TM-Builder: Um Construtor de Ontologias baseado em Topic Maps

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 tem como objetivo introduzir uma arquitetura genérica de um extrator de ontologia (*Ontology Builder*) a partir de uma família de documentos XML. Este Ontology Builder é obtido através de uma especificação de ontologias. Após, é apresentada uma instância da referida arquitetura, a qual ao processar uma família de documentos XML, gera uma especificação em XTM (*XML Topic Maps*). Para descrever este processo de extração do conhecimento dos documentos XML, é definida uma nova linguagem XML chamada XSTM (*XML Specification for Topic Maps*). A arquitetura e a linguagem propostas são ilustradas por meio de um estudo de um caso real: a especificação da ontologia ligada a autores e artigos apresentada em uma conferência sobre XML (XATA) realizada na Universidade do Minho.

Palavras chaves: XML, Semantic Web, Topic Maps, Ontologia, XSL.

Abstract

This paper aims at introducing a generic architecture of a ontology extractor (*Ontology Builder*) from instances of a family of XML documents. This Ontology Builder comes from an ontology specification. After, it presents an instance of this architecture that process a family of XML documents, resulting, as output, a specification in XTM (*XML Topic Maps*). To describe the extraction of knowledge from XML documents to produce a XTM, a new XML language called XSTM (*XML Specification for Topic Maps*) is defined. The proposed architecture and language are illustrated by a real case study: the ontology specification connected to the authors and papers presented in a conference about XML (XATA) realized at University of Minho.

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

*Bolsista CNPq - Brasil

1 Introdução

A cada dia, milhares de novos recursos de informação são disponibilizados na World Wide Web (doravante designada abreviadamente por *Web*). Desta forma, a Web está crescendo de maneira rápida, tornando as tarefas de procura mais complexas. Para minimizar o problema, algumas iniciativas fizeram com que uma nova área de pesquisa e desenvolvimento surgisse: *Semantic Web*.
Quando se refere a Semantic Web [2], fala-se sobre uma rede de conceitos, por oposição a uma rede de documentos. Cada conceito tem um grupo de recursos associados e pode estar relacionado com outros conceitos. Pode-se usar esta rede de conceitos para navegar sobre os recursos da Web, ou simplesmente sobre os recursos de informação. Para organizar esses recursos de informação de forma a permitir uma navegação sobre os conceitos, é necessário um paradigma que tenha a indexação de recursos como característica. Este objetivo, entre outros, é encontrado na definição de Topic Maps [5].
Como forma de dar suporte ao leitor do artigo, esta seção dedica-se a introduzir os conceitos básicos da área abordada: Semantic Web e Ontologias; assim como apresentar também os objetivos deste artigo. A norma XTM (XML Topic Maps) será descrita na seção 2, uma vez que é usado no trabalho que se propõe. Topic Maps é um formalismo para representar conhecimento sobre a estrutura de um recurso de informação e para organizá-la em tópicos. Na seção 3, será apresentado o extrator de ontologias genérico, o *OntBuild*, que é o modelo defendido. A descrição do sistema que propõe-se, o extrator de Topic Maps a partir de documentos XML – TM-Builder – é feita na seção 4. A definição da linguagem XSTM será encontrada na seção 5. Como caso de estudo, a extração da ontologia de um congresso é abordada na seção 6. Por fim, uma síntese do artigo e os trabalhos futuros são apresentados na conclusão.

1.1 Semantic Web

Sob qualquer parâmetro que se queira avaliar, a Web é sem dúvida um dos maiores sucessos na história dos empreendimentos humanos, contando com usuários de todo o mundo, manipulando e acessando uma quantidade sem precedentes de informação.
Enquanto o tamanho e a complexidade da Web aumentam, o mesmo não pode ser dito sobre as tecnologias utilizadas para sua manipulação. A maior parte das tarefas de acessar, extrair, interpretar e manter a informação disponível ainda é deixada a cargo dos usuários.
Os motores de busca são ineficientes quando se trata de fazer inferências complexas e correlacionar assuntos aparentemente disjuntos. A simples anotação de páginas HTML por intermédio das tags <META> ou mesmo o emprego de padrões de metadados não é suficiente para incluir a semântica desejada, que possibilitaria a execução de tarefas mais sofisticadas e mais úteis do que as atualmente existentes.
Na abordagem de Tim Berners-Lee [2], as construções orientadas para entendimento humano levam a limitações e a um tratamento trivial por parte dos computadores, do conteúdo das páginas Web – limita-se a um cabeçalho, links para outras páginas (mas, em geral, as máquinas não possuem uma forma confiável de processar o conteúdo semântico das informações contidas em uma página).
Com base nessas premissas, surgiu a idéia da Semantic Web, na qual o conhecimento do significado de recursos da Web é armazenado por meio da utilização de (meta) dados processáveis por máquinas. Pretende-se que a Semantic Web não seja separada da Web, mas uma extensão da tecnologia corrente. Basicamente, os mecanismos a serem desenvolvidos para o estabelecimento da Semantic Web compreendem duas vertentes: a disponibilização de um conjunto de coleções estruturadas de informações e regras de inferência associadas a esses conjuntos; e a criação de agentes de software capazes de percorrer a Web realizando tarefas complexas com base nessas estruturas de conhecimento.

1.2 Ontologias

Atualmente, os sistemas convencionais de consultas utilizam técnicas de base sintática sobre uma forma de adequação léxica, mais do que uma aplicação da base de conhecimento do campo de interesse. Em muitos domínios, o usuário está interessado em encontrar informação onde a relevância dos documentos não pode ser medida através do uso de sistemas de busca por palavras chaves. Neste contexto, algumas propostas envolvem a criação de metadados que seguem modelos de ontologias.
Uma ontologia [11] é uma especificação explícita de uma conceitualização. Também pode ser entendido como um conjunto de termos hierarquicamente estruturado para a descrição de um domínio o qual pode ser utilizado como um esqueleto fundamental para uma base de conhecimento.
Uma ontologia também pode ser vista como uma teoria lógica para descrever o significado pretendido de um vocabulário formal, isto é, seu comprometimento com uma conceitualização particular do mundo. Estas incluem estruturas que permitem manipular termos de uma forma muito eficiente e útil para o usuário e mecanismos de validação para comunicação inter-programas. A importância de seu uso é devida à capacidade de representar hierarquias de classes de objetos (taxonomias) e seus relacionamentos.

