Modeling Applications for the Semantic Web

Fernanda Lima and Daniel Schwabe

Depto de Informática, PUC-RIO, R. Marquês de São Vicente, 225, Gávea,
Rio de Janeiro – RJ – 22453-900 – Brasil
{ferlima, schwabe}@inf.puc-rio.br
http://www-di.inf.puc-rio.br/~schwabe

Abstract. This paper proposes the Semantic Hypermedia Design Method, SHDM. By extending OOHDM with primitives taken from Semantic Web languages such as DAML+OIL, we show how a larger, easier to evolve, set of applications can be specified. Such applications also allow tapping the richness of resource descriptions that are becoming available with the Semantic Web.

1 Introduction

The Semantic Web is currently an active topic of research and industry efforts, as it is regarded as the next evolutionary step of the current Web. The main goal of this future Web is to have a large amount of data available with its metadata, to help machines and humans find and process useful resources as well as reuse data across various applications [21].

The major emphasis so far has been in the search for useful resources, and many interesting proposals for organizing and searching Web data based on the Semantic Web are being put forward. For most people, though, the Web is important because of the functionality provided by Web applications, through which they can not only access, but also process the information stored in the Web itself. Therefore, much of the Semantic Web's promise can only be delivered if Web applications are able to fully take advantage of its added (meta) information. Therefore, in addition to metadata about Web resources, we must also provide metadata about applications.

We have been investigating Web application design models for several years in the context of Object Oriented Hypermedia Design Method (OOHDM) [9, 8]. Being a model-driven approach, it stands as a logical candidate to be integrated with the Semantic Web approach, since its models can be used as metadata describing the application. Our goal is to provide authoring methods that help designing applications that can tap on the Semantic Web. In particular, the purpose of this paper is to show that we can express this information building SHDM - Semantic Hypermedia Design Method, a new version of the original method. A complementary goal is to evolve more traditional design methods such as OOHDM by enriching the languages used to specify its various models with Semantic Web languages such as RDF [16], RDF Schema (RDFS) [17], DAML+OIL [19], and OWL[22], while keeping the basic underlying fundamental abstractions.

J.M. Cueva Lovelle et al. (Eds.): ICWE 2003, LNCS 2722, pp. 417–426, 2003.

In addition to these general languages, we also enrich navigation modeling by providing primitives to specify Faceted Navigations, where we can express abstractions for faceted access structures and faceted contexts in a concise model. This allows the design of richer access structures, providing the user with more flexible ways to reach the set of objects that are relevant to task at hand. We also offer primitives that allow the Web application designer to describe concisely hierarchical faceted metadata.

This paper is organized as follows. In Section 2, we describe the main concepts of the SHDM method, its main design steps, showing how one can take advantage of more expressive modeling primitives extending the original OOHDM, both for data and navigational models. In this section, we use an Art Ontology as an example to show a summarized example of the method. In Section 3 we conclude and make brief comments about our future work.

2 The SHDM Method

One of the cornerstones of OOHDM is that it explicitly separates conceptual from navigation design, since they address different concerns in Web applications. Whereas conceptual modeling and design must reflect objects and behaviors in the application domain, navigation design is aimed at organizing the hyperspace taking into account users' profiles and tasks. Navigational design is a key activity in the implementation of Web applications and we advocate that it must be explicitly separated from conceptual modeling [8].

The foundations mentioned above are maintained in this new version of the method called SHDM, enriched with several new mechanisms inspired by the languages being proposed for the Semantic Web. The first step is treating "information items" described in described in the Conceptual Model, and in the Navigation Class Model of OOHDM as resources manipulated in Semantic Web languages, such as the W3C Resource Description Framework (RDF), which is used to describe resources and their properties. By generalizing the concept of "Web resource", RDF can be used to represent information about anything that can be identified on the Web. The characterization of resources in SHDM is done using ontology definition languages such as RDF Schema, DAML+OIL and the recent "work in progress" W3C Web Ontology Language (OWL), expressing more advanced features such as constraints (restrictions), enumeration and datatypes according to XML Schema[1].

