Ontology driven Websites
Metamorphosis: a framework to specify and manage ontology driven websites

José C. Ramalho[1] **, Giovani R. Librelotto**[1*]**, Pedro R. Henriques**[1]

[1]Departamento de Informática – Universidade do Minho
Campus de Gualtar – 4710-057 – Braga – Portugal

{jcr,grl,prh}@di.uminho.pt

Abstract. *Website development has always been an hard task: it consumes time and resources. What is new today is normally taken as granted tomorrow by users. This is to say that users always want more. Today they want up to date information and they want to access it according to their point of view or particular preferences.*
To cope with these demands, websites must be dynamic and must be able to reconfigure automatically their structure, content and appearance.
This scenery has favored the creation of tools for automatic generation and management websites.
In this paper we propose not a new tool of this kind but a new approach to the problem.
In our approach we consider two layers. A physical layer that we call the resources layer, composed by databases, XML documents, directory subtrees, and the whole sort of files you can think of to represent your information. A metadata layer called the ontology layer, that provides a view to those resources.
Our framework consists of several parts. In this paper the focus will be the navigation component. This component takes an ontology and uses it to navigate through the resources layer. We are using XML technology to implement the whole framework and this component is implemented through an XML transformation process.

1. Introduction

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 (also called ontology) to navigate among web resources or simply among information resources.

The main idea behind Metamorphosis is to integrate the specification of these concept networks or ontologies with the navigation and their storage. We intend to do this using XML technology in every component of the system.

[*]Bolsista CNPq - Brasil

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. However there are other languages that should be considered like: RDF [Con99], RDFS [Con00], DAML [DAR01], OIL [OIL02] or OWL [Con02]. The most important are Topic Map (the XML version – XTM [tS01], 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 derived.

There are many tools that use ontologies to derive websites or to navigate through a set of resources. Those tools are normally implemented with a traditional programming language like JAVA or PYTHON. We achieved the same purpose with an XML transformation. The idea behind our navigational component is quite simple: we have defined a view over a point of the ontology; that view is composed by a series of information data-sets; some of those data-sets are hyperlinks; each hyperlink is a pointer to another XML transformation; each time a transformation is issued another view of the same ontology is presented to the user but positioned in another basepoint (the one passed as a parameter). This way to have a website running we just need a parametric transformation and an ontology specified in XTM.

In the next section we give a brief overview of the technical issues necessary to understand the rest of the paper: Topic Maps, XML, and XML transformations. Section 3. introduces the architecture of the environment we are developing. Afterwards, section 4., we discuss the implementation of the navigational component. Finally, section 5. presents one of the case studies used in the development of the system. The paper ends with some conclusions and describing the work to follow.

2. Background

Nowadays, the *World Wide Web* is one great human business, with members over the all world, handling and accessing an amount of information without precedents. But this it is not reflected in the current state of technologies being used to handle information. Most part of the access, extract, interpret, and maintain tasks available is user's responsibility.

The search engines are inefficient when they need to do complex inferences and to correlate subjects seemingly different. The simple annotation of HTML pages by <META> tags (or by use of metadata) is not sufficient to include the desired semantic. This Semantic Web will allow users to execute more sophisticated and more useful tasks then current ones.

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better in processing and ”understanding” the data that they merely display at present.

An important technology for developing the Semantic Web is already in place: *eXtensible Markup Language* (XML). XML lets everyone create their own tagshidden labels such as the ones that annotate Web pages or sections of text on a page. Scripts, or programs, can make use of these tags in sophisticated ways, but the script writer has to know what the page writer uses each tag for. In short, XML allows users to add arbitrary structure to their documents but says nothing about what that structure

means.

Meaning can be expressed by XML Topic Maps [tS01], which encodes it in topics and associations. These topics and associations can be written using XML tags. In XTM, a document makes assertions that particular things (people, Web pages or whatever) have properties (such as ”is a sister of” ”is the author of”) with certain values (another person, another Web page). This structure turns out to be a natural way to describe the vast majority of the data processed by machines. Subject and object are each identified by a Universal Resource Identifier (URI), just as used in links inside a Web page (URLs, Uniform Resource Locators, are the most common type of URI.) The verbs are also identified by URIs, which enables anyone to define a new concept, a new verb, just by defining a URI for it somewhere on the Web.

2.1. Ontologies

The component that represents and maintains these structured collections is the ontology. 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 [Gua97].

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 [ST99]. According to Chandrasekaran [Cha99], 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[1] has been accepted mainly as representation to the architecture of the Semantic Web.

The development of these mechanisms depends of 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, 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.

