Engineering Web Applications Using Roles

Gustavo Rossi, Jocelyne Nanard, Marc Nanard

Engineering Web Applications using Roles

Gustavo Rossi[1], Jocelyne Nanard[2], and Marc Nanard[2]

[1] Facultad de Informática, Universidad Nacional de La Plata, Argentina and Conicet[1]
gustavo@sol.info.unlp.edu.ar
[2] LIRMM, CNRS/Univ. Montpellier, 161 rue Ada, F34392 Montpellier ceddex 4, France
[jnanard, mnanard]@lirmm.fr

Abstract. In this paper we discuss the use of the role concept in the Web Engineering life cycle. We claim that introducing roles in the modeling and design armory of existing web engineering methods will improve their expression power and help to solve design problems that appear frequently in Web Applications. We first survey those situations in which objects must change their properties according to the context in which they are used; next we introduce the role concept and discuss how it has been used so far in the software engineering community. Using existing methods (like UWE and OOHDM) as an example we show how to introduce roles during the Web Engineering process. We compare our work with other similar approaches and then outline some further work we are pursuing.

1 Introduction

The increasing use of the Web as a software platform together with the advance of technology has given raise to a completely new generation of Web Applications; these applications allow ubiquitous access from fixed and mobile devices, provide personalized features to individuals, support complex business processes and workflows, etc. The Web engineering community has already discussed and hopefully solve many of the design problems arising from this complexity [5], [10], [21]. In this paper we focus on one important issue that has been so far ignored (or at most only partially addressed): the evolution of objects features during the application's execution, for example when they interact with other objects.

Usually, application objects must be designed to provide different services (data, behavior) according to the context in which they are accessed. In an academic application, when a person gets enrolled it is treated as a student; the same person may then become a teaching assistant in a course, or an employee of the university: the corresponding software object(s) will presumably exhibit different behaviors. In a typical e-commerce Web software, the same product object will provide different services according to the application object (or user) interacting with it, e.g. in the context of a stock application, when being part of an order, when accessed as a recommended item, when treated as a gift to an employee, etc. Notice that in these two

[1] This work has been partially funded by the Universidad Abierta Interamericana project: Conceptual Modeling of Web Applications

examples, application objects (persons, products) vary their features when interacting, collaborating or being accessed by other objects, as if they were dynamically changing their base class. This variation occurs not only when we send specific messages to those objects but also when we navigate them, even in simple Web applications. The examples above shows only a small sub-set of the different kinds of situations in which semantic flexibility is needed for modeling.

Some of the situations above can be modeled using combinations of well-known object-oriented primitives or patterns (See for example [20]); moreover, we can surely use programming language features to solve them, such as Java interfaces. However, it is important to deal with these concerns in different levels of abstraction and in particular, we need a clear way to express this kind of variability in a higher level, while modeling the application. In this paper we argue that we can solve those modeling problems in an elegant and clear way, by introducing the role concept.

Informally, a role is the set of features, which an object can assume in a particular context (or when participating in a certain relationship); we thus say that the object is playing that role. For example, a person object might play the role of a student, a teaching assistant, etc. A product may be playing the role of a gift or a recommendation. When the person is playing the role of a student it "acquires" some new attributes and services while still being a person; when she plays the role of a teacher other features are necessary. The teacher might even play the role of a tutor when interacting with a particular student and so on.

Surprisingly, the role concept has not been used so far in the hypermedia community (interesting exceptions are IUHM [14] and [1]) and thus has been also ignored in the Web Engineering community.

In this paper we discuss why roles should be used as first-class citizens in Web design methods, and propose some simple ways of introducing them in some well-known design approaches, like UWE [10] and OOHDM [21]. The contributions of this paper are three fold:

- From a pedagogic point of view, we aim to make the Web Engineering community aware of the role modeling principle usefulness; we explain how to deal with different design concerns using roles.
- We show that the introduction of the role concept not only improves existing methods but also unify some existing primitives in a simple but powerful formalism;
- Finally, while comparing our approach with others using roles we clearly show when roles should be used and when other design structures or patterns are preferred.

