Dublin Core Application Profiles

Separating Validation from Semantics

A paper submitted to the W3C RDF Validation discussion

July 3, 2013

Karen Coyle and Tom Baker
kcoyle@kcoyle.net tom@tombaker.org

Historical Context

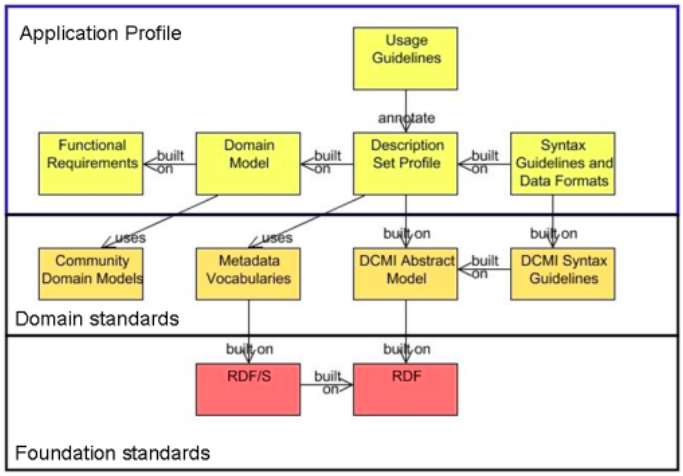
The first Dublin Core workshop (1995) pre-dated the first W3C RDF Working Group by about two years. "Metadata" was a new thing, and much of the discussion in early Dublin Core workshops was about defining a generic model for metadata —"elements" in 1995, "qualifiers" for "schemes" and "sub-elements" in 1997, and so on. Indeed, W3C's RDF effort was initially framed in 1997 in part as a response to requirements that had emerged for Dubin Core metadata, and the Dublin Core Metadata Element Set was in March 2000 one of the very first vocabularies to be posted on the Web as an RDF schema. The community around Dublin Core at the time, however, largely had what we might today characterize as a "validatable record mindset" — the notion of metadata as sets of fields and sub-fields nested in records, essentially as data documents, as in XML or in MARC-format library data.

The Dublin Core community coined the notion of a Dublin Core Application Profile in 2000. [7] The Application Profile mixes and matches elements from multiple vocabularies to meet a specific application need. Part of the Dublin Core community understood this in terms of validatable records, while others saw profiles as syntax-independent expressions of semantics. The push to formalize the model for application profiles from 2000 to 2008 was intended to bridge the gap between the XML and RDF mindsets.

This paper describes the Dublin Core community model that had emerged by about 2008 after which, for lack of widespread uptake (and funding), further development of the model tapered off. The model is presented to the RDF Validation Workshop as the culmination of a discussion of metadata requirements that matured in over a decade and half of discussions in the Dublin Core community.

Singapore Framework for Dublin Core Application Profiles

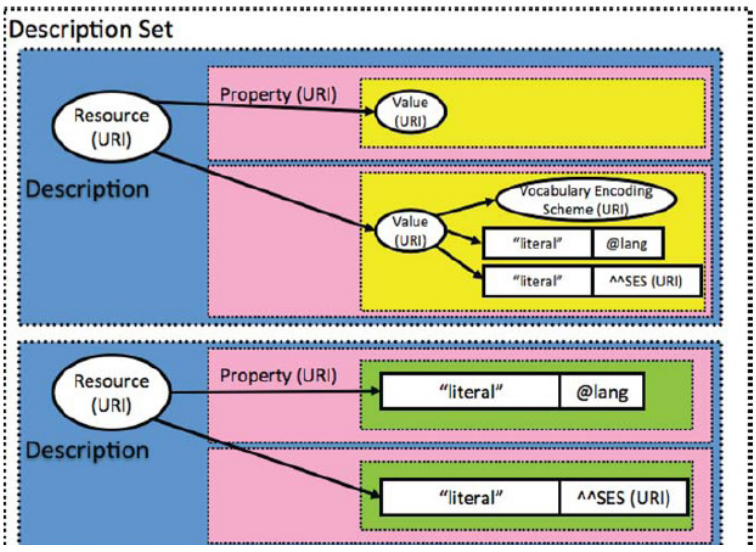
Introduced in 2000, the concept of a Dublin Core Application Profile is based on the idea that vocabularies (also known as "element sets") are *declared* in one schema (e.g., an RDF schema) and merely *reused* in an application profile [6]. The "Singapore Framework for Dublin Core Application Profiles" of 2007 [1], pictured below, builds on the distinction between "declaring" and "reusing." It situates a Description Set Profile (DSP) in the context of models and vocabularies defined outside of the Description Set Profile itself. A DSP is a document specifying the content of metadata records in terms of validatable constraints. A DSP may build on a Domain Model that is a set of classes for things being described (such as Author and Book). A Domain Model should ideally be anchored in the functional requirements for a specific application and may be based on a Community Domain Model such as the Functional Requirements for Bibliographic Records (FRBR), developed by the international library community. [2]



A Description Set Profile *uses* but does not *define* properties and classes from RDF vocabularies. The RDF vocabularies, in turn, are based on the foundational RDF specifications. Application Profiles do not place constraints directly on the RDF vocabularies used, that is, they do not re-define the meaning of RDF vocabularies. Rather, they document constraints on how those vocabularies are used in creating instance data. The same classes and properties can be used, with different usage constraints, in any number of application profiles without effect on their global meaning. Usage constraints in one application profile have no effect on usage constraints in another.

