Modular, Best-Practice Solutions for a Semantic Web-Based Digital Library Application

Andrew Russell Green and José Antonio Villarreal Martínez

Instituto Mora (National Council for Science and Technology, Mexico)
and Instituto de Investigaciones Estéticas (National Autonomous University of Mexico)
ahg@servidor.unam.mx,quetzal1910@gmail.com

Abstract. Digital libraries and archives stand to benefit greatly from the Semantic Web, which may provide a basis for novel end-user functions targeted at research and teaching. The project "Image Preservation, Information Systems, Access and Research" seeks to develop an adaptable digital library application based on a back-end of semantically modeled data. By "adaptable" we mean able to adapt to diverse library and archive scenarios, especially involving the integration of different types of material (photographic prints, negatives, drawings, periodicals, books, etc.) in a single system. This requires "mixing and matching" standard and non-standard catalogue record formats and ontologies to model them. A central problem we have encountered is: how to structure application logic in this context in a way that follows best-practice principles. In this paper we discuss the problem and propose, as a tentative solution, a Semantic Component Architecture, which would provide an integrated, encapsulating way of selecting vocabularies and establishing inference rules, recurrent path patterns, graph constraints, catalogue record templates and arbitrary logic. We also review the related issue of encapsulation of low-level graph structures.

Key words: Semantic Web application architecture, ontology modularization, path definition languages, digital libraries

1 Introduction

Digital libraries and archives stand to benefit greatly from the Semantic Web (SW). Semantically modeled catalogues should provide a basis for new functions to help users sift through large and diverse repositories, discover patterns, explore associations among objects, find relevant information, and create and share descriptions of objects in a structured, flexible manner. This is the promise the SW holds for knowledge repositories, and one can hardly underestimate its potential impact in History and other Social Sciences: archives are primary sources—essential deposits of partially processed information, used for research in these disciplines—and despite the high degree of interrelation among data in different archives, catalogues are often isolated and employ divergent record formats that are hard to align using standard information technology.[1]

This issue is one of the reasons the project Image Preservation, Information Systems, Access and Research (IPISAR) set out to build a SW-based digital library application. The project investigates the dissemination, study and management of heritage resources, and attempts to provide solutions to common problems in these areas.

The application being built, called "Pescador", will store catalogue data in a persistent triple store (whose function will be similar to that of a relational database in traditional systems). The requirements for the application include the ability to integrate data in various catalogue formats and adapt to the cataloguing needs of diverse archives. In this paper, the terms "catalogue format" and "record format" refer to the selection, organization and meaning of fields used to describe objects in an archive or library catalogue, as well as other conventions related to catalogue creation. Since Pescador will use the SW to model catalogues, each record format will correspond to a distinct kind of graph structure, often requiring specialized vocabulary and rules, and related to specialized application logic.

The application will have three main types of users: (1) regular users (or "patrons") who will consult the material provided by the digital library, (2) cataloguers, who will provide and manage the library's materials and metadata, and (3) catalogue designers/modelers/programmers, who will select or create the catalogue record formats and corresponding ontologies, and adapt the system to the needs of a given scenario. Pescador will provide a Web interface for the first two kinds of users; on this level, numerous functions targeted at research, teaching and cataloguing are planned [6]. When these users view data from the catalogue, they will see a user-friendly organization of information extracted from the SW graph; similarly, when cataloguers modify elements in the catalogue, they will employ an easy-to-use form, and the SW graph will be changed according to their input.

The third type of user, the catalogue designer/modeler/programmer, will use a programming interface. The problem considered in this paper is: how to design an interface that allows users of this type to adapt the application to their needs, in a way that follows best-practice information engineering principles such as the encapsulation of complexity, the separation of concerns and the non-repetition of declarations. To achieve this we propose a Semantic Component Architecture (SCA), that is, an adaptation of component architecture principles to a SW application context, in which data structure rules and application logic are closely linked. In addition, we propose a mechanism for encapsulating low-level graph structures, which should also help catalogue designers/modelers/programmers follow best-practice principles.

