Using the Ontology Paradigm to Integrate Information Systems Oveia: Expanding the Topic Maps frontier

Giovani R. Librelotto
University of Minho
Dept. of Informatics
Braga, Portugal
grl@di.uminho.pt

Weber Souza
UNEB
Dept. Ciências Exatas
Salvador, Brasil
wsouza@nexen.com.br

José C. Ramalho
University of Minho
Dept. of Informatics
Braga, Portugal
jcr@di.uminho.pt

Pedro Rangel Henriques
University of Minho
Dept. of Informatics
Braga, Portugal
prh@di.uminho.pt

***Abstract*—Ontology based websites are one possible implementation of the Semantic Web. There are several languages for ontology specification: RDF, OWL, Topic Maps. Topic Maps follow a structure formally specified what makes them a good choice for semantic website specification. The process of ontology development based in topic maps is complex, time consuming, and it requires a lot of human and financial resources, because they can have a lot of topics and associations, and the number of information resources can be very large. To overcome this problem a new environment is proposed, Oveia. Oveia is composed by four components which have relevant contributions to the Semantic Web area. This paper describes these components in detail. Two components representing a metadata extractor: heterogeneous data integration (through XSDS specifications) and an homogeneous intermediate data representation for the extracted metadata (datasets). The Ontology builder who builds an ontology from metadata stored in a set of datasets (construction rules are specified in a new domain specific language: XS4TM). The Ontology builder stores the result in XTM files or in relational databases according to the Topic Map structure. Finally, *Ulisses*, the navigational component, generates web interfaces through which is possible to move inside the topic map and among information resources.**

Keywords: *Semantic Web, Ontology, Topic Maps, Ontology Extraction, Conceptual Navigation.*

I. Introduction

Typically, on a daily basis a lot of data is produced in an institution or in a company. To satisfy the storage requirements, these organizations use most of the times relational databases, which are quite efficient to save and to manipulate structured data.

On the other side, The Topic Maps paradigm – ISO/IEC 13250 – is an attractive choice to represent complex structures of knowledge. Topics and associations compose a structured semantic network over information resources. This hierarchical topic network is known as ontology.

Topic Maps are being nowadays increasingly accepted as an excellent option when it is necessary to organize information according to different points of view simultaneously. Topic Maps are designed to manage the infoglut, building valuable information networks from an overabundance of information; the approach enables the structuring of unstructured information resources of any kind. A topic map can be seen as a collection of interconnected electronic indexes.

According to [8], Topic Maps is more high-level than RDF (Resource Description Framework) [4]. In RDF resources have properties that have values (which may be other resources). In topic maps, topics have characteristics of various kinds: names, occurrences and roles played in associations with other topics. The essential semantic distinction between these different kinds of characteristic is absent in RDF. This difference in level of semantics also accounts for the fact that a useful generic browser – like *Ulisses* – can be built for topic maps, but not for RDF.

An ontology extractor supported by Topic Maps was presented in [5]. This environment processes a set of XML documents – belonging to the same family (DTD or XML-Schema) – with a processor (TM-Builder) created from an ontology specification in XSTM (XML Specification for Topic Maps). This TM-Builder process generates an XML Topic Map (XTM). So, this platform is fundamentally XML based.

So, when the information resource is not a XML document, a conversion of this source to an XML file was necessary. This task is not the best way to do the extraction because the data synchronization is complex; when the resource is modified, it was necessary to regenerate the XML file with the new content.

A Topic Map Builder (named Oveia) was developed to avoid the database dump and data synchronism between the database and the dumped XML file. Oveia extracts the data directly from information resources. In this way, the generated topic map is stored in a database structured accordingly [2], in addition to allow the storage in an XTM file. This target database is called *OntologyDB* and its design follows the standard Topic Maps concepts and some rules published in [10] to do the conversion between XML models and relational databases.

The extracted ontology is specified in XS4TM (XML Specification for Topic Maps). This language has the same objective that XSTM but it has several upgrades. XS4TM covers all XTM elements, and it follows closely the concepts present at XTM. XS4TM is designed for ontology extraction in heterogeneous information resources.

