Semantic Web Development with WSDM

Peter Plessers, Sven Casteleyn, Olga De Troyer

Vrije Universiteit Brussel, Pleinlaan 2, 1000 Brussels, Belgium
{Peter.Plessers, Sven.Casteleyn, Olga.DeTroyer}@vub.ac.be

Abstract. Currently, one of the important focal points of the evolution of the World Wide 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 the designer a well-structured, systematic approach to web design, now face new opportunities and challenges: on one hand, semantic web technology can be used internally in web design methods to make the semantics of the different design models explicit, on the other hand, a major challenge for existing design methods is to (semi-) automatically generate the semantic annotations, effectively enabling the semantic web. In this paper, we describe how WSDM, one of the first web design methods, was adapted to (internally) use semantic web technology. In addition to its existing strengths (i.e. solid design support for website structure and presentation) as a traditional web design method, we show how the internal use of semantic web technology can be exploited to support the generation of semantically annotated websites.

1 Introduction

With over a decade of evolution, the World Wide Web has undergone some dramatic changes. While websites at first consisted of a handful of linked (static) pages, they are now complex applications, offering (rapidly changing) information and functionality to a highly diversified audience. To keep up with the growing needs and demands of visitors, web technology evolved at an equally dazzling rate. In this setting, it gets more and more difficult to design a website in an ad hoc way. The resulting websites lack consistency, both in structure and presentation, and transparency. Visitors fail to build a mental model of the website, causing them to feel 'lost in hyperspace'.

Web site design methods were conceived to help the web designer in coping with the complexity of designing and creating websites. By offering abstractions for the different design concerns, and a systematic approach to web design, these design methods succeeded in helping the designer to create more usable websites.

With the conception of the semantic web, and related technologies (e.g. RDF, OWL, …), new challenges and opportunities for web design methods arose. The aim of the semantic web is to make the semantics of the available web content explicit, thereby facilitating machine understanding and processing (of the content). Building on their strength in providing design support for website (navigation) structure and presentation, a challenge for web design methods is to support the (semi-) automatic

generation of semantic annotations. An opportunity however lies in the use of semantic web technologies internally in the web design method. More particularly, the use of ontologies, which capture semantics, allows to explicitly express the semantics of the different design models, as well as the semantics of the represented data.

Some web design methods use ontology languages to internally represent information (see related work). As far as the authors are aware of, only limited support exists for automatic generation of semantic annotations based on web design models. There exist however manual and semi-automatic (based on natural language parsing) annotation techniques (see related work). In this paper, we discuss how WSDM, one of the first web site design methods, was adapted to suit the needs of the semantic web on one hand, and to benefit from semantic web technology on the other hand. We further extend our previous work on semantic annotations as described in [16]. Major changes include the use of web ontology language OWL, both to explicitly define the different WSDM design models and to model data and functionality of the website. Subsequently, we show how the adoption of semantic web technology helps to (semi-) automatically generate semantic annotations. As the semantic annotations are supported on a conceptual level, and the actual annotations are generated based on this conceptual level, the approach provides some benefits over existing annotation approaches: static and dynamic websites are supported, changes in site structure or presentation do not invalidate the annotations and the generated annotations are more consistent.

The remainder of this paper is structured as follows. Section 2 gives an overview of the WSDM approach and informally discusses the WSDM Ontology capturing the different design models. Chapter 3 discusses how WSDM supports semantic annotations at a conceptual level (during design), and points out problems and solutions. Chapter 4 describes in more detail the actual generation process of both website and semantic annotations. Chapter 5 gives an overview of related work and finally section 6 states conclusions.

2 WSDM Overview and Ontology

WSDM was developed in 1998 by De Troyer and Leune [2] and aims to separate design concerns by offering a systematic, multi-phase approach to web design. Each design phase focuses on one specific aspect of the web design cycle: requirements and task analysis, data and functionality modeling, navigation modeling, presentation modeling and implementation. More than other web design methods, WSDM is a methodology, i.e. it not only provides modeling primitives that allow a web developer to model the web application from different perspectives and at different levels of abstraction, but it also provides a systematic way for the designer to obtain the different design models and the resulting website.

