Using OWL to specify and build different views over the Emigration Museum resources

Flávio Xavier Ferreira and Pedro Rangel Henriques

Departamento de Informática, CCTC
Universidade do Minho
{flavioxavier,prh}@di.uminho.pt

Abstract. This paper discusses the approach used to create the exhibition rooms of the virtual (Web-based) section of Portuguese Emigration Museum (Museu da Emigração e das Comunidades, MEC) founded by the Cultural Department (Casa da Cultura) of Fafe's Town Hall (Câmara Municipal de Fafe). The museum's assets are made up of documents (paper or digital format) of more than 8 kinds, ranging from passport records to photos/cards or building-drawings. Each room is no more than a view over the information contained in those single or interrelated resources. The information exhibited in each room is described by an ontology, written in OWL. That ontology is used to specify the information that is extracted from the resources and to provide a semantic-network navigator to the museum visitor. That approach can be automatised to allow a very systematic way to deal with the huge and rich museum assets; we will also discuss some technical details concerning its complete implementation in the near future.

1 Introduction

Fafe, as many other Portuguese towns and villages, mainly at the north, has a huge cultural heritage characterising the social phenomena of emigration (especially to Brazil) along the nineteenth and first half of twentieth centuries.

In this context Miguel Monteiro[1], supported by the staff of Fafe's Town Hall (via Cultural Department), started some years ago collecting information from passports' governmental records into a database. But soon from this project arise the the idea to gather all sorts of documentation and create a web-based virtual museum that makes easily accessible this rich cultural heritage to emigrants and theirs descendants as well as to all those researching in that area, and of course the general public.

The Museum was born in 2001 with the designation of Museu da Emigração e das Comunidades (hereafter referred as MEC). Its material is inherited mainly from official documents or personal writings reporting on the departure, travel, and stay abroad, but there are also a large number of assets bearing witness

[1] An History professor, responsible for the MEC creation, and its current director.

to the less usual phenomena of *emigrants' return* – besides the documents, a large set of buildings (private or public, professional or philanthropic, and other non-physical evidences) left by the emigrants around the country can be also considered assets. The MEC is structured upon six Rooms (see `http:///www.museu-emigrantes.org/museu.htm`), but at the moment these rooms are handmade, difficult to maintain and to add new information, and most important, they are lacking a systematic way to for information acquisition, treatment and exhibition; inconsistencies are evident from room to room but even inside the same room.

Some years ago the MEC, started a collaboration with University of Minho Language Specification and Processing Group (GEPL), to develop a project aiming at systematic approach for the acquisition, archiving, treatment and exploration of the Museum's documental resources. In our perspective, each room is seen just as *a specific view over a common information repository*. The repository should be a digital archive (in database format or as a collection of XML files) of all the information resources referred above as museum's assets. Each *view* (the knowledge enclosed in the respective room) can be specified by an *ontology*, as traditionally done by philosophers to organise the discourse over a certain closed-world. The extraction process[2] can be automatise by resorting to a standard notation for the ontology description, and moreover, the browsers that will implement the user-interface in each room (as a semantic network navigator) can also be automatically built.

In the past, that approach was realised using Topic Maps[13], at the moment we are experimenting with OWL. This paper describes our more recent research work exploring RDF/OWL.

So, we start, in section 2, with a catalogue of the MEC information resources, which constitute the Museum's assets; as the basis for all the exhibition rooms, it gives the motivation to our work and proposal. In section 3 we introduce the ontology concept and present the main standards for ontology description; Topic Maps are just briefly referred as they where explored in previous work; RDF/OWL is studied in detail because it is going to be the main focus of this work. Section 4, not the biggest but the central one, presents a detailed discussion of our methodological approach, briefly introduced above as *the use of ontologies to specify and construct each museum's room*. Section 5 is concerned with MusVis, the extraction and navigation system we are developing; its architecture is defined and its technical implementation is briefly referred. The paper ends at section 6 with the traditional remarks and future work.

2 Emigration Museum and its Information Resources

The MEC is a web-museum (although it also has physical headquarters and some exhibitions), that gathers knowledge, and resources about the Portuguese emigration.