In section II there is an overview about Semantic Web and Topic Maps. It briefly introduces the subject and it shows some of its characteristics. The proposed architecture and its main characteristics are presented in section III. The section IV presents the conceptual navigator, called *Ulisses*. The Oveia application to a concrete case study is discussed in section V. The comparison with related work is found in section VI. A synthesis of the paper and hints for future work are presented in the section VII.

II. Topic Maps

Everyday thousands of new information resources are linked to the web. This way the web is growing very fast, making search tasks more and 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 the authors refer to the Semantic Web they are thinking about a network of concepts. Each concept has a group of related resources and can be related to other concepts, though it 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 [2].

Topic Maps is a formalism to represent knowledge about the structure of a set of information resources and to organize it in *topics*.

A topic, 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 characteristics. It constitutes the basis for the topic maps creation.

A topic can have one or more occurrences. An occurrence represents the information that is specified as relevant to a given topic. One or more addressable information resources of a given topic, constitutes the occurrence.

Occurrences and topics exist on two different layers (domains), but they are *connected*. Occurrences establish the routes from the topics to the information resources, providing the reason why that route exists from a topic to an information object.

These topics also have associations that represent and define relationships between them. As described in ISO/IEC 13250 [2], a topic association specifies a relationship among specific topics (e.g. that Professor *is supervisor of* Student or Student *studies at* University). A topic association is a link between topics, each of which plays a role as a member of that association. This is important because a relationship that holds between topics is probably interesting even without the given context that the topic and its associations were created for (e.g. City *is located in* Country, that a city is located in a country relation is valid in most contexts).

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 there is about a certain domain, by formally declaring topics, and by linking the relevant parts of the information set to the appropriate topics [7].

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 [3] 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 are kept in their 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, enabling an incremental growth.

XML Topic Maps (XTM) 1.0 specification [9] was developed by *topicmaps.org* which is an independent consortium of parties developing the applicability of the topic map paradigm (ISO/IEC 13250:2000) to the World Wide Web by leveraging the XML family of specifications.

III. Oveia

The ontology extractor – Oveia – is based on ISO/IEC 13250 Topic Maps [2], like the TM-Builder [5]. The Oveia extracts information fragments from heterogeneous information resources according to an ontology specified in *XS4TM* (*XML Specification for Topic Maps*) language. Such as TM-Builder, Oveia has a language to specify the extraction. This new language, called XS4TM, was designed to cover all the concepts in the Topic Maps paradigm. This topic map extractor is an upgrade of the TM-Builder introduced in the previous section.

This extractor interacts with a information system that holds the set of information about a particular world, like an university, an academic department, a workshop, or a library.

In this new generation of the TM-Builder, the system core is a processor that extracts the topics and associations from information resources and builds a topic map. As illustrated in figure 1, the processor takes as input two specifications defined by the user: the description in XS4TM and the XSDS (*XML Specification for Datasources*).

The extractor stores the generated topic map in an XTM file or in a relational database (*OntologyDB*) that contains just the pieces of information in agreement with the *XS4TM* specification. The *OntologyDB*'s structure is presented in section III.A.7).

Initially, the Oveia was developed to extract topic maps from relational databases. But the present version is based on a more general model that allows the extraction from several data sources, supported by the concept of extraction drivers. An extraction driver should be defined for each different data source (database, XML document, CSV file, HTML page, etc). To make this possible it was created an intermediary representation for data sources that was named *datasets*. An extraction driver is a bridge from the data sources to this intermediate representation.

A new XML language, named XSDS, was added to the architecture to specify the extraction process. XSDS specification defines transformations and filters over the information resources. This can be done because it uses the query language appropriate for each resource to select the desired entities.

A. Architecture

The extraction process in the Oveia architecture starts with the physical information resources, and an XSDS specification that defines which data will be retrieved by the *Dataset Extractor*; this extractor generates the datasets in agreement with XSDS specification. The XS4TM processor takes the generated datasets and an ontology specification in XS4TM language and generates a topic map.