As ontologias colaboram no sentido de se obter uma Web onde os recursos disponíveis são acessíveis não somente por seres humanos, mas também por processos automatizados. Esta automação provoca a elevação do status da Web de *machine-readable* (lida automaticamente) para algo que é chamado de *machine-understandable* (entendida automaticamente). Isto reflete a visão de Berners Lee sobre Semantic Web [2].
A fim de prover o primeiro mecanismo necessário à *Semantic Web*, a anotação da informação em XML (*eXtensible Markup Language*) [10] vem sendo reconhecida como relevante. XML permite representar dados em formato semi-estruturado, o que ocorre com frequência no mundo real. Entretanto, XML por si mesmo, não permite acrescentar significado a tais estruturas. Ao usar XML como sintaxe para transmissão de dados semi-estruturados, a descrição do significado deve ficar a cargo de alguma linguagem de especificação semântica. Esse conjunto coerente de coleções estruturadas de informação forma uma ontologia.
O desenvolvimento de ontologias irá prover o mecanismo de construção da parte semântica da Semantic Web. O modelo em camadas proposto por Berners-Lee [3] tem sido aceito principalmente como representação para a arquitetura da Semantic Web. O desenvolvimento de tais mecanismos depende, obrigatoriamente, de linguagens que expressem a informação de maneira a ser entendida por máquinas. O desafio é proporcionar uma linguagem que manipule igualmente, de maneira eficiente, dados e regras para deduções sobre esses dados e que permita que regras existentes em qualquer sistema de representação de conhecimento possam ser exportadas para a Web.
O desenvolvimento de ontologias deverá representar uma parcela significativa de esforço no desenvolvimento de qualquer aplicação no futuro. Dessa forma, o desenvolvimento de ambientes para construção e manipulação de ontologias é fundamental. Tais ambientes devem ser compostos de um repositório de ontologias que possa ser manipulado por desenvolvedores, usuários e programas de aplicação, permitindo a navegação, pesquisa e reuso de termos. Quando novos termos forem acrescidos à ontologia, o ambiente deve verificar a consistência do repositório.

1.3 Extração de ontologia

A fim de facilitar a criação de ontologias a partir de documentos XML, decidiu-se criar um extrator que retirasse automaticamente uma ontologia a partir da referida coleção de documentos, com base numa especificação que explicita quais os elementos de tais documentos que devem ser retirados e como devem se associar entre si. A esse extrator, chamaremos de *Ontology Builder* (*OntBuild*).
Um *OntBuild* é fortemente dependente da estrutura dos recursos de informação. Trabalhando com vários tipos de documentos XML, é necessário implementar diversos *OntBuilds* (um para cada esquema XML). Para minimizar o esforço envolvido foi criada uma linguagem XML para descrever a extração do conhecimento de documentos XML para produzir uma ontologia. É, assim, apresentada uma proposta de linguagem para especificação de ontologias a partir de um tipo de documentos XML, chamada de XSO (*XML Specification for Ontologies*).
A partir de uma especificação em XSO, é possível gerar, também automaticamente, um *OntBuild* para a família de documentos XML respectiva. Ao processar esta família de documentos no *OntBuild*, obtém-se uma ontologia que deve ser representada em uma linguagem apropriada: RDF (*Resource Description Framework*), DAML+OIL (*DARPA Agent Markup Language + Ontology Inference Layer*) ou XTM (XML Topic Maps).

1.4 Por que usar Topic Maps como padrão?

Em [8] e [9], apresenta-se uma série de diferentes características entre Topic Maps e RDF. A conclusão que se chega é que RDF é designado para prover metadados sobre recursos de informação, enquanto que Topic Maps provém uma visão alto-nível do domínio coberto pelos recursos. Como é justamente este o objetivo do trabalho, a adoção de Topic Maps é totalmente indicada.
Além dos apontamentos citados por Lars Marius Garshol, Steve Pepper [16] acrescenta o fato de que Topic Maps foi desenvolvido para suportar um alto-nível de indexação de conjuntos de recursos de informação para tornar esta informação acessível. RDF, por outro lado, foi projetado para suportar a visão de uma rede semântica provendo metadados estruturados sobre os recursos e uma fundamentação para inferência lógica.
A partir da comparação apresentada, ficou claro perceber que Topic Maps possui um nível mais alto que RDF, no sentido de que um mapa de tópicos contém mais informação sobre si do que um modelo RDF. O uso de DAML+OIL é relacionado com RDF; assim, ao rejeitar RDF, DAML+OIL também é preterido, pelo que ficou escolhido Topic Maps para representar as ontologias geradas pelo *OntBuild*.

2 XML Topic Maps

Um Topic Map é basicamente um documento XML onde diferentes *element types* são usados para representar: Tópicos; Ocorrências de tópicos; e Relacionamentos (ou Associações) entre os tópicos [15].
XML Topic Maps (XTM) [17] 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 os tópicos 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.
Topic Maps pode ser visto como uma descrição de um ponto de vista sobre uma coleção de recursos, organizado formalmente por tópicos e pela ligação de partes relevantes do conjunto de informação aos tópicos apropriados.
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. *Charles Goldfarb* [10] (o pai das linguagens de anotação) geralmente compara Topic Maps com GPS (*Global Positioning System*) aplicado ao universo da informação. Falar sobre Topic Maps é falar sobre estrutura de conhecimento.
Os principais objetivos de XTM são:

- Estruturar recursos de informação não estruturados;
- Permitir procuras que recuperem a informação requisitada;
- Criar visões diferentes para usuários ou finalidades específicas, filtrando a informação.

