Conceptual design of web application families: the BWW approach

Roberto Paiano, Andrea Pandurino, Anna Lisa Guido
University of Lecce
Via per Arnesano, 73100 Lecce (LE)
Tel. +390832297229

Abstract: A good analysis of the application domain is the most complex and crucial phase in information system design, regardless of its dimensions. Many well-known methodologies exist for the following development phase, but to design a specific application domain many approaches follow one another: from UML to goal-oriented approach. All these became critical when the final application will be a large web application where it is important to also model the user experience. In this case it is not possible to isolate a well-structured conceptual model of the domain where the computer technology concepts are not taken in consideration but the modeling concepts map directly to the domain concepts.
We explain here our experience of a research industrial project founded by the Italian Government about environmental monitoring. In this work, we adopt the UML-Like approach in order to obtain the domain specific conceptual model. The target is not a single application but a family of applications. After this phase, we adapt in an opportune way the BWW (Bunge-Wand-Weber) ontology to design the application domain and we evaluate which domain representation is more objective and useful for the following web information system design and development phases.

1. Introduction and background

Web application became in the last years complex application where several actor, several type of information, several technologies, several roles are involved. In order to link together these different key aspects it is very important to provide a methodological approach to helps in the design and later in the development of web application. We focus on methodologies that consider the information, navigation and transactional aspects typical of web application with several user on several devices. Very often when methodologies are applied they result strictly related to the designer experience about the domain knowledge. When the application domain is very large and several application (not only a single application) can be obtained from the design, it is important to formalize in some way the domain knowledge in order to not lies the application of the methodology only on the experience of the designer but in order to provide a common knowledge on which designer can work. In a few words it is important to take into account the conceptual design. We can see in Fig. 1 three different layers of analysis: the first, the domain layer, allows us to acquire the rigth knowledge of the entire application domain (named conceptual model); in the second, the web application layer, several design methodologies such as IDM, P-IDM and AWARE provide an engineeristic approach to the well-known web application paradigm (in order to solve the information and navigation open issues); and finally in the application layer, implementation details are taken into consideration.

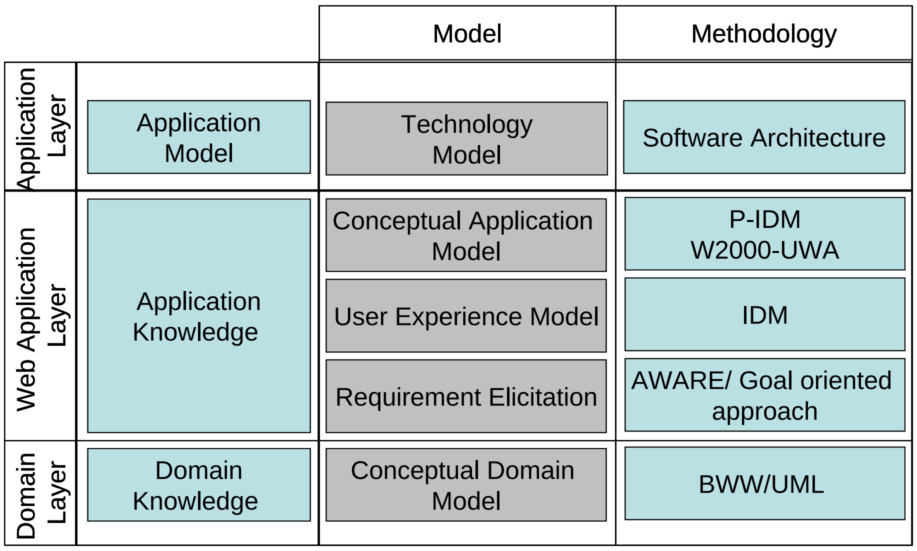

Fig. 1: Layers of analysis

In the modeling of an application family, surely a fundamental role is performed by the domain knowledge. The domain design must represent all the system complexity and, at the same time, the domain layer must be of high level of abstraction and must not take into consideration technology aspects. The conceptual model can be interpreted as the "broader view of information systems requirements engineering" [DIES00]. The conceptual model goal is to obtain a detailed domain analysis independent from the system functionalities, the processes, the information structures. Moreover the language used to describe the model must be near to the domain concepts. In practice, the conceptual model allows us *to acquire* the application domain knowledge *and to store* it using a well-defined grammar. The conceptual model will be the starting point from which to obtain the following analysis phase. Due to the conceptual model importance, it is essential to consider an efficient approach with several key aspects:

- *Complete*: it covers all the aspects of the particular application domain;
- *Objective*: it must represent the reality in its wholeness without focus on a particular aspect of the reality as it can be seen by a particular actor involved in a specific aspect of it. The conceptual model could be the model where all the actors can recognize its own particular part of the application domain.
- *Abstract*: at this modeling level it is not important to go down to the detail, but rather to remain at a high abstraction level. This allows a simple and efficient information exchange between the conceptual model and the models obtained in the next phases for instance the process model, the user experience model etc. [CHEN99]
- *Independent from the implementation technology*: to be tied up to the implementation technology results, in some cases, in designing the application domain in terms of the selected implementation technology. As a consequence, this can put little emphasis on the critical application domain aspects (because they are, for example, hard to implement) or, contrarily, can emphasize irrelevant aspects. Both factors can result in a product of poor quality.
- *Simple*: Surely a model can not be simple if the application domain is complex, so it is important to think to a methodology and to a relative notation in order to manage the application domain complexity and to represent exactly the complexity of the application domain without add another complexity level.

In order to meet all these key aspects we focus on two different approaches to the conceptual domain model:

- *The use of classical techniques within software engineering*. Due to the domain dimension and complexity, it is too hard to use a full-compliant UML approach because it force to constraint often hard to take in consideration in order to provide the key aspects considered above. For this

reason we decided to adopt an UML-like approach that is to adopt the main concepts of Object Oriented customized for our purpose.
- *The use of the formal ontology*. We adapt, through a methodological approach that we propose, the classification of the concepts proposed by BWW in order to represent the domain concepts and its relationships with a well-known semantics.

In this paper we present in the section 2 an overview of the BWW approach that we propose for the conceptual model and in the section 3 we speak about the environmental domain as a case study of our research work. In the section 3.1 we discuss the conceptual model of the application domain an UML-Like approach and in the section 3.2 we discuss about the conceptual model of the same application domain with the proposed BWW approach. In the section 4 we compare the two approaches provided and finally we provide conclusion and future works.

2. The BWW approach

To achieve an objective conceptual model, not tied up to a particular technology and whose modeling primitives allow us to provide for a suitable semantics to the model, we explored two possible directions tied up to the application of the Information Systems design philosophy.
The first is proposed by Chisholm [CHI92][CHI96][LEW97] immediately rejected because it doesn't provide an objective model: the approach is based on the *common sense*, that is it leaves a lot of scope for free interpretation from who read the model without providing norms and interpretation rules.
We follow in our research work another approach: the BWW approach [WEB97]. BWW provides an effective and suitable classification of the concepts that allows from one side to not flatten the whole modeling in a few concepts (classes, relationships and so on) and from the other side to assure a good level of objectivity and the correct semantics: each part of the domain model under analysis can be associated, without leaving scope for free interpretations, with the classification of the concepts implicit in BWW. In this way the model obtained is objective: the concept classification helps to represent the application domain without focus on a particular point of view but only categorize the application domain concept in a well specific classification. Furthermore it is easy to realize and easy to use for those who will deal with the following phases of analysis and implementation. The model obtained is not tied up to any implementation technology.
Obviously we adopt the BWW approach not in the philosophy point of view but we adapt BWW concepts classification in order to define the conceptual model in a specific application domain. In this way it is possible to define the domain model direct using domain concepts.
In order to define the domain model in the environmental domain (target of our research work) we defined the BWW ontological meta-model that reflects the concepts classification proposed by BWW and, on the base of this classification, we realized the environmental domain model. The language used to define both the meta-model and the model is OWL [W3C04]. The domain model so obtained has well-defined semantics: the concepts of the environmental domain have easily been classified among those proposed by BWW. It is clear that the realized meta-model is also valid for the definition of the conceptual modeling of other application domains.

3. Case study: enviromental domain

We consider a very complex application domain: the environmental protection (understood as habitat of all the organisms and as organic structures of systems and subsystems). Environmental protection has evolved considerably in recent years. In the sixties, the public administration main goal was the control laid down by law. Today great importance is given to knowledge acquisition of the factors that heavily affect the environment quality; this choice is determined by the high growth rate of the population and by the evolution of the productive system that exercise pressure on the environment.

