TM-Builder: An Ontology Builder based on XML 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

Abstract

Everyday a huge number of new information resources are linked to the web. This way the web is growing very fast, making search tasks more and more difficult with worse results. To solve the problem several initiatives were undertaken and a new area of research and development emerged: the one called Semantic Web.
When we refer to the semantic web we are thinking about a network of concepts. Each concept has a group of related resources and can be related to other concepts; we can then use this concept network to navigate among web resources or simply among information resources. From the undertaken initiatives one became an ISO standard: Topic Maps ISO 13250.
The aim of this paper is to introduce a *Topic Map (TM) Builder*, that is a processor that extracts topics and relations from instances of a family of XML documents.
A *TM-Builder* is strongly dependent on the resources structure. So, to extract a topic map for different collections of information resources (sets of documents with different structures) we have to implement several *TM-Builders*, one for each collection. This is not very easy! To overcome this inconvenient we have created an XML abstraction layer for *TM-Builders* that enables us to specify the topic map we want to build from a concrete family of resources, in order to generate automatically the intended extractor.
To describe that process, i.e. the extraction of knowledge from XML documents to produce a TM, we present a language to specify topic maps for a class of XML documents, that we call XSTM (XML Specification for Topic Maps). We also discuss a XSL processor that automatically generates the Extractor from its formal specification written in XSTM, the XSTM-P.

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

*Bolsista CNPq - Brasil

1 Introduction

This paper is concerned with knowledge extraction from documents marked up in XML. To go straight to our target topic, we clearly assume that the reader is familiar with XML and companion for document's structuring and processing. For details about these topics we suggest the reading of [13] [5] [14].
There are many tools for creation of Topic Maps (TM for short) like Mapalizer[1]. However, we do not know anyone that creates automatically the topic map from a XML specification of relevant items of the information using just XML tools. There is a chapter about *Automated/Automatic Topic Map Construction* in [1] but it does not explain how it is possible to implement this topic map's constructor.
Our aim is, precisely, the introduction of a XSL tool that reads a family of XML resources and builds a topic map – *TM-Builder*. Moreover, we intend to show that such a tool can be generated automatically instead of creating by hand a new one each time the documents type changes.
In this context, we understood that a Topic Maps specification language was necessary to enable the systematic derivation of a *TM-Builder*. XSTM (*XML Specification of Topic Maps*), the proposed language, is an XML language; so it becomes possible to create a *TM-Builder* that extracts Topic Maps from XML documents, using an XML dialect. That approach offers a complete XML framework to the user. The benefits of such an approach are obvious.
Section 2 presents the basic concepts in this paper. In section 3 there is an overview about Topic Maps [4]. We give a brief introduction to the subject and we show some of their characteristics. At the end of the section we present a real example (with fragments of the complete specification) that will help the reader to understand how a topic map is defined.
Section 4 describes the steps followed to develop the *TM-Builder*. The information extraction from XML resources depends on the schema of the resources. To overcome the problem of having to code a new Extractor every time we have a different class of XML resources, we have created an abstraction layer. This layer is composed by one specification in XSTM, a new XML language we have created for this purpose.
A formal specification of the proposed language, XSTM, is provided in section 4.1, showing a diagram that depicts the XML-Schema [10], and listing the respective DTD (Document Type Definition); in that section, we also detail the elements introduced in the DTD and illustrate their use through examples. For those more familiar with grammar based language definitions, we include a CFG (Context Free Grammar) for XSTM as well. The details concerning the implementation of the XSTM processor are introduced in section 4.2.
Section 5 presents a case study to illustrate all the concepts referred, and to show a real situation where we took profite of the proposed system (approach and tools), to create a semantic-driven website.
A synthesis of the paper and hints on future work are presented in the last part, section 6, together with some metrics about XSTM use.

2 Ontologies

An ontology is a logical theory to relate the intended meaning of a formal vocabulary, i.e., it is a binding with a particular conceptualization of a world. The ontology includes structures that allow manipulating terms in a more efficient way; it is useful to the human understanding and validation mechanisms operating on inter-agents communication. The importance of its use is the ability to represent hierarchies of object classes (taxonomies) and their relationships.
In the same area, different ontology definitions and classifications can be found . In Artificial Intelligence, Guarino defines ontology as a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world [12].
In the area of Information Systems, ontology is defined as a set of concepts and terms, that can be used to describe some area of knowledge, or construct a representation of the knowledge [20]. According to Chandrasekaran [7], ontologies are theories of content about the object types, object properties, and relationships between objects that are possible in a domain of specific knowledge.
The ontology's development will continue to provide the construction mechanism of the semantic part of Semantic Web. The model proposed by Berners-Lee[2] has been accepted mainly as representation to the architecture of the Semantic Web.
The development of these mechanisms depends on languages that express the information in a way that machines can understand. The challenge is to provide a language that will enable the manipulation, in an efficient way, of data and rules for queries about these data, and that will allow existing rules in any knowledge representation system to be exported to the Web.
The ontology's development will have to represent an important part of the effort in the development of any application in the future. That way, the development of environments to the construction and manipulation of ontologies is crucial and important. Such environment must be composed by an ontology deposit that can be manipulated by developers,

[1] http://www.topicmapping.com/mapalizer
[2] Available in http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

users, and application programs, allowing the navigation, search and terms reuse. When new terms are added to the ontology, the environment must verify the deposit consistency and act accordingly.
The idea of using XML to specify ontologies is not new and several markup languages have been developed. From the undertaken initiatives one became an ISO standard: Topic Map ISO 13250 [19]. However there are other languages that should be considered like: RDF [15], RDFS [6], DAML [8], OIL [9] or OWL [2]. The most used ones are Topic Maps (the XML version – XTM, is being widely used and became recently an appendix to the ISO standard) and RDF. People using them normally have the same goals in mind but they differ in their basic philosophy: XTM follows a top-down approach while RDF works in some sort of a bottom-up approach.
When one is developing a website thinks in a top-down perspective. So the XTM approach seems the best approach to specify an ontology from which the website will be generated.