Permitindo criar 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 maneiras, por diferentes mapas de tópicos. Como é possível e fácil modificar um mapa, a reutilização da informação é conquistada.

2.1 As características do modelo XTM

Tópicos são o ponto principal de XTM [14]. Em um sentido mais genérico, pode ser qualquer coisa: uma pessoa, uma entidade, um conceito. Eles constituem a base para a criação de XTM. Podem ser vistos como um link-múltiplo, o qual aponta para todas as suas ocorrências [4].
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. Os tipos representam as classes onde os tópicos estão agrupadas, i.e., a categoria de cada instância tópico. Pela definição standard, os tipos de tópicos também são tópicos.
Um tópico pode ter um ou mais nomes. A opção de especificar mais de um nome ao tópico pode ser utilizada em diferentes contextos (*scopes*), como idiomas, estilos, domínios, área geográfica, período histórico, etc.
Um tópico pode ter uma ou mais ocorrências. Um ou mais recursos de informações endereçáveis de um tópico constituem o conjunto de ocorrências de tópicos (*Topic Occurrences*). As ocorrências de tópicos podem ser, por exemplo, um artigo sobre um tópico em uma enciclopédia; ou uma imagem ou vídeo descrevendo o tópico; ou qualquer de outras muitas formas nas quais os recursos de informação podem ter alguma relevância a um tópico. As ocorrências podem ser endereçáveis através de uma URI (*Universal Resource Identifier*). Uma ocorrência de tópico representa a informação que é especificada como relevante para um certo assunto [15].
Ocorrências e tópicos existem em duas camadas diferentes, mas elas são ”conectadas” entre si. As ocorrências estabelecem rotas dos os tópicos para os recursos de informação, possibilitando também prover uma razão de o porque que a rota existe. Neste ponto, a separação em duas camadas é percebida: tópicos e suas ocorrências; uma das grandes vantagens de XTM.
Entre todas as ocorrências de um tópico, uma distinção pode ser feita através de subgrupos. Cada subgrupo é definido por um papel de atuação (*role*) em comum. Os papéis de atuação em ocorrências (*occurrence role*) podem ser utilizados para distinguir gráficos de texto, ocorrências principais de ordinárias, menções de definições, etc. Os papéis de atuação em ocorrências são definidos pelos usuários, sendo assim podem variar em cada XTM [4].
O padrão XTM também define papéis de atuação como tópicos. Se um papel de atuação em ocorrência é definida explicitamente como um tópico, XTM facilmente pode ser usado para ceder informações sobre ele (como seus nomes e os relacionamentos aos quais eles participam). Mas, para fazer a real distinção entre diferentes tipos de ocorrências, XTM também utiliza o conceito de tipo de papel de atuação em ocorrência (*occurrence role type*).
Associações (*associations*) são responsáveis pelos relacionamentos entre os tópicos. Elas são ligações independentes da fonte dos documentos onde as ocorrências de tópicos são encontradas; elas representam a base do conhecimento, a qual contem a essência da informação que alguém criou e atualmente representa seu valor essencial. Um ilimitado número de tópicos podem ser relacionados por uma associação.
O poder de XTM aumenta com a criação de associações porque, deste modo, é possível agrupar um conjunto de tópicos que de algum modo são relacionados. Isso é de grande importância ao prover interfaces intuitivas e amigáveis para a navegação de grandes quantidades de informação.
Assim como os tipos de tópicos agrupam vários tópicos e tipos papeis de atuação também suportam várias ocorrências, as associações entre tópicos devem ser agrupadas de acordo com seu tipo de associação (*association type*).
É importante referir que cada tópico que participa em uma associação tem um papel (*association role*) que expressa a sua atuação nessa associação. Os papeis de atuação em associação também são vistos como tópicos no modelo XTM.

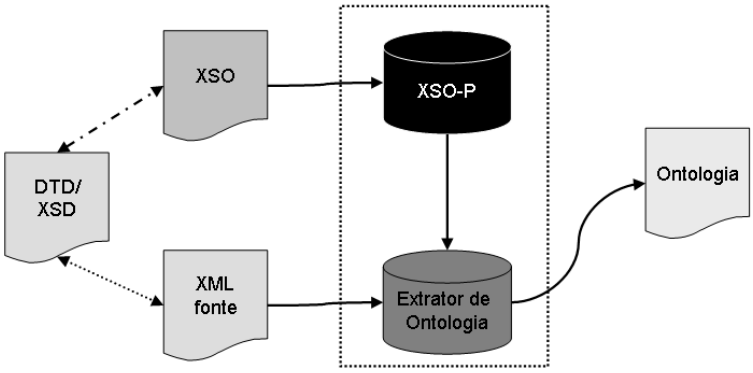

Figure 1: A arquitetura do sistema *OntBuild*

3 *OntBuild*: O Extrator de Ontologias Genérico

A partir da criação manual de algumas ontologias (descritas em XML), verificou-se que tal tarefa de criação, além de consumir tempo, é bastante repetitiva. Com base nesta verificação, decidiu-se desenvolver um extrator de ontologias a partir de um conjunto de documentos XML, isto é, um *Ontology Builder* (*OntBuild*).
Neste contexto, o *OntBuild* é um conversor de uma linguagem XML em outra linguagem XML. O *OntBuild* é uma folha de estilos XSL (eXtensible Stylesheet Language) que recebe um documento XML e gera outro documento XML que contém a especificação de uma ontologia. O *OntBuild* aceita como entrada documentos XML devido aos seguintes fatores:

- XML é, por excelência, a linguagem atual para a anotação de documentos;
- XML é presentemente a plataforma para intercâmbio de informação mais utilizada;
- Outras fontes de dados (não-XML) podem facilmente ser convertidas para XML (adaptando-se, assim, sem problemas para o *OntBuild*).

