Extração de Topic Maps no *Oveia*: Especificação e Processamento

Giovani Rubert Librelotto*
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
grl@di.uminho.pt

and

José Carlos Ramalho
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
jcr@di.uminho.pt

and

Pedro Rangel Henriques
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
prh@di.uminho.pt

Resumo

Este artigo apresenta uma solução para conquistar, com o uso de ontologias, interoperabilidade entre sistemas de informação heterogêneos (formados por bases de dados relacionais e documentos anotados, entre outros). O *Oveia* é um extrator de ontologias representadas no formato *Topic Maps*. Sua arquitetura é composta por duas especificações e os referentes processadores: a primeira, escrita na linguagem XSDS (*XML Specification for DataSources/DataSets*), especifica os dados a serem extraídos das fontes de informação; enquanto que segunda, escrita na linguagem XS4TM (*XML Specification for Topic Maps*), é responsável por declarar as ontologias a serem geradas. Com base nestas especificações, o extrator busca as informações nas fontes de informação e produz um *topic map*. Este *topic map* gerado pode ser armazenado em formato XTM (*XML Topic Maps*) ou em uma base de dados relacional. Essa dupla capacidade (manipular vários tipos de fontes de informação e armazenar o resultado em um suporte diferente) é uma clara vantagem na comparação com outras ferramentas de extração de ontologias; nomeadamente com o seu antecessor, o TM-Builder, que apenas permitia lidar com documentos XML, relativo ao qual este representa uma evolução justificativa.

Palavras chaves: *Topic Maps*, Extração de Ontologias, *Semantic Web*, XML, XSL.

Abstract

This paper presents a proposal based on ontology to achieve semantic interoperability in a heterogeneous information system. *Oveia* is an ontology extractor, following *Topic Maps* approach. *Oveia* was conceived to overcome the drawbacks of the known ontology extraction tools; namely, *Oveia* is a sucessor of *TM-Builder*. It provides an extraction model supported on an ontology specification language, XS4TM (*XML Specification for Topic Maps*) – a language to define the ontology to be extracted (topics, association, and instances) – but it also takes into consideration the characteristics of each data sources, interpreting a resource specification written in XSDS. The proposed extractor processes the XSDS and XS4TM specification and generates a *topic map*. This generated *topic map* can be stored in XTM syntax or in a relational database. This double capacity (to manipulate many kinds or information resources and to store the generated *topic map* in two different formats) is a clear advantage in comparison with the ontology extraction tools available.

Keywords: Topic Maps, Ontology Extraction, Semantic Web, XML, XSL.

*Bolsista CNPq - Brasil

1 Introdução

A chamada *Sociedade de Informação* necessita de ter acesso completo à informação disponível, a qual é geralmente heterogênea e distribuída. Para estabelecer uma partilha de informação eficiente, muitos problemas técnicos têm sido resolvidos. Primeiramente deve ser encontrada a fonte de informação apropriada (contendo os dados necessários para uma determinada tarefa). Encontrar recursos de informação apropriados é um problema tratado nas áreas de *recuperação de informação* (*information retrieval*) e *filtragem de informação* (*information filtering*) [2].
Uma vez localizada a informação, é necessário permitir o acesso aos dados. Isto significa que os recursos de informação, encontrados no primeiro passo, devem integrar e possibilitar que o sistema que efetuou a procura trabalhar com eles em conjunto. O problema de unir sistemas heterogêneos e sistemas distribuídos é conhecido como o *problema da interoperabilidade*.
Resumidamente, a partilha da informação não requer apenas que se tenha acesso completo aos dados; também requer que os dados acessados possam ser processados e interpretados pelo sistema remoto. Problemas que podem surgir da heterogeneidade dos dados são bem conhecidos na comunidade de sistemas de base de dados distribuídas (veja [7] e [6]): heterogeneidade estrutural (heterogeneidade esquemática) e heterogeneidade semântica (heterogeneidade de dados) [7]. Heterogeneidade estrutural significa que diferentes sistemas de informação armazenam seus dados em diferente estruturas. Heterogeneidade semântica resulta dos diferentes significados atribuídos a conteúdos semelhantes.
Para conseguir interoperabilidade semântica em sistemas heterogêneos de informação, o significado da informação que é intercambiada deve ser compreendido claramente pelo sistema interpretador [11]. Conflitos semânticos ocorrem toda vez que dois sistemas usam diferentes interpretações para a mesma informação, ou seja, quando há uma ambiguidade. Goh [4] identifica três causas principais para heterogeneidade semântica:

- Conflitos em geral ocorrem quando ítens de informação parecem ter o mesmo significado, mas de fato, diferem. Por exemplo, contextos temporais diferentes;
- Conflitos de escala ocorrem quando diferentes sistemas de referência são usados para medir um valor. Exemplo disto são diferentes moedas (valores monetários);
- Conflitos de nomes ocorrem quando os nomes em esquemas de informação diferem significativamente. Um fenômeno freqüente é a presença de homônimos e sinônimos.

O uso de ontologias para a explicação do conhecimento implícito e oculto é uma possível abordagem para resolver o problema da heterogeneidade semântica. Em [14], menciona-se a interoperabilidade como uma aplicação chave de ontologias, sendo ai referidas muitas abordagens baseadas em ontologias para integração de informação. É neste contexto que surge a nossa motivação para lidar com ontologias, tendo-se optado pela sua própria representação na forma de *Topic Maps*.
O processo de desenvolvimento de ontologias baseadas em *Topic Maps* é complexo, consumidor de tempo e requer grande quantidade de recursos humanos e financeiros, devido ao fato de qualquer *topic map* (por mais simples que seja) possuir um conjunto significativo de tópicos e associações; além disso, para que a ontologia extraída seja realmente significativa, pode envolver um grande número de recursos de informação que poderão ser de tipos diferentes.
Para resolver este problema, este artigo propõe um extrator de ontologias, chamado *Oveia*, que constrói *topic maps* a partir de recursos heterogêneos de informação, tais como bases de dados relacionais e documentos XML. Para isso, a ontologia a ser extraída é definida em uma linguagem de especificação denominada XS4TM (*XML Specification for Topic Maps*). O *topic map* extraído pode ser armazenado em uma base de dados relacional (permitindo que as ontologias possam crescer, sem restrições), ou em um documento no formato *XML Topic Maps* (XTM).
O artigo inicia apresentando o paradigma *Topic Maps* na seção 2; *Topic Maps* é um formalismo para representar conhecimento sobre um recurso de informação, organizando por tópicos. A descrição do sistema que se propõe, o extrator de *Topic Maps* a partir de recursos heterogêneos de informação – *Oveia* – é feita na seção 3. A definição da linguagem XS4TM será encontrada na seção 3.5. A seção 4 apresenta os trabalhos relacionados. Por fim, uma síntese do artigo e os trabalhos futuros são apresentados na conclusão.

