On Generating Content and Structural Annotated Websites using Conceptual Modeling

Sven Casteleyn[1], Peter Plessers[1], Olga De Troyer[1]

[1] Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene – Brussel
Belgium
{Sven.Casteleyn, Peter.Plessers, Olga.DeTroyer}@vub.ac.be
http://wise.vub.ac.be/

Abstract. An important milestone in the evolution of the Web is the Semantic Web: a Web in which the semantics of the available content and functionality is made explicit. Web design methods, originally aimed at offering a well-structured, systematic approach to Web design, now face new opportunities and challenges: Semantic Web technology can be used to make the semantics of the conceptual design models explicit; however a major challenge is to (semi-) automatically generate the semantic annotations, effectively enabling the Semantic Web. In this paper, we describe how WSDM, a well-known Web design method, was adapted to use Semantic Web technology for its conceptual modeling and how this can be exploited to generate semantically annotated websites. We consider two types of semantic annotations: content-related annotations and structural annotations. The first type allows to describe the semantics of the content of the website, the latter are annotations that explicitly describe the semantics of the different structural elements used in the website.

1 Introduction

Websites have evolved from a handful of statically linked pages into complex applications, serving a vast amount of rapidly changing information and functionality to a highly diversified audience. Web design methods were conceived to help the Web designer in coping with the complexity of designing and creating websites. Current Web design methods offer conceptual modeling primitives for different design concerns (i.e. most methods distinguish between data, navigation and presentation) combined with a systematic development approach.

The latest developments in the field of the Web are related to the vision of the *Semantic Web*. To allow making the semantics of the available Web content explicit, several (Semantic Web) technologies were introduced (e.g., RDF, OWL). With the arrival of the Semantic Web and its related technologies, new opportunities and challenges for Web design methods arose. A first opportunity lies in the use of Semantic Web technologies internally in the Web design method. More in particular, the use of ontologies allows to explicitly express the semantics of the different design models (meta-models), as well as the semantics of the represented data. In addition, the use of

Semantic Web technology in combination with semantically rich conceptual modeling concepts allows the generation of semantically annotated websites: websites in which the semantics of the content is made explicit by means of annotations. We call this kind of annotations *content-related annotations* to distinguish them from a second type of annotations, the so-called *structural annotations*. Indeed, it is also possible to annotate a website so that not only the semantics of its content are made explicit, but also the semantics of its structure. Dedicated ontologies describing the semantics of structural elements for a particular use (e.g., the WAfA ontology [19] is dedicated to assist visually impaired users while browsing) can be used to make the semantics of the different structural elements (e.g., a navigation menu, a logo, an advertising banner) explicit. These structural annotations can be generated by exploiting the conceptual design information captured during the design process. The more semantically rich design modeling concepts are used, the more of these semantically rich structural annotations can be generated. These structural annotations can subsequently be exploited by external applications requiring specific knowledge on the website structure: e.g., page transcoders (to transcode a webpage in a form more appropriate for screen readers used by visually impaired users) or search engine indexers.

In this paper, we explain how WSDM [4], an existing Web design method, combines Semantic Web technology and Conceptual modeling to allow the development of websites that satisfy the needs of the Semantic Web (section 2). We discuss how the adoption of Semantic Web technology is exploited to (semi-) automatically generate content-related semantic annotations (section 3), and to fully-automatically generate structural annotations (section 4). We also illustrate the benefits of structural annotations with two useful applications: facilitate accessibility for visually impaired users, and provide aid for search engines indexing websites. As the annotation process is performed on a conceptual level and the actual annotations are generated, the approach provides benefits over existing (manual) annotation approaches: (1) annotation is automatic (for structural annotations) or semi-automatic (for content-related annotations), (2) static as well as dynamic websites are supported, (3) changes in site structure or presentation do not invalidate the annotations (in contrast to manual annotation approaches) and (4) the generated annotations are more consistent. We discuss the actual implementation of the annotation generation process in section 5. Section 6 discusses related work, and finally, section 7 gives conclusions.

2 WSDM Overview and its Ontology

WSDM (Web Semantics Design Method), developed in 1998 [4], aimed to offer a systematic, multi-phase approach to Web design. It makes a clear distinction between the conceptual design and the implementation aspects. Each design phase focuses on one specific aspect: requirements specification, task modeling, content and functionality modeling, navigational design, presentation modeling and implementation.

With the emergence of the Semantic Web, WSDM has been adapted to support the development of *semantic* websites, i.e. the method supports the semantic annotation of content and structure. To achieve this, (1) an (OWL) ontology is used to formally

define the different WSDM design models, and (2) OWL is used as conceptual modeling language for the content and the functionality. The OWL ontology, which formally defines the different design models used in WSDM, is called the WSDM Ontology. The WSDM ontology can be compared to a set of meta-models. When using the method, the WSDM ontology is populated and will contain the design models created by the designer for the website under development.