A maior parte dos atuais sistemas de gestão de base de dados tem facilidades para descarregar o conteúdo das tabelas em formato XML; então, para esses casos (que representam uma grande maioria), um *front-end* já existe. Para os restantes, não é difícil desenvolver tradutores.
A arquitetura do sistema proposto, XSO, pode ser visualizada na Figura 1. Na prática, após o processamento da folha de estilos XSL, uma ontologia será obtida em um documento XML, em uma linguagem de representação a ser definida.
Mesmo recorrendo aos serviços de um extrator, como o *OntBuild*, a tarefa de criar uma ontologia para montar uma Semantic Web é complexa e lenta, pois o extrator depende do tipo de documento XML a processar. Isso significa que será necessário recodificar o *OntBuild* cada vez que a família de documentos a manipular obedeça a um esquema diferente. Esta constatação levou a pensar em unir ao *OntBuild* um gerador automático de *OntBuilders*, formando assim um sistema completo cuja arquitetura genérica é ilustrada na figura 1.
Para poder concretizar esta idéia de gerar o *OntBuild*, é necessário especificar formalmente o processo de extração concreto, para cada tipo de documentos a processar. Isso levou à definição de uma nova linguagem XML: a XSO (*XML Specification for Ontologies*). A linguagem XSO especifica o processo *OntBuild*, habilitando a codificação sistemática (em XSL) da tarefa de extração de ontologia.
Porém, XSO por si só não é suficiente para montar o sistema da figura 1; falta escolher a linguagem de representação de ontologias que o *OntBuild* irá utilizar. Para esta decisão, foram analisadas as características de cada uma das 3 alternativas mais importantes para descrição de ontologias: RDF [6], DAML+OIL [7] e XTM [17] (este estudo comparativo estará contido em outro artigo).
Nessas circunstâncias, é possível gerar automaticamente um extrator de ontologias desenvolvendo outro processador XSL, que tenha a habilidade de transformar as especificações em XSO em especificações na linguagem de representação de ontologias a ser escolhida nas seções posteriores (RDF, DAML+OIL ou XTM).
O processador XSO (XSO-P) é uma das principais peças nesta arquitetura, como pode ser visto na figura 1. A partir de uma especificação em XSO (uma instância XML), tal processador (codificado em XSL) gera uma folha de estilos XSL, a qual que irá processar o conjunto de documentos de entrada XML para extrair a ontologia desejada.

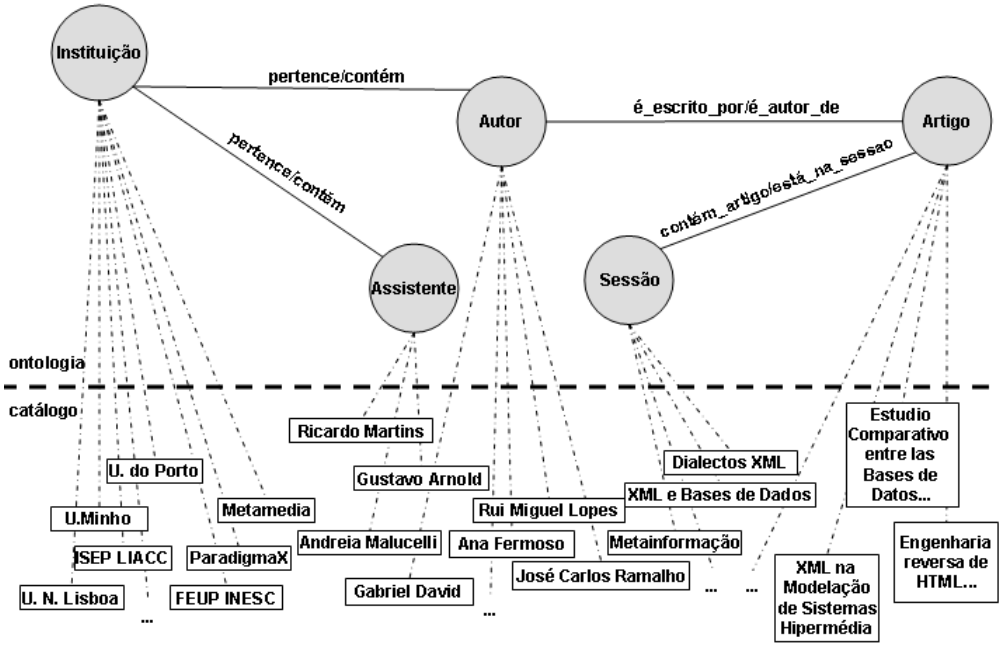

Figure 2: A ontologia e o catálogo do XATA.

4 *TM-Builder*: O Extrator de Topic Maps

Ao analisar XTM, identificam-se duas partes distintas: uma ontologia e um catálogo. A ontologia é definida pelo tipo de tópico, tipo de associação e pelo tipo de papel de atuação em ocorrências. O catálogo é a base de conhecimento associada à ontologia e é composto por 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 2 dá uma representação esquematizada desta visão, usando, como exemplo, os participantes e artigos envolvendo o workshop *XATA* descrito na seção 6.

Há muitas ferramentas para apoio à criação de XTM, como Mapalizer[1]. Contudo, não se tem conhecimento de alguma que através de uma especificação em XML, dos items de informação relevantes, permite criar automaticamente o Topic Map usando apenas ferramentas XML. Há um capítulo sobre *Automated/Automatic Topic Map Construction* em [1], mas não clarifica uma possível implementação do construtor de XTM.

Nesse contexto, entende-se a necessidade de uma linguagem de especificação de *Topic Maps* para permitir a derivação sistemática de um extrator de TM. De acordo com as características de XTM, apresentadas na seção anterior, a arquitetura genérica de *OntBuild* foi então adaptada para permitir a geração de *XML Topic Maps*, como formato de saída. Sendo assim, a linguagem XSO passa a ser denominada por linguagem XSTM (*XML Specification for Topic Maps*), pois a ontologia passa a ser especificada em formato XTM. O extrator – genericamente designado *OntBuild* – passa a denominar-se *TM-Builder*.