2.2. Topic Maps

A topic map 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 rela-

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

tionships between them. Information about the topics can be inferred by examining the associations and occurrences linked to the topic. A collection of 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 [tS01].

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 [GP01] usually compares Topic Maps to a GPS (Global Positioning System) applied to the information universe. Talking about Topic Maps is talking about knowledge structures.

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.

2.2.1. The characteristics of Topic Maps model:

Topics are the main building blocks of topic maps[PHE03]. 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". 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 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 [Pep00]. The occurrences can be an *Universal Resource Identifier* (URI). A topic occurrence represents the information that is specified as relevant to a given subject.

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. Occurrences establish the routes from the topics to the information resources, enabling also to provide the reason why that route exist. 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" [tS01]. The standard also defines occurrence roles as topics.

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" [Pep00].

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.

2.2.2. 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 to describe the TM building process, we will present a case-study where the subject is a workshop. This workshop *XML: Aplicações e Tecnologias Associadas (XATA)* took place at University of Minho, in February, 2003. Considering this example, we have four topic types: Institution, Paper, Participant, and Session.

In the following examples, we will assume that a participant’s name is *Pedro Rangel Henriques*, and that he is *Participant* that belongs to *Department of Informatics*, and he wrote a paper called *Especificação XML de Aplicações para WWW* that was presented in session *Dialectos XML*. The basic ontology therefore consists of the topic types *Participant*, *Institution*, *Paper*, and *Session*, the association roles *belongs-to/contains*, *is-in-the-session/contains the paper*, and *is-author-of/is-wrote-by*, and the association types *Participant and Institution*, *Paper and Session*, and *Author and Paper*.

First of all, 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="Participant">
    <baseName>
      <baseNameString>Participant</baseNameString>
    </baseName>
  </topic>
  <topic id="pedrorangelhenriques">
    <instanceOf>
      <topicRef xlink:href="#Participant"/>
    </instanceOf>
    <baseName>
      <baseNameString>Pedro Rangel Henriques</baseNameString>
```

```
    </baseName>
  </topic>
</topicMap>
```

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

Next we add the occurrence definitions, using the element (*resourceRef*) that contains the URL (resource address) as the value for the *xlink:href* attribute. 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="pedrorangelhenriques">
  ...
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#website"/>
    </instanceOf>
    <resourceRef xlink:href="http://www.di.uminho.pt/~prh"/>
  </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 define the *Author-and-Paper* association between *Especificação XML de Aplicações para WWW* and *Pedro Rangel Henriques*. The first one having the role *was-wrote-by* and the second *is-author-of*.

```
<association>
  <instanceOf>
    <topicRef xlink:href="#Author-and-Paper"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#was-wrote-by"/>
    </roleSpec>
    <topicRef xlink:href="#especificacaoxmldeaplicacoesparawww"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#is-author-of"/>
    </roleSpec>
    <topicRef xlink:href="#pedrorangelhenriques"/>
  </member>
</association>
```

The references *was-wrote-by* and *is-author-of* are association role types, and they are declared like a topic type, i.e., with a identifier and a base-name only. The reference *Author-and-Paper* is an association type, that defines the type of this association. The declaration of this topic is shown below:

```
<topic id="Author-and-Paper">
  <baseName>
    <baseNameString>Author and Paper</baseNameString>
  </baseName>
  <baseName>
```