WSDM originally targeted 'traditional' websites. With the emergence of the semantic web, WSDM has been adjusted to support the generation of *semantic* websites: the web content is annotated with semantic information. To facilitate the specification of semantic information (during design), and the generation of semantic annotations (during implementation generation), an OWL ontology is used to formally

define the different design models, and to perform data and functionality modeling. This OWL ontology formally specifying the different WSDM design models is called the WSDM Ontology[1]. In the remainder of this section, an overview of WSDM is given, and the OWL concepts that describe the relevant design models are informally described.

Figure 1 shows an overview of the different phases of WSDM, and the relevant design models constructed in each design phase.

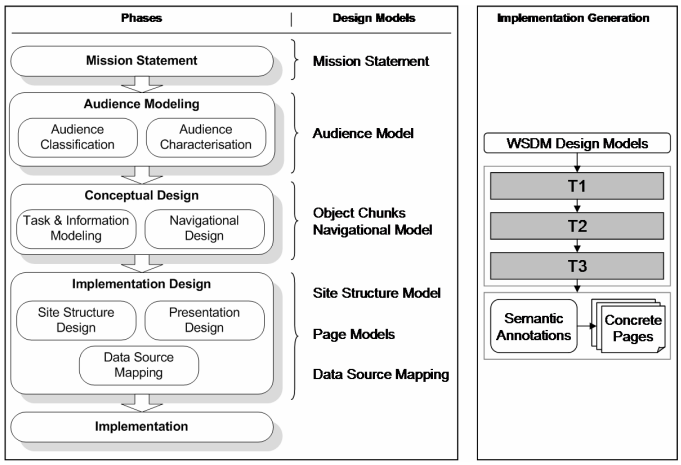

Fig. 1. WSDM Overview

Fig. 2. Implementation generation.

Mission Statement

In the first phase of WSDM, the mission statement for the website is formulated. The intention of this phase is to identify the subject of the website, the purpose and the target users. The mission statement is formulated in natural language. In the WSDM Ontology, the 'mission statement' is an OWL concept with a textual data property.

Audience Modeling

In the Audience Modeling phase, the targeted users, identified in the mission statement, are classified in so called *audience classes*. An audience class is a group of visitors that have the same information and functional requirements. Audience classes can be subclassed: an audience class that has the same and more requirements than another audience class is called an audience subclass. Also during audience modeling, for each audience class, their characteristics and usability requirements are expressed.

In the WSDM Ontology, OWL concepts for 'requirement' and 'characteristic' are described. Requirement is subclassed into 'Usability-', 'Information-' and 'Functional-' requirements. Finally, the 'Audience Class' concept describes the WSDM audience classes. Each 'Audience Class' has as object properties 'hasRequirement' and 'hasCharacteristic', and can be subclassed. The model representing the audience

[1] See http://wise.vub.ac.be/ontologies/WSDMOntology.owl

class hierarchy as well as for each audience class their characteristics and their set of (informal specified) requirements is called the audience model.

Conceptual Design

The conceptual design phase is used to specify the content, functionality and structure of the website at a conceptual level. The conceptual design makes an abstraction from any implementation or target platform. The content and functionality are defined during the Task Modeling phase; the navigational structure is defined during the Navigational Design.

The purpose of the Task Modeling phase is to analyze in detail the different tasks each audience class needs to be able to perform, and to formally describe the data/functionality that is needed for those tasks. The tasks each audience class needs to be able to perform are based on the requirements formulated for each audience class (during audience modeling). WSDM uses a slightly modified version of CTT [15] to decompose each (high level) task into a set of elementary subtasks, and describe the temporal relations among them. Such decomposition is called a task model. For each elementary task of a task model, an object chunk is created to formally describe the necessary information and functionality needed to fulfill the requirement of this elementary task [3]. OWL itself is (re-)used as modeling language for the object chunks. In the WSDM Ontology, the OWL concept 'Object Chunk' is composed of OWL classes, object and data properties. Furthermore, 'Object Chunks' can have associated 'Object Chunk Functions', which allow to model system functionality and interaction (e.g. instance creation, select functions, upload function, …). Due to space restrictions, we don't go into deeper detail on these functions.

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. 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 further composed of nodes (conceptual units of navigation) and links (connecting nodes).