The Oveia architecture is shown in figure 1 and it is composed mainly by seven components: the *dataset extractor* receives an *XSDS specification* describing the metadata in the *information resources* that will be useful in the ontology construction; so, this extractor generates the intermediate representation (called *datasets*). These datasets contain the metadata (in a homogeneous way) extracted from information resources. The *XS4TM processor* takes these datasets and an *XS4TM specification*; this specification defines which metadata are topics, which are their occurrences, and what describes the association between them. Finally, the XS4TM processor generates a topic map that can be stored in an *OntologyDB* (section III.A.7)) or in an XTM file.

The following subsections describe this architecture in detail.

1) Information Resources: This component is composed by data resources: databases, XML documents, web pages, etc, that are kept unchanged, this is, Oveia does not modify these sources, it just copies the relevant information components to build the topic map.

These data sources are mapped to an intermediate (universal) representation, called *dataset*. This mapping is described in a XSDS specification. There is a specific driver to extract data for each kind of information resource. This driver (section III.A.4)) will interprete the XSDS specification and will retrieve the desired fields/elements building the intended representation.

2) XSDS Specification: A XSDS specification is an XML document that gives precise information about each information resource that should be scanned to extract topics and associations. In a formal way, the Context Free Grammar (CFG) shown below defines the abstract language XSDS:

```
resources    = datasources datasets
datasources  = datasource+
datasource   = @extractorDriver @name parameter+
parameter    = @name
datasets     = dataset+
dataset      = @name @database

extractorDriver = br.uneb.dcet.tmbuilder.drivers.DataBase
                | br.uneb.dcet.tmbuilder.drivers.XMLFile
```

A XSDS specification is separated in two parts: *datasources* and *datasets*.

The first one defines the path to the physical resources. Each resource is defined in a *datasource* element. This element has an attribute called *extractorDriver* that indicates which extractor driver will be used by *Dataset Extractor*. If the extraction is from a database, it is necessary to define several parameters, like database's user and password, the URL connection and the JDBC driver. If the resource is an XML file, the only needed parameter is the path to this document in the file system tree. So, each driver has its own attributes.

The second part of this specification is composed by the *datasets*. A *dataset* declares which data (record fields, or DTD elements) are to be extracted from each *datasource*. It is possible to declare several *datasets* for one single *datasource*. The extraction data is expressed in the query language adequate to the type of source in use: SQL will be used to extract information from a relational database while XPath will be used for the extraction of XML elements and attributes.

3) Datasets: Intermediate Representation: The *datasets* compose the intermediate representation that contains the extracted data from the information resources. Each *dataset* has a relation to an entity in these resources and it is represented through a table, where each line is a record following the structure specified

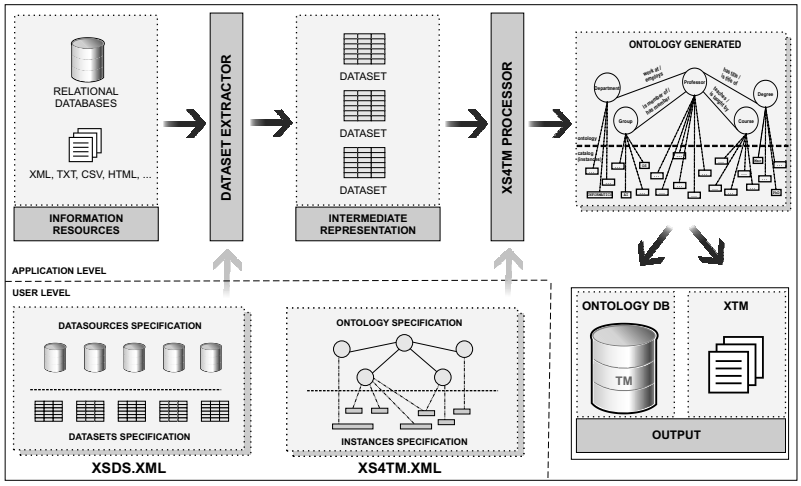

Fig. 1: Oveia Architecture.

in XSDS. The *datasets* representation guarantees that Oveia sees an uniform data structure that represents all the participating data resources.

The *dataset* declaration is composed by a query to extract the data from information resources. Each *dataset* has an unique identifier. This identifier will be used throughout the architecture to reference a particular *dataset*.