Das ferramentas disponíveis, nenhuma é tão simples de instalar e usar como o *TM-Builder*. Muitas dessas ferramentas necessitam de outras tecnologias, softwares, bibliotecas, linguagens de programação ou utilitários para executar suas tarefas. Como o caso do *Omnigator*[13], do *TM4J*[20] e do *Nexist*[19], que necessitam do *Tomcat*[18] instalado; ou da instalação de uma ferramenta *Python*[2] para o uso do *SemanText*[3]; ou ainda de um plug-in para o *Protégé 2000*[12]. Portanto, o *TM-Builder* provém uma ferramenta independente de outra tecnologia, pois depende apenas de um simples *parser* XSL, para efetuar as transformações.

A linguagem XSTM, proposta na seção seguinte, é uma linguagem XML que permite criar um TM-Builder, o qual extrai um topic map de uma família de documentos XML. Essa abordagem oferece um completo framework XML para o usuário. A arquitetura do *TM-Builder* tem então a forma apresentada na Figura 3. Esta arquitetura é uma instância da apresentada na Figura 1, agora com XTM sendo como linguagem alvo para a especificação de ontologias.

[1] `http://www.topicmapping.com/mapalizer`
[2] `http://www.python.org/`
[3] `http://www.semantext.com/`

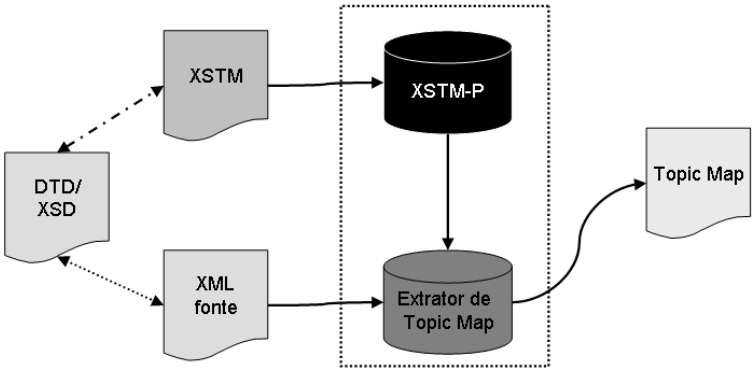

Figure 3: A arquitetura do sistema TM-Builder

5 XSTM: Uma linguagem XML para especificar Extratores de Topic Maps

Conforme a seção anterior, a linguagem XSTM é um dialeto XML para especificar o topic map que se pretende construir ao analisar documentos anotados pertencentes a um mesmo esquema XML.
A linguagem XSTM fornece todos os construtores necessários para especificar a tarefa de extração de Topic Maps. Ela permite a definição dos tópicos, seus tipos e ocorrências, assim como as associações, seus tipos e papeis de ocorrências. De um modo formal, a Gramática Livre de Contexto (GLC) da linguagem é mostrada abaixo.

```
xstm       ::= topic+ topicType+ assoc* assocType*
topic      ::= xpath TTypeID
topicType  ::= TTypeID TTypeName
assoc      ::= assocClass ATypeID LElem RElem
assocClass ::= "one2N" split=("true"|"false")  || "N2N" split=("true"|"false")
               || "one2one" type=("attribute"|"subelement") || all2all
assocType  ::= ATypeID ATypeName LElem RElem
LElem      ::= TTypeID EName? TTypeID
RElem      ::= TTypeID EName? Param? TTypeID
```

Cada especificação XSTM é uma instância XML. Portanto, na prática a linguagem XSTM é definida, não por uma GLC, mas sim por um *Document Type Definition* (DTD), ou um *XML-Schema* (XSD), de modo a permitir o uso de todos os ambientes de processamento XML.

5.1 XSTM-P: O processador XSTM

A linguagem XSTM, previamente definida, especifica o processo que vai ser executado pelo TM-Builder, habilitando uma codificação sistemática (em XSL) desta tarefa de extração. Além disso, é possível gerar automaticamente esse extrator XSL desenvolvendo outro processador XSL (a que vulgarmente é chamado XSL de 2º nível) para traduzir uma especificação XSTM em um código TM-Builder.
O processador de XSTM (XSTM-P) é o gerador de TM-Builders; é uma das peças principais nesta arquitetura, como pode ser vista na figura 3. Ele toma uma instância XML, escrita de acordo com a especificação XSTM, e gera uma folha de estilos XSL que processará o documento XML o qual deseja-se extrair sua ontologia.
Ambas folhas de estilo XSL (o gerador de extrator e o próprio extrator) são processados por um processador XSL standard, como Saxon[4] ou Xalan[5], sendo este mais um dos benefícios desta proposta.

6 Caso de estudo: XATA

A fim de demonstrar o uso do TM-Builder para a extração de ontologias a partir de uma fonte XML, esta seção apresenta o caso de estudo da workshop *XML, Aplicações e Tecnologias Associadas* (*XATA*[6]), que ocorreu na Universidade do Minho, em Braga - Portugal, nos dias 13 e 14 de Fevereiro de 2003, visando reunir a comunidade XML de língua portuguesa. Participaram pesquisadores e utilizadores de XML, tanto oriundos de universidades quanto de empresas, permitindo assim um compartilhamento de informação entre o mundo acadêmico e o mundo profissional.
Nesta workshop, vários artigos foram submetidos para avaliação; e os aprovados foram devidamente apresentados durante a conferência. As apresentações dos artigos foram divididas em sessões, cada qual com um tema associado, como Tecnologia e Web Services, XML e Base de Dados, entre outras.