In the remainder of this section, an overview of WSDM is given (see figure 1), and the different models are formally described using Description Logic syntax[1] [1][2].

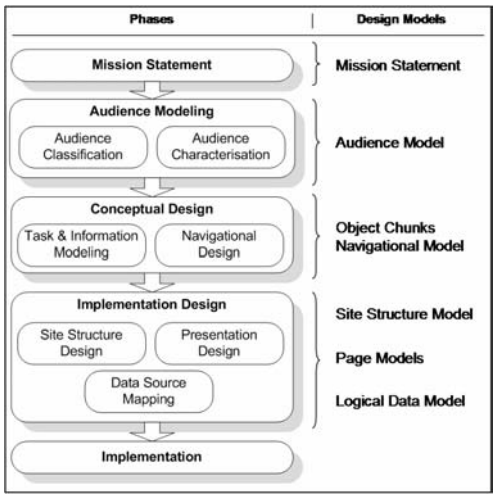

Fig. 1. WSDM Overview

Mission Statement Specification: In this first phase the *mission statement* of the website is formulated. The intention is to identify the purpose of the website, the topics and the target users. The mission statement is formulated in natural language. The WSDM Ontology fragment describing the mission statement is as follows: $\{MissionStatement \sqsubseteq (= 1\ hasValue),\ \top \sqsubseteq \forall hasValue.String\}$. Note that for the remainder of this section we will omit specification of datatype properties for reasons of clarity.

Audience Modeling: In this phase, the targeted users identified in the mission statement, are classified into so called *audience classes*. An audience class is a group of visitors that has the same information and functional requirements. An audience class that has the same and more requirements than another audience class is defined as an audience subclass. This results in an audience class hierarchy. For each audience class, the characteristics of the members of the class and their usability requirements are formulated. The output of this phase is *the audience model* consisting of the audience class hierarchy, and the characteristics and requirements of each audience class. The WSDM Ontology fragment describing the relevant concepts for the Audience

[1] Description Logic is the formal underlying framework for OWL(-DL)

[2]The full specification of the WSDM Ontology can be found at http://wise.vub.ac.be/ontologies/WSDMOntology.owl.

Modeling phase look as follows: {*UsabilityRequirement* ⊑ *Requirement*, *InformationRequirement* ⊑ *Requirement, FunctionalRequirement* ⊑ *Requirement,* ∃*hasAudienceSubclass*.⊤ ⊑ *AudienceClass,* ⊤ ⊑ ∀*hasAudienceSubclass.AudienceClass*, ∃*hasRequirement.*⊤ ⊑ *AudienceClass,* ⊤ ⊑ ∀*hasRequirement.Requirement*, ∃*hasCharacteristic.*⊤ ⊑ *AudienceClass,* ⊤ ⊑ ∀*hasCharacteristic.Characteristic*}.

Conceptual Design: In this phase, conceptual models are made starting from the requirements formulated in the previous phase. The designer creates conceptual models for the content, functionality and structure of the website. The conceptual design makes an abstraction from any implementation detail or target platform. The content and functionality are modeled during the *Task & Information Modeling* sub phase; the navigational structure is defined during the *Navigational Design* sub phase.

Information and functionality modeling is based on the requirements identified during Audience Modeling. Tasks are defined for the different requirements. These tasks are analyzed and modeled in detail using a slightly modified version of CTT (Concurrent Task Trees) [5]. Tasks are decomposed (step by step) into a set of elementary subtasks, and temporal relations among them are indicated. The result is a *task model*. For each elementary task, an *object chunk* is created to formally describe the information and functionality needed to perform this task [5]. OWL is used as conceptual modeling language for the object chunks. The output of the Task & Information Modeling phase is a set of *object chunks*.

The relevant part of the WSDM Ontology describing object chunks is given next: {∃*isComposedOf.*⊤ ⊑ *ObjectChunk*, ⊤ ⊑ ∀ *isComposedOf.*(*Class* ⊔ *DatatypeProperty* ⊔ *ObjectProperty*)}.

The goal of the Navigational Design is to define the conceptual structure of the website and to model how the members of the different audience classes can navigate through the website and perform their tasks (from a conceptual point of view). For each audience class, a dedicated navigation structure, called *navigation track*, is defined. A navigation track can be considered as a sub site containing all and only the information and functionality needed by the members of the associated audience class. Such a navigation track is composed of *nodes* (conceptual units of navigation) and *links* (which connect nodes). Links may be parameterized. Note that during the conceptual navigation design, no actual page structure is yet created. This is done during implementation design (see next). The output of this phase is the *navigational model*.