2 Topic Maps

Topic Maps [3] é um formalismo para representar conhecimento acerca da estrutura de um conjunto de recursos de informação e para o organizar em *tópicos*. Esses tópicos têm ocorrências e associações que representam e definem relacionamentos entre os tópicos. A informação sobre cada tópico pode ser inferida ao examinar as associações e ocorrências ligadas ao tópico. Uma coleção desses tópicos e associações é chamada *topic map*. Também pode ser visto como um paradigma que permite organizar, manter e navegar pela informação, permitindo transformá-la em conhecimento. Falar sobre *Topic Maps*, é falar sobre estrutura de conhecimento.

Um mapa de tópicos expressa a opinião de alguém sobre o que os tópicos são, e quais as partes do conjunto de informação que são relevantes para cada tópico [13].
Permitindo a criação de um mapa virtual da informação, os recursos de informação mantém-se em sua forma original e não são modificados. Então, o mesmo recurso de informação pode ser usado de diferentes formas, por diferentes mapas de tópicos. Como é possível e fácil modificar um mapa, a reutilização da informação é conquistada.
Tópicos são o ponto principal de *Topic Maps* [12]. Em um sentido mais genérico, um tópico pode ser qualquer coisa: uma pessoa, uma entidade, um conceito. Eles constituem a base para a criação de *Topic Maps* (TM). Cada tópico tem um tipo de tópico (*topic type*), ou talvez múltiplos tipos. Cada tipo de tópico pode ser visto como uma típica relação classe-instância.
Ao analisar *Topic Maps*, identificam-se duas camadas distintas: os tópicos e as ocorrências. Os tópicos podem ser divididos em duas partes: os que representam conceitos abstratos e os que representam conceitos concretos. A ontologia é definida pelos conceitos abstratos, ou seja, os que serão instanciados por outros tópicos; por exemplo: tipo de tópico, tipo de associação e pelo tipo de papel de atuação em ocorrências.
Os tópicos restantes formam a base de conhecimento associada à ontologia, os quais compõem um conjunto de objetos de informação que permite organizar e indicar os reais recursos de informação (um objeto pode ter múltiplas ocorrências nos recursos de informação). A figura 1 dá uma representação esquematizada desta visão.

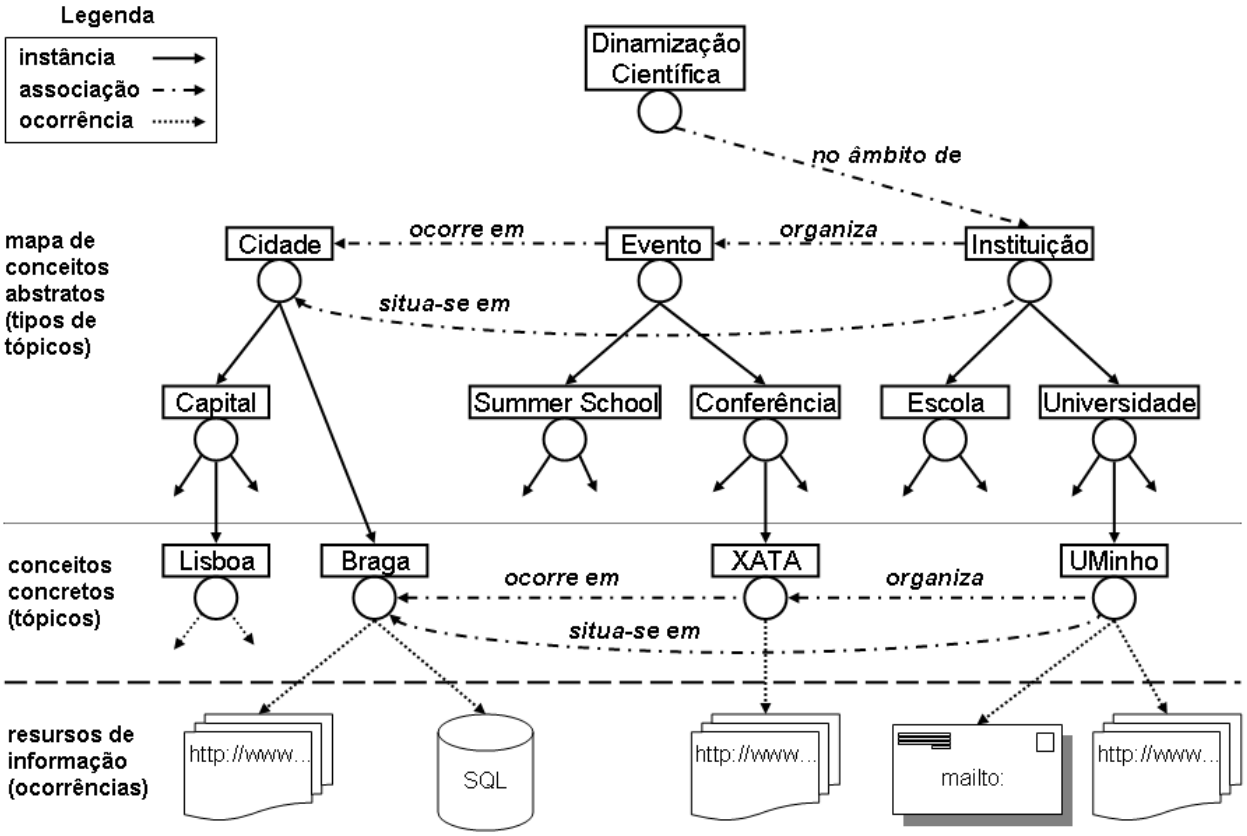

Figure 1: O Mapa do Conceito *Dinamização Científica*.

O conceito de associação (*association*) permite descrever relações entre tópicos. Uma associação é (formalmente) um elemento de vínculo que define uma relação entre dois ou mais tópicos. Um ilimitado número de tópicos podem ser relacionados por uma associação.