[4] http://saxon.sourceforge.net/
[5] http://xml.apache.org/xalan-j/
[6] http://www.di.uminho.pt/~jcr/XML/conferencias/xata2003/

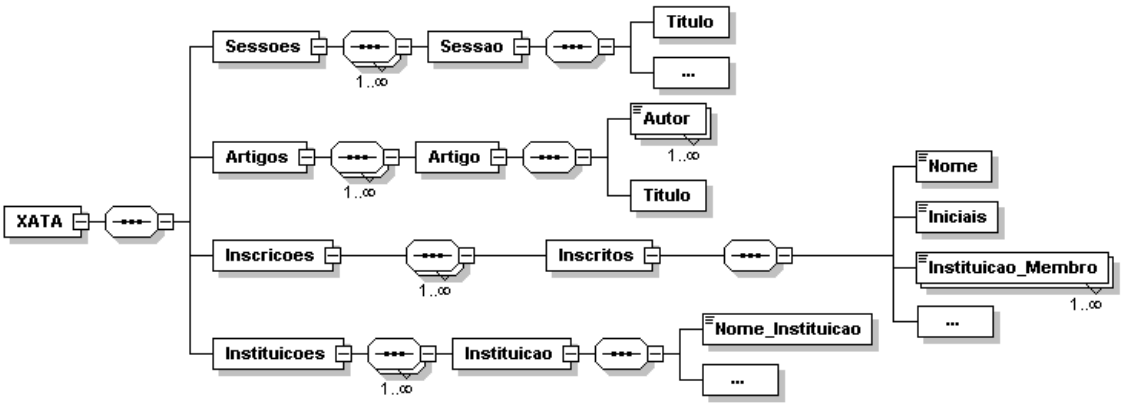

Figure 4: XML-Schema do XATA.

Como não poderia deixar de ser, o evento foi todo baseado em XML, desde sua divulgação, quanto sua produção. Portanto, todas as informações referentes ao *XATA* estão armazenadas em documentos XML. O XML-Schema do evento é apresentado na figura 4. Obviamente este XML-Schema está incompleto, porém, para nosso caso de estudo, o importante está ressaltado nessa figura.
Como a linguagem XSTM depende apenas da estrutura do documento, e não da instância XML, a partir deste XML-Schema é possível definir a especificação da ontologia do XATA. Portanto, são cinco etapas que devem ser seguidas: definição dos tipos de tópicos, dos próprios tópicos, dos papeis de atuação em ocorrências, dos tipos de associação e, por fim, das próprias associações.

6.1 A especificação XSTM para o XATA

O elemento raiz do XML-Schema de XSTM é *xstm*, o qual possui quatro sub-elementos, cada um referente a uma parte da ontologia expressada por XTM. Os seus sub-elementos são: *topicType*, *topic*, *assocType* e *assoc*.
Inicialmente, são definidos os tipos de tópicos. Nesta ontologia, os tópicos são agrupados em Instituição, Autor, Assistente, Sessão e Artigo. Em XSTM, os tipos de tópicos são declarados pelo elemento *topicType*, contendo um identificador (*id*) – para ser referenciado em outros momentos na especificação – e um nome (*name*) – para a visualização do XTM em um navegador. Como exemplo, é mostrado abaixo a declaração do tipo de tópico *Artigo*.

```
<xstm>
    <topicTypes>
        <topicType>
            <id>ID-Artigo</id>
            <name>Artigo</name>
        </topicType>
        <topicType>...</topicType>
    </topicTypes>
    ...
</xstm>
```

Enquanto que os tipos de tópicos são conceitos abstratos definidos pela ontologia, tópicos são elementos reais nos documentos XML tomados como entrada. Para sua definição é usado o elemento *topic*, o qual possui dois sub-elementos: o caminho XPath referente ao próprio elemento (*xpath*) e o seu tipo (*type*). A seguir, apresenta-se a especificação XSTM para a definição dos tópicos referentes ao tipo de tópico *Artigo*.

```
    <topics>
        <topic>
            <xpath>//Artigo/Titulo</xpath>
            <type>ID-Artigo</type>
        </topic>
        <topic>...</topic>
        ...
    <topics>
```

Até este ponto, todos os tópicos, e seus respectivos tipos, encontram-se declarados. Mas em XTM, tópicos sem qualquer associação relacionada aos mesmos, possuem pouca funcionalidade. A rede de conhecimento é obtida através das associações entre os tópicos. Várias associações podem ser inferidas a partir do *XATA*; por isso, vamos tomar como exemplo a associação entre os tipos de tópicos *Artigo* e *Autor*.
Uma vez definidos os tópicos e seus tipos, o próximo passo é a definição dos tipos de associação. Ele define o papel de atuação de cada um dos membros das associações. É declarado com o elemento *assocType* que possui um identificador (*id*), um nome (*name*) e os membros deste tipo de associação (*memberAssoc*). Cada membro é definido

por um contexto *scope* – o identificador do papel de atuação de um tópico em uma associação – e a sua respectiva descrição (*description*). Cada um dos papéis de atuação será um tópico, no XTM final gerado. Sendo assim, é visualizado abaixo a especificação do tipo de associação entre *Autor* e *Artigo*.

```
    <assocTypes>
        <assocType>
            <id>ID-autor_artigo</id>
            <name>Autor e Artigo</name>
            <memberAssoc>
                <scope>ID-escrito_por</scope>
                <description>é escrito por</description>
            </memberAssoc>
            <memberAssoc>
                <scope>ID-autor_de</scope>
                <description>é autor</description>
            </memberAssoc>
        </assocType>
        <assocType>...</assocType>
    </assocTypes>
```

Para finalizar a especificação em XSTM, o elemento *assoc* permite a especificação de todas as associações que envolvem dois ou mais tópicos; elas são encontradas e extraídas a partir do documento XML fonte.
Neste âmbito, quando refere-se a relacionamentos entre nodos da árvore XML (elementos e atributos), não está se referindo ao modelo entidades-relacionamento. Portanto, os nomes 1-para-1, 1-para-N, M-para-N e todos-para-todos não têm exatamente o mesmo significado usado na perspectiva tradicional. Neste contexto, há cinco tipos de relacionamentos entre elementos (ou atributos) que são descritos por quatro elementos filhos de *assoc*:

- o elemento *one2one* descreve as associações entre elementos e seus atributos, com o atributo *type* com o valor *attribute*;
- o elemento *one2one* também descreve as associações entre elementos distintos, com o atributo *type* com o valor *subelement*;
- o elemento *one2N* define as associações *um para muitos*;
- o elemento *M2N* define as associações *muitos para muitos*;
- o elemento *all2all* define as associações entre tópicos que estão relacionados através de uma tabela intermediária.