The WSDM Ontology fragment describing the relevant Navigation Design concepts is as follows: {∃*hasChunk.*⊤ ⊑ *Node,* ⊤ ⊑ ∀*hasChunk.ObjectChunk*, ∃*hasSource.*⊤ ⊑ *Link,* ⊤ ⊑ ∀ *hasSource.Node*, ∃*hasTarget.*⊤ ⊑ *Link,* ⊤ ⊑ ∀*hasTarget.Node*, ∃*hasCondition.*⊤ ⊑ *Link,* ⊤ ⊑ ∀*hasCondition.Condition*, ∃*hasParameter.*⊤ ⊑ *Link,* ⊤ ⊑ ∀*hasParameter.Parameter*}.

Implementation Design: Here, the conceptual design models are complemented with information required for the actual implementation: the distribution of nodes and links on pages (Site Structure Design), presentation issues (Presentation Design) and logical data source (Logical Data Design).

During **Site Structure Design**, the conceptual navigation structure of the website is mapped onto pages, i.e. it is decided which nodes (with associated object chunks) and links defined in the navigational model will be grouped onto Web pages. Different site structures can be defined, targeting different devices, contexts or platforms.

The output of this phase is the *site structure model*. The WSDM Ontology fragment describing the relevant Site Structure Design concepts is as follows: $\{\exists hasNode.\top \sqsubseteq Page,\ \top \sqsubseteq \forall hasNode.Node$, Page $\sqsubseteq \exists hasNode.Node\}$.

The goal of the **Presentation Design** is to describe the layout of the pages, i.e., positioning and style. First, page templates are designed. Different kinds of templates may be needed, e.g., a homepage template, a title page template, leaf page templates. WSDM provides several Template Concepts (e.g., 'Footer', 'Header', 'Sidebar') to model page templates. For styles, Cascading Style Sheets are currently used. Next, it is specified how the information and functionality (modeled by means of the object chunks and grouped by means of nodes and assigned to a page) should be presented. Therefore, WSDM offers several Presentation Concepts to model the layout and presentation of a page. These Presentation Concepts vary from primitive ones (e.g., 'Grid', 'Row', 'MultimediaConcept', 'FormConcept') to high-level concepts (e.g., 'Menu', 'Section'). Also during Page Design, the designer must decide on labels and presentation styles for links. The output of this phase is the *presentation model* consisting of a set of templates, and for each page defined in the site structure model a *page model*.

The **Logical Data Design** is needed for data-intensive websites that maintain their data in a data source. In this phase, this data source must be defined and the relationship between the conceptual level (i.e. the object chunks) and the data source must be expressed. This last issue is explained into more detail in the next section.

3 Content-Related Semantic Annotations

In this section, we describe how WSDM allows designing websites of which the content is semantically annotated. Important to our approach is that this is supported at a conceptual level. The approach extends and refines our previous work as described in [15]: multiple existing domain ontology can be used, if needed, an appropriate (application) ontology can be extracted from the design, but most importantly, the use of OWL facilitates easier specification of semantic annotations.

Conceptual Design

The goal of our approach is to generate a website of which the content is automatically annotated with one or more domain ontologies which are related with the topics covered by the website. Details about the actual generation process are given in section 5. Here, we describe the principles of the approach and what must be done by the designer to obtain a semantically annotated website. In practice, three different cases may occur when designing a website:

1. *No appropriate domain ontology exists or is available.* A new ontology will be created incrementally as a result of the creation of the object chunks, i.e. by integrating all object chunks (see [6]). The object chunks are expressed as views on

this ontology. Note that there is no additional effort required from the designer. Such an ontology is often called an *application ontology*.

2. *A single domain ontology exists that covers completely the domain of the website.* In this case, this ontology is taken as the basis for the conceptual design. The designer needs to express the concepts and relations used in the object chunks in terms of concepts from this domain ontology, e.g., by referring to an ontology concept instead of defining a new one. In this way, the object chunks are defined as views on this domain ontology.
3. *Multiple domain ontologies are needed to cover the domain of the website.* In this case, the different domain ontologies must be aligned first. This is done by defining a so-called *reference ontology* and by defining mappings between the domain ontologies and this reference ontology. Then, the concepts used in the object chunks can be defined in term of the concepts of this reference ontology, and the object chunks will be views on the reference ontology.

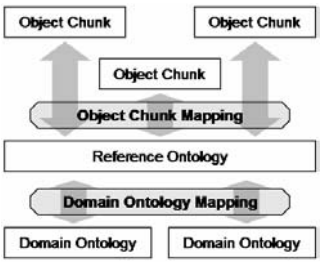

Fig. 2. General architecture illustrating the different mappings.

