Design Method Meta Model

Tom Vleminckx

Promotor: Prof. Dr. Olga De Troyer
Guiding assistants: Sven Casteleyn and Peter Plessers

A thesis presented for a
License Degree in Applied Computer Science

WISE - Web & Information System Engineering
Department of Applied Computer Science
Vrije Universiteit Brussel
Belgium
Acadademic Year 2004-2005

Ontwerp Methoden Meta Model

Tom Vleminckx

Promotor : Prof. Dr. Olga De Troyer
Begeleiders: Sven Casteleyn en Peter Plessers

Verhandeling voorgedragen tot het behalen van de graad
Licentiaat in de Toegepaste Informatica

WISE - Web & Information System Engineering
Department Toegepaste Informatica
Vrije Universiteit Brussel
België
Academiejaar 2004-2005

Abstract

Engineering Web sites is no longer possible in an ad-hoc manner. Instead, Dynamic content and integrated business logic require a careful website engineering process. There exist a wide range of methodologies that support the design of a website in such a systematic way. Unfortunately, most methods lack structured documentation and formal specifications. This results in a poor usability and ambiguous semantic specification of the models. Furthermore, the design methods all use different terms for concepts in their models, this makes it difficult to compare methodologies.

This thesis gives an overview and a formal specification of four recent model-driven website design methods : WebML, Hera, OOHDM/SHDM and WSDM. These formal specifications are taken as a starting point to identify common sub models and (design) concepts. Subsequently, a design method meta model is constructed. This meta model introduces common terminology and an abstraction layer on top of existing models.

Abstract

Web sites ontwikkelen is niet langer mogelijk op een snelle, eenvoudige manier. Dynamische informatie en geïntegreerde business logica vragen om een duidelijk gestructureerd ontwikkelings proces. Er bestaan reeds veel methoden om websites te ontwikkelen. Helaas ontbreekt er bij veel methoden goede documentatie en wordt er bijna nooit een formele beschrijving gegeven. Dit alles resulteerd in verwarrende en slecht bruikbare modellen. Bovendien gebruikt elke methode andere termen voor concepten met als gevolg dat het moeilijk is om methoden te vergelijken.

Deze thesis geeft een overzicht en een formele specificatie van vier recente website ontwerp methoden : WebML, Hera, OOHDM/SHDM en WSDM. Deze formele specificatie wordt gebruikt om overeenkomende concepten aan te duiden in de verschillende methoden. Vervolgens wordt er een ontwerp methode meta model geconstrueerd. Dit meta model introduceert een algemene terminologie en zorgt voor een abstractie laag op de bestaande modellen.

Acknowledgements

First of all, I would like to thank Prof. Dr. Olga De Troyer, for allowing me to do both my thesis and practical training at her lab.

Special thanks go to Sven Casteleyn and Peter Plessers of the WISE research group. They were always prepared to provide me with the necessary feedback and support. Their proofreading and suggestions certainly improved the quality of my thesis.

I would also like to thank my girlfriend Loes for the support she gave me during the writing of this thesis. Furthermore, I would like to thank my friends Bart and Robin and my co-students Dirk, Len and Lies. They gave me some nice and relaxing moments during this period of hard work.

At last but not least, I thank my parents for their moral and financial support during the year and for giving me the chance to study at the Vrije Universiteit Brussel.

Contents

List of Figures

List of Tables

Introduction

The World Wide Web(WWW) evolved from a network of linked pages into a data-intensive network with large Web applications. This evolution of the WWW has different causes [9, 14]:

- *More and more data comes available* : databases with large amounts of data are connected to websites and provide real-time information.
- *Business logic becomes integrated into Web applications*: online webshops and business to business websites use the WWW as a virtual marketplace. People use the Web to perform all sorts of tasks (e.g., hotel booking, flights reservation, etc.).
- *New software technologies*: programming and scripting languages specially designed for the WWW are gaining more and more expression power (e.g, Java[1]).
- *Increasing network capabilities*: faster network connections and better Internet servers are capable of transfering large amounts of data (e.g., on demand TV).
- *More and more dynamic content*: webpages no longer contain static information but perform queries on databases to provide dynamic information.
- *More and more people use the WWW*: more and more people and companies use the WWW as a medium to share and exchange data.

Because of this increased complexity the design of a website in an ad-hoc manner became obsolete and lead to a lot of problems [9, 14, 37]:

- *Usability problems:* the website has no clear mission statement. There is lots of redundant and inconsistent information. Furthermore, the provided information is not up-to-date.
- *Badly structured information:* information is not stored in databases or the database schema is not good designed. This results in maintainability problems.

[1] http://java.sun.com/

- *Bad navigation structure:* an unclear and confusing navigation structure which results in the *lost-in-hyperspace syndrome [25].*
- *Websites are difficult to maintain/extend:* there is no separation of concerns. Presentation, data and navigation are mixed together.
- *Inconsistent look & feel:* different styles and layout on a website.

To overcome these problems a number of website design methods have been proposed that support the systematic design of websites at a conceptual level. As representatives ones we mention: WSDM [8, 10, 26, 36], Hera [1, 17, 23, 38, 39], OOHDM [3, 30–32], WebML [6, 11, 12] and Ontoweaver [2]. Most of these methodologies are data-driven, which means they take the data as a starting point to construct a website. Other methods, like WDSM, are audience driven, which means they take the intended audience as a starting point to design a website. However, all these design methods are model based, meaning they use various models to specify a website at a conceptual level. The majority of these design methods describe their models in lots of articles and publications. However, there are so many articles on each design method that it is difficult for the reader to grasp or even use these methods. Furthermore, only informal descriptions of the methods are given which are lacking clear and explicit semantics.

This thesis will first give an informal overview of four design methods: WebML, Hera, OOHDM or SHDM and WSDM. This will give the reader a clear and structured overview of each methodology. Next, each methodology is analysed and formally specified using the ontology language OWL. This formal specification will be done in context of the Dexter hypertext reference model [20]. The Dexter reference model allows to capture the different abstractions used in a hypertext system. With these abstractions it should be possible to indicate where these various design methods have resemblances in their models. Having this done, this thesis proposes a *design method meta model* which attempts to cover the models from each discussed design method. The existence of such a meta model offer a wide range of benefits:

- **Interoperability:** the meta model acts as a mapping mechanism between models of different methodologies. This opens the path towards design method interoperability.
- **Standardization:** the meta model can be seen as a standard, models that are not able to map to the meta model may lack some modeling features.
- **Comparison of methodologies:** since each model can be mapped to a model in the meta model, it becomes easy to compare methods.
- **Standard terminology**: the design methods discussed in this thesis all use different terms for their concepts. The meta model introduces a standard terminology.

- **A foundation for future design methods**: the meta model can be used as a starting point for future design methods.

For the convenience of the reader, this thesis is structured in two parts : a background part and a research part.

Background Part

The background part will provide the reader with all information necessary to understand this thesis. Chapter 1 gives a discussion about ontologies. Chapter 2 will give an overview of a few website design methods. In Chapter 3 the Dexter hypertext reference model is discussed. This is the reference model which is used as a framework when constructing the design method ontologies.

Research part

The research part will report the research that has been done concerning website design methods. Chapter 4 discusses the construction of an ontology for each individual design method. Chapter 5 contains the most important part of this thesis namely the construction of a *design method meta model* based on the ontologies constructed in Chapter 4.

Part I

Background

Chapter 1

Ontologies

Introduction

Ontologies were developed in the domain of artificial intelligence to facilitate knowledge sharing and re-use. Nowadays, more and more research communities study the use of ontologies. The reason ontologies are becoming so popular is largely due to what they promise *: a shared and common understanding of a domain that can be communicated between people and application systems [14].*

This chapter will first explain *conceptualizations*, a term frequently used in ontology definitions. Thereafter, a definition for an ontology is given. Finally, a few recent ontology languages are discussed.

1.1 Conceptualization

Before we take a closer look at ontologies it is important to clarify the term "conceptualization". The definition from *Genesereth & Nilsson [18]* states the following:

> *Conceptualizations are the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them.*

This definition mentions two important aspects:

- *Objects:* an object in the real world. Objects can be concrete (e.g., this car, this paper) or abstract (e.g., the concept car, the concept paper). Object can be primitive (e.g, an atom) or composite (e.g., a car contains wheels).
- *Relations:* a semantic relation between objects (e.g., a car drives on a road).

According to Wordreference[1] a conceptualization is:

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

> *Inventing or contriving an idea or explanation and formulating it mentally.*

This definition relates a conceptualization to a *mental model.* Humans establish mental models of how things work, or how they would behave in a particular situation. A mental model contains a set of believes that hold for a particular situation. It is important to notice that a mental model is not an exact reflection of a real or imaginary situation. For example, someone could have a mental model of a "a bottle of water". The set of believes in this mental model *could* contain the following:

- It contains water.
- It is possible to open it.
- It is is made out of plastic.
- etc.

But another human could have a completely different mental model about a bottle of water (e.g., it is made out of glass). The same holds for a conceptualization. There are many conceptualizations of the same situation.

The following important issues were mentioned about ontologies :

- A conceptualization is not an exact reflection of some situation.
- There are many different conceptualization for the same situation.

1.2 Ontology Definition

With the notion of conceptualization explained we give a definition of an ontology. *Thomas Gruber* gives the following definition for an ontology:

> *An ontology is a specification of a conceptualization* [19].

This definition relates an ontology to a conceptualization. The defintion states that an ontology is a description of the concepts and relationships that can exist for an agent or a community of agents.

According to Wikipedia[2] an ontology is the following.

> *In computer science, an ontology is the attempt to formulate an exhaustive and rigorous conceptual schema within a given domain, typically a hierarchical data structure containing all the relevant entities and their relationships and rules within that domain.*

[2]http://en.wikipedia.org/

So an ontology expresses a particular domain of knowledge by means of a conceptual schema (a conceptualization). This conceptual schema is described by making use of the following kinds of concepts:

- Classes (general things) in the many domains of interest (e.g., the class of cars).
- The relationships that can exist among things (e.g., a car has four tires).
- The properties (or attributes) those things may have (e.g., a tire has a color).

The two previous definitions clearly make the relation between an ontology and a conceptualization. Figure 1.1 shows how ontologies and conceptualizations are related. An ontology describes, with a formal specification, a conceptualization. A conceptualization describes a particular domain of knowledge. This domain of knowledge is a representation of (a part of) a world.

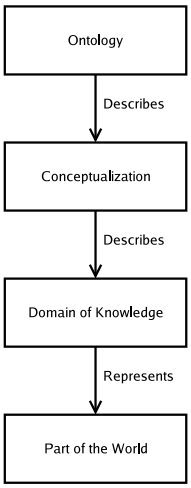

Figure 1.1: Ontologies and conceptualization.

To formaly represent ontologies an ontology language is used. The next section will describe a few recent ontology languages for the WWW.

1.3 Web Ontology Languages

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties and relations [4].

This section will discuss two ontology languages : the Resource Description Framework Schema (RDFS)[3] and the OWL Web ontology language (OWL)[4].

1.3.1 RDF Schema

The Resource Description Framework (RDF)[5] provides a way to express simple statements about resources, using named properties and values. However, with RDF it is not possible to define the vocabularies (terms) they intend to use in those statements, specifically, to indicate that they are describing specific kinds of classes or resources and will use specific properties in describing those resources. RDF(S) provides the facilities needed to describe such classes and properties and to indicate which classes and properties are expected to be used together [40].

Classes

A class in RDF(S) corresponds to the generic concept of a type or category. A class can be used to represent almost any kind of thing, such as trees, vehicles, computers etc. Classes can be organized hierarchical by the subclass property. Classes are described using the RDF(S) resources 'rdfs:Class', 'rdfs:Resource' and the properties 'rdf:type' and 'rdfs:subClassOf'. The following example[6] shows an RDF(S) for a Vehicle Class Hierarchy:

```
<rdfs:Class rdf:ID="MotorVehicle"/>
<rdfs:Class rdf:ID="PassengerVehicle">
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
...
<rdfs:Class rdf:ID="Truck">
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
</rdf:RDF>
```

The example describes a class "MotorVehicle" and two subclasses "PassengerVehicle" and "Truck".

Properties

In addition to describing classes it is also possible to describe properties that characterize classes. For example, a motor vehicle may have a weight. In RDF(S), properties are described using the RDF class "rdf:Property" and the RDF(S) properties

[3] http://www.w3.org/TR/rdf-schema
[4] http://www.w3.org/TR/owl-features/
[5] http://www.w3.org/RDF/
[6] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#schemaclasses

"rdfs:domain", "rdfs:range" and "rdfs:subPropertyOf". The next example[7] describes a property "hasWeight" for the class "MotorVehicle":

```
<rdf:Property rdf:ID="hasWeight">
 <rdfs:domain rdf:resource="#MotorVehicle"/>
 <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>
<rdfs:Datatype rdf:about="&xsd;integer"/>
```

The domain of a property, in this case "MotorVehicle", describes the object of the property. The range, in the example "integer", describes the subject of a property. A property can be specialized with the subproperty relation:

```
<rdf:Property rdf:ID="driver">
 <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>
<rdf:Property rdf:ID="primaryDriver">
 <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

The previous example[8] says that if an instance of a person is a "PrimaryDriver" this person is also a "Driver".

Containers and Collections

There is often a need to describe groups of things: for example, to say that a book was created by several authors, or to list the students in a course, or the software modules in a package. RDF provides several predefined (built-in) types and properties that can be used to describe such groups. Three different kinds of containers exist:

- *"rdf:Bag":* represents a bag of unordered things.
- *"rdf:Seq":* represents an ordered sequence of things.
- *"rdf:Alt":* represent an alternative container.

Besides containers, RDF also provides collections, which is a more restrictive form of a container. A collection can be closed so it is not possible to add anymore members.

[7] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#schemaclasses
[8] http://www.w3.org/TR/2004/REC-rdf-primer-20040210

Limitations

RDF(S) is a language to specify simple ontologies. However, many types of knowledge cannot be expressed in this language. RDF(S) has the following limitations[9]:

- cardinality constraints on properties, e.g., that a Person has exactly one biological father.
- specifying that a given property (such as ex:hasAncestor) is transitive, e.g., that if A ex:hasAncestor B, and B ex:hasAncestor C, then A ex:hasAncestor C.
- specifying that a given property is a unique identifier (or key) for instances of a particular class.
- specifying that two different classes (having different URIrefs) actually represent the same class.
- specifying that two different instances (having different URIrefs) actually represent the same individual.
- specifying constraints on the range or cardinality of a property that depend on the class of resource to which a property is applied, e.g., being able to say that for a soccer team the ex:hasPlayers property has 11 values, while for a basketball team the same property should have only 5 values.
- the ability to describe new classes in terms of combinations (e.g., unions and intersections) of other classes, or to say that two classes are disjoint (i.e., that no resource is an instance of both classes).

OWL, the ontology language described in the next section, overcomes these problems by providing a framework built on top of RDF(S).

1.3.2 OWL

The OWL Web Ontology Language is a language for defining and instantiating Web ontologies. OWL is an extension of RDF(S), it overcomes the limitations of RDF(S) discussed in the previous chapter. OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users [10].

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.

[9] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#richerschemas
[10] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3

- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).

- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary.

The most important constructs of OWL are discussed below.

Classes and Individuals

Just as in RDFS, a class groupes concepts of the same type. Individuals are instantiations of a class. The next example shows a very simple wine ontology in OWL.

```
<owl:Class rdf:ID="Wine">
 <rdfs:subClassOf rdf:resource="PotableLiquid" />
</owl:Class>
...
<Wine rdf:ID="PinotNoir" />
```

This ontology defines a class “Wine” which is a subclass of the class “PotableLiquid”. There is one individual defined named "PinotNoir".

Properties

Like in RDFS, properties describe characteristics of classes. There are two types of properties:

- *Datatype properties:* relations between instances of classes and RDF literals and XML schema datatypes

- *Object properties:* relations between instances of two classes.

We extend our example with an object property:

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
 <rdfs:domain rdf:resource="#Wine"/>
 <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

```
...
<owl:Class rdf:ID="Wine">
 <rdfs:subClassOf rdf:resource="PotableLiquid"/>
</owl:Class>
<owl:Class rdf:ID="WineGrape">
 <rdfs:subClassOf rdf:resource="Grape"/>
</rdfs:Class>
<owl:Restriction>
 <owl:onProperty rdf:resource="#madeFromGrape"/>
 <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
 </owl:minCardinality>
</owl:Restriction>
...
```

Just as in RDF(S) a property has a domain and a range. The above example places a minimum cardinality constraint on the relation between the class "Wine" and "WineGrape" : a wine is made from at least one grape. In addition to RDF(S), properties can be characterized (e.g., inverseOf, Cardinality, FunctionalProperty, value constraints etc.). A full definition of the OWL syntax is given here [33].

Chapter 2

Website Design Methods

Introduction

The WWW evolved from a network of linked pages to an information system with large Web applications. A website is not longer constructed in an ad-hoc manner by simply writing some HTML pages. Instead dynamic content and integrated business logic require a careful website engineering process. Such a website engineering process is described by a website design method. There are too many website design methods to describe them all here. Instead, this chapter gives a detailed overview of four representative website design methods.

2.1 Web Modeling Language (WebML)

The **Web Modeling Language (WebML)**, is a model-driven method to design websites. The WebML group was created by researchers in the data base group of the Dipartimento di Elettronica e Informazione at Politecnico di Milano. The WebML modeling language is a visual notation for specifying the composition and navigation features of hypertext applications [6]. The method has a clear separation between the *conceptual model* and the *hypertext model*. The conceptual model or data model is modeled with the *Entity-Relationship(ER) [13]* notation. The hypertext model or navigational model is modeled in WebML notation. As shown in figure 2.1 the design method contains seven steps:

1. *Requirements Specification:* a formalization of the essential information about the application domain (e.g., business rules).

2. *The Data Design:* this design phase produces the data model by making use of business rules and the requirements specifications.

3. *The Hypertext Design:* this design phase produces the hypertext model which serves as a navigation model on top of the data model.

4. *Architecture Design:* specification of hardware, network and software components.

5. *Implementation:* the actual implementation of all models.

6. *Testing and evaluation:* test, evaluate and repeat each step if necessary.

7. *Maintenance and Evolution.*

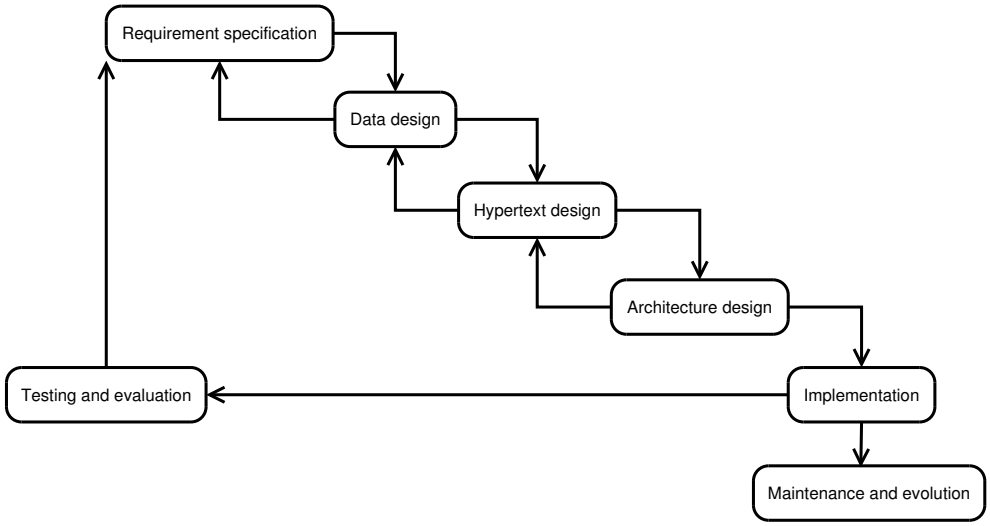

Figure 2.1: The WebML design process.

The next sections gives an overview of the most important modeling phases. A full description of the WebML methodology can be found here [6, 11].

2.1.1 Data Design

There exist many formal languages to describe conceptual data models. WebML does not propose yet another data modeling language, but instead builds on the notation of the ER modeling language. The result of this modeling phase is the conceptual data schema or the data model. This thesis foresees no detailed explanation of the ER model. Instead, the reader is referred to the ER model introduced by Peter Chen [13].

2.1.2 Hypertext Design

The hypertext design phase uses the data model to construct a navigational model for a website. To express this model they proposed the WebML syntax. The WebML syntax has both a graphical and a textual notation. For a full syntax description we refer to [11]. The first aim of the hypertext model is the modeling of content

publishing: how is the information shown to the user (2.1.2.1). The second aim is to model content management: how are changes made to the available information (2.1.2.2).

2.1.2.1 Content Publishing

The key ingredients of WebML are *pages, units* and *links* organized into modularization constructs called *areas* and *site views.* We first take a look at the definition of a unit:

> *Units are atomic elements for specifying the content of a webpage.*

A unit extracts information from entities in the data schema and is able to publish this information dynamically. Units are the building blocks of pages. There are six types of units :

- *Data Units:* Show information about a single object (e.g., information about a car).
- *Multidata Units:* present information about a set of objects (e.g., a list of movies).
- *Index Units:* show a list of descriptive properties of some objects, without showing details about those objects (e.g., a list of movie titles).
- *Scroller Units:* enables the browsing of an ordered set of objects with next and previous operations (e.g., scrolling through a photo collection).
- *Entry Units:* model entry forms that gather input (e.g., a registration form).
- *Global Units:* used to store information that needs to be available to all units (e.g., a country of origin).

The first four units enable the publishing of information, they call this *content units.* An entry unit expresses the acquisition of information from users (e.g., information entered by a user into a form). Global units are used to store global variables. Content units are characterized by the following properties:

- *Name:* a user-defined name.
- *Source:* the entity from the conceptual model that provides the content for this unit.
- *Selector(Optional):* a selection predicate determing which information is shown in the unit.
- *Order Clause(Optional):* a clause that determines the order in which the information is shown.

Some additional properties exist like *included attributes* or *block factors* (see [11] for more information). Entry units, which support form-based data entry are characterized by the following properties:

- *Name:* a user-defined name.
- *Fields:* a number of input fields.

The next building block of WebML is a page. A page is the actual element that is shown to the user who browses the hypertext by accessing webpages. The definition of a page is as follows:

> *Pages are the actual interface elements delivered to the user, who browses the hypertext by accessing pages in the desired sequence.*

Pages typically consists of several units grouped together to form a block of information.

The next concept in WebML is a link. Links are the navigational constructs that connect pages and units. A definition of a links is as follows:

> *Links allow the modeling of navigation paths between pages and units.*

The part of modeling that specifies the links between pages is often called *navigation modeling.* A link generalizes a fundamental notion of hypertext : the concept of an anchor. According to the WebML terminology a link can have the following properties:

- *inter-page link:* links that cross the boundaries of pages.
- *intra-page link:* a link inside the same page.
- *contextual link:* the link transports information.
- *non-contextual:* there is no information transport.

So a link allows navigation and information transport between pages.

The last concepts in the WebML terminology are site views and areas.

> *Site view and areas are modularization constructs that allow the grouping of pages and links.*

A set of pages can be grouped into a site view. Such a site view can satisfy the needs of a particular group of users. Large site views can be decomposed in areas, which are clusters of pages with homogeneous information.

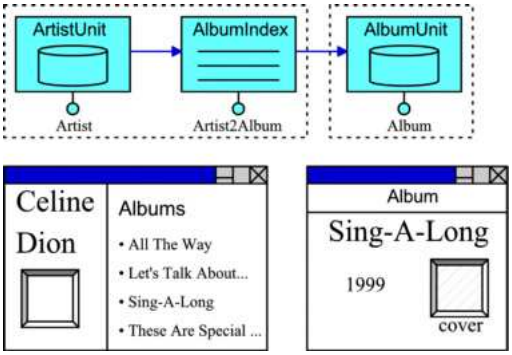

Figure 2.2: An example of a WebML hypertext model.

Figure 2.2 (taken from [11]) shows an example of a hypertext model that contains two pages. The first page contains one data unit and one index unit. The second page contains one data unit. The figure also shows the corresponding webpages. There is a navigational link from "album title" to a new webpage "album" that contains detailed information about the followed link.

2.1.2.2 Content Management

Often Web applications perform operations on data. For example, a user can add a comment to a website that reviews movies. The modeling of such operations is done by the *content management model*. This model is an extension of the hypertext model of section 2.1.2. The content management model introduces a few new concepts to model operations:

- *Operation Units:* they express the processing executed as the result of navigating a link. An operation unit receives input parameters from other units and performs an operation on data according to this parameters (e.g., an operation unit receives a name and creates a new user with this name).
- *KO-links and OK-links:* these links are attached to an operation unit and express the failure and success of an operation respectively.

WebML provides a number of built-in operations, a short description of each predefined operation is given:

- *Object Creation:* create a new entity instance in the conceptual model.
- *Object Deletion:* delete an instance in the conceptual model.

- *Object Modification:* edit an object instance in the conceptual model.
- *Relationship creation:* create a new relation between instances of the conceptual model.
- *Relationship deletion:* delete a relationship between two instances of the conceptual model.

The content management model allows the formal description of data management in website applications.

2.1.3 Presentational Design

The presentational design addresses how the information is shown to the user (look & feel) and where this information is placed on the screen(layout).

The layout of a page is organized with a *page template skeleton* which only contains the minimal HTML mark-up needed to define the layout grid of the page and the position of the various units in such a grid. The template skeleton is transformed with XSLT[1] presentation rules. There are two kinds of such rules:

- *Page Rules:* they match the outermost part of the skeleton"s layout (e.g., a table tag).
- *Unit Rules:* they match the class of units (for instance, index units) and produce the markup for their presentation.

Note the HTML produced by the XSLT rules exploits the CSS[2] styles associated to the corresponding units, so that the XSLT rules are only concerned with layout and not with graphic properties. WebML comes with a tool called WebRatio Site Development Studio (WebRatio) [7]. This tool provides a GUI for the WebML design process and is able to organize the layout and presentation of a page.

2.2 Hera

Hera is a model-based methodology for designing Web Information Systems (WIS). Hera is a program in the Information Systems group at the department of Mathematics and Computer Science of the Technische Universiteit Eindhoven [1]. Based on the principle of separation of concerns it distinguishes three design steps: conceptual design, navigational design and presentation design [39].

[1] http://www.w3.org/TR/xslt
[2] http://www.w3.org/Style/CSS/

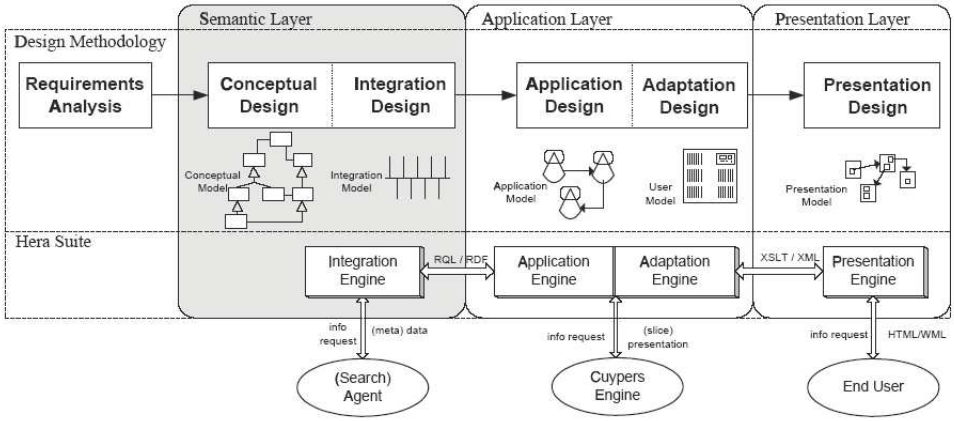

Figure 2.3: The Hera methodology.

The Hera methodology defines three layers in which the modeling occurs:

- *The Semantic Layer:* This layer defines the content that is available in terms of a conceptual model. This content can originate from a database or some external source like a search agent. All this information is brought together in the integration design.
- *The Application Layer:* This layer addresses the navigational design in terms of a navigational model. The navigational model is a view on the conceptual model that satisfies the needs of a particular user class.
- *The Presentational Layer:* This layer defines how content is presented to the user in terms of a presentation model.

Figure 2.3 (taken from [1]) shows the design methodology and the mapping between the different models. Hera makes use of RDFS [34] to represent the conceptual and the application model. The mapping between the conceptual model and the application model is done by the RDF Query Language (RQL). The presentation model is in fact an XSLT transformation on entities from the conceptual model. Note the integration design and adaptation design is beyond the scope of this thesis. For more extensive explanation of the Hera methodology we refer to [39].

2.2.1 Conceptual Design

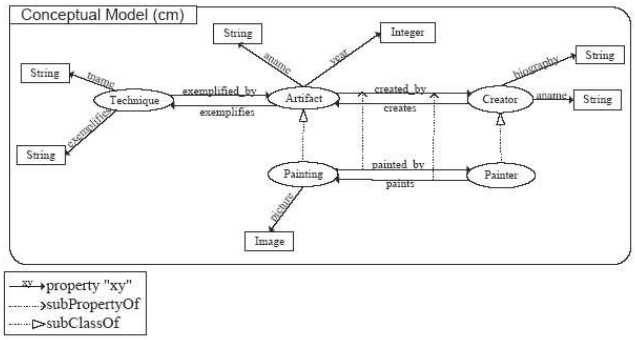

Figure 2.4: Hera conceptual model example

During conceptual design the data schema is defined. Figure 2.4 (taken from [39]) presents an example of a Conceptual Model (CM) in Hera. The CM is composed of *concepts* and *concept properties* that together define the domain ontology. There are two types of concept associations : *concept attributes* and *concept relationships.* Concept attributes associate concepts with media items (e.g., Image, String). Concept relationships define relations between concepts.

The CM is expressed in RDF(S) using two additional RDF(S) descriptions : *CM properties* and system *media types.* A CM property describes the inverse and cardinality of concept relations. The system media types describe the possible types of a concept (e.g., video, audio, string). An instance of a CM is expressed in RDF.

In Hera its possible to attach a *conditional inclusion* to an element. A conditional inclusion is specified in XSLT[3], it allows adaptation at a conceptual level. The reader is referred to [41] for a more detailed explanation, since adaptation is not a subject of this thesis.

Hera provides an extention of the CM with a *Navigational Data Model(NDM).* The purpose of the NDM is to complement the CM with a number of auxiliary concepts that do not necessarily exist in the CM. Figure 2.5 (taken from [38]) show an example of a NDM with the following concepts:

- *"SelectedPainting":* a subclass of the "Painting" concept in the CM. It represents the paintings the user can select in a multi-selection form.
- *"Order":* an order represent a selection of one painting and a quantity.

[3]http://www.w3.org/TR/xslt

- *"Trolley":* a troley represents a shopping basket containing a set of orders.

From the example we can remark a NDM contains two kinds of concepts:

- *Views over concepts from the CM* (e.g., "SelectedPainting" is a subset of paintings the user selected).
- *Concepts based on user interaction* (e.g., Order, Trolley).

The instantiation of both concepts occurs by triggering a certain action (e.g., pushing a submit button) in the application model, which is defined in the next section.

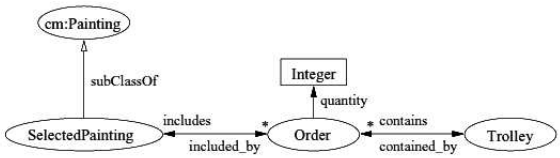

Figure 2.5: Hera navigational data model example.

2.2.2 Application Design

2.2.2.1 Information Publishing

The application design results in an *application model* (AM) which specifies how the user can navigate through the data. The AM is composed of *slices* and *slice relationships*. There exist two sorts of slice relationships: *navigational relationships* and *compositional relationships*. The navigational relation is used to express navigation between slices. Compositional relations are used to aggregate slices. A compositional relation between slices can be one-to-many. This is specified by a "Set" on the relation. A slice is always owned by a concept from the conceptual model and each slice has attributes that correspond to attributes in the conceptual model. A slice can contain nested slices. Top-level slices correspond to pages that are presented to the user. Note slices can be associated to an *appearance condition,* this condition specifies adaptable behaviour. Since Adaptation is beyond the scope of this thesis, the reader is referred to [41] for more information.

Figure 2.6 (taken from [39]) shows an AM that contains two top-level slices : technique and painting. The technique slice contains two attributes and a compositional relation to painting. Note the Set notation is used in this relation because technique has a one-to-many relation with painting. The top-level painting slice contains two compositional relations to technique and painter.

The AM in Figure 2.6 corresponds to two webpages. One webpage containing a

name, a description of a technique and a list of painting pictures. Each of these pictures is linked to a second webpage that contains more detailed information about the painting. The second page is linked to the first page by the name of the technique.

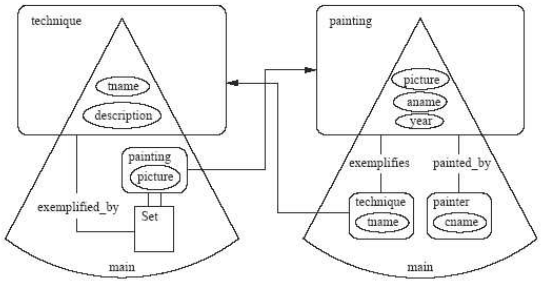

Figure 2.6: Hera application model example.

2.2.2.2 User Input and Data Manipulation

The AM presented in section 2.2.2.1 only considers simple user interaction : navigating through the hypertext by clicking on anchors and following links. Hera provides an extended AM [38] that allows:

- *Dynamic information* (e.g., a user sends a query and receives dynamic information).
- *User input* control with automated processing of navigation information (e.g., selecting items for a shopping basket).

The Hera AM forsees two concepts that allow the modeling of user interaction: forms and queries. Figure 2.7 (taken from [38]) show where these concepts are situated in the AM. A form is part of a slice and contains the following properties:

- *Owner Concept:* the concept related to this form (e.g., a form to select paintings is related to the "Painting" concept).
- *Input fields:* the fields the user has to fill in (e.g., input of a quantity, selection of a painting). A field can be a selection field, multi-selection field, text input etc.
- *Output fields:* the values that are sent to the next navigation object.

Forms are connected with navigational relationships that are in fact queries (Q1 & Q2). A query uses the form input values to select information from the CM. Note

Hera implements forms using the XFORMS [15] standard, queries are expressed in SeRQL (Sesame RDF Query Language) [5].

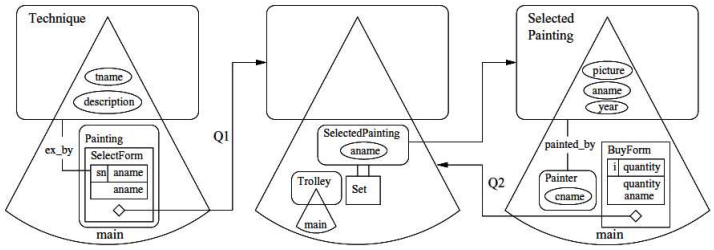

Figure 2.7: Hera application model with forms and queries.

2.2.3 Presentational Design

The result of the presentational design is the *presentation model (PM)* which describes how data is rendered. The presentation model of Hera makes use of AMACONT[4]. The AMACONT project uses a *component-based XML document* format that enables to compose Web presentations from reusable document components encapsulating adaptive content, behaviour and layout [41]. There are three kinds of components to describe a document, each component is defined at a different layer in AMACONT:

- *Media Components:* encapsulate concrete media assets by describing them with technical meta-data (e.g., image resolution, size).
- *Content Units:* group media components by declaring their layout in a device-independent way.
- *Document Components:* define a hierarchy out of content units to fulfill a specific semantic role.

Figure 2.8 (taken from [16]) shows the situation of components in the different layers.

In order to describe the presentation of a component-based Web document, AMACONT allows to associate XML-based layout descriptions to components. Such a description is called a *layout manager*. Currently there are four such managers:

- *"BoxLayout":* layout of multiple components either vertically or horizontally.
- *"BorderLayout":* arranges components to fit in five regions : north, south, east, west and center.

[4]http://www-mmt.inf.tu-dresden.de/english/projekte/AMACONT

- *"OverlayLayout":* allows to arrange components on top of each other.
- *"GridLayout":* allows the layout of components in a configurable grid.

Similar, the PM of Hera allows the definition of layout managers and the assignment of layout managers to slices from the AM. The layout of slice attributes and subslices can be defined by assigning them to "subcomponent" specifications. For more information on the presentation design of Hera and AMACONT the reader is referred to [41].

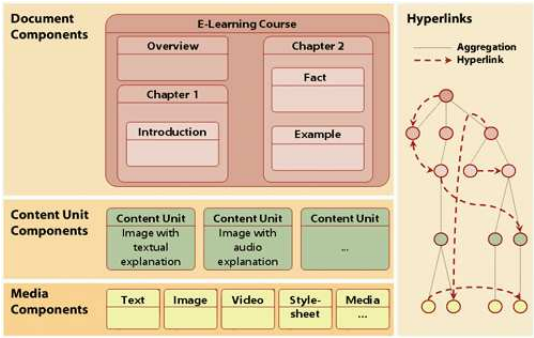

Figure 2.8: The AMACONT document model.

2.3 OOHDM/SHDM

The **Object Oriented HyperMedia Design Method (OOHDM)** is developed by Daniel Schwabe at the Departement of Informatics, Universidade Católica de Brasília. The last version of OOHDM is called the **Semantic Hypermedia Design Method (SHDM)**. SHDM differs from OOHDM since it uses RDF(S) and OWL to represent its various models. The SHDM method is a model-driven approach to design Web applications in five steps : Requirements Gathering, Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. Classification, aggregation and generalization/specialization are used throughout the process to enhance abstraction power and reuse opportunities. More information on OOHDM and SHDM can be found here [28,30,31]. The following sections give a brief overview of the most important design steps of SHDM.

2.3.1 Conceptual Design

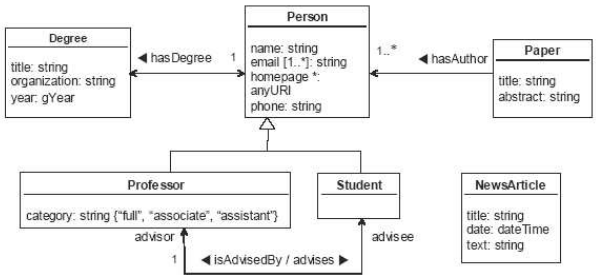

Figure 2.9: SHDMf conceptual model example.

The first design phase is the conceptual modeling which results in a conceptual model. This model is a conceptual model for an application domain described as some ontology using either OWL or RDF(S).

An example of a conceptual model is given in Figure 2.9 (taken from [32]). This is a graphical representation for an ontology of an academic department.

2.3.2 Navigational Design

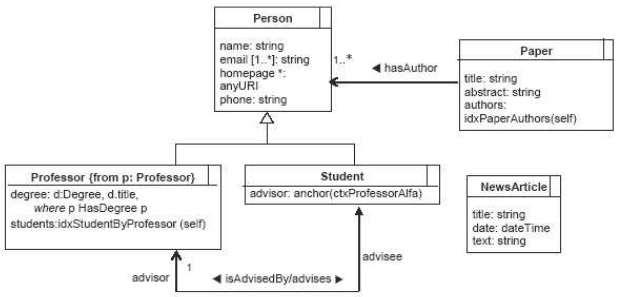

Figure 2.10: SHDM navigational class example.

This step produces two models : the *navigational class model* and the *navigational context model.* The navigational class model defines *what* information can be reached, the navigational context schema defines *how* this information can be accessed.

The navigational class model is a view over the conceptual model. It shows which information in the conceptual model can be reached by the user. Navigational

classes represent views of conceptual classes. Attributes from conceptual classes are mapped to attributes in navigational classes. There are three mappings defined:

- *Directly mapped conceptual attributes :* attributes from the conceptual schema (e.g., "name" in the navigational class "Person").
- *Derived attributes:* attributes not directly visible in the conceptual schema (e.g., calculating the age from the birthday).
- *Attributes from other conceptual classes* (e.g., "students" in the navigational class "Professor" do not belong to the conceptual class "Professor").

The mapping from conceptual model to navigational model is done with RQL [27], a query language for RDF. These RQL statements are described in the attributes of the navigational classes, they perform a query on data in the conceptual model. Each class in the navigational model corresponds to a node. Navigational classes are connected with each other by navigational links. An example is given in Figure 2.10 (taken from [32]).

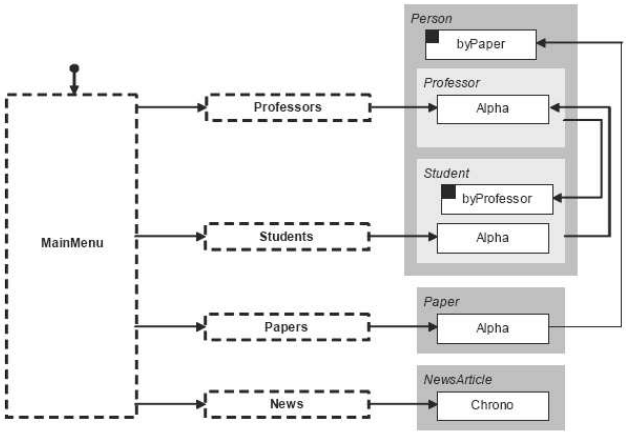

Figure 2.11: A navigational context model in SHDM.

The navigational context model describes how navigation is organised in different contexts. A navigational context schema is built out of *navigational context objects* and *access structures.* Access structures are indexes (a collections of links) that allow the user to reach navigation objects within some context. Context objects refer to navigational classes and define how the user can browse the information in the current context (in the example each navigational class is associated with a navigational context).

Figure 2.11 (taken from [32]) shows a navigational context model defined on the

navigational class model of Figure 2.10. The navigational context schema contains five access structures : “MainMenu”, “Professors”, “Students” , “Papers”, and “News”. It also contains four navigational context objects : “Professor”, “Student”, “Paper” and “NewsArticle”. The access structures define how the user can reach the navigational context objects. For example, the navigational context object "Paper" can be reached by following "MainMenu" and "Paper". The navigational context objects define how the user can navigate in a particular context. For example, when a user is at the navigational node “Student” he has the possibility to browse the students either alphabetically or by professor. We say the user is in the context of student with two navigational access patterns.

Context: Professor Alpha
Parameters:
Elements: prof Professor
Context class:
Order: prof.name ASC
Navigation: index (idxProfessorsAlfa), sequential
Operations:
Restrictions:
Comments: All professors in alphabetical order of name.

Figure 2.12: A Context Definition Card.

In addition, the full definition of a context is given by a *Context Definition Card (CDC)*. The CDC in Figure 2.12 (taken from [32]) shows the definition of the context “Professor”. The professors are shown in ascending alphabetical order. The navigation in this context is sequential on alphabetical index.

2.3.3 Interface Design

Abstract interface design [29] focuses on making navigation objects *functional* perceptible to the user. The abstract interface design only provides the support to model the information exchange between the user and the website. Since this information exchange is driven by the tasks being supported, it is less sensitive to hardware and software specific aspects. In SHDM, the abstract interface design is specified using the *Abstract Widget Ontology* which will be discussed in section 4.3.3.1 on page 56.

At a more concrete level there is the *concrete interface design* which defines the actual look & feel and layout of the website. For this purpose, SHDM provides the *Concrete Widget Ontology* which provides basic elements to design an interface. The rest of the interface design is left to the graphic designer since it is almost totally dependent on the particular hardware and software runtime environment.

2.4 WSDM

The **Website Design method (WSDM)** project was initiated (1998) and is led by Prof. Dr. Olga De Troyer of the Vrije Universiteit Brussel, department of Computer Science and head of the research group WISE [8]. This method takes a somewhat different approach than the other methods. Most design methods take the data or database as starting point to construct a website, those are the so-called *data-driven* methods. WSDM takes as starting point the needs and requirements of the intended audience(s) of the website. They call this an *audience-driven* method. The method contains five phases : mission statement specification, audience modeling, conceptual design, implementation design and implementation. WSDM makes a clear distinction between the conceptual design and the presentational design to avoid implementation details at an early design stage. Figure 2.13 (taken from [9]) shows an overview of the different design phases. The following sections describe each of the modeling phases.

2.4.1 Mission Statement Specification

The mission statement specifies the purpose, subject and intended users of the website. The mission statement is formulated in natural language.

2.4.2 Audience Modeling

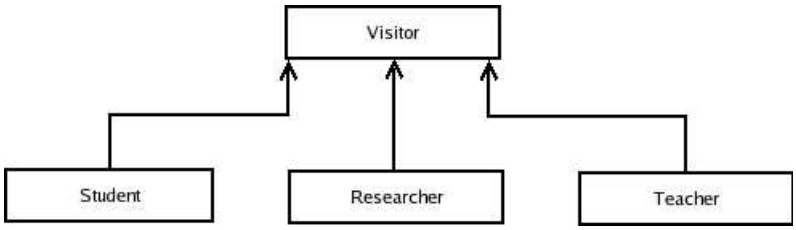

Figure 2.14: An audience model example.

During *audience modeling* the mission statement is taken as a starting point to group the intended visitors of the website into *audience classes.* Users with the same requirements and characteristics are put in the same class. This phase is called *audience classification.* The audience classes are organised in a hierarchical tree to form the *audience model.* Figure 2.14 shows an example of such an audience model.

The second step in the audience modeling phase is the *audience characterization.* During this step the following statements are summed-up for each audience class:

- Characteristics (e.g., colour-blind).

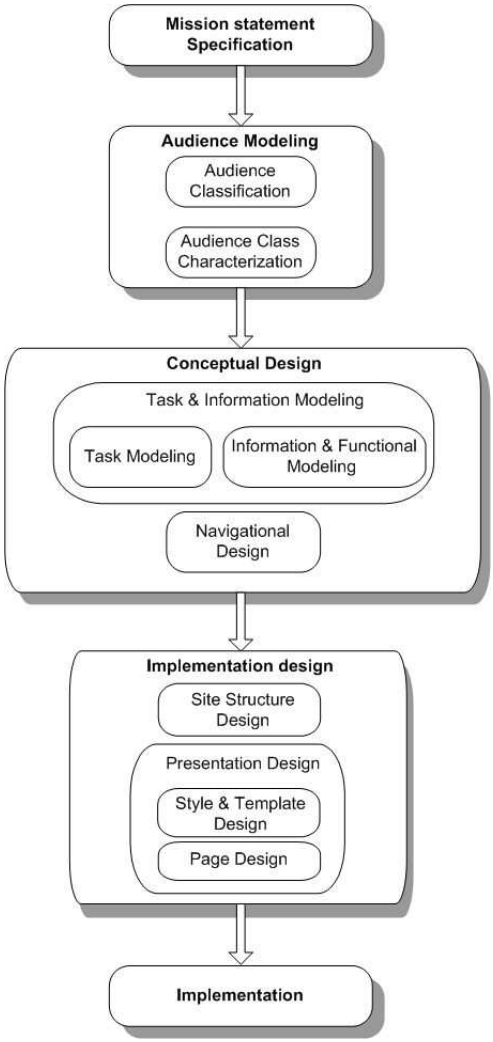

Figure 2.13: WSDM overview.

- Navigational requirements (e.g., easy navigation for children).
- Usability requirements (e.g., use of different languages).
- Information - and functional requirements.

An example off functional - and information requirements for a reviewer visiting a scientific conference website could be:

```
Functional Requirements:
 - Submit reviews on papers
 - Upload new papers
Information Requirements:
 - View all papers
 - View all authors
```

The result is an audience model with for each audience class a summation of characteristics and requirements.

2.4.3 Conceptual Design

The conceptual design phase describes the content, functionality and structure of the website at a conceptual level. The conceptual design does not specify implementation details like used technologies, used software etc. The conceptual design can be divided in three parts : the *task & data modeling* and the *navigational design.*

2.4.3.1 Task Modeling

The goal of *task modeling* is to model the different information and functional requirements of the different audience classes and the processes that are necessary to support these requirements [35]. For each information and functional requirement a task analysis is done (specify which tasks are needed to perform this requirement). Tasks are further decomposed into elementary tasks.

2.4.3.2 Data Modeling

During data modeling we see which data is needed to perform the elementary tasks specified in the task modeling phase. For each elementary task a small conceptual schema is constructed. Such a schema is called an *object chunk.* The result of this phase is a collection of object chunks. WSDM uses OWL to model object chunks.

Figure 2.15 shows an example of an object chunk vizualized in ORM [21]. This object chunk describes the concept "Paper". According to this schema a paper has one Author and many co-authors, a paper also has a unique PaperId and one Paper Title.

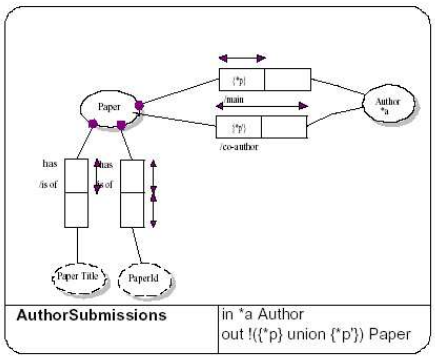

Figure 2.15: An object chunk in WSDM.

2.4.3.3 Navigational Design

The objective of navigational design is to construct the *navigational model* for each audience class. The navigational model imposes how the user navigates through the information that is available on the website. The navigational model is built out of nodes grouping one or more object chunks. Nodes are connected by navigational links. Figure 2.16 shows a navigational model for an audience class "Author".

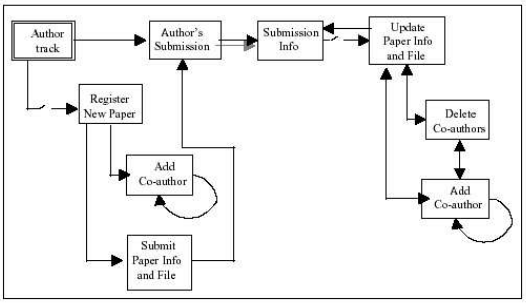

Figure 2.16: A navigational model in WSDM.

2.4.4 Implementation Design

The *implementation design* further extends the conceptual model with implementation details. The implementation design consists of two sub phases : *Site Structure Design* and *Presentation Design.* During site structuring design, it is decided which nodes from the navigational model are grouped together to form pages. The result is a *site structure model.*

The presentation design is divided in two sub-phases. The *Style & Template Design* phase defines page templates for the website (e.g., homepage template). The result of this phase is the *style & template model.* These templates are used by the next sub-phase called the *Page Design.* This phase describes how information is presented on the pages. The result of this modeling phase is the *page model.*

Chapter 3

The Dexter Hypertext Reference Model

Introduction

The Dexter hypertext model is a reference model for hypertext systems. The model captures, both formally and informally, the important abstractions found in a wide range of existing hypertext systems [20].

In this thesis the model is used as a principle basis to construct a *meta-model* for the website design methods discussed in the next chapter. As illustrated in Figure 3.1 the Dexter reference model consists of three layers, the *runtime layer*, the *storage layer* and the *within-component layer*. There are also two interfaces that connect these layers, the *presentation specifications* - and *anchoring* interfaces.The emphasis of the Dexter reference model for hypertext systems is on the storage layer, the anchoring and the presentation specification.This chapter first gives a few definitions for the terminology used in the description of the layers and interfaces. Next, a description of each layer and interface is given.

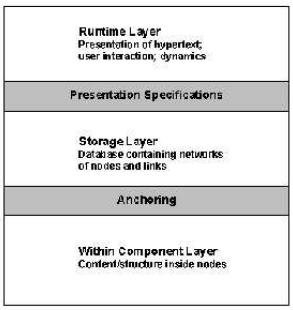

Figure 3.1: The Dexter Reference model for hypermedia systems.

3.1 Definitions

Before we continue with the discussion of the Dexter reference model a few definitions [22] are given to provide the reader with enough background knowledge to understand the following sections.

3.1.1 Components

A component is a pair < uid,cinfo >, where uid is a global unique identifier for component, cinfo represents "component info" and consists of:

- *a set of attribute-value pairs.*
- *a sequence of anchors.*
- *a presentation specification.*

Attributes can be of any type and used for any purpose. They are not elaborated further. We also do not specify the detail of the presentation specification. We distinguish two types of components : atomic components and composite components. An ***atomic component*** *is a component for which the anchor values and possibly some attribute values belong to the within-component layer (This means that we cannot describe their internal structure in the model). A* ***composite component*** *is a component where the anchor values are the unique identifiers of other components (that we refer to as subcomponents).*

3.1.2 Anchors

An anchor is a pair < aid,avalue >, where aid is a unique identifier for the anchor within the scope of its component and avalue is an arbitrary value that specifies some location, region, item or substructure of a component. Anchor values of atomic components belong to the within-component layer and are not elaborated in Dexter. Anchor values of composite components are uid-aid pairs, where the uid identifies a subcomponent of the composite and the aid identifies an anchor within that subcomponent.

3.1.3 Specifiers

A specifier is a 4-tuple < uid,aid,dir,pre > where uid is the unique identifier of a component, aid is the identifier of an anchor, dir is a direction (which is FROM, TO, BIDIRECT or NONE), and pre is a presentation specification.

3.1.4 Links

A link is a component < uid,ss,cinfo > where uid is a unique identifier, ss is a sequence of specifiers, and cinfo is the component information. A component"s information consists of:

- *a set of attribute-value pairs*
- *a sequence of anchors*
- *a presentation specification*

3.2 Within-component Layer

> *The within-component layer is specifically concerned with the contents and structure within the components of the hypertext network.*

This layer hides all data and media-specific details, who are encapsulated in the components, from the other layers.The elaboration of this layer is beyond the scope of the model. Instead, other reference models have to specify this layer to define, in conjunction with the dexter model, a full hypertext model.

3.3 Storage Layer

> *The storage layer describes how the components and links form a hypertext network.*

The storage layer models a hypertext network as a set of:

- atomic components
- composite components
- link components

The storage layer contains two functions : *a resolver function* and *an accessor function.* Both functions provide addressing of components by a unique identifier(UID). The accessor function provides a mapping between the UIDs and components. Because this way of addressing components is too restrictive there also exists a resolver function. A resolver function can take a component specification as input and "resolves" this into a UID. Figure 3.2 shows the organization of the storage layer. There is one atom, one composite and one link, each with their unique identifier.

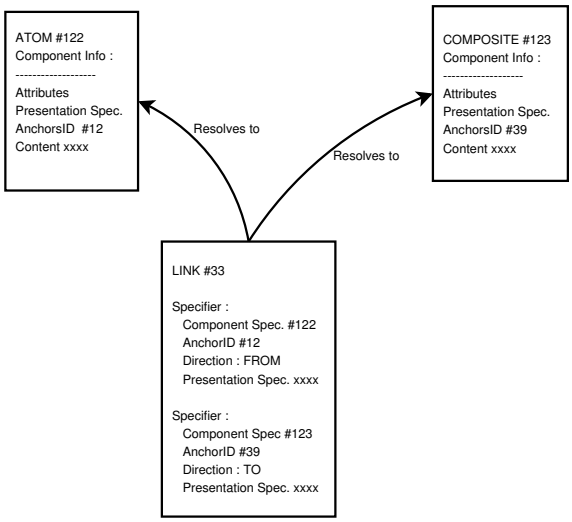

Figure 3.2: The organization of components in the storage layer.

3.4 Runtime Layer

> *The runtime layer describes how instantiated components are presented to the user.*

This layer instantiates components and uses the presentation specifications to present the instance to the user. There can be many simultaneous instantiations of a component. Each instantiation has a unique identifier (IID). When a component is instantiated its anchors are also instantiated. An instantiated anchor is known as a *link marker.* The runtime layer includes *session* entities. These are used to keep track of moment-by-moment mappings between components and their instantiation. There is also a resolver function in this layer, *the runtime resolver.* Just like in the storage layer the runtime resolver provides a mapping between components and UIDs. The only difference is that the runtime resolver can use the session entities. At the heart of the runtime model is the session's *instantiator* function. This function takes as input a component UID and a presentation specification. The function return an instantiation of the component as part of the session.

3.5 Anchoring

> *The anchoring mechanism acts as an interface between the storage layer and the within component layer.*

This interface provides an indirect way to address parts within components. The anchoring addressing provides communication between the two layers while preserving the boundary between these layers. There is no concrete reference to parts in the within-component layer, only anchors are used.

3.6 Presentation Specification

The presentation specification forms part of the interface between the storage layer and the runtime layer.

The presentation specification is a primitive that specifies how the component is presented by the system during instantiation. Since presentation specifications are application and/or media dependent, Dexter makes no attempt to model their internal structure. Only the basic aspects needed for communication between these layers are defined.

Part II

Research

Chapter 4

Design Method Ontologies

Introduction

In literature, most website design methods are described in an informal manner. Such a description does not provide a clear syntactic and semantic specification. In this chapter I will describe each of the design methods from Chapter 2 in an ontology to provide a formal specification of the method. The representation of a website design method in an ontology language offers a wide range of benefits:

- *Unambiguous Semantics:* an ontology representation gives a clear and unambiguous formal semantic description of the method.
- *Reusability:* a formal description of a model makes it possible to reuse the model in other or future design methods.
- *Interoperability:* design models can be combined and even use features of each other.
- *Semantic Annotation:* the definition of the different models in an ontology language allows the reuse of existing domain ontologies. This way websites can provide semantic annotated data.

During the construction of the ontologies only the most important design phases of each methodology will be considered:

1. *Data Modeling or Conceptual Modeling.*
2. *Navigational Modeling or Hypertext Modeling.*
3. *Presentation Modeling.*

Some methods contain more extensive design models but they are not described as part of the ontology. The Dexter hypertext reference model will be used as a framework to construct these ontologies. This will allow us to better pinpoint

the meaning of the different concepts used in the different methodologies. As a concequence, each of the sections in this chapter will follow the structure of the Dexter reference model. As described in Chapter 3 his model contains three layers:

1. *Within Component Layer*
2. *Storage Layer*
3. *Runtime Layer*

We will situate each part of a design method meta-ontology in one of these layers. This means a mapping will be provided between the different design model concepts and Dexter reference model concepts wherever this is possible. The meta-ontologies are all created with a case tool called Protégé, a free, open source ontology editor and knowledge-base framework [24].

4.1 WebML Ontology

The WebML ontology contains three major parts: *"DataModelingConcept"* , *"HyperTextModelingConcept"* and *"PresentationModelingConcept"*. The first part describes the modeling of the conceptual model in E-R. The second part describes the modeling of the navigational model and the content management model in WebML notation. The last part describes how presentation is managed. Figure 4.1 shows the global structure of the WebML ontology.

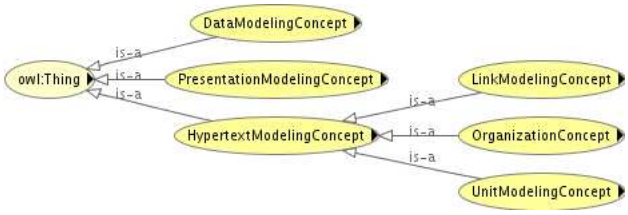

Figure 4.1: WebML ontology structure.

4.1.1 Within Component Layer

4.1.1.1 Conceptual Model

WebML uses the Entity Relationship (ER) model to describe the data used by the website. Figure 4.2 shows how the ontology is structured. The essential ingredients of the ER model are entities and relationships between these entities. A relationship can contain two or more roles between entities, this allows the modeling of n-ary relationships. Entities contain one or many attributes that describe the entity, each attribute has a name and a datatype. An attribute can be part of a primary key, a primary key provides a unique identity for an entity. Entities can be organized

into a hierarchy with the generalization relation. A generalization hierarchy has one super-entity and one or more sub-entities.

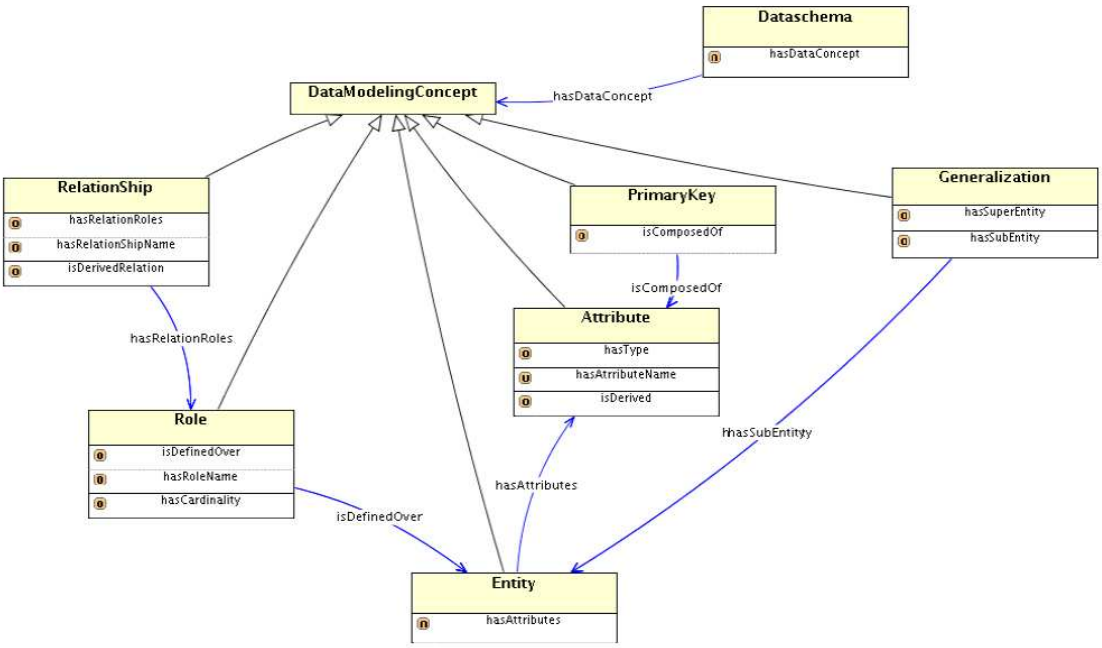

Figure 4.2: ER ontology

4.1.2 The Storage Layer

4.1.2.1 Units

Units are the atomic elements for specifying the content of a webpage. They correspond to atomic components in the Dexter reference model. We distinguish four different types of units (see Figure 4.3 and Figure 4.4):

- *"ContentUnit":* used to publish information on a webpage. A content unit is associated with one or more entities in the conceptual model.
- *"OperationUnit":* used to perform operations on information available on the website.
- *"EntryUnit":* used to enter new information.
- *"GlobalUnit":* used to hold global parameters for the website.

Content units

Content units present content extracted from the data model. There are five types of content units (see 2.1.2.1 on page 15) which all contain the following properties:

- *"hasName":* the name of the unit.
- *"hasSourceEntity":* a reference to entities in the data model used to specify where the information comes from.
- *"hasSelector":* a reference to a "UnitSelector" which contains a conjunction or disjunction of one or many predicates identifying which objects the unit should display. There are three types of predicates: a simple predicate is of the form: [attribute = value], a "ValueDisjunction" predicate is of the form [attribute operator value1 | value2 ...] and an "AttributeDisjunction" is of the form [attribute1 | attribute2 ... operator value].
- *"hasOrderBy":* a reference to an "OrderBy" object. An "OrderBy" contains a list of attributes used to sort the objects that the unit should display. Nesting is possible with the "NestedOrderBy" object.

Operation Units

Unlike content units, operation units do not display content but they perform operations on data. There are different types of operation units: object operation units and relation operation units. The first operates on objects in the conceptual model; the latter on relations in the conceptual model. Internally, they contain the following properties:

- *"hasName":* the name of the unit.
- *"hasSourceEntity":* the entity in the data model to which the operation applies.
- "*hasParameter":* the parameters used to perform the action.

An operation unit may have one or many input links, providing values for its input parameters. Each operation unit is connected with one "OKLink" and one "K0Link"; the former is followed when the operation succeeds; the latter when the operation fails.

Entry Units

Entry units are used to enter new information which can be provided to an operation unit (e.g., a HTML form). These units contain the following properties:

- *"hasName":* the name of the unit.
- *"hasEntryField":* one or many entry fields in which the user can enter or upload his information. Each entry field has an associated type and default value.

Global Units

Global units provide information that has to be available "globally" to all pages of a site. Each global unit is associated with a "GlobalParameter". There are two types of global units: a "SetUnit" or a "GetUnit", the former sets the value of a "GlobalParameter"; the latter gets the value stored in the global parameter. A global parameter contains the following properties:

- *"hasName":* the name of the parameter.
- *"hasGlobalParameterType":* the type of the parameter.
- *"hasDefaultValue":* the default value of the parameter.

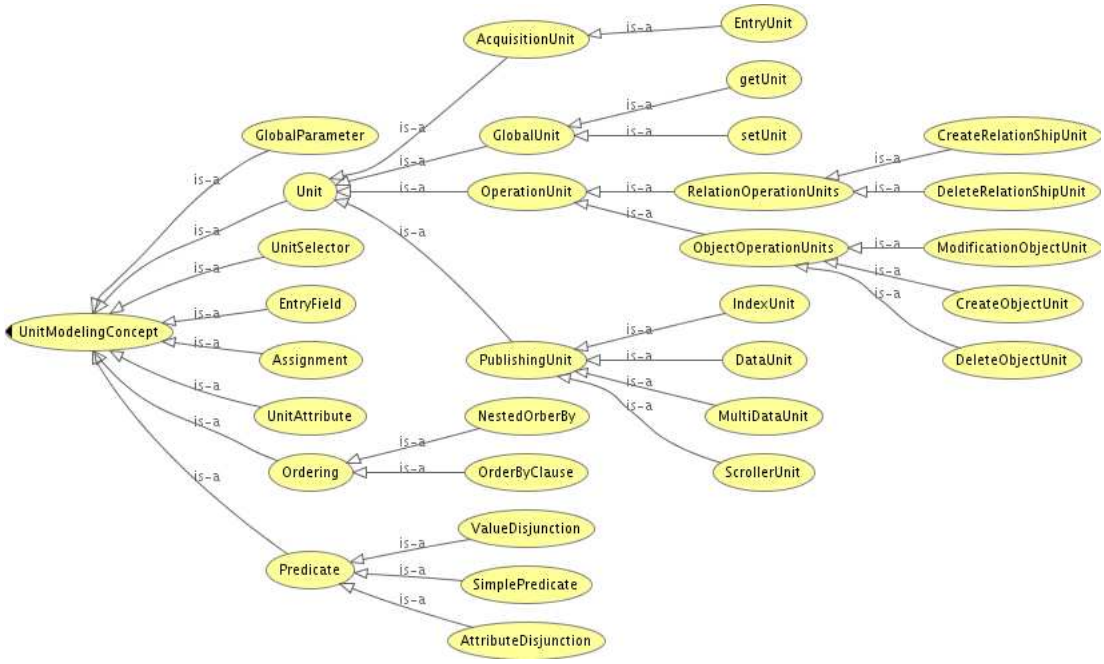

Figure 4.3: WebML unit modeling concepts.

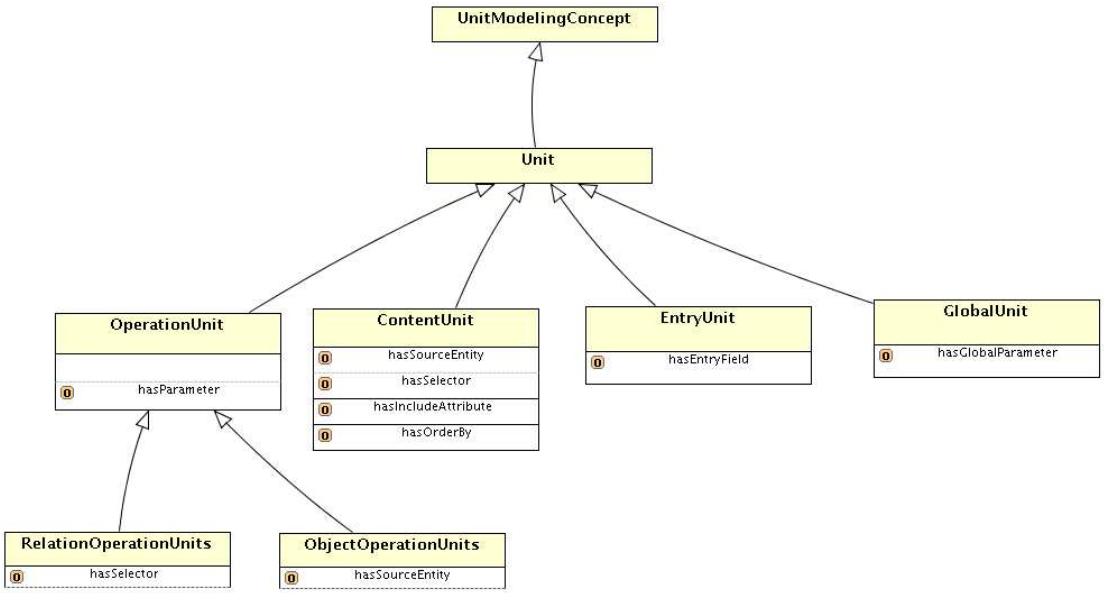

Figure 4.4: WebML units modeling concepts.

4.1.2.2 Navigational Model

The navigational model describes the collection of components and the links that interconnect them. Our ontology makes use of the following concepts to describe the WebML navigational model:

- *"OperationLink":* links attached to an operation unit used to denote the failure or success of an operation. An "OKLink" is followed when the operation is a success, a "KOLink" is followed when the operation fails. For example, when an on-line banking transaction is a success the "OKLink" is followed otherwise the "KOLink" is followed.
- *"NavigationLink":* these links are used to navigate through the hypertext. A navigation link contains a source unit or page and a destination unit or page.
- *"LinkParameter":* a link parameter is a property of a link that transports information from the source unit to the destination unit.
- *"Page":* a page is a collection of well-defined units. Pages are the actual interface elements delivered to the user.
- *"SiteView":* a collection of pages that form a particular view on the website.
- *"Area":* Areas are containers of pages or sub-areas, which can be used to give a hierarchical organization to a site view.

Figure 4.5 and 4.6 show the hierarchy of the WebML navigational ontology. Figure 4.7 gives a more detailed overview of the navigational ontology. Links can be between pages or units and always contain a source and destination denoting the begin and endpoints of the link. Links can contain many parameters used to transport information between units. A link parameter is described by a label and a type. The rest of the ontology is concerned with modularization of units. A siteview contains a number of areas which in contain one or many pages.

The Storage Layer of the Dexter reference model is composed of links, atom - and composite components. Pages, areas and siteviews correspond to composite components since they are composed out of several atomic components. Obviously, the link concept in the Dexter reference model refers to the link concept in our ontology. A link is specified by one or more endpoints. The endpoints contain a component specification which, in WebML, is given by the name of the units or pages. The presentation specification and anchor id is not given during navigation design since this is handled by the presentation model. WebML does not support bidirectional links or one-to-many links.

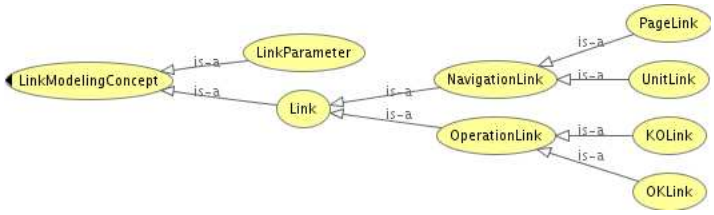

Figure 4.5: WebML organizational modeling concepts.

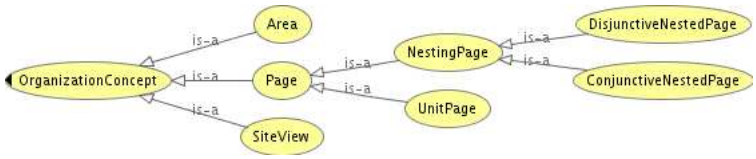

Figure 4.6: WebML link modeling concepts.

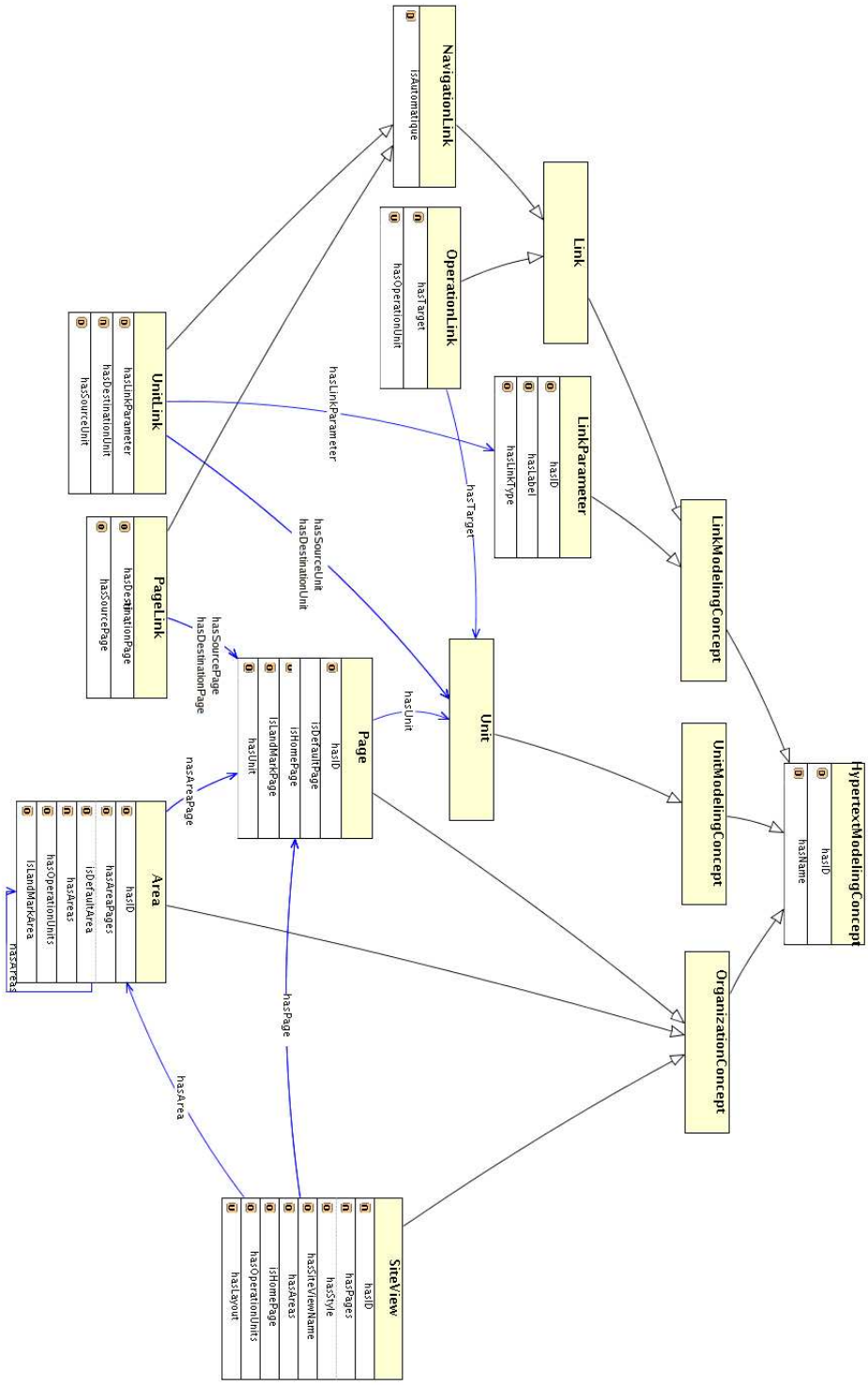

Figure 4.7: WebML Navigational model ontology.

4.1.3 Runtime Layer

4.1.3.1 Presentation Model

The Runtime Layer is concerned with the presentation of components to the user. Presentation addresses two concerns: layout and style. In WebML the layout and style of a webpage is accomplished with a case tool called WebRatio [7]. WebRatio uses a grid containing rows and cells to organize the layout of a webpage (see Figure 4.8). Each cell is associated with a page, unit, attribute or field from the navigational model, this association is done with the "hasReference" object property. An XSL style sheet defines the style of each cell. There is no additional information available about the presentation design in WebML since this is part of the case tool.

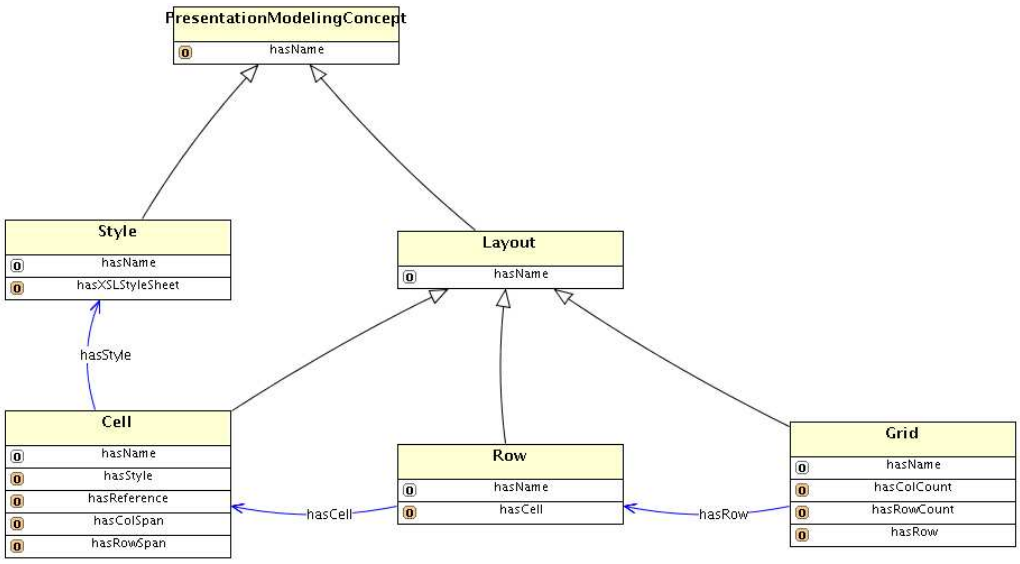

Figure 4.8: WebML presentation model ontology.

4.1.4 Anchoring & Presentation Specification

In the Dexter reference model anchoring serves as an interface between the Within Component Layer and the Storage Layer. In WebML anchoring is addressed by a unique ID that is assigned to every concept in the hypertext model. These IDs are used to refer to concepts within units. The presentation specification is an interface between the Storage Layer and the Runtime Layer. The same principle of unique IDs is used to refer to concepts in the Storage Layer. The specific modeling of anchoring and presentation specifications is done during the presentation design which is done in WebRatio.

4.2 Hera Ontology

The Hera ontology contains four important classes:

- *"ConceptualModelingConcept":* the class used for the construction of the conceptual model.
- *"ApplicationModelingConcept":* the class used to specify navigation modeling.
- *"PresentationModelingConcept":* the class used to specify the presentation model.
- *"Conditions":* used to specify adaptive behaviour on Hera components.

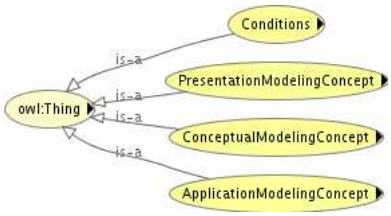

Figure 4.9: Hera ontology structure.

4.2.1 Within Component Layer

4.2.1.1 Conceptual Model

The CM is expressed in RDF(S) using two additional RDF(S) descriptions: CM properties and system media types. A CM property describes the inverse and cardinality of concept relations. The system media types describe the possible types of a concept (e.g., video, audio, string). An instance of a CM is expressed in RDF. Figure 4.10 shows how the CM ontology is structured. It is important to notice the mapping to RDF(S): "Attribute" and "Relation" are mapped to a “rdf:Property”, a “Concept” is mapped to a “rdfs:Class”. Since the conceptual model is defined in RDF(S) it is possible to use existing RDF(S) ontologies. The reader is referred to Chapter 1 for more information on RDF(S).

4.2.2 The Storage Layer

4.2.2.1 Slices

A slice is a meaningful presentation unit of some media items, which can both group content attributes and other slices. Each slice is associated with a concept in the CM. There are both top level slices, which represent pages of the website, and primitive slices, which only contain concept attributes. In the Hera ontology, a slice contains the following properties:

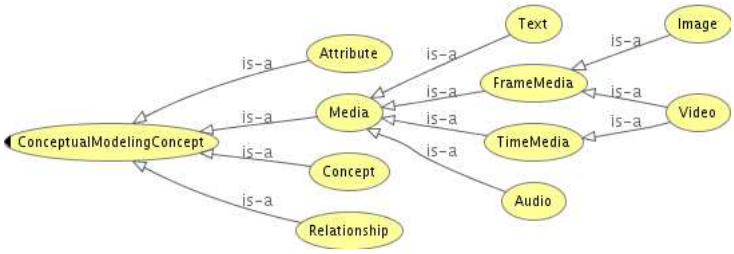

Figure 4.10: Hera conceptual model.

- *"hasName":* a slice name.
- *"hasConcept":* a reference to a concept in the conceptual model.
- *"hasAttribute":* slice attributes that refer to attributes in the conceptual model.
- *"hasAppearanceCondition":* a condition used to specify adaptation (e.g., only show this slice on a PDA).

Figure 4.11 shows where a slice is situated in the storage layer. Note slices represent both atomic and composite components in the Dexter reference model since a slice can contain sub-slices.

4.2.2.2 Application Model

The Hera application model is a navigation model on top of the data model. The navigational model contains slices which are connected with slice relationships. There are four kinds of slice relationships (see Figure 4.11):

- *"NavigationalRelation":* defines navigation between a source slice and a destination slice.
- *"CompositionalRelation":* define aggregation of slices (a slice contains another slice).
- *"SetOfLinks":* defines a one-to-many relation between slices, only the links to the other slices are shown in the source slice.
- "*SetOfSlices":* defines a one-to-many relation between slices, the slices are included as a whole in the source slice.

In the ontology, a relationship is specified with the following properties:

- *"hasAppearanceCondition":* an adaptation condition on the relationship (e.g., only show this link on a PDA).
- *"hasDestination":* a reference to the destination slice.

- *"hasSource":* a reference to the the source slice.
- *"hasCMReference":* a reference to a relation in the conceptual model.

Navigational relationships represent links in the Dexter reference model. A link in the Dexter model is specified by a set of endpoints which contain a set of specifiers. The Hera application model specifies the component specification with the name of the slices. The presentation specification is given in the runtime layer. There is no support for bidirectional links. Hera supports one-to-many links with the Set concept.

Hera also allows data entry by users with forms. Forms are owned by a slice and contain a various number of input fields. A form has the following properties:

- *"hasName":* a form name.
- *"hasConcept":* a reference to a concept in the CM. This concept is use to extract information from (e.g., a list of paintings to select from).
- *"hasFormItem":* a set of input fields. Each input field contains an input type (e.g., selection, text) and a value.
- *"hasHandOverItem":* a set of attributes this form hands over to the next in the navigation chain.

Forms are connected to other slices with queries. A query uses the information which is entered in the form to construct a specialized view over the conceptual model (e.g., only paintings from the selected artist are shown).

4.2.3 Runtime Layer

Figure 4.12 shows how presentation modeling ontology of Hera is structured. The PM is based on the AMACONT document format and it is specified in RDF(S). The PM contains four "LayoutManagers" and a "SubComponent" concept (for a description of the layout managers see section 2.2.3 on page 23). A layout manager is associated with a slice to define the overall layout of that slice. In addition, there are two types of "SubComponents": a "SliceRef" to define the layout of sub-slices and an "AccesElementRef" to define the layout of a "SetOfLinks" or "SetOfSlices". The next example shows an instantiation of a presentation model in RDF:

```
<Slice rdf:about=''#Slice.technique.main''>
  <layout rdf:resource=''#BoxLayout1''/>
</Slice>
...
<BoxLayout rdf:id=''BoxLayout1''>
  <axis>y</axis>
```

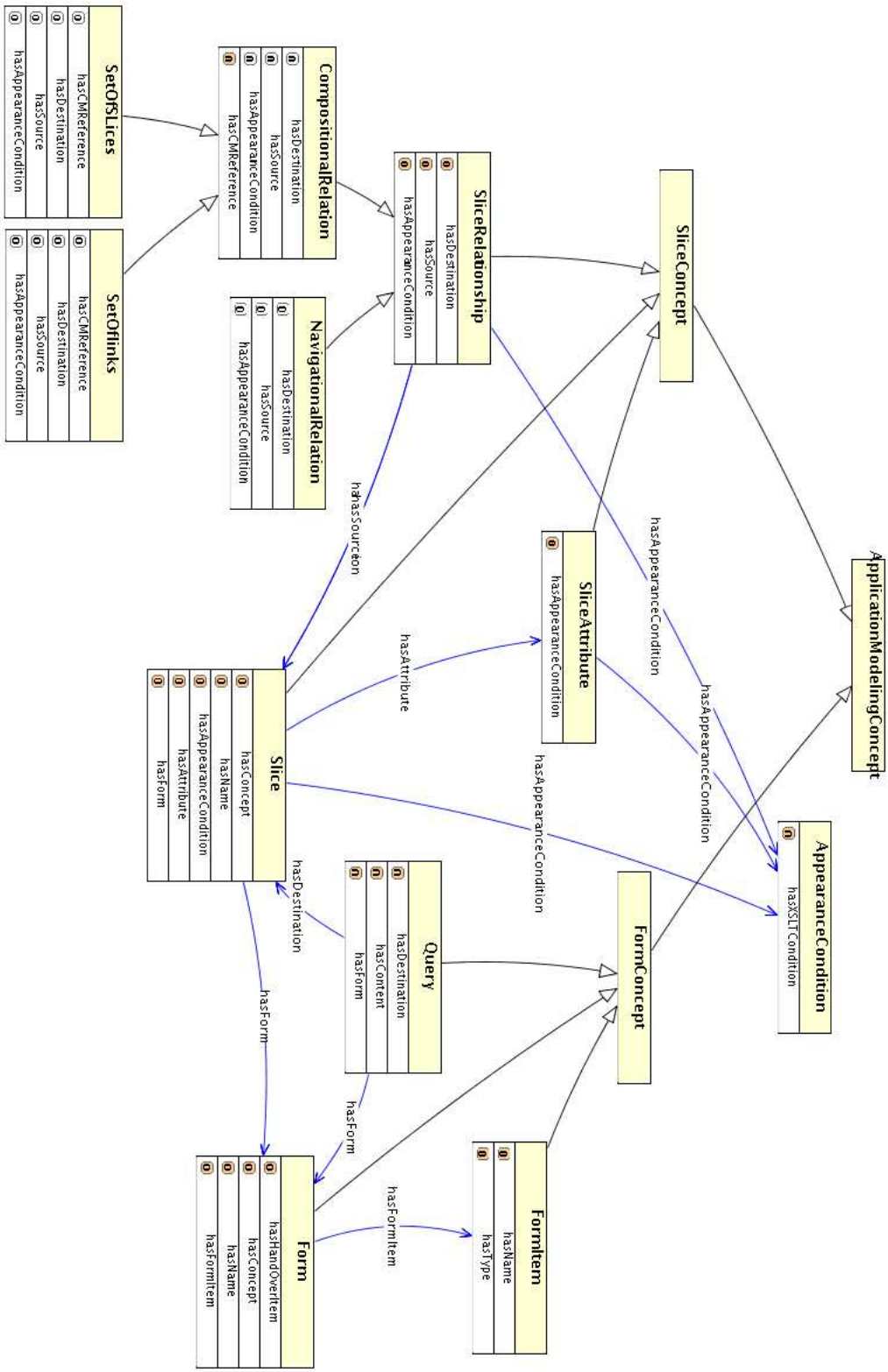

Figure 4.11: Hera application model ontology.

```
    <slice-ref rdf:resource="Slice.technique.tname" pres:halign="left"/>
  ...
  </BoxLayout>
```

The example describes the layout of a top-level slice "technique" as a BoxLayout.

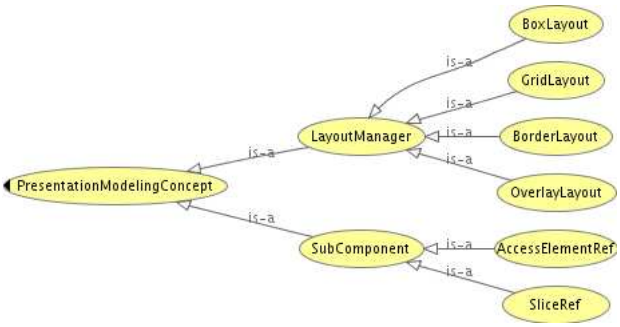

Figure 4.12: The presentation modeling ontology hierarchy.

4.2.4 Anchoring & Presentation Specifications

Anchoring provides a mechanism to refer to parts within components. In Hera it is possible to refer to parts of slices by using a concatenation of slice name and attribute name (see the example in section 4.12). The same way of addressing is used in the presentation specification which is specified in the PM.

4.3 OOHDM/SHDM Ontology

4.3.1 Within Component Layer

4.3.1.1 Conceptual Model

The newest version of OOHDM, called SHDM, expresses its conceptual model in OWL, but with additional restrictions such as requiring that all properties have a range and a domain. However, more information on this additional restrictions could not be found [3, 28–32]. See Chapter 1 for more information on OWL.

4.3.2 Storage Layer

4.3.2.1 Attributes

In SHDM, node attributes represent the atomic components in the Dexter reference model. Attributes from conceptual classes are mapped to node attributes and each node is associated to a conceptual class. There are three mappings defined in SHDM (see Figure 4.13):

- *"DirectlyMappedAttribute":* an attribute directly mapped from a conceptual class to a node attribute.
- *"DerivedAttribute":* an attribute derived from attributes in a conceptual class.
- *"ForeignAttribute":* an attribute from another conceptual class than the conceptual class associated to the node.

The mappings from attributes in the conceptual model to attributes in the navigational model (i.e. node attributes) are specified with RQL. This is possible since both the CM and the NM are specified in RDF(S) or OWL. The "NodeAttribute" concept is described in the ontology with the following data properties:

- *"isAnchor":* a boolean to define this attribute as an anchor. An anchor defines the starting point of a link.
- *"isIndex":* a boolean to define if this attribute is an indexed attribute (e.g., an index of student names).
- *"hasName":* the name of the attribute.

Moreover, a "ForeignAttribute" and a "DerivedAttribute" contain an RQL query statement on the CM.

4.3.2.2 Navigational Model

The navigational model contains two sub-models:

- *The navigational class model:* describes a view over the conceptual model or conceptual model (see Figure 2.10 on page 25 for an example).
- *The navigational context model:* describes how the user can navigate the navigational model in different contexts (see Figure 2.11 on page 26 for an example).

Navigational Class model

The navigational class model contains the following concepts (see Figure 4.13):

- *"Node":* a node is a collection of "NodeAttributes". Each node is associated with a class from the CM. The class "node" is a subclass of "rdfs:Class".
- *"Link":* a link is a connection between a source node and a destination node.
- *"NodeAttribute":* defined in the previous section.

A node corresponds to a composite component in the Dexter reference model since it contains many attributes which are atomic components. A link corresponds to a link in the Dexter reference model. Unlike Dexter, SHDM doesn't allow to to specify one-to-many links, instead a collection of one-to-one links can be used. Bidirectional links are not supported by SHDM.

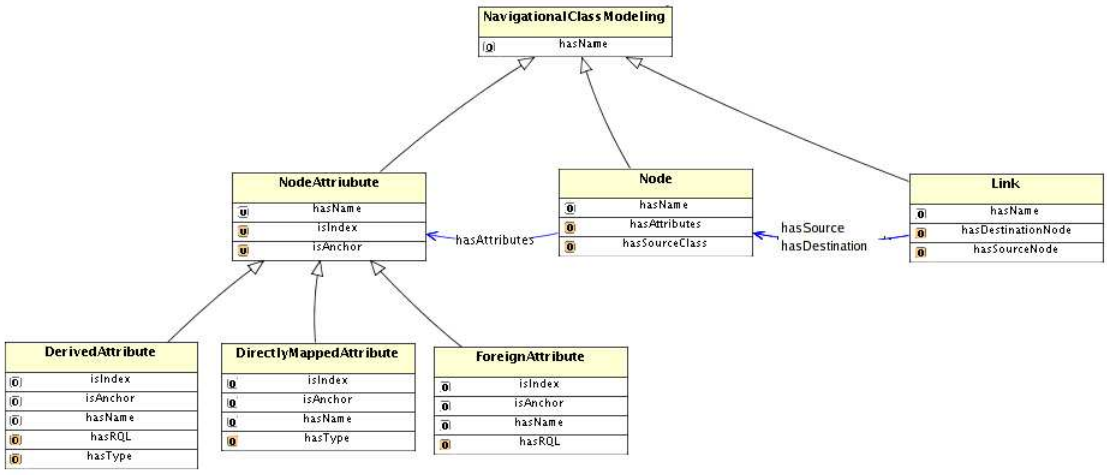

Figure 4.13: Navigational class model.

Navigational Context model

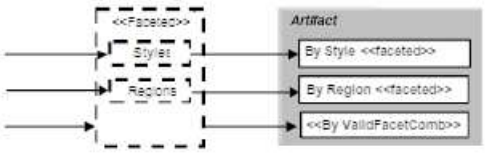

Figure 4.14: SHDM facet example.

The navigational context model is composed of:

- *"ConxtextClass":* a context class contains a group of context objects or sub context classes. A context class refers to a navigational class.
- *"ContextObject":* a context object describes how a set of objects can be browsed in a particular context. A context object is always owned by a context class. The detailed definition of a context object is given in the associated context definition card.
- *"ContextLink":* a context link is a link between access structures and/or context objects.
- *"AccessStructure":* an access structure defines how a context object or another access structure can be reached. It is associated with a class in the navigational class model.

- *"AccessStructureFaceted":* An access structure can contain many facets. The navigational context object can be reached by any valid combination of the facets. Figure 4.14 shows an example of an SHDM faceted access structure. The navigational context object "Artifact" can be reached by any valid combination of "Styles" and "Regions".

- *"AccessStructureBySubClass":* An access structure can contain a subclass tag. This allows navigation on all subclasses of the associated navigational class (e.g., when a user can navigate on person, he can also navigate on student).

- *"ContextDefinitionCard":* a detailed definition of a context object.

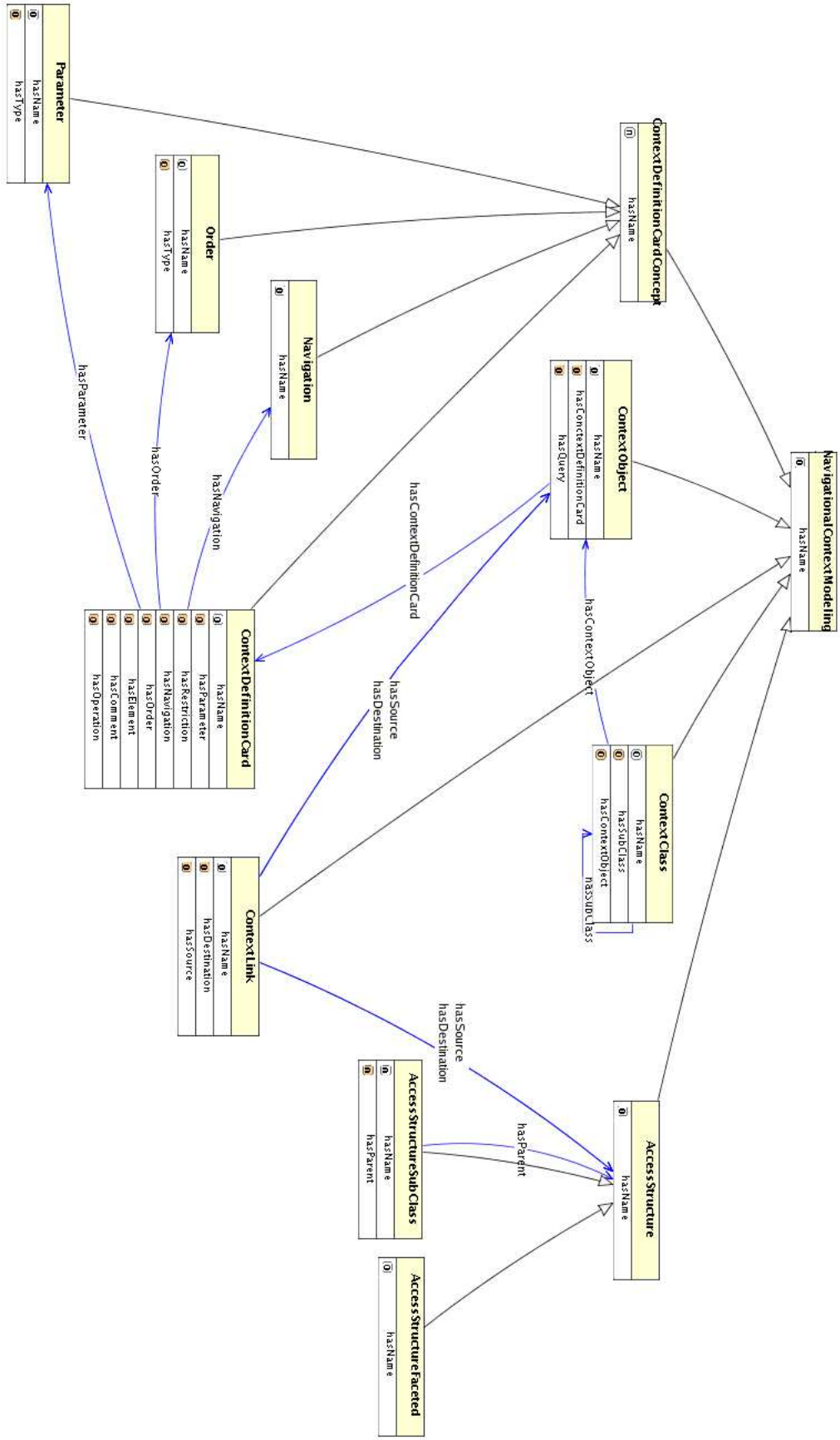

Figure 4.15: Navigational context model.

4.3.3 Runtime Layer

4.3.3.1 Abstract Widget Ontology

SHDM provides presentation design at a very abstract level: only information exchange between the user and the application is considered. This way the model is less sensitive to runtime environment aspects. The model is called the *Abstract Interface* and is specified with the *Abstract Widget Ontology* given in Figure 4.16. An abstract interface widget can be any of the following:

- *"SimpleActivator":* capable of reacting to external events, such as mouse clicks.
- *"ElementExhibitor":* able to exhibit some type of content, such as text or images.
- *"VariableCapturer":* able able to receive(capture) the value of one or more variables. This includes text boxes, pull down menus and checkboxes.
- *"IdefiniteVariable":* this widget allows entering of unknown values, such as a text string entered by a user.
- *"PredefinedVariable":* this allows the selection of some predefined values. *"DiscreteGroup"*, *"MulitpleChoices"*, *"ContiniousGroup"* and *"SingleChoices"* are all sub concepts that specify some choice method.
- *"CompositeInterfaceElement"*: a composition of any of the above.

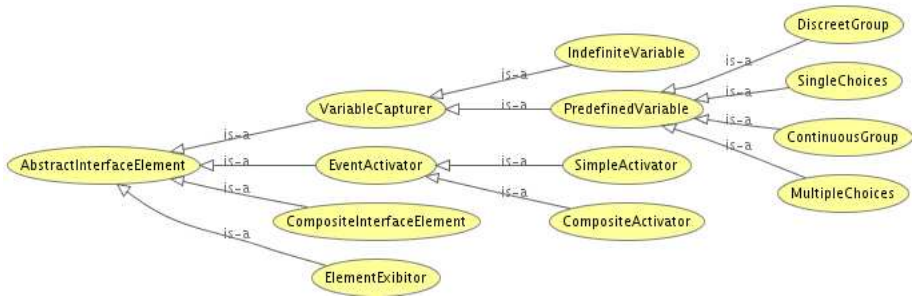

Figure 4.16: SHDM abstract widget ontology.

4.3.3.2 Concrete Widget Ontology

Besides the abstract widget ontology, SHDM also provides a concrete widget ontology (see Figure 4.17). Each abstract widget contains a mapping to a concrete widget. The concrete widget ontology contains the basic building blcoks to construct the interface of a website.

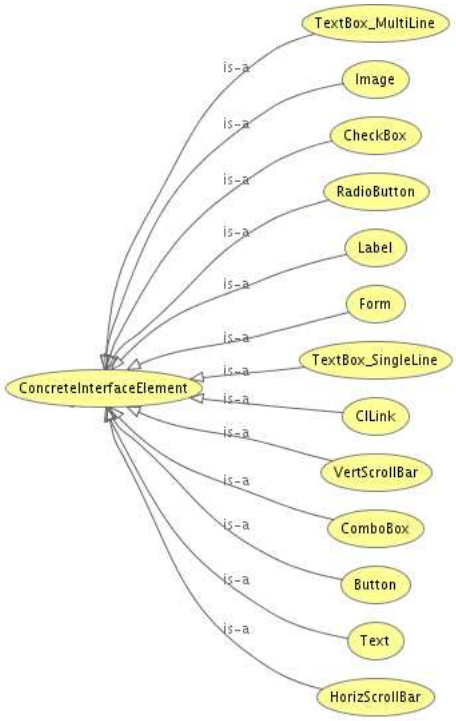

Figure 4.17: SHDM abstract widget ontology.

4.3.4 Anchoring & Presentation Specifications

The anchoring specification is in SHDM given by placing an "anchor" tag before an attribute in the navigational class model. This denotes this attribute is an anchor that is used to navigate to a different context. Presentation specification is not defined in SHDM since presentation modeling is not worked out in the methodology.

4.4 WSDM Ontology

The WSDM ontology was already elaborated by Sven Casteleyn and Peter Plessers who are both assistants at the WISE research group of the Vrije Universiteit Brussel [9]. The following sections will describe their ontology.

4.4.1 Within Component Layer

4.4.1.1 Object Chunks

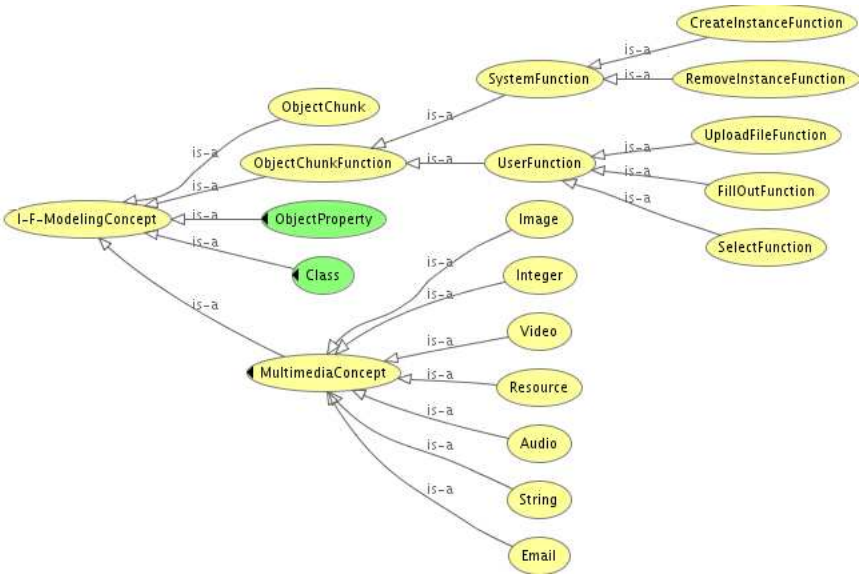

Figure 4.18: Information & Functional modeling concepts.

The first part of this design phase is Information Modeling. During this phase we model the different object chunks that form the business model. An object chunk can be seen as a tiny conceptual model that formalizes the necessary concepts to fulfil a particular information or functional requirement of a user [9].

In the WSDM ontology an object chunk is described by a collection of OWL statements consisting of two classes connected by a property. This means OWL itself is used to model the object chunks. As a consequence every ontology defined in OWL can be used by the WSDM design method. The darker ellipses in Figure 4.18 show how the mapping between WSDM object chunks and OWL is done. The ontology defines two classes: "Class" and "ObjectPopery" that both inherit from "owl:Class" and "owl:ObjectProperty" respectively. This means that every class of an object chunk is also an OWL class and each object property of an object chunk is also an OWL property.

Object chunks contain the following properties in the ontology (see Figure 4.19):

- *"hasInputReference":* references from other object chunks.
- *"hasOutputReference":* references to other object chunks
- *"isComposedOf":* references to the set of classes and properties belonging to this object chunk.

The functional modeling of a website is represented by the concept "ObjectChunkFunction". An "ObjectChunkFunction" represents a function that is applied to an object chunk. There are two sorts of functions (see Figure 4.18):

- *"SystemFunction":* a function applied by the system without any user interaction.
- *"UserFunction":* a function applied by interaction of a user.

There are two types of system functions:

- *"RemoveInstanceFunction":* indicates the system should create a new instance of the associated concept.
- *"CreateInstanceFunction":* indicates the system should remove an instance of the associated concept.

And we distinguish three user functions:

- *"FillOutFunction":* indicates the user has to fill out a value for the associated concept.
- *"UploadFileFunction":* indicates the user needs to upload a file for the associated concept.
- *"SelectFunction":* indicates the user has to select a value for the associated concept.

4.4.2 Storage Layer

4.4.2.1 Navigation Model

From Figure 4.19 we notice a navigation concept is either a "Node" or a "Link".

There are two special types of nodes: "ExternalNode" and "RootNode". A root node represents the root of the navigation hierarchy, an external node represents some external resource. Nodes(except external nodes) are composed of object chunks created during the Information & Functional modeling phase, they resemble a container of logical related information. Nodes represent the atomic components in the dexter reference model.

Links are the connections between nodes, they allow the navigation between nodes. There are four types of links:

- *Structural links:* these links provide the actual structure of the information and functionality being offered on the website.

- *Semantic links:* these links represent some semantic relation that exists between concepts of the nodes involved. For example, a teacher teaches a course. This means there is a semantic relation between teacher and course. In the navigation model this would resemble in a navigational link from teacher to the courses he/she teaches.

- *Navigational aid links:* these links are placed on top of the existing structural links to better facilitate navigation for the visitor. Examples of navigational aid links are home links and landmark links.

- *Process logic links:* these links connect nodes that express part of a workflow or invocation of an (external) functionality. An example is the pay process on e-commerce websites. The different nodes that are part of the payment process are connected with process logic links.

Another important concept is a *link parameter* which is used to transport information between a *source node* and a *target node*. A parameter is a reference to a concept of an object chunk connected to the source node. For example, a visitor selects an item on one page that is to be displayed on another page. The identifier of the selected item is a link parameter. Next to parameters there is also the possibility to define a *link condition*. Link conditions restrict the availability of the link to certain user classes, devices or time frames. WSDM allows, as does Dexter, to specify one-to-many links. This indicates a user can navigate from a start node to a set of end nodes. Since HTML doesn't support one-to-many links, they are implemented as a collection of one-to-one links. Bidirectional links are not supported by WSDM

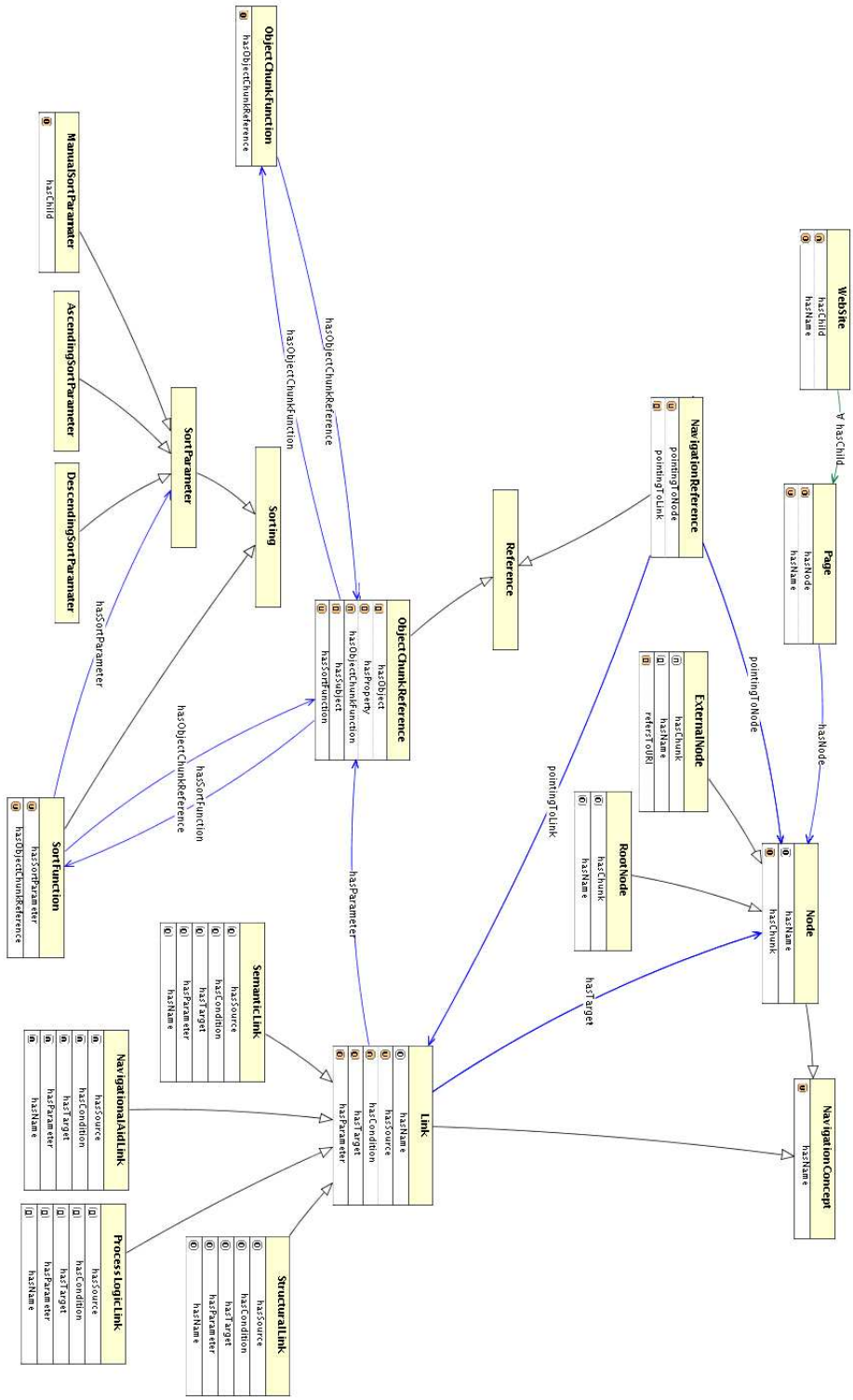

Figure 4.19: WSDM navigation model ontology

4.4.3 Runtime Layer

4.4.3.1 Style & Template Modeling

The aim of the Style & Template Modeling phase is to define templates that are used for webpages and to define styles for the objects on these pages. The specification of style is not included in the WSDM ontology instead stylesheets are used. Template specification is included in the ontology by the class "TemplateConcept". There are two types of template concepts: *independent template concepts* and *dependent template concepts.* The dependent template concepts are the building blocks used by the independent template concepts. A template can contain an "EditableRegion" concept, these are the concepts filled in during the Page Modeling phase. Figure 4.20 shows the organization of the ontology.

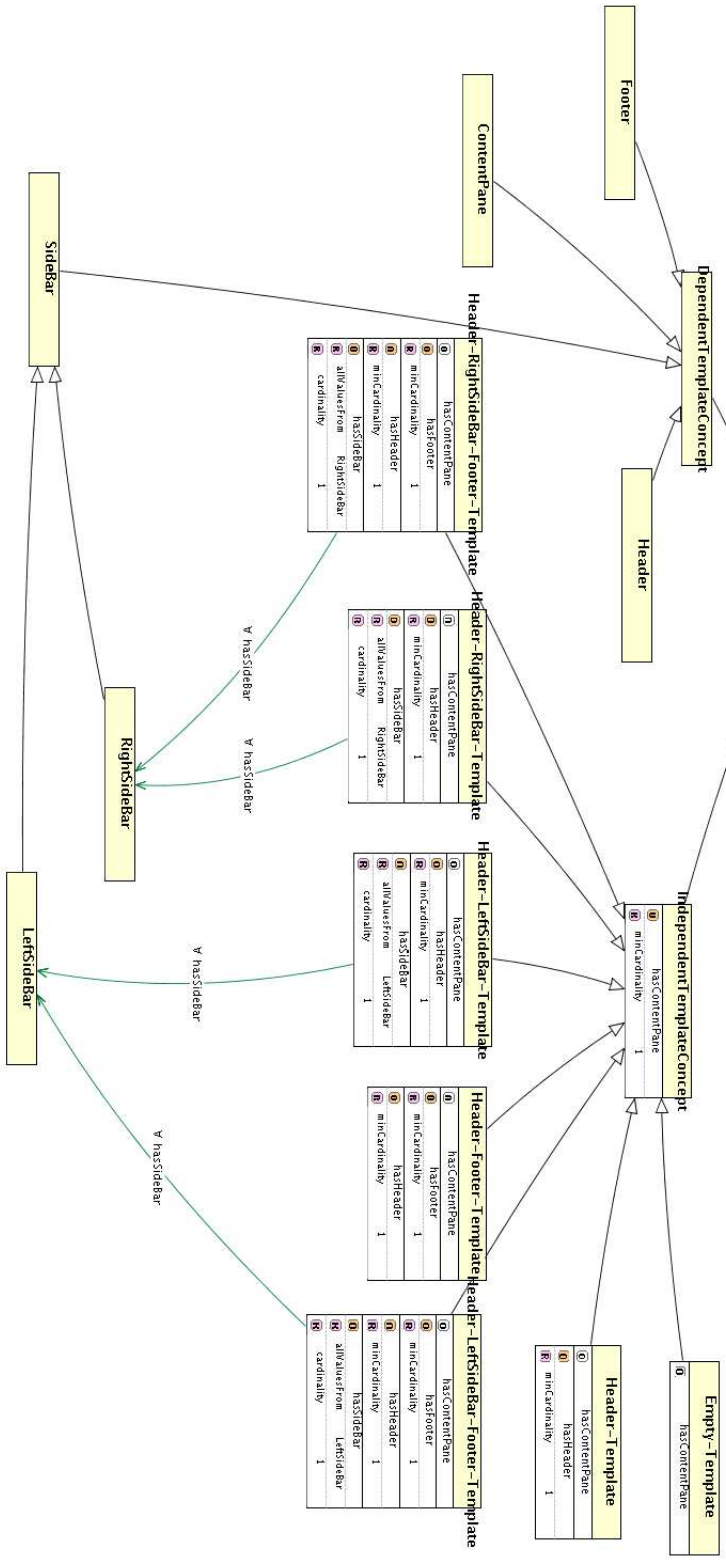

Figure 4.20: WSDM template concepts.

4.4.3.2 Page Modeling

The page model describes the structure and the layout of the different pages. A page is based on a template defined during the Style & Template Modeling phase. During this design step the editable region(s) specified in the template need to be filled out with presentation concepts. The ontology defines two types of presentation concepts: *primitive presentation concepts* and *complex presentation concepts.* Primitive presentation concepts are used to define page layout at a low level while complex presentation concepts allow design at a high and more intuitive level. This thesis only discusses primitive concepts, the reader is referred to [9].

Primitive Presentation Concepts

Figure 4.21 gives an overview of the primitive presentation concepts. Elements are positioned in a grid which contains rows, cells and editable regions (elements are not show in the ontology for simplicity). A presentation concept contains a "NavigationReference", which is a reference to a concept in the navigation model. This indicates which navigation concept needs to be displayed by the presentation concept. It is important to notice that a "NavigationReference" is the connection between the presentation model and the navigation model.

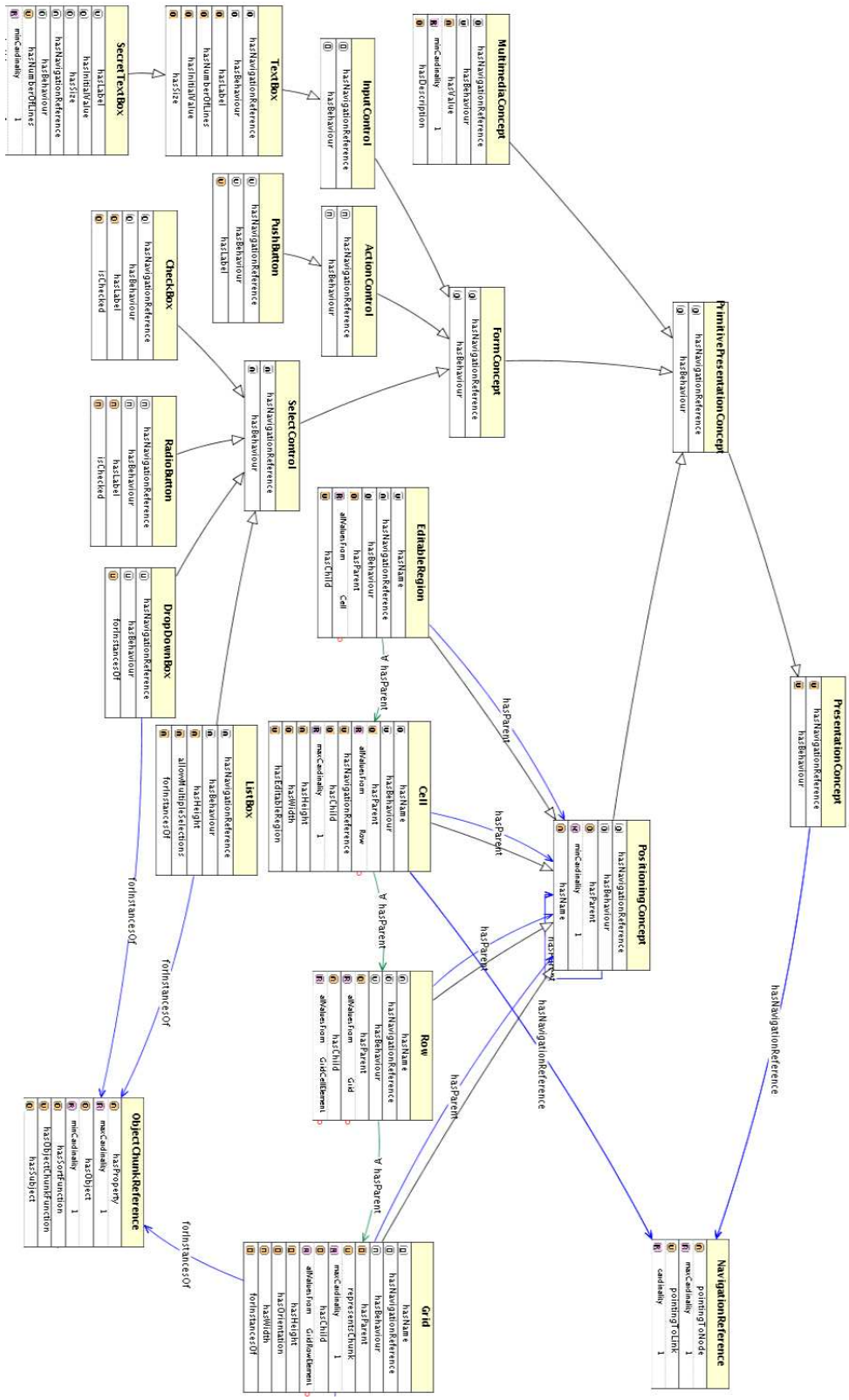

Figure 4.21: WSDM primitive presentation model ontology.

4.4.4 Anchoring & Presentation Specification

Anchoring provides a mechanism to refer to parts within components. In the WDSM ontolgy, "ObjectChunkReferences" are used to refer to parts within object chunks. A part of an object chunk is either a class or a statement . Similar, the concept "NavigationReference" is used to refer to the navigational concepts node and link. The reference mechanism provides a clear separation between the different layers.

Chapter 5

The Meta Model

Introduction

Chapter 4 described the construction of an ontology for each individual design method. The Dexter hypertext model was used as a reference model for each ontology. Concepts in the Dexter reference model were related to concepts in each methodology. In this chapter common concepts are abstracted from each design method. These abstractions will be used in the construction of a *design method meta model.* The aim of this design method meta model is to cover modeling techniques used in *all* methodologies. Therefore it should be possible to express the models of each design method of Chapter 2 in terms of concepts of the meta model ontology. The existence of a design method meta model offers the following benefits:

- **Interoperability:** the meta model acts as a mapping mechanism between model of different methodologies. This opens the path towards design method interoperability.
- **Standardization:** the meta model can be seen as a standard, models that are not able to map to the meta model may lack some modeling features.
- **Comparison of methodologies:** since each model can be mapped to a model in the meta model, models can be easy compared.
- **Standard terminology:** the design methods discussed in this thesis all use different terms for their concepts. The meta model introduces a standard terminology.
- **A foundation for future design methods:** the meta model can be used as a starting point for future design methods.

Section 5.1 uses the Dexter hypertext reference model to capture informally the important abstractions found in the design models of Chapter 4. Section 5.2 explains the meta ontology into detail.

5.1 Meta Model Construction

5.1.1 Within Component Layer

The within-component layer of the Dexter reference model is specifically concerned with the contents and structure *within* the components of the hypertext network. This layer is purposefully not elaborated within the Dexter model. The meta model ontology specifies the within-component layer by making abstractions of concepts explained in the within-component sections of each ontology described in Chapter 4.

5.1.1.1 WebML

WebML uses the ER (Entity-Relationship) model to specify its CM. The ER data model views the real world as a set of basic objects (entities) with attributes and relationships with roles among these objects. See 4.1.1 on page 39 for a detailed explanation of the WebML within-component layer.

5.1.1.2 Hera

In Hera, the CM is defined in RDF(S). RDF(S) uses classes, properties and relations to represent a conceptual schema. Hera extends RDF(S) with two additional properties: cardinality and media types. See 4.2.1 on page 47 for a detailed explanation of the Hera within-component layer.

5.1.1.3 SHDM/OOHDM

SHDM/OOHDM uses RDF(S) with additional constraints to model its CM. See 4.3.1 on page 51 for more information.

5.1.1.4 WSDM

WSDM uses object chunks to represent information on a website. An object chunk is a tiny conceptual schema modeled in OWL. The reader is referred to 4.4.1 on page 58 for more information.

5.1.1.5 Meta Model

The previous sections clearly indicate that each design method has an atomic component to represent a meaningful piece of information on a website. Moreover, each atomic component is a view over information stored in a conceptual schema which is modeled in a particular modeling language. It is possible to abstract common features from each modeling language. Table 5.1 shows how concepts from different modeling languages are mapped to the meta ontology. The table describes the following concepts:

- *CM Language:* the language used to model the conceptual schema. The conceptual schema is the foundation of an atomic component.
- *Class:* a class is a description of a group of objects with similar attributes, common operations, common relationships and a common semantic purpose.
- *Property:* a property is a characteristic of an class.
- *Relation:* a relationship is a connection or link between classes or objects.
- *Specialization:* a relationship between a subclass and a superclass which indicates a "is-a" relation.
- *Cardinality Constraint:* a constraint on a relation denoting the minimum and maximum number of objects of the destination entity to which an object of the source entity can be related.
- *Value Constraint:* a constraint that restricts the possible values an object or data property can have.

It is important to notice that most of the mentioned design methodologies use a Web ontology language like OWL or RDF(S) to specify their conceptual model. The use of a Web ontology language has the following advantages:

- *Reuse* of existing domain ontologies.
- *Exchange* of conceptual models without mapping.
- *Semantic annotation* of the information.

Only WebML uses the ER model which is originally used for database modeling. However, it is possible to map an ER model to an OWL or RDF(S) model. As said in Chapter 1, RDF(S) does not have the same modeling capabilities as OWL. It is obvious that a conceptual model expressed in OWL is preferred above any other. WSDM is the only method that specifies its model in OWL. SHDM also claims to specify their CM in OWL but they provide no concrete examples, only CM in RDF(S) are provided. Hera specifies its CM in RDF(S), but with additional properties such as cardinality and media types.

Method	Meta Model	WebML	Hera	SHDM OOHDM	WSDM
CM language	OWL	ER	RDF(S), with cardinality and media types	RDF(S), with cardinality.	Object chunks in OWL
Class	Class	Entity	Class	Class	Class
Property	Data type property	Attribute	Property	Property	Data type property
Relation	Object property	Relationship, role	Property	Property	Object property
Specialization	SubclassOf	Sublclass relation	SubclassOf	SubclassOf	SubClassOf
Cardinality Constraint	yes	yes	yes	yes	yes
Value Constraint	yes	yes	no	no	yes

Table 5.1: Within-component layer mapping.

5.1.2 Storage Layer

The storage layer describes how components and links form a hypertext network. The storage layer models a hypertext network as a finite set of components consisting of:

- *Atomic components:* the atomic pieces of information represented on a webpage.
- *Composite components:* a composition of atomic components.
- *Link components:* connections between components.

The storage layer is described in a navigational model. The following sections will give a brief overview of the components used in the navigational model of each design method.

5.1.2.1 WebML

WebML models the storage layer in the hypertext model. The key ingredients of the WebML hypertext model are pages, units, and links, organized into modularization constructs called areas and site views. WebML allows the modeling of a hypertext system which allows publishing, acquisition, manipulation and organization of information in the data schema.

Information Publishing

Units are the atomic pieces of publishable content, the correspond to atomic components in the Dexter reference model. Content units offer a way to dynamically arrange content extracted from the conceptual schema. In fact, content units perform a query on the conceptual schema. The result of this query is sorted and placed on a page. As a consequence, a page can contain the result of many queries performed by units. There exist many different types of information publishing units like an index unit, multi-data unit etc., these are all specializations of the global type content unit.

Information Acquisition

Units can also be used to acquire information from the user, this is done with entry units. Entry units support form-based data entry from users. Each entry unit contains a number of input fields in which the user can enter or select information. Entry units are used to perform searches over the objects of an entity or to supply parameters for operations.

Information Manipulation

Data manipulation is modeled with operation units. An operation unit denote either data manipulation or the execution of a generic external service. WebML provides a set of predefined operation units which offer the most commonly used primitives for updating, creating, modifying and deleting objects in the data schema.

Information Organization

Units are connected with each other by links. Links offer a way to navigation and/or transport information between units, they allow basic interaction between the user and the hypertext. Information transport is accomplished with link parameters. These are parameters attached to a link which transport information from a source unit to a destination unit. Organization of units is done with areas, pages and site views. Units are grouped on pages, areas and site views offer a way to group pages into modularization constructs.

Information Adaptation

WebML does not support adaptation.

For a detailed explanation of the WebML design method the reader is referred to 2.1 on page 13 and 4.1 on page 39.

5.1.2.2 Hera

The application model of Hera specifies the navigational aspects of the hypertext system. Hera uses slices, slice composition and slice navigation to organize the information on webpages. Slices are both atomic and composite components since a slice can contain many sub-slices.

Information Publishing

A slice is a meaningful grouping of concept attributes that need to be shown together in the hypermedia presentation. Slices provide a view over concepts in the conceptual model. As a consequence, each slice is associated with a concept in the conceptual model. Slices can contain other slices with the compositional relation, top-level slices (slices at the top of the hierarchy) represent webpages of the website.

Information Acquisition

Hera allows the acquisition of information from users through forms. Forms contain a various number of input fields in which the user can enter or select information. Queries use the information entered in the form to create specialized views over the conceptual schema.

Information Manipulation

Hera does not allow manipulation of data in the conceptual schema, only queries over the conceptual schema are allowed.

Information Organization

Slices are organized in a hierarchy with relationships. There are two kinds of relationships: compositional relationships (aggregation between slices) and navigational relationships (navigation between slices). Slices can be used to group concepts from the CM.

Information Adaptation

Hera supports adaptation with with conditions on slices and relations. A condition returns a boolean value that specifies if the associated component is shown or not (e.g. only show this information on a PDA when condition "system = PDA" is true).

For a detailed explanation of the Hera design method the reader is referred to 2.2 on page 18 , the ontology is explained on 4.2 on page 47.

5.1.2.3 SHDM/OOHDM

SHDM uses two different models to model the hypertext network: a navigational class model and a navigational context model. The first is used to model what information is available; the latter is used to model how the available information can be browsed.

Information Publishing

SHDM extracts information from the conceptual schema with attributes. Attributes are part of navigational nodes and perform a query on the conceptual schema. A navigational node can contain many attributes, each node represents a webpage.

Information Acquisition

SHDM allows the acquisition of data through the abstract widget ontology. The abstract widget ontology allows the modeling of simple structures that capture user input. However, there is no information available on how SHDM maps abstract widgets to navigational classes.

Information Manipulation

SHDM does not provide modeling support to manipulate data in the conceptual schema.

Information Organization

The navigational class schema describes a collection of navigational classes and links. It describes what information is available on the website and how users can navigate through this data. The navigational context schema is a collection of access structures (links) and navigational context classes. The navigational context schema defines how the information can be reached and browsed. Context definition cards provide a more detailed way to define context classes (e.g., sorting of information). SHDM does not provide modeling support to organize data into modularization constructs.

Information Adaptation

SHDM does not support adaptation.

For a detailed explanation of the SHDM design method the reader is referred to 2.3 on page 24 , the ontology is explained on 4.3 on page 51.

5.1.2.4 WSDM

WSDM uses nodes, object chunks and links to represent its navigational schema.

Information Publishing

WSDM uses nodes to represent information that needs to be available on the website. A node is a collection of object chunks. The information in these object chunks can be sorted with a sorting function.

Information Acquisition

User functions are functions defined on object chunks which allow acquisition of information from users. The acquired information is stored in an instance of an object chunk.

Information Manipulation

System functions are functions defined on object chunks, they allow to create, delete and modify instances of object chunks.

Information Organization

WSDM defines the navigational schema as a collection of nodes interconnected by links. Links can be equipped with link parameters to allow information transport between nodes. Nodes are grouped on pages which represent the actual webpages of the website.

Information Adaptation

WSDM supports adaptation with conditions on links (e.g., only show a link when a condition is true).

For a detailed explanation of the WSDM design method the reader is referred to , the ontology is explained on

5.1.2.5 Meta Model

From the previous sections we can abstract a few common functionalities supported by (most) models:

- *Information publishing* with atomic or composite components. The published information can be sorted according to sorting parameters.
- *Information acquisition* from users. The resulting information is stored in the CM.
- *Information manipulation* allows users to create, modify and delete objects in the CM.
- *Information organization* allows to group and structure information in a hypertext network.

- *Information adaptation* allows the information to adapt to the user or the runtime environment.

Table 5.2 shows how concepts from the different design methods are mapped to the meta model.

5.1.3 Runtime Layer

Remind from chapter 3 that the runtime layer describes how instantiated components are presented to the user. The next sections will give an overview of the structure of the runtime layer in the different design methods. The reader is referred to Chapter 4 for more details on this subject.

5.1.3.1 WebML

WebML uses a grid layout to organize concepts on a webpage. A grid contains rows, each row contains a number of cells. Every cell is associated with a concept and has its own style and behaviour. The style of elements is defined with stylesheets. A case tool called WebRatio is used to construct the presentational model. There is no additional information in literature about this case tool.

5.1.3.2 Hera

Hera uses the layout managers from AMACONT to organize concepts on a webpage. Layout managers provide the same functionality as a grid layout but at a higher abstraction level. Style of concepts is defined in stylesheets. Hera does not provide support for complex interface and behaviour modeling.

5.1.3.3 SHDM/OOHDM

SHDM does not provide support for layout and style modeling, this task is left to the graphic designer of the website. However, it is possible to define a user interface with the abstract and concrete widget ontologies. It is not possible to associate a behaviour to concepts.

5.1.3.4 WSDM

WSDM models the layout of a webpage with templates and presentation concepts. A template is defined that defines the overall layout of a webpage, this template is filled in with presentation concepts which define the actual interface elements delivered to the user (e.g., a form, audio etc.). WSDM also allows to associate a behaviour with presentation concepts.

Method	Meta Model	WebML	Hera	SHDM OOHDM	WSDM
Information publishing	Atom component, sorting functions on components	Content units, orderby clause to sort units	Slice, no sorting of information.	Attribute, no sorting of information.	Node with object chunks, sorting with functions on object chunks
Information acquisition	User functions	Entry unit	Forms in slices	Abstract widget ontology	User functions
Information manipulation	System functions	Operation unit	No manipulation of information	No manipulation of information	System functions
Information organization	Composite component, link between components, page	Page, area, siteview, links between units and pages	Slice composition, slice navigation	Navigational node, context classes, navigational link	Node, page, links between nodes and pages
Information adaptation	Adaptation conditions on components	No adaptation support	Condition on slices and attributes	No adaptation support	Conditions on object chunks

Table 5.2: Storage layer mapping.

5.1.3.5 Meta

Again, there are some common functionalities in each of the presentational models. Table 5.3 summarizes the following aspects for each design method:

- *Layout:* the layout of a website describes the placement of concepts on a webpage.
- *Style:* style defines the look & feel of a website (e.g., text size, color etc.).
- *Interface:* the interface describes the collections of elements provided by the model to construct a user interface (e.g. form, vertical scrollbar, audio..).
- *Behaviour:* behaviour defines actions that occur on certain events (e.g., a link colors red on mouse click).

The presentation layout of WebML is completely managed by WebRatio, a case tool for WebML. There are no implementation details available for this case tool. Hera uses AMACONT to specify the layout of its webpages. The style of webpages is defined with stylesheets. There is little support to model a concrete user interface. SHDM only provides support for basic user interfaces with the abstract and concrete widget ontology, presentation is left to the graphic designer. WSDM uses templates and a grid with presentation concepts to manage the layout of a webpage, style of presentation concepts is defined with stylesheets. Only WebML and WSDM provide support to model a complete user interface including layout, style, presentation concepts and behaviour.

Method	Meta Model	WebML	Hera	SHDM OOHDM	WSDM
Layout	Grid, row and cells	Grid, row and cells	Layout managers from AMACONT	not supported	Page templates, grid, row and cells
Style	Reference to stylesheet	WebRatio, stylesheets.	Stylesheet	Stylesheet	Stylesheet
Interface	Wide range of interface elements	WebRatio	Only media types	Abstract & concrete widget ontology	Wide range of presentation concepts
Behaviour	Events & actions on presentation concepts	WebRatio	not supported	not supported	Events & actions

Table 5.3: Runtime layer mapping.

5.2 Design Methods Meta Ontology

The ontologies for the website design methods described in Chapter 4 all distinguish three layers: a conceptual layer, a navigational layer and a presentational layer. This three layer ontology specification has one important purpose: *separation of concerns*. The design method meta model also uses this structure and distinguishes three sub-models (see Figure 5.1 for an overview):

- *"ConceptualModelingConcept":* concepts to describe the application domain, these concepts are used in the conceptual schema.
- *"NavigationalModelingConcept":* concepts uses to describe the hypertext network, navigation and user interaction. These concepts are used in the navigational schema.
- *"PrestationalModelingConcept":* concepts used in the presentation schema to describe presentation and behaviour of concepts.

The "WebSiteConcept" contains two concepts to describe a website as a collection of pages.

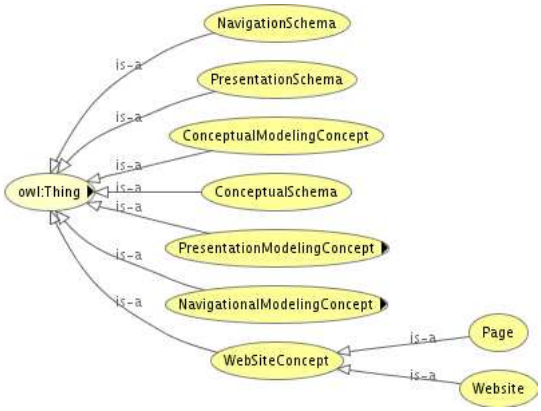

Figure 5.1: Meta model global overview.

5.2.1 Conceptual Meta Model

5.2.1.1 Conceptual Meta Model Description

Figure 5.2 shows how the *Conceptual Meta Model ontology* is structured. The figure shows how concepts in the conceptual meta model are mapped to OWL concepts (see 1.3.2 on page 10 for more details on OWL). The CM of the meta model is completely specified in OWL. The the data types of OWL are extended with "MultiMediConcepts" these are data types frequently used in websites (e.g., video, image etc.).

5.2.1.2 Conceptual Meta Model Mapping

Table 5.4 shows how concepts in the CM from the different design methods are mapped to concepts in the conceptual meta model. The CM from WebML needs a mapping from ER to OWL. Hera and SHDM use RDF(S) to describe their CM. Since the meta model is described in OWL, almost no mapping is required (OWL is based on RDF(S)). Note that both Hera and SHDM extend RDF(S) with cardinality constraints. Hera also uses media types to specify the data type of an attribute. The CM of WSDM does not require a mapping since it is already defined in OWL.

Meta Model	WebML	Hera	SHDM OOHDM	WSDM
Class	Entity	Class	Class	Class
Relation	Role and Relation	RDF(S) properties	RFD(S) properties	OWL Properties
Property	Attribute	Attribute	Attribute	Property
Cardinality Constraint	Cardinality on roles	Cardinality on relations	Cardinality on relations	Cardinality on relations
Subclass	Specialization	Subclass	Subclass	Sublclass
Value Constraint	Enumeration	Not supported	Not supported	OWL value constraints
Multimedia concepts	ER data types	Media types	Not supported	Media concepts

Table 5.4: Conceptual meta model mapping.

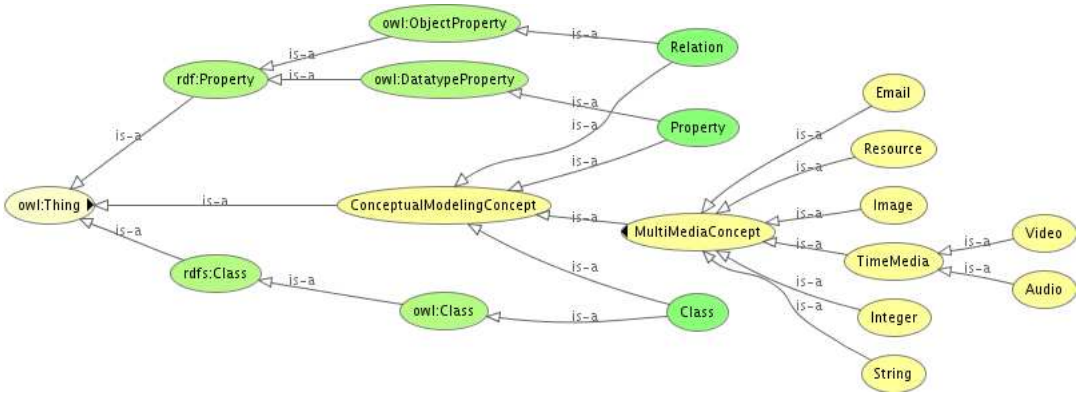

Figure 5.2: Conceptual meta model.

5.2.2 Navigation Meta Model

Basically, the navigation meta model is a collection of atoms and composite components connected by links. Each atom contains information which is the result of a query on the conceptual schema. Composites provide a mechanism to group this information on pages, links provide a way to navigate through this information. In addition, functions on atoms provide a way to manage information in the conceptual schema. Some navigation models allow webpages to adapt themselves to the runtime environment (e.g., no color on WAP). Figure 5.3 gives a hierarchical overview of the navigational meta model. Figure 5.4 gives a more detailed overview. Table 5.5 shows how concepts from the different design methods are mapped to the meta ontology.

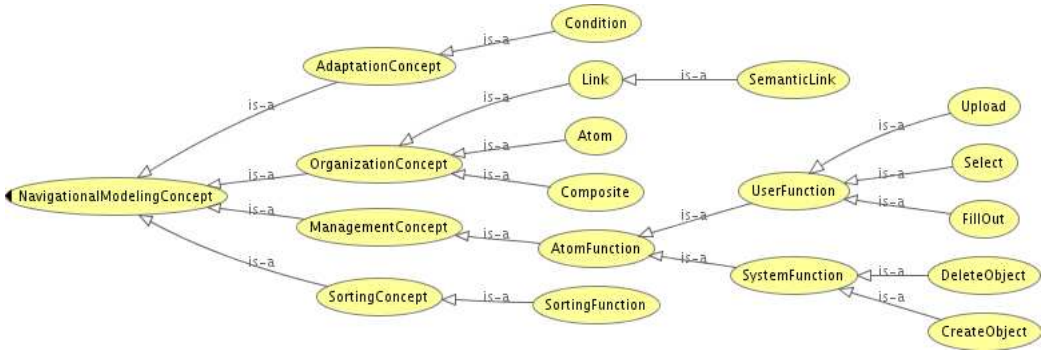

Figure 5.3: Navigational meta model hierarchy.

5.2.2.1 Navigation Meta Model Description

Atom

- *"hasName":* a unique name.
- *"hasSourceEntity":* a reference to one or more classes (i.e. entities) in the conceptual model.
- *"hasSetting":* a setting to stor boolean variables.
- *"hasSourceAttribute":* a reference to one or more attributes in the source entities.
- *"hasAtomFunction":* a function on the atom that allows this atom to acquire information or manage information in the conceptual schema.
- *"hasSelectionClause":* a selection predicate that determines which objects are selected from the conceptual schema.
- *"hasSorting":* a sorting function that allows to sort the information in this atom on a sorting parameter (e.g. ascending).
- *"hasAdaptation":* an adaptation condition that specifies in which situations the information in this atom is shown.

Composite

- *"hasName":* a unique name, mostly the name of the webpage.
- *"hasAtom":* a reference to one or more atoms.
- *"hasAdaptation":* an adaptation condition.

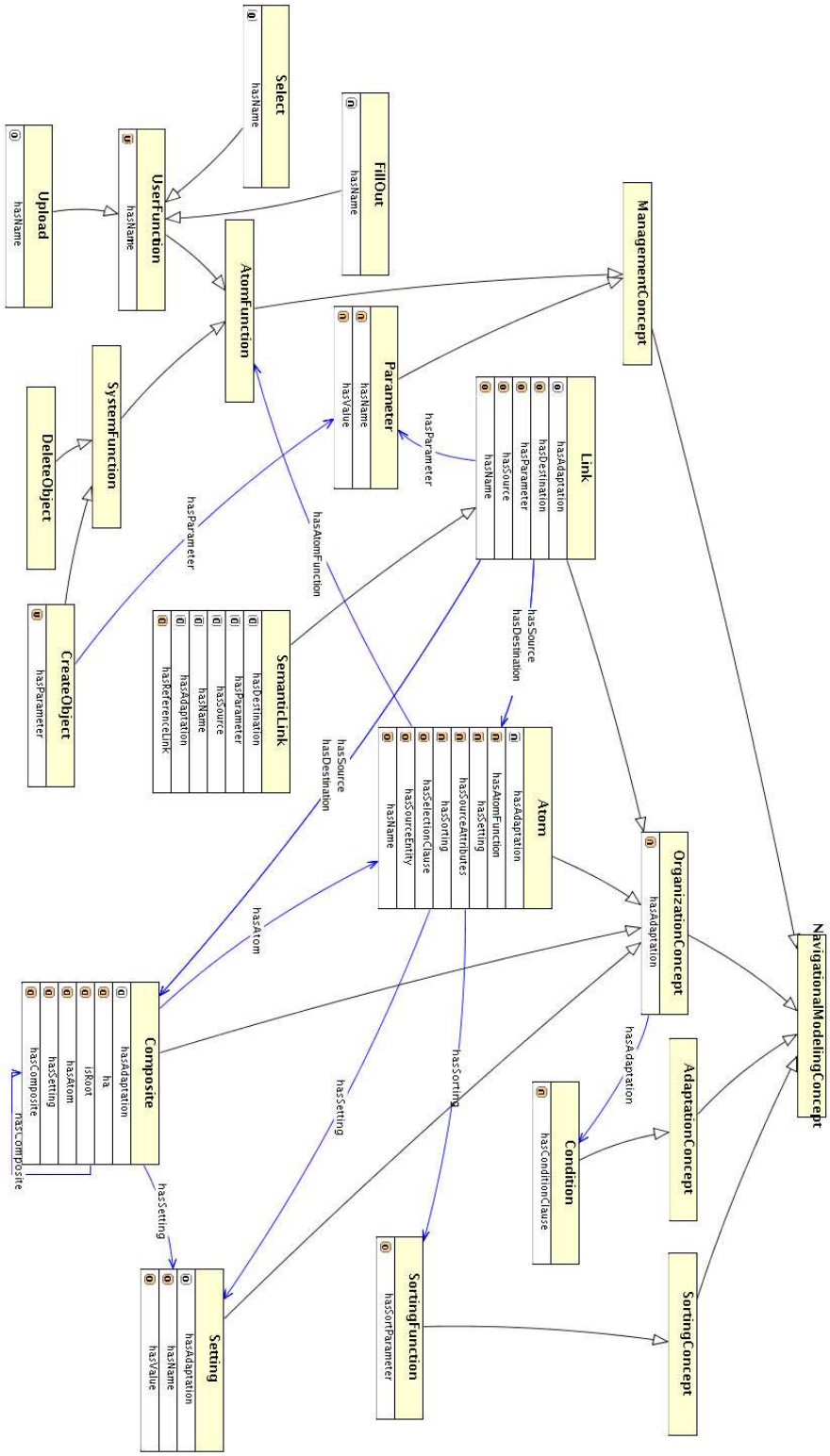

Figure 5.4: Navigational meta model.

Link

- *"hasName":* a unique name
- *"hasSource":* a reference to an atom denoting the source of this link.
- *"hasDestination":* a reference to an atom denoting the destination of this link.
- *"hasAdaptation":* an adaptation condition.
- *"hasParameters":* one or more parameters that are able to store information which need to be transported between atoms.
- *"hasReferenceRelation":* [only in case of a semantic link] a explicit reference to a relation in the conceptual schema.

5.2.2.2 Management Concepts

Management concepts provide functionality to manage information in the conceptual schema. There are two sort of functions: *user functions* and *system functions.* User functions allow to acquire information from users and insert this information in the conceptual schema. System functions allow to create, delete, and modify objects in the conceptual schema with minimal user interaction.

5.2.2.3 Sorting Concepts

A sorting function is associated with an atom and allows to sort the information stored in this atom. Sorting functions are associated with a sorting parameter which defines how the information is sorted (e.g. ascending, alphabetic, etc.).

5.2.2.4 Adaptation Concepts

Adaptation concepts are associated with components in the navigational model. They allow the navigation model to adapt to a specific runtime environment (e.g., do not use color if the runtime environment is WAP).

5.2.2.5 Navigation Meta Model Mapping

WebML

WebML uses content units to publish information on webpages, these units are mapped to the meta model concept "Atom". All properties from the content unit can be mapped to properties in an atom (see 4.1 on page 39 for details on unit properties):

- "hasIncludeAttributes" =>"hasSourceAttributes"
- "hasOrderby" => "hasSorting"

Meta Model	WebML	Hera	SHDM OOHDM	WSDM
Atom	All units	Slice	Attribute, context object, context definition card	Node referring to object chunks
Composite	Page, area, siteview	Slice aggregation	Node, context class, access structure	Page
Link	All links with optional properties	Slice-Relationship and queries	Navigation links, context link	All links
Condition	/	Conditions on navigation concepts	/	Conditions on links
Sorting function	Orderby clause in content units	/	Context definition cards	Sorting function on object chunks
User function	Entry units	Forms in slices	Abstract widgets	User function
System function	Operation units	/	/	System functions

Table 5.5: Navigation meta model mapping

- "hasSelector" =>"hasSelectionClause"
- "hasSourceEntity" => "hasSourceEntity"
- "hasName" => "hasName"

Operation units are also mapped to "Atom". The attributes "hasSourceEntity" and "hasName" are mapped just like in content units. In case of "ObjectOperationUnits" the atom contains an "AtomFunction" of the type "CreateObject", the "hasParameter" attributes from the operation units are mapped to the "hasParameter" attributes in this atom function. In case of an object delete the "AtomFunction" is set to a "DeleteObject" and the "hasSelector" from the operation unit is mapped to the "hasSelector" in the atom function.

Entry units are mapped to "Atom". The attributes "hasSourceEntity" and "hasName" are mapped just like in content units. The atom is associated with one or many user functions through the "hasManagementFunction" attribute. Each "hasEntryField" attribute is mapped to a user function. The name of the user function indicates on which part of object the function is used.

Global units are also mapped to an "Atom", the only attribute which is mapped is "hasSetting". This attribute is used to store the global variable.

WebML supports links between units and links between pages. In both cases their source and destination unit or page is mapped to the attributes "hasSource" and "hasLink", which are both attributes of the meta model concept "Link". The link parameters can be directly mapped to the link parameters in the meta model since they are exactly the same.

Page can be mapped to "Composite", the "hasUnit" property is mapped to "hasAtom". Area and siteview are also mapped to "Composite", they can refer to other composites with the "hasComposite" property. The "hasArea" and the "hasPage" are both mapped to this property. The boolean settings in "Area", "Siteview" and "Page" are all mapped to "hasSetting" , this property contains a name and a boolean value.

Hera

Hera uses slices to present information on webpages. Slices are mapped to the concept "Atom" in the following way:

- "hasConcept" => "hasSourceEntity"
- "hasAttribute" => "hasSelectionClause"
- "hasAppearanceCondition" => "hasAdaptation"
- "hasName" => "hasName"

A slice can contain a form, which is used to acquire data from users. Each "FormItem" is mapped to an "UserFunction", which is a specialization of "AtomFunction". Note Hera does not support "SystemFunction". The type of the form item stipulates the sort of atom function (e.g., a selection type is mapped to "Select").

- "hasHandOverItem"=> "hasName" of "UserFunction"
- "hasConcept"=> "hasSourceEntity" from "Atom"

A "NavigationalRelation" is mapped to "Link", the "hasSource" and "hasDestination" properties can be directly mapped. The property "hasCMReference" is mapped to

"hasReferenceLink", this denotes a reference to a relation in the CM. The "SetOfSlices" and "SetOfLinks" can be mapped to a collection of links in the meta model. The "CompositionalRelation" can be mapped by using the meta model concept "Composite", the "hasDestination" property is mapped to "hasAtom".

A query is a link between a form and a slice, it is used to transport information between a form and a slice. A query is mapped to a link in the meta model, the "hasForm" property is mapped to the "hasSource" property of link , the "hasDestination" property can be directly mapped. The "hasContent" property is mapped to "hasParameter" property in "Link".

SHDM

SHDM uses nodes with attributes to publish information. The concept "NodeAttribute" is mapped to "Atom". The "isIndex" and "isAnchor" properties are mapped to the "hasSetting" property in node. This is used to store boolean variables. "DerivedAttribute" and "ForeignAttribute" also contain SeRQL queries used to perform a query on the conceptual model. Many values in this query can be mapped to properties in the atom (they be directly mapped since the query is specified in SeRQL):

- The select of the query => "hasSourceAttributes"
- The from in the query => "hasSourcEntity"
- The where in the query => "hasSelectionClause"

A "Node" and "Link can be directly mapped to the meta concepts "Composite" and "Link" respectively.

SHDM also uses a navigational context model to model how information can be reached and browsed. The concept "ContextClass" is mapped to "Composite", since it can contain other context classes or context objects. The "hasSubClass" property is mapped to "hasComposite", the rest of the mapping is obvious.

A "ContextObject" is mapped to "Atom", each context object contains a context definition card, the mapping is defined as follows:

- "hasName" => "hasName"
- "hasParameter"=>"hasSelectionClause"
- "hasRestriction" =>"hasSelectionClause"
- "hasOperation" => "hasSelectionClause"
- "hasNavigation" => "hasSorting"

- "hasOrder" => "hasSorting"
- "hasElement" => "hasSourceEntity"

The "hasComment" attribute contains meta data and can be merged into the name of the atom.

The "AccessStructure" concept is mapped to "Composite" since it can contain many "AccessStructureSubClass" concepts. The "AccessStructureSubClass" and "AccessStructureFaceted" are mapped to "Atom", only the "hasName" property is mapped. A "ContextLink" can be mapped to the meta concept "Link".

WSDM

WSDM uses nodes to group information in object chunks. Each node is mapped to "Atom". WSDM does not perform queries in on the CM, instead all the information in the object chunks is used. There is no redundant information because WSDM is an audience driven method, the audience is taken as a starting point not the data. As a consequence, most of the atom properties are not set. WSDM contains a concept "RootNode" which denotes the root of the navigation hierarchy. The "hasSetting" property from atom can be used to store this boolean variable.

The mapping from "ObjectChunkFunction" to "AtomFunction" is obvious since the meta model uses the same ontology structure. A "SortingFunction" can also be directly mapped to the meta concept "SortingFunction".

The four link types defined in WSDM can all be mapped to the meta concept "Link". The different link typed only have a semantic difference.

5.2.3 Presentation Meta Model

5.2.3.1 Presentation Metal Model Description

The presentation model defines how instantiated components are presented to the user. Presentation describes layout, style, interface and behaviour of a components. Figure 5.5 and 5.6 give an overview of the presentation meta model. Table 5.6 describes a mapping between concepts from the design methods and the meta model.

5.2.3.2 Presentation Metal Model Description

Layout Concepts

The meta model specifies the layout of webpages at a very abstract level: a grid contains rows, each row contains cells. Grids can be nested to create a more complex layout. In the ontology, a cell has the following properties:

- *"hasHeight":* an integer defining the height of this cell.
- *"hasWidth":* an integer defining the width of this cell.
- *"hasInterfaceConcept":* a reference to an interface concept (e.g., audio, video, a form etc.).
- *"hasStyle":* a reference to a stylesheet.
- *"HasNavigationConcept":* a reference to a component in the navigation schema. This is the component which is displayed in the cell. It is important to notice this is an association with the navigational model.

Style Concepts

Each cell can be associated with his own style. The ontology does not provide functionality to model style properties. Instead, a reference to a stylesheet is used.

Interface Concepts

Interface concepts are used to define the actual user interface delivered to the user. Only the most abstract interface elements are provided to model a user interface. Multimedia concepts provide a way to present basic media on a webpage. Form concepts allow the modeling of simple forms uses to acquire information from users.

Behaviour Concepts

Behaviour concepts are associated with a presentation concept. The define which actions occurs on a certain event (e.g., color red when mouse over).

5.2.3.3 Presentation Metal Model Mapping

For WebML a total mapping to the meta model is not available since their presentation design is completely done in WebRatio. The stylesheet concept can be mapped which is a reference to a stylesheet. The WebRatio tool also uses a grid structure to organize layout, this can be mapped to the grid in the meta model.

Hera uses the AMACONT document model as a presentation model. AMACONT uses layout managers to structure content on webpages. These layout managers can be mapped to the grid concept. The media components can be directly mapped.

SHDM uses the abstract and concrete widget ontology. The abstract widget ontology supports forms and events, which can be mapped to the "FormConcept" and "Behaviour" respectively.

Most WSDM concepts can be directly mapped to the meta ontology. However, the meta ontology doesn't provide support for complex presentation elements.

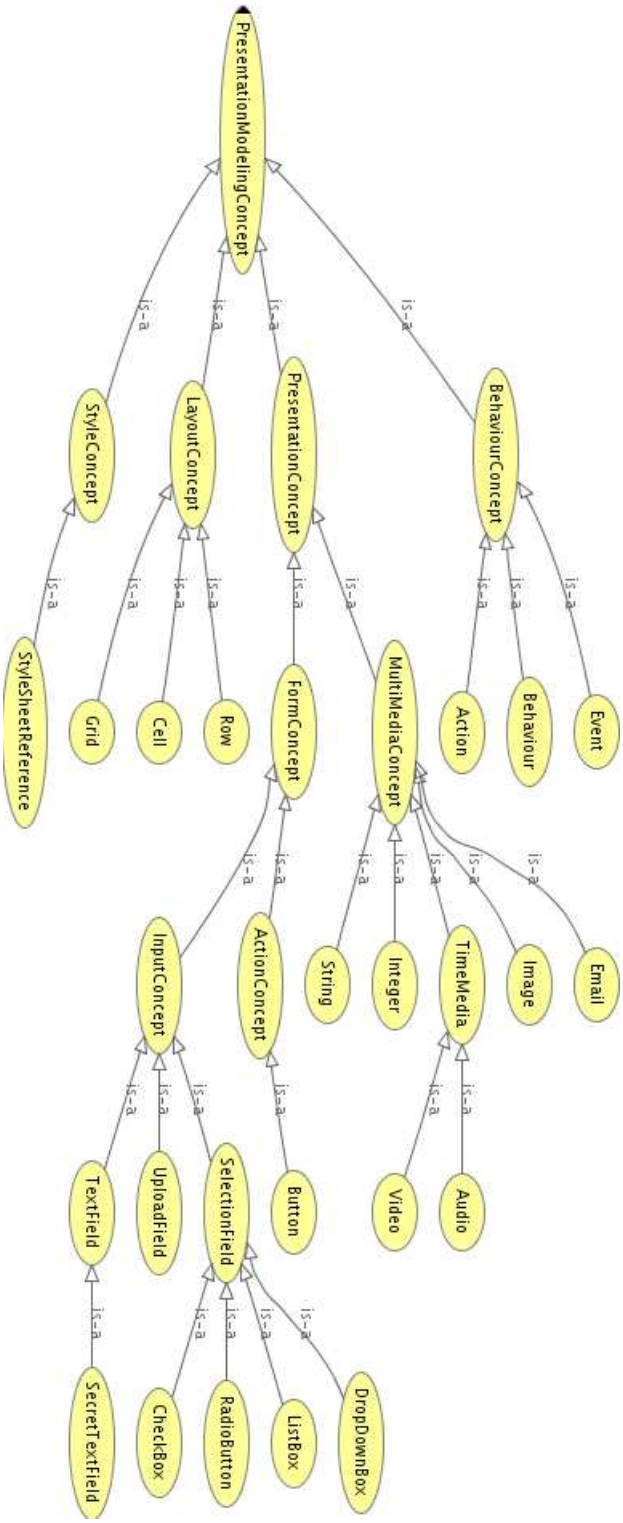

Figure 5.5: Presentational meta model hierarchy.

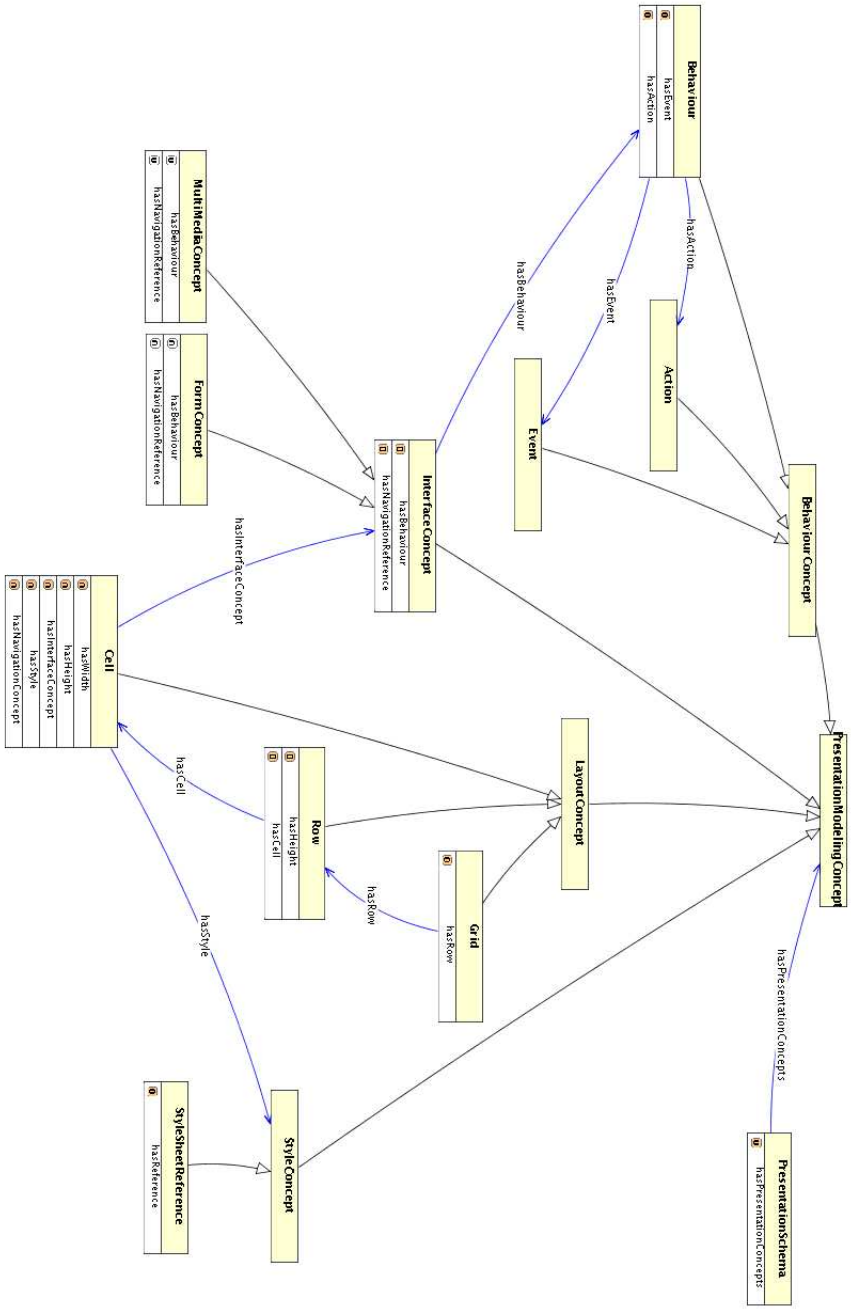

Figure 5.6: Presentation meta model.

Meta model	WebML	Hera	SHDM OOHDM	WSDM
Events with actions	WebRatio	/	Event activator in the abstract widget ontology	Events with actions
Multimedia Concepts	WebRatio	Media components	/	Multimedia concepts
Form concepts	WebRatio	/	Variable capturer in the abstract widget ontology, form concepts in the concrete widget ontology	Form concepts
Grid with row and cells	WebRatio	Layout managers from AMACONT are basically grids	Layout is left to the graphic designer	Grid with row and cells
Stylesheet reference	Stylesheet	Stylesheet	Stylesheet	Stylesheet

Table 5.6: Presentation meta model mapping.

5.3 On-line Material

The ontologies described in the previous sections can be found on-line: `http://wilma.vub.ac.be/~tvleminc/metamodels`.

Chapter 6

Conclusion

The World Wide Web(WWW) evolved from a network of linked pages into a data-intensive network. This evolution of the WWW requires a new, more systematic, kind of website design. This thesis discusses, both formally and informally, four website design methods (WebML, Hera, SHDM/OOHMD and WSDM) that support the systematic design of websites at a conceptual level. The research part of this thesis describes each method in the OWL Web ontology language. The use of an ontology as a representation language for a website design method delivers a wide range of benefits:

- **Unambiguous Semantics:** an ontology representation gives a clear and unambiguous semantic description of the models used in the method.
- **Reusability:** a formal description of a model makes it possible to reuse the model in other or future design methods.
- **Interoperability:** models of different design methods can work together or maybe extend each other.
- **Semantic Annotation:** the definition of the different models in an ontology language allows the reuse of existing domain ontologies. This way websites provide semantic annotated data.

During the construction of the ontologies the Dexter hypertext reference model was used. Concepts from each design method where related to concepts in the Dexter reference model. This relation to Dexter made it possible to create a structured overview of each method. It became clear that the methodologies where in accordance with each other:

- **Separation of concerns***:* all methods use different models for different concerns. A conceptual model models the data used on the website, a navigational model specifies navigation, content publishing and user interaction. A presentation model is used to model presentation of concepts to the user.

- **A conceptual model in a Web ontology language**: all methods, except WebML, use a Web ontology language to specify their conceptual model. This makes it easy to map between conceptual models of different methods.
- **Navigational model features:** all methods use a navigation model consisting of nodes, interconnected with links. Nodes are views over entities in the conceptual model, they are the atomic components to publish content on a website. Most navigational models also specify management of data in the conceptual schema.
- **Presentational model features:** the presentation model of each methodology allows to specify presentation of concepts to the user. Some methods, like WSDM, provide complex presentation modeling. Others, like SHDM, only provide support to model a basic user interface.

Since the design methods agreed on many points, it was possible to make abstractions of concepts and features used in each methodology. These abstractions where used to construct a *design methods meta model.* The existence of such a design method meta model offers the following benefits:

- **Interoperability:** the meta model acts as a mapping mechanism between model of different methodologies. This opens the path towards design method interoperability.
- **Standardization:** the meta model can be seen as a standard, models that are not able to map to the meta model may lack some modeling features.
- **Comparison of methodologies:** since each model can be mapped to a model in the meta model, it becomes easy to compare methods.
- **Standard terminology:** the design methods discussed in this thesis all use different terms for their concepts. The meta model introduces a standard terminology.
- **A foundation for future design methods:** the meta model can be used as a starting point for future design methods.

The design method meta model ontology which is purposed in this thesis can be used to specify each of the discussed methodologies. A informal mapping is provided between concepts in the discussed methodologies and the meta model. The model provides an abstraction layer on top of these design methods. Furthermore, the meta model can be used as a basis for future design methods.

Bibliography

[1] The Hera design method. http://wwwis.win.tue.nl/ hera/.

[2] Ontoweaver website. http://kmi.open.ac.uk/projects/akt/ontoweaver/.

[3] The OOHDM design method. http://www.oohdm.inf.puc-rio.br:8668/.

[4] OWL web ontology language use cases and requirements. http://www.w3.org/TR/webont-req/.

[5] Sesame RDF query language. http://www.openrdf.org/.

[6] The WebML design method. http://www.webml.org/.

[7] WebRatio, a case tool for WebML. http://www.webratio.com.

[8] The WSDM design method. http://wsdm.vub.ac.be.

[9] S. Casteleyn and P. Plessers. Use and advantages of ontology-based web design. 2004.

[10] S. Casteleyn, O. De Troyer, and S. Brockmans. Design time support for adaptive behaviour in web sites. In *Proceedings of the 18th ACM Symposium on Applied Computing*, pages 1222–1228, Melbourne, USA, 2003. ACM, Academic Press.

[11] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications.* Morgan Kaufmann Publishers Inc., 2002.

[12] S. Ceri, P. Fraternali, A. Bongio, and A. Maurino. Modeling data entry and operations in WebML, 2000.

[13] P. P.-S. Chen. The entity-relationship model: Toward a unified view of data. In J. Mylopoulos and M. L. Brodie, editors, *Artificial Intelligence & Databases*, pages 98–111. Kaufmann Publishers, INC., San Mateo, CA, 1989.

[14] J. Davies, D. Fensel, and F. van Hamelen. *Towards The Semantic Web - Ontology-Driven Knowledge Managemen.* Wiley, West Sussex, England, 2003.

[15] M. Dubinko, L.L. Klotz, R. Merrick, and T.V. Raman. Xforms 1.0 (w3c recommendation 14 octaber 2003), 2003.

[16] Z. Fiala, M. Hinz, K. Meißner, and F. Wehner. A component-based approach for adaptive, dynamic web documents. *J. Web Eng.*, 2(1-2):58–73, 2003.

[17] F.Â Frasincar and P. Barna. Adaptation and reuse in designing web information systems.

[18] M. R. Genesereth and N. J. Nilsson. Logical foundations of artificial intelligence, 1987.

[19] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[20] F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[21] T. Halpin. *Information modeling and relational databases: from conceptual analysis to logical design.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[22] W. Hongjing. A reference architecture for adaptive hypermedia applications, 2002.

[23] G.J. Houben. Hera: Development of semantic web information.

[24] Stanford Medical Informatics. Protege, http://protege.stanford.edu/.

[25] H. Maurer. The next generation web solution, 1996.

[26] P. Plessers, S. Casteleyn, Y. Yesilada, O. De Troyer, R. Stevens, S. Harper, and C. Goble. Accessibility: A web engineering approach. In *Proceedings of the 14th International World Wide Web Conference (WWW2005)*, pages 353–362, Chiba, Japan, 2005. ACM, Academic Press.

[27] D. Plexousakis, F. V. Vouton, G. Karvounarakis, M. Scholl, S. Alexaki, and V. Christophides. RQL: A declarative query language for RDF*, March 10 2002.

[28] G. Rossi, D. Schwabe, and F. Lyardet. Web application models are more than conceptual models. In *ER Workshops*, pages 239–253, 1999.

[29] D. Schwabe and S. Moura. Interface development for hypermedia applications in the semantic web, 2003.

[30] D. Schwabe and G. Rossi. Developing hypermedia applications using OOHDM, 1998.

[31] D. Schwabe and G. Rossi. An object oriented approach to web-based applications design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.

[32] D. Schwabe, G. Szundy, de S. Moura, and F. Lima. Design and implementation of semantic web applications. In *Proc. of the Workshop on Application Design.*

[33] M. K. Smith, C. Welty, and D. L. McGuinness. OWL web ontology language guide. Technical Report 20040210, World Wide Web Consortium, 2004.

[34] R. R. Swick and World Wide Web Consortium. Resource description framework (RDF) model and syntax, July 02 1998.

[35] O. De Troyer and S. Casteleyn. Modeling complex processes for web applications using WSDM. 2003.

[36] O. De Troyer and S. Casteleyn. Designing localized web sites. In *Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE2004)*, pages 547–558, Brisbane, Australia, 2004. Springer-Verslag.

[37] Olga De Troyer. Designing well-structured websites: Lessons to be learned from database schema methodology. In Tok Wang Ling, Sudha Ram, and Mong-Li Lee, editors, *Conceptual Modeling - ER '98, 17th International Conference on Conceptual Modeling, Singapore, November 16-19, 1998, Proceedings*, volume 1507 of *Lecture Notes in Computer Science*, pages 51–64. Springer, 1998.

[38] R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Modeling user input and hypermedia dynamics in Hera.

[39] R. Vdovjak, F. Frasincar, G. J. Houben, and P. Barna. Engineering semantic web information systems in Hera, 2003.

[40] World Wide Web Consortium (W3C). Rdf primer, 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[41] F. Zoltan, F.Flavius, M. Hinz, G. J. Houben, P. Barna, , and K. Meissner. Engineering the presentation layer of adaptable web information systems, 2004.