The fundamental idea is that all objects have labels that describe their meaning. For instance, the following object represents a member's category: <1, PhD>, where the string "1" is a identifier of this category, and "PhD" is a human-readable label. The datasets are very simple, while providing the expressive power and flexibility needed for integrating information from disparate sources.

4) Dataset Extractor: The *Dataset Extractor* is a processor that extracts data from information resources and creates the *datasets*, in agreement with an XSDS specification. This component processes an XSDS specification, which specifies both the information resources (*datasources*) and the respective piece of information that will be extracted, and it generates an intermediate representation (*datasets*).

This intermediate representation is composed by a set of tables that contains the information extracted from the *datasources*. These tables have only the data selected in *datasets* elements of the XSDS specification.

The *Dataset Extractor* has several extraction drivers that, as told above, are the module (program) responsible for the information extraction from the *datasources*. There is an extraction driver for each kind of *datasource*.

Two extractor drivers were developed: to connect with databases (br.uneb.dcet.tmbuilder.drivers.DataBase); and to deal with XML documents (br.uneb.dcet.tmbuilder.drivers.XMLFile). The implementation of new extraction drivers for other information resources will happen in a demand driven way.

5) XS4TM Specification: XML Specification for Topic Maps (XS4TM) is a domain specific language conceived to allow XML designers to specify the process of ontology extraction from information resources; in our case, from an intermediate representation called *dataset*.

XSTM and XS4TM have the same aim (to specify topic maps extraction), but they have some differences. The main difference between them is the grammar: XSTM has a particular grammar defined by the authors, while XS4TM has a grammar based in XTM DTD.

A specification in XS4TM is composed by two parts:

Ontology: the definition of the ontology requires in XS4TM the same effort as in XTM; it is necessary to specify every single topic type, association type, occurrence role type, ...;

Instances: the instances definition describes each topic and association that will be extracted from the information resource.

The XS4TM's Context Free Grammar is based in XTM 1.0 DTD[1]. The *ontology* and *instances* elements have the same syntax that the *topicMap* element in XTM model.

The XS4TM language is intended to make the specification of Topic Maps extraction more flexible. However, the use of XS4TM is not much more difficult because this language is an extension of the XTM standard; this is, the DTD that defines XS4TM includes and augments the XTM DTD. In XS4TM, the ontology is specified like in XTM: with the same elements and attributes. So, if the

[1]http://www.topicmaps.org/xtm/1.0/#dtd

designer knows XTM syntax, he does not need to learn another syntax to specify ontology in XS4TM.

The XTM and XS4TM syntaxes have a singular difference. The *topic* and *association* elements, children of *instances* element, have the *dataset* attribute instead of *id* attribute. This new attribute references the *dataset* (specified in XSDS) that contains the information necessary to construct the topics and associations. This difference should be highlighted because in XTM each single topic is specified with a topic element; in XS4TM the authors are referencing a *dataset* inside each topic element, this will have the effect of creating a topic for each data item in the *dataset*.

6) XS4TM processor: This component uses the XS4TM specification and retrieves the information it needs to build the ontology from the *datasets*. It is an interpreter that takes advantage of the information organization in *datasets* (an internal universal representation for data elements) and generates all the associations between the relevant topics according to XS4TM.

The XS4TM processor is one of the most important components of Oveia. Its behavior can be described in three steps: reads the the XS4TM specification and extracts from the *datasets* the topics and associations found; creates the topic map; finally, stores this generated topic map into an *OntologyDB* or an XTM file.

7) Oveia Output: Ontology Database or XTM file: As previously told, the user can choose the output format that best fits his application: a single XTM file or a relational database.

XTM files can grow exponentially. Huge XTM files are space and time consuming making their processing a hard task, specially from the web server side; and the performance tends to be worse as the interaction activity grows. So, in real cases it is crucial to find other ways to store very big ontologies. Therefore, it was decided to use also database technology.

The Topic Maps model is not hierarchic and it maps quite well into the relational model. This way it was decided to create a relational model for Topic Maps, named *OntologyDB*, following the structure mapping adopted in [10]. This model is easy to understand and implement, systematically and cleanly. The generated topic map is then stored in a relational database according to this model.

