Lifting and Lowering the Data from Digital Library “Virtual Encyclopedia of Bulgarian Iconography”

Ivo Marinchev

Abstract: In this paper we present an algorithm for lifting and lowering the data from the database of the digital library “Virtual Encyclopedia of Bulgarian Iconography”. The data is transformed between its native relational or web service form and the OWL instances defined by the corresponding base ontology developed in the scope of the SINUS project [9]. We implement data transformations with the help of the table that maps the corresponding fields and structures of the original syntactic data and the corresponding OWL classes and properties.

Key words: Semantic Web, Ontologies, Instance Data, Lifting, Lowering, OWL, RDFS, RDF.

INTRODUCTION

Semantic Web [7] and Semantic Web Services are research efforts towards automation of the use of information in the web and web services, enhancing existing capabilities with intelligent and automated integration and reasoning. The most useful approach for applying semantic descriptions to existing web services is the bottom-up semantic annotation of Web services. It has very relaxed requirements on completeness of semantic descriptions, which enables building incremental layers of semantics on top of existing service descriptions.

No matter how semantic annotations are applied, during the service invocation the data have to be converted between its native and semantic representation and vice versa. The transformation between the native representation (for example XML instance document) and semantic representation (for example OWL [2] instances in RDF [6]) is called lifting. The opposite transformation is called lowering.

As in the general case there are many different mappings between the semantic schema in the form of ontologies (OWL) and its native data schema (XML schema or relational database schema) neither annotations nor lifting and lowering can be created completely automatic. For that reason and the fact that most interoperability data today are represented in XML format or can be easily converted to XML the research papers and recommendations propose the usage of XSLT transformations to service annotations for lifting and lowering implementation.

In this paper we propose a slightly different approach to implementation of lifting and lowering process with the help of a table (we call it mapping table) that represents the mappings between the structure of the native data and their corresponding semantic instances. The advantages of this approach are that mappings are easier to read and maintain as they are in practice a form of configuration of the transformation algorithm and at the same time two way transformations can be implemented with the same mapping table.

---

1 This work was partially supported by Research Project No. D-002-189 funded by the National Science Fund of Bulgaria.
DATABASE OF “VIRTUAL ENCYCLOPEDIA OF BULGARIAN ICONOGRAPHY”

The goal of the project “Virtual encyclopedia of the Bulgarian iconography”[5] is to develop the information content, structure, and the realisation of a digital library with multimedia content as a demonstrator of “Virtual encyclopedia of the Bulgarian iconography”. That library includes several hundred specimens of Bulgarian icons from different artists, historical periods, and schools. The chosen architecture represents a hypermedia digital library [4], which means that presentation of a complex multimedia content in the Internet is simplified. The resources are digital objects of different formats – text, graphics, and other media. They are structured in a hypermedia manner, i.e., some digital objects point to other ones. In this way the user can navigate quickly, in a non-linear fashion, within areas of related topics, using the hyperlinks. The digital objects are grouped according to their topics into thematic collections [4]. For each object and collection, special meta-descriptions are created. They include data about the title, the artist, the period (in years and centuries), the school, the dimensions (width/height/thickness), the technique, the base material (type of wood, ground coat, etc) the category, the location, the title description, the author description (biographic data), the comment (icon features such as state, founder’s and other signatures, previous restorations), etc. Also, they contain links to other digital objects and collections, keywords, and so on. That information is used for the semantic annotation and indexing of the digital objects, which facilitate their locating during search requests, and their web-based representation [3].

Virtual encyclopaedia of the Bulgarian iconography has several interfaces. One is the main web based graphical interface that was built during its initial development. Later given the needs of SINUS project [9] two programming interfaces were created – SQL and web service. SQL is the interface with direct access to its relational database (MySQL). The second interface is implemented as a web services that represents the data in a data structure that represents the iconographic object as a single complex object.

