XML Topic Map Builder: Specification and Generation

Giovani Librelotto
University of Minho
Department of Informatics
Braga, Portugal, 4710-057
grl@di.uminho.pt

José C. Ramalho
University of Minho
Department of Informatics
Braga, Portugal, 4710-057
jcr@di.uminho.pt

Pedro R. Henriques
University of Minho
Department of Informatics
Braga, Portugal, 4710-057
prh@di.uminho.pt

ABSTRACT

Everyday thousands of new information resources are linked to the web. This way the web is growing very fast what makes search tasks more difficult. 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 resources, thought we can use this concept network to navigate among web resources or simply among information resources. From the undertaken initiatives one became an ISO standard: Topic Map 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 Topic Map Builder is strongly dependent on the resources structure. So if we have an heterogeneous set of information resources we have to implement several Topic Map Builders. This is not very useful, and to overcome this problem we have created an XML abstraction layer for Topic Map Builders.

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, the XSTM-P.

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 documents structuring and processing. For details about these topics we suggest the reading of [6] [3] [7].

There are many tools for creation of Topic Maps (TM for short) like Mapalizer[1]. However, we do not know anyone that from a XML specification of relevant items of the information creates automatically the Topic Map 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.

In this context, we understood the need for a Topic Map specification language 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 Topic Map Builder that generates 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.

In section 2 there is an overview about Topic Maps. We give a brief introduction to the subject and we show some of its characteristics. At the end of the section we present a case study that will help the reader to understand how a Topic Map is created.

Section 3 describes the steps followed to develop the Topic Map Builder. The information extraction from XML resources depends of the DTD/Schema of the resources. To overcome the problem of having to code a new Extractor everytime we have a different class of XML resource 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, showing a diagram that depicts the XML-Schema [4], and listing the respective DTD; 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 for XSTM as well.

The details concerning the implementation of the XSTM processor are introduced in section 5.

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. XML 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 [2].

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 [5] 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,

[1] http://www.topicmapping.com/mapalizer

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.

2.1 The characteristics of Topic Maps 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 [9].

Topics are the main building blocks of topic maps[8]. 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" [2]. 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 [9]. 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" [2].

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" [9].

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.

2.2 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, roles, and occurrences.

To illustrate all the ideas so far introduced and describe the TM building process, we will present a case-study when the subject is a 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 that follow, we will assume that a professor's name is *Pedro Rangel Henriques*, and that he 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 of 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 themselves (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"
                xlink:type="simple"/>
    </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="#xpath"
                xlink:type="simple"/>
    </instanceOf>
    <resourceRef
xlink:href="/university[1]/professors[1]/prof[1]"/>
  </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.

3. THE TM BUILDER

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.

In our context, a TM Builder is a converter from one XML language to another XML language. The TM Builder is a 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 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[2] 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;
> 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 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/attrubute 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

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

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.

Finally, we conclude that the Topic Map built by hand and partially listed in the subsection 2.2, could be automatically extracted by the two XSL stylesheets describe above (that constitute our TM Builder) from the XML source file below:

```
<?xml version="1.0" encoding="UTF-8"?>
<university>
  <dept>
    <name>Department of Informatics</name>
    <local>Campus de Gualtar</local>
  </dept>
  <professors>
    <prof title="Dr">
      <name>Pedro Rangel Henriques</name>
      <phone>4469</phone>
      <groupID>1</groupID>
    </prof>
    <prof title="Dr">
      <name>Jose Carlos Ramalho</name>
      <phone>4479</phone>
      <groupID>1</groupID>
    </prof>
    ...
  </professors>
  <courses>
    <course>
      <name>LESI</name>
      <class>MP1</class>
      <class>MP3</class>
    </course>
    <course>
      <name>LMCC</name>
      <class>PED1</class>
      <class>PL1</class>
    </course>
    ...
  </courses>
  <groups>
    <group>
      <id>1</id>
      <name>gEPL</name>
    </group>
    <group>
      <id>2</id>
      <name>gLMF</name>
    </group>
    ...
  </groups>
</university>
```

In the next section, we will introduce a language that can be used to specify the extraction process.

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

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 DTD and a respective XML-Schema. These diagrams (figure 1 to 8) were obtained with the XML Spy 5.0[3], from Altova.

```
<?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)>
```

[3]http://www.xmlspy.com/

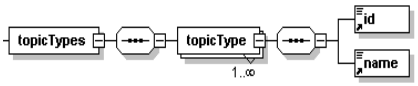

Figure 1: The topicTypes element.

