Separation of Structural Concerns in Physical Hypermedia Models

Silvia Gordillo[1,3], Gustavo Rossi[1,4,**], and Daniel Schwabe[2,††]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{gordillo, Gustavo}@sol.info.unlp.edu.ar
http://www-lifia.info.unlp.edu.ar
[2] Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil
dschwabe@inf.puc-rio.br
http://www.inf.puc-rio.br
[3] Also CICPBA
[4] Also CONICET

Abstract. In this paper we propose a modeling and design approach for building physical hypermedia applications, i.e. those mobile applications in which physical and digital objects are related and explored using the hypermedia paradigm. We show that by separating the geographical and domain concerns we gain in modularity, and evolution ease. We first review the state of the art of this kind of software systems, arguing about the need of a systematic modeling approach; we next present a light extension to the OOHDM design approach, incorporating physical objects and "walkable" links; next we generalize our approach and show how to improve concern separation and integration in hypermedia design models. We compare our approach with others in the field of physical and ubiquitous hypermedia and in the more generic software engineering field. Some concluding remarks and further work are finally presented.

1 Introduction

The idea of physical hypermedia (PH) was first introduced in [7] to support design activities and to organize collections of different media, and in [15] as a formalism to build augmented reality applications. In these software systems, physical objects are augmented with digital information which can be accessed by the mobile user, for example while standing in front of the object. Physical objects can be further considered as nodes in a hypermedia network and thus linked with other nodes either physical or digital. When dealing with digital objects, the user will traverse the link using the well-known navigation paradigm (e.g. as in the WWW); in other cases (e.g. when physical objects are involved) the link will have to be "walked" by the user [9].

** Gustavo Rossi is partially funded by Universidad Abierta Interamericana in its project "Conceptual Modelling of Web Applications".
†† Daniel Schwabe is partially funded by CNPq – Brasil.

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 446 – 459, 2005.

A simple example scenario is a museum in which visitors are equipped with portables computer devices. When the visitor stands in front of an artwork, he gets multi media information about the artwork in his portable device. Additionally, he is presented with a set of anchors that allow him to navigate to other objects (hypermedia nodes) related with the artwork. When one of these nodes is another artwork in the museum, he can be shown how to reach this artwork; he can then choose to traverse the physical space (walk the link) towards this node, or just continue his actual tour. Notice that we are not just augmenting the physical object (artwork) with some digital information but also providing some kind of linking to other digital or physical objects.

These ideas allow to apply the hypermedia paradigm to the real world; it has been shown elsewhere [5] that we can also build rich social interactions when users can link their own comments to a physical object, for example as digital graffiti or recommendations, and these comments can be accessed or further discussed by other users.

In our research we are interested in how to model and design these applications, i.e. in analyzing which software modeling and design issues we face while building PH applications, which software abstractions we need to clearly indicate the intended structure and behavior of a PH network, and the way in which those abstractions relate with each other.

In this paper we present a novel approach for the design of PH applications; this approach seamlessly extends the Object-Oriented Hypermedia Design Method (OOHDM) [18] to support physical objects and "walking" navigation. We show that, by clearly separating the fundamental concerns in this kind of software, we improve modularity and ease of evolution. We also show how to go further with this approach in complex application domains, by applying well-known techniques for separation of orthogonal (or partially overlapping) concerns.

The main contributions of this paper are the following:

- We indicate which design issues must be faced while modeling PH applications,
- We present a concise design approach that can be easily adapted by other hypermedia and Web modeling methodologies like UWE [11] or WebML [3] to extend their scope to the physical world.
- We present an approach for designing navigational structures that depend on application concerns. We define concern-driven navigation as an extension of our approach of concern decoupling in physical hypermedia.

The rest of the paper is organized as follows: In Section 2 we introduce our modeling approach; we briefly explain the OOHDM framework and indicate how we extended it to support PH and we analyze several navigation issues. In Section 3 we discuss how to generalize our basic approach to include other concerns. In Section 4 we compare our work with others both in the hypermedia and software engineering fields. Some further work and concluding remarks are finally described in Section 5.

2 Modeling Physical Hypermedia Applications

While researchers have emphasized the feasibility of the PH paradigm by building software infrastructures that support these ideas [8,9,15] and performing usability studies [7], modeling and design issues have been so far ignored. We believe that the inherent complexity of these applications requires a special emphasis in modeling and design.