```
    <scope>
      <topicRef xlink:href="#was-wrote-by"/>
    </scope>
    <baseNameString>was wrote by</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#is-author-of"/>
    </scope>
    <baseNameString>is author of</baseNameString>
  </baseName>
</topic>
```

The TM above explains that the *Pedro Rangel Henriques* is author of *Especificação XML de Aplicações para WWW* and, at the same time, indicates that the *Especificação XML de Aplicações para WWW* was wrote by *Pedro Rangel Henriques*.

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 technicians to marketing to information architects. Third, a lot of our work is about creating sites and prototypes of various designs and layouts, so to ease the process, we made *DINavigator* aware of relations and contextual menus. To bind all these, topic maps seem the perfect technology.

Also, even if there is a growing amount of tools available, none of them are simply distributed and installed, most of them requiring some other technology, software, library, programming language or utility to work. This is fine for die-hard technologists, but not need a knowledge of how to setup *Tomcat* with *Omnigator*, *TM4J* or *Nexist*, or install a *Python* tool like *SemanText*, or even figure out how to add a plug-in to *Protégé 2000*. Hence, *DINavigator* provides an out-of-the-box solution.

The XTM file is where all metadata about your site is stored, and the *DINavigator* framework uses this for all the structuring, relations and so forth you might have. The XTM file is not there for manual editing, but if you do, do note that the next build will overwrite any changes made to it.

3. Metamorphosis architecture

Although we are discussing *DINavigator*, the tool created to navigate among a semantic network. This navigation component is part of a bigger project called *Metamorphosis*.

The idea behind metamorphosis goes a little further than what was discussed so far. To give you a better idea let's pick a real example: suppose you several data resources like XML documents, relational databases or PDF documents; you want all these items to be accessible through the Web; at the first try you have built a complete index of all your data items; that index is huge and you know that there is no browser capable of displaying it; here is the point where you will start discussing the organization of your data resources; and it is here that we introduce you to *Metamorphosis*; you do not need to change anything in your data resources; you just need to create an ontology for your data resources.

Figure 1 further illustrates this idea and gives an idea of *Metamorphosis* architecture.

This architecture can be described as follows:

(1) Resources layer This component is composed by our data resources: XML documents, Web pages, Databases, ... Metamorphosis does not interfere with any of it, it will only use part of the information to build the semantic network.

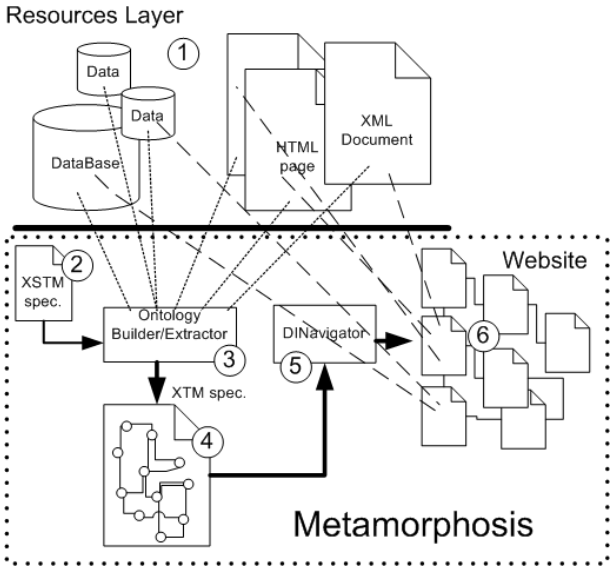

Figure 1: Metamorphosis

(2) XSTM spec This XML document gives precise information on where to go to extract the relevant information; this component is not being discussed in this paper.

(3) Ontology Builder/Extractor This component uses the XSTM spec and retrieves the information it needs to build the ontology from the data resources. It is implemented as a XSL stylesheet and it takes advantage of information organization: for instance, when processing a relational database it looks for the relations information and automatically generates all the associations between the relevant topics.

(4) XTM spec This is the topic map specification automatically generated by the ontology builder.

(5) DINavigator This is the component being discussed in this paper. Also implemented in XSL. It takes a topic map and produces a whole website according to some principles.

(6) Website The final generated website through which is possible to navigate semantically through the data resources.

In the next section we are going to discuss the navigational component and the website generation.

4. DINavigator - the navigational component

At this point we can say that ontologies, specified with XTM, are a set of records, each record represents a concept, it points to some resources (physical information records) and participates in several relations (associations). So, using these relations between concepts to query and navigate through information concepts (first) and then through information resources (secondly) seems the right way to do it.

The main idea about navigation can be described as: when you are positioned at a certain concept the navigation framework shows you a particular concept, the resources it points to, and all the concepts related to it; if you choose one of the related concepts the position changes to that concept and the view will change accordingly; if you choose one of the resources the system will show that resource view.

