Using WEESA to Semantically Annotate Cocoon Web Applications

Gerald Reif, Harald Gall
Department of Informatics, University of Zurich
Binzmuehlstrasse 14
CH-8050 Zurich, Schwitzerland
{reif,gall}@ifi.unizh.ch

ABSTRACT

The Semantic Web is based on the idea that Web applications provide semantically annotated Web pages. This meta-data is typically added in the semantic annotation process which is currently not part of the Web engineering process. Web engineering, however, proposes methodologies to design, implement and maintain Web applications but lack semantic annotation. In this paper we show how WEESA, a mapping from XML documents to ontologies, can be used in Apache Cocoon Web applications to semantically annotate Web pages. We introduce Cocoon transformer components that use the WEESA mapping definition to automatically generate RDF meta-data from XML documents. We further show how existing Cocoon Web applications can be extended to Semantic Web applications and discuss the experiences gained in an industry case study.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Information Storage and Retrieval; D.2 [**Software**]: Software Engineering

General Terms

Design

Keywords

Web Engineering, Semantic Web, Semantic Annotation, Ontology

1. INTRODUCTION

The increasing popularity of the WWW has lead to an exponential growth in the number of Web pages available, which makes it increasingly difficult for users to find required information. In searching the Web for information, one gets lost in the vast number of irrelevant search results and may miss relevant material. Current Web applications provide Web pages in HTML format representing the content in natural language only and the semantics of the content is therefore not accessible by machines. To enable machines to support the user in solving information problems, the Semantic Web proposes an extension to the existing Web that makes the semantics of the Web pages machine-processable [3]. The semantics of the information of a Web page is formalized using RDF meta-data describing the meaning of the content. The existence of semantically annotated Web pages is therefore crucial in bringing the Semantic Web into existence.

In [20, 21] we introduced WEESA, a technique to extend existing XML-based Web engineering methodologies to develop semantically annotated Web applications. The novelty of this approach is the definition of a mapping from XML elements/attributes to concepts defined in an ontology. This mapping can then be taken to automatically generate RDF meta-data from XML content documents. WEESA can therefore be used to extend XML based Web applications to provide RDF meta-data in addition to HTML Web pages.

The contribution of this paper is the integration of the WEESA meta-data generation into the Apache Cocoon Web development framework. For this purpose we developed two WEESA enabled transformer components. We show how these transformers can be used to develop semantically annotated Web applications and discuss the experiences gained while implementing an industry case study.

The remainder of the paper is structured as follows. Section 2 introduces WEESA. Section 3 presents the Apache Cocoon pipeline model. Section 4 discusses the association of HTML Web pages and RDF meta-data and Section 5 and 6 shows the integration of WEESA into Cocoon transformers and how this transformer can be used in XML based Web applications. Section 7 presents the case study, 8 discusses the related work, and Section 9 concludes the paper.

2. WEESA META-DATA GENERATION

In this section we briefly introduce Web engineering and show how WEESA [21] can be used to engineer *Semantic Web applications*, that provide HTML content and machine-processable RDF meta-data.

Web Engineering focuses on the systematic design, development, and maintenance of Web applications [6]. Most Web engineering methodologies are based on separation-of-concerns to define strict roles in the development process and to enable parallel development. A popular way to separate the content from the graphical design is the use of XML

SAAW2006 1st Semantic Authoring and Annotation Workshop 2006 at the 5th International Semantic Web Conference, November 5-9, 2006, Athens, Geogria, USA

and XSLT, where XML focuses on the content of the Web page and XSLT defines the design.

In order to develop Semantic Web applications, the traditional Web engineering techniques have to be extended by the semantic annotation process. In this process, the Web pages have to be semantically annotated with RDF meta-data. This enables machines to have access to the content of the pages. Several tools such as the SHOE Knowledge Annotator [9], the CREAM OntoMat [7], and SMORE [12] have been proposed to support the user when annotating existing Web pages. However, to manually annotate Web pages is only feasible if the number of Web pages is small and the content does not change frequently.