[1] The situation of historical archives varies greatly from one archive to another. Other recurring difficulties include access restrictions and insufficient funding; the first of these is also a major focus of the project described in this article. See [7] and [2].

To date, two incomplete versions Pescador have been created. Both are currently used for Web sites that offer simple consultation functions for on-line archives (available at [9] and [4]). Our proposals stem from the experience of developing these versions of the application. Though the project IPISAR may yet generate new archival Web sites using the second version, it is clear that to implement all proposed features, a major rewrite is unavoidable. It should be noted that work on the rewrite and on the detailed design and implementation of the SCA has not yet begun. The general nature of the proposals outlined here is a reflection of this.

All versions of Pescador are provided under the terms of the free GNU GPL license.

2 Previous related development experiences

Since before work on Pescador began, it was clear that to integrate data in various formats and adapt to diverse scenarios, the system would have to offer ways of selecting vocabulary, establishing graph constraints[2] and configuring the algorithms used to transform information as it passed between the model and the UI. In retrospect, we can say that we have consistently underestimated the complexity of the mechanisms required for these and related processes. The first version of Pescador (version 0.1) allowed catalogue designers/modelers/programmers to select RDF schemas and set presentation information, which was modeled using an augmented version of the Fresnel Display Vocabulary [3]. In the subsequent version (0.2), we abandoned Fresnel due to its limited scope (it was designed only for specifying how to display SW data) and text manipulation features, and instead created a domain-specific language (DSL) that provided an integrated way of establishing vocabulary, graph constraints, inference rules, path definitions and display specifications. Our DSL also allowed the creation of executable code fragments that could be hooked into various parts of the system. Though never implemented in full, the DSL was, we believe, an important step in the right direction, as it recognized the need to take into account best-practice principles when setting out configuration information [5]. However, our version 0.2 design was flawed precisely because it defined a configuration system—it offered a very limited range of possibilities for creating arbitrary logic and encapsulating complexity, as compared to our current proposal, the SCA, which by its very nature calls for opening as widely as possible the catalogue designer/modeler/programmer's options. The difficulties we faced in organizing and re-using code written in our own DSL highlight the need for mechanisms that do not impose the limits that our DSL did.

[2] By graph constraints, in this context and in the rest of the paper, we mean the specifications of how to structure and link subgraphs that describe elements in the catalogue. This includes, but is not limited to, establishing which properties may be used with resources of a which classes, and the properties' allowed cardinality.

3 General conception of an SCA

The SCA we envision would coordinate pluggable "components" or "bundles"[3] that would wrap interrelated elements of any of the following types: schemas, constraints, inference rules, ontologies, path definitions, executable code, display specifications, Abox configuration information, and links to external data sources. As in other component frameworks, bundles would be able to provide hooks of various kinds and use the hooks exposed by other bundles. These hooks would be the bundle's means of controlling and simplifying access to the elements it wraps. A bundle could also "depend on" other bundles, and would thereby know which external hooks are available to it. The standard definition of a component as "a software artifact consisting of three parts: a service interface, a client interface and an implementation"[11] might, with little or no modification, provide adequate theoretical grounding for an SCA—though we would have to ask how a SW application context would modify our understanding of what constitutes an interface and what constitutes an implementation.

4 Justification

An SCA would help solve issues of encapsulation and re-usability. Consider, for example, an archive that includes books, photographic prints, negatives and handwritten manuscripts. Catalogue record formats for each type of object would certainly vary, though they would also have elements in common. For example, all objects would probably be associated with an author and a creation date. But only the negatives, photographic prints and books could be associated in a chain of image reproductions (as in, a negative could be reproduced on a print, and the print could be touched up and cropped, then reproduced in a book). Many objects might have a place of creation, but the precise meaning of this concept would not be the same for a book (place of publication) and for a negative (place the photo was taken). Of course, the SW excels in the alignment of data structures like these. However, setting up a digital archive to integrate records of such objects involves much more than establishing data structures and their intersections; bits of logic have to be defined, and it makes sense to define them alongside related data structure information in an encapsulated way—in other words, to "keep together what goes together".