The rest of the paper is organized as follows. We first review the role concept and how it has been used in the software engineering field. Next, we focus on the Web Engineering process and discuss why roles are useful in this context. Then, we show how to introduce roles in existing Web applications design methods; in particular we outline how to use roles to explicitly represent the use of Web patterns. Finally we compare our work with other similar approaches and present some further research we are pursuing.

2 The role concept in software engineering

As stated in [25], the role concept has been introduced early in software engineering but, unfortunately, from different points of view (See for example [15]). In this section, we point out the fundamental definitions of the role concept, which are important for its use in Web engineering. For the sake of clarity we distinguish two perspectives: conceptual modeling and object-oriented design. This distinction is useful to emphasize the intentional aspect associated to the use of roles from more technical aspects, thus allowing us to extract the main points to introduce in Web engineering design methods.

2.1 Roles in conceptual modeling

Fundamental for the relationships between roles and objects is the distinction between *natural types* and *roles*, originally introduced by Sowa [24].

A natural type is a semantically rigid and non-founded type insofar as an entity that has the type cannot stop being of the type without loosing its identity and does not depend on any collaboration. As an example, consider the Type *Person*: a person exists (even in software) independently of any relationship with other types; it is not possible to end being a person without losing the object's identity.

On the contrary, a role type is a founded and semantically non-rigid type insofar as it characterizes an entity by some role it plays in relationship to another entity or other entities, and if left, does not give up identity of entities. Thereby a role type characterizes the dynamic state of an entity when it is involved in some collaboration with others. For example *Student* is a role type; for a person to be a student it must be enrolled in a course (or in a college), thus it depends on collaboration with other entity. A person can end being a student without loosing its identity.

As said before, when an object plays a role it acts as if it has changed its class, incorporating the properties belonging to the role type. As objects might be involved in many relationships, an object may play several roles at the same time. Besides, a role type can be "assigned" to different natural types or even to other role types.

Introducing roles in conceptual modeling as first class entities allows us to clearly identify and separate those object' properties such as attributes and services that correspond to its belonging to a natural type (*Person, Product, Course*) from those corresponding to the roles played by the object (*Student, Employee, Gift*, etc).

However, conceptual modeling approaches have not incorporated this concept completely. For example, in UML [28], roles have been mainly introduced to serve three different purposes; first, they label association ends (similar to entity relationship models), thus illustrating the conceptualization of roles as named places in a relationship. Less popular, though more related with the definitions in [24] is their use to indicate slots in collaborations and as dynamic classes. Steinman has recently shown [26] some problems of these two interpretations, and has proposed some changes to the meta-model of UML [29], incorporating roles as core structural elements following the ideas expressed above.

2.2 Roles in object-oriented design

Roles have been considered from two points of view in object-oriented design: as an abstraction mechanism in object-oriented programs and as a means to describe design patterns and thereby to provide better support to composite patterns and framework description.

In [11], the authors detail the interest of making explicit those class attribute subsets associated to roles, in order to formally specify operations such as composition, generalization, point of view, etc. While the definition of roles is similar to the one proposed in [24], the focus is put on extending an object-oriented language with role-based constructs, for example improving the language class browser (in Smalltalk for example) with roles. Different techniques for implementing roles are presented in [2], for instance the use of interfaces [27], like in Java. A set of strategies for introducing roles in object-oriented analysis is described in [12].

The above discussion on the difficulties for expressing properties of objects and classes when involved in some relationship lead Riehle [16] to propose representing design patterns using role diagrams rather than class diagrams, and even to define the Role pattern to reify roles as objects for decoupling them from the role they play in some collaborations [3].

2.3 Summary and Notations

So far, the role concept has not been used in Web engineering design methods even those, which rely on object-orientation and UML-like modeling. As we will show, it is important for Web engineering methods to incorporate roles as first-class abstraction concepts at the same level of classes.

We can summarize the usage of the role concept for Web engineering methods in the following way:

- Roles as abstraction mechanisms: we distinguish the notion of *Role* as characterizing the dynamic state of an object in some collaboration context and *Role type* as representing the abstract properties of one role. We argue for using explicitly role types and not only roles as places in a collaboration.
- Roles as indications of context: when different objects are involved in some relationships leading to specific constraints, roles are used for expressing those constraints.

