# Getting Ready Web Engineering Methods for the Semantic Web. Putting Ontologies into Practice.

Victoria Torres[1], Joan Fons[1], Oscar Asensi[2] y Vicente Pelechano[1]

[1]Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia
{vtorres, jjfons, pele}@dsic.upv.es

[2]Departamento de Lenguajes y Sistemas Informáticos.
Universidad de Alicante.
Campus de San Vicente del Raspeig.
Apartado 99. 03080 Alicante. España.
oasensi@dlsi.ua.es

**Abstract.** Current Web Engineering methods develop "closed" web applications from conceptual models. This fact makes difficult the integration and the interoperability of different web applications. In this context it is necessary to establish a technological framework where the application data and functionality could be represented and shared between different web applications. Semantic web languages provide an appropriate framework to achieve these non-functional requirements. Ontologies are proliferating to enable interoperability between Internet-connected applications. Web Engineering methods should be prepared to face up this new challenge. A first attempt in this community is based on the transformation of conceptual schemas into ontologies. This strategy does not take into account navigational and presentation models. This work takes advantage from all these models (navigational and presentation) enriching the web implementations with all the knowledge gathered during the modeling and design process. Our approach provides different ontologies as a basis to enable a more fruitful interchange of knowledge between web applications. We provide a semantic representation of web applications that enables not just to merely access to static information. We also provide a navigational ontology that can be queried through the use of a semantic query language.

**Keywords.** Web Engineering; Conceptual modeling, Semantic Web, Ontologies, UML.

## 1. *Introduction*

From a methodological point of view, the most outstanding approaches (OOHDM [11], WebML [1], OOH [4], WSDM [2], UWE [5], etc.) focus their efforts on defining web applications from conceptual models that allow them to systematically obtain closed implementations. These approaches provide abstraction mechanisms that make it easy to conceptualize and develop the web applications allowing the analyst to specify hypermedial and functional requirements. The mechanisms for hypermedia modeling allow the analyst (1) to define web pages as conceptual schema views and (2) to interconnect these views to define the navigational structure of web applications.

Web engineering methodologies end up with closed software applications. This fact makes difficult to integrate different web systems that should desirable to be cooperating somehow. Then, it is necessary to establish a framework where the application domain schema, data and functionality could be represented and shared between all these different systems. In this context, the semantic web languages provide an appropriate framework to make all this information available to any web system.

It is well known that the Semantic web is supposed to be the next web generation [12]. But until this new generation could be considered as a fact, more research and work has to be done in this area. If ontologies proliferate and consolidate to enable unprecedented interoperability between Internet-connected devices web engineering community has to be prepared to face up this new challenge and not to fall behind.

Transforming models produced during the Web development process into a semantic web language is a first attempt to get ready for the next web generation. With this work done, at least we will be able to share knowledge between a set of web applications developed for the web environment. Usually, those approaches taken by web engineering researchers (SHDM [6]) are based only on the conceptual models, not taking into account other models developed during the web engineering process. The proposal introduced in this work is based on the idea of making advantage from all models (including navigational and presentation model) specified during the web development process.

The main contribution of this work is to enrich the web implementations with all the knowledge gathered during the modeling and design process, providing different ontologies as a basis to enable a more fruitful interchange between systems. Once we have a semantic representation of web applications we will be able not just to merely access to static information but as a next step to automatically locate, select, employ, compose and monitor Web-based services.

The rest of the paper is structured as follows: section 2 shows how the web is evolving and how web engineering methodologies have to adapt in order to deal with this new situation using semantic web languages as the key element. Section 3 briefly presents the navigational concepts of a web application by means of a metamodel. Then, it is presented the transformation of this metamodel into a specific semantic web language. An example for querying some navigational properties is also shown. Finally, section 4 gives some conclusions and presents further work.

## *2. The role of the Semantic Web in Web-Engineering Methods. The Ontology way*

As the Web is evolving into a new generation, the kind of applications developed for this environment has to be adequate to this change. This indeed means methodologies developed to create web applications have to face this change in some way. Nowadays web application designers are entrusted to build distributed and interoperable systems. To do this in a more accurate way it is necessary to provide a semantic description about all data and functionality that is accessible through the web.