To annotate dynamic Web pages, that obtain their content from a background logic such as a database, the annotation process should be integrated in the engineering process of the Web application. During the engineering of a Web application, information items can be identified more easily than in the generated HTML Web pages. Once the information item is identified, it is mapped to a concept defined in the ontology. For example, the result of a database query is mapped to a property in the ontology, to indicate that a new RDF statement has to be generated with the query result as value for the property.

The aim of WEESA (WEb Engineering for Semantic web Applications) is on the one hand, to integrate the annotation process into the engineering process of the Web application. On the other hand, to reuse the existing design artifacts of XML based Web applications to develop Web applications that provide semantic meta-data in addition.

In WEESA the structure of the XML document is used to identify XML elements/attributes which are mapped to concepts in an ontology. This mapping definition can then be taken to automatically generate RDF meta-data from XML content documents. WEESA not only allows one-to-one mappings but uses a more flexible approach to overcome the gap between the information available in the XML document and the information needed by the ontology. The content of several XML elements/attributes can be selected and further processed in Java methods to meet the semantics of the concept in the ontology.

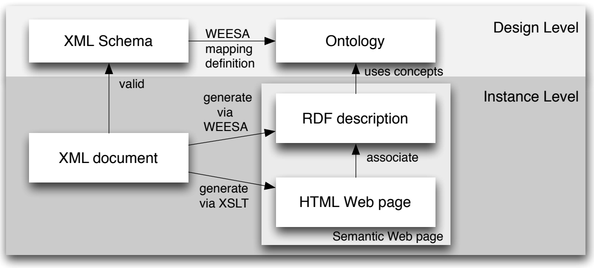

Figure 1: WEESA design and instance level.

At the design level of the Web application only the structure of the XML document is known and no XML instances are available. Therefore we use the structure information from the XML Schema to define the mapping to the ontologies. In this way, the same XML document that follows the XML Schema is used for the XSLT transformation to generate HTML and for WEESA to generate RDF meta-data. Therefore, the XML Schema can be seen as the contract the content editor, responsible for writing the XML documents, the layout designer, responsible for the graphical appearance, and the engineer defining the WEESA mapping have to agree on. Figure 1 shows the definition of the WEESA mapping at the design level of the Web application and how this mapping is used at instance level to automatically generate RDF meta-data and the HTML page from XML documents.

3. COCOON PIPELINE CONCEPT

In this section we introduce the concept of Apache Cocoon Web applications. Apache Cocoon [4] is a Web development framework built around the concepts of separation-of-concerns and component-based Web development. Cocoon uses component pipelines to build Web applications where each component on the pipeline is specialized on a particular operation.

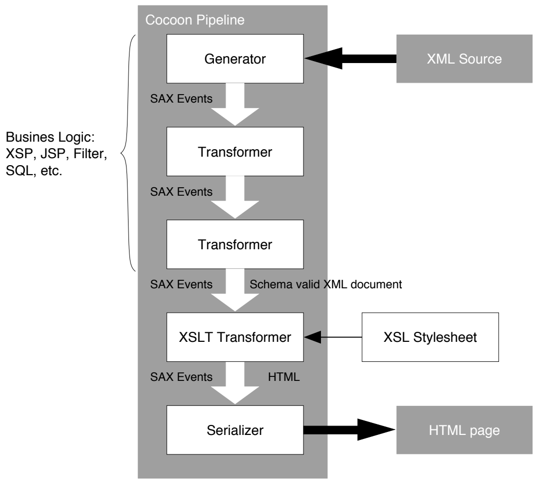

Figure 2: Pipeline of a typical Cocoon Web application.

Figure 2 shows the pipeline of a typical Cocoon Web application. A Cocoon pipeline consists of a *generator*, an arbitrary number of *transformer*, and a *serializer*. An HTTP request triggers the pipeline and causes the generator to read XML from a data source and produces a stream of SAX[1] events as output representing the XML document. This output is the input of a transformer or a serializer. A transformer takes the SAX events, does some transformation (e.g. XSLT transformation), and the results are again SAX events. These events can then be taken by another transformer or a serializer. In a typical Cocoon Web application, the business logic (e.g. SQL queries, Java code) is processed by the transformers at the beginning of the pipeline. The output of the business logic is a schema valid XML document that fulfills the Web engineering contract and has the structure the XSLT stylesheet was designed for. This document is then taken by the XSLT transformer which uses

[1]SAX: Simple API for XML

the XSLT stylesheet to produce the HTML page. The serializer finally takes the SAX events and processes them into a character stream for client consumption.