A estrutura dos sub-elementos de *assoc* são muito similares. Cada um dos quatro sub-elementos acima descritos possui o seu tipo (*type*) – o identificador do tipo de associação correspondente – e os membros que fazem parte desta associação (*members*). Os membros possuem dois elementos filhos: *topicAssoc* que identifica o tipo de tópico pertencente a esta associação e *role*, que demonstra o papel de atuação do tópico na atual associação.
O elemento *one2one* expressa relacionamentos que podem ser obtidos a partir de algum elo de ligação entre os tópicos encontrados no documento XML. Como por exemplo, no caso específico da associação entre *Autor* e *Artigo*, os autores de cada artigos podem ser identificados devido ao conteúdo do caminho XPath *//Artigo/Autor*, o qual é uma referência às iniciais dos autores encontradas em *//Inscritos/Iniciais*. Assim, a associação entre os tipos de tópicos *Autor* e *Artigo*, referente ao *XATA*, foi especificada da maneira abaixo demonstrada:

```
    <assocs>
        <one2one type="subelement">
            <type>autor_artigo</type>
            <members11>
                <element>
                    <topicAssoc>Artigo</topicAssoc>
                    <role>eh_escrito_por</role>
                </element>
                <elementRef target="//Artigo/Autor">
                    <topicAssoc id="./Iniciais">Inscritos</topicAssoc>
                    <role>eh_autor</role>
                </elementRef>
            </members11>
        </one2one>
    </assocs>
    ...
</xstm>
```

Na figura 5, é encontrado a visualização do Topic Map referente ao *XATA*, (criado pelo TM-Builder) no Ontopia Omnigator[7]. Este navegador fornece o total acesso à ontologia extraída, permitindo a navegação através dos conceitos definidos na especificação em XSTM.
Esta figura mostra a ontologia que foi descrita em XSTM, pois:

[7] Demonstração online em `http://www.ontopia.net/omnigator/`

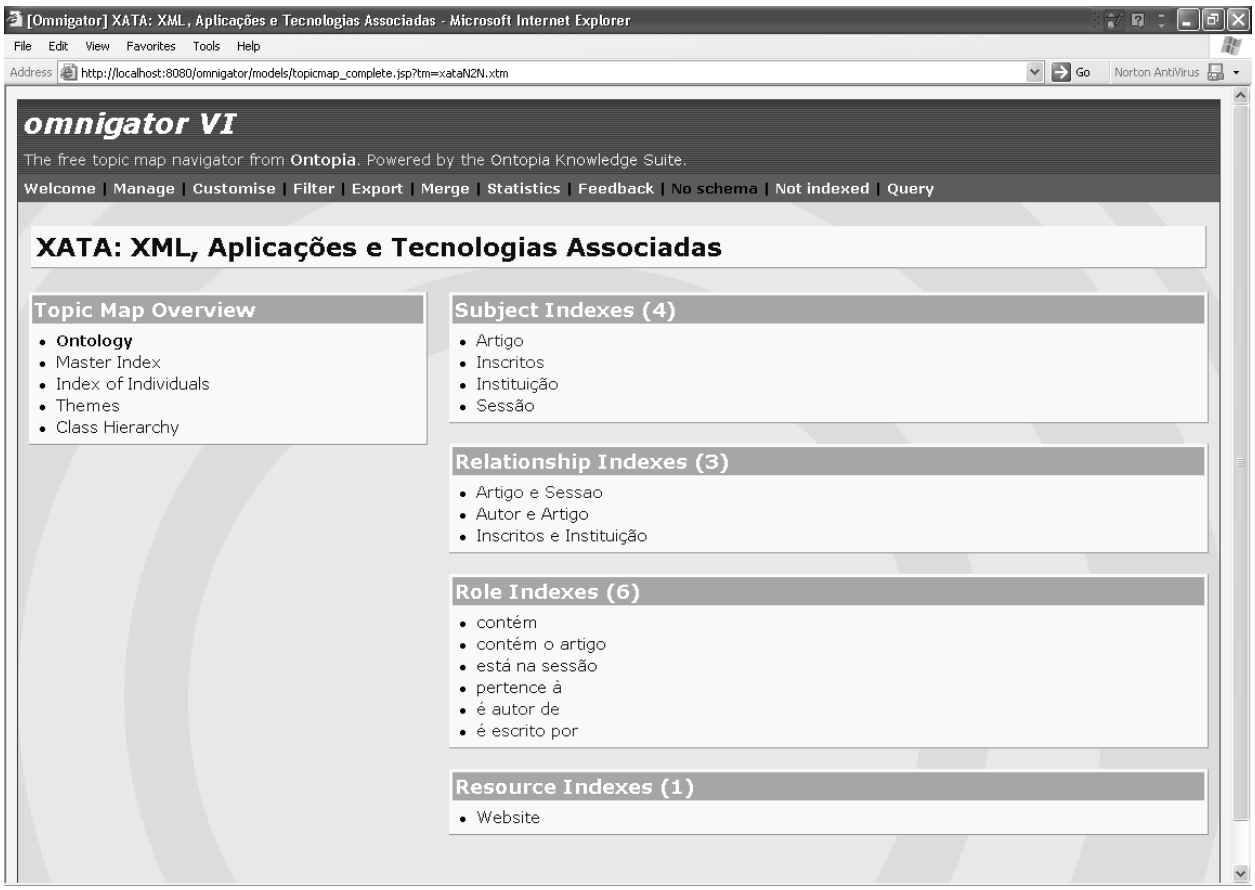

Figure 5: Visualização do Topic Map do XATA no Omnigator.

