Conceptual Modelling versus Semantic Web: the two sides of the same coin?

Oscar Pastor, Joan Fons, Victoria Torres, Vicente Pelechano
Politechnic University of Valencia
Department of Information Systems and Computation
{ opastor, jjfons, vtorres, pele } @dsic.upv.es

Abstract

A Web Application must have a precise semantics. In currently Web Engineering environments, this can be achieved either by specifying a Web Conceptual Schema, or by using a Semantic Web-language where the Web Application meaning is properly captured. In any case, the set of relevant conceptual primitives has to be properly represented. In this paper both approaches are studied as two different strategies to solve a common problem. The required conceptual primitives are presented, including those data and functional more conventional primitives, and those navigation and presentation more specific of Web Appls.

1. Introduction

Philosophers are facing for centuries the problem of how to represent properly reality. The existence or non existence of universals -concepts whose representation we can perceive- has been an issue from the platonic realism –that defend the existence of universals- till the more modern nominalism and conceptualism philosophical approaches –arguing that universal or concepts are just a linguistic mechanism to model real world phenomena-.

In some way, this philosophical discussion has been present in Software Engineering through Conceptual Modelling approaches. We could see Conceptual Modeling-based methods as a projection on Software Engineering of realism, in the sense that the only software components that are present in a final product are those that have a conceptual counterpart in the corresponding source conceptual schema. Agile Methods, Extreme Programming-based would constitute the nominalism approach, not accepting the need of having previous, pre-existing modelling constructs as a required basis for any software representation at the solution space.

Within the emergent Web Engineering community, model-based approaches are providing sound methods to deal with a precise Web Application Production Process, where the features associated to system structure, dynamics, functionality, navigation and presentation are properly managed (OOHDM [7], WebML[1], OOWS[2],…). Furthermore, how to go from the conceptual schema (system specification) to the implementation (final software product) is precisely stated by defining a set of mappings between conceptual primitives and their corresponding software representations (OlivaNova [6]). Some concrete tools are even already present in industry, giving some kind of automated support to this web-oriented software production process

According to these approaches, any Web Application is the result of systematically applying a set of transformation rules specified at a higher level of abstraction in a Web Conceptual Schema. If Web Applications would have been built from the beginning following these ideas, the semantic of any Web Application would be precisely characterized by its corresponding specification. Obviously this has not been the case.

Generally speaking, Web Applications Development in practice has been during the last years and ad-hoc, informal process, where modelling support has not been considered at all an essential approach to deal with the complexity of Web development. In consequence, we face a situation where a huge number of Web Applications are running in the Web, with a mostly unknown semantic structure, and all of them independent from each other.

But as we commented before, humans are insistent in trying to structure the world. The World Wide Web has not escaped to this human goal. If we want to communicate Web Applications providing efficient web services to exploit the advantages of the global web, the semantic of a Web site needs to be precisely known. Semantic Web languages are introduced to represent Web site modes. They play the role of the Conceptual Schema in the Model-based Web Development approaches provided in the context of the most advanced Web Engineering methods. Semantically tagged data start to become available: the Semantic Web technology is just here.

But it is very interesting to remark that Conceptual Modelling for Web Applications and Semantic Web related technologies are facing a common, well-known problem: to understand the world, by providing a clear

system specification. Conceptual Modelling selects a top-down strategy –from the model to the implementation- while current Semantic Web technology opts for a bottom-up approach –we have the software product: let's provide any kind of structured specification in a clear-enough language.

According to this idea, a basic aspect is to characterize the set of conceptual primitives required to model a Web Application. Either if we chose a top-down or a bottom-up approach, the required conceptual primitives should be the same. For representation purposes, different languages can be selected to specify them, but the important point is to describe precisely those conceptual constructs needed to characterize the structure of any Web Application. In this paper, we basically introduce such a set of conceptual constructs, independently of any particular language or conceptual modelling approach. Our final intention is to characterize the expressiveness that has to be provided by any particular solution, either coming from the Conceptual Modelling or the Semantic Web domain.