3 Topic Maps

A TM is a formalism to represent knowledge about the structure of an information resource and to organize it in ”topics”. These topics have occurrences and associations that represent and define relationships between them. Information about the topics can be inferred by examining the associations and occurrences linked to the topic. A collection of these topics and associations is called a topic map.
Topic Maps can be seen as a description of what is about a certain domain, by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics [3].
A topic map expresses someone’s opinion about what the topics are, and which parts of the information set are relevant to which topics. Charles Goldfarb [11] usually compares Topic Maps to a GPS (Global Positioning System) applied to the information universe. Talking about Topic Maps is talking about knowledge structures. Topic Maps are the basis for knowledge representation and knowledge management.
Enabling to create a ”virtual map” of information, the information resources stay in its original form and so they are not changed. Then, the same information resource can be used in different ways, for different topic maps. As it is possible and easy to change the map itself, information reuse is achieved.
Topic Map architecture was also designed to allow merging between topic maps without requiring the merged topic maps to be copied or modified.

3.1 The characteristics of Topic Map model

A topic map is basically an XML document (or set of documents) in which different element types, derived from a basic set of architectural forms, are used to represent topics, occurrences of topics, and relationships (or associations) between topics [18]. The most known XML version – XTM [19] – is being widely used and became recently an appendix to the ISO standard.
Topics are the main building blocks of topic maps[17]. In its most generic sense, can be anything. A person, an entity, a concept, really anything regardless of whether it exists or has any other specific characteristic. It constitutes the basis for the topic maps creation. It can be seen as a ”multi-headed link, that points to all its occurrences” [3]. This ”link” aggregates information about a given subject (the thing that the topic is about).
Each topic has a topic type or perhaps multiple topic types. *Topic Type* could be seen like a typical class-instance relationship. Types represent the classes in which topics are grouped in, i.e., the category of one topic instance. Topic types are also topics (by standard definition).
A topic can have a name or more than one. However, topics do not have always names: a cross reference (e.g. - page 105), is considered to be a link to a topic that has no (explicit) name. The ability to specify more than one topic name can be used to name topics within different scopes, such as language, style, domain, geographical area, historical period, etc.
A topic can have one or more occurrences. One or more addressable information resources of a given topic, constitutes the set of *Topic Occurrences*. It might be a monograph devoted to a particular topic, for example, or an article about the topic in an encyclopedia; it could be a picture or video describing the topic, a simple mention of the topic in the context of something else, a commentary on the topic (if the topic were a law, say), or any of a lot of other forms in which an information resource might have some relevance to a topic [18]. A topic occurrence represents the information that is specified as relevant to a given subject.
Occurrences and topics exist on two different layers (domains), but they are ”connected”. Occurrences establish the routes from the topics to the information resources, enabling also to provide the reason why that route exist.
At this point it is very clear the separation in two layers of the topics and their occurrences, one of the great features of Topic Maps.
Among all occurrences of a given topic, a distinction can be made among subgroups. Each subgroup is defined by a common role. *Occurrence role* can be used to distinguish graphic from text, main occurrences from ordinary

occurrences, mentions from definitions, etc. ”The occurrence roles are user-definable and therefore can vary for each topic map” [3].
The standard also defines occurrence roles as topics. If an occurrence role is defined as a topic explicitly, topic map facilities can be used to say useful things about them (such as their names, and the relationships they participate in).
But to make the real distinction between different types of occurrences, Topic Maps uses also the concept of *occurrence role type*. This is different of the occurrence role in the sense that the last one is simply a mnemonic and the first one is a ”reference to a topic in the map, which further characterizes the relevance of the role” [18].
The order used to specify topics and their associations is irrelevant; however, if necessary, a certain semantic order can be imposed.
Topic associations are almost ordinary links, except that they are constrained to only relate topics together. Because they are independent of the source documents in which topic occurrences are to be found, they represent a knowledge base, which contains the essence of the information that a someone is creating, and actually represents its essential value. An unlimited number of topics can be associated within ”topic associations”.
The power of topics maps increases with the creation of topic associations because that way, it is possible to group together a set of topics that are somehow related. This is of great importance in providing intuitive and user-friendly interfaces for navigating large pools of information.
As topic types group different kinds of topics and occurrences roles supports occurrences of different types, associations between topics can also be grouped according to their type (*Association Type*).
It is important to refer that each topic that participates in an association has a corresponding *association role* which states the role played by the topic in the association. Association roles are also regarded as topics in the topic map standard.