4. ASSOCIATING HTML AND RDF

As shown in Figure 1, we have to perform two steps on the schema valid XML document to generate the semantically annotated Web pages: (1) The XSLT transformation has to be performed to generate the HTML page. (2) The WEESA mapping has to be processed to generate the RDF meta-data description of the Web page.

Once we have generated the HTML page and its RDF meta-data description we have to put them into relation. Unfortunately, no standardized approach exists for associating RDF descriptions with HTML. In [17] Palmer discusses several possible approaches. These techniques can be classified in the following categories:

Embedding RDF in HTML: With this association style the RDF description is directly embedded in the HTML page. Several ways have been proposed such as adding RDF/XML in the `<script>` element, using XML notations and `CDATA` sections, or adding RDF base64 encoded in the `<object>` element. A detailed discussion of these methods can be found in [17].

A recently proposed technique to embed RDF into HTML is RDFa [1]. RDFa is the current W3C working draft for integrating RDF data into XHMTL documents and is not yet supported with the approach presented in this paper.

Linking to an external RDF description: The RDF meta-data description is stored in an external document and the HTML page references to its meta-data description. This reference can be done using the HTML `<link>` element [17] or using an common HTML `<a href="">` link [8].

In the following two sections we introduce a WEESA enabled Cocoon transformer for both categories of association styles.

5. WEESA COCOON TRANSFORMER TO GENERATE HTML+RDF

One way to associate HTML and RDF is to embed the RDF/XML description in the `<script>` element within the HTML `<head>`. The HTML `<script>` element can be used to include non-HTML media in HTML Web pages [10]. This section introduces the `WEESAReadDOMSession` transformer that can be used to realize this kind of RDF – HTML association.

When developing Semantic Web applications that add the RDF meta-data to the HTML page we have to introduce new steps to the pipeline discussed in Section 3. Since we need the schema valid XML document for the XSLT transformation and for the WEESA meta-data generation, we have two options. We can either integrate WEESA in a modified XSLT transformer that generates RDF and HTML or we can split up the pipeline and use a specialized transformer for the RDF meta-data generation.

One of the strength of Cocoon is, that every component is specialized on a specific task. This gives the developer the flexibility to configure the components in the pipeline according to his requirements. Therefore we decided, to split up the pipeline to generate HTML pages with embedded RDF/XML.

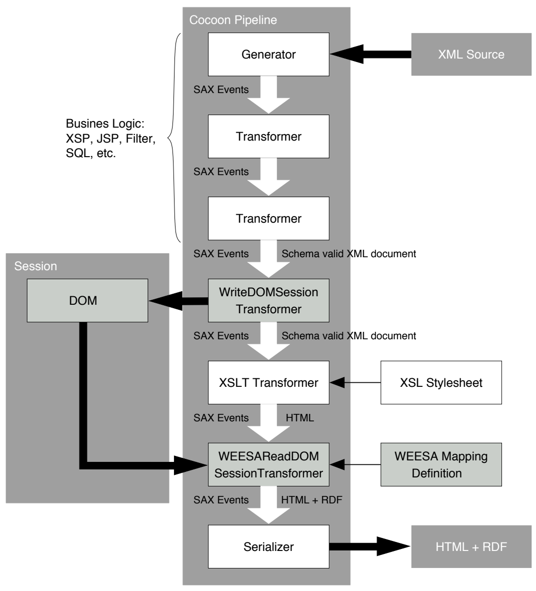

Figure 3: Cocoon pipeline for the WEESA HTML+RDF generation.