In the WSDM Ontology, the Navigation Model is represented by the OWL concepts 'node' and 'link'. Nodes have one object property 'hasChunk' which identifies the object chunks which are connected to the node. There are two subtypes of nodes: 'RootNode' and 'ExternalNode'. Links have object properties 'hasSource' and 'hasTarget', identifying the two nodes that are linked. Furthermore, a link can have a parameter ('hasParameter') identifying flow of information along links, and a condition ('hasCondition') to restrain the appearance of the link. There are four subtypes of 'link', one for each link type supported in WSDM: navigation aid, process logic, semantic and structural link. For an in depth discussion of these link types, see [4].

Implementation Design

During the implementation design phase, the conceptual design models are completed with information required for the actual implementation. The implementation design consists of three sub phases: the Site Structure Design, Presentation Design and Data Source Mapping.

During Site Structure Design, the conceptual structure of the website (defined during navigational design) is mapped onto pages, i.e. it is decided which nodes (with 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.

In the WSDM ontology, 'pages' are OWL concepts which have one object property 'hasNode' to denote the nodes that are contained in that particular page.

The Presentation Design defines the look and feel of the website as well as the layout of the pages (i.e. positioning of page elements). First, the sub phase Style & Template Design, aims at designing page templates. Typically, a website may require different kinds of templates, e.g. a homepage template, a title-page template, leaf page templates, etc. Furthermore, the style of page elements (e.g. font, color, alignment, etc.) is also specified. Next, the sub phase Page Design aims at describing how the information/functionality (modeled by the object chunks and represented by means of nodes) assigned to a page should be presented. Also link labels are decided, and presentation styles are given to the (different) links. The layout of a page is based on one of the templates defined during the Style & Template Design. This is done for each page type. 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 main concepts in the WSDM Ontology describing the presentation design are 'TemplateConcepts' (concepts related to modeling templates) and 'PresentationConcepts' (concepts related to modeling positioning of page objects). Style is currently not included in the WSDM Ontology; instead Cascading Stylesheets are used. Going into further detail on the WSDM presentation concepts is outside the scope of this paper, and left for a forthcoming publication.

Finally, the Data Source Mapping phase is discussed in the next section (see Data Source Mapping).

3 Semantic Annotations

In this section, we describe how WSDM supports semantic annotations at the conceptual level and explain the benefits of our approach. The approach extends our previous work as described in [16]: next to building a domain ontology during design, we also show how one or more existing domain ontologies can be re-used to generate semantic annotations, and the problems (and solutions) involved.

Conceptual Design

As described in the overview of the WSDM method (see Section 2), Object Chunks are used to model the concepts and relations between these concepts necessary to fulfill a particular (elementary) requirement of the website. Object Chunks are represented using OWL. As each Object Chunk models only one specific requirement, they can be seen as tiny ontologies. All Object Chunks together cover the complete domain of the website.

The goal of our approach is to automatically generate a website which content is annotated with one or more domain ontologies. Details about the generation process

are given in Section 4. In practice, three different cases may occur when designing a website. They are described below:

1. *No appropriate domain ontology exists or is available.* In this case, we incrementally build a new domain ontology as a result of the creation of the Object Chunks. This domain ontology covers the domain covered by the union of the Object Chunks, and therefore of the complete website. The Object Chunks can be seen as views on this new domain ontology.
2. *One domain ontology exists that covers the complete domain of the website.* The domain ontology is taken as basis for the conceptual design. The Object Chunks are defined as views on this domain ontology by selecting the appropriate concepts.
3. *Multiple domain ontologies are needed to cover the domain of the website.* To be able to define Object Chunks as views, we first have to align the different domain ontologies. This is done by defining a reference ontology and mappings between the domain ontologies and this reference ontology. The Object Chunks are then defined as views on this reference ontology.