[2] The task of building up the ontology from the information resources data

The MEC wants to discover and show the effects of mixing people and cultures, in the social, cultural and economical history of Portugal. It focus, mainly on the past Portuguese emigration to Africa and the more recent emigration to Brazil (19th and 1st half of 20th century) and to Europe (2nd half of the 20th century), but it is by no means restricted to them [2].

The MEC assets are vast and multifaceted, this is supported by the fact that emigration documents and objects come from the most diversified sources, ranging from official government records to old newspapers and photo albums. The document types are themselves heterogeneous (from official travel reports to local stories). Some documents where converted to an electronic format (plain ASCII text, Ms-Word, Ms-Excel, Ms-Access, HTML, etc.), but many others are, still, in paper format stored in Archives and Libraries.

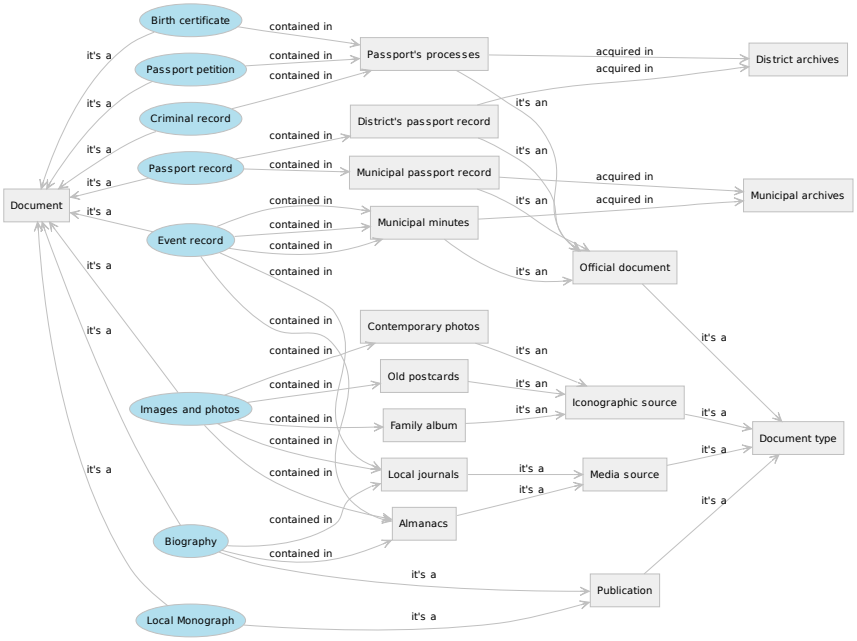

Fig. 1. Information sources and resources semantic web

This tremendous amount of resources and their variety gives the MEC an enormous potentiality as a museum, but at the same time it is very difficult to organise and display all this information in a straightforward way. To overcame this problem the sources of information (the so-called museum's assets) were catalogued. The *conceptual map* (a *graph of concepts*) in Figure 1 shows the organisation of the sources and the types of the documents (*ellipses* denote doc-

ument types). Based on that classification, we have defined (using XSD) an XML format to enable the encoding of those documents to a structured (annotated) digital format, adequate for archiving and subsequent processing (this will be addressed in section 5). We are also developing an editor to assist the acquisition phase and the creation of the XML files.

3 Ontologies and their Notation

An ontology is originally a philosophic concept, concerned with the study of being or existence and forms the basic subject matter of metaphysics. In computer science an ontology represents a set of concepts and there relations in a given domain; it can also be used to infer knowledge and information about the domain's objects[9].

Technically speaking an ontology is defined by a set of classes, individuals, attributes, and relations. *Classes* or *concepts*, are abstract sets of objects, that can contain other individuals and other classes (subclasses). *Individuals* or *instances* are the actual objects we want to represent, they can be people, animals, numbers, web pages, etc.. The ontology objects can be described using *attributes*, each attribute is a name/value pair. *Relations* are connections between the ontology objects, they allow the representation of concepts and the creation of associations within the ontology objects. In the example seen in Figure 2, we

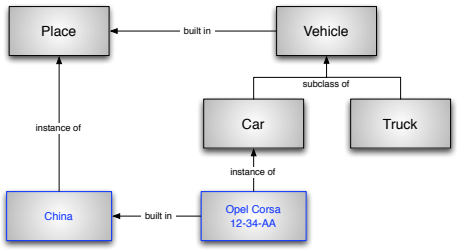

Fig. 2. An ontology example

can identify the classes `Vehicle`, `Car`, `Truck` and `Place`; and the individuals `China` and `Opel Corsa`; the licence plate `12-31-AA` can be seen as an attribute; and there are also the relations `built-in`, `subclass-of` and `instance-of`.