To split up the pipeline we use the `WriteDOMSession` transformer. This transformer takes the input document and writes it first as DOM[2] into the servlet session, and second as SAX events to its output. This is how the pipeline is split up and the XML document can be reused later in the pipeline. After the HTML page is generated by the XSLT transformer the `WEESAReadDOMSession` transformer takes the DOM-XML from the session and uses the WEESA mapping definition to generate the RDF meta-data representation in RDF/XML format. The `WEESAReadDOMSession` transformer further embeds the generated RDF/XML in the `<script>` element which is then added in a user defined element of the HTML page. This element is typically the `<head>` element. The serializer finally delivers the HTML+RDF page to the client. The additional steps are shown in Figure 3 as light gray pipeline components. The problems that comes with embedding the RDF meta-data in the `<head>` element are discussed at the end of the this section.

The Cocoon framework uses the `sitemap.xmap` configuration file to define the pipelines. Figure 4 shows a fragment of the sitemap file for the pipeline from Figure 3. Lines 3-6 instruct Cocoon to start a servlet session. In line 9 the generator is instructed to read the XML document `AlanisMorissetteUnplugged.xml` from the `content/` directory. In lines 14-17 the `WriteDOMSession` transformer is defined to write the XML document to the session. The `dom-name` parameter gives the DOM-object the name `rdf` in

[2]DOM: Document Object Model

```
 1 <map:match pattern="AlanisMorissetteUnplugged.html">
 2   <!-- create the session -->
 3   <map:act type="session">
 4     <map:parameter name="action" value="create"/>
 5     <map:parameter name="mode" value="immediately"/>
 6   </map:act>
 7
 8   <!-- read the XML file -->
 9   <map:generate type="file"
10      src="content/AlanisMorissetteUnplugged.xml"/>
11
12   <!-- here comes the business logic (if needed) -->
13
14   <!-- write the XML document to the session -->
15   <map:transform type="writeDOMsession">
16     <map:parameter name="dom-name" value="rdf"/>
17     <map:parameter name="dom-root-element"
18                    value="content"/>
19   </map:transform>
20
21   <!-- do the XSLT transformation -->
22   <map:transform type="xslt"
23                  src="style/albumInfo.xslt"/>
24
25   <!-- configure WEESAReadDOMSession transformer -->
26   <map:transform type="WEESAReadDOMSession">
27     <map:parameter name="dom-name" value="rdf"/>
28     <map:parameter name="trigger-element"
29                    value="head"/>
30     <map:parameter name="position" value="in"/>
31     <map:parameter name="weesa-mapping-definition"
32                    value="mapping/albumMapping.xml"/>
33   </map:transform>
34
35     <!-- serialize the output -->
36   <map:serialize type="html"/>
37 </map:match>
```

Figure 4: Pipeline definition using the `WEESAReadDOMSession` transformer.

the servlet session. The use of the `dom-root-element` parameter is explained below. Line 20 configures the XSLT transformer to use the XSLT stylesheet `albumInfo.xslt`. In lines 23-29 the `WEESAReadDOMSession` transformer is configured. The `dom-name` parameter tells which DOM-object should be taken from the session, the `weesa-mapping-definition` parameter names the mapping file, and the `trigger-element` and `position` parameters tell that the generated RDF/XML should be placed `in` the HTML `<head>` element. Finally, in line 32 the serializer is instructed to write the XML stream as `HTML` to the consumer.

In praxis, most Web pages consist of several parts, for example a header, the navigation part, and a part with the actual page content. A sample XML document following this structure is shown in Figure 5. Depending on the structure of the Web page, not all parts of the XML document have to be looked at when generating the RDF meta-data. For example, only the `<content>` part contains information that is needed for the meta-data generation. In this case, we have to define the WEESA mapping only for the subtree starting with the `<content>` element and we have to inform the `WriteDOMSession` transformer only to write the subtree following the `<content>` element as DOM to the session. This is done with the `dom-root-element` parameter in line 16 from Figure 4.

Including RDF/XML meta-data in the HTML page using the `WEESAReadDOMSession` transformer has the advantage that the business logic, typically processed at the beginning of the pipeline, has to be computed once only for both the HTML and RDF generation. Embedding the RDF/XML meta-data in the `<head>` tag of a HTML document, however, breaks HTML 4.01 and XHTML validity [19].

```
<page>
  <header>
    <!-- here goes the XML for the header -->
  </header>
  <navigation>
    <!-- here goes the XML for the navigation -->
  </navigation>
  <content>
    <!-- here goes the XML for the content -->
  </content>
</page>
```

Figure 5: Sample XML document aggregated from several parts.