Furthermore, roles can be used to help separate some abstract design properties from their implementation using pure class-based models.

In the following we will use two different notations to introduce roles in Web engineering methods: the notation proposed in [26] (See Figure 1) in which roles are expressed using UML (in a similar way as the UML notation for interfaces). We will also use the notation of [18] as shown in Figure 2. The rationale for choosing each one of them is fully explained in Section 4.

In the next section we further discuss the need of using roles in Web engineering methods.

Figure 1. UML-like notation for roles

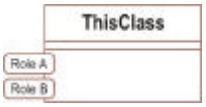

Figure 2. Riehle´s notation for roles

3 Using Roles in the Web Engineering Process

Web application development methods like UWE [10], OOHDM [21], OO-H [7] and WebML [5] partition the design space in (at least) two different activities, conceptual and navigational design, providing modeling primitives for each activity (for the sake of conciseness we ignore subtle differences between these approaches). There is a consensus in our community that during conceptual design, the overall application functionality is specified; application objects, behaviors and relationships are specified using well-known notations such as UML.

Being Web applications a particular kind of hypermedia software, navigational design aims at defining which hypermedia objects the user will perceive and how he will navigate them; hypermedia primitives such as nodes, links and indexes are used in this activity. Nodes and links are generally defined in terms of application objects and relationships, using some mapping mechanism.

3.1 Why roles are needed?

These methodologies have not incorporated the concept of role in their design primitives, and thus problems associated with objects (nodes) behaving differently when playing different roles, e.g. when relating with other objects (nodes), are generally solved in an ad-hoc way both in conceptual and navigational models. For example, there is no easy way to express the fact that a node will show different information according to the path, which has been followed to reach it (the OOHDM *navigational context* primitive, as shown later, solves only a particular case of this situation).

3.2 Where roles are needed?

From the above, we can derive some conclusions about the usefulness of role-based modeling in the development process. As discussed in Section 2, role types can be introduced in the object-oriented design process to indicate different sets of behaviors when application objects play different roles. In the following sub-sections we refine

the discussion considering the different design dimensions of Web Engineering methods.

3.2.1 Conceptual design

During conceptual design, role types can be used in a straightforward way, without changing usual Web design methods' heuristics to build the conceptual schema. Simply, we enhance our modeling armory with the role concept (See Section 4), i.e. when we identify that an application object will exhibit different behaviors when participating in different interactions we define the corresponding role types and their associated role behaviors. These role types might be eventually mapped to different classes of nodes (or roles) in the navigational model

3.2.2 Navigational design

Introducing roles in the navigational design activity requires a slightly different view. As mentioned in Section 2, an object cannot play a role in isolation: an object plays a role in relationship with other objects and it might play different roles according to the objects with which it interacts. Being hypermedia a discipline in which relationships (materialized for example as links) constitute such an important feature, it is clear that the role concept can be applied in a simple, intuitive way to allow nodes to exhibit different features (such as displayed information or outgoing links) when accessed with different links, i.e. when they are related with different nodes.

Let us suppose, in an e-commerce site, a node class *Product* built from conceptual class *Product* (for the sake of simplicity, we ignore here possible roles defined in the conceptual level). By analyzing the different incoming link types we can decide whether we want the product to show different attributes or behaviors. For example when the product is accessed from an index of similar products we might want to provide a link to the next product in the index (this is called set-based navigation in OOHDM and is implemented with navigational contexts as discussed in Section 4). At the opposite, when we navigate to the product from an order in which the product is included, we might want to disallow further navigation. Finally, when the Product is a recommended item, for example navigated from the home page, we might want to add some additional comments. More generally, we could define different role types for each incoming link type of a node class.

Using the role abstraction does not pose implementation problem; implementing role-based diagrams in state-of-the-art Web architectural languages and tools is straightforward using plain object-oriented techniques and applying either the Role pattern [3], variants of the Decorator pattern [6] or other programming techniques [12], [27].