To accomplish our objective, after this introduction we present in the next sections the quoted set of conceptual primitives: in section 2 we introduce the conventional static and dynamic primitives, following an Object-Oriented Model to characterize the data and functional system architecture. In the section 3, the navigation and presentation conceptual primitives that complement the previous static and dynamic system views are presented. The work is ended with the conclusions and the corresponding references.

2. Conceptual Primitives related to Data and Functionality

Of course, a Web Application is still an application… What we mean by this is that beyond specific web-oriented aspects that we will face in the next section, a Web Application must be based on a precise Class Architecture –to characterize the static system's view- and a precise Functional Model –to characterize the dynamic system's view.

We chose an Object Oriented Model for Conceptual Modelling purposes because it has been proved in many previous works that the OO Model is especially appropriate for conceptual modelling purposes due to its proximity to human cognitive mechanisms. It seems to be a natural way of modelling to look at the world as a society of interacting objects, belonging to classes where data and functionality are formally specified. The question now is to fix what conceptual primitives need to be taken into account.

From the static point of view, the list of conceptual primitives is composed of:

- **classes**, and
- **relationships** between classes

The class specification includes the definition of **attributes** and **operations**. Every attribute has associated its type, its default value, whether it is a constant, variable or derived and if it accepts null values. For every operation, its arguments must be specified, together with a special label to distinguish new and destroy operations, and shared operations with other classes when this is the case.

The valid relationships between classes are those of **association** / **aggregation** and **generalization** / **specialization**. Association / aggregation are characterized by the binary cardinality (minimum and maximum) and by the constant or variable property of the established relationship. If the relationship is unidirectional, the induced part-of relation converts the association in aggregation.

Generalization/specialization conceptual primitives include the specification of roles as specialization mechanisms that gets activated only in periods of a given object live. The condition or operation that activates the role, and the condition or operation that deactivates it has to be specified.

Finally, **integrity constraints** allow specifying conditions that must hold in any valid state of an object. They are specified within the class scope as well-formed formulas built on attributes.

From the dynamic point of view, the list of conceptual primitives is composed of:

- **preconditions** of operations, to state what conditions must hold for activating an operation;
- **valuations** of operations, to state what is the change of state generated by the occurrence of an operation, in terms of new attribute values or object creation / destruction;
- **transaction** definition, which provide complex operations consisting of a set of operations belonging to the same class (if the transaction is local) or belonging to different classes (if global);
- **trigger** specification, to fix when an operation will be activated in an automated way, because a given condition is fulfilled.

These primitives have traditionally been present in a sound model-based software development process, and they need to be present in particular if we want to provide a method for developing Web Applications.

But the data and functional specification is not all. A Web Application needs to specify particular navigation and presentation characteristics, specific of Web environments.

3. Conceptual Primitives related to Navigation and Presentation

It is not an easy task to define navigation, as there is no general definition accepted by everybody. According to interesting discussions hold within previous IWWOST editions (International Workshop of Web-Oriented Software Technology, [3], [4], [5]), our position is that navigation implies the change of a conceptual node through the activation of a navigational link. This implies that what do we mean by conceptual node –interaction unit that provides access to relevant data and functionality for a given agent- and navigational link –reachability relationship between conceptual nodes to satisfy a given agent's goal- is basic for characterizing a navigational model.

For navigational purposes, we assume that any valid navigation must be accomplished by traversing a path that exists in the underlying class model. This means that we can navigate from one class to another if and only if there is specified a relationship between the involved classes.

The navigation specification must fit the features of particular agents. In consequence, the main navigation conceptual primitive is the **navigational map** that will be attached to any particular type of user. The navigational map represents the valid paths that any user of the corresponding type can go through.