3 Oveia – Um Extrator de Topic Maps a partir de Recursos de Informação

O *Oveia* é um extrator de ontologias em sistemas heterogêneos de informação baseado em *Topic Maps*. O *Oveia* foi desenvolvido com o objetivo de suprir as deficiências encontradas nas atuais ferramentas de extração de ontologias. O *Oveia* surge na seqüencia do projeto que resultou no *TM-Builder* [8], o qual fornece um modelo de extração suportado numa linguagem própria para a especificação da ontologia a ser extraída.
Um fato que representa a evolução do *Oveia* em relação à sua versão inicial é a capacidade de extrair ontologias a partir de fontes de dados diversas. No *TM-Builder*, quando a fonte de informação é diferente de um documento XML, há necessidade de converter esta fonte para um documento XML. Portanto, considerando que a maior parte dos

recursos de informação em empresas e instituições estão armazenadas em base de dados, para realizar uma extração de uma ontologia a partir delas, inicialmente seria necessário a geração de documentos XML com o conteúdo da base de dados.

Assim como o *TM-Builder*, o *Oveia* faz uso de uma linguagem de especificação de extração (XS4TM), a qual permite extrair *Topic Maps* de forma genérica e adaptativa. A especificação de extração de ontologias em XS4TM tornou-se mais flexível e completa, contemplando todos os elementos do padrão *Topic Maps* [3]. Isso garante maior flexibilidade de especificação para propósitos diversos de extração.

Na fase de especificação dos recursos de informação, é possível fazer transformações e filtros nessas fontes de dados, pois é utilizada a linguagem de consulta de cada recurso para sua especificação.

A linguagem de especificação de extração foi inspirada no modelo XTM. Isso significa que a especificação da ontologia a ser extraída (em XS4TM) é feita em um esquema XML similar ao esquema de XTM. Essa característica permite maior facilidade de compreensão da especificação proposta, pois o modelo XTM é um padrão que vem sendo adotado amplamente pela comunidade acadêmica. Assim, o projetista da ontologia apenas deve conhecer a sintaxe de XTM e a estrutura das fontes de dados, para estar habilitado a especificar extrações de ontologias em XS4TM.

A arquitetura do *Oveia* pode ser expressa conforme demonstra a figura 2: inicialmente, é feita em uma especificação XSDS (*XML Specification for DataSources/DataSets*), a qual define quais dados devem ser recuperados pelo *Extrator de Datasets*; a informação extraída é armazenada em um formato intermediário, chamado *Datasets*. O próximo passo é a especificação da ontologia em XS4TM; essa fase determina o que é relevante para a extração dos tópicos e associações, assim como clarifica os limites que devem ser impostos ao *topic map*. O processador XS4TM recebe os *datasets* gerados e a especificação da ontologia na linguagem XS4TM (*XML Specification for Topic Maps*) e gera o *topic map* final. Por fim, o *Oveia* armazena o *topic map* gerado na *BD Ontologia* ou no formato XTM.

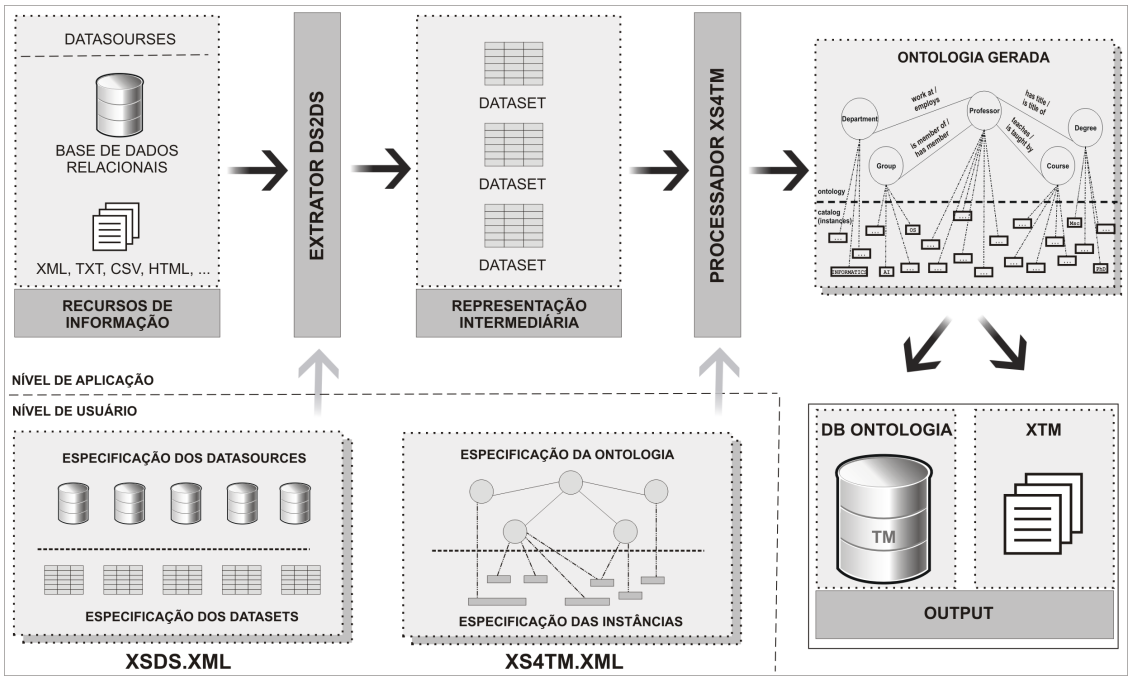

Figure 2: Arquitetura do Oveia

As próximas sub-seções apresentam cada um dos componentes do *Oveia*.

3.1 Recursos de Informação: Os *Datasources*

Este componente é composto pelos recursos de dados: bases de dados, documentos XML, páginas HTML, etc. Ao final do processo de extração de ontologias, os recursos manterão-se inalterados, ou seja, o *Oveia* não modifica as fontes; somente copia as partes relevantes de informação para a construção do *topic map*. Esses recursos de dados são mapeados para uma representação intermediária, chamada *datasets*. Esse mapeamento é descrito pela linguagem XSDS.

3.2 Representação Intermediária da Informação: Os *Datasets*

Os *datasets* são a representação intermediária que contém os dados extraídos das fontes de informação. Cada *dataset* tem uma relação com uma entidade dos *datasources*, e seu conteúdo é representado na forma de uma tabela, onde

cada linha é um registro segundo a estrutura definida em XSDS. Os *datasets* garantem que o *Oveia* tenha uma visão uniforme sobre a estrutura de dados que representam as fontes de dados participantes.
Cada *dataset* tem uma identidade única, a qual será usada pelo *Oveia* para o referenciar. A idéia fundamental é que todos os objetos tem rótulos que descrevem o seu conhecimento. Por exemplo, o seguinte objeto representa um registro da categoria *tipo de professor*:

<1,PhD>

onde "1" é o identificador da categoria, enquanto que "PhD" é um rótulo legível por humanos. Os *datasets* são simples, enquanto provém um poder de expressividade e flexibilidade necessário para integrar sistemas de informação de diferentes fontes.

