Chapter #

WSDM: WEB SEMANTICS DESIGN METHOD

Olga De Troyer, Sven Casteleyn, and Peter Plessers
Research Group WISE, Department of Computer Science, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel, Belgium

Abstract: In this chapter we describe WSDM, an audience-driven Web design method. WSDM was one of the first Web design methods and different from the other methods because of its audience-driven approach. Now the method is adapted to the use of Semantic Web technology and support the generation of semantically annotated web sites and web applications.

Key words: web design, audience driven, WSDM, OWL, localization, adaptation, semantic annotations, structural annotations

1. WSDM OVERVIEW

WSDM was introduced by De Troyer and Leune in 1998 (De Troyer and Leune, 1998). At that time the acronym stood for **W**eb **S**ite **D**esign **M**ethod and only targeted information-providing web sites. In the meantime, the method has evolved a lot and now also allows designing traditional web applications as well as semantic web applications, hence the renaming of the method into **W**eb **S**emantics **D**esign **M**ethod.

More than other web design methods, WSDM is a methodology, i.e. it not only provides modeling primitives that allow a web developer to construct models that describe the web site/application from different perspectives and at different levels of abstraction, but it also provides a systematic way to develop the web application. Developing a web site/application with WSDM starts with the formulation of the so-called "Mission Statement" and follows a well-defined design philosophy that offers the designer the necessarily support to structure the web site. The method consists of a sequence of phases. Each phase has a well-defined

output. For each phase, a (sub) method describing how to derive the output from its input is provided. The output of one phase is the input of a following phase. As already indicated, currently the method allows developing web sites as well as web applications. For the sake of simplicity, we will use the term "web systems" to indicate both web sites and web applications. It is also important to notice that WSDM allows developing web systems that are semantically annotated, in this way effectively enabling the Semantic Web. Content-related (semantic) annotations as well as structural annotations can be generated. Content-related annotations make the semantics of the content explicit. Structural annotations are annotations that explicitly describe the semantics of the different structural elements of the web systems. Structural annotations can be exploited by third parties to transcode the web system to a different format, for example to formats appropriate for a screen reader, or they can be exploited by search engines for their page segmentation (see e.g., (Deng Cai et al., 2004) for an overview of page segmentation by search engines).

WSDM follows an *audience-driven* design approach. An audience-driven design philosophy means that the different target audiences (visitors) and their requirements are taken as starting point for the design and that the main structure of the web site is derived from this. Concretely, this results in different navigation paths (called audience tracks) offered from the homepage, one for each different kind of visitor.

Figure #-1 shows an overview of the different phases of WSDM. The different phases are shown sequential. However, in practice the design process is rather iterative. In this section, a brief description of the different phases is given. In the following sections, the different phases are explained into more detail and illustrated with examples from the common example, the Internet Movies Data Base web site[1] or IMDb for short. Note, that it is not realistic and also not the purpose to redesign the entire system here. Simplifications are made were necessarily to reduce the size of the drawings and the examples.

In the first phase of the method, the Mission Statement Specification, the mission statement for the web system is formulated. The goal of this phase is to identify the purpose of the web system, as well as the subject and the target users. Without giving due consideration to the purpose, there is no proper basis for making design decisions, or for evaluating the effectiveness of the web system. The target users are the users that we want to address or that will be interested in the web system. The subject of the web system is of course related to the purpose and the target users of the web system. The subject must allow fulfilling the purpose of the web system and it must be adapted for the target users. The output of this phase is the *mission*

[1] http://www.imdb.com

statement. It is formulated in natural language and must describe the purpose, subject and target users of the web systems. In fact, the mission statement establishes the borders of the design process. It allows (in the following phases) deciding which information or functionality to include or exclude, how to structure it and how to present it.

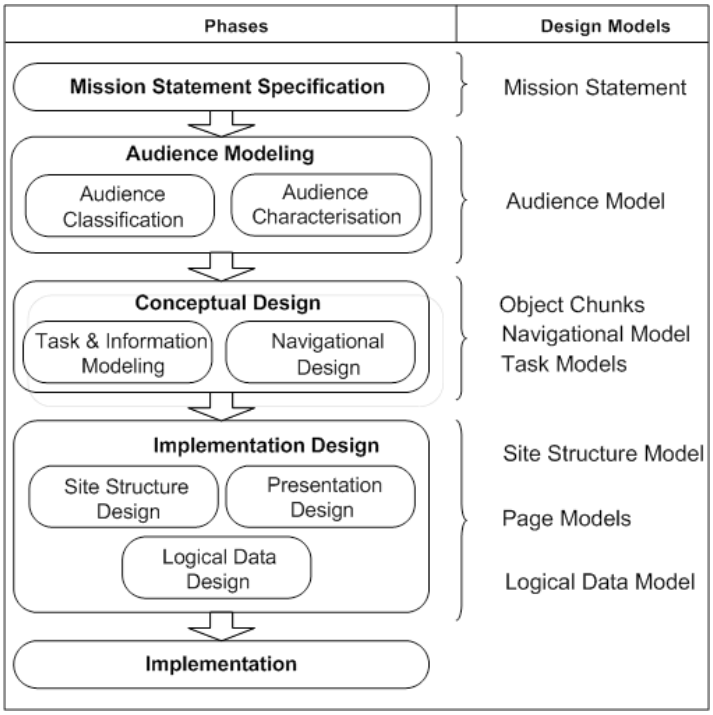

Figure #-1. WSDM Overview

The next phase is the Audience Modeling phase. The target users identified in the mission statement are refined into so-called *audience classes*. This means that the different types of users are further identified and classified into audience classes. The classification is based on the 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 information, functionality and structure of the web system at a conceptual level. A conceptual design makes an abstraction from any implementation technology or target platform. The information and functionality are

specified during the Task & Information Modeling sub phase. The overall conceptual structure including the navigational possibilities for each audience class is specified during the Navigational Design sub phase.

During the Implementation Design phase, the conceptual design models are complemented with information required for an actual implementation. It consists of three sub phases: Site Structure Design, Presentation Design and Logical Data Design. 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 Logical Data Design is only needed for data-intensive web systems. 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. Evidently, other types of data sources are possible (XML, RDF, OWL, …).

The last phase is the Implementation. The actual implementation can be generated automatically from the information collected during the previous phases.

2. MISSION STATEMENT SPECIFICATION

In the first phase of the method, the mission statement for the web system should be formulated. To develop a successful web system, it is necessary to first reflect on the purpose of the web system otherwise there will be no proper basis for making design decisions, or for evaluating the effectiveness of the web system. E.g., for a company, the purpose may range from simply "having an identity on the Web", to "advertising some of its products", to "provide a full-fledged e-shop"; for public and local authorities it may range from providing general information to a full-fledged e-government system that allows to arrange official matters (e.g., apply for official documents) using the Web. The purpose should be established in consultation with the different stakeholders.

The different stakeholders should also agree on the topics that should be covered by the web system. Even if the purpose is clear, it may be necessary to explicitly name the topics the system will deal with. For example, a company may decide to offer on-line information about products but only for their products in a higher price range. Another example is a high school that

decides only to offer information about their education system and courses, but not about their research activities.

Furthermore, also the target users should be identified. In principle everybody can visit a public web site, also people for which the web site is not relevant. However, it is impossible to satisfy the expectations of each possible visitor. It is better to focus on the users needed to make the web system successful, called the target users. For example, consider a company that only sells very specialized technical items. In this case, the web site should focus on the people that need these products. These people are likely not a general public but instead probably very technical and specialized people. When building a web system for a university, an important group of users you probably want to address are potential students.

It is clear that the purpose, subject (topics) and the target users are highly related. The subject and the target users of the web system must allow fulfilling the purpose, and the subject must be suitable for the target users. So, one may argue that if the purpose is stated, the topics and target users are implicitly stated as well. However, to avoid misunderstandings, it is better to explicitly identify all three aspects of the mission statement. The mission statement is formulated in natural language.

Later on, in the following phases, the mission statement serves as the basis to decide what information or functionality to include (or not), how to structure it, and how to present it. Information or functionality that is not needed for the purpose or is not covered by the subject should not be considered. How to present and structure information and functionality is highly dependent on the target users, e.g., information should be presented and organized differently to professionals than to a common public. In addition, the mission statement can be used during validation to check if the web system has indeed achieved the formulated purpose.

To illustrate this phase, we have formulated a mission statement for the example web system. Currently the web system mainly focuses on movies but there is also a part about games. Therefore, the mission statement is formulated as follows:

"To be the biggest and best movie and game site on earth. For movies, this will be achieved by providing as much information as possible on movies, including their actors, directors and producers, as well as to provide news, allow exploring show times, buy tickets in selected cinemas, and to share personal opinions about movies. For games, information about games is offered as well as news, and game lovers should be able to exchange information". This mission statement defines the purpose, subject and target users as follows:

- **Purpose**: to be the biggest and best movie and game site by (1) providing information and news on movies and games, (2) allow to explore show

times of movies and to buy tickets and (3) to allow movie lovers and game lovers to share personal opinions and exchange information.
- **Subject**: movies and related information such as actors, directors and producers; selected cinemas; games.
- **Target users**: movie lovers and game lovers.

As you may notice, the purpose may involve multiple goals. Here, the stakeholders want to realize their long-term purpose (to be the biggest and best movie and game site) by means of three (related) goals. The subject may also deal with different topics. Here, movies as well as cinemas and games are considered. Also the target users may be composed of different groups. Here, two different groups of users are involved: movie lovers and game lovers.

3. AUDIENCE MODELING

The mission statement formulated in the first phase is only a first and very incomplete description of the system that should be developed. Because WSDM is an audience-driven design method, the first concern that is elaborated is the set of target users. The target users identified in the mission statement are refined into so-called *audience classes*. This is done by means of two sub phases: the Audience Classification and the Audience Characterization. During Audience Classification, the different types of users are identified into more detail and classified into so-called audience classes. During Audience Characterization relevant characteristics are specified for each audience class. We describe each sub phase into more detail in the following subsection, but first we discuss why it is important to distinguish between different types of users.

In general, a web system has different types of visitors that may have different needs. Consider as an example the web site of a university. Typical users of such a web site are potential students, enrolled students, and researchers. Potential students are looking for general information about the university and the content of the different programs of study. The enrolled students need detail information about the different courses, timetables and contact information of the lecturers (telephone extension, room number, and contact-hours). Researchers look for information on research projects, publications and general information on the researchers (full address, research interests, and research activities). This illustrates that different types of users (WSDM uses the term audience class) may have different information and/or functional requirements. To ensure a good usability, this should be reflected in the web system. For example, a student should be able to follow a navigation path that leads him to the information he is interested

in without having to travel through pages of other (for him) non-relevant information. If this is not the case and all information is provided to all users, a user has to scan the page(s) in order to find the links, the pages and the pieces of information or functionality that are relevant for him. Providing too much non-relevant information enhances the lost-in-hyperspace syndrome.

Next to the fact that different types of users may have different information and functional requirements, it may be necessary to represent the (same) information or functionality in different ways to different kinds of users. This depends on the characteristics of the users. As an example we again consider the university example. Potential students, especially secondary school students are not familiar with the university jargon and should be addressed in a young and dynamic way. Also, by preference, the information should be offered in the native language. The enrolled students are familiar with the university jargon. They also prefer to have the information in the native language, however for foreign students (e.g., who follow exchange programs) English should be used as communication language. For researchers, it may be sufficient to use English. When the information and functionality is grouped based on the requirements of the different types of users, it is also possible to adapt the presentation to the characteristics of the different type of users without the need to rely on adaptive web systems or personalization (Brusilovsky, 1996). Although personalization may be for some situations undoubtedly the best solution, in other situations it may be less appropriate.

