Vrije Universiteit Brussel

FACULTY OF SCIENCE AND BIO-ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

WSDM Web Development Adapted to Website Genres, Design Patterns and, Content Management Systems

Graduation thesis submitted in partial fulfillment of the requirements for the degree of Master in de Ingenieurswetenschappen: Computerwetenschappen

Kevin Van Gyseghem

Promoter: Prof. Dr. Olga De Troyer
Advisor: Pejman Sajjadi

Academic year 2013-2014

Vrije Universiteit Brussel

FACULTEIT WETENSCHAPPEN EN BIO-INGENIEURSWETENSCHAPPEN
VAKGROEP COMPUTERWETENSCHAPPEN

WSDM Web Development Adapted to Website Genres, Design Patterns and, Content Management Systems

Masterproef ingediend in gedeeltelijke vervulling van de eisen
voor het behalen van de graad
Master in de Ingenieurswetenschappen: Computerwetenschappen

Kevin Van Gyseghem

Promoter: Prof. Dr. Olga De Troyer
Advisor: Pejman Sajjadi

Academiejaar 2013-2014

Abstract

The purpose of this thesis is to adapt the Web Semantic Design Method (WSDM) for implementations that use a Content Management System (CMS). The problem is that the models of the WSDM methodology specify everything in much detail, which does not match the level of control one have when using a CMS. An adapted WSDM methodology, called WSDM-Lite, is proposed to solve this problem.

This thesis introduces the concept of Web Design Patterns (WDPs) which are conceptual descriptions of features that can be found in websites. These WDPs roughly match functionality that can be found in installable modules for CMSs. The WDPs are reusable components that are modeled together using a feature assembly diagram.

WSDM-Lite uses website genres to speed up the development process. By defining Web Genre Patterns (WGP) web developers can reuse conceptual models of web applications and adapt them to the need of their own project. The conceptual models of a website genre contain commonly used WDPs.

To further streamline the modeling process in WSDM-Lite a support tool is introduced as a Software as a Service application (SaaS). The application does not only provide the necessary tools to create the various models of a WSDM-Lite project, but also automates a lot of work and improves the overall quality of a design. An e-commerce example project is discussed. Multiple approaches are discussed to automate website generation from these models, each with their own benefits and drawbacks.

The result of the thesis is the new, adapted, WSDM version: WSDM-Lite. This methodology is a more compact version of WSDM that works well with implementations based on CMSs. Patterns, website genres, and a support tool are used to speed up and improve the design process.

Abstract

Het doel van deze thesis is de Web Semantic Design Method (WSDM) methodologie aan te passen aan implementaties die gebruik maken van een Content Management Systeem (CMS). De originele versie van WSDM vereist veel detail in de verschillende modellen, dit strookt met het gebruik van een CMS waarin er vooral met grotere componenten wordt gewerkt zoals bijvoorbeeld installeerbare modules.

In deze thesis wordt een aangepaste methode, WSDM-Lite, voorgesteld. Deze methode maakt gebruik van web design patronen (WDP). WDP zijn conceptuele beschrijvingen van features in een web applicatie. Deze patronen komen overeen met de grootte van installeerbare modules in een CMS. Een totaalbeschrijving van een web applicatie wordt gegeven door deze patronen in een feature assembly diagram te plaatsen.

WSDM-Lite gebruikt ook website genres om het modeleerproces te versnellen: website genre patronen(WGP) zijn herbruikbare conceptuele modellen die gebruikt kunnen worden door de web ontwikkelaar en kunnen aangepast worden aan de noden van het project. Deze WGP bevatten vaak gebruikte WDP in een specifiek website genre. Verder wordt het modeleerproces ook versnelt door een ondersteunende applicatie, een "Software as a Service" (SaaS) applicatie. Deze applicatie bevat niet alleen de nodige tools om de verschillende modellen aan te maken maar automatiseert handelingen van de web ontwikkelaar en heeft invloed op de algemene kwaliteit van het design. Een E-commerce voorbeeld werd uitgewerkt.

Verschillende mogelijkheden, elk met voor en nadelen, werden besproken om de modellen van WSDM-Lite te gebruiken om automatisch de web applicatie deels op te zetten. Het resultaat van deze thesis is een nieuwe WSDM versie: WSDM-Lite. WSDM-Lite is een compacte WSDM versie dat nog steeds de principes volgt van de originele maar gebruikt maakt patronen, website genres en een ondersteunende applicatie.

Acknowledgements

First of all I want to express my gratitude towards the promotor of this thesis, Prof. Dr. Olga De Troyer for all the insights and proof-reads. I also want to thank the Advisor, Pejman Sajjadi, for all the help with the thesis and the critical view towards the contents of it.

I also wish to thank all the lecturers during my time at the VUB who gave me the competences to complete this thesis.

Contents

1 Introduction

In the last two decades the manner in which web applications are built has come a long way. Early systems used an ad hoc development approach but it became soon clear that there was a need for a more disciplined approach. A new field of study "Web Engineering"[20][11] emerged and several web design methods or methodologies were developed such as WebML[6], OOWS[21], WSDM[25], ...

WSDM, introduced by De Troyer and Leune in 1998[25], is a very interesting web design method for two reasons. Firstly, WSDM not only provides the modeling techniques needed to develop web applications, but also a step-by-step approach where each step is based on the results of previous steps, ending up with the final product. Next, WSDM is also using an audience-driven approach[25] rather than data-driven or organization-driven.

The domain of the web is more than any other technology a fast evolving domain. We see a clear trend in the use of CMS-based applications: Drupal[1] had on 10th of May 2014 more than 907.084 installations[2] and WordPress[3]

[1] http://www.drupal.com
[2] https://drupal.org/project/usage/drupal
[3] http://www.wordpress.com

has over 409 million people reading articles[4] each month with a clear growing trend. These are only two of the more popular CMS.

1.1 Problem Statement

Although WSDM is an interesting web development approach, the original method does not consider the use of CMS to implement the web system, while the popularity of CMS-based web applications is only increasing. Moreover, there is a clear mismatch between these two technologies: The levels of abstraction in which WSDM requires the developer to model a web application does not match the level of control one has when using a CMS. Sajjadi [18] already investigated how to adapt WSDM to recent developments in the web domain.

Besides the practical mismatch between the levels of abstraction we also wish to solve a conceptual mismatch: One of the reasons why CMSs are used is because they provide a readymade core with the possibility to extend it with various components. This is beneficial for a fast deployment strategy. Modeling everything from scratch by hand does not match with this approach and web developers would be tempted to omit the models even though they have clearly shown their benefits.

1.2 Thesis Contribution

In this thesis we will discuss a new lightweight WSDM, called WSDM-Light, for the development of web applications using a CMS, while still closely adhere to the principles of the original WSDM. As an extention of this new methodology, we will provide a supporting tool which speeds up the modeling phases even more. Such a tool has several advantages:

- A consistent interface across each type of modelling technique reducing the learning time
- Information collected in earlier modeling phases are used to initialize later phases, using the power of a methodology and speeding up the modeling process.

[4] http://en.wordpress.com/stats/

- Using website genres, models using common structures found across the web are provided.
- A repository containing several conceptual models for a website genre that relate to the setup of different CMSs.

We will also discuss shortly other possible benefits of the tool such as traceability, collaboration, and (partial) website generation.

1.3 Thesis Structure

In chapter 2 we will give background information about technologies used in the rest of the thesis. Chapter 3 discusses related work with pro's and cons of the approaches. Next we will introduce the adapted WSDM methodology, WSDM-Lite. Chapter 5 introduces a support tool for the new WSDM-Lite methodology, of which will the design will be described with the WSDM methodology in chapter 6. An example project using the new methodology and tool can be found in chapter 7. In chapter 8 partial website generation will be discussed based on WSDM-Lite and the support tool. Future work is discussed in chapter 9 and chapter 10 concludes this thesis with a summary.

2 Background

2.1 Original WSDM Version

WSDM stands for Web Semantic Design Method and is an audience driven web development methodology that uses a five-step process starting from a mission specification and ending with the implementation. It provides a clear methodology to develop the web system as the output of each phase is the input for the next. The original WSDM method will be used as basis for the new methodology but will be changed where necessary to meet the goals of this thesis. We will give a short overview of the different steps (figure 2.1) of the original WSDM method [8]. The different phases are illustrated for a fictive e-commerce website.

2.1.1 Mission Statement Specification

The first step that has to be done is reflecting on the purpose of the system and the targeted users. These specifications will be used to make design decisions later on and determine the borders of the project. In the end the effectiveness of the system can be coupled back to these specifications. The mission statement specification is written in natural language and must specify the purpose, subject and target users of the system. The mission statement for the example is given in Table 2.1.

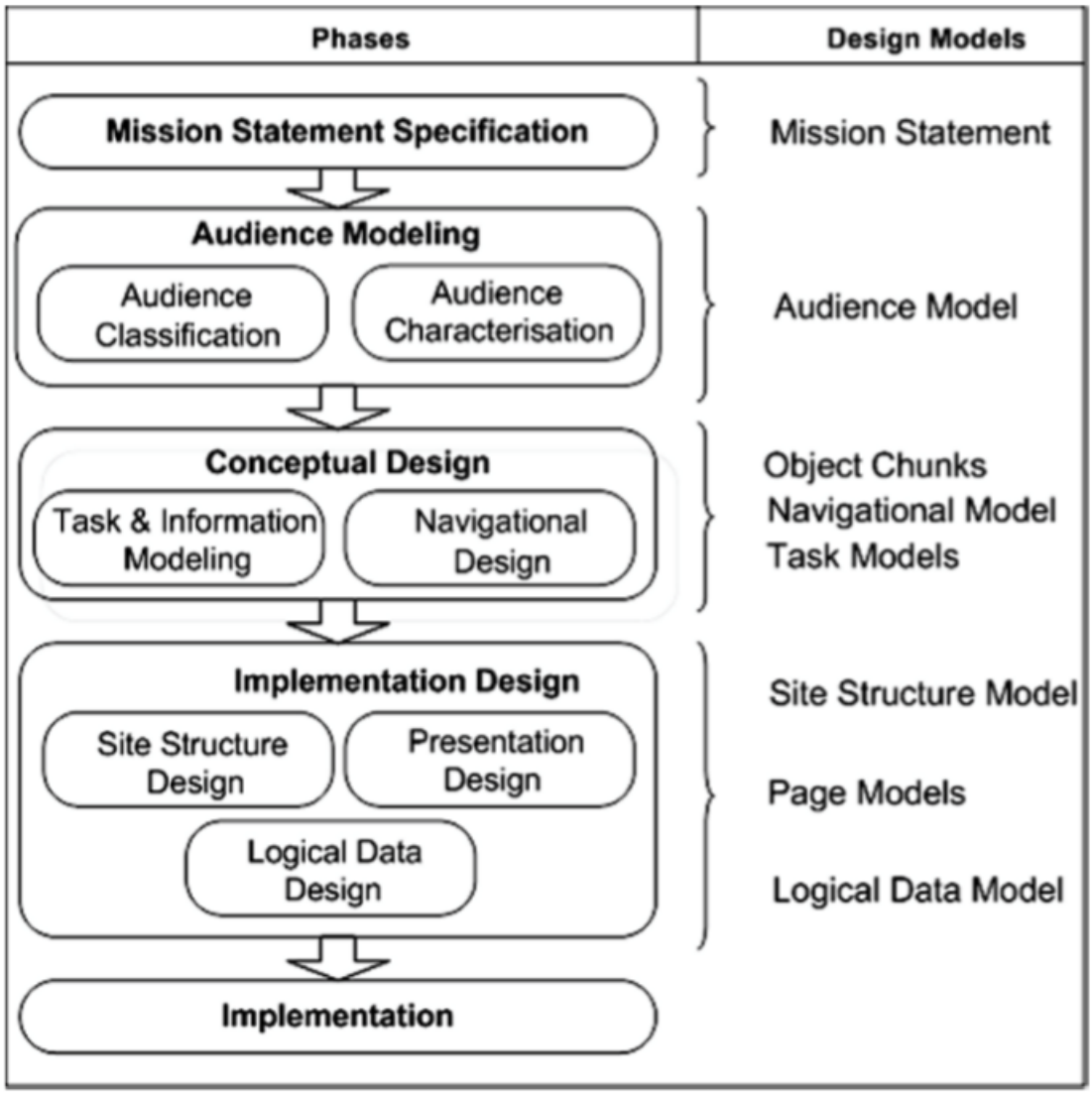

Figure 2.1: WSDM Steps

2.1.2 Audience Modeling

The first phase is a very incomplete description of the system but provides us with a focus where to go. Since WSDM is an audience-driven method the next step is to model the audience of the system. WSDM starts from the requirements of the audience and therefore does not have the problems that come from a poor underlying design or an organization or data driven design [5].This is done in two sub-phases, the first one called "audience classification" and the next "audience characterization".

The purpose of the e-commerce website is to provide a platform for music lovers where they can (1) buy their favorite albums, (2) get background information about the band and their activities and (3) buy merchandising of the bands. The subjects of the website are band albums, merchandising and related information such as biography, discography, concerts, news, reviews, etc. The target users are general music fans that listen to the genres Rock, Poprock, Indierock and Britpop and fans of specific band belonging to this genre.

Table 2.1: Example

Audience Characterization

It is clear that different users of a system have different needs. The audience classification refines the targeted users denoted in step 1 (Mission statement specification) and turns them into audience classes; this classification is based on specific functional and informational requirements. All members of an audience class have the same set of requirements. If different members have different requirements, a new audience class has to be added.

Some audience classes have the same needs as another audience class but need more. WSDM uses audience class subtyping in this situation. An audience class B has the same needs as an audience class A but B has extra needs. The people of the class B belong also to the class A and are a subset of it. In this way there is no redundancy while specifying requirements. The notation used by WSDM in illustrated in Figure 2.2

class subtyping.png

Figure 2.2: Audience class subtyping

The first thing we have to do in the audience classification is identify the people possibly involved and determine if they are part of the target audience

of the system. This is done by first considering the activities of the organization related to the purpose of the system. For the example this results in the following activities:

- Ordering music albums
- Ordering merchandising
- Providing information about bands and their background
- Providing news and reviews about bands and their albums and concerts.

The next thing we have to do is identify the people involved and assign them to target users. This is visualized in figure 2.3.

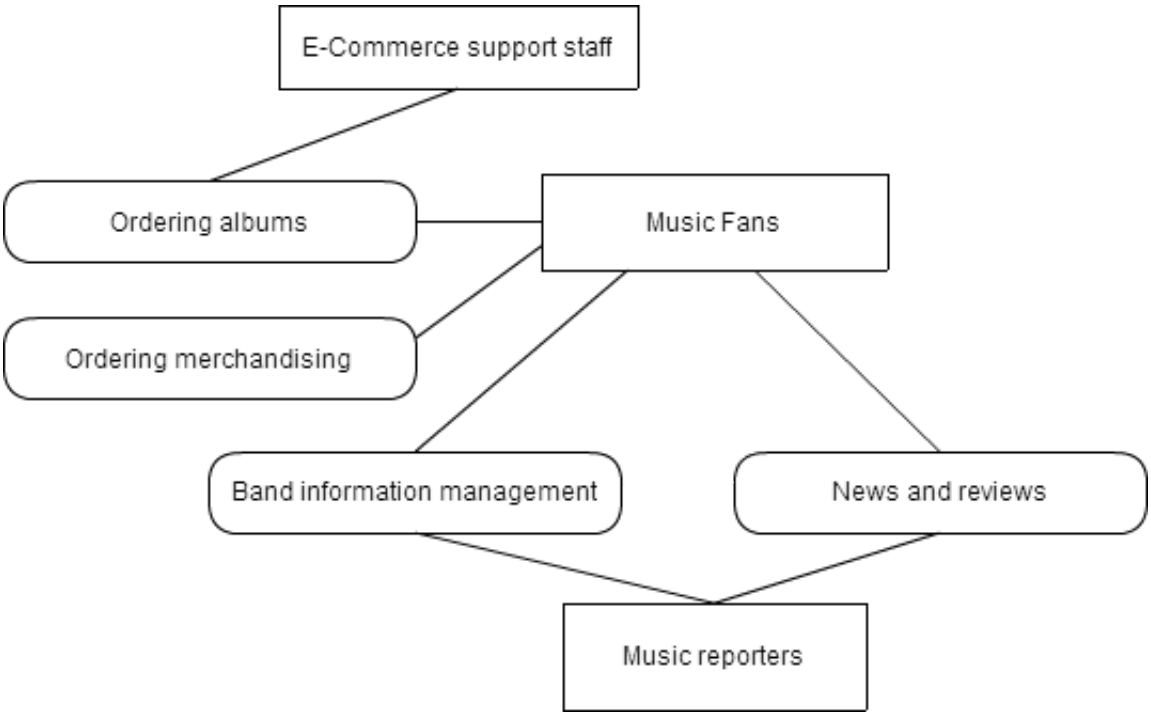

Figure 2.3: Involved activities and target users

Now we need to identify and specify the different requirements for the different user. Because this is an example we will not specify all the requirements in full detail:

- E-Commerce support staff (ESS)

1. Process orders
2. Respond to customer questions
3. Manage the product database
4. Add promotions

- Music Fans (MF)

1. Place order
2. Search for bands
3. Search for albums
4. Search on song titles
5. Browse on genre
6. Listing to an album using a streaming service
7. Pose questions about the products or service

- Music Reporters (MR)

1. Add news about a band
2. Add reviews about a band concert
3. Add reviews about albums

Until now we divided the target users into groups and added requirements to each group. Now we need to classify them into a real audience classes. This can be done by using the matrix requirement method [5]. An NxN matrix, with N the total number of requirements, is created. Every row and column is a representation of a requirement, in the same order. The entry on each position is the answer to the question "Does every user that has the requirement of this row, also have the requirement of this column?". The matrix for our example is given in table 2.2.

The audience classes are then derived from this matrix. Identical rows define an audience class. If the set of "Y" values of an audience class A is a subset of another audience class B, it is its superclass in the audience classification, since B elaborates the requirements of class A. In our example we end up with the audience classification visualized in figure 2.4.

.	Requirement	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ESS.1	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N
2	ESS.2	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N
3	ESS.3	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N
4	ESS.4	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N
5	MF.1	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
6	MF.2	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
7	MF.3	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
8	MF.4	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
9	MF.5	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
10	MF.6	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
11	MF.7	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N
12	MR.1	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
13	MR.2	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
14	MR.3	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 2.2: Example

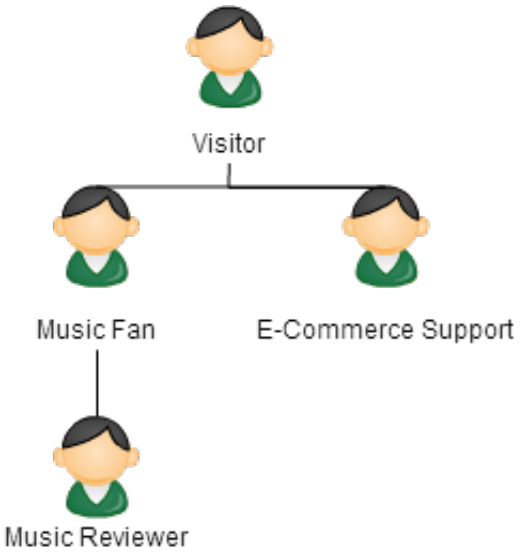

Figure 2.4: Audience Classification

Audience Characterization

In the audience characterization phase we take each audience class derived in the previous phase and specify their characteristics if relevant. Our website will be published in English, so characteristics about English skills are also relevant. The details of the audience characterization can be found in table

2.3.

Music Fan	Music reviewer	E-Commerce support
Understanding of English	Flawless English writing skills, A broad knowledge about music, Adult	Good English writing skills, Good knowledge about the products, Adult

Table 2.3: Example

The audience characterization does not follow the inheritance suggested by the audience classification. This is illustrated by another example: Imagine we are developing a website for a university. The audience classes for this websites are parents which need only information and students which need information and functionality to check courses. The students are in this case a child class of the parents, but the age of the parents (+- 36 or above) is clearly different than the average age of students (+-18 to 25).

2.1.3 Conceptual Design

We now have a better knowledge about the users that will be using our system. Based on the output of the previous phases we can now model our web system in our user-centered methodology. The conceptual design phase consist first of task and information modeling, and then the navigational design. Instead of making a complete model of the tasks and information needed in the system, WSDM starts from the requirements of each audience class and adds small pieces of task and information models to it. Only later on will they be merged into one big global model. This has some advantages described in [8].

Task and Information Modeling

Task Model

The task models are based on ConcurTaskTrees [22] but are modified to better suit web systems [8]. They describe the tasks associated with the requirements on a conceptual level. In our example, the task "Search for albums" is modeled using the adapted CTTs in figure 2.5.

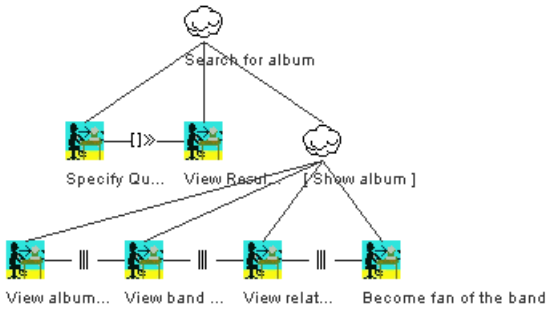

Figure 2.5: CTT model for the task "Search for albums"

Information Model

The information model uses the ORM graphical notation[13] to specify object chunks associated with a task model. For each elementary task such an object chunk must be created. An object chunk is a part of an ORM diagram that describes the information needed by the task at hand. WSDM extended the ORM language to deal with data handling like we see on the web with sending and processing forms[7]. The object chunk related to the "Show album" task can be found in figure 2.6.

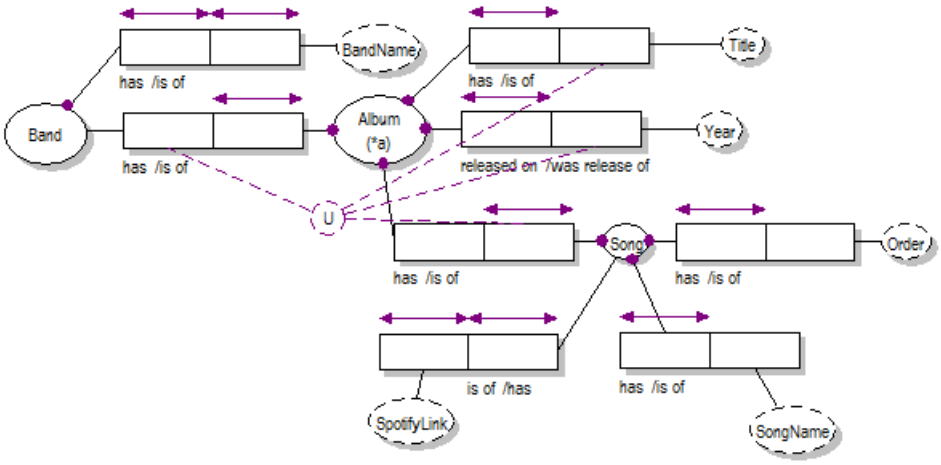

Figure 2.6: Object chunk for the task 'Show album' (IN: *a Album)

Navigational Design

In this step we again depart from the audience classes. For each audience class a navigation track is created to specify how a user can navigate through the system. Such an audience track is actually a subsystem of the whole system that only depicts the information and functionality required by the specific user class. An audience track is derived from the task models created for an audience class. Once all audience tracks are defined, they are combined into a navigational structure by the means of navigational links. The hierarchy of the navigational tracks of the audience classes follows the hierarchy of the audience classes themselves defined in the audience classification phase. The output of this phase is reached when the navigational structure model is augmented with navigational aid links and semantic links and we end up with the navigational model.

This conceptual navigational structure is composed of components and the different links between them. Components are navigational units that group information and functionality conveyed in one or more object chunks [8].

For each task model, a navigational task model is created. An example task model can be found in figure 2.7.

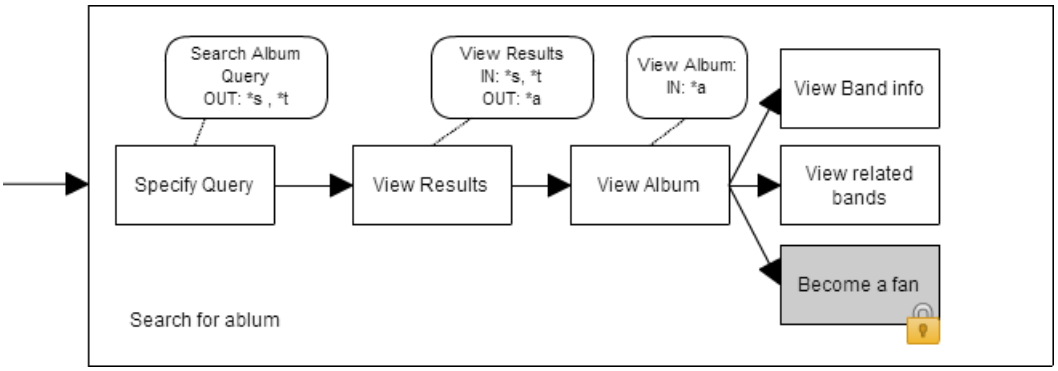

Figure 2.7: Navigational task model

By creating and combining the navigational task models of each task model, a navigational track for the audience class is created (figure 2.8).

After this phase, the different audience class tracks are combined into a single structure called the conceptual navigational structure. This is done

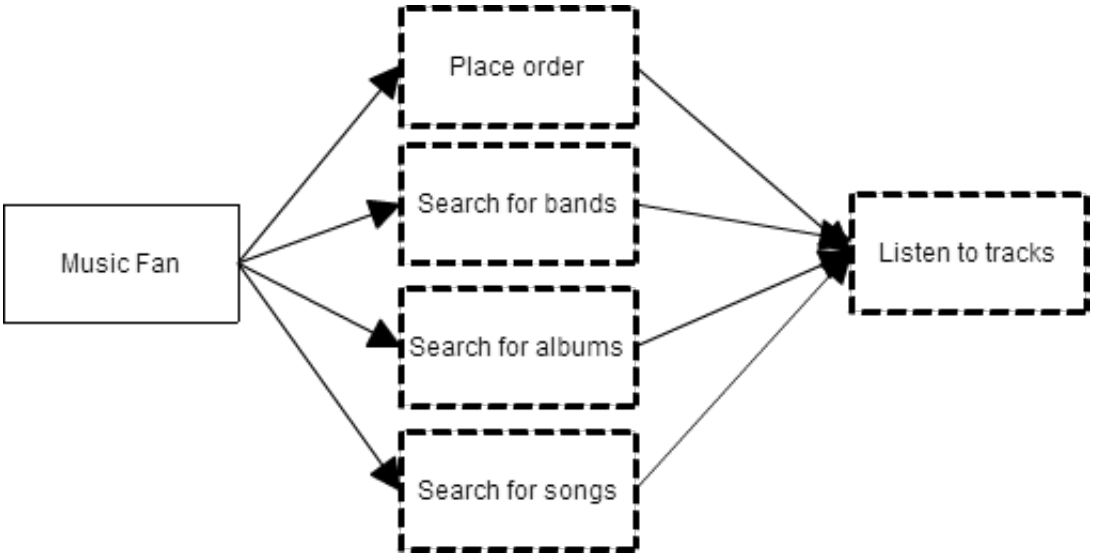

Figure 2.8: Navigational track

using the same hierarchy as the audience classification model. After this semantic and navigational aid links are added.

2.1.4 Implementation Design

We now have reached a complete conceptual design for our web system. To achieve a complete description ready for implementation, the conceptual models have to be extended by implementation details. This is done by defining the site structure design, the presentation design, and the logical design.

Site Structure Design

The site structure design model is an extension of the conceptual navigational model and specifies how the different components should be grouped on pages. Graphically, a page symbol is drawn around one or more components to specify they should be on the same page. These pages are abstract pages because they can adapt to data. For example: the content of the show album page is different per album.

In the case of our example scenario, each component is put on a different page but the search functions are placed on the same page (figure 2.9).