Figure 2 shows an overview of the architecture covering these three cases. The different domain ontologies used are aligned by defining a mapping between the domain ontologies and the reference ontology (called *domain ontology mappings*). This reference ontology can also be used to define additional concepts not present in the available domain ontologies but relevant for the application. Note that in the case of just one domain ontology, the references ontology plays the role of this domain ontology (possibly also augmented with additional concepts). In the case where there is no domain ontology available, the reference ontology plays the role of application ontology that is incrementally constructed. The second type of mappings, called *object chunk mappings*, defines the object chunks as views on the reference ontology. A view mechanism is required because the conceptualization as specified by a domain ontology may not always exactly suit the requirements of the website. E.g., a domain ontology may specify an address as composed of a street, number and city, but the website may prefer to consider the address as a single entity (i.e. a single string).

To illustrate the different mappings, we give a small example. As in section 2, we use Description Logic syntax, this time to describe object chunks, reference ontology and domain ontologies. Suppose, a first (existing) domain ontology contains (besides other axioms) the following axioms: {*Man* $\sqsubseteq$ *Person*, *Woman* $\sqsubseteq$ *Person,* $\top \sqsubseteq \forall$*hasMaternityLeave.*{*true, false*}*,* $\exists$*hasMatenityLeave.*$\top \sqsubseteq$ *Woman*, *Woman* $\sqsubseteq$ (= 1 *hasMaternityLeave*)} (Informally: 'Man' and 'Woman' are subtypes of 'Person', and for a 'Woman' it is specified if she is on maternity leave or not). A second (existing) domain ontology describes a partly overlapping domain, and contains the following

axioms: {$\top \sqsubseteq \forall hasSex.\{M, F\}$, $\exists hasSex.\top \sqsubseteq Person$, $\exists hasStreet.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasStreet.String$, $\exists hasCity.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasCity.String$, $\exists hasCountry.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasPostalCountry.String$} (a 'Person' is either male or female, specified by the 'hasSex' property, and a 'Person' has an address which is specified by the 'hasStreet', 'hasCity' and 'hasCountry' properties). To align these two domain ontologies, it is necessary to resolve the different ways of representing a person's sex (i.e. respectively by using subtypes, and by using a hasSex property) and furthermore to merge the non-overlapping parts of both ontologies. Suppose this is done by constructing the following reference ontology:
{$Man \sqsubseteq Person$, $Woman \sqsubseteq Person$, $\top \sqsubseteq \forall hasMaternityLeave.\{true, false\}$, $\exists hasMaternityLeave.\top \sqsubseteq Woman$, $\exists hasStreet.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasStreet.String$, $\exists hasCity.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasCity.String$, $\exists hasCountry.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasCountry.String$}

Then, the following domain ontology mappings express the relations between the reference ontology and the two domain ontologies (trivial mappings are omitted):

Reference ontology	**Domain Ontology1**	**Domain Ontology2**
Man	Man	Person WHERE hasSex = 'M'
Woman	Woman	Person WHERE hasSex = 'F'
hasMaternityLeave	hasMaternityLeave	-

Now assume that the Web designer wants to consider an 'address' as a single string. This is expressed in the object chunk as follows: {$Man \sqsubseteq Person$, $Woman \sqsubseteq Person$, $\exists hasAddress.\top \sqsubseteq Person$, $\top \sqsubseteq \forall hasAddress.String$} ('Man' and 'Woman' are subtypes of 'Person', and a 'Person' has an address specified as a single string). Now, 'hasAddress' cannot refer in a one-to-one way to a concept in the reference ontology. Instead, the following Object Chunk Mapping is needed (trivial one-to-one mappings are again omitted). The '+'-sign indicates the concatenation of strings.

Object Chunk	**Reference ontology**
hasAddress	hasStreet + hasCity + hasCountry

Data Source Mapping

When the website is generated (from the models), the actual pages need to be filled with data. The designer may decide to use a data source (e.g., a relational database) to maintain the data. To be able to generate the actual pages a mapping is needed between the conceptual level (i.e. the object chunks) and this data source. The mapping is defined between the reference ontology and the data source. E.g., in the case of a relational database, the data source mapping indicates the tables and columns where instances of concepts of the reference ontology can be found. Note that, similar as for object chunk mappings and domain ontology mappings, no one-to-one mapping can be assumed. For example, for a relational table 'Person(ID, street, city, country, gender, hasMaternityLeave), we have the following data source mappings:

Reference ontology	**Data Source**
hasMaternityLeave	SELECT hasMaternityLeave FROM Person WHERE gender='F'

Woman	SELECT ID FROM Person WHERE gender='F'
Man	SELECT ID FROM Person WHERE gender='M'

The different mappings will be used to generate the actual annotations (see section 5). This approach is different from the usual annotation approaches that define mappings with the ontology directly on the implementation level (see also section 6 on related work). In our approach, the mappings are defined at the conceptual level. This has several advantages. We mention the most important ones:

1. *Implementation independent*: the basis for the annotations is made on the conceptual level, and therefore the actual website annotations can be generated along with different implementations.
2. *Consistency of annotations*: as concepts (in the object chunks) are linked to Reference Ontology concepts and only one link per concept is given, it is not possible (like in other annotation approaches) that the actual annotations (for different instances) are not consistent.
3. *Both static and dynamic websites supported*: the implementation generation process of WSDM (see section 5) does not distinguish between static and dynamic websites; annotations are effortlessly generated for both types of websites.

