Building Semantic Web Portals with WebML

Marco Brambilla and Federico M. Facca

Dipartimento di Elettronica e Informazione
Politecnico di Milano
P.za Leonardo da Vinci 32, I-20133 Milano, Italy
{marco.brambilla,federico.facca}@polimi.it

Abstract. Current conceptual models and methodologies for Web applications concentrate on content, navigation, and service modeling. Although some of them are meant to address semantic web applications too, they do not fully exploit the whole potential deriving from interaction with ontological data sources and and from Semantic annotations. This paper proposes an extension to Web application conceptual models toward Semantic Web. We devise an extension of the WebML modeling framework that fulfills most of the design requirements emerging for the new area of Semantic Web. We generalize the development process to cover Semantic Web and we devise a set of new primitives for ontology importing and querying. Finally, an implementation prototype of the proposed concepts is proposed within the commercial tool WebRatio.

1 Introduction

Evolution of Web applications toward complex Web-based Information Systems dramatically increases the complexity of the requirements and of the technological issues associated to the design and development phases. Modern Web applications comprise distributed data integration, remote service interaction, and workflow management of activities, possibly spawned on different peers. In this scenario, a wider attention to semantics of data and applications is mandatory to allow effective design and evolution of complex systems, that can be possibly set up and manipulated by different organizations. Indeed, if semantics of data and applications is known, their integration becomes more feasible. Moreover, explicit semantic annotation of Web applications can facilitate content search and access and foster a future generation of Web client that exploit the semantic information to provide better browsing capabilities to customers.

The Semantic Web is an evolution of the World Wide Web, promoted by Tim Berners-Lee to bring "semantics" to the human-readable information so as to make them machine-readable and allow better and easier automatic integration between different Web applications. To address this challenge many semantic description languages arose, like RDF, OWL and WSML; some of them are currently W3C Recommendations. 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. This languages are supported by a wide range of tools and APIs, that support design of knowledge (e.g. Protégé, OntoEdit), provide storing facilities (e.g. Sesame and Jena), and offer reasoning on the data (e.g., Racer and Pellet). Based on these modeling languages, a set of querying languages have been devised too; among them, we can mention SPARQL, a W3C recommendation.

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. This leads to high costs of implementation for Semantic Web features, even if embedded within traditional Web applications. These extra costs are related not only to the design of the architecture and deployment of the Semantic platforms, but only to the repetitive and continuous task of semantic annotation of contents and application pages.

We claim that conceptual modeling can increase dramatically the efficiency and efficacy of the design and implementation of such applications, by offering tools and methodologies to the designer for specifying semantically-rich 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 [2]. 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. Now we focus on the extension of WebML as a Model Driven method to model and develop Semantic Portals. A Web Portal is a Web site providing personalized capabilities to its visitors and is designed to use distributed and different sources. A Semantic Web Portal adopts semantic Web technologies to better integrate distributed data sources and to provide semantic descriptions of its contents so as to make them machine-readable.

The introduction of Semantic Web applications brought a new set of requirements, to be fulfilled by methodologies and tools for such applications: e.g. easy reuse of existing ontological models, support for semantic web languages, advanced ontology query paradigms, easy specification of semantically rich descriptions of services, contents, and interfaces. Some of these requirements have been addressed by existing modeling methodologies for semantic Web applications and Semantic Portals [3,4,5,6,7].

The paper is organized as follows: Section 2 specifies the new requirements of Semantic Web; Section 3 presents the case study used throughout the paper; Section 4 briefly summarizes the principles of the WebML language; Section 5 presents the extensions to the language for supporting Semantic Web features, in terms of extensions to development process, content model, and hypertext model; Section 6 exemplifies the approach on the running case; Section 7 describes the implementation experiments; Section 8 discusses the related work; and finally Section 9 draws some conclusions.

