Using Web Services to put Metamorphosis on the Web

Giovani Rubert Librelotto, Jonas Bulegon Gassen
UNIFRA – Centro Universitário Franciscano
Rua dos Andradas, 1614, Santa Maria, RS – Brazil
{librelotto,jbgassen}@gmail.com

and

José Carlos Ramalho, Pedro Rangel Henriques
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
{jcr,prh}@di.uminho.pt

Abstract

This paper describes the exposure on the Web of *Metamorphosis* main functionalities. This exposure will be accomplished through an XML Web Service. *Metamorphosis* is a Semantic Web tool aiming at the automatic creation of knowledge views for heterogeneous data sources or information systems. Each view is characterized by an ontology that corresponds to a semantic network composed of concepts and relations that are built with data extracted from several data sources.
There are several formal notations for knowledge representation; Topic Maps were chosen due to their advantages like: having a frozen syntax which enables the creation of tools like extractors and navigators; and being abstract enough to specify anything (everything can be represented as a topic and these topics can relate to each other).
The knowledge views are the final result of *Metamorphosis* but this tool has four different stages. We intend to expose each one of these stages. This way the desired XML Web Service will have four different modes, offering the following functionalities:

f1: TM-Builder - the user provides a data source specification, and the Service will return an extractor for that data source.

f2: TM - the user provides a data source specification together with the data source, and the Service will give the user the extracted Topic Map.

f3: WebSite - the user provides a data source specification together with the data source, and the Service will send back the Website.

f4: Remote WebSite - the user provides a data source specification together with the data source, and the Service will give the user an access point (URL) to the Website that will be hosted by the server.

In the paper we will give enough information to understand *Metamorphosis* and all the layers developed so far. The four exposed functionalities of *Metamorphosis* will be discussed in detail.

Keywords: Web Services, Topic Maps, Ontology Extraction, Ontology-driven websites, XML, XSL.

1 Introduction

Two years ago we start exploiting the Topic Maps concept with the idea of easily creating intelligent web interfaces (semantic navigators) for any data source.
With that aim in mind we picked a set of case studies and started developing a software architecture that we called *Metamorphosis*. *Metamorphosis* [8] has several components: a topic map extractor, an automatic web site generator, and a constraint checker processor.
One thing that is quite evident to whom works with knowledge management is that the creation of ontologies, in this case a topic map, is hard. To ease this task we created the ontology extraction component called *TM-Builder*. This component has several layers and enables the user to specify which parts of the data sources should be retrieved to build the intended topic map.
The navigational component, called *Ulisses*, is composed of an interface generator. At present it generates a generic web interface using the information in the topic map. This interface can later be used to navigate throw the topic map, and to access information in the data sources.
The constraint checker component is still under development but will verify if the actual topic map instances complies all the constraints settle down in the topic map specification. This component will not be discussed in this paper.
The first version of *Metamorphosis* was developed inside a complete XML framework with the advantage of being completely standard and platform independent and the disadvantage of only being able to process XML data sources. The only thing a user has to do is to specify the ontology the topic map instances extraction schema. From this specification, *Metamorphosis* will generate a topic map, and from this topic map it will generate a web interface that can be used to navigate among information sources.
XSTM (XML Specification for Topic Maps) language describes the process of the knowledge extraction from a class of XML documents to produce a topic map.
The aim of this paper is to make the *Metamorphosis* available on the internet with Web Services. Joining the characteristics of this technology and *Metamorphosis*, any kind of application can generate its own topic maps or/and website conceptual based on topic maps from XML files.
This paper begins with a presentation the basic concepts on this area: Ontologies and Topic Maps, in Section 2. Section 3 describes the *Metamorphosis* and their processors: *XSTM-P* – the XSTM processor that creates a topic map extractor; *TM-Builder* – the topic maps extractor for XML resources according to a XSTM specification; and *Ulisses* – the conceptual navigator topic map-based. The description of the integration of *Metamorphosis* and Web Services is found in the Section 4. A synthesis of the paper and hints on future work are presented in the Section 5.

2 Ontology and 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 [4] 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.