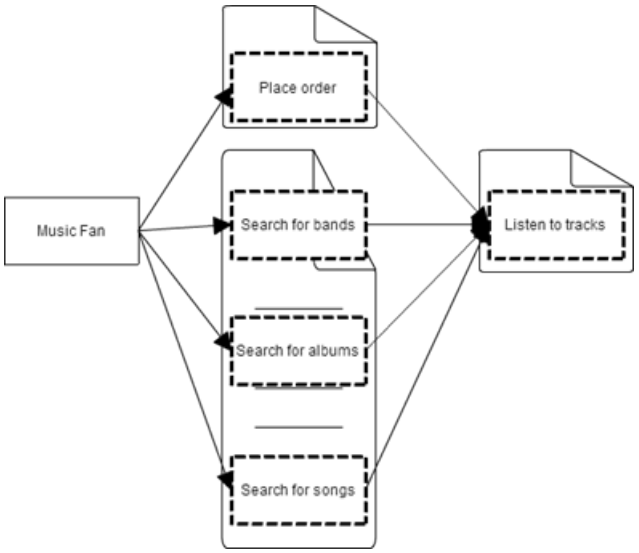

Figure 2.9: Site structure

Presentation Design

The web is a very visual environment and until now we did not yet specify any visualization details. In this phase, the look and feel of the entire web system is defined as well as standard templates for the different pages defined in the site structure design. Some pages may look alike and others may differ. Therefore different templates may be created for the different pages. Templates ensure a consistent look and feel of the web system.

A template can consist of a header, footer, sidebars, ... but must at least have one editable region where the content of the page can be filled in. In this editable region the concepts of the components are placed using a grid system with rows of cells. Content can thus be placed in these cells.

In this phase also a style needs to be provided ensuring consistency across the system. This is done by using a Cascaded Style Sheet (CSS). The output of the presentation design is thus a set of templates, linked to the different pages of the site structure design and a style.

Logical Design

If there is no existing data source available a complete data model has to be constructed using the different object chunks defined in the conceptual

phase. This complete data model can then be transformed to, for example, a relational database model using the 7STEP algorithm [14].

2.1.5 Implementation

In this phase the actual web system is implemented based on the different models created in the previous phases. This can be done by hand coding or by using web implementation frameworks or tools.

2.2 Feature Assembly Framework

Feature assembly [1] [30] is a high level modeling technique that finds its origins in software product lines. A software product line is a group of closely related programs that share some components or so called features. A feature diagram is a visual representation of such a program in terms of its features and the relation between these feature.

Feature assembly is used to model variability in software products of a same family. When we look at websites we see that websites of a certain type have a common base and only differ in some features. We can say that a software family is closely related to a genre of websites and that these genres vary in features but have a lot of commonalities. Websites genres will be discussed in more detail in the next chapter.

Features are high level descriptions of components in a software system and different types of relationships between features can be expressed. Feature models are usable in our WSDM-Lite version because they hold the level of abstraction we are looking for in our adapted version. In Feature Assembly, relationships between features are categorized as:
Decomposition relations:

- Mandatory - the subfeature(s) are required, and
- Optional - the subfeatures are optional

Specification relations:

- An abstract feature can have different option features. A cardinality m:n indicates the minimum (m) and the maximum number of option features (n) to be selected for the abstract feature.

Constraints:

- Exclusion - the two features cannot be selected in the same product,
- Required - the selection of one product requires the selection of another product.

In figure 2.10 an example is given from [1]

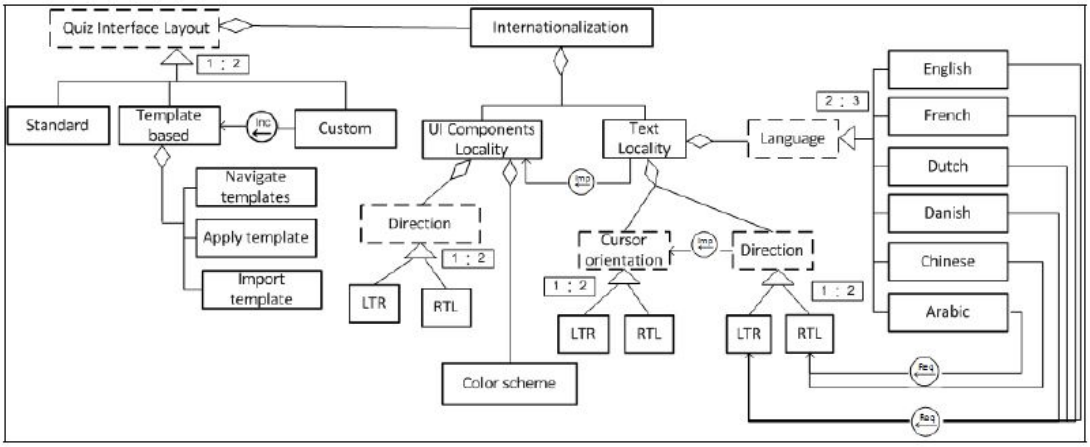

Figure 2.10: Feature Assembly Diagram Example

The feature assembly diagram shows an overview of the complete system. The problem with using the feature assembly diagram for web applications as-is is not enough as demonstrated by the example specified in figure 2.10: a feature 'Import template' is shown. By seeing the name of the feature we get a general idea what the feature will do but it does not contain enough information to attach an implementation to this. Questions such as 'what information does a template contain?' and 'in what format can a template be imported' are not answered in the diagram. More information will need to be attached to a feature, but the feature assembly diagram is a good start to model a general overview of a web application. There exist many graphical notations for feature diagrams which are interchangeable in the WSDM-Lite methodology. In this thesis we will use the feature assembly notation.

3
Related work

In this chapter we will look at related work. We will first discuss the WCMS based WSDM version proposed by Sajjadi [18] which will be the basis of the new WSDM-Lite methodology. Next we will discuss the Web Engineering Method (WEM) which is one of the only methodologies designed for CMSs. Next we will discuss WebRatio, which is a support tool for the WebML[6] method. Last we will discuss website genres which will play a large role in initializing models for the WSDM-Lite methodology.

3.1 WCMS Based WSDM Version

An alternative version of the legacy WSDM for CMS was already proposed by Sajjadi [18]. The version, called WCMS based WSDM, follows roughly the same methodology as the legacy WSDM but is modified to the needs of CMS-Based web development. The different steps of this methodology are illustrated in figure 3.1. The method uses web design patterns as building blocks for the conceptual design of the application. These web design patterns are incorporated in to a feature assembly diagram to design, on a high conceptual level, the system.

These patterns are classified in several possible website genres. By deciding on a website genre early in the methodology the web developer can select,

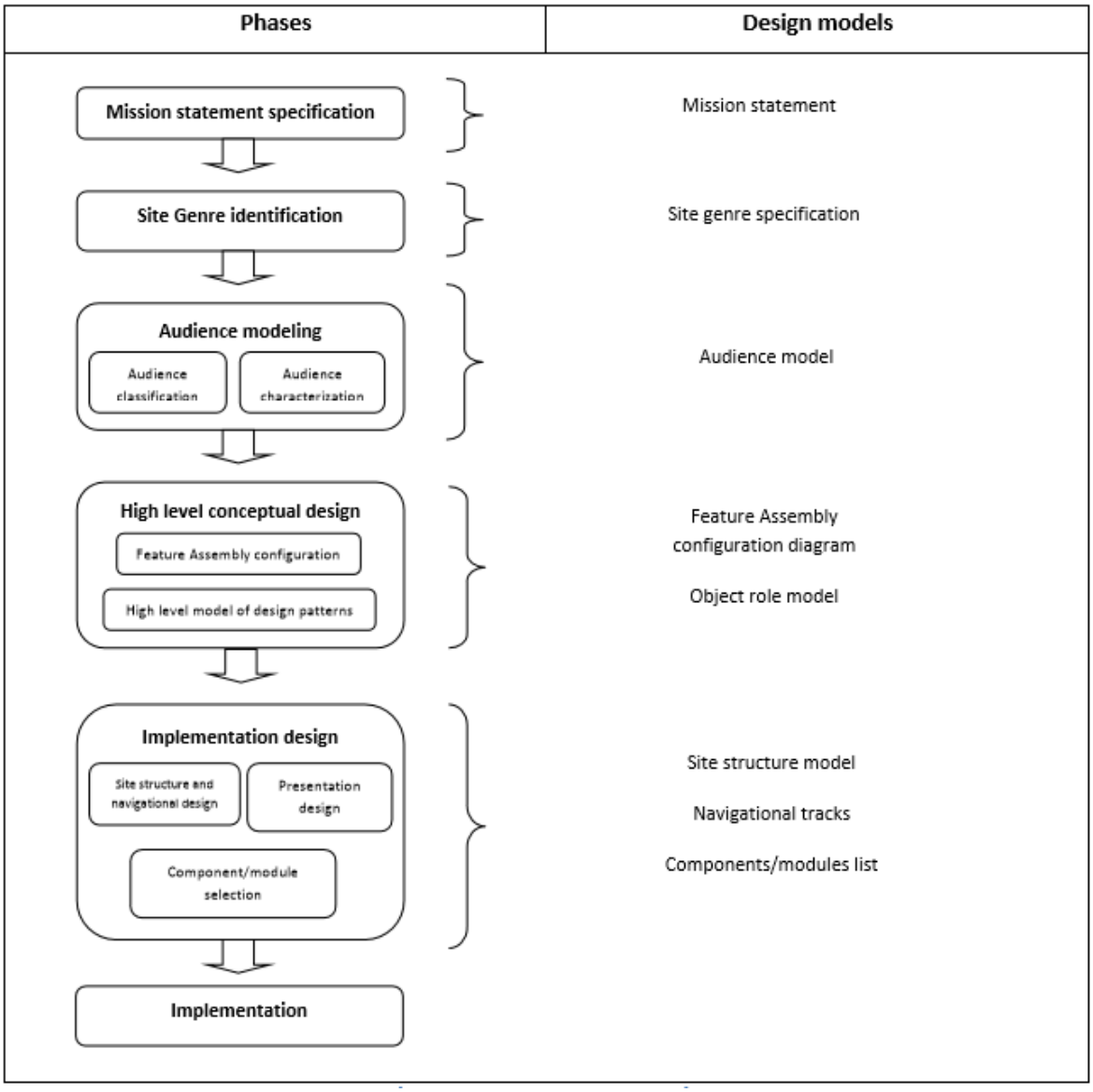

Figure 3.1: Phases of WCMS-Based WSDM

but is not limited to, existing web design patterns. A predefined feature assembly diagram per website genre can be used to initialize the conceptual design. This approach is based on the fact that recognition is much easier than recall. The idea behind the proposed method is sound but it still lacks depth and needs more elaboration. As discussed in the background chapted a feature assembly diagram alone is not enough to give a conceptual design of a system. The combination of features in the feature assembly diagram

and the proposed high level design patterns in the WCMS based WSDM is a good suggestion but there is not yet a clear definition of such patterns. We will propose a definition of a web design pattern that can be used as features in the feature assembly diagram containing enough information to conceptually model the system.

The WCMS based WSDM suggests only one feature assembly diagram for the whole system. When we look at for example an E-Commerce website, we can have customers and support staff audience classes. Both audience classes need for example a feature 'browse products' but a customer needs a feature 'shopping cart' while the support staff needs 'order processing'. Different feature assembly diagrams per audience class seems more suitable in this situating. By using the inheritance of the audience class hierarchy we can develop a smart way that eliminates most of the content duplication that would arise in these situations.

The WCMS based WSDM version suggests only one feature assembly diagram per website genre that contains mandatory features with the core of the website genre and optional features to be selected by the website designer to model the required system. We believe a more open approach will allow the method to follow the quickly evolving world wide web: a website genre can have multiple feature assembly diagrams of which the web designer can select one that relates the most to the required system. There will also be no distinction between mandatory and optional features: a feature that is considered here as 'mandatory' has a high probability of being present in the feature assembly diagrams of the website genre, but we cannot foresee all the cases in which a feature must or must not be present. Therefore we will not put restrictions on this.

Both the original WSDM version as the WCMS based WSDM version display phase evolutions in multiple dimensions. The high level conceptual design contains both the feature assembly configuration and the high level model of design patterns. We believe that a linear progress of consecutive phases is easier to follow and such an approach will be used in the adapted WSDM version.

The phase 'Component/Module selection' will not be a part of the adapted WSDM version since we believe that phase is part of the implementation phase rather than the implementation design: in the original WSDM version coding is part of the implementation phase. When we use a CMS we ac-

tually select code that was already created and is grouped inside a module. By installing such a module we actually add code to the application. The difference is that this code wasn't written by the designer himself.

3.2 WEM

The GX Web Engineering method (WEM) [28] was developed at the company GX and is specifically designed for the GX WebManager CMS. The method is based by combining techniques from other web or software development techniques such as UML-Based Web Engineering and the Unified software development process.

Because it was developed in a company it has some unique considerations that make it well-suitable for business environments such as proposal construction for client validation which act as some kind of requirement analysis, and based on this module selection is done.

Although the method seems very business-oriented the activities relate very closely to general software development phases. However, because it is based on the modules of a specific CMS of one company, this approach is to narrow for universal use.

3.3 WebRatio

WebRatio[1][2] is a software program based on the WebML[2][6] web design method. WebML is not a CMS specific approach but the fact that it has also a tool supporting it makes it interesting for us. WebRatio is an Eclipse-based environment to develop models for the web project (see figure 3.2). WebML is not a true methodology, i.e. WebRatio does not follow a step-by-step approach but a project is divided is several 'folders' and each folder contains specific models. WebML is now evolving (and WebRatio with it) to support the IFML[12] (Interaction Flow Modeling Language) standard for visual modeling language. Based on the IFML model, WebRatio can generate a full web application in Java code.

[1] www.webratio.com
[2] www.webml.org

IFML is becoming a more populair technique and holds some interesting properties. It is a platform independent interaction flow modeling technique that allows us to model different views and connections between them activated by triggers. It focuses on the 'view' part of an MVC application which a lot of web applications use. It references controller actions and model information to complete the design.

IFML contains techniques to deal with more recent developments in rich internet applications (RIA). Containers can be nested and in case of an XOR view the different containers can be translated in the implementation as tabbed panes.

IFML is on itself a good way to design a web application because it is independent of the actual implementation. IFML however talks about 'views' and 'container' which roughly map 1-to-1 with pages and elements on web pages. The philosophy behind the conceptual phase of WSDM is that we model the tasks and information at hand and not how these tasks are translated into a page-like structure. Therefore IFML is not suitable to use in the conceptual phase of a WSDM version. An IFML model could be added to a WSDM version in the implementation phase. Adding an IFML model could be useful to better connect the more abstract models of the conceptual phase and an actual implementation. However, in WSDM-Lite we higher the level of abstraction and do not which to talk about details happening on every page. Therefore, IFML has no place in our new WSDM-Version.

WebRatio is a commercial application and provides a broad set of extra functionality that saves the developer a lot of work. However, WebRatio is not implementation independant so it is strongly advisable to use the generated Java code. Once the code is generated, it is still advisable to continue using WebRatio to update the design of the web application and generate new code.

3.4 Website Genres

Sajjadi[18] observed that many websites are based on similar patterns and more and more, websites can be categorized[23] in different genres. However, the categorizations of these genres are based on vague descriptions and some genres may overlap, but Sajjadi argued that it could be useful to identify certain patterns that we see in these genres and reuse them when we are creating a website belonging to this genre.

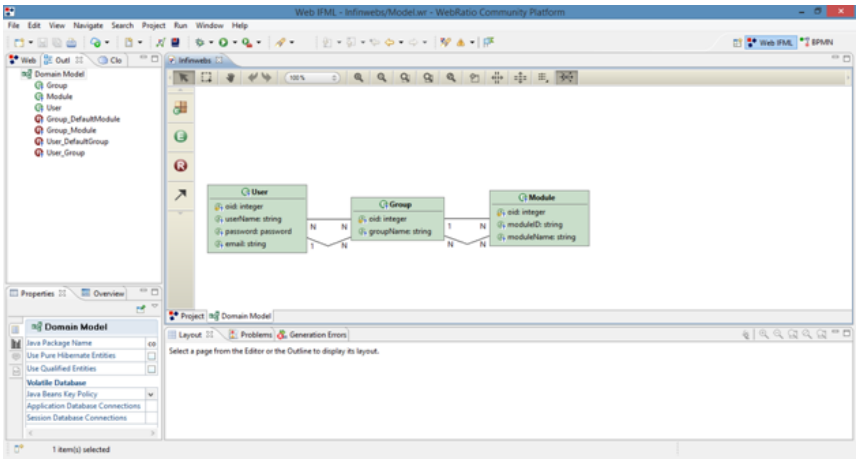

Figure 3.2: WebRatio screenshot

Sajjadi identified several common website genres. In the adapted WSDM version we do not wish to limit ourselves to a fixes set of website genres. Instead we let web designers define their own website genres and commonalities in them. When there are a large amount of projects being created these website genres will start to emerge. Designs for web applications that are often being reused imply that these web applications belong to a common website genre. The work of Sajjadi will not be used as a catagorization for projects but as possible designs to start from. The catagorization of web applications in WSDM-Lite will use the principle of a folksonomy: possible website genres will emerge out of the projects already done with the WSDM-Lite methodology.

In previous paragraph we mentioned that web applications of a same genre have commonalities between them. This implies that websites have common features but still differ. It is impossible to define a fixed set of commonalities that all websites in a website genre share and which are optional. We cannot model a modal web application and use it as basis for all applications in that genre. To account for we must allow a website genre to be modeled in different ways.

As an example website genre, we take e-commerce websites. We can identify several patterns for such as a homepage portal, product overviews, filters, checkout, ...[26] . If there are a lot of website genres containing only a product overview and others contain every pattern. This will result in two populair designs for the e-commerce website genre and other designs. Other

designs that are less popular will be displayed with less importance (for example: in an ordered list). Different designs for one website genre will need to have a rating to weed out the bad designs and let the good designs emerge.

By using a folksonomy for the website genre and a rating for the designs of these website genres, the methodology is robust enough to follow trends on the web without changing the methodology itself. If new trends emerge and a new genre of websites pops up, the folksonomy will evolve with it and the new trend will show up as a website genre.

4
WSDM-Lite

In this chapter we introduce the new WSDM-Lite methodology. We will explain in detail what is changed and why it is changed. Keep in mind that the main purpose of the changes introduced are the need to adopt the level of abstraction to the use of CMS systems. The WSDM-Lite methodology consists of following phases:

1. Mission statement specification
2. Website genre selection (new)
3. Audience modeling
4. Conceptual modeling (adapted)
5. Navigational modeling (adapted)
6. Structural modeling (adapted)
7. Template modeling (adapted)

4.1 Mission Statement Specification

The mission statement specification phase defines the boundaries of a project and is not directly related to the implementation by means of a CMS. There-

fore this phase will remain the same as in the legacy WSDM method and requests the web developer to specify the purpose, subject and target users of the system.

4.2 Website Genre Selection

Using the information we reviewed in the related work, we saw that there exist website genres that have a lot of commonalities. More and more, websites belong to a website genre. By specifying a website genre we define the scope of the project, specified in the mission statement specification, further.

Using this genre we can look up specifications in some kind of repository and reuse work of others to speed up our process.

4.3 Audience Classification and Characterization

The WSDM-Lite method is still an audience-driven methodology so the audience modeling is still an early and important phase in the method. The audience of the website is already specified at a high level of abstraction and is not affected by the use of a CMS later on. Therefore the audience modeling phase remains the same as in the legacy WSDM version.

4.4 Conceptual Modeling phase

The purpose of the conceptual modeling is to design the website independent of the actual implementation. We create models to specify how our website will be structured but these models should not be based on any CMS that could be used to implement the website. In the original WSDM version, there were very detailed models of elementary tasks and the data these tasks were used. When these models are used to implement a web application based on a CMS, we have a mismatch between the level of detail in our models, and the level of control we have when installing modules or components. Therefore it is useless to model components in detail while we will not change the ready-made components of a CMS. To solve this, we propose conceptual models that do take into account that a CMS will be used to implement the website, but we abstract from which CMS we will use. Therefore, we model

possible CMS components conceptually as web design patterns.

Furthermore, we keep the concept of web site genre. But, instead of providing a model for each website genre, i.e., composed of web design patterns and relations between them (as done by Sajjadi[18]), we also consider them as patterns. We call these patterns, Website Genre Patterns.

We first discuss the different types of patterns used and how we represent them. Next, we describe the adapted conceptual modeling phase.

4.4.1 Patterns

Website Genres and Patterns

As stated by Sajjadi[18] and explained in the related work in section 3.1, a web design pattern can be seen as a feature in the feature assembly diagram and the website genre as a product line. We can even go further because some patterns can belong to many genres. For example: A login can belong to an e-commerce website but also to a community website. In such a case we better consider all websites as a product line and use the website genres merely as a classification for patterns, where patterns can belong to many classeswebsite genres.

Web Design Pattern (WDP)

Every CMS is using the concept of modules, components, plug-ins, ... In this way, CMSs are extensible and customizable. Just like in the legacy WSDM methodology we first model the structure of the website at a conceptual level, meaning that we consider these modules without considering their actual implementation. A conceptual 'module' can be called a Web Design Patterns (WDP) (Sajjadi). It is possible that we need more than one actual module to implement a WDP, or that none exist and we have to implement it ourselves. It is up to the person who implements the system to find a good match and adopt where necessary.

There are two extreme cases we need to consider when conceptually modeling a pattern:

1. It is not always possible or needed to model details when using a CMS. For example: If we want to use a "login" functionality, and we will use the functionality provided by the CMS, it is not needed to model this functionality in detail.

2. It is possible that a very specific module we want is not yet available. In this case there is other choice than to implement it.

These two extreme cases show us that the level in which we need to model a pattern may differ greatly when using a CMS. Our pattern definition will take this into account by providing optional specifications which can be omitted in case 1 and added in case 2. It is also possible to omit details in a first phase of the design and add details later on when it has been decided upon the CMS to use for the implementation and the limitations and functionality provided by the CMS are known.

A web design pattern consists of following specifications:

- **Pattern name (required):** The name of the pattern is a unique identifier within the boundaries of the project.
- **Website genre (optional):** The name(s) of the website genre(s) the pattern belongs to.
- **Problem (required):** What problem the pattern solves.
- **Solution (required):** The solution the patterns offers to solve the problem.
- **When to use (optional):** (cfr. "context" Le Van Huyen [17]) Descriptions or situations in which the pattern can be applied.
- **Warnings (optional):** (cfr. "problems" Sajjadi[18]) Issues that may arise when using this pattern.
- **Rational (required):** Describes how the goal is accomplished by the pattern.
- **Structure (optional):** More concrete information is specified here about the navigation structure (CTTs) and object chunks (ORM Diagrams).
- **Related patterns (optional):** A list of related patterns can also be presented. This gives the web developer the possibility to have associative[4] navigation next to the top-down navigation from the website genres. For example: when we browse to the list of WDP of the e-commerce website genre and we look at the WDP 'Cross-selling', the WDP 'Up-Selling' will be a related pattern because they are both techniques used to sell more to a customer.

The data model (ORM-notation) of a web design pattern can be seen in figure 4.1

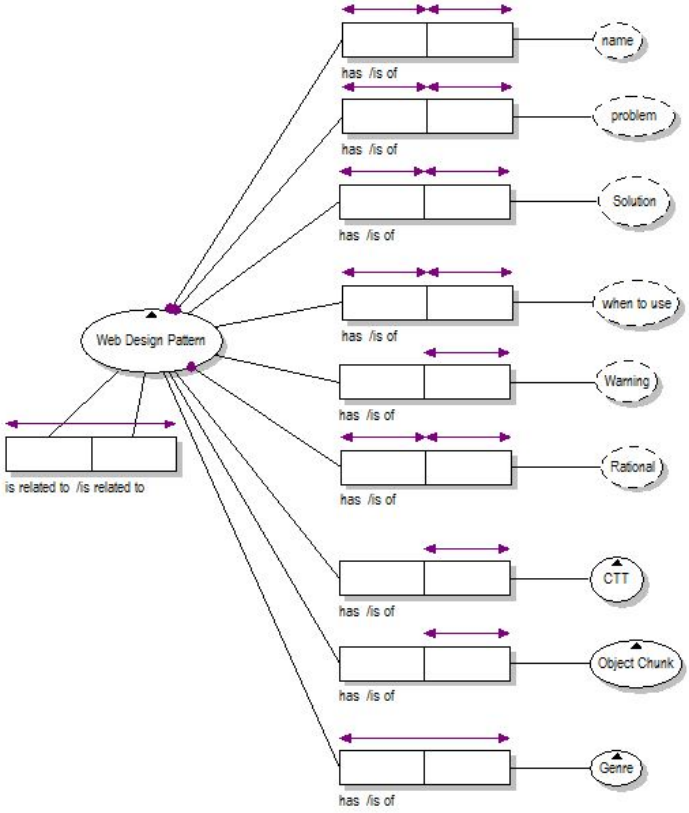

Figure 4.1: ORM information model of a WDP

Reusing Web Design Patterns

An advantage of patterns is that they can be reused in as long as they are accessible and searchable. When a pattern is stored in a repository that can be searched by a community, the pattern definition could be extended with a more elaborate human readable description, tags, rating, commenting, etc.

Note that a pattern does not always need to be reused as-is but can be modified in the context of a web project.

Designing New Patterns

Everyone is free to design patterns. If they are not relevant for other people they can be kept private to the web project; when they are relevant for other

people they can be shared and the community can decide on the importance and quality of the pattern by e.g., some kind of rating system.

Website Genre Patterns (WGP)

In the second phase of the WSDM-Lite methodology, a website genre is selected. Based on this website genre, patterns common to this website genre can be proposed to the designer. For example: for the e-commerce website genre, the patterns products, checkout, payment terminal and many more can be proposed. Some web design patterns will require others or exclude some other patterns.

Web design patterns with relations between them can also be seen as patterns but on a higher level. Therefore, to capture this idea, we introduce the concept of Website Genre Patterns (WGP) and provide a definition.
To express the relationships between the web design patterns in a Website Genre Pattern, we propose to use Feature Assembly technique. Features are used to represent the web design patterns. The feature will hold the pattern's name as label, and the information about the pattern can be stored inside the feature. The relationships between the features are expressed as relationships and constraints.

A WGP contains the following information:

- **Website genre name (required):** A WGP belongs to a website genre. The name of the website genre is actually the website genre for which te pattern is created (For example: E-commerce), but the actual 'genre' of the website is not a clearly defined set of genres but rather the combinations of WGP with the same website genre name.
- **Feature Assembly Diagram (required):** A WGP contains a set of WDP (features) and relations between them specified using the feature assembly technique.
- **Description (optional):** A human readable description of the modeled system.
- **Related CMS (optional):** Website Genre Patterns that are being reused imply that they are already used before. Therefore at least one developer has implemented this conceptual model and chosen a suitable CMS. By adding this information to the WGP it can save a developer time during the implementation phase of the project.

4.4.2 Specifying the Conceptual Models

Based on the website genre, specified in the second phase of the method, the WGP associated to this website genre is proposed to the designer. As we represent the WGP by means of a feature model, the designer can now select the required WDPs (features) with respect to the relationships and constraints specified. However, we want to provided the necessarily flexibility. Therefore, this (configuration) diagram is only used as a starting point for the web developer and may be modified or completed.

Different Audience Classes

Different audience classes may not have access to each feature in our design. Therefore, and similar as in the legacy WSDM, this needs to be specified. Making a different configuration diagram for each audience class can accomplish this. However, we also explore the subclass properties of audience classes stating that a sub-audience class inherits the requirements from its parent audience class(es). By also using inheritance on the models we can elaborate the diagram of class A with the features for class B when B is a sub-audience-class of A. This makes sense because a class B is a subclass of A if it has the same requirements as A, but more. Therefore, class B should have at least the same features as A. We use the e-commerce example to illustrate this: figure 4.2 has only the registration and login WDP, while figure 4.3 also has various product features.