2 Requirements for Semantic Web Engineering

To collect the requirements that a Semantic Web application should comply with, we analyzed some current online Semantic Web Portals (e.g., [8,9,10]) and we extracted the following set of needs:

– **Support of semantic languages.** Semantic Web applications should be aware of and support (i.e., be able to query and manage) different Semantic Languages and metamodels (RDFS [11], OWL[], WSML [2], . . .).
– **Semantic application models.** Semantic Web applications should be designed and specified by means conceptual models that include and support semantic descriptions.
– **Flexible integration.** Semantic Web applications should embrace the philosophy of flexibility and heterogeneity integration of Semantic Web.
– **Classes and instances access and queries.** Both domain ontology classes and instances should be easily and seamlessly accessible by Semantic Web applications, through appropriate querying primitives. Notice that, while queries in data-driven Web applications are only on data instances, a Semantic Web application may exploit structure querying too.
– **Inference and verification.** Ontology-based web applications should exploit available inferencing systems on ontological data, both for semantic queries and verification of data.
– **Semantic data sources.** A Semantic Web application relies on semantic data (e.g., ontologies) that offer a machine understandable data description that may be not only used to populate and generate Web pages, but also to automatically enrich such Web pages with semantic annotations.
– **Importing and reuse of ontologies.** Semantic Web applications shall allow to: *(i)* import new (possibly distributed) data conforming to the Web application ontology; *(ii)* to seamlessly integrate new ontologies, not fitting the default ontology; and *(iii)* to reuse existing and shared ontologies.

From the previous set of requirements, we derived the following requirements for the conceptual metamodels pursuing the design of Semantic Web applications:

– Metamodels should be aware of and support semantic languages.
– Metamodels themselves should be “semantic”, i.e., grant self-annotation and explicit semantic extraction.
– Metamodels should allow flexible integration of heterogeneous sources and applications.
– Metamodels should allow transformations towards a query language able to capture all the aspects of ontologies, including inference, verification, query on instances, and query on classes.
– Metamodels should easily allow: to specify semantic data sources as underlying level of the application; to exploit these sources for populating Web pages, and for (automatically) annotating such Web pages.
– Metamodels shall be able to import and reference distributed data and ontologies, aiming at the reuse and sharing of the knowledge.

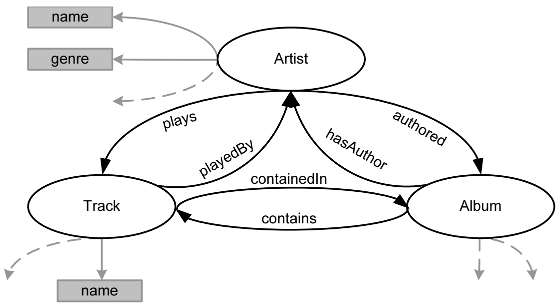


Fig. 1. A fragment of the MusicBrainz ontology representing Artist, Album, Track and their relationships

3 A Semantic Web Portal for the Music Domain

To discuss our approach, a running example will be used throughout the paper. To implement a realistic scenario, we will consider the reuse of one of the existing ontologies available on the Internet that can be easily used and integrated to create new Semantic Web Portals. We integrate two ontologies for the musical domain to build a Web application offering access to this kind of contents, considering also users profile information. In particular, we exploit the MusicBrainz ontology [12] for the music domain information; the MusicMoz [13] hierarchy to classify music genres; the RDF Site Summary [14] for music news; and the Friend Of A Friend (Foaf) ontology [15], a widely used formalism to describe for user's profiles and relationships among them. A fragment of the MusicBrainz ontology is reported in Figure 1. This application is similar to other existing Semantic Web applications (e.g., [9]), that provide personalized access to the contents exploiting distributed semantic information. The presented application, although rather simple because of space reasons, can be considered a Semantic Web Portal since it aggregates different sources of information spanned across the Intenet. In Section 6 we will show how to model such application with the extended version of WebML.

4 WebML: An Overview

Our approach to Semantic Portals specification is based on the WebML language. WebML (Web Modeling Language) is a methodology for Web application design that comprises the definition of a development process and of a modeling language (composed by several metamodels for describing orthogonal aspects of a Web application) [16]. WebML offers a set of visual primitives for defining conceptual schemas that represent the organization of the application contents and of the hypertext interface. The WebML primitives are also provided with an XML-based textual representation, which allows specifying additional detailed properties, not conveniently expressible in the visual notation.

For specifying the data underlying the application, WebML exploits the Entity-Relationship model, which consists of *entities* (classes of data elements), and *relationships* (semantic connections between entities).

WebML also allows designers to describe hypertexts, called *site views*, for publishing and managing content. A site view is a piece of hypertext, which can be browsed by a particular class of users. Multiple site views can be defined for the same application. Site views are then composed of *pages*, and they in turn include containers of elementary pieces of content, called *content unit*, typically publishing data retrieved from the database, whose schema is expressed through the E-R model. WebML offers a set of predefined units for extracting data from the datasource, submitting data to the application, and specifying the navigation behaviors. Finally, WebML models the execution of arbitrary business actions, by means of operation units. An *operation unit* can be linked to other operations or content units. WebML incorporates some predefined operations for creating, modifying and deleting the instances of entities and relationships, and allows designers to extend this set with their own operations.