To use ontologies in the MEC we still need a specific notation to write them. There are several ontology description languages available[6], but our attention (in the next subsections) goes to: the International Organization for Standardization (ISO) standard Topic Maps (TM)[11]; and the World Wide Web Consortium (W3C) standards Resource Description Framework (RDF), RDF Schema (RDFS) and Web Ontology Language (OWL)[3].

[3] All three are part of the W3C semantic web effort[5].

3.1 Topic Maps

Topic Maps can be seen as a way to share and represent knowledge, with focus on information retrieval; this notation was created to allow knowledge description that are equally processable by machines and humans[12]. TM is a standard (ISO 13250:2003) for the representation and interchange of knowledge, with an emphasis on the findability of information, it provides a standardised notation for interchangeably representing information about the structure of information resources used to define topics, and the relations between topics[11]. In a topic map, information is represented using topics, associations and occurrences:

topic is the basic element of a topic map, representing some subject (ex. persons, contries, organisations, software modules, etc.);
association connects two or more topics, definning a semantic relationship between the themes represented by those topics;
occurrence represents a relationship between topics and information resources relevant to them.

There are several notations for TM. The most usual is the standard XML-based interchange syntax called XML Topic Maps (XTM).

TM lacks a schema language, that defines the topics structure and constraints. One of the possible solutions is the ISO standard Topic Maps Constraint Language (TMCL)[1]; this proposition is still underdevelopment. XTche language, developed by Giovani Librelotto[12], is a concrete proposed intended to comply with the TMCL requirements.

Topic Maps proved in practise to be natural and easy to use, allowing the effective construction and the handling of semantic networks. However, the scientific comunity is nowadays more inclined to use the W3C standards mainly on account of RDF.

3.2 RDF and OWL

OWL was chosen to represent the ontologies for the MEC, but OWL is not a standalone technology, it benefits and uses many RDF and RDFS constructs, and is considered an extention of these languages. As such, we will take a deeper look into those three W3C recommendations.

RDF language was design as metadata model, but is largely used as a general method for modeling information. RDF metadata model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples in the RDF terminology.

A RDF resource is any data or information source we want to describe. It can be anything from physical objects to web resources (ex. a city, a database, a web page, etc.). A resource is allways represented using a Universal Resource Identifier (URI) (the URI does not need to be on a web accessible path, nor has any normalisation rules; it will just suffice that its meaning is known by the reading application).

The subject, is the resource we are describing. The predicate or property denotes a characteristic and expresses a relation between the subject and the object, it is represented by a RDF resource. The value or object is represented either by a literal string or a resource[12]. As an example, the sentence "Ana lives in Portugal" in RDF, is the triple "Ana" (subject), "lives in" (predicate) and "Portugal" (object). This example is shown in Figure 3 using RDF/XML[4].

```
1  <rdf:RDF
2          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3          xmlns:pro="http://people.org/predicates#">
4
5          <rdf:Description rdf:about="http://people.org/Ana">
6                  <pro:lives_in>
7              <rdf:Description rdf:about="http://countries.org/Portugal"/>
8          </pro:lives_in>
9          </rdf:Description>
10 </rdf:RDF>
```

Fig. 3. RDF/XML example

RDFS is a RDF extension, that allows the definition of classes of resources, restrictions and properties over RDF, in a way that establishes the application vocabulary. Figure 4 illustrates this new abstraction layer over RDF.

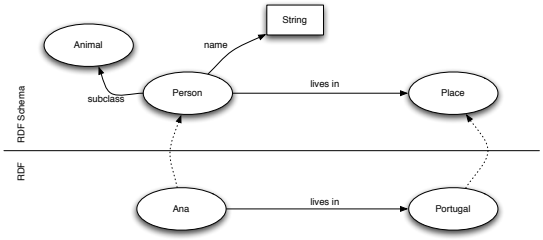

Fig. 4. A RDF statement and its corresponding RDF Schema

RDFS adds some constructs to the RDF language like, `rdfs:Class`, `rdfs:subClassOf`, `rdf:Property`, they allow the creation of a class hierarchy. Figure 5 shows the RDFS description[4] for the sample sentence used above (Figure 3).

