Web Design for the Semantic Web

Peter Plessers*, Olga De Troyer
Vrije Universiteit Brussel, Department of Computer Science, WISE, Pleinlaan 2, 1050 Brussel, Belgium
{Peter.Plessers, Olga.DeTroyer}@vub.ac.be

Abstract

To be able to realize the vision of the semantic web an important bottleneck that needs to be solved is an easy and intuitive approach for the annotation of websites with semantic information. Annotating websites defines the containing data in a form which is suitable for interpretation by machines. In this paper, we present a new approach to annotate websites by taking the annotation process to a conceptual level and by integrating it into an existing website design method. By this means, we are able to solve some of the problems current annotation solutions have.

1. Introduction

The importance of being able to express the semantics of the presented information on the Word Wide Web (WWW) was neglected for a long time. It was the vision of the Semantic Web [1] that brought this issue to the foreground. The idea of the Semantic Web states that the information available on the WWW should be defined such that it remains usable for human interpretation, but also becomes usable for machines. Realizing this vision, some limitations of the current WWW (e.g. its restricted query possibilities) can be solved. Although a lot of work has been done in recent years in the research domain of the Semantic Web, an easy and intuitive approach for authoring websites with semantic markup still remains an important bottleneck. As mentioned in [13], the generation of such semantic markup should be a by-product of normal computer use.

A step towards this goal has been taken in recent years by annotation approaches such as SHOE [11] [12], MindSwap [7] and CREAM [9]. The earliest annotation systems were based on a manual editing of the HTML pages to add the needed semantic information. Already soon, this process of manual editing proved to be a cumbersome and erroneous task and the necessity of supporting tools became undisputable. The most well-known annotation tool is the SHOE Knowledge Annotator [12] of the SHOE project which allows the user to annotate existing web pages using a graphical user interface. While such tools solve a number of issues like syntactic mistakes or inconsistencies with the used ontology, a number of fundamental problems still remain.

The main reason for these problems is that current tools define a linkage between an ontology and the actual data of the website on an implementation level resulting in a strong weaving of semantics and implementation. We list some of the problems we encounter in current annotation approaches:

- Despite the introduction of supporting tools, the annotation process remains a very heavy and time consuming task. In addition, in most current approaches this process is an additional activity and the ones that will benefit from the annotations are usually not the ones that should accomplish the job. Therefore, the motivation for performing the annotation process is low.
- It is usually assumed that the granularity of the concepts defined in the ontology matches exactly the granularity of the data on the website, although this assumption cannot be taken for granted. It must therefore be possible to define a link between semantically equivalent concepts but with a different level of granularity.
- Most of the supporting tools only allow annotating static websites, page by page on an implementation level. Even approaches that support the annotation of dynamically generated websites (by annotating the database) create a direct link between the implementation structure of the database (i.e. tables and columns for a relational database) and concepts in the ontology. For static web pages this has as a consequence that the work done for one page needs to be repeated for similar structured web pages and that the maintenance of the metadata becomes a heavy task with a huge cost. Also note that for both static and dynamic websites, every time one changes the implementation of the website or database, even though nothing has changed to the semantics of the presented data, the defined linkage between the web pages or database and the ontologies can be affected.

* This research is partially performed in the context of the e-VRT Advanced Media project (funded by the Flemish government) which consists of a joint collaboration between VRT, VUB, UG, and IMEC.

In this paper we show that elevating the annotation process to a conceptual level, provides an answer to the problems mentioned before. It is also our belief that (whenever possible) the annotation is best performed while designing the website, not after it is implemented. In this way we can take advantage of the information available during the website design process to ease and improve the annotation process. Therefore, we propose to integrate the annotation process into an existing website design method. Several website design methods have already been proposed in literature. We will use WSDM (Web Site Design Method) [3] [4] in our approach as this method is well suited for our purpose. It uses an explicit information-modeling step at a conceptual level. In fact, we propose an approach that bridges classical website design methods and annotation techniques developed for the Semantic Web. Using website design methods in the context of the Semantic Web can provide great value and benefits for the annotation process.

The rest of the paper is organized as follows. In section 2 we give a short overview of existing annotation approaches. We present an overview of our approach in section 3. In section 4 and 5, more details on the important aspects of the method are given, making use of a small example. The next section lists the advantages of our approach and the paper is concluded with future work and conclusions.