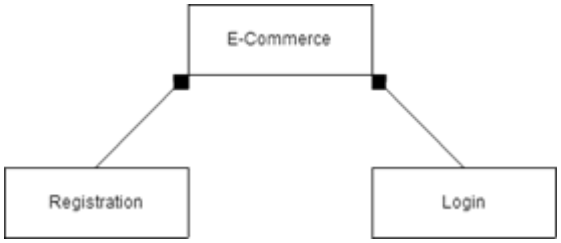

Figure 4.2: Conceptual model of parent audience class

4.5 Navigational Design

In the navigational design of the legacy WSDM, the links between all tasks in the system were modeled. Because, in WSDM-lite, we do not model each task in detail in the conceptual phase, but use patterns, we cannot do this in the same way as in the navigational design of the legacy WSDM where

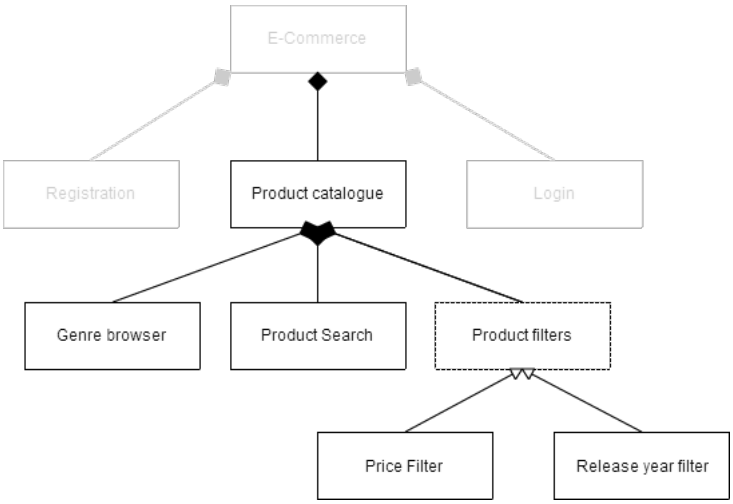

Figure 4.3: Conceptual model of child audience class

we created a task navigational model component for each individual task. The purpose of this task navigational model was to express the process logic (i.e., workflow) of the task. In these navigational models, several components could be grouped into transactions. Very often, these transactions were necessary to group tasks that have a single goal (e.g., paying a purchase).

In WSDM-lite, we have modified the navigational design phase such that it omits the creation of atomic components (corresponding to atomic tasks) but takes the features of the conceptual phase as building blocks for the navigational design. This avoids the need to model process logic between atomic tasks and transaction, which will usually be cover by a module in the CMS. In practice we only need to take the feature assembly configuration model for each audience class, remove the edges between the features and replace them by links of the navigational design. We can also apply the inheritance of the feature assembly diagrams to the navigational design. An example is given in figure 4.4

4.6 Site Structure Design

In the structural design, which was called the implementation phase in the legacy WSDM, the conceptual design is taken as starting point and the features are grouped onto pages. Visually we take our navigational model literally and draw pages around features that should be on the same page. The same inheritance structure is used as in the conceptual and the navigational

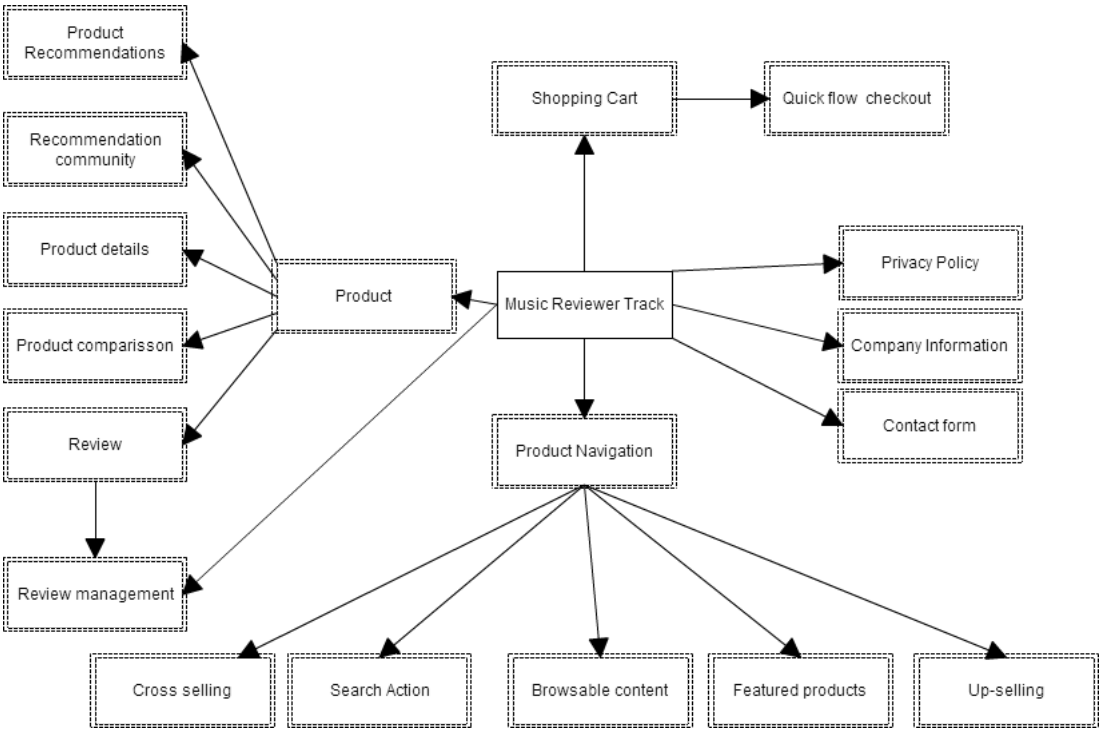

Figure 4.4: Example of a navigational design

modeling phase. There are three possible scenarios:

1. A feature maps exactly to one page, then we draw one page around the feature.

2. Several features belong to the same page (for example: search band, search album, ...), then we draw a page so that it contains all these features.

3. It is also possible that a feature is spread across multiple pages (if it a complex process or it contains a lot of information). To keep our model clean, we will not draw several pages around this single feature but just draw one page, and indicate that multiple pages are possible by writing a * in the upper right corner. This relates to the notations in regular expressions. An example is given in figure 4.5.

4.7 Template Design

A web application needs to have a consistent look and feel. To ensure this, we design page templates that are used in the different pages defined in

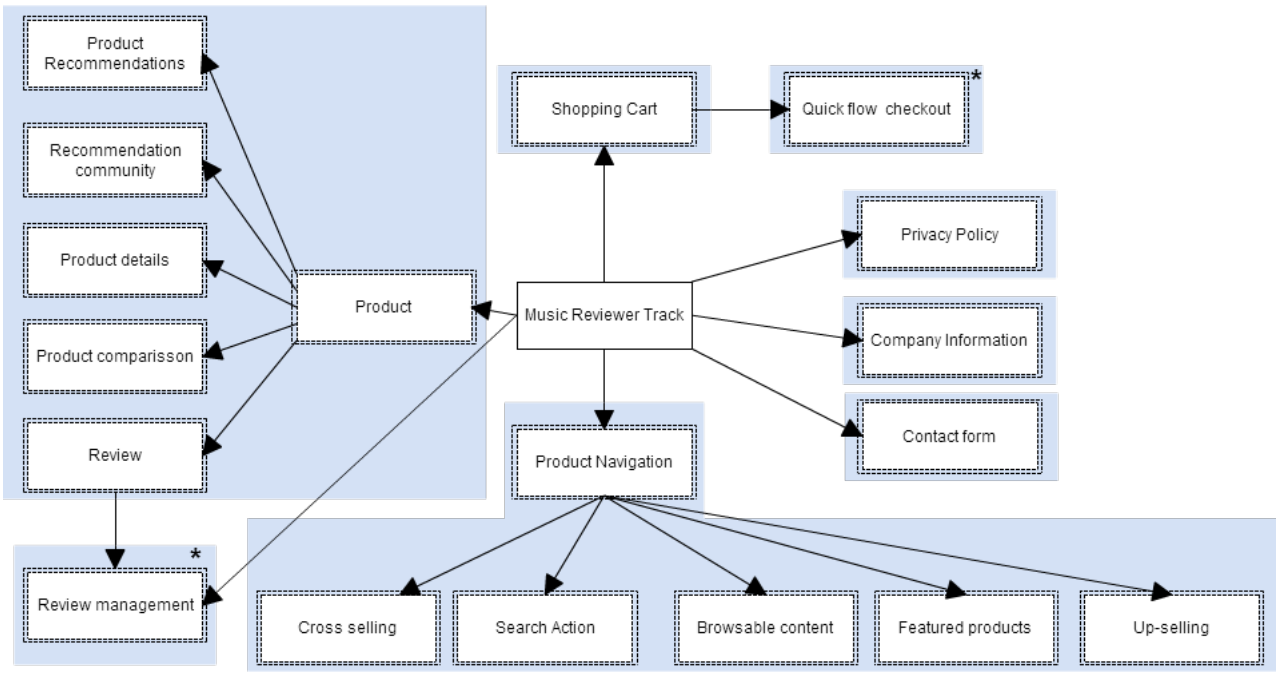

Figure 4.5: Example of a site structure design

the previous phase. There are two levels of detail on which we can design templates. It is up to the web developer to choose one of these two:

1. Each CMS uses a template system of its own. The templates are already programmed in HTML, JavaScript, etc. and the templates (or elements of templates) are then used appropriately by the CMS on the correct pages.

2. Already programming a template for the CMS can be too early for this design stage. Therefore we can still use the template approach of the original WSDM version. The high-level presentation design concepts(tables, lists, ...), forms and events are not available in the template design of WSDM-Lite because the data for these concepts is not modeled at this level of detail. Grids with rows and cells will still be used but the cells can be designed freely as the web developer chooses.

5

WSDM-Lite tool support

To further streamline the methodology WSDM-Lite users would benefit of having a support tool which helps them in various ways to speed up the design process and even improve the design itself by enforcing consistency. We will now discuss various functionalities such a tool may have and the benefits for the method. In the next chapter we will describe the implementation of a proof of concept of such a tool.

5.1 Initialization

In WSDM-Lite the models of one phase are based on the information collected in a previous phase. This means that when a designer finishes a phase and goes to the next, the tool can use the results of the phase to initialize the next phase. We will give an overview in which cases this is possible.

5.1.1 Mission Statement Specification > Audience Classification

In the mission statement specification we have to specify the target users. The target users are not yet formulated as true audience classes, as this is part of the audience classification phase. However, in the audience classification phase we can use these preliminary identified target user groups to assign

requirements to them. Next, Afterwards, the requirement matrix can be created and the audience classes can be derived. This ensures that all targets users identified in the mission statement will be taken into consideration. Identifying new target users, during audience classification, should force a revision of the mission statement.

5.1.2 Audience Classification > Audience Characterization, Conceptual Design, Navigational Design and Site Structure Design

The result of the audience classification phase is a set of audience classes structured in an audience hierarchy. The audience characterization and the models of the conceptual design, the navigational design and site structure design need to be specified for each audience class. Based on the audience classification we can already initialize these phases to account for the different audience classes.

5.1.3 Website Genre Selection > Conceptual Design

The website genre selection and the conceptual design are not directly consecutive phases, but the conceptual design is based on the website genre selection. The idea behind the website genre selection is that websites from this genre have a lot of commonalities. These commonalities can be seen as a website pattern and as a sets of possible web design patterns, modeled in a feature assembly diagram. Therefore, the feature model associated with the web site genre selected in the website genre selection can be presented to allow the web designer to create a configuration model from this feature model, and possibly extend or adapt it as relevant for his web application.

5.1.4 Conceptual Design > Navigational Design

In the previous chapter we discussed that, in WSDM-lite, the navigation model can easily be derived from the feature assembly configuration models of the different audience classes. Therefore, the tool can initialize the navigational design with the features of those conceptual design models. The only thing left to do is specify the links between these features. As in the legacy WSDM, the overall navigational design of the website can be derived from the audience hierarchy. Also here, the tool can generate a first design that the web designer can adapt if needed.

5.1.5 Navigational Design > Site Structure Design

The site structure design uses the navigational design to assigns the features to pages. The site structure design only adds extra information to this navigational model. So, the navigational design can be directly copied to an initial site structure design.

5.2 Repositories for WGP and WDP

In previous chapter, we introduced the concept of Web Design Patterns (WDP) and Website Genre Patterns (WGP). The power of patterns is that other web developers can be reused them. For this, we need a means to share already created patterns with other developers.

5.2.1 WGP Repository

There can exist various website genre patterns for each website genre. When the conceptual modeling phase is initialized, the web developer can pick one of the available WGP from the WGP repository. When the web developer cannot find a WGP to suit his needs, he can opt to not pick a WGP and create a conceptual model from zero. If he wishes, he can share this new conceptual model as WGP in the WGP repository.

5.2.2 General WDP Repository

The WDP repository must be available during the entire conceptual design phase. When the web designer starts with a WGP or is creating one from scratch, he should be able to add new features (WDP) to the feature assembly diagram. Other designers may already have designed suitable WDP (for example: a shopping cart), so by browsing the WDP repository these WDPs can be added easily. When a WDP is not yet available, the developer can add it to the repository when relevant.

5.2.3 Local WDP Repository

Some features used in a web design, may be specific to the application under consideration and should therefore not be added to the general WDP repository. However, there could also be a need to be able to reuse these local WDPs. Consider, for example, the audience classification in figure 5.1 . Both C and D extend the feature assembly diagram in the conceptual phase of audience class B. F extends the diagram of E. When we add for example

the feature 'Premium support' for the audience class C, it is not accessible in E or F because the feature is not present in a common parent. To avoid having to duplicate the feature to be able to use it in E or F, we introducing a local WDP repository were we can access all features used in the current design.

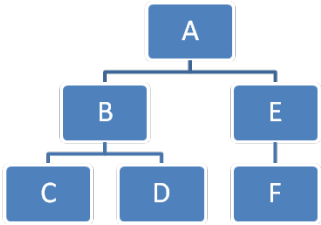

Figure 5.1: Example audience classification hierarchy for local WDP repository use

5.2.4 Maintaining Repository Quality

By allowing everyone to add patterns in the repository we make sure we have a wide range of WDP available. This will however also imply that people can add patterns that are badly designed or too specific to be reused. By introducing a score based on the quantitative usage and a rating system we can assign a relative quality to the patterns and move the good patterns to the foreground and hiding the bad patterns in the background.

5.3 Model Modifications: Validation, Automatic Updates and, Notifications

WSDM is an iterative methodology, meaning that a few iterations may be needed before the design is finished. Therefore it is for example possible that we add a feature to the conceptual phase in a second iteration; however the navigational design was initialized with the features from the first iteration. The new feature still needs to be added to the navigational design. It would be possible to propagate the adding of the new feature to the navigational design and site structure design, but we believe that this would confuse the web developer because the feature does not only needs to be added, but links and pages must be added in respectively the navigational design and site structure design. Instead, we propose to do validations on the models and give warnings when inconsistencies are found. In some cases it may be possible to do automatic updates in later phases.

5.3.1 Mission Statement Specification: Adding Target Users

When we add a target user group we need to be able to assign requirements to it in the audience classification phase. Adding the target users to the audience classification phase can be done automatically because it only adds an extra column to the matrix table but it has no effect until the web designer assigns requirements to it. We do not need to update any other phase because the added target users are not a new audience class since it has no requirements assigned. Only after assigning requirements to this new audience class and creating the audience hierarchy again, a new audience class is created and this may have an impact on the rest of the design.

5.3.2 Website Genre Selection: Changing the Website Genre

It is improbable that one would want to change the website genre after a complete iteration, but it is plausible. The website genre is used to initialize the conceptual design phase, but the initialization was already completed in the first iteration. This means that the conceptual design was already initiated once and processed by the website designer. We assume that a web designer created a conceptual model that is for the most part correct and only needs some minor modifications. If a project changes from the website genre 'E-Commerce' tot 'Blog', the e-commerce may still be present but the blog part gets a more prominent role. The conceptual model still needs to contain the WDP for the e-commerce. Therefore, changing the website genre after the first iteration will have no automated effect on the conceptual design phase but it is up to the website designer to update it.

5.3.3 Audience Classification: Modification to the Audience Classification

Adding a new Audience Class

When a new audience class is added after the requirement matrix was filled, a conceptual model for this audience class must be created. The conceptual models use the same inheritance as the audience classification. If we look at figure 5.2, and B is the new audience class that was added, the conceptual model for audience class B extends the one for audience class A. Which means that when we look at the conceptual model for B at this moment, all the WDPs and relations of the conceptual model of A are visible and we are

able to add new WDPs and relations that are only applicable for audience class B. This means that we do not need to initiate the conceptual model of B because a model is already available thanks to the inheritance. It is up to the website designer to extend this model for the audience class B with the extra requirements added which causes B to be a child class of A.

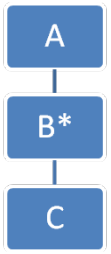

Figure 5.2: Example audience classification hierarchy when adding audience class

Removing an Audience Class

When an audience class is removed from the audience classification, the audience characterization, its conceptual model and navigational model are no longer relevant and should therefore also be removed. This can be done automatically. The site structure design is a combination of the navigational models of all audience classes, so features in this model only need to be removed when they do not belong to any other audience class.

Updating an Audience Class

Updating an audience class means adding or removing requirements that belong to this audience class. These updates are particularly complex when these changes modify the tree structure (hierarchy) of the audience classification. When the parent class of an audience class A (figure 5.3) shifts from B to C, the conceptual model no longer extends the conceptual model of audience class B but that of C.

Thanks to the local WDP repository we can identify identical features in the diagram of audience class B and audience class C. This enables us to keep the features that extend the parent diagram and also keep the relations to features in the parent diagram. Relations to features that are present in diagram B but not in C are lost. The web developer can be notified of these removals. These changes also affect the navigational and the site structure design.

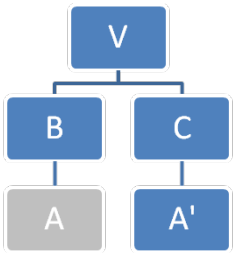

Figure 5.3: Example audience classification hierarchy when updating an audience class

5.3.4 Conceptual Design: Modifying the Feature Assembly Diagram

Just like in the previous section, we need to consider the three possible cases to modify the feature diagram: add, update and remove.

Adding a WDP

When adding a feature, we can either add a completely new feature or a feature from the local repository. When adding a completely new feature , the tool can add this feature to the navigational design for the current audience class and to the site structure design. For both models the feature will be added but it won't have any connections, which makes the models invalid. Therefore, the developer needs to be notified of this. In the next section, we will make sure this invalidation remains visible to the developer the entire time.

When adding a feature from the local repository, it only needs to be added to the navigational design. The site structure design is a combination of the navigational designs of all audience classes, since the feature was in the local repository it was already present in the site structure design from another audience class. We add the feature to the navigational design in the same way as we would add a new feature.

Updating a WDP

A feature or WDP is a relatively complex dataset, however, the only modification that would affect other phases of the design is the modification of the name. However, changing the name of a WDP can be easily automatically propagated in the design. We do not need to update the name in other audience classes since we use the concept from the local repository. This ensures

that in the different audience classes, we are talking about the same WDP.

Removing a WDP

A developer could remove a WDP in the conceptual design phase when it is no longer needed. Therefore it can also be removed in the navigational design of that audience class. The site structure design is a combination of all navigational models of all audience classes. So the WDP that was removed in the conceptual phase can only be removed from the site structure design if it isn't being used in the conceptual model of another audience class. If this would be the case, the combination of all navigational designs would still contain the feature.

5.3.5 Modifications in the Navigational Design and the Site Structure Design

Both the navigational designs and the site structure design are 1-to-1 mapping from the conceptual design. Until now a modification only affected later phases. If we would allow a developer to modify the features here, previous phases would also be affected. To control the complexity of possible updates we do not allow the developer to modify features in these phases. If the developer wants to modify the features, he needs to go back to the conceptual phase. Modifications to the links in the navigational design and pages in the site structure design will not affect other phases.

5.4 Model Validations

By using specific tools for each type of model we have the obvious advantage that we can do some validations on them. For example: "Is the feature configuration model free of inconsistencies", or "Is each feature in the site structure design assigned to at least one page?". The checking of the internal structure of a model is outside the scope of the thesis.

Consistency between models should also be ensured. For example: "Is each audience class captured in the site structure design". In principle, these kinds of inconsistencies cannot occur as the tool uses the models of previous phases to initiate the model in later phases, and, as discussed in the previous sections, warning and flags can be raised when a model update has an impact on a later model. In previous section we discussed several automatic updates when modifying a model. In some cases, updating one model causes another

model to invalidate. Because we have these model validations, correcting the whole methodology after updating one model boils down to fixing the errors from the model validations. The user does not need to manually check everything again.

5.5 Requirement Satisfaction

At the moment we did not discuss any relation between the requirements and the features. In the methodology, the website genre initiates the conceptual phase. It would however not be unreasonable to say that a website genre could also initiate the requirements and the requirements in turn could initiate the conceptual phase. This approach would even further help the developer to design the system.

The problem with such an approach is that requirements are in natural language. Natural language is inherently ambiguous so we cannot do an errorless interpretation of the requirements to the conceptual phase.

It would however still be useful to check if the conceptual design fulfills all requirements. This validation is an extra effort a designer could do, but is not part of the methodology itself. By checking for each feature the requirements it fulfills, we create a manual mapping. This mapping has two benefits:

1. After we completed this mapping we can generate a list of requirements that are yet to be fulfilled. This ensures that the developer does not forget an important requirement in the system.

2. When updating a requirement, we can see the different features that implement this requirement. We have immediately the link to the features we need to update to accommodate the change in the requirement.

6

WSDM Design of support tool

In previous chapters we discussed the adaption of the original WSDM methodology because it does not work well with implementations that use CMSs. In this chapter, we will discus the design of the support tool. The support tool is a web application. Therefore, WSDM can be used for its design. However, because of its complexity, we will not use a CMS. Therefore, we have used the original WSDM method instead of our own WSDM-Lite version.

6.1 Mission Statement Specification

Subject: The subject of this tool is the design of web application projects which will be implemented using a CMS. These designs are divided into several different phases (or views): the mission statement specification, the audience classification, the audience characterization, the conceptual model, the navigational model, the site structure design and the template design. The actual implementation is not part of the design. The output of each phase is a model.

Purpose: The purpose of the tool is to support a web engineer to design the web application. The program needs to provide tool support to create the models as well as meta-support such as: initialization, validation, collaboration, etc.

Target users: The target users of this tool are web designers. In a later stage, more users could be added since mostly more people than only the web designers are involved in web projects.

6.2 Audience Classification

Web development projects are private projects. Therefore we will put most of its functionality after a login.

6.2.1 Visitors

1. Visitors should be able to sign up for an account. We do not require much personal information of this visitor. He should provide his e-mail address and a password which will be used to log in. We will also ask for the web designerâĂŹs name in order to have a more personal interaction after logging in.

2. Visitors should be able to login when they have already created an account. The login uses their e-mail address and a password.

6.2.2 Web Designers

Because the WSDM-Lite methodology is divided into different phases, we will sort the requirements accordingly. We will start by the more general requirements not related to any specific phase.

Projects

1. A web designer should be able to manage multiple projects. He must be able to get an overview of all of his active projects and in which phase they are. Besides the models the project also must get a title so that is easily recognizable.

2. The project consists of the different phases of the WSDM-Lite methodology. Each phase must be finished completely before going to the next. In the phases where this can be checked this must be enforced. In cases where this cannot be checked it must be encouraged.

Mission Statement Specification

3. In the mission statement specification, the web developer needs to enter the subject, purpose and target users of the web application to be built. All of these are mandatory. It must be possible to add some minimal styling (bold, italic, underlining, colors, listing) to the subject and purpose.

Website Genre Selection

4. In this phase a web designer must be able to select a website genre for the new application. The possible selections are those which are available in the WGP repository. The possibilities must be sorted in such a way that the more popular are visible more prominent and the WGPs that are not frequently used are in the background.

Audience Classification

5. The audience classification should be initialized by the data entered in the mission statement specification. The target user groups must be initialized as the columns of the requirement matrix. These target user groups are not yet real audience classes until the requirement matrix is filled and they are placed in an audience hierarchy and doubles are removed.

6. The WSDM-Lite methodology does not enforce any structure in the requirements so they can be entered in any structure. A text-document like structure such as in this chapter is recommended. The requirements still need to be individually addressable so they can be used in the requirement matrix.

7. Requirements themselves must be able to have minimal styling options (bold, italic, underlining, listing, ...).

8. Web designers must be able to assign requirements to target user groups while still having a good overview of the requirements and the possible target user classes.

9. Web designers must still be able to add or remove target user classes in this phase in case they forgot any.

10. Web designers must be able to filter requirements per (set of) target user groups to get a better insight in their requirements.

11. The web designer can verify the requirements matrix which results in an audience classification hierarchy. Only when this verification is completed the project can continue to the next phase. Each time the requirement matrix is changed, also by adding a new row, the matrix needs to be verified before the designer can continue.

Audience Characterization

12. The audience characterization must be initialized with the models resulting from the audience classification phase, namely the audience classification.

13. The more frequently used characteristics must be suggested such as language, country, age, ... It is impossible to foresee all types of characterization so web designers must be able to add their own characteristics.

14. The audience characterization does not follow the inheritance resulting from the audience class hierarchy but it must be easy to assign a characterization to more than one audience class since there is a high probability that they will share some characteristics.

Conceptual Modeling Phase

15. The conceptual model takes as input both the website genre selection phase and the audience classification phase. To initialize this phase the website designer must select one of the WGPs related to the website genre phase. This WGP will be used as starting point for the conceptual phase. The web developer needs to select a WGP corresponding to the selected website genre from the WGP repository.

16. The web designer must be able to switch between the conceptual models of the audience classes in a transparent way, so he knows at all times which conceptual model is visible.

17. The conceptual models use the inheritance of the audience classification. Features and relations of the feature assembly diagram of a parent class P are also visible in a child class C. You can look at this as having a paper with a diagram P and a second, transparent paper that adds a second layer C to P.

18. Features can only be deleted at the top-level audience class. When a feature is present in audience class A (figure 6.1), it is also visible in B

and C thanks to the inheritance. We cannot remove this feature in B because this will affect A (a feature cannot be present in A and not in B). Therefore, we can only delete features at the top level where they are visible.

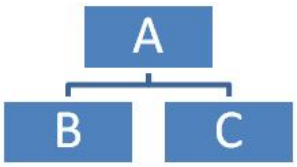

Figure 6.1: Audience Classification Example Situation

19. When we delete a feature at level A (figure X), we only delete the feature at this level. The feature gets two new top-level audience class locations: B and C. We can then delete the feature in both places again to completely remove the feature from the conceptual model. When the project has a more complex audience classification structure, this can become cumbersome. Therefore a second 'delete' action needs to be introduced which deletes the feature from the whole (sub-)tree. This can again only be done at the top level.

20. Adding features again needs to be done at the top level. When we add a feature to the conceptual model of audience class A in figure X, this feature will be also accessible in B and C.

21. When a feature is added, this can either be a completely new feature, a feature from the general WDP repository, or from the local WDP repository.

    - When adding a new WDP, the mandatory specifications (name, problem, solution and rational) need to be added immediately. Other specifications can be done later.
    - The local WDP repository contains all distinct patterns that are being used in the current project. When this WDP is inserted, no copy is made but simply a reference to the already used pattern. Duplication should be avoided.
    - The general WDP repository contains all shared patterns of all projects. These WDPs can be filtered on website genre and are sorted on popularity.
    - When viewing a pattern from one of the repositories, the user can navigate to the related patterns

22. Modifying a feature can be done from anywhere and the changes will be visible in all places.

23. The features in the feature assembly diagram of the conceptual model only display the name of the WDP. When the user clicks on it he can access the rest of the WDP information and modify this.

24. The problem, solution, when to use, rational and warnings are blocks of text which can be edited.

25. Related patterns and the website genre can only be optionally specified when sharing the WDP in the WDP repository. Patterns that are not shared do not benefit from this information.