Figure 3 shows an overview of the architecture covering all three cases. The different domain ontologies used are aligned by defining mappings to a reference ontology (called *Domain Ontology Mappings*). This reference ontology can also be used to define additional concepts not present in the used domain ontologies. Note that in the case of just one domain ontology, the sole purpose of the reference ontology is the latter one. In the case where there is no domain ontology available, the reference ontology plays the role of domain ontology that is incrementally constructed. The second mapping, called *Object Chunk Mappings*, defines Object Chunks as views on the reference ontology. This 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 that a person may have multiple email addresses (general case), but the website may require that persons can only enter exactly one email address (specific case). Both the Domain Ontology Mappings and the Object Chunk Mappings are defined using MAFRA (MApping FRAmework for Distributed Ontologies) [12].

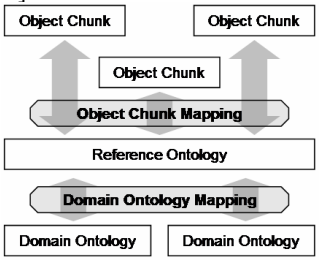

Fig. 3. General architecture illustrating the different mappings.

We give a small example illustrating the different mappings. We use the (shorter) description logic syntax for the object chunks, reference ontology and domain ontologies. Also a shorthand notation is used for the mappings themselves instead of the XML syntax of MAFRA. A domain ontology 1 describing the university domain contains the following axioms: {*Person* $\sqsubseteq$ $\forall$*name*.*String*, *Student* $\sqsubseteq$ *Person* $\sqcap$

$\forall email.String\}$ (a person and student have respectively as name and email address a string as value). A second domain ontology 2 describes social aspects of citizens and contains the following axioms: $\{Person \sqsubseteq \forall hasName.String \sqcap \forall hasSSN.String\}$ (a person has as name and social security number a string as value). To align the two domain ontologies, we construct a reference ontology containing the following statements: $\{Person \sqsubseteq \forall name.String \sqcap \forall SSN.String, Student \sqsubseteq Person \sqcap \forall email.String\}$. The following Domain Ontology Mappings are defined to link the reference ontology and the domain ontologies (only showing the properties involved):

Reference ontology	Domain Ontology 1	Domain Ontology 2
name	name	hasName
SSN	-	hasSSN
email	email	-

Assume that the website requires a further refinement of *name* into *first name* and *surname*. An Object Chunk would contain the following axiom: $\{Person \sqsubseteq \forall firstName.String \sqcap \forall surname.String\}$. The following Object Chunk Mappings are defined:

Object Chunk	Reference ontology
firstName + surname	name

The advantages of defining mappings at the conceptual level compared to approaches defining mappings on an implementation level are manifold. We mention the most important ones below:

1. *Implementation independent*: annotations are done on a conceptual level, and can effortlessly be re-generated along with different implementations
2. *Consistency of annotations*: as the annotation process is performed on a conceptual level, the actual annotations (at instance level) are guaranteed to be consistent.
3. *Both static and dynamic websites supported*: the implementation generation process of WSDM (see Section 4) does not distinguish between static and dynamic websites; annotations are effortlessly generated for both types of websites

Conflicts

When defining the Object Chunk and the Domain Ontology Mappings we may need to resolve possible conflicts between respectively the Object Chunks and the reference ontology, and the domain ontologies and the reference ontology. The different types of conflicts can be classified into the following categories:

- *Structural heterogeneity problems.* Different designers may have a different view on the same domain. As ontologies are only an abstract view of a domain, this means that a same domain can be structured differently by different designers. This includes naming conflicts, datatype conflicts, granularity differences, ...
- *Semantic heterogeneity problems.* Two concepts with the same name can refer to different real world objects in the domain (homonyms); and two concepts with different names can refer to the same real world object (synonyms).

Due to space limitations, we cannot go into deeper detail on solving these conflicts. Instead, we refer to [5].

Data Source Mapping
When the implementation of the website is generated, the different web pages need to be filled out with actual data (as specified in the *page design*). The web designer may decide to use a data source (e.g. a relational database) to store this data. To be able to fill out these web pages correctly, a Data Source Mapping needs to be defined between the reference ontology and the data source. E.g. in the case of a relational database, the Data Source Mapping determines in which tables and columns instances are stored of which concepts of the reference ontology. Note that, as with Object Chunk Mappings and Domain Ontology Mappings, no one-to-one relation can be assumed. Consider as example a table 'Person(ID, name, SSN, email, code)' in a relational database storing name, SSN from persons and email address from students. The following mappings are defined (note the additional condition for email due to lack of built-in subtyping support in relational databases):

Reference ontology	Data Source
name	name
SSN	SSN
email	email WHERE code='S'

4 Implementation Generation Process

To generate the actual implementation of a semantically annotated website, a transformation pipeline is used. This pipeline takes the object chunks, navigational model, site structure design, template design and page design as inputs. The transformations necessary to generate the implementation of the website consists of three steps (without the annotations). Figure 2 gives an overview.