3.1 Audience Classification

The target users informally identified in the mission statement are the input for the Audience Classification. These target users are refined and classified into audience classes based on differences in their information and functional requirements. All members of an audience class must have the same set of information and functional requirements.

Sometimes, the set of information and functional requirements of one audience class is a subset of the set of requirements of another audience class. To accommodate such situations the concept of *audience subclass* is used. Figure #-2 gives the graphical representation of an audience class and an audience subclass. The audience class B is an audience subclass of the audience class A, which means that the audience class B has all the same information and functional requirements as audience class A but also some extra requirements. In other words, the set of requirements of audience class A is a subset of the set of requirement of the audience (sub-)class B. From the point of view of their populations, the population of the audience

subclass is a subset of the population of the audience superclass. Indeed, the members of the audience subclass have more requirements that the members of the audience superclass, so these can only be less people.

Figure #-2. Graphical Representation of Audience Class and Subclass

WSDM prescribes two alternative methods for discovering the audience classes. The first method uses the activities of the organization for which the web systems needs to be developed and the role people play in these activities. Only activities that are related to the subject of the web system are considered. Each activity involves people, which may be potential users for the web system. These people should be identified. If they belong to the target users of the web system, their requirements (information-, as well as functional-, usability- and navigational requirements) are formulated (in an informal way and at a high level). Users with the same information and functional requirements become members of the same audience class. Whenever the information and functional requirements of a set of users differ, a new audience class is defined or if possible, an audience subclass is introduced. If possible, the activities are decomposed in order to refine the audience classes. This may introduce audience subclasses. The decomposition stops if no new subclasses emerge or if decomposition is not possible anymore. In summary, the method is as follows:

Step 1: Consider the activities of the organization related to the purpose of web system.

Step 2: For each activity

1. Identify people involved
2. Restrict them to the target users
3. Identify their requirements
4. Divide them into audience classes based on different information or functional requirements
5. Decompose the activity if possible and repeat Step 2

In this way a hierarchy of audience classes is constructed. At the top of this *audience class hierarchy* we always place the audience class *Visitor*. The Visitor audience class represents all targets users. The requirements

associated with the Visitor class are the requirements that are common to all users.

The second method starts with identifying all possible information and functional requirements of the target users, without wondering how to classify them into audience classes. The different audience classes and the subclass relations between them are derived from a matrix that is constructed using the different requirements as dimensions of the matrix. The cells in the matrix answer the question "Does every user who has the requirement of this row, also (always) have the requirement of this column?". Informally, every row i of such a matrix characterizes the users having the requirement associated with this row in term of the other requirements they have. From this requirements matrix the audience class hierarchy can be (automatically) derived as follows:

If two or more rows are exactly the same, this means that the users that are represented by these rows are the same in term of possible requirements, and thus the users having these requirements should belong to the same audience class. If the set of "Y" entries of a row *k* is a subset of the "Y" entries of another row *l*, this means that the users represented by row *l* have the same requirements as the users represented by row *k* and some extra requirements. So, the audience class represented by row *l* is an audience subclass of the audience class represented by the row *k*.

This algorithm is formally specified in (Casteleyn and De Troyer, 2001). In summary it looks as follows:

Step 1. Construct the audience class matrix based on all the requirements formulated for the web system to be built.

Step 2. Determine the equivalence rows. Each set of equivalence row represents an audience class. The user requirements for this audience class are the requirements associated with the rows. Meaningful names should be given to the audience classes.

Step 3. Identify subset relation between the rows of the different equivalence classes. These subset relations results in audience subclass relations.

Step 4. Construct the integrated audience class hierarchy (using the audience classes and the audience subclass relations).

We now illustrate the Audience Classification phase for the IMDb example. We first illustrate the activity-based method and then the method based on the requirements matrix.

We suppose that IMDb is run by a separated organization. In this case, the activities of this organization could be:

- Provide information about movies and games
- Sell cinema tickets

- Maintain the information about movies and games

The people (and other organizations) involved in these activities are movie lovers, game lovers, cinemas and the organization's database administrators. Figure #-3 illustrates this.

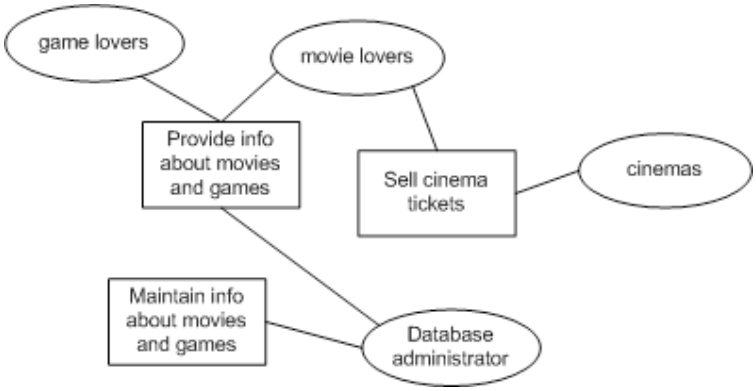

Figure #-3. Activity Diagram for the IMDb Example

The organization's database administrators are not part of the target users and maintaining the information about movies and games is also not part of the purpose of the web system. Therefore, the database administrators are not considered in the audience classification. The cinemas are involved in the activity "Sell cinema tickets" because the information about the movies, show times and available seats must be synchronized with the ticket information of the cinemas. However, we suppose that this synchronization is done by means of interactions with the cinemas' information systems and no manual intervention is needed for this. Therefore also cinemas can be discarded. This leaves us with the movie lovers and the game lovers. The requirements for these users can be specified as follows:

Movie lovers

1. To get an overview of the movies currently playing
2. To get an overview of the movies coming soon
3. To find a movie by means of a search function
4. To browse the movie database
5. To get an overview of new DVDs
6. To get an overview of DVDs coming soon
7. To obtain information about a movie
8. To obtain information about a person involved in a movie
9. To read news about movies
10. To have links to other interesting sites in relation to movies

11. To explore show times at different cinemas of movies currently playing
12. To buy tickets for a show time in a selected cinema
13. To manage a personal movie list
14. To post messages on the movie message boards
15. To enter a comment about a movie

Game lovers

1. To get an overview of games
2. To get information about a game
3. To read news about games
4. To have links to other interesting sites in relation to games
5. To post messages on the game message boards

The requirements for these two groups are sufficiently different to put them in separate audience classes. This results in two different audience classes: Movie Lover and Game Lover. Note that these classes don't need to be disjoint: a person may be a movie lover as well as a game lover. However, such a person will in general not want to look for movie information and game information at the same time. Also, in general a movie lover is not necessarily a game lover and vice versa.

There is no useful further decomposition for these activities. The resulting audience class hierarchy is shown in figure #-4. Note that Visitor is the top of the hierarchy.

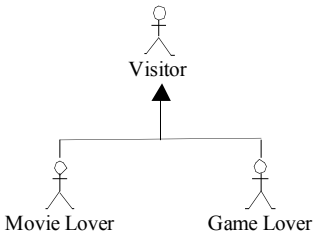

Figure #-4. Audience Class Hierarchy for the IMDb Example

Some parts in the IMDb system require authorization to access and are protected by a login. Therefore additional (authorization) requirements are needed. A user must be able to register and once he is registered he is able to login. Also logout out must be possible. These are requirements common to the movie lovers and the game lovers and are therefore assigned to the Visitor class.

Visitor

1. To register

2. To login
3. To logout

To use the matrix method to derive the (final) audience class hierarchy, we first have to list all information and functional requirements of the target users, here being movie lovers and game lovers. These are defined as follows:

1. To get an overview of the movies currently playing
2. To get an overview of the movies coming soon
3. To find a movie by means of a search function
4. To browse the movie database
5. To get an overview of new DVDs
6. To get an overview of DVDs coming soon
7. To obtain information about a movie
8. To obtain information about an actor
9. To obtain information about a director
10. To read news about movies
11. To obtain links to other interesting sites in relation to movies
12. To explore show times at different cinemas of movies currently playing
13. To buy tickets for a show time in a selected cinema
14. To manage a personal movie list
15. To post messages on the movie message boards
16. To enter a comment about a film
17. To get an overview of games
18. To get information about a game
19. To read news about games
20. To obtain links to other interesting sites in relation to games
21. To post messages on the game message boards
22. To register
23. To login
24. To logout

Then the requirements matrix is constructed. For each requirement a row and a column is created. Then the cells are filled by answering the questions "Does every user who has the requirement of this row, also (always) have the requirement of this column?". For example for row 1 (requirement 1) we obtain the following answers:

- For column 1 to 16: "Y"
- For column 17 to 21: "N"
- For Column 22 to 24: "Y"

The other cells are filled in a similar way. Table #-1 shows a reduced version of the matrix where equal rows and columns are displayed as a single row or column.

Table #-1. Reduced Requirements Matrix

	1-16	17-21	22-24
1-16	Y	N	Y
17-21	N	Y	Y
22-24	N	N	Y

The following audience classes can be derived from this matrix (equal rows):

- Row 1 to 16: Movie Lover
- Row 17 – 21: Game Lover
- Row 22 to 24: Visitor (this audience class only has the requirements which are common to all users)

The following subclass relations can be derived (subset between rows):

- Movie Lover (rows 1-16) is an audience subclass of Visitor (rows 22-24)
- Game Lover (row 17-21) is an audience subclass of Visitor (rows 22-14)

This results is the same audience hierarchy as the one given in figure #-4.

Once the audience classes are identified, it should be investigated if the members of those audience classes have special usability or navigational requirements. Different examples of navigational requirements can be found in the IMDb example. For example for the Movie Lover audience class we can formulate the following navigational requirements:

- The user should be able to navigate directly from the information of a movie to the show times and the ordering of tickets when the movie is currently played.
- If more information about some item shown is available, then the user should always be able to directly navigate to this information. E.g., from the movie information to the information about its directors, its actors, its genre, etc.

3.2 Audience Characterization

In the second sub phase of the Audience Modeling, the Audience Characterization, relevant characteristics should be specified for each audience class. Examples of characteristics are level of experience with web sites in general, frequency of use, language issues, education/intellectual abilities, age, income, lifestyle…. Some of the characteristics may be translated into usability requirements while others may be used later on in the Implementation Design phase to guide the design of the “look and feel”

of the navigation tracks of the different audience classes, e.g., choice of colors, fonts, graphics, etc.

The target users of the IMDb example are a very broad audience; they don't have very specific characteristics. There are also no difference worth mentioning between the characteristics of the audience class Game Lover and the audience class Movie Lover. Therefore, the characteristics for all audience classes are specified as follows:

Characteristics for all audience classes IMDb example:

- Able to communicate in English
- Have reasonable experience with the Web
- Young people or adults

4. CONCEPTUAL DESIGN

So far in the method, the information-, functional-, usability- and navigational requirements as well as the characteristics 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 or semi-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" is covered by the Task & Information Modeling sub phase and deals with the modeling of the content and functionality of the web system; the conceptual "how" is covered by the Navigational Design sub phase and specifies the conceptual structure of the web system and the navigation. We describe these two sub phases into more detail in the next subsections.

4.1 Task & Information Modeling

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 descriptions, called *object chunks*, which model the information and functionality needed to satisfy these requirements. These conceptual descriptions are integrated into an overall conceptual model. This approach is used because WSDM follows the audience driven design philosophy. It has the following advantages:

- The developer is forced to concentrate on the actual needs of the users rather than on the information (and functionality) already available in the organization. In this way, the chance that information is missing in the

actual system will be less than in a data-driven approach where the available data is taken as the starting point. In addition, the information and functionality already available in the organization is not necessarily what the users need. Also 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 and that it may be necessary to use different structures or terminology for different types of users. By modeling the requirements for each audience class separately we can give due consideration to this.