26. CTTs and object chunks can be added as much as the web designer seems fit. Both can be given a name for recognition purposes but don't have a specific purpose in the design of the project.

27. Both for the CTT models and the ORM models a specific modeling tool needs to be provided which allows the designer to easily create models in these languages.

28. The CTT, ORM and Feature assembly diagram models need to be validated against their modeling language syntax. Only when these validations are successfully completed the web designer is allowed to complete this phase and go to the next.

Navigational Design Phase

29. The input of the navigational design is the models created in the conceptual design phase. The way the inheritance is used stays the same. The features that are present in the models also stay the same. The notation of the features is adapted to the navigational notation described in the WSDM book chapter [8]. All relations between features are replaced by navigational links to initialize the model. Some of these links will need to be removed and other will need to be added.

30. Features cannot be modified in this phase to avoid unnecessary complications and confusion for the web designer. If the web designer wants to modify features (add, edit or remove) he has to go back to the previous phase.

31. Specific modeling tools need to be provided to the web designer in order to create a valid navigational model.

32. The web designer must be able to switch between the navigational models of the audience classes in a transparent way, so he knows at all times which conceptual model is visible.

33. The navigational model needs to be valid before the web designer can continue to the next phase. This means that all WDPs should have at least one link associated with it in order to be connected to the main application.

Site Structure Design Phase

34. The site structure design needs to display the navigational models similar to the previous phase but they can't be edited anymore. When changes need to be made to these models, the web designer has to return to the previous phase.

35. In the site structure design phase the web designer has access to the right modeling tools to group the features into web pages. An '*' icon can be added to grouped features which will be displayed on more than one page.

36. The grouping of WDPs onto pages follows the same hierarchy as the conceptual and navigational models: If features X and Y are on the same page for the site structure design for audience class A, they are also on the same page for audience class B and C. B and C could however add extra features to this page.

Template Design

37. The web designer may choose not to use the WSDM-Lite template design phase and instead design the template for the CMS itself directly. In this case, the template design phase is removed from the available phases in the project.

38. The template design is initialized by the site structure design. Each grouping of pages result into one editable region in a template (Headers and footer of a template may remain the same for example). For groupings containing a '*' icon more editable regions for a template may be created at will.

39. The correct tools need to be provided described in the presentation design phase in the WSDM Book chapter[8].

Cross-Phase Requirements

40. How phases should be initialized is discussed in section 5.1

41. How the tool should handle updates after the first iteration is discussed in section 5.3

Navigational Requirements

42. The web developer should be able to easily navigate and jump to other WSDM-Lite phases of the project. In the first iteration the web developer should not be able to jump ahead to a phase that was not yet initialized.

Requirement Matrix

Because we only have one target users group we have a fairly simple requirement assignment as seen in table 6.1.

Requirement	Visitors	Web Designers
Visitor.1	Yes	Yes
Visitor.2	Yes	Yes
Web developer.1	No	Yes
Web developer.. . .	No	Yes
Web developer.N	No	Yes

Table 6.1: Example

Which results in the audience classification seen in figure **??**.

Figure 6.2: Audience Classification

6.3 Audience Characterization

Web developers: Web developers are people who are experienced in creating websites and but have a background in computer sciences. They understand the need for a methodical approach to developing web applications and are able to model these in various aspects. They have an understanding of how WSDM-Lite works as described in chapter 4.

6.4 Conceptual Design

6.4.1 Task and Information Modeling

We now give the task and related information models. Because the support tool is still in a prototyping phase we will not give the object chunks with all the related actions but just the base ORM diagrams of the object chunks. Giving all the object chunks would lead us to far from our purpose. When possible, the official data model of models will be given.

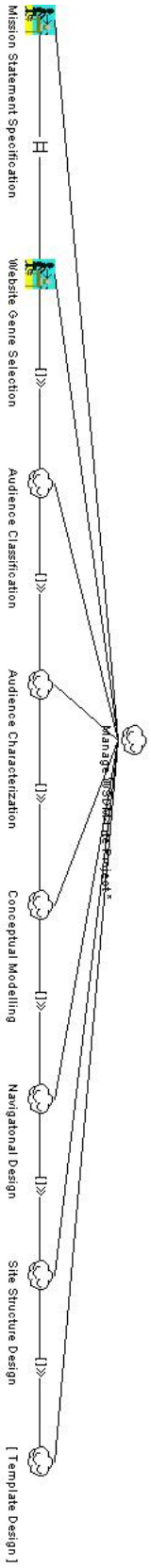

Figure 6.3: Task model

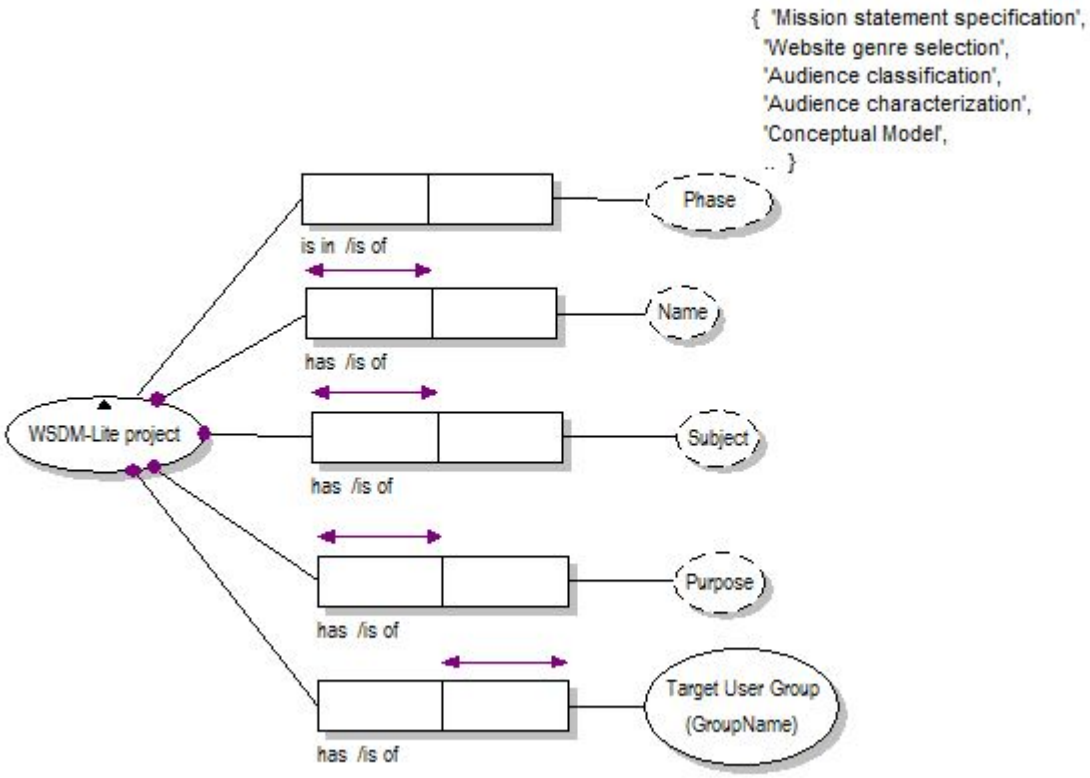

Figure 6.4: Information model for the mission statement specification

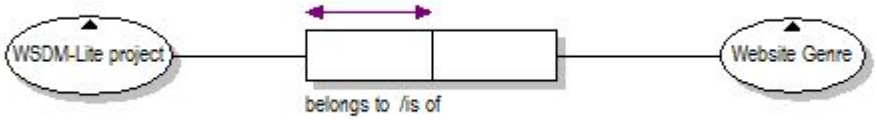

Figure 6.5: Information model for the website genre selection

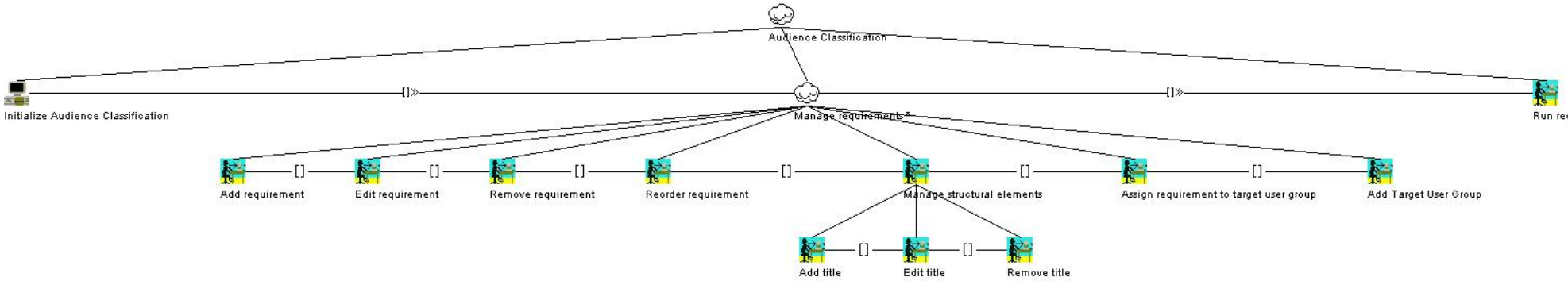

Figure 6.6: Task model for the Audience Classification Phase

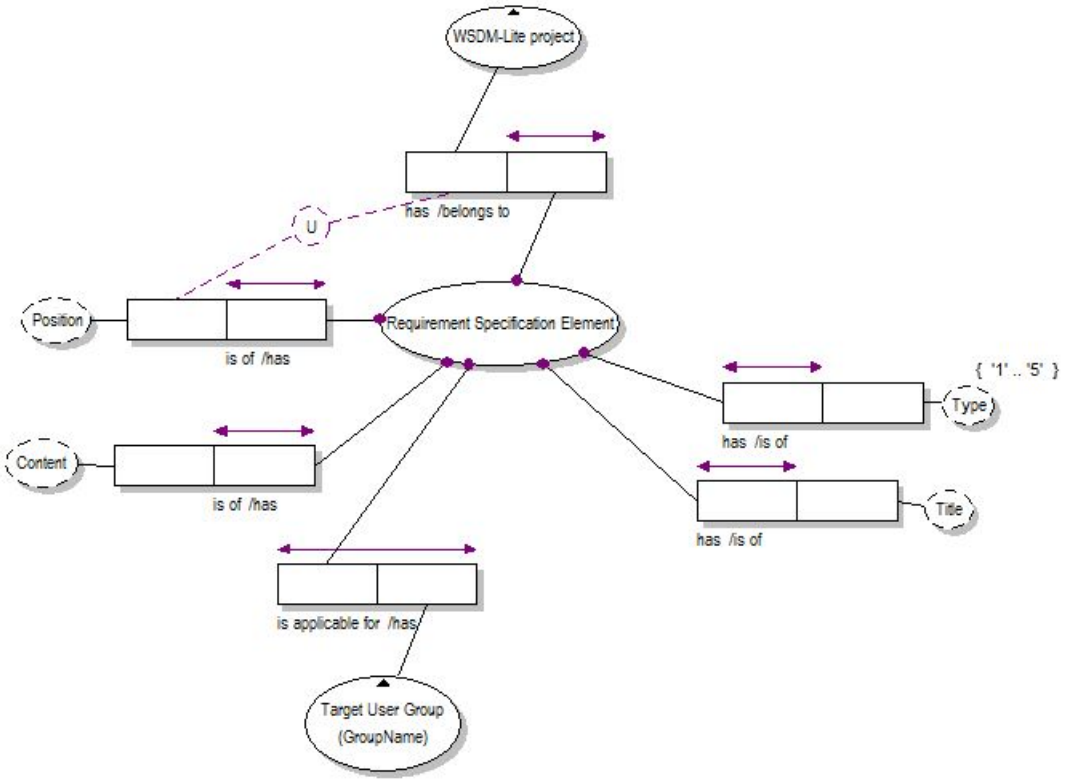

Figure 6.7: Information model for a requirement

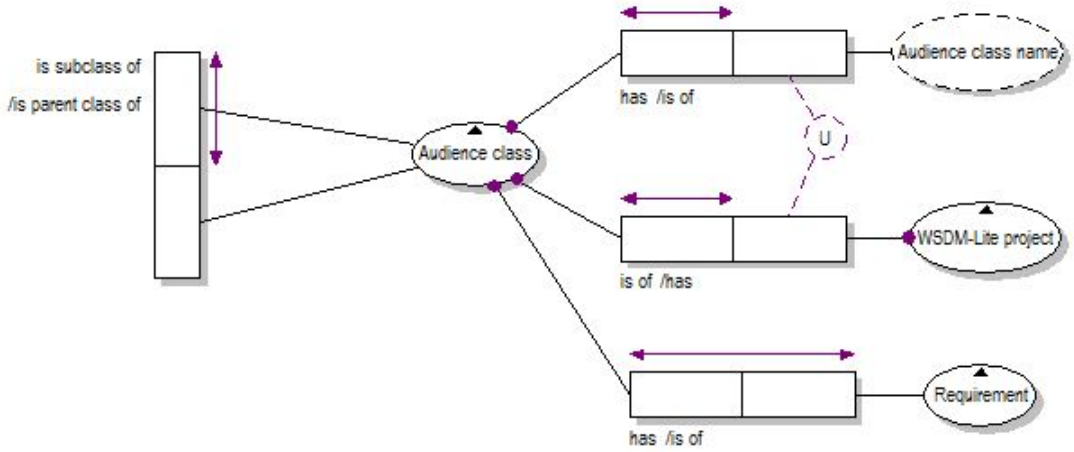

Figure 6.8: Information model for an audience class

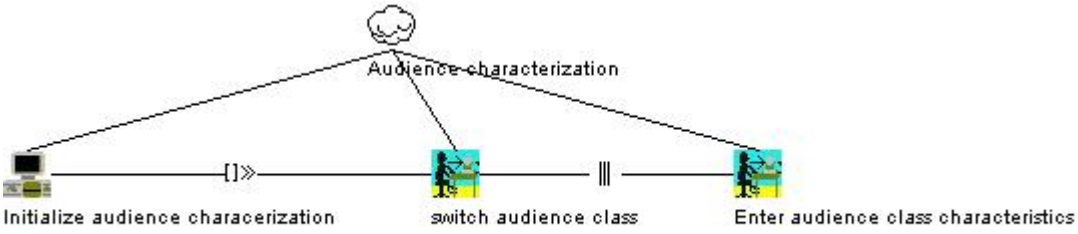

Figure 6.9: Task model for audience characterization

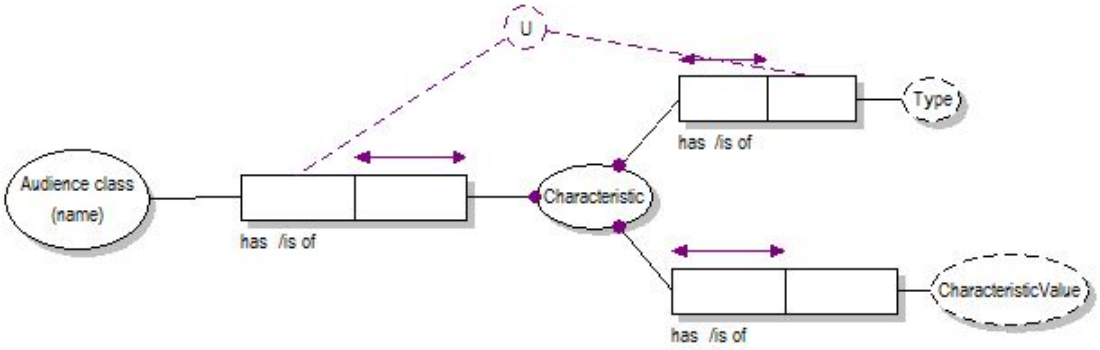

Figure 6.10: Information model for audience characterization

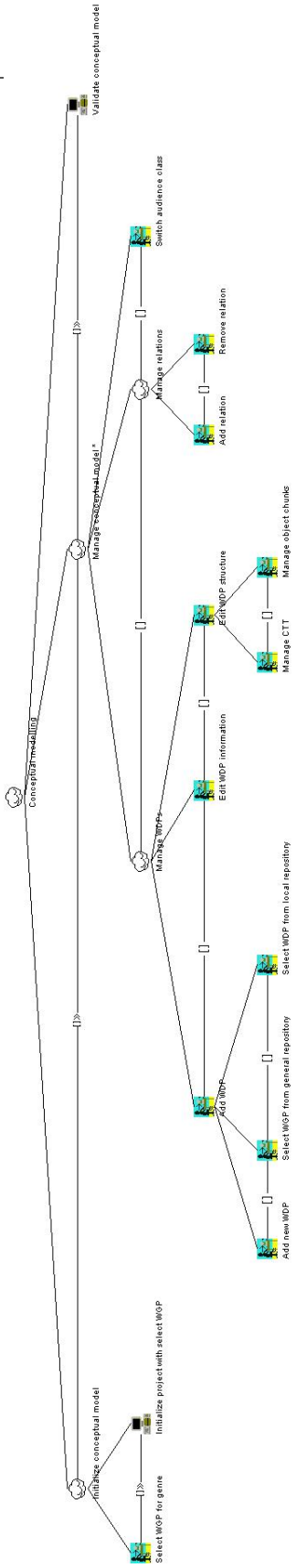

Figure 6.11: Task model for the conceptual modeling phase

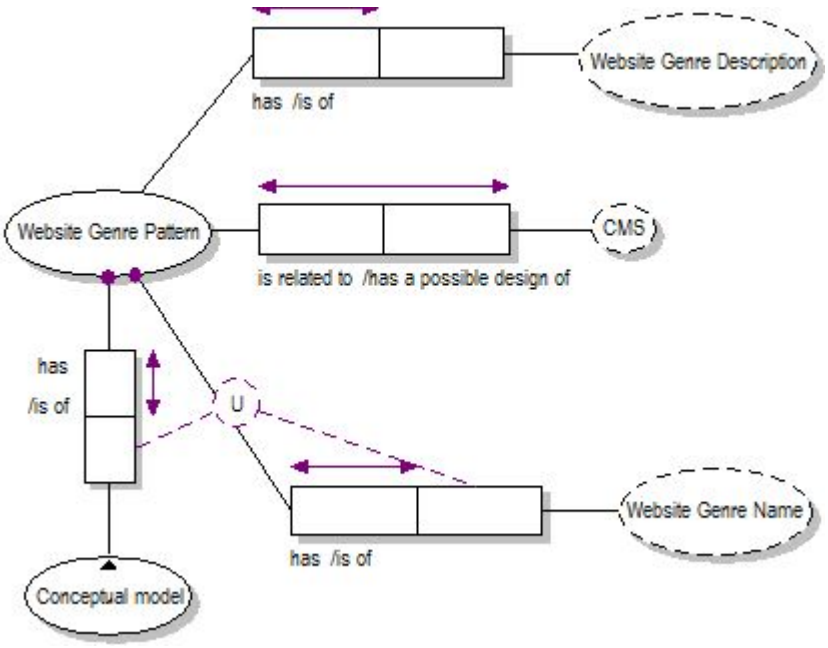

Figure 6.12: Information model for a WGP

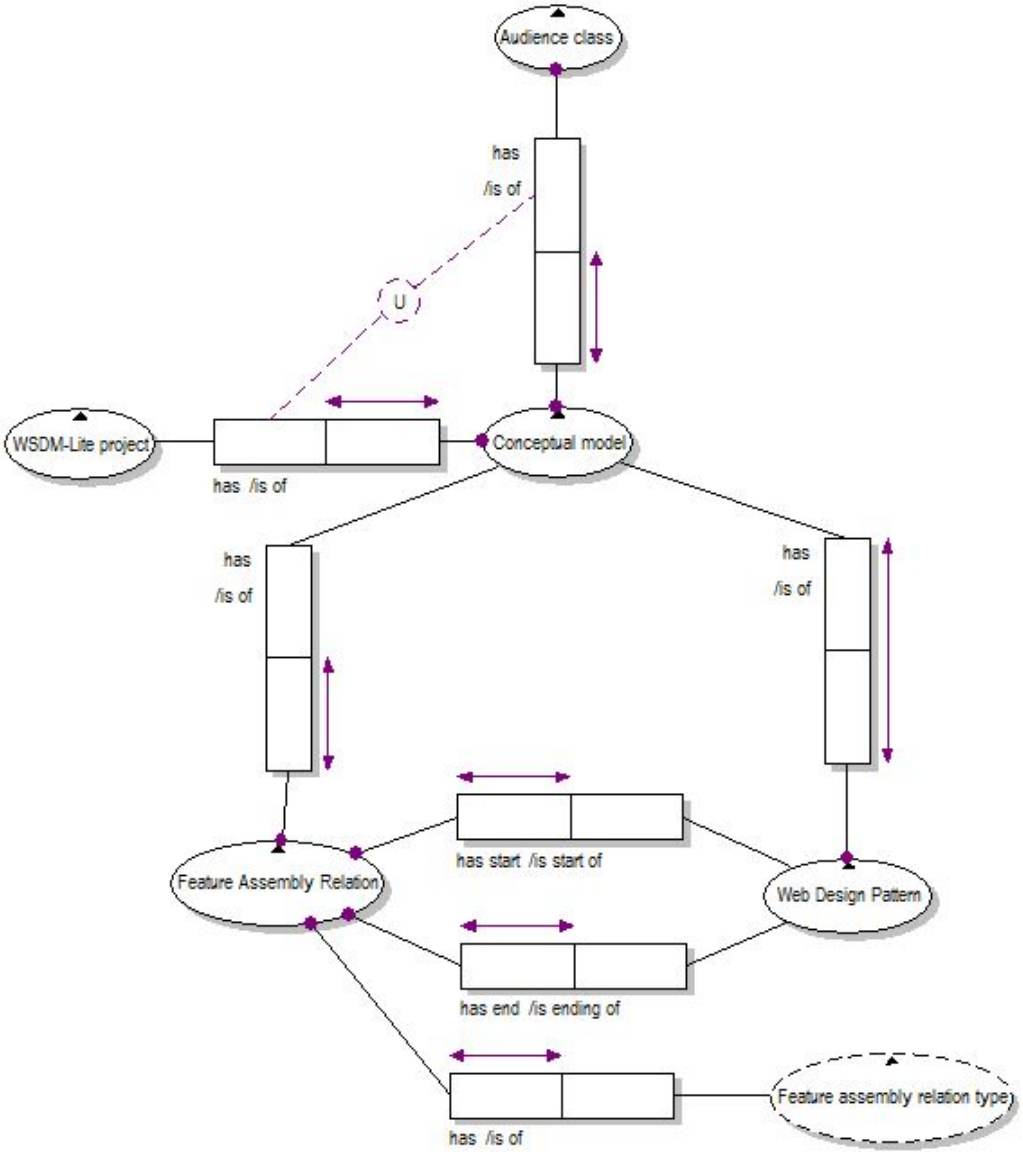

Figure 6.13: Information model for a conceptual model

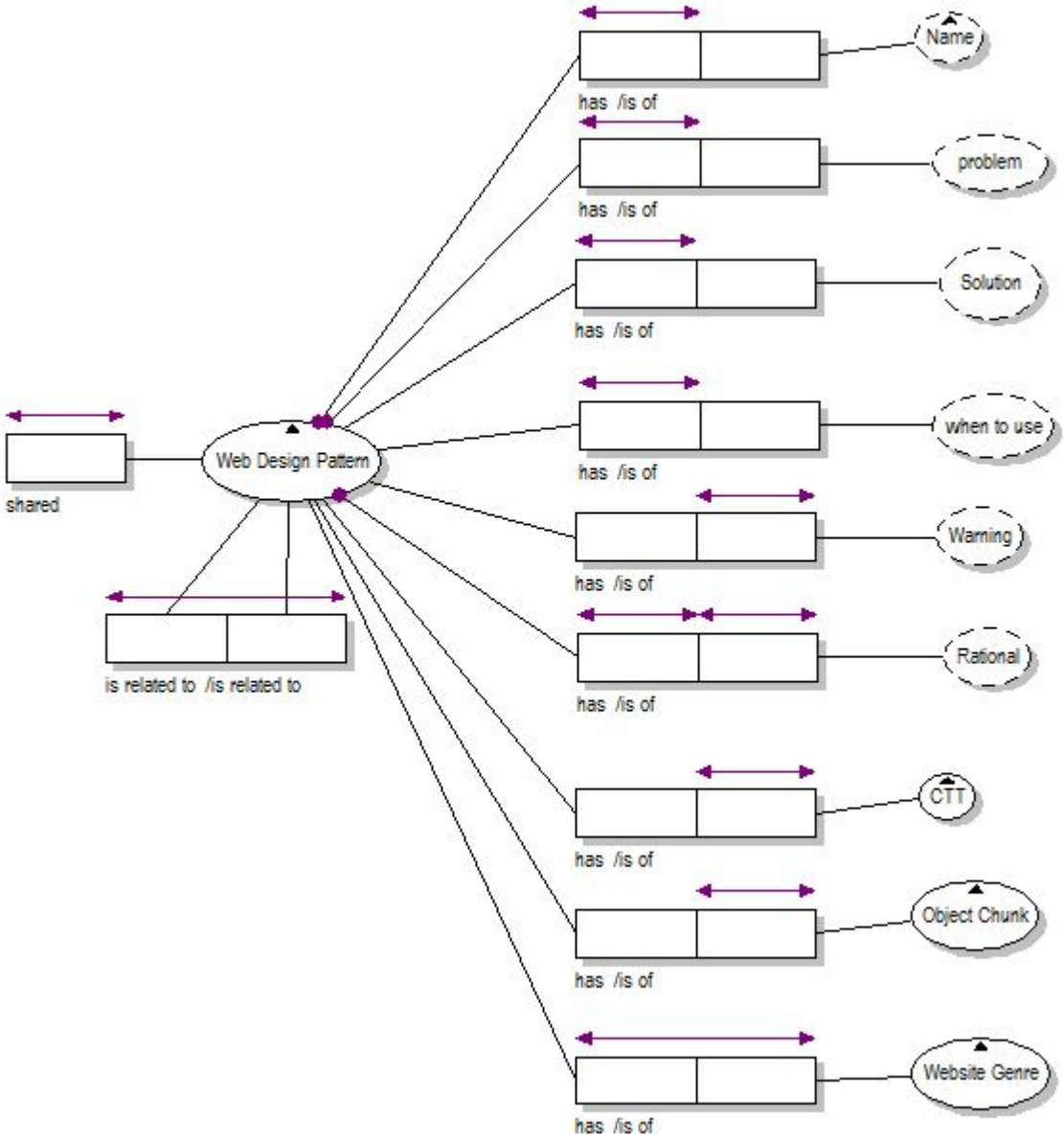

Figure 6.14: Information model for a WDP

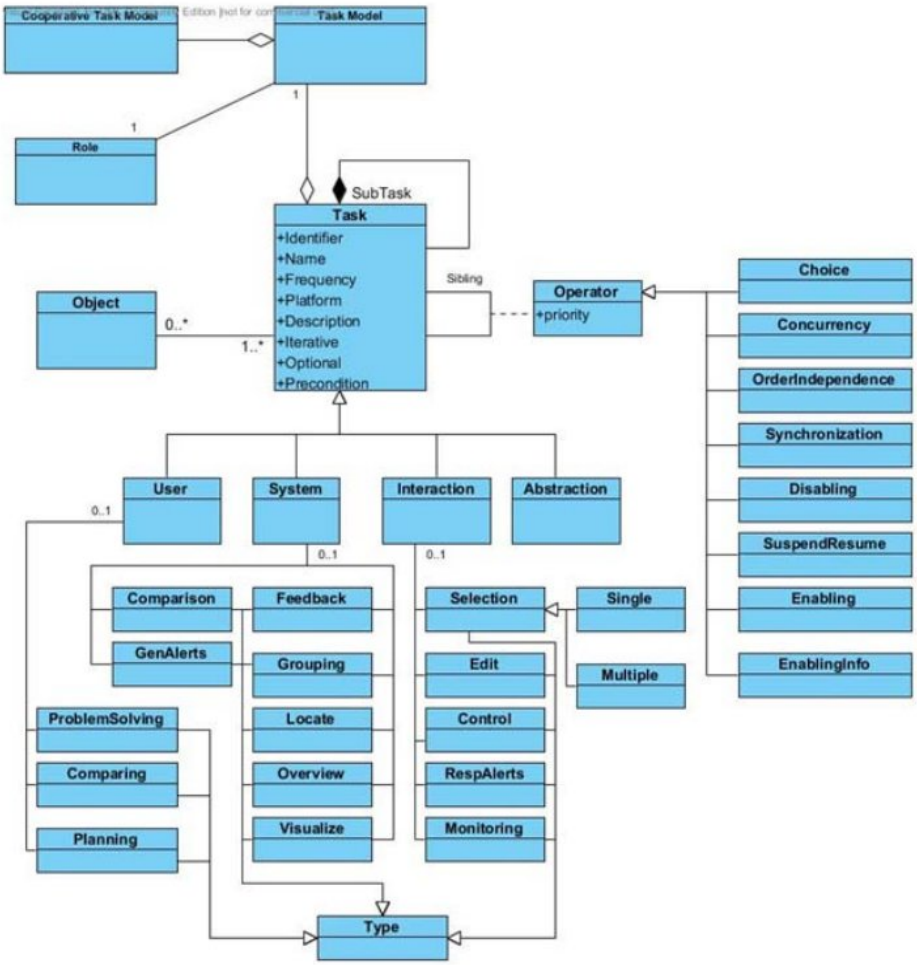

Figure 6.15: Information model for a CTT [10]