2.1 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 [11]. The most known XML version – XTM [12] – is being widely used and became recently an appendix to the ISO standard.
Topics are the main building blocks of topic maps [10]. 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 [11]. 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" [11].
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 Metamorphosis

To understand the design of *Metamorphosis* [9] (its architecture and flowchart), it is important to remember the motivation for its development aiming at providing access to the information contained in a large collection of XML documents though the Web, it is mandatory to catalog the resources.
Creating a complete index of all the data items in the documents gives rise to a practical problem: that index can easily scale up to such an huge size that no browser will be able to upon it.
To overcome that problem, we decide to create an ontology to control the access to the referred data items. According to the benefits of Topic Maps this representation was elected to describe the desired ontology. In that control, we have identified the need for a tool to help in the creation of the topic map and in the production of the HTML pages to browse it.
The first experiments with those tools pointed out the possibility and the usefulness of developing another tool do improve the extraction of the topic map, that was precisely the origin of *Metamorphosis*.
Figure 1 illustrates this idea and shows the *Metamorphosis* architecture.
This architecture can be described as follows:

(1) Resources layer: This stage is composed of data resources, more specifically XML documents. *Metamorphosis* does not interfere with any of it, it will only use part of the information to build the semantic network.

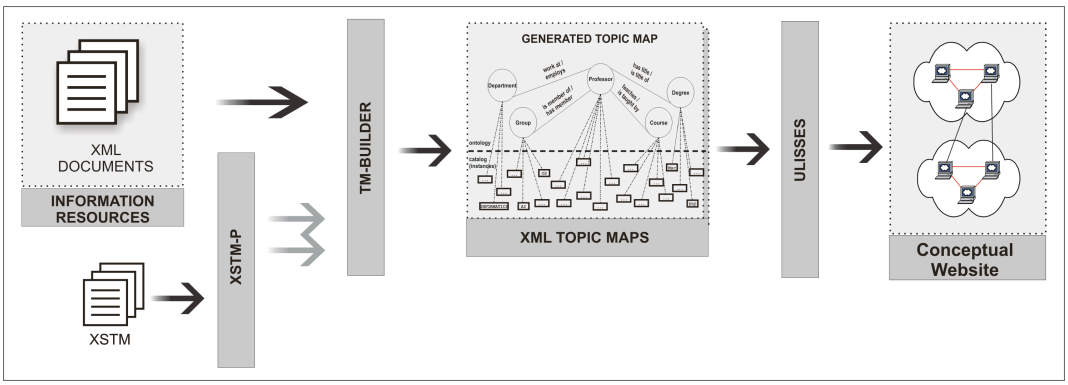

Figure 1: Metamorphosis Architecture.

(2) XSTM specification: This XML document defines the topic map (topics and associations) and gives precise information on where to go to find data items (inside the resources) to create topic instances.

(3) XSTM-Processor: This module, an XSL stylesheet [14], takes the XSTM specification, as input, and produces, as output, a concrete topic map builder (*TM-Builder*), this is, another tool that will perform the construction of the desired ontology for that specific family of XML documents.

(4) TM-Builder: This component retrieves the information it needs to build the ontology from the data resources. It is implemented as a XSL stylesheet.

(5) XTM ontology: This is the desired topic map specification automatically generated by TM-Builder.

(6) Ulisses: Also an XSL stylesheet [5], this tool produces an HTML interface to navigate on the ontology generated.

(7) Conceptual Website: The HTML file (actually a set of them), produced by *Ulisses*, is in fact a semantic website throw which we can browse the data resources in a conceptual manner drive by the ontology.

In the next sub-sections we will detail the three *Metamorphosis* processors: *XSTM-P* and *Ulisses*, the built-in components; and *TM-Builder*, the generated extractor.

3.1 TM-Builder: the ontology extractor

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 concepts and relations from a set of XML resources, or by other words, a topic map builder.
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 [3]. Kal Ahmed [1] wrote chapter about *Automated/Automatic Topic Map Construction* but it does not explain how it is possible to implement this topic map's constructor.
So, we developed an XSL tool that reads a family of XML resources and builds a topic map – *TM-Builder* [7]. Moreover, we intend to show that such a tool can be generated automatically instead of creation by hand a new one each time the documents typed changes.
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.
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 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:

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

- 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* to extract a Topic Map from an XML document is: initially, for the given ontology creates all the topics types, occurrences roles, occurrences types, and associations types; after, during a document tree traversal, for each association, define the: association type and association members; and for each element in the source that is seen as a topic, create the: topic ID, topic type, topic names, and topic occurrences. In our system, this 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 present the XSTM; this language is used to specify the extraction process.

3.1.1 XSTM: an XML Language to specify topic map extractors

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.
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.
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        ::= 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. Notice that the XSTM DTD is obtained direct and systematically from the CFG shown.
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 leaf-tree topics), 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.
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 complexity of its DTD. So, if this XML document grows up, the same *TM-Builder* is able to process it.