Today, monitoring activity and environmental control is not only developed by the government public administrations (Municipality, Provinces, Regions) but also by a number of associations and organizations and, above all, from several agencies for the protection of the environment in national and international territory.
The great number of stakeholders and the necessity to acquire the knowledge about the quality of the environment and soil, force a coherent and reliable informative exchange. To achieve this goal, organization and institution nets were born in order to facilitate the collaboration for applying a common environmental policy. The efforts to collect and to deliver environmental knowledge in the Italian and European areas do not match at regional institutional level, because the technological infrastructures do not support informative exchange. In this context, the research project GENESIS-D[1][MAIN05]. The project goal is the creation of a "framework" for the design and the development of software systems for environmental management at regional and/or sub-regional level. The software systems obtained starting from the framework Genesis-D, will be web application that will allow the exchange of environmental information among different institutional subjects such as Regions, Provinces, Municipalities, ARPA (Regional Agencies for the Environmental Protection), etc.
The problems related to the application domain model are of two types:

- The application domain is very complex so it is hard to cover all the complexity and to take into consideration all the application domain aspects without guidelines;
- The applications obtained starting from the framework will be web-based applications so the design focus is not only on the data element (represented in the form of object or relational entity) but also on the user experience (end-user and his perception of the information). This constraint force to consider the user experience design that is to take under control simultaneously of all the aspects of the application domain, to acquire and to document a deepened domain knowledge in order to highlight the differences and the shared points among different stakeholders.

3.1 Conceptual modelling through UML-like approach

At the conceptual modelling abstraction level it seem not correct to use a full-compliant Object Oriented approach; we define a class diagram but we can not speak about “object” as they are defined in the UML definition. In UML an object is an entity with a well defined boundary and identity that encapsulate state and behaviour [OMG]. Because in this phase we are not able to define state and behaviour we define the generic concept of “Entity” that may contain some attributes in order to characterize it. We define a static view of the overall system through the class diagram where we represent Entity and relationship between them. As it regards the dynamic view of the system, it could be defined using other UML diagram but this introduce to another important problem: the overall view of the application domain is fragmented, so a change in one diagram may make obsolete other diagrams[BOOCH].
We named this approach where we define a class diagram as relationships between Entity and without dynamic view of the system the “UML-like” approach.
In the UML-like approach the modeling process is made up though several iterations: each iteration has the goal to refine the analysis and therefore to describe the application domain in a more and more precise way. Particularly, the first step aims to provide a description of the information entities and the relationships among them. These entity are refined in order to describe the overall application domain. After this phase entities are refined (through the definition of the attributes, of their type and of the methods), in order to define a class diagram that is the model of the application domain. This process starts from the "generalization" in which the designer is forced to aggregate

[1] The leading company is Edinform SpA in collaboration with the University of Lecce and the Polytechnic of Milan

them in terms of shared attributes. At the end of this process abstract objects are specialized obtaining a full UML-compliant design useful for the implementation of the application. In a few word this process bring to a class diagram. These classes will be invoked in order to develop a single specific application. Obviously the application will be realized with an OO technology.
Taking into consideration a scratch of the environmental domain (Fig 2) we consider several objects such as: alteration biological variety, climatic change, consumption of soils, soil degradation, eutrophication, pollution, wastes production and disposal, radiation, noise, and so on. Each of these phenomena is characterized by a variation of some environmental parameters which may or may not be homogeneous. In the UML-like approach the designer describes the phenomena creating the *FIA* class (Fact of Environmental Interest) and all the parameters are grouped in the *IIP* class (Indicator, Index, Parameter) strictly related to the *Metrics* class. It is clear that in the class diagram FIA, IIP and Metrics are Abstract Classes that must be specialized when they are instanced. Using an analogous procedure, other abstract classes are identified as *Objects*, *Subjects* and *Structures* (*OSS*) to which each *FIA* makes reference. The class *OSS* contains the Subjects (physical or juridical person that can be interested or involved in facts and environmental phenomena), the *Objects* (any object or territorial structure inside which the characteristic processes of the human social lifetime are based and are developed) and the Structures. The class *OSS* is directly connected with the class *FIA* and it will be decomposed in *OST and SOI.*