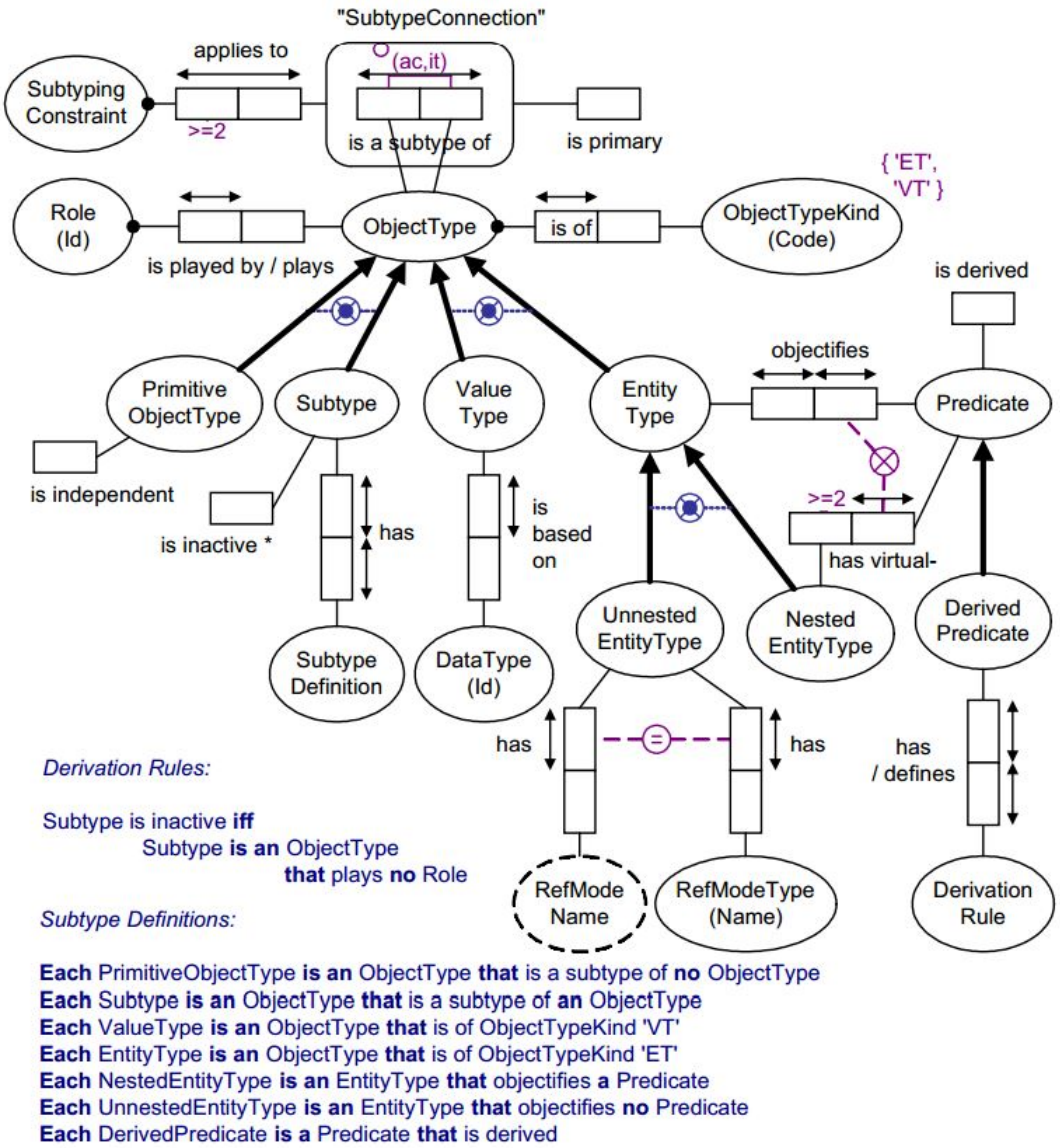

Figure 6.16: Information model for an object chunk [19]

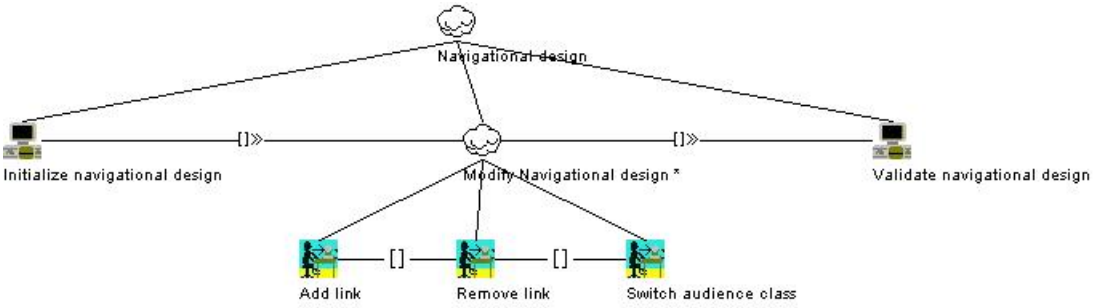

Figure 6.17: Task model for the navigational design phase

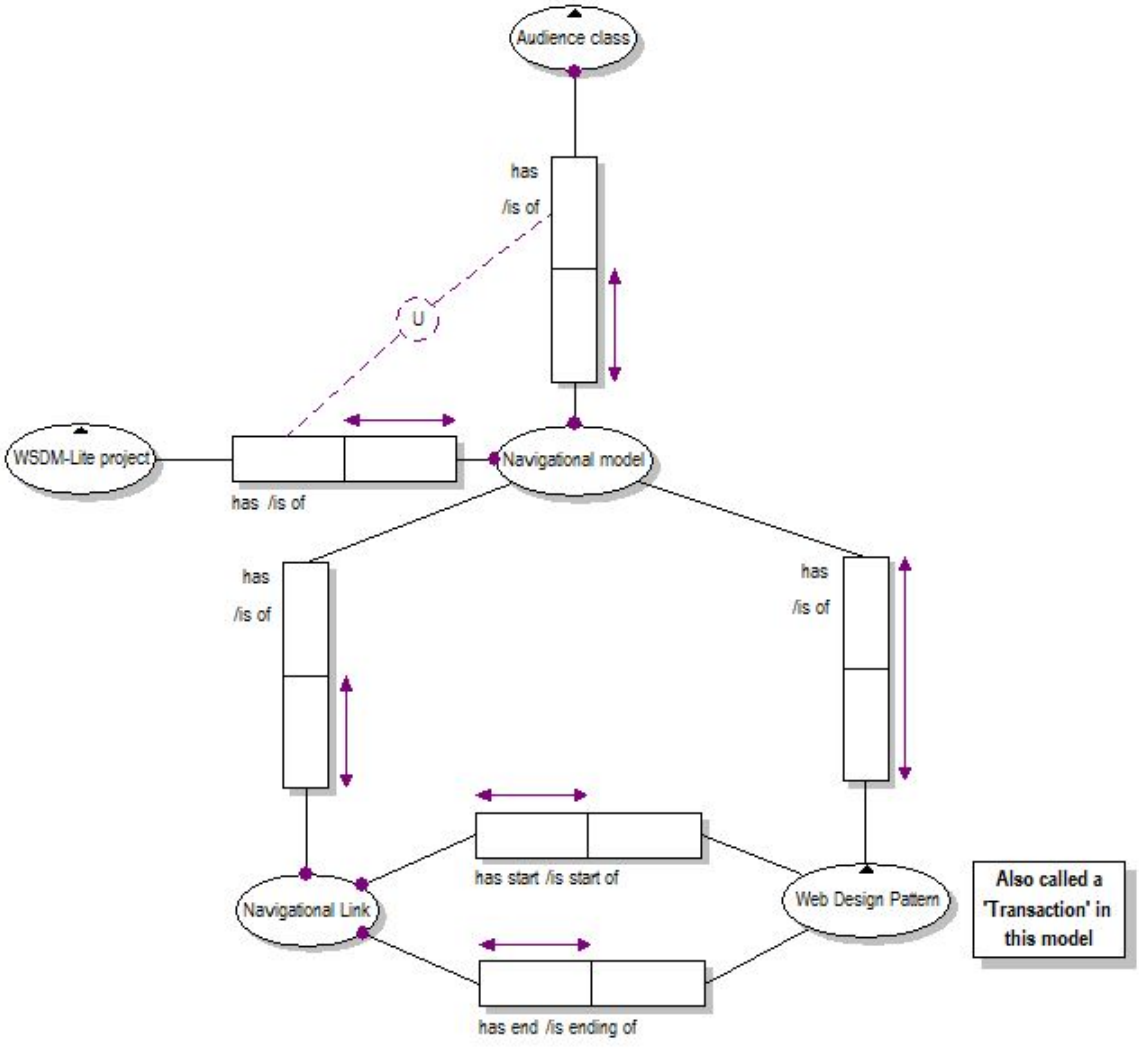

Figure 6.18: Information model for the navigational model

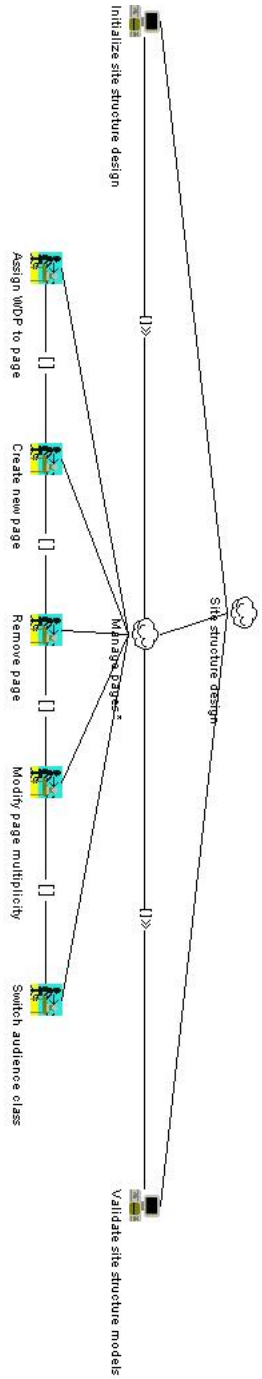

Figure 6.19: Task model for the site structure design phase

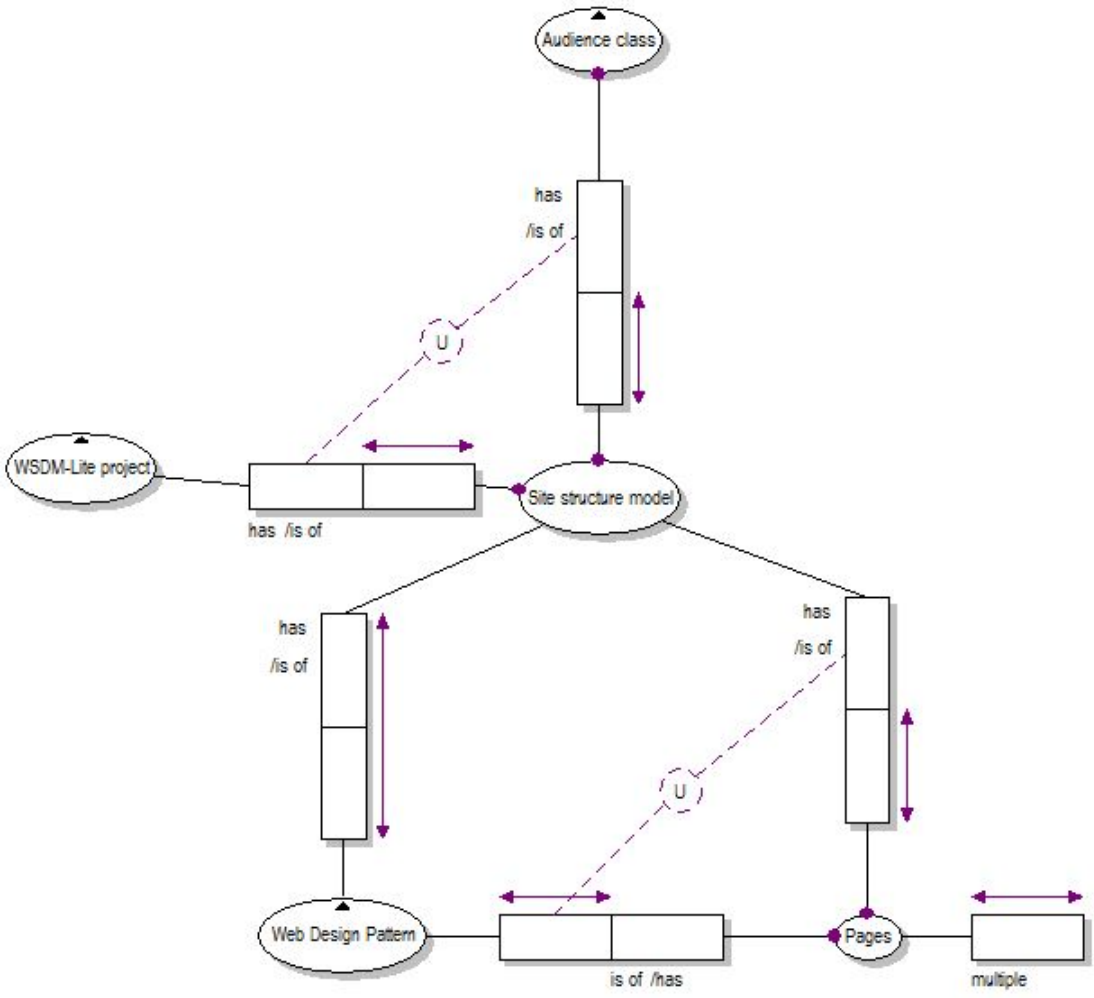

Figure 6.20: Information model for the site structure model

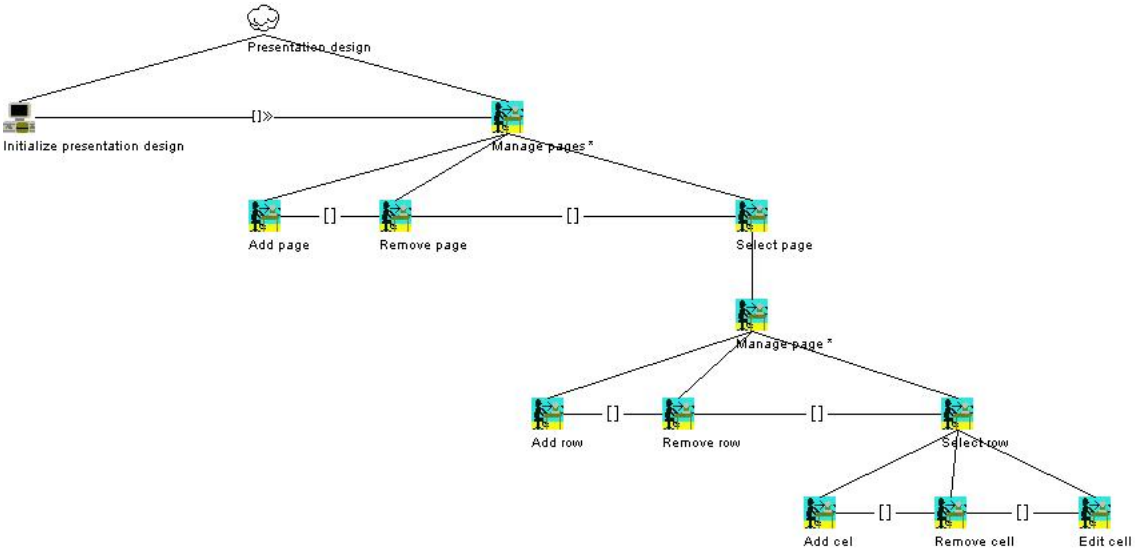

Figure 6.21: Task model for the template design phase

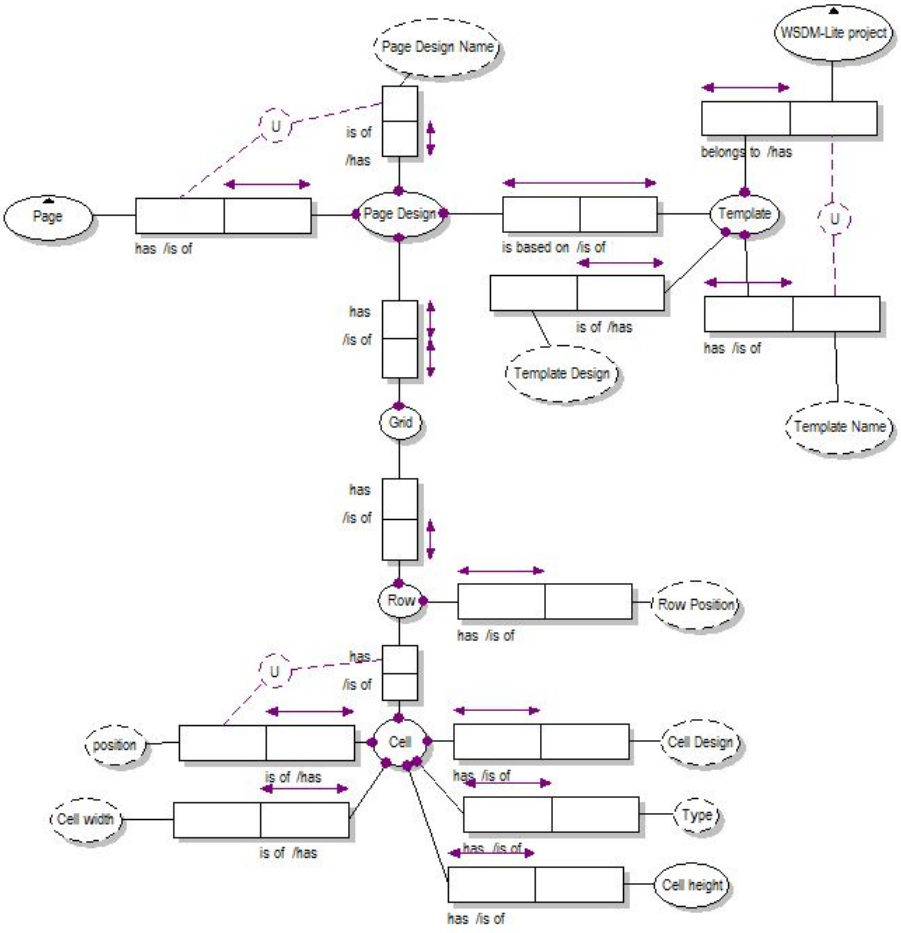

Figure 6.22: Information model for the template design phase

6.4.2 Navigational Design

The navigational model can be found in figure 6.23.

6.4.3 Site Structure Design

The navigational model can be found in figure 6.24.

6.4.4 Template Design

Because the first implementation will be a prototype, no presentation design will be created since the prototype will be used to explore various possibilities. Fragments of the presentation design can be found in the screenshots in the description of the implementation.

6.5 Implementation

An implementation base was set up for an evolutionary prototyping, i.e., the initial prototype should be able to result in the final product. The base contains login and registration for new users, project management and navigation through the different phases of the WSDM-Lite design. The implementation status of the different phases differs, some are completely implemented, others are only visual prototypes.

6.5.1 Application Core

The application is implemented as a "software as a service" (SaaS) [27] application. The application will be hosted on one location on which users can register and create projects. The application's implementation is based on the CakePHP[1] framework for backend implementation and a MVC structure and Twitter bootstrap[2] is used to make the visualization nicely viewable across different devices. Two open source projects, both called CakeStrap[3] [4], were used to speed up the implementation process. One project implemented the CakePHP bakery while the other one already implemented the login (Figure 6.25) and registration (Figure 6.26). Some improvements to the design were made.

[1] http://cakephp.org/
[2] http://getbootstrap.com/
[3] https://github.com/hugodias/cakeStrap
[4] https://github.com/Rhym/cakeStrap

6.5.2 Project Overview

The project overview page (Figure 6.27) was implemented and gives the web developer access to project management (add, edit or delete project). When clicking on a project the web developer goes to the current WSDM-Lite phase of the project.

6.5.3 Mission Statement Specification and Website Genre Selection

As specified in the site structure design of the WSDM-Lite support tool, the mission statement specification and the website genre selection can be found on one page (figure 6.28). For the purpose and subject the rich text editor TinyMCE[5] was used. For adding and removing target users jQuery[6] was used to give the experience of a rich internet application (RIA). This phase is completely implemented. On the bottom navigational aid links can be seen to the other phases which can be found in every phase.

6.5.4 Audience Classification

The audience classification is initialized by the target user groups specified in the mission statement specification as seen in the available columns in figure 6.29. Assigning a requirement to an audience class can simply be done by checking the checkbox in the correct column of a requirement. Editing, removing and reordering requirements or titles can be done by using the icons on the right. Adding a requirement or title in the correct place is done by hovering over a requirement or title above the targeted place, a menu will appear below to either add a requirement (+ sign) or a title (H1 to H3). This phase is completely implemented.

6.5.5 Audience Characterization

In the audience classification phase it is possible to enter characteristics about an audience class in a very open manner. Changing to another audience class is done by clicking the button (figure 6.30) and selecting another audience class. For this phase only a static prototype is provided.

[5] http://www.tinymce.com/
[6] http://jquery.com/

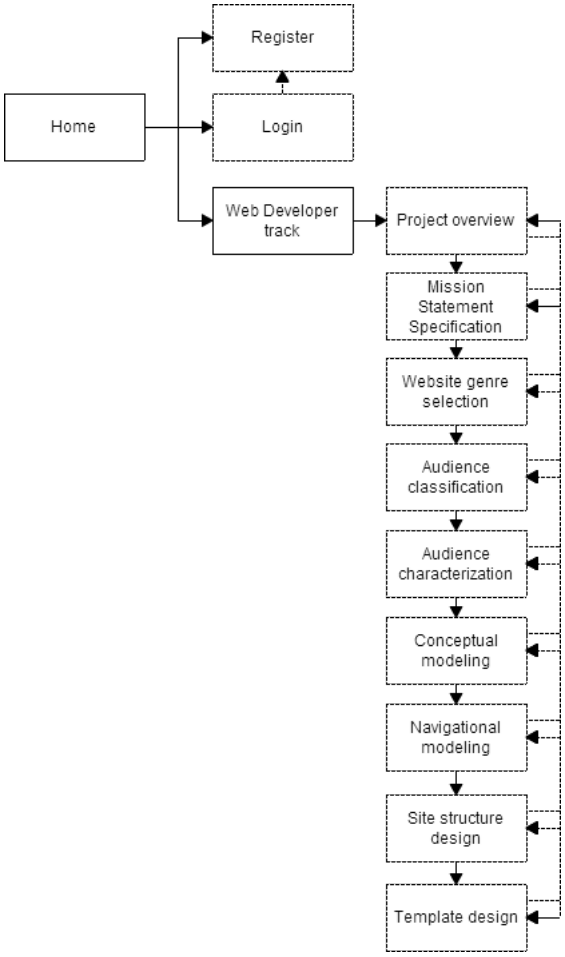

Figure 6.23: Navigational model

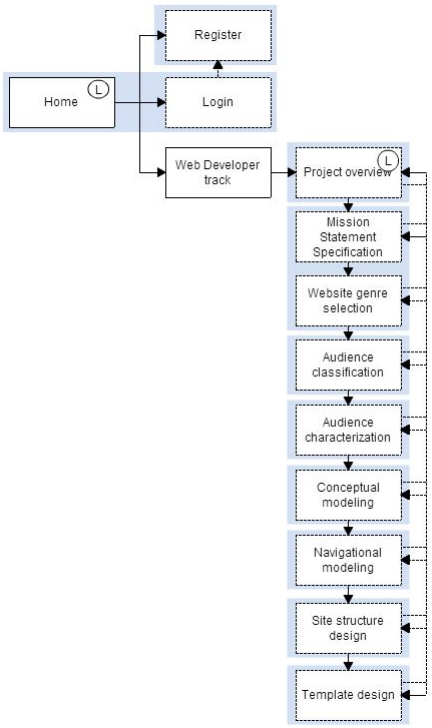

Figure 6.24: Site structure model

Figure 6.25: Screenshot: login

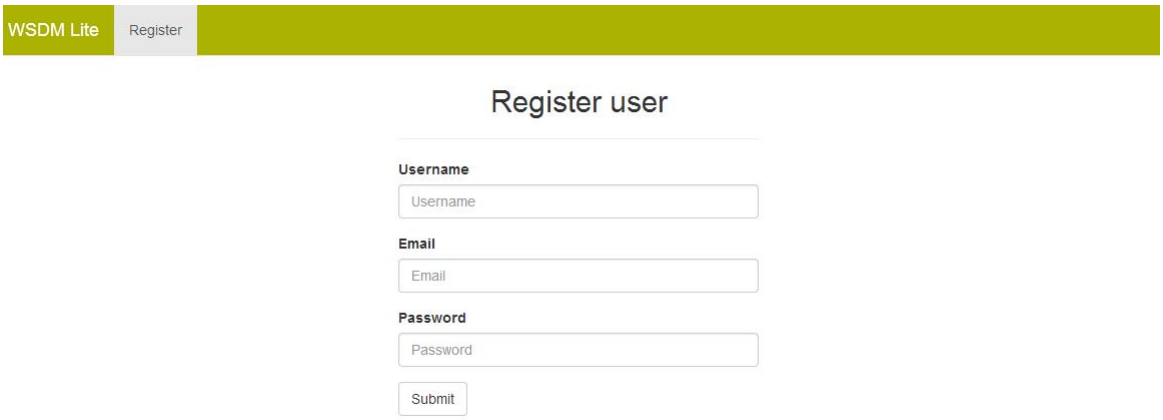

Figure 6.26: Screenshot: registration

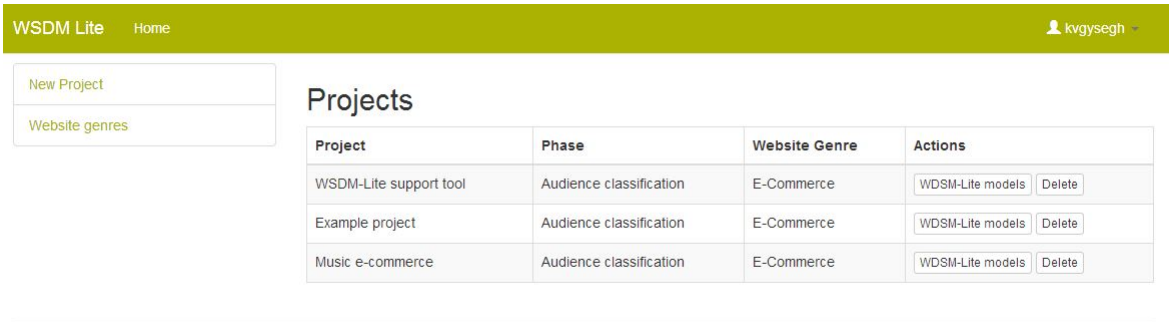

Figure 6.27: Screenshot: projects

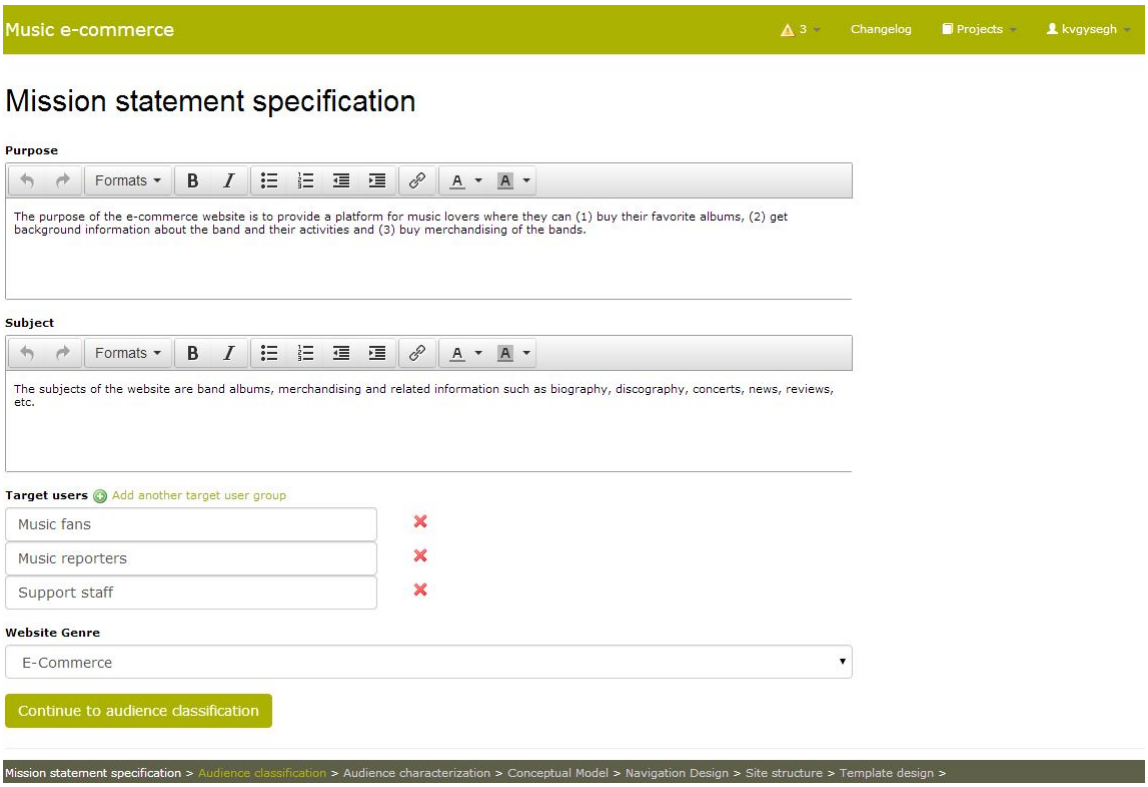

Figure 6.28: Screenshot: mission statement specification and website genre selection

