Extending WebML towards Semantic Web

Federico M. Facca Marco Brambilla
Dipartimento di Elettronica e Informazione
Politecnico di Milano
P.za Leonardo da Vinci 32, I-20133 Milano, Italy
{facca,mbrambil}@elet.polimi.it

ABSTRACT

Available methodologies for developing Sematic Web applications do not fully exploit the whole potential deriving from interaction with ontological data sources. Here we introduce an extension of the WebML modeling framework to fulfill most of the design requirements emerging for the new area of Semantic Web. We generalize the development process to support Semantic Web applications and we introduce a set of new primitives for ontology importing and querying.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia— *Architectures, Navigation*; D.2.2 [**Software Engineering**]: Design tools and techniques—*Computer-aided software engineering (CASE)*

General Terms

Design, Languages, Theory

Keywords

Semantic Web, Design Method, Ontology, Web Engineering, Conceptual Model

1. INTRODUCTION

Modern Web applications comprise distributed data integration, remote service interaction, and workflow management of activities, possibly spawned on different peers. In this scenario, if semantics of data and applications is known, integration becomes more feasible. To address this challenge many semantic description languages arose like RDF, OWL and WSML. All these languages allow to formally model knowledge by means of ontologies: the resulting formal models are the starting point to enable easy information exchange and integration between machines. These languages are suitable for reasoning and inferencing, i.e., to deduct more informations from the model by applying logic expressions. This makes the modeling task easier since not all the knowledge has to be modeled. Unfortunately, although the theoretical bases and some technological solutions are already in place for Semantic Web support, the techniques and methodologies for Semantic Web application design are still rather rough.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

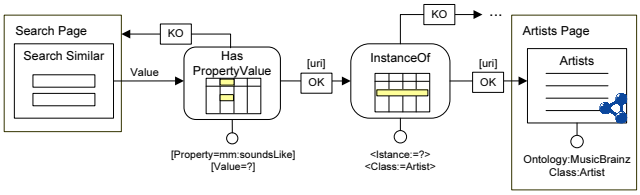

Figure 1: A piece of WebML model describing a Semantic Web application.

We claim that conceptual modeling can increase dramatically both efficiency and effectiveness of the design and implementation of such applications. At this purpose, we present an extension of the WebML domain specific language for Web application design, to model and develop Semantic Web applications that adopt Semantic Web technologies to better integrate distributed and semantic data sources or to provide semantic annotations of the deployed applications.

2. EXTENDING THE WEBML METHOD

We enriched every level of the WebML methodology proposed first in [2] by: (i) extending the development process to describe the tasks related to the design of ontologies and semantic aspects of the web applications/services; (ii) extending the data model to support semantic data sources (i.e., ontologies); (iii) extending the hypertext model for querying ontologies, with particular attention to advanced and inferencing queries; (iv) supporting semantic annotations of the applications in the presentation model.
In particular we extended the WebML basic primitives provided by the hypertext model (e.g., `Index` and `Data` units) to support ontological data sources (e.g., RDF/OWL ontologies) and we defined a new set of primitives specifically designed to exploit ontology features and reasoning over ontological data. This new units are aggregated primitives that, depending on the type of parameters, execute differently. These units (`SubClassOf`, `InstanceOf`, `HasProperty`, `HasPropertyValue`, `PropertyValue`, `SubPropertyOf`) aim at providing explicit support to advanced ontological queries. They allow to extract classes, instances, properties, values; to check existence of specific concepts; and to verify whether a relationship holds between two objects.

Figure 1 depicts a fragment of a WebML application that retrieves artists or albums whose names sound similarly to the name specified by the user. The sample ontology is the MusicBrainz ontology [4]. The `value` submitted in the form

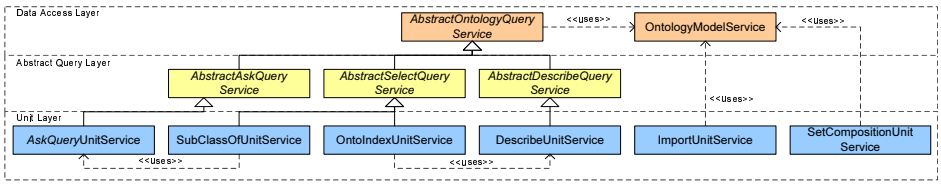

Figure 2: UML class diagram of the internal software architecture of the newly implemented units.

is passed to the `HasPropertyValue` unit that extracts a set of URIs of instances (albums or artists) that have `value` as value of the *mm:soundsLike* property. If no instance is found, the user is redirected to the `Search` page through the `KO` link. Otherwise, the set of URIs is passed to the `InstanceOf` unit that checks if they are instances of the class Artist. In this case, the URIs are passed through the `OK` link to an `Index` unit showing list of Artists, otherwise the URIs are passed on the `KO` to publish a list of Albums (not shown in the figure).