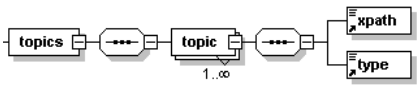

Figure 2: The topics element.

```
<!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.

4.1 The *topicTypes* element

The *topicTypes* element is a sequence of *topicType* elements, where each one is a new topic type in the Topic Map. The topic type has an identifier and a name. The *id* element is the global identifier of this topic type in the Topic Map. This identifier will be referenced by others topics. The *name* element is the topic type name, that will be shown in a visualization of the Topic Map. Figure 1 shows the respective XML-Schema.

For instance, in our case-study the XSTM description of topic type is:

```
<topicTypes>
  <topicType>
    <id>Department</id>
    <name>Department</name>
  </topicType>
  <topicType>
    <id>Professor</id>
    <name>Professor</name>
  </topicType>
   <topicType>
    <id>Degree</id>
    <name>Degree</name>
  </topicType>
  ...
</topicTypes>
```

As was observed in the section 2, a topic type also is a topic. So, each topic type defined in XSTM language is transformed in a new topic if the new TM, which will be referenced by other topics (*instanceOf*).

4.2 The *topics* element

While topic types are abstract concepts defined by the ontology, topics are actual elements in the XML documents taken as input. To define them, we added the *topics* element, that is a sequence of *topic* element. This last one is a sequence of two required elements, as can be seen in the figure 2. The *xpath* element means the XPath expression that describes the path to the element that will be the topic, in the input XML document. The *type* element has a text content that specifies the identifier of its topic type.

Below we find the XSTM description of the topics for the case-study under consideration.

```
<topics>
  <topic>
    <xpath>//dept/name</xpath>
    <type>Department</type>
  </topic>
  <topic>
    <xpath>//prof/name</xpath>
    <type>Professor</type>
  </topic>
  <topic>
    <xpath>//prof/@title</xpath>
    <type>Degree</type>
  </topic>
  ...
</topics>
```

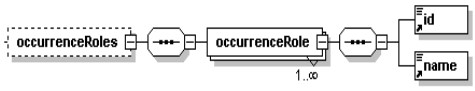

Figure 3: The occurrencesRoles element.

In this case we are selecting several XML elements at same time, through the use of an XPath expression. In the last section, we will discuss the gain we obtain from this kind of statement.

4.3 The *occurrenceRoles* element

The *occurrenceRoles* element defines the possible role types (once again an abstract concept) for the occurrences of each topic in an association. It has a child called *occurrenceRole* that is a sequence of an identifier and a name, as can be observed in the figure 3. It has the same structure of *topicType* element. The identifier (*id* element) can be referenced in an association and in a association type. The *name* element contains the name that will be displayed in the visualization of this Topic Map in a TM's application. Each occurrence role type will be a topic.

Below, there are the XSTM specification (we just show a fragment) of the *occurrence role types* in the ontology of our case-study.

```
<occurrenceRoles>
  <occurrenceRole>
    <id>works-at</id>
    <name>works at</name>
  </occurrenceRole>
  <occurrenceRole>
    <id>employs</id>
    <name>employs</name>
  </occurrenceRole>
  ...
</occurrenceRoles>
```

4.4 The *assocTypes* element

The *assocTypes* element allows to define another abstract concept in the ontology, the association types. It is composed by one or more of *assocType* that is a sequence of three elements (figure 4): an identifier (*id*) that will be referenced by an association; a name (*name*) that will be shown in a visualization of the Topic Map in a application like Omnigator[4] or TM-Design[5] and; and a member.

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.

[4]http://www.ontopia.net

[5]http://www.topicmap-design.com/en/topicmap-designer.htm

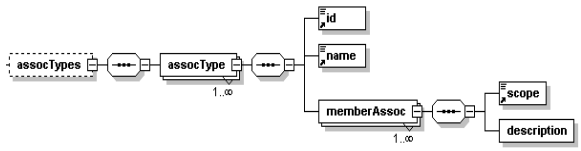

Figure 4: The assocTypes element.

An XSTM specification for *association types* can be seen below:

```
<assocTypes>
  <assocType>
    <id>work</id>
    <name>Work</name>
    <memberAssoc>
      <scope>employs</scope>
      <description>employs</description>
    </memberAssoc>
    <memberAssoc>
      <scope>works-at</scope>
      <description>works at</description>
    </memberAssoc>
  </assocType>
  ...
</assocTypes>
```

4.5 Relationships Types

The *assocs* element allows the specification of all actual associations that involve two or more topics and that can be found and extracted from the XML documents taken as input.

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 and M-to-N, 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;

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*, or *M2N*, as shown in Figure 5; all of these elements have a similar structure: a sequence of a *type* and *membersXX* elements. The *XX* means the kind of relationship (1 to 1, 1 to N, M to N).

4.5.1 Relationships 1 to 1

This relationship represents two kinds of associations: between element and attributes, and between element and its subelement referenced.

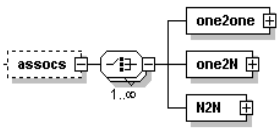

Figure 5: The assocs element.

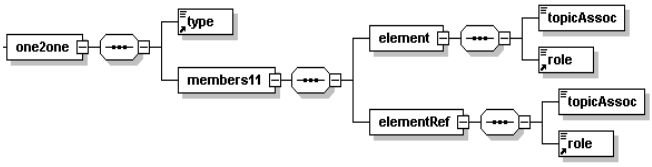

Figure 6: The one2one element.

The structure of the *one2one* element is a sequence of a association type and the members of this association, as can be seen in the figure 6. This element has an attribute named *type* that defines the kind of this association. The value of this attribute must be *attribute* or *subelement*.

The *members11* element (11 means 'one to one') is a sequence of *element* and *elementRef* elements. The first one specifies the main element in this relationship. The second one is an attribute or an subelement referenced by the first one.

The relationship between the topic types *Professor* and *Degree* illustrates a one2one relationship because actually it relates one element (*prof*) with one of its attributes (*title*). Below we present the XSTM specification for that association *academic-title*.

```
<one2one type="attribute">
  <type>academic-title</type>
  <members11>
    <element>
      <topicAssoc>Professor</topicAssoc>
      <role>has-title</role>
    </element>
    <elementRef>
      <topicAssoc>Degree</topicAssoc>
      <role>is-title-of</role>
    </elementRef>
  </members11>
</one2one>
```

We define the content of *type* attribute (of *one2one* element) as *attribute*, because the first one is an element and the second one is an attribute; the *element* specifies the topic *Professor* and its role, as well as the *elementRef* specifies the topic *Degree* and its role.

As said above, this kind of relationship also represents associations between an element and a subelement.

For instance, the relationship between the topic types *Professor* and *Group* illustrates another case of the one2one relationship because actually it relates one element (*prof*) with one of its subelement (*groupID*) that references the *group* element. Below we present the XSTM specification for that association *research*.

```
<one2one type="subelement">
  <type>research</type>
  <members11>
    <element>
      <topicAssoc>Professor</topicAssoc>
      <role>is-member-of</role>
    </element>
    <elementRef target="//prof/groupID">
```

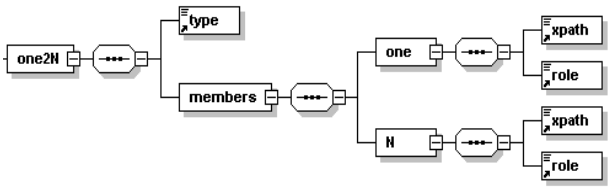

Figure 7: The one2N element.

```
      <topicAssoc name="./name" id="./id">
          Group</topicAssoc>
      <role>has-member</role>
    </elementRef>
  </members11>
</one2one>
```

We define the content of *type* attribute (of *one2one* element) as *subelement*, because the first one is an element and the second one is an subelement; the *element* specifies the topic *Professor* and its role, as well as the *elementRef* specifies the topic *Group* and its role.

In this case, there is one more attribute, *target*, of *elementRef*, whose value specifies the subelement (*//prof/groupID*) that makes the link between the two topics.

Notice that the element *topicAssoc* has two attributes: *name* to state the name to be shown by a visual tool; and *id* to specify the key of the index table.

4.5.2 Relationships 1 to N

This kind of relationship represents *one to many* associations between elements.

The *one2N* element is a sequence of an association type and the members of this association, as can be seen in the figure 7.

The *members1N* element (1N means 'one to N') is a sequence of *one* and *N* elements.

The *one2N* element has an attribute, named *split*, that specifies the format of this relationship. If the value of this attribute is `true`, it means that will be created an association for each occurrence of the second topic (referred in the *N* member) found in the input XML document. Otherwise, if the value is `false`, it means that will be created only one association grouping all the occurrences found for the second topic.

For instance, the association between the topic types *Department* and *Professor* illustrates a case of the *one2N* relationship, because actually it relates one element (*dept*) with multiple occurrences of the element *prof*. Below we present the XSTM specification for that association *work*.

