STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

AN ONTOLOGICAL APPROACH TO WEB APPLICATION DESIGN USING W2000 METHODOLOGY

ANNA LISA GUIDO, ROBERTO PAIANO, AND ANDREA PANDURINO

Abstract. Applications users claim ever more quality in the software product; that does not mean only an improved performance but also a better usability or a more efficient interaction paradigm; in other words, a better "user experience". The designer focus is not on the system but on the user and his applications. This point of view creates new requirements that force the designer to use structured approaches for improving the quality of the design and of the final web application as well. If we consider the web and the applications on this channel (where the "user experience" aspects have an important role), it's possible to evaluate the weight of the new features in the software engineering process.
The new application features don't exactly match design methodologies, which must evolve in order to follow the new requirements. Many methodologies are introducing new primitives and structures that did not exist before. The well-known UML describing language (and its MOF metamodel) is evolving; the Jim Connallen's Web Application Extensions for UML are an example of the need to update the language to these application trends. Supporting the UML goodness, in this paper we describe our experiences in the use of Web Application (WA) design methodology called W2000 (which uses UML) and in the use of ontology language to represent the same methodology.

1. Introduction & Background

During the standard application development, the designer can use well-known traditional software engineering techniques that focus on the system aspects, such as data and transactions; on the contrary, the new user requests put the stress on the hypermedia aspects, such as information, navigation and presentation aspects; the combination of these factors improves the user experience and produces a better quality product. However linear the modeling technique and however legible the language used may be, the need to describe in sufficient details the multiple static and dynamic aspects of the application is a highly complex task. The

Received by the editors: September 19, 2005.

1991 *Mathematics Subject Classification.* [**Information Systems**: hypertext navigation, interfaces, decision support etc.](68U, 35).

1998 *CR Categories and Descriptors.* D.2.0 [**Software**]: Subtopic – *Software Engineering-General* .

designer has to manage the individual aspects and predict the inevitable interactions among them; in many cases, a navigation choice that is valid for one device or context turns out to be completely wrong or unworkable in another; thus, the complexity increases and the methodological instruments to manage it seem to be insufficient.
The complexity to design a modern application is very high; in particular, we need to combine the multi-user and multi-device features with the typical transactional aspects. To obtain high quality applications, a structured methodological approach to model all the user experience aspects is needed.
The design methodologies used to model traditional application are not sufficient to meet the new application requirements and to improve the quality of the final product; therefore, new design methodologies, in particular in the web application domain, are born such as OOHDM [1] ,WSDM [9] , WebML [13] , HDM [3] . The traditional design tool is also updated to include new primitives and construct.
UML (de facto standard in model representations and independent of methodology) must evolve to include the new features: UML [12] defines a formal mechanism to extend its semantics. The "Extension Mechanism" allows to customize and extend UML with the use of stereotypes, constraints, tag definition and tagged values. (The same mechanism, defined in UML 1.x, is also valid in UML 2.0 [11]).
In this paper, we describe our experience in the use of web application design methodology (called W2000 [6] [5]) with UML and highlight the benefits and the problems connected with the large use of stereotypes. Considering that stereotype is the only possible choice to describe the primitives not directly implemented into the UML paradigm, our studies could be applied to all the methodologies that make a large use of stereotypes.
Many problems arise from MOF (core of UML approach). In this paper, we highlight these problems and try to use the ontological language as an alternative.

2. Our experience with MOF approach: W2000 methodology

W2000 methodology uses UML and, thus, has a MOF representation (proposed by OMG [16]) made up of three packages: Information, Navigation and Publishing Model. Each package has sub-packages where a class diagram represents W2000 methodology primitives and relationships among them. Packages, sub-packages and OCL constraints [10] (OCL is an external language used to express all the methodology constraints not directly supported by UML) represent, in MOF terminology, the design methodology *metamodel*, that is a collection of concepts and their relationships; in our case it is a collection of W2000 methodology primitives. The specific WA *model* is produced according to this metamodel. Although MOF W2000 methodology approach [7] is formally correct, there are some practical problems (related to UML1.x version and also present in UML2.0), due to dependence of UML on MOF.
The main problems that we highlight are:

Metamodel and semantics: In MOF approach there are a lot of primitives not directly supported by UML and thus all primitives are represented by stereotypes so metamodel semantic coincides with stereotypes semantic. Furthermore, the lack of semantics creates confusion to the unskilled designer during the practical applications of modeling concepts. The explicit presence of semantics helps the designer to understand how the modeling concepts should be used.
Another problem strictly connected to *semantics concerns semantic relationships among classes*: we can observe that MOF allows to use only two relationships aggregation and association while in W2000 metamodel it is necessary to define specific methodology relationships with its relative semantics.
Relationships among classes: another problem is that relationships among classes are lost in the transition from metamodel to model. Supposing that in the metamodel we have a relationship among classes: when we define the model, relationships must be redefined because they are not inherited by the model. This problem could be solved creating intermediate classes to represent the relationships; the disadvantage of this solution is that it will make the model unreadable for the large number of intermediate classes.
Finally, in MOF approach, if an attribute is the same for two different concepts it is defined once for each class (each attribute is strictly connected to each class). This MOF limit creates confusion letting designers think that each attribute has its semantic but it is not true.
Model Flexibility: another problem is the flexibility, that is the possibility to enrich the model with new primitives or to add new characteristics to the primitives already defined. The solution proposed by UML (both 1.x and 2.0) is to enrich the UML metamodel with Extension Mechanism but this mechanism require a good knowledge of UML and then may require a lot of time compromising the evolution of the WA design methodologies. Another problem related to the language evolution concerns the *unique name assumption* principle: in UML approach different words must refer to different objects. In order to meet WA evolution, it is often necessary to define new version of concepts (defined before) and to use the same name. The unique name assumption makes it impossible. The UML and MOF do not support the *dynamic classification of classes*. It is possible that, when metamodel is extended to include the methodology evolution, two classes must be replaced by their intersection: the instance of the new class contains both previous classes. This is not possible in UML, since every instance can be only the instance of a class and not the instance of two classes at the same time.
Standard description of the model: it is important to have a machine readable description of the model. In MOF approach we use XMI (OMG standard) as a model representation language (but we are free to use any XML description). There are different formats according to the graphic editors that produce XMI but a model description must be understandable in an easy and univocal way by software agent and preferably should be a W3C standard.

An external language to represent constraints: in MOF approach there is an external language, OCL, to describe the methodology constraints. OCL is hard to understand by designer who are unskilled both in W2000 methodology and in the OCL language. If the designer is also an OCL language expert, the model obtained will be formally correct but difficult to understand by developers.

3. Ontological approach to web application design methodology representation

The main problem related to MOF approach is the *lack of semantics* that brings to represent all the W2000 primitives through the same UML primitives: stereotypes.
Another problem is that WA methodologies must follow the evolution of WA requirements and the methodology representation language must meet this evolution too: in MOF approach it is often hard to represent the new WA requirements and the solution is to add new stereotypes that will be increasing the semantic problem. Finally, the language used to represent the methodology must be easy to learn and read for designers and guidelines must be provided during the whole design process; instead, the necessity to learn an external language such as OCL may be a problem.
Considering these comments, the use of UML as a representation language, in our experience, forces methodology to adapt itself to the paradigm imposed by UML, and thus semantics turns flat and it is hard to read and understand the model. *We need to adapt the representation language to methodology and not methodology to representation language as in UML approach.* To solve this problem, we need a *new language* easy to use, more flexible and expressive than UML, allowing to represent directly methodology primitives and helping the designer in his/her task through a better semantics. We explored several alternatives but the use of semantic language able to describe, in general, a domain of knowledge seems more flexible and compliant with our goal which consists in representing WA methodology in a more meaningful way. The language used is OWL [14], the *ontological language*: a text language without graphical notation. Our use of ontology is quite different from the semantic web which is the traditional one. We use OWL to represent both WA design methodology and the WA model obtained through this methodology. We choose to use ontology as a language to represent a methodology starting from its definition: "*ontology is a formal, explicit specification of a shared conceptualization*" (Gruber in 1993 [2]), that is ontology is an abstraction of some concepts made through the definition of its peculiar characteristics. Defining a *metamodel* as a set of concepts and rules necessary to specify a model in the domain of interest, we can state that: "*A valid metamodel is an ontology but not all the ontologies are expressly modeled as metamodel*" [15]. From these considerations we state that *it would be possible to use the ontological languages to express a metamodel and to obtain from it a model.*

3.1. **OWL language in a nutshell.** Before explaining the use of OWL in our approach, it's necessary to understand its main concepts. OWL primitives are:

- *Classes*: allow the abstraction of some concepts. Each class has a set of *properties* (each one for specific concept characteristics). A class would be composed by subclasses.
- *Properties*: There are two types of properties: *DataType* specific to each class and *ObjectProperty* used to create a link between classes. Object-Property has both domains: class (to which the property is connected) and range (the possible values of the property). In each class we can indicate "restrictions" that define constraints.
- *Individuals* are objects that have the characteristics defined by classes and properties. Both classes and properties may have individuals.