Moreover, each WebML semantic unit can automatically extract a RDF description of its contents. The designer has to specify how he wants to use the RDF fragments; for instance, he can aggregate the fragments of all the units in the page and publish the aggregate at the bottom of the page, as a global semantic annotation of the page itself; or he can maintain them separated and publish the RDF annotation for each unit in the page.

Besides the units for ontological data query, we introduce also three new units: the `Set Composition` unit performs set operations (i.e., union, intersection, difference) over two input sets of URIs, considering the hierarchy of the URIs involved; the `Import Ontological Source` unit adds a remote or local data source that must be consistent with ontological model of the web application (it's validated against it before being added to the ontology); the `Describe` unit returns the RDF description of a URI, thus enabling data exporting and semantic annotation of pages. The above mentioned querying units can be used to compose reasoning tasks over ontological data. E.g., suppose that we want to discover the common super concepts between two ontology classes; we can use two `SubClassOf` units to extract the two set of super classes to which the two classes belongs to; then we can find the common set of superclasses by means of the `SetComposition` unit. For instance, if we apply the previous pattern to two music genre classes like *Progressive Rock* and *Urban Hip-Hop*, we get their common superclasses (e.g., the *Rock* class).

3. ARCHITECTURE

We extended the WebRatio CASE tool [5] and its runtime libraries to support the new units. The prototype implementation is based on Jena framework [3] to interact with OWL/RDF ontologies. The design environment offered by Webratio has been extended exploiting the plug-in mechanism of the toolsuite: we devised a general purpose data access layer to ontological data sources, plus a runtime Java component and an XML descriptor for each unit.

To handle interaction with ontologies we defined a new data access layer, comprising a set of general purpose Java classes to be reused by all the new units for querying the ontology repositories. These classes provide facilities to import ontologies and to select OWL/RDF classes, properties, and instances (possibly filtered by one or more conditions). The main aspects of the class structure are represented in Figure 2.

The `OntologyModelService` enables connections to local and remote ontologies specified at design time or imported at runtime by mean of the `Import Ontological Source` unit. Three abstract classes offer the query services corresponding to the query methods offered by SPARQL on the ontology contents: the `AbstractSelectQueryService` class performs selection over data; the `AbstractDescribeQueryService` retrieves the RDF describing a given URI, the `AbstractAskQueryService` verifies simple predicates. The `AbstractAskQueryService` is extended by the `AskQueryService` that is used by some of the advanced querying units to verify predicates (e.g., to check whether a class is subclass of another). The new ontological primitives use or implement these services for performing their task.

4. CONCLUSION

We presented an extension to the WebML methodology and models for supporting the design and the specification of Semantic Web applications. In [1] we presented our vision on the needs and the opportunity of applying Web Engineering methods to the development of Semantic Web Services in the context of the WSMO framework. In particular we showed how, starting from a rich and annotated model of a Web Service, it is possible to automatically generate both the implementation of the Web Service and a large part of its semantic description. Here we described a solution that provides a full coverage of the development process, and allows the designer to specify at a high level of abstraction basic and advanced queries on ontological data sources, to import existing sources, and to annotate Web pages with semantic descriptions of the contents and of the models. We support our proposal with a prototype implementation within the CASE tool WebRatio.

5. REFERENCES

[1] M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. Della Valle, and F. M. Facca. A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *Proceedings of the 5th Int.l Semantic Web Conference (ISWC 2006)*, Nov 2006.

[2] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kauffmann, 2002.

[3] Jena Team. Jena a semantic web framework for java, 2007. http://jena.sourceforge.net.

[4] MusicBrainz. Musicbrainz project, 2007. http://musicbrainz.org.

[5] WebModels s.r.l. Webratio site development suite, 2007. http://www.webratio.com.