RDFS has some limitations as a standalone ontology language because: there is no distinction between the language constructs and the ontology vocabulary; it does not allow to define class and property restrictions; it is too week to describe resources in detail[3,5,12].

[4] This example is in the RDFS abbreviated format, the extended format its much similar to the notation of RDF, both formats have however the same meaning

```
1  <rdf:RDF
2      xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4      xml:base=  "http://people.org/rdf#">
5
6      <rdfs:Class rdf:ID="Animal" />
7      <rdfs:Class rdf:ID="Person">
8          <rdfs:subClassOf rdf:resource="#Animal"/>
9      </rdfs:Class>
10     <rdfs:Class rdf:ID="Place" />
11
12     <rdf:Property rdf:ID="lives_in">
13         <rdfs:range rdf:resource="#Place"/>
14         <rdfs:domain rdf:resource="#Person"/>
15     </rdf:Property>
16     <rdf:Property ID="name">
17         <rdfs:range rdf:resource="rdfs:Literal"/>
18         <rdfs:domain rdf:resource="#Person"/>
19     </rdf:Property>
20
21         <Person rdf:ID="Ana">
22         <name>Ana</name>
23         <lives_in>
24             <Place rdf:ID="Portugal"/>
25         </lives_in>
26         </Person>
27 </rdf:RDF>
```

Fig. 5. An RDFS example coded in RDF/XML

The OWL was built on top of RDF and RDFS as a language for the represention of web ontologies[3]. This language was designed to be used by applications that process the information content instead of just presenting it to humans[10,12]. An OWL ontology includes class descriptions, along with there associated properties and instances, as well as related restrictions.

An OWL file, as seen in Figure 6, is opened by a namespaces declarations, followed by the `owl:Ontology` element; this element contains the ontology URI (line 10), generic information's about the ontology (`rdfs:comment` - line 11), version control (`owl:priorVersion`) and imported ontologies (`owl:imports` - line 12).

OWL uses the constructs `owl:Class` (lines 15 and 16) and `rdfs:subClassOf` (line 17) to represent classes and subclasses . Ontology relations are defined in OWL by the `owl:ObjectProperty` element (lines 20 and 23). Ontology attributes are defined by `owl:DatatypeProperty` (line 28); this element relates a OWL class to an XML Schema (XSD) datatype. In OWL, the individuals are created using the classes identifiers (lines 35 and 37).

The code presented in Figure 6 is a straitforward example, but there are some things to notice:

- the usage of the elements `rdfs:domain` (lines 22 and 30) and `rdfs:range` (lines 21 and 29) within `owl:ObjectProperty` to define the domain and range of a property;

```
1  <rdf:RDF
2      xmlns:="http://people.org/owl#"
3      xmlns:per="http://people.org/owl#"
4      xmlns:pla="http://places.org/places#"
5      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6      xmlns:owl="http://www.w3.org/2002/07/owl#"
7      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
8      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
9
10   <owl:Ontology rdf:about="http://people.org/owl">
11      <rdfs:comment>This document contains the class definition of people and
            animals, there properties and some instances.</rdfs:comment>
12      <owl:imports rdf:resource="http://places.org/places"/>
13   </owl:Ontology>
14
15   <owl:Class rdf:ID="Animal"/>
16   <owl:Class rdf:ID="Person">
17      <owl:subClassOf rdf:resource="#Animal"/>
18   </owl:Class>
19
20   <owl:ObjectProperty rdf:ID="lives_in">
21      <rdfs:range rdf:resource=pla:#Place"/>
22      <rdfs:domain rdf:resource="#Person"/>
23   </owl:ObjectProperty>
24   <owl:ObjectProperty rdf:ID="address_of">
25      <owl:inverseOf rdf:resource="#lives_in"/>
26   </owl:ObjectProperty>
27
28   <owl:DatatypeProperty rdf:ID="name">
29      <rdfs:range rdf:resource="xsd:#string"/>
30      <rdfs:domain>
31            <owl:Class rdf:about="#Person"/>
32      </rdfs:domain>
33   </owl:DatatypeProperty>
34
35   <Person rdf:ID="Ana">
36      <lives_in>
37         <pla:Place rdf:ID="Portugal"/>
38      </lives_in>
39      <name>Ana</name>
40   </Person>
41 </rdf:RDF>
```