```
<one2N split="true">
  <type>work</type>
  <members1N>
    <one>
      <topicAssoc>Department</topicAssoc>
      <role>employs</role>
    </one>
    <N>
      <topicAssoc>Professor</topicAssoc>
      <role>works-at</role>
    </N>
  </members1N>
</one2N>
```

4.5.3 Relationships M to N

This kind of relationship denotes *many to many* associations among elements.

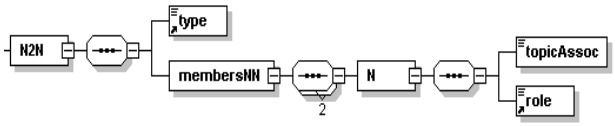

Figure 8: The M2N element.

The *M2N* element is a sequence of an association type and the members of this association, as can be seen in the figure 8.

The *membersMN* element (MN means 'M to N') is a sequence of two *N* elements.

The *M2N* element has an attribute, named *split*, that specifies the format of this relationship. This attribute has the same meaning as the one described in the previous relationship. If the value of this attribute is `true`, it means that will be created an association for each pair of topic occurrences (first or second *N* member) found in the input XML document (resulting MxN pairs). Otherwise, if the value is `false`, it means that will be created only one association grouping all the pair occurrences found for all the topics.

For instance, an association between the topic types *Professor* and *Course* illustrates a *M2N* relationship because actually it relates one element (*prof*) with the element *course*. Below we present the XSTM specification for the association *education*.

```
<M2N split="true">
  <type>education</type>
  <membersMN>
    <N>
      <topicAssoc>Professor</topicAssoc>
      <role>teaches</role>
    </N>
    <N>
      <topicAssoc>Course</topicAssoc>
      <role>is-taught-by</role>
    </N>
  </membersMN>
</M2N>
```

5. 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 9. 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[6] or Xalan[7], 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
```

[6] http://saxon.sourceforge.net/
[7] http://xml.apache.org/xalan-j/

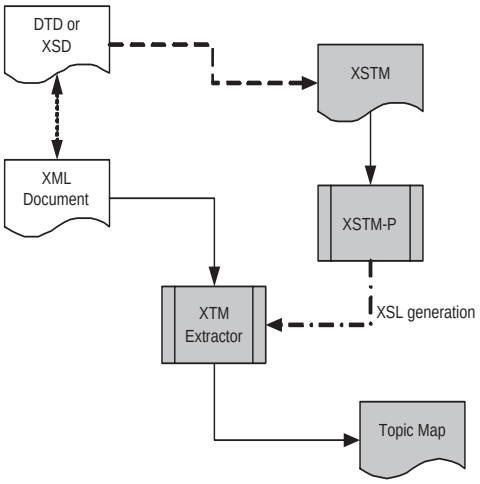

Figure 9: The XSTM architecture.

```
    xstm:topic;
  * create the occurrence types;
  * for each association type: create
    xstm:topic, which includes the members
    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;
```

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 file with 528 XSLT lines. The Topic Map extracted from a source file with 342 lines is a XTM file with 4205 lines[8] with 98 topics and 173 associations.

6. CONCLUSION

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 10 gives a schematic representation of this vision.

In XSTM, the ontology definition takes the same effort to specify as 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.

[8]The files can be found in http://alfa.di.uminho.pt/ grl/xstm

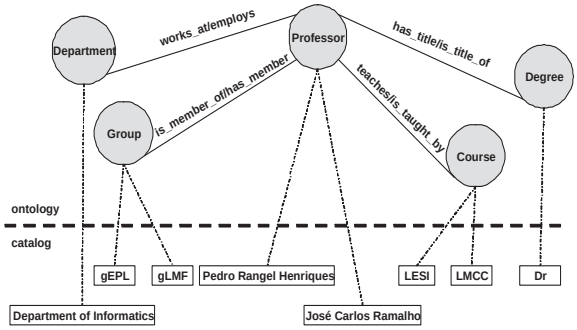

Figure 10: The case-study's ontology and catalog.

7. REFERENCES

[1] K. Ahmed, D. Ayers, M. Birbeck, J. Cousins, D. Dodds, J. Lubell, M. Nic, D. Rivers-Moore, A. Watt, R. Worden, and A. Wrightson. *Professional XML Meta Data*. Wrox Programmer to Programmer Series, 2001.

[2] M. Biezunski and S. R. Newcomb. Topic Maps Frequently Asked Questions. www.infoloom.com, September, 1999.

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

[4] J. Duckett, O. Griffin, S. Mohr, F. Norton, N. Ozu, I. Stokes-Rees, J. Tennison, K. Williams, and K. Cagle. *Professional XML Schemas*. Wrox Press, 2001.

[5] C. F. Goldfarb and P. Prescod. *XML Handbook*. Prentice Hall, 4th edition, 2001.

[6] E. Harold and W. Means. *XML in a Nutshell*. O'Reilly & Associates, 2001.

[7] S. Holzner. *Inside XML*. New Riders Publishing, 1st edition, 2000.

[8] J. Park and S. Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*. Prentice Hall, 2003.

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