2. Related work

Current annotation approaches in use are fully decoupled from existing web design methods. The most well-known approach is the SHOE Knowledge Annotator [12] of the SHOE project. It provides the user a form-based graphical user interface to markup existing web pages using SHOE ontologies without having to worry about syntax. This tool only supports the annotation of static web pages, no support for dynamic pages is provided. The annotation process also remains an additional task that needs to be performed after the website is completed. Furthermore, it doesn't give any support to solve the granularity problem between the data on a website and the concepts of an ontology (as mentioned earlier in the introduction).

Another system is the SMORE (Semantic Markup, Ontology and RDF Editor) application [15] of the MindSwap project which is based on the same principles as the SHOE Knowledge Annotator, but provides a more advanced user interface. It contains an embedded HTML editor, web – and ontology browser which allow the user by means of drag and drop to create web page elements as instances of ontology concepts. The Ont-0-Mat tool [10] of the CREAM project uses a similar graphical user interface. Both tools allow annotating web pages by markup and by authoring. SMORE has also the possibility to create a new ontology borrowing concepts of existing web ontologies.

CREAM is, as far as we know, the only approach that supports the annotation of dynamically generated websites. Opposite to the annotation tools previously mentioned, the database is annotated instead of the HTML page. The following information is published to a web page to be able to link concepts of a given ontology to tables and columns of a data source: 1) which database is used and how the database can be accessed; 2) which query is used to retrieve data from the database; and 3) which elements of the query result are used to create the dynamic web page. Using this information it can be defined which data on the web page is originated from which column of which table. By defining a linkage between the database columns and concepts of an ontology, semantic meaning is added to the data stored in the columns.

Nevertheless the linkage between the database and the ontology is defined at a somewhat higher level than is done between static HTML pages and ontologies, the linkage is still done in an implementation-dependent way. As can be seen in the case of CREAM which supports dynamic pages, the direct linkage between the database columns and the concepts in the ontology can be easily broken by a change in the structure of the database. This shows that an annotation approach on a higher level - a conceptual level - is necessary.

3. Overview of the approach

Figure 1 gives an overview of the global architecture of our annotation approach. The different phases of WSDM that are relevant for our annotation approach are at the left: Task Modeling, Navigational Design, Page & Presentation Design, Database Design and finally the Implementation. Our approach is integrated into the original phases of the WSDM design method. A short overview of each step of the WSDM method, together with the enhancements (if any) we made for our annotation approach, is given below.

- *Mission Statement Specification:* Specifies the subject and goal of the website and declares the target audience. No enhancements are needed in this step.
- *Audience Modeling:* In this phase the different types of users are identified and classified into audience classes. For each audience class, the different requirements and characterizations are formulated. Also in this step, nothing additional is needed.
- *Task Modeling:* A task model is defined for each requirement of each audience class. Each task defined in the task model is elaborated into elementary tasks. For each elementary task a data model (called 'object

chunk') is created, which models the necessary information and/or functionality needed to fulfill the requirement of that elementary task. ORM (Object Role Modeling) [8] is used as the representation language for the object chunks. For our purpose, we added an annotation process to the Task Modeling phase. This results in the creation of a linkage between the object types and roles of the different object chunks and the concepts of one or more ontologies. This annotation is called the *conceptual annotation* (arrow A in Figure 1) because it is performed on a conceptual level. In this way we define the semantic meaning of the object types and roles used in the object chunks. This conceptual annotation is performed for static as well as dynamic websites.

- *Navigational Design:* In this phase of WSDM the navigational structure of the website is described by defining components, connecting object chunks to those components and linking components to one another.
- *Page Design:* During Page Design, the components of the navigational structure and their associated object chunks are mapped onto a Page structure defining the pages that will be implemented for the website. We determine which object chunks will be placed on a certain page. Using this step as well as the previous one (the navigational design) we can identify which object chunks will be placed on a page. This is necessary to know for the actual implementation which annotations we have to add to a page.
- *Presentation Design:* For each page defined in the Page Design a page template is created defining the layout of the page. This layout is defined in an implementation independent way. To implement the actual web pages making use of a chosen implementation language (e.g. HTML, XML, …), an instantiation of these page templates can be generated. For this, the templates are filled using the proper data to obtain the actual pages.
- *Data Design:* As explained in [6] we can derive an integrated conceptual schema from the object chunks made during Task Modeling. This integrated object schema is called the Business Information Model (BIM) and can be used as the basis for a database schema from which an underlying database can be created. The Data Design is only done when we deal with dynamically generated websites querying a database. For static web pages the data design step is omitted as the actual data will not originate from a database, but will be supplied by the designer during implementation. For our approach, we need to keep track of two mappings: 1) the mapping from the object types and relationships of the different object chunks to their correspondence in the integrated BIM (called *object chunk mapping*) (B in Figure 2); and 2) the mapping between the BIM, used as the conceptual database schema, and the actual implementation (called *database mapping*) (C in Figure 2). In this way we are able to determine the mapping between the queries specified at the (conceptual) level of the object chunks, and the actual database.