3.3 XSDS: Especificação das Fontes de Dados

XSDS (*XML Specification for DataSources/DataSets*) é a linguagem definida com o intuito de especificar quais fontes de informação fornecerão dados para a criação de *topic maps*, de acordo com uma ontologia posteriormente especificada. Essa especificação fornece todos os elementos necessários para especificar as fontes de dados passíveis de extração de informação.
No inicio deste trabalho foi previsto a criação de uma linguagem universal para realizar consultas em recursos heterogêneos de informação. Mas a conversão das linguagens de consulta (SQL, XPath, ...) para a linguagem idealizada como universal não seria a solução ideal pelo fato de que o projetista da ontologia teria a necessidade de aprender uma nova linguagem, ao invés de usar as linguagens amplamente conhecidas e difundidas. Dessa forma foi adotada a estratégia de manter o padrão de consulta de cada recurso e processar o resultado gerado por cada uma das linguagens de consulta. Assim, os *datasets* seriam o ponto de unificação de representação da informação entre os metadados extraídos de cada recurso de informação.
De um modo formal, a *Context Free Grammar* de XSDS é definida a seguir:

```
1 Resources   ::= Datasources Datasets
2 Datasources ::= extratorDriver name Datasource+
3 Datasource  ::= Parameter+
4 Parameter   ::= name parameter
5 Datasets    ::= Dataset+
6 Dataset     ::= name database dataset
```

Na prática, a gramática de XSDS é dividida em duas partes: a definição dos *datasources* e a definição dos *datasets*. A primeira parte refere-se aos recursos físicos, ou seja, define-se quais fontes reais de informação serão usadas para a obtenção de dados; a segunda parte refere-se a quais campos de dados das fontes de informação devem ser extraídos, usando a linguagem de query de cada fonte em questão. Assim, pode-se dizer que a partir de um mesmo *datasource*, podem ser construídos vários *datasets*.
A definição da arquitetura do extrator foi idealizada para suportar extensão a diversos recursos de informação como fontes de dados. Para isso, essa arquitetura baseia-se no conceito de *drivers* de extração.

3.3.1 Especificação dos Datasources e Datasets em XSDS:

Os *datasources* definem a localização física do recurso de informação. A declaração de cada uma das fontes de informação é feita no elemento <*datasource*>. Este elemento possui um atributo, chamado *extractorDriver* que indica qual *driver* de extração será utilizado: de acordo com o tipo de fonte de informação. Por exemplo: no caso de uma base de dados, além da localização da mesma, são passados parâmetros como o usuário e a senha a ser utilizada nesta base de dados, juntamente com o *driver* de extração que fará este processo; no caso da fonte de informação ser um documento XML, é necessário apenas o nome do arquivo com o seu caminho na árvore de diretórios do sistema operacional.
Para cada conjunto de dados dos recursos de informação (*datasets*) que se queira mapear a partir dos recursos de informação, é necessária a declaração do elemento <*dataset*>. Neste elemento, é necessário indicar qual fonte de dados provém os dados para a construção do *dataset* em questão.
O conteúdo do elemento <*dataset*> é uma expressão na linguagem de consulta referente ao tipo da fonte de informação. Caso esta fonte seja uma base de dados, o conteúdo deste elemento será uma expressão SQL para recuperar os dados referentes ao *dataset* em questão. Se a fonte de informação é um documento XML, o conteúdo deste elemento será uma expressão XPath, indicando o caminho para a informação deste *dataset*.

3.4 O Extrator DS2DS

O *Extrator DS2DS* (*DataSource to DataSet*) é um processador que extrai dados de recursos de informação e faz a criação dos *datasets*, de acordo com a especificação XSDS. Este componente processa uma especificação XSDS, a

qual especifica a fonte dos dados a serem extraídos (*datasources*) e o destino das informações extraídas, as quais definem a representação intermediária (*datasets*).
Esta representação intermediária é composta por um conjunto de tabelas que contém a informação extraída dos *datasources*. Estas tabelas contém somente os dados especificados em XSDS, nos elementos <*datasets*>. Cada um destes elementos será responsável pela geração de uma tabela, a qual será preenchida pelos dados extraídos das fontes de informação. Estas tabelas são armazenadas temporariamente em uma estrutura de dados uniforme, de onde fornecerão os dados para a montagem do *topic map* pelo Processador de XS4TM. Assim, os dados que outrora estavam em formatos heterogêneos, estão agora representados em uma estrutura homogênea.
O *Extrator DS2DS* recorre a *drivers* de extração que, como dito anteriormente, são os módulos responsáveis por ir buscar os dados às fontes de informação; portanto, há um *driver* desenvolvido para cada tipo de recurso de informação, garantindo assim a independência dos módulos fixos do sistema em relação às diversas fontes. Atualmente, o protótipo do *Oveia* possui dois drivers implementados: para conexão com bases de dados relacionais; e para a recuperação de informação em documentos XML.
O funcionamento de um *driver* de extração pode ser descrito como uma sequência de passos, os quais iniciam com a conexão ao recurso de informação físico (para bases de dados relacionais, a conexão é feita via JDBC, *Java DataBase Connectivity* [10] – o que assegura o acesso à quase totalidade das bases de dados atuais – e para documentos XML o acesso é realizado através de um *parser* XML). Posteriormente, o *driver* lê os dados desta fonte de acordo com as tabelas/campos ou elementos especificados em XSDS; esta extração é realizada através da linguagem de consulta da referida fonte (o *driver* utiliza SQL para consultar bases de dados e XSL para documentos XML). Por fim, os dados extraídos são armazenados em memória na estrutura de dados designada por *datasets*, descrita em 3.3.1.
A implementação de novos *drivers* para outros recursos de informação é um processo relativamente fácil e pode ser realizado conforme a necessidade. O desenvolvimento de um *driver* depende apenas do mapeamento da estrutura na qual os dados estão armazenados no recurso de informação para os *datasets*. Este *driver* deve incluir mecanismos para o acesso aos dados através da linguagem de consulta referente a este recurso.