- *Implementation Mapping (T1):* the implementation platform is chosen (e.g. XHTML[2]), and the integrated model derived by the previous transformations is partially transformed towards the chosen platform. References to data (i.e., references to object chunks) are not yet processed.
- *Data Source Mapping (T2):* the references to statements in the object chunks are resolved and mapped to their data source. This results into executable queries using the appropriated querying formalism. This mapping can be performed fully automatically using the mapping from the reference ontology to the data source.
- *Query Execution (T3):* finally, the queries are executed and the actual pages can be generated by inserting the actual data. When the query execution phase is performed offline, a static website is created but when it is performed at runtime, a dynamic site is the result.

To be able to also generate the semantic annotations, the transformation pipeline needs to be extended. For each query that is executed, we make the result of the query

[2] See http://www.w3.org/TR/xhtml1/.

explicit in terms of the associated Object Chunk. Next, an annotation is created between the data on the web page and the instantiated Object Chunk. Let us clarify this with an example. Imagine that a query asking for the surname of a person returns "Plessers" and "Casteleyn". In the generated HTML code, we surround them with a span tag containing a unique ID:

```
<span id="7">Plessers</span>
<span id="8">Casteleyn</span>
```

Following the references to statements in object chunks that lead to the creation of the query executed (see T4), we instantiate the associated object chunk *c*:

```
<Person rdf:ID="001">                <Person rdf:ID="002">
  <surname>Plessers</name>             <surname>Casteleyn</name>
</Person>                            </Person>
```

Finally, we link the generated HTML code and object chunk instantiations together (semantic annotations) using XPointer expressions. Consider the following example where 'page.html' is the generated HTML page and 'c' refers to the name of the object chunk:

```
page.html#xpointer(id("7")) <=> c#xpointer(id("001")/surname)
```

5 Related Work

When reviewing the literature concerning semantic annotations, we can distinguish three different, basic 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. with NLP techniques). Examples of automatic approaches include Melita [1], KMI annotation framework [11], etc.

Manual annotation approaches offer the user tool support to define annotations for HTML documents. The first tool was the SHOE Knowledge Annotator [8] 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 [7] (adding support for dynamic web pages by annotating database implementations).

Both manual and automatic approaches suffer some disadvantages that can be solved by integrating the annotation process into a web engineering method. The adequacy of automatically generated annotations is generally lower compared to manual approaches, the disadvantage of manual approaches however is that the annotations are defined on an implementation level (making them more vulnerable for changes) and require a substantial effort from the designer after the website is already implemented. Recently, research has therefore been focused on integrating semantic web technology into web design methods. Examples of semantic web design methods

include SHDM [14], Hera [6], OntoWeaver [10], OntoWebber [9], Seal [13], etc. These methods use ontology languages (e.g. RDFS, OWL) as modeling language for their internal design models. This has the advantage that existing ontologies can be reused in the design process and that a verification of the design models is feasible. Some of these approaches offer the possibility to make the internally constructed data models externally available (in the form of RDFS or OWL). However, none of these approaches allow *annotating* the web content 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 its semantics (semantic annotations) is required to support for example content rating and filtering[3].

As far as the authors are aware of, the only similar approach to the one described in this paper, is WEESA [17]. However, WEESA is not a design method by itself, but rather an extension to existing design methods that specify their design models in XML. It is able to generate semantic annotations by defining a mapping between the XML schemas and existing ontologies. The disadvantage of WEESA is that they cannot directly use domain ontologies created/reused during the web design process, but instead need to define this mapping regardless if a domain ontology was used during the design process or not. This means that data modeling is done twice: once in the XML schema, once in the domain ontology used.