Units may be connected through *links*, represented as oriented arcs between source and destination units. The aim of links is twofold: defining the navigation (possibly displaying a new page or a piece of content in the same page) and the data parameters passing from the source to the destination unit.

WebML-based development is supported by the WebRatio CASE tool [17], which offers a visual environment for designing the WebML conceptual schemas, storing them in XML format, and automatically generates the running code (through XSLT model transformations), which is deployed as pure J2EE code.

5 Extending WebML Towards Semantic Web Portals

This section discusses the extensions to the WebML metamodels that are needed for complying with the new requirements of semantic web applications, according to the specifications of Section 2. The extensions apply to any aspect of the WebML methodology:

– **Development process:** extensions to describe the tasks related to the design of ontologies and semantic aspects of the web applications/services;
– **Data model:** extensions to support semantic data sources (i.e., ontologies);
– **Hypertext model:** extensions to support querying on ontologies, with particular attention to advanced and inferencing queries;
– **Presentation model:** extensions to support semantic annotations of the applications.

5.1 Extensions to the Development Process

The methodology adopted in the development of "traditional" Web applications needs to be extended with additional tasks that formalize the new design steps required by the injection of semantics within Web applications. Figure 2 depicts

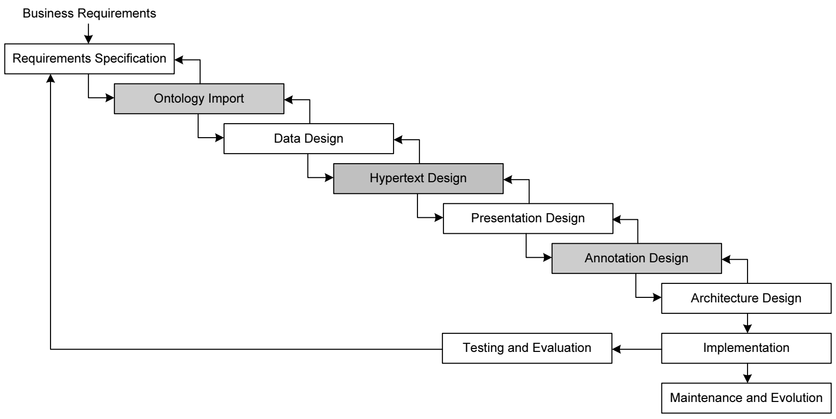


Fig. 2. The extended development process for Semantic Web applications

the extended version of the development process for Web applications. The original version was proposed in [16] and is adopted with slight variations by most of the existing Web Engineering approaches. The gray blocks represent the new tasks we introduced to fulfill Semantic Web application requirements:

- **Ontology Import.** This task addresses the selection and the importing of existing domain ontologies that may be exploited for the domain of the Web application under development. The imported ontologies can be possibly modified or merged to better suite the Web application purposes.
- **Hypertext Design.** This step, which already existed in the original approach, needs to be extended to specify how to query ontologies by means of proper primitives. Notice that the design of interactions with relational data remains unchanged.
- **Design Annotation.** In this phase the Web engineer specifies how the hypertext pages will be annotated using existing ontological knowledge. This step enriches the Hypertext Design and relies on the Presentation Design for deciding the actual position and display style of annotations.

Notice that the waterfall representation may be adjusted for some design experiences, considering that in some cases some steps are not needed at all (e.g., if only imported ontologies are necessary, the data design step can be skipped). We did not depict all the variants for sake of clarity.

5.2 Extensions to the Data Model

The existing metamodels for Semantic Web applications either evolved from existing ones by extending their data source coverage to ontologies , or have born with native support of semantic data sources.

Although ontology support is obviously necessary for semantic Web application design, we think that relational data sources can still provide great added value to Web applications, since the representation of every piece of information as a semantic concept is not realistic in short/medium terms. Indeed, relational databases are still effective to describe substantial parts of web applications. Therefore, allowing seamless interaction between ontologies and databases is a desiderata of current Semantic Web applications (see Section 2 too). Notice that now we do not aim at extending the data model of WebML so as to model ontologies (see [18]), but at allowing Web Portals to query imported semantic knowledge together with relational sources. By adopting a model-based paradigm like WebML, interaction between ontology instances and database instances can be quite straightforward and will be exploited within the hypertext model.