Thus, in an SCA, the logic needed to transform the SW model of a book's metadata into a standard bibliographic citation might be stored in the same bundle as the vocabulary, graph constraints, and remaining display information for books. Similarly, a component could be created for chains of image reproductions (like the one described above). The component could enclose vocabulary, constraints, path definitions, and code for generating visual representations of these chains. Other components—such as those for books and photographs—could depend on it, interfacing with the hooks provided and thus employing in a simplified manner the enclosed complexity.

[3] Herein the terms "bundle" and "component" are used as synonyms.

The advantages of component architectures, and of the modularity and reusability they offer, have long been recognized, and countless frameworks and related technologies are used in applications of all shapes and sizes.[4] Component architectures are increasingly viewed as a programming paradigm in their own right.

Our experience shows that a SW-based system that relies, as Pescador 0.1 did, on partially overlapping RDF data sets coupled with fragments of related logic scattered throughout the application, ends up grossly violating best-practice principles in many respects, to the detriment of flexibility and scalability. An SCA would address these problems in a thorough manner, and provide a basis for the re-use of ontologies coupled with application logic.

5 Path Definitions in the Context of an SCA

In both versions of Pescador, the algorithms that extract information from the graph frequently traverse paths from one resource to another. These paths vary greatly in length and complexity. To allow for encapsulation and separation of concerns as proposed, an SCA must include a means of defining path patterns that supports these principles. (We first ran into path-related encapsulation issues in version 0.1 of Pescador, which used SPARQL to extract information from the graph. In that version, we found ourselves writing SPARQL queries that repeated information already set out in several other parts of the application—quite the opposite of a maintainable application structure.) The solution we suggest is to adapt an existing path definition mechanism (of which there are many [12]) to allow bundles to define paths as aggregates of other paths, which may in turn be defined opaquely in other bundles.

Let us illustrate this proposal with reference to the hypothetical archive described in 4 Justification. Consider: one bundle might define paths from any photographic image to its original negative, along a chain of image reproductions; another might define paths from cities to the continents on which they are located. In such a scenario, it should be possible, in a third bundle, to aggregate paths and define a route from any photographic positive to the continent on which the negative that created the photo was snapped. Fig. 1 presents this example.

A preliminary survey of existing path definition mechanisms indicates that the SPARQLeR extension to SPARQL may be the best candidate for adaptation to our SCA and Pescador. The authors of SPARQLeR, working in the life sciences domain, found they needed a means of detecting "semantic associations"—undirected paths "that connect two entities in the knowledge base using named relationships" [8]—and that the associations they had to detect could not be "explicitly defined by the fully specified structure of a graph pattern". In modeling catalogues and Social Science data, we have noticed a similar requirement, which SPARQLeR has the potential to fulfill.

[4] Examples of frameworks and technologies include OSGi, Enterprise Java Beans, COM and CORBA. See [1] for the history of the concept.