3.2 How to define a Topic Map

Before we start, we need to know exactly what we want to represent in the topic map. There are two phases to this: delimiting the scope of the TM (that is, deciding the extent of the domain it should cover); and designing the basic ontology. In TM terminology, an ontology is a precise description of the kinds of things that are found in the domain covered: in other words, the set of topics that are used to define classes of topics, associations, association types, association roles, and occurrences.
To illustrate all the ideas so far introduced and describe the TM building process, we will present an example whose subject is an university and its professors, research groups, departments, and courses. The scope can easily be extended to cover the students, the employees, the projects, etc.
In the examples fragments that follow, we will assume that a persont with name *Pedro Rangel Henriques* is *doctor* and a *professor* at *Department of Informatics*, where he teaches courses in *LMCC* and *LESI* degrees. Also he is a member of *gEPL* research group.
The basic ontology therefore consists on the *topic types Professor*, *Department*, *Course*, *Degree*, and *Group*, the *association roles works-at/employs*, *teaches/is-taught-by*, *has-title/is-title-of*, and *is-member-of/has-member*, and the *association types work*, *education*, *academic-title*, and *research*.
In the first step, we define the topics (topic types and their instances), specifying their identifiers and base names. Below is an incomplete example:

```
<?xml version="1.0" encoding="UTF-8"?>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0"
       xmlns:xlink="http://www.w3.org/1999/xlink">
  <topic id="Professors">
    <baseName>
      <baseNameString>Professors</baseNameString>
    </baseName>
  </topic>
  <topic id="Pedro-Rangel-Henriques">
    <instanceOf>
      <topicRef xlink:href="#Professors"/>
    </instanceOf>
    <baseName>
      <baseNameString>Pedro Rangel Henriques</baseNameString>
    </baseName>
  </topic>
</topicMap>
```

We only show the definition of the topic *Pedro-Rangel-Henriques* and its type *Professors*. The other topics and their types are created in a similar way.

In the next step, we add the occurrence definitions, using the element (*resourceRef*) that contains the URL (resource address) as the value for the attribute *xlink:href*. For instance, one occurrence for the topic *Pedro Rangel Henriques* is specified in XPath writing the path to the XML file used as the resource, as illustrated below.

```
<topic id="Pedro-Rangel-Henriques">
  ...
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#email"/>
    </instanceOf>
    <resourceRef xlink:href="prh@di.uminho.pt"/>
  </occurrence>
</topic>
```

Notice the use of *#xpath* as a *xlink:href*; it is possible because *xpath* is also a topic, more precisely it is a occurrence type.
The "..." in the code above stands for the definition of the topic *Pedro-Rangel-Henriques* as appeared in the previous specification fragment; we do not repeat it to keep the example as light as possible.
In the third step, we define the associations among topics, stating their type and their members (a topic with an explicit role). In the example below, we show the *work* association between *Department of Informatics* and *Pedro Rangel Henriques*. The first one *employs* the second that *works-at*.

```
<association>
  <instanceOf>
    <topicRef xlink:href="#work"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#employs"/>
    </roleSpec>
    <topicRef
      xlink:href="#Department-of-Informatics"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#works-at"/>
    </roleSpec>
    <topicRef
      xlink:href="#Pedro-Rangel-Henriques"/>
  </member>
</association>
```

The references *employs* and *works-at* are association role types, and they are declared like a topic type, i.e., with a identifier and a base-name only. The reference *work* is an association type, that defines the type of this association. The declaration of this topic is shown below:

```
<topic id="work">
  <baseName>
    <baseNameString>Work</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#employs"/>
    </scope>
    <baseNameString>employs</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#works-at"/>
    </scope>
    <baseNameString>works at</baseNameString>
  </baseName>
</topic>
```

The TM above explains that the Pedro Rangel Henriques works at Department of Informatics and, at the same time, indicates that the Department of Informatics employs Pedro Rangel Henriques.
At this point we can say that ontologies, specified with XTM [19], are a set of records, each record represents a concept, it points to some resources (physical information records) and participates in several relations (associations).

4 The TM-Builder – The Topic Map Extractor

Looking at a TM we can think of it as having two distinct parts: an ontology and an object catalog. The ontology is defined by what we have been designating as topic type, association type, and occurrence role type. The catalog is composed by a set of information objects that are present in information resources (one object can have multiples occurrences in the information resource) and that are linked to the ontology. Figure 1 gives a schematic representation of this vision.