Description Set Profiles

Description Set Profiles are based on the DCMI Abstract Model, also known as DCAM [3], which defines the notions of Description Set, Description, and Statement. A Description is a set of statements about one and only one resource. A Description Set groups the one or more Descriptions that constitute the full set of metadata for some application or purpose, such as a single metadata record that describes both a Book and its Authors.[5] The diagram below shows how the components of RDF graphs translate into components of the DCMI Abstract Model. Description Sets group Descriptions. Descriptions group Statements. Statements group components such as *Resource URIs*, *Property URIs*, *Value URIs, Vocabulary Encoding Scheme URIs* (equivalent to SKOS Concept Scheme URIs), literal values, language tags, and *Syntax Encoding Scheme URIs* (also known as RDF Datatype URIs).



A Description Set Profile specifies constraints on Descriptions, Description Sets, and Statements — constraints that can be used either to create or to validate data. The Description Set Profile Constraint Language [6], a Working Draft of 2008, provides a language for customizing application-specific Description Set Templates. Description Set Templates hold one or more Description Templates for the one or more things intended to be described together in a validatable Description Set instance, such as a metadata record describing a Book together with its Authors. A Description Template may be defined, for example, with the constraint *standalone* (can a record describe just an Author without describing a Book?), *resourceClass* (what class of thing is the description about?), and *minOccurs/maxOccurs* (how many such Descriptions may be contained in a Description Set?).

A Description Template, in turn, holds one or more Statement Templates. A Statement Template may be defined in terms of constraints such as *valueURI* (to mandate use of a particular controlled value list), *language* (to specify the language of literal values), or *propertyList* (to specify the set of properties allowed in a given Statement Template).

The DCAM-based abstract constraint language supports the design of metadata compatible with RDF (e.g., with properties, classes, datatypes), but with additional features not supported (either yet, or at the time) by W3C's RDF Recommendations. DCAM Descriptions and Description Sets are like RDF named graphs. Vocabulary Encoding Schemes are like SKOS concept schemes. The abstract representation of DCAM allows instance metadata to be expressed using any concrete implementation technology that can be mapped to DCAM, such as XML, HTML, or relational databases. The full DCAM family of specifications [8] includes syntax guidelines such as DC-HTML [4], which describes a way to embed DCAM-compatible metadata in HTML using the Meta and Link elements and specifies which features of DCAM are expressible in HTML given the limitations of this approach. (Note that the 2008 DC-HTML specification has effectively been superseded by RDFa.)

Proposal for a Revised Description Set Profile Constraint Language

The DCAM/DSP approach has the following strengths:

1. It specifically separates underlying RDF vocabularies, with their constraints on an ontological "reality," from syntactically validatable constraints on the content of data. Defining constraints in terms of valid data instead of constraints on ontological meaning helps metadata designers avoid overspecifying their ontologies with complex constraints that may be intended to help control data quality in specific applications. Overspecified ontologies can create unwanted entailments and complications when used in new and unanticipated contexts. In other words, by cleanly separating underlying vocabularies from overlaid constraints, Application Profiles are, in principle, easier to create than comparably complex ontologies and their use encourages the design of data that will play well as Linked Data.
2. The notion of specifying the content of metadata in terms of templates and constraints still seems quite useful. In contrast to approaches to RDF validation that focus on SPARQL-based pattern matching on existing data, the DCAM/DSP approach specifies a language usable in the phase of designing and creating metadata. The approach is not at all incompatible with SPARQL-based validation inasmuch as it should be possible to translate Description Set Profiles into SPARQL queries over existing data.
3. The specific set of templates and constraints defined in the DSP language seems like a good starting point for future abstract constaint languages. The DCAM notion of a Description (for a set of statements about one and only one resource) is well-named, as is the Description Set. The language of constraints such as *minOccurs* and *maxOccurs* captures requirements that emerged in many years of discussion and implementation experience in the Dublin Core community.

To be deployable, the constraint language would need to be revised in light of recent RDF developments and subjected to rigorous testing. A revised constraint language for Application Profiles, if widely promoted, and implemented and supported by software tools, could make it easier for people to design good RDF-compatible data.

Application Profiles will be the topic of a special working session at the Dublin Core 2013 meeting in Lisbon, Portugal, September 4. [9]

References

[1] Singapore Framework for Application Profiles, http://dublincore.org/documents/singapore-framework/

[2] Functional Requirements for Bibliographic Records at Wikipedia. http://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records

[3] Dublin Core Abstract Model, http://dublincore.org/documents/abstract-model/

[4] Expressing Dublin Core metadata using HTML/XHTML meta and link elements, http://dublincore.org/documents/dc-html/

[5] Guidelines for Dublin Core Application Profiles, http://dublincore.org/documents/profile-guidelines/

[6] Description Set Profile: A Constraint Language for Dublin Core Application Profiles, http://dublincore.org/documents/dc-dsp/

[7] Application Profiles: Mixing and Matching Metadata Schemas, http://www.ariadne.ac.uk/issue25/app-profiles

[8] DCAM family of specifications, http://wiki.dublincore.org/index.php/Glossary/DCAM_Family_of_Specifications

[9] Application Profiles as an alternative to OWL Ontologies http://dcevents.dublincore.org/IntConf/index/pages/view/APaltOO