3.1.2 An editor for XSTM

To facilitate the XSTM edition, we have created a web interface to produce this kind of specification. The user only needs to complete some steps for an automatic creation of the required file. After that steps, XSTM specification – that will represent the concepts and the relationship according to its point of view – will be created.
The XSTM file creation process is divided in six steps :

Schema definition: In this step, the user specifies a schema of XML documents; which elements (and their attributes) he wants to have into XML files.

Topic Maps hierarchy definition: Now, the user clearly describe the topic map hierarchy tree: the topic types and superclass/subclass relationships.

Topic definition: In a list of all elements created or loaded, the user checks a checkbox to declare if the selected element will be topics. In addition, it is necessary to define what kind of topic each one is: its topic types. This way, each topic is being inserted into the topic map tree. When submitted this information, the system automatically examines the XML document in memory and gets all XPath [6] address of the selected topics.

Topic characteristics: Here, the user must to define the topic characteristics – the subject identify, the base names (including their type, scope, and content), and the occurrences (including their type, scope, references, and data). In a form, the user must to associate an element or attribute to each one of these topic characteristics. This element (or attribute) will provide the information to fill the topic characteristic in. The system determines the XPath path to each element automatically.

Association types definition: In this step, the user defines the association types. There are 5 kinds of relationship[2] between topics: "is a", "contains", "1 to N", "M x N", and "has 1 or more". Each one of these relationships can be associate two or more topics. Each topic that participates in an association plays a role in that association called the association role. So, in a form the user choices 2 or more association roles and selects their respective relationship. The result of this process is all the association types possible in the final topic map.

Topics association: A topic association asserts a relationship between two or more topics. That way, in this step the user selects two or more topics and he defines which association type (previously defined) provides semantic to this relationship. A topic can have relationship with more than one topic and it can be related in several associations; with this definition is possible to get all information for a topic in the semantic network composed of the associations.

After all, the created specification is saved in a XSTM file containing the representation and characteristics of all topic and associations. The user can download that generated XSTM file. The XSTM specification will be parsed by *XSTM-P* processor (an XSL file that generates another XSL) and the system creates a *TM-Builder* according to the user definition. The generated *TM-Builder* is then ready to process a family of XML documents and to produce the topic map file.
At this moment, the user has two options:

- If the user wants to execute the topic map generation into the server; that way he needs to send the XML files to the server, after processing the XML sources, the server returns the final topic map to the client;
- If the user wants to perform the extraction in his machine, he should download the *TM-Builder* produced. This is the best option to keep the privacy because the XML sources are not sent to the server. With the *TM-Builder* in his machine, the user can parse the XML files generating the topic map.

3.2 XSTM-P: the TM-Builder generator

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) [7] is the *TM-Builder* generator; it is one of the main pieces in our architecture, as can be seen in Figure 1. 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[3] or Xalan[4], what in our opinion is one of the benefits of the proposal.

[2] All these relationship is described in details in [7], including a case study.
[3] http://saxon.sourceforge.net/
[4] http://xml.apache.org/xalan-j/

3.3 Ulisses: the conceptual web site generation

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.
XML Topic Maps allow the browsing of the information of a certain domain driven by an ontology. That semantic driven web site is obtained using *Ulisses*, a conceptual navigator for XML Topic Maps.
Ulisses was created as a direct consequence of creating a similar framework in XSLT for the XSTM language. This component takes an ontology and uses it to navigate through the resources layer. 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. *Ulisses* 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.

Topic maps is used in *Ulisses* 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 *Ulisses* and view its full metadata map.
To produce a web site from a topic map, *Ulisses* follows a systematic conversion process that can be formally described by a mapping from topic map domain into HTML domain. Table 1 shows this mapping that supports the generation. For instance, we can see that a topic will be translated into a new HTML home page, while the topic occurrences will be the text, images, logo, etc, that constitute the body of that page.

Topic Map	Web Site
Topic	Web Page
Topic Associations	Site map
Occurrences	Text, Images, Logo, HTML fragments, External Links
Topic Names	Page Headers, Titles, Hyperlinks

Table 1: Mapping between Topic Maps and web sites

Ulisses offers a develop environment for making rapid changes to a whole site, be it page setup, content, topics, relations or added/deleted pages. A standards-compliant XSLT parser analyzes the grammatical structure of the XTM document, generating a set of HTML files as output. 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.
So we can think of *Ulisses* as a web site generator, based on XSL and using topic maps for this purpose. It was conceived to be a simple way of creating full sites, with design, content and topical links. Therefore, the topic map obtained with *TM-Builder* will be the input for *Ulisses*; the output is a complete web site in HTML. 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.