The output of the Task & Information Modeling phase is a collection of *task models* and associated *object chunks*. We first explain the task modeling and afterwards the information modeling.

4.1.1 Task Modeling

The purpose of the Task Modeling is to model in detail the different tasks the members of each audience class need to be able to perform, and to formally describe the data and functionality that is needed for those tasks. The tasks that a member of an audience class needs to be able to perform is based on the requirements formulated for the audience class during Audience Classification, i.e. for each information and functional requirement formulated for an audience class, a task is defined that should allow to satisfy this requirement. Each task is modeled into more details using an adapted version of the task modeling technique CTT (Paterno et al., 1997 and Paterno, 2000). Essentially, in CTT, tasks are decomposed into subtasks until elementary tasks are obtained. In addition, temporal relationships between subtasks are specified to indicate the order in which the subtasks need to be performed. The result is a *task model*.

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). CTT also 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 Web and Web design:

1. WSDM does 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. This means that we only use the following categories of tasks (see figure #-5 for the graphical notation):

- *Application tasks*: tasks executed by the application. Application tasks can supply information to the user, perform some calculations or updates, e.g., checking username and password is typically an application task.
- *Interaction tasks*: tasks performed by the user by interaction with the system, e.g., entering information using a form.
- *Abstract tasks*: tasks that consist of complex activities, and thus require decomposition into sub tasks, e.g., ordering tickets for a movie.

2. A (complex) task is decomposed into (sub)tasks. 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. WSDM does not follow this rule. We 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 at a conceptual level who will initiate a subtask, or who will make a choice between possible subtasks.
3. CTT has a number of operators to express temporal relations among tasks. For some of the operations, we have changed the meaning slightly and an extra operator for transactions has been added:
   - *Order independent* (T1 |=| T2): the tasks can be performed in any order.
   - *Choice* (T1 [] T2): one of the tasks can be chosen and only the chosen task can be performed.
   - *Concurrent* (T1 ||| T2): the tasks can be executed concurrently.
   - *Concurrent with information exchange* (T1 |[]| T2): the tasks can be executed concurrently, but they have to synchronize in order to exchange information.
   - *Deactivation* (T1 [> T2): the first task is deactivated once the second task is started.
   - *Enabling* (T1 >> T2): the second task is enabled when the first one terminates.
   - *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.
   - *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.
   - *Iteration* (T*): the task can be performed repetitively. In CTT 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 in WSDM is that the task can be

repeated several times and ends when the one in charge decides not to repeat the task (e.g., the user who decides not to re-do the task 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.
- *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.
- *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").

4. In WSDM, the level of detail provided in the task model is less than in the original CTT method. The reason for this is the use of the object chunks. As we will explain, with each elementary task an object chunk can be associated which further describe the task in terms of information and functional needs.

Figure #-5. Graphical Notation for the Different Types of Tasks

We illustrate the task modeling for some of the requirements formulated for the IMDb example. Figure #-6 shows the task model for the tasks defined for the requirement "To find a movie by means of a search function" of the audience class Movie Lover. For this requirement, the task "Search IMDB" is defined. This abstract task is decomposed into three sequential tasks: "Specify Query", "View Results", and "Show Movie". The task "Show Movie" is optional and is further decomposed into "Show Movie Info" (to show the information like title, director, etc) and "Provide Extras". This task allows the user to choose between "Show Photos" (to display photos of the movie), "Add to My List" (to add the movie to the user's personal movie list) and "Post Message" (to post messages on the message boards associated with the movie). The task "Add to My List" is composed of an application task "Update My List" that will add the movie to the personal movie list of the user, and the task "Manage My List". To not overload the figure, the

abstract tasks “Manage My List” and “Post Message” are not further elaborated in this CTT.

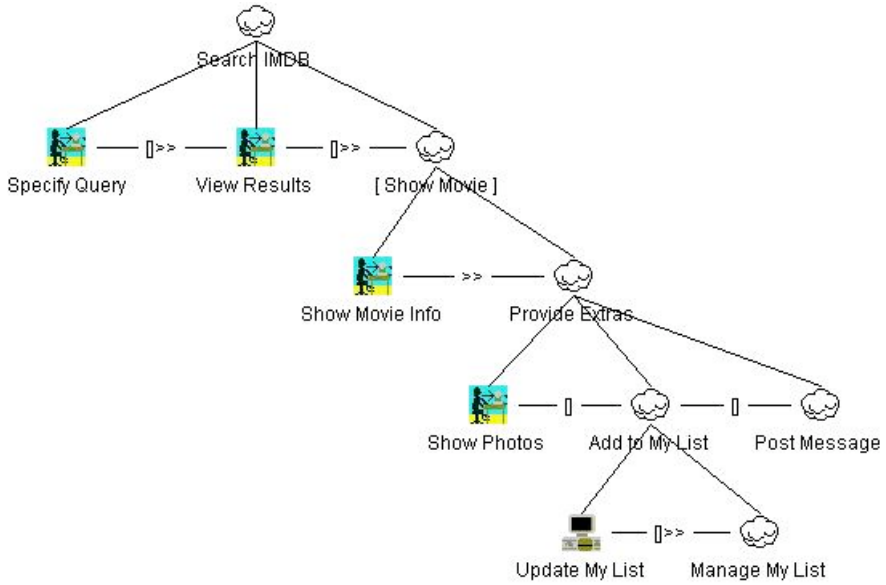

Figure #-6. Task Model for the Task "Search IMDB"

Figure #-7 shows the task model for the task defined for the requirements “To obtain show times of movies currently playing” and “To buy tickets for a show time in a selected cinema” of the audience class Movie Lover. We decided to support both requirements by a single task, as buying tickets requires knowing the show time, and offering the possibility to buy tickets while exploring show times may stimulate the sale of tickets. The task “Showtimes & Buy Tickets” is decomposed in two sequential tasks. First, the location must be specified using the “Specify Location” task. This is done by means of an interaction task to enter the location and the movie(s) (task “Enter Location Movie”), optionally followed by an interaction task to choose the location if more than one location exists for the information entered (Task “Choose Location”). Next, the user can explore the show times and optionally buy tickets by means of the task “Select Showtime & Buy Tickets”. This task is decomposed into the task “Explore Show Times”, which can be repeated, followed by the optional task “Buy Tickets”. For the task “Explore Show Times”, first the show times associated with the requested location, movie(s), and date are showed (“View Showtimes”) and then the user may change these parameters (“Change Parameters”) and obtain the show times again.

The task models created in this way, allow a first level of description of the functionality to be provided by the web system (i.e. they describe a kind of workflow). More details are given by means of the object chunks. The object chunks are described in the next section.

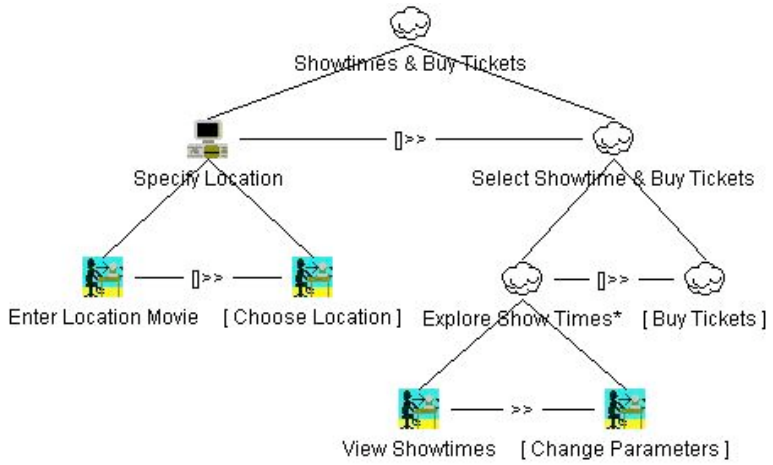

Figure #-7. Task Model for the Task "Show Times & Buy Tickets"

4.1.2 Information Modeling

When a task model is completed, an object chunk is created for each elementary task in this model. The main purpose of an object chunk is to describe formally the information and functionality needed by the user when he has to perform the associated task. If the requirement associated with the task is a pure informational requirement (i.e. the user is only looking for information; he doesn't need to perform actions) then the object chunk can be considered as the conceptual description of the information that will be displayed on (a part of) the screen. For this purpose a standard conceptual modeling language is sufficient. However, to be able to deal with functionality (e.g., filling in a form and process it), also a data manipulation language is needed. We first discus the modeling of information requirements and then discus what is needed to allow modeling functionality.

WSDM uses OWL[2] to model the information needs. OWL is becoming the standard for the Semantic Web. Its use as specification language for the objects chunks allows an easy integration with and use of existing domain ontologies and allows making the objects chunks available as local application ontology in case no relevant domain ontology already exists. It also provides the basis for the generation of semantic annotations (see section 6). Note that, as there isn't yet a generally used and compact graphical notation for OWL, we use the ORM graphical notation (Halpin, 2001) to give a graphical representation of OWL. We have opted for the ORM notation because ORM is very close to OWL and therefore the mapping from ORM to OWL is straightforward. In addition, because of the purpose of the object chunks, there is no need to specify any of the advanced types of restrictions supported by OWL. An ORM data type is graphically represented as a dotted circle, an ORM object type as a solid circles, an ORM subtype is connected to its supertype object type by means of an arrow, roles are represented as boxes connected to their respectively data type or object type, a mandatory constraint on a role is represented as a black dot on its connection, and a uniqueness constraint is represented as an arrow over one or two role boxes. See figure #-8 for some examples. The mapping from ORM to OWL is sketched in table #-2. Suppose L is an ORM data type; N, N', N1 and N2 are ORM object types; and r and r' are roles. Informally, we can state that an ORM object type is mapped on an OWL class; an ORM role connected to a data type to an OWL datatype property; and an ORM role connected to an object type to an OWL object property.

As already indicated, the use of OWL allows an easy way of coupling the concepts used in the object chunks to concepts in existing (external) ontologies. This coupling is later on (in the implementation phase – see section 6) used to generate semantic annotations. To refer in an object chunk to concepts defined in ontologies, the namespace mechanism of OWL is used. To refer to a concept in an ontology, the identifying prefix of the ontology is used to qualify the names of the concept. For example, "FOAF:Person" refers to the class Person defined in the ontology identified by the prefix FOAF.

Figure #-8 shows an example object chunk "ShowMovie". This object chunk is associated with the elementary task "Show Movie Info" of the task model "Search IMDB" given in figure #-6. In this object chunk the use of two external ontologies is illustrated: a basic IMDB ontology[3] (prefixed with "IMDB") and the well-known FOAF ontology[4] (prefixed with "FOAF" and used to describe persons). For example, the classes "IMDB:Movie",

[2] www.w3.org/TR/owl-features
[3] http://www.csd.abdn.ac.uk/~ggrimnes/dev/imdb/IMDB.rdfs
[4] http://www.foaf-project.org/

"FOAF:Image", "IMDB:Genre", and "FOAF:Person" refer to classes from these ontologies. Also properties can refer to properties defined in ontologies e.g., "IMDB:genres" refers to such a property.

Table #-2. Mapping between ORM and OWL

ORM	OWL
N	<Class rdf:ID="N"/>
N' subtype of N	<Class rdf:ID="N'"> <subClassOf rdf:resource="#N"/> </Class>
(*N1, r, L*)	<DatatypeProperty rdf:ID="r"/> <Class rdf:about="#N1"> <subClassOf> <Restriction> <onProperty rdf:resource="#r"/> <allValuesFrom rdf:resource="L"/> </Restriction> </subClassOf> </Class>
(N1, r', N2)	<ObjectProperty rdf:ID="r"/> <Class rdf:about="#N1"> <subClassOf> <Restriction> <onProperty rdf:resource="#r'"/> <allValuesFrom rdf:resource="N2"/> </Restriction> </subClassOf> </Class>
(r,r')	<Property rdf:about="#r"> <inverseOf rdf:resource="#r'/> </Property> (for object properties only)
Mandatory role r	… <Restriction> <onProperty rdf:resource="#r"/> <someValuesFrom rdf:resource="…"/> </Restriction> …
Uniqueness constraint of role r	… <Restriction> <onProperty rdf:resource="#r"/> <maxCardinality> 1 </maxCardinality> </Restriction> …

To allow communication between tasks, parameters (input- as well as output parameters) can be specified for object chunks. E.g., the object chunk "ShowMovie" (figure #-8) has an instance of type "IMDB:Movie" as input parameter (represented by *m). Input parameters are in general used to

restrict the information that should be presented to the user. E.g., the input parameter *m of type IMDB:Movie is used to express that only the information related to this particular movie should be shown. This is graphically represented by putting this parameter in the corresponding class. In fact, placing the parameter *m in the class "IMDB:Movie" restricts the complete schema to a view.

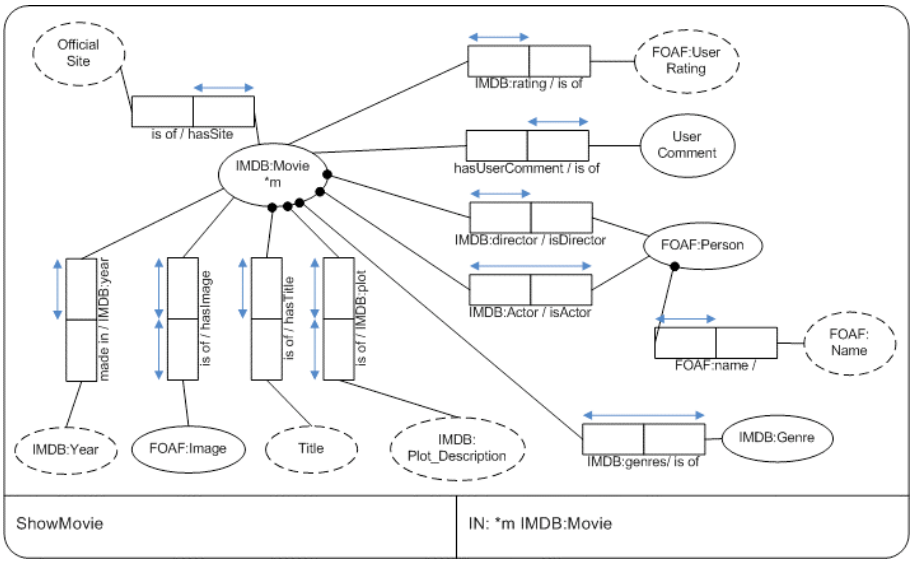

Figure #-8. Object Chunk "ShowMovie"

To be able to deal with functionality (e.g., fill in a form, update, add or delete information), a (conceptual) data manipulation language is needed. WSDM provides a graphical conceptual data manipulation language. However, notice that this language has limited expressive power and has not the intention to allow specifying complex functionality. We don't believe that a graphical language is appropriate for this. However, using the primitives provided by the language, most commonly used functionality for web systems, such as adding, deleting and changing information, can be specified. For more complex functionality, WSDM supports the use of (external) web services. More in particular, the conceptual data manipulation language of WSDM provides support for:

- Specifying that the user can select one or several instances from a class or a property (e.g., to allow the user to select an actor from the list of available actors in the IMDb example). For this the symbols "!" (single selection) and "!!" (multiple selection) are used.

- Expressing interactive input (e.g., needed to fill in a form). 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 (datatype properties). Class instances cannot be entered directly. They should be created using the "NEW" operator (see next bullet)
- To manipulate the data itself, a number of primitive operators are available. "NEW" indicates the creation of a new class instance; "REMOVE" is used to indicate the removal of one or more class instances; the symbol "+" above a relation indicates the addition of a property (and its inverse) and the symbol "-" the removal of properties;
- Furthermore, an assignment operator ("=") is available to assign values to variables (also called *referents*), as well as several built-in functions and default referents. For example, *USER is available to refer to the current user of a session.

A more detailed description of this graphical language can be found in (De Troyer and Casteleyn, 2001) and (De Troyer et al., 2005).

Figure #-9 shows an object chunk involving functionality: adding a movie to the personal movie list of the user. This object chunk is created for the elementary (application) task "Update My List" in the task model "Search IMDB" given in figure #-6. The movie-instance that needs to be added to the personal list is given by means of the input parameter *m (and passed to this task after the user has opted for the task "Add to My List" in the task "Show Movie" were the user was viewing a certain movie). The list to which the movie needs to be added is denoted by the referent *l and refers to the "My List"-instance that "belongs to" the "User"-instance *USER (which is the pre-defined referent used to refer to the current user). The "+" below the object properties "is in" and "has" specifies the addition of the relationship (i.e. instantiation of both object properties for *l and *m). Note that *l is returned as output parameter because in the task "Add to My List" this information need to be passed to the task "Manage My List".

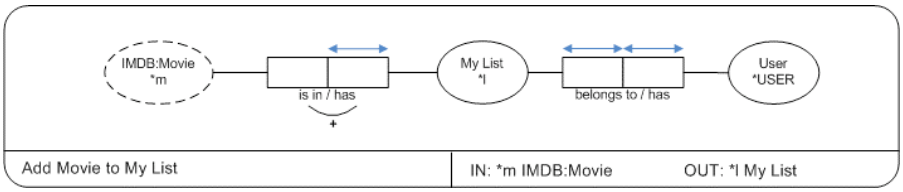

Figure #-9. Object Chunk "Add Movie to My List"

4.2 Navigational Design

The goal of the Navigational Design is to define the conceptual structure of the web system and to model how the members of the different audience classes can navigate through the web system and perform their tasks. Because of the audience driven approach of WSDM, a *navigation track* is created for each audience class. Such an audience track can be considered as a subsystem containing all and only the information and functionality needed by the members of the associated audience class. The internal structure of an audience track is derived from the task models made for this audience class. In addition, navigational requirements formulated during audience modeling are also taken into account. In the next subsection we explain in detail how a navigational track is created. Next, all audience tracks are combined into a basic conceptual navigation structure by means of so-called *structural links* (see section 4.2.2). In WSDM, the structure defined between the audience tracks corresponds to the hierarchical structure defined between the audience classes in the audience class hierarchy.

Once the main conceptual navigation structure has been derived, so-called *semantic* and *navigational aid links* are added (see also section 4.2.2). Semantic links are navigational links based on semantic relationships that exist between objects in the domain and which are modeled in the object chunks by means of object properties. Semantic links express task independent navigation (which may have been expressed in the form of navigational requirements during Audience Modeling). Navigational aid links are links that enhance navigation, such as home links, landmarks, quick links, etc. In contrast to structural links and semantic links, navigational aid links are strictly speaking not necessary but are added to enhance the usability of the web system. A more detailed discussion on the different types of links used in WSDM can be found in (De Troyer and Casteleyn, 2003a) and in (De Troyer and Casteleyn, 2003b).

The output of the navigational design phase is the *navigational model*. The navigational model is expressed in term of *components* and *links* between components. Components can be considered as (conceptual) navigation units that group the information/functionality conveyed in one or more object chunks. As indicated, WSDM distinguishes between structural links, process logic links, semantic links, and navigational aid links. The process logic links express part of a workflow or the invocation of (external) functionality (e.g., a web service). In general, a link may be defined from one component to one (other) component (one-to-one link), but also from one component to a set of components (one-to-many link), or from a set of components to one single component (many-to-one link), or from a set of components to a set of components (many-to-many link). As typical

implementation formats, such as HTML do not support one-to-many, many-to-one or many-to-many links, these kinds of links need to be implemented as a collection of one-to-one hyperlinks when generating (HTML) output. However, these kinds of links are useful to consider during conceptual design because they allow abstracting from the current implementation limitations and provide more semantics. For example, in the Presentation Design (see section 5.2) a one-to-many link can be represented as a single menu, and structural annotations can be generated to indicate the semantics of the links (see section 6).

Note that the links specified in the navigational model are actually link types. That means that even a one-to-one link may result in different hyperlinks in the actual web system. For example, if we specify in the navigational model that the user can navigate from a "Movie"-component to an "Actor"-component, this is modeled by means of a one-to-one link between these two components. However, a movie may involve several actors and therefore, for each individual movie the one-to-one link may give rise to several hyperlinks, one to each of its actors.

Links can have parameters to indicate that relevant information should be passed from the source component to the target component when a user will follows the link. A parameter is usually an output parameter from an object chunk connected to the source component.

Next to parameters, also conditions can be specified for links. A condition allows restricting the availability of the link for e.g., different users, devices or timeframes. For example, a link may become unavailable after a certain date, or only users that are logged in are presented certain links. To some extend, conditional links allows for adaptation of the web system.

The graphical representation of components and the different kinds of links are given in figure #-10. An external component refers to an external system or a web service.

Note that the navigational model only provides the *conceptual* structure (including navigation) of the web system. The mapping of this conceptual structure onto (web) pages and hyperlinks is specified during Site Structure Design, which is part of the Implementation Design phase (see section 5.1).

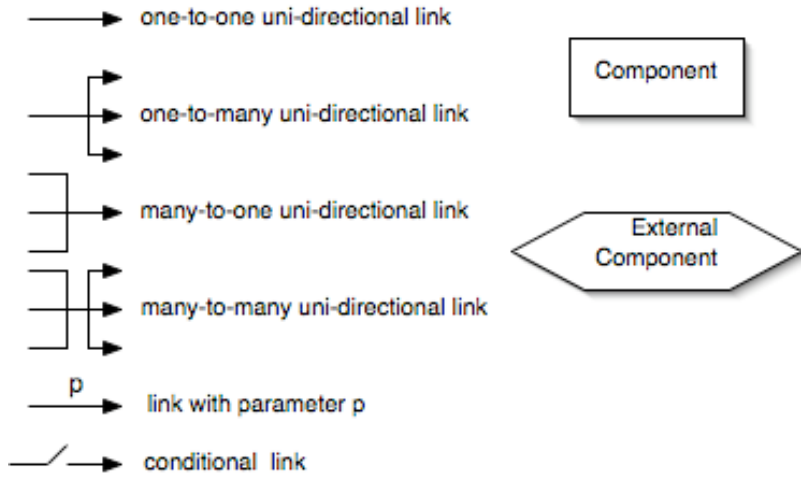

Figure #-10. Graphical Representation of Components and Links

4.2.1 Creating the Navigational Tracks

To create a navigational track for an audience class, for each task a *task navigational model* is created. A task navigational model is the translation of the task model (represented by means of a CTT) into a navigational structure.

A task navigational model is created by defining a component for each elementary interaction task defined in the hierarchical decomposition of the task. The object chunk that was created for the elementary task is connected to this component. See figure #-11 for the graphical representation. Also object chunks created for application tasks can be attached to components. In fact, components are a kind of placeholders for the object chunks (which represents the actual information and/or functionality). By linking components instead of object chunks, it is possible to use the same object chunk in different task navigational models and even in different navigational tracks without loosing the modeling context of the different links.

Next, process logic links between components are used to express the workflow or process logic, which is expressed in the task model by means of the temporal CTT relations. In fact, the temporal CTT relations are translated into links (one-to-one, one-to-many, many-to-one or many-to-many links; conditional- or non-conditional links). Components and process logic links can be grouped into a *transaction* to indicate that they constitute a conceptual unit (for which the all-or-nothing property holds). To avoid

complex diagrams, complex subtasks may be modeled separately, in so-called *Subtask Navigational Models*.

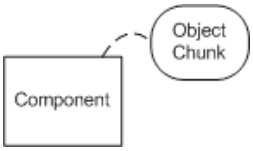

Figure #-11. An Object Chunk connect to a Component

Until now, we permitted the designer to neglect the fact that not all information and functionality should be freely available. In many web systems, parts of the information and functionality need to be protected in some way. This can be modeled during the Navigational design by means of a *protection area*. Graphically, this is represented by including the component(s) (together with their associated object chunks) that need to be protected into a named box labeled with a key symbol (an example can be found in figure #-12). In a similar way, it is possible to indicate that some information transfer needs to be secure. This protection area concept for expressing security and validation, allows abstracting (in the different navigational models) from how to achieve the validation or security. If relevant, how this must be achieved can be specified by means of a separated navigational model. An example is given in figure #-14 and is explained later on.

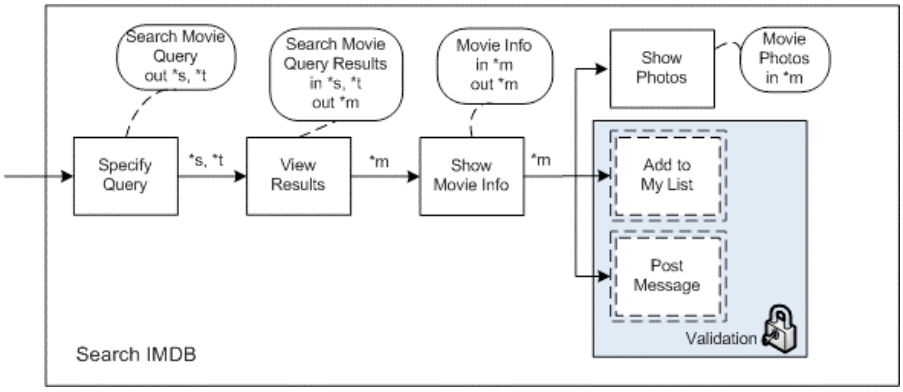

Figure #-12. Task Navigational Model for the Task "Search IMDB"

We illustrate the task navigational models by means of some examples. Figure #-12 shows the task navigational model for the task “Search IMDB”.

Its task model was given in figure #-6. The navigational sub task models "Add to My List" and "Post Message" are protected by the "Validation" protection area. These parts of the web system can only be accessed after the user has been authorized. How this must be done is specified in the task navigation model given in figure #-14, which has been derived from the task model given in figure #-13.

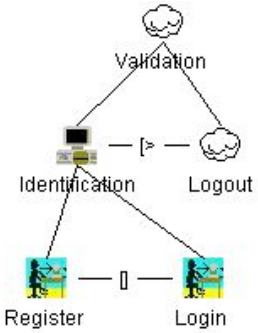

Figure #-13. Task Model "Validation"

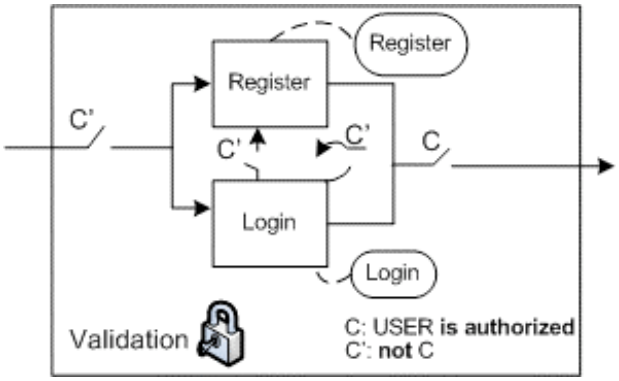

Figure #-14. Task Navigational Model for "Validation"

Figure #-15 gives the task navigational for the task "Showtimes & Buy Tickets" for which the task model was given in figure #-7. Note the use of an external component "Buy Tickets" to indicate that this is handled by an external service. Link parameters, input and output parameters are omitted in this diagram.

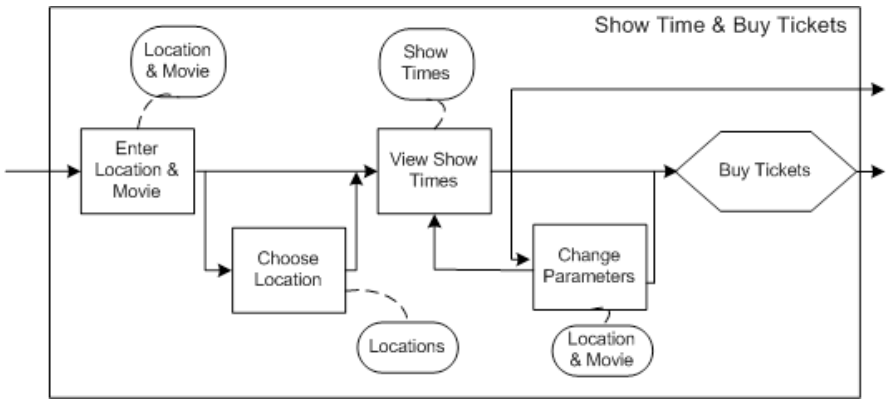

Figure #-15. Task Navigational Model for the Task "Showtime & Buy Tickets"

Once the task navigational models are constructed for the different tasks of an audience class, they should be composed into a navigational track using structural links. This can be done by first making a task model of how the members of the audience class are allowed to select the different tasks, and then translating this task into a task navigational model. In the simple case that, at any moment in time, a member of the audience class can freely select between the different tasks, this modeling process can be reduced to the introduction of a new component that is linked to the different task navigational models by means of a one-to-many link. This principle is illustrated in figure #-16, which shows the navigational track for the Game Lover audience class of the IMDb example. For the sake of simplicity, the navigational models are represented by means of their shorthand notation (dotted double-lined rectangles). The newly introduced component is not connected to an object chunk because it does provide any information or functionality itself. Instead, its sole role is to allow navigation to the different tasks. In the actual web system, this may result in a menu that provides links to the different tasks. However, if there are a lot of tasks for an audience track, it may be better (from a usability point of view) to structure the tasks in some way and to provide groups of tasks (which may result in groups of menu's).

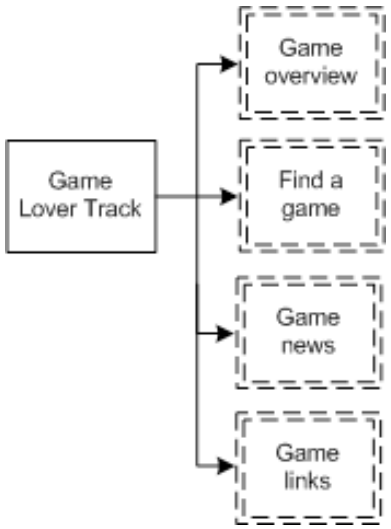

Figure #-16. Navigational Track for the Game Lover Audience Class

4.2.2 Creating the Navigational Model

Once the navigational tracks for the different audience classes are constructed, they need to be composed into a single structure. This will be the main conceptual structure of the web system. Because WSDM is audience-driven, the structure between the audience tracks must correspond to the hierarchical structure defined between the audience classes in the audience class hierarchy. We illustrate this with the IMDb example. The navigational model for the IMDb example is given in figure #-17. Note that for space limitations the different task navigational models are given by means of their shorthand notation and that the navigational track for the Movie Lover is also given by means of its shorthand notation (a double-lined rectangle). Note the correspondence with the audience class hierarchy given in figure #-4.

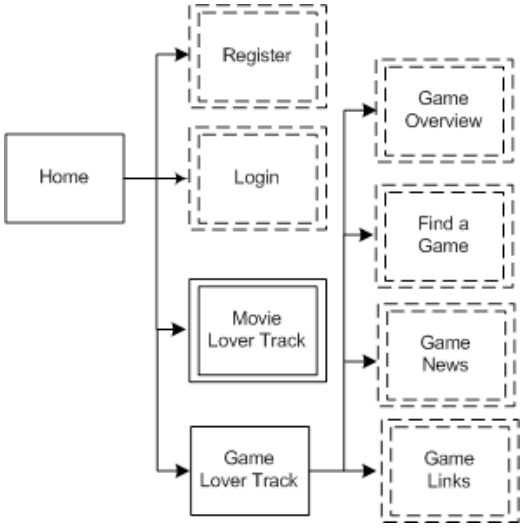

Figure #-17. Main Conceptual Navigation Structure for the IMDb Example

During Navigational Design, we also define semantic links that will enhance the navigation. Semantic links are based on semantic relationships that exist between objects in the application domain and which are modeled in the object chunks by means of object properties. For example, in the IMDb example, there are semantic relationships between movie and actor, between movie and director, and between movie and cinema. Also, in the navigational requirements for the Movie Lover audience class it was stated that the user should be able to navigate directly from the movie to the information of each of these related items. This facility can be modeled by means of semantic links. A semantic link is a link between two object chunks and must be based on the existence of a semantic relationship between two classes (e.g., movie having director). The source object chunk should contain the object property that expresses this semantic relationship between the two classes (movie having director). The target object chunk should provide the request information (information about director). We illustrate this with an example. Consider the object chunk "ShowMovie" as shown in figure #-8. For a movie we decided to provide the name of the director (modeled in "ShowMovie"). Suppose that the object chunk "Director Info" models the information provided for a director. Then, based on the object property "IMDB:director" we can define a semantic link from the object chunk "ShowMovie" to the object chunk "Director Info". See figure #-18 for a graphical representation. If needed, the link can be labeled with the name of the property used. Semantic links are independent of a particular task. This means that in each task where the chunk "ShowMovie" is used, the link will be available. Therefore, they don't need to be

represented in the different task navigation models and they will not overload these diagrams.

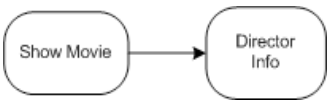

Figure #-18. Graphical Representation of a Semantic Link

In the IMDb example many semantic links are possible. In figure #-19, a selection of possible semantic links is given. Note that in this example all links are bi-directional because the user must always be able to navigate to more information if this is available. E.g., a link from the movie page to the director page must be available and vice versa.

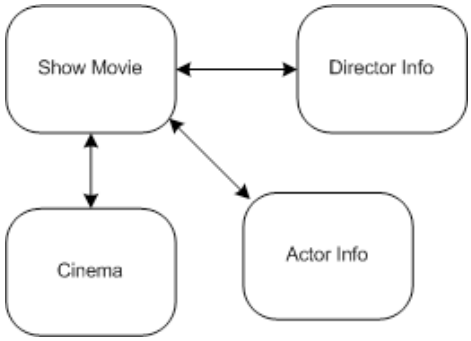

Figure #-19. Some Example Semantic Links for the IMDb Example

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 links, navigational aid links can be added to ease the navigation even more and to enhance the usability. From the viewpoint of being able to reach information and functionality, they are strictly speaking not needed; all information and functionality should also be reachable by means of the navigational tracks. Navigational aid links can 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 to access the information more easily. A typical example of a navigational aid link is the home link, which can often be found on each page of a web site. Also landmarks are examples of navigational aids links. Not to overload the diagrams, the home component and the landmark components are represented by means of a symbol. Later on, during the implementation

phase, home links and landmark links can be generated. In Figure #-20, the conceptual structure of the IMDb example is enhanced with navigational aid links (home and landmarks and an link from the "Login" navigational task model to the "Register" navigational task model).

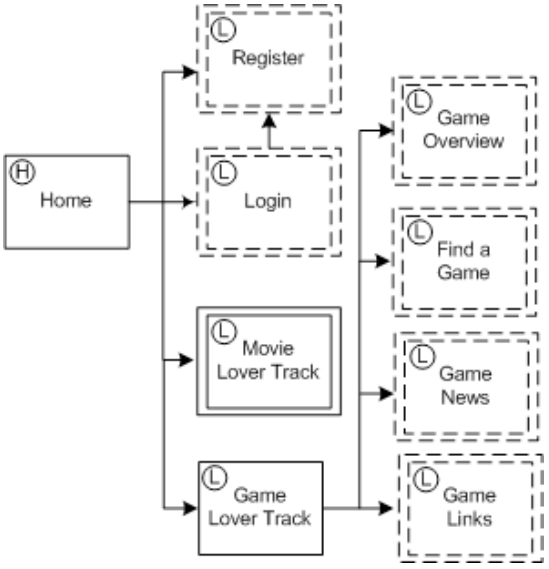

Figure #-20. Navigational Aids Links

5. IMPLEMENTATION DESIGN

The goal of the implementation design is to complement the conceptual design with the necessary details for the implementation. In principle, it would be possible to generate an implementation from the conceptual design but this is not realistic for several reasons. Firstly, the Web is very visual oriented and the standards for presentation have become very high in the last years. Especially professional web systems need to have a professional look and feel and usually graphical designers are involved to achieve this. If the web system is generated directly from the conceptual design, only standard and rather simplistic presentations can be obtained. Therefore, a presentation design is needed. Secondly, the information provided on the web system may already be available from some data source (e.g., a relational database). In this case, no new data source needs to be created but a mapping should be defined from the conceptual description of the information (the object chunks) to the actual data source. Thirdly, Web users don't like to make

unnecessarily clicks (clicks that don't lead to new information) but on the other hand too much information on a single page will overload the page and also decrease the usability. Therefore, information and functionality should be grouped onto pages in such a way that a good balance is reached between the amount of information on a page and the number of clicks needed to reach information.

For these reasons, WSDM has an Implementation Design phase consisting of three sub phases: the Site Structure Design, the Presentation Design and the Logical Data Design. The following subsections describe these sub phases into more detail.

5.1 Site Structure Design

During Site Structure Design the designer decides how the components from the navigational model will be grouped into pages. The characteristics of the different audience classes may be taken into account when deciding which information to group on a page. E.g., for an audience class with the characteristic that the age is over 50, the designer might want to limit the amount of information on a single page. It is also possible to define different site structures for the same design, each supporting a different device. For a device with a small screen it may even be necessary to distribute the information related with a single component onto different pages.

By default each component (with its associated object chunks and links) is placed on a single page. However, the designer can decide to group different components on a single page or to use different pages for a single component. When components are grouped on a page, the designer should respect the conceptual structure expressed by means of the different links, e.g., if a component cannot be reached by a link from another component these two components cannot be placed on the same page.

The output of this phase is the *site structure model*. The site structure is graphically presented by drawing pages over the components that should be grouped. Figure #-21 illustrates a part of the site structure design for the IMDb example. The Home component and the links from this component is placed on a single page; Register and Login are each on a different page; and the components "Find a Game" and "Game Links" and all links coming from the "Game Lover Track" component are also put together on a single page. The rest has been left unspecified in this figure.

Note that the pages defined during the Site Structure Design are abstract pages. Each abstract page will give rise to one or a set of concrete pages when the actual implementation is generated. For example the page containing the "ShowMovie" object chunk will result (in case of a static web site) in many different concrete pages, one for each movie.

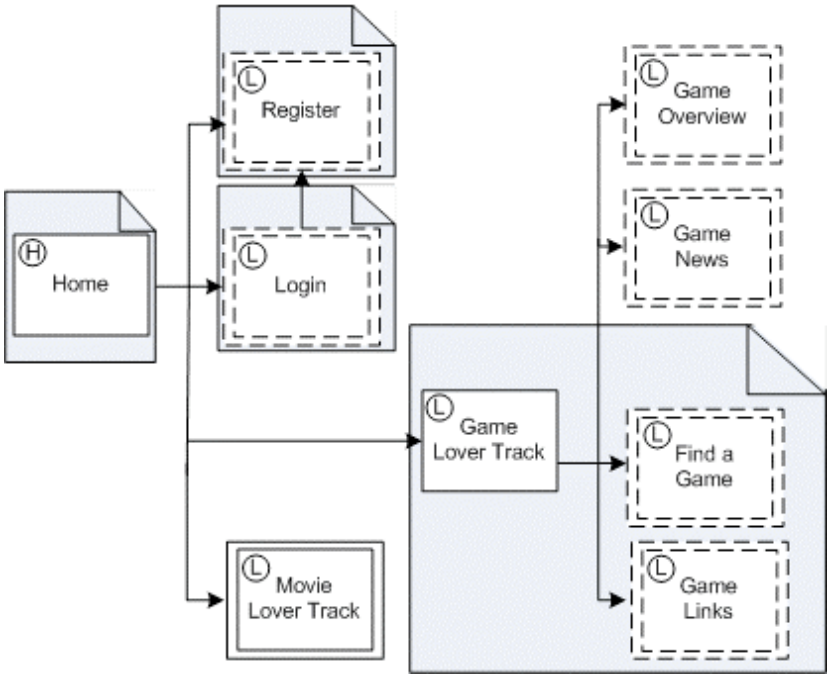

Figure #-21. Part of the Site Structure Design for the IMDb example

5.2 Presentation Design

During Presentation Design the look and feel of the web system, as well as the layout of the pages (i.e. positioning of page elements) is defined. To enhance a consistent look and feel, templates are used. Therefore, page templates are defined. Typically, a web system may require different kinds of pages e.g., a homepage, a title page, leaf page. For each of those page types a template can be created. These templates are subsequently used in the page design, when for each of the pages defined in the site structure model the layout is defined. The layout describes how the information and functionality (modeled by means of the object chunks and) assigned to a page (by means of the components) should be laid out on the page.

For both the template and page design a number of presentation modeling concepts are available. To position information, the concept of a *grid* is used. A grid contains *rows* with *cells*. A cell contains a *multimedia value*, another grid (nesting of grids), or a *referent* from an object chunk assigned to the

page (remember that a referent refers to an instance/value (or a set of instances/values) from a class or a property). Absolute and relative height and width can be specified for grids, rows and cells. By nesting grids, specifying width and height of the different grids, rows and cells, information can be positioned on the page.

The value of a cell can be associated with a *hyperlink* (which must be based on a link contained in the page and defined during navigational modeling). Furthermore, when a grid or cell is associated with a referent that represents a set (of object instances or values) then, in the actual implementation, the grid will be repeated for each instance of this set.

To display multimedia values correctly, additional properties may be required. For instance, an image and a video require a height and a width property.

Furthermore, also a number of high-level presentation modeling concepts are provided. The high-level presentation modeling concepts are more powerful and more intuitive for the designer. They also are useful to capture the semantics of a presentation element. Most of the concepts have a well-known meaning: *OrderedBulleteList*, *BulletList*, *Table*, *Menu*, *TableOfContent*, *BreadcrumbTrail*, *Section*, *Banner*, *Copyright*, *Advertisement*, *Figure*, *Icon*, *Logo*, *Marquee* (a string or an image that scrolls horizontally across the screen).

Forms are a widely used in web system. To support them the following control concepts are available: a select control to model that a selection can be made out of multiple options, an input control to model that a value can be entered, and an action control to specify that an action should be performed. These controls can be associated with a presentation concept. Types of select controls are a *RadioButton*, a *CheckBox*, a *ListBox* and a *DropDown* box. An input control is either a *TextBox* or a *SecretTextBox* (typically used for entering passwords). A typical action control is a *PushButton*. The behavior associated with an action control is defined by associating an event and an action to the control. It expresses the fact that when the specified event occurs for the associated presentation concept, the specified action will be performed. Possible events are *OnClick*, *OnLoad*, *onHoover*; possible actions are *PopUp*, *Show*, *Scroll*, *Reset*, *Submit, and Cancel*. A popup menu for instance can be defined using the Menu presentation concept with associated OnClick event and associated PopUp action; an expandable menu can be defined using the Menu presentation concept where the elements of the menu are associated with the event OnClick and action Show.

Templates are specified using the presentation modeling concepts mentioned. A template can also be composed out of a *Header*, *Footer*, *SideBar*, and/or *ContentPane*. Each template furthermore contains at least

one *editable region*. An editable region denotes an area that one needs to specify further when the template is used for a page design. An editable region can be placed anywhere in a grid.

To specify style, WSDM currently relies on Cascading Style Sheets (CSS)[5]. This allows style specification for any particular element and has enough expressive power to describe most styles commonly found in web systems.

For each page, the designer chooses a template and then specifies how the links and the information (specified by means of the object chunks) will be positioned in the editable regions of the template. This is done using the presentation modeling concepts mentioned. For each object chunk connected to a component included in the page, a grid is constructed. Each datatype property of an object chunk is placed in a cell of the grid. For functionality, control concepts are used. If needed, multimedia values can be added to enhance the presentation (e.g., titles, labels, graphics, etc).

For each link contained in the page, the designer needs to specify the anchor. This is done by adding the link to the relevant cell of the grid. Note how this linking mechanism does not differentiate between the type of anchor (e.g., a text element, an image, a table): a link is uniformly specified on a cell of a grid, no matter what its content is.

The characteristics and usability requirements of the audience classes should be taken into account when designing the different templates and pages.

The output of this phase is the *presentation model* consisting of a set of *templates*, a set of *styles* and for each page defined in the site structure model, a *page model*.

Figure #-22 shows a simple example template and figure #-23 a simple example page model.

[5] www.w3.org/Style/CSS/

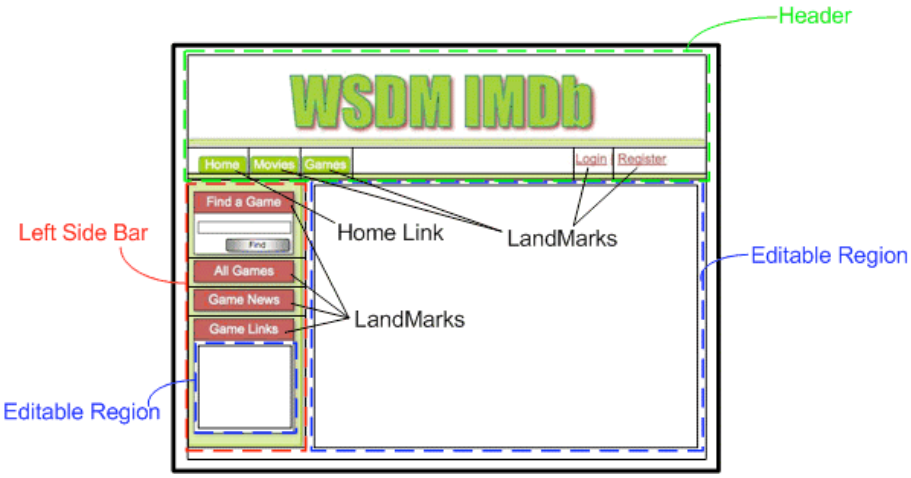

Figure #-22. Example Template for the IMDb example

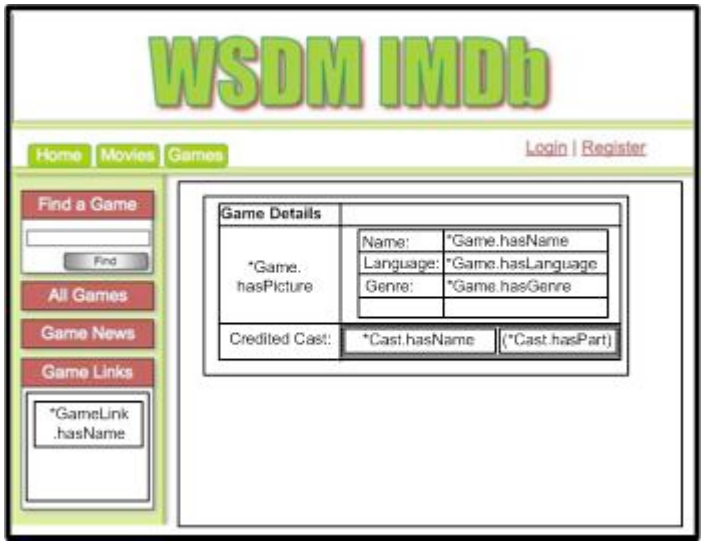

Figure #-23. Template Page Model for the IMDb Example

5.3 Logical Data Design

The information provided by the web system is described by means of the different object chunks made during Task & Information Modeling. The different object chunks are related by means of the *reference ontology* that

contains the different concepts used in the different object chunks. The objects chunks are views on the reference ontology that is incrementally constructed during Information Modeling. The reference ontology may be based on one or more external ontologies or created from scratch. This reference ontology can be considered as the conceptual schema for the data to be provided by the web system. In case no data storage is already available, a logical data schema needs to be created from this conceptual schema. This is comparable to the creation of a relational database schema from a conceptual ER schema or UML schema. The logical data schema can be a relational database schema, an XML schema, an RDF schema or even the OWL schema of the reference ontology itself. While generating the logical data schema, it is important to keep track of the mapping between the reference ontology and the logical data schema, because later on (in the implementation phase) the conceptual queries and updates expressed in the object chunks need to be translated into queries and updates onto the logical database schema. Because of space limitations, it is not possible to describe in this chapter how a logical data schema can be generated from a reference ontology and how the mappings can be expressed. Normally, this process should be supported by a CASE-tool, in which case the designer is not burdened with the creation of the logical data schema and the mappings.

In a second scenario, an existing data store is available. In this case it is only needed to define the mapping between the reference ontology and this data store.

The output of this phase is a *logical data schema* and a *data source mapping*.

6. IMPLEMENTATION

The actual implementation can be generated automatically from the information collected during the different design phases by means of the different design models. As proof-of-concept, a transformation pipeline (using XSLT) has been defined, which takes as input the object chunks (with corresponding data source mapping), navigational-, site structure-, style & template- and page models, and outputs the actual implementation for the chosen platform and implementation language. This transformation is performed fully automatically. A description of this transformation pipeline is out of the scope of this chapter. An overview can be found in (Plessers et al., 2005b). An example of page (showing game details) is given in figure #-24.

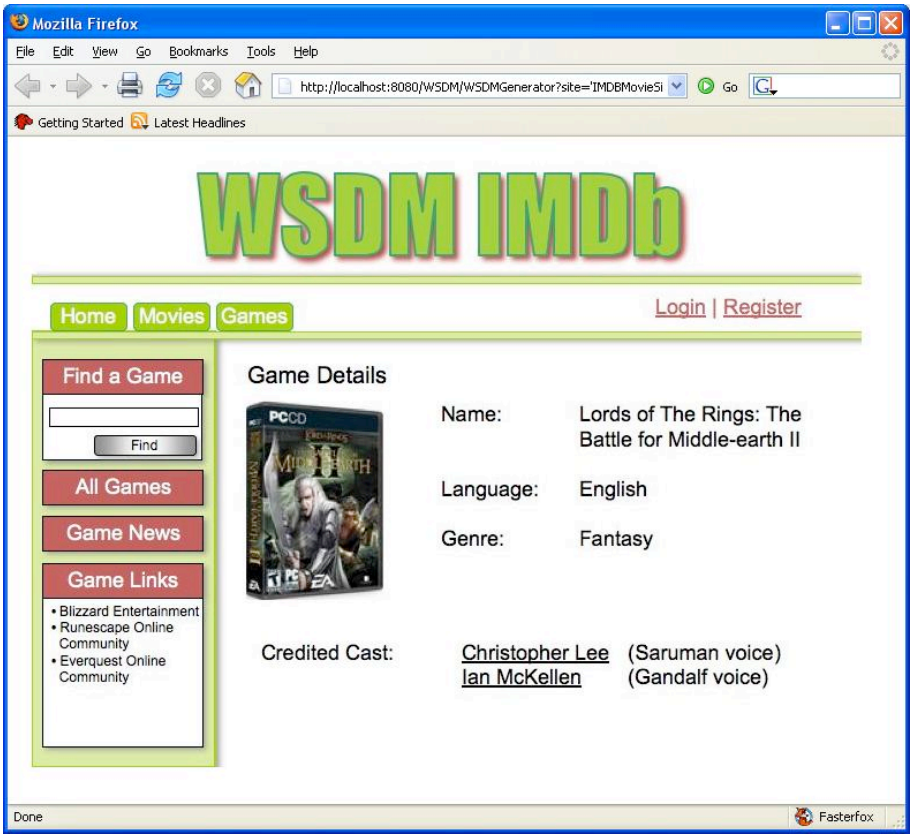

Figure #-24. Example Page from the IMDb example

Important to notice is that, based on the design information collected, semantic annotations can be generated. More in particular, the use of one or more external ontologies during the Conceptual Design allows expressing explicitly the semantics of the presented data by means of content-related semantic annotations using these ontologies. However, it is also possible to annotate the web system such that also the semantics of its structure are made explicit. Dedicated ontologies (e.g., the WAfA ontology (Yesilada et al., 2004) developed in the context of accessibility for visually impaired user) can be used to make the semantics of the different structural elements (e.g., a navigation menu, a logo, an advertising banner) explicit. These so-called structural annotations can be exploited by third parties that require specific knowledge about the web system's structure: page transcoders to transcode a webpage in e.g., a format more appropriate for screen readers or search engine indexers. Structural annotations can be generated without any additional effort from the designer by exploiting the design information captured by means of the design models. How the content-related and

structural annotations are generated is out of the context of this chapter. A description of this can be found in (Plessers et al., 2005a), (Plesssers et al., 2005b), (Plessers and De Troyer, 2004a), and (Plessers and De Troyer, 2004b).

As an example, consider (a part of) the generated content-related semantic annotations for a web page showing the movie details for "The Terminator" movie, based on the object chunk "ShowMovie" (recall figure #-8):

```
<IMDB:Movie rdf:ID="23">
<hasTitle>The Terminator</hasTitle>
<IMDB:year>1984</IMDB:year>
<FOAF:plot>A human-looking cyborg from the future …<FOAF:plot>
…
</IMDB:Movie>
```

When generating the actual data on a web page, span tags enclose the data originating from the data source. E.g., for the example above, the generated movie title and year is as follows:

```
<span id="1">The Terminator</span>
<span id="2">1984</span>
```

Finally, exploiting the mapping between the reference ontology and the actual data source (defined in Logical Data Design sub phase), the (HTML) code and the generated annotations are linked together:

```
page.html#xpointer(id("1"))<=>page.owl#xpointer(id("23")/hasTitle)
page.html#xpointer(id("2"))<=>page.owl#xpointer(id("23")/IMDB:year)
```

7. FURTHER ISSUES

WSDM has been extended in different directions. The most important ones are the extensions to support localization (De Troyer and Casteleyn, 2004) and the extensions for adaptation (Casteleyn, 2005).

We will briefly describe the principles used for these extensions.

7.1 Localization of Web Systems

Public web systems are accessible from all over the world. This offers opportunities for companies and organizations to attract visitors from across

the country borders and to do business with them. Two different approaches are possible to address this issue: develop one single web system to serve everyone or develop "localized" web systems for particular localities. The "one size fits all" approach may be appropriate for particular communities (like researchers) but in general it will be less successful. In general, it is recommended to localize a global web system, i.e. to create different versions and adapt those versions to the local communities they target. Members of a community do not only share a common language, but also common cultural conventions. Since measurement units, keyboard configurations, default paper sizes, character sets and notational standards for writing time, dates, addresses, numbers, currency, etc differ from one culture to another, it is self-evident that local web systems should address these issues. Some jokes, symbols, icons, graphics or even colors may be completely acceptable in one country, but trigger negative reactions in another country. Sometimes even the style or tone of the site's text might be considered offensive by a particular cultural entity, as a result of which the text needs to be rewritten rather than merely translated. Next to culturally differences, it may also be necessary to adapt the content to regional differences, like differences in the services and products offered, differences in price, and differences in regulations.

As for classical software, web system localization is often done once the web system is completely developed and available for a particular community. We believe that the globalization process[6] could benefit from taking localization requirements into consideration while designing the web system. Then, it may be easier to actually realize globalization because the internationalization activities[7] may already be considered and prepared for during the design process. For this reason, WSDM was extended to support web localization. We shortly explain how the different (sub) phases have been adapted.

The Mission Statement Specification

To be able to take localization into account during the design process, the mission statement should also mention the different localities for which the web system needs to be developed. A locality describes a particular place, situation, or location. Localities are identified by means of a name and a label. Examples of localities are: the US, Japan, and the Flemish community in Belgium.

[6] According to LISA (Localization Industry Standards Association) (http://www.lisa.org) localization of a thing is adapting it to the needs of a given locality. Globalization is about spreading a thing to several different countries, and making it applicable and useable in those countries.

[7] Internationalization consists of all preparatory tasks that will facilitate subsequent localization

As an example, suppose that next to the English version (which is targeted to the US), we also want localized versions of the IMDb web system for France and Germany. Then, the mission statement can be reformulated as follows: "To be the biggest and best movie and game site on earth. For movies, this will be achieved by providing as much information as possible on movies including their actors, directors and producers, as well as to provide news, allow exploring show times, buy tickets in selected cinemas in the US, and to share personal opinions about movies. For games, information about games is offered as well as news, and game lovers should be able to exchange information. Next to the US version, localized versions for France and Germany should be offered; information dependent on the country, such as the movies currently playing, should be adapted for each version. Exploring show times and buy tickets is only available for the US". Here, the localities that are targeted are US, France and Germany.

Audience Modeling

To support localization, a distinction is made between requirements and characteristics typical for an audience class and those typical for a locality.

The requirements and characteristics that are typical for a locality are related to the language, culture, habits or regulations of the locality. Some examples of locality requirements are: an address should always include the state; for each price it should be indicated if tax is included and if not the percentage of tax that need to be added should be mentioned. Locality characteristics will typically deal with issues as language use, reading order, use of color, and use of symbols.

Then, the localities are linked to the different audience classes. An audience class may span different localities. For example, in the IMDb example all classes identified so far are applicable for all localities, but in fact only people from the US should be able to explore show times and buy tickets. Therefore, we can refine the audience class hierarchy and introduce a new subclass for the people that can explore show times and buy tickets. Then, this class only needs to be supported in the US locality.

Conceptual Modeling

During Task Modeling, a task model is defined for each requirement. Now, we also have requirements formulated for the different localities. These requirements also need to be considered during task modeling. When constructing the task models, we need to inspect the locality requirements to check if additional or different steps are needed when decomposing a task. If a task must be completely different for a specific locality (which is rarely the case), a different CTT must be created and labeled with this locality. If only some additional steps are needed, then these steps are labeled with the localities for which they are needed.

Also when constructing the object chunks, we need to inspect the locality requirements to check if additional information is needed. If this is the case, this information is added to the object chunk and labeled with the locality for which it is needed. In the object chunks, we should also indicate which information is dependent on the locality. For example, in the IMDb example, the description of a movie need to be given in the language of the locality, also the movies currently played and the movies coming soon will be different for each locality. Labeling the classes and properties that are locality dependent indicates this.

In the Navigational Design, the audience tracks are labeled with all the localities for which they are applicable.

Implementation Design

Usually the Site Structure Design will be independent of the locality, i.e. for each locality the site structure will be the same. However, if some task models are very different for different localities, a different site structure may be needed.

During the Presentation Design the localization characteristics formulated during Audience Modeling need to be taken into consideration. Different templates should be created for different localities if this is needed (e.g., different colors, different labels).

When creating a logical data schema, we need to take into account that the information may be different for different localities as indicated by the labeling of and within the object chunks. Depending on the situation different data sources for each locality may be needed or only different field for some properties. Many different solutions are possible; we will not go into detail here. More information can be found in (De Troyer and Casteleyn, 2004).

7.2 Adaptation

WSDM provides flexible design-support for the specification of (automatic) re-organization of structure and content of the web system (at runtime), based on the way users access and use the web system. Note that this type of adaptation, also called *optimization* (Perkowitz and Etzioni, 1997) differs from personalization (which is what is usually intended when the term ‘adaptation’ is used): optimization improves the web system as a whole (for all users), where personalization adapts the web system for a single user (i.e. the current user). The possibility to take into account and anticipate during design, the actual use of the web system at runtime, offers the following advantages:

1. **Anticipate and react on runtime browsing behavior**: e.g., make popular pages more directly available (add navigational aids links)

2. **Automatic evaluation and use of design alternatives**: e.g., merge audience tracks if their separation seems to be less useful
3. **Detect and correct design flaws**: e.g., detect and correct misplaced information
4. **Better tailor the web system to satisfy business goals:** e.g., add or replace strategic business information in such a way that it appears on the most popular pages

To specify this type of adaptive behavior a dedicated language, called the Adaptation Specification Language (ASL), was introduced. ASL is a high-level rule-based adaptation specification language, which allows the designer to specify adaptation strategies (i.e. *which* adaptation needs to be done) and adaptation policies (i.e. *when* adaptation needs to be done). ASL is event-based: user-generated *events* (e.g., clicking a link, visiting a page, starting a session) will trigger the adaptation strategies. The strategies themselves are specified using *rules* (e.g., iterations, conditional execution of an action, pre-defined transformations on the relevant design models). By allowing the designer to specify which event(s) need to be tracked, and when and how adaptation should be performed based upon these events, the designer has a powerful mechanism to specify how the organization and structure of the web system should be improved (at runtime) based on the actual use of the web system. The remainder of this section explains an example of a useful adaptation strategy (and –policy), and highlights some interesting features of ASL. For an in-depth discussion of ASL (including formal specification, example strategies, experimentation results), we refer to (Casteleyn, 2005).

Consider as an example adaptation strategy the *promotion* strategy, discussed in the context of WSDM in (Casteleyn et al., 2003). Promotion of a component makes the component easier to find by moving it closer to the root (e.g., the homepage) of the web system. Here, the promotion is here based on the popularity of the components: the most popular component(s) (the component(s) with the highest amount of accesses) are promoted. ASL allows specifying for which components the amount of accesses needs to be tracked and how this should be done (i.e. per session, per load, per click). For the IMDb example, a useful adaptation strategy might be to promote the movie and game mostly visited (overall) during the past month. Therefore, the accesses to each individual movie and game need to be counted. A general script for counting the access to the elements of some set (of design elements) is used for this. This script can be used in different adaptation strategies:

```
script trackAmountOfAccesses(Set) :
  forEach element in Set
```

```
begin
   addTrackingVariable element.amountOfAccesses ;
   monitor load on element do element.amountOfAccesses :=
                                 element.amountOfAccesses + 1
end
```

Intuitively, the for-each rule in this script states that a tracking variable *amountOfAccesses* is declared (i.e. addTrackingVariable) and attached to each element of the given set. Furthermore, load events on the elements (i.e. for all users and for all sessions) will give rise to the increment of the *amountOfAccesses* tracking variable of that particular element.

The actual promotion in this case consists of linking the most popular movie and game to the root of the web system. First, a script implementing the general principal of promotion is given. Here, the original link(s) to the promoted component are kept (so only a navigational aids link is added). Alternative promotion strategies can be defined. The promotion script is specified as follows in ASL (note that in ASL the shorter term "node" is used instead of "component"):

```
script promoteNode(Set, promoteTo) :
begin
   let promoteNodeMaxAccesses be
      max(Set [MAP on element: element.amountOfAccesses]);
   forEach node in Set :
      if  node.amountOfAccesses = promoteNodeMaxAccesses
      then addLink (navigationAid, promoteTo, node)
end
```

Having defined the adaptation strategy, we are able to specify the adaptation policy, i.e. *when* the adaptation should be performed. In this example, we collect the accesses to movies/games for one month and perform a promotion once every month:

```
when initialization do
  begin
      call trackAmountOfAccesses(ALL MovieDetailNode);
      call trackAmountOfAccesses(ALL GameDetailNode)
  end

when 1 month from now do
  begin
      call promoteNode(ALL MovieDetailNode, root);
```

```
        call promoteNode(ALL GameDetailNode, root)
        reset(ALL MovieDetailNode
            [MAP on element: element.amountOfAccesses]);
        reset(ALL GameDetailNode
            [MAP on element: element.amountOfAccesses]);
    end
```

Note that the ALL keyword is used to obtain a set of all instances of a particular (conceptual) component. In this case, all concrete Movie- and Game Detail nodes (i.e. each individual movie or game detail page) are the subjects of the adaptation strategy. The first part of the adaptation policy specifies that, when the web system is initialized, the amounts of accesses to Movie- and Game Detail nodes are initialized. The second policy specifies that after one month (note that this will be repeated each month), the promotion strategy is applied, and the components containing the most popular movie and game are promoted to the root. Finally, all tracking variables are reset for a next month of tracking.

8. SUMMARY

WSDM is a semantic web design method based on an audience-driven design philosophy. This means, that not the data available in the organization or its internal organization, but the requirements of the target audience is the starting point of the modeling process. The different audience classes and their different requirements are also reflected in the actual structure of the web system. This approach is used to offer the designer a well-defined method to identify the information and functionality needed for a web system and to structure it in an appropriate way. This must prevent that the developers only provide information that happened to be available and structure it in a way that is only obvious for them. The method is based on the principle that a web system should be designed for and adapted to its target audiences.

The method also makes a clear distinction between the conceptual design and the implementation design. Issues like grouping of information and functionality in pages and graphical presentation and layout are not considered to be conceptual issues but implementation design issues, because more than one grouping into pages or more than one presentation design is possible for the same conceptual design.

Last but not least, WSDM allows developing web systems that are semantically annotated by means of one or more ontologies. Next to the regular content-related semantic annotations, also structural annotations can

be generated. These are annotations that describe the semantics of the different structural elements of the web system and can be exploited by other application to transcode the web system to formats more suitable for other purposes than human reading.

Furthermore, the clear separation of design concern by means of different design concepts and models, as well as a clear separation between conceptual issues and implementation issues have showed to pay: the modeling of a new design concern can easily been added. This has been demonstrated for adding localization and adaptation.

9. REFERENCES

Brusilovsky, P., 1996, Methods and techniques of adaptive hypermedia, In: *User Modeling and User-Adapted Interaction*, 6 (2-3), Springer Science+Business Media B.V, ISSN 0924-1868, pp. 87-129

Casteleyn, S., 2005, *Designer Specified Self Re-organizing Websites*, PhD Thesis, Vrije Universiteit Brussel

Casteleyn, S., De Troyer, O., 2001, 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*, Springer-Verlag, ISBN 3-540-44-122-0, pp. 198 - 211

Casteleyn, S., De Troyer, O., Brockmans, S., 2003, Design Time Support for Adaptive Behaviour in Web Sites, In: *Proceedings of the 18th ACM Symposium on Applied Computing*, ACM, ISBN 1-58113-624-2, pp. 1222 - 1228

Deng Cai, Shipeng Yu, Ji-Rong Wen, Wei-Ying Ma., 2004, Block-based web search, In: *SIGIR 2004: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, July 25-29, 2004, Sanderson, M, Järvelin, K., Allan, J, Bruza, P., eds., ACM, ISBN 1-58113-881-4, pp 456-463

De Troyer, O., Casteleyn, S., 2001, 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/), Oscar Pastor, ed., Valencia University of Technology, Spain

De Troyer, O., Casteleyn, S., 2003a, 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), Daniel Schwabe, Oscar Pastor, Gustavo Rossi, Luis Olsina, eds. Oviedo, Spain

De Troyer, O., Casteleyn, S., 2003b, Exploiting Link Types during the Conceptual Design of Web Sites, In: *International Journal of Web Engineering Technology (IJWT),* Vol 1, No. 1, Underscience Publishers, ISSN 1476-1289, pp. 17-40

De Troyer, O., Casteleyn, S., 2004, Designing Localized Web Sites, In: *Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE2004)*, Zhou, X., Su, S., Papazoglou, M.P., Orlowska, M.E., Jeffery, K.G., eds., Springer-Verlag, Brisbane, Australia, ISBN 3-540-23894-8, pp. 547 - 558

De Troyer, O., Casteleyn, S., Plessers, P., 2005, Using ORM to Model Web Systems, In: *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops, International*

Workshop on Object-Role Modeling (ORM'05), *Lecture Notes in Computer Science 3762* Springer, ISBN 3-540-29739-1, pp. 700-709

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

Halpin, T., 2001, *Conceptual Schema and Relational Database Design: From Conceptual Analysis to Logical Design*, Morgan Kaufmann

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

Paterno, F., Mancini, C., and Meniconi, S., 1997, ConcurTaskTrees: a Diagrammatic Notation for Specifying Task Models, In: *Proceedings of INTERACT 97*, Chapman & Hall,pp. 362-366Plessers, P., Casteleyn, S., De Troyer, O., 2005a, Semantic Web Development with WSDM, in: *5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005)*, Galway, Ireland

Perkowitz, M. and Etzioni, O., 1997, Adaptive web sites: an AI challenge, in: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 16-23

Plessers, P., Casteleyn, S., De Troyer, O., 2005a, Semantic Web Development with WSDM, In 5th International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2005), Galway, Ireland

Plessers, P., Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., Goble, C., 2005b, Accessibility: A Web Engineering Approach, In: *Proceedings of the 14th International World Wide Web Conference (WWW2005)*, Allan Ellis, Tatsuya Hagino, eds., ACM, Chiba, Japan ISBN 1-59593-046-9, pp. 353 – 362

Plessers, P., De Troyer, O., 2004a, Web Design for the Semantic Web, In: *Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, CEUR Workshop Proceedings, Vol 105 Web, WWW2004 Workshop*, Christoph Bussler, Stefan Decker, Daniel Schwabe, Oscar Pastor, eds., ISBN 1613-0073, New York, USA

Plessers, P., De Troyer, O., 2004b, Annotation for the Semantic Web during Website Development, In: *Proceedings of the ICWE 2004 Conference, Lecture Notes in Computer Science 3140*, Nora Koch, Piero Fraternali, and Martin Wirsing, eds., Springer, Munich, Germany, ISBN 3-540-22511-0, pp. 349-353

Yesilada, Y., Harper, S., Goble, G., Stevens, R., 2004, Screen Readers Cannot See (Ontology Based Semantic Annotation for Visually Impaired Web Travellers), In: *Web Engineering, 5th International Conference, ICWE 2005, Sydney, Australia, July 27-29, 2005, Proceedings*, *Lecture Notes in Computer Science 3579*, Springer, ISBN 3-540-27996-2, pp 445-458