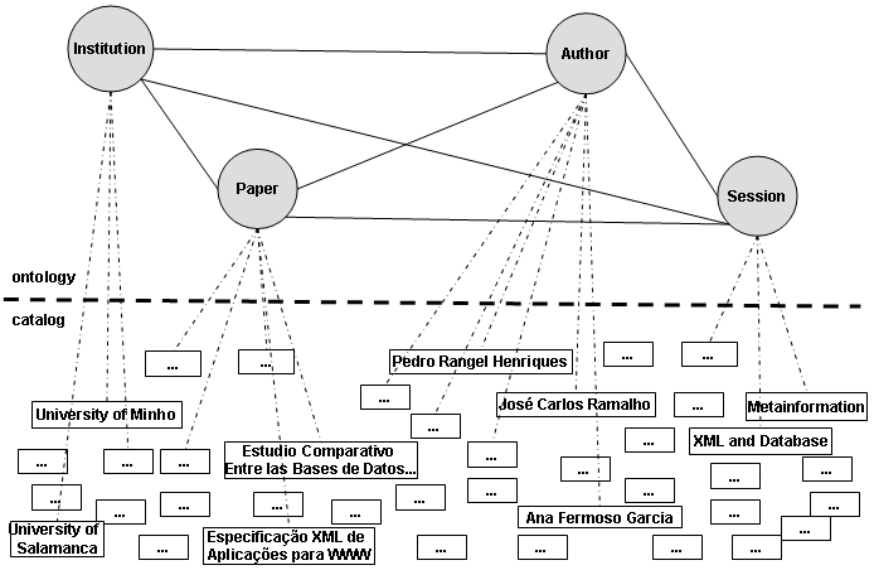

Figure 1: The Ontology and Catalog for our case-study

After creating some Topic Maps by hand, it is easy to conclude that such task is time consuming and very repetitive. Thus gave us the idea to develop an Extractor of topics and relations from a set of XML resources, or by other words, a *TM-Builder*.
The reasons of the use of topic maps are that, first of all, there are too few topic maps-based tools and programs out there to ensure widespread use, and this was first of all our contribution to the distribution process. Second, topic maps is an outstanding technology and standard that bridges several levels of professionals, from teaches to marketing to information architects.
In our context, a *TM-Builder* is a converter from one XML language to another XML language. The *TM-Builder* is an XSL stylesheet that receives an XML document as input and generates another XML file that contains a Topic Map. The reasons why the proposed *TM-Builder* accepts XML documents as input are:

- XML is the current language to markup documents;
- XML is becoming the platform for information interchange;
- New data sources (non-XML) can be easily added to our extracting system just by using a translator to XML. Most of the actual information systems, like Database Management Systems, have facilities to dump their information in XML; so, for these cases the front-end is already there.

The main algorithm of the *TM-Builder*[3] to extract a Topic Map from an XML document is shown below:

```
> Initially, for the given ontology creates all the:
  * topics types;
  * occurrences roles;
  * occurrences types;
  * associations types;
```

[3]We assume that it contains the definition of the working ontology.

```
> During a document tree traversal, for each association, define the:
  * association type;
  * association members;
  > for each element in the source that is seen as a topic, create the:
    * topic ID;
    * topic type;
    * topic names;
    * topic occurrences;
```

In the next section, we will introduce a language that can be used to specify the extraction process.
In our system, that algorithm was coded in a XSL stylesheet. In practice we found that after the XSL processing, an XML Topic Map file will be generated. This Topic Map can have a problem: a set of topics with the same identifier. This problem occurs when an element or attribute (that was defined as a topic) is found more than once in an input XML file. Each time that this element/attribute is found, a new topic is created. So, many topics will be created with the same identifier. We must substitute all these topic definitions by just one definition: the identifier, topic type, and base name, shall appear once; different occurrence definitions must be created for each topic found.
To solve the problem, another XSL stylesheet was develop. This stylesheet is called when the first finishes the tree traversal; it will look for these problematic patterns in the generated XTM producing the final XTM file.
In the new stylesheet, for each set of topics with the same identifier, an unique topic will be created with the same topic type common to all, with the union of all individual occurrences as the occurrence set. All others topics are deleted.
In the next section, we will introduce a language that can be used to specify the extraction process.

4.1 XSTM: an XML Language to specify Topic Map extractors

The TM extractor discussed in the last section is tied to the structure of a specific XML source. The creation of a TM for an XML source with a different structure would imply the development of a new extractor. To solve this problem we have created an abstraction layer based on a new XML language: XSTM (XML Specification of Topic Maps).
The XSTM language supplies all the constructors that are needed to specify the extraction task, the Topic Map Builder process; it allows the definition of topics and their types and occurrences, as well as associations and their types and occurrence roles.
In a more formal way, we show below the CFG (Context Free Grammar) for that language:

XSTM's Context Free Grammar

```
xstm        ::= topicType+ topic+ assoc* assocType*
topicType   ::= TTypeID InstanceOf TTypeName
topic       ::= xpath TTypeID InstanceOf Resource*
Resource    ::= resourceData | resourceRef
assocType   ::= ATypeID ATypeName MemberAssoc*
MemberAssoc ::= Scope Description
assoc       ::= assocClass ATypeID Members*
assocClass  ::= "N2N" split=("true"|"false") |
                "one2one" type=("attribute"|"subelement") |
                "one2N" split=("true"|"false") |
                "all2all"
Members     ::= ElemIndex Element*
Element     ::= TTopicAssoc RoleID
```

Each XSTM specification is defined as an XML instance and the XSTM language is defined by a DTD and/or an XML-Schema.
Nowadays, XML-Schema has overcome the DTD approach for the definition of classes of the markup documents. We also made that upgrade; however, as XML-Schema is much more verbose than the correspondent DTD, we decided to include here the XSTM DTD.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT xstm (topicTypes, topics,
         occurrenceRoles?, assocTypes?, assocs?)>
<!ELEMENT topicTypes (topicType+)>
<!ELEMENT topicType (id, name)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT topics (topic+)>
<!ELEMENT topic (xpath, type)>
```