4 Structural Semantic Annotations

By exploiting the semantics of the modelling concepts (e.g., menu, header, node) used in the different design models (and captured in the WSDM Ontology), useful annotations concerning the structure of the website can be generated. This is realized by defining a mapping between the WSDM Ontology concepts and an external ontology describing the semantics of structural elements (tailored for a certain use). An example of such an ontology is the WAfA ontology [19]. Subsequently, these mappings can be used to annotate the actual website with concepts from this external ontology. As the mappings are dependent on the ontology used, we will illustrate the approach for two different ontologies: the WAfA ontology (developed to assist visually impaired users) and a (newly created) block-ontology (to assist search engines in more accurately indexing a website). Evidently, it is possible to annotate one website using multiple ontologies, each describing different types of structuring elements.

Structural Annotations to support Accessibility for Visually Impaired User

Currently, most visually impaired users rely on screen readers to access websites. These screen readers sequentially read a page. This is not only time-consuming for the user but in addition a lot of information that is conveyed by means of layout (e.g., white space, tables used for structuring) is lost. The Dante approach [19] allows annotating Web pages using the WAfA ontology, which defines concepts that allow indicating how objects on a page are presented and the role they fulfil in the presentation. These annotations allow (external applications) to transcode Web pages in a form more suitable for accessing pages using screen readers. However, currently it is a manual annotation process, and this is an effort that is too labour intensive to be

usable in general. Moreover, the resulting annotations are typically sensitive to changes in the websites content or structure (and re-annotation is required).

By defining a mapping between the modelling concepts in the WSDM ontology and the concepts in the WAfA ontology, the WAfA annotations can be generated automatically when developing a website using WSDM. Here we give the mapping for two representative concepts. To describe the mapping rules, we use the following notational convention: first, the WAfA concept is given in bold, followed by it's meaning (in italic). Where needed an informal explanation of the mapping rule is given and finally a formal definition using Semantic Web Rule Language (SWRL)[3] which is particularly suited to handle OWL specifications and makes automatic annotation generation possible (see section 5). Some mapping rules between the WSDM and WAfA-ontology are straightforward one-to-one mappings; others are more complex and need to exploit the knowledge captured by means of several concepts, and/or the relationships between them. Two examples follow:

- **WAfA:TableOfContent**: *A list of available sections and a link to the beginning of each section*

  `wsdm:NavigationTableOfContent(?i) ⇒ WAfA:TableOfContent(?i)`[4]

- **WAfA:DropDownLinkMenu:** *A DropDownLinkMenu is a menu that appears below an item when the user clicks on it.* A linkmenu corresponds to a wsdm:Menu represented as a wsdm:List in WSDM. Furthermore, to denote it is a *dropdown* menu, it should have an associated wsdm:Behaviour defined with wsdm:Event 'onClick' and wsdm:Action 'dropDown'. Each menu with this behaviour is a DropDownLinkMenu in WSDM.

  ```
  wsdm:Menu(?i) ∧ wsdm:representedBy(?i, ?x) ∧ wsdm:List(?x)  ∧
  wsdm:hasBehavior(?x, ?y) ∧ wsdm:Behavior(?y) ∧ wsdm:onEvent(?y,
  'on Click') ∧ wsdm:doAction(?y, 'dropDown')
  ⇒      wafa:DropDownLinkMenu(?i)
  ```

Other mapping rules are defined in a similar manner. Currently, we have defined mapping rules for 74% of the WAfA Ontology concepts (see [16]). Note that this mapping is a once-only activity. Thereafter, it can be used to automatically generate structural annotations for any website[5].

Structural Annotations for Search Engine Support

To improve search results, search engines apply a technique called page segmentation (see e.g. [3] for an overview). The aim of page segmentation is to distinguish meaningful "blocks" (also called "passage") in a Web page according to the logical structure, the presentation and the semantics of page objects. This information is subsequently exploited in page-rank and website indexing algorithms (e.g. [3, 10]). Extensive research has been done in devising information retrieval algorithms that are able to extract the relevant blocks from a given Web page. Unfortunately, as valuable design knowledge about the structure and semantics of page objects is not available in typical Web pages, output of these algorithms is unavoidably limited. Similar as in

[3] See http://www.daml.org/2003/11/swrl/.

[4] Both ontologies were developed independently, which explains different names for similar concepts.