Implementation: In this phase of WSDM the actual implementation of a website, based on the models created in the previous phases, is generated. To this step we added the generation of the actual annotation of the website (called the *page annotation*) (D in Figure 2). Here we have to distinguish between static websites and dynamically generated websites. For static websites only the conceptual annotation is needed. For dynamic websites also the chunk integration and the database mapping have to be taken into consideration. A more detailed explanation is given in section 5.

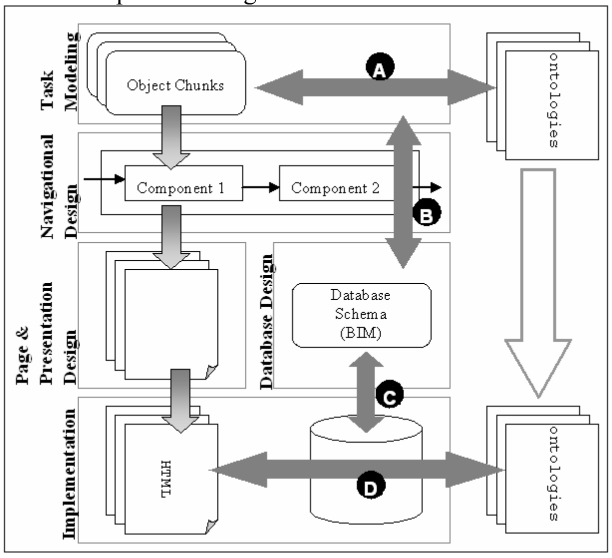

Figure 1 - Architectural overview

4. Conceptual Annotation

To explain the different steps in our approach, we introduce a simple example situated in the domain of universities. Assume the following two requirements for a university website:

1. We want to be able retrieve a list of all the labs with their associated research domain(s) and the name of the professor who is the head of the lab.
2. It must be possible to see some detailed information of all employees (professors, assistants, technical personnel, …) working for a certain department.

These requirements are formulated during the Audience Modeling phase of the WSDM method. The information needed to fulfill these requirements is expressed by means of two object chunks given respectively in Figure 2 and

Figure 3. These object chunks are constructed during Task Modeling.

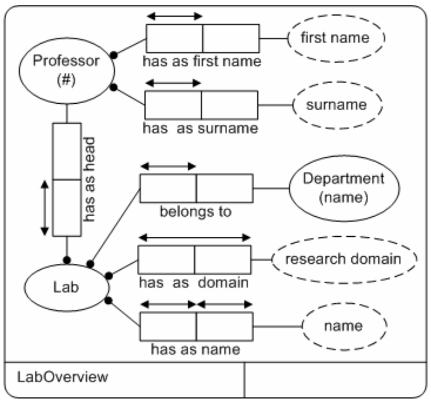

Figure 2 – Object Chunk LabOverview

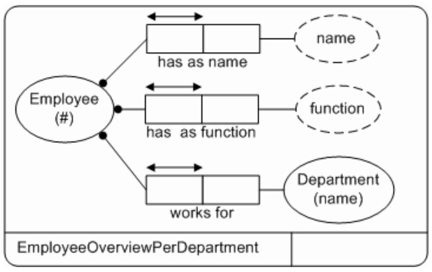

Figure 3 - Object Chunk EmployeeOverview

As already explained, while creating an object chunk, the designer performs the *conceptual annotation*. The designer will create associations between the concepts used in the object chunk (the object types, e.g. 'Professor', 'Lab', 'research domain', … and the roles e.g. 'works for', 'has name', …) and semantically equivalent concepts defined in one or more ontologies. In this way, we allow designers to define the meaning of the different object types and roles they introduce during conceptual modeling. As already indicated, this conceptual annotation is used to generate automatically the actual page annotation for the website implementation.

