Engineering the Presentation Layer for Semantic Web Information Systems

Flavius Frasincar
flaviusf@win.tue.nl

Geert-Jan Houben
houben@win.tue.nl

Peter Barna
pbarna@win.tue.nl

Richard Vdovjak
richardv@win.tue.nl

Eindhoven University of Technology
PO Box 513
NL-5600 MB Eindhoven, the Netherlands

ABSTRACT

This paper presents a design methodology that deals with the presentation aspects involved in the development of a Semantic Web Information System. The methodology is driven by two main design models: the conceptual model and the application model. The application model extends the conceptual model with presentation abstractions that capture the application logic. During the presentation generation the input data goes through a sequence of transformation steps. Using semantic web technology we chose to represent the models and their instances in RDF(S). The RDF/XML model serialization facilitates the specification of the different transformations in XSLT.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and principles; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*Architectures, Navigation*; I.2.4 [**Artificial Intelligence**]: Knowledge representation formalisms and methods

General Terms

Design, Experimentation

Keywords

WIS, SWIS, RDF(S), Semantic Web, RDF/XML, XSLT

1. INTRODUCTION

Five years ago the term Web Information System (WIS) [5] was coined to denote an information system that is based on Web technology. The next-generation Web, the Semantic Web [1], extends the current Web with technologies that support the representation of data semantics. Similar to the introduction of the WIS concept, we use the term Semantic Web Information System (SWIS) to label an information system that makes use of these new technologies during its development/deployment process. A SWIS is different from a WIS in the sense that it needs to fulfill (at least partially) the semantic interoperability prophecy of the Semantic Web.

At present there are not many design methodologies that use Semantic Web technology. One such methodology is the eXtensible Web Modeling Framework (XWMF) [7]. XWMF focuses on presentation modeling aspects using an extensible set of RDF schemas and descriptions. XWMF disregards the issue of presentation adaptation and doesn't elaborate on the methodology phases. The methodology presented in this paper not only considers the modeling aspects in the presentation design of a SWIS but also addresses the presentation adaptation and identifies the different methodology steps.

2. PRESENTATION METHODOLOGY

The proposed presentation methodology is part of the broader Hera methodology for designing SWISs. Hera has two main layers: the data retrieval and integration layer, and the presentation layer. The focus of this paper is on the presentation layer. In previous work [4] we have identified the main models which are presented only briefly here. In this section these models are extended with adaptation features and a stepwise methodology to automate the presentation generation for the input data is proposed.

From the existing Semantic Web technologies we chose to use RDFS [2] for representing the different models and RDF [8] to describe the model instances. The RDF/XML serialization of the models and their instances facilitate the usage of an XSLT [6] processor to perform the different methodology transformations.

The Conceptual Model (CM) is the schema that the application input data needs to comply with. It specializes two vocabularies: the CM vocabulary and the system media vocabulary. The CM vocabulary extends RDFS with `cardinality` and `inverse` properties. The system media vocabulary defines a hierarchy of media types. The CM is composed of concepts and concept properties. Concept properties refer to different concepts or to media types.

The Application Model (AM) is the presentation schema of the application. It specializes the AM vocabulary. The AM vocabulary defines the `Slice` and `Link` classes, and their list variants. The AM is composed of slices and slice properties. A slice refers to media properties from the CM. The `owner` of a slice is a concept from the CM. There are two types of slice properties: slice composition and slice navigation (hyperlink abstraction).

One of the advantages of using RDFS is the ability to reuse existing vocabularies like the CC/PP vocabularies for modeling device capabilities and user preferences. The user/platform profile

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

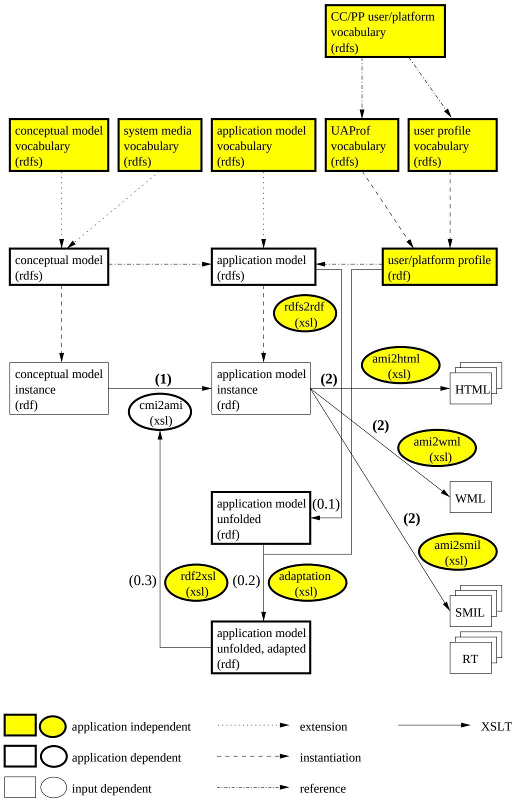
CC/PP user/platform vocabulary (rdfs)
conceptual model vocabulary (rdfs)
system media vocabulary (rdfs)
application model vocabulary (rdfs)
UAProf vocabulary (rdfs)
user profile vocabulary (rdfs)
conceptual model (rdfs)
application model (rdfs)
user/platform profile (rdf)
rdfs2rdf (xsl)
conceptual model instance (rdf)
(1)
cmi2ami (xsl)
application model instance (rdf)
(2)
ami2html (xsl)
HTML
(2)
ami2wml (xsl)
WML
(2)
ami2smil (xsl)
SMIL
RT
application model unfolded (rdf)
(0.1)
(0.3)
rdf2xsl (xsl)
(0.2)
adaptation (xsl)
application model unfolded, adapted (rdf)
application independent
application dependent
input dependent
extension
instantiation
reference
XSLT