We would like to stress that, even though we know that the Semantic Web underlying framework (RDF) is not object-oriented (OO), we still find it useful to use some of the OO modeling principles, mainly because it decreases the level of granularity, suppressing details by allowing grouping of descriptions.

[1] In our examples we used DAML+OIL only because OWL in still at "W3C Working Draft" status, i.e., not completely specified as a Recommendation. Since OWL is derived from the DAML+OIL we plan to make any adaptations easily as soon as it reaches a more mature status.

In this paper we focus more on the main SHDM novelties; more details can be found in [4]. In the following subsections we will specify two of the activities, namely Conceptual Design and Navigational Design.

2.1 The Conceptual Design

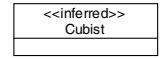
During the SHDM Conceptual Design step we build a model (the Conceptual Model) showing classes and their relationships specifically related to a domain. Classes are described as in object-oriented (OO) UML models [6] with three distinguished details on attributes: they can be multi-typed (representing different perspectives of the same real-world entity), they are described with multiplicity (referring to the number of times the attribute may occur in instances) and they can have explicit enumerations (defining the possible values for that attribute in instances). Relations are described also as in OO UML models, with one additional detail: relations can be specialized creating subrelation hierarchies.

The conceptual model obtained using the UML class diagram can be mapped to a RDF/XML serialization format according to heuristic rules [4] summarized next. In the following subsections we present the notations of the new Conceptual Model primitives, their general semantics and their mapping to one of the Semantic Web languages, DAML+OIL.

When comparing the object-oriented model (OO) with the RDF model it is possible to state that the concepts of classes and subclasses (specialization and generalization relations) can be modeled equivalently. However there is a significant difference in modeling OO attributes and OO association relations in the RDF model. These two OO abstractions are modeled through RDF properties indistinctly, i.e., in RDF models there is no distinction between a property that describes a class (attribute) and a property that describes an association relation with another class. In addition, RDF properties can be specialized through subsumption relation, allowing the creation of subproperties. Our Conceptual Schema takes advantage of these characteristics as shown below; for reasons of space, only the main ones are detailed.

Every class is mapped to a DAML+OIL Class, modeling attributes and relationships as properties. We use DAML+OIL extensions defined as *Datatype* and *ObjectType* Properties to represent attributes and relationships, respectively. Attribute multiplicity is mapped to *minCardinality* and *maxCardinality* on specific properties. Attribute enumerations are mapped to the constructor *one of*, providing a means to define a class by direct enumeration of its members, in such a way that no other individuals can be declared as belonging to the class. Datatypes are defined as in XML Schema.

In addition to defining classes and instances declaratively, DAML+OIL and other Description-Logics languages let us create intensional class definitions using Boolean expressions and specify necessary, or necessary and sufficient, conditions for class membership. These languages rely on inference engines (classifiers) to compute a class hierarchy and to determine class membership of instances based on the properties of classes and instances [5]. SHDM incorporates these Semantic Web languages approaches using Inferred Classes, represented graphically as UML stereotypes (see Fig. 1).




```
<daml:Class rdf:about="Cubist">
        <daml:Restriction>
          <daml:onProperty rdf:resource="#style"/>
          <daml:hasValue rdf:resource="#Cubism"/>
        </daml:Restriction>
</daml:Class>
<daml:Class rdf:about="#Cubist">
   <rdfs:subClassOf rdf:resource="#Painter"/>
</daml:Class>
```

```
< daml:Class rdf:ID="Cubist">
    <daml:intersectionOf rdf:parseType="daml:collection">
      <rdfs:Class rdf:about="#Painter"/>
      <daml:Restriction rdf:about="#Cubism-Restriction"/>
    </daml:intersectionOf>
</daml:Class>
<daml:Restriction rdf:ID="Cubism-Restriction">
    <daml:onProperty rdf:resource="#style"/>
    <daml:hasValue rdf:resource="#Cubism"/>
</daml:Restriction>
```