It has been suggested that there exists a classic chicken and egg situation with the Semantic Web; i.e. without applications using semantics no one will build semantic content and conversely without content no one will build applications. Just like the original appearance of Web sites, early adopters of the new technology benefited and were better prepared for when the Internet medium matured. Early adopters of semantic web technologies can benefit today while at the same time preparing for the future. If web methodologies make an effort to deal with the Semantic Web, they will both contribute to the development of the semantic web generation and will be prepared to attend any semantic web consumer.

Nowadays, our web applications cannot be completely closed as they were built until now. It is very important to provide in a structured and schematic way the data and functionality that our systems are dealing with. Moreover, to ensure that consumers are doing a proper use of this data and functionality it is necessary to provide some semantics associated to all these information. It is essential when we are thinking that our applications will be integrated in a more complex system, made up of different and existing applications.

Web application methodologies are fairly ready to share all its data and functionality because they work based on conceptual models that gather all the domain information about the system intended to build [10]. *All they have to do is defining a strategy pot "put in practice" the gathered concepts at conceptual level introducing them at implementation level using the semantic web concepts.*

The research community is doing a great effort developing languages and technology to contribute in the development of the Semantic Web. As result languages as RDF, RDF-S, DAML+OIL or OWL has been developed in order to represent semantic information about arbitrary resources. These languages usually provide mechanisms for describing groups of related resources and the relationships between these resources. These languages are designed to be used by applications that need to process the content of information instead of just presenting information to humans.

In particular, the OWL Web Ontology Language (OWL [8]) has more expressive power than earlier W3C languages. It adds more vocabulary for describing properties and classes (among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties and enumerated classes). It enables the creation of ontologies for any domain and the instantiation of these ontologies in the description of specific Web sites. This language (1) can be used to formalize a domain by defining classes and properties of those classes, (2) define individuals and assert properties about them, and (3) reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language.

Analyzing the models produced in web engineering it is almost commonly agreed the necessity of at least three models. These models are: the conceptual[1], the navigational and the presentation model. The *Conceptual Model* is aimed to describe the domain of the system being developed. This means, describing the concepts and the relationships existing between all of them. The *Navigational Model* specifies the navigation semantics associated to the system users. Web sites and portals are used today to provide different views of an enterprise's data and processes, getting the right data to the right user at the right time. This idea encourages us to say that navigational ontologies could help data consumers to access the right data and to place the content in the right place. Finally, the *Presentation Model* is aimed to organize the views defined for each kind of user in a specific fashion.

At this point we should ask ourselves which of these models are interesting to get transformed into these semantic web languages. Depending on what kind of information we want to be made available it is interesting to transform some of them.

It is quite clear that the ***conceptual model*** should be published in order to provide consumers the concepts our system is about. In fact, this is the first step followed by all these web developers that want to provide semantic knowledge of their web sites. Then, no one should doubt about the importance of this model.

Regarding to the ***navigational model*** we initially could doubt about the importance of mapping this information into semantic web languages. But we think this model can help data consumers in the way of making clear issues such as "to who is suitable this information (what kind of user)" or "which would be the most interesting way to access some specific data".

The ***presentation model*** can be seen as a proposal made by the designer (who is supposed to have a deep knowledge in the system domain) where it is specified some presentation requirements (based on predefined patterns such as layout, information paging, ordering, etc.) of web applications to obtain the final web interface.

Making all these semantics available, web application methodologies will be able to produce artefacts not just as isolated systems, instead as set of data structured accessible and understandable by others.

As it is said previously in this section, web-engineering methodologies are based on models that gather all different requirements collected during the initial development phases. These models can be quite easily

[1] called Conceptual Schema in the Information Systems Modeling Area

mapped to semantic web languages because these kinds of languages have enough expressiveness to characterise the elements and constraints modelled in the web-engineering models.

OWL provides elements to represent classes, properties, instances of classes and relationships between these classes. Relationship properties such as cardinality constraints can also be represented in OWL through property constraints (see in section 3 some of these transformations). Then, this kind of information can be directly transformed into OWL.

However, there exists some information that cannot be mapped so easily. This is the case of complex datatypes or constraints defined in a language such as OCL. Complex datatypes again have to be artificially transformed into classes providing them the same importance degree as a real class. Constraints usually give important information that has to be considered when manipulating the class diagram. At this moment, this information only could be included in OWL as a property defined as a string.