Figure 1: Hera presentation methodology

is defined based on two CC/PP vocabularies: the existing UAProf vocabulary and our own vocabulary for defining user preferences. Based on this profile we add adapt the AM by adding visibility conditions to slices.

Figure 1 depicts the Hera presentation methodology. The above models are depicted by rectangles and can be application independent, application dependent, or input dependent. Their dependencies are classified as: extensions, instantiations, or references. Figure 1 also gives the methodology transformation steps, the XSLT transformations, that operate based on these models. These transformation steps are depicted by ellipses and they are application independent or application dependent. Each transformation is discussed in the sequel of this paper.

2.1 Preparation

The main transformation (labeled (1) in the figure) populates the AM with the input CM instance. In order to ease the specification of a stylesheet that performs the main transformation, three other substeps are identified during the preparation step. In the application model unfolding (labeled (0.1) in the figure) the skeleton of the AM instance is created. The RDFS representation of the AM is unfolded using the following stripping: class, property, class, property etc. In the next step, the application model adaptation (labeled (0.2) in the figure), the skeleton of the AM instance is adapted based on the slice visibility conditions. Slices that have their condition not fulfilled are discarded and links pointing to them are hidden. Considering the adapted skeleton of the application instance, a stylesheet to be used in the main transformation is generated (labeled (0.3) in the figure). This process is facilitated by the defined `owner` property that relates slices from the AM to concepts from the CM.

2.2 Main transformation

As stated above, the main transformation (labeled (1) in the figure) fills the AM with the input data (the CM instance). The result is an AM instance that is ready to be presented in different user browsers. The algorithm used in the main transformation is straightforward: each slice is projected on the instances of the associated (association based on the `owner` property) concept. A context based on the current concept instance identifier helps an appropriate traversal through the related concept instances.

2.3 Code generation

Three types of browsers have been considered HTML, SMIL, and WML. For each browser there is an associated transformation stylesheet (labeled (2) in the figure) that produces browser specific code for the AM instance. The code generation phase uses a media directed translation to produce browser specific code for each media item. The WML code generation discards images from the presentation. Note that the WML presentation consists only of one document (that contains all cards), the HTML presentation has one document per page, and the SMIL presentation has one SMIL document for each page and one RealText (RT) clip for each text item.

3. CONCLUSIONS

An experimental prototype using the proposed methodology was developed. RDF(S) proved to be a suitable formalism to represent the different models and their instances. Its extensibility feature was successfully used in interconnecting the different models. Nevertheless we had to add appropriate extensions to RDFS; this suggest to look at the possibility to use in the future a Web ontology language like OWL [3]. XSLT was designed as a transformation language for XML and not for RDF representations which limits the expressive power of the RDF(S) transformations specification. An RDF-aware transformation language (to our knowledge not existing yet) would be more appropriate in this context.

4. REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[2] D. Brickley and R. Guha. Rdf vocabulary description language 1.0: Rdf schema. W3C Working Draft 23 January 2003.

[3] M. Dean, D. Connoly, F. van Harmelen, J. Hendler, J. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Owl web ontology language 1.0 reference. W3C Working Draft 29 July 2002.

[4] F. Frasincar, G. J. Houben, and R. Vdovjak. Specification framework for engineering adaptive web applications. In *The Eleventh International World Wide Web Conference, Web Engineering Track*, 2002.

[5] T. Isakowitz, M. Bieber, and F. Vitali. Web information systems. *Communications of the ACM*, 41(1):78–80, July 1998.

[6] M. Kay. Xsl transformations (xslt) version 2.0. W3C Working Draft 16 August 2002.

[7] R. Klapsing and G. Neumann. Applying the resource description framework to web engineering. In *Electronic Commerce and Web Technologies, First International Conference, EC-Web 2000*, volume 1875 of *Lecture Notes in Computer Science*, pages 229–238. Springer, 2000.

[8] G. Klyne and J. J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. W3C Working Draft 23 January 2003.