[5] Compare to manual annotation approaches, where each website needs be processed by hand.

the previous case, semantics concerning structure available in the WSDM design models can be used to automatically generate semantic annotations describing the "blocks" required to sophisticate search engine's indexing algorithms. As no ontology describing these blocks and their relationships exists, we have created a prove-of-concept block-ontology describing different semantic blocks (e.g., topics, sections, units) and their relationships (both semantic, e.g., 'isSubTopicOf', and spatial, e.g., 'below'). Note that it would be possible to directly use the WSDM Ontology to make the annotations. However, this would require knowledge of the WSDM Ontology by the page segmentation algorithms. Two example mapping rules are:

- **block:Section:** *A block representing a section in a Web page*
  `wsdm:Section(?i) ⇒ block:Section(?i)`
- **block:SemanticBlock:** *A block representing a semantic unit (presented together).* In WSDM, an object chunk represents (a unit of) information needed for a single task. The wsdm:Grid representing a wsdm:ObjectChunk can be annotated as a block:SemanticBlock:
  `wsdm:Grid(?i) ∧ wsdm:representsChunk(?i, ?x) ∧`
  `wsdm:ObjectChunk(?x) ⇒ block:SemanticBlock(?i)`

5 Implementation Generation Process

To generate the actual semantically annotated website, a transformation pipeline is used. We will not explain the complete pipeline but instead focus on the generation of the annotations. The pipeline takes all the models of the conceptual and the implementation design as inputs. The transformations to generate the implementation of the website (without annotations) consists of four steps (T1, T2, T3 and T4 in Figure 3):

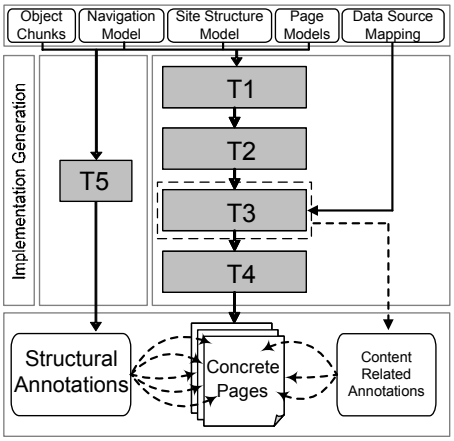

Figure 3 Implementation Generation Overview

- *Model Integration (T1):* integrates the different input models into one single model. In principle, this transformation can be omitted, but it simplifies the following transformations.

- *Implementation Mapping (T2):* the implementation platform is chosen (e.g., HTML, XHTML, WML), and the integrated model derived in T1 is transformed towards the chosen platform. References to data (i.e., to instances in object chunks) are not yet processed; this is done in the next transformation T3.
- *Query Construction (T3):* the references to instance values in the object chunks are resolved and mapped onto queries on the data source. This is performed fully automatically because the mappings from the object chunks to the reference ontology and from the reference ontology to the data source are available (see section 5.1 for an example).
- *Query Execution (T4):* finally, the queries derived in T3 are processed, and the actual pages are generated by inserting the data at the proper places. When the query execution phase is performed offline, a static website is created; when it is performed at runtime, a dynamic website is the result.

Generating Content-Related Annotations

The content of the website is annotated by means of the OWL reference ontology (see section 3). Remember that the object chunks are defined as views (or conceptual queries) on the reference ontology. In the query construction (T3), these conceptual queries are transformed into executable queries using both the object chunk- and data source mappings. We explain by means of a small example (based on the example of section 3) how these mappings allow us to generate content-related annotations.

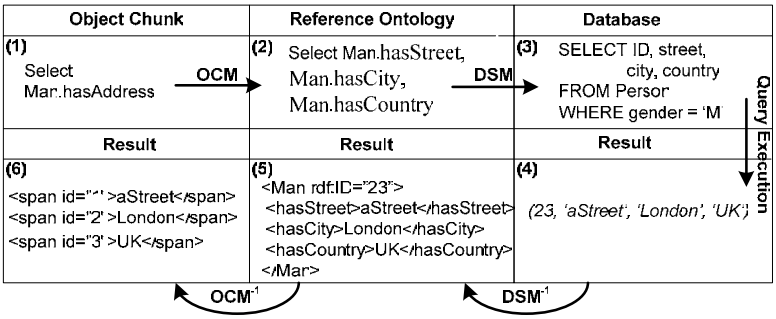

Figure 4 Generating Content-Related Semantic Annotations

An overview of the content-related annotation generation process is illustrated in figure 4. Consider a conceptual query expressing the address of men (1). Using the object chunk mapping (OCM), this conceptual query on the original object chunk is transformed into a conceptual query on the reference ontology (2). Using the data source mapping (DSM), the resulting conceptual query is transformed into an executable (SQL) query (3) on the actual data source (here a relational database). The result of this query is a set of instances, in the form of a table, of which an example tuple is shown in (4). The data in this table is subsequently transformed into a set of instances of the reference ontology using the inverse data source mapping (DSM^{-1}) (5). Finally, the address is presented as a single string (as was specified by the object chunk), using the inverse object chunk mapping (OCM^{-1}) (6). Note that, by inserting -tags surrounding the individual attributes, we are still able to refer to the individual parts of the address string on the Web page, i.e. no semantic information present in