3.5 XS4TM: Uma linguagem XML para especificar a extração de Topic Maps

A linguagem XSTM (*XML Specification for Topic Maps*), proposta em [8], foi inicialmente definida como sendo um dialeto XML para especificar o *topic map* que se pretende construir ao analisar documentos anotados pertencentes a um mesmo esquema XML. Por essa definição, o XSTM está diretamente ligado a extrações a partir de documentos XML. Por outro lado, a necessidade de abranger novas fontes de dados fez com que se propusesse uma nova arquitetura para extração de ontologias. Dessa forma tornou-se necessário repensar e redesenhar o XSTM; o qual passa a ser denominado por XS4TM.
Cada especificação XS4TM é uma instância XML. Portanto, na prática a linguagem XS4TM é definida por um DTD (e/ou um XML-Schema), de modo a permitir o uso de todos os ambientes de processamento XML.
A linguagem XS4TM tem por objetivo tornar a especificação da extração de *Topic Maps* mais completa e flexível. XS4TM é caracterizado por transformar o atual padrão XTM em um subconjunto da sua especificação.
Formalmente, XS4TM é representado pela *Context Free Grammar* abaixo:

```
1  XS4TM           ::= Ontologies Instances
2  Ontologies      ::= (Topic | Association)*
3  Instances       ::= (Topic | Association)*
4  Topic           ::= id (InstanceOf* SubjectIdentity? (BaseName | Occurrence)*)
5  InstanceOf      ::= id (TopicRef | ResourceRef | SubjectIndicatorRef)
6  SubjectIdentity ::= id (TopicRef | ResourceRef | SubjectIndicatorRef)*
7  TopicRef        ::= id xlink:type xlink:href
8  SubjectIndicatorRef ::= id xlink:type xlink:href
9  BaseName        ::= id (InstanceOf? Scope? BaseNameString Variant*)
10 BaseNameString  ::= id baseNameString
11 Variant         ::= id (Parameters VariantName? Variant*)
12 VariantName     ::= id (ResourceRef | ResourceData)
13 Parameters      ::= id (TopicRef | ResourceRef | SubjectIndicatorRef)+
14 Occurrence      ::= id (InstanceOf? Scope? (ResourceRef | ResourceData))
15 ResourceRef     ::= id xlink:type xlink:href
16 ResourceData    ::= id resourceData
17 Association     ::= id (InstanceOf? Scope? Member+)
18 Member          ::= id (RoleSpec? (TopicRef | ResourceRef | SubjectIndicatorRef)*)
19 RoleSpec        ::= id (TopicRef | ResourceRef | SubjectIndicatorRef)
20 Scope           ::= id (TopicRef | ResourceRef | SubjectIndicatorRef)+
```

A especificação XS4TM é subdividida em duas partes distintas (ambas seguem o esquema de XTM 1.0 DTD):

- Na primeira parte são declarados os elementos responsáveis pela definição da ontologia, como os tipos de tópicos, os tipos de associações, ou qualquer outra definição de acordo com o modelo XTM que possa ser utilizada para expressar a estrutura de conhecimento a ser extraída – ou seja, a hierarquia de tópicos, relações super-tipo/sub-tipo e associações entre os tópicos;

- A segunda parte representa as instâncias de tópicos e associações. Nesse momento, os *Datasets* serão utilizados para expressar quais recursos de informação fornecerão metadados para a construção de tópicos e associações.

A figura 3 mostra uma referência da especificação XS4TM para um elemento do *dataset DS_aluno* declarado em XSDS.

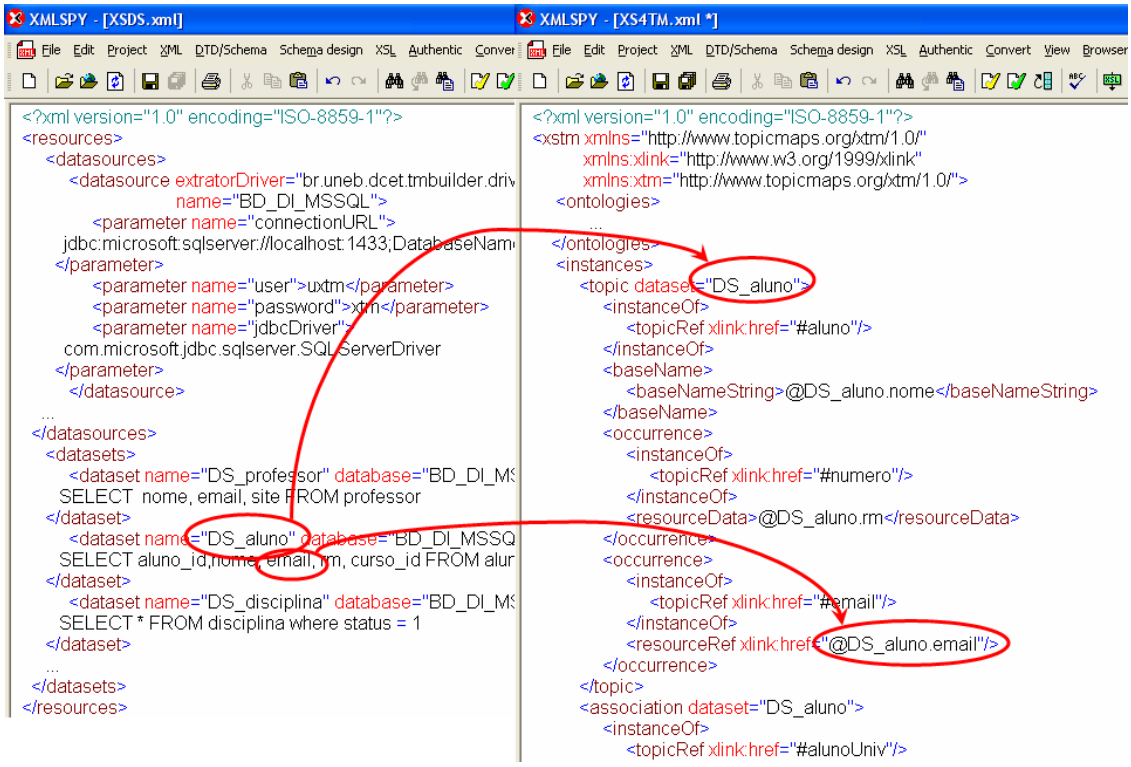

Figure 3: Relações entre XSDS e XS4TM

Para o preenchimento das informações referentes a cada tópico, é necessário buscar tal informação no *dataset* que a contém. Assim, identifica-se essas propriedades com a expressão:

@ + ”dataset” + ”.” + ”atributo”

Mais detalhadamente, isto significa:

- O @ apenas indica que esta declaração é referente a uma propriedade de um *dataset*;
- Após o @, encontra-se o identificador do *dataset* (especificado em XSDS) ao qual deseja-se recuperar a informação. No exemplo da figura 3, o *dataset* selecionado é o *DS_aluno*.
- O *atributo* é uma referência ao campo do *dataset* que contém a informação desejada. Na figura 3, o atributo recuperado para a construção da ocorrência do tipo *email* é o campo *email* extraído pelo *dataset DS_aluno*.