The conceptual annotation is defined as a mapping from the different object chunk entities (object types and roles) onto the different ontology entities (concepts and relationships). We distinguish between three different cases:

- One-to-one mapping: an object chunk entity can be mapped in a one-to-one way onto an ontology entity;
- One-to-many mapping: an object chunk entity cannot be mapped onto one single ontology entity but on a combination of ontology entities;
- Many-to-one mapping: an object chunk entity cannot be mapped onto one single ontology entity but a combination of object chunk entities can be mapped on a single ontology entity;

The conceptual annotation is illustrated for the 'LabOverview' object chunk (see Figure 2) in Table 1. The left column contains the different object chunk entities; the right column lists the corresponding ontology concepts and relationships. The used ontology itself is omitted in this paper due to space limitations. Note that for the entities 'first_name' and 'surname', we define the conceptual annotation as a many-to-one mapping between the tuple <first name, surname> and the ontology concept 'name'. It would be incorrect to define a direct annotation between the object type 'first name' and the ontology concept 'name' or/and the object type 'surname' and the ontology concept 'name'. The other conceptual annotations are all defined as one-to-one mappings.

Object Chunk Entities	Ontology Entities
Professor	Professor
Lab	Lab
<first name, surname>	name
research domain	researchField
has name	hasName
name	labName
…	…

Table 1- Conceptual Annotation example

We conclude this section with a possible outline of implementation of this Conceptual Annotation. For a one-to-one mapping, the annotation is straightforward as we can define a direct link between the entity in the object chunk and the ontology entity. This is not possible for the one-to-many and many-to-one mappings. To solve this, we introduce an intermediate ontology, called *Extended Ontology*. This ontology extends the ontologies used; it contains new entities that are constructed from the existing ones by applying some operators (e.g. the concatenation) on these entities. We introduce this intermediate ontology because it is not always allowed to modify or extend an existing ontology (e.g. because of a lack of sufficient permissions). For our example, the Extended Ontology will contain three new concepts: 'first_name', 'surname' and 'name', where we define 'name' 1) equivalent to the concept 'name' in the original ontology; and 2) as the concatenation of 'first_name' and 'surname' in the Extended Ontology. Then, a one-to-one mapping from respectively the object type 'first name' to the Extended Ontology concept 'first_name' and from the object type 'surname' to the Extended Ontology concept 'surname' is possible.

5. Generating the page annotation

Starting from the conceptual annotation provided by the designer(s), the actual page annotation can be generated. Note that the conceptual annotation is the only information that is requested from the designers (concerning the annotation process), as the following steps can be done automatically. For this generation process a distinction has to be made between static and dynamic websites. For static web pages, at this point of the method, all necessary information is gathered. Through the conceptual annotation we can trace which ontology concepts are associated with the object chunk entities and by the Page Design we know which object chunk entities will be implemented on a page.

5.1 The object chunk mapping

In case of a website dynamically generated from the content of a database a database design need to be done. In WSDM, the database design is done during the Data Design phase by integrating the different object chunks into one integrated schema, called the Business Information Model (BIM). The conceptual annotation can be used to drive the integration process as it identifies semantically equivalent and related object types (e.g. it can be derived that the object type 'Professor' in the 'LabOverview' object chunk is a subtype of the object type 'Employee' in the 'EmployeeOverview' object chunk if the ontology concepts linked with these object types are also involved in a subtype relationship). For a more in depth overview of the object chunk integration itself, we refer to [6]. We illustrate the chunk integration with our example. Assume that the conceptual design only consists of the two object chunks given in figure 2 and 3., Then, the integrated schema is shown in Figure 4. During integration it was recognized that a 'name' (of an employee) is equivalent with the concatenation of 'first name' and 'surname' (of this employee) (this can e.g. be derived from the conceptual annotation). Therefore, it was decided to keep the 'first_name' and 'surname" for an employee and to drop the 'name'. Therefore, the object type 'name' (as an employee's complete name) is not included in the BIM because it would be superfluous.