- no item *Subject Indexes* são encontrados os tipos de tópicos;
- no item *Relationship Indexes* são encontrados as associações;
- no item *Role Indexes* encontram-se os papeis de atuação;
- no item *Resource Indexes* estão os tipos de recursos mapeados pelo TM.

7 Conclusão

O objetivo deste artigo foi a apresentação de uma arquitetura para a construção automática de de topic maps através de folhas de estilo XSL, a partir do processamento de documentos XML de uma mesma família. Esse sistema, todo ele inserido dentro do ambiente XML, foi designado por TM-Builder e resultou de uma proposta genérica, o *OntBuild*, discutido na seção 3.

Em XSTM – linguagem para especificar a extração de taxonomias a partir de documentos XML do mesmo tipo – a definição da ontologia exige o mesmo esforço que em XTM; é necessária a especificação cada um dos tipos de tópicos, dos tipos de associações e dos tipos de papeis de atuação em ocorrências. Contudo, em Topic Maps tudo isto é visto como tópico. XSTM adicionalmente classifica esses tópicos, dando-lhes uma semântica concreta maior, através da associação de um tipo de tópico, um tipo de associação ou um tipo de papel de atuação em ocorrências. Então, do ponto de vista de descrição da ontologia, o ganho é obtido através de se passar a dispor de uma semântica mais precisa. Para o catálogo, a situação é completamente diferente. Na especificação de tópicos e associações usam-se expressões XPath que atuam como consultas. Desta forma, o ganho obtido é igual ao número de ocorrências recuperadas pela expressão de consulta. No caso das associações, o ganho é ainda maior: de N para as relações 1:N e de MxN para as relações M:N.

Para ilustrar todos os conceitos envolvidos e a idéia aqui proposta, estudou-se com cuidado um caso concreto: a construção de uma ontologia associada a toda organização do workshop XATA. Após processar a especificação completa do estudo de caso demonstrado neste trabalho, uma descrição XSTM com 160 linhas, o XSTM-P produziu um TM-Builder (uma folha de estilos XSL) com 413 linhas. Ao extrair os tópicos e associações de um documento XML

com 1139 linhas, o TM-Builder gerado criou um Topic Map com 4017 linhas, com 145 tópicos e 116 associações. Julgamos que esses números são claros indicadores de ganho obtido com o sistema.
O mais interessante desta proposta é que, por mais que aconteçam modificações no documento XML (obviamente modificações a nível de seu conteúdo, e não na sua estrutura), incluindo-se novos artigos, autores, etc, não é necessário modificar a especificação XSTM. Sendo assim, este mesmo TM-Builder pode processar o novo documento XML. Ou seja, pode-se usar o TM-Builder obtido para processar qualquer documento que pertença ao mesmo esquema XML.
Como sequência deste projeto, está em desenvolvimento um navegador conceitual, chamado *DINavigator*, que habilita a navegação conceitual sobre a rede semântica definida pelo XTM. Portanto, este navegador fornece o total acesso à ontologia extraída pelo *TM-Builder*, permitindo a navegação através dos conceitos definidos na especificação em XSTM. O navegador também é baseado na tecnologia XML, desta maneira garante-se a portabilidade e a independência relativamente a plataformas de hardware e software, ao contrário do Omnigator, por exemplo, que depende das ferramentas *Tomcat* e *JDK*.

References

[1] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Joshua Lubell, Miloslav Nic, Daniel Rivers-Moore, Andrew Watt, Rob Worden, and Ann Wrightson. *Professional XML Meta Data*. Wrox Programmer to Programmer Series, 2001.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. In *Scientific American*. http://www.sciam.com/2001/0501issue/0501berners-lee.html, May 2001.

[3] Tim Berners-Lee. W3C – Semantic Web – XML 2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html, 2000.

[4] Michel Biezunski and Steven R. Newcomb. Topic Maps Frequently Asked Questions. www.infoloom.com, September, 1999.

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

[6] World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax Specification, February, 1999. http://www.w3.org/TR/REC-rdf-syntax.

[7] DARPA. Reference description of the DAML+OIL ontology markup language., March, 2001. http://www.daml.org/2001/03/reference/.

[8] Lars Marius Garshol. Topic maps, RDF, DAML, OIL: a Comparison. In *Ontopia*. http://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html, 2002.

[9] Lars Marius Garshol. Living with topic maps and RDF. In *Ontopia*. http://www.ontopia.net/topicmaps/materials/tmrdf.html, 2003.

[10] Charles F. Goldfarb and Paul Prescod. *XML Handbook*. Prentice Hall, 4th edition, 2001.

[11] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[12] Stanford University School of Medicine. Welcome to the Protégé Project, 2003. http://protege.stanford.edu/.

[13] Ontopia. The Ontopia Omnigator, 2002. Online demonstration at http://www.ontopia.net/omnigator/.

[14] Jack Park, Sam Hunting, and Douglas C. Engelbart. *XML Topic Maps: Creating and Using Topic Maps for the Web*. Prentice Hall, 2003.

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

[16] Steve Pepper. Ten Thesis on Topic Maps and RDF. http://www.ontopia.net/topicmaps/materials/rdf.html, Agosto, 2002.

[17] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. http://www.topicmaps.org/xtm/1.0/, Mar, 2001. TopicMaps.Org Specification.

[18] The Apache Jakarta Project. Apache Tomcat, 2003. http://jakarta.apache.org/tomcat/.

[19] SourceForge. Nexist, 2003. http://nexist.sourceforge.net/.

[20] SourceForge. Topic Maps for Java - TM4J, 2003. http://tm4j.org/.