Desta forma, cria-se uma forma de habilitar o uso das informações contidas nos *datasets*.

3.6 Processador de XS4TM

Este componente utiliza a especificação XS4TM para selecionar quais campos dos *datasets*, extraídos dos recursos de informação, são necessários para a formação do *topic map*. Este processador é um interpretador que tira vantagem da organização das informações em um formato uniforme.
O seu processo de execução pode ser resumido em três passos: (1) ler a especificação XS4TM e selecionar os dados que encontram-se nos *datasets*, os quais serão úteis para o *topic map* em questão; (2) criar o *topic map* baseado na própria especificação XS4TM; (3) armazenar o *topic map* gerado na *BD Ontologia* ou em um documento no formato XTM.
Referente ao passo 3, para o armazenamento de *Topic Maps* na *BD Ontologia*, é inicialmente necessário que a estrutura da mesma seja criada. Para isso, o *Oveia* inclui uma série de *scripts* em SQL (uma para cada base de dados relacional suportada: MySQL, SQL Server 2000, Oracle, DB2, Sybase e Postgres), as quais são responsáveis pela criação das

tabelas e relacionamentos na base de dados escolhida como repositório para a *BD Ontologia*. A partir do momento em que a estrutura estiver criada, o *Oveia* utiliza conexão via JDBC para o armazenamento do *topic map* na referida base de dados.

3.7 Base de Dados de Ontologias

Um dos diferenciais desta ferramenta é o armazenamento dos *Topic Maps* extraídos em uma base de dados relacional. De acordo com [17], referente aos métodos de mapeamento de documentos XML para o modelo relacional, adotou-se na *BD Ontologia* o modelo de mapeamento por estrutura.
Conforme o mapeamento por estrutura, foi criada uma tabela para cada elemento de XTM 1.0 DTD. Esse processo consiste em identificar as características e os tipos de associações entre os elementos do DTD e representá-los no modelo relacional.
Esse mapeamento pode ser entendido facilmente tomando-se, como exemplo, o segmento do XTM 1.0 DTD, que apresenta a definição dos elementos <*topic*>, <*baseName*>, <*subjectIdentity*> e <*baseName*>. A definição do primeiro elemento indica que ele é composto pelos elementos subsequentes (além do elemento <*occurrence*>, o qual não será mencionado aqui).

```
<!ELEMENT topic (instanceOf*, subjectIdentity?, (baseName | occurrence)*)>
<!ATTLIST topic
    id ID #REQUIRED
>
<!ELEMENT instanceOf (topicRef | resourceRef | subjectIndicatorRef)>
<!ATTLIST instanceOf
    id ID #IMPLIED
>
<!ELEMENT subjectIdentity (topicRef | resourceRef | subjectIndicatorRef)*>
<!ATTLIST subjectIdentity
    id ID #IMPLIED
>
<!ELEMENT baseName (scope?, baseNameString, variant*)>
<!ATTLIST baseName
    id ID #IMPLIED
>
```

A figura 4 representa o diagrama entidade-relacionamento correspondente ao DTD em causa: para cada um dos quatro elementos do DTD foi criada uma tabela, acrescentando-se ainda uma tabela de ligação. Por exemplo, a tabela *topic* (que contém todos os tópicos) tem uma chave estrangeira indicando qual o *subjectIdentity* associado ao tópico em questão. Também pode ser visto que a tabela *baseName* possui uma chave estrangeira para indicar o tópico a que este nome diz respeito. Por fim, a tabela *instanceOfTopic* relaciona as instâncias com o seu tipo (que também é um tópico), isto porque uma instância pode pertencer a vários tipos de tópicos, assim como um tipo de tópico pode ter várias instâncias (relação M:N). Portanto, esta tabela armazena as chaves primárias das tabelas *topic* e *instanceOf*, para cada par *topic – instanceOf*.

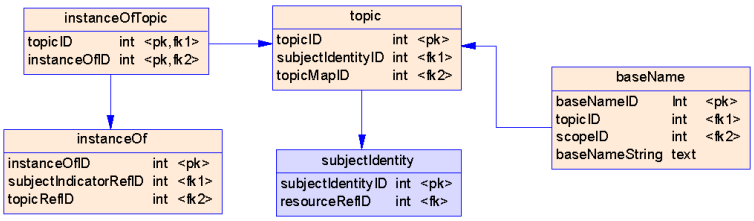

Figure 4: Trecho do Modelo ER do BD Ontologia

A facilidade de compreensão do modelo ER adotado é garantida principalmente pelo fato de que este modelo segue o padrão XTM, o qual é bastante conhecido pela comunidade acadêmica. Essa foi umas das vantagens trazidas por essa opção de modelagem, preservando o padrão *Topic Maps*. Assim, a partir dessa base de dados, é possível navegar no *Topic Maps* utilizando consultas SQL.

3.8 Caso de estudo: um departamento acadêmico

Para ilustrar as idéias até aqui apresentadas, e também para descrever o processo de construção de Topic Maps, foi desenvolvido um caso de estudo onde o tema principal é um departamento de uma universidade.
O departamento em questão possui uma base de dados MySQL contendo toda a informação sobre si. Esta base de dados tem 44 tabelas representando os membros, as categorias dos membros, as publicações, os projetos, os cursos, etc.

A especificação XSDS que descreve esta base de dados (um *datasourse*) e seus *datasets* possui 115 linhas. A especificação XS4TM para este caso possui 1562 linhas para descrever uma ontologia composta por 109 tópicos e instâncias definidas à custa de 53 tópicos e 46 associações.
O resultado final foi um topic map (armazenado na *OntologyDB*) com um total de 4420 tópicos e 4223 associações. Ao converter-se esta *OntologyDB* para o formato XTM, obteve-se um documento XTM com 172490 linhas.
A construção do topic map final foi concluída em 9:23 minutos. O tempo médio para cada elemento (tópico ou associação) foi de 0,0651 segundos. Esta extração foi executada no protótipo do *Oveia*, desenvolvido em Java. O ambiente o qual foi executada esta extração é composto de um PC Intel Pentium II, com 600MHz e 512MB Ram. Todos os serviços, tais como base de dados MySQL (fonte de informação), base de dados SQL Server 2000 (para a *OntologyDB*) e o protótipo do *Oveia* foram executados no mesmo computador.