Next two sections show in detail a study about the kind of knowledge that can be extracted from the conceptual, navigational and presentation models.

## Extracted Knowledge from Conceptual Models

The conceptual model describes the conventional application necessities through the use of OO models. Specifically, class diagrams are designed to represent the static part of the real world that is intended to automate. Usually, this diagram is a model composed mainly by classes and relationships. So, the class diagram provides the ontology that describes the content of the system in a formal shared representation of a domain. The dynamic behaviour of the system is represented by the dynamic and functional models. They provide primitives to represent operations, event pre and post conditions, transactions, triggers, etc.

## Extracted Knowledge from Navigational Models

Current Web Engineering approaches (OOHDM [11], OOH [4], WSDM [2], UWE [5], etc.) extend classical Software Engineering methods by introducing some kind of navigational model to specify the navigational characteristics of web applications. This model defines the navigational view of the system and it is related to a group of users. Usually, navigational descriptions are represented by graphs specifying the connected views over the data and functionality defined in the structural and behavioural model. Nodes of these graphs represent system views and those nodes can be related by means of navigational links. Also, when it is necessary, access structures (such as indexes, guided tours, data search mechanisms, etc.) can be defined to speed up the information exploration.

This model captures some knowledge that would be very interesting to share with other applications:

- Which is the most suitable way to access some data? Which sequence of navigational links and navigational nodes can be followed to access some data and functionality? It is important to remark that this information cannot be obtained from the class diagram, because in this diagram we do not explicit the best way of traversing the structural relationships for each specific user. Some methodologies argue that navigational links has not always to be related to structural relationships from the class diagram. So, in those cases, the importance of the extracted knowledge from this model is clearer.

- Which is the most suitable element (anchor) to access some data? When representing a resource, in terms of the domain knowledge or static personalization, it is useful to know which property has enough semantics to represent the resource itself.

- In which navigational nodes is correct (semantically speaking) to present some data? Navigational nodes gather related information that has sense to be available all together. At the same time, some data can be suitable in different navigational nodes. These nodes include related data that should be presented together from the analyst or system designer point of view.

- Which are the most appropriate ways to distribute/present data and functionality to each kind of use? The navigational nodes give this information for each kind of user. Usually, the distribution of the data and functionality for each kind of users depends on the system specific requirements (who can see some properties and who can execute some functionality).

- Which is the most appropriate view of the conceptual schema to be presented to each kind of user? Navigational nodes give this information distributing data and functionality in a different way to each kind of user.

- Which access structures and searching mechanisms are defined to browse the huge amount of data? Search mechanisms or access structures (such as indexes) are required to access huge amount of data based on some data properties.

### Extracted Knowledge from Presentation Models

The presentation model captures the abstract requirements of data presentation. For instance, this model could provide information such as the most appropriate information arrangement, some properties ordering criteria or whether data is provided gradually or all at once.

Although all three models provide interesting information for very different issues during systems interoperability, in this work we focus just on the navigational model, which adds some value to data and functionality consumers that is not usually taken into account.

## *3. From Navigational Models to Ontologies*

From the software engineering field, the natural way of describing concepts and properties has commonly been done by using models with a precise semantics. Moreover, meta-models have been used to build specific models within a domain of interest. In these terms, the OOWS [3] web engineering method has developed a navigational metamodel that describes the navigational primitives (terms, concepts, relationships) which define a navigational domain. This precise semantics can be described in any ontological language, commonly represented in UML[13] (conceptual schemas) and in a web environment using a semantic web language [10].

It would be desirable to have a public navigational ontology that could be extended by other ontologies in order to provide additional definitions. So, this lack encouraged us to made available the OOWS navigational metamodel as a navigational ontology accessible to anyone.

### Metamodeling Navigational properties

Metamodeling has been a very popular way to define the concepts of specific domain ontologies and unambiguously establish the relationships between those concepts. The UML[13] is the most used metamodel notation in the Object Orientation field of research.
This section introduces a Navigational Metamodel that takes into account the navigational properties of web applications following the Web Engineering principles [7]. This metamodel represents the concepts of the

OOWS approach [ER2003], an Object Oriented method that use conceptual models as basis to develop web applications.
The navigational model of a web application is defined by a set of navigational maps. Each navigational map structures the navigational view of the system for a specific kind of user. A navigational map is made up of a set of navigational nodes (called navigational contexts) and navigational links. A navigational context represents a view over the class diagram. Navigational link define valid node attainability inside the navigational map (graph).
A navigational context is made up of a set of navigational classes and navigational relationships. A navigational class represents a view over a set of attributes and operations of a class from the class diagram. A navigational relationship defines a requirement for retrieving some related data using a structural relationship (association-aggregation-composition or specialization-generalization) at the class diagram.