5.3 Extensions to the Hypertext Model

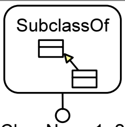
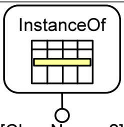
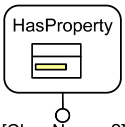
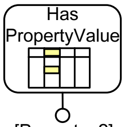
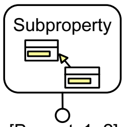
Ontological Queries Within WebML Hypertext Primitives. The main asset of WebML is the ability of effectively capturing the parameters passed between the different components (units) of the Web application. We believe that this added value is still valid even when we are considering Semantic Web Applications. Indeed, no matter if the underlying data are ontologies or databases, the basic hypertext primitives (units) and the parameter passing mechanism remain the same. Thus, parameters on the links become the actual contact point between traditional and semantic data and provide the mechanism for orthogonalizing data issues and hypertext issues.

The basic WebML primitives for data access (e.g., `Index` unit, `Multidata` unit, the `Data` unit) have a general purpose meaning and are perfectly fitting in the role of query and navigation primitives for both relational and ontology sources. They only need a few simple extensions for supporting the additional expressive power and the different data model of the ontological sources.

Let us consider the `Index` unit: besides extracting lists of relational instances, we want to use it to produce lists of instances of a particular class within an ontology model. Some requirements were highlighted about this kind of queries: *(i)* the possibility to show only direct instances or also inferred instances; *(ii)* the need for querying both instances and classes, thus mixing instances and classes in the results too. The same discussion applies to `Multidata` unit and `Data` unit. Another primitive already introduced by WebML is the `Hierarchical Index` unit. This unit acquires a first class role in the context of Semantic Web applications, because it provides a mechanism to browse and publish a portion of an ontology in a hierarchical tree representation; for instance, given a class, it allows to publish the hierarchical tree underlying it, comprising subclasses and instances.

Although the basic primitives remain valid, they require some extensions to cover the new kind of datasources. Indeed, queries on ontological data require a much wider expressive power and some different modeling rules for the information with respect to relational data. This affects the notations that the primitives must use for defining the conditions and the selection of the data. Typical examples of new features and queries of the ontological models are:

Table 1. Summary of new WebML inference units

Name	Symbol	Input	Output
subClassOf	 [ClassName1=?] [ClassName2=?]	c_1, c_2	**true** if c_1 is subclass of the class c_2
		$c_1, ?$	the list of superclasses of the class c_1
		$?, c_2$	the list of subclasses of the class c_2
instanceOf	 [ClassName=?] [Instance=?]	i, c	**true** if i is an instance of the class c
		$i, ?$	the list of classes to which the instance i belongs
		$?, c$	the list of instances of the class c
hasProperty	 [ClassName=?] [Property=?]	c, p	**true** if the class c has the property p
		$c, ?$	the list of properties of the class c
		$?, p$	the list of classes having the property p
hasPropertyValue	 [Property=?] [Value=?]	p, v	the list of URIs where property p has value v
		$p, ?$	the list of possible values for the property p
		$?, v$	the list of properties with value v
subPropertyOf	 [Property1=?] [Property2=?]	p_1, p_2	**true** if the property p_1 is subproperty of p_2
		$p_1, ?$	the list of superproperties of the property p_1
		$?, p_2$	the list of subproperties of the property p_2

– there is *no distinction between relationships and attributes* within the set of properties of a class. E-R style relationships might be considered as ontological properties having an *URI* as value, and attributes to ontological properties having a literal as value.
– several Semantic Web framework (e.g., OWL, RDF) assume that any instance of a class may have an arbitrary number (zero or more) of values for a particular property.
– Cardinality constraints and classes can be defined when specifying properties. In this case, it is possible to publish as values also structured objects and not only atomic attributes.

In general, while in a E-R model the selection of attributes to be published is straightforward, in the ontology model some navigation over the model may be required to publish the data. The extensions to the existing WebML primitives

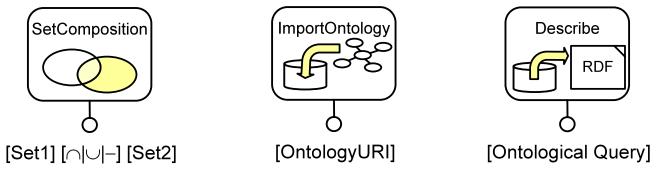


Fig. 3. Symbols of the new WebML semantics management units

provide these advantages. By means of the extended WebML primitives we can express visual queries over an ontology.