Fig. 6. An OWL example, using RDF/XML syntax

- the element `owl:inverseOf` (line 25) allows the definition of inverse properties;
- the declaration of the individual `Portugal` (line 37) is inside another individual `Ana` (line 35), witch is completely valid and serves to show the OWL syntax freedom;
- the individual `Portugal` (line 37) does not belong to a local class (`Place`), in fact this individual belongs to the ontology `http://places.org/places`, this happens because in OWL it is valid to extend existing ontologies in other files.

OWL also offers various other elements such as `owl:Restriction`, `owl:cardinality`, `owl:intersectionOf` and `owl:disjointWith`, they allow restrictions, cardinality, class intersection and disjoint classes respectively. They, along with

others improve the task of ontology modeling. Regarding properties there are also the elements `owl:TransitiveProperty`, `owl:SymmetricProperty`, `owl:FunctionalProperty` and `owl:FunctionalInverseProperty` that are specialised types of `owl:ObjectProperty`.

It can also be noticed that in Figure 6 the `address_of` property is not present in the individual `Portugal`, but it could be inferred from its inverse (the `lives_in` property) by using a OWL reasoner. A reasoner is a tool that produces valid logical conclusions about an ontology.

OWL has three sub-languages, OWL Lite, OWL DL, OWL Full. We used OWL DL in our project. OWL lite is the most simple as it uses only a subset of OWL constructs, therefore is very simple to implement. OWL Full is the full language without any semantic restrictions, its very close to RDFS. OWL DL is an intermediate solution, that is very expressive, but also maintains a computable completeness and decidability; it includes all the language constructs but they can only be used under certain conditions[10].

4 Using Ontologies to Create Museum Rooms

The MEC needs a simple and organised way to show its assets to the public. For that purpose, we created theme oriented museum exhibition-rooms, or as we call them, *views*. Those views are described in a rigorous way by means of semantic networks, this is, concept maps. This is a new approach, that uses related information gathered from the various information sources rather than just showing each one of them (Figure 7). This approach allows the user to browse in an interactive and differentiated way through the information, and also allows to create more than one perspective over the same information. Views are

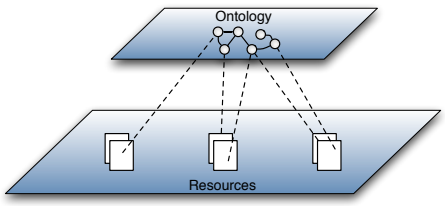

Fig. 7. The ontology and resources planes interaction

represented by ontologies. So a OWL ontology[5] is defined for each view created. Each view is intended to focus in a particular aspect or theme, for instance:

– Emigrants by date: view that shows, for an given time interval, all the known emigrants and associated data.

[5] For each view there is only one ontology, however many OWL files per view can exist (the definition file, and many files with individuals).

– Event Surroundings: taking as input an event in the life of a given emigrant (ex. departure), this view will show information about the physical and social surrounding environment at the epoch and place where the event occurred.
– Emigrant's Places (V1): view that reports on the different places of emigration cycle (birth, departure, arrival, etc.).

We will now take a closer look at this last view and its specification.

V1 shows the main events of the emigrant's life and their location, along with images of the events and places. That information is retrieved from passport petitions, passport records, birth certificates, criminal records, events records, and images, postal cards and photos, as can be seen in Figure 1.