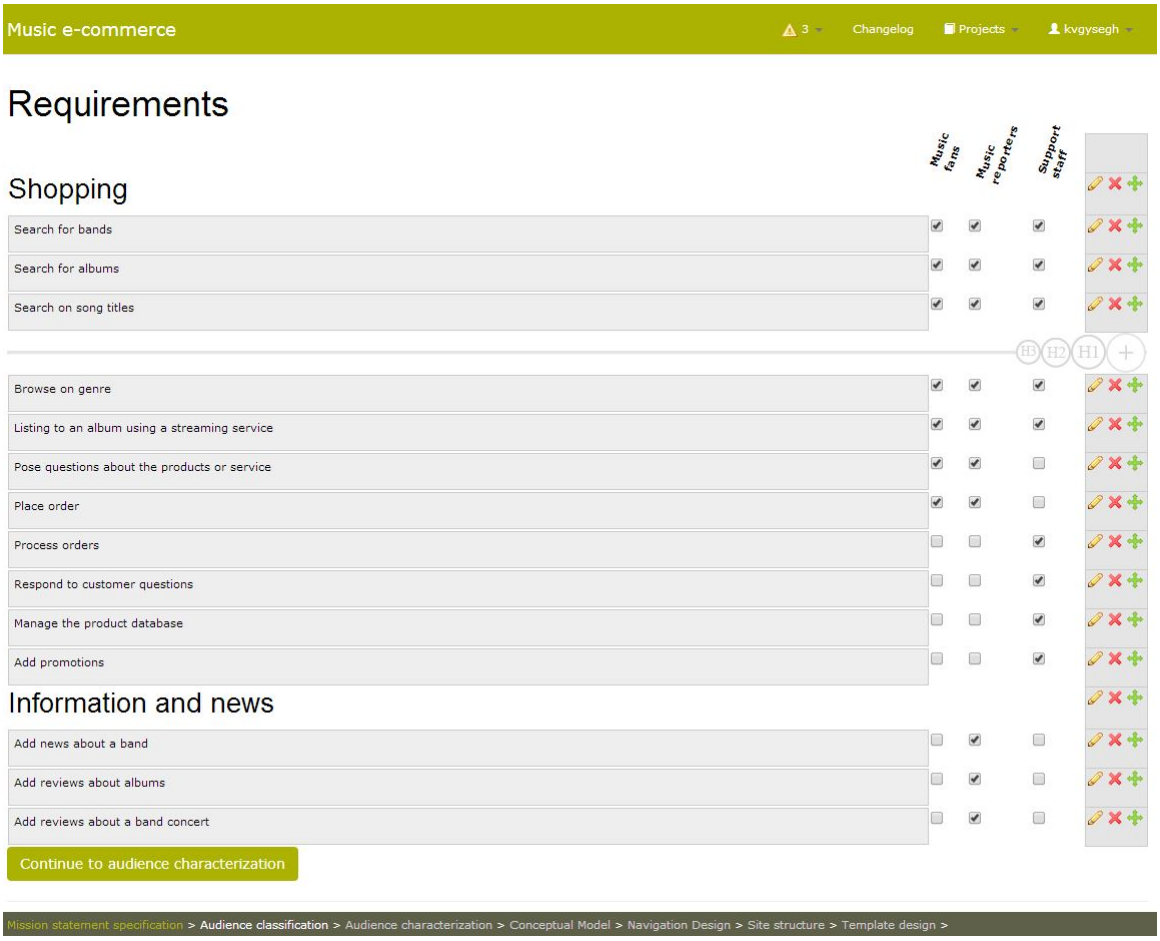

Figure 6.29: Screenshot: mission statement specification and website genre selection

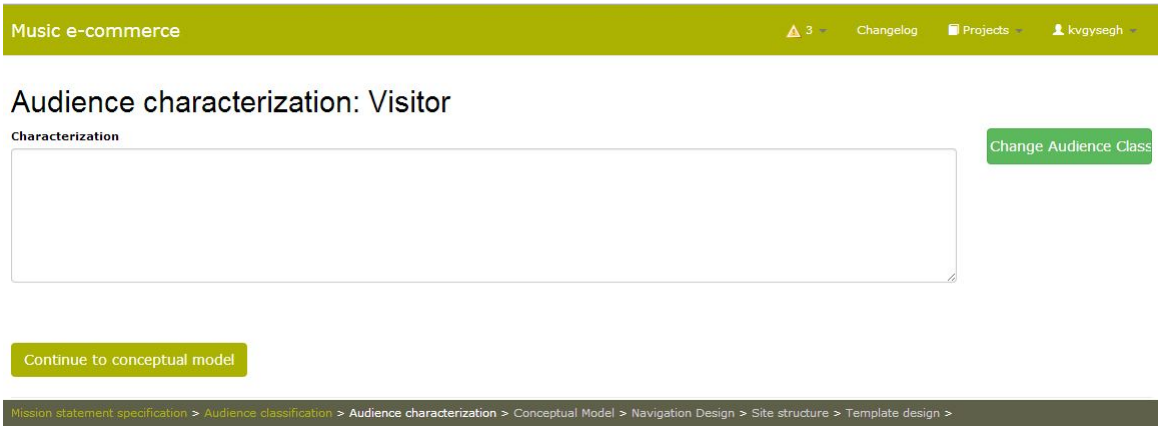

Figure 6.30: Screenshot: mission statement specification and website genre selection

6.5.6 Conceptual Model

In the conceptual model phase we see the feature assembly diagram (figure 6.31). Again we have a button to change to another audience class. We can click on a feature to see and modify the rest of the WDP specifications (figure 6.32). This overview overlays the conceptual design schema. It still remains available in the background. Here we can for example add and edit a CTT (figure 6.33). The CTT modeling tool is completely implemented, but for the rest of the phase only a prototype is provided.

6.5.7 Navigational Design

In the navigational design (Figure 6.34) we have a very similar design as in the conceptual modeling phase. The same features are visible but got another notation. The relations of the conceptual design phase have been removed and replaced by links. On this page we can only add or remove links. Clicking on a feature won't have any effect. We can change the audience class again by clicking on the button. For this phase only a visual prototype is provided, no interaction is possible.

6.5.8 Site Structure Design

In the site structure design (Figure 6.35) we see the same model as in the navigational design. The actions we can do on this model have changed however. We can only create pages and assign features to them. We can either add a page as a single page or as a set containing multiple pages, depicted with the * icon. The button to change the audience classes is again present. Pages can be given a name to identify them in the template design phase. For this phase only a visual prototype is provided, no interaction is possible.

6.5.9 Template Design

The last phase is the template design phase. This phase is not mandatory in the support tool so the web developer is allowed to leave this phase empty. The developer can select a page out of the list specified in previous phase. When a page was designated as 'multiple' in the previous phase, more than one page template can be created. The specification of page design template consists of adding rows to a grid and cells to rows. These cells can be of various types and a design image can be uploaded to give a cell a graphical representation.

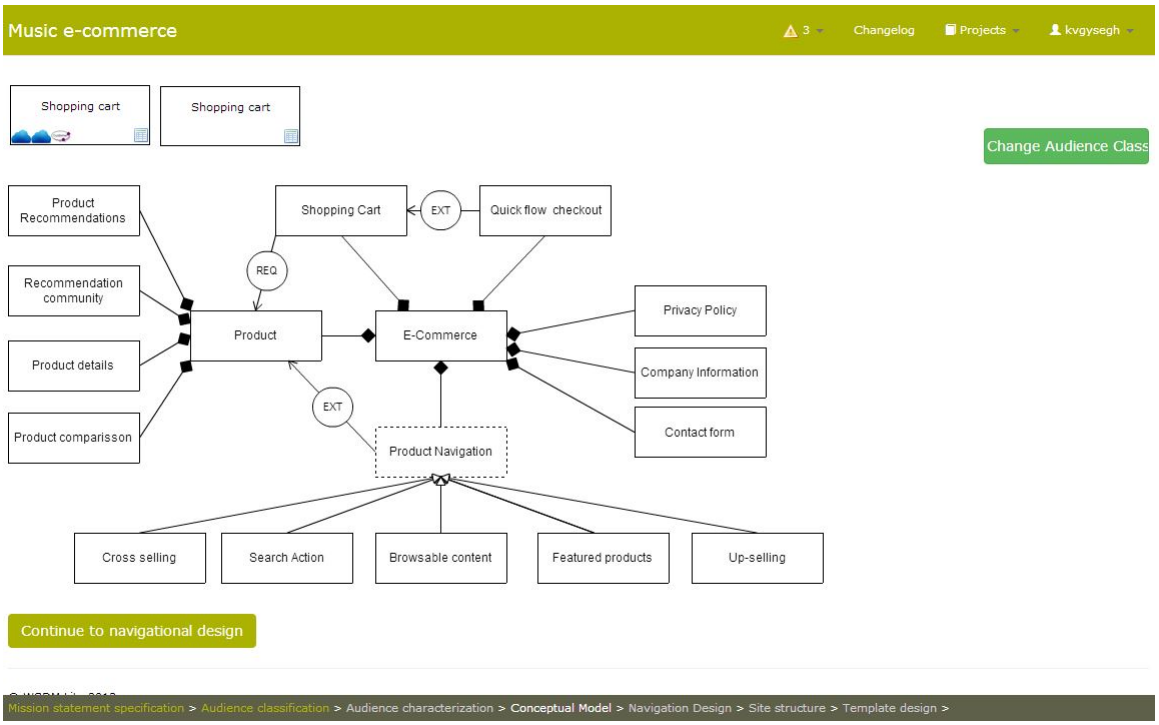

Figure 6.31: Screenshot: mission statement specification and website genre selection

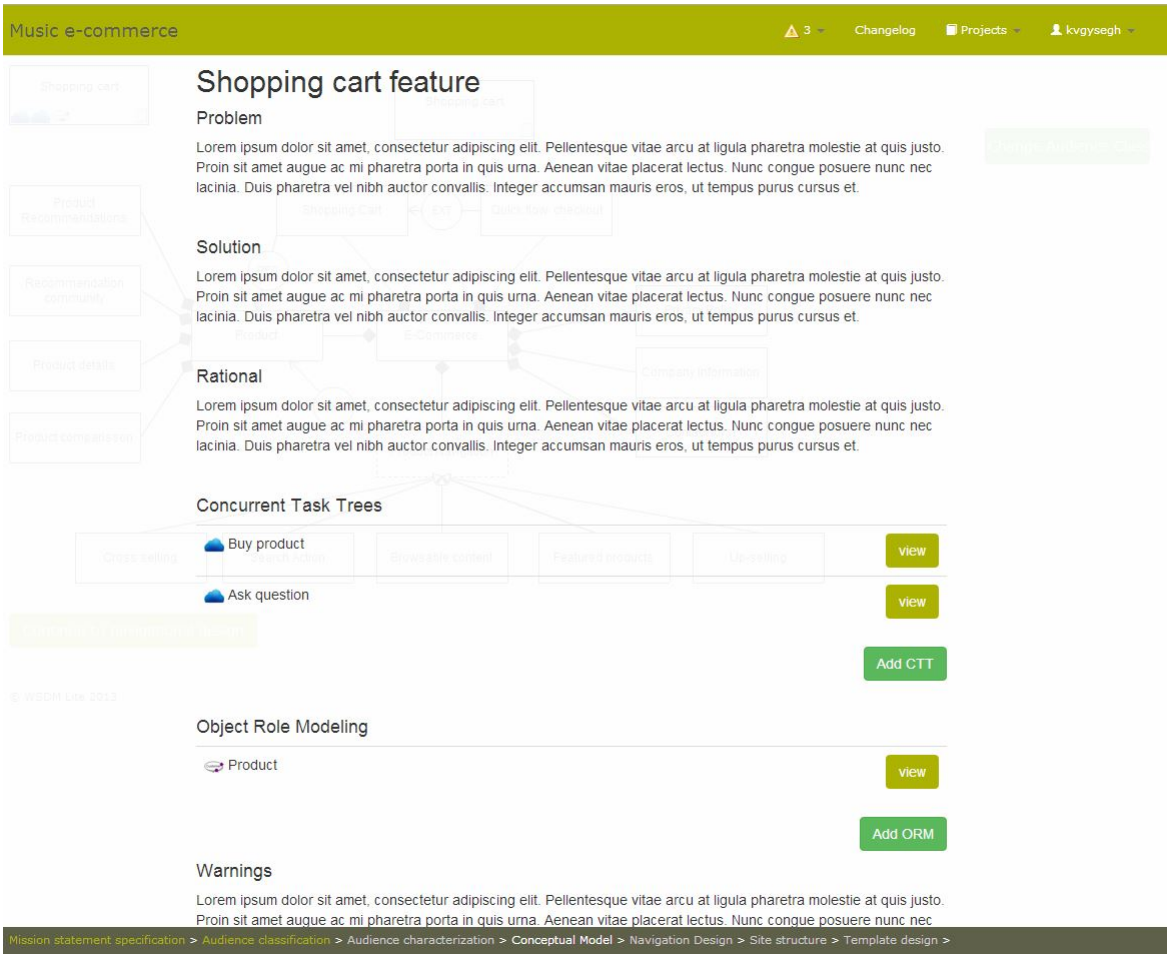

Figure 6.32: Screenshot: mission statement specification and website genre selection

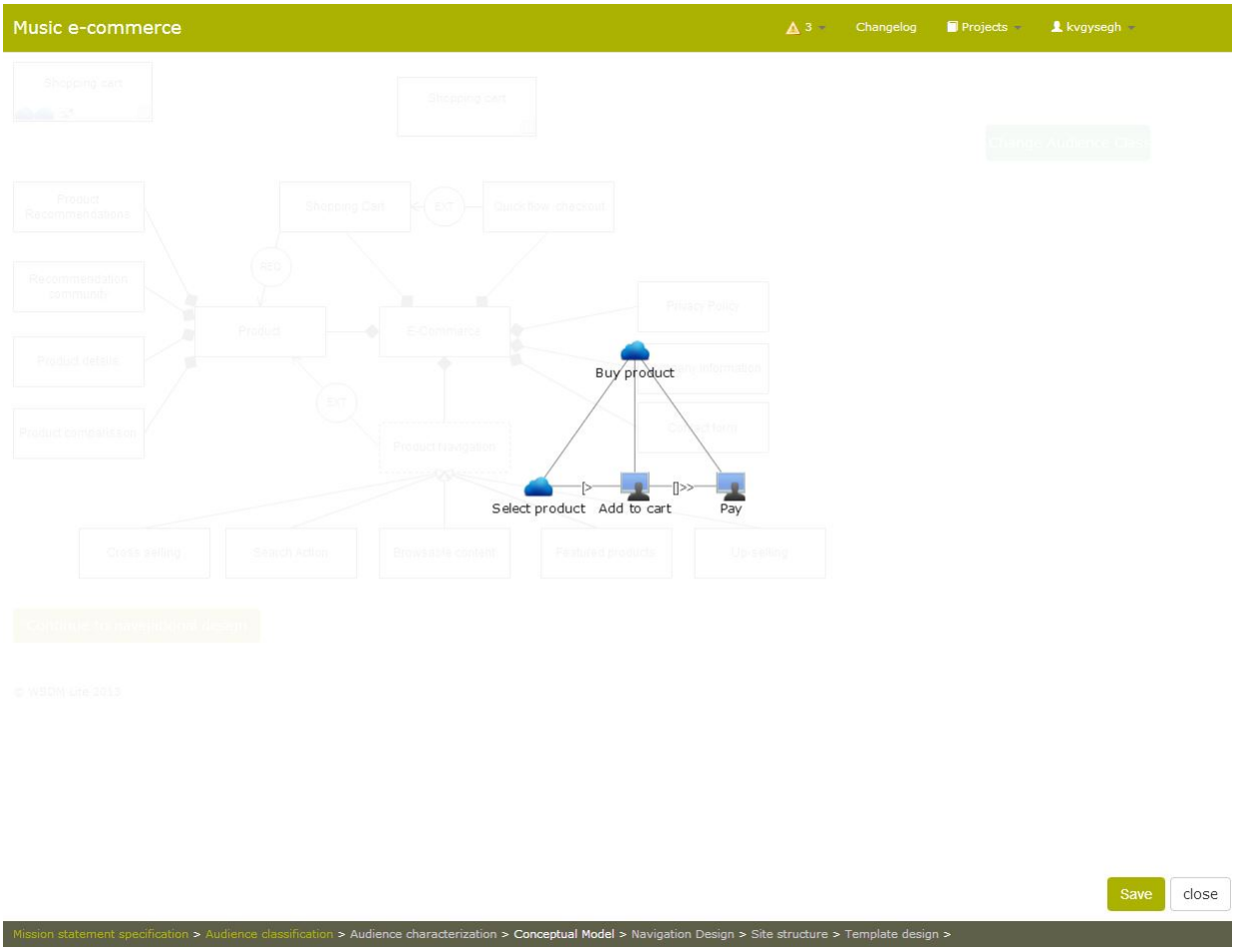

Figure 6.33: Screenshot: mission statement specification and website genre selection

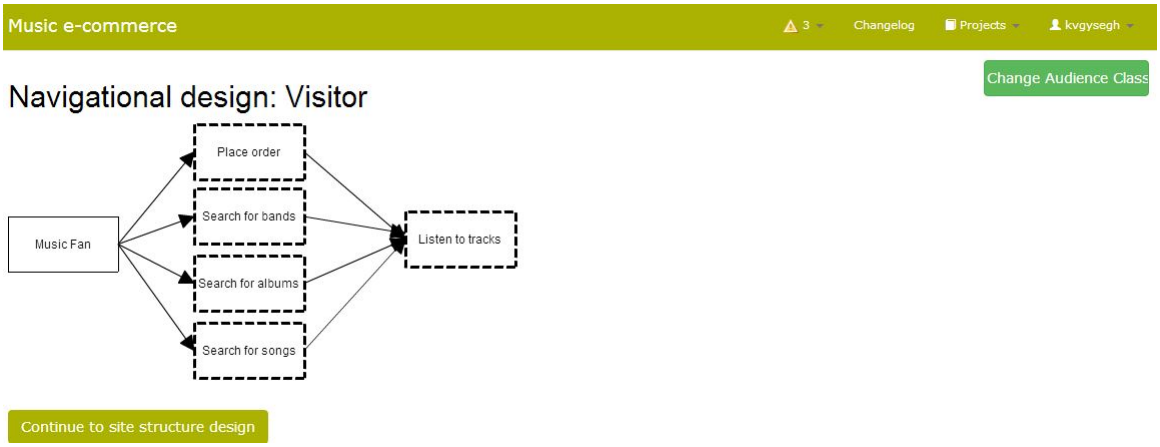

Figure 6.34: Screenshot: mission statement specification and website genre selection

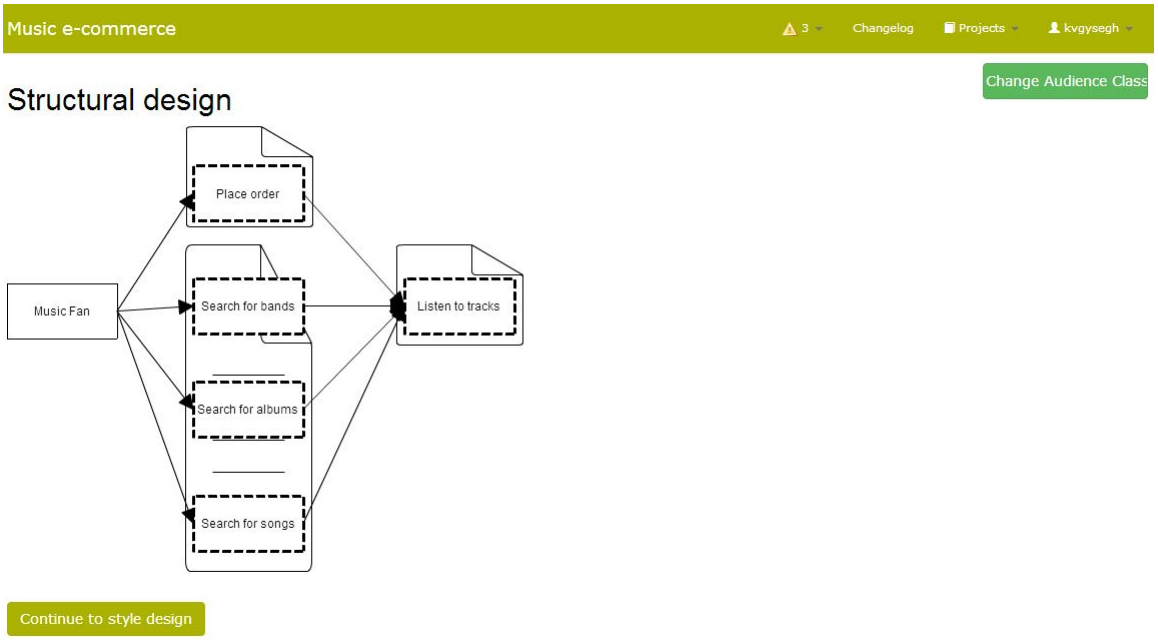

Figure 6.35: Screenshot: mission statement specification and website genre selection

7

Example: E-commerce

After our discussion in chapter 4 about the WSDM-Lite methodology and the discussion of the support tool in chapter 5, we will present an example. The website to be designed is similar to the one given as example in section on the original WSDM (section 2.1), in order to show the difference between the two methods. We also use the example to illustrate the use of the support tool.

7.1 Mission Statement Specification and Website Genre Selection

The mission statement specification is exactly the same as in the original WSDM version, but now we can enter the information in the support tool as seen in figure 7.1.
To minify the number of different screens the website genre selection is on the same page as the mission statement specification in the support tool. The website genre for this project is obvious e-commerce.

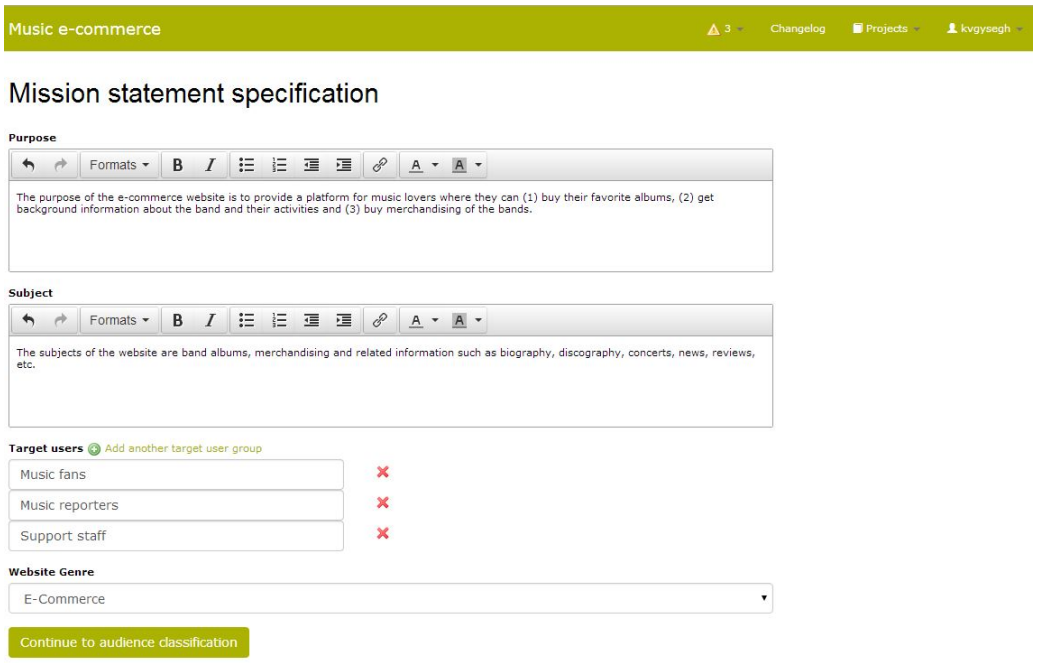

Figure 7.1: Mission statement specification and website genre selection of music e-commerce

7.2 Audience Classification and Audience Characterization

The audience classification is also the same as in the legacy WSDM version. Again we can enter the requirements in the tool. By entering each requirement individually we can connect a requirement to a WDP as discussed in section 5.5. Note that the user can decide in which order to place the requirements. By checking the appropriate checkboxes to the right of each requirement as seen in figure 7.2, the user does the audience classification, i.e., he indicates which requirement belongs to which audience. The checkboxes matrix contains the same data as the requirement matrix, but is represented in the order chosen by the user (i.e., designer).

The result of this phase is exactly the same audience classification as in the legacy WSDM version since requirements are not directly related to an implementation strategy. The output of this phase is the audience classification, provided for the example in figure 7.3. The audience characterization (figure 7.4 and 7.5) is also exactly the same as in the original WSDM version.