4 Putting Roles to Work in Web Engineering Methods

The role abstraction mechanism complements existing object-oriented concepts mainly the concept of a class. The sole introduction of this concept in the modeling

armory gives us the possibility to think about applications in a higher-level way; however, the introduction of the role concept in existing methods has to be done in a seamless way, in order to maintain the corresponding notation coherent. In this section, we show how to improve some existing methods with roles. We have chosen two object-oriented methods for keeping the discussion simple; however the discussion below can be easily applied to other methods such as OO-H [7] and WebML [5] in a straightforward way. As introducing a new concept in an existing methodology involves much more than changing a notation, the sub-sections below are just the starting point for a much wider discussion in the Web Engineering community.

4.1 Introducing roles in UWE

The UWE methodology [10] is an object-oriented approach based on the standard UML, i.e. the notation and diagrams in UWE are restricted to those provided by UML. The UWE meta-model elements are the basis for the UWE notation, defined as a light weighted UML profile, because it uses UML extension mechanisms, such as stereotypes, tagged values and OCL constraints [29]. The advantage of this approach is that UWE can benefit from all tools that support UML.

While the conceptual model in UWE uses plain UML primitives, the UWE profile introduces some specific modeling elements for navigation and presentation. In particular, the navigation space model is represented by a stereotyped class diagram, including the classes of those objects, which can be visited during navigation. The <<navigation class>> and the <<navigation link>> stereotypes are used to model nodes and links. Relationships with conceptual classes are expressed using the UML Object Constraint Language (OCL) [29].

Other stereotypes such as <<index>> and <<guided tour>> are used for constructing the navigation structure model as a refinement of the navigation space model. An index for example allows direct access to instances of a navigation class. This is modeled in UWE by a composite object, which contains an arbitrary number of index items. Each index item is in turn an object, which has a name that identifies the instance and owns a link to an instance of a navigation class. Any index is a member of some index class, which is stereotyped by «index» with a corresponding icon.

One way to introduce roles in UWE is to use the notation in [26] and shown in Figure 1 which, as explained in Section 2, re-elaborates the role concept in UML. Roles can thus be introduced in the conceptual, navigation space and navigation structure models. The problem with this approach is that it is not a light weighted extension of UML (since it modifies the UML meta-model) so it does not fit well with the UWE philosophy. Another possibility would be to define a new stereotype <<role>> and use the Role pattern [3] to decouple roles from the base classes. Role classes will use the stereotype <<role>> thus simplifying the diagram. For the sake of illustrating the impact of using roles in UWE we will use a third and simpler approach: UML interfaces. Interfaces allow partitioning the operations of a class in groups (for example corresponding to the different roles played by objects of that class).

In Figure 3, we show part of an UWE navigation structure model taken from [10] and improved with roles. In the diagram, papers can be accessed from different indexes (Accepted, All Papers by ID, etc); in each case, a different role (interface) is de-

fined providing additional information (links to the next article in the list for example).

The benefits of using roles in UWE are twofold: first we can use roles in both the conceptual and navigational model keeping the notation UML-compliant; second, the use of roles in the navigation model allows to improve the method, as expressed in Section 3, and in particular introducing the OOHDM concept of navigational contexts not present yet in UWE as discussed in Section 4.1.

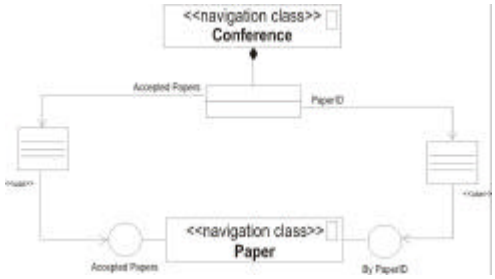

Figure 3. UWE navigation structure model using roles

4.2 Introducing roles in OOHDM

OOHDM [21] shares the same basic ideas with UWE, in particular a clear separation of conceptual from navigation models; however OOHDM uses a more "eclectic" notation; while the conceptual model is described using UML, the navigational model uses a somewhat proprietary notation, which favors expressive power. Introducing roles in OOHDM is also easy: in the conceptual model we can use the notation in Figure 1 that, as discussed before, is based on UML.

The introduction of roles in the navigational model requires a more careful discussion. In OOHDM the navigational structure of the application is specified with a navigational schema, showing nodes and links and a navigational contexts schema.