Fig. 1. Inferred Class, with two alternative DAML+OIL equivalent definitions

This simple example states that a "painter" belongs to the "cubist" class if the property value of his/her style is "cubist" or that the "painter" subclass cubist is the intersection of classes "painter" and the set of resources whose "style" property satisfies the condition of having its value equal to "cubist". Since this language relies on inference engines to compute a class hierarchy, we could validate our model using any DAML+OIL inference engine. Other than the inferred classes, we defined in our metamodel another stereotype to represent class hierarchies with arbitrary depth, called "arbitraryClassHierarchy", but due to space restrictions we will not detail it in this article.

2.2 The Navigational Design

During the Navigation Design step we produce a Navigational Model over a conceptual domain, according to user profiles and the tasks that will be supported. As stated in [9, 8], during Navigation Design we are interested in specifying which objects will be reached by the user (the nodes) and the relations between these nodes (the links). We also specify the sets of objects within which the user will navigate (called contexts) and in which way s/he will access these contexts (the access structures). We are also able to specify different contents for the nodes according to the contexts within which they are reached (inContext classes). The basic SHDM navigation primitives are defined at this point, as in OOHDM.

For SHDM, we have identified the need to model some new access structure primitives in order to take advantage of the increased availability in the WWW of taxonomies, which we have named Faceted Access Structures, inspired by the facet concept initially proposed in library and information sciences [7]. Simply put, a facet can be considered as a category. In [12] Taylor defines facets as "clearly defined, mutually exclusive, and collectively exhaustive aspects, properties, or characteristics of a class or specific subject".

In a faceted classification scheme, the facets may be considered to be dimensions in a Cartesian classification space, and the value of a facet is the position of the artifact in that dimension. Within each facet, subfacets or more specific topics are listed. The breakdown continues into subfacets within subfacets. The items in each

subfacet, in general, are ordered from more general to more specific, complex or concrete.

We define facet hierarchies based on our navigational attribute types- which are in fact metadata about our Web application. Each hierarchy is defined independently, in order to organize content along a particular dimension. This will be exemplified later on.

Navigational Contexts remains a very important navigational primitive in our approach, since it allows us to describe sets of navigational objects relevant to the user during a task. The novelty lies in the fact that the language used to define contexts is more expressive than the previous one.

Next we show the notations of the new Navigational Model primitives, together with their mapping to a Semantic Web language such as DAML+OIL.

The Navigation Design activity generates two schemas, the Navigational Class schema and the Navigational Context schema. The first defines all navigable objects as views over the application domain. The navigable relations are links between nodes and also the new subrelations that allow a new type of navigation based on subsumption relations between links. The second schema defines navigational contexts (the main structuring primitive for the navigational space), access structures used to reach these contexts and links that connect them.

The representation of navigational classes is graphically identical to OOHDM, using the same innovations introduced for conceptual attribute notations. Navigational classes represent views of conceptual classes, including directly mapped conceptual attributes, derived attributes and also attributes from other conceptual classes. The mappings are specified using an RQL [3] query, exemplified below.

RQL mapping:

RQL query	Description
select y from { Artist } firstName { y }	retrieves the firstName of Artists
select y from {x} creates { y } where x= “parameterA”	retrieves all Artifacts of a specific Artist

As in the conceptual schema, SHDM also allows sub-relations in the navigational class schema.

In addition to using sub-relations defined in the conceptual model, it is possible to use sub-relations in the mapping of the conceptual model into the navigational class model. For example, it is possible to define navigational sub-relations of “creates” by restricting its subclasses, for instance, only those whose counter-domain is a subclass of Painting. In Fig. 2 we illustrate a combination of DAML+OIL and RQL to specify the mapping from the conceptual to the navigational model (identified with namespace shdm).