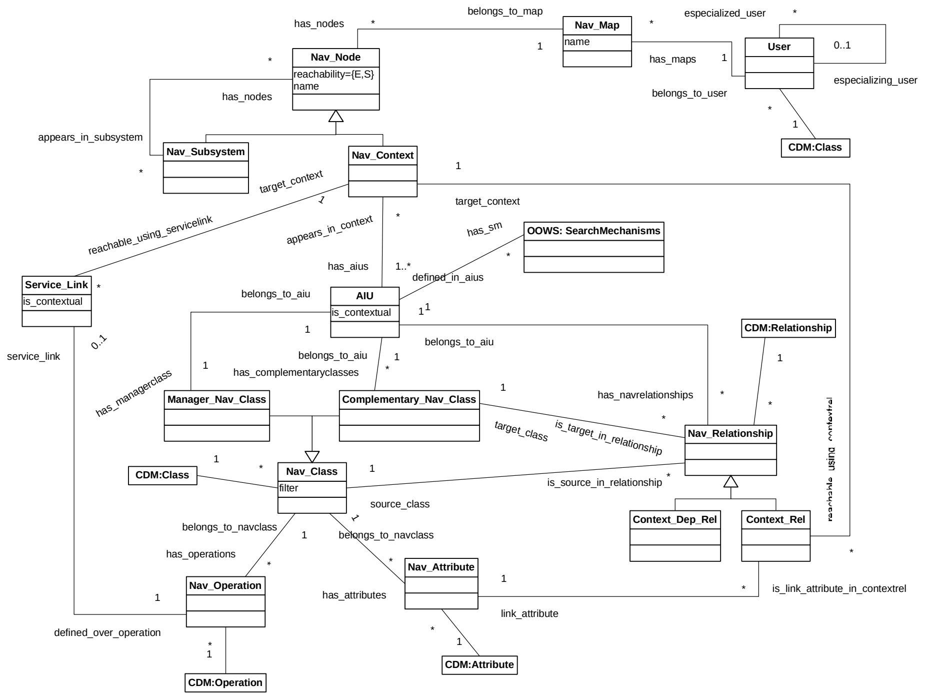

**Figure 1 - Portion of the OOWS Metamodel**

## Navigational properties representation in OWL

Domain schemas or ontologies can be expressed as sets of triples using the classes and properties defined in the semantic web languages (RDF-S or OWL). One thing to take into account when mapping from object-oriented models (i.e. UML) to OWL is that OWL properties are first-class objects and are not defined within the context of a particular class. So, it is necessary to provide mapping mechanisms to avoid conflicting range declarations when the same property is used to represent a field in two different classes. This can be solved by prefixing each property name with the name of the class to which belongs.