An original feature of OOHDM is the introduction of the notion of *navigational context* defined as a set of nodes sharing some property, e.g. books of an author, CDs performed by a group, products in a shopping cart, etc. When a node is navigated in a particular context, it might exhibit specific features, which are specified by using InContext classes, a kind of decorator [6] for the corresponding node class. The navigational contexts schema shows the indexes and contexts of the application. Indexes have similar semantics as their UWE counterpart; they specify the selectors (how the different index entries "look like") and the target objects. A simplified navigational contexts schema for a conference review system taken from [23] is shown in Figure 4. From the main menu it is possible to enter (via indexes) into different contexts related with papers. When navigating to a paper in the *By Pc Member* context, it is possible to easily access other papers in the same context, i.e. the other papers assigned to the same PC member; this is possible because the corresponding InContext class contains those additional links that are necessary to implement set-based navigation features.

Since nodes involved in a navigational context play a specific role when being accessed in the context, it is natural to use a role-based notation for representing differ-

ent features of those nodes in the context. In fact what we can see in Figure 4 are the different contexts (roles) in which a Paper is accessed.

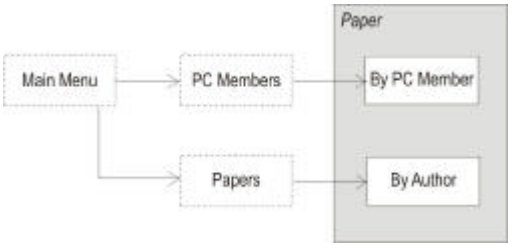

Figure 4. Navigational Contexts in OOHDM

We propose the use of the notation in Figure 2 [18] both in the navigational schema and in the navigational contexts schema since this notation is very similar to the original OOHDM notation for contexts. As a result we can express all the roles a node may play including those induced by navigational contexts using the same notation.

In Figure 5 we show the node class *Paper* with the different roles played by papers: when accessed in the context of papers of an author or assigned to a PC member and when a paper is considered a recommended paper (for example the best in each section). In the case of contexts (See Figure 4), each of these roles contains the information formerly defined in the corresponding InContext class. A paper playing the role of a recommended paper might exhibit further comments and be linked to additional references.

A set of pre-defined roles can be defined which contains usual features of navigational contexts, such as set-based navigation anchors. They can be eventually differentiated from the other roles using a gray background as shown in Figure 5.

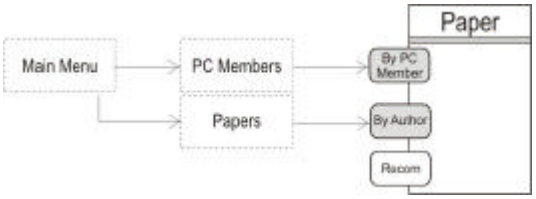

Figure 5. Describing different navigational contexts with roles

This way of introducing roles in the OOHDM's navigational model has a further advantage because it allows building a diagram (similar to the navigation structure diagram in UWE) that refines the navigational schema incorporating indexes and navigational contexts.

A further discussion not addressed in this paper is the impact of the combination of roles in the conceptual and navigational models with the use of navigational contexts. In particular, when roles defined in the navigational model are the counterpart of cor-

responding roles in the conceptual model and when they are not. Roles involved in navigational contexts are an example of this latter case, i.e. they are not derived from "conceptual" roles of a class, they are purely "navigational" roles.

4.3 Documenting the use of Web Patterns

The Hypermedia and Web Engineering communities have explored the use of patterns in the development process for some years [4], [13], [19]. Patterns record valuable experience and convey this experience to ease its reuse in software projects. Incorporating patterns in Web applications methods is critical but difficult, since patterns express abstract design ideas that have to be instantiated to be applied in a particular application; roles can be also useful to solve this problem.

In [17] the authors propose the use of role-based models to describe composite patterns; the basic idea takes into account that class-based descriptions for patterns in the style of [6] represent, in fact, roles that must be played by actual classes when the pattern is instantiated.