4 Trabalhos Relacionados

Vários projetos tem objetivos similares ao *Oveia*. O TSIMMIS [1] é um projeto que objetiva prover ferramentas para acessar, de uma maneira integrada, múltiplos recursos de informação e garantir que a informação obtida é consistente. O TSIMMIS foi desenvolvido para extrair propriedades de objetos não-estruturados, permitindo a combinação entre várias fontes e a navegação nesta informação; ele fornece uma visão centralizada da informação que esta dispersa nos recursos de informação. O *Oveia* foi desenvolvido para permitir uma navegação conceitual sobre recursos heterogêneos de informação. Essa navegação conceitual é obtida por uma ontologia, representada de acordo com a norma *Topic Maps*, criada a partir de dados extraídos dos sistemas de informação.
O KAON[1] [5] é um projeto *open-source* que fornece uma infra-estrutura para gestão de ontologias, voltado para aplicações de negócios. A ferramenta KAON REVERSE é um *plug-in* do *framework* KAON. O KAON REVERSE permite o mapeamento de bases de dados relacionais para uma ontologia, com o objetivo de extrair instâncias e relacionamentos entre instâncias, a partir da base de dados. Entre as ferramentas conhecidas, esta é a que mais se aproxima do *Oveia*.
A tabela 1 mostra as características e funcionalidades do *Oveia* e do KAON REVERSE[2].

	KAON REVERSE	OVEIA
Linguagem	Java	Java
Uso de APIs	Sim	Sim
Uso de Engenharia Reversa	Sim	Não
Especificação	Árvore(GUI)	Documento XML
Fontes de Extração de Ontologias	BDs relacionais via JDBC	BDs relacionais via JDBC, XML, extensível a outras fontes
Padrão de Representação de Ontologias	RDF	Topic Maps
GUI (Interface Gráfica)	Sim	Não
Resultado Gerado	Documento RDF	Base de Dados de Ontologias

Table 1: Comparativo entre KAON REVERSE e Oveia.

De acordo com a tabela 1, é perceptível as vantagens de cada ferramenta em pontos distintos. Seria mais sensato dizer que existe uma certa tendência à união das funcionalidades, do que comparar qual seria a melhor ferramenta. Partindo deste ponto de vista, destaca-se as vantagens de cada uma. Por um lado, a KAON REVERSE apresenta vantagens em relação ao uso de interface gráfica para a especificação de ontologias e o uso de engenharia reversa dos recursos de informação para auxiliar o mapeamento. Por outro lado, o *Oveia* se destaca por ser mais flexível em relação aos recursos passíveis de extração e em relação ao processo de especificação. Além disso, o *Oveia* se diferencia por gerar uma base de dados de ontologias capaz de manter os dados extraídos.
Em relação ao padrão de representação, as duas ferramentas possuem focos diferentes. Enquanto uma faz uso de RDF, a outra utiliza *Topic Maps*. A opção pelo paradigma *Topic Maps* foi tomada, principalmente, por seu poder de representatividade ser superior ao do RDF e permitir a criação de navegadores conceituais.
Ao fazer uso de uma linguagem de especificação semelhante ao padrão *Topic Maps* (XS4TM), o *Oveia* flexibiliza o processo de extração para o projetista da ontologia. Esta vantagem é considerável quando se refere ao processo de criação de ontologias, visto que este é diretamente ligado a sua finalidade. Ou seja, para cada caso, poderá ser criada uma visão diferente da fonte de dados, expressando assim uma estrutura de conhecimento específica sobre um determinado domínio.

[1]Mais informações em: http://kaon.semanticweb.org/
[2]Os dados presentes nesta tabela foram utilizados a partir da análise feita em [15].

Um outro extrator de ontologias que pode ser referido é o *OntoExtract* [16]. O *OntoExtract* extrai ontologias a partir do esquema de bases de dados relacionais. Este extrator apresenta a proposta de definir o RDF como linguagem base para definição de objetos de ontologias, como uma espécie de camada de modelo; enquanto que o XTM seria utilizado como uma linguagem de definição de ontologias, uma (meta)ontologia.
Assim como o KAON REVERSE, o *OntoExtract* baseia-se na extração exclusiva em bases de dados a partir de engenharia reversa do esquema relacional. Limitando, assim, a faixa de recursos de informação passíveis de extração.