6. WEESA COCOON TRANSFORMER TO GENERATE RDF/XML

Another way of associating RDF and HTML to one another is to use the `<link>` element in the `<head>` of the HTML page to reference the corresponding external RDF/XML meta-data description [17]. To generate this stand-alone RDF/XML description within Cocoon Web applications we developed the WEESA transformer.

Cocoon Web applications that use the `<link>` element to associate RDF and HTML need two pipelines: one for the generation of the HTML page, and another for the WEESA meta-data generation. The pipeline for the HTML generation is similar to the one introduced in Section 3. Only the reference to the RDF description has to be added in the `<head>` of the HTML page. The reference looks as follows:

```
<link rel="meta" type="application/rdf+xml"
               href="AlanisMorissetteUnplugged.rdf"/>
```

There are basically two ways of adding the `<link>` element to the HTML page. One is to modify the Web application to add the element in the business logic or in the XSLT stylesheet. The other possibility is to use the `AddRDFLink` transformer we developed. This transformer is added to the pipeline for the HTML page generation between the XSLT transformer and the serializer. The `AddRDFLink` transformer extracts the URL of the incoming request, replaces the `".html"` suffix of the path with `".rdf"`, and adds the `<link>` with the `".rdf"` URL in the `<head>` of the HTML page.

Since the `AddRDFLink` transformer searches for a `".html"` suffix in the URL and replaces it with `".rdf"` it can only be used in Web applications that obey the following naming convention. The path in URLs that trigger the pipeline for the HTML pages have the suffix `".html"` such as:

```
http://www.mytunes.com/album.html?id=1234
```

The path in URLs that trigger the pipeline for the RDF meta-data generation have the suffix `".rdf"` such as:

http://www.mytunes.com/album.rdf?id=1234

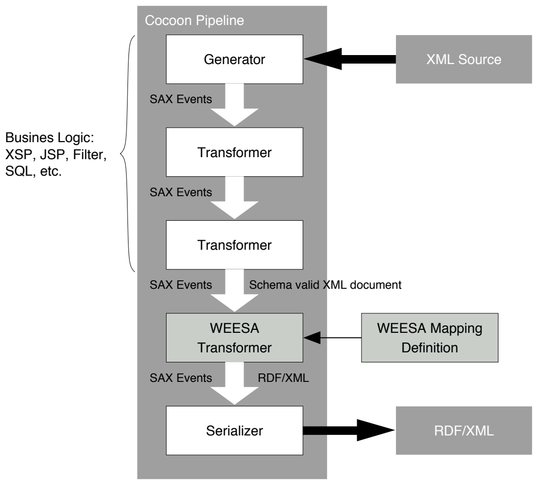

Figure 6: Cocoon pipeline for the WEESA RDF/XML generation.

In the pipeline for the WEESA meta-data generation that uses the WEESA transformer the business logic is also processed at the beginning of the pipeline. The schema valid XML document is sent to the WEESA transformer that takes the mapping definition and processes the WEESA mapping. The RDF/XML output of the transformer is then taken by a serializer and sent to the client. The pipeline is shown in Figure 6 and Figure 7 shows a snipped from `sitemap.xmap` that is used to configure the WEESA transformer. Again, the `dom-root-element` parameter defines the start element of the XML subtree that should be considered for the WEESA transformation as described in the section above, and the `weesa-mapping-definition` parameter defines the mapping definition to be used.

```
<map:transform type="WEESA">
  <map:parameter name="dom-root-element"
                                  value="content"/>
  <map:parameter name="weesa-mapping-definition"
                 value="mapping/artistMapping.xml"/>
</map:transform>
```

Figure 7: Configuration of the WEESA transformer.

Using the `<link>` element to associate RDF and HTML has the advantage that the RDF description has to be generated on request only. This, however, has the drawback that the schema valid XML document has to be generated a second time. Fortunately this does not have to be a big disadvantage if the Cocoon caching mechanism can be used.