For example, in the Observer design pattern [6], we have two roles: Observer and Subject; when using this pattern, specific application classes will provide interfaces for playing those roles. Unfortunately, when implementing those classes, other interfaces are necessary and thus the use of a pattern (generally an important architectural decision) becomes obscured. Clearly documenting those interfaces that implement pattern related roles allows tracking their use in a software system.

The former observation shows a possible approach for indicating the use of Web patterns in a design model. In the upcoming discussion we concentrate on navigational patterns; conceptual and interface patterns can be treated analogously.

A navigational model describes the linking relationships among nodes. Links usually express relationships derived from the conceptual model and with strong semantic meaning. However, it has been shown [19] that Web patterns allow us to describe elaborated micro-architectures, in which the basic guidelines for "discovering" links are not enough. For example the *News* pattern [19] shows that when we want to make evident some "novel" nodes, we must link a home page with those nodes, even breaking the taxonomic structure of the hyper graph. In an electronic bookstore application for example we can allocate a section in the home page for "news" (such as in www.amazon.com) and build those opportunistic links to the recent books. When a book is a novelty we might want to include some additional comments in the corresponding node to make it evident.

We can describe the generic structure of the *News* pattern using a role diagram in which there will be an outstanding role, *Novelty* played by those nodes that are linked from the home page. However, when we use the pattern in a particular application such as the bookstore, we might want to make it evident that we are using the pattern.

A simple solution to this problem can be obtained by explicitly indicating that the role *Novelty* in the navigational diagram is played by books. In this way, we show which additional issues are shown when the node is navigated in that context, and we also show why the home page is directly linked to particular products. Therefore, we can add some semantics to a link that reflects an important design decision. In Figure

6, we show part of the resulting navigational diagram in which we have also indicated other roles, such as those derived from OOHDM navigational contexts.

Generalizing, in a role-enhanced navigational diagram we can indicate the use of a Web pattern by specifying those roles played by nodes when participating in the pattern.

The information provided by roles allows us to make the use of patterns explicit in a Web application project and, thereby, to benefit from the pattern properties for the ongoing design. This is useful for making design more explicit, precise and formal. The description of a process for discovering which patterns can be applied in a particular project is outside the scope of this paper (See the concluding remarks section).

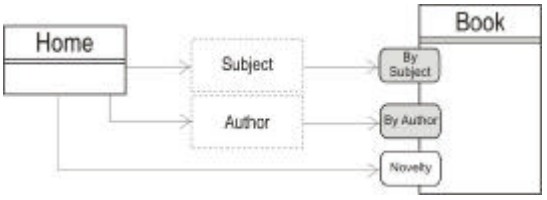

Figure. 6. Representing patterns usage in the navigational diagram

5 Related work and Discussion

As mentioned in Section 2 some authors have already proposed the use of the role abstraction in software engineering methods; however, so far, the concept has been almost ignored in the hypermedia and Web engineering community.

In [14], design and development of service-based architectures is viewed as building a hypertext structure that specifies structural and semantic behavior externally to items. According to the Information Unit Hypermedia Model (IUHM) [9], each item (data, metadata, service, etc.) of an application is encapsulated in an information unit (IU). Relationships between the components of the application structure are represented as a typed-links hypertext among information units. The *role* concept is used to type the links that define a semantic structure on information units independently of the *type* link structure that defines the kind of data contained in information units. A *role* link of an IU *A* references an IU *B*, which models the semantic behavior of all the IUs sharing the same role. A *type* link specifies how an information unit is syntactically handled independently of its usage. Both *role* links and *type* links are typed. The role link structure and the type link structure are interpreted together to determine the dynamic behavior of the application. The role and type links are used both to model and to implement service based application as a hypertext structure. The role concept is used to define the semantic of information units usage. We also use the role concept to give additional semantics to application objects; however our concern is different: while the IUHM is thought as an architectural approach for building service-based applications, we intend to use roles as modeling artifacts.

Another interesting exception is [1]. The authors propose a role-based modeling approach in which different navigational schema are built as role models and nodes are described as different roles of conceptual classes, thus unifying the conceptual and

navigational schema. The main motivation of this approach is to overcome the lack of support for logical modeling of navigation in class-based approaches (like OOHDM, WebML and UWE), and to take into account navigation issues already at the domain level. Role models can be seen as independent conceptual models, and for each context one can have a role model. The authors claim that this additional dimension in conceptual modeling can lead to clearer and more focused models, better customized to different domain contexts [1].