5 Conclusão

O objetivo deste artigo foi a apresentação de uma arquitetura para a construção automática de *Topic Maps*, para extração de informação de bases de dados ou documentos XML. Esse sistema, designado por *Oveia*, resultou de uma proposta inicial denominada *TM-Builder*, o qual é um extrator de ontologias a partir das fontes XML totalmente escrito em XML/XSL.
A extração de informação de fontes heterogêneas de informação é especificada pela linguagem XS4TM, a qual define quais os conceitos e relações encontradas nestas fontes de informação que serão mapeadas para tópicos e associações, respectivamente, no *topic map* gerado pelo *Oveia*. Para funcionar, o sistema requer que todas as fontes estejam descritas na linguagem XSDS, de modo a ser convertidos para uma representação interna uniforme, os *Datasets*.
XS4TM é a linguagem para especificar a extração de *Topic Maps* a partir de recursos de informação. XS4TM classifica os tópicos, dando-lhes uma semântica mais concreta, através da associação de um tipo de tópico, um tipo de associação ou um tipo de papel de atuação em associações (valores encontrados nas fontes).
O *Oveia* é uma evolução do *TM-Builder*, o qual estava limitado ao processamento de documentos XML. A fim de evitar esse problema, foi construída uma nova arquitetura que fornece uma abstração dos tipos de fontes de dados baseada em *drivers* de extração. O resultado obtido foi a introdução de uma representação intermediária, os referidos *Datasets* que asseguram independência da fonte de dados através da criação de uma nova camada de abstração, o que torna possível a extração em fontes de dados diversas, de forma transparente e em uma única especificação.
A principal vantagem desta proposta é a possibilidade de adaptação do processo de especificação e extração de ontologias a um maior número de casos reais, sem acarretar mudanças de formato dos recursos de informação. Claramente não se está aqui a afirmar a supremacia da nova solução, *Oveia*, em relação à anterior, quer porque ainda não foram realizados testes comparativos, quer porque à partida a primeira solução tecnológica (*TM-Builder*) se afigura com um desempenho superior.
Outra importante vantagem surge a nível de manutenção dos *topic maps*: modificações nas fontes de informação (obviamente mudanças ao nível de seu conteúdo, e não estruturais – incluindo-se novos dados ou excluindo-os), não implicam uma modificação da especificação XS4TM; basta voltar a executar o processo de extração sobre a fonte de informação alterada para obter um novo mapa de tópicos atualizado.
Um dos projetos a ser desenvolvido em breve será um módulo que permita a conversão de um documento XTM para uma *BD Ontologia*, assim como possibilite a extração de um *topic map* da *BD Ontologia* para um documento XTM. Com este módulo, o utilizador pode rapidamente ter um conjunto de *topic maps* armazenados no formato desejado, seja em documentos XML (no padrão XTM) ou em base de dados relacionais (em uma *BD Ontologia*).
Uma bateria de testes será realizada em um futuro próximo para comprovar a eficiência do *Oveia*. O estudo de desempenho do sistema será realizado medindo os tempos e memória consumidos em situações de carga diversificadas, variando-se o número e o tipo das fontes. Outras medições serão realizadas para analisar o comportamento do protótipo quando confrontado com especificações XSDS e XS4TM de diversos tamanhos e complexidades, para um conjunto fixo de fontes de informação.
Até o presente momento, não foi realizada uma validação formal dos módulos do *Oveia*; em contrapartida, validações pragmáticas provaram que os topic maps gerados pelo sistema condizem com as ontologias especificadas em XS4TM e com os dados encontrados nas fontes de informação. No âmbito deste projeto, o passo seguinte consistirá em conceber um bancada de teste que permita sistematizar a comparação do *input* (fontes + especificação) com o *output* (topic map), a fim de comprovar que a ontologia gerada é fiel à especificação e às fontes. Uma validação formal dos módulos do sistema, se bem que complexa, seria a maneira mais segura de comprovar matematicamente a correção da proposta apresentada neste artigo.
Uma metodologia que será aplicada para verificar a correção do topic map gerado pelo *Oveia* será o uso de uma linguagem para especificação de restrições (*constraints*) associadas ao esquema de Topic Maps, a qual será descrita em [9]. Esta linguagem – XTCHE – permitirá a especificação de restrições sobre a ontologia representada em um topic map, além de permitir a definição de esquemas para topic maps.
O *Oveia* não possui um ambiente amigável para o utilizador criar suas especificações. Aparentemente o processo de criação de um documento XS4TM ou XSDS mostra-se trabalhoso. Isso exige que o utilizador conheça a linguagem de descrição dos documentos de especificação, que é o padrão XTM. É sabido que esta tarefa pode ser auxiliada por ferramentas de edição de documentos XML, como por exemplo o *XMLSpy*[3], porém essa tarefa de criar uma interface

[3]Ferramenta para construção de documentos XML, desenvolvido pela ALTOVA. Mais informações em: http://www.altova.com

de especificação agradável e fácil de usar será um outro tópico de investigação futura.

References

[1] TSIMMIS – The Stanford-IBM Manager of Multiple Information Sources, April, 1998. http://www-db.stanford.edu/tsimmis/tsimmis.html.

[2] N.J. Belkin and B.W. Croft. Information filtering and information retrieval: Two sides of the same coin? In *Communications of the ACM*, volume 35(12), pages 29–38, http://www-agki.tzi.de/buster/IJCAIwp/Finals/wache.pdf, December 1992. ACM.

[3] Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December, 1999. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf.

[4] Cheng Hian Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources*. PhD thesis, Phd, MIT, 1997.

[5] S. Handschuh, A. Maedche, L. Stojanovic, and R. Volz. KAON, 2001. http://kaon.semanticWeb.org/kaon/white-paper.pdf.

[6] V. Kashyap and A. Sheth. Schematic and semantic semilarities between database objects: A context-based approach. In *The International Journal on Very Large Data Bases*, volume 5(4), pages 276–304. VLDB Journal, 1996.

[7] Won Kim and Jungyun Seo. Classifying schematic and data heterogeinity in multidatabase systems. In *IEEE Computer*, volume 24(12), pages 12–18. IEEE, 1991.

[8] Giovani Librelotto, José C. Ramalho, and Pedro R. Henriques. TM-Builder: Um Construtor de Ontologias baseado em Topic Maps. In *XXIX Conferencia Latinoamericana de Informtica*, La Paz, Bolívia, 2003.

[9] Giovani Rubert Librelotto, José Carlos Ramalho, and Pedro Rangel Henriques. XTCHE - A Topic Maps Schema and Constraint Language. In *XML 2004 Conference and Exposition*, Washington D.C., U.S.A, 2004. IDEAlliance. (to be published).

[10] Kevin Mukhar, Todd Lauinger, and John Carnell. *Beginning Java Databases: JDBC, SQL, J2EE, EJB, JSP, XML*, volume ISBN 1861004370. Wrox Press Ltd, 2001.

[11] Steven R. Newcomb. A Semantic Integration Methodology. In *Extreme Markup Languages 2003: Proceedings*. IDEAlliance, 2003. http://www.idealliance.org/papers/extreme03/html/2003/Newcomb01/EML2003Newcomb01.html.

[12] J. Park and S. Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.

[13] Steve Pepper. The TAO of Topic Maps - finding the way in the age of infoglut. Ontopia, 2000. http://www.ontopia.net/topicmaps/materials/tao.html.

[14] M. Uschold and M. Grniger. Ontologies: Principles, methods and applications. In *Knowledge Engineering Review*, volume 11(2), pages 93–155. Cambridge University Press, 1996.

[15] André Accioly Vieira. Extra o de Ontologias a partir de Esquemas Relacionais, 2002. Instituto Militar de Engenharia. Mestrado em Sistemas e Computação.

[16] André Accioly Vieira, Astério Kiyoshi Tanaka, and Ana Maria de Carvalho Moura. OntoExtract – Ferramenta para Extração de Ontologias a partir de Bancos de Dados Relacionais. *WTDBD/SBBD/2002*, 2002.

[17] Kevin Williams, Michael Brundage, Patrick Dengler, Jeff Gabriel, Andy Hoskinson, Michael Kay, Thomas Maxwell, Marcelo Ochoa, Johnny PaPa, and Mohan Vanmane. *Professional XML Databases*. Wrox Press, 2000.