7. CASE STUDY

We have evaluated the WEESA transformers in the annual Vienna International Festival[3] (VIF) industry case study. VIF is a database supported Web application that comprises a ticket shop, over 60 event descriptions, reviews, and an archive over the last 52 years. For the WEESA case study we use the event descriptions and the ticket shop and semantically annotate the corresponding Web pages. Below we list the Web pages that make up the case study and discuss the meta-data the pages provide:

VIF Homepage: Entry point to the VIF Web application. It provides general information about the festival and a navigation bar to the features of the Web application. The meta-data consists of the contact information of the festival.

Program overview pages: List of events of the festival. There is one list of all events and several lists for the events of a specific event category (such as concert, performing arts, etc.) The meta-data consists of the events, location, and the category of the event.

Event description: Detailed information about the event. It contains information such as the title, description, location, date, etc. of the event. The meta-data reflects the event details provided by the Web page.

Ticket shop receipt: Receipt of the bought ticket. It is the final acknowledgement of the shopping process in the online shop containing all the details of the specific event. The meta-data reflects event details of the bought ticket.

When we chose the case study application we decided not to design a new Web application from scratch but to adopt an existing one. This decision has the advantage that we are able to work out the differences between the design of traditional Web applications and Semantic Web applications.

The VIF Web application was originally implemented using MyXML [13]. MyXML is a Web application framework that is based on separation-of-concerns. For our case study we took the existing database and reimplemented the Web application based on the Cocoon Web application framework. To do so, we followed the steps introduced in Section 3. We first defined the XML Schema as contract of the Web application. In the following steps we implemented the business logic, designed the XSLT stylesheets, and defined the Cocoon pipelines, as we would do for a traditional Web application. To semantically annotate the Web application we defined the WEESA mapping for each kind of Web page and added the WEESA enabled transformer into the pipeline, as introduced in Section 5 and 6. Since WEESA follows the concept of separation-of-concerns all these steps could be performed in parallel.

During the life cycle of a Web application a new version of the used ontology may be issued or a new ontology become the standard ontology to describe information in the domain of the Web application. Therefore, WEESA was designed to enable the change of the used ontology without reimplementing the whole Web application. To demonstrate this flexibility we implemented the case study first with a self-defined VIF ontology and changed later to the iCalendar [11] ontology. We further did the implementation for both HTML – RDF association styles introduced in Section 4. Figure 8 shows a snipped from the generated RDF graph that is based on the self-defined VIF event ontology.

[3]http://www.festwochen.at

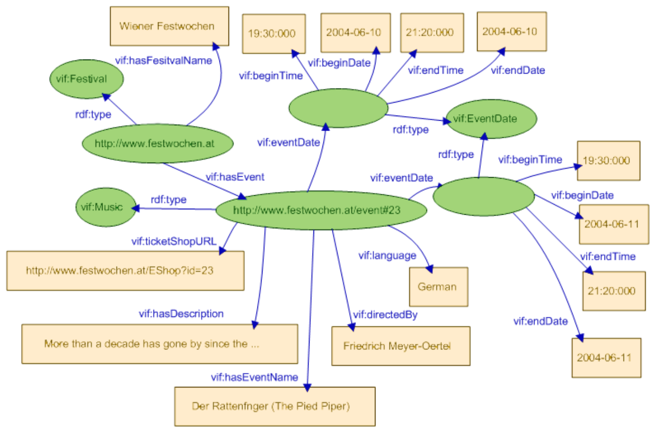

Figure 8: Snipped of the generated RDF graph.

In the following we list experiences gained while developing the case study:

Annotation by configuration: The change from a traditional Web application to a WEESA Semantic Web application requires basically two steps: the WEESA mappings have to be defined and the Cocoon pipelines have to be modified. Both of these steps can be done by either writing new XML files (mapping definition) or modifying existing ones (pipeline configuration). No Java programming is involved in these steps.

In addition, the change between the two association styles of HTML and RDF, also could be done by only modifying the pipeline configuration. The WEESA mapping definitions remains the same.

Change of ontologies: We implemented the VIF case study Semantic Web application for different ontologies. For the change from one ontology to the other, only the WEESA mapping definition used had to be changed. The rest of the implementation remained untouched and no additional programming effort was needed.