Figure 1 depicts simplified structure of the raw data in the encyclopaedia. The web service interface reconstructs this structure in the same way.

Figure 1: The simplified structure of the iconographic object in the database [5].
BASE ONTOLOGY

The base ontology is the ontology that is created to be as close as possible to the raw data. The main objectives are to encapsulate enough semantics about the knowledge domain but at the same time to facilitate the semantic annotation of the legacy data as much as possible. Hence the classes in the base ontology correspond closely to complex types in the web service schema. The differences are on the instance level. In the original database as well as the web service interface there are no separated objects for most of the complex types (for example there is no separated object Author, it exits just as part of the Icon object). But on the ontology level it is required instances to be enumerated in order to be possible for the client applications to infer the semantics of the data and distinguish between a string as a literal and a string as an instance of the ontology class.

Base ontology classes are (left indented are subclasses of the corresponding class):

- IconographicalObject:
  - Icon
  - WallPainting
  - Miniature
  - Vitrage
  - Mosaic
- PlasticIconographicalObject
  - Iconostasis
  - Throne
- Identification
- Description
- Technology
- Author
- IconographicalClan
- IconographicalSchool
- ObjectDate
  - ObjectPeriodDate
  - ObjectDating
- Century
  - TheBeginningOf
  - TheMiddleOf
  - TheEndOf
- TheFirstHalf
- TheSecondHalf
- Dimensions
- ObjectLocation
  - Monastery
  - Church
  - Chapel
- PrivateCollection
  - Museum
  - Gallery
- ObjectLocationAddress
- Country
- Province
- Town
- Village
- Character
- CanonicalCharacter
- IconographicalScene
  - ConditionState
  - IconographicalTechnique
- BaseMaterial
- Collection

GENERATING OF UNIQUE IDENTIFIERS

Every ontology instance requires a unique identifier. This identifier is used in the references to the corresponding instance. The original data do not have such identifiers for all objects (only the top level iconographic objects have). Additional requirement is that the identifiers are generated in such way that every time a given instance is lifted it gets the same identifier. Our solution to identifier generation is the usage of hash functions as MD5 [1] or SHA-1 [8] on the concatenation of the data properties of an instance. As there is always a possibility the hash function will return the same value for different inputs one can use as an identifier the concatenations of the hashes of all of the values of data properties of an instance or even the plain concatenations of the values but in the last two cases the identifiers will become rather long. Another solution applicable, when one lifts the dataset to the semantic level as a whole, he can use the first solution but check for duplicated
hash values and if they occurs to concatenate a sequence number to the duplications to make them unique.

**MAPPING TABLE AND ALGORITHM**

Our implementation of lifting and lowering is based on a table (we call it mapping table) that represents the mappings between the structure of the native data instances and their corresponding semantic instances. The advantages of this approach are that mappings are easier to read and maintain as they are in practice a form of configuration of the transformation algorithm and the lifting and lowering phases use the same table.

The algorithm requires the input and output data to be in XML format. If it is not the case (for example input data is in different format – SQL Tables, Object model, etc) a filter or transformer can be put in the XML pipeline to convert it to XML. The mapping table consists of XPath like expressions that refer to the elements of the input document and output ontology instances graph and how the corresponding value(s) is/are transformed.

The lifting algorithm is recursive and works the following way:
1. Starting from the root level of the input XML document.
2. Looping through all the siblings from a given level:
   a. If the current element corresponds to a simple type and data property in the mapping table, it is transformed according the corresponding value transformation rule.
   b. If the current element corresponds to a complex type or object property in the mapping table the algorithm invokes itself recursively for any of its children in order to process the nested elements of the current complex type and when nested elements are processed and their unique identifiers are returned the complex type element is transformed to the output document.
3. Generating a unique identifier of the newly created instance and return it.
4. Exit

Important part of the algorithm is value transformations. First observation is that the database contains just one class of objects (Icon) but in the ontology there are several subclasses of the IconographicObject class. Which class instance will be the result of the transformation depends on the value of the “type of Iconographic Object” property.