To make this discussion concrete, we define a PH application as a hypermedia application (i.e. the access to information objects is done by navigation) in which all or some of the objects of interest are real-world objects which are visited by the user "physically". The most usual scenario for these applications involves a mobile user and some location sensing mechanism and underlying software that can determine for example when the user is within interaction range of one of these objects.

In this specific domain we need to express, in an implementation-independent way, which are the objects of interest and their properties (including their location), how they are linked, which links should be implemented as conventional and which should be "walked" by the user. We should be able to cope with technology evolution and heterogeneity, i.e. the design model should not be compromised with details on location-sensing technology and at the same time it should allow to build models that can gracefully evolve together with new technical possibilities.

We have extended the OOHDM [18] design approach by adding a few concepts such as physical objects and slightly changing some navigation semantics to adapt them to the physical hypermedia field. The object-oriented nature of OOHDM and its open meta-model allowed us to achieve this objective easily, without changing the basic assumptions and primitives of the methodology. In the following sub-sections we stress those modeling constructs that are fundamental to the development of this kind of context-aware software. In particular we emphasize how to describe basics structures and behaviors in a high level way. We purposely ignore aspects related with user modeling and user context-aware adaptation that have been discussed elsewhere [1,10,19].

2.1 The OOHDM Design Approach

OOHDM partitions the development space into five activities: requirements gathering, conceptual design, navigation design, abstract interface design and implementation. The first step is to elicit stakeholders' requirements which helps to identify actors and the tasks they must perform. Scenarios are collected by means of User interaction Diagrams, a special form of Use Cases. During conceptual design we describe the application classes and their relationships using UML [21].

For each user profile we can define a different navigational structure according to the tasks this kind of user must perform. The linking structure of a Web application is then defined by a navigational schema, built from navigational meta-classes such as nodes, links, anchors and access structures such as indexes. Each node is defined as a view over conceptual objects, acting as an Observer on those objects [6]. The separation between objects and their views allows customizing the structure of nodes and the linking topology to the needs of the corresponding user profile and task.

Additionally, the navigational contexts schema defines the meaningful sets of nodes that the user will traverse and the intra-set navigation topology. For example, we can specify the navigational context "Artworks by Painter" to denote the set of paintings of a particular painter, and specify that sequential navigation (from one artwork to the next, according to some specified ordering) is allowed. Since the same artwork might belong to different sets, e.g. Artworks in a Time Period or Artworks in a Room, the differences when accessing it in one context or others, are expressed using InContext classes, that extend the basic features of a node in the particular navigational context. Details on these primitives can be found in [19].

The abstract interface model defines which interface objects the user will perceive (in particular how nodes will look like) and which interface transformations will take place. Finally, during the implementation activity the whole set of models is mapped into a run-time environment. Though OOHDM does not prescribe a particular strategy for implementing a hypermedia or Web application, its approach can be naturally mapped to object-oriented languages and architectural styles, such as the Model-View-Controller. In the following sub-sections we concentrate in the conceptual and navigation design activities.

2.2 Dealing with Physical Objects

We extended the OOHDM conceptual meta-model by adding the concept of Physical Object and a simple User Model. Though the user model is not described here for the sake of conciseness, it suffices to say that it contains information about the current user's position together with his (general or application specific) preferences. Details on the introduction of user's information in OOHDM models are discussed in [19].

A physical object is an application object that can be explored "physically", i.e. it has a physical presence in the system and the user can be tracked if he is within interaction range of it. In the museum example, we can be interested in modeling artworks, rooms, corridors or other places as physical objects.

To find a suitable approach for modeling physical objects, we need to consider that not all objects of the same class (e.g. Artwork) should be tagged as physical: for example, in the museum, we might want to relate artworks in the museum with others that are not in exhibition, or are in another geographical place, or simply do not exist anymore. The "physicality" of an object is a completely separate concern from its other characteristics, such as its (sub) type. Therefore, representing physical objects as sub-classes of a particular class (e.g. Artwork) introduces a specialization criteria that might collide with others in the intended domain (paintings, sculptures, etc) and also prevents us for considering an artwork alternatively as physically accessible or not for when it is not longer in the museum or when it is presented in an exhibition. To accommodate this, we have then chosen to model physical objects as roles that can be assumed by conceptual objects. Roles have been extensively used to model and integrate different points of views on the same reality [13], both as conceptual modeling and design artifacts [14,20]. Roles can easily be mapped to simple implementations either using decorators as shown in [14] or even Java interfaces.