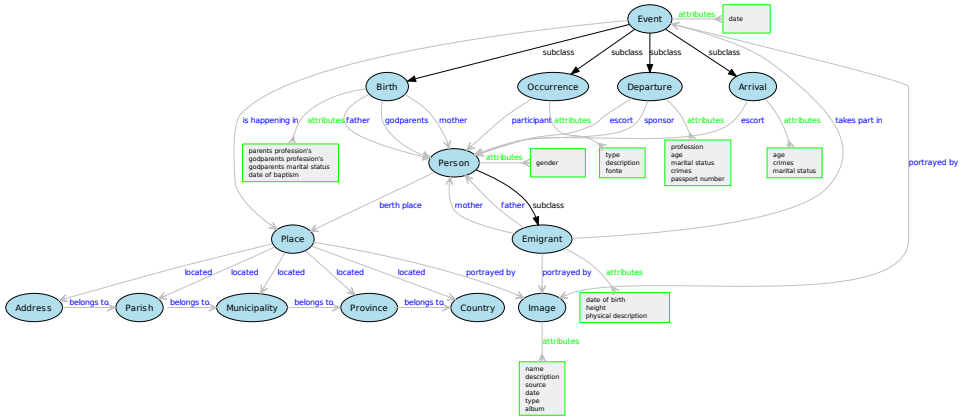

Fig. 8. V1 ontology

Figure 8 shows the ontology specification, i.e, its classes, its properties[6] and its attributes. In a global analysis of the ontology, it is clear that the `Emigrant` class is the centrepiece, followed by the `Event`, `Place` and `Image` classes. This four classes represent V1 main idea: *the emigrant, his life events, where those events happened, and the images of the emigrant, the events and places.*

5 MusVis - an ontology navigation system for the museum visitors

In this section, a short description of our ontology navigation system (MusVis) is presented. MusVis is more than a browser, will be a modular software system

[6] Inverse properties are not represented in the image

that allows one to gather MEC basic information from the various sources, build an ontology with that information, and create an web-based navigator over the ontology. MusVis architecture is presented in Figure 9; it is made up of four modules, each with a specific data transformation task. Now we will take a closer look into each module. As seen in section 2, the MEC assets are multifaceted so

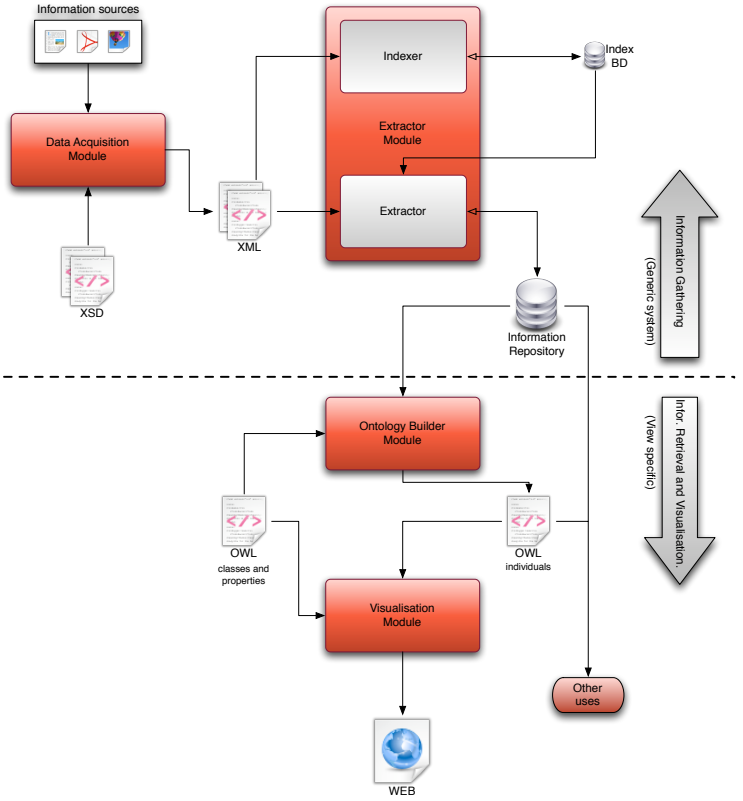

Fig. 9. The MusVis architecture

a common digital file format for the various documents presented in Figure 1 was needed—XML markup system was the principle adopted. As already told, an XML-Scheme was defined for each type of document, setting up the XML tags that can be used each time a new document has to be added to the Museum's digital repository. The Data Acquisition module assists in this task; it is similar to a graphical text editor, and allows an historian to translate and markup documents. The markup is done by means of colours and symbols, always hiding the XML backend from the historian. ThisThese module is based upon previous work developed inside our research group [8,7].