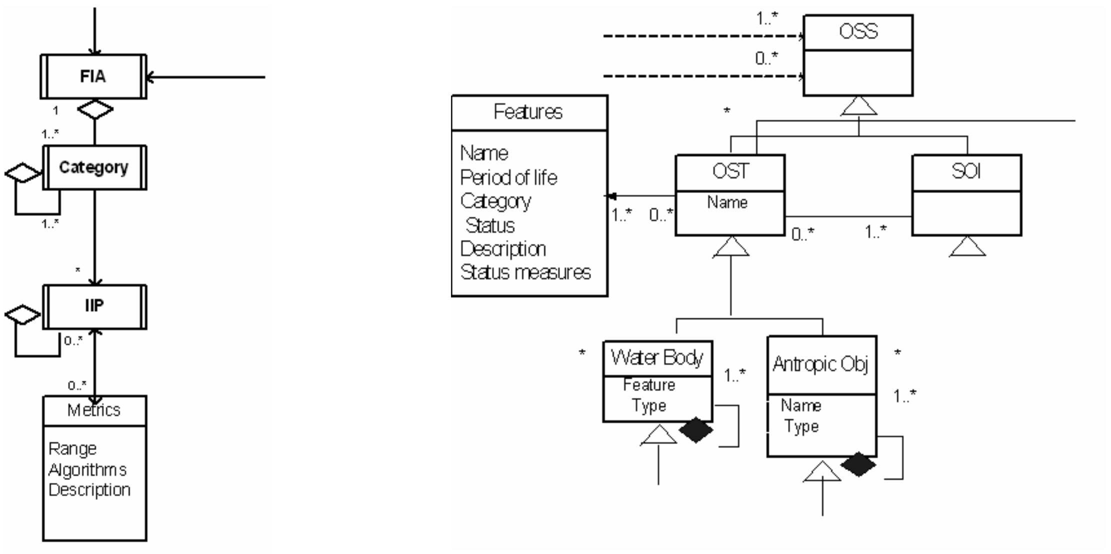

Fig. 2. A scratch of UML-Like model

3.2 Conceptual modelling through BWW

In the conceptual modelling made by BWW approach, the language used to define the model is OWL and the classification of concepts that we adopt is defined by BWW ontology. To be precise, we adapt the BWW ontology to our goal and we follow several steps in order to obtain a complete conceptual model of the domain in a well-defined language (OWL) (see fig. 3). We explain here the steps followed to define a domain-specific model using BWW concepts in the environmental domain. Obviously these steps, here applied to the design of the environmental domain, can be followed in order to model any application domain.
Having acquired the necessary knowledge of the application domain, in Step 1 it is possible to individualize the *Things* [WAND95] distinguishing them, as far as possible, from the *Classes*[WEB96](step 2). In the BWW approach the difference between *Things* and *Classes* is very subtle: the *Things* are well defined in reality in examination; they are tangible objects and therefore they are easily identified. *Classes* group together poorly defined *Things* that have some shared

properties (*mutual property*)[WAND95] (the definition of mutual property is in step 3). We, for instance, consider the *Water Basin*: it has the same properties as the *Water Body*. The properties shared between *Water Basin* and *Water Body* define a class (*Hydrographic object*): *Water basin* and *Water Body* are things and they have both mutual and intrinsic property. In the conceptual model phase it is also possible to know some mutual properties but to not know what *things* belong to this class, because their individualization requires a degree of specialization and detail in the model that would make it considerably complex and it would stray from the desired degree of abstraction.

Step	Step description
1. Analysis of things	Highlight things of the domain
2. Analysis of classes	Highlight classes of the domain
3.Mutual property	Select mutual property of each class
4. intrinsic property	Select intrinsic property of things and classes
5. Characterization of mutual/intrinsic property	Select peculiar characteristics of mutual/intrinsic property
6.Events	Define what it happens in a particular application domain
7.System	Select different parts of the same conceptual modeling

