Navigation Analysis and Navigation Design in OO-H and UWE [⋆]

Cristina Cachero[1] and Nora Koch[2]

[1] Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{ccachero}@dlsi.ua.es
[2] Ludwig-Maximilians-Universität München. GERMANY
kochn@informatik.uni-muenchen.de

Abstract This technical report supplements the paper entitled 'Conceptual Navigation Analysis: a device and platform independent navigation specification'. It presents how a navigation analysis model can be integrated in two well-known user-centered conceptual modeling approaches, namely OO-H and UWE. Our aim with this integration process is to illustrate how other user-oriented hypermedia modeling proposals might also include a similar navigation analysis model as part of their specification. Also, this paper discusses the kind of restrictions that the provision of a navigation analysis model may impose on underlying design navigation models. We claim that a navigation analysis model such as the one presented in the main paper might substitute, or at least complement, the domain structural model on which most hypermedia modeling proposals are currently basing its navigation definition.

1 Introduction

One of the perhaps more controversial concepts used in the Web community is the concept of navigation. In the paper entitled 'Conceptual Navigation Analysis: a device and platform independent navigation specification' ([2]) we define the concepts of *Navigation Semantic Unit* and *Navigation Semantic Link*, and propose to use these concepts to build a navigation analysis model. Furthermore, we justify how a clear separation between navigation analysis and navigation design activities would improve the development of device and platform independent Web applications.

In this article we go a step further and analyze, based on the navigation analysis model presented in [2], which consequences the inclusion of such a navigation analysis model may have on existing hypermedia (Web) design methods.

[⋆] Technical Report 0205, Institute of Computer Science, Ludwig-Maximilians University of Munich, April 2002. This paper has been partially supported by the Spain Ministry of Science and Technology (project TIC2001-3530-C02-02), and by the European Union within the IST project AGILE: Architectures for Mobility (IST-2001-32747).

In order to get such aim, in this paper we describe the inclusion process for two well known methods, namely OO-H (Object-oriented Hypermedia method) [6] und UWE (UML-based Web Engineering) [8]. We hope that this integration effort will provide an illustrating example as well as a starting point for a fruitful discussion on the advantages and limitations of including a navigation analysis model such as the one presented in [2] in these and other Web development proposals.

For the sake of clarity in the remaining of the paper we will employ the same example presented in the discussion in [2] to illustrate the navigation analysis concepts applied to OO-H and UWE. It is a small example about a *Hotel Management* Web application. In this system, let's imagine that we have identified two roles: the *receptionist* and the *hotel administrator*. Let's also suppose that the receptionist is allowed to *view current reservations*, *reserve a room* (adding also the related customer if necessary), register a *customer check-in*, note down the services (laundry, room breakfast, etc.) provided to such customer while staying in the hotel (*manage charges*) and register the customer departure (*check-out customer*). On the other hand the hotel administrator may not only perform the above mentioned tasks, but also generate a set of reports (namely, *generate Invoice Report* and *generate Occupation Report*), *modify the customer data* and manage any other hotel characteristic (*manage Hotel Data*) such as service types offered to customers, room characteristics, etc. These requirements may be gathered by means of a UML-compliant [11] Use Case Diagram such as the one depicted in Fig. 1.

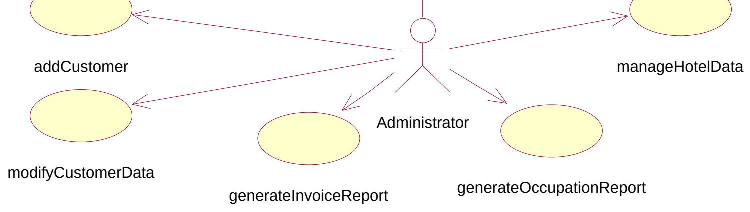

Figure 1. *Hotel Management System: Use Case Diagram*

This article is structured as follows: Section 2 presents a summary of the concepts related to semantic navigation defined in [2]. Based on the navigation analysis model built in [2], section 3 describes the results produced by the design modeling activities of OO-H and UWE. Finally, section 4 presents the conclusions of this work.

2 Navigation Analysis

The Use Case diagram representing the user functional requirements provides enough information to define a sound navigation analysis model that can be shared among different implementations (be them for different users, platforms and/or target devices). This navigation analysis model is based on two constructs: *Navigation Semantic Units (NSU)* and *Navigation Semantic Links (NSL)*. A NSU can be defined as a set of information and navigation structures that collaborate in the fulfillment of a subset of related user requirements. This concept is the basis for the concept of *Semantic Navigation* understood as a **voluntary invocation of a NSU**, with the purpose of fulfilling a certain user requirement. The set of possible invocations on each NSU are captured by means of different NSL. Each NSL can either connect different NSU or provide the user with a direct access path to fulfill a given requirement. From this definition we can conclude that **the activation of a NSL always implies a change in the user interaction aim**.

In order to derive the different NSU from the Use Case Diagram the designer must perform a grouping activity on the different use cases. This activity may be based on different types of criteria:

- Similar purpose: some requirements share a general purpose, e.g. *generate invoice report and generate occupation report*, all of them aiming at showing any kind of statistics.
- Functional dependency: other requirements are always used in conjunction with (to support) others, e.g. *create customer*, which support the *customer check-in* and *customer room reservation* requirements.
- Data involved: still other requirements deal with the same domain concept, e.g. *add customer, modify customer data*, both dealing with the *customer* data structure.
- Same object state: also, some requirements are applied to objects in the same state, e.g. *check-in customer and manage charges* are both invoked on customers that have already arrived in the hotel, while *reserveRoom and viewReservations* are related to customers not yet in the hotel, and *check-out customer and generate invoice report* are related to clients that are about to leave (or have already left) the hotel.
- Temporality: last, some requirements may be related by temporality of use, e.g. (although not part of our example) imagine that we had the requirements *set hotel rooms, set available services and set hotel personnel*, all of them related to the application start-up.

These criteria are usually combined in the boundaries of the same system, and the election of one or another usually depends on the subjectivity of the designer.

Graphically speaking, an NSU is depicted as a UML class symbol decorated with a ≪*semantic unit*≫ stereotype. Back to our example, and departing from the Use Case diagram presented in Fig. 1, in Fig. 2 we can observe the Navigation Analysis Model corresponding to the *Hotel Management System*, and how the answers to the set of requirements specified in Fig. 1 are encapsulated in five NSU:

– Reservations: structure and behaviour related to hotel reservation management.
– Stays: structure and behaviour related to hotel occupation management.
– HotelAccounts: structure and behaviour related to hotel departure and payment management.
– Customer: structure and behaviour related to customer management.
– HotelManagement: structure and behaviour related to general hotel characteristics management.

As the reader might already have inferred, the *Reservation, Hotel Management and Customer* NSU are examples of the application of the *data* criterion, and gather information related to the reservation, hotel and customer concept respectively. In the case of the *add customer* requirement, the data criterion (grouping requirements related to client maintenance) has also prevailed over its functional dependency with respect to *checkInCustomer* and *reserveRoom*. This decision implies that adding a client remains relatively independent from checking in a customer or making a room reservation, thus simplifying its (possible) future inclusion as a new high level user requirement. On the other hand, we observe how *Stay* and *Hotel Accounts* are derived from applying a state criterion, and gather the information and functionality needed to answer the requirements that might arise while the customer is in the hotel or once it has left the hotel, respectively. This criterion has prevailed on the *similar purpose criterion* that would have caused e.g. the grouping of *generate Invoice Report and generate Occupation Report* on an independent NSU.

For each NSU, we must define the set of attributes and operations that are relevant from a user perspective in order to fulfill their requirements. Note (see Fig. 2) how these attributes and operations are not necessarily equal to those that will appear in the class diagram, that reflects the domain structure and not the user perspective of that domain. In this figure we also observe how attributes and methods may be repeated among different NSU. We would like to stress that this repetition does not necessarily imply a common mapping rule to any underlying domain concept. For example, we may be interested in different attribute *perspectives* [4] appearing in different NSU, which in turn may imply different structural attributes providing the contents to that navigation analysis

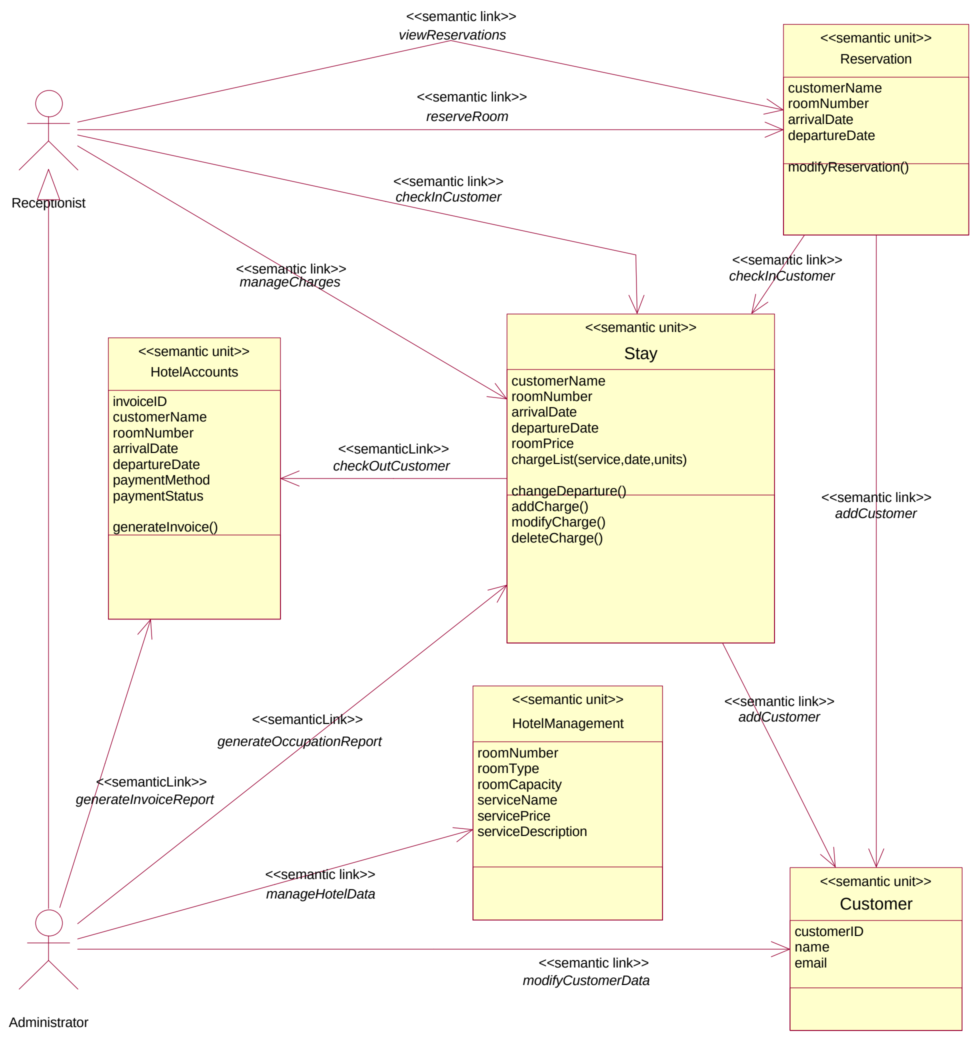

Figure 2. *Hotel Management System: Navigation Analysis Model*

attribute. Also, the way operation calls are performed may differ from one NSU to another, due to the fact that NSU provide a context for the different operation calls. For example, it could happen that certain parameters were implicitly provided by such context in some cases and explicitly introduced by the user in others. Also, the fact that this model is user-independent allows the identification of reusable views among different user types.

NSL on the other hand are depicted as arrows decorated with a ≪*semantic link*≫ stereotype. These links represent a change in the user aim, and so should be independent from designer punctual decisions, such as the inclusion of a new index to further organize the information, the application of certain interface patterns (e.g. addition of a header and a footer to every page) or the pagination of the view. Furthermore, with this concept we are introducing a hierarchy [12] in the importance of the links, which in turn may influence several design features:

– Visually speaking, the definition of a NSL may imply its automatic inclusion in direct access menus during the design phase. They could be seen as functional landmarks, that is, main gates to the functionality offered by the applications. Additionally, special colours, fonts etc. might be defined to set them off from less relevant links.
– Logically speaking, the definition of a NSL may imply a physical navigation step. Otherwise stated, information giving an answer to two different user requirements should not appear in the same 'design page'.
– Also the kind and amount of contextual information that can be transmitted along the link should be limited. If NSL imply a semantic step (a change in the user intention), they should be as light as possible, in order to make this cognitive change easy to be performed.

The way this navigation analysis influences the construction of a navigation design model depends on the proposal under consideration. In order to illustrate this fact, next we are showing the influence that the inclusion of a navigation analysis model causes on two well-known hypermedia methods, namely OO-H (see section 3.2) and UWE (see section 3.3).

3 Navigation Design

Current hypermedia modeling proposals usually depart from any kind of domain model (either complete [6] or with a previous simplification based on *core* concepts [1, 10, 3]) to define the navigation design view of the system under development. In this article, on the contrary, we propose a different approach, based on three artifacts:

– Domain Model
– Analysis Navigation Model
– User Interaction Model

It is commonly avowed that the **the Domain Model** provides very useful information not only to specify the correspondence between user and domain concepts, but also to ease the step towards implementation, including the (possible) application of code generation techniques. In our example, the class diagram corresponding to such model can be seen in Fig. 3. In this example, let's assume that our hotel system is made up of a set of *Rooms* which can be of different *Types*. Rooms can be booked by *Customers*. For each *Reservation* whose status is set to *Registered* (that is, the customer is already in the hotel) the system keeps track of the *Services* provided (laundry, drinks, etc), in order to *Charge* them to the customer. On a customer departure, an *Invoice* is generated and the method of *Payment* is registered.

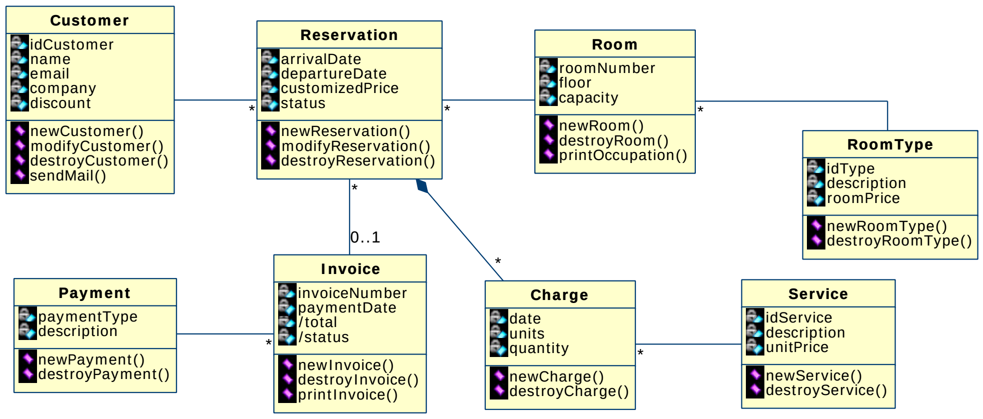

Figure 3. *hotel management System Class Diagram*

However, a navigation model based on a domain model is relatively rigid when faced to new, often unpredictable, use contexts. The reason is that the OO paradigm is specially suited to encapsulate data concerns into classes, but it is not so well suited to represent other concern types, such as business-related or functional. Otherwise stated, a new requirement may cause a reorganization in the class diagram. This fact is called by some authors *the tyranny of the dominant decomposition* [9]. Consequently, if we let the navigation model depend on the domain model, a new requirement may imply not only a change in the domain most suitable structure but also an arbitrary change in the different navigation diagrams, what, according to us, is a clear drawback of current approaches.

On the other hand, we think that the user requirements gathered in use case diagrams such as the one depicted in Fig. 1 are too far away from this navigation view and so fall short to provide (on their own) the designer with the kind of information s/he needs. The analysis model presented in Fig. 2 aims at filling this gap between use cases (too abstract) and domain models (too concrete), thus providing a concern-independent view that at the same time can serve as an starting point from which to derive the different navigation design models. In this sense, one of the main advantages of **the inclusion of an Analysis Navigation Model** such as the one presented in [2] is precisely the achievement of a greater degree of isolation from data structure concerns. This analysis navigation model provides a set of views, each of which encapsulates a subset of concerns, not necessarily data-related. In this way, the addition of a new requirement may imply the extension of the model with a new NSU that, additionally, may introduce a new perspective (concern type) over the system, but will never imply a restructuring process on the existent navigation analysis model.

Also, before stepping forward to the navigation design, and based on both the textual description of the problem, the domain model and the navigation analysis model built so far, we propose the construction of **an interaction model**, whose objective is to represent the action flow (user-system communication) required to fulfill the system functional requirements. The user interaction model is depicted by means of a UML activity diagram which includes an object flow. This model (1) shows the responsibilities and collaboration between the actors of the system, (2) documents the business process involved in the system and, due to the object flow, (3) includes details of the computation. A similar approach, solely based on the information provided by use cases and with a propietary notation, is the user interaction model approach presented in [5].

Part of the user interaction model corresponding to the Hotel Management System, whose navigation design models in both OO-H and UWE are presented in Sections 3.2 and 3.3 respectively, can be seen in Fig. 4. This User Interaction Model drives many navigation design decisions, from which the set of pages that will make up the final interface can be derived.

3.1 Influence of the navigation analysis on existing navigation models

We can think of two ways of using an analysis model during the design navigation process:

1. Used in an indirect way, the designer departs from the domain model to construct the design navigation model, and uses the analysis model to take decisions about certain design model features.
2. Used in a direct way, the designer departs from the navigation analysis model and, getting help from the domain model, defines both further navigation structures (inside the boundaries of each NSU) and how analysis attributes, operations and relationships can be mapped to the underlying domain structure, in order to achieve a full traceability of the models.

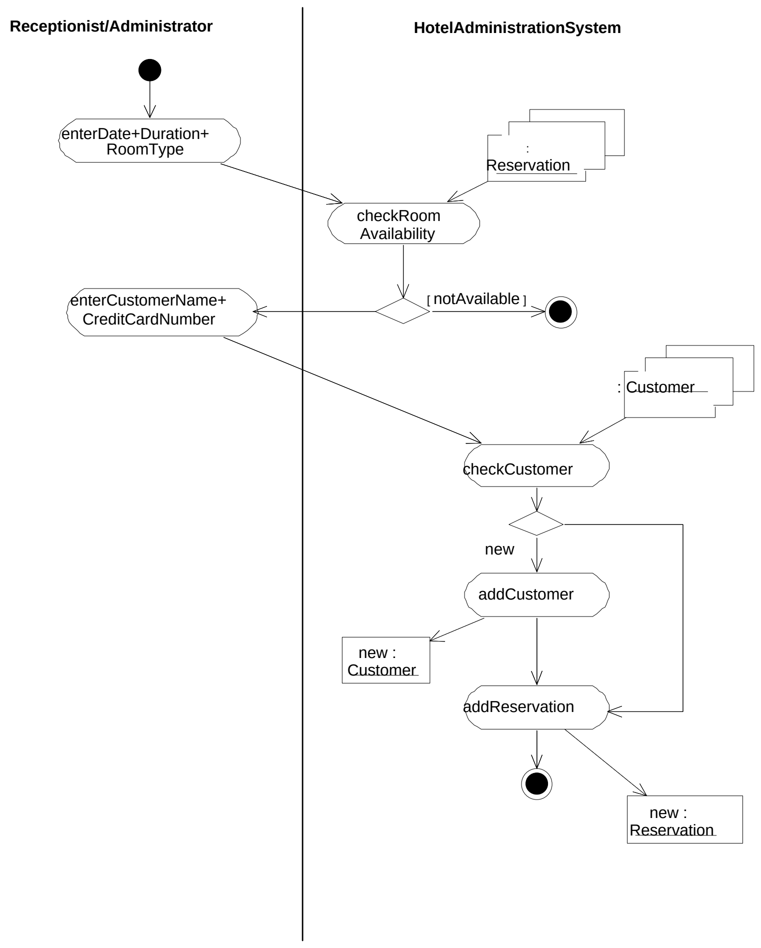

Figure 4. *Hotel Management System: Partial View of the User Interaction Model*

The first approach (followed in OO-H, see Section 3.2), implicitly provides the 'gathering' paths through the domain model that make up the information encapsulated within the boundaries of each NSU. The main disadvantage of this approach is however that diagrams may become quite complex. The second approach (followed in UWE, see Section 3.3) on the contrary, generates very intuitive diagrams, where the domain structure may remain hidden if desired. However, this approach requires the inclusion of a new *mapping activity* where any kind of derivation rule or even new diagrams showing the domain elements collaborating in the definition of each NSU must be defined.

Whatever the case, we believe that the mapping process from navigation structures to domain structures should be dealt with at this design level of abstraction. It is now that the general navigation structure has been defined (a user-oriented *what*) that the equivalence between its elements and underlying domain structures and/or design navigation constructs (a domain-related *how*) should be defined.

Next, the design navigation models corresponding to two well-known hypermedia modeling proposals, OO-H [6] and UWE [8], based on both the class model and the analysis navigation model depicted so far, are shown, together with the philosophy followed to create them. We would like to stress how these diagrams may implicitly include the application flow captured by means of any kind of User Interaction Model, e.g. the one presented in Fig. 4.

3.2 Design navigation modeling in OO-H

The OO-H (Object Oriented Hypermedia) Method [6] is a generic approach, based on the object oriented paradigm, that provides the designer with the semantics and notation necessary for the development of web-based interfaces and its connection with previously existing application logic modules. Although an extensive description of this method is out of the scope of this report, next we are briefly introducing some of the most relevant concepts regarding its navigation design modeling phase. This phase is reflected in OO-H in a set of Navigation Access Diagrams (NAD), one for each actor identified in the system, that are partially based on the class diagram that represents the domain structure of the system. Departing from these OO-H NAD models, and applying default settings, a functional prototype of the application interface can be directly derived, without need to perform a *mapping process* between navigation and information structures.

In order to define the NAD view, the first step consists on defining the different navigation subsystems, known as *Navigation Targets (NT)*, which are directly derived from the navigation analysis performed over the user requirements. In Fig. 5 we can observe the notation and typical structure of this first level of the NAD diagram. An inverted triangle stands for a *collection*, which for the sake of simplicity can be seen in the context of this example as a menu structure. This collection gathers the different entries to the set of defined NT, which correspond to the set of NSU defined at during the navigation analysis phase.

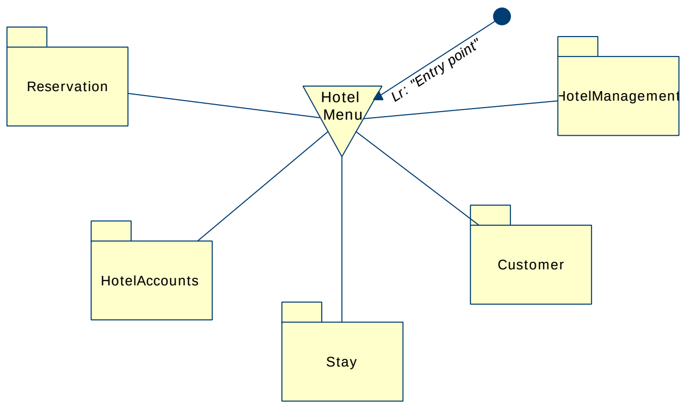

Figure 5. *OO-H NAD level 0 corresponding to the* Hotel Management System

Once this structure has be derived, the designer must tackle, inside each NT, the (possible restricted) views each user type has over the different conceptual classes. These views, known as *Navigational Classes* (NC), may include both attributes and operations, and must be connected by means of *Navigational Links* (NL), which provide the logical navigation paths that give a response to each user requirement. The notation for these constructs can be seen in Fig. 6, where a partial view of the NAD corresponding to the internal navigation of the *Reservation* NSU (and thus defined inside the boundaries of the corresponding *Reservation* NT) is defined. There we can observe how the NC *Reservation*, which is a restricted view on the domain class *Reservation* depicted in Fig. 3, provides information about the arrival and departure dates and the price each customer pays for each room reserved. Also, this NAD includes the definition of the *newReservation()* operation interface, which allows the receptionist to introduce new reservations in the system.

NL in OO-H are categorized according to its purpose. For the sake of our example, three types are especially relevant:

- Requirement Links (Lr) establish the point from where the user begins to navigate inside each NT. They are depicted by means of an arrow with a filled circle. In Fig. 6 the arrow labeled *View Reservation Info*, that points to the *Reservation Info* NC, belongs to this type.
- Internal links (Li) connect information constructs inside a given NT. In Fig. 6 we observe how the *Reservation Info* NC is the source of two Li (*room* and

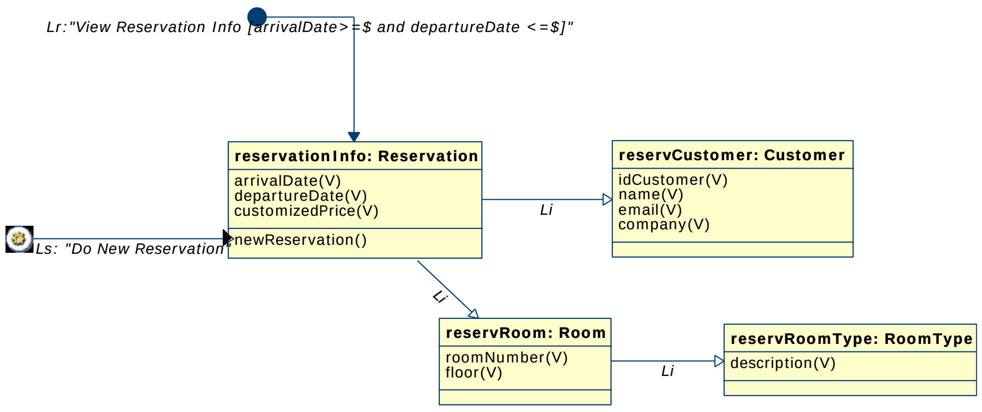

Figure 6. *Partial view OO-H NAD level 1 corresponding to the* Reservation *NSU*

customer) that relate the general information regarding each reservation with information regarding the specific room and customer involved in that reservation instance.

– Last, Operation Links (Ls) provide the interface to the operations that a given actor is allowed to invoke. Ls are depicted by arrows with a serrated wheel on one of its ends. Back to our example, in Fig. 6 we can observe how, inside the *Reservation* NT, the receptionist has been granted access to the *newReservation()* operation.

Also in our example we can observe how NL may have, among other characteristics, one or more *filters* associated. Filters are OCL formulae [11] that constrain navigation. In order to illustrate this concept, in Fig. 6 the filter *arrivalDate >= $ and departureDate <= $*, where the $ symbol stands for user input, has been associated to the Lr *View Reservation Info*. This filter restricts the set of target objects to those reservations that are inside the boundaries of a given period.

It is important to stress the fact that the navigation design decisions influence the number and type of pages that will finally make up the user interface[1]. The separation of attributes and operation calls among pages is performed by means of a *Show-in* tagged value associated to links. This tagged value can be set to *origin* or to *destination*. NL with *Show-in=origin* (depicted by means of a hollow arrow head, see Fig. 6) gather in a single page the information contained in the origin and target NC. On the contrary, NL with *Show-in=destination* (depicted as a filled arrow head, see Fig. 6) cause a new page to appear in the interface.

[1] These pages can be later refined and integrated in physical views during the OO-H presentation design modeling phase.

In our example, the two *destination* NL (namely the Lr *View Reservation Info* and the Ls *do new Reservation()*) and their associated filters bring about three design pages. The first one (see Fig. 7) shows a form that asks for the user-dependent values that are used in the filter associated to the Lr *View Reservation Info*. Once the dates have been entered, the second page (see Fig. 7) shows a *Reservation* list that contains the objects that fulfill the constraint imposed by the filter. Each reservation object is presented together with the associated *room* and *customer* information. This fact has been modeled by two Li with the tagged value *Show-in=origin*. In the *View Reservation Info* view we can also observe how the *Show-in=destination* Li cause an anchor to appear on the page. This anchor is used in our example to navigate to the *newReservation()* operation parameter input page, whose definition falls out of the scope of this article.

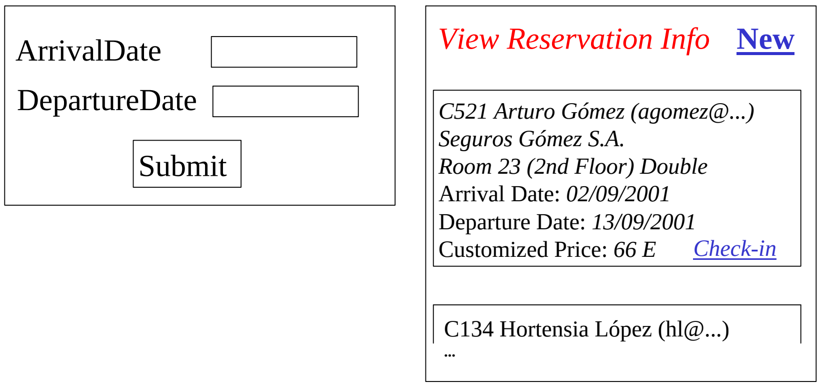

Figure 7. *Partial Storyboard for the Reservation NSU*

Navigation analysis influence in OO-H As previously stated, OO-H is mainly based on the domain model. Up to now, it gets help from the navigation analysis model in order to adopt certain design decisions such as the top level interface structure. Also, the inclusion of a navigation analysis model such as that depicted in Fig. 2 causes a set of restrictions to occur on design navigation models. For the sake of this article, the most relevant is that NAD design links corresponding to NSL (in our example the Lr *View Reservation Info* and the Ls *do new Reservation*) must be set to *Show-in=destination* and *Activation=manual* (meaning that the user must explicitly activate the corresponding implementation navigation construct). Furthermore, the parallelism drawn between design links and NSL provide a powerful mechanism to make sure the final interface is complete, that is, provides at least one way to answer every user requirement.

3.3 Design Navigation Modeling in UWE

Regarding UWE (UML-based Web Engineering), and before focusing on its navigation modeling activity, next we are presenting an overview of the complete UWE design process. This overview provides a better understanding of how introducing a navigation analysis model influences the design of Web applications in UWE [7, 8]. We illustrate the UWE design method using the hotel management example defined in the previous sections.

UWE defines an iterative UML-based process for Web applications that includes the following design steps: requirements analysis, conceptual, navigation, presentation and task design and which produce the following artifacts: a use case model and a conceptual, a navigation, a presentation and a task model. Fig. 8 depicts these models as UML packages and shows how the models are related representing them by UML trace dependencies.

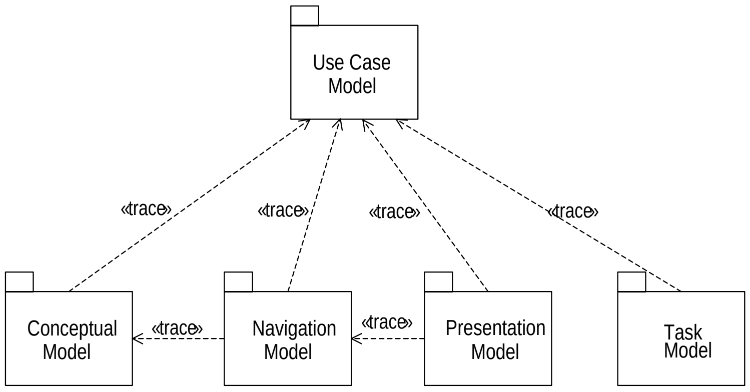

Figure 8. Overview of the UWE Design Process

The use case model describing the functional requirements of the *Hotel Management* Web application (see Fig. 1) is the source of information for the conceptual model (see Fig. 3) as well as for the navigation, presentation and task model. If we add the construction of a navigation analysis model previous to build the navigation design model, the UWE process will change as the navigation design model will then be based both on the navigation semantic units and semantic links identified during navigation analysis (Fig. 2) and, as before, on the classes of the conceptual model (Fig. 3). Note that the class diagram shown in Fig. 8 will not change as both the navigation analysis model and the navigation design model are part of the package navigation model.

UWE proposes special modeling elements and a set of guidelines to construct a navigation model based on both the conceptual model and the navigation analysis model following the indirect way of using the navigation analysis model, i.e. departing from the domain model to construct the navigation design model and enhancing it with modeling features obtained from the navigation analysis model. A first result in navigation modeling is a representation of the *navigation space* which consists of *navigation classes* and associations we call *direct navigability* (links). The navigation classes are defined as classes whose instances are visited by the user during navigation. The direct navigability links represent direct access from the source navigation class to the target navigation class. Navigation classes will be given the same name as the corresponding conceptual classes. For the representation of these modeling elements UWE uses the UML stereotypes ≪*navigation class*≫ and ≪*direct navigability*≫. Navigation classes contain derived attributes (denoted by a slash *(/)* before its name). For example, the attribute *paymentType* included in the *class Invoice* is derived from the conceptual class *Payment* which is not included in the navigation model. The UWE navigation space model is shown in Fig. 9. The formulas to compute the derived attributes can be given by OCL expressions.

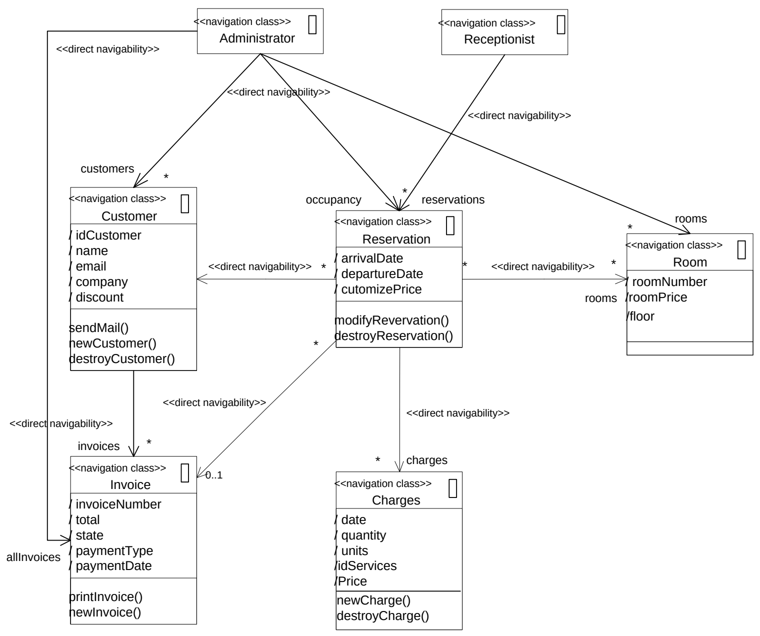

Figure 9. *Hotel Management* Application: First step in the construction of the UWE Navigation Design Model

There is obviously no way to fully automate the construction of the navigation space model, but UWE provides a set of guidelines that can be followed by the designer:

1. Include those classes of the conceptual model that are relevant for the navigation as navigation classes (i.e. navigation classes can be mapped to conceptual classes). If a conceptual class is not a visiting target in the use case model, it is irrelevant in the navigation process and therefore omitted in the navigation space model, such as the classes *Service* and *Payment* in this example.
2. Add from the navigation analysis model, if necessary and to allow starting navigation, navigation classes for the actors of the Web application. In our example, we include classes *Receptionist* and *Administrator*.
3. Keep information of the omitted classes (if required) as attributes of other classes in the navigation design model (e.g. the newly introduced attributes *paymentTypes* and *paymentDate* of the navigation class *Invoice*). All other attributes of navigation classes map directly to attributes of the corresponding conceptual class.
4. Associations of the conceptual model are kept in the navigation model.
5. Add associations between classes that are taken over from the navigation analysis model and starting navigation classes, usually as many associations as semantic links depart from these actors in the navigation analysis model.
6. Additional associations can be added for direct navigation to avoid navigation paths of length greater than one. Examples are the newly introduced navigation associations between *Customer* and *Invoice* (see Fig. 9).
7. Add additional associations based on the semantic links of the navigation analysis model.
8. Add constraints to specify restrictions in the navigation space, e.g. invariants to specify that navigation from *Reservation* to *Invoice* should only will possible if the *Customer* has checked out.

In successive steps, and following a systematic (partially automated) procedure, the navigation design model built so far is enhanced with access elements, such as indexes, guided tours and menus. The result of these further steps in navigation modeling is the navigation structure model. These access primitives introduce design decisions of the developer, such as whether the navigation should be supported by a guided tour or by an index.

The UWE method includes additional UML stereotyped modeling elements first proposed in [1], namely *≪index≫, ≪guided tour≫, ≪query≫ and ≪menu≫*. They are used in the construction of UML class diagrams to represent the navigation structure model. The resulting UWE navigation design model, built with these modeling elements and based on the model shown in (Fig. 9), is depicted in Fig. 10. Note that we do not included the navigation class *Receptionist* and the corresponding *ReceptionistMenu* in the diagram visualized in Fig. 10, as the only difference is that the menu includes only the two last menu items. Most of the steps described in the above method can be performed in a fully automatic way. As a result we obtain a comprehensive navigation design model of the application.

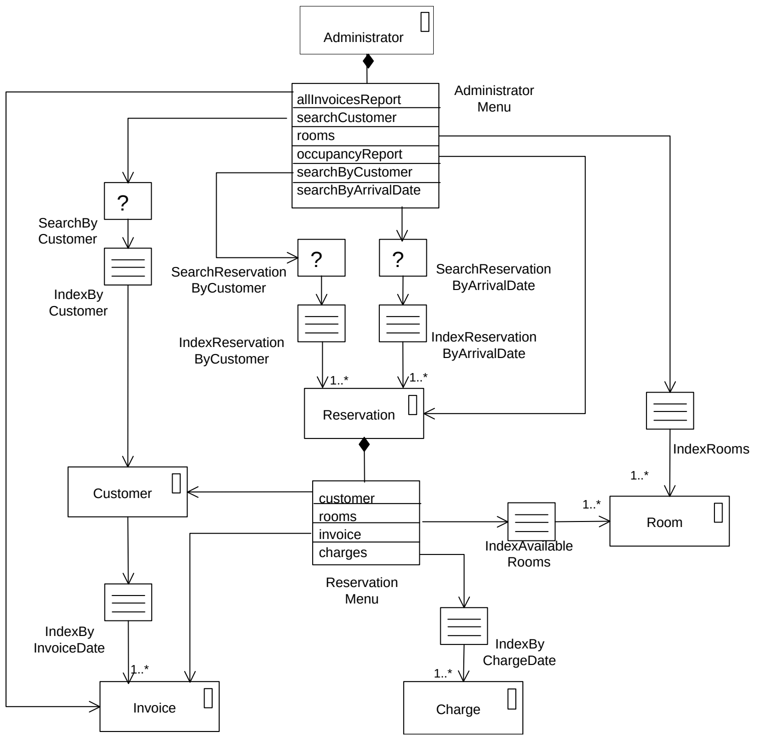

Figure 10. UWE Navigation Structure Model of the *hotel management* Application

1. Add one or more access primitives to access the starting point of the application. In our example, we add queries *SearchByCustomer* and *SearchByArrivalDate* with corresponding indexes *IndexReservationByCustomer* and *IndexReservationByArrivalDate*.
2. Replace all bi-directional associations, which have multiplicity greater than one at both associations ends by two corresponding unidirectional associations.
3. Consider only those associations of the navigation space model, which have multiplicity greater than one at the directed association end. For each association of this kind, choose one or more access primitives to realise the navigation. In our example, e.g. for the association with role *charges* we add an index *IndexByChargeDate*.
4. Enhance the navigation space model correspondingly. Role names of navigation in the navigation space model are now moved to the access elements.

5. Introduce a main menu, necessary if more than one access primitive is used to start the application.
6. Associate to each navigation class, which has at least one outgoing association, a corresponding menu class. The association between a navigation class and its corresponding menu class is a composition. In our example, the navigation class that requires a menu is the class Reservation.
7. Introduce for each role, which occurs in the previous model at the end of a directed association a corresponding menu item, such as *customer*, *rooms*, *invoice* and *charges*. By default, the role name is used as the constant name of the menu item.
8. Any association of the previous model which has as its source a navigation class now becomes an association of the corresponding menu item introduced in step 5.

Navigation analysis influence in UWE Note that if we change the UWE process to include navigation analysis modeling this will only influence the construction of the first steps of the navigation design model, i.e. the second group of guidelines given for the modeling of the navigation design model will remain unchanged.

In the UWE process without a navigation analysis model, the navigation space model (result of the first steps in the navigation design) has some similarities with the navigation analysis model. Both are based in the identification of nodes and links. Both introduce two concepts and define two stereotypes for class and association, respectively. The concepts used in the navigation analysis model are the navigation semantic unit and semantic link, while the concepts used in the navigation space model are the navigation class and the direct navigability. The difference is less support for the designer when he/she takes decisions based on the use case model instead of on the navigation analysis model.

4 Conclusions

The aim of this article is to analyse what consequences the construction of a navigation analysis model, such as defined in [2] has to existing hypermedia (Web) design methods. In order to achieve that goal, we have described the inclusion process for two well known methods, namely OO-H [6] and UWE [8]. We hope that our proposal serves as starting point of a fruitful discussion on advantages and limitations of the inclusion of a navigation analysis model in the context of other Web development methods.

References

[1] Hubert Baumeister, Nora Koch, and Luis Mandel. Towards a UML extension for hypermedia design. In *UML´99 The Unified Modeling Language - Beyond the Standard, LNCS 1723*, Fort Collins,USA, 1999. Springer Verlag.

[2] C. Cachero, N. Koch, J. Gómez, and O. Pastor. Conceptual Navigation Analysis: a device and platform independent navigation specification. In *Sent to the Second International Workshop on Web-Oriented Software Technology IWWOST'02*. ECOOP Workshop Proceedings, 06 2002.

[3] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites WWW9 Conference. In *First ICSE Workshop on Web Engineering, International Conference on Software Engineering*, 05 2000.

[4] F. Garzotto and P. Paolini. HDM A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems (TOIS)*, 11(1):1–26, 01 1993.

[5] Natacha Güell, Daniel Schwabe, and Patricia Vilain. Modeling Interactions and Navigation in Web Applications. In *Proc. ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling*, pages 115–127. Springer, 10 2000.

[6] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, pages 26–39, 04 2001.

[7] Nora Koch. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. UNI-DRUCK Verlag, 2001. PhD. Thesis, Ludwig-Maximilians-Universität München.

[8] Nora Koch, Andreas Kraus, and Rolf Hennicker. The Authoring Process of the UML-based Web Engineering Approach. In Daniel Schwabe, editor, *First International Workshop on Web-oriented Software Technology*, June 2001. To appear.

[9] H. Osser and P. Tarr. Using multidimensional separation of concerns to (re)shape evolving software. In *Communications of the ACM*, volume 44, pages 43–50, 10 2001.

[10] D. Schwabe, L. Esmeraldo, G. Rossi, and F. Lyardet. Engineering Web Applications for Reuse. *IEEE Multimedia. Special Issue on Web Engineering*, pages 20–31, 01-03 2001.

[11] OMG Unified Modelling Language Specification. http://www.rational.com/uml/, 06 1999.

[12] J. Yoo and M. Bieber. Towards a Relationship Navigation Analysis. In *Prooceedings of the 33rd Hawaii International Conference on System Sciences*, 01 2000.