The Extractor module is responsible for indexing and organising in a common repository all the information obtained from the XML files created in the previous module. The data model chosen for the information repository allows a straightforward implementation of the next module, the Ontology Builder, as an information retrieval system.

The Ontology Builder module is responsible, as said above, for the creation of MEC ontologies that support each museum's room. This module is view specific, which means that needs a different instantiation (although with a similar behaviour) for each view. Taking into account the ontology definition (classes, properties, etc.) and the view specific input, it extracts all the information relevant to set up that view, and creates the ontology individuals with the information repository data. These last three modules update the ontologies with new data, every time a XML file is added to the system; this allows the new data visualisation by the last module, without any configuration, since it retrieves the information from the ontologies dynamically.

Fig. 10. A MusVis screenshot

Finally, the Visualisation module traverses the semantic network (corresponding to the ontology), and exhibits its content (each ontology component) to the user. It accomplishes this by using the ontology data and a set of standard and custom (one for each ontology) presentation rules to generate a set of dynamic web pages (JSP) for each ontology. Those pages are entity-oriented, and centred on each individual attributes and connections. Figure 10, a screenshot of V1, illustrates such a webpage; it is precisely the webpage generated by MusVis from the V1 ontology seen on figure 8.

The system modules are being implemented using Java. JAXP framework is used for XML processing (Saxon implementation for XPath); and the Jena OWL

framework with Pellet as a reasoner, for RDF/OWL processing. Sqlite is used for the database. Working environments are: Protégé for OWL editing; XMLSpy to create XSD and XML documents; and Eclipse for Java programming.

6 Conclusion

Along this paper the idea that exhibition rooms of the virtual Emigration Museum, are no more then structured views over the the museum digital archive of documents (its assets) was defended. Although it was assumed along the paper that the repository is made up from XML documents, if a subset of them is supported in databases or whatever digital format, the approach can be the same. OWL documents can be represented in it's XML standard notation or we can use a database[7].

That perspective allows to systematically extract the information from data sources and automatically build the OWL ontology that formally describes the meaning of each view (by other words, the content of each room). An OWL navigator, general purpose or a specific one, can then be used to implement the visitor interface.

Future work goes in two directions. On one hand, more implementation work should be done to finish the first and last modules, the Data Acquisition and the Visualiser. Additionally, some extra analysis must be made to improve the automatisation degree. After that, MusVis can be deployed and real tests carried on to measure its performance, and effective impact and usability. On the other hand, other views should be specified, in order to build the respective ontologies. At this moment (without a full implementation) a OWL vs TM, realistic comparison can't be achieved.

References

1. Document Description and Processing Languages. `http://www.isotopicmaps.org/tmcl/`.
2. Museu da Emigração - O que somos? `http://www.museu-emigrantes.org/ficha_tecnica.htm`.
3. OWL Web Ontology Language Guide. `http://www.w3.org/TR/owl-guide/`.
4. Resource Description Framework (RDF): Concepts and Abstract Syntax. `http://www.w3.org/TR/rdf-concepts/`.
5. W3C Semantic Web Activity. `http://www.w3.org/2001/sw/`.
6. O. Corcho and A. Gomez-Perez. A Roadmap to Ontology Specification Languages. *Knowledge Engineering and Knowledge Management: Methods, Models, and Tools: 12th International Conference, Ekaw 2000 Juan-Les-Pins, France, October 2-6, 2000 Proceedings*, 2000.
7. Flávio Ferreira, Hugo Pacheco, and José Vilas Boas. Pda's no levantamento de informação em arquivos históricos. Technical report, Universidade do Minho, 2007.

[7] With Jena we can access a OWL in a database with the same API as a OWL serialised as RDF/XML

8. Rafael Félix. Sistemas de Digitalização e Anotação de Documentos. Technical report, Departamento de Informática, Universidade do Minho, 2002. Relatório de Projecto (Opção III).
9. T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
10. I. Horrocks and P.F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, pages 17–29, 2003.
11. ISO/IEC. *ISO/IEC JTC1/SC34: Topic Maps Constraint Language* .
12. Giovani Rubert Librelotto. *[Topic maps: da sintaxe à semântica.* PhD thesis.
13. G.R. Librelotto, J.C. Ramalho, and P.R. Henriques. Topic maps aplicados ao sistema de informação do Museu da Emigração. 2006.