It should be noted that it couldn't be assured that there always exists a one-to-one mapping between an object chunk entity and an entity of the BIM. In general, an entity of an object chunk is mapped onto a view of the BIM. Let us illustrate this with our example. Take for instance the role 'has as first name' defined between the object types 'Professor' and 'first name' in our 'LabOverview' object chunk (see Figure 2). In the integrated BIM this information is modeled by means of a role that is more general, i.e. a role between 'Employee' and 'first_name' and the fact that 'Professor' is a subtype of 'Employee'. Then, the mapping of the role 'has as first name' is as follows:

```
'has as first name' → 'has as first
name'where <'Employee' is 'Professor'>
```

Note that `'has as first name' <'Employee' is a 'Professor'>` is the view expressing that we only should consider the role 'has as first name' for those 'Employee' instances which are also instances of 'Professor'.
If we consider the second object chunk (Figure 3) with the object type 'name', the mapping of this object type would be as follows ('X' s the operator to express the Cartesian Product):

```
'name' → 'first_name' X 'surname'
```

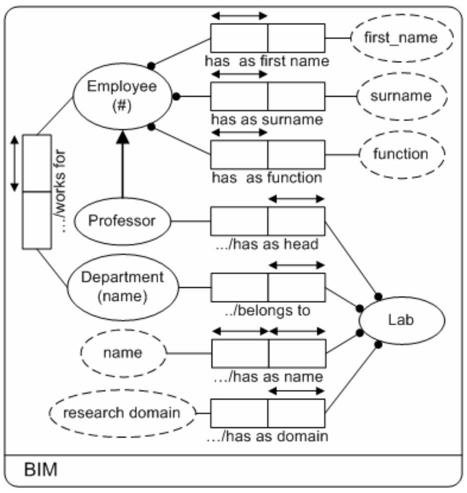

Figure 4 - BIM example

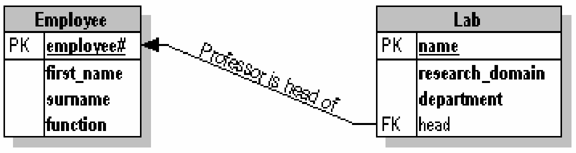

Figure 5 - Database schema

5.2 The database mapping

The next step is to generate an actual (relational) database schema from the BIM. This can be done using one of the known mapping algorithms for ORM like for example RMap [8]. Which mapping algorithm is used is not important, but the mapping has to be made explicit. This is essential, because we need to know in which database columns we can find the instances of a particular object type or role. Again, we cannot assume that the algorithm maps an object type or role to exactly one column. In general, an entity of the BIM will be mapped onto a (relational) view in the data schema.

We have applied the RMap algorithm to our example. Figure 5 shows the resulting database schema. Note that the column 'function' in the table 'Employee' is used to check if an 'Employee' instance is an instance of 'Professor' (for professors, the value of 'function' will be 'P'). If we take the object type 'Professor' (see Figure 4), we see that this object type is mapped on a part of the Employee table, expressible by a relational view:

```
  'Professor' → Employee
  where <function = 'P'>
```

5.3 The page annotation

We conclude this section with the generation of the actual page annotation for the chosen implementation.

As stated already during the overview of our approach in section 3, a page template is created for each page defined in the Page Design phase of WSDM. These templates will be instantiated to construct the actual web pages. Using the previously defined conceptual annotation, these page templates can be extended (automatically) with semantic annotations (the page annotations). It is know which object chunk entities will be instantiated by a template, and by the conceptual annotation we know the ontology entities to which these object chunk entities refer. This suffices to generate the page annotation. Below, the HTML version of the page template for the 'LabOverview' object chunk is given. This page template will allow listing all labs ordered by department, their associated research domain(s) and the professor who is the head of the lab (given by his first name and surname). By the definition of a template, the actual data is absent and is substituted by comment fields (v1, v2, …). Note that the template already defines the semantic annotation of these data. Also note that although several instances of e.g. 'Lab' can be listed on a page, we only had to annotate the concept and not each instance (in contrary to existing annotation approaches).

```
<!-- ontologies used -->
<html xmlns:u="http://.../university"
  xmlns:e="http://.../extended-ontology">
…
<body>
<table>
…
<h1><span ID="1"><-- v1 --></span></h1>
<ul>
 <li>
  <h3><span ID="2"><!-- v2 --></span></h3>
  <table width="60%" border="0">
   <tr>
    <td width="20%">Research domain:</td>
    <td width="80%">
     <span ID="3"><!-- v3 --></span>
    </td>
   </tr>
   <tr>
    <td width="20%">Head:</td>
    <td width="80%">
     <span ID="4"><!-- v4 --></span>
     <span ID="5"><!-- v5 --></span>
    </td>
   </tr>
  </table>
 </li>
</ul>
…
</table>
</body>
</html>

<u:depName ID="1">
  <label><-- v1 --></label>
</u:depName>
<u:labName ID="2">
  <label><-- v2 --></label>
</u:labName>
<u:researchField ID="3">
  <label><-- v3 --></label>
</u:researchField>
<e:first_name ID="4">
    <label><-- v4 --></label>
</e:first_name>
<e:surname ID="5">
    <label><-- v5 --></label>
</e:surname>
<e:name ID="6">
  <collection>
    <rdf:li rdf:resource="#4" />
    <rdf:li rdf:resource="#5" />
  </collection>
</e:name>
<u:Department ID="7">
  <u:hasDepName rdf:resource="#1" />
</u:Department>
<u:Lab ID="8">
  <u:hasName rdf:resource="#2" />
  <u:hasAsDomain rdf:resource="#3" />
  <u:belongsTo rdf:resource="#7" />
</u:Lab>
<u:Professor ID="9">
  <u:hasName rdf:resource="#6" />
  <u:isHeadOf rdf:resource="#8" />
</u:Professor>
```