Fig.3 Methodological steps to define a conceptual model using BWW

The next step (Step 4) is to identify the related *intrinsic properties*. At this point the designer presents an initial description of the application domain; surely it is not complete but it succeeds in giving an idea of the application domain complexity.
The following step (Step 5) consists of identifying the peculiar characteristics of each mutual and intrinsic property. Particularly, each mutual / intrinsic property is tied up to both human and natural laws. This characterization of the properties introduced by BWW, has led us to reflect them in the environmental domain and to reveal *some concepts that in the UML-like approach had not been considered*. The territorial competences of the *Public Corporation* are, for instance, defined by *decrees*. The *decrees* result, therefore, in a characterization of the Intrinsic Property *territorial competences* and, particularly they are *Human Law*: this means that territorial competencies are constraints by the Human Law decree. BWW attributes allow us to take into consideration the *correct semantics* of a concept (*decree*). In the UML-like approach, because decree is an instance of a DIA (Provision Intervention Action) it will be related to this class, so it is not highlighted that decree constraint the territorial competencies. We decide to not show the concept of *decree* (Fig. 4) because it should has the same characteristics of the DIA therefore cannot be connected with the *Territorial Competencies* with the correct semantic (Fig. 4). In the BWW approach the problem is solved thanks to the “BWW Law” concept. The concept of decree that naturally fits the concept of Law is hard to represent using UML notation, in fact a study in this field say that there is not a correspondence between the concept of BWW law and any kind of representation in UML[HOPDAL02].

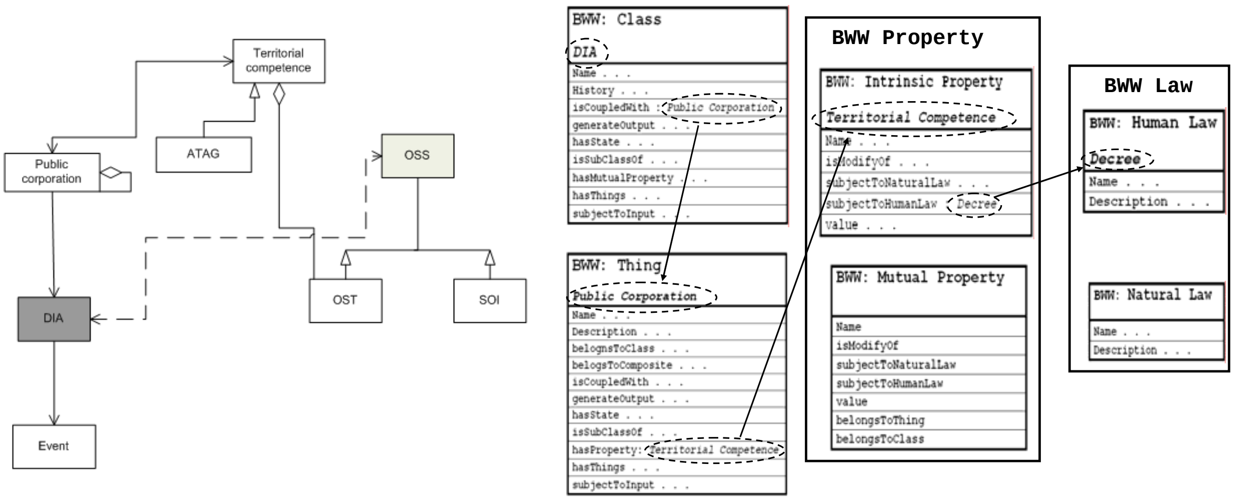

Fig. 4:The two approaches compared

The following step (step 6) is the identification of the *Events*. The concept of events allows us to define what happens in a particular application domain. BWW attributes well defined semantics to the *event*: the events are in partnership with a *thing* or class and, when they occur, change a well defined property of the Thing or of the *class* to which they are associated. The possibility to define the events induces the designer to understand thoroughly the functioning of the application domain and to identify the consequences that an event can provoke (which property the event modifies). When an event modifies the value of different properties of a same *thing/class*, the *thing/class* changes state. All the states make a *history*. In BWW the concept of history is defined to follow changes of state so the changes of state are associated to *opportune semantics*.
At this point we have a complete conceptual model of the application domain in examination and in order to organize the obtained conceptual model, it is possible (but not obligatory) to define the *systems (step 7)*. BWW defines the systems as the joining of things intending that the things belonging to the system influence the properties of each other: this is a guideline that facilitates the designer to define the subsystems and therefore to structure the model appropriately.
The conceptual model obtained, as in the case of the conceptual modelling obtained from the UML-like approach, will be the starting point for the design and implementation of specific web application: at this point, because we are not lies in any way to the object oriented approach, we are free to design and develop the web application taking the knowledge stored in the model and without any constraint about the technology to adopt.