The other primitives are hierarchically structured. Any navigational map is made up of:

- **navigational nodes**, that includes a set of *navigational contexts*, that are the basic user interaction units, containing a set of *navigational classes* and *navigational relationships*
- **navigational links**, which are binary relationships specifying a reachability relationship between two navigational nodes.

Any Web Conceptual Modelling or Semantic Web based approach has to provide the way to specify the required specific properties of both navigational classes and relationships. A **navigational class** includes the set of attributes and the set of operations that a user can access. These accessible properties must exist in the structural class diagram introduced in section 2. This allows us to define a navigational class as a view of a class, where the subset of visible and accessible class properties is specified. As not all of the class population has to be available, additionally filters on the class population can be defined associated to any navigational class.

Finally, a **navigational relationship** is defined as unidirectional, binary relationship that exists between two navigational classes of a given navigational context. They need to have a structural relationship counterpart in the associated Class Diagram. Depending on if the navigational relationship induces or not navigation, we have navigation relationships of two different types.

If the navigational relationship does not imply navigation, we are just adding more information to the basic user interaction unit that the navigational context is. If it implies navigation some more properties have to be specified:

- which is the target navigational context
- which is the attribute that will be used as "anchor" for activating the navigation

These are the most basic conceptual navigation primitives that must be provided. We also talked about presentation patterns. Now, we briefly introduce a set of conceptual presentation patterns, intended to complement the navigational view by specifying some presentation properties. These properties will also guide the user interaction provided by the Web Application.

The conceptual presentation patterns are basically:

- **information layout** (register, table, tree, master-detail, etc.)
- **ordering criteria** to indicate the chosen order to view the required information
- how to **group the visualization** of objects (page cardinality, access mode)

With them, how the user will "see" and interact with the information provided at any navigational step, is precisely specified.

4. Conceptual Primitives Representation: Conceptual Modelling vs. Semantic Web

Once established the set of conceptual primitives that must be captured to properly specify the requirements of a Web Application, we must follow an approach to tackle with the software development process.

From a Conceptual Modelling point of view, the representation of those concepts must be defined before system implementation. At a higher level of abstraction, any conceptual modelling approach must provide with graphical notations to represent the system requirements by using those conceptual primitives. Usually these primitives are organized into different diagrams. In an OO paradigm, we use a Class Diagram to represent the structural system properties (see Figure 1) and a Functional Model to represent the dynamics (see Figure 2).

Figure 1.- Static Primitives: Class Diagram

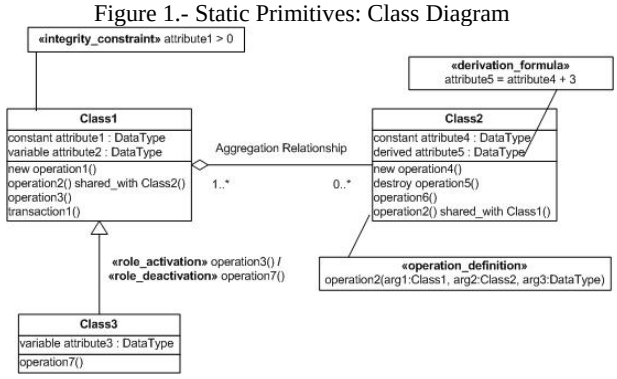
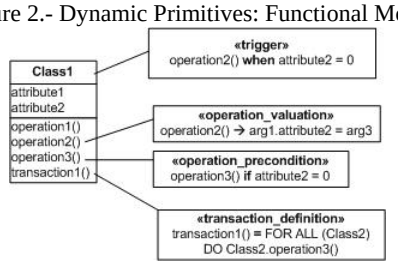


Figure 2.- Dynamic Primitives: Functional Model



In addition, the navigational primitives are represented by means of a Navigational Model (see Figure 3) and the presentation primitives within a Presentation Model (see Figure 4).