We share some of these motivations: conceptual and navigational design must be better integrated and performed in an incremental, iterative way; in fact, some of the previously mentioned methods have defined their own notations to incorporate user interaction into the design process such as User Interaction Diagrams [8].

However, our approach is different: we propose to enrich both conceptual and navigational models with roles, while preserving the viewing or mapping relationship between these two models, as defined in current methods. In our proposal, materialized as explained in Section 4 with different notations according to the chosen method, roles express different sets of behavioral services for an object in the conceptual model, and different "faces" of a node when being navigated in different contexts.

The discussion above is related with the use of design primitives and abstraction and reuse mechanisms during the Web Engineering life cycle. We think that the design concerns of conceptual (domain) and navigation modeling are different in essence and thus different modeling primitives and abstraction mechanisms are necessary.

Domain modeling and design must be done using well-known object-oriented heuristics and guidelines; a clear identification of classes, their attributes and behaviors is critical during this process. The role abstraction helps to improve this activity by allowing to specify those object behaviors that depend on relationships with other objects.

Meanwhile, navigational design has its roots in mature hypertext theories. Even though navigational components are specified using software engineering methods, the guidelines and heuristics for building a good linking topology are different from those aimed at obtaining a good object-oriented design. It has been shown elsewhere (See for example [22]) that a good navigational model requires opportunistic design decisions both for defining nodes' attributes, links, indexes and other hypertext structures. Some of these decisions such as those related with the relationship among conceptual and navigational objects are better expressed using a viewing mechanism than with the role mechanism. From a more architectural design point of view nodes can be seen as applications of the Observer design pattern [6], while roles allow to express different signatures (sets of services) for the same class.

As Web applications become mainstream, more complex design problems arise; it is important to use the best abstraction mechanism that is suited to solve each of those problems. Roles can be a powerful tool that must be combined with other existing (class-based) primitives and patterns. However, a deeper discussion on abstraction mechanisms is far beyond the intent of this paper though we hope to have settled the starting points for this discussion.

6 Concluding Remarks and Further work

In this paper we have discussed the use of the role modeling abstraction in the context of the Web Engineering process. We have shown that incorporating roles in our vocabulary and design armory can help us obtain more compact and expressive design models. We have argued that introducing roles in existing Web design methods can improve their modeling power; we have also shown that existing notations for roles can, in general, be used in well-known methods maintaining their consistency. We have also discussed how our approach differs from other uses of roles in the literature.

We are now working on two different research areas: We are looking for good modeling practices (using the role abstraction) when applications are built for different user profiles and common abstractions are necessary for those applications, e.g. when building different navigational models (as in OOHDM) which share some node class definitions. The same problem appears when the user profile needs to be explicitly modeled in the application, for example to provide different access rights according to the current user. In this case we need to integrate two different interpretations of the role idea: roles played by application objects and roles played by the user.

We are also working on the use of patterns all along the engineering process of Web applications. By using a formal representation of navigation patterns with role diagrams, we aim to obtain a systematic approach to improve the derivation of class models from higher level (pattern-based) role models.

We strongly believe that these ideas open a wide research spectrum in the Web Engineering community: there are still many open issues such as how to effectively use roles from the early requirement stages, when roles are to be used instead of other modeling abstractions, the impact of the use of roles in mobile Web applications, etc.

References

1. Allert . H., Dolog, P., Nejdl, W., Siberski, and W., Steimann, F.: Role-oriented Models for Hypermedia Construction – Conceptual modeling for the Semantic Web -. Technical Report, Univ. Hannover (2003)
2. Assman, U.: Role-Based Design. A concept for understanding Design Patterns and Frameworks. In http://www.ida.liu.se/~TDDB84/slides/7-role-based-design.pdf
3. Bäumer, D., Riehle, D., Siberski, W. and Wulf, M.: The Role Object Pattern. In Proceedings of Pattern Languages of Program Design (PloP) (1997) Available at: http://jerry.cs.uiuc.edu/~plop/plop97/Proceedings/riehle.pdf
4. Bernstein, M.: Patterns of hypertext. In Proceedings of the 9th ACM International Conference on Hypertext and Hypermedia (Hypertext '98), Pittsburgh, USA (1998) 20-24
5. Ceri, P., Fraternali, P., and Bongio, A.: Web Modeling Language (WebML): a modeling language for designing web sites. Computer Networks and ISDN Systems, 33(1-6), June (2000) 137-157
6. Gamma, E., Helm, R., Johnson, and R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison Wesley, Reading, (1995)
7. Gomez, J., Cachero, C., and Pastor, O.: Conceptual Modeling of device independent Web Applications. IEEE Multimedia 8(2) (2001) 26-39