A context groups objects related to each other by some aspect (e.g., common attributes or being related to a common object) and organizes these objects as sets of nodes, defining in which way they may be accessed (e.g., sequentially).

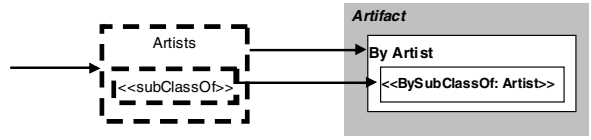
Navigation contexts may be further specified as groups of contexts, since it is possible to sometimes parameterize their defining property. For example, “Sculpture by Material” is actually a set of sets; each set is a context, determined by one value of the “material” attribute.

```
<daml:ObjectProperty rdf:ID="etches">
        <daml:subPropertyOf rdf:resource="#creates">
        <shdm:rql query:value =”select y  from { x } creates { y : cult:Watercolor” }
        <!-- retrieves all instances according to the description in the text above -->
        <daml:range rdf:resource="#Painter"/>
        <daml:domain rdf:resource="#Watercolor"/>
</daml: ObjectProperty >
```

Fig. 2. Navigational Class attribute mapping using sub-relations

There is an analogous definition for contexts whose property is based on 1-to-n relations, such as “Sculpture by Sculptor”.

Access structures are indexes (collections of links) that allow the user to reach navigation objects (within some context). SHDM allows defining both Access Structures and Navigational Contexts using meta-data properties. The <<subClassOf>> stereotype indicates that the corresponding element (access structure of navigational context) is a set of elements, one for each sub-class.



RQL mapping:

RQL query	Description
http://www.icom.com/schema.rdf#Artist	retrieves the IDs of Artist instances
select y from { x }creates { y } where x= “parameterA”	retrieves the IDs of Artifacts that were created by a chosen Artist (the parameter). This query includes the subrelations paints and sculpts
select y from { x }paints { y } where x= “parameterA” and $y=Paintings	retrieves the IDs of Painting that were created by a chosen Artist/Painter (the parameter)

Fig. 3. Access Structures and Navigational Contexts defined based on meta-model properties

In Fig. 3 we show the graphical notation, and the RQL statements for an example. In The Artist access structure represents a list of links to all artists (the order is specified in the corresponding card). The inner dashed box represents sub-sets of Artist defined according to its subclass, for example Painter and Sculptor. The context Artifact by Artist that is composed of all artifacts created by a specific artist. This context can be access by choosing an artist as a parameter of selection. The innermost box signifies that the user can also choose any subclass of Artist to group the artifacts.

Faceted Access Structures and Faceted Navigational Contexts are defined using the <<faceted>> and <<ByValidFacetComb>> stereotypes. In Fig. 4 the outside dashed box denotes the valid combinations of facets to reach the Artifact navigational class and three inside dashed boxes that indicate the possibility of choosing just one of the facets. The context Artifact by ValidFacetCombination exemplifies the possibility of

accessing Artifacts by any combination of Region or Style. Similarly, Artifact by Style – Faceted stands for all sets of artifacts grouped by Style and by its subclasses.

Faceted elements are detailed in the corresponding specification cards, illustrated in Fig. 5. The designer can use a graphical notation to annotate in the facet hierarchies numbers that represent the invalid combinations. When the designer describes the combinations, he/she does not have to make it extensively; it is enough to only annotate the nodes that are superclasses of the invalid combinations, at any level of the trees. The enumerated combinations can be generated by an algorithm such as proposed by Tzitzikas in [13].