Figure 3.- Navigational Primitives: Navigational Model

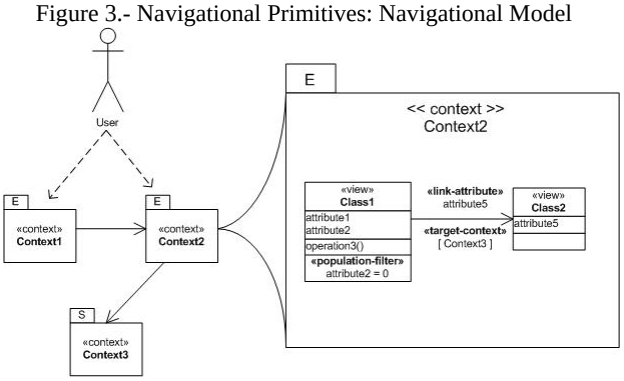
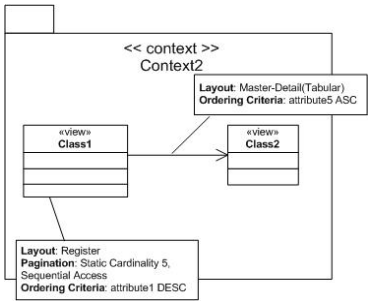


Figure 4.- Presentation Primitives: Presentation Model



From a Semantic Web point of view, the representation of those primitives must be placed at implementation level. Development strategies oriented to apply the semantic web use implementation semantic languages (such as RDF [8]) to take into account those conceptual primitives. The specification of these properties is usually specified in schema files (such as RDF-Schema) defining the valid implementation structures. A piece of the schema that defines some static and some navigation primitives is shown in Figure 5.

Figure 5.- Static and Navigation Primitives

```
<!-- Static Primitives -->
...
<owl:Class rdf:ID="Class">
  <rdfs:label>Class</rdfs:label>
  ...
</owl:Class>

<owl:DatatypeProperty rdf:ID="className">
  <rdfs:label>Class name</rdfs:label>
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

...

<owl:Class rdf:ID="Attribute">
  <rdfs:label>Attribute</rdfs:label>
  ...
</owl:Class>

<owl:DatatypeProperty rdf:ID="attributeName">
  <rdfs:label>Attribute name</rdfs:label>
  <rdfs:domain rdf:resource="#Attribute"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="attributeType">
  <rdfs:label>Attribute type</rdfs:label>
  <rdfs:domain rdf:resource="#Attribute"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="ConstantAttribute">
  <rdfs:label>Constant Attribute</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Attribute"/>
</owl:Class>
...

<owl:Class rdf:ID="Aggregation">
  <rdfs:label>Aggregation Relationship</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Relationship"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#composite"/>
       <owl:cardinality
           rdf:datatype="&xsd;nonNegativeInteger">1
       </owl:cardinality>
    </owl:Restriction>
    ...
  </rdfs:subClassOf>
...
</owl:Class>
...
<!-- Navigational Primitives -->
...
<owl:Class rdf:ID="User">
  <rdfs:label>User</rdfs:label>
</owl:Class>

<owl:Class rdf:ID="NavigationalMap">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#NavigationalUser"/>
     <owl:cardinality
       rdf:datatype="&xsd;nonNegativeInteger">1
     </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
...
<rdf:ObjectProperty rdf:ID="context">
  <rdfs:domain rdf:resource="#NavigationalMap"/>
  <rdfs:range  rdf:resource="#Context"/>
</ObjectProperty>
...
<owl:Class rdf:ID="E_Context">
  <rdfs:label>Exploration context</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Context"/>
</owl:Class>
...



<owl:ObjectProperty rdf:ID="navigationalClass">
  <rdfs:label>Class View</rdfs:label>
  <rdfs:domain rdf:resource="#Context"/>
  <rdfs:range rdf:resource="#ClassView"/>
</owl:ObjectProperty>
...
```

```
  <owl:Class rdf:ID="ClassView">
    <rdfs:label>Class View</rdfs:label>
  ...
  </owl:Class>
  ...
 <owl:ObjectProperty rdf:ID="navigationalRelationship">
    <rdfs:label>Navigational Relationship</rdfs:label>
    <rdfs:domain rdf:resource="#ClassView"/>
    ...
 </owl:ObjectProperty>
  ...
```

Figure 6 shows the representation of some static primitives and Figure 7 shows the representation of some navigation primitives.

Figure 6.- Static Primitives (in use)

```
<!DOCTYPE owl [
 <!ENTITY cm "http://oomethod.dsic.upv.es/onto/04/cm#">
  ...
]>

<rdf:RDF
  xmlns:rdf = "&rdf;"
  xmlns:rdfs = "&rdfs;"
  xmlns:cm = "&cm;"
  ...
>

 <cm:Class rdf:about="&cm;Class1">
  <cm:className>Class1</cm:className>
  <cm:attribute>
    <rdf:List>
        <rdf:first>
      <cm:ConstantAttribute rdf:about="&cm;attribute1">
        <cm:attributeName>attribute1</cm:attributeName>
        <cm:dataType>integer</cm:dataType>
          </cm:ConstantAttribute>
        </rdf:first>
        <rdf:rest>
         <rdf:List>
          <rdf:first>
      <cm:VariableAttribute rdf:about="&cm;attribute2">
      <cm:attributeName>attribute2</cm:attributeName>
             <cm:dataType>integer</cm:dataType>
            </cm:VariableAttribute>
          </rdf:first>
          <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
         </rdf:List>
        </rdf:rest>
    </rdf:List>
  </cm:attribute>

  <cm:operation>
    <rdf:List>
        <rdf:first>
     <cm:NewOperation rdf:about="http&cm;operation1">
      <cm:operationName>operation1</cm:operationName>
            <cm:operationArgument>
             <rdf:List>
              ...
             </rdf:List>
       </cm:operationArgument>
      </cm:NewOperation>
        </rdf:first>
        <rdf:rest>
        ...
        </rdf:rest>
    </rdf:List>
  </cm:operation>

  <cm:IntegrityConstraint>
         <cm:constraintFormula>attribute1 gt 0
        </cm:constraintFormula>
  </cm:integrityConstraint>

 </cm:Class>
```

```
 ...
 <cm:Aggregation
   rdf:about="&cm;AggregationRelationship">
      <cm:composite rdf:resource="&cm;Class1">
      <cm:component rdf:resource="&cm;Class2"/>
   ...
 </cm:Aggregation>
 ...
```

Figure 7.- Navigational Primitives (in use)

```
...
<cm:User rdf:about="&cm;User">User
...
</cm:User>

  <cm:NavigationalMap>
     <cm:navigationalUser rdf:resource="&cm;User"/>
     <cm:context>
        <rdf:List>
          <rdf:first>
             <cm:context rdf:resource="&cm;Context1"/>
          </rdf:first>
          ...
        </rdf:List>
     </cm:context>
     <cm:navigationalLink>
     ...
     </cm:navigationalLink>
   </cm:NavigationalMap>

  <cm:E_Context rdf:about="&cm;Context1">
  ...
  </cm:E_Context>

  <cm:E_Context rdf:about="&cm;Context2">
    <cm:navigationClass>
      <rdf:List>
        <rdf:first>
    <cm:NavigationalClass rdf:about="&cm;Class1View1">
      <cm:navigationalAttribute>
       <rdf:List>
        <rdf:first>
          <cm:Attribute rdf:resource="&cm#attribute1"/>
        </rdf:first>
        <rdf:rest>
        ...
        </rdf:List>
      </cm:navigationalAttribute>
     </cm:NavigationalClass>
        </rdf:first>
    ...
    </cm:navigationalClass>

    <cm:navigationalRelationship>
     <cm:NavigationalRelationship
              rdf:about="&cm;NavRelationship1">
      <cm:nrSourceClass rdf:resource="&cm;Class1View1">
      <cm:nrTargetClass rdf:resource="&cm;Class2View1">
      </cm:NavigationalRelationship>
    </cm:navigationalRelationship>
  </cm:E_Context>
  ...
  <cm:NavigationalLink>
   <cm:fromContext rdf:resource="&cm;Context1"/>
   <cm:toContext rdf:resource="&cm;Context2"/>
  </cm:NavigationalLink>

  <cm:NavigationalLink>
   <cm:fromContext rdf:resource="&cm;Context2"/>
   <cm:toContext rdf:resource="&cm;Context3"/>
  </cm:NavigationalLink>

  ...
```

In this way, we have a different representation but a common set of basic concepts. Having a fix set of conceptual primitives, it is feasible to define a set of mappings between conceptual primitives and their

corresponding software representations, making possible the implementation of Web Conceptual Model Compilers.

In this environment, a fruitful strategy could be based on taking the best of these two approaches by:

- having complete conceptual models of web applications, at a higher conceptual (problem space) level, and
- implementing final applications by applying a set of systematic translation rules from those conceptual primitives into web semantic concepts representation, at the solution space level.

5. Conclusions

The main goal of the emerging Web Engineering discipline is to develop correct Web Applications, where structure, functionality, navigation and user interaction have to be properly represented. To make this possible, any Web Application has to provide a precise semantic associated to it. Only if such a precise meaning is given, it makes sense to provide web services whose structure and functionality is clearly specified, and that can be accessed and used by different agents.

This semantics can be provided in a top-down way, by defining a Web Conceptual Schema where all the relevant modelling components are specified. The resulting Web software product is the corresponding representation of the Conceptual Model at the solution space level.

Alternatively, a bottom-up strategy can be used. In this case, a Semantic Web-based language (i.e. RDF) allows to specify those relevant conceptual constructs that characterize the meaning of the corresponding Web Application. This specification makes possible the connection of the application to any external potential agent. Web site models can be represented in this way by Semantic Web languages. The available Semantic Web infrastructure is immediately applicable for the Web Engineering field, thus making the processing of Web site models effective.

In any case, the set of conceptual primitives required to fix the semantics of a Web Application must be clearly defined. In this paper, this set is introduced. They are structured in data and functional conceptual primitives, and more web-oriented navigational and presentation conceptual primitives. The final intention is to fix the required expressiveness for any Web Conceptual Modelling strategy, or any Semantic Web-based ontology language. According to that, we could conclude that Conceptual Modelling and Semantic Web are really the two sides of the same coin: the coin required to develop correct Web Applications.

References

[1] S. Ceri, P. Fraternalli and A. Bongio, "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites", *WWW'00*, Elsevier, Amsterdam, The Nederlands, May, 2000, pp. 135-157.

[2] J. Fons, V. Pelechano, M. Albert and O. Pastor, "Development of Web Applications from Web Enhanced Conceptual Schemas". *ER'2003*, Springer-Verlag, Volume 2813, Chicago, USA, October, 2003, pp. 232-245

[3] International Workshop on Web-Oriented Software Technology, First Edition. June, 2001. Valencia, Spain. http://www.dsic.upv.es/~west/iwwost01/

[4] International Workshop on Web-Oriented Software Technology, Second Edition. June, 2002 Malaga, Spain. http://www.dsic.upv.es/~west/iwwost02/

[5] International Workshop on Web-Oriented Software Technology, Third Edition. July, 2003 Malaga, Spain. http://www.dsic.upv.es/~west/iwwost03/

[6] OlivaNova Model Execution System. CARE Tecnologies S.A. http://www.care-t.com/

[7] D. Schwabe, G. Rossi and S. Barbosa, "Systematic Hypermedia Design with OOHDM", *ACM Conference on Hypertext*, USA, 1996

[8] W3C Recommendation 10 February 2004. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/