Advanced Data Access Primitives. The evolution of the basic data access primitives, introduced in the previous paragraph, is still not enough to exploit the rich set of possible queries over semantic instances and classes. Therefore, we introduce a new set of operational primitives to describe advanced queries over ontological data. We introduced these new units largely inspired by SPARQL [19] and RDF Schema syntax [11].

A first set of new units allow advanced ontological queries. The units are aggregated primitives that, depending on the type of parameters, execute differently. The complete summary of the behavior of these units is available in Table 1. 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.

Besides the units for ontological data query, we introduce also three new units depicted in Figure 3. The `Set Composition` operation unit, is able to perform classic set operations (i.e., union, intersection, difference) over two input sets of URIs, considering the hierarchy of the URIs involved. E.g. suppose we have two set of classes: $A = \{ProgressiveRock, Jazz, Metal\}$ and $B = \{Rock, JazzFusion\}$ In this case, the set operation will give the following results: $A \cap B = \{ProgressiveRock, Metal, JazzFusion\}$ and $A \cup B = \{Rock, Jazz\}$ since *Rock* is superclass of *ProgressiveRock*, and *Jazz* is superclass of *JazzFusion*.

The `Import Ontology` unit imports at run time an ontological data source that must be consistent with one or more of the ontology models used at design time of the web application (it's validated against them before being added): according to the designer choice, it is possible to store only the url of the newly imported ontology (i.e., it will be accessed remotely for each query) or to import the ontology in the local OWL/RDF repository (i.e., it will be accessed locally, but modifications to the original data will not be propagated to the application). Notice that the navigational model of the Web application does not change at runtime, thus if the imported ontology contains new pieces (e.g., a new class unrelated with already existing classes) that were not considered in the hypertext, these pieces of knowledge will not be reachable.

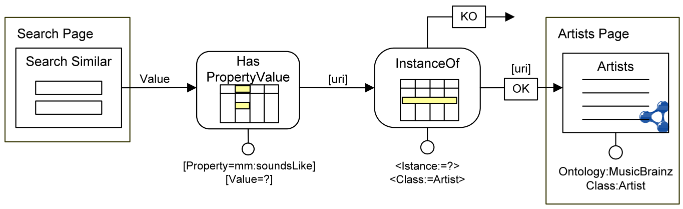


Fig. 4. A piece of Semantic Web application described by the new WebML units

The `Describe` unit returns the RDF description of an 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 find the set of common properties between two classes (e.g., Track and Song), first we can use two `HasProperty` units to extract the two set of properties characterizing the two classes; and finally we can find the common set of properties by means of the `SetComposition` unit, that will be in charge of calculating the intersection between the two sets.

Figure 4 reports a fragment of the portal that allows to retrieve artists or albums whose names sound in a similar way to the name specified by the user. The `value` submitted in the form 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. The set of URIs is then passed to the `InstanceOf` unit that checks if they are instances of the class Artist. In this case, the URIs are passed over 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).

6 Modeling the Semantic Web Portal for Music Domain

Thanks to the extensions introduced so far, we are now able to model a Semantic Web Portal scenario like the one proposed in Section 3 for the music domain. Figure 5 reports a fragment of the WebML model (including the semantic extensions) for that application. The publication units with the RDF symbol () rely on ontological data sources (e.g., *Artists* index unit), while the other units publish data from the a relational database (e.g., *User Data* data unit). The integration between the two kinds of content happen at the parameter level: once the results are transferred as parameters through links, they become homogeneous pieces of contents. The user starts his navigation from the *User Home Page*, where he can find his *Foaf Profile*; he can import a profile if it is not available yet. This part of the application actually shows how integration between ontological data sources and relational data can be achieved using parameters transported over links: when the user imports his Foaf profile, he actually stores the uri of the profile in the *User* relational entity; this uri is later used to publish his Foaf profile from the ontology repository according to the database schema.