6 Conclusion

In this paper, we described how the website design method WSDM has been adapted to suit the needs of the semantic web. The most important changes are the use of semantic web technology (i.e. OWL) to formally describe (the semantics of) the different WSDM design methods, and to model the available data and functionality of the website. We informally described the WSDM Ontology, and subsequently showed how the use of OWL to model information and functionality during the design process can be exploited to (semi-) automatically generate semantic annotations for the resulting website.

Our approach takes into account the different cases where 1) no existing domain ontology is available, 2) an existing domain ontology is used, and 3) multiple existing domain ontologies are used. In the latter case, we pointed out possible conflicts, and indicated how to solve them. In any of those cases, the modeling effort of the web designer is effectively re-used to generate semantic annotations. Furthermore, the following advantages over existing annotation methods can be pointed out: implementation independence, consistency of generated annotations and support for both static and dynamic websites.

To conclude this paper, we proposed an approach that bridges WSDM, a classical website design method and the semantic annotation process to generate annotated websites for the Semantic Web. The annotation process has become an intrinsic part of web design.

[3] See http://www.w3.org/TR/rdf-pics

References

1. Ciravegna, F., Dingli, A., Petrelli, D., Wilks, Y.: User-System Cooperation in Document Annotation based on Information Extraction. In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 02), Sigüenza Spain (2002)
2. De Troyer, O. and Leune, C. : WSDM: A User-Centered Design Method for Web Sites. In Computer Networks and ISDN systems Volume 30, Proceedings of the 7th International World Wide Web Conference, Elsevier (1998) 85-94
3. De Troyer, O., Casteleyn, S.: Modeling Complex Processes for Web Applications using WSDM. In Proceedings of the Third International Workshop on Web-Oriented Software Technologies (2003)
4. De Troyer, O., Casteleyn, S.: Exploiting Link Types during the Conceptual Design of Web Sites. In International Journal of Web Engineering Technology, Vol 1, No. 1, Inderscience, ISSN 1476-1289 (2003) 17-40
5. De Troyer, O., Plessers, P., Casteleyn, S.: Conceptual View Integration for Audience Driven Web Design. In CD-ROM Proceedings of the WWW2003 Conference, IW3C2 (also http://www2003.org/cdrom/html/poster/), Budapest Hungary (2003)
6. Frasincar, F., Houben, G.-J.: Hypermedia presentation adaptation on the semantic web. In Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, volume 2347 of Lecture Notes in Computer Science. Springer, ISBN 3-540-43737-1 (2002) 133-142
7. Handschuh, S., Staab, S.: Authoring and annotation of web pages in CREAM. The Eleventh International World Wide Web Conference (WWW2002), Honolulu Hawaii USA (2002)
8. 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) 35-40
9. Jin, Y., Xu, S., Decker, S., Wiederhold, G.: OntoWebber: A Novel Approach for Managing Data on the Web. In Proceedings of ICDE (2002) 488-489
10. Lei, Y., Motta, E., Domingue, J.: Modelling Data-Intensive Web Sites with OntoWeaver. In proceedings of the International Workshop on Web Information Systems Modelling (WISM 2004), Riga Latvia (2004)
11. 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)
12. Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA - A Mapping Framework in the Semantic Web. In Proceedings of the ECAI Workshop on Knowledge Transformation, Lyon France (2002)
13. Maedche, A., Staab, S., Studer, R., Sure, Y., Volz, R.: Seal - Tying up Information Integration and Web Site Management by Ontologies. IEEE Data Engineering Bulletin, 25(1) (2002) 10-17
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, ISBN 0-7695-2237-8 (2004) 106-113
15. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer-Verlag, ISBN 1-85233-155-0 (1999)
16. Plessers, P., De Troyer, O.: Annotation for the Semamtic Web during Website Development, In Proceedings of the ICWE 2004 Conference, Lecture Notes in Computer Science 3140, Eds. Nora Koch, Piero Fraternali, and Martin Wirsing, ISBN 3-540-22511-0, Munich Germany (2004) 349-353
17. Reif, G., Gall, H., Jazayeri, M.: WEESA - Web Engineering for Semantic Web Applications. In Proceedings of the 14th International 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. The 13th International Conference on Knowledge Engineering and Management (2002)