Another observation is that a scalar value in the database can correspond to ontology class instance on the semantic level. Let see one example how one XML instance element is lifted to the corresponding ontology instance:

The fragment of the XML instance document defining the location of certain icon is:

```xml
<Identification>
  <Location>
    <Country>Bulgaria</Country>
    <Province>Sofia</Province>
    <Gallery>National Art Gallery</Gallery>
  </Location>
</Identification>
```

The transformation rules that correspond to the Location element are defined as follows:

Source Location: /Identification/Location/Province/
Semantic Location: /ObjectLocation/ObjectLocationAddress/address_has_Province/
The above rules mean that the values available in the XML instance document at the positions defined with the source locations (XPath expressions) are transformed to the places in the ontology instance graph specified by the corresponding semantic locations. Note that the semantic location expressions are not XPath expressions. They just look similar because we deliberately selected similar notation to specify the paths in the graph containing ontology instances and relations (object and data properties) between them.

In the above example the XML elements Province has the value Sofia. When it is lifted to semantic level the corresponding relation in the semantic location from the mapping rule is address_has_Protvince that is an object property with domain class ObjectLocationAddress and range class Province. Therefore a new instance of the class Province is created. The unique MD5 hash value that corresponds to the province_has_name data property is EE416842BA544449D15A2F41316A8339. Hence the ontology instance of class Province created to represent the province of Sofia get the following identificator Province_EE416842BA544449D15A2F41316A8339. The algorithm checks whether this instance has already been created during the lifting process of previous icon and if it is not available it generates and stores what is shown below.

```xml
<Province rdf:about="#Province_EE416842BA544449D15A2F41316A8339">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <province_has_name>Sofia</province_has_name>
</Province>

<Country rdf:about="#Country_5002A5722B7CDCF3ADF428618F5C2F93">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <country_has_Name>Bulgaria</country_has_Name>
</Country>

<Gallery rdf:about="#Gallery_1AEC592210F0F2BAFF65E36CCA899844">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <gallery_has_name>National Art Gallery</gallery_has_name>
</Gallery>

Then using the instance identificators retuned from the deeper levels of the recursion the corresponding ObjectLocationAddress and object properties address_hasCountry, address_has_Province, and address_has_Gallery are created with references to the instances discussed above:

```xml
<ObjectLocationAddress rdf:about="#ObjectLocationAddress_5730C383B6FD8314949C5B9109505950">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
  <address_has_Country rdf:resource="#Country_5002A5722B7CDCF3ADF428618F5C2F93" />
  <address_has_Province rdf:resource="#Province_EE416842BA544449D15A2F41316A8339" />
  <address_has_Gallery rdf:resource="#Gallery_1AEC592210F0F2BAFF65E36CCA899844" />
</ObjectLocationAddress>
```
The lowering algorithm works the same way as the lifting one but it uses the mapping rules table in the opposite direction. It processes the ontology instance graph and maps the semantic locations to their corresponding paths in the generated the XML instance document.

CONCLUSIONS AND FUTURE WORK

In this paper we presented an algorithm for lifting and lowering the data from the database of digital library “Virtual Encyclopedia of Bulgarian Iconography”. This algorithm was developed in the scope of the SINUS project [8]. Our objective was to transform the data from its relational form to semantic level given the previously defined ontology. The later will be used for building and executing wide range of semantic queries against the ontology instances in order to infer information that can not be obtained from the original digital library. At the same time our work is also a practical example for upgrading existing legacy system to semantic web level.

REFERENCES

[2] OWL http://www.w3.org/TR/owl-ref/

ABOUT THE AUTHOR

Assoc. Prof. Ivo Marinchev, PhD, Department of Technologies for Knowledge Management and Processing, Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Phone: (+ 359 2) 870 75 86, E-mail: ivo@iinf.bas.bg.