Figure 2 shows part of the OWL representation of the navigational metamodel.

```
...
  <!-- Class Navigational Context -->
  <owl:Class rdf:ID="Nav_Context">
    <rdfs:label xml:lang="en">Navigational context</rdfs:label>
    <rdfs:label xml:lang="es">Contexto navegacional</rdfs:label>
    <rdfs:subclassOf rdf:resource="#Nav_Node" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#has_aius"/>
           <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <!-- Navigational Context properties -->
  ...
  <owl:ObjectProperty rdf:ID="has_aius">
    <rdfs:domain rdf:resource="#Nav_Context"/>
    <rdfs:range rdf:resource="#AIU"/>
  </owl:ObjectProperty>

  <!-- Class AIU -->
  <owl:Class rdf:ID="AIU">
    <rdfs:label xml:lang="en">Abstract Interaction Unit</rdfs:label>
    <rdfs:label xml:lang="es">Unidad de interacion abstracta</rdfs:label>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#has_manager_nav_class"/>
           <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="has_manager_class">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#AIU"/>
    <rdfs:range rdf:resource="#Manager_Nav_Class"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="has_complementary_class">
    <rdfs:domain rdf:resource="#AIU"/>
    <rdfs:range rdf:resource="#Complementary_Nav_Class"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="has_nav_relationships">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#AIU"/>
    <rdfs:range rdf:resource="#Nav_Relationship"/>
  </owl:ObjectProperty>


  <!-- Class Navigational Class -->
  <owl:Class rdf:ID="Nav_Class">
    <rdfs:label xml:lang="en">Navigational class</rdfs:label>
    <rdfs:label xml:lang="es">Clase navegacional</rdfs:label>
  </owl:Class>

  <!-- Navigational Class properties -->
  ...
  <owl:ObjectProperty rdf:ID="is_source_in_relationship">
    <rdfs:domain rdf:resource="#Nav_Class"/>
    <rdfs:range rdf:resource="#Nav_Relationship"/>
  </owl:ObjectProperty>

  <!-- Class Manager Navigational Class -->
  <owl:Class rdf:ID="Manager_Nav_Class">
    <rdfs:label xml:lang="en">Manager navigational class</rdfs:label>
    <rdfs:label xml:lang="es">Clase navegacional directora</rdfs:label>
    <rdfs:subclassOf rdf:resource="#Nav_Class" />
  </owl:Class>

  <!-- Manager Class properties -->
  <owl:ObjectProperty rdf:ID="Manager_Nav_class.belongs_to_aiu">
    <owl:inverseOf rdf:resource="#has_manager_class" />
  </owl:ObjectProperty>

  <!-- Class Navigational Relationship -->
  <owl:Class rdf:ID="Nav_Relationship">
```

```
    <rdfs:label xml:lang="en">Navigational relationship</rdfs:label>
    <rdfs:label xml:lang="es">Relacion navegacional</rdfs:label>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#related_to_structural_relationship"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <!-- Navigational Relationship properties -->
  <owl:ObjectProperty rdf:ID="target_class">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#Nav_Relationship"/>
    <rdfs:range rdf:resource="#Complementary_Nav_Class"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="source_class">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#Nav_Relationship"/>
    <rdfs:range rdf:resource="#Nav_Class"/>
  </owl:ObjectProperty>
  ...

  <!-- Class Context Relationship -->
  <owl:Class rdf:ID="Context_Rel">
    <rdfs:label xml:lang="en">Context relationship</rdfs:label>
    <rdfs:label xml:lang="es">Relacion de contexto</rdfs:label>
    <rdfs:subclassOf rdf:resource="#Nav_Relationship" />
  </owl:Class>

  <!-- Context Relationship properties -->
  <owl:ObjectProperty rdf:ID="target_context">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:domain rdf:resource="#Context_Rel"/>
    <rdfs:range rdf:resource="#Nav_Context"/>
  </owl:ObjectProperty>
   ...

</rdf:RDF>
```

**Figure 2 - Part of OOWS Metamodel representation in OWL**

Once we have the navigational metamodel represented into a semantic web language we can instantiate it to define the adequate navigability for each kind of user. Next section shows an example about the sort of questions that can be queried to the navigational metamodel.

## Extracting Navigational Knowledge

In this section we introduce a research group web site as an example to show how the ontology can be queried through the use of a semantic query language. For this purpose we use the OWL-QL query language which is intended to be a candidate standard language and protocol for query-answering dialogs among Semantic Web computational agents. As it is not the objective of this work to give the details of this semantic query language, more information about this project and the syntax overview is available at [9].

Figure 3 shows a portion of the specification of the navigational properties related to the *Member* kind of users for the web application of the example. The navigational capabiblities for this user are represented in a *Navigational Map* on the left side of the figure. This map defines node (context) accessibility. On the right side of the figure, it is shown the detailed description for the *Members* context. This context shows the partial view over the class diagram that Member users will have when he navigates to this node.