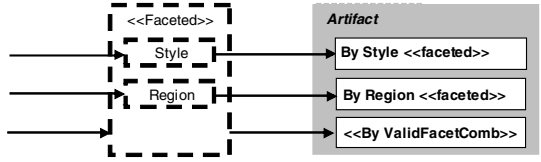
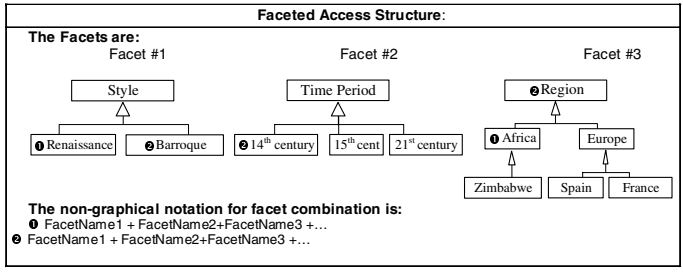


Fig. 4. Faceted Access Structures and Faceted Navigational Contexts



Valid facet combinations are:
Renaissaince>TimePeriod>Europe *(ie, all combinations of TimePeriod values and Region>Europe values)*
Barroque>15th century>Region
Barroque>21st century>Region

Example of the concise declarations of invalid facet combinations are:
❶ Renaissance+ Africa
❷ Barroque+14th century

Explicitly the invalid facet combinations are:
Barroque>14th century>Region
Barroque>14th century>Africa
Barroque>14th century>Europe
Barroque>14th century>Africa>Zimbabwe
Barroque>14th century>Europe>Spain
Barroque>14th century>Europe>France
Renaissance>Africa
Renaissance>Africa>Zimbabwe

Fig. 5. Navigational Faceted Access Structures Specification Card

We developed an example inspired by a Museum example [2] that we briefly outline, focusing on illustrating the novelties in the navigation schemas.

Fig. 6 presents the Conceptual Model for the example. The superclass Artist has an association relation ("creates") with the superclass Artifact, specialized by: "paints" and "sculpts", meaning that whenever somebody *paints* something, he/she is also *creating* it. It also means that a query for the instances of this model to ask for domain and range of the "creates" relation will also retrieve the union of the subrelations.

Fig. 7 shows the Navigational Class schema. Notice that node Artifact includes attributes that did not belong to the original Conceptual Class, such as *style*.

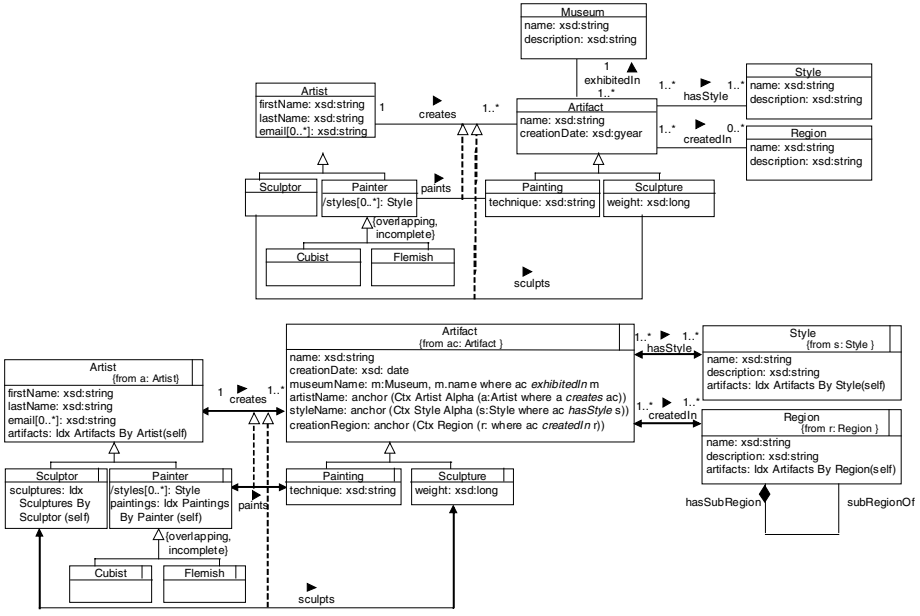


Fig. 6. Art Conceptual and Navigation Class Schema

In Figure 19 we present some novelties in the Navigational Context schema.