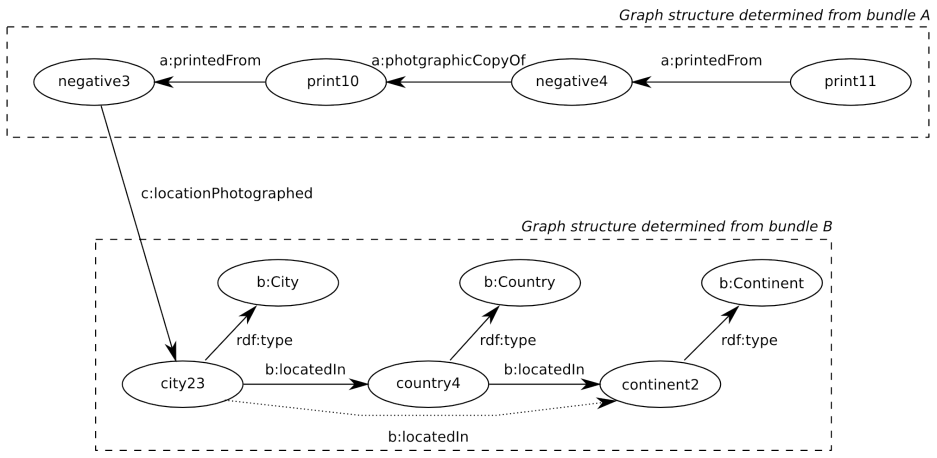

Fig. 1. Encapsulation and Separation of Concerns in Path Definitions

In this example, bundle A defines vocabulary and related logic for chains of image reproductions, bundle B does the same for geographic locations, and bundle C manages photograph catalogue records at a higher level, and depends on bundles A and B. Bundle A might define a path `originalNegative` to go from any photographic print to its original negative, and bundle B might define a path `onContinent` to traverse from any place to the continent on which that place is located. Bundle C could then define an aggregate path that traverses first `originalNegative`, then the property `c:locationPhotographed`, and finally `onContinent`, to go from a photographic print to the continent on which the photo was taken (`print11` to `continent2`). Bundle C would not need to know the inner workings of `originalNegative` and `onContinent`, and in the event that the specifics of the graph structures determined by bundles A or B were to change, the required changes to path definitions would be limited to one bundle.

6 Low-Level Encapsulation: SW Abstraction Layer

This section describes a mechanism called "the SW graph abstraction layer", which seeks to apply, in low-level graph operations, the same principles that underlie the SCA. The abstraction layer will offer simplified access to and correct processing of very basic graph structures that occur repeatedly in a wide variety of models that we have worked with. It will be integrated directly with the Pescador's triple store; graph extraction and modification operations will access it in a transparent manner. The abstraction layer will be logically above a SW-conformant graph. Both the abstraction layer and the SW-conformant graph will represent exactly the same information; each will simply provide a different view of that information. For interchange with other SW applications, the Pescador will be able to export the SW-conformant view of the contents of its triple store in standard SW serialization formats (like RDF/XML).

Below are some examples of simple graph structures as seen by the abstraction layer and as reflected in the underlying SW specifications-compliant graph. The first two examples are presented in detail and the rest are summarized.

Note that version 0.2 of Pescador already implements many of the features described here. Unlike the SCA, the abstraction layer has demonstrated its usefulness in a functioning implementation, and can be considered a relatively mature proposal. Nonetheless, substantial work remains to be done to develop a theoretical grounding for this solution, analyze its implications in formal terms, and design a new implementation.

Multilingual Values A common idiom in SW graphs is the multilingual value: conceptually a single value with alternate versions for different languages. It is often modeled with a blank node that is an instance of `rdf:Alt`. The problem we have encountered is the recurring need to select an appropriate version of a multilingual value on the basis of information available at runtime. Fig. 2 shows the abstraction layer's view of this idiom, together with the underlying standards-compliant model.

This type of structure is created only when specifically requested for a given property and a given subject resource. The query system takes the abstraction layer view of the model; queries are executed in a given language context, and only one of the "switchable" triples is available during a given operation. This makes for cleaner path definitions, which need only specify the main property to be traversed (in this example, `sys:name`). Without this feature, the sub-pattern for selecting the correct language alternative would have to be repeated in many path definitions.

Ordered Multiple Values In catalogue records, it is often necessary to present several values of the same field in a specific order—for examples, the authors of a publication must be listed in the same order in which they appear on the publication itself. Similarly, if several topics from a thesaurus are associated with a given entity, the most relevant topics are listed first. These