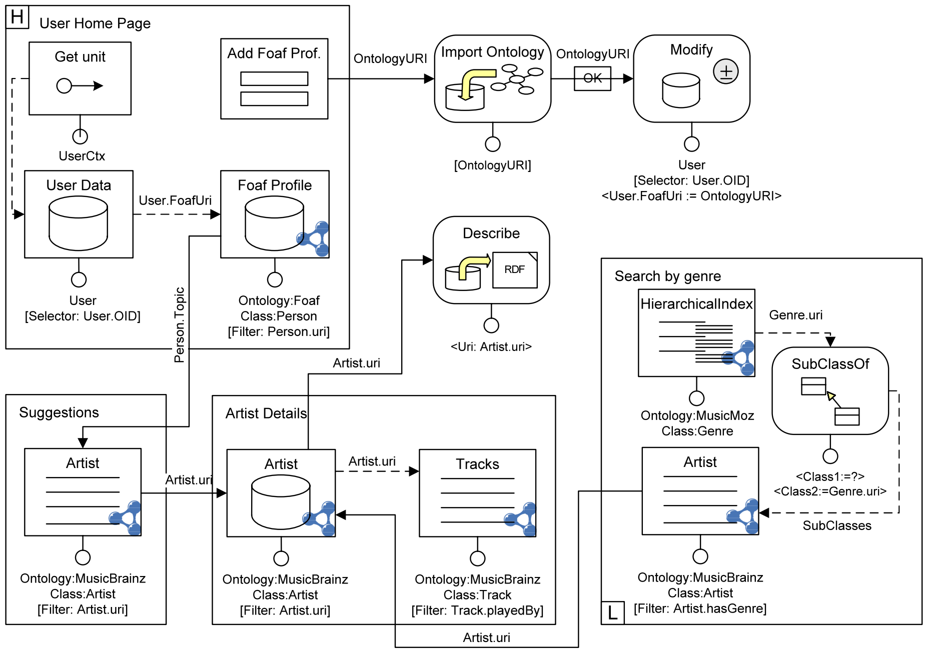


Fig. 5. A portion of a WebML diagram for a Semantic Music Portal

Navigating the outgoing link from *Foaf Profile*, the user can access the *Suggestion* page showing an index of *Artists* corresponding to his preferences. From here the user can navigate to the *Artist details* page, where detailed informations about the selected *Artist* and his *Tracks* are presented. The user can ask for the exporting of the RDF description of the artist he is currently browsing. Finally, the *Search by genre* page provides a hierarchical representation of the class *Genre*, and then displays all the artists that are related to the selected genre. The `SubClassOf` unit extracts indirect sub-genres of the chosen one, thus allowing to display associated artists.

7 Implementation and Architectural Issues

This section discusses the implementation and architectural issues related to the proposed extensions to the WebML metamodel. These issues are discussed with respect to the reference implementation of the Webratio toolsuite. For our prototype implementation we adopted the Jena framework [20] to interact with OWL/RDF ontologies. We provided reasoning support by means of the Jena integrated reasoner, and by means of the integration of Pellet [21] with the Jena framework. The design environment offered by Webratio has been extended exploiting the plug-in mechanism of the toolsuite: we devised a general purpose

```
<SWINDEXUNIT class="mf:Track" id="swinu1"
 name="Tracks" ontology="onto1">
  <DisplayedProperties
   property="mf:title"/>
  <DisplayedProperties
   property="mf:descriptor"/>
  <SortProperties order="ascending"
   property="mf:title"/>
  <Filter boolean="or">
    <FilterCondition id="fselector1"
     property="mf:playedBy"
     predicate="eq" name="Artist"/>
  </Filter>
</SWINDEXUNIT>
```

```
<descriptor service="org.webml.onto.
 SWIndexUnitService">
  <onto>onto1</onto>  ...
  <input-params>
    <input-param type="mf:Artist"
     name="swdau3.Artist" />
  </input-params>
  <query type="SELECT">
   DISTINCT ?instance ?p1 ?p2
   WHERE {?instance rdf:type mf:Track .
          ?instance mf:title ?p1 .
          ?instance mf:descriptor ?p2 .
          ?instance mf:playedBy ?fs1 .
          FILTER (?fs1 = $swdau2.Artist$)}
   ORDER BY DESC(?p1)
  </query>
</descriptor>
```

Fig. 6. Design time (left) and runtime (right) descriptors for a semantic index unit

ontology data access layer to be exploited by every unit; moreover, we developed a runtime Java component and an XML descriptor for each unit.