4 Metamorphosis via Web Services

After developing *Metamorphosis*, described in Section 3, we were convinced that it is not just an interesting and challenging system, but it can be useful for other people working on this area. So we decide to share with the community its functionalities. That decision implies the distribution, installation, and maintenance of numerous copies of the platform.
To give access to the services provided by *Metamorphosis* without delivering the platform to every consumer, we propose the use of Web Services technology. In that way, users and applications can take profit of the *Metamorphosis* facilities without need to get and install the system; moreover just standard protocols are necessary to communicate with the server.

According to [13], Web Services can be defined as "a software component identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols".
Web Services are a new generation of Web applications. They are self-contained, self-described, modular applications that can be published, located, and invoked across the Web. Web Services perform functions, which can be anything from simple requests to complicated business processes. Once a Web Service is deployed, other applications (and other Web Services) can discover and invoke the deployed service.
All data traffic generated by Web Services is transmitted over HTTP protocols. Every message, sent or received, by a Web Service is packed in a XML-based protocol (that can be transmitted by HTTP protocol) encapsulating all data to be exchanged with the other applications. In this way, data circulates over the internet, sharing information across different systems and platforms; all data transmitted can there be understood in the other side through XML parsers.
The Web Services technology was chosen because:

- Web Services are a W3C standard, what means that any software developer can produce solutions using this technology;
- Web Services are built on industry standard protocols, such as TCP, SSL, XML, SOAP, and WSDL;
- Web Services do not requires a complex infrastructure in place to be developed. The only real requirements are a Web server and a connection to the Internet;
- All data traffic is made through XML protocols, what means that all data is sent over HTTP;
- All devices can access data through Web Services using. The only requirement is that the language support it.

The envisaged Web Service will make *Metamorphosis* available in four different modes. Each mode offers a different functionality, as described in next section.

4.1 f1: TM-Builder

As can be seen in Figure 2, the Client develops the XSTM specification and just sends it to the Web Service, which returns the *TM-Builder*. That way, the Client is responsible for the subsequent processing.

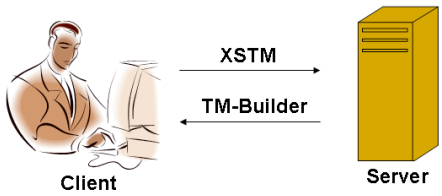

Figure 2: f1 mode

Step by step, the usage protocol[5] for **f1 mode** is:

C: Uploads the XSTM specification;

S: Generates the TM-Builder and to sent it to client;

C: Extracts a topic map with the TM-Builder;

C: Ulisses produces a website with the topic map;

C: Puts the website online.

This functionality is recommended when the Client has enough computational resources. The **f1 mode** advantages are:

- Data privacy, as the extraction is executed in client side, a copy of XML sources do not need to be sent to the server;
- Offline extraction, as it is processed in the client side, than is no need to be online;
- Autonomous update, the topic map can be result after changes (insertions, deletions, or alterations) in data sources, without the need to call the Web Service for regeneration of *TM-Builder*. The new construction of *TM-Builder* is just necessary if the document's schema changes.

[5] **C** means *Client* and **S** means *Server*

4.2 f2: TM

When the Client does not have power requirements to process (or if the user does not want to do it), the job can be sent to the Server, that makes the conversion and provides the option of download the results or stores within it.
As can be seen in Figure 3, the Client develops the XSTM specification and sends it together with XML sources. Therefore, the Server produces the *TM-Builder* and processes it, taking as input the XML resources. After all, the Client downloads the generated topic map.

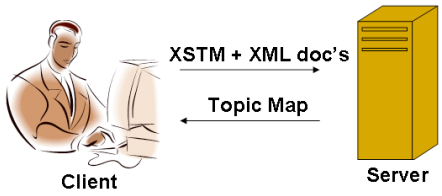

Figure 3: f2 mode

Step by step, the usage protocol for **f2 mode** is:

C: Uploads the XSTM specification;

C: Uploads the XML sources;

S: Generates the TM-Builder and to produce a topic map;

C: Downloads the topic map;

C: Ulisses produces a website with the topic map;

C: Puts the website online.

The main advantages of this functionality is that all topic map transformations (*XSTM-P* and *TM-Builder*) are made in the Server. If the Client only wants a topic map, this functionality is the best choice.

4.3 f3: WebSite

In this functionality, the Server is responsible for all XSL transformations (*XSTM-P*, *TM-Builder*, and *Ulisses*). The Client sends the XSTM specification and XML sources, like in **f2 mode**. But here, the Server returns the complete website, as can be seen in Figure 4. Hence the Client can publish this website as he wishes.