the reference ontology is lost. Finally, we link the generated HTML code (given in (6)) and the instantiation of reference ontology concepts together using XPointer expressions: page.html#xpointer(id("1"))<=>refOnt#xpointer(id("23")/hasStreet)

Generating Structural Annotations

Taking as input the design models in the WSDM Ontology and the mapping between the WSDM Ontology and an (external) ontology describing structural elements (e.g., the WAfA ontology), a transformation *T5* can be added to the transformation pipeline to generate the structural annotations. To illustrate the generation of structural annotations, consider the example in which a (WSDM) menu with menu-items is transformed to a bulleted list in HTML, including structural annotation denoting the presence of a menu for accessibility purposes (using the WAfA ontology):

(1) Result after T1

```
<wsdm:Menu rdf:id="menu1">
 <wsdm:hasItem>
  <wsdm:MenuItem rdf:id="item1">
   <wsdm:Label>item 1</wsdm:Label>
   <wsdm:hasNavRef
       rdf:resource="#ref1"/>
  </wsdm:MenuItem> </wsdm:hasItem>
 <wsdm:hasItem>
  <wsdm:MenuItem id="item2">
   ...
  </wsdm:MenuItem> </wsdm:hasItem>
 <wsdm:representedBy
    rdf:resource="#bulletedList"/>
</wsdm:Menu>
```

(2) Result after T4

```
<ul id="menu1">
 <li id="item1">
   <a href="a link">item 1</a>
 </li>
 <li id="item2">...</li>
</ul>
```

(3) Result after T5

```
http://.../wafa.owl#linkMenu
http://www.example.com/page.html
#xpointer(id("menu1"))
```

Note how the unique ids, originating from the WSDMOntology instances, are maintained through the transformation pipeline and reflected in the final code. In our example, the bulleted list in **(2)** carries the same id as the high-level presentation concept wsdm:Menu in **(1)**, denoting that the (bulleted) list structure actually represents a menu, and it is annotated with a WAfA:linkMenu concept.

A prototype implementation of the transformation pipeline was made using Semantic Web technology: OWL for the WSDM Ontology and (instantiations of the) design models and object chunks, XSLT to perform the transformation steps, and xPointer to link annotations and actual implementation (i.e. HTML).

6 Related Work

When reviewing the literature concerning semantic annotations, we can mainly distinguish three different approaches: manual, (semi-)automatic, and Web engineering approaches. The difference between manual and automatic approaches consists of the fact that the former ones require a (manual) mapping between content and semantics, while the latter attempt to extract the semantics automatically (e.g., using NLP techniques). Examples of automatic approaches include Melita [2] and KMI annotation framework [13]. Manual annotation approaches offer the user tool support to define annotations for HTML documents. The first tool in this context was the SHOE

Knowledge Annotator [9], which only supports static Web pages. In course of time, other manual annotation tools arose: SMORE [18] (adding authoring support by using an embedded HTML editor), Ont-O-Mat [8] (adding support for dynamic Web pages by annotating database implementations).

Both manual and automatic approaches suffer some disadvantages. The adequacy of automatically generated annotations is generally lower compared to manual approaches; the disadvantage of manual approaches is that the annotations are defined on an implementation level (making them more vulnerable to changes) and require a substantial effort from the designer after the website is already implemented.

Recently, research has also been focused on integrating semantic Web technology into Web design methods. Examples of semantic Web design methods include SHDM [14], Hera [7], OntoWeaver [12], OntoWebber [11]. These methods use ontology languages (e.g., RDFS, OWL) as modeling language for their design models. This has the advantage that existing ontologies can be used in the design process and that a verification of the design models is feasible. Some of these approaches offer the possibility to make the data models internally constructed externally available (in the form of RDFS or OWL). However, none of these approaches actually generates websites that are annotated i.e., they rather offer the content (independently) in user- (e.g., HTML) and machine-readable form (e.g., RDF). Explicitly linking Web content with ontologies that describe the semantics (semantic annotations) is required to support for example content rating and filtering (see http://www.w3.org/TR/rdf-pics). These methods also do not provide support for structural annotations.