4. Comparison between the two approaches

The UML-like approach allows us to describe a domain using an incremental method: the designer improves the analysis step by step; this option is very useful when, in an application domain such as the environmental domain, knowledge is very fragmented. In our case study, the project started with an analysis of environmental models (National, European and American models) that describe the environmental domain in general and then it has been possible to describe in detail all the information needed to produce the class diagram.
The developer can directly use the output obtained through the UML standard notation to create the framework.
The problem strictly related to this approach is that the output will be an Object Oriented model that will implement the typical constructs of this paradigm (generalization and specialization) but, although correct, it moves away from the objective reality.
The process is strongly dependent on the experience of the designer who must analyze the application domain and must be skilled at OO modeling.
The final output of UML-like approach, even if complete, is strongly tied up to the implementation logic. Furthermore, the output (the model and its notation) complex that is add to the complexity of

the application domain another level of complexity due to the fact that the semantic in the UML-like approach is flatten in the concept of entity and relationships between them.
The concept of entity is used also to model very different concepts; for instance, a *FIA* is determined by varying a particular indicator. In the UML-like modeling, the *FIA* and the indicator are both modelled as classes but the *FIA* is not a pure class but a set of values (not a-priori defined) that the indicator can assume at the time: a *FIA* is registered when some value of an indicator change. Using the BWW approach the *FIA* has been modelled as a history of the value of indicators; thus, the concept of *FIA* has the correct semantics. This example makes it clear that the application domain model (particularly the environmental domain), using an UML-like approach, compresses the different semantics into the same concept of entity (or of relationship among entity). These make not only the creation of the model difficult but also its understanding from he who must use the conceptual modeling for the following phases of analysis and implementation. The model created using UML-like approach, gives freedom to understand the model because it is not tied up to a particular semantics, therefore the model is not objective.
With BWW approach, it is possible to represent some relationships that in the UML-like vision it could be possible to characterize only by using other diagrams different from class diagram (in this case we obtain a fragmented design not useful in the conceptual modelling phase); for instance, an *OSS* creates a *FIA* that, in turn, is created when the variation of an indicator produces an alarm. The alarm condition is an input event to the *public corporation* that in response produces a *DIA* (Provision Intervention Action), a regulation that the *OSS* must respect. According with the *DIA*, *OSS* finishes the alarm and, hence, it produces another change to the indicator and so another *FIA*. In the UML-like approach it has not been possible to define this important aspect: it can describe only a relationship among *OSS*, *FIA* and *DIA* and the cause-effect relationship is not clear. We have to consider that the cause-effect relationship (in UML-like approach) could take place adding the specific methods to the objects but this is not compliant with the need to define a conceptual model of the domain without going in detail. On the contrary in the BWW approach the class *OSS* is related to the thing indicator and produces the event *FIA*. In the event *FIA* it is possible to see which properties of the thing indicator are modified by the event. The indicator produces the alarm condition (modelled through the property generateOutput of things), that is the Event source in the thing *public corporation*. The thing *public corporation*, related with *DIA*, issues the rules of *DIA*, modelled with an *Event*. At this point the *DIA*, engages the *OSS* and forces the *OSS* to conform to the new indicator value. To model this is very simple with the BWW approach thanks to the concept of “*coupled*” (that allows the joining of two classes: class modifies the value of one or more properties of the other), to the concept of *Event* and to the concept of input and output in relationship with the concept of thing. It is important that the relationship’s name (*is coupled with*, *generate output* and so on) is intrinsic in BWW approach so the right semantics is given directly by the BWW approach. Last, but not least, thanks to the BWW approach the domain model is directly expressed using well defined and categorized domain concept through real world concept such as thing, classes, law and so on where the domain concept is easy to map.

5. Conclusions and future trends

The BWW approach allows the provision of the right semantics and therefore it expresses all the domain details directly using the domain concepts. The model is objective thanks to the classification of the concepts provided by BWW and it is not tied to a specific implementation technology.
The conceptual model has been immediately understood in the correct way both from the stakeholders (who, helped by the classification of the concepts, have identified their own activities and goals), and from the buyer (who has succeeded in understanding all application details without having specific skills in BWW ontology). The modeling realized through the BWW approach meets, therefore, all the requirements described above.