A role type (in this case a sub-type of the basic role Physical) indicates those properties and behaviors of an object when playing that role, i.e. when the object has a physical presence. In Fig. 1 we show a simple conceptual model for the physical museum; roles are described using the notation in [14]. Small white boxes indicate that the corresponding class can play the corresponding physical role, i.e. any object of the class can be considered physical.

Physical objects are characterized by an attribute *location* whose semantics depends on the location model being used, and an operation to change location (if the object is mobile or can be changed of place); *location* can be just an identifier (e.g. if we use code bars or infrared sensing), or we might need a more complex representation. In our meta-model we provide a set of basic geometries and reference systems

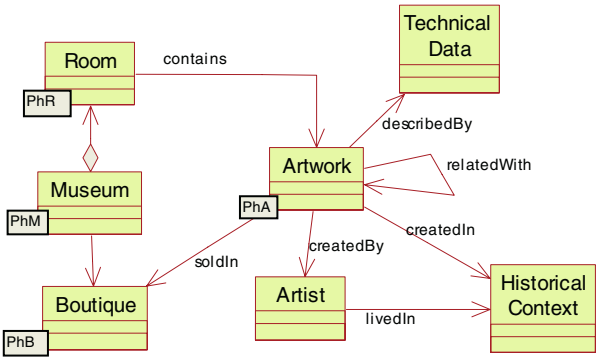

Fig. 1. Conceptual Model of the museum including physical objects

for locations as shown in Fig. 2. A designer might choose one of them or define his own location model. Using the simple solution of Fig. 2, we can deal with different representations and location-sensing technologies even for the same class of objects; for example if an artwork is located in a square outside the museum we could use global coordinates, as usual with GPS technology. This design structure also simplifies evolution when location or sensing technology changes, as the design model can be seamlessly adapted to the evolution, for example by adding a new type of reference system.

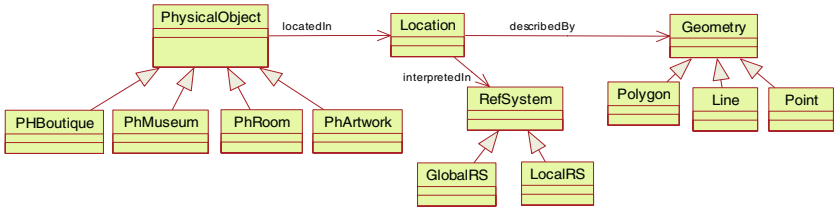

Fig. 2. Decoupling Location model from physical objects

The location of a physical object involves a value expressed in a Geometry (point, line, polygon, etc.) and interpreted in one reference system. For the sake of simplicity we do not explain details of location interpretations, which has been well covered in the literature of geographic information systems [12].

Physical (role) objects possess a default behavior that allows them to handle the event signaled by the user being within interaction range, by opening (activating) the corresponding node. Additionally, they could be able to inform how the user can reach them from any location; this behavior is triggered by “walkable” links (See 2.3) to indicate how the user can “navigate” physically to the object. The object can either answer its absolute location, a plan or the route one must follow to reach it from the actual location. The designer must specify this last and eventually other behaviors as they are application dependent. In Fig. 3 we schematize the fundamental spatial behaviors.