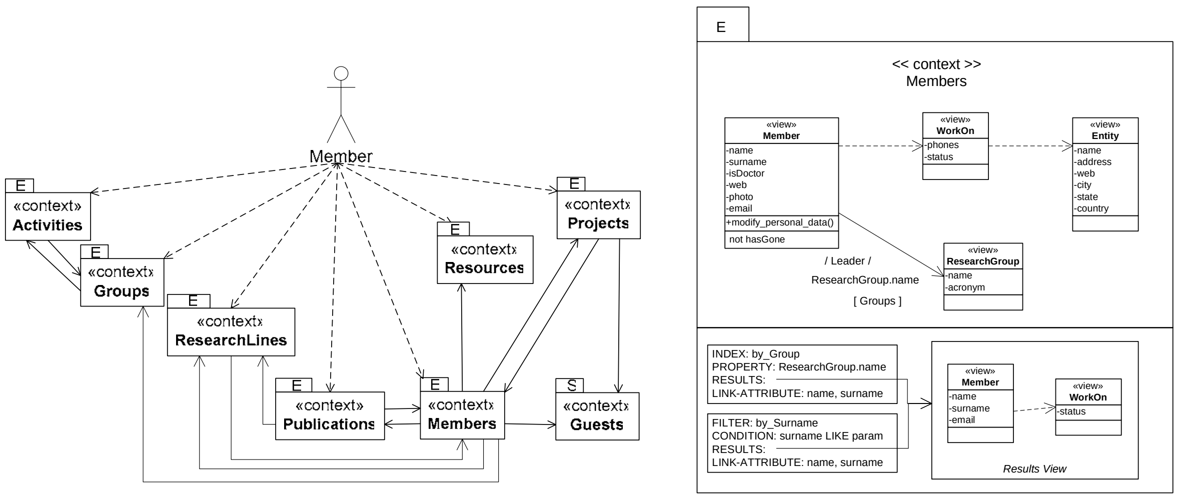

**Figure 3 - Research Group Web Site *Member* Navigational Map and *Members* context**

Now, we show how those conceptual navigational properties could be represented in the OWL semantic web language.

```
...
<nm:User rdf:ID="Member">
 ...
</nm:User>

<nm:Nav_Map rdf:ID="Nav_Map_Member">
    <nm:belongs_to_user="#Member"/>
    ...
</nm:Nav_Map>


<nm:Nav_Context rdf:ID="Groups">
    <nm:belongs_to_map="#Nav_Map_Member"/>
    ...
</nm:Nav_Context>

<nm:Nav_Context rdf:ID="Members">
    <nm:belongs_to_map="#Nav_Map_Member"/>
    <nm:has_aius rdf:resource="#aiu_member"/>
    ...
</nm:Nav_Context>

<nm:AIU rdf:ID="aiu_member">
    <nm:has_managerclass rdf:resource="#Member"/>
    <nm:appears_in_context rdf:resource="#Members"/>
    ...
</nm:AIU>

<nm:Manager_Nav_Class rdf:ID="Member">
    <nm:Manager_Nav_Class.belongs_to_aiu rdf:resource="#aiu_member"/>
    ...
</nm:Manager_Nav_Class>

<nm:Context_Rel rdf:ID="Context_Rel11">
    <nm:source_class rdf:resource="#Member"/>
    <nm:target_class rdf:resource="#RGroup"/>
    <nm:target_context rdf:resource="#Groups"/>
    ...
```

```
</nm:Context_Rel>

<nm:Nav_Context rdf:ID="Activity">
    <nm:belongs_to_map="#Nav_Map_Member"/>
    <nm:has_aius rdf:resource="#aiu_activity"/>
    ...
</nm:Nav_Context>

<nm:AIU rdf:ID="aiu_member">
    <nm:has_managerclass rdf:resource="#Activity"/>
    <nm:appears_in_context rdf:resource="#Activities"/>
    ...
</nm:AIU>

<nm:Manager_Nav_Class rdf:ID="Activity">
    <nm:Manager_Nav_Class.belongs_to_aiu rdf:resource="#aiu_activity"/>
    ...
</nm:Manager_Nav_Class>

<nm:Context_Rel rdf:ID="Context_Rel21">
    <nm:source_class rdf:resource="#Activity"/>
    <nm:target_class rdf:resource="#RGroup"/>
    <nm:target_context rdf:resource="#Groups"/>
    ...
</nm:Context_Rel>
  ...
```

**Figure 4 – Semantic representation of a piece of web application**

As we said in section 2, with the navigational ontology we want to provide extra information about web applications. For instance, it could be interesting to some kind of user to know which nodes (contents) have to be traversed to get a specific piece of information.

