Using ORM to model Web Systems

Olga De Troyer[1], Sven Casteleyn[1], Peter Plessers[1]

[1]Vrije Universiteit Brussel, WISE, Pleinlaan 2,
1050 Brussel, Belgium
{Olga.DeTroyer, Sven.Casteleyn, Peter.Plessers}@vub.ac.be
http://wise.vub.ac.be

Abstract. In this paper, we describe how ORM is extended, and combined with Concurrent Task Trees (CTT) to model the content as well as the functionality of a web system in the web design method WSDM. As WSDM uses an audience driven design approach, the use of ORM is somewhat different from its use for data modeling in the context of databases. We discuss the differences. We also discuss the benefits of using ORM for our purpose, modeling web systems using an audience driven approach.

1 Introduction

In the last years, web sites have evolved from a simple collection of hypertext pages towards large web applications supporting both static and dynamic content and complex (business) processes. Although it is still easy to publish a couple of pages, more and more it is recognized that appropriate design methods are needed to develop large and complex web sites. In the past, web sites were created opportunistically without any regard for methodology. The increased complexity soon lead to usability problems with sites developed in an ad-hoc manner [1]: the information is badly organized, the navigation structure is non-transparent and difficult to grasp for visitors and the look & feel may be inconsistent. Consequently, users fail to make a comprehensive mental model of the website. Furthermore, maintainability, evolution and extensibility of these web sites are highly problematic. Therefore, in the past, a number of web design methods have been proposed. Examples include WSDM [2], WebML [3], OOHDM [4], OO-H [5].

Most of the existing web design methods recognize that in order to design a web system[1] due consideration should be given to the following aspects: content, functionality, navigation and presentation. To model the content of a web system, the methods usually rely on an existing data modeling technique, e.g., WebML uses the ER model, OOHDM as well as OO-H applies the class diagrams of UML. Recent methods like HERA [6] uses semantic web technology (e.g., RDF (S)). As far as we are aware, WSDM is the only web design method that uses ORM to model the content of a web system. To model functionality, WebML provides operation units that can be associated to data entry units to process information entered by a user, and

[1] We will use the term web system to indicate web sites as well as web applications

predefined units for the standard data management functionality. Also a workflow data model (composed of processes, activities, etc) can be defined. OOHDM uses UML's user interaction diagrams to model the interaction with the user. In Hera, the modeling of user interaction is based on forms and queries. Forms are used to retrieve information from the user and are based on the XForms standard; queries are used to update and retrieve data and are expressed in SeRQL. To model functionality, WSDM combines an extended form of ORM with Concurrent Task Trees [7] (CTT).

The purpose of this paper is to explain how ORM is used in WSDM to model the content of the web system and how it has been extended and combined with CTT to allow for the modeling of web functionality. We suppose that the reader is familiar with ORM. The rest of the paper is organized as follows. Section 2 provides an overview of the different phases of WSDM. Section 3 elaborates on the use of ORM to model content and functionality. In Section 4 we discuss the differences with ORM as data modeling method for databases and provide conclusions.

2 WSDM Overview

The web site design method WSDM follows the *audience driven* design philosophy. This means that the different target audiences and their requirements are taken as starting point for the design and that the main structure of the web system is derived from this. Concretely, this will result in different navigation path (called tracks) for different kinds of visitors in the organizational structure of the web system.

WSDM is composed of a number of phases. In the first phase the mission statement for the web system is formulated. The goal is to identify the purpose of the web system, as well as the subject and the target users. The *mission statement* is formulated in natural language and must contain the elements mentioned. The mission statement establishes the borders of the design process. It allows (in the following phases) to decide what information or functionality to include or not, how to structure it and how to present it.

The next phase is the 'Audience Modeling' phase. The target users identified in the mission statement should be refined into so-called *audience classes*. The different types of users are identified and classified into audience classes. How this is done can be find in [8]. The classification is based on the requirements (information-, as well as functional-, and usability requirements) of the different users. Users with the same information and functional requirements become members of the same audience class. Users with additional requirements form audience subclasses. In this way a hierarchy of audience classes is constructed. For each audience class relevant characteristics (e.g., age, experience level) are given.

The next phase, the 'Conceptual Design' is used to specify the content, functionality and structure of the web system at a conceptual level. A conceptual design makes an abstraction from any implementation or target platform. The content and functionality are defined during the 'Task Modeling' sub phase. This will be described in detail in the next section. The overall conceptual structure including the navigational possibilities for each audience class is defined during the 'Navigational Design' sub phase. We will not elaborate on this sub phase.

During the 'Implementation Design' phase, the conceptual design models are completed with information required for the actual implementation. It consists of three sub phases: 'Site Structure Design', 'Presentation Design' and 'Data Source Mapping'. During Site Structure Design, the conceptual structure of the web system is mapped onto pages, i.e. it is decided which components (representing information and functionality) and links will be grouped onto web pages. For the same Conceptual Design, different site structures can be defined, targeting different devices, contexts or platforms. The Presentation Design is used to define the look and feel of the web system as well as the layout of the pages. The 'Data Source Mapping' is only needed for data-intensive web sites. In case the data will be maintained in a database, a database schema is constructed (or an existing one can be used) and the mapping between the conceptual data model and the actual data source is created.

The last phase is the 'Implementation'. The actual implementation can be generated automatically from the information collected during the previous phases. As proof-of-concept, a transformation pipeline (using XSLT) has been implemented, which takes as input the different models and generates the actual implementation for the chosen platform and implementation language. This transformation is performed fully automatically. An overview of this transformation pipeline is given in [9].

3 Modeling Content and Functionality in WSDM

During Audience Modeling, the information-, functional-, and usability requirements of the potential visitors are identified and different Audience Classes are defined. The goal of the Conceptual Design is to turn these (informal) requirements into high level, formal descriptions which can be used later on to generate (automatically) the web system.

During conceptual design, we concentrate on the **conceptual** "what and how" rather than on the visual "what and how". The conceptual "what" (content and functionality) is covered by the Task Modeling step, the conceptual "how" (conceptual structure and navigation possibilities) by the Navigational Design.

Instead of starting with an overall conceptual data model, like most web design methods do, WSDM starts with analyzing the requirements of the different audience classes. This will result in a number of tiny conceptual schemas, called *Object Chunks*. Later on these conceptual schemas are integrated into an overall conceptual data model [10]. This approach has several advantages:

- It allows the developer concentrating on the needs of the actual users rather than on the information (and functionality) already available in the organization (which is not necessarily the information needed by the users. In addition the way the information is organized and structured in the organization is not necessarily the way external users need it).
- It gives consideration to the fact that different types of users may have different requirements: different information or functional requirements; but they may also require a different structure or terminology for the information. By modeling the requirements for each audience class separately we can give due consideration to

this. E.g., we can avoid that the user is overloaded with information of which most is not relevant to him.

Analyzing and modeling the information and functionality needed for the different audience classes is done in the Task Modeling phase.

3.1 Task Modeling

The tasks the members of an audience class need to be able to perform are based on their requirements (informally) formulated during audience classification. To come to detailed task models, a task analysis is done. I.e. for each information and functional requirement formulated for an audience class, a task is defined. Each task is modeled into more details using an adapted version of the task modeling technique CTT [11] (called CTT+). This has been described into more detail in [7].

CTT was developed in the context of Human-Computer Interaction to describe user activities. CTT looks like hierarchical task decomposition but it distinguishes four different categories of tasks (user tasks, application tasks, interaction tasks, and abstract tasks) and it also allows expressing temporal relationships among tasks. In addition, CTT has an easy to grasp graphical notation. For its use in WSDM, we slightly adopted the technique to better satisfy the particularities of the Web:

1. WSDM do not consider user tasks. User tasks are tasks performed by the user without using the application (such as thinking on or deciding about a strategy). They are not useful to consider at this stage of the design. We only use:
   - *Application tasks*: tasks executed by the application. Application tasks can supply information to the user, perform some calculations or updates.
   - *Interaction tasks*: tasks performed by the user by interaction with the system.
   - *Abstract tasks*: tasks that consists of complex activities, and thus requiring decomposition into sub tasks.
2. A (complex) task is decomposed into (sub)tasks. Tasks are identified by means of their name. The same task can be used in different sub-trees, and a task may be re-used inside one of its sub-trees (expressing recursion). CTT prescribes that if the children of a task are of different categories then the parent task must be an abstract task. WSDM do not follow this rule. We use the category of the task to explicitly indicate who will be in charge of performing the task. The convention is that for an interaction task, the user will be in charge; for an application task the application will be in charge.
3. CTT has a number of operators (with corresponding graphical notation) to express temporal relations among tasks. E.g., tasks can be *order independent, concurrent* (with or without information exchange); tasks may exclude each other (*choice*); one task may *enable* (with or without information passing) or *deactivate* another task; a task can be *suspended* to perform another task and *resumed* when this last task is completed; a task can be *iterated* (a defined or undefined number of times); and a task can be *optional* or mandatory. An extra operator for dealing with *transactions* has been added.

We illustrate this task modeling process by means of an example. The example is taken from the Conference Review System case [12], which aimed at providing a web

system for supporting the paper review process of a conference. We start with the following requirement formulated for the 'Author' Audience Class:

"*An author must be able to consult the information about its submissions*"

For this requirement, the following (abstract) task is defined: "*Find Information about Author's Submissions"*. This task can be decomposed into two more elementary tasks (see figure 1). The two subtasks are informally described as follows:

1. *Find Author's Submissions*: Give paper-id and title of all papers submitted by the author as main author or as co-author and allow the user to select a paper. This is an interaction task because the task requires interaction with the user.

2. *Give Submission's Information*: For a selected paper give paper-id, title, abstract, main author (name) and co-authors (names), track and if available the subjects, and file (url) containing the paper. This is an application type of task.

The temporal relationship expressed between the two sub tasks (symbol []>> in figure 1) expresses that finishing the task *Find Author's Submissions* enables the task *Give Submission's Information* and some information must be passed from the first to the second task (i.e. the selected submission).

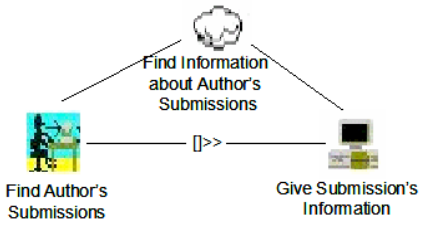

Fig. 1. Task Model for the task "Find Information about Author's Submissions"

The task models created in this way allow for a first level of description of the functionality to be provided by the web system (i.e. they describe a kind of workflow). The elementary tasks will be further specified by creating object chunks, which are tiny conceptual schemas (see next section). In the following sub phase the task models and the temporal relationships are used to derive the navigational model, which expresses the navigational possibilities for the different audience classes.

3.2 Detailed Information and Functional Modeling

When a task model is completed, for each elementary task an object model, called *Object Chunk*, is created that (formally) models the necessary information and functionality needed for this elementary task. For this an extended version of ORM is used. Figure 2 shows the Object Chunk for the task *Find Author's Submissions* of Figure 1. We will explain it later on.

The main purpose of an Object Chunk is to represent the information needed by the user when he has to perform the associated task. If the requirement is a pure information requirement (i.e. the user is only looking for information; he doesn't need to perform actions) then an Object Chunk can be considered as a conceptual

description of the information that will be displayed on (a part of) the screen. For this purpose standard ORM is sufficient. However, to be able to deal with functionality (e.g., filling in a form), it was necessary to extend ORM:

1. To express functionality we need to be able to refer to instances of Object Types. For this we use *referents*. Graphically, a referent is placed inside the circle depicting its Object Type. For Value Types, instances (and therefore also referents) are values, e.g., 'Casteleyn' is an instance of the *PersonName,* 1 is an instance of *Rate*. A *generic marker* is used to represent an instance of an Entity Type, e.g., **a* in figure 2 represents an instance of type *Author*. Sets of referents are represented using the traditional set brackets '{' and '}', e.g., {*p} placed in the Object Type *Paper* would represents a set of Paper instances. The traditional set operators (union, intersection, etc.) can be used on sets. The notation {…}@ indicates the cardinality of the set.
   To represent a relationship between instances, we can place the referents either in the Object Types (if no confusion is possible) or in the boxes of the roles of the relationship.
2. To allow communication between tasks, parameters (input- as well as output parameters) are specified for Object Chunks. E.g., the Object Chunk *AuthorSubmission* (figure 2) has an instance of type Author as input parameter (represented by the referent *a) and an instance of type Paper as output parameter (represented by the referent *p). Input parameters usually restrict the information that should be presented to the user. E.g., the input parameter *a of type Author, is used to express that only the information related to this particular author should be shown. This is indicated by putting this parameter in the corresponding Object Type. In this way only the Paper-instances 'with main' Author equal to *a, and the Paper-instances 'with co' Author equal to *a will be considered (and displayed on the corresponding page of the actual web system). In fact, the use of the parameter *a in the Object Type Author restricts the complete schema to a kind of view. The fact that for the Object Type Paper two Value Types (PaperTitle and PaperId) are included indicates that the relevant Paper-instances should be shown by means of their paper title and paper id.
3. To model that some information must be selectable (e.g., papers should be selectable to ask for the details of a paper; or when filling in the review form, it must be possible to select the rate from a given list) a selection operator is used, represented by the symbol '!'. E.g., !{*p} expresses that the user can select an instance from the set {*p}. The symbol '!!' is used to indicate that more than one instance is selectable.
   The selection symbol is placed inside the circle depicting an Object Type when an instance can be selected from the (possibly restricted) population of this Object Type. Similar, to indicate that it is possible to select an instance from the (possibly restricted) population of an Object Type role, the selection symbol is placed inside the box of this role. E.g., in figure 2, the '!' in the roles 'with main' and 'with co' indicates that the user is able to select a Paper-instance from the set of Paper-instances 'with main' Author equal to *a and from the set of Paper-instances 'with co' Author equal to *a. Intuitively, this means that the user can select a paper from all papers submitted by the author *a either as main author or as co-author.

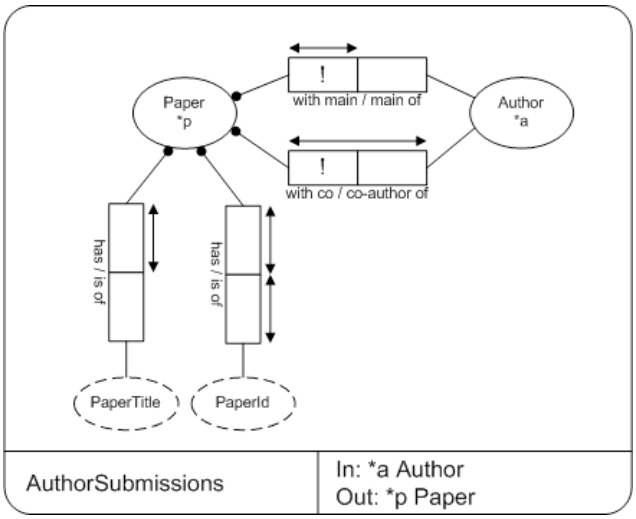

Fig. 2. Object Chunk *AuthorSubmissions*

4. To express interactive input (e.g., needed to fill in a form) a principle similar to that of the selection is used. The symbol '?' is used for the input of a single value, '??' is used for expressing interactive input of more then one value. Note that these symbols can only be applied to Value Types. Entity Type instances cannot be entered directly. They should be created by the system using the NEW operator (see further on).
5. To assign values to referents the assignment operator ('=') is used, e.g., in figure 3, which shows the Object Chunk for a task *Register New Paper*. **t = ?* in the Value Type PaperTitle indicates that the user should enter the paper title, which is then assigned to the referent *t.
   The value of a referent can be changed by means of the operator '→'. *t → ? in the Value Type PaperTitle would allow the user to change the given paper title (represented by *t) by entering a new title; {*a} → Ø will replace the value of the referent set {*a} by the empty set.
6. To manipulate the data itself, a number of primitives are available:
   - 'NEW' indicates the creation of a new Object Type instance. E.g., in figure 3, *p = NEW in the Object Type Paper indicates the creation of a new Paper instance, which is assigned to the referent *p;
   - 'REMOVE' is used to indicate the removal of one or more instances from an Object Type;
   - '+' above a relation indicates the addition of a relationship. In figure 3, these are used to specify that the relationships (*p, *a), (*p, *t) and (*p, *id) should be added.
   - '-'above a relation indicates the removal of a relationship;
7. Furthermore, we have:
   - '==' for testing equality of values;
   - 'IS' for testing membership of an Object Type;
   - 'EXIST' to test the existence of an instance;

- **‘**NEXT’ to generate a unique system identifier (used in figure 3 to generate the following paper id);
- ‘TODAY’ represents the current date;
- ‘UPLOAD’ and ‘EMAIL’ are supposed to be built-in functions.

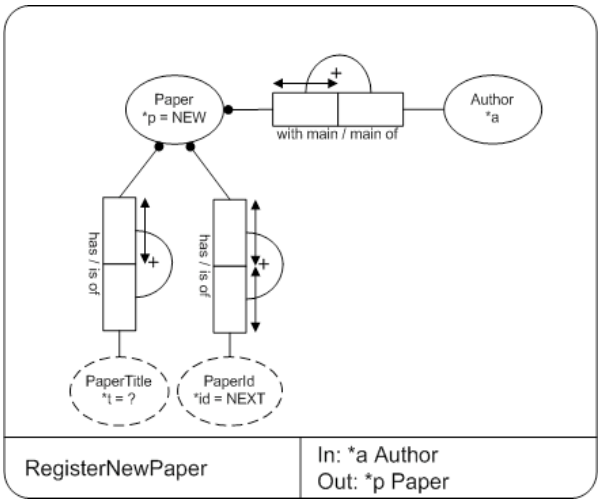

Fig. 3. Object Chunk *RegisterNewPaper*

3.3 The Business Information Model

The disadvantage of modeling the requirements of the different audience classes by means of separated Object Chunks is that they need to be related to each other and possibly also integrated. Indeed, different chunks may deal with the same information. For example, Authors as well as PC-Members need information about papers. This means that different Object Chunks may model (partially) the same information. This redundancy is deliberately (the same information may need different representations or different navigation paths for different audience classes) and will not cause any problems as long as we recognize that it is indeed the same information and that it is maintained in a single place. Therefore, the different chunks are related to each other. In addition, an integrated schema may be useful if one wishes to maintain the information in a database. In that case, such an integrated schema, called *Business Information Model*, will be the conceptual schema of this database. It is possible that the web system will use an existing database. In this case the different chunks need to be defined as views on the schema of this database. How this should be done is outside the scope of this paper. In [13] we have proposed an approach to link (already during modeling) the concepts used in the different Object Chunks to an ontology. In this way the integration can be done in an automatic way and it has the additional advantage that it paves the way for semantic annotations.

The mapping of the Object Chunks (or the Business Information Model) onto the actual data source(s) is considered in the Data Source Mapping step of the Implementation Design.

4 Discussion and Conclusions

In this paper, we have presented how ORM is used in the web design method WSDM to model the content and the functionality of a web system. For this purpose, ORM was extended with referents to be able to refer to instances, and with a number of operators for modeling selection of information, interactive input and data manipulation. Workflow is captured by means of task models, expressed in an extended version of CTT. The task models and the ORM descriptions are linked by defining an Object Chunk for each elementary task in a task model.

Next to the capability to define functionality, there are a number of fundamental differences in how ORM is used in WSDM, compared with the use of ORM as data modeling technique for databases:

- ORM is used to model the information requirements of a single task. This results in tiny conceptual schemas, called Object Chunks, comparable to views in databases. Later on the different Object Chunks can be integrated into an overall data model.
- When using ORM for data modeling, constraints are specified to allow checking the integrity of the database and are valid for the entire model. This is not the case for our Object Chunks. Because chunks are used to model requirements in the context of certain task and for a particular audience class, the constraints in a chunk only express constraints formulated or applicable in that context. These are not necessarily the constraints that apply in general for the information considered. E.g., in general a paper has different reviews, but for the task of reviewing papers for the audience class PC Member, a paper only has one review.

In the early days of WSDM, class diagrams were used to model content [2]. However, very quickly it turns out that these diagrams were not suitable for the use in Object Chunks. People always tended to put all possible attributes in the classes, even if there were not needed for the purpose of the task at hand. With ORM, in which all relationships are modeled at the same level, we didn't have this problem. Also, the integration of the Object Chunks turned out to be simpler with ORM.

5 References

1. Nielsen, J.: Designing Web Usability: The Practice of Simplicity. New Riders Publishing (2000)
2. De Troyer, O., Leune, C.: WSDM: A User-Centered Design Method for Web Sites. In Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference, Publ. Elsevier, Brisbane, Australia (1998) 85-94

3. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML) : a modeling language for designing Web sites. In Proceedings of the ninth World Wide Web Conference (2000)
4. Schwabe, D., Rossi, G., Barbosa, S.: Systematic Hypermedia Application Design with OOHDM. In Proceedings of ACM Hypertext'96 Conference (1996) 116-128
5. Gómez, J., Cachero, C., Pastor, O.: Conceptual Modelling of Device-Independent Web Applications. IEEE Multimedia Special Issue on Web Engineering (2001) 26-39
6. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. Journal of Web Engineering (JWE), Vol. 2, Nr. 1-2, Rinton Press (2003) 3-26
7. De Troyer, O., Casteleyn, S.: Modeling Complex Processes for Web Applications using WSDM. In Proceedings of the Third International Workshop on Web-Oriented Software Technologies (held in conjunction with ICWE2003), IWWOST2003 (also on http://www.dsic.upv.es/~west/iwwost03/articles.htm), Eds. Daniel Schwabe, Oscar Pastor, Gustavo Rossi, Luis Olsina, Oviedo, Spain (2003)
8. Casteleyn, S., De Troyer, O.: Structuring Web Sites Using Audience Class Hierarchies. In Conceptual Modeling for New Information Systems Technologies, ER 2001 Workshops, HUMACS, DASWIS, ECOMO, and DAMA, Lecture Notes in Computer Science , Vol. 2465, Publ. Springer-Verlag, ISBN 3-540-44-122-0, Yokohama, Japan (2001)
9. Plessers, P., Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., Goble, C.: Accessibility: A Web Engineering Approach. In Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan (2005)
10. De Troyer, O., Plessers, P., Casteleyn, S.: Conceptual View Integration for Audience Driven Web Design. In CD-ROM Proceedings of the WWW2003 Conference, IW3C2 (also http://www2003.org/cdrom/html/poster/), Budapest, Hongary (2003)
11. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Eds. Ray, P., Thomas, P., Kuljis, J., Springer-Verlag Londen Berlin Heidelberg, ISBN 1-85233-155-0, 2000
12. De Troyer, O., Casteleyn, S.: The Conference Review System with WSDM. In First International Workshop on Web-Oriented Software Technology, IWWOST'01, (also http://www.dsic.upv.es/~west2001/iwwost01/), Eds. Oscar Pastor, Valencia (University of Technology), Spain (2001)
13. De Troyer, O., Plessers, P., Casteleyn, S.: Solving Semantic Conflicts in Adience Driven Web Design. In Proceedings of the WWW/Internet 2003 Conference (ICWI 2003), Vol. I, Eds. Pedro Isaías and Nitya Karmakar, Publ. IADIS Press, ISBN 972-98947-1-X, Algarve, Portugal (2003) 443-450