OntologyDB represents the target topic map, containing the topics and association inferred from the information resources, automatically generated by Oveia; appropriate SQL-based tools are available to navigate over it.

In a pratical way, there is a processor that stores an XTM document into an *OntologyDB*. This processor also allow the conversion in the opposite direction: extract XTM documents from an *OntologyDB*.

IV. Ulisses - the navigational component

At this point this paper says that ontologies specified with Topic Maps 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 among 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 the navigator is positioned at a certain concept it offers the user a particular view of the topic map from that concept, the resources it points to, and all the concepts related to it; if the user chooses one of the related concepts the position changes to that concept and the view will change accordingly; if the user chooses one of the resources the system will show that resource view.

In order to use web browsers as Topic Map navigators all the views discussed above are HTML pages automatically generated from the Topic Map source.

A new version of *Ulisses* is under development, actually adapted to each one of the possible output formats. Like Omnigator[2], it enables the navigation over an information system using concepts and relations; however, different drivers are provided to deal with XTM documents or an *OntologyDB*.

In this perspective *Ulisses* can be seen as a website generator. It was conceived to be a simple way of creating full sites, with design, content and topical links.

Ulisses was created as a direct consequence of creating a similar framework in XSLT for the XTM paradigm. 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 domain, 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.

Ulisses offers a framework for making rapid changes to a whole site, be it page setup, content, topics, relations or added/deleted pages. Since *Ulisses* 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

[2]http://www.ontopia.net/omnigator/models/index.jsp

tool to go into a production environment. This radically decreases development cost and time.

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 [1].

V. Case study

To illustrate all the ideas so far introduced and to describe the TM building process, this section presents a case-study where the main subject is an academic department. The department is composed by research groups, courses, projects, members, categories, courses, etc.

This case study starts with a MySQL database which contains all the information about the department. So, it is necessary to declare this information resource in an XSDS specification. In XSDS language, the *datasource* element declares the parameters needed to make the connection to the database, and the *extractorDriver* attribute points to the extractor driver that shall be used in this case. The XSDS specification below defines the department's database as a datasource called *DB_DI* and the chosen extraction driver to connect with this database.

```
<resources>
 <datasources>
  <datasource name="DB_DI"
    extratorDriver="br.uneb.dcet.tmbuilder.drivers.DataBase">
    <parameter name="connectionURL">
       jdbc:mysql://localhost/department
    </parameter>
    <parameter name="user">root</parameter>
    <parameter name="password"/>
    <parameter name="jdbcDriver">
      org.gjt.mm.mysql.Driver
    </parameter>
  </datasource>
  <datasource> ... </datasource>
  ...
 </datasources>
 <datasets>
  <dataset name="DI-Members" database="DB_DI">
     SELECT CodMember,Name,URL,Category FROM Members_table
  </dataset>
  <dataset name="DI-Categories" database="DB_DI">
     SELECT CodCat, Description FROM Categories_table
  </dataset>
  <dataset> ... </dataset>
  ...
 </datasets>
</resources>
```

The dataset *DI-Members*, also included in the example above, declares the extraction of the following fields from the database table *Members_table*: *CodMember*, *Name*, *URL*, and *Category*. These fields will be mapped to topics in XS4TM specification.

XS4TM language is divided in two parts: ontology and instances. The first one is shown in the code below.

```
<xs4tm xmlns="http://www.topicmaps.org/xtm/1.0/"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xmlns:xtm="http://www.topicmaps.org/xtm/1.0/">
 <ontology>
  <topic id="DI">
   <baseName>
    <baseNameString>Department of Informatics
    </baseNameString>
   </baseName>
  </topic>
  <topic id="Members">
   <instanceOf>
    <topicRef xlink:href="#DI"/>
   </instanceOf>
   <baseName>
    <baseNameString>Members</baseNameString>
   </baseName>
  </topic>
  ...
 </ontology>
 <instances>...</instances>
</xs4tm>
```

The Topic Maps abstract concepts (topic types, association types, occurrence role, ...) are defined in the *ontology* element. The *instances* element specifies the topics and associations that are built with data found in the *datasets*.