The comment fields v1, v2, v3, v4 and v5 are substitutes for the actual data that will be used to instantiate the page template. The surrounding span tags refer to instances of ontology entities defined at the bottom of the page template. As one can see, v1 refers to the instance with ID="1" of a concept 'depName' (referring to the name of a Department). Note that v1 is also used to instantiate the label property of the ontology concepts. An instance of a Professor (see ID="9") has a name (see ID="6"), which is the collection of both his first name and surname, and is the head of a Lab (see ID="8") with a certain name (see ID="2"), an associated research domain (see ID="3") and belongs to a Department (see ID="7").

To instantiate such a template, the comment fields are being replaced with the actual data. We distinct two different manners to instantiate these page templates depending on whether we deal with static or dynamic web pages. For static websites, it is the designer himself who is responsible for instantiating the different page templates. Note that the designer only has to add the data to the template and doesn't need to worry in any way about the semantics of the data as this is already included in the page template. For dynamic web pages, the content is generated using the underlying database. Both the object chunk mapping and the database mapping are used to accomplish this task. The object chunk mapping describes the mapping of the object chunk entities onto the BIM, and the database mapping describes the mapping of the BIM onto the database schema. If we compose both mappings, we can derive the exact (SQL) query needed to obtain the data from the (relational) database using the conceptual query formulated at the level of the object chunk. The output can be inserted into the page templates where the value comment tags are replaced with the actual data. Details about the query are omitted due to space limitations. Whether the results of the query are placed on either one or more, separated pages is decided during the Page Design.

6. Advantages

The goal of our approach is to add semantic knowledge to the web pages of a new to create website. Opposed to current approaches, which perform the annotation on the web page level or on the database level (for dynamic websites), we define the annotation on a conceptual level. Web designers will provide the annotation during the conceptual design. Compared to currently existing annotation methods, this approach has a number of advantages:

- *The annotation is implementation independent.* Current methods define the annotations directly in the implementation of the website: in case of static websites the annotation is unswervingly weaved with the markup codes; for dynamic websites there is a direct association defined between the ontology and the database implementation. Changing for example a web page, without altering the meaning of the content, can needlessly require modifications to the annotations. Using our approach, an implementation will be generated (HTML, XML, …) and changes can be generated without breaking the annotation, resulting in a greater level of maintainability of the annotation.
- *From the designer's point of view, the annotation process is uniform for static and dynamic websites.* In current approaches the annotation for static and dynamic websites is done in a different way: respectively annotating web pages or a database. In our approach, the annotation step is done at the conceptual design which is independent on whether the website will be static or dynamic.
- *Workload reduction.* It is our belief that the best moment to define the meaning of the content of a website is when you are defining it. In most other methods, the annotation process is executed only after the website is completely implemented. In addition, in our approach, each designer is only responsible for defining the meaning of the object chunks he creates, and not for the entire content of the website. For large websites, developed by different persons, this can be a serious reduction of the workload and reduces the need for an overall domain expert.
- *Reuse of the annotations.* In current annotation methods (for static websites), if a certain concept is used on different pages, the annotation has to be repeated for each page. In our approach, the annotation has to be defined only once and the same concept can be reused in different object chunks. Moreover, all copies of an entity used over several object chunks will be updated automatically if the annotation of one copy has changed.
- *Improvement of the design process.* An important aspect of integrating the annotation into the design process is that it enables us to improve the consistency during the website design process and to speed it up by making use of the metadata already provided. It is for example possible to make suggestions to the designer about information to be included based on earlier conceptual annotations made (possibly made by the co-designers).