Ontological Units implementation Each unit is implemented by means of a generic class representing the runtime component that is executed for every instance of that kind of unit. Then, for each new unit (including the revisited traditional units that access ontologies) we developed an XML descriptor specifying its parameters, its properties, and the binding to the implementation classes, and so on. To better clarify the structure of the descriptor, we show an example of an ontological index unit descriptor (see left part of Figure 6). By mean of an associated XSLT transformation, design time descriptors are translated to runtime descriptors that include automatically generated template of SPARQL queries (right part of Figure 6). Units are implemented by Java components that behave according to the logics specified in the runtime descriptors, defined for each instance of the unit.

Ontology Data Access Layer. 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 7. 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 perform selection over data (SPARQL `SELECT` query); the `AbstractDescribeQueryService` retrieves the RDF describing a given URI (`DESCRIBE` query), the `AbstractAsk-QueryService` verifies simple predicates (`ASK` query). The `AbstractAskQuery-Service` 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). In general, unit services use or implement these services.

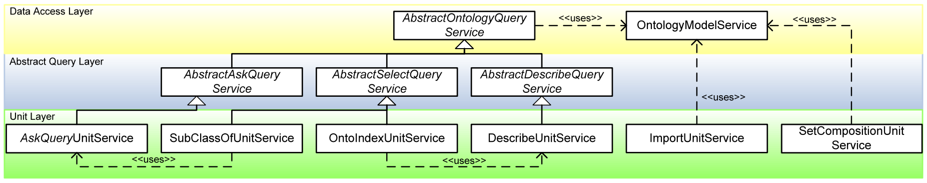


Fig. 7. A UML class diagram that shows part the of class hierarchy of the new implemented units

8 Related Works

While design methodologies for traditional Web applications offer rather mature and established solutions, Semantic Web application methodologies are still in a development phase. Realizing the benefits of the Semantic Web platform (e.g., interoperability, inference capabilities, in-creased reuse of the design artifacts, etc.) traditional design methodologies are now focusing on designing Semantic Web applications: e.g., OOHD evolved in SHDM [5]. New methodologies like XWMF [3], OntoWebber [4] and Hera [6] were specifically designed by considering the Semantic Web peculiarities.

Table 2 reports a summary that compares the features of the previously cited models for Semantic Web Portals and the WebML extensions presented in this paper. All the models, except XWMF, have a complete development methodology that covers all the needed aspects to create a Semantic Web applications. They also offer a wide support for ontology languages: basically all the models support both RDF and OWL (except for XWMF).

However, our extension is the only one that leverages on Semantic Web query languages to offer advanced query primitives that allows both query on schema and instances, together with simple reasoning patterns over data. The others models, in some cases (e.g., Hera) offers query on data schema and instances. Hera and OntoWebber offer direct to support to integration by mean of an integration model that can be used to query different data schema using the same query, while our proposal offers only a basic integration of different data sources thanks to the parameter flow between the different units in the hypertext. WebML offers the chance to integrate relational, XML and ontology data sources, while other methodologies seems to support explicitly only ontologies (off course, this issue can be solved adopting extraction techniques to import other data sources within ontologies). SHDM does not allow to import ontologies but only to create them from UML diagrams. Then, it offers a tricky way to link these ontologies to the external ones. Even if all the analyzed models are based on an ontology representation, only our proposal and a WSDM extension [22] provide an approach to annotate pages so as to make them machine readable.

Most of the new methodologies offer runtime frameworks that include or allow integration of reasoners, while some of them do not clarify if the reasoning is supported also at design time. An important factor to assure the success of a Web information System design methodology is the existence of CASE tool support, since

Table 2. Comparison of methodologies for modeling Semantic Web Portals

Requirement	XWMF	OntoWebber	SHDM	Hera	WebML+Sem.
Methodology	Partial	Yes	Yes	Yes	Yes
Semantic Model Description	Yes	Yes	Yes	Yes	Partial
Advanced query support	No	Partial	Partial	Partial	Yes
Flexible integration	No	Partial	Yes	Yes	Partial
Heterogeneous data sources	No	No	Partial	Partial	Yes
Distributed data sources	No	No	No	Yes	Yes
Reuse of ontologies	Yes	Yes	Partial	Yes	Yes
(Automatic) Annotation	No	No	No	No	Yes
Reasoning Support	No	No	Yes	Yes	Yes

a powerful methodology that is not accompanied by adequate tools will make the designer tasks very difficult to fulfill. While most of the traditional design methodologies have powerful CASE tools, no established tool support is provided for Semantic Web design, although all the cited methodologies offer some basic tools. Among them, the most complete are Hera and SHDM. Our methodology is completely supported by a commercial tool, Webratio [17] that we extended with the new components to enable design of Semantic Web applications.