```
<!ELEMENT type (#PCDATA)>
<!ELEMENT xpath (#PCDATA)>
<!ELEMENT occurrenceRoles (occurrenceRole+)>
<!ELEMENT occurrenceRole (id, name)>
<!ELEMENT assocTypes (assocType+)>
<!ELEMENT assocType (id, name, memberAssoc+)>
<!ELEMENT memberAssoc (scope, description)>
<!ELEMENT scope (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT assocs (one2one | one2N | N2N)+>
<!ELEMENT one2one (type, members11)>
<!ATTLIST one2one
type (attribute | subelement) #IMPLIED
>
<!ELEMENT one2N (type, members1N)>
<!ATTLIST one2N
split CDATA #IMPLIED
>
<!ELEMENT N2N (type, membersNN)>
<!ATTLIST N2N
split CDATA #IMPLIED
>

<!ELEMENT members11 (element, elementRef)>
<!ELEMENT members1N (one, N)>
<!ELEMENT membersNN (N)+>

<!ELEMENT element (topicAssoc, role)>
<!ELEMENT elementRef (topicAssoc, role)>
<!ATTLIST elementRef
target CDATA #IMPLIED
>
<!ELEMENT role (#PCDATA)>
<!ELEMENT one (topicAssoc, role)>
<!ELEMENT N (topicAssoc, role)>
<!ELEMENT topicAssoc (#PCDATA)>
<!ATTLIST topicAssoc
name CDATA #IMPLIED
id CDATA #IMPLIED
>
```

Notice that the XSTM DTD listed above is obtained direct and systematically from the CFG shown.

4.2 XSTM-p: an XSTM Processor

The XSTM language, defined in the previous section, specifies the *TM-Builder* process, enabling the systematic codification (in XSLT) of the extraction task.
In that circumstances we understood that it was possible to generate automatically the Extractor developing another XSL processor to translate an XSTM specification into the *TM-Builder* code.
The XSTM processor (XSTM-P for short) is the *TM-Builder* generator; it is one of the main pieces in our architecture, as can be seen in Figure 2. It takes a TM specification (an XML instance, written according to the XSTM language), and generates an XSL stylesheet that will process an input XML document to extract the topic map.
Both XSL stylesheets (the generator and the extractor) are processed by a standard XSL processor like Saxon[4] or Xalan[5], what in our opinion is one of the benefits of the proposal.
The main algorithm of the XSTM-P is now:

```
> Define the KEY tables, to create associations from cross references.
> During the tree traversal:
  * for each topic type: create xstm:topic;
  * for each occurrence role: create xstm:topic;
  * create the occurrence types;
  * for each association type: create xstm:topic, which includes the members
```

[4] http://saxon.sourceforge.net/
[5] http://xml.apache.org/xalan-j/

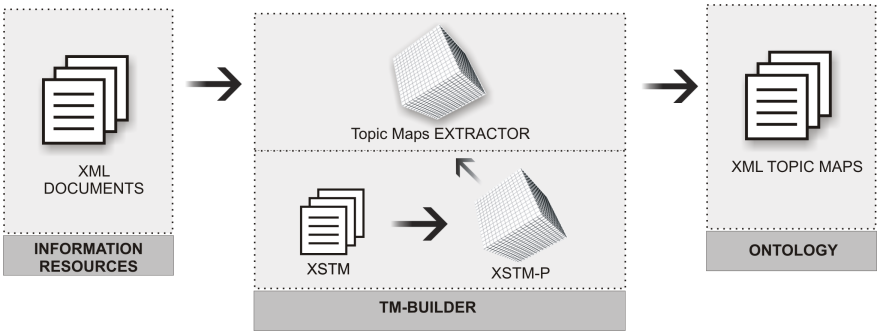

Figure 2: The TM-Builder architecture.