Taking as example the knowledge presented in the previous figure we can think about the following question: *From which nodes an Member user has to navigate to reach the content of the research group"?* Figure 5 shows the query pattern that represents this question.

```
  Query Pattern:{(target_context ?c_rel ?t) (Nav_Context.name ?t Group)
                 (source_class ?c_rel ?n_class)(belongs_to_aiu ?n_class ?aiu)
                 (appears_in_context ?aiu ?n_context)
                 (belongs_to_map ?n_context ?n_map)
                 (belongs_to_user ?n_map ?user)
                 (User.name ?user Member)}
  Must-Bind Variables List: (?n_context)
```

**Figure 5 – Example of a semantic query**

The previous figure shows the query formulated in OWL-QL. Although this query can be translated into an XML syntax [9], we rather prefer to present the query just as OWL-QL to clarify the example.

## *4. Conclusions and Further Work*

In this work we have argued that web engineering methodologies have to adapt to provide applications that deal with the new scenarios that are appearing in the next web generation. To achieve this goal, web engineering methodologies should enrich their implementations reusing the knowledge gathered in its own models. Making this knowledge available to others facilitate the interoperability between distributed and heterogeneous web systems. We have studied in what way these models are useful to provide the required interoperability mentioned above.

It is also studied how the developments done in the semantic web can help to achieve this new challenge and which languages seem to be suitable to represent this knowledge. Finally, it is presented a semantic web representation of the navigational metamodel and how a navigational schema can be queried trough the use of a semantic web query language.

As ontologies deal with consensus it would be interesting to associate different navigational ontologies or establish a web site ontology and then extend it with the peculiarities of each method. Web engineering methodologies will also have to change to provide web service definition to complete the interoperability scenario. In this direction a collaborative effort made by researchers at several organizations have end up in the OWL-S language. Then, users and software agents will be enabled to automatically discover, invoke, compose and monitor Web resources offering services under specified constraints.

## *5. References*

[1] Ceri S., Fraternali P., Bongio, "A. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites". *In WWW9*, Vol. 33 (1-6), pp 137-157. Computer Networks, 2000.

[2] De Troyer O. and Leune C. "WSDM: A user-centered design method for Web sites". In *Proc. of the 7th International World Wide Web Conference*, 1998.

[3] Fons J., Pelechado V., Albert M. and Pastor O. "Development of Web Applications from Web Enhanced Conceptual Schemas". *Proc. Of the International Conference on Conceptual Modelling, 22nd Edition*, ER'03, pp 232-245. Chicago, EE.UU, 13 - 16 October 2003..

[4] Gómez J., Cachero C., and Pastor O. "Extending a Conceptual Modeling Approach to Web Application Design". *Proc. Conference on Advanced Information Systems Engineering*, CAiSE'00, Springer- Verlag, LNCS 1789, pp. 79-93, 2000.

[5] Koch, N. and Wirsing, M. "Software Engineering for Adaptive Hypermedia Applications". In: *3rd Workshop on Adaptive Hypertext and Hypermedia*, 2001.

[6] Lima F. and Schwabe D. "Exploring Semantic Web Modeling Approaches for Web Application Design" *2nd International Workshop on Web Oriented Software Technology* (IWWOST'02)

[7] Muruguesan, S. and Desphande, Y., "Web Engineering. Software Engineering and Web Application Development". *Springer LNCS - Hot Topics*, 2001.

[8] Web Ontology Language Overview, W3C Recommendation 10 February 2004 , http://www.w3c.org/TR/owl-features/

[9] A proposing OWL query language (OWL-QL).Knowledge Systems Laboratory Stanford University, http://ksl.stanford.edu/projects/owl-ql/

[10] Pastor, O., Fons, J., Torres, V. and Pelechano, V. "Conceptual Modeling versus Semantic Web: the two sides of the same coin? www2004. workshop in semantic web.

[11] Schwabe D. and Rossi G., "An Object Oriented Approach to Web-Based Application Design", *Theory and Practice of Object Systems* 4(4), 1998. Wiley and Sons, New York, ISSN 1074-3224.

[12] Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web", *Scientific American Journal*, May 2001

[13] Object Management Group, "Unified Modling Language (UML). Specification Version 1.4 draft". http://www.omg.org, February 2001