Modeling Complex Processes for Web Applications using WSDM

Olga De Troyer, Sven Casteleyn
Vrije Universiteit Brussel
WISE Research group
Pleinlaan 2, B-1050 Brussel, Belgium
Olga.DeTroyer@vub.ac.be, Sven.Casteleyn@vub.ac.be

Abstract
We present the latest version of WSDM, an audience driven web site design method, which is better tailored to model complex business processes. For this the Conceptual Design phase has been equipped with a Task Modeling step that uses ConcurTaskTrees (CTT), a task modeling technique from the HCI field. This is an easy and powerful technique and only some minor modifications were needed to make it fit for its use in WSDM. In addition, the Navigational Design step has been adapted slightly to allow for a better separation between structure and workflow.

1 Introduction

In the last few years, with e-commerce as a main driving force, web sites have evolved from a simple collection of hypertext pages, through form-based applications, towards applications supporting complex business processes and workflows. In this kind of applications we encounter issues such as transactions, processes that can be suspended and resumed later on, activities that must be synchronized appropriately, activities that should be performed by different actors, persistence, and so on. Most of the current web site design methods only have limited support to model these advanced issues. WSDM as well was lacking such concepts. Therefore, we adapted the Conceptual Modeling phase of WSDM in the following way: the sub phases Information- and Functional Modeling have been replaced by one single sub phase called Task Modeling. This sub phase subsumes Information- and Functional Modeling, but in addition an explicit task modeling activity has been added. The Navigational Design has been better defined, and now allows for a better separation between structure and workflow. Finally, some modeling concepts have been added which allow a better description of advanced task modeling issues, e.g. transaction support. In this paper, we present and discuss this revised Conceptual Modeling phase.

The paper is structured as follows. In section 2 we give a short overview of the different phases of WSDM. In the section 3 we discuss into more detail the new aspects of the Conceptual Design phase and explain how an e-commerce process like the one in the Amazon web site (www.amazon.com) can be modeled using WSDM. Section 4 discusses issues that were not yet covered and draw conclusions.

2 WSDM Overview

In this section, a short overview of the WSDM method is given. More information can be found in [De Troyer 2001a; Casteleyn and De Troyer 2001; De Troyer and Casteleyn 2003; De Troyer and Casteleyn 2001b; De Troyer and Leune 1998]. The two basic characteristics of WSDM are the audience driven approach, and the explicit conceptual design phase. With the first characteristic, the audience driven approach, WSDM puts, already in a very early stage of the development, the emphasis on the different kind of users (called audience classes). WSDM starts the design with the identification of the different audience classes, their needs and requirements. These different audiences and their requirements will drive the entire development process. The second important characteristic is the explicit conceptual design phase. First a conceptual design free of any presentation or implementation detail is made. This allows for different presentations for different users and/or platforms.

The first step of WSDM is defining the *Mission Statement*. The Mission Statement should express the purpose and the subject of the web site and declare the target audience. Based on this Mission Statement a two-step *Audience Modeling* phase is performed. In the first step, *Audience Classification*, the different types of users are identified and classified. Members of the same *Audience Class* have the same information and functional requirements. In the next step, called *Audience Class Characterization*, the characteristics of the different audience classes are given. The result of the Audience Modeling is a hierarchy of audience classes together with an informal description of their requirements: information- and functional - as well as navigational- and the usability requirements, and their characteristics.

Next, the *Conceptual Design* is done. This phase is divided in two steps: *Task Modeling* and *Navigational Design.* During Task Modeling, the information & functional requirements of the different audience classes are modeled by means of *Task Models* and *Object Chunks*. During Navigation Design, we design the conceptual structure of the web site and model how the members of the different audience classes will be able to navigate through the site. For each audience class a *Navigation Track* is created. Navigational requirements are taken into consideration in this step. All navigation tracks together form the *Conceptual Structure* of the site. The Conceptual Design Phase is described in detail in the next section.

During *Implementation Design* the page structure as well as the 'look and feel' of the web site is defined. The aim is to create a consistent, pleasing and efficient look and feel for the conceptual design made in the previous phase by taking into consideration usability requirements and characteristics of the audience classes. The Implementation Design should also provide the specification for the logical data design (if needed). This could be any suitable data definition format: a relational database design, XML, RDF, ….

The last phase, *Implementation*, is the actual realization of the web site using the chosen implementation environment, e.g.HTML, XML, ….

3 The Conceptual Design of WSDM

In this section, we explain the conceptual design phase of WSDM, and more in particular, the steps and concepts that have been changed or added in order to better accommodate the modeling of complex processes.

As running example, we use an e-commerce process like the one in the Amazon web site (www.amazon.com) where people can browse or search for items (books, music, gadgets, …) and buy them electronically. We will refer to this example as the "Amazon example". We have tried to stay as close as possible to the situation in the current Amazon web site, but for reason of clarity we have reduced the process to its most elementary form (without losing complexity).

As explained in the overview, the Conceptual Design phase of WSDM consists of the sub phases *Task Modeling* and *Navigational Design*. They are described into more detail in the following sections.

3.1 Task Modeling

The goal of the Task Modeling is to model the information- and functional requirements of the different audience classes and the processes that are necessary to support these requirements. The requirements of the different audience classes are identified at a high level in the previous phase, the Audience Modeling. In this phase, those high level requirements are elaborated and modeled more rigorously. They are modeled using *Task Models* and *Object Chunks*. Before we explain how this is done, we first give the main requirement identified during Audience Modeling for the Visitor class in the Amazon example.

Requirement *Buy Items* for the Audience Class *Visitor*:

> *The visitor of the web site should be able to browse through the items offered or search for items and obtain information about those items (such as the price, the availability, the authors in case of a book, …). While doing this he should be able to put items in his shopping cart; inspect and manage his shopping cart and finally he must be able to check out (buy) the items that are in his shopping cart.*

During Task Modeling, we first define a *task* for each requirement. Then, the task is elaborated into more detail and decomposed into *elementary tasks.* For this purpose, we use the ConcurTaskTrees (CTT) notation [Paterno et al. 1997; Paterno 2000]. This notation was developed in the context of Human-Computer Interaction (HCI) 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. This makes it much more powerful from a modeling point of view than e.g. HTA (Hierarchical Task Analysis). In addition, CTT has an easy to grasp graphical notation. However, we do not completely follow the original specifications of CTT, but adopted them slightly to better satisfy the particularities of the WWW and web design. We will discuss the differences briefly:

1. CTT was developed to support the specification of flexible and expressive task models in HCI. In the HCI domain, such models are mainly used to analyze the models for complexity and completeness and to validate usability. Therefore, in these models, tasks performed by the user without using the application (such as thinking on or deciding about a strategy) are also considered, because they may be important cognitive activities and useful to take into account during analysis. We will **not** consider such user tasks here. We don't think they are useful for inclusion in the design phase. This means that we will only distinguish between the following categories of tasks:
   - *Application tasks*: tasks executed by the application. Application tasks can supply information to the user, perform some calculations or updates, or determine navigation paths. An example of an application task is validating a login.
   - *Interaction tasks*: tasks performed by the user by interaction with the system, e.g. entering personal information
   - *Abstract tasks*: tasks that consists of complex activities, and thus requiring decomposition into sub tasks, e.g. buying items at Amazon.

   The graphical presentation of these task categories is given in figure 1.

Figure 1: Categories of Tasks

2. A (complex) task can be decomposed. This results in a hierarchical structure for a task. The same task can be used in different sub-trees. Tasks are identified by means of their name. CTT prescribes that if the children of a task are of different categories then the parent task must be an abstract task. We will not follow this rule. We will use the category of the task to explicitly indicate who will be in charge of performing the task. For an interaction task, the user will be in charge; for an application task the application will be in charge. In this way, we can indicate for example who will initiate a subtask, or who will make a choice between possible subtasks.
3. TCC uses the following operators to express temporal relations among tasks in a task hierarchy. Note that for some of the operations, we have changed the meaning slightly (this will be indicated):
   - *Order independent* (T1 |=| T2): the tasks can be performed in any order. E.g. in the Amazon example, viewing the details of an item and adding the item to the shopping cart can be done in any order.
   - *Choice* (T1 [] T2): one of the tasks can be chosen and only the chosen task can be performed. E.g. to select items in the Amazon example, the user can either use a search specification or browse through the different categories.
   - *Concurrent* (T1 ||| T2): the tasks can be executed concurrently, for example browsing a file and printing a file (there is no example of this in the Amazon example).

- *Concurrent with information exchange* (T1 |[]| T2): the tasks can be executed concurrently, but they have to synchronize in order to exchange information. For example, editing a file and scrolling its content. They need to exchange information because it is possible to edit information that is visible in the scrolling. Also for this relation, there is no example in the Amazon example.
- *Deactivation* (T1 [> T2): the first task is deactivated once the second task is started. For example, in the Amazon example, once an item is deleted from the shopping cart it cannot be edited anymore.
- *Enabling* (T1 >> T2): the second task is enabled when the first one terminates, for example consulting the database is only possible after a successful login.
- *Enabling with information exchange* (T1 []>> T2): the second task is enabled when the first one terminates but in addition, some information is provided by the first task to the second task, for example checking a password is possible after the password was entered and the value of the password should be provided for checking it.
- *Suspend-resume* (T1 |> T2): indicates that T1 can be interrupted to perform T2 and when T2 is terminated, T1 can be reactivated from the state reached before the interruption, e.g. in the Amazon example, filling the shopping cart can be interrupted at any time to inspect the shopping cart.
- *Iteration* (T*): the task can be performed repetitively. Note that in CTT the meaning is different. There, the meaning is that the action is performed repetitively: when the action terminates, it is restarted automatically until the task is deactivated by another task. The interpretation here is that the task can be repeated several times. In the Amazon example, the activity of filling the shopping cart with an item can be repeated several times, and it will end when the user decides not to re-enter the iteration but to continue with the next task.
- *Finite iteration* (T(n)): if the task has to be repeated a fixed number of times (number known in advance).
- *Optional* ([T]): indicates that the performance of the task is optional, e.g. viewing the details of an item is optional in the Amazon example.
- *Recursion*: occurs when the sub tree that models a task contains the task itself. This means that performing the task can be a recursive activity. An example of a recursive task in the Amazon example is browsing through the categories.

We found it useful to extend this list with an extra operator to express a transaction:

- *Transaction* (-> T <-): the task must be executed as a transaction. This means that if the task, or in case of a complex task one of the tasks in the task's sub tree, is not completed successfully, the whole task will not be successful and all activities should be rolled back (i.e. 'all or nothing').

Figure 2 gives the task model for the task "Buy Items" using this (modified) CTT notation. "Buy Items" is the task that we defined for the requirement *Buy Items* formulated earlier.

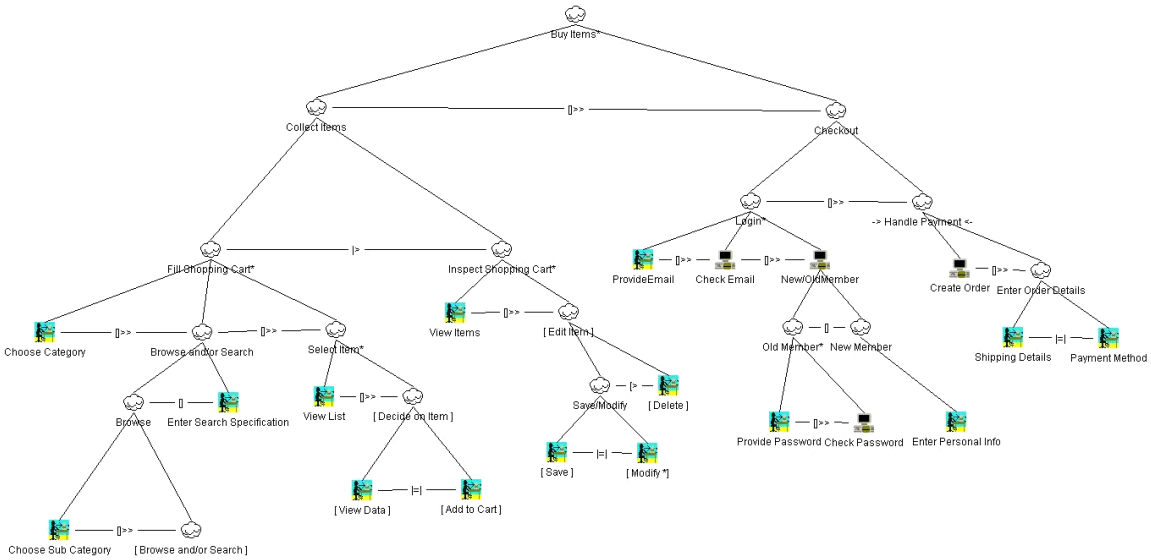

Figure 2: Task Model for Buy Items

We briefly describe the task model for Buy Items:

- The task Buy Items is decomposed into Collect Items and Checkout. The relation between the two is enabling with information exchange. Indeed, first items should be collected into the shopping cart before checkout is possible. The information that needs to be exchanged is the shopping cart.
- Collect Items is decomposed into Fill Shopping Cart (which can be repeated) and Inspect Shopping Cart. The relation between the two is suspend-resume, which indicates that Fill Shopping Cart can be interrupted by Inspect Shopping Cart at any point and will be reactivated from the state reached before the interruption once the Inspect Shopping Cart task is ended.
- To perform Fill Shopping Cart, first the user has to select a category (Choose Category), then locate some items either by browsing or using a search specification (Browse and/or Search), and next he can repeatedly select an item (Select item) from the result list to view its details and/or add it to the shopping cart. Note that the user doesn't need to view details or put an item in his cart (View Detail and AddToCart are optional).
- To Checkout, first the user should login (Login). Logging in can be repeated in case of unsuccessful login. To login, the user first provides his Email address (ProvideEmail). This address is validated by the application (CheckEmail) and depending whether or not it is the email address of a member the application will ask the user to provide his password (Provide Password) and check it (Check Password), or ask the user for his personal information and create a new member (Enter Personal Info). If the login is terminated, the member is identified and the payment handling (Handle Payment) can be done (enabling with information exchange). Handle Payment is modeled as a transaction. First an shopping order is created by the application (Create Order) and next, the order details are added (Enter Order Details). This is done by entering the shipping details (Shipping Details) and the payment method (Payment Method). The order in which this is

done is not important (order independent relation), but both must be given for the task to be successful. If the Handle Payment task is quitted before all subtasks are performed, a rollback will be done and there will be no shopping order.

After a task model is created for a task, we start modeling the information and/or functionality needed to perform the task. This is done by creating an *object chunk* for each elementary task in the task model. Such an object chunk should model the information and/or functionality required by this elementary task. Two example chunks are given in figure 3 and figure 4. Figure 3 gives the object chunk for the task View List. It models the information that should be displayed about the items selected (obtained by means of the search or browsing). To keep the drawing small, we have limited the items to books and music. Figure 4 gives the object chunk for the task Payment Method. It models the information that should be requested from the user as well as how this information should be added to the order (created in Create Order and passed as parameter). We will not elaborate on this step because this has already been explained in another paper [De Troyer and Casteleyn 2001b]. The notation used is an extended ORM notation (for ORM see [Halpin 2001]). Sometimes, e.g. in case of application tasks, a task may be too complicated to model it (completely and only) using a graphical notation. Then, no (or only a partial) chunk is created and the specification is given in a different way, e.g. in an informal way or by using some pseudo code.

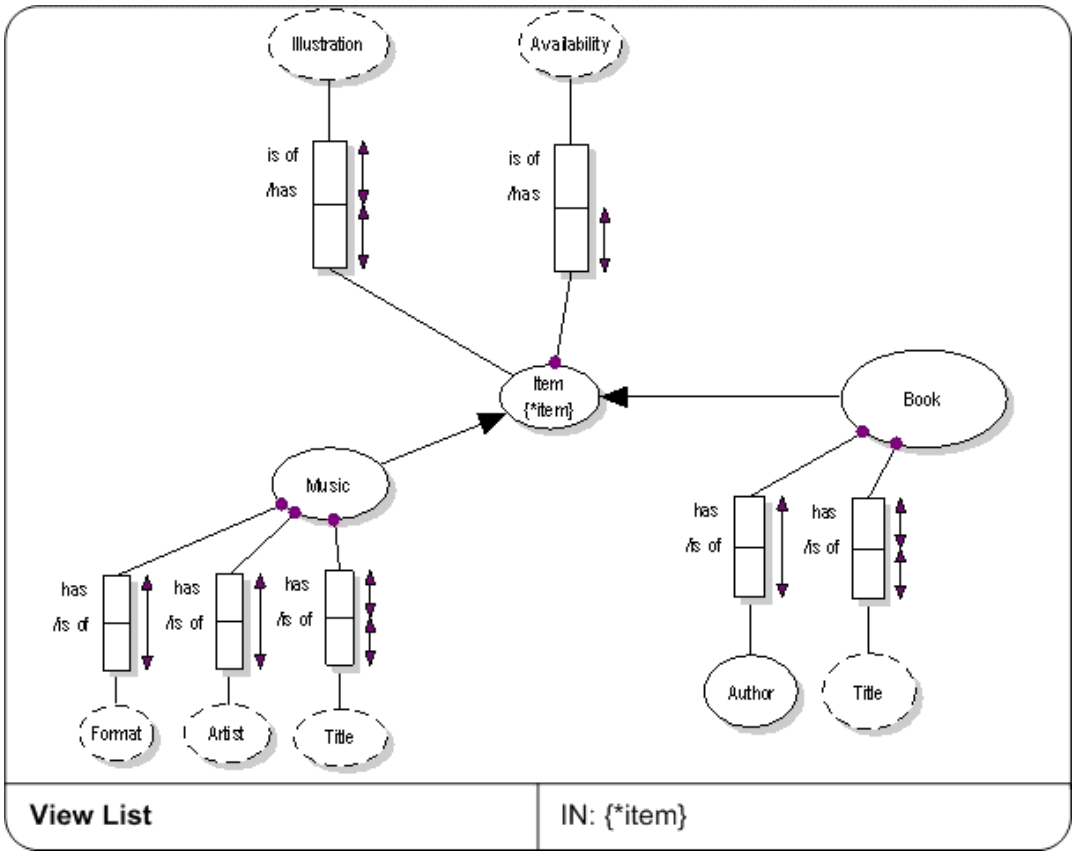

Figure 3: Object Chunk for the Task View List

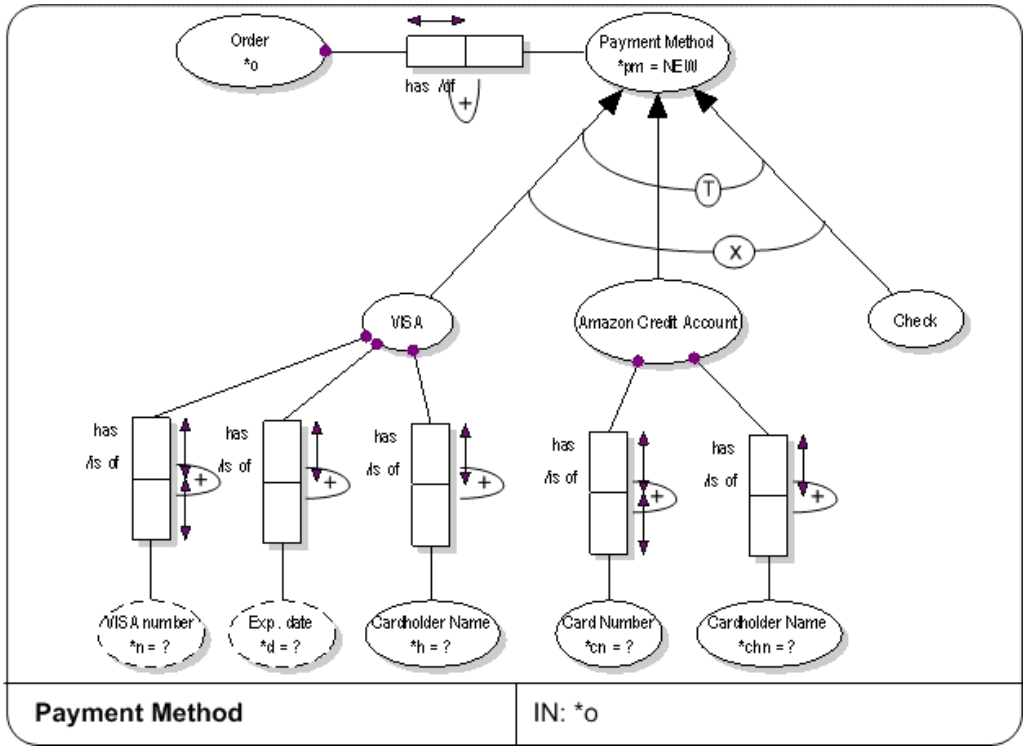

Figure 4: Object Chunk for the Task Payment Method

3.2 Navigational Design

The purpose of the Navigational Design is to define the conceptual structure of the web site and to model how the members of the different audience classes will be able to navigate through the site and perform the tasks.

Because of the audience driven approach of WSDM, a *Navigation Track* is created for each audience class. A navigation track can be considered as a sub site containing all information and functionality needed by the members of the associated audience class. Next, all audience tracks are combined into the *Conceptual Structure* by means of structural links. The structure defined between the audience tracks should correspond to the hierarchical structure defined between the audience classes in the Audience Class Hierarchy. How this is done exactly is described in detail elsewhere [Casteleyn and De Troyer 2001] and will not be discussed again. We rather focus on the construction of a Navigation track, and more in particular on the construction of the *Task Navigation Models.*

The task navigation models describe how the user will be able to perform the tasks (defined during Task Modeling) in the web site. The tasks models and the object chunks defined (during Task Modeling) for the different tasks of an audience class are the input for this. *Components* and *project logic links* are used to construct a task navigation model for each task model defined. This is done by defining a component for each elementary interaction task defined in the hierarchical decomposition of the task. Next, so-called *process logic links* between components are used to express the workflow or process logic expressed in the task model by means of the temporal CTT relations. In fact, the

temporal CTT relations need to be translated into links (one- or bi-directional links; one-to-one, one-to-many, many-to-one or many-to-many links; conditional- or non-conditional links; interrupt-resume links). (More on the different types of links used in WSDM can be found in [De Troyer and Casteleyn 2003]). Components and process logic links can be grouped into a *transaction* to indicate that they constitute a conceptual transaction (as indicated in the task model). To provide for a better overview, complex subtasks may be modeled separately, in so-called *Subtask Navigation Models*. The exact algorithm to map a task model into a task navigation model will be given elsewhere, but the mapping for the Amazon example is provided here.

Figures 5 to 9 shows the different task and subtask navigation models resulting from the task model "Buy Items" (given in figure 2). Components are represented as rectangles and process logic links as arrows. A transaction is indicated by means of an octagon (see for example figure 9). A subtask navigation model is represented as a dotted gray rectangle inside another (sub)task navigation model. Figure 6, 7, 8 and 9 are subtask navigation models. In figure 6, the arrow from inside the dotted gray rectangle, representing the subtask navigation model Fill Shopping Cart, to Inspect Shopping Cart is a suspend-resume link and indicates that from every component in Fill Shopping Cart there is in fact a link to start of Inspect Shopping Cart (e.g. the first component).

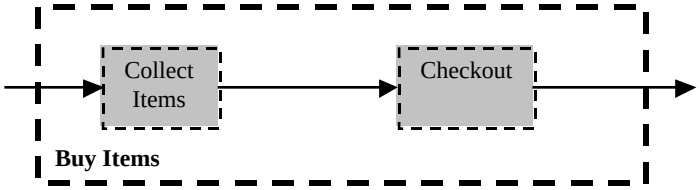

Figure 5: Task Navigation Model for Buy Items

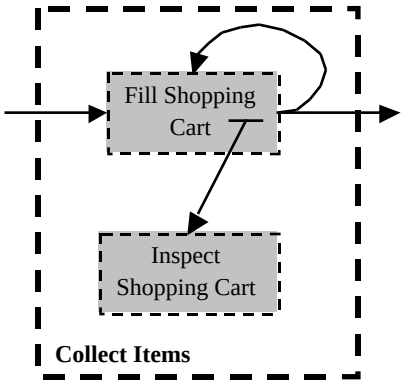

Figure 6: Subtask Navigation Model for Collect Items

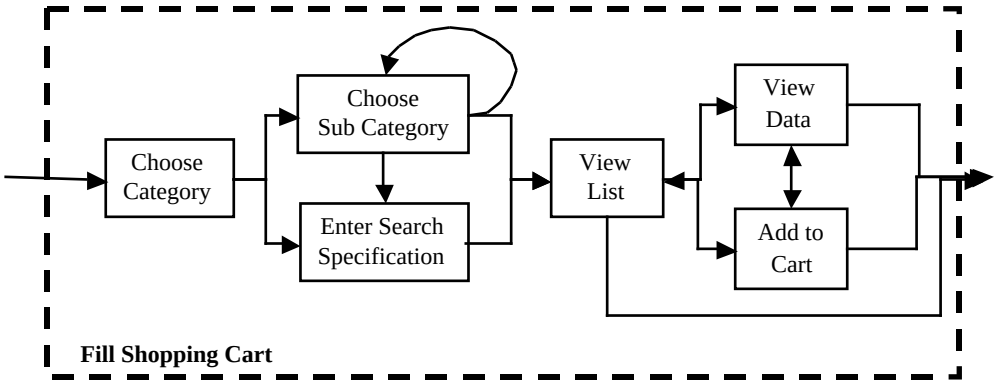

Figure 7: Subtask Navigation Model for Fill Shopping Cart

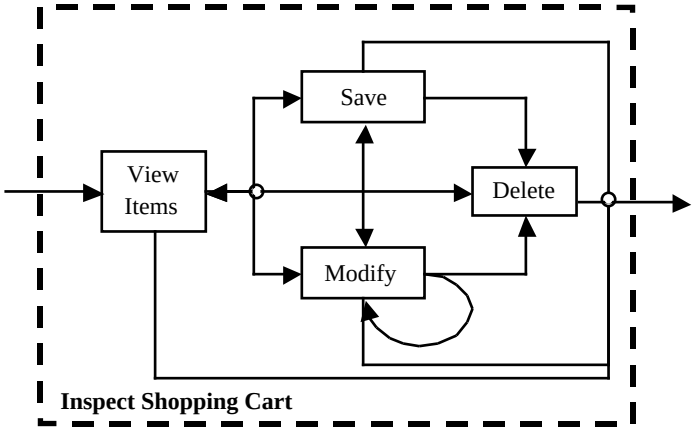

Figure 8: Subtask Navigation Model for Inspect Shopping Cart

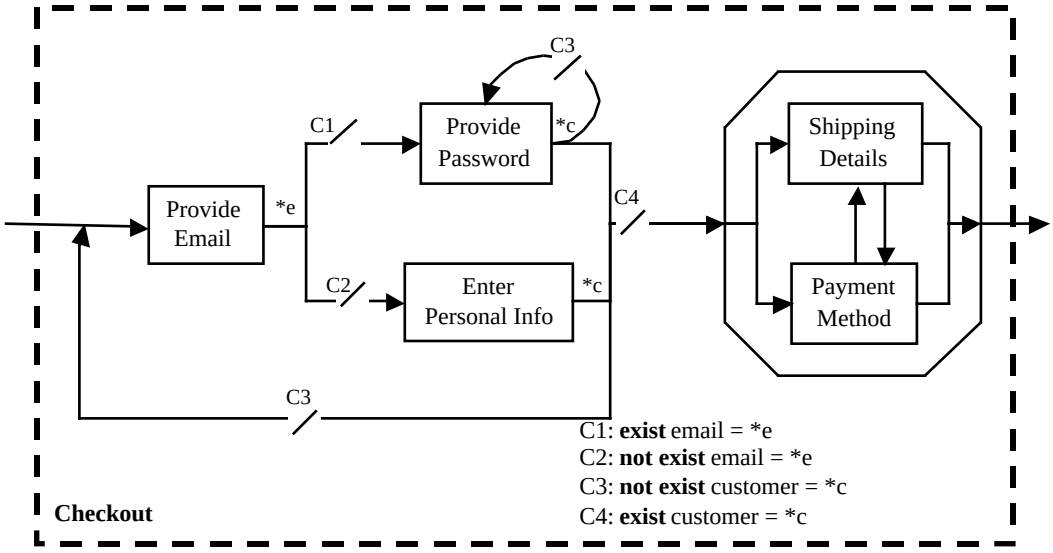

Figure 9: Subtask Navigation Model for Checkout

To complete the task navigation models, the components are linked to the object chunk(s) created for the respective elementary tasks (not shown in the figures). Components are a kind of placeholders for object chunks. They allow using the same object chunk in different contexts (tasks). Because each audience class has its own set of tasks, a set of task navigation models is made for each audience class. These task navigation models need to be grouped (structured) into a navigation track. This is done using structural links. More information on this can be found in [De Troyer and Casteleyn 2003]. In simple cases (if the number of tasks is limited), the tasks may be presented as independent alternatives (e.g. using a menu in the actual web site).

During Navigational Design, we also define *semantic relationship links* between object chunks. Semantic relationship links are based on relationships that exist between objects

in the domain and which are modeled in the object chunks. E.g. in the web site of Amazon, when you have selected a (number of) book(s), you can click on one of the author names to view more information about this author (such as other books of this author). This facility can be model by means of a semantic relationship between the object chunk "View Items" (that models which information about an item should be visible) and the object chunk "Author Info" (that models the information about an author). See figure 6 for a graphical illustration of this semantic relationship link. The link is based on the existence of a semantic relation between the object types "Item" (the subtype Book in this case) and "Author": Book written-by/of Author. The semantic relationship links are independent of a particular task. This means that in each task where the chunk "View Items" is used, the link will be available.

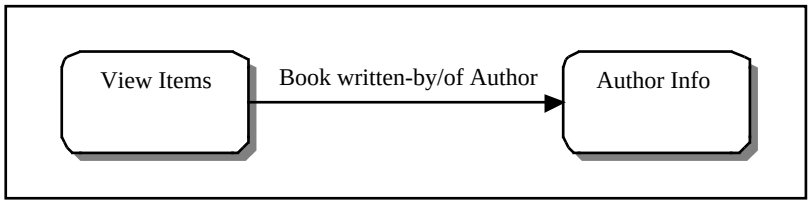

Figure 10: Semantic Relationship Link

On top of the conceptual structure defined by means of the structural links, and navigation possibilities defined by the process logic links and the semantic relationship links, *navigational aid links* can be added to ease the navigation through the web site and to enhance the usability of the web site. From the viewpoint of being able to reach information, they are strictly speaking, not needed; all information and functionality should also be reachable without the navigational aid links. They could be compared to adding an index and post-it pointers to chapters in a book: the information in and the structure of the book stays the same, but the user is provided with shortcuts for more easily accessing the content of the book. A typical example of a navigational aid link is the home link, which usually you can find on each page of a web site. This is also the case for the Amazon web site. For the home link, WSDM uses a graphical notation to indicate the component that represents the home.

4 Discussion and Conclusions

In this paper, we have demonstrated how to model complex user tasks within the framework of WSDM. According to the audience driven philosophy of WSDM, we start by describing the user requirements of the different users. Each requirement is then translated into a user task, and if necessary, decomposed into (elementary) sub-tasks. This is done using a modified form of ConcurTaskTree, which allows do describe the temporal relations between the different tasks (e.g. sequential, concurrent, optional, etc.). For every elementary task, a chunk (a tiny conceptual schema describing what exactly the user needs to see; or which information he is supposed to enter; or the functionality required from the application) is created. From these ConcurTaskTrees with temporal

relations between the different task described, and the chunks formally describing the information and/or functionality required by the user or the application, we are able to generate the navigation structure for the different users, offering the right tasks at the right time to the users.

An aspect of complex business process and workflow that is not covered by the Amazon example is when activities performed by different actors (more in particular users) must be synchronized. Also in such a situation, CTT can be used. In [Paterno 2000], it is discussed how the CTT task models can be used for cooperative applications. They recommend to make a task model for each user involved and to express the cooperation in a separated tree, the cooperative tree. As WSDM already makes task models per audience class, only a cooperative task tree need to be added during Task Modeling to accommodate synchronization between tasks of different users.

.

5 References

Casteleyn, S. and De Troyer, O., 2001. Structuring Web Sites Using Audience Class Hierarchies. Proceedings of DASWIS 2001 (ER) workshop. Yokohama, Japan.

Casteleyn, S. and De Troyer, O., 2002. Exploiting Link Types during the Web Site Design Process to Enhance Usability of Web Sites. Proceedings of the IWWOST 2002 (ECOOP) workshop. Malaga, Spain.

De Troyer, O. and Leune, C., 1998. WSDM : A User-Centered Design Method for Web Sites. Computer Networks and ISDN systems, Proceedings of the 7th International World Wide Web Conference, Elsevier, 85-94.

De Troyer, O., 2001a. Audience-driven web design. Information modeling in the new millennium, Eds. Matt Rossi & Keng Siau, IDEA GroupPublishing, ISBN 1-878289-77-2

De Troyer, O. and Casteleyn, S., 2001b. The Conference Review System with WSDM. IWWOST 2001 (http://www.dsic.upv.es/~west2001/iwwost01/), Valencia, Spain.

De Troyer, O. and Casteleyn, S., 2003. "Exploiting Link Types during the Conceptual Design of Web Sites", In International Journal of Web Engineering Technology, Vol 1, No. 1

Halpin, T., 2001. Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design, 1st Edition. Morgan Kaufmann Publishers.

Paterno F. 2000, “Model-Based Design and Evaluation of Interactive Applications”, Springer Verlag

Paterno, Mancinii and Meniconi, 1997. “ConcurTaskTress: a Diagrammatic Notation for Specifying Task Models”, In Proceedings of INTERACT 97, Chapman & Hall, 362-366