```
    of this association type;
> For each topic defined in XSTM file:
  * create a xstm:for-each to each element with the path to each one;
> For each association defined in XSTM file:
  * create the association among all members:
    - defined in one2one;
    - defined in one2N;
    - defined in M2N;
    - defined in all2all.
```

5 Case-Study – Conference website specification and generation

This section presents a case concerned with the organization of the workshop "*XML, Aplicações e Tecnologias Associadas*" (*XATA*); the case-study demonstrates the use of *TM-Builder* to build automatically the conference website from a set of XML documents that describe the event. This workshop[6] was held at University of Minho, Braga, in the beginning of 2003, joining together the XML portuguese community (Researchers and Users of XML from universities or companies) to share information about XML research and development between academic and professional worlds. Many papers were submitted to this workshop; the accepted ones were presented in the conference. The paper presentations were grouped by sessions, each one associated with a specific theme, like *Technology and Web Services*, *XML and Database*, among others.

The Information System that has supported the event (from the first announcement to the proceedings edition and last hour diffusion material) was completely XML-based. Therefore, all the XATA related information is stored in XML documents. The XML-Schema of the workshop documentation is presented in the figure 3. Obviously this XML-Schema is incomplete, but the relevant elements for this study are included and shown.

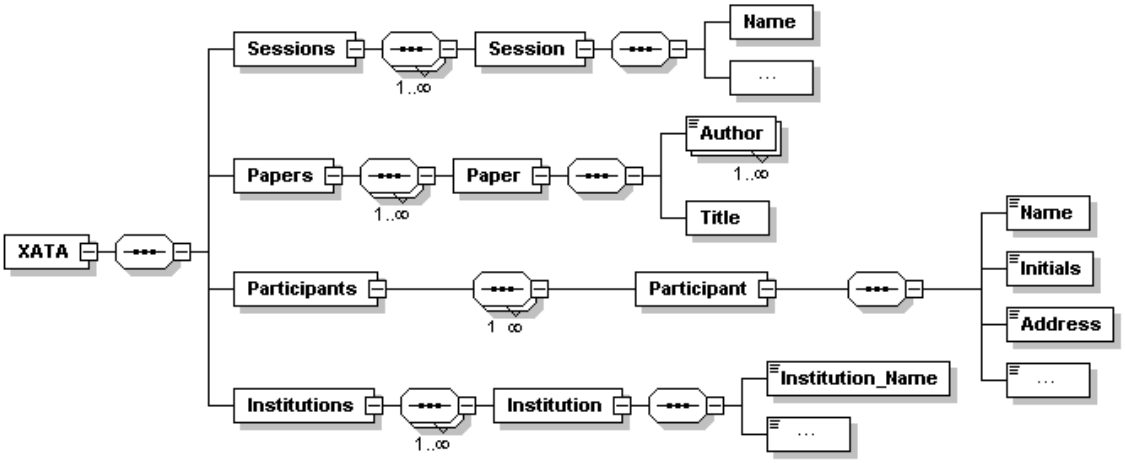

Figure 3: XATA's XML-Schema.

As the XSTM language only depends on the documents' structure, and not on the XML instance, the ontology specification can be defined over the XML-Schema shown; this task will be accomplished in five steps: definition of the

[6] http://www.di.uminho.pt/~jcr/XML/conferencias/xata2003/

types of topics, of the topics itself, of the occurrence roles, of the association types, and finally, of the associations itself.

5.1 The XSTM specification for the XATA

The XSTM root element is *xstm*, that has four sub-elements. Each sub-element references a piece of the ontology expressed by XTM. Its sub-elements are: *topicType*, *topic*, *assocType*, and *assoc*.
First of all, it is defined the topic type. In this ontology, the topics are grouped in *Institution*, *Author*, *Participant*, *Session*, and *Paper*.
In XSTM, the topic types are declared by *topicType* element, that contains a *id* identifier – to be referenced at other moments in the specification – and a *name* name – to visualization in a XTM browser. For instance, the *Paper* topic type declaration is shown below:

```
<xstm>
  <topicTypes>
    <topicType>
      <id>ID-Paper</id>
      <instanceOf>XATA</instanceOf>
      <name>Paper</name>
    </topicType>
    <topicType>...</topicType>
  </topicTypes>
  ...
</xstm>
```

In XSTM, the topic type are abstract concepts defined by the ontology. The topics are real elements in the input XML documents. The topic element is used to its definition. This element has two sub-elements: the XPath path to the element itself (*xpath*) and its type (*type*). The attributes *id* and *name* mean the XPath path to the topic identifier and to the topic name. The XSTM specification to the topic definition referent to the Paper topic type is presented below:

```
<topics>
  <topic>
    <xpath id="@number" name="Title">//Paper</xpath>
    <type>ID-Paper</type>
  </topic>
  <topic>...</topic>
  ...
<topics>
```

Until here, all the topics and its types are declared. But in XTM, topic without any association has little functionality. The knowledge web is obtained from the associations among topics. Many associations can be inferred from the *XATA*; therefore, the example is the association among *Paper* and *Author* type topics.
Once defined the topics and its types, the next step is the definition of association types. They define the occurrence role for the association members. It is declared with the *assocType* element that has a *id* identifier, a *name* name, and its *membersAssoc* association type.
The *member* element means the member of the association and it has two child. The *scope* element specifies the reference to the topic that is the scope of this occurrence role. The other one, named *description*, is a name that will be displayed in a Topic Map's application. Each association role will be a new topic, in the output XTM. An *association types* specification in XSTM, between *Author* and *Paper*, can be seen below:

```
<assocTypes>
  <assocType>
    <id>Author-and-Paper</id>
    <name>Author and Paper</name>
    <memberAssoc>
      <scope>is_wrote_by</scope>
      <description>is wrote by</description>
    </memberAssoc>
    <memberAssoc>
      <scope>is_author_of</scope>
      <description>is author of</description>
```

```
    </memberAssoc>
  </assocType>
  <assocType>...</assocType>
</assocTypes>
```

To finish the XSTM specification, the *assoc* element allows the specification of all the associations. This associations can involve two or more topics. They are found and extracted from the input XML document.
In the following, when we refer to relationships between tree nodes (XML elements and attributes) we are not talking about relations in the sense of the well-known entity-relationship model. So the usual names 1-to-1, N-to-1, M-to-N, and all-to-all, do not have exactly the same meaning used in that traditional perspective.
In our context, there are four kinds of relationships between elements, that are described by the three alternative children of the *assocs* element:

- associations between an element and its attribute. It is defined by the *one2one* element whose *type* attribute has the value *attribute*;
- associations between an element and a subelement referenced in the element's context. It is defined by the *one2one* element with the *subelement* value in the *type* attribute;
- associations one to N, defined by the *one2N* element;
- associations M to N, defined by the *M2N* element;
- associations between topics that are connected through another table, defined by the *all2all* element.

The *assocs* element contains specification of its type and all its members. The *type* always means the association type (see the previous subsection), i.e., a reference to the identifier of the topic that represents its association type.
The members are a choice of *one2one*, *one2N*, *one2N*, or *all2all*; all of these elements have a similar structure: a sequence of a *type* and *members* elements.
The *members* element is composed by one or more of *member* that defines the members of this association, and it is composed by three elements: the *xpath* means the XPath path to the element (or attribute) that is a member in this association; and the *role* element, that is a occurrence role of this member.
The one2one element expresses relationships that can be obtain from some connection among the topics found in XML document. For instance, in the specific association between *Author* and *Paper*, the authors of each paper can be identified by the content of XPath path *//Paper/Author*, which is a reference to the inicial letters of authors name found in *//Participant/Initial*. Thus, the association among the *Author* and *Paper* topic types, referent to the XATA, was specified in the way shown below:

```
<assocs>
  <one2one type="subelement">
    <type>Author-and-Paper</type>
    <members11>
      <element>
        <topicAssoc ref="Author">Paper</topicAssoc>
        <role>is_wrote_by</role>
      </element>
      <elementRef>
        <topicAssoc ref="Initial">Participant</topicAssoc>
        <role>is_author_of</role>
      </elementRef>
    </members11>
  </one2one>
</assocs>
```

After processing the complete specification for the case-study under work, a XSTM description with 194 lines, we produced a *TM-Builder* that is a XSLT file with 413 lines. The Topic Map extracted from a source file with 10132 lines is a XTM file with 44018 lines with 1465 topics and 1216 associations.
Although short[7], we are convince that the example is expressive enough.
Figure 4, shows the topic map visualization obtained with *Ulisses* [16]. The input topic map was created by *TM-Builder*. This navigator gives access to the information contained in the source documents filtered by the Topic Map specification, allowing the navigation through the topic instances driven by the associations, defined in the ontology specified in XSTM.
Figure 5 shows the ontology described in XSTM:

[7] We have simplified the ontology to deal only with a small number of topics and associations.

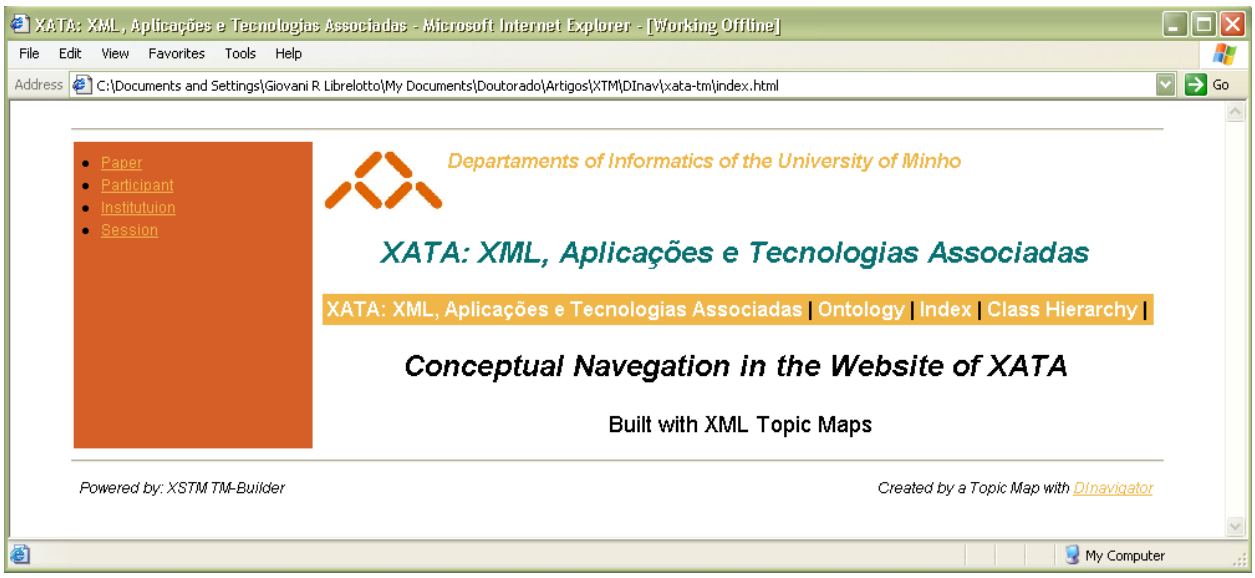

Figure 4: XATA's topic map visualization in the Ulisses.

- the topic types in the item *Subject Indexes*;
- the associations in the item *Relationship Indexes*;
- the role associations in the item *Role Indexes*;
- the resources types in the item *Resource Indexes*.

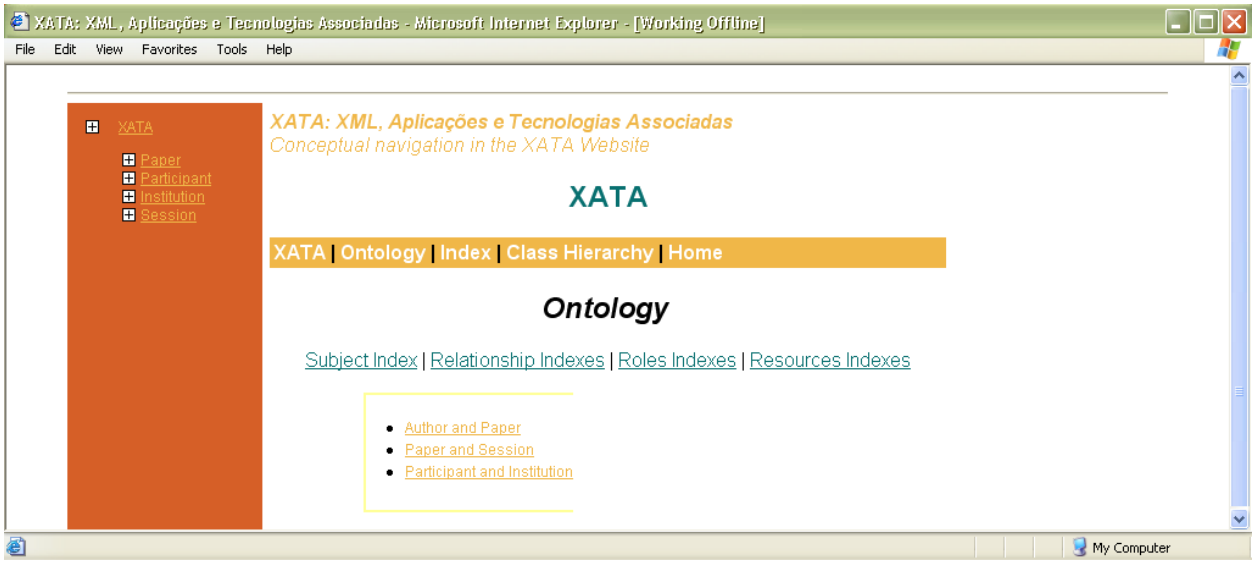

Figure 5: XATA's topic map visualization – Ontology section.

6 Conclusion

In this paper we introduced an architecture for the automatic construction of topic maps using XSL stylesheets to process a family of XML documents – the information resources. This system, entirely based on XML tecnologies is called *TM-Builder*. This Topic Map Builder is supported by an specification language, called XSTM (XML Specification of Topic Maps), that describes the desired topic map relating the underlying ontology with the instance data items to catalog according to the elements in DTD of the resources under consideration.
The effort to describe the ontology in XSTM is similar to that required in XTM: we have to specify every single topic type, association type, and occurrence role type. However, in XTM everything is a topic. XSTM further classifies

those topics, giving them a more concrete semantics, naming them *topic type*, *association type*, or *occurrence role type*. So, for the ontology part, the gain is achieved through a more precise semantics.
For the catalog, the situation is completely different. In our *topic and association specifications* we use *XPath expressions* that act like queries. This way the gain we obtain is equal to the number of occurrences retrieved by the query expression. In the case of the associations the gain is even higher: N for the 1:N relations and MxN for the M:N relations.