Java Mapping Library: In the two paragraphs above we argued, that no Java programming is involved to semantically annotate a Cocoon Web application using WEESA. This is true regarding the Web application. To not only support one-to-one mappings from XML to the ontology Java methods are used in the WEESA mapping definition. WEESA provides a library of Java methods for common tasks. In some cases, however, the developer has to implement a Java method for a user-specific task. This user specific method can be added to the library for later reuse.

Free-Text and Mixed Content: Since WEESA uses the structure of the XML document to identify the concepts that are mapped to the ontologies, free-text and mixed content can not be annotated. Natural language understanding would be needed to do so. However, in our experience this is not a problematic limitation since the concepts that can be found in many ontologies available today can also be found in the structure of an XML document.

Database Keys for RDF Resource Identifier: The VIF case study further showed that database keys should be accessible in the XML documents to be able to generate unique resource identifiers for the RDF representation. The database keys help to ensure that the same identifier is used for the same resource throughout the whole Web application.

Well Formatted data in the XML document: Since the RDF meta-data description of the Web pages is intended to be machine-processable the literals used should store information in a well defined format. For example to represent time and date information the XML Schema `xsi:time` respectively, `xsi:date` format should be used. This requirement should also be kept in mind when defining the XML Schema as contract for a Web page. The schema should force content editors to provide information split up into logical units that are stored in a well defined format. For example, to specify the begin and end time of an event, it is better to have a specific XML element for the begin and end time instead of an arbitrary string such as `"from 11:00 to 12:00"`. Using arbitrary strings results in pattern matching, when defining the WEESA mapping.

8. RELATED WORK

To our knowledge not much work has been done to integrate the Semantic annotation process into Web engineering. In [18], the authors suggest an extension of the Web Site Design Model (WSDM). In this approach object chunk entities which are artifacts in the Web application design process are mapped to concepts in the ontology.

Many Web engineering methodologies have been proposed that use Semantic Web technologies such as ontologies and RDF to formalize engineering artifacts. The Extensible Web Modeling Framework (XWMF) [14] aims to use a machine-processable format for the Web engineering artifacts to make them exchangeable between the multitude of tools that are involved in the Web application life cycle. The Semantic Hypermedia Design Method (SHDM) [16] heavily uses OWL ontologies for domain and methodology specific issues. Ontologies are used for the conceptual model and the navigational model of the application domain. The SHDM further defines method specific ontologies for the abstract and concrete widget interface to model the user interface of the Web application. Despite these methodologies use engineering artifacts that rely on Semantic Web technologies, the actual Web application does not contain any semantic annotations.

There is further related work in the area of Semantic annotation. CREAM/OntoMat [7] is a Semantic annotation framework that offers several annotation methods such as manual annotation, authoring of annotated documents, semiautomatic annotation, and the annotation of dynamic pages. This flexible approach is, however, not integrated in the Web engineering process.

In the area of interpreting XML as RDF data several approaches exist. In [15], XML documents are interpreted as RDF data via a RDF Schema to enable machines to interpret XML unambiguously as a set of statements in the RDF data model. The round-tripping tool between XML and RDF [2] allows to directly interpret XML documents with a RDF model using the XML schema as basis for describing how XML is mapped into RDF and back. In [5] the idea is that every element/attribute name maps to a RDF property, viewing the structure of the XML document as relational model between parent nodes and their children. All these approaches rely on the equality of the XML element/attribute names and those of the class/property names in the ontology. This, however, cannot be guaranteed, since ontologies are typically defined by third parties.

9. CONCLUSIONS

The success of the Semantic Web crucially depends on the existence of semantically annotated Web pages. Current Web engineering methodologies and frameworks, however, lack techniques to provide semantic meta-data. To annotate Web pages, information items have to be identified and mapped to the concepts that are defined in ontologies. Since it is easier to identify these information items in structured data-sources (e.g., databases, XML), which are available while engineering the Web application, than in the generated HTML pages, we argue that the semantic annotation process should be integrated into the engineering process of the Web application.