In the specification below, there is a topic definition. For each tuple found in *DI-Members* dataset a new topic in the resulting TM will be built. All these built topics are instances of *Members* topic. The base name is filled with the *Name* field from the *DI-Members* dataset. As well the *resourceRef* element (that is instance of Website) is filled it in with the *URL* field.

```
<instances>
 <topic dataset="DI-Members">
  <instanceOf>
   <topicRef xlink:href="#Members"/>
  </instanceOf>
  <baseName>
   <baseNameString>@DI-Members.Name</baseNameString>
  </baseName>
  <occurrence>
   <instanceOf>
    <topicRef xlink:href="#Website"/>
   </instanceOf>
   <resourceRef xlink:href="@DI-Members.URL"/>
  </occurrence>
 </topic>
 <association dataset="DI-Members">
  <instanceOf>
   <topicRef xlink:href="#Members-Category"/>
  </instanceOf>
  <member>
   <roleSpec>
    <topicRef xlink:href="#category-of"/>
   </roleSpec>
   <topicRef xlink:href="@DI-Categories.CodCat"/>
  </member>
  <member>
   <roleSpec>
    <topicRef xlink:href="#member-of"/>
   </roleSpec>
   <topicRef xlink:href="@DI-Members.Category"/>
  </member>
 </association>
 ...
</instances>
```

The *association* element defines an association between members and categories. The XS4TM processor compares the *Category* fields from *DI-Members* dataset with *CodCat* fields from *DI-Categories*; for each combination a new

association will be created. All the built associations will be instances of *Members-Category* topic; this association type should be declared in *ontology* element, like shown below.

```
<ontology>
 ...
 <topic id="Members-Category">
  <baseName>
   <baseNameString>Members and Category</baseNameString>
  </baseName>
  <baseName>
   <scope>
    <topicRef xlink:href="#member-of"/>
   </scope>
   <baseNameString>is member of</baseNameString>
  </baseName>
  <baseName>
   <scope>
    <topicRef xlink:href="#category-of"/>
   </scope>
   <baseNameString>category of</baseNameString>
  </baseName>
 </topic>
 ...
</ontology>
```

The department's database has 44 tables representing members, documents, categories, courses, etc. The XSDS file describes this database and its datasets have 115 lines. The XS4TM document for this case has 1562 lines; in this document there are 109 topics in the ontology specification; in the instances specification there are 53 topics and 46 associations definitions.

The final result was a topic map (stored in *OntologyDB*) with 440 topics and 422 associations; the same topic map could have 17240 lines in the XTM format. This topic map construction task was concluded in 1:53 minutes. The average time for each element (topic or association) was 0,1311 seconds.

The environment which was implemented in this experiment is composed by an Intel PC Pentium II, 600MHz and 512 MB Ram. All services such as MySQL database (source), *OntologyDB* managed by Microsoft SQL Server 2000, and our prototype was carried out at the same machine.

VI. Related Work

Numerous other projects have the similar goals than *Oveia* has. TSIMMIS[3] is a project that aims to provide tools for accessing, in an integrated fashion, multiple information sources, and to ensure that the information obtained is consistent.

TSIMMIS was developed to extract properties from unstructured objects allowing the combination between several sources and the browsing of information; it gives a centralized view of the information that is distributed in the information system. Oveia was developed to allow a conceptual navigation over the heterogeneous information systems. This conceptual navigation is driven by an ontology specified from metadata extracted from information systems.

KAON[4] is an open-source ontology management infrastructure targeted for business applications. The KAON REVERSE tool is one of the component of KAON environment. An important focus of KAON is on integrating traditional technologies for ontology management with those used in business applications, such as relational databases. This tool has the same aim as Oveia. The table I shows the comparison between the Oveia and the KAON REVERSE.

Table I
KAON REVERSE x Oveia

	KAON Reverse	**Oveia**
Language	Java	Java
APIs	Yes	Yes
Reverse Engineering	Yes	No
Specification	Tree (GUI)	XML Document
Ontology Extraction Source	Relational DB via JDBC	Relational DB via JDBC, XML, extensible to other sources
Ontology Representation	RDF	Topic Maps
GUI	Yes	No
Final Result	RDF Document	OntologyDB and XTM file

According to table I, KAON REVERSE has advantages concerning the use of a graphical interface for the specification of the ontology. It also allows the use of reverse engineering of data sources to help creating the mapping. On the other side, Oveia is more flexible concerning data source formats and the specification process. To represent the ontology, KAON REVERSE adopts RDF (XML file); Oveia generates ontologies and stores them in an ontology database (*OntologyDB*) or in an XTM file.