In terms of implementation we wanted web browsers to be our navigators. This way all the views discussed above would be HTML pages. So, the implementation of this navigational component

is reduced to an XML transformation. However, there were two possibilities: runtime or batch transformations.

4.1. Runtime transformations

Runtime transformations were implemented with an XML transformation inside a CGI script. The initial page is a call to the CGI parametrized with the main concept. The CGI applies the transformation to the ontology and produces the HTML view. In this view all the links are CGI calls with a different parameter, the new position inside the ontology.

This is a nice solution because it only needs two files, the ontology (specified in XTM) and the CGI. However, runtime transformations are time consuming. This solution is being used in small prototypes and is still under development.

4.2. Batch transformations

This solution corresponds to the actual component being used in *Metamorphosis*. We have created an XSL stylesheet that transforms the ontology into a website. This website corresponds to all the possible views of the ontology.

Though we can think of *DINavigator* as a website generator, based on XSL and using topic maps for this purpose. It is made to be a simple way of creating full sites, with design, content and topical links.

All in all; the words are ”simple” and ”powerful.”

DINavigator was created as a direct consequence of creating a similar framework in XSLT for the XSTM [LRH03] language. When this language was developed, and as the need for a good prototyping-tool emerged at his work, the time was right to look for a way of making all these come together. *DINavigator* was created out of three reasons;

- Be an effective prototyping tool in his work, and lowering the time from prototype to production.
- To create a tool that uses Topic maps, and that is easily distributed, installed and used by all.
- To bridge the technical languages and methods of technicians, programmers, designers and information architects.

DINavigator offers a framework for making rapid changes to a whole site, be it page setup, content, topics, relations or added/deleted pages. Since *DINavigator* is parsed through a standards-compliant XSLT parser, the result, through careful auditing of all accompanying HTML, is production ready. This simply means that you can quickly create prototypes, and simply expand them using the same tool to go into a production environment. This radically decreases development cost and time.

Topic maps is used in *DINavigator* for all structures, relations, content preparation, resources and occurrences, naming and ontological expression. This means that any other topic maps able tool out there can grab the topic map file from *DINavigator* and view its full metadata map.

5. Our Case Study: XATA 2003

This section presents a case study about the workshop *XML, Aplicações e Tecnologias Associadas* (*XATA*[2]). This case study demonstrates the use of *Metamorphosis*. This workshop happened at University of Minho, in Braga, Portugal, at February 13-14, 2003. XATA gathered the XML portuguese

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

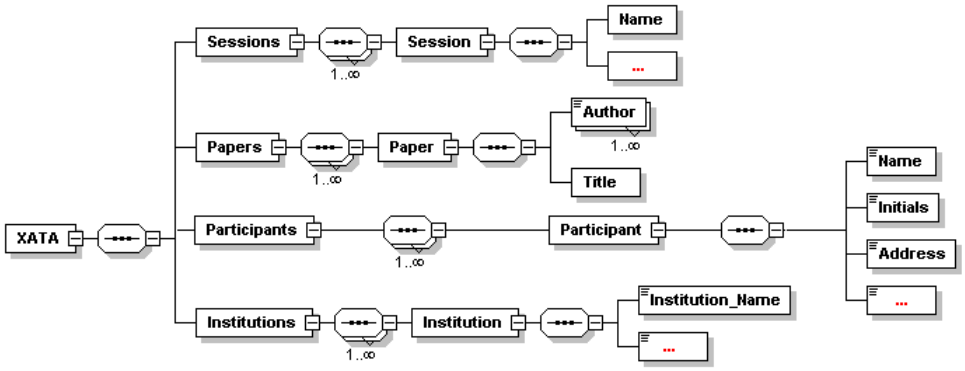

Figure 2: XATA's XML-Schema.

community. Researchers and users of XML from university or companies have participated. Thus allowing a sharing of information between the academic world and the professional world.

In this workshop, many papers were submitted to evaluation; the accepted ones were presented in the conference. The papers presentations were separated in sessions, each one with an associated theme, like *Tecnologia e Web Services* (Technology and Web Services), *XML e Base de Dados* (XML and Databases), among others.

The event was all XML based, since its diffusion, until its production. Therefore, all the information referents to the *XATA* are stored in XML documents. The XML-Schema of the workshop is presented in the figure 2. Obviously this XML-Schema is incomplete, but the important items to this case study are in the figure.

As the XSTM language only depends on document structure, and not on the XML instance, the ontology specification of XATA can be defined from this XML-Schema. So, there are five steps: definition of the 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 to 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 are 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>
            <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 XSTM specification to the topic definition referent to the Paper topic type is presented below:

```
<topics>
    <topic>
        <xpath>//Paper/Title</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>ID-author_paper</id>
        <name>Author and Paper</name>
        <memberAssoc>
            <scope>ID-was-wrote-by</scope>
            <description>was-wrote-by</description>
        </memberAssoc>
        <memberAssoc>
            <scope>ID-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 *//Artigo/Autor*, which is a reference to the initial letters of authors name found in *//Inscritos/Iniciais*. 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>ID-author_paper</type>
            <members11>
                <element>
                    <topicAssoc>Paper</topicAssoc>
                    <role>ID-was-wrote-by</role>
                </element>
                <elementRef target="//Paper/Author">
                    <topicAssoc id="./Initiais">Participants</topicAssoc>
                    <role>ID-is-author-of</role>
                </elementRef>
            </members11>
        </one2one>
    </assocs>
    ...
</xstm>
```

Figure 3, shows the Topic Map visualization within DINavigator. The input topic map was created by TM-Builder. This navigator gives the total access to the extracted ontology, allowing the navigation through the concepts defined in the XSTM specification.

6. Conclusion and Future Work

Metamorphosis is being used in several small to medium projects, to build web interfaces to information systems.

Until now it has proven to be a useful prototyping tool. The web interfaces are created quite easily and fast.

There are some situations where this system has helped a lot:

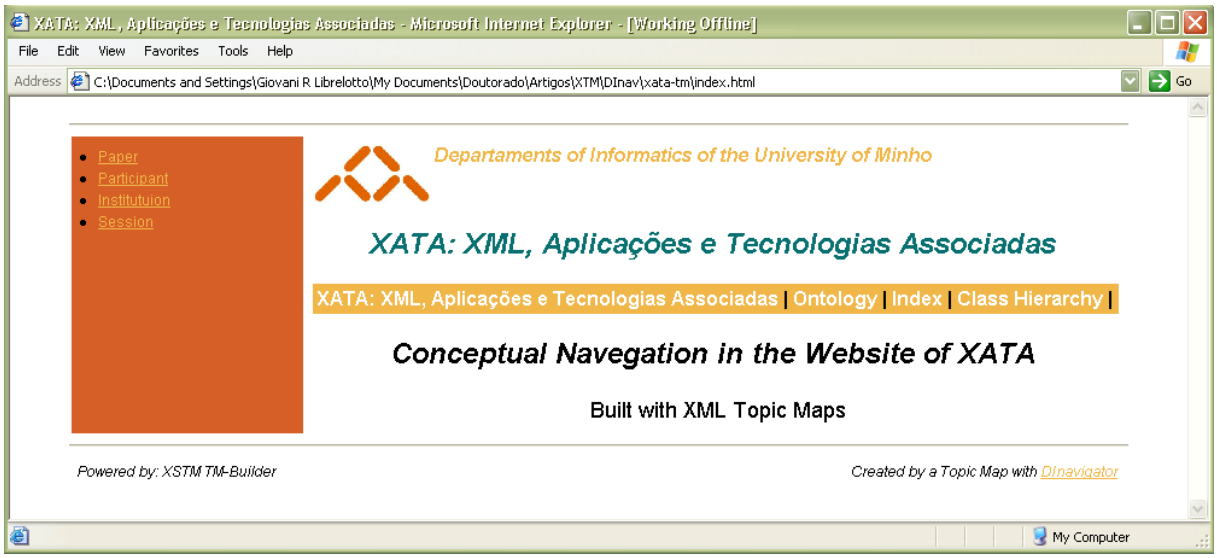

Figure 3: Topic Map visualization in the DINavigator.

- suppose you have an heterogeneous information system;
- you want to make it accessible through the web;
- when you start creating your HTML index pages you end up with pages of about 5 Megabytes;
- web browsers can not hold pages above 1 or 2 Megabytes;
- you have a problem ...

We have been solving similar problems creating an ontology for the intended information system and using *Metamorphosis* to manage that interface.

However *Metamorphosis* is not finished:

- we are starting to test it with big XML documents (up to 25 Megabytes);
- we are adapting it to work with relational databases and other sources of information, besides XML documents;
- we are studying a relational model for ontologies (if your ontology grows the XML file would no longer be the proper way to store it);
- we are working in new versions for *Metamorphosis* components enabling them to work with new forms of representing information.

In the moment a web page is being created for *Metamorphosis* and should be announced soon.

References

B. Chandrasekaran. *What Are Ontologies, and Why do We Need Them?*, volume vl 14, n 1. IEEE Inteligent Systems and their applications, Jan/Fev 1999.

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

World Wide Web Consortium. Resource Description Framework (RDF) Schema Specification 1.0., March, 2000. Avaliable in http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

World Wide Web Consortium. Web Ontology Language (OWL) Reference Version 1.0, November, 2002. Avaliable in http://www.w3.org/TR/owl-ref/.

DARPA. DAML. Darpa Agent Markup Language Program., 2001. Avaliable in http://www.daml.org/.

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

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.

Giovani Librelotto, José C. Ramalho, and Pedro R. Henriques. XML Topic Map Builder: Specification and Generation. In *XATA: XML, Aplicações e Tecnologias Associadas*, 2003.

OIL. Welcome to OIL, 2002. http://www.ontoknowledge.org/oil.

Steve Pepper. The TAO of Topic Maps - finding the way in the age of infoglut, 2000.

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

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

topicmaps.org Specification. XML Topic Maps (XTM) 1.0, Mar, 2001.