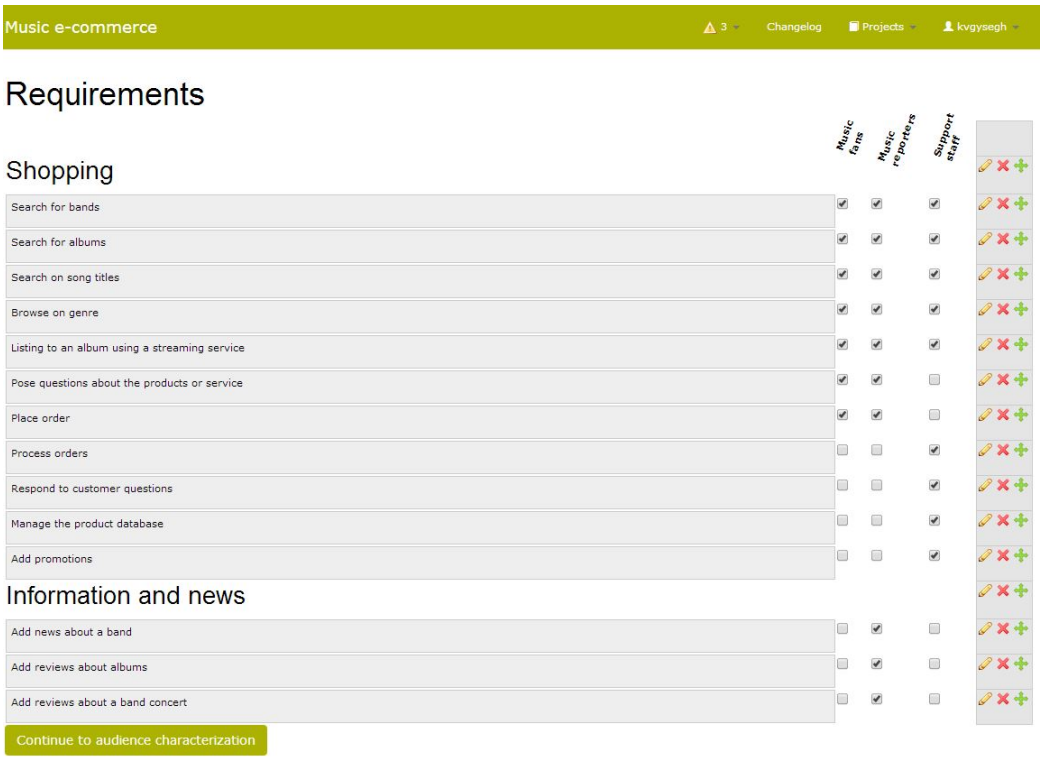

Figure 7.2: Audience classification of music e-commerce

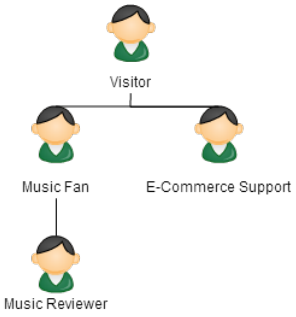

Figure 7.3: Audience classification hierarchy of music e-commerce

7.3 Conceptual Model

7.3.1 Initialization of the Conceptual Model

From the conceptual phase on, the difference with the original WSDM version becomes more notable. When opening the conceptual phase for the first time, the tool suggests several WGP to select from as seen in figure 7.6. It is up to the designer to select the one most appropriate for his system.

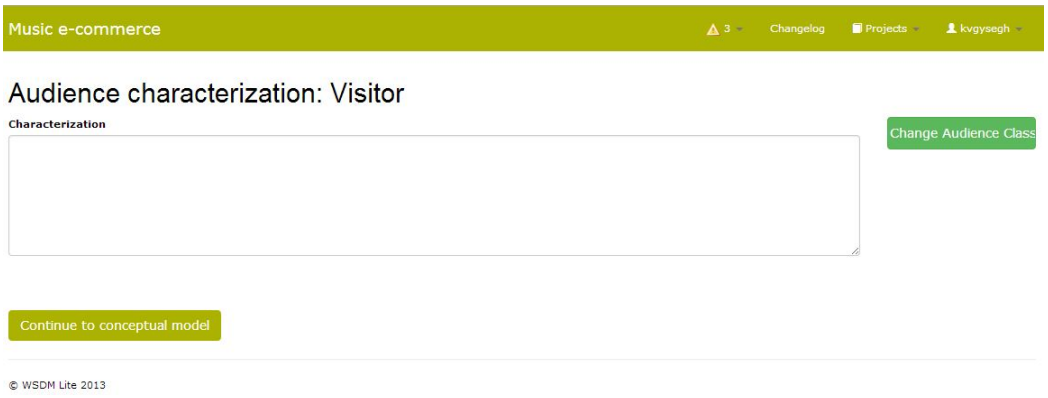

Figure 7.4: Audience characterization of visitor class

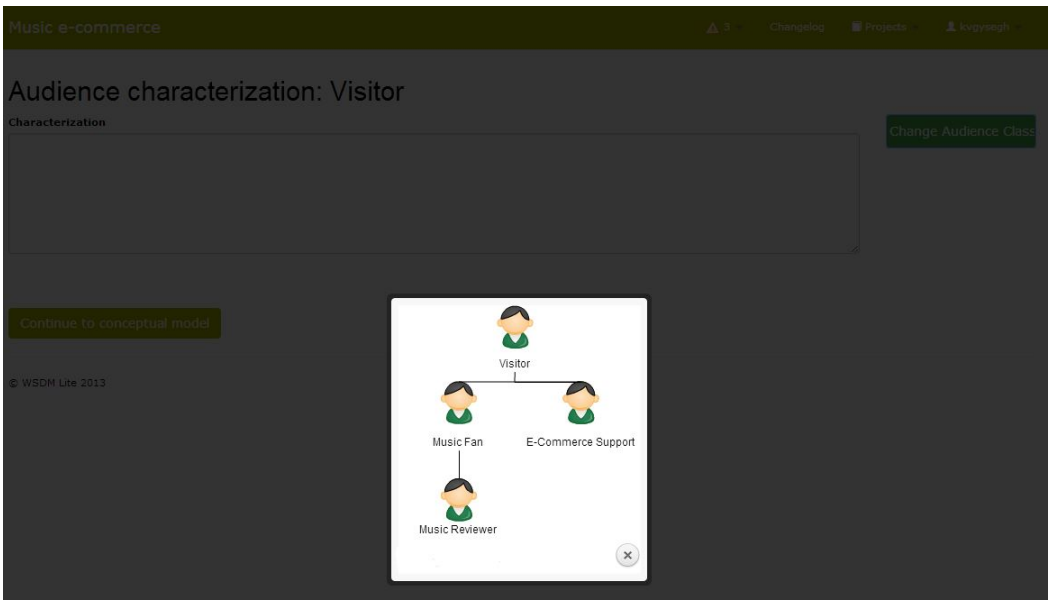

Figure 7.5: Changing audience class for audience characterization

Figure 7.6: Selection of a WGP from the WGP repository

After selecting a WGP the conceptual model is initiated and the designer is able to modify the WGP to the needs of the project. Because he selected

a WGP and did not create a new one, he sees WDPs in the WGP that may be he did not think of (see figure 7.7). In this example, the designer chooses not to remove them from the design because it gives the web shop a more complete design. The WDP 'Product recommendations' is for example added which was not present in the requirements phase. It can be added to the requirements in a next iteration.

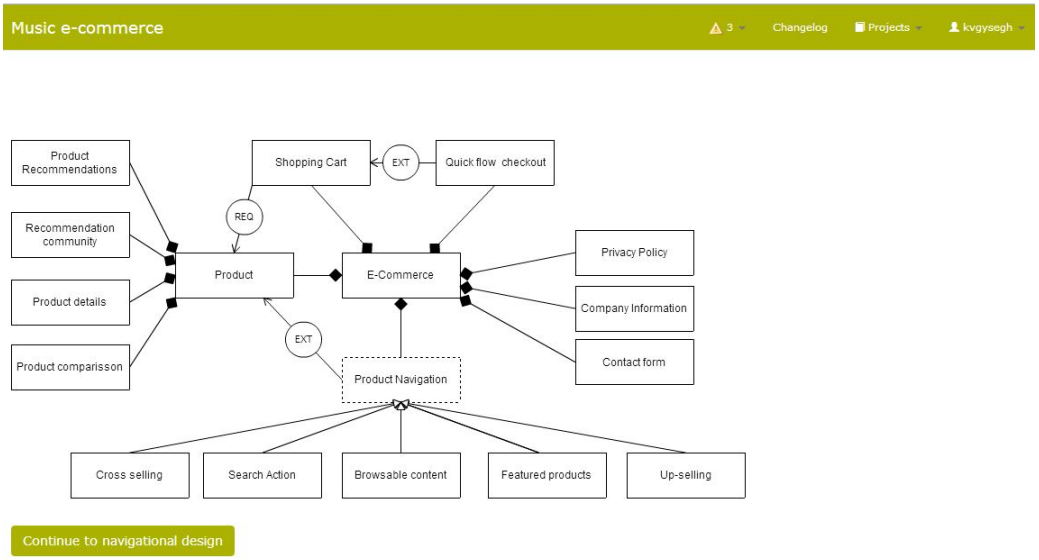

Figure 7.7: Conceptual design

The WGP is a conceptual design for the whole system but doesn't take the different audience classes into account. To do this we take a top-down approach in the audience classification model: the designer starts with the visitor class and removes all WDPs that are not relevant for this class. The WDPs that are removed from the conceptual model for this class but are not (yet) removed for the classes that extend the visitor class (and their children). The effect of the removal of a WDP is that the conceptual models of the child-audience-classes of the visitor class extend its conceptual model by one WDP, the one that was removed.

In this project, the shopping cart and the quick flow checkout are not available to the visitor class neither for the e-commerce support staff. By removing these WDPs they become only available for the music fans and music reviewers. Removing a WDP is simply done by selecting it and pressing the delete button.

When a WDP should not be present for any audience class, the WDP should be removed for each audience class. To make this more user-friendly, the tool introduces a second way to delete the WDP such that it is removed for all classes and disappears in the project. Removing a WDP from all conceptual schema's is done by clicking right on a WDP and clicking the action 'remove all'.

A top-down approach needs to be taken because when we should remove a WDP from a child audience class and this WDP remains present in the parent class, then the audience classification will violate the definition given for the audience subclassing by WSDM, i.e., an audience class can only extend the requirements of the parent class.

7.3.2 Adapting the Conceptual Design

Once the initial conceptual model is divided properly to the different audience classes, the designer can start adding extra WDPs. More experienced users will be able to do this concurrently with the task described in the previous section. When we go over the requirements of our example, we notice that the requirements 'Process orders' , 'Add reviews about albums' and 'Add reviews about a band concert' are not covered by the current conceptual model, so we will have to add WSPs to satisfy these requirements.

The review system actually consists of two parts: The reviews related to the products and the review management system for the music reviewers to manage their work. The reviews may be available for the visitors, music fans, music reviewers, and e-commerce staff. Based on the inheritance concept of the conceptual models we should add it to the class closed to the root of the audience hierarchy because the WDP will then be present in all child classes as well. In this case, this is the root itself, the visitor class. The conceptual schema of the Music Reviewer audience class is given in figure 7.8.

The order processing is added to the conceptual schema of the e-commerce support staff, which does not contain the review management (figure 7.9).

7.3.3 Modeling the Details

By now, the designer has modeled the global structure of the web application. We added for example the 'Order processing' WDP to the system, but the details of this new WDP are not yet specified. Until now we only gave it a name but the designer can elaborate this WDP more as discussed in chapter 4. We illustrate this for the 'Review' WDP:

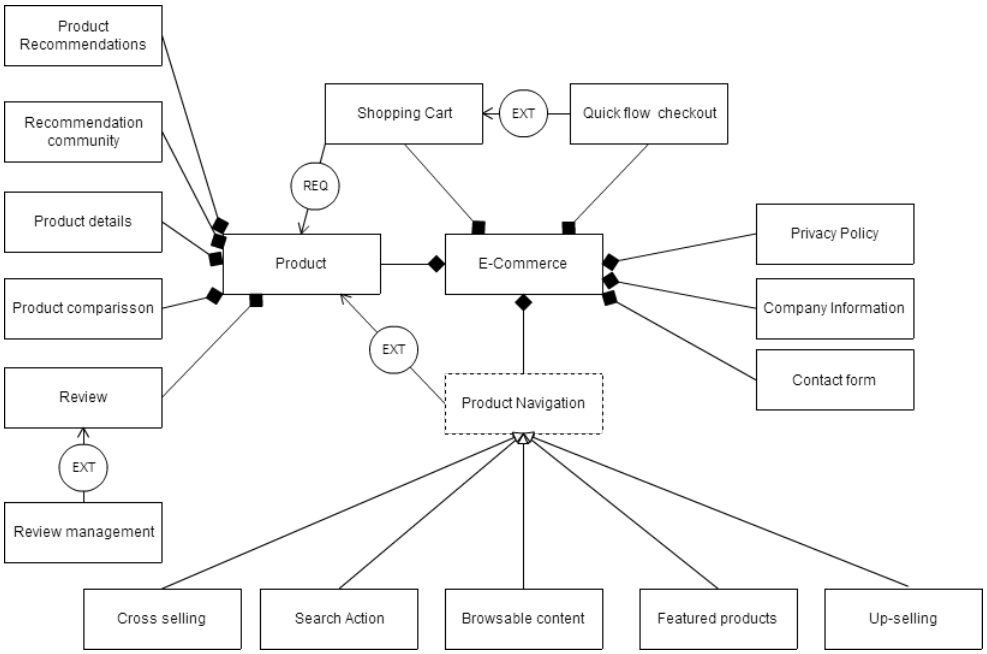

Figure 7.8: Conceptual design of music reviewer audience class

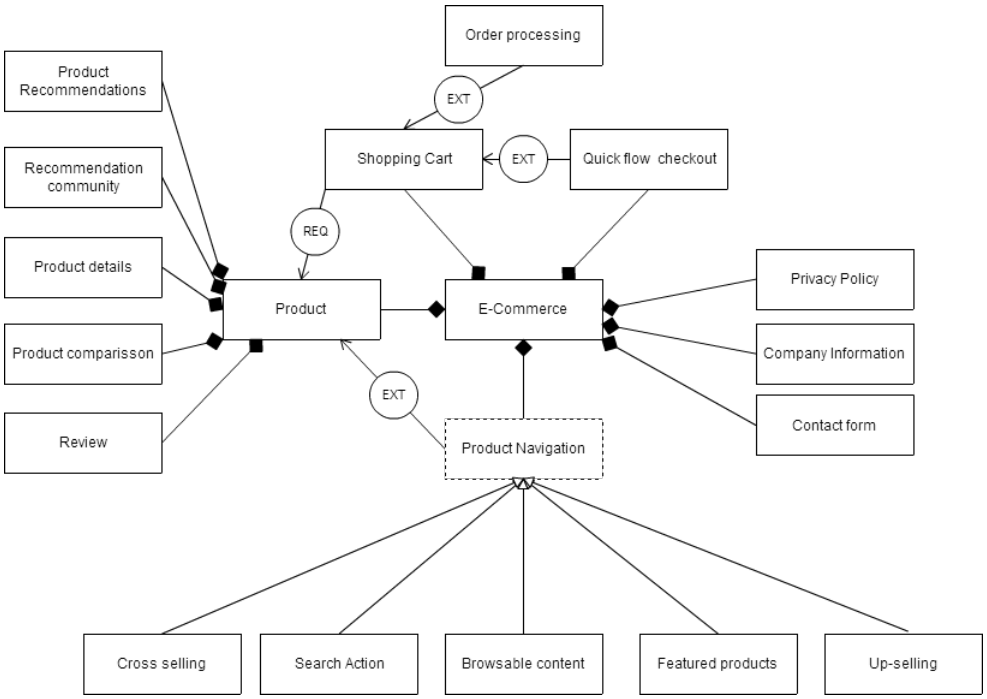

Figure 7.9: Conceptual design of the support staff audience class

- **Problem:** If a new album comes out people do not know yet a lot about it. If they don't know a lot about an album they will hesitate to buy it.
- **Solution:** By providing professional reviews about albums and bands the visitors of the website they learn to know the band and get more inclined to buy an album.
- **Rational:** The review contains a score for the quality of the album (1 to 5 stars), a number by number discussion and a conclusion. The review also contains references to older work and youtube movies of this older work. People can listen to that and will probably know the older songs.
- **Object chunk: :**

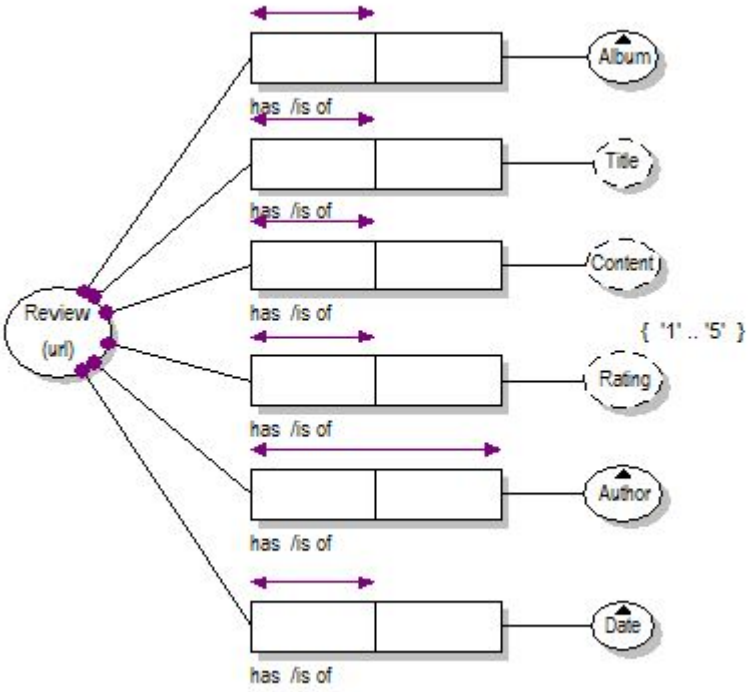

Figure 7.10: Object chunk for 'Review'

Each WDP in the conceptual modal should be more elaborated with at least the problem statement, solution and rational. Optionally warnings can be specified and the structure. The structure contains specifications about

the task (CTT) and the used information (ORM object chunk). Most of the conceptual model was initiated by the WGP, which contains already existing WDPs. These existing WDPs already provide detail information. All the designer needs to do is verify whether the specified information meets the project requirements. For our example we need to modify the object chunk for the "product" WDP such that it contains the information about our products (i.e., albums with release date, artist, ...). See figure 7.10.

The conceptual phase for our project is now finished.

7.4 Navigational Design

The navigational design determines which WDPs are accessible from which WDPs. It is possible that links between components are not cross page hyperlinks but, for example, anchor links. Anchor links can even be omitted if the WDPs are small and they are closely positioned to each other on a single page.

When the conceptual phase is completed, a corresponding navigational design model is initiated automatically for each audience class. This model contains every WDP of the conceptual schema for this audience class with all links, by default, replaced by navigational links since WDPs that are connected in the conceptual schema also tend to be connected in a navigational design.

We again take a top-down approach for creating the navigational design, since the links that are added in the navigational design of a parent class will be inherited by the child class. In figure 7.11 we see the full navigational design for the music reviewers audience class, which inherits from the music fans and the visitor class.

7.5 Site Structure Design

In the structural design, we model the pages that will be available in the web application. When opening this phase in the tool, the designer sees the navigational design from the previous step. The only action he can do now is selecting components and assigning them to a page or a set of pages. This phase is closely related to the next phase where the design of the structure

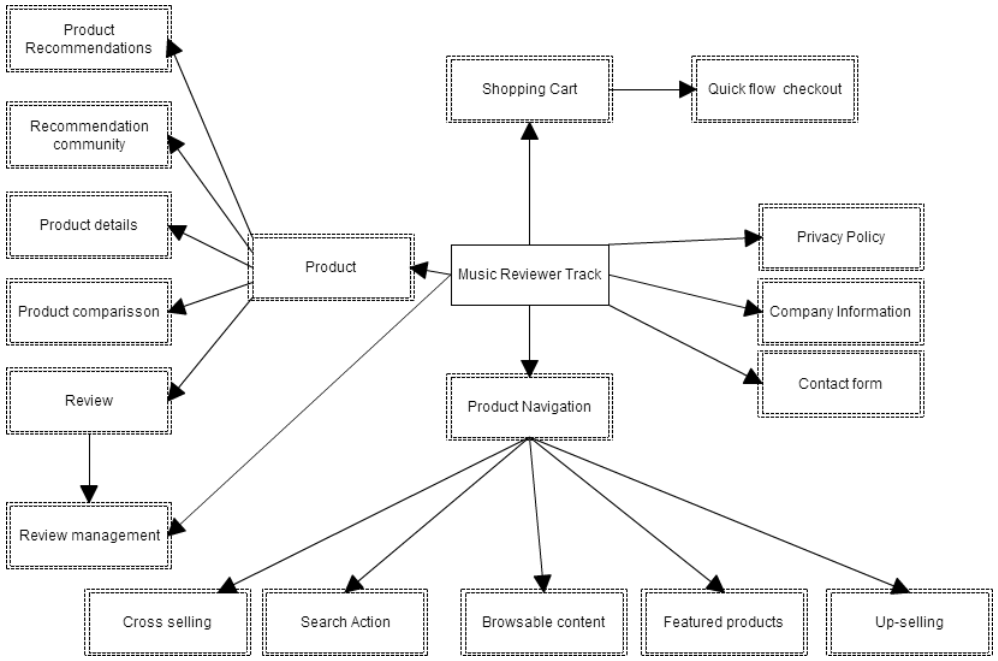

Figure 7.11: Navigational design for music e-commerce

of these pages is made.

Note that the inheritance is different from the original WSDM version. In the original WSDM version, the navigational design had inheritance in the sense that the music reviewer track would connect to the music fan track and all the links of the music fan track would be available. Here we have inheritance in the same way as for the previous phases: by going to a lower audience class in the audience hierarchy we expand the model with WDPs only relevant for that audience class.

The structural design for the music reviewer track is shown in figure 7.12. We see that some WDPs are displayed on just one page (Contact Form), some WDP are displayed on several pages (Review management) and some are displayed with other WDPs on just one page (Product). The product page could contain a content table with anchor links to go to the different parts on the page.

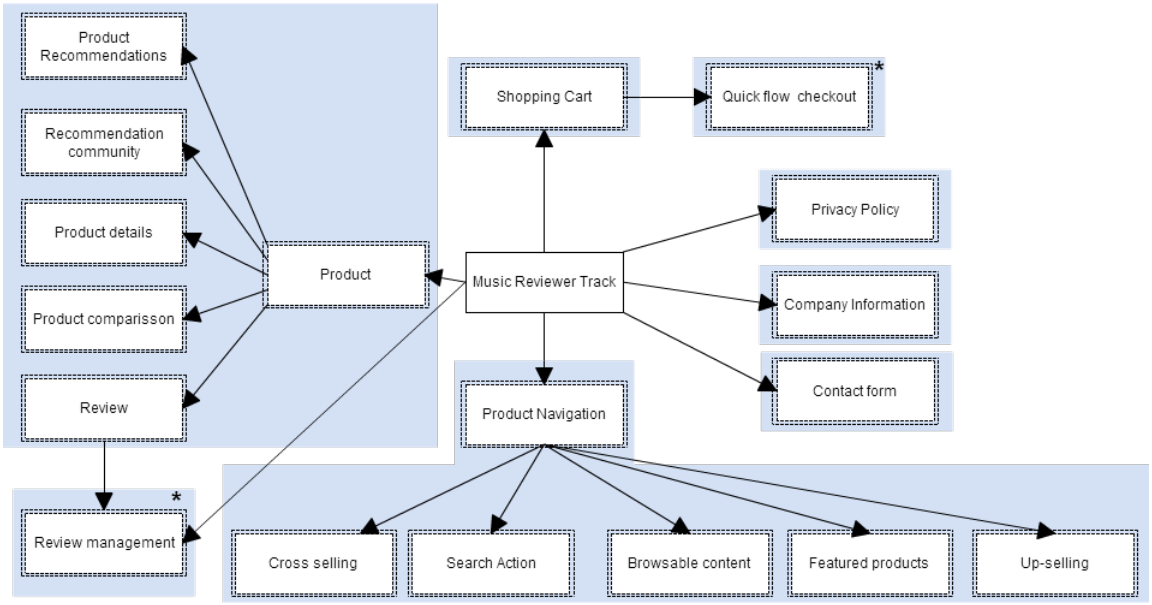

Figure 7.12: Navigational design for music e-commerce

7.6 Template Design

The structure of our example website is now well defined and we have definedmodeled the different pages that will be available in the system. The only thing left to model is the structure of the individual pages. As discussed in chapter 4, there are two possibilities: create a template directly for a content management system or use the template modeling technique from the original WSDM version. In this example we will do the latter. The pages we need to design are:

1. Product overview (Product navigation)
2. Product details
3. Shopping cart
4. Privacy policy
5. Company information
6. Contact form
7. Shopping cart
8. Quick flow checkout*

- Enter customer information
- Enter delivery information/method
- Choose payment method
- Summary
- Confirmation page

9. Review management*
    - Reviews overview
    - Create review
    - Edit review

It is possible that one design will be used for more than one page such as the privacy policy and the company information. In this example we will limit ourselves to the design of the product overview (figure 7.13) and the product details (figure 7.14). Both are based on a free available template[1]. More specifics about the design of the templates can be found in [8].