VII. Conclusion

This paper describes the integration of information systems using the ontology paradigm. The proposal is an architecture for the automatic construction of Topic Maps with data extracted from information systems. This environment, called Oveia, is an upgrade of TM-Builder [5], an ontology extractor handling just XML.

The Oveia contributions to the area can be summarized as follows:

1. XSDS allows the integration of heterogeneous data sources; this component provides an abstraction of

[3]http://www-db.stanford.edu/tsimmis/tsimmis.html

[4]http://kaon.semanticweb.org/

information resources based on the concept of *extraction drivers* making Oveia independent from source formats because only the driver knows the source specific structure;

2. Intermediate Representation as Datasets is a proposal for a normalized way of representing different and heterogeneous data structures;

3. XS4TM supports ontology extraction process; This is one of the most important contributions to the area, since it optimizes specification and reduces time to setup topic maps based applications;

4. *Ulisses* supports conceptual navigation over XTM files and OntologyDB databases; all the features in this navigator are being formally specified; from this specification it will be possible to derive a generic navigator generator.

Currently Oveia is being used in several scenarios:

E-Learning In this context, Oveia is being use to integrate the output of some content provider applications [6];

General websites Personal webpages, departmental websites, ...

Systems Integration To generate an homogeneous view of heterogeneous information systems.

A. Future Work

Merging is the process of joining two topic maps or joining two topics. It is a built-in feature of the Topic Maps standard because the *merging of indices* was one of the requirements underlying the development of the Topic Map paradigm. The process of merging two topic maps is not implemented in this version of Oveia. The next step will be its implementation.

In addition, there is a plan to build a constraint language that will allow constraint specifications over the information systems and the ontology represented in a topic map.

Talking about *Ulisses*, another family of navigators is under development: graphical SVG[5] and X3D[6] navigators.

The Oveia does not have a friendly interface to create XS4TM specifications. The development of a XS4TM or a XSDS specification still is a hard task. To facilitate this task, a friendly user-interface is also under development.

A module that extracts relationships from the databases metadata will be built. This module will support the reverse engineering of a database (like in KAON). So, it will be possible to generate the XSDS specification automatically.

[5]Scalable Vector Graphics – http://www.w3.org/TR/SVG/

[6]Web 3D Consortium – http://www.web3d.org/

VIII. References

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

[2] M. Biezunsky and M. Bryan and S. Newcomb. ISO/IEC 13250 – Topic Maps. In *ISO/IEC JTC 1/SC34*, http://www.y12.doe.gov/sgml/sc34/document/0129.pdf, December, 1999.

[3] C. Goldfarb and P. Prescod. *XML Handbook*. Prentice Hall, 2001.

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

[5] G. Librelotto and J. Ramalho and P. Henriques. TM-Builder: Um Construtor de Ontologias baseado em Topic Maps. In *Procs. XXIX Conferencia Latinoamericana de Informática*, La Paz, Bolívia, 2003.

[6] G. Librelotto and J. Ramalho and P. Henriques. ADRIAN – a platform for e-learning content production. In *Procs. Second International Conference on Multimedia and Information & Communication Technologies in Education*, Badajoz, Spain, 2003.

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

[8] S. Pepper. Ten Thesis on Topic Maps and RDF. In *Ontopia*, http://www.ontopia.net/topicmaps/materials/rdf.html, August, 2002.

[9] S. Pepper and G. Moore. XML Topic Maps (XTM) 1.0. In *TopicMaps.Org Specification*, http://www.topicmaps.org/xtm/1.0/, August, 2001.

[10] K. Williams and M. Brundage and P. Dengler and J. Gabriel and A. Hoskinson and M. Kay and T. Maxwell and M. Ochoa and J. PaPa and M. Vanmane. *Professional XML Databases*. Wrox Programmer to Programmer Series, 2000.