The only known approach similar to the one described in this paper, is WEESA [17]. However, WEESA is not a design method by itself, but can be used after the design and for design methods that specify their design models in XML. It is able to generate content-related semantic annotations by defining a mapping between the XML schemas and existing ontologies. The disadvantage of WEESA is that it cannot benefit from the Web design process itself, but instead needs to define the mapping regardless if a domain ontology was used during the design process or not. As far as we are aware of, no other Web design method generates structural annotations.

7 Conclusion

In this paper, we described how in the website design method WSDM Semantic Web technology (OWL) and conceptual modeling is used to generate two types of semantic annotations: content-related annotations and structural annotations. The use of ontologies for the conceptual modeling of information and functionality during the design process allows (semi-) automatically generation of content-related semantic annotations. Three different situations are considered 1) no existing domain ontology is available, 2) a single existing domain ontology can be used, and 3) multiple existing domain ontologies must be used. Next to content-related semantic annotations, we also discussed structural semantic annotations: annotations which (semantically) describe the structure of the website. This type of annotations is generated exploiting the semantics of the different design modeling concepts. The approach is illustrated for two types of structural annotations and their usefulness has been pointed out.

The integrating of the annotation generation in the design process of a website, as described here, has the following advantages over existing (post-website-deployment) annotation approaches: smaller effort required (i.e. content-related annotations are semi-automatically generated, structural annotations fully automatically), robustness (annotations are not invalidated when re-designing the website), higher consistency, and support for dynamic websites.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook (2003)
2. Ciravegna, F., Dingli, A., Petrelli, D., Wilks, Y: User-System Cooperation in Document Annotation based on Information Extraction. In Proc. of EKAW 02, Sigüenza Spain (2002)
3. Deng Cai, Shipeng Yu, Ji-Rong Wen, Wei-Ying Ma.: Block-based web search. ACM SIGIR. (2004), pp. 456-463
4. De Troyer, O. and Leune, C.: WSDM: A User-Centered Design Method for Web Sites. In Proceedings of the 7th WWW Conference, Elsevier, (1998), pp. 85-94
5. De Troyer, O., Casteleyn, S.: Modeling Complex Processes for Web Applications using WSDM. In Proceedings of the 3rd Int. IWWOST workshop (2003)
6. De Troyer, O., Plessers, P., Casteleyn, S.: Conceptual View Integration for Audience Driven Web Design. In CD-ROM Proc. of the WWW2003 Conference, Budapest Hungary (2003)
7. Frasincar, F., Houben, G.-J.: Hypermedia presentation adaptation on the semantic web. In Proceedings of AH 2002, LNCS, Springer (2002), pp. 133-142
8. Handschuh, S., Staab, S.: Authoring and annotation of web pages in CREAM. The 11th Int. World Wide Web Conference (WWW2002), Honolulu Hawaii USA (2002)
9. Heflin, J., Hendler, J.: Searching the web with SHOE. Artificial Intelligence for Web Search, Papers from the AAAI Workshop, WS-00-01, AAAI Press (2000), pp. 35-40
10. Jiang X.-M., Xue, G.-R., Song W.-G., Zeng, H.J., Chen, Z., Ma W.-Y.: Exploiting PageRank at Different Block Level. In Proceedings of the WISE 2004, (2004), pp. 241-252
11. Jin, Y., Xu, S., Decker, S., Wiederhold, G.: OntoWebber: A Novel Approach for Managing Data on the Web. In Proceedings of ICDE (2002), pp. 488-489
12. Lei, Y., Motta, E., Domingue, J.: Modelling Data-Intensive Web Sites with OntoWeaver. In proceedings of the International Workshop WISM2004, Riga Latvia (2004)
13. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic Annotation, Indexing, and Retrieval. Elsevier's Journal of Web Semantics, Vol. 2, Issue (1) (2005)
14. Moura, S., Schwabe, D.: Interface Development for Hypermedia Applications in the Semantic Web. In Proceedings of LA Web 2004, Ribeirão Preto, Brasil. IEEE CS Press (2004)
15. Plessers, P., De Troyer, O.: Annotation for the Semamtic Web during Website Development, In Proceedings of the ICWE 2004 Conference, Munich Germany (2004), pp. 349-353
16. Plessers, P. Casteleyn, S., Yesilada, Y., De Troyer, O., Stevens, R., Harper, S., Goble, C.: Accessibility: A Web Engineering Approach, In Proceedings of the 14th Int. World Wide Web Conference, Chiba Japan (2005), pp. 353-362
17. Reif, G., Gall, H., Jazayeri, M.: WEESA - Web Engineering for Semantic Web Applications. In Proceedings of the 14th Int. World Wide Web Conference, Chiba Japan (2005)
18. Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In Proc. of the 13th International Conference on Knowledge Engineering and Management (2002)
19. Yesilada, Y., Harper, S., Goble, G., Stevens, R. Screen Readers Cannot See. In ICWE 2004 Proceedings, (2004), pp 445-458