3.2. **The architecture of our approach.** MOF approach, proposed by OMG is based on a 4-level architecture. It allows to define a language for the methodology representation and to use this language for model definition. The use of a 4-level architecture is a good choice because it allows to separate different levels of abstraction so we use it but with few changes. Fig. 1 shows the 4-level architecture compared with our ontological approach.

In MOF approach, M3 level is the level where the MOF language, that is the

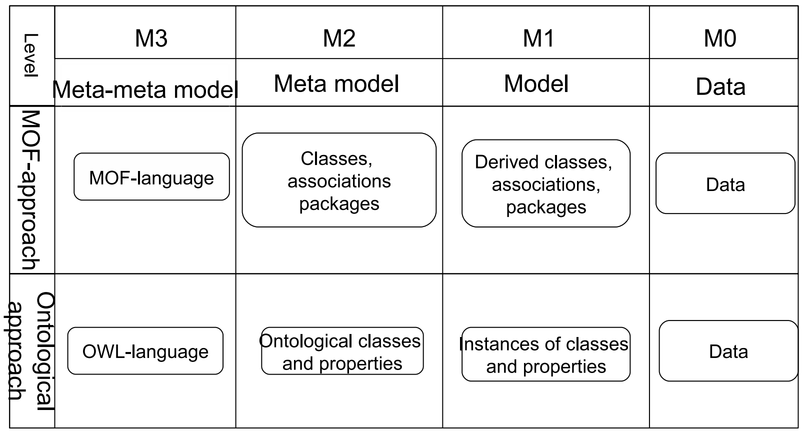
Level
M3
M2
M1
M0
Meta-meta model
Meta model
Model
Data
MOF-approach
MOF-language
Classes, associations packages
Derived classes, associations, packages
Data
Ontological approach
OWL-language
Ontological classes and properties
Instances of classes and properties
Data

FIGURE 1. MOF and Ontological approaches compared

abstract language used to describe MOF metamodel, is defined. MOF is Object Oriented and strictly connected to UML: UML notation is used to express MOF metamodel. In the ontological approach we use in M3 level OWL language instead of MOF language.

In the M2 level we define, both in MOF and ontological approaches, a metamodel, that is the abstract language that allows the definition of the model in the M1 level. In MOF approach the metamodel definition is made up of classes, association, packages and OCL constraints aimed to representing the particular methodology. In the ontological approach we define the metamodel through ontological classes, that allow to define the methodology primitives, and ontological properties (DataType and Object Property) that allow to give other details about each methodology primitive. Object Property, using domain and range definition, represents the semantic network of the methodology. Restriction on properties allows to define methodology rules without using any external language.
M1 level is the level where the designer, using metamodel (guidelines for methodology), designs the specific application. Finally M0 level represents data of a specific model.
From a technological point of view, in our ontological approach, we use both in the M2 and M1 level, Protégé[4], an open source ontology editor developed by Stanford University. Protégé manages separately the metamodel (classes and properties) and the model (the instances); thus, it's possible to define the metamodel (M2 level) and model (M1 level) layer. To create the ontological metamodel we followed the guidelines defined by the Stanford University researchers [8].

4. Our experience with ontological approach: W2000 methodology

To understand the benefits of our ontological approach we have to explain our experience with WA design methodology W2000: we explain how problems presented in section 3 can solved.
First of all, it's possible to give a *semantic meaning* to each W2000 primitive directly through OWL without the UML stereotypes in order to improve the designer comprehension of the model. In figure 2 there is the OWL code of the relationship between Entity and Component(Entity and Component are W2000 methodology primitives): the ObjectProperty "madeOfComponent". It has a restriction that forces each entity to have at least one component. There is also the relationship between "Component" and "Entity"("belongsToEntity") defined as inverse of "madeOfComponent": this allows to read in a bi-directional way the concepts linked to the Object Property.

```
<owl:ObjectProperty rdf:about="#madeOfComponent">
    <owl:inverseOf>
        <owl:ObjectProperty
        rdf:ID="belongsToEntity"/>
    </owl:inverseOf>
```

Figure 2. OWL code: "inverse of"

The ontological approach allows to define an attribute once and to use it in different primitives: the domain of this attribute will be the union of all classes