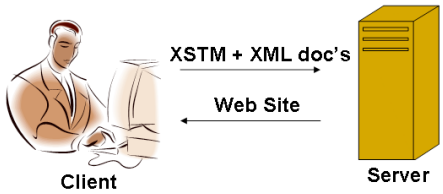

Figure 4: f3 mode

Step by step, the usage protocol for **f3 mode** is:

C: Uploads the XSTM specification;

C: Uploads the XML sources;

S: Generates the TM-Builder and to produce a topic map;

S: Ulisses produces a website with the topic map;

C: Downloads the website;

C: Puts the website online.

This functionality is recommended when the Client has not enough computational resources. The **f3 mode** main advantage is that the Server processes all XSL transformations, generating the complete website for the Client.

4.4 f4: Remote WebSite

As can be seen in Figure 5, the Client sends an XSTM specification, the XML sources, and an authorization to the Server, which will generate and publish a website in an own domain like `http://www.topicmapsfactory.com/userlogin`, where *userlogin* represents the login name of the Client.

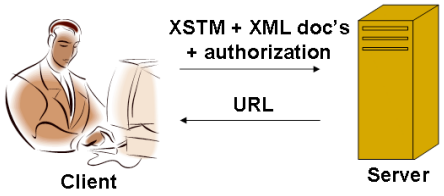

Figure 5: f4 mode

Step by step, the usage protocol for **f4 mode** is:

C: Uploads the XSTM specification;

C: Uploads the XML sources;

C: Sends an authorization to publish the website;

S: Generates the TM-Builder and to produce a topic map;

S: Produces a website with the topic map;

S: Puts the website online;

C: Gets URL to the website.

This functionality is recommended when the client has not enough computational resources and/or wants to expose his website. The **f4 mode** advantages are:

- Client does not need to have strong processors and memory;
- Server processes all XSL transformations: *TM-Builder* and *Ulisses*.
- Server publishes the generated website.

The choice of the most appropriate mode for a particular case depends on several factors that should be carefully balanced. The computational resources available, the desired autonomy (independence from the server), the ability to host a web site, are examples of those factors that will influence the decision of the consumer.
However, no matter what choice is taken, the client procedure to access and consume the service in the same, and is based on standard XML protocols. If it is not difficult to design and implement the Web Service, it is even easier to program the client application.

5 Conclusion

Metamorphosis is a XML-based system that produces automatically a semantic, on conceptual, navigator to browse XML documents. These semantic views are expressed as topic maps: a conceptual network where topics are connected by associations. *Metamorphosis* builds the topic map extracting data items from the information resources and linking them according to the XSTM specification. Then, from a topic map, *Metamorphosis* generates automatically a complete semantic web site that can be used to navigate among information sources, driven by the concepts in the ontology.
Metamorphosis is made up of three processors: *XSTM-P*, *TM-Builder*, and *Ulisses*. A module to check for the semantic integrity of topic maps, based in a constraint specification, is under development and was not discussed in this paper.
Still talking about 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 deal with XML documents or databases in a transparent manner. It will be able to save the topic map as an XTM file, or in a new database, structured according to XTM format. However, it will rely upon the same XSTM language for the specification of the topic maps.
In this paper we proposed the use of XML Web Services to make available on the Internet a platform, called *Metamorphosis*, for the automatic construction of topic maps.

The use of Web Services provides a lot of advantages like interoperability, ubiquity, just-in-time integration, industry support, and etc., due to the fact that is a free international standard and its open protocol can be implemented by any software developer, do not matter the operating system or the programming language.
In the approach discussed along the paper, the Web Service plays an important role; it allows us to offer the complete functionality (four modes) of *Metamorphosis* avoiding the distribution of the system and all the implied coast and maintenance problems.

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] Michel Biezunski and Steven R. Newcomb. Topic Maps Frequently Asked Questions, September 1999. `http://www.infoloom.com/faq.htm`.

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

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

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

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

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

[8] Giovani Rubert Librelotto. *XML Topic Maps: da Sintaxe Semntica*. PhD thesis, Departamento de Informtica, Escola de Engenharia, Universidade do Minho, 2005.

[9] Giovani Rubert Librelotto, Jos Carlos Ramalho, and Pedro Rangel Henriques. Metamorphosis - A Topic Maps Based Environment to Handle Heterogeneous Information Resources. In *Lecture Notes in Computer Science*, volume 3873, pages 14–25. Springer-Verlag GmbH, 2006.

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

[11] 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`.

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

[13] Jeffrey C. Schlimmer. Web Services Description Requirements. World Wide Web Consortium, October 2002. `http://www.w3.org/TR/ws-desc-reqs/`.

[14] Doug Tidwell. *XSLT*. O'Reilly, 2001.