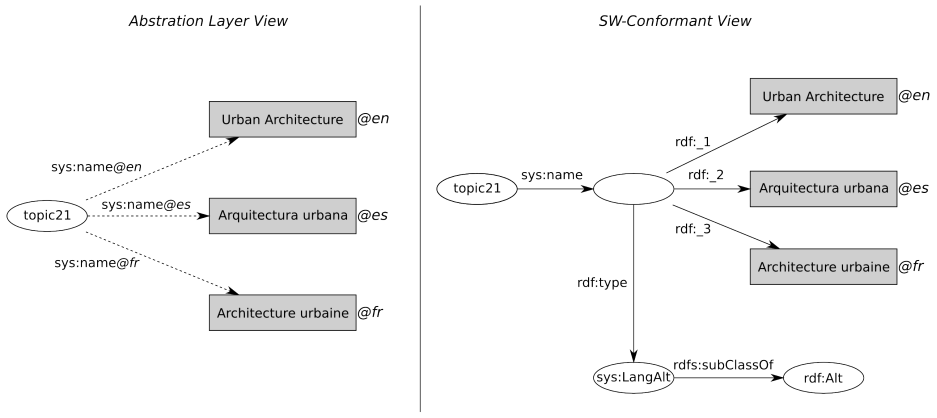

Fig. 2. Multilingual Values
In the abstraction layer view, the arcs shown with a dotted line are seen by extraction operations as "switchable" versions of a single triple. Queries are executed in a given language context, and only one arc may be "switched on" during a given operation.

fixed orders must be reflected in the model. Since in RDF multiple, identical properties on a single subject are unordered, when a fixed order is needed, multiple values may be grouped using an `rdf:Seq`. Fig. 3 shows how the abstraction layer understands this low-level structure.
As in the case of multilingual values, this structure is only created when it is requested for a given property and a given subject. The query system traverses from the main subject (`photo2`) directly to the objects proper (`topic21`, `topic42` and `topic25`) and extracts them in the correct order. Again, no special sub-pattern need be repeatedly included in the path definitions that traverse this type of structure.

Missing Value Explanations It is frequently useful to include in a catalogue an explanation for a value's absence (common reasons are "unknown" or "field does not apply"). The abstraction layer provides a special mechanism for modeling such explanations. When a path query does not produce any results, it may expose a pertinent missing value explanation, if one is expressed in the model.

N-ary Relations As explained in [10], there are many ways of modeling n-ary relationships with the SW. The abstraction layer integrates facilities for working with structures that model these relationships, allowing clean path definitions and encapsulating code that deals with corresponding low-level subgraphs.

Structured Values The RDF specification establishes an idiom called a "structured value", which consists of a blank node that is at once the object of a property and the subject of additional triples used to model the aforementioned property's value. The abstraction layer builds on this concept by

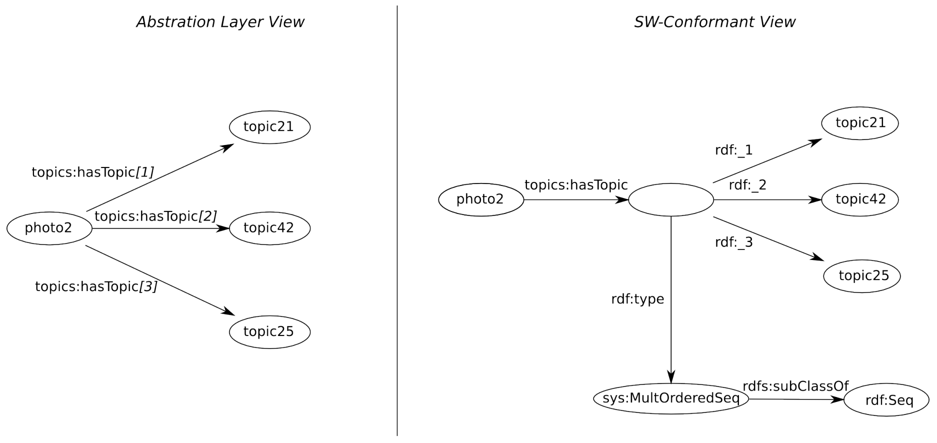

Fig. 3. Ordered Multiple Values
In the abstraction layer view, only the main property appears (`topics:hasTopic`), and queries automatically extract the object resources in the correct order.

offering a mechanism for explicitly identifying nodes that are structured values, thus allowing the implementation of special features for constructs of this type.