A case study was included in order to illustrate all the concepts involved in this paper. This real case study deals with the construction of *semantic website* for a workshop (XATA 2003): from an XML document describing the event (sessions, papers, participants, etc) we automatically extracted with a TM-Builder a topic map that was used to allow the browsing of the XATA information driven by an ontology defined for that specific workshop. That semantic driven XATA website was obtained using *Ulisses*, a conceptual navigator for XML Topic Maps, developed by our team at that time. The results of this case show the real gain obtained with XSTM specification; the *TM-Builder*, generated to extract topic maps from XATA XML documents, is able to process every XML document – independently of its size – according schema previously defined.
The most interesting achievement of our proposal is that the size of the XML resource does not influencies the size of the XSTM specification, because what accounts is just its structure, i.e., the size of its DTD. So, if this XML document grows up, the same *TM-Builder* is able to process it.
Concerning future work, we can announce the evolution of *TM-Builder* to support also databases as information resources. This second generation of *TM-Builder* will use a completely different approach for the internal representation and data-handling in order to extract topics from XML documents or databases saving the topic map or in an XTM file, or in a new database structured according to XTM format. However, it will the same XSTM language for the specification of the intended topic map.

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] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Web Ontology Language (OWL) Reference Version 1.0. World Wide Web Consortium, November, 2002. http://www.w3.org/TR/owl-ref/.

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

[4] 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.

[5] Neil Bradley. *The XML Companion*. Addison-Wesley, 3rd edition, 2002.

[6] Dan Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. World Wide Web Consortium, March, 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[7] B. Chandrasekaran. What Are Ontologies, and Why do We Need Them? In *IEEE Inteligent Systems and their applications*, volume vl 9, n 1. IEEE, Jan/Fev 1999.

[8] DARPA. DAML. Darpa Agent Markup Language Program., 2001. http://www.daml.org/.

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

[10] Jon Duckett, Oliver Griffin, Stephen Mohr, Francis Norton, Nikola Ozu, Ian Stokes-Rees, Jeni Tennison, Kevin Williams, and Kurt Cagle. *Professional XML Schemas*. Wrox Press, 2001.

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

[12] Nicola Guarino. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, pages 139–170. Springer Verlag, http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/SCIE97.pdf, 1997.

[13] Elliote Rusty Harold and W. Scott Means. *XML in a Nutshell*. O'Reilly & Associates, 2001.

[14] Steven Holzner. *Inside XML*. New Riders Publishing, 1st edition, 2000.

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

[16] 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 – CLEI, La Paz, Bolívia*, 2003.

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

[18] 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.

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

[20] W. Swatout and A. Tate. Ontologies. In *IEEE Inteligent Systems and their applications*, volume vl 14, n 1. IEEE, Jan/Fev 1999.