In this paper we introduced Apache Cocoon transformer components that can be used to generate semantic meta-data out of existing Web engineering artifacts. These transformer use WEESA, a mapping from XML elements/attributes to concepts in ontologies, to automatically generate RDF from XML documents and can therefore be used to develop Semantic Web applications. In this paper we further presented the experiences gained while using WEESA for developing a Semantic Web application.

10. REFERENCES

[1] B. Adida and M. B. eds. *RDFa Primer 1.0 - Embedding RDF in XHTML*, 16 May 2006. `http://www.w3.org/TR/xhtml-rdfa-primer/`.

[2] S. Battle. Poster: Round-tripping between XML and RDF. In *International Semantic Web Conference (ISWC)*, Hiroshima, Japan, November 2004.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific America*, 284(5):34–43, 2001.

[4] The Apache Cocoon project homepage, Last visited February 2005. `http://cocoon.apache.org/`.

[5] M. Ferdiand, C. Zirpins, and D. Trastour. Lifting xml schema to owl. In *4th International Conference on Web Engineering*, pages 354–358, Munich, Germany, July 2004.

[6] M. Gaedke and G. Graef. Development and evolution of web-applications using the webcomposition process model. In *Int. WS on Web Eng. at the 9th Int. WWW Conf.*, Amsterdam, the Netherlands, May 2000.

[7] S. Handschuh and S. Staab. Annotation of the shallow and the deep web. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in Artificial Intelligence and Applications*, pages 25–45. IOS Press, Amsterdam, 2003.

[8] J. Hartmann and Y. Sure. An infrastructure for scalable, reliable semantic portals. *IEEE Intelligent Systems*, 19(3):58–65, May 2004.

[9] J. Heflin, J. Hendler, and S. Luke. Shoe: A blueprint for the semantic web. In D. Fensel, J. Hendler, H. Liebermann, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 29–63. The MIT Press, 2003.

[10] HTML 4.01: Definition of the script element, Last visited March 2005. `http://www.w3.org/TR/html401/interact/scripts#edef-SCRIPT`.

[11] iCalendar OWL ontology definition, April 7 2004. `http://www.w3.org/2002/12/cal/ical`.

[12] A. Kalyanpur, J. Hendler, B. Parsia, and J. Golbeck. SMORE - semantic markup, ontology, and RDF editor. Technical report, University of Maryland, 2003. `http://www.mindswap.org/papers/SMORE.pdf`.

[13] C. Kerer and E. Kirda. Web engineering, software engineering and web application development. In *3rd Workshop on Web Engineering at the 9th World Wide Web Conference*, pages 135 – 147, Amsterdam, the Netherlands, May 2000. Springer-Verlag.

[14] R. Klapsing and G. Neumann. Applying the resource description framework to web engineering. In *Proceeding of the 1st International Conference on Electronic Commerce and Web Technologies: EC-Web 2000*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[15] M. Klein. Using RDF Schema to interpret XML documents meaningfully. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*, volume 96 of *Frontiers in AI and Applications*, pages 79–89. IOS Press, Amsterdam, 2003.

[16] F. Lima and D. Schwabe. Application modelling for the semantic web. In *Proceedings of the 3th International Conference on Web Engineering (ICWE 2003)*, pages 417–426, Oviedo, Sapin, July 2003. Springer-Verlag.

[17] S. B. Palmer. RDF in HTML: Approaches, June 2002. `http://infomesh.net/2002/rdfinhtml/index.html`.

[18] P. Plessers and O. D. Troyer. Annotation for the semantic web during website development. In *4th International Conference on Web Engineering*, pages 349–353, Munich, Germany, July 2004.

[19] W3C: Frequently Asked Questions about RDF: How do I put some RDF into my HTML pages?, September 2004. `http://www.w3.org/RDF/FAQ#How`.

[20] G. Reif. *WEESA - Web Engineering for Semantic Web Applications.* PhD thesis, TU Vienna, 2005. `http://seal.ifi.unizh.ch/fileadmin/User_Filemount/Publications/reif-phdthesis05.pdf`.

[21] G. Reif, H. Gall, and M. Jazayeri. WEESA - Web Engineering for Semanitc Web Applications. In *Proceedings of the 14th International World Wide Web Conference*, pages 722–729, Chiba, Japan, May 2005.