The UML-like approach appears particularly effective because, being a lot close to an implementation technology, allows us to directly reach the realization of a family of applications in a particular application domain. Nevertheless, the objectivity and the simplicity of the BWW approach encourage the use of this approach for the conceptual modeling of application domains of great dimensions.
Using the BWW approach, the effort and the elapsed time to define the application domain conceptual model, are smaller than those needed in the UML-like approach. In our research work the knowledge acquisition for the conceptual model of the environmental domain required 3 months and it has been realized both through interviews with the various stakeholders and through the study of several documentation pages. The team, composed of 3 units, after having acquired the domain knowledge, was split: 2 units used the UML-like approach while the third unit made the conceptual model with the BWW approach. The UML-like conceptual model required 3 months while the one using the BWW approach required just 1 month.
The case study presented in this research work where we compare the UML-like approach with the BWW approach to the conceptual design is very complex, so we think that consideration made for this case study are also valid for other case study that we planning to conclude as soon as possible.
In order to hold under control the typical aspects of web applications and therefore to manage the user experience, we design two web application starting from the conceptual modelling of the family of web application. Our efforts are focused on the definition of a methodology to obtain an IDM [PER05] design starting from the domain specific modeling of the whole application domain. To achieve this goal, it appears simpler to start from the conceptual model realized according to the BWW approach characterized by an elevated degree of objectivity that results independently from the implementation rather than to start from the UML-like model.
As future work we plain to design and implement an editor in order to help in the application of the methodology that we define.

References

[BOOCH99] Booch, G.; Rumbaugh, J.; Jacobson, I. “The unified Modeling Language User Guide” Addison Wesley 3rd Printing Febraury 1999.

[CHEN99] Chen,P.P.; Thalheim, B;Wong,L. Y.:Future Directions of Conceptual Modeling. Lecture Notes in Computer Science 1565, Springer, 1999.

[CHIS96]Chisholm,R. M: A Realistic Theory of Categories – An Essay on Ontology, *Cambridge University Press*,1996.

[CHIS92]Chisholm,R. M.:In Language, Truth, and Ontology Kluwer. *Academic Publishers, Dordrecht*, 1992.

[DIES00]Dieste O.; Juristo N; Moreno A.M.; Pazos J.; Sierra A: Conceptual Modelling in Software Engineering and Knowledge Engineering: Concepts, Techniques and Trends, *Handbook of Software Engineering and Knowledge Engineering*, World Scientific Publishing Company, 2000.

[OPDAL02]Opdal, L. A; Henderson-Sellers, B.: Ontological Evaluation of the UML using the Bunge-Wand-Weber Model. Softw Syst Model, 43-47 Digital Object Identifier 10.1007/s1027-002-0003-9

[LEW97]Lewis, E. H.: Chisholm’s Ontology of Things The Philosophy of Roderick M. Chisholm, *Lewis E. Hahn* 1997.

[MAIN05]Mainetti, L.; Perrone, V.: A UML Extension for Designing Usable User Experiences in Web Applications, *Proceedings of V International Workshop on Web Oriented Software Technologies* June 2005,Porto, Portugal

[OMG] Object Management Group, UML 2.0 Superstructure Specification OMG Adopted Specification

[PER05] Perrone, V.; Bolchini, D; Paolini, P.; A stakeholders centered approach for conceptual modeling of communication-intensive applications. Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information pp: 25 – 33, ISBN:1-59593-175-9 2005, September 21, 23 2005, Coventry, United Kingdom

[W3C04]W3C : OWL Web Ontology language Reference, *W3C Recommendation, 2004*.

[WAND95]Wand, Y.;Weber, R.: On the deep structure of information systems. *Information Systems Journal, 5* 1995

[WEB97]Weber, R. :Ontological Foundations of Information Systems, Coopers and Lybrand Accounting Research Methodology. Monograph No. 4. Melbourne, 1997.

[WEB96]Weber, R.; Zhang, Y.: An analytical evaluation of NIAM's grammar for conceptual schema diagrams. *Information Systems Journal, 6: 147–170*, 1996