[1] http://themes.shopify.com/themes/minimal/styles/music

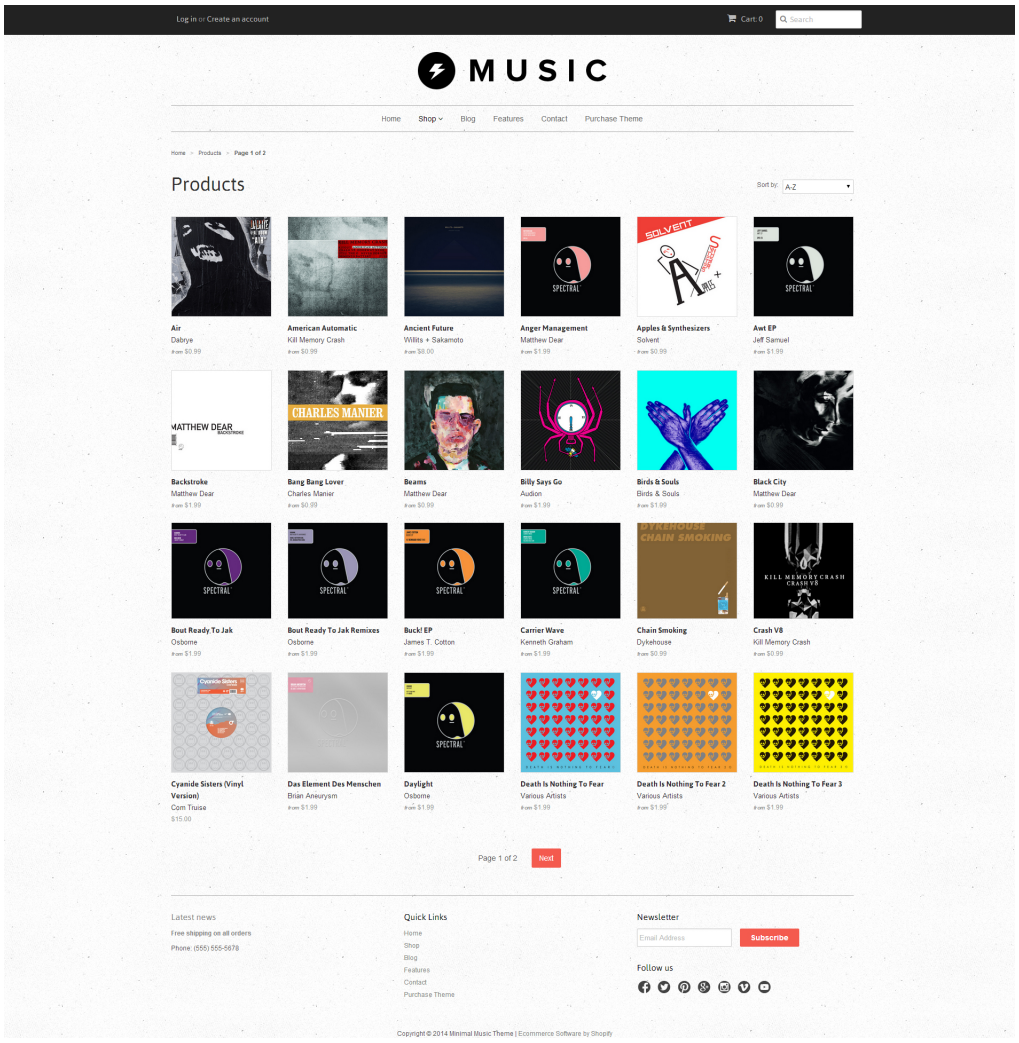

Figure 7.13: Page design for a product overview

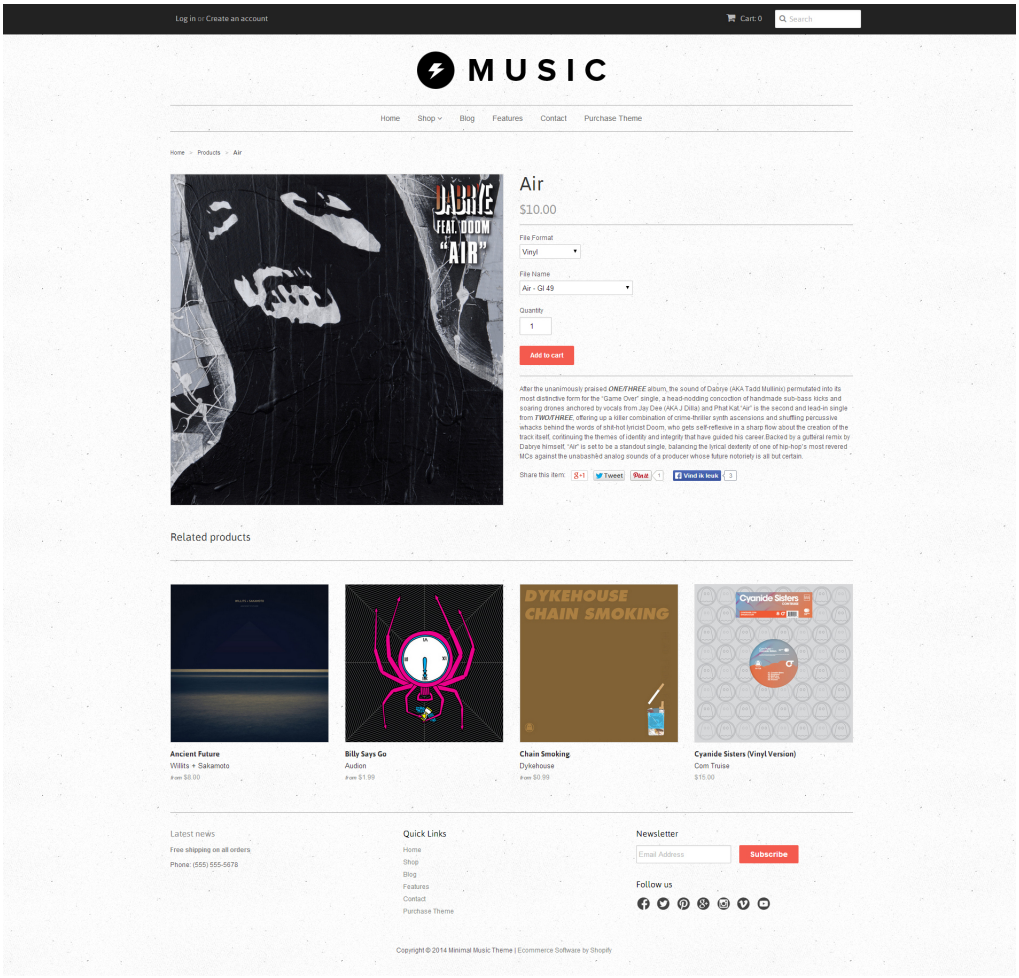

Figure 7.14: Page design for product details

8

Website Generation

Models specify applications by describing them at various levels of abstraction and by specific views on the application. The application can be implemented based on these models. To speed up the implementation, and to ensure that all models are followed when implementing the system, it is useful to research to which extend and with which limitations these models can be used to (partially) generate the application. A discussion of implementation generation for WSDM was already done [29], we will take a similar approach here but using the specifications of WSMD-Lite. There are several possibilities, each with their own benefits and drawbacks. We discuss them in this chapter. The focus is on an implementation based on a CMS. In the first approach the WSDM-Lite support tool will try to exactly replicate the actions a web designer would undertake to install and configure a CMS. It will become clear that this approach is very open and due to this fact it will be very hard to partially generate a website. Therefore we will introduce a second approach in which the WSMD-Lite methodology itself will be restricted and a best possible website will be generated. A third approach will restrict the WSDM-Lite methodology even further and we see that the methodology approaches a new domain.

8.1 Installing a CMS

We can identify several steps common to installing any CMS. Source files of a CMS must first be uploaded to a webhosting with supporting technologies (PHP, MySQL, ...) (1). Most CMS like drupal[1][24] or wordpress[2][3] have a wizard-installation that can be used after all files are uploaded. This installation is a user-friendly way of setting up the rest of the CMS. However, on a technical level this boils down to three actions: rewriting a configuration file with for example the database connection information (2), setting up file and folder permissions (3), and initializing the database with the tables and content needed for the CMS to work (4). All four actions can be done relatively easy on a technical level and this won't be the issue when we want to initialize the CMS. Installing custom modules also consists of these 4 actions or less, depending on the type of module. The real problem for automated (partial) website generation consists of selecting an appropriate CMS and selecting appropriate modules. This is more a semantic issue because we need to select existing modules and map these to features we used conceptually. This mapping may no be an easy task. We will use the term 'modules' for the installable of interchangeable parts in CMS although they each CMS uses a different term for this concept (e.g., plug-ins in WordPress).

8.2 Approach 1: The WSDM-Lite Support Tool in the Role of Web Developer

8.2.1 Technical

In the technical setup of this first approach the WSDM-Lite support tool takes over the role of the web developer manually installing a CMS and the modules. The WSDM-Lite tool is able to upload a CMS to a specific webhosting using FTP, setting up the file permissions, writing configuration file data and making a connection to a database to initialize the database. All these actions can be implemented relatively easily using a server side language such as PHP, ASP, or Java.

[1] www.drupal.com
[2] www.wordpress.com

8.2.2 Linking Components on WDP Level

Approach

In the semantic part we discuss the selection of a specific CMS and it's modules. A first naive approach could be to have corresponding modules attached to conceptual WDPs. The advantage of this approach is that we have a fairly straightforward mapping. Several modules of different CMS can be attached to a WDP. To generate the web application the tool could select the CMS that best covers the required WDPs, i.e., for which all or the most WDPs has a related module. The modules can then be installed directly.

Issues

This naive technique is very straightforward but a number of issues arise when using this approach. We will try to eliminate these issues in other approaches we will discuss further on.

- **Number of WDPs:** Across all the WGPs there exist a lot of features. Each feature can in turn be modeled in several ways. This will result in a lot of WDPs. Maintaining relations between all those WDPs and the related implemented modules is a near impossible task.
- **Number of CMSs:** There exist a lot of CMSs. One WDP may have related implementations in tens of CMSs.
- **Number of modules:** Several modules may implement a certain WDP.
- **Number of CMS Versions:** Some CMS do not update with backward compatibility. For examples: modules implemented for Drupal 6 are not usable in Drupal 7. The modules and the CMS in which they are available need to be tracked.

With the four issues mentioned we clearly see the bigger issue of the vastness of possible connections: #WDPs x #CMS x #modules x #CMSVersions = #Total number of possible connections. Maintaining these connections as WSDM-Support tool administrator is clearly not possible. A possible solution could be crowd sourcing, however, there are still other issues that make this approach infeasible:

- **Level of abstraction:**When we take for example the WDP products and reviews, these can be implemented individually for a specific CMS

but a better solution would be a module where they are both combined. If we map the WDP to modules on WDP level we cannot (easily) take this into account.

- **Features provided by CMS core:** Some features may be provides by the core of the CMS. When all CMSs provide the feature, this is no problem but consider the following example. A shopping cart is directly available in the CMS OsCommerce but not in the CMS Drupal. If we have a mapping between WDPs and modules we cannot (easily) take this situation into account.

- **Conflicting CMS selection:** Imagine the situation where we create a new WDP, which will be quite common in projects. This WDP has no implemented module associated with it. It is possible that this component can only be realized in for example Drupal, but for the other WDPs WordPress would be a better fit. This has as consequence that the wrong CMS may be selected.

- **Motivation for crowdsourcing:** WDPs and WGPsare not a fixed set but can be created and shared freely by web developers. Which means that quite a lot new patterns will be created. When a WDP has no related module, the web developer has search for it manually. Web developers may not have a lot of incentive to couple a found module back to the WDP and make it available to other web developers since they don't have a direct benefit from doing this, unless they will use it again in a later project.

- **Technical representation of the modules:** This approach assumes we have a database of modules available in the WSDM-Support tool. This database must for each module hold the files of the module, the configuration settings, the file permissions, and the database structure. Together with the number of possible connections it is impossible to maintain this manually.

8.2.3 Linking Components on WGP and WDP Level

Approach

Instead of having a mapping between WDPs and modules, we could associate a WGP design to a specific CMS and a set of modules. Instead of automatically finding a suitable CMS, modules related to WDPs, and verifying if there is a combination of versions that work, this combination could

be specified beforehand. You can look at it as a 'snapshot' of an installed CMS with all the needed modules for that WGP. When a combination of features is implemented by one module, only one module must be specified. When features are already implemented by the core of the specified CMS, no module needs to be added.

Issues

Solved issues:

Number of WDPs, Number of CMSs, Number of modules, Number of CMS Versions, Level of abstraction, Features provides by CMS core

Issues that still remain:

Conflicting CMS selection, Motivation for crowdsourcing, Technical representation of the modules

New issues:

- **Number of WGP:** A set or one CMS and several modules still need to be specified for each WGP. Because each WGP contains several WDPs, it is reasonable to say that there will be less WGP than WDPs, so an improvement is made. However, maintainability will still remains an issue.

- **Updates of WGP:** A WGP is mainly used as a basis to design a web project. The web designer can add and remove WDPs in order that the design fits the requirements. When the designer updates the conceptual model the modules specified in the WGP will no longer match the required ones. This can be fixed in a limited way by defining which WDP use which module. If there are no more connections between a module and WDPs, the module is no longer needed and doesn't need to be installed. However, for added WDPs no modules may be available in the preselected CMS.

8.2.4 Module Market Place with Search

Approach

When we look at the websites of WordPress or Drupal , we notice that both have a searchable index of all the available modules. If one would be able to combine these indexes for a large number of CMSs and make a similar page

in the WSDM-Lite tool, a web developer could find suitable modules across CMSs and connect the WDP with modules in a module selection phase at the end of the method as suggested by Sajjadi[18]. The critical issue with this approach is that these indexes of modules are not freely available and have limited and rather simplistic search mechanisms, which make it hard to search easily for a suitable module.

Based on the vast modules available at both WordPress and Drupal,we suggest that it would be possible to setup a similar platform attached to the WSD-Lite tool where developers can add their own modules because there are no real differences in incentive to uploading a module to for example the Drupal site. By allowing developers to upload their modules directly to the WSDM-Lite platform more meta-information could be added such as the website genre so that the module can be found easily.

Issues

Extra issues solved:

Technical representation of the modules (the developer knows how it should be installed), Conflicting CMS selection (mapping can be done during the search), Motivation for crowdsourcing (mapping can be done during the search), Number of WGP (matching is done manually at runtime), Updates of WGP (finding modules is only done after WGP are updated).

Issues that still remain:

All issues mentioned in the two approaches before are solved

Extra issues:

- **Success of marketplace:** Although people seem to upload their modules to the website of a specific CMS, it is unlikely that a lot of people will do this for the WSDM-Lite tool: a big incentive is the fact that these website have a lot of traffic and the chance that their module will actually be downloaded and used is large. The WSDM-Lite tool is a new project and at this moment still a prototype. Developers won't have a lot of downloads when they add their module to our system. On the other hand, when there are not a lot of modules available in the marketplace, the marketplace won't be successful, which is a vicious circle.

8.2.5 Conclusions

The first two approaches are based on a manual maintenance of the mappings between the conceptual model and implementation modules by the web developer of the project. Maintainability is one of the biggest issues that comes with this approach but technically it should be possible.

The third approach uses a marketplace similar to the marketplaces of specific CMS tools. The mapping of the conceptual model to an implementation is still done by the web developer but is supported by a search engine. Modules are directly linked to the conceptual model so users can use this mapping. This approach seems the most appropriate one to keep on working, however it is unlikely that enough developers will maintain the WSDM-Lite marketplace.

8.2.6 Alternative Technical Setup

In practice, the choice of CMS is sometimes based on more than the availability of modules but also on issues such as overall quality of the CMS, possibilities to expand, personal experience, and so on. Therefore we could opt to provide module selection only for a particular CMS, e.g., Drupal, because of its quality, popularity, community support, etc. An extra WSDM-Lite module could be developed for this CMS that connects to the WSDM-Lite tool through an API. The WSDM-Lite support tool doesn't need to decide anymore which CMS to choose and doesn't need to maintain available modules of every CMS. This makes the WSDM-Lite tool more maintainable and the data collected in the WSDM-Lite tool can be also used by any application. A lot of web hosting servers do not allow external connections to the database, which was a requirement of the original setup. By doing all communication between the WSDM-Lite support tool and a CMS, we do not only resolve this issue, but we can make the communication more secure[16].

8.2.7 Other Advantages of Connection WDPs and Modules

The link between a WDP and a module is used to initialize the CMS when a website is constructed. This link can also be used over time to validate if the documentation is still valid with the CMS. When a module is removed from the CMS in the administration panel of the CMS, the WDP will have lost its connection and the web developer can be warned of this. When a new module is installed, this module won't have any connection in the conceptual

design. The web developer can be warned of this and update the models accordingly.

8.3 Approach 2: Strictly Limit the Mapping

It is clear that the biggest problem with translating a conceptual design into a mapping between WDPs and modules is the amount of possible connections between the vast number of WDPs and the vast number of implemented modules. There are no means to automate these connections so everything has to be done manually. Although this manual labor can be done by a group of people, it is still cumbersome.

Only by exhaustively limiting the options in the conceptual design we can create a more realistic setting in which actual website generation seems more possible. The main difference between this approach and the ones mentioned before is that previous approaches were additions to the current WSDM-Lite methodology and this approach also restricts the WSDM-Lite methodology. The technical setup of this approach can be either the original or the alternative setup discussed in previous section.

8.3.1 Limited WDP for each Website Genre

The WSDM-Lite methodology introduced in thesis allows for web developers to create their own WDPs. On the other hand, we have the vast amount of available modules. We can limit the number of possible connections to create and maintain by limiting the available WDP and/or limiting the modules that relate to these WDP. In this approach, we will do both.

First we limit the number of available WDPs for the web designer. Web developers are no longer allowed to create their own WDPs. This significantly decreases the number of available WDP. Secondly, we merge all possible WDPs of a feature into one, but ensure that this single feature is adaptable enough to account for most variations. This results in a limited and fixed set of WDPs that can be used in a project of a specific website genre.

Next we limit the number of available modules. The easiest thing to do is to select modules of only one CMS. Next we will select for each WDP available in the conceptual phase, which was a fixed and limited set, one module. Because we merged several WDPs into one we need the module that implements the most common cases. Note that modules that implement the

most common cases, directly relates to the popularity of that module, which is information that is provided by most module indexes.

Issues

The problem with this approach is that is very restrictive. We have no control over which modules will be installed. This issue is not that hard to overcome because when the website is installed, we can still remove the module if it does not fit our needs and install another one.

Another problem is that the designer can no longer model a WDPs, which restricts the web developer in using only commonly used features and he will have no possibilities to include new features in his website. This issue can be overcome by not completely restricting the WDPs. A web developer is encouraged to use the available WDPs, which relate to a module, but is allowed to create own WDPs. Custom created WDPs are not related to an implemented module and these WDPs will not be available in the generated website. They should be added manually. WDPs which are commonly used, have a high probability of being available to web developers as implemented modules. When we select the module that is most frequently used, we have the least probability the module will need to be removed and replaced by another.

For custom WDPs are WDPs which do not belong to the most popular ones in the website genre, the probability of an implemented module being available is also low and the web developer would have to create his own custom module anyway.

8.4 Approach 3: Extend the Support Tool with Implementations for the Website Genres

Until now we only discussed installing modules. When we wish to take all models into consideration for the website generation, things become more complicated: we need to use the structure specifications (CTTs and object chunks) of the WDPs to configure the modules, use the navigational models to set up navigational paths of the web application and divide these modules into pages using the site structure design. This implies that we do not only need to install and configure the modules, but also the core of the CMS.

An approach in this direction was made by Mouratidou, Despoina[9] who modified the WSDM methodology to a lightweight methodology but only for the website genre of web shops. All models were used to (partially) generate a web shop in one of three CMSs: WordPress, Joomla and XCart.

CMSs are still not fully adapted to match the models for 100

8.5 Conclusion

In this chapter, we have investigated possible options to partially generate websites using WSDM-Lite. However, this turns out to be harder than expected. The problem why this will be nearly impossible is the maintainability of mapping conceptual designs to the various CMSs, modules and versions. Only when we limit the possible features and as a consequence the possible subsets of these features we obtain a more realistic scenario. There is a clear tradeoff between the complexity we can model and the amount of the web application that can be generated. In the complete and open WSDM-Lite, we can create our own features (WDP) and describe their contents. This gives us a lot of freedom in modeling but it becomes hard to generate a website based on functionality provided by a CMS. By restricting the modeling possibilities we create more maintainable solutions but we are no longer able to model the most complex applications.

9 Future Work

This chapter discusses possible future work. We consider work in the context of the tool support, as well as work in the context of the method itself.

9.1 Tool Support

The tool implemented for this thesis was a prototype and proof of concept. Only some phases of the WSDM-Lite method are implemented and usable. The development of a complete software program requires a lot more work since there are about 7 different modeling tools to be implemented.

9.2 User Studies

The proposals in this thesis still need to be evaluated. This can be done by applying the method on real case scenario's and by web developers. The reason why this is not yet done is because the existence of tool support is vital for this. Testing the methodology with real -life case studies without a tool will be time consuming and laboriously and therefore the evaluation obtained in this way may not reflect correctly the quality of the method.

9.3 User Roles

The methodology focuses only on the web developers in the project. In a real life project more stakeholders are often involved, such as customers, users, project managers, analysts, senior management, quality assurance staf, ...[15] Because the data of each model is individually accessible, views on these models could be created for the other stakeholders. For example, a list of s could be generated and be associated with a cost price. This would allow business people of software companies to generate automatic proposals for their clients. This idea is similar to the WEM [28].

9.4 Template Design

More recently, websites are accessed more and more on mobile devices as seen in figure 9.1, data collected by ShopVisible [1]. The template design phase is still based on a premises that websites have a certain grid and that grid doesn't change to the width of the screen a website is viewed upon. More recent technologies such as Twitter Bootstrap[2] have shown that a design should be able to adapt to the device it is showed on for a better user experience. WSDM-Lite should take this new trend into account either by allowing different but related templates for each possible width or, and preferably, have a template system that uses the same column approach as the implementation would have.

[1] http://www.shopvisible.com/pdf/influence-impact-2013-review.pdf
[2] http://getbootstrap.com/

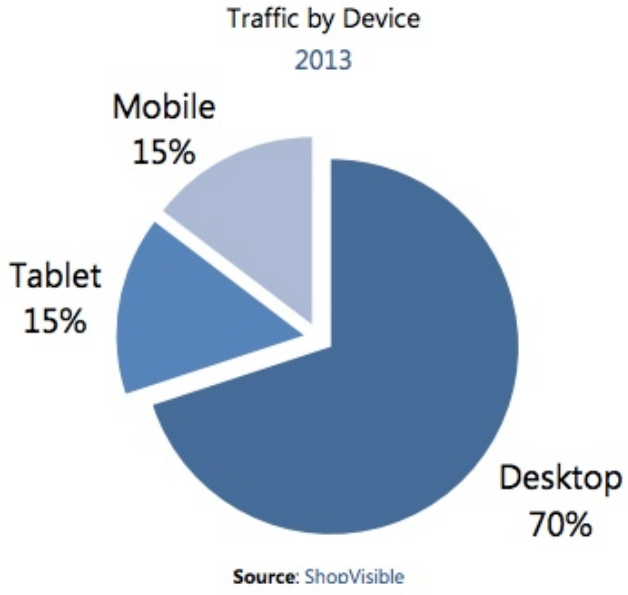

Figure 9.1: ShopVisible

10 Conclusion

In the introduction we stated that the original WSDM version is not compatible with implementations using a CMS because they fundamentally differ in the level of abstraction in which web application are modeled in WSDM and the level of control of the configuration of a CMS. In chapter 4 we introduced the concept of web design patterns which match on a conceptual level the modules that can be installed in a CMS. A feature assembly diagram of these WDP was used to depict the global structure of a web application. The navigational design and the site structure design phases were adapted to the use of this feature assembly diagram in the conceptual phase to follow the philosophy of the original WSDM version in which the output of a phase could be used as input of the next. The use of a CMS significantly reduces the workload of implementing a web application. The use of WSDM-Lite reduces the workload of modeling such an application in contrast to a WSDM design.

Website genre patterns were introduced which contain commonly used global website structures, which were modeled using the feature assembly diagram and WDPs. Using website genre patterns a designer no longer needs to model a web application from nothing but by personalizing a commonly used design. This reduces again the workload of a web developer.

By constructing a WSDM-Lite support tool we further reduced the work-

load of a web developer. The program provides the necessary tools and support to create the models of the WSDM-Lite methodology in an efficient manner. The tool does not only provide the means to create the models but can also take over work from the web developer and even improve the design. The design of the tool was elaborated with a WSDM design since it is not based on a CMS because of its complexity. The introduced concepts were shown in a realistic example for a music e-commerce.

Several approaches were discussed in which the models of WSDM-Lite could be used to partially generate a CMS. The possibilities of this generation was directly related to the restrictions imposed on the WSDM-Lite methodology.

List of Figures

List of Tables

Bibliography

[1] Lamia Abo Zaid, Frederic Kleinermann, and Olga De Troyer. Feature assembly: A new feature modeling technique. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson Woo, and Yair Wand, editors, *Conceptual Modeling âĂŞ ER 2010*, volume 6412 of *Lecture Notes in Computer Science*, pages 233–246. Springer Berlin Heidelberg, 2010.

[2] Roberto Acerbis, Aldo Bongio, Marco Brambilla, and Stefano Butti. Webratio 5: An eclipse-based case tool for engineering web applications. In Luciano Baresi, Piero Fraternali, and Geert-Jan Houben, editors, *Web Engineering*, volume 4607 of *Lecture Notes in Computer Science*, pages 501–505. Springer Berlin Heidelberg, 2007.

[3] Aaron Brazell. *WordPress Bible*. Wiley Publishing, 1st edition, 2010.

[4] Vannevar Bush and Jingtao Wang. As we may think. *Atlantic Monthly*, 176:101–108, 1945.

[5] Sven Casteleyn and Olga De Troyer. Structuring web sites using audience class hierarchies. In Hiroshi Arisawa, Yahiko Kambayashi, Vijay Kumar, HeinrichC. Mayr, and Ingrid Hunt, editors, *Conceptual Modeling for New Information Systems Technologies*, volume 2465 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin Heidelberg, 2002.

[6] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web modeling language (WebML): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137 – 157, 2000.

[7] Olga De Troyer, Sven Casteleyn, and Peter Plessers. Using ORM to model web systems. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, volume 3762 of *Lecture Notes in Computer Science*, pages 700–709. Springer Berlin Heidelberg, 2005.

[8] Olga De Troyer, Sven Casteleyn, and Peter Plessers. WSDM: Web semantics design method. In Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, editors, *Web Engineering: Modelling and Implementing Web Applications*, Human-Computer Interaction Series, pages 303–351. Springer London, 2008.

[9] Mouratidou Despoina. WSDM-lite: A lightweight web design methodology for web shops. Master's thesis, Vrije Universiteit Brussel, 2014.

[10] Lucio Davide Spano Fabio Paterno, Carmen Santoro. Concur task trees (ctt). `http://www.w3.org/2012/02/ctt/#ctt`. Accessed: 2014-08-12.

[11] A Ginige and San Murugesan. Web engineering: an introduction. *MultiMedia, IEEE*, 8(1):14–18, Jan 2001.

[12] Object Management Group. Interaction flow modeling language (ifml). Technical report, March 2013.

[13] Terry Halpin. ORM/NIAM object-role modeling. In Peter Bernus, Kai Mertins, and GÃijnter Schmidt, editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems, pages 81–101. Springer Berlin Heidelberg, 1998.

[14] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2008.

[15] Hubert F. Hofmann and Franz Lehner. Requirements engineering as a success factor in software projects. *IEEE Softw.*, 18(4):58–66, July 2001.

[16] M. Hondo, N. Nagaratnam, and A. Nadalin. Securing web services. *IBM Syst. J.*, 41(2):228–241, April 2002.

[17] Le Van Huyen. Design patterns for the web semantics design method. Master's thesis, Vrije Universiteit Brussel, May 2010.

[18] Seyed Pejman Sajjadi Karmaseh. Adapting the website design method WSDM towards recent common practices in web development. Master's thesis, Vrije Universiteit Brussel, June 2012.

[19] John Krogstie, T. A. Halpin, and Keng Siau. *Information Modeling Methods and Methodologies*. IGI Global, Hershey, PA, USA, 2004.

[20] San Murugesan, Yogesh Deshpande, Steve Hansen, and Athula Ginige. Web engineering: a new discipline for development of web-based systems. In San Murugesan and Yogesh Deshpande, editors, *Web Engineering*, volume 2016 of *Lecture Notes in Computer Science*, pages 3–13. Springer Berlin Heidelberg, 2001.

[21] Oscar Pastor, Joan Fons, Vicente Pelechano, and Silvia Abrahao. Conceptual modelling of web applications: The OOWS approach. In Emilia Mendes and Nile Mosley, editors, *Web Engineering*, pages 277–302. Springer Berlin Heidelberg, 2006.

[22] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, INTERACT '97, pages 362–369, London, UK, UK, 1997. Chapman & Hall, Ltd.

[23] Chaman Thapa, Osmar Zaiane, Davood Rafiei, and AryaM. Sharma. Classifying websites into non-topical categories. In Alfredo Cuzzocrea and Umeshwar Dayal, editors, *Data Warehousing and Knowledge Discovery*, volume 7448 of *Lecture Notes in Computer Science*, pages 364–377. Springer Berlin Heidelberg, 2012.

[24] Todd Tomlinson. *Beginning Drupal 7*. Apress, Berkely, CA, USA, 1st edition, 2010.

[25] O.M.F. De Troyer and C.J. Leune. WSDM: a user centered design method for web sites. *Computer Networks and {ISDN} Systems*, 30(1-7):85 – 94, 1998. Proceedings of the Seventh International World Wide Web Conference.

[26] D.K. Van Duyne, J.A. Landay, and J.I. Hong. *The Design of Sites: Patterns for Creating Winning Web Sites*. Prentice Hall, 2007.

[27] Toby Velte, Anthony Velte, and Robert Elsenpeter. *Cloud Computing, A Practical Approach*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2010.

[28] Inge Van De Weerd. WEM: A design method for cms-based web implementations. Technical report, institute of information and computing sciences, utrecht university, 2005.

[29] Kevin Van Wilder. Implementation generation for WSDM using web applications framework. Master's thesis, Vrije Universiteit Brussel, 2009.

[30] Lamia Abo Zaid, Frederic Kleinermann, and Olga De Troyer. Feature assembly framework: Towards scalable and reusable feature models. In *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '11, pages 1–9, New York, NY, USA, 2011. ACM.