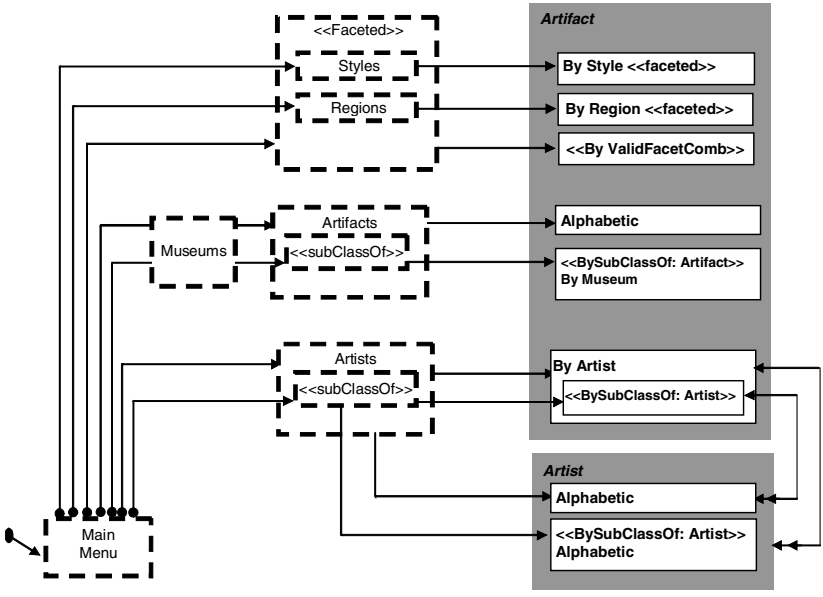


Fig. 7. Art Navigational Context Schema

The abstraction power of the notation proposed is exemplified in several places. The use of compact facet specifications avoids explicit enumeration of all possible combinations, including those not know at design time. The same is true for the use of the <<subclassOf>> stereotype, since it allows definitions of access structures and contexts for an arbitrary class hierarchy. Since we have used RQL, we are able to query both data and metadata. For instance, we can now define a context "Artifact by Style", without knowing ahead of time all possible values (or subclasses) of "Style". If the user later adds a new subclass to "Style", and its corresponding instances, the same application specification still applies. In this sense, SHDM specifications could be regarded as specifying frameworks (as in [10]). Although not shown in here, similar reasoning can be applied to inferred classes.

3 Conclusions

In this paper we have argued that Web application design methods can benefit from modeling language primitives being proposed for the Semantic Web, such as RDF, RDFS, and DAML+OIL. Some approaches, such as HERA [1] propose directly using RDF and RDFS, or slight extensions, as the basic ontology modeling language, equivalent to our conceptual modeling. Others, such as OntoWebber [2], add additional ontologies on top of them, to cover other aspects of application design, such as site structure. In contrast, we have kept the traditional UML-like object model, extending it with a few primitives such as sub-relations, from RDF, and anonymous classes defined through restrictions, from DAML+OIL.

We have followed the original OOHDM approach of defining the Navigational Class model as a mapping of the Conceptual Model, but using RQL as the mapping specification language, which is able to query DAML+OIL models. Another benefit brought by SHDM is the ability to concisely specify faceted navigation structures. It was shown how facet specification is equivalent to very large enumerations of possible navigations paths. With the increasing availability of domain taxonomies, this will allow such taxonomies as part of the navigation structure of applications designed using SHDM. In addition, the resulting applications are able cater to varying user profiles by providing alternative navigation paths better suited to each particular case.

We are now investigating how SHDM can be extended to personalized and adaptable web applications. Additional topics being pursued include integration of interface and interaction models, of application functionalities. In [4] an implementation architecture is outlined, based on the Sesame [11] environment.

