
              <thing about="#bibliotecalivro" />
            </unionof>
          </toClass>
        </onProperty>
      </restriction>
    </subclassof>
    <onProperty resource="#ConceptMappings">
      <toClass resource="#biblioteca:\documents and settings\fabio\deskt" />
    </onProperty>
  </class>
  <class id="periodico">
    <subclassof resource="#NonLexicalConcept" />
    <subclassof>
      <restriction>
        <onProperty resource="#RelatedConcepts">
          <toClass resource="#bibliotecaperiodico" />
        </onProperty>
      </restriction>
    </subclassof>
    <onProperty resource="#ConceptMappings">
      <toClass resource="#periodico:\documents and settings\fabio\deskt" />
    </onProperty>
  </class>
</rdf:RDF>
```

(d) Exemplo de OWL Gerada.

Elemento	categoria	tipo	mixed	anytype
biblioteca	all	null	false	false
periodico	empty	null	false	false
livro	all	null	false	false
autor	all	null	false	false
nome	simpleType	string	false	false
titulo	simpleType	string	false	false

Elemento	categoria	tipo	mixed	anytype
biblioteca	all	null	false	false
periodico	empty	null	false	false
livro	all	null	false	false
autor	all	null	false	false
nome	simpleType	string	false	false
titulo	simpleType	string	false	false

Figura 1. Exemplo de execução da ferramenta.

Considera-se por limitações de espaço do artigo, uma execução sobre dois documentos XML, mostrados na Figura 1(c). Uma vez definidos os documentos na GUI, estes são passados para a Lógica. A partir do primeiro documento gera-se uma árvore contendo os atributos em cada nodo necessários para a qualificação dos elementos e sua conversão para XSD, como mostrado na Figura 4.1. Quando o segundo documento entra no processo ele é comparado com os nodos da árvore, mantendo a forma mais abrangente de classificação dos atributos, como mostrado na Figura 4.1.

Após a conclusão da varredura nos documentos, é gerado o esquema XSD que é passado para o segundo módulo. A ontologia OWL gerada é mostrada na Figura 1(d).

5. Principais Trabalhos Relacionados

Em [Bohring and Auer 2002] é apresentada uma ferramenta para o mapeamento de um documento XML para um documento OWL. O mapeamento é feito através de um *framework* baseado em XSLT que infere um esquema XSD a partir do documento XML e, a partir deste, aplica uma transformação para obter um esquema OWL. Entretanto, este trabalho não permite a conversão de múltiplos documentos XML nem uma interação do usuário, de forma que se possa melhorar a qualidade do resultado do processo.

Em [Isaías et al. 2006] supõe-se uma ontologia existente e foca-se em alinhar os dados dos documentos XML com esta ontologia. Os resultados deste trabalho mostram que uma ontologia pré-pronta ajuda na qualidade da conversão de um documento XML em OWL. Entretanto, a criação da ontologia não é tratada.

6. Conclusão e Trabalhos Futuros

No domínio de Bibliotecas Digitais, o projeto Digitex propõe uma solução para a integração de várias fontes de dados XML. Com o objetivo de auxiliar este processo, a ferramenta apresentada neste trabalho foi elaborada. Ela é capaz de extrair uma ontologia a partir de dados XML, ontologia esta que serve como base conceitual a outros projetos para consulta e integração de fontes de dados. A principal contribuição desta ferramenta é a consideração de múltiplos documentos XML para a geração da ontologia, com intervenção reduzida de um usuário especialista.

Como limitações atuais, a ferramenta não identifica tipos de dados mais complexos, limitando aos tipos básicos. Heurísticas para identificar tipos mais complexos (data por extenso, etc.) estão sendo desenvolvidas. Uma geração mais apurada do XSD em relação as cardinalidades (máxima, mínima) dos elementos é outro ponto em estudo. Iniciou-se também a conversão da ferramenta em um *WebService* que possa ser integrado a outros sistemas existentes.

Referências

- Bohring, H. and Auer, S. (2002). Mapping xml to owl ontologies.
- BREITMAN, K. (2005). *Web Semântica A internet do Futuro*.
- Gonçalves, R. and dos Santos Mello, R. (2007). Improving XML instances comparison with preprocessing algorithms. In Wagner, R., Revell, N., and Pernul, G., editors, *DEXA*, volume 4653 of *Lecture Notes in Computer Science*, pages 13–22. Springer.
- Isaías, P., Nunes, M. B., and Martínez, I. J., editors (2006). *Mapping XML to Existing OWL ontologies*. International Conference WWW/Internet 2006, (Eds).
- Leonardo G. Garcia, R. S. M. (2006). Definição e implementação de regras de conversão de um esquema xml schema para um esquema conceitual. page 6.
- Santos, F., Júnior, C. S., Gonçalves, R., and Mello, R. (2007). Um framework para suporte à preparação e comparação de similaridade de documentos xml. In *III Escola Regional de Banco de Dados*.