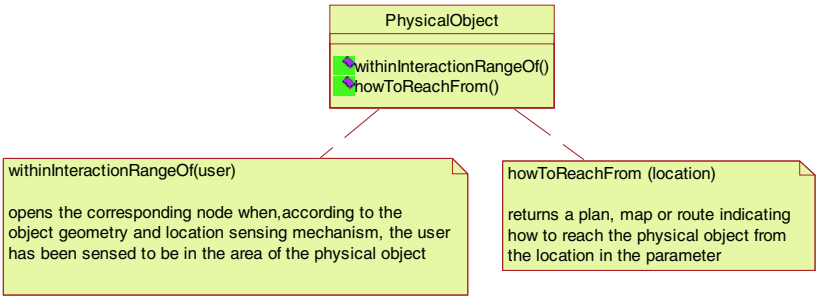

Fig. 3. Spatial Behaviors of Physical objects

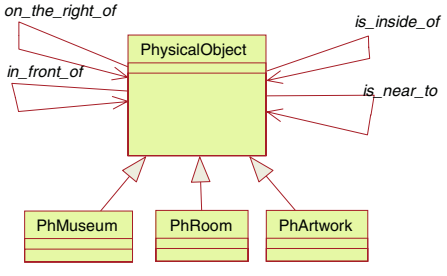

Fig. 4. Relationships among Physical Objects

Physical objects may be related using a variety of spatial and geographical relationships and predicates which give us a tool to build interesting navigational structures as will be described in Section 2.3. In Fig. 4 we show some well-known spatial relationships. Notice that their implementation in a particular system might need that they are specified in an instance basis (e.g. an artwork *in_front_of* another). In other cases they can be calculated using the corresponding geometrical behaviors [12]. Designers can devise new relationships specific to their domain of interest. These relationships will be shown in the model of Fig. 1 as relationships between the physical roles of the corresponding classes.

2.3 Navigation Issues

Navigation in PH applications is also described by a navigational schema with the OOHDM semantics. This allows us to eventually let a user explore a physical object even when not physically near it, by just defining the corresponding Node class; in our example this is useful for those artworks under restoration, which will be only accessed physically by specialists.

There are, however, two major differences between a conventional and a physical hypermedia regarding the operational semantics expressed by the navigational schema: the activation of nodes and the semantics of link traversal.

In conventional hypermedia a node is opened when we navigate a link having that node as a target. While we want to preserve this behavior for "pure" digital nodes, a node that stands for a physical object should be opened when the user is within interaction range of the object. Of course we might also want to build a "conventional" web interface for that object which can be trivially done using the viewing mechanisms of OOHDM, so we won't discuss this possibility here.

We decided not change the basic Node class hierarchy; as we explained in Section 2.2, a node corresponding to a physical object will be opened when the user is within interaction range of the object as a consequence of the default behavior specified for its role as a physical object. We can of course implement more sophisticated context-aware or personalized activating behaviors; this can be done by using the strategies discussed in [19].

Meanwhile, to implement a different navigation semantics, we defined "walkable" links (or WLinks) as those links whose target node is the digital counterpart of a physical object. The main difference between the operational semantics of a navigational and a walkable link is that, while the former usually closes the current node and opens the target node, the latter just indicates the user intention to reach the corresponding physical object.

WLinks are designed by changing the default link traversal algorithm, which is expressed in OOHDM as a Strategy [6] on Link classes as described in [18] and shown in the left of Fig. 5. In order to achieve the desired behavior, the link traversal algorithm invokes the *howToReachFrom* behavior (Fig. 3) in the physical object corresponding to the target node; the actual user location used as a parameter is the location of the link source node. A schema of the decision structure of the WalkingLink traversal is shown in Fig. 5.

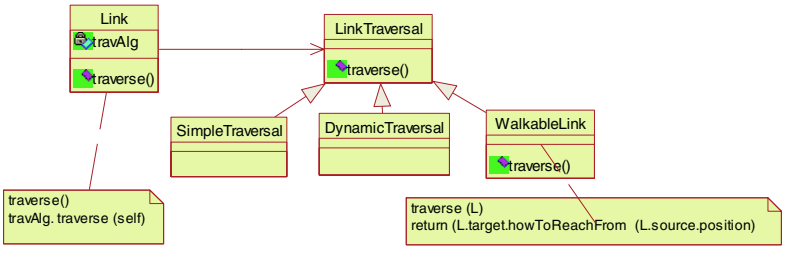

Fig. 5. Walkable links as Strategies on Links

This design style allows implementing different types of "walking" semantics by specifying a different algorithm as another Strategy class, either at the same level of *WalkableLink* or as a concrete sub-class that re-writes the method *traverse*. For example, in a production line application, we might want that the target object moves towards the user, so instead of asking for a map we could send a message such as *L.target moveTo (L.source.position).*

Decoupling links from their traversal algorithms also allows us to express differences at the link instance level, for example when an instance of a WLink class has an exceptional "non-walking" semantic, i.e. it behaves as a conventional link, and we

show digital information of the object even if the user is not in front of it. In Fig. 6 we show the navigational schema for the visitor user role that corresponds to the conceptual model in Fig. 1. WLinks are shown with a <<W>> in the style of UML stereotypes [21]. We omit the name of link classes for the sake of simplicity. With this solution we also cope with evolution problems that are mapped to changes in instances of the corresponding Link class instead of requiring changes in the overall class.