8. Guell, N., Vilain, P. and Schwabe, D.:Modeling Interactions and Navigation in Web Applications. In Proceedings of the International Workshop on Conceptual Modelling and the Web (2000) 115-127
9. King, P, Nanard M., and Nanard J., Rossi G.: A structural computing model for dynamic service-based systems. Symposium MetaInformatics, Lecture Notes in Computer Science, Vol. 2994 (2003)
10. Koch, N. and Kraus, A.: The authoring process of UML-based Web Engineering Approach. In Proceedings of the 1st International Workshop on Web-Oriented Software Construction (IWWOST 02), Valencia, Spain (2001) 105-119
11. Kristensen, B.B. & Osterbye, K.: Roles, conceptual abstraction theory and practical language issues. Theory and Practice of Object Systems, 2(3) (1996) 143-160
12. Mosse, F.: Modeling Roles. A practical set of Analysis patterns. In http://www.objectdiscovery.com/papers/roles/
13. Nanard, M., Nanard, J., and Kahn, P.: Pushing reuse in hypermedia applications. Golden Rules, Design Patterns and Constructive Templates. In Proceedings of the 9 th. ACM International Conference on Hypertext and Hypermedia (Hypertext '98), Pittsburgh, USA (1998) 11-20
14. Nanard J., Nanard M., and King P.: A hypermedia-based model for integrating open services and metadata. 14th. International ACM Conference Hypertext'2003, ACM Press (2003) 128-137
15. Pernici, P.: Objects with Roles. Proceedings of the ACM-IEEE Conference on Office Information Systems (1990) 205-215
16. Riehle, D.: Describing and composing patterns using role diagrams. In Proc. Ubilab Conference I (1996) 137-152
17. Riehle, D. (1997). Composite design patterns. In Proc. OOPSLA'97, (1997) 218-228
18. Riehle, D.: Role Model Based Framework Design and Integration. In Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'98). ACM Press (1998) 117-131
19. Rossi, G., Schwabe, D., and Lyardet, F.: Improving Web Information systems with navigational patterns. Computer Networks 31 (1999) 1667-1678, Elsevier 1999.
20. Rossi, G and Schwabe, D.: Abstraction and Reuse Mechanisms in Web Applications Models. In Proceedings of the International Workshop on Conceptual Modelling and the Web, 2000.
21. Schwabe, D and Rossi, G.: An Object-Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems (TAPOS), Vol 4 (1998) 207-225.
22. Schwabe, D. and Rossi, G.: Web Application Models are more than Conceptual Models. In Proceedings of the International Workshop on Conceptual Modelling and the Web, 1999.
23. Schwabe, D., Rossi, G.: "Conference Review System in OOHDM". In Proceedings of the International Workshop on Web Oriented Software Technology, Valencia, Spain, 2001. Avaliable at http://www.dsic.upv.es/~west2001/iwwost01/
24. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison Wesley (1984)
25. Steimann, F.: On the Representation of Roles in Object-Oriented and Conceptual modeling. Data and Knowledge Engineering 35 (2000) 83-106
26. Steimann, F. A radical revision of UML's Role Concept. UML Proc. 2000, Springer (2000) 194-209
27. Steimann, F.: Role=Interface: a merger of Concepts. Journal of Object-Oriented Programming, Oct./Nov. (2001) 23-32
28. Fowler, M.: UML Distilled. Addison Wesley (1997)
29. Object Management Group: The UML 1.5 Specification. In http://www.omg.org/uml/