7. Conclusion

In this paper, we presented an approach for the semi-automatic annotation of static as well as dynamic websites. The actual annotation process is performed during the design phase of the website at a conceptual level. We presented the proposed approach integrated into an existing website design method, WSDM. This design method provides us a conceptual model of the website that can be used to annotate (at a type level) the information that will be available on the website with concepts from an ontology. This is done by annotating the entities (object types and roles) used in the conceptual model of the website. Next this "conceptual" annotation can be used to generate the actual page annotation by keeping track of the different transformations performed during the development process to derive at an implementation

To realize our approach, we only had to add a single step to the already existing phases of the WSDM method: the *conceptual annotation,* which defines the association between the entities in the object chunks and concepts in an ontology. The actual *page annotation* which defines data on the website as instances of the associated ontology concepts is generated automatically. Note that it was needed to keep explicitly track of the *object chunk mapping* and *database mapping* to be able to generate this page annotation.

The current conceptual annotation is still limited: it only provides limited support for multiple ontologies, there is not yet any support for solving semantic conflicts, and it still neglects the problem of domain- and structural conflicts. A lot of improvement can and must be achieved in this area. This will be the topic for further work.

References

[1] T. Berners Lee, J. Hendler, O. Lassila, "The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities", *Scientific American* 2001: 5(1).

[2] S. Ceri, P. Fraternali, A. Bongio, "Web Modeling Language (WebML) : a Modeling Language for Designing Web Sites", *In proceedings of the 9th World Wide Web Conference (WWW2000)*, Amsterdam, 2000.

[3] O. De Troyer, C. Leune, "WSDM: A User-Centered Design Method for Web Sites", *Computer Networks and ISDN Systems, proceedings of the 7th International World Wide Web Conference*, Brisbane Australia, 1998, pp. 85–94.

[4] O. De Troyer, "Audience-Driven Web Design", *Information Modeling in the New Millennium, Eds. Matt Rosi & Keng Siau,* IDEA GroupPublishing, ISBN: 1-878289-77-2, 2001.

[5] O. De Troyer, S. Casteleyn, "Modeling Complex Processes for Web Applications using WSDM", *Proceedings of the IWWOST2003 workshop*, Oviedo Spain, 2003.

[6] O. De Troyer, P. Plessers, S. Casteleyn, "Solving Semantic Conflicts in Adience Driven Web Design", *Proceedings of the WWW/Internet 2003 Conference*, Algarve Portugal, 2003.

[7] J. Golbeck, M. Grove, B. Parsia, A. Kalyanpur, J. Hendler, "New Tools for the Semantic Web", *Proceedings of EKAW 2002, LNCS 2473, Springer*, 2002, pp. 392–400.

[8] T. Halpin, "Information Modeling and Relational Databases", 3rd edition, Morgan Kaufmann, 2001.

[9] S. Handschuh, S. Staab, A. Maedche, "CREAM – Creating Relational Metadata with a Componentbased, Ontology Driven Framework", *Proceedings of K-Cap*, Victoria Canada, 2001.

[10] S. Handschuh, S. Staab, "Authoring and annotation of web pages in CREAM", *The Eleventh International World Wide Web Conference (WWW2002)*, Honolulu Hawaii USA, 2002.

[11] J. Heflin, J. Hendler, S. Luke, "SHOE: A knowledge Representation Language for Internet Applications", *Technical report CS-TR-4078 (UMIACS TR-99-71)*, 1999.

[12] J. Heflin, J. Hendler, "Searching the web with SHOE", *Artificial Intelligence for Web Search, Papers from the AAAI Workshop, WS-00-01*, AAAI Press, 2000, pp. 35-40.

[13] J. Heflin, J. Hendler, "Agents and the Semantic Web", *IEEE Intelligent Systems Journal, 16(2)*, 2001, pp. 30–37.
[14] D. Schwabe, G. Rossi, "An Object Oriented Approach to Web-Based Application Design", *Theory and Abstraction of Object Systems 4(4)*, Wiley and Sons New York, 1998.

[15] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, F. Ciravegna, "MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup", *The 13th International Conference on Knowledge Engineering and Management*, 2002.