WLinks, in the same way as navigation links, allow us to explore physical objects by mapping conceptual relationships shown in Fig. 1 into walkable links. For example two artworks that are closely related might be linked to suggest the user a less conventional museum visit. Notice that additional usability issues arise, e.g. to avoid suggesting the user to perform long walks: they also become a design concern. Interesting discussions on this subject can be found in [7,8].

Physical objects can also be related through rich spatial relationships either generic as shown in Fig. 4 (e.g. near, in front of) or application specific (e.g. in the same room), that may also induce interesting navigation relationships and structures. These relationships can help to create navigational contexts by grouping objects according some location property; for example we can describe the set of Rooms close to the Boutique, or the set of Artworks located not far from a certain zone in the Museum. In a tourist application we might want to visit the monuments along a road or a river, or those that are near a particular place. It is interesting to note that, when using WLinks to implement intra-set navigation in a navigational context, the context represents a real guided tour along a geographical space. Furthermore, these location-based relations can be combined with the other types of relations in defining meaningful contexts; for example, while traveling in southern France, one may wish to see the Artworks by Van Gogh depicting scenes in that region.

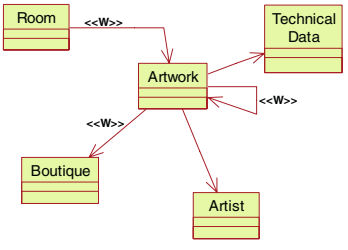

Fig. 6. Navigational Schema with WLinks

3 Advanced Concern Separation

In previous sections we showed how to decouple conceptual properties from spatial properties of physical objects. We have also shown that the spatial perspective opens new yet unexplored possibilities for building navigation structures. While the spatial concern is crucial for physical hypermedia, we can generalize this discussion to other non-spatial concerns. Some of them will be orthogonal with the others: for example, the user model could be described without directly affecting application classes [19].

Other types of concerns however are more difficult to handle. Suppose that in the Museum example we want to model the history of the museum, e.g. how the building and collections evolved over time, when artworks arrived at the museum, and why. Should we clutter the existing model (e.g. Fig. 1) with information and relationships related to this new concern? Additionally, how do we deal with the cognitive overhead eventually produced by links belonging to different "themes"? We introduce these ideas in the following sub-sections by first formalizing how to clearly separate spatial from conceptual modeling.

3.1 Conceptual Versus Spatial Modeling

By using well known techniques for concern separation such as Role Modeling [16] or Subject-Oriented Design [4], we can re-think the conceptual model of Fig. 1, as dealing with two separate sub-models and specify them as shown in Fig. 7, with UML packages, namely *KernelMuseum* and *PhysicalMuseum.*

Thus, each model can be engineered separately, and relationships can be thought and evolve independently thus improving modularity. When two classes with the same name exist in both models (e.g. Room, Artwork, Boutique) we need to solve this overlap when integrating the models. We did this as shown in Fig. 1 by using roles in classes with the same name. Fig. 1 therefore represents a design refinement of the higher level model in Fig. 7.

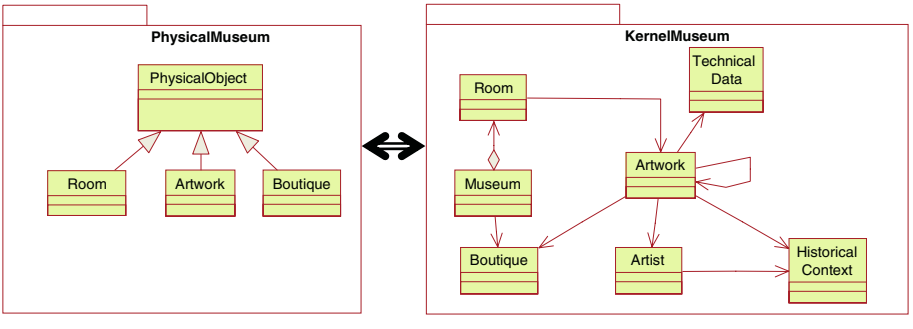

Fig. 7. Decoupling design concerns

Notice that the PhysicalMuseum model might also contain relationships between application classes, that will be mapped onto relationships between the corresponding (physical) role classes.

3.2 Extending the Approach to Other Concerns

We can use the same design philosophy shown in Fig. 7 when introducing other concerns into the conceptual design activity. Generalizing the ideas behind Fig. 7, we show in Fig. 8 a model that adds the *Historical* concern of the Museum to the previous example.

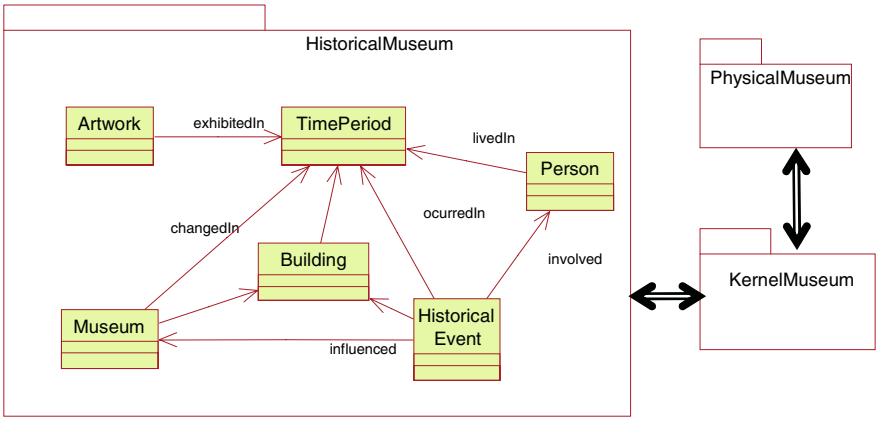

Fig. 8. Multiple concerns in conceptual models

In Fig. 8 we considered the *KernelMuseum* concern as the central one; historical and physical classes and relationships should be further integrated with those in the kernel. This can be done again using roles in overlapping classes; in this example we will have a *Historical* role for *Artwork* and *Museum*; each role will host attributes and relationships that belong to the corresponding concern. Interesting problems arise when dealing with more complex applications or behavioral requirements (see for example [4]). Other techniques such as Subject-Oriented Design [4] or Aspect Oriented Design [2] can be used to describe the conceptual model when cross-cutting requirements exist, though they are usually applied to more technical or programmatic concerns as discussed in section 4.

3.3 Discussion: Towards Concern-Driven Navigation

It is interesting to analyze the impact of introducing multiple concerns in the navigational model; in our example, what does it mean to navigate the hypermedia space following one of those concerns? What kind of software abstractions do we need in order to support this kind of navigation?

We say that a hypermedia application supports concern-driven navigation, when it is possible to choose one particular application concern and emphasize the information and links corresponding to that concern, even at the expense of eliminating others. For example, a user exploring the *Historical* concern of the museum might want to ignore all information not related with this concern, or perhaps be able to "switch" from one concern to another. Meanwhile, a person on a tourist trip might just want to be "pointed" to places of interest, instead of just navigating through digital data. Some concerns might be mutually exclusive, while others might be pervasive; e.g. in the physical museum, the spatial concern is always accessible, since the user is traversing the museum physically.

The discussion in 3.2 also applies to the navigation model; a designer might derive different navigational schemas, one for each possible concern, using OOHDM primitives, and then integrate them in a single schema, once again using roles. In the role

model, while the base node class specifies information that the nodes always exhibit, a role type will indicate which additional information and links a node will show when playing the corresponding role, i.e. when accessed within the corresponding concern. Navigational role types are derived from those roles representing concerns in the conceptual model. It is a designer choice to decide which navigation concerns are useful for a particular user profile or task, e.g. using the OOHDM viewing mechanism for a specific hypermedia application. A similar solution has been used in [17] for solving more general navigation problems.

We do not impose a particular kind of association among roles and nodes. Whereas the standard unidirectional association is taken as a default (roles know about base nodes but not vice versa), a designer might want to make nodes aware of the roles it can play, in order to provide additional navigation operations. He can design an operation for changing the actual concern (role) by another to allow more flexibility in the user navigation; this can be useful, for example, for adaptive hypermedia. For example, while exploring one node from a historical point of view, we might want to see the "other face" of the node and continue exploring the physical concern. The idea of concern-driven navigation opens many additional design issues, outside the scope of this paper.

4 Related Work

Some of the design problems addressed in this paper have been the focus of interesting research projects in the hypertext and object-oriented communities. We next discuss some of them to highlight our contributions.

4.1 Hypermedia

In [8] a comprehensive framework (HyCon) for deploying applications in which the hypermedia paradigm is extended to the physical world is presented. The authors not only show how to provide situated authoring and browsing but also show different usage patterns in this kind of applications. In [15] meanwhile, an object-oriented framework called HyperReal, based on the Dexter hypertext reference model is presented. As in [8] the authors show a powerful software substrate for building augmented reality software. Our research is more oriented towards the modeling and design of physical hypermedia applications: once the intended structure and behavior of a PH application has been specified using the extended OOHDM, it could be implemented for example using HyCon or HyperReal. We believe, however, that for these technologies to become mainstream, some standardization at the level of implementation architectures is needed. We are now studying how to extend the MVC metaphor to include the physical dimension (See Section 5).

The goal of Adaptive Hypermedia (AH) [1] has been to improve the usability of hypermedia applications by taking into account user interests and profiles. In this way adaptive hypermedia has studied how to adapt the contents (nodes) and topology (links) of the application according to a user model. Mature research in AH has led to separating the user model from adaptation rules and from the domain model. The UWA project [10] meanwhile has dealt with providing ubiquitous access to Web

applications. A comprehensive modeling and design approach including user models and adaptation rules has been devised.

Although PH is a kind of ubiquitous and adaptive hypermedia software, our research has a different intent with respect to providing adaptive behaviors. First, we are studying how to design applications in which real and digital objects are linked with the hypermedia paradigm, and exploring how to build meaningful navigation structures that take into account the spatial domain. Besides, we are carrying separation one step further than in UWA and AH by applying it to specific application concerns, e.g. the physical concern.

4.2 Object-Oriented Modeling and Design

Separation of concerns has been a recurrent theme in the software engineering and in particular the object-oriented field. Many researchers have argued that the object abstraction is not enough to solve problems such as cross-cutting concerns, misalignment between requirements and designs and evolving behaviors. These problems have been addressed using Aspect-Oriented Programming [2], Subject-Oriented Programming and Design [4] and Role Modeling [16]. Aspect-orientation has focused mainly on technical domains, such as persistence, caching, security, etc. Subject-Oriented Programming has been first used at the programming level and more recently for aligning requirements with designs.

Our work is grounded on the ideas of Role Modeling (in fact we use the role construct heavily in our approach), but with an original application focus: to separate physical from more conceptual aspects. Subject and Aspect Orientation have not been used in the field of hypermedia so far. Our concept of concern-driven navigation meanwhile has not been addressed previously in the literature, although the original OOHDM InContext class primitive can be seen as a first step in this direction.

5 Concluding Remarks and Further Work

In this paper we have presented an original approach for modeling physical hypermedia applications, i.e. those applications in which physical and digital objects are related using the hypermedia paradigm. We have shown how to extend our approach to a broader domain: to build hypermedia applications in which there are many different concerns, for example corresponding to application “themes” or subjects. For space reasons we have not discussed customization and personalization issues; given that PH applications are a particular example of context-aware software, these issues are fundamental. It is relatively straightforward to apply previous ideas on Web applications customization [19] in this extended OOHDM framework.

We are currently working on several research directions. One of them relates with providing better modeling tools to express navigational structures. Physical navigation introduces a new kind of context, different from the OOHDM idea of navigational context: the actual location of the user is relevant to provide him (physical) navigation cues, for example in the form of links or landmarks. Suppose that the user is in front of an object of interest (e.g. an artwork); the corresponding node exhibits two kind of links: navigational and Wlinks, the latter ones allow navigation to other

physical objects in the environment. A reasonable design decision could be to keep those links active (visible) while the user navigates to other digital objects related with the artwork (as he is still in the same physical position). In this way, he is always aware of the actual physical navigation options he has from this location. This is certainly a navigational design problem; we want to express that some of the nodes reachable from one particular node (the one corresponding to a physical object) "inherit" the walkable links of this node. We are studying how to solve this design requirement by using a slight modification of the OOHDM concept of composite nodes.

We are also improving our notation to make it more "standard" by exploring the use of UML stereotypes and the OCL [21] to express design constraints. We are also researching on the exploitation of the geo/spatial dimension in order to find a better way to integrate it into other design concerns. We are finally working on implementation issues, such as adapting the MVC architectural style to physical hypermedia applications. In the context of a prototype implementation for the Museum of Natural Sciences in La Plata we are also experiencing usability aspects and their relationships with the concepts presented in this paper.

References

1. Adaptive Hypermedia Home Page: http://wwwis.win.tue.nl/ah/
2. Special Issue on Aspect Oriented Programming. Comm ACM, October 2001.
3. Ceri, P., Fraternali, P.: Web Modeling Language (WebML): a modeling language for designing web sites. Computer Networks and ISDN Systems, 33(1-6), June (2000) 137-157
4. Clarke, S.: Composition of Object-Oriented Software Design Models. Ph.D. Thesis, January 2001, Dublin City University. In: www.cs.tcd.ie/Siobhan.Clarke/papers/SClarkeThesis.pdf
5. Espinoza, F ,Persson, P, Sandin, A., Nystrom, H., Cacciatore, E. , Bylund, M.: GeoNotes: Social and Navigational Aspects of Location-Based Information Systems". Proceedings of Third International Conference on Ubiquitous Computing (Ubicomp 2001), Springer Verlag, 2-17
6. Gamma, E., Helm, R., Johnson, J., Vlissides, J. : Design Patterns. Elements of reusable object-oriented software, Addison Wesley 1995
7. Gronbaek, K., Kristensen, J., Eriksen, M.: Physical Hypermedia: Organizing Collections of Mixed Physical and Digital Material. Proceedings of the 14th. ACM International Conference of Hypertext and Hypermedia (Hypertext 2003), ACM Press, 10-19
8. Hansen, F., Bouvin, N., Christensen, B., Gronbaek, K, Pedersen, T. ,Gagach, J.: Integrating the Web and the World: Contextual Trails on the Move. Proceedings of the 15th. ACM International Conference of Hypertext and Hypermedia (Hypertext 2004), ACM Press. 2004
9. Harper, S., Goble, C., Pettitt, S.: proximity: Walking the Link. In Journal of Digital Information, Volume 5, Issue 1, Article No 236, 2004-04-07. Available at: http//jodi.ecs.soton.ac.uk/Articles/v05/i01/Harper/
10. Kappel, G., Proll, B., Retschitzegger, W.: Customization of Ubiquitous Web Applications. A comparison of approaches. International Journal of Web Engineering and Technology, Inderscience Publishers, January 2003
11. Koch, N., 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

12. Laurini, R., Thompson, D.: Fundamentals of Spatial Information Systems, Academic Press Ltd, 1992
13. Pernici, B.: Objects with Roles. Proceedings of the ACM-IEEE Conference on Office Information Systems (1990) 205-215
14. 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
15. Romero, L., Correia, N.: HyperReal: A Hypermedia model for Mixed Reality. Proceedings of the 14th ACM International Conference of Hypertext and Hypermedia (Hypertext 2003), ACM Press, 2-9
16. Reenskaug, T: Working with objects. The OOram Software Engineering Method. Manning/Prentice Hall 1996.
17. Rossi, G., Nanard, J., Nanard, M: Engineering Web Applications using Roles. Technical Report LIRMM, University of Montpellier, May 2004
18. Schwabe, D, Rossi, G.: An object-oriented approach to web-based application design. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, October, 1998, 207-225.
19. Schwabe, D, Guimarães, R., Rossi, G.: Cohesive Design of Personalized Web Applications. IEEE Internet Computing 6(2): (2002), 34-43
20. Steimann, F.: On the Representation of Roles in Object-Oriented and Conceptual modeling. Data and Knowledge Engineering 35 (2000) 83-106
21. The UML home page: www.omg.org/**uml/**