9 Conclusions

In this paper we presented an extension to the WebML methodology and models for supporting the design and the specification of Semantic Web applications. The described solution provides a full coverage of the development process, and allow the designer to specify 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. Our approach provide substantial added value with respect to the existing frameworks for Semantic Web application design, although some of them offer more advanced solutions on some aspects (e.g., seamless integration of different ontologies). We support our proposal with a prototype implementation within the CASE tool WebRatio. Finally we showed how the proposal can be adopted to develop a Semantic Web Portal for the musical domain reusing existing knowledge.

Future work includes providing a integration layer to allow for seamless integration of different ontologies, extended testing of the new framework and integration of existing Eclipse based solutions for ontology editing with in the CASE tool.

Acknowledgments

We would like to thank Emanuele Della Valle and Irene Celino for the useful discussions on the engineering of Semantic Web Portals and their support in the definition of the case study.

References

1. Brambilla, M., Celino, I., Ceri, S., Cerizza, D., Della Valle, E., Facca, F.M.: A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (eds.) ISWC 2006. LNCS, vol. 4273, Springer, Heidelberg (2006)
2. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer-Verlag, New York, Inc., Secaucus, NJ, USA (2006)
3. Klapsing, R., Neumann, G., Conen, W.: Semantics in Web Engineering: Applying the Resource Description Framework. IEEE MultiMedia 8(2), 62–68 (2001)
4. Jin, Y., Decker, S., Wiederhold, G.: OntoWebber: Model-Driven Ontology-Based Web Site Management. In: Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L. (eds.) The first Semantic Web Working Symposium. Proceedings of SWWS'01, Stanford University, California, USA, July 30 - August 1 (2001), pp. 529–547 (2001)
5. Lima, F., Schwabe, D.: Application Modeling for the Semantic Web. In: 1st Latin American Web Congress (LA-WEB 2003), Empowering Our Web, Sanitago, Chile, 10-12 November 2003, pp. 93–102. IEEE Computer Society, Los Alamitos (2003)
6. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Eng. 2(1-2), 3–26 (2003)
7. Maedche, A., Staab, S., Stojanovic, N., Studer, R., Sure, Y.: Semantic portal - the seal approach. In: Fensel, D., Hendler, J., Lieberman, H., W.W. (eds.) Spinning the Semantic Web, pp. 317–359. MIT Press, Cambridge, MA (2003)
8. MIND LAB: Mindswap - Maryland Information and Network Dynamics Lab Semantic Web Agents Project (2007) `http://www.mindswap.org`
9. Music Technology Group, Universitat Pompeu Fabra: Foafing the music (2007) `http://foafing-the-music.iua.upf.edu`
10. AIFB, University of Karlsruhe: ontoworld.org (2007) `http://ontoworld.org`
11. W3C: Rdf vocabulary description language 1.0: Rdf schema (2007) `http://www.w3.org/TR/rdf-sparql-query`
12. MusicBrainz: Musicbrainz project (2007) `http://musicbrainz.org`
13. MusicMoz: Musicmoz - open music project (2007) `http://musicmoz.org/`
14. RSS-DEV Working Group: Rdf site summary (rss) 1.0 (2000) `http://web.resource.org/rss/1.0/`
15. Miller, L., Brickley, D.: Foaf project (2007) `http://www.foaf-project.org`
16. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kauffmann, Seattle, Washington, USA (2002)
17. WebModels s.r.l.: Webratio tool. (2007) `http://www.webratio.com`
18. Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A.: Creating Semantic Web contents with protege-2000. IEEE Intelligent Systems 16(2), 60–71 (2001)
19. W3C: Sparql query language for rdf (2007) `http://www.w3.org/TR/rdf-sparql-query`

20. Jena Team: Jena a semantic web framework for java (2007) `http://jena.sourceforge.net`
21. Parsia, B., Sirin, E., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: An owl dl reasoner (Technical report)
22. Casteleyn, S., Plessers, P., Troyer, O.D.: Generating semantic annotations during the web design process. In: ICWE '06. Proceedings of the 6th international conference on Web engineering, New York, NY, USA, pp. 91–92. ACM Press, New York (2006)