7 Other Related Issues

Among the numerous issues facing Pescador development, several relate to the SCA and the system's adaptability, including:

Model generation, concurrency and validation To develop a Pescador Web interface for creating, deleting and modifying catalogue items requires the study of algorithms for changing the model in a multi-user environment. Input validation logic and graph constraint checking must also be considered and integrated with the SCA. Our intention to offer an "undo" mechanism further complicates the matter.

Inference rules In many situations it is desirable to establish custom inference rules for a model. Yet again, the details of how to do this must be established, and whatever solution we choose must integrate with the SCA.

Networking A proposed feature of the system is to allow meta-searches in the repositories of multiple Pescador Web servers linked over a peer-to-peer network. To make this a reality we must study, among other things, model alignment and the sharing of SCA bundles over a network.

8 Conclusion

In this paper we have considered some issues encountered in the development of an adaptable SW-based digital library application. Specifically, the issues discussed relate to modularization, the structuring of application logic and a programming interface for catalogue designers/modelers/programmers that allows the use of best-practice information engineering principles. We have also outlined solutions to these difficulties. Although the implementation and detailed design of these solutions is far from complete, our proposals as they stand establish research directions and provide starting points for further work towards the creation of the application we envisage.

References

1. Atkinson C., Paech B., Reinhold J., Sander T.: Developing and Applying Component-Based Model-Driven Architectures in KobrA. IEEE: Alamitos, California (2001) `http://ieeexplore.ieee.org/iel5/7549/20561/00950441.pdf`
2. Aguayo, F., Roca, L.: Estudio introductorio. In: Aguayo, F., Roca, L. (eds.): Imágenes e investigación social. Instituto Mora, México (2005) 9-28 `http://durito.nongnu.org/docs/Aguayo_Roca_2.html`
3. Bizer, C., Lee, R., Pietriga, E.: Fresnel Display Vocabulary for RDF: User's Manual. World Wide Web Consortium (2005) `http://www.w3.org/2005/04/fresnel-info/manual-20050726/`
4. Fototeca Digital: Fotógrafos y Editores Franceses en México. Siglo XIX. Instituto Mora and Instituto de Investigaciones Estéticas, National Autónomous University of Mexico (2007) `http://afmt.esteticas.unam.mx`
5. Green, A. R.: Logic and a Little Language for Heritage Resource on the Semantic Web. Poster accompanying a system demonstration, presented at the 4th European Semantic Web Conference (June, 2007) `http://durito.nongnu.org/docs/innsbruck2.pdf`
6. Green, A. R.: Metadatos transformados: Archivos digitales, la Web Semántica y el nuevo paradigma de la catalogación. In: Amador C., P., Robledano A., J., Ruiz F., R. (eds): Quintas Jornadas: Imagen, Cultura y Tecnología. Universidad Carlos III de Madrid: Madrid (2007) 11-22 `http://durito.nongnu.org/docs/metadatos_transformados_green.pdf`
7. Green, A. R.: Rescate de la memoria. Ciencia y Desarrollo (Sept. 2006). Consejo Nacional de Ciencia y Tecnología, Mexico
8. Kochut, K. and Janik, M., SPARQLeR: Extended Sparql for Semantic Association Discovery (2007) `http://www.eswc2007.org/pdf/eswc07-kochut.pdf`
9. Marcas de Fuego de la Biblioteca "José María Lafragua" de la BUAP. Autonomous University of Puebla (2006) `http://www.marcasdefuego.buap.mx/`
10. Noy, N., Rector, A.: Defining n-ary relations on the semantic web. Working Draft for the W3C Semantic Web best practices group (2005)
11. Parrish, A., Dixon, B., Hale, D.: Component-Based Software Engineering: A Broad-Based Model is Needed (1999) `http://www.kiv.zcu.cz/publications/`
12. RDF Path Languages And Templating. In: ESW Wiki. World Wide Web Consortium `http://esw.w3.org/topic/RdfPath`
13. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press: Cambridge, UK (2000)