References

1. Frasincar, F., Houben, G-J., Vdovjak, R.: "Specification Framework for Engineering Adaptive Web Applications", In Proceedings of the WWW2002, Honolulu, USA, 2002.
2. Jin, Y., Decker, S., Wiederhold, G.: "OntoWebber: Building Web Sites Using Semantic Web Technologies", http://www-db.stanford.edu/~yhjin/docs/owedbt.pdf

3. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D. and Scholl, M.: "RQL: A Declarative Query Language for RDF", In Proceedings of the 11th International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA, May 2002, http://139.91.183.30:9090/RDF/RQL/index.html
4. Lima, F.: “Modeling applications for the Semantic Web”, PhD Thesis (in preparation), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2003.
5. Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R., Musen, M.A.: “Creating Semantic Web Contents with Protégé-2000”, IEEE Intelligent Systems, Vol. 16, No. 2, March/April 2001, special issue on Semantic Web, 60–71.
6. OMG: "Unified Modeling Language Specification version 1.3 (UML 1.3)", June 1999.
7. Ranganathan, S.: “Colon Classification, Basic Classification”, 6th ed., New York: Asia Publishing House, 1963.
8. Rossi, G., Schwabe, D. and Lyardet, F.: "Web Application Models Are More than Conceptual Models" In Proceedings of the ER'99, Paris, France, November 1999, Springer, 239–252.
9. Schwabe, D. and Rossi, G.: "An object-oriented approach to Web-based application design" Theory and Practice of Object Systems (TAPOS), October 1998, 207–225.
10. Schwabe, D., Rossi, G., Esmeraldo, L. and Lyardet, F.: "Engineering Web Applications for reuse", IEEE Multimedia 8(1) – Special Issue on Web Engineering, Jan-Mar 2001, 20–31.
11. Sesame.aidministrator bv.: "Sesame: A Generic Architecture for Storing and Querying RDF and RDF-Schema", Technical Report, http://sesame.aidministrator.nl/, 2001.
12. Taylor, A. G.: “Introduction to Cataloging and Classification”, 8th ed. Englewood, Colorado: Libraries Unlimited, 1992.
13. Tzitzikas, Y., Spyratos, N., Constantopoulos, P. and Analyti, A.: "Extended Faceted Taxonomies for Web Catalogs", Third International Conference on Web Information Systems Engineering, WISE 2002, Singapore, December, 2002.
14. van Harmelen, F., Horrocks, I. and Patel-Schneider, P.: "Reference Description of the DAML+OIL (March 2001) Ontology Markup Language", http://www.daml.org/2001/03/reference.html
15. W3C1998: "A Discussion of the Relationship Between RDF-Schema and UML", http://www.w3.org/TR/NOTE-rdf-uml/, 1998.
16. W3C1999: "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
17. W3C2000: "Resource Description Framework (RDF) Schema Specification 1.0", W3C, Candidate Recommendation 27 March 2000, http://www.w3.org/TR/2000/CR-rdf-schema-20000327/
18. W3C2001a: "XML Schema Part 2: Datatypes", W3C Recommendation 02 May 2001, http://www.w3.org/TR/xmlschema-2/
19. W3C2001a: “DAML+OIL (March 2001) Reference Description”, W3C Note 18 December 2001, http://www.w3.org/TR/daml+oil-reference
20. W3C2002a: "Requirements for a Web Ontology Language", W3C Working Draft 07 March 2002, http://www.w3.org/TR/webont-req/
21. W3C2002c: “Semantic Web Activity Statement”, http://www.w3.org/2001/sw/Activity/, retrieved 2002/11/03.
22. W3C2002c: “Web Ontology Language (OWL) Guide Version 1.0”, W3C Working Draft 4 November 2002, http://www.w3.org/TR/2002/WD-owl-guide-20021104/