that represent the primitives in which the attribute will be used.
Concerning the *model flexibility*, the architecture proposed allows to add new primitives to metamodel inserting new classes and its properties into the metamodel or new properties to classes. The changes to metamodel are very fast and do not require technical competences about any language: it will be available immediately in the model and also the existing model will be updated automatically with new elements. The OWL language allows to define intersection or union between two classes (dynamic classification of classes) and the equivalence between classes (the unique name assumption is not valid in OWL).
The *relationships among classes* in the transition from the metamodel to the model is naturally taken. The OWL metamodel is immediately ready to create a WA model in the OWL language: to create a model starting from metamodel is sufficient to add instances of classes and properties defined in the metamodel. In our approach the property instances allow to tie together classes without adding, as in MOF approach, classes that represent the relationship between classes defined in the metamodel.
The problem of *standard description of the model* is solved with the OWL use (recommended by W3C); also with the same language it's possible to represent both the metamodel and the model.
The ontological approach avoids the use of *external language to represent constraints* that are described directly in the OWL language using its restriction. For example, to express that an Entity is made up of at least one Component, a restriction on the property "madeOfComponent" is defined (Fig. 3).

```
<owl:Restriction>
   <owl:minCardinality rdf:
datatype="http://www.w3.org/2001/XMLSchema#int>1
   </owl:minCardinality>
<owl:onProperty>
<owl:ObjectProperty rdf:about="#madeOfComponent"/>
```

FIGURE 3. OWL code Property restriction

This approach helps the designer to understand the methodology and then to make a design following all the rules presented in the methodology *without learning an external language.* The model obtained is also understandable by developers who do not have knowledge of the OCL language.

5. Conclusion and future work

Considering that WA design methodologies have peculiar characteristics, different from standard applications, and that the traditional design approaches are not sufficient to take into account all these new features, we highlight the limits of MOF. Then we introduce a new ontology approach that uses the OWL language.

Our approach applied to W2000 methodology is more flexible and allows methodology to adapt itself to WA evolution. The model obtained from the metamodel is more clear, effective and complete than the one obtained with MOF approach and makes the designer work easier. In fact, a better semantics provides clear guidelines for the designer.
At present our work is focused on the development of an editor able to design WA through W2000 methodology and to obtain an OWL description of the model. We are also planning to extend metamodel to include a good description of the operation in an ontological way.

References

[1] D. Schwabe, G. Rossi, S. D. J. Barbosa: *Systematic Hypermedia Application Design with OOHDM* Proc. ACM Conf. Hypertext '96 - Mar 1996 - ACM Press.

[2] Dieter Fensel Ontologies: *A silver Bullet for Knowledge Management and Electronic Commerce* Second Edition, Revised and Extended. Foreword by Michael L.Brodie. Springer-Verlag Berlin Heidelberg 2004 ISBN 3-540-00302-9.

[3] F. Garzotto, P. Paolini, D. Schwabe: *HDM - A Model for the Design of Hypertext Applications*, in Proceedings ACM Hypertext '91, S. Antonio (TX, USA), ACM Press, Dec. 1991.

[4] http://protege.stanford.edu/.

[5] L. Baresi, F. Garzotto, and P. Paolini, *Extending UML for Modeling Web Applications*, Proceedings of 34th Annual Hawaii International Conference on System Sciences (HICSS-34). IEEE Computer Society, 2001.

[6] L. Baresi, F. Garzotto, Paolo Paolini, *From Web Sites to Web Applications: New Issues for Conceptual Modeling*, Proceedings WWW Conceptual Modeling Conference, Salt Lake City, October, 2000.

[7] L.Baresi, F. Garzotto, M.Maritati: *W2000 as a MOF Metamodel* In Proceedings of The 6th World Multiconference on Systemics, Cybernetics and Informatics - Web Engineering track. Orlando (USA), July 2002.

[8] Natalya F. Noy and Deborah L. McGuinnes Stanford University, Stanford, CA, 94305 *Ontology Development 101: A Guide to Creating Your First Ontology*

[9] O.M.F. De Troyer, C.J. Leune, *WSDM: a user centred design method for Web sites*, Proceeding of WWW7 Conference, April 14-18, Brisbane, Australia, 1998.

[10] OMG, ad/97-08-08, *Object Constraint Language Specification*, version 1.1, 1 September 1997.

[11] OMG. *UML 2.0 Superstructure Specification. Version 2.0*, September, 8 2003.

[12] OMG. *Unified Modeling Language Specification.* Version 1.4, September 2001, http://www.omg.org/uml.

[13] S. Ceri, P. Fraternali, A. Bongio: *Web Modeling Language (WebML): a modeling language for designing Web sites*, Proc. Int. Conf. WWW9, Amsterdam, May 5 2000.

[14] *OWL Web Ontology language Reference* W3C Recommendation 10 February 2004.

[15] www.metamodel.com : *What are differences between a vocabulary, a taxonomy, a thesaurus, an ontology and a metamodel.*

[16] www.omg.org.

Dipartimento Ingegneria dell'Innovazione Universitá di Lecce Via per Arnesano, 73100 Lecce, Italy Tel: +39 0832 297229 Fax: : +39 0832 297279
E-mail address: annalisa.guido@unile.it, roberto.paiano@unile.it, andrea.pandurino@unile.it