Formalizing Dublin Core Application Profiles Description Set Profiles and Graph Constraints

Mikael Nilsson, Alistair J. Miles, Pete Johnston, Fredrik Enoksson

mikael@nilsson.name, A.J.Miles@rl.ac.uk, Pete.Johnston@eduserv.org.uk, fen@csc.kth.se

Abstract. This paper describes a proposed formalization of the notion of Applications Profiles as used in the Dublin Core community. The formalization, called Description Set Profiles, defines syntactical constraints on metadata records conforming to the DCMI Abstract Model using an XML syntax. The mapping of this formalism to syntax-specific constraint languages such as XML Schema is discussed.

Introduction

The term *profile* has been widely used to refer to a document that describes how standards or specifications are deployed to support the requirements of a particular application, function, community or context, and the term *application profile* has recently been applied to describe this tailoring of metadata standards by their implementers (Heery & Patel, 2000).

Since then, the Dublin Core Metadata initiative (DCMI) has published a formalization of the Dublin Core metadata model called the DCMI Abstract Model (Powell et al, 2007), which provides the necessary foundation for a formalization of application profiles that lends itself to machine processing.

This paper describes a proposed formalization of the notion of Applications Profiles as used in the Dublin Core community, called Description Set Profiles, or *DSPs*. This formalization is simplified by focusing on the core aspect of application profiles: the need for syntactically constraining the metadata instances.

Dublin Core Application Profiles

As described in the Singapore Framework for Dublin Core Application Profiles (Singapore Framework, 2008), a DSP is part of a documentation package for Dublin Core Application Profiles (DCAPs) containing

- Functional requirements, describing the functions that the application profile is designed to support, as well as functions that are out of scope
- Domain model, defining the basic entities and their relationships using an formal or informal modeling framework.
- Description Set Profile, as described in this paper
- Usage guidelines, describing how to apply the application profile, how the used properties are intended to be used in the application context etc.
- Encoding syntax guidelines, defining application profile-specific syntaxes, if any.

The DSP thus represents the machine-processable parts of a Dublin Core Application Profile.

There are existing attempts at defining a formal model for Dublin Core Application Profiles. Two important attempts have been documented in CEN CWA 14855, defining an overarching model for documenting application profiles, and CEN CWA 15248 that defined a machine-processable model for DCAPs.

These models depend on single-resource model for application profiles, where the DCAP describes a single resource and its properties. In the light of emerging multi-entity application profiles such as the Eprints Application Profile (Allinson et al 2007), where a five-entity model is used, a one-entity DCAP model is clearly insufficient. Also, earlier attempts at defining DCAPs have not had the benefit of a formal model for Dublin Core metadata, the DCMI Abstract Model, (Powell et al, 2007).

The Singapore Framework described above is intended to support DCAPs at the level of complexity represented by the ePrints DCAP.

Description Set Profiles

The DSP model relies on the metadata model defined in the DCMI Abstract Model and constrains the set of valid metadata records. Thus, a DSP defines a set of metadata records that are valid instances of an application profile. The De-

scription Set Profile model is being developed within the Dublin Core Architecture Forum and is in progress of being put forward as a DCMI Working Draft.

The first part of the paper describes the design of the DSP specification in the context of Dublin Core Application Profiles, uses it is intended to support, and some examples of applying it to relevant problems. Later in the paper, we discuss how the approach could be generalized to graph-based metadata such as RDF, and the potential benefits of such an approach.

The Role of Application Profiles

The process of profiling a standard introduces the prospect of a tension between meeting the demands for efficiency, specificity and localization within the context of a community or service on the one hand, and maintaining interoperability between communities and services on the other. Furthermore, different metadata standards may provide different levels of flexibility: some standards may be quite prescriptive and leave relatively few options for customization; others may present a broad range of optional features which demand a considerable degree of selection and tailoring for implementation.

It is desirable to be able to use community- or domain-specific metadata standards or component parts of those standards in combination. The implementers of metadata standards should be able to assemble the components that they require for some particular set of functions. If that means drawing on components that are specified within different metadata standards, that should be possible. They should also be safe in the knowledge that the assembled whole can be interpreted correctly by independently designed applications. Duval et al (2002) employ the metaphor of the Lego set to describe this process: an application designer should be able to snap together selected building blocks drawn from the kits provided by different metadata standards to build the construction that meets their requirements, even if the kits that provide those blocks were created quite independently.

In a Dublin Core Application Profile, the terms referenced are, as one would expect, terms of the type described by the DCMI Abstract Model, i.e. a DCAP describes, for some class of metadata descriptions, which *properties* are referenced in statements and how the use of those properties may be constrained by, for example, specifying the use of *vocabulary encoding schemes* and *syntax encoding schemes*. The DC notion of the application profile imposes no limitations on whether those properties or encoding schemes are defined and managed by DCMI or by some agency: the key requirement is that the properties referred to in a DCAP are compatible with the RDF notion of property.

It is a condition of that abstract model that all references to terms in a DC metadata description are made in the form of URIs. Terms can thus be drawn from any source, and references to those terms can be made without ambiguity. This set of terms can be regarded as the vocabulary of the application or community that the application profile is designed to support. The terms within that vocabulary may also be deployed within the vocabularies of many other DCAPs.

It is important to realize that the semantics of those terms is carried by their definition, independent of any application profile. Thus, semantic interoperability is addressed outside of the realm of application profiles, and therefore works between application profiles. Instead, application profiles focus on the *set of metadata records* that follow the same guidelines. Therefore, application profiles are more about high-level syntactic or structural interoperability than about semantics.

The Design of Description Set Profiles

The Dublin Core Description Set Profile model is designed to offer a simple constraint language for Dublin Core metadata, based on the DCMI Abstract Model and in line with the requirements for Dublin Core Application Profiles as set forth by the Singapore Framework. It constrains the resources that may be described by descriptions in the description set, the properties that may be used, and the ways a value may be referenced.

A DSP does, however, *not* address the following:

- Human-readable documentation.
- Definition of vocabularies.
- Version control.

A DSP contains the formal syntactic constraints only, and will need to be combined with human-readable information, usage guidelines, version management, etc. in order to be used as an application profile. However, the design of the DSP information model is intended to facilitate the merging of DSP information and external information of the above kinds, for example by tools generating human-readable documentation for a DCAP.

A DSP describes the structure of a Description Set by using the notions of "templates" and "constraints".

A *template* describes the possible metadata structures in a conforming record. There are two levels of templates in a Description Set Profile:

- **Description templates**, that contains the statement templates that apply to a single kind of description as well as constraints on the described resource.
- **Statement templates**, that contains all the constraints on the property, value strings, vocabulary encoding schemes, etc. that apply to a single kind of statement.

While templates are used to express structures, *constraints* are used to limit those structures. Figure 1 depicts the basic elements of the structure.

Thus, the DSP definition contains constructs for restricting

- what properties may be used in a statement and the multiplicity of such statements
- what languages and syntax encoding schemes may be used for literals and value strings, and if they may be used or not
- what vocabulary encoding schemes and value URIs that may be used, and if they may be used or not.

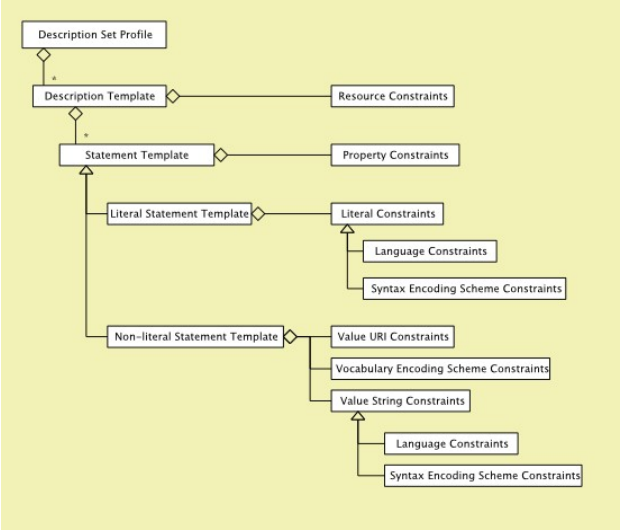

Figure 1: Templates and constraints in a DSP

The DSP specification also contains a pseudo-algorithm that defines the semantics of the above constraints, i.e. how an application is supposed to process a DSP. The algorithm takes as input a description set and a DSP, and gives the answer matching or non-matching .

The Book DSP example

To show some of the features of the DSP model, consider the example of an application profile that wants to describe a book and its author. We would like to describe the following:

- A book
  - The title (dcterms:title) of the book (a literal string with language tag)
  - The creator (dcterms:creator) of the book, described separately
    - A single value string for the creator is allowed
    - No value URI for the creator is allowed
    - No vocabulary encoding scheme for the creator is allowed
- The Creator of the book
  - The name (foaf:name) of the creator (a literal string)

Using the XML serialization of a DSP, we would end up with the following XML:

```
<DescriptionSetTemplate>
  <DescriptionTemplate maxOccur="1" minOccur="1">

    <StatementTemplate maxOccur="1" type="literal">
      <Property>http://purl.org/dc/terms/title</Property>
      <LiteralConstraint>
        <SyntaxEncodingSchemeOccurrence>disallowed</SyntaxEncodingSchemeOccurrence>
        <LanguageOccurrence>optional</LanguageOccurrence>
      </LiteralConstraint>
    </StatementTemplate>

    <StatementTemplate maxOccur="1" type="nonliteral">
      <Property>http://purl.org/dc/terms/creator</Property>
      <NonliteralConstraint descriptionTemplateID="creator">
        <ValueURIOccurrence>disallowed</ValueURIOccurrence>
          <VocabularyEncodingSchemeOccurrence>disallowