



Vrije Universiteit Brussel  
Faculteit Wetenschappen  
Departement Informatica & Toegepaste Informatica  
Academiejaar 2001-2002

# Ontwerpen van adaptive web sites door middel van WSDM

Saar Brockmans [sbrockma@vub.ac.be]

*Proefschrift ingediend met het oog  
op het behalen van de graad van  
Licentiaat in de Toegepaste Informatica*

Promotor: Prof. Dr. Olga De Troyer

## Samenvatting

Tegenwoordig bevatten web sites explosieve hoeveelheden informatie. De laatste jaren wordt er dan ook heel wat aandacht besteed aan het ontwerpen van web sites. Ook adaptieve web sites krijgen de nodige aandacht. Dit zijn web sites waarvan de structuur en/of de inhoud (al dan niet automatisch) aangepast wordt aan het surfgedrag van de bezoekers. Voor zover als geweten, hebben bestaande web site design methoden geen voorzieningen om reeds tijdens het ontwerp aan te geven welke aspecten of delen van een web site adaptief zijn.

Deze thesis start met een inleiding over adaptief gedrag in web sites en beschrijft daarna enkele bestaande web site design methoden. Er wordt dieper ingegaan op WSDM, de web site design methode die als framework zal gebruikt worden doorheen de thesis. Vervolgens wordt een formalisatie van het navigatieve model gegeven, om daarna een oplossing voor te stellen om adaptief gedrag te modelleren. Uiteindelijk wordt door middel van twee cases aangetoond dat de theoretische oplossing ook praktisch uitvoerbaar is.

**Keywords:** adaptieve web sites, WSDM, Navigatieve Model, regels.



Vrije Universiteit Brussel  
Faculteit Wetenschappen  
Departement Informatica & Toegepaste Informatica  
Academiejaar 2001-2002

# Designing adaptive web sites using WSDM

Saar Brockmans [sbrockma@vub.ac.be]

## **Abstract**

Nowadays, web sites contain explosive amounts of information. Therefore, the design of web sites gets a lot of attention during the last years. Also adaptive web sites are receiving attention. In adaptive web sites the structure and/or the content of the web site is adapted (automatically or manually) to the surf behavior of the visitors. However, as far as we are aware of, existing web site design methods do not support the specification of adaptive behavior during modeling.

This thesis starts with an introduction of adaptive behavior in web sites and describes some existing web site design methods. WSDM, the web site design method which will be used as framework through the thesis, is discussed in detail. Afterwards, a formalization of the navigational model is given, to propose a solution for modeling adaptive behavior afterwards. Finally, we show by means of two cases that the theoretical solution is also practical applicable.

**Keywords:** adaptive web sites, WSDM, Navigational Model, rules.

## Acknowledgements

First of all, I want to thank **Prof. Dr. Olga De Troyer**, for allowing me to do my thesis at her lab. I also want to thank her for spending so much time on advising me, solving problems we encountered and proofreading my thesis several times. It was a nice experience to work together.

Secondly, I'm very thankful to **Sven Casteleyn**, who helped me from the beginning with my work. He thought about problems and solutions, gave me a lot of advice and also gave me lots of suggestions after proofreading draft versions of the thesis. He did a great job to me so, many thanks Sven!

For proofreading drafts and for supporting me during the whole period, I would like to thank my boyfriend **Wim**. He meant much to me this year.

A lot of gratitude goes to **my parents**, who gave me the opportunity to study and educated me to the person that I am now.

I also want to thank my sister and brother, **Sofie** and **Thomas**, who supported me in several ways.

Finally, I want to thank all my friends and family who supported me during my years at the VUB. A special thanks goes out to all members of **Muzemix**, for the lovely moments.

## *Table of contents*

<b>Introduction</b>	<b>5</b>
<b>I Background</b>	<b>6</b>
<b>1 Adaptive behavior in web sites</b>	<b>7</b>
1.1 What is adaptive behavior?.....	8
1.2 Classification.....	8
1.3 Difference between 'adaptive', 'adaptable' and 'dynamic'.....	9
<b>2 Summary of existing web site design methods</b>	<b>10</b>
2.1 Object-Oriented Hypermedia Design Method (OOHDM).....	10
2.2 Web Modeling Language (WebML).....	13
2.3 Relationship Management Methodology (RMM).....	14
2.4 Object-Oriented Hypermedia Method (OO-H).....	18
2.5 Other existing methods.....	19
<b>3 Web Site Design Method (WSDM)</b>	<b>20</b>
3.1 Characteristics.....	20
3.2 An Overview.....	21
3.2.1 Mission Statement Specification.....	22
3.2.2 Audience Modeling.....	22
3.2.3 Conceptual Design.....	24
3.2.4 Implementation Design.....	27
3.2.5 Implementation.....	27
3.3 Conclusion.....	27
<b>II Research</b>	<b>28</b>
<b>4 Problem statement: design time support for adaptive behavior in web sites</b>	<b>29</b>
<b>5 Suggestion to design adaptive behavior</b>	<b>30</b>
5.1 Definition of the Navigational Model.....	30
5.1.1 The Navigational Model.....	30
5.1.2 Audience Tracks.....	32
5.2 Operations on the Navigational Model.....	34
5.2.1 Operations on nodes.....	35
5.2.2 Operations on chunks.....	35
5.2.3 Operations on links.....	36
5.3 Rules to design adaptive behavior.....	36
5.3.1 Transformations.....	36

5.3.1.1	Promotion and demotion.....	37
5.3.1.2	Linking and unlinking.....	37
5.3.1.3	Clustering.....	38
5.3.2	Syntax for rules.....	39
5.3.2.1	Some syntax-elements.....	39
5.3.2.2	The rule-syntax.....	40
5.4	Future work.....	42
<b>6</b>	<b>Cases</b>	<b>43</b>
6.1	Case 1: Youth organization 'Muzemix'.....	43
6.1.1	Mission Statement Specification.....	43
6.1.2	Audience Modeling.....	44
6.1.2.1	Audience Classification.....	44
6.1.2.2	Audience Class Characterization.....	47
6.1.3	Conceptual Design.....	48
6.1.3.1	Information Modeling.....	48
6.1.3.2	Functional Modeling.....	49
6.1.3.3	Navigational Design.....	49
6.1.4	Implementation Design.....	54
6.1.4.1	Page Design.....	54
6.1.5	Implementation.....	54
6.1.6	Conclusion.....	55
6.2	Case 2: Online Bookstore bekend.....	55
6.2.1	Mission Statement Specification.....	55
6.2.2	Audience Modeling.....	55
6.2.2.1	Audience Classification.....	55
6.2.2.2	Audience Class Characterization.....	57
6.2.3	Conceptual Design.....	58
6.2.3.1	Information Modeling.....	58
6.2.3.2	Functional Modeling.....	59
6.2.3.3	Navigational Design.....	60
6.2.4	Implementation Design.....	63
6.2.4.1	Page Design.....	63
6.2.5	Implementation.....	63
6.2.6	Conclusion.....	64
<b>7</b>	<b>Conclusion</b>	<b>65</b>

## List of figures

1.1	WWW Growth.....	7
2.1	Summary of the OOHDM methodology.....	12
2.2	The RMM methodology within the context of the complete software development cycle.....	15
2.3	The RMDM primitives.....	16
2.4	OO-H method: the design process.....	18
3.1	WSDM: Overview.....	21
3.2	Audience Class Hierarchy for the Bookstore example.....	24
3.3	Book-Data Model for the Bookstore example.....	25
3.4	Functional Model 'submit books' for the Bookstore example.....	26
3.5	Part of the Author Track of the Bookstore example.....	26
6.1	Audience Class Hierarchy for the Muzemix example.....	47
6.2	Informational Chunk for Presentation.....	48
6.3	Functional Chunk for Submit Presentation Info.....	49
6.4	Main Track Structure of the Navigational Model.....	50
6.5	Navigation Track for Non Member.....	51
6.6	Page structure for the Non Member Track.....	54
6.7	Audience Class Hierarchy for the Online Bookstore example.....	57
6.8	Informational Chunk for Book.....	59
6.9	Functional Chunk for Add information about a book.....	60
6.10	Main Track Structure of the Navigation Model.....	60
6.11	Navigation Track for Reader.....	61
6.12	Page structure for the Reader Track.....	63



## Abbreviations

APD	Abstract Presentation Diagram [6, 7]
BNF	Backus Naur Form [34]
ER	Entity-Relationship [25]
HTML	HyperText Markup Language [27]
NAD	Navigation Access Diagram [6, 7]
OMT	Object Modeling Technique [23]
OO-H	Object-Oriented Hypermedia Method [6, 7]
OOHDM	Object-Oriented Hypermedia Design Method [1, 2, 3, 17, 18]
OQL	Object Query Language [24]
ORM	Object Role Modeling [21]
RMDM	Relationship Management Data Model [5]
RMM	Relationship Management Methodology [5]
UML	Unified Modeling Language [20]
WAP	Wireless Application Protocol [28]
WebML	Web Modeling Language [4]
WSDM	Web Site Design Method [13, 14, 15, 16]
WWW	World Wide Web [22]
XML	Extensible Markup Language [26]

## Introduction

This thesis consist of two parts: the background part and the research part.

### ***Background***

The first part of the thesis provides the information that is needed to understand the research part. Firstly, we discuss adaptive behavior in web sites. Next, the most important existing web site design methods are described, including their support for adaptive behavior. Finally, an extensive depiction of WSDM is provided.

### ***Research***

The second half of the thesis, the research part, describes the research that has been done. First we discuss the problem, which is that existing web site design methods, and in particular WSDM, don't provide any support for adaptive behavior in web sites. Consequently, a solution to this problem in the context of WSDM is proposed: representing adaptive behaviour using rules with conditions and operations. To be able to do this, a formal definition of the Navigational Model of WSDM is given. Finally, the proposed solution is illustrated by two cases.

# Part I

## Background

## 1. Adaptive behavior in web sites

The World Wide Web is becoming a very popular medium for sharing data. Many companies and organizations, even individuals have found interest in delivering information through the internet. Companies, for example, currently use the World Wide Web for fast data exchange, necessary when becoming international. Like this, everyone has its own reason to use the World Wide Web.

Figure 1.1 [29] shows the growth of interest in the WWW.

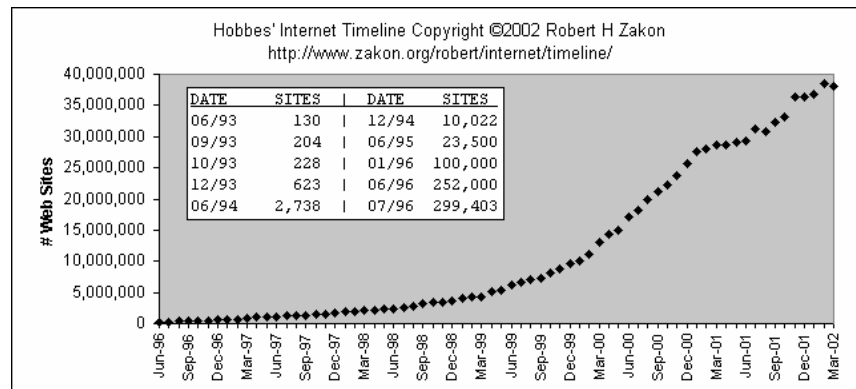


Figure 1.1. WWW Growth.

Considering this growth of interest in the Web, web sites now contain many valuable pieces of information. Explosive amounts of information is presented in web sites nowadays. Thus, to structure and maintain the sites properly, great attention needs to be devoted to the design. In fact, designing web sites becomes a complex process that involves multiple aspects: data management, hypertext structure, etc. Web site design methods exist to help us properly integrating these different aspects.

But while designing web sites, there are several other elements that must also be taken into consideration. The first item we observe is that not every visitor has the same goal. Also, the same visitor may be looking for different information at different times. Secondly, a lot of sites change in time from their original design. Links and pages are accumulating in unlikely places and a site may be used in many different ways, while originally designed for a particular kind of use.

These points are addressed by adaptive web sites [8, 9, 11], which are discussed in this chapter. It is self-evident that web sites containing adaptive elements like described above, need to be designed and maintained in a more expressive way. Currently, as far as we are aware of, there are

no design methods described in the literature that provide abstract design mechanisms to automate and model adaptive behavior.

## 1.1 What is adaptive behavior?

As we already mentioned in the introduction, several issues arise when designing web sites. Every web site is made for a different organization (or individual), with different goals and objectives in mind. Similarly, various kinds of visitors frequent the site, all with different needs, different characteristics and different goals. Changes in the web site affect those visitors, and are very hard to maintain and keep track of by the web master. Some sites evolve in their functionality offered, while others adapt their motives and objectives, leading to various changes and updates. These changes range from small content updates<sup>1</sup> to more substantial, structural changes. The latter can be made by redesigning the site (manually), or they could be automatically done by observing certain parameters relevant to the (structure of the) web site.

Adaptive web sites address exactly this issue:

*Adaptive web sites are sites that automatically improve their organization and presentation by learning from visitor access patterns<sup>2</sup>.*

This automatic reorganization of web sites is called *adaptive behavior*. Several tactics can be applied to realize adaptive behavior: changing presentation (e.g. highlighting interesting links, changing text or link appearance, which typically include changing labels, link color, text size, ...), offering information better tailored to the particular visitor (e.g. connecting related pages, clustering similar pages together) or altering the navigation (e.g. moving, adding or removing links).

## 1.2 Classification

Adaptive behavior can be classified in two dimensions:

- Customization and transformation
- Content-based adaptation and Access based adaptation

We explain each dimension below.

---

<sup>1</sup> This includes both deleting or changing existing information, and providing additional information

<sup>2</sup> web server logs automatically record user behaviour at the site. These recorded information form visitor access patterns.

- *Customization – transformation*
  - *Customization* is adapting a web site to suit an individual user; the page returned by the site depends on knowledge (gathered in some way) about those individuals (for example, My Yahoo!)
  - *Transformation* is modifying a web site to facilitate navigation for a group of users (e.g. move a link)
- *Content-based – access-based*
  - *Content-based adaptation* is adapting a web site based on the contents of the pages (e.g. connect pages with the 'same' content)
  - *Access-based adaptation* is adapting a web site based on the way users interact with the site (e.g. links to a frequently asked page are put in the homepage of the site)

These two classifications are orthogonal. This means a site can use both customization or transformation and content-based or access-based adaptation. While customization and transformation are mutually exclusive, content-based and access-based are complementary: they can be combined in one adaptive site.

### 1.3 Difference between 'adaptive', 'adaptable' and 'dynamic'

Before continuing, it might be useful to note there is some confusion about the exact meaning of the terms *adaptive*, *adaptable* and *dynamic* when speaking of web sites, leading to frequent misconceptions.

*Adaptable* sites *don't change* a user profile *automatically*, but the user is able to change it himself, after receiving hints from the system. *Adaptive* web sites do update user profiles *automatically*, by observing the user's browsing behavior. So, while an adaptive web site may be adaptable, the opposite is (by definition) not the case.

*Dynamic* sites are constructed to include content that changes *from time to time*. This allows the site to display things such as the current time or the latest news from your favorite portal. An *adaptive* site will have changing items *while someone is browsing it*.

## 2. Summary of existing web site design methods

---

In this chapter, we discuss some existing design methods for web sites. For each method, the approach and the different phases are described. This thesis is about support of adaptive behavior in a design method for web sites, hence we shortly discuss the support of each method.

### 2.1 Object-Oriented Hypermedia Design Model (OOHDM)

The OOHDM Method, as discussed in [1, 2, 3, 17, 18], uses an object-oriented approach for building hypermedia applications. It comprises 4 steps namely conceptual design, navigational design, abstract interface design and implementation. They are performed in a mix of incremental and iterative development styles. Each step concerns a certain part of the design and builds an object-oriented model, or enriches a model from previous iterations. The clear separation of the three design-phases makes the designs more modular and reusable. Besides, the interface design primitives can easily be mapped onto non-object-oriented implementation languages or environments (like HTML). So, OOHDM can be applied, no matter which target system is used.

In the next paragraphs, we describe each step in detail.

#### *Conceptual Design*

In this phase a model of the application domain is built using well-known object-oriented modeling principles with a notation similar to UML. The model consists of classes, subsystems and relationships, which are built using aggregation and generalization/specialization hierarchies. The most important in this phase is that the domain semantics must be taken as neutral as possible; users and tasks are not yet to be considered.

OOHDM doesn't demand to use a particular method to produce the conceptual class schema; any of the well-known methodologies (OMT, UML) may be employed.

#### *Navigational Design*

OOHDM sees an application as a navigational view over the conceptual model. During navigational design, user profiles and tasks are taken into account. With one conceptual schema, different navigational models are possible; each navigational model corresponds with another view or application on the same domain.

The navigational structure of a hypermedia application is defined by a schema specifying navigational classes. OOHDM includes a number of predefined types of navigational classes: nodes, links and access structures.

The semantics of nodes and links are the same as we are used to in hypermedia applications, while access structures represent alternative ways to access nodes, e.g. indices, guided tours, etc. These navigational classes are organized in something called a navigational context. An example of a navigational context is a n-to-m link that permits sequentially passing all link targets. Navigational contexts can be nested. Navigational contexts and classes take into account the types of intended users and their tasks. A node represents the view on a conceptual class, like defined by the conceptual design.

Navigational semantics are defined in terms of nodes and links; in this way, we can represent movement in the navigational space, i.e. the nodes that the users can reach on a certain moment.

Because the navigational design is independent of the conceptual model, the navigational model can evolve independently of the conceptual one, which simplifies maintenance.

#### *Abstract Interface Design*

Once we have a navigational design, an abstract interface model is built. During abstract interface design, the interface objects which the user will perceive are defined: how will the navigational objects look like, which interface objects will activate navigation, etc.

The clear distinction between navigational and abstract interface design leads to a higher degree of independence from user-interface technology: different interfaces may be build for the same navigational model.

OOHDM uses Abstract Data Views to describe the user interface. Abstract Data Views are formal models for interface objects (like a picture or a city map) in terms of interface classes. They show the way in which the interface objects are structured, the way in which they are statically related to navigational objects and how they respond to external events. Interface classes are defined as aggregations of primitive classes, like buttons and text fields, and recursively of interface classes.

#### *Implementation*

During implementation, the navigational and abstract interface models are transformed to concrete objects, available in the chosen implementation environment. The uniform modeling constructs, objects and classes, used in the OOHDM method, allow a fluent transition of the domain modeling to navigational and interface design.

It is not necessary to use object-oriented mechanisms in the implementation-phase, but it will make it easier. Anyway, there are lots of techniques to map an object-oriented specification on a non object-oriented runtime environment.



### Support for adaptive elements

Much effort has been spend for some years on personalization and customization using OOHDM, but OOHDM has not yet any support for adaptive behavior in web sites.

Figure 2.1 shows an overview of the OOHDM methodology.

Activities	Formalisms	Mechanisms	Design Concerns	Products
Conceptual Modeling	Object-Oriented Modeling constructs (classes, relationships, use cases)	Classification, aggregation, generalization and specialization	Model the semantics of the application domain	Classes, subsystems, relationships, attribute perspectives
Navigational Design	Object-Oriented Views, Object-Oriented State charts, Context Classes, User centered Scenarios, Design Patterns	Classification, aggregation, generalization and specialization	User's profile and task. Emphasis on cognitive aspects.	Nodes, links, access structures, navigational contexts, navigational transformations
Abstract Interface Design	Abstract Data Views, Configuration Diagrams, ADV-Charts, Design Patterns	Mapping between navigation and perceptible objects. Composition and generalization/ Specialization	Model perceptible objects, implementing those metaphors. Describe interface for navigational objects, define lay-out of interface objects	Abstract interface objects, responses to external events, interface transformations
Implementation	Those supported by the target environment	Those provided by the target environment	Performance, completeness	Running application

Figure 2.1. Summary of the OOHDM methodology.

## 2.2 Web Modeling Language (WebML)

WebML, as also demonstrated in [4], provides a model-driven approach for specifying websites on conceptual level, with both a graphical notation and a textual XML-syntax.

The specification of a site in WebML consists of four perspectives:

- The structural model: concerning data content
- The hypertext model: concerning the pages and the links between pages
- The presentation model: concerning the layout and graphical requirements
- The personalization model: concerning the customization features for one-to-one content delivery

These models are explained in the next section.

### *Structural model*

This model describes the data content of the website, in terms of relevant entities and relationships. Classical notations such as the E/R model [referentie], the ODMG object-oriented model [referentie] and UML class diagrams [referentie] can be used for the data modeling. Calculated information can be expressed using a simplified OQL-like query language.

### *Hypertext model*

The hypertext model formulates the hypertexts that correspond with the different site views. Site view descriptions in turn have two sub models: the composition model and the navigation model. The composition model contains the pages of the hypertext and the content units of the pages, while the navigation model expresses the linking between pages and content units.

### *Presentation model*

In this phase, the layout and graphical look of pages are formulated by means of an abstract XML syntax. Presentation specifications are page specific or generic:

- *page-specific* specifications give the presentation of a specific page and include explicit references to page content (e.g., they dictate the layout and the graphical appearance of the title)
- *generic* presentation specifications are based on predefined models independent of the specific content on the page and include references to generic content elements (for instance, they dictate

the layout and graphic appearance of all attributes of a generic object included in the page).

#### *Personalization model*

The personalization model gives a model of the users and user groups, specified in terms of predefined entities called User and Group. The properties of these entities permit for saving group-specific or individual content (e.g. shopping suggestions or list of favorites).

Then, OQL-like declarative expressions can be added to the structure schema, to define derived content, based on the profile data in the User and Group entities. With this personalized content, units can be composed and presentation specifications can be defined.

With WebML, high-level business rules, written in simple XML syntax, can also be defined to handle site-related events like user clicks and content updates. This way, new user-related info (e.g. shopping histories) is formed and site content is updated (e.g. new offers matching users' preferences).

#### *Support for adaptive elements*

WebML is a general-purpose design method and doesn't include specific construct for adaptive behavior.

### **2.3 Relationship Management Methodology (RMM)**

RMM [5] is a methodology to design and construct hypermedia applications and has seven steps: E-R design, Entity design, Navigational design, Conversion protocol design, User-interface design, Run-time behavior design and Construction and testing. Figure 2.2 shows the RMM methodology within the context of the complete software development cycle.

The cornerstone of the methodology is the RMDM data model, like explained below. After that, the seven phases are described in detail.

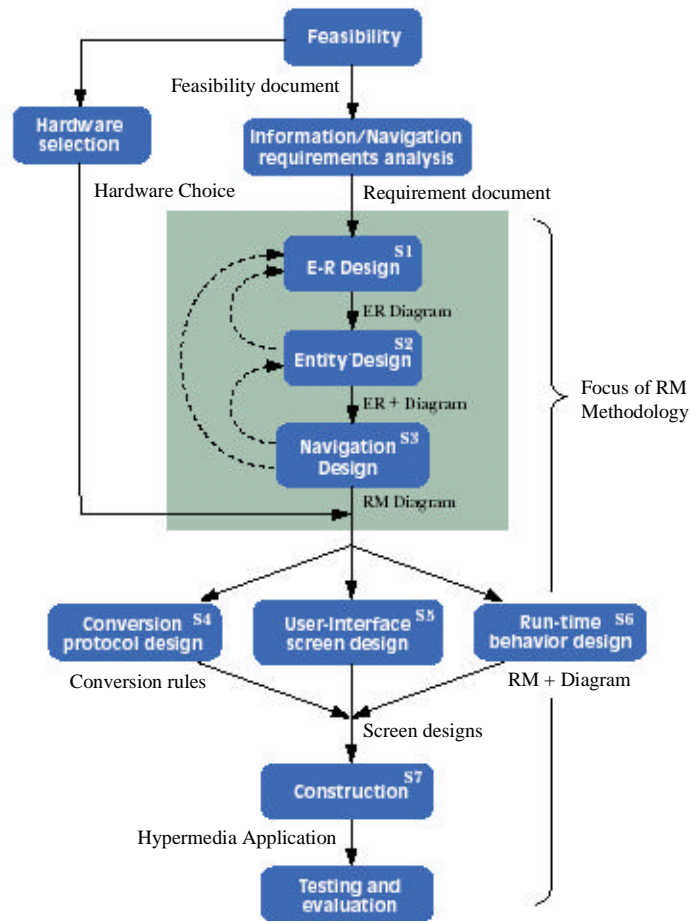


Figure 2.2. The RMM methodology within the context of the complete software development cycle.

#### *Relationship Management Data Model (RMDM)*

A data model is a set of logical objects, necessary to express an application's design. Like this, RMDM provides a language for describing the information objects and the navigation mechanisms in hypermedia applications.

Figure 2.3 shows RMDM's modeling primitives. Some information to the figure: We can see that RMM has the notion of slices. Because entities can have a lot of attributes of different natures (e.g. biographical data or salary

information), it may be impractical to represent all attributes of an entity at once. Therefore, attributes are grouped into slices. For example, a person entity may have a general slice containing name, age and photograph, and a biography slice with name and biography.

An index acts as a table of contents to a list of entity instances, providing direct access to each listed item.

A guided tour implements a linear path through a collection of items, allowing the user to move either forward or backward on the path.

The conditions qualifying indices and guided tours determine which instances of an entity are accessible from the construct.

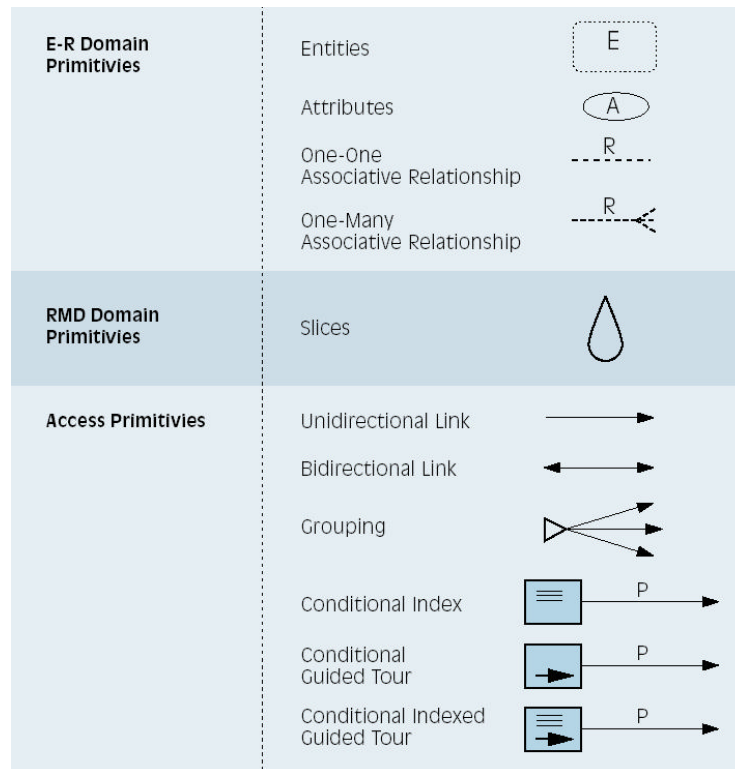


Figure 2.3. The RMDM primitives.

### *E-R Design*

This phase constructs an Entity-Relationship (E-R) diagram, which illustrates the information domain of the application. The entities and relationships form the basis of a hypermedia application and most of them become nodes and links in it.

Sometimes, the E-R diagram already exists; if this is the case, this step can be skipped.

#### *Entity Design*

The entity design phase concentrates on issues like how is the information of the entities presented to the user and how they are accessed. Entities are split into slices, which are put in a hypertext network.

The output of this step is an E-R+ model, an E-R diagram enlarged with a slice design diagram for each entity.

#### *Navigational Design*

This model provides the possible navigation paths between entities; each associative relation in the E-R+ diagram is analyzed.

RMM is a methodology for domains that are relatively frequently updated, so all navigational paths are expressed in generic terms. This means that there are no hard coded links between entity instances; links are specified by referring to properties of entities and relationships. There are three navigational elements making this possible: conditional indices, conditional guided tours and conditional indexed guided tours, like described in the session on RMDM.

#### *Conversion Protocol Design*

In this step, each RMDM-object is transformed into an object on the target platform using conversion rules, e.g. a Tool book list-box or an HTML form can be used to implement an index. For the moment, programmers do the conversion manually.

#### *User-Interface Design*

The product of the user-interface design is a screen layout for every object appearing in the RMDM diagram obtained in the navigational design. This includes buttons layouts, location of navigational aids and appearance of nodes or indices.

#### *Run-time behavior design*

During this step, decisions about how to implement link traversal, history, backtracking and navigational mechanisms are made. While designing the run-time behavior, the volatility and the size of the domain are also considered, to decide whether node contents and link endpoints are to be

built during application development or dynamically computed on demand at runtime.

#### *Construction and testing*

This step consists of construction and testing as in traditional software engineering projects. Extra attention is made to test all navigational paths.

#### *Support for adaptive elements*

The RMM method has no support for designing adaptive elements in web sites. As far as we are aware of, no concrete steps have been undertaken in this direction.

## 2.4 Object-Oriented Hypermedia Method (OO-H)

The OO-H method design process, as also demonstrated in [6, 7], departs from an UML-based class diagram of the domain information structure and offers Navigation Access Diagrams (NAD) and Abstract Presentation Diagrams (APD). A NAD defines a navigation view, while an APD gathers the concepts related to presentation.

Figure 2.3 presents an overview of the design process of the OO-H method. We consider the NAD and the APD in detail below.

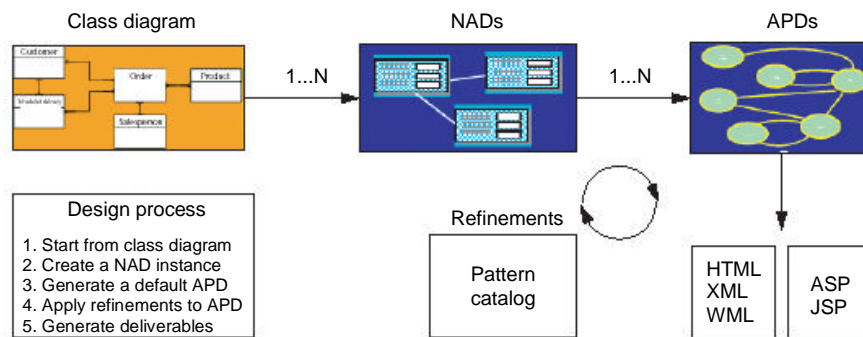


Figure 2.4. OO-H method: the design process.

#### *Navigation Access Diagram (NAD)*

Each user type corresponds with a view of the system. Starting from the class diagram, a NAD instance is built for each user type. A NAD shows information, services and needed navigation paths for the navigational requirements of the associative user. NADs are based on four construct types:

- *navigation classes* are enriched domain classes: the visibility of the attributes and methods are limited according to the access permissions and navigation requirements of the associative user.
- *navigational targets* group the elements of the model that collaborate in the coverage of each user navigational requirement.
- *navigational links* present the paths a user can follow in the system, making use of four different types of links.
- *collections* are possibly hierarchical structures defined on navigation classes or navigational targets; they give the user new ways to access information.

#### *Abstract Presentation Diagram (APD)*

After building the navigation access diagrams, the abstract presentation diagrams are constructed. An APD specifies the visual appearance and page structure of the system. Per NAD, one or more APDs can be built, each representing another view of the associative navigational requirements. The structure of the APDs can be directly extracted from the NADs, using the pattern catalog.

#### *Support for adaptive elements*

OO-H has no full support for adaptive behavior yet, but they are working on it. Actually, the support is limited to customization (namely different interface models for different actors in the system, and a set of patterns to allow the user to introduce her preferences on certain attributes of the domain model).

The developers are creating a framework for supporting adaptive behavior.

## **2.5 Other existing methods**

Other design methods exist (e.g. Strudel, Tiramisu or ADM), but they are either outdated or they do not provide any support for adaptive behavior. Hence they are not included in this summary. For more detail about these methods, we refer the interested reader to [30, 31, 32].



### 3. Web Site Design Method (WSDM)

---

After considering other design methods, we now give description of WSDM [12, 13, 14, 15, 16]. Because WSDM is the methodology on which the research is based, this description is more extensive than the sections of chapter 2. First, we discuss the most characteristic elements of the method and after that, we describe all WSDM phases in detail.

#### 3.1 Characteristics

##### *Audience driven*

WSDM has an audience-driven approach in contrast to most other web site design methods, which are data driven. The audience-driven approach takes the requirements of the users of the web site as a starting point and uses this as basis for the structuring of data afterwards. Data-driven methods take the data of the organization as a starting point: they consider how to structure the web site based on the data.

##### *Audience classes*

A website has different 'kinds' of users, e.g. for the web site of a university we may distinguish the following users: professors, students, future students. Different users have different requirements, sometimes even different presentational needs, e.g. children will appreciate a flashy presentation, while bank directors maybe prefer a sober and clear interface. WSDM classifies the users of a web site into audience classes: each audience class has its own requirements and characteristics. This is reflected in the context (not all info for all users) and in the interface (language, jargon, look-and-feel, etc.).

The division into audience classes has as an advantage that the site is more adapted to each user's needs, which enhances the usability and user satisfaction.

##### *Explicit conceptual design phase*

The conceptual design, free of implementation details, is explicitly separated from the implementation design, like using a specific implementation language, grouping into pages, use of menus, etc. Hence we are no longer dependent of the current technology (e.g. HTML). Moreover, it is possible to offer different presentations on different devices, e.g. Palm Pilot, PC, WAP, etc.

### 3.2 An Overview

Figure 4.1 gives an overview of the WSDM method. The first step is to specify the *Mission Statement*. The Mission Statement Specification expresses the purpose and subject of the web site, and declares the target audience.

Based on the Mission Statement, the *Audience Modeling* is performed, in two steps: Audience Classification and Characterization. During the *Audience Classification* phase, the different types of users are identified, while *Audience Characterization* consists of defining the characteristics of the members of the different Audience Classes.

Next, we perform *Conceptual Design*, in three steps: Information, Functional and Navigational Modeling. During *Information Modeling* we observe what kind of information is apparent, while in the *Functional Modeling* phase we observe the functionality. The *Navigation Design* phase, we consider the global navigation through the information and the functionality.

The next phase, *Implementation Design*, performs *Page, Presentation and Logical Database Design*. During these phases, grouping in pages, specifying the look and feel, and designing the database are done.

After all design phases done, the web site can be implemented.

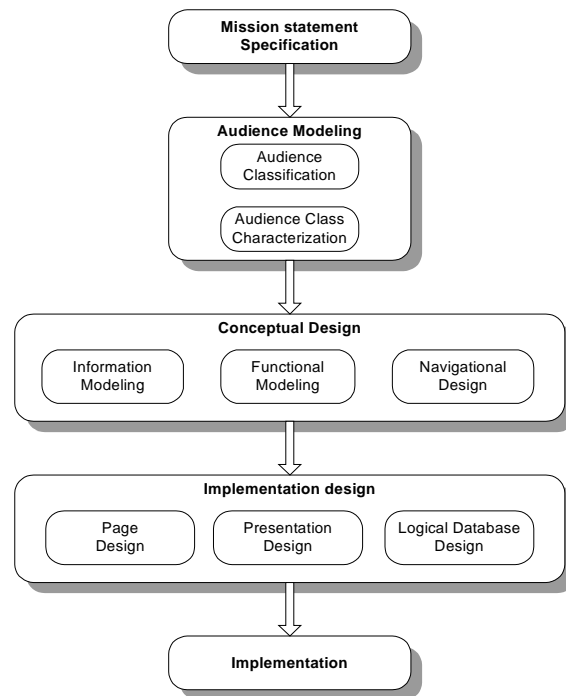


Figure 3.1. WSDM: Overview.

We discuss all mentioned phases in detail below. As illustration of the different steps, we use the following example:

*“Provide an online bookstore where users can browse a big bibliography, get information about books or authors and buy books. Authors must be able to add or delete information about his own books.”*

### 3.2.1 Mission Statement Specification

While the Mission Statement Specification, the purpose and subject of the web site are considered, and the target audience is declared.

#### Bookstore example

- *Purpose*
  - *Offer information of books and authors online*
  - *Offer books online to all interested people*
- *Subject*
  - *Books*
- *Target audience*
  - *Authors*
  - *Everyone who wants information about books*
  - *People who wants to buy books online (private & bookshops)*

### 3.2.2 Audience Modeling

As mentioned earlier, the phase Audience Modeling consists of two phases, Audience Classification and Audience Class Characterization.

#### **Audience Classification**

During this phase, different types of users are identified. Users with the same information- and functional requirements, form an Audience Class. Audience Classes need not be disjoint, so one user can belong to several Audience Classes.

#### Bookstore example

*Target audience: Authors and everyone who wants information about books or wants to buy books online*



#### *Audience Classes*

- *Author*
- *Reader*
- *Bookshop-keeper*
- *Private-buyer*

*Audience Class Author*  
*Information Requirements:*  
 - information about their own books  
 - information about him self

*Functional Requirements:*  
 - submit a book  
 - add information about a book  
 - delete information about a book  
 - add information about him self  
 - delete information about him self

*Usability Requirements:*  
 - flexible way to submit a book  
 - flexible ways to add or delete information about books and authors

*Audience Class Bookshop-keeper*  
*Information Requirements:*  
 - information about books  
 - information about authors

*Functional Requirements:*  
 - search (by ISBN, title, author, etc) for books  
 - buy large quantities of books online  
 - get specialized information about books

*Usability Requirements:*  
 - flexible ways to search for books  
 - flexible ways to buy books online

*Audience Class Reader*  
*Information Requirements:*  
 - information about books  
 - information about authors

*Functional Requirements:*  
 - get information about books  
 - get information about authors

*Usability Requirements:*  
 - flexible ways to search for information about books and authors

*Audience Class Private-buyer*  
*Information Requirements:*  
 - information about books  
 - information about authors

*Functional Requirements:*  
 - search (by ISBN, title, author, etc) for books  
 - buy a book online online

*Usability Requirements:*  
 - flexible ways to search for books  
 - flexible ways to buy books online

In our example, the Audience Classes *Bookshop-keepers* and *Private-buyers* have some requirements in common. WSDM has a feature to support this: subclasses. The members of a subclass have all the requirements of the members of the super class, plus additional requirements. So *Bookshop-keepers* and *Private-buyers* are both a subclass of the same superclass. We call this superclass *Buyer*.

With all Audience Classes together, we can form an audience class hierarchy, which shows all Audience Classes in terms of sub- and superclasses. The class on the top is always the *Visitor*-class. Visitor groups all requirements common to all audience classes.

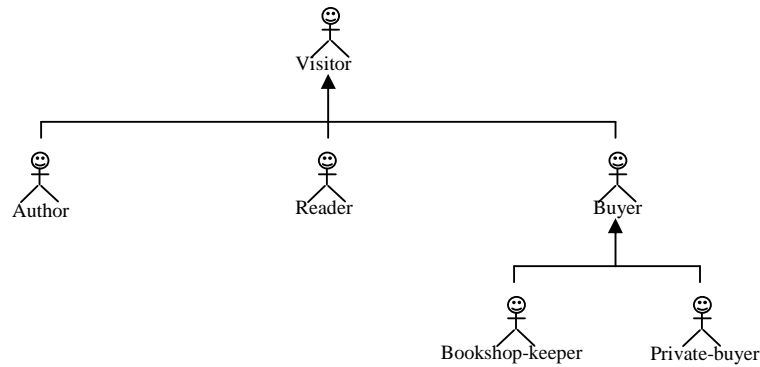
Bookstore example

Figure 3.2. Audience Class Hierarchy for the Bookstore example

**Audience Class Characterization**

After specifying the separate audience classes with their requirements, the characteristics of the members of the different Audience Classes are defined.

Bookstore example

We illustrate the characteristics of the Audience Classes Author, Reader, Bookshop-keeper and Private-buyer.

*Audience Class Author*  
 Characteristics:  
 - mostly adults  
 - experience with WWW may vary  
 - all languages possible

*Audience Class Bookshop-keeper*  
 Characteristics:  
 - all ages  
 - mostly experienced with WWW  
 - all languages possible

*Audience Class Reader*  
 Characteristics:  
 - all ages  
 - mostly experienced with WWW  
 - all languages possible

*Audience Class Private-buyer*  
 Characteristics:  
 - all ages  
 - mostly experienced with WWW  
 - all languages possible

**3.2.3 Conceptual Design**

The conceptual design is divided into three phases: Information Modeling, Functional Modeling and Navigation Design. The three phases are discussed below. Since the specifications of these models are really large, we just give a fragment from the entire picture.

## Information Modeling

During Information Modeling, the information requirements of the different Audience Classes are modeled. Currently, ORM (with extensions) is used to model the information in WSDM, but other techniques like ER, UML, etc can be used too. Information requirements are elaborated into elementary information requirements, for which we make an Object Chunk.

### Bookstore example

Figure 3.3 shows the information model for the information requirement 'information about books'.

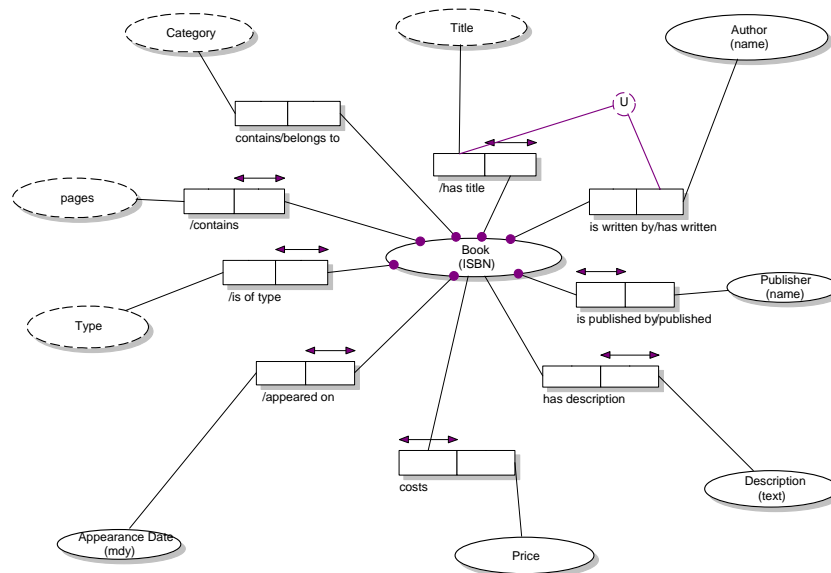


Figure 3.3. Book-Data Model for the Bookstore example.

## Functional Modeling

In this phase, the functionality for the different Audience Classes is defined. The functional requirements are, as well as the information requirements, elaborated into elementary requirements, and for each elementary requirement we make a Functional Chunk using ORM with extensions. Audience subclasses inherit the Object Chunks of their audience super classes.

Bookstore example

We picture the functional model for the functional requirement 'submit books' in figure 3.4.

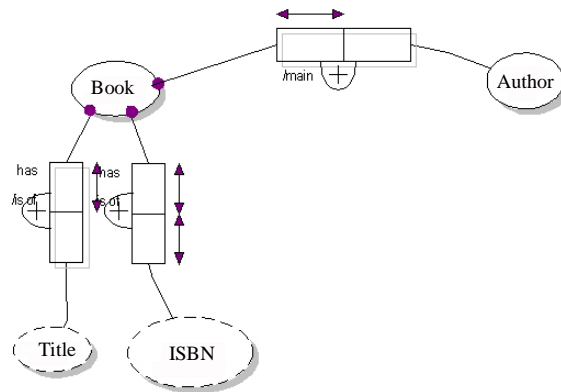


Figure 3.4. Functional Model 'submit books' for the Bookstore example.

**Navigational Design**

The Navigational Design models the navigation possibilities of the web site. For each audience class, a navigation track is defined. Tracks are composed of components and links. Components represent bundles of information or functionality, while links connects components.

A Track is a connected directed graph and needs to fulfill all information, functional and navigational requirements of its audience class. For each requirement, following the decomposition structure of the requirements, a component is defined.

Bookstore example

Figure 3.5 shows part of the Navigational Model for the Author Track.

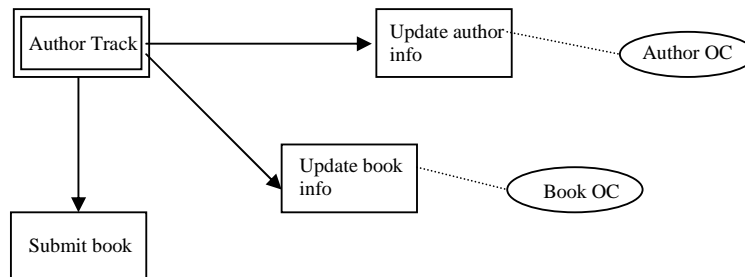


Figure 3.5. Part of the Author Track of the Bookstore example.

### 3.2.4 Implementation Design

The last phase of the design is the Implementation Design, consisting of three phases, Page Design, Presentation Design and Database Design. The Implementation Design phase is not described in detail in the literature yet, hence our discussion is limited too. Because the Implementation Designs are extremely large, we just give a part of the models.

#### Page Design

The Page Design groups information in pages. The page structure is derived from the Navigational Model. For the components and links in the navigational model we define pages and hyperlinks. The web developer can decide to cluster components connected by links and to represent the information on a single page, or he can decide to distribute the information of a single component onto different pages.

#### Presentation Design

During the Presentation Design, the look and feel is specified and the page layout is defined. Different navigation tracks may have different styles. This step of the method is not yet elaborated, hence we don't illustrate with an example.

#### Logical Database Design

If the web site needs a database, this is designed during this phase. From the conceptual Information Model, a logical database schema can be generated (using tools like InfoModeler [33]) or manually built.

### 3.2.5 Implementation

This phase consists of the actual implementation of the web site. The implementation environment can be chosen: HTML, XML, WML, ... The implementation could be automated using available tools.

## 3.3 Conclusion

The most important characteristic of WSDM is that the different users of a web site find the *right information* in a *faster* way: each user doesn't get all data and each user gets the data on his way.



Part II  
Research

#### **4 Problem statement: design time support for adaptive behavior in web sites**

---

In the background part, we first described adaptive behavior in web sites. Afterwards, we briefly discussed some existing web site design methods and made a more extensive study of the Web Site Design Method (WSDM), which will be used as a framework later on in the research section.

Unfortunately, none of these methods provides any support for adaptive elements. This includes WSDM, which has some support for dynamic behavior, but no structural support for adaptive behavior. However, as we explained in chapter 1, the need for adaptive behavior in web sites is increasing rapidly.

In the research part of this thesis, we will demonstrate a solution for designing adaptive behavior. The proposed solution, in chapter 5, is applied on WSDM but could be used in other methods as well. Chapter 6 illustrates the solution with two extensive cases studies.

## 5 Suggestion to design adaptive behavior

---

In this chapter, we propose a solution to design adaptive behavior in web sites. The first solution you should think of, is placing conditions on the links in the navigational model of WSDM. These conditional links can denote dynamic behavior like ‘if user has condition  $c$ , then link  $l$  can be followed’. But what if you want to define adaptive behavior like “if a chunk is mostly accessed via a certain path of a particular length, put this chunk some levels higher in the path”? It is not possible to describe this with a condition on a link. The following sections describe a solution for such adaptive elements. Rules are defined to indicate which actions are to be undertaken when some condition is fulfilled. These rules have two parts: a condition and an operation. The operations are described in section 6.2 and the syntax for the rules in 6.3. But first we give a formal definition of the navigational model in section 6.1. With this formalization, we will be capable of defining operations on the model. Note that the definitions for the navigational model are general, and thus applicable for any web design methods. Since our working model will be WSDM, we will provide some additional definitions that deal with the peculiarities of the audience driven approach for WSDM.

### 5.1 Definition of the Navigational Model

This section gives a formal definition of the Navigational Model. This formalization will allow us to define operations on the model, which are used in the rules of section 5.4. We first define the Navigational Model with some properties in section 5.2.1. After this, section 5.2.2 gives the needed definitions for the Audience Track, and defines an Audience Based Navigational Model. This Audience Based Navigation Model represents the Navigational Model used in the current version of WSDM.

#### 5.1.1 The Navigational Model

First we define a Navigational Model and some properties. Later, we will define audience tracks as sub models of this model.

##### **Definition 5.1:** Navigational Model

We define a *Navigational Model*  $M$  for a website  $W$  as a 4-tupel  $(N, H, L, C)$ , where:

- $N$  is a finite non-empty set of nodes  $N$ ;
- $H$  is a finite non-empty set of chunks;
- $L \subseteq N \times N$ , is the set of links;
- $C \subseteq N \times H$ . is the set of connections between nodes and chunks.

**Interpretation**

Consider a web site with Navigational Model  $(N, H, L, C)$ :

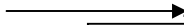
A link (element of  $L$ ), is a 2-tupel  $(n_s, n_t)$ , where  $n_s$  is the source and  $n_t$  the target of the link.

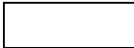
For  $n_s$  we write source ( $l$ ); for  $n_t$  we write target ( $l$ ).

A connection (element of  $C$ ) is a 2-tupel  $(n, c)$ , where  $n$  is the node of the connection and  $c$  is the chunk of the connection.

We write node ( $l$ ) and chunk ( $l$ ).

**Notations**

A link is denoted by 

A node is represented by: 

**Comments**

Let  $W$  be a web site with Navigational Model  $(N, H, L, C)$ : an element  $n$  of  $N$  represents a bundle of information- or functionality units. Elements of  $N$ , nodes, are also called *component*.

**Comments**

Consider a web site  $W$  with a set of chunks  $H$ .

A chunk  $h \in H$ , represents an informational or functionality chunk defined in the conceptual design.

**Definition 5.2**: root

Given a Navigational Model  $M = (N, H, L, C)$ , one of the elements of  $N$  can be indicated to be the *root* of the Navigational Model.  $\text{Root}(M)$  will be used to denote the root of  $M$ .

**Comment**

Given a Navigational Model  $M = (N, H, L, C)$  for a web site  $W$ ,  $\text{root}(M)$  represents the homepage of  $W$ .

**Definition 5.3**: path

Let  $n_1$  and  $n_2$  be two nodes of a Navigational Model  $M = (N, H, L, C)$ .

A  $n_1$ - $n_2$  *path* in  $M$  is a (loop-free) sequence of links  $l_1 \dots l_r \in L$  where:

If  $r = 1$  ( $l_1 = l_r$ ):

- source ( $l_1$ ) =  $n_1$
- target ( $l_r$ ) =  $n_2$

If  $r > 1$ :

- $\forall l_h, l_{h-1} \in L$  where  $1 < h \leq r$ : target ( $l_{h-1}$ ) = source ( $l_h$ )

The *length* of a path  $p$ , denoted by  $\text{length}(p)$ , is the number of links in  $p$ .

**Comment**

The length of a path corresponds with the navigation depth in the site.

**Definition 5.4: connected**

Consider a web site with Navigational Model  $(N, H, L, C)$ .

Two nodes  $n_1$  and  $n_2$  of the set  $N$  are called *connected* if there is a path between  $n_1$  and  $n_2$  in  $M$ .

We define 'well-connected' to check if every node from the Navigational Model can be reached (from the root of the web site). So this will be used to check if there are no lost components in the web site.

**Definition 5.5: well-connected**

Given a web site  $W$  with Navigational Model  $M = (N, H, L, C)$  and a root  $r$ , then  $M$  is called *well-connected* if:

$\forall n \in N$ :

A  $r$ - $n$  path exists in  $M$ .

### 5.1.2 Audience Tracks

Now that the Navigational Model is described, we define Audience Tracks as sub models of a Navigational Model. Finally, we define an Audience Based Navigational Model to represent a Navigational Model as it is currently used in WSDM.

The definition of sub models is necessary to define Audience Tracks later on. Audience Tracks will be defined as sub models with specific properties.

**Definition 5.6: sub model**

Consider a Navigational Model  $M = (N, H, L, C)$ . The Navigational Model  $M_1 = (N_1, H_1, L_1, C_1)$  is a *sub model* of  $M$  if:

- $\emptyset \neq N_1 \subseteq N$
  - $\emptyset \neq H_1 \subseteq H$
  - $L_1 \subseteq L$ ,
  - $C_1 \subseteq C$ ,
- where  $\forall l_x \in L_1$ : source  $(l_x) \in N_1$  and target  $(l_x) \in N_1$   
and  $\forall c_x \in C_1$ : node  $(c_x) \in N_1$  and chunk  $(c_x) \in H_1$

**Lemma 5.1: a sub model is a Navigational Model**

This can be directly derived from definition 5.6.

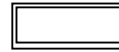
**Definition 5.7: Audience Track**

Given a web site  $W$  with a set of audience classes  $A = \{A_1, \dots, A_n\}$  and Navigational Model  $M = (N, H, L, C)$ .

An *Audience Track* for an audience class  $A_m \in A$  is a sub model  $TA_m = (N_1, H_1, L_1, C_1)$  of  $M$ .

**Notation**

The root of an Audience Track is denoted as:

**Comment**

The root of an Audience Track represents the first page of the audience track.

The property 'nested' for Audience Tracks is defined to express that one Audience Track is 'contained' within another one. This property is used in the definition of Audience Based Navigational Models to denote that, when an Audience Class is a subclass of another Audience Class, their corresponding Audience tracks are nested.

**Definition 5.8: nested**

Given a web site  $W$  with audience classes  $A = \{A_1, \dots, A_n\}$  and Navigational Model  $M = (N, H, L, C)$ .

Let  $T_1 = (N_1, H_1, L_1, C_1)$  and  $T_2 = (N_2, H_2, L_2, C_2)$  be two Audience Tracks of  $M$  for the respective audience classes  $A_1$  and  $A_2$ , then  $T_1$  is *nested* inside  $T_2$  if:

- $T_1$  is a sub model of  $T_2$
- AND
- $\text{root}(T_1) \neq \text{root}(T_2)$

While Audience Tracks can be nested, they could also be disjoint. This means they have no nodes in common.

**Definition 5.9: disjoint**

Let  $W$  be a web site with Navigational Model  $M$  and Audience Tracks  $T_1 = (N_1, H_1, L_1, C_1)$  and  $T_2 = (N_2, H_2, L_2, C_2)$ . Then  $T_1$  is *disjoint* from  $T_2$  if:

$N_1$  and  $N_2$  are disjoint.

**Comment**

Note that, although this definition requires the nodes of the different Audience Tracks to be disjoint, there *can* be shared chunks. Indeed, since chunks may be re-used, two different components may have a shared chunk connected to them.

The final definition defines the Navigational Model as it is currently used in WSDM, and is thus specific for the audience driven philosophy of WSDM. The formal model defined so far could easily be extended to reflect a data driven, or organization driven approach as followed by other web design methods; yet this is beyond the scope and goal of this thesis.

**Definition 5.10: Audience Based Navigational Model**

Given a web site  $W$  with a set of audience classes  $A = \{A_1, \dots, A_n\}$  and a Navigational Model  $M = (N, H, L, C)$ .  $M$  is an *Audience Based Navigational Model* if:

- $\forall A_m \in A$ , where  $1 \leq m \leq n$ :  $M$  has an Audience Track  $TA_m$  for  $A_m$
- if number of  $A > 1$ , then:
  - $\forall$  Audience Track  $TA_m = (N_m, H_m, L_m, C_m)$ :  $\text{root}(M) \notin N_m$
- $\forall$  audience class  $A_1$  which is a subclass of audience class  $A_2$ : the Audience Track for  $A_1$  is nested inside the Audience Track for  $A_2$
- $\forall$  audience class  $A_1$  which is a brother/sister of the audience class  $A_2$ : the Audience Track for  $A_1$  and the Audience Track for  $A_2$  are disjoint

Lemma 5.2 is a direct consequence of definition 5.10

**Lemma 5.2**

Consider a web site  $W$  with Navigational Model  $M$  and Audience Tracks  $T_1$  and  $T_2$ . If  $T_1$  is not a nested Audience Track in  $T_2$ , then:

The nodes of  $T_1$  are not accessible from  $T_2$  and vice versa, unless via root ( $M$ ).

**5.2 Operations on the Navigational Model**

With a formal definition of the Navigational Model, we can now formulate operations on the model. We assume the following while defining the operations:

*Let  $M = (N, H, L, C)$  be a Navigational Model of a web site  $W$ , then we will use  $M' = (N', H', L', C')$  as the resulting Navigational Model after execution of an operation  $Op$  on  $M$ , noted as  $Op(M)$ .*

### 5.2.1 Operations on nodes

The only needed operations on nodes are *deleteNode* and *addNode*. The operation *move* can be composed by operations on links, hence this operations is not defined separately. Supposing  $n \in N$ :

*deleteNode* ( $n$ )

$$N' = N \setminus \{n\}$$

and  $\forall l \in L$ , where  $\text{source}(l) = n$  or  $\text{target}(l) = n$ :  
 $L' = L \setminus \{l\}$

and  $\forall c = (n, h) \in C$ :  
 $C' = C \setminus \{c\}$

*addNode* ( $n$ )

$$N' = N \cup \{n\}$$

Remark that it is not useful to add a node without a chunk. Meta-information, as discussed in 5.4, should check that no nodes exist without chunks with it.

The two defined operations are the only basic operations on nodes. Other operations can be composed of *deleteNode* and *addNode*.

### 5.2.2 Operations on chunks

As with nodes, we define *deleteChunk* and *addChunk* operations to add or delete a chunk. However for chunks, we need two extra operations. We must be able to connect or disconnect a chunk to/from a node. We suppose  $h \in H$  and  $n \in N$ :

*deleteChunk* ( $h$ ):

$$H' = H \setminus \{h\}$$

and  $\forall c = (n, h) \in C$ :  
 $C' = C \setminus \{c\}$

*addChunk* ( $h$ ):

$$H' = H \cup \{h\}$$



*disconnect* ( $h, n$ ):

$$C' = C \setminus \{(n, h)\}$$

*connect* ( $h, n$ ):

$$C' = C \cup \{(n, h)\}$$

All other possible operations on chunks can be composed of the four defined operations. *Move*, for example: if we want to move a chunk  $h$  from component  $n_1$  to component  $n_2$  ( $h \in H$ ,  $n_1, n_2 \in N$ ), we first use *disconnect*( $h, n_1$ ) and afterwards *connect*( $h, n_2$ ).

### 5.2.3 Operations on links

*Delete* and *add* are the only operations on links that cannot be composed of other operations. To define these operations, we suppose  $l \in L$ .

*deleteLink* ( $l$ ):

$$L' = L \setminus \{l\}$$

*addLink* ( $l$ ):

$$L' = L \cup \{l\}$$

We have just defined the basic operations on links. All other necessary operations can be composed of the defined ones.

## 5.3 Rules to design adaptive behavior

In this chapter, we formulate a syntax for the navigational rules. Section 6.2.1 formalizes several kinds of adaptive improvements [19] to be designed with rules. Afterwards, 6.4.2 demonstrates the syntax, which can be used by the designer by filling in the variables. From now on, variables are shown *italic*.

### 5.3.1 Transformations

We first present three basic kinds of transformations that cannot be designed on the Navigational Model simply using conditions on links. For each, we give a more formal description.

Note that, when implementing a web site, there must be given meta-information to tell the system when adaptations may be made. For

example: in case of a university web site, the system may not put a link to a certain course between a list of links to faculties. More about this in 5.4.

Before beginning, it's worth explaining what we mean with the number of visits of web sites and components (used in the following paragraphs).

When a user visits a web site and visits a number of components and chunks, this is called a session. When talking about the number of visits of a component, we mean the number of sessions that visit that component. So when during one web site-visit, the component is visited three times, this is calculated as one.

The number of visits of a web site is the number of sessions.

### 5.3.1.1 Promotion and demotion

*Promotion* makes a link or page easier to find by moving a link closer to the home page of the web site.

Promotion and demotion are based on the popularity of links and pages. Promotion means adding a link to the model, while *demotion* means removing a link.

The x- and y-values in the following conditions are decided by the designer of the web site, but will probably be large.

#### Promotion

condition:

The number of visits of chunk/component  $c$  via a certain path  $p$ , is  $x$  % of the total number of visits of chunk/component  $c$  and  $y$  % of the number of visits of the web site  $W$ .

transformation:

Put  $c$  some steps higher in the path.

#### Demotion

condition:

The number of visits of chunk/component  $c$ , is  $x$  % of the number of visits of the web sites  $W$ .

transformation:

Put  $c$  some steps lower in the path.

### 5.3.1.2 Linking and unlinking

*Linking* connects two pages that were previously unconnected by adding new links between them. The fact that many users visit two pages in the same session suggests that they are conceptually related in users' minds, even if

the designer of the web site made no explicit connection. Similarly, *unlinking* is based on observing a lack of correlation; if links between two pages are never followed, we might infer that they are unrelated in users' minds, even though the designer connected them.

Again, the x-variable in the following conditions is decided by the designer, but would probably be large.

### **Linking**

condition:

Components  $c_1$  and  $c_2$  are related (based on access statistics or on contents)

transformation:

Add links between  $c_1$  and  $c_2$ .

### **Unlinking**

condition:

The number of traversings of link  $l$  is  $x\%$  of the number of visits of web site  $W$ .

transformation:

Remove link  $l$ .

### **5.3.1.3 Clustering**

*Clustering* associates a collection of related pages and makes them accessible as a group on a newly created page. The system recognizes a collection of similar documents that are not grouped together anywhere at the site, creates a new page for them and adds a reference to the new page. Documents may be considered related based on their filenames, their locations in the site hierarchy, their correlation in visitor paths, etc.

### **Clustering**

condition:

Two documents (chunks) are related (like described above), but not grouped somewhere on the site.

transformation:

add component with links to the chunks.

### 5.3.2 Syntax for rules

Finally, after formally defining the Navigational Model and operations on it, we can formulate the syntax for the rules. First we give some syntax-elements and in part 5.3.2.2, the whole syntax is defined.

#### 5.3.2.1 Some syntax-elements

Before describing the final syntax for rules, we first give some syntax-elements, used in the rule-syntax.

##### 1. Rule-specific elements

- numberOfVisits ( $c$ )  
*stands for:* # visits of component  $c$
- numberOfVisits ( $W$ )  
*stands for:* # visits of web site  $W$
- numberOfVisits ( $c, p$ )  
*stands for:* # visits of component  $c$ , via path  $p$
- numberOfTraversings ( $l$ )  
*stands for:* # traversings of link  $l$
- related ( $c_1, c_2$ )  
*stands for:* components/chunks  $c_1$  and  $c_2$  are related
- grouped ( $c_1, c_2$ )  
*stands for:* chunks  $c_1$  and  $c_2$  are grouped in one component
- forEach( $s, r$ )  
*stands for:* apply rule  $r$  on each element of the set defined in  $s$ .

The *forEach*( $s, r$ ) is necessary to define rules on groups of nodes, chunks or links. Rules will mostly be used with the *forEach*-element.

##### 2. General elements

- NOT ( $exp$ )  
*stands for:* negation
- ( $exp$ ) [AND ( $exp$ )]<sup>+</sup>

	<i>stands for:</i> conjunction
- /	<i>stands for:</i> division
- <	<i>stands for:</i> smaller then

### 5.3.2.2 The rule-syntax

This section describes the final syntax to describe rules, in BNF notation [34], with variables of a certain type. Two kinds of rules exist: rules specific on one element, and rules on groups of elements. Rules on a specific element are the simplest. They occur when the designer anticipates adaptive behavior on a certain element, and adds a rule specific for this element. Although this works fine in particular cases, in general, user behavior will be unpredictable and it will be extremely difficult for the designer to correctly anticipate user navigation behavior.

In other words, the designer should also be able to define certain adaptive behavior which are not hooked to a particular element, but instead are defined on all the nodes of the navigation model. A refinement will enable the designer to specify rules on groups of nodes: nodes that somehow have similar properties<sup>3</sup>, and may thus be subject of similar adaptive operations. The `forEach` in the syntax below is used to specify adaptive behavior on sets of nodes. The *italic* letters in the syntax denote variables ( $C_1, C_2$  and  $n_x \in N, c \in N$  or  $c \in H, p$  is a path,  $l \in L, h_x \in H$ ).

rule	::=	<expression>   forEach ((<set>), (<expression>)).
set	::=	<variable>* of <elements>   <variable>* of <elements> with (<property> (AND <property>)*).
elements	::=	N   H   L.
property	::=	audience track = T   length from root = x.
expression	::=	if <condition> then (<operation>+).
condition	::=	(<promotioncond>)   (<demotioncond>)

---

<sup>3</sup> For example: nodes within the same audience track, nodes that are semantically similar,  
...

		( < linkingcond > )   ( < unlinkingcond > )   ( < clusteringcond > ).
promotioncond	::=	( numberOfVisits ( c, p ) / numberOfVisits ( c ) ≥ x ) AND ( numberOfVisits ( c, p ) / numberOfVisits ( W ) ≥ y ).
demotioncond	::=	numberOfVisits ( c ) / numberOfVisits ( W ) ≤ x.
linkingcond	::=	related ( c <sub>1</sub> , c <sub>2</sub> ).
unlinkingcond	::=	numberOfTraversings ( l ) ≤ x.
clusteringcond	::=	( related ( c <sub>1</sub> , c <sub>2</sub> ) ) AND ( NOT ( grouped ( c <sub>1</sub> , c <sub>2</sub> ) ) ).
operation	::=	deleteNode ( n )   addNode ( n )   deleteChunk ( h )   addChunk ( h )   connect ( h, n )   disconnect ( h, n )   deleteLink ( l )   addLink ( l ).

Remark that the variables of the *forEach* statement are elements of N, H or L. In this syntax, two properties to denote the sets for the *forEach* are given: 'audience track = T' and 'length from root = x'. These are not the only possibilities. Arbitrary sets can be defined by designers to define adaptive behavior, e.g. all nodes related on a certain semantic area.

*Set* uses the non-terminal <variable>. A variable is a string of alphanumeric characters. The first character must be a letter:

**Terminal Alphabet:**

0..9, A..Z, a..z (all digits and alphabetic characters)

**Non-terminal Alphabet:**

<variable>, <alphaSequence>, <alphanumeric>, <letter>, <digit>

**Syntactic Rules:**

<variable> ::= <letter> <alphaSequence>  
 <alphaSequence> ::= <alphanumeric> <alphaSequence>  
 <alphanumeric> ::= <letter> | <digit>  
 <letter> ::= a | b | .. | z | A | B | .. | Z

`<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

#### 5.4 Future work

To complete the picture, the defined rules can be used in the implementation of web sites. A tool can be developed which interacts with the Navigational Model, so the implementation and adaptations of the web site are automated. Hence the designer can design the web site using the tool, which will implement the web site. While the web site is being used, the tool assures that the adaptations are being performed. Some issues should be thought yet of, like:

- An algorithm to check for completeness should be performed after adapting the site. This assures for example that no empty nodes exist.
- Not all adaptations make sense. For example: a link to a certain course cannot exist between a list of links to the different faculties on a page of a university site. Providing the site with meta-information can solve this: information about its content, structure and organization. We refer to [11] for more information.
- There should also be a way of specifying how often (which time intervals) the rules must be applied. Applying the rules on arbitrary moments would cause problems. For example if the rules are applied after one hour, the statistics will give a totally wrong representation. The designer should be possible to tell the system when to apply the rules. For example, the system is told to apply the rules every two weeks. It's worth to test this the intervals experimentally.

## 6 Cases

---

To illustrate our solution, we treat two cases. 6.1 shows a WSDM design of a web site of a youth organization called 'Muzemix' and section 7.2 treats a web site of an online bookstore.

## 6.1 Case 1: Youth organization 'Muzemix'

The first case is about a youth organisation called Muzemix. Muzemix has two partitions: a musical-atelier and a pop-youth-coir. The coir, for teenagers from 12 till 20 year-old, presents regularly and the musical-atelier, for children/youths from 7 till 17 year-old, has one big presentation of some days per year.

The purpose of the website is:

*"Promote the presentations and provide general information about Muzemix to non members. Provide specific information to members, tutors, workgroups and the board of directors."*

In the following sections, we elaborate each phase of WSDM. Purposes and details of all phases are explained in chapter 3, hence these are not included in this chapter.

### 6.1.1 Mission Statement Specification

- Purpose
  - Offer information of the organization to non members
  - Offer information of the ateliers to the members
  - Promoting the presentations
  - Offer reports of meetings online to the tutors, the workgroups and the board of directors.
- Target audience
  - non members
  - musical-actors (= members of the atelier)
  - members of the pop youth-coir
  - tutors
  - members of the workgroups
  - members of the board of directors
- Subject
  - musical-ateliers
  - coir-repetitions
  - presentations

### 6.1.2 Audience Modeling



### 6.1.2.1 Audience Classification

**Target audience:** non members, members of the atelier, members of the pop youth-coir, tutors, members of the workgroups, members of the board of directors



#### Audience classes

- Non member
- Atelier-member
- Coir-member
- Tutor
- Workgroup-member
- Administration-member

#### Requirements

##### Audience Class **Non member**

###### Information requirements

- information about the organization (general, atelier and coir)
- information about presentations

###### Functional requirements

- get information about the organization
- get information about presentations

###### Usability requirements

- flexible ways to get information about the organization
- flexible ways to get information about presentations

##### Audience Class **Atelier-member**

###### Information requirements

- information about the atelier
- information about presentations

###### Functional requirements

- get information about the atelier
- get information about presentations

###### Usability requirements

- flexible ways to get information about the atelier
- flexible ways to get information about presentations

##### Audience Class **Coir-member**

###### Information requirements

- information about the coir-repetitions
- information about presentations

###### Functional requirements

- get information about the coir-repetitions
- get information about presentations

###### Usability requirements

- flexible ways to get information about the coir-repetitions

- flexible ways to get information about presentations

#### Audience Class **Youth-member**

##### Information requirements

- information about presentations

##### Functional requirements

- get information about presentations

##### Usability requirements

- flexible ways to get information about presentations

#### Audience Class **Tutor**

##### Information requirements

- information about the atelier
- information about the coir-repetitions
- information about presentations
- reports of tutor-meetings

##### Functional requirements

- get information about the atelier
- get information about the coir-repetitions
- get information about presentations
- add information about the atelier
- add information about the coir-repetitions
- get reports of tutor-meetings
- add reports of tutor-meetings

##### Usability requirements

- flexible ways to get information about the atelier
- flexible ways to add information about the atelier
- flexible ways to get information about the coir-repetitions
- flexible ways to add information about the coir-repetitions
- flexible ways to get information about presentations
- flexible ways to get reports of tutor-meetings
- flexible ways to add reports of tutor-meetings

#### Audience Class **Workgroup-member**

##### Information requirements

- reports of workgroup-meetings
- information about presentations
- information about coir-repetitions
- information about the atelier

##### Functional requirements

- get reports of workgroup-meetings
- add reports of workgroup-meetings
- get information about presentations
- get information about coir-repetitions
- get information about the atelier

##### Usability requirements

- flexible ways to get reports of workgroup-meetings
- flexible ways to add reports of workgroup-meetings
- flexible ways to get information about presentations

- flexible ways to get information about coir-repetitions
- flexible ways to get information about the atelier

#### Audience Class **Administration-member**

##### Information requirements

- reports of core-meetings
- reports of workgroup-meetings
- reports of tutor-meetings
- information about presentations
- information about the musical-atelier
- information about the coir-repetitions

##### Functional requirements

- get reports of core-meetings
- get reports of workgroup-meetings
- get reports of tutor-meetings
- add reports of core-meetings
- get information about presentations
- get information about the atelier
- get information about the coir-repetitions
- add information about presentations

##### Usability requirements

- flexible ways to get reports of core-meetings
- flexible ways to get reports of workgroup-meetings
- flexible ways to get reports of tutor-meetings
- flexible ways to add reports of core-meetings
- flexible ways to get information about presentations
- flexible ways to get information about the atelier
- flexible ways to get information the coir-repetitions
- flexible ways to add information about presentations

#### Audience Class **Organization-member**

##### Information requirements

- information about presentations
- information about the atelier
- information about the coir-repetitions

##### Functional requirements

- get information about the atelier
- get information about the coir-repetitions
- get information about presentations

##### Usability requirements

- flexible ways to get information about the ateliers
- flexible ways to get information about the coir-repetitions
- flexible ways to get information about presentations

#### **Audience class hierarchy**

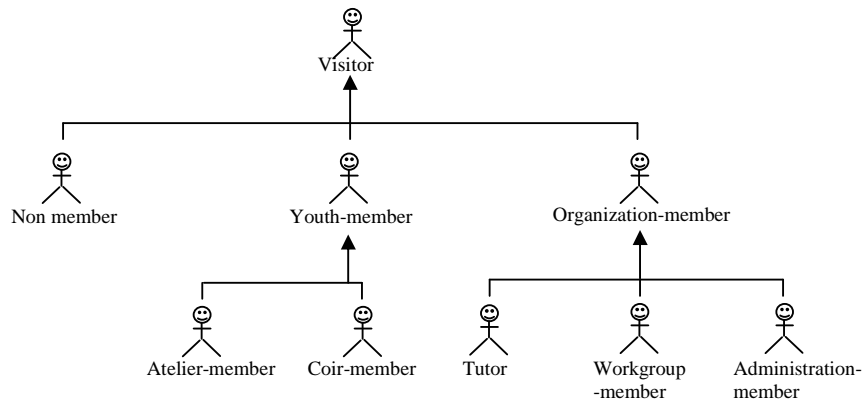


Figure 6.1. Audience Class Hierarchy for the Muzemix example.

### 6.1.2.2 Audience Class Characterization

#### Audience Class **Non member**

##### Characteristics

- all ages
- mostly experienced with WWW
- all languages possible

#### Audience Class **Atelier-member**

##### Characteristics

- 7 to 17 year-old boys and girls
- experience with WWW may vary
- language: dutch

#### Audience Class **Coir-member**

##### Characteristics

- 12 to 20 year-old boys and girls
- experience with WWW may vary
- all languages possible, mostly dutch

#### Audience Class **Youth-member**

##### Characteristics

- 7 to 20 year-old boys and girls
- experience with WWW may vary
- all languages possible, mostly dutch

#### Audience Class **Tutor**

##### Characteristics

- adults
- experience with WWW may vary
- language: dutch

### Audience Class **Workgroup-member**

#### Characteristics

- mostly adults
- experience with WWW may vary
- all languages, mostly dutch
- 

### Audience Class **Administration-member**

#### Characteristics

- adults
- experience with WWW may vary
- language: dutch

### Audience Class **Organization-member**

#### Characteristics

- mostly adults
- experience with WWW may vary
- all languages: mostly dutch

## 6.1.3 Conceptual Design

### 6.1.3.1 Information Modeling

We only show the informational model for the information requirement 'information about presentations'.

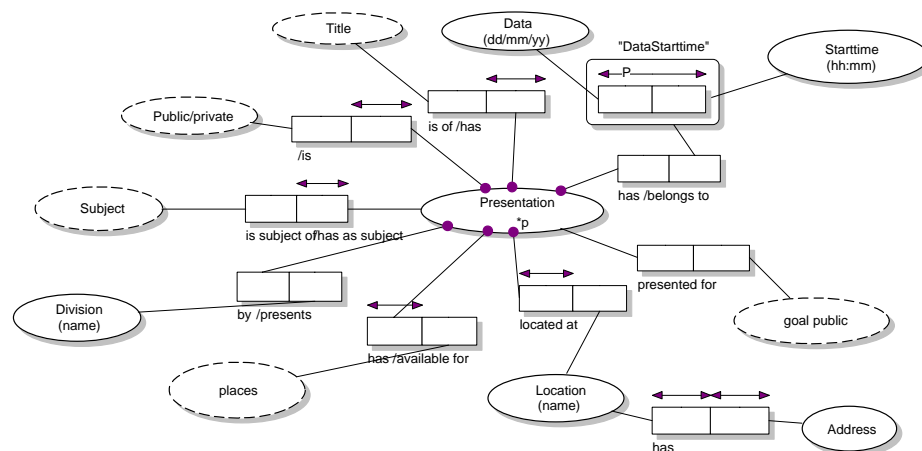


Figure 6.2. Informational Chunk for Presentation.

### 6.1.3.2 Functional Modeling

The decomposition of the functional requirement 'add information about presentations' results in:

Add information about presentations

- Register new presentation
- Submit presentation info

**Register new presentation**

The user enters the title of the presentation.

**Submit presentation info**

The user enters the date, time, goal public, location, number of places, subject and division, and indicate the presentation to be public or private.

We show the functional model for the requirement 'submit presentation info' in figure 6.3.

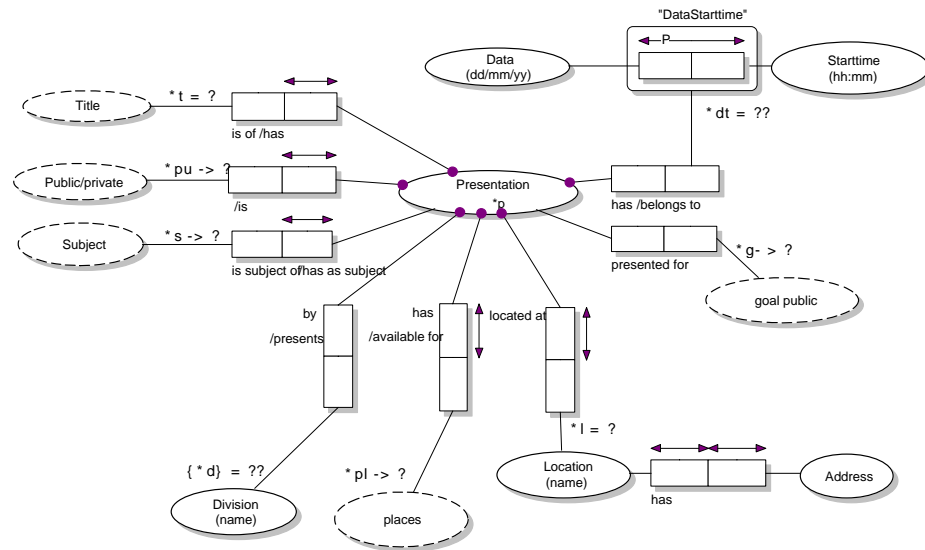


Figure 6.3. Functional Chunk for Submit Presentation Info.

**6.1.3.3 Navigational Design**

Figure 6.4 gives the track structure of the Muzemix web site. The next step is to elaborate the Navigation Track for the different Audience Classes.

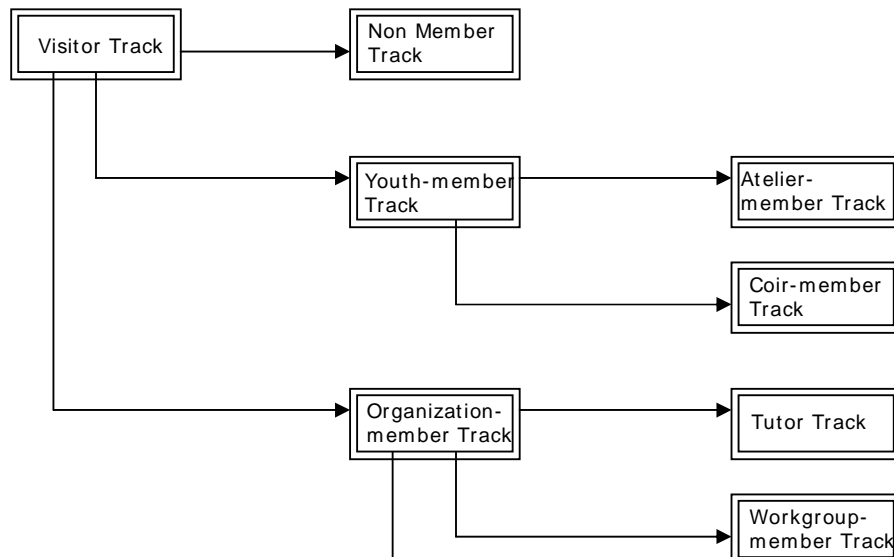
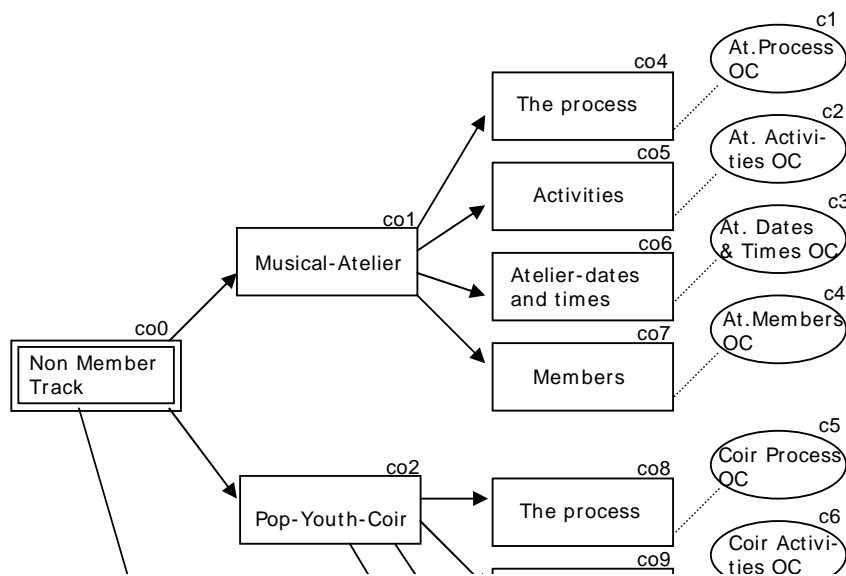


Figure 6.4. Main Track Structure of the Navigation Model.

Figure 6.5 shows the Navigation Track structure for the Audience Class *Non Member*. The rules that describe the adaptive behavior of the Track, are given below. To identify the links, components and chunks, they got an ID on the picture.



Now, during navigational design, our solution proposed in chapter 5 can be used to denote adaptive behavior. Below we give some rules a designer



could write for the track of figure 6.5. The first rules are rules on a specific node, chunk or link. These can be used when the designer conjectures that there is a correlation between the elements and he will know from the user's access behavior if this is true or not. The last rules are rules on groups of elements.

#### 1 Rules on a specific element

- We state as designer that it could be possible that the presentations of the organization (co13) are a very popular issue. If  $W$  is the web site that is be designed, we define the rule:

```

if ((numberOfVisits (co13, ((co0,co3),(co3,co13))) /
      numberOfVisits (co13)  $\geq$  1)
AND (numberOfVisits(co13, ((co0,co3),(co3,co13))) /
      numberOfVisits (W)  $\geq$  0.1))
then addLink ((co0, co13)).

```

The first part of the rule is a condition which is always true, while the second part is a condition that states that the component co13 is, in 10 % of the visits of  $W$ , accessed (via a certain path). This rule handles an example of *promotion*.

- The second rule which could be added to the navigational design, is an example of *linking*.

We know that it may be possible that users that visit the component of the Board of Directors, they are also interested in the Tutors component. If statistics prove that this is true, a link between the two is needed. This is given in the following rule:

```

if related (co14, co15)
then addLink ((co14, co15))
AND addLink ((co15, co14)).

```

- As third example, we give a rule that handles *clustering*. If a certain number of visits of the web site show that the components *Musical-Atelier* and *Pop-Youth-Coir* are related, it would be a solution to make a new component with links to these two components:

```

if (related (co1, co2))
AND (NOT (grouped (co1, co2)))
then addNode (co19)
AND addLink ((co0, co19))
AND deleteLink ((co0, co1))
AND deleteLink ((co0, co2))
AND addLink ((co19, co1))
AND addLink ((co19, co2)).

```

- The next example is an example of *linking*. We think about the fact that visitors of the site that are looking for the atelier (or the coir), are also interested in the presentations they give. So we write a rule that links the two if necessary:

For the musical-atelier:  
**if** related (co1, co13)  
**then** addLink ((co1, co13))  
**AND** addLink ((co13, co1)).

For the pop-youth-coir:  
**if** related (co2, co13)  
**then** addLink ((co2, co13)),  
**AND** addLink ((co13, co2)).

## 2 Rules on groups of elements

This section describes some rule on groups of elements. Remark that, with this rules, some rules of the previous section are overflowing. However, we gave that rules to have more examples to demonstrate our solution of chapter 5.

- The first rule in the Muzemix example is a rule on *promotion*. Now we use such a rule with *forEach* to denote that we want that rule applied on every element of the audience class 'non member' and on a certain level, namely three.

```
forEach ((element1 of H with audience track = non member,  

length from root = 3),  

if ((numberOfVisits (element1, ((co0,co3),(co3, element1))) /  

numberOfVisits (element1) ≥ 1)  

AND (numberOfVisits(element1, ((co0,co3),(co3, element1))) /  

numberOfVisits (W) ≥ 0.1))  

then addLink ((co0, element1))).
```

- The second rule on a specific element is an example of *linking*. We now write a *forEach* rule that uses this rule, to describe that we want to add a link between all related components. This rule is useful when some information is not expected to be related, however seems to be related in users' mind.

```
ForEach((element1, element2 of N),  

if related (element1, element2)  

then addLink ((element1, element2)  

AND addLink ((element2, element1))).
```

### 6.1.4 Implementation Design

Like already mentioned, the method identifies the different steps of the implementation design, but is not yet completely elaborated on this point. Hence we only briefly discuss the implementation design phase.

#### 6.1.4.1 Page Design

Figure 6.6 shows the page design for the *Non Member Audience Track*.

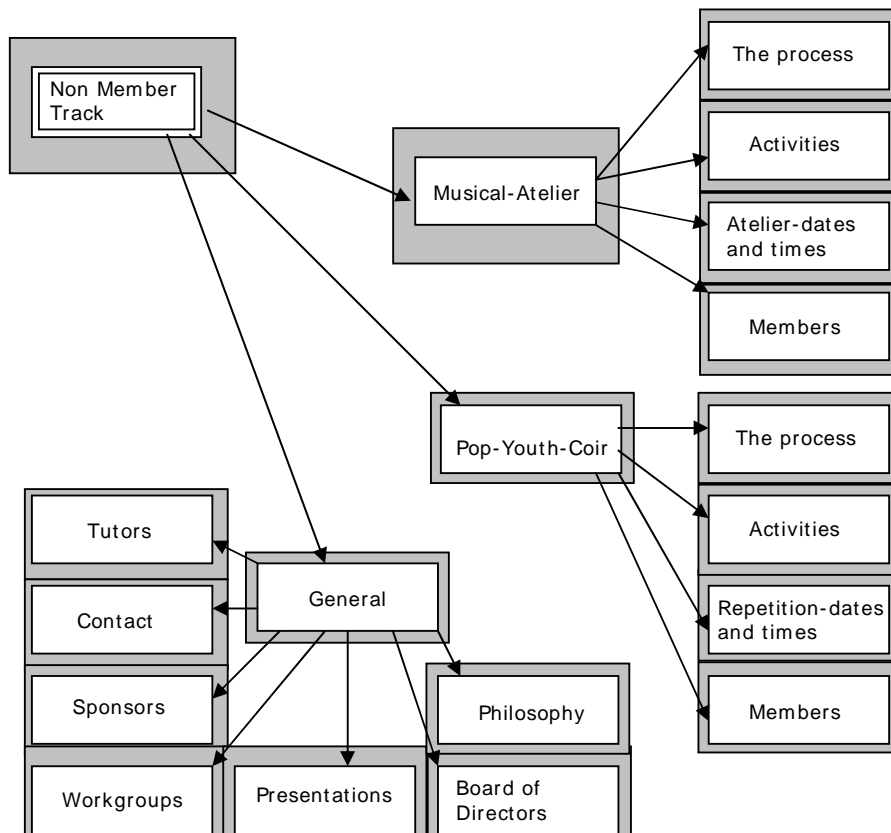


Figure 6.6. Page structure for the Non Member Track.

#### 6.1.5 Implementation

The last phase, the actual construction of the web site using the chosen environment, is not in the scope of this chapter, because this thesis is about the design and not the implementation of web sites.

#### 6.1.6 Conclusion

The Muzemix example shows that the proposed solution in chapter 5 is not just theory, but is really applicable while designing web sites. We gave some examples of rules. Some of *linking*, one about *clustering* and one about *promotion*. The next case, the online bookstore, in 6.2 give some other example-rules.

## 6.2 Case 2: Online Bookstore

The second case is an online bookstore. The web site provides information of books and authors, and users can buy books online.

The purpose of the website is:

*“Offer information of books and authors. Also offer books online to all interested people, private buyers as well as bookshop-keepers. Authors must be able to add (information of) books.”*

Again, we don't include purposes and details of all phases, because these are already elaborated in chapter 3.

### 6.2.1 Mission Statement Specification

- Purpose
  - offer information of books online
  - offer information of authors online
  - offer books online
- Target audience
  - everyone who wants information about books or authors
  - everyone who wants to buy a book online
  - authors
- Subject
  - books
  - authors

### 6.2.2 Audience Modeling

#### 6.2.2.1 Audience Classification

**Target audience:** everyone who wants information about books or authors, everyone who wants to buy a book online, authors



#### **Audience classes**

- Reader (= people who just want information)
- Private-buyer
- Bookshop-keeper
- Author

## Requirements

### Audience Class **Reader**

#### Information requirements

- data about books
- data about authors

#### Functional requirements

- get information about a book
- get information about an author

#### Usability requirements

- flexible ways to get information about a book
- flexible ways to get information about an author

### Audience Class **Private-buyer**

#### Information requirements

- data about books

#### Functional requirements

- search (by ISBN, title, author, ...) for a book
- buy a book

#### Usability requirements

- flexible ways to search for a book
- flexible ways to buy a book

### Audience Class **Bookshop-keeper**

#### Information requirements

- data about books

#### Functional requirements

- search (by ISBN, title, author, ...) for a book
- get specialized information about books
- buy large quantities of books

#### Usability requirements

- flexible ways to search for a book
- flexible ways to get specialized information about books
- flexible ways to buy large quantities of books

### Audience Class **Buyer**

#### Information requirements

- data about books

#### Functional requirements

- search (by ISBN, title, author, ...) for a book
- buy books

#### Usability requirements

- flexible ways to search for a book
- flexible ways to buy books

### Audience Class **Author**

#### Information requirements

- data about his/her own books

- data about his/her own
- Functional requirements
- add a book he/she has written
  - add information about a book he/she has written
  - change information about a book he/she has written
  - delete information about a book he/she has written
  - add information about his/her own
  - change information about his/her own
  - delete information about his/her own
- Usability requirements
- flexible ways to add a book
  - flexible ways to add information about a book
  - flexible ways to change information about a book
  - flexible ways to delete information about a book
  - flexible ways to add information about his/her own
  - flexible ways to change information about his/her own
  - flexible ways to delete information about his/her own

#### Audience class hierarchy

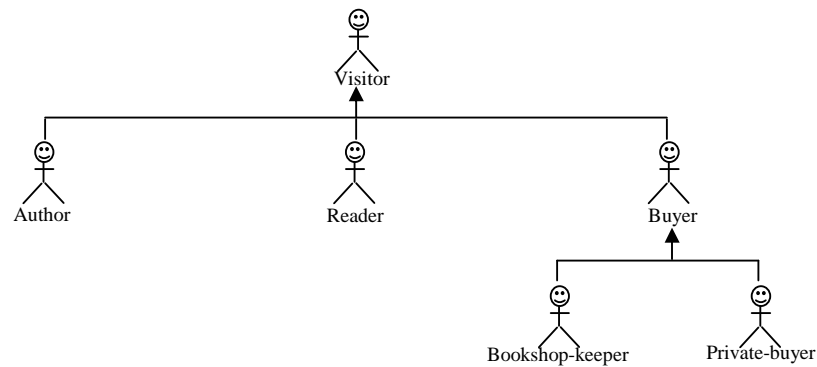


Figure 6.7. Audience Class Hierarchy for the Online Bookstore example.

#### 6.2.2.2 Audience Class Characterization

##### Audience Class **Author**

###### Characteristics

- mostly adults
- experience with WWW may vary
- all languages possible

##### Audience Class **Reader**

###### Characteristics

- all ages

- mostly experienced with WWW
- all languages possible

Audience Class **Private-buyer**

Characteristics

- all ages
- mostly experienced with WWW
- all languages possible

Audience Class **Bookshop-keeper**

Characteristics

- all ages
- mostly experienced with WWW
- all languages possible

Audience Class **Buyer**

Characteristics

- all ages
- mostly experienced with WWW
- all languages possible

## 6.2.3 Conceptual Design

### 6.2.3.1 Information Modeling

We only show the informational model for the information requirement 'information about books', in figure 6.8.

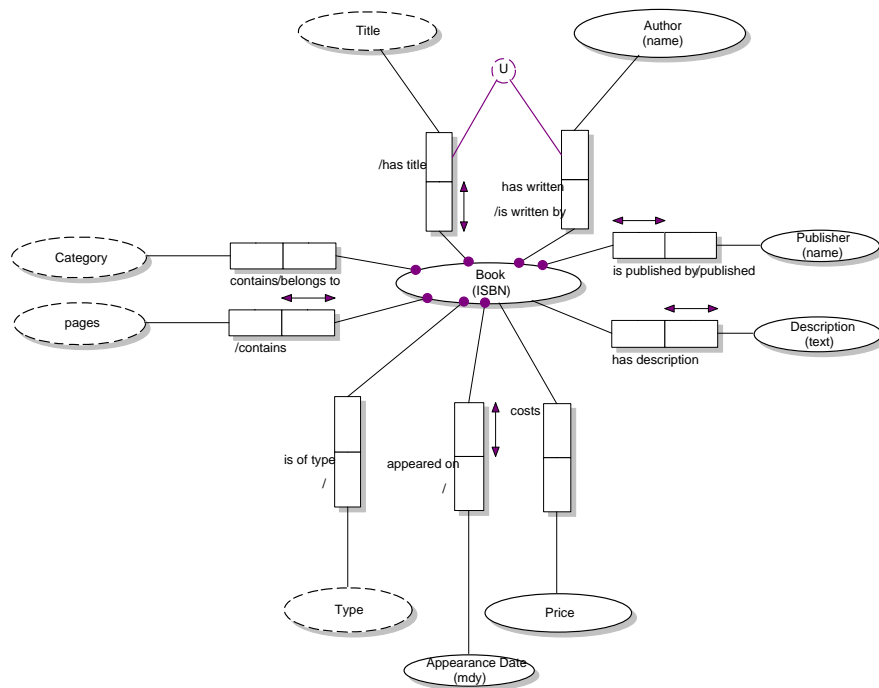


Figure 6.8. Informational Chunk for Book.

### 6.2.3.2 Functional Modeling

The decomposition of the functional requirement 'add a book' results in:

Add a book

- Register new book
- Submit book info

#### **Register new book**

The user enters the title of the book.

#### **Submit book info**

The user enters the authors, publisher, price, appearance date, type, pages and category.

We show the functional model for the requirement 'add information about a book' in figure 6.9.



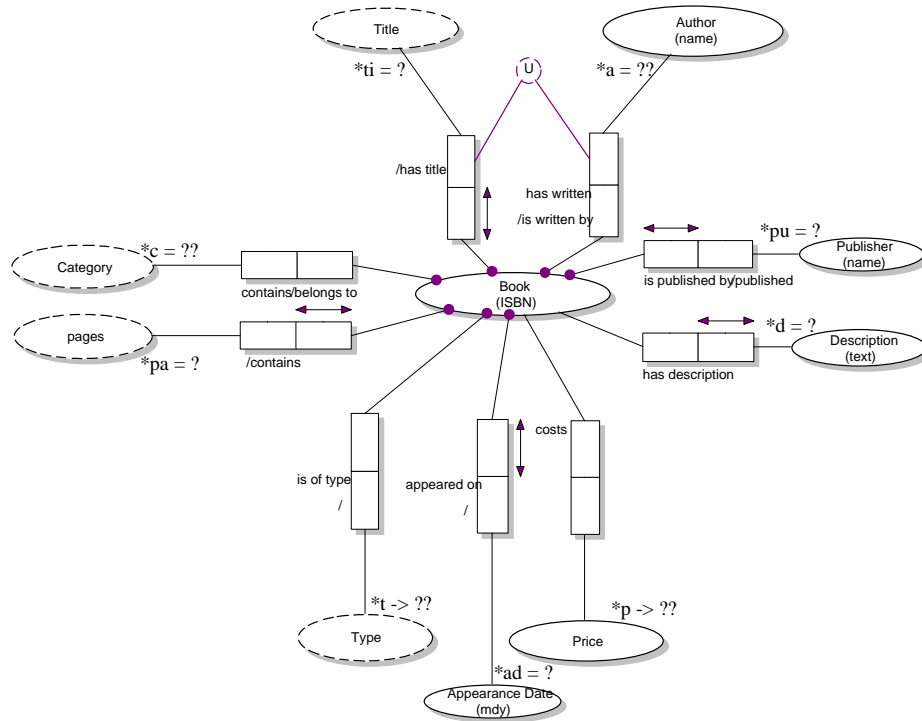


Figure 6.9. Functional Chunk for Add information about a book.

### 6.2.3.3 Navigational Design

Figure 6.10 gives the track structure of the online bookstore. The next step is to elaborate the Navigation Track for the different Audience Classes.

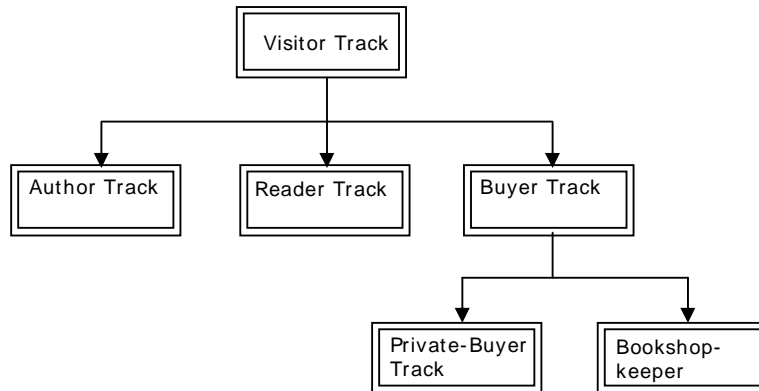


Figure 6.10. Main Track Structure of the Navigation Model.

Figure 6.11 shows the Navigation Track structure for the Audience Class *Reader*. The rules that describe the adaptive behavior of the Track, are given below. To identify the links, components and chunks, they got an ID on the picture.

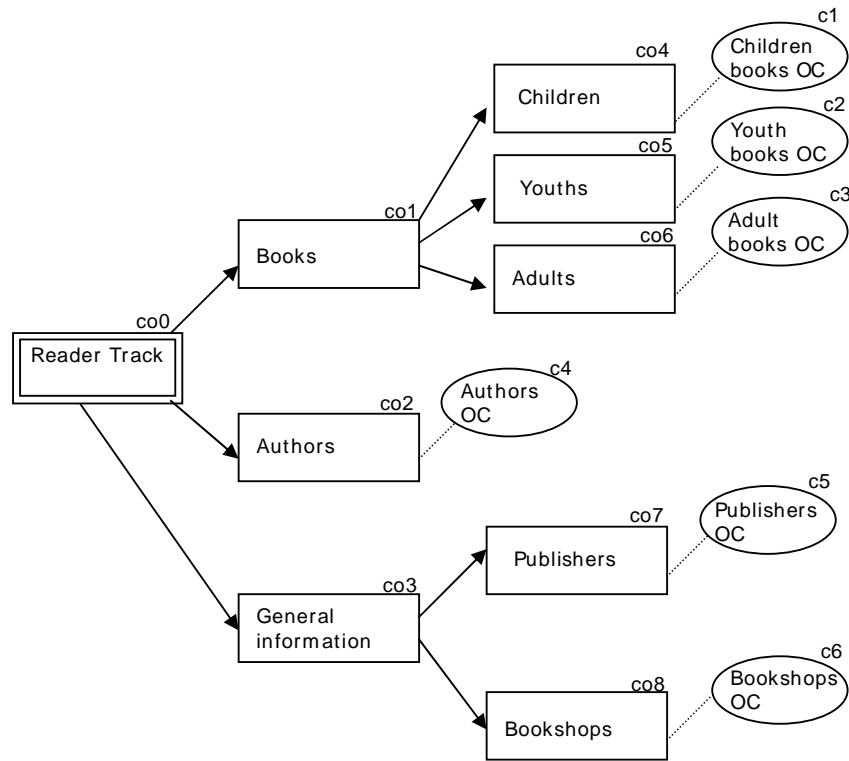


Figure 6.11. Navigation Track for Reader.

Now we can use the proposed solution from chapter 5 to describe adaptive behavior. Below we write some possible rules for the track of figure 6.11. There are two kinds of possible rules: rules on a specific node, chunk or link; and rules on groups of links. The former are described in the first part and the latter in the second part. The rules on a specific element can be used when the designer conjectures that there is a correlation between the elements and he will know from the user's access behavior if this is true or not.

## 1 Rules on a specific element

- The first rule is an example of linking. We know that users that search for adults-books, often also look for books for their children. If users' access behavior proves this, we want to add a link between them:

```

if related (co4, co6)
then addLink ((co4, co6))
AND addLink ((co6, co4)).

```

- As second example, a rule with promotion is given. We state as designer that the information about bookshops can be a very popular issue. Supposing  $W$  is the web site, we define the rule:

```

if ((numberOfVisits (co8, ((co0,co3),(co3,co8))) /
      numberOfVisits (co8)  $\geq$  1)
AND (numberOfVisits(co8, ((co0,co3),(co3,co8))) /
      numberOfVisits (W)  $\geq$  0.1))
then addLink ((co0, co8)).

```

- The third example is a rule with *demotion*. We think that it can be possible that information about authors is not as popular as we suggest. Maybe it is better to put the component *authors* a level lower. In the rule,  $W$  is the web site.

```

if numberOfVisits (co2) / numberOfVisits (W)  $\geq$  x
then deleteLink(co0, co2)
AND addLink(co3, co2).

```

## 2 Rules on groups of elements

In this section, we give an example of a rule on a group of elements. Remark again that the rules on groups of elements will be mostly used when designing a web site.

- This example is a rule on a group of elements that handles *linking*. It tells the system that if any two nodes are related, there must be added a link between them. We write this rule using the *forEach* statement:

```

ForEach((element1, element2 of N),
  (if related (element1, element2)
then addLink ((element1, element2))
AND addLink ((element2, element1))))).

```

## 6.2.4 Implementation Design

Like already mentioned, the method identifies the different steps of the implementation design, but is not yet completely elaborated on this point. Hence we only briefly discuss the implementation design phase.

### 6.2.4.1 Page Design

Figure 6.12 shows the page design for the *Reader Audience Track*.

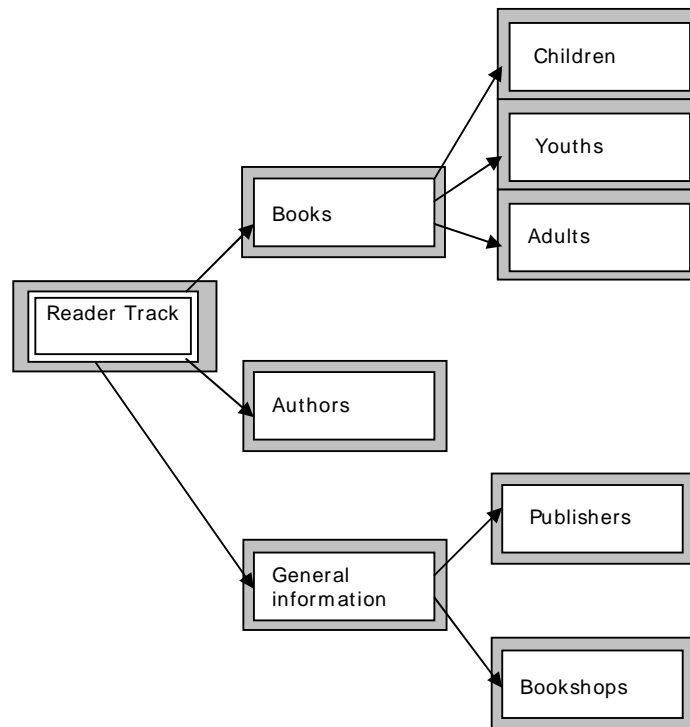


Figure 6.12. Page structure for the Reader Track.

## 6.2.5 Implementation

The last phase, the actual construction of the web site using the chosen environment, is not in the scope of this chapter, because this thesis is about the design and not the implementation of web sites.

### **6.2.6 Conclusion**

The Online Bookstore proves, as well as the Muzemix example from section 6.1, that the solution proposed in chapter 5 is really applicable for designing web sites.

## 7 Conclusion

---

The goal of this thesis was to find a way to support adaptive behavior of web sites at design time. We thoroughly discussed existing web site design methods and their support for adaptive behavior in web sites. Until now, existing design methods are limited equipped with the necessary design constructs for designing adaptive web sites.

We proposed a solution to design adaptive behavior. The solution consists of the following parts:

- First we gave a formal definition of the Navigational Model and Audience Tracks. This definition was necessary to define operations on the Navigational Model in the next step of the solution.
- Having a formal definition of the Navigational Model and the Audience Tracks, we defined operations on the model. These include *add-* and *delete-* operations on chunks, components and links. Also *connect-* and *disconnect* operations are provided, to connect and disconnect chunks from components. Other possibly needed operations like *move* can be composed of the defined ones. The operations define the adaptive transformations and are used in the rules of the next step.
- Finally, we produced a syntax in BNF notation [34] to define rules. With this rules, the designer of a web site can indicate which adaptive transformation have to be performed. A rule consists of two parts, a condition and an operation:  
rule ::= if <condition> then (<operation>+).

The proposed solution is then applied and demonstrated in two case studies to make clear that it offers a good manner to design adaptive behavior in web sites.

We can conclude this thesis by stating that the goal we put upfront, namely proposing a solution to design adaptive behavior in web sites, is met by the definition, operations and rules. We concentrated on WSDM but the proposed solution is generally applicable, hence it could be used in other methods as well.

## Bibliography

- [1] D. Schwabe and G. Rossi (1998), "Developing Hypermedia Applications using OOHDM", Departamento de Informática, PUC-RIO, Brazil and LIFIA, Fac Cs. Exactas, UNLP, Argentina.
- [2] D. Schwabe, G. Rossi and S.D.J. Barbosa (1995), "Abstraction, Composition and Lay-out Definition Mechanisms in OOHDM" in Proceedings of the ACM Workshop on Effective Abstractions in Multimedia, San Francisco, California.
- [3] D. Schwabe and G. Rossi (2001), "The Object-Oriented Hypermedia Design model (OOHDM)", <http://www.telemidia.puc-rio.br/oohdm/oohdm.html>.
- [4] S. Ceri, P. Fraternali and A. Bongio (2000), "Web Modeling language (WebML): a modeling language for designing Web sites", Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy.
- [5] T. Isakowitz, E.A. Stohr and P. Balasubramanian (1995), "RMM: A Methodology for Structured Hypermedia Design", Communications of the ACM, Vol. 38, No. 8.
- [6] J. Gomez, C. Cachero and O. Pastor (2001), "Conceptual Modeling of Device-Independent Web Applications", <http://www.dsic.upv.es/~west2001/iwwost01/files/contributions/JaimeGomez/ieeeMultimedia.pdf>.
- [7] J. Gomez, C. Cachero and O. Pastor (2001), "Extending a Conceptual Modelling Approach to Web Application Design", <http://www.dsic.upv.es/~west2001/iwwost01/files/contributions/JaimeGomez/caise00.pdf> .
- [8] M. Perkowitz, O. Etzioni (1999), "Towards adaptive Web sites: Conceptual framework and case study", University of Washington, Department of Computer Science and Engineering, Box 352350, Seattle, USA.
- [9] H. Wu, E. de Kort, P. De Bra (2001), "Design issues for General-Purpose Adaptive Hypermedia Systems", Eindhoven University of Technology.
- [10] R.P. Grimaldi (1994), "Discrete and combinatorial mathematics", third edition, Rose-Hulman Institute of Technology, Addison-Wesley Publishing Company.
- [11] M. Perkowitz, O. Etzioni (1997), "Adaptive Web Sites: an AI Challenge", Department of Computer Science and Engineering, University of Washington.

- [12] O. De Troyer, S. Casteleyn (2001), "The Conference Review System with WSDM", in Proceedings of IWOST 2001, Valencia, Spain.
- [13] O. De Troyer (2001), "WiSDoM Short description", <http://wsgm.vub.ac.be/ShortDescription.htm>, Brussels, Belgium.
- [14] O. De Troyer (2002), "WSDM Web Site Design Method", used for VUB courses, Vrije Universiteit Brussel, Belgium.
- [15] S. Casteleyn, O. De Troyer (2001), "Structuring Web Sites Using Audience Class Hierarchies", Proceedings of DASWIS 2001 workshop (attached to the ER 2001 conference), Yokohama, Japan.
- [16] O. De Troyer (2000), "Audience-driven Web Design", Vrije Universiteit Brussel, WISE research group, Belgium.
- [17] G. Rossi, A. Fortier, J. Cappi and D. Schwabe (2001), "Seamless Personalisation of E-Commerce Applications", LIFIA-Facultad de Informatica-UNLP, Argentina and Depto de Informatica, PUC-Rio, Brazil.
- [18] G. Rossie, D. Schwabe and R. M. Guimaraes (2001), "Designing Personalised Web Applications", LIFIA-Facultad de Informatica-UNLP, Argentina and Departamento de Informatica, PUC-RIO, Brazil.
- [19] M. Perkowitz and O. Etzioni (1997), "Adaptive Sites: Automatically Learning from User Access Patterns", Department of Computer Science, University of Washington, Seattle.
- [20] Object Management Group Inc. (2002), "UML Home Page", <http://www.uml.org/>.
- [21] T. Halpin (2001), "Conceptual Schema & Relational Database Design", Second Edition, WytLytPub.
- [22] W3C (2001), "The World Wide Web", <http://www.w3.org/Security/Faq/wwwsf.html>.
- [23] Software Design Center (2002), "How to Draw Rumbaugh (OMT) Diagrams", <http://www.smartdraw.com/resources/centers/software/omt.htm>.
- [24] O<sub>2</sub> Technology (1998), "ODMG OQL User Manual", <http://www.cis.upenn.edu/~cis550/oql.pdf>.
- [25] V. Markowitz (1996), "The Entity-Relationship (ER) Model", [http://gizmo.lbl.gov/DM\\_TOOLS/OPM/ER/ER.html](http://gizmo.lbl.gov/DM_TOOLS/OPM/ER/ER.html).
- [26] W3C (2000), "Extensible Markup Language (XML)", <http://www.w3.org/XML/>.



- [27] W3C (2002), "HyperText Markup Language Home Page", <http://www.w3.org/MarkUp/>.
- [28] W3C (2002), "Wap Tutorial", <http://www.w3schools.com/wap/default.asp>.
- [29] R. H. Zakon (2002), "Hobbes' Internet Timeline", <http://www.zakon.org/robert/internet/timeline/>.
- [30] M. Fernandez, D. Florescu, A. Levy, D. Suciu, J. Kang, S. Sudarsky and I. Tatarinov (1999), "Strudel Web-site Management System User's Guide", <http://www.research.att.com/%7Emff/strudel/>.
- [31] C. Anderson, A. Levy, D. Weld (1999), "Declarative web-site management with Tiramisu", University of Washington, Department of Computer Science and Engineering.
- [32] P. Atzeni, A. Parente (2001), "Specification of Web applications with ADM-2", Dipartimento di Informatica e Automazione, Roma.
- [33] Visio Corporation (1997), "Visio Technology", <http://www.visio.com/>.
- [34] M. Marcotty and H. Ledgard (1986), "The World of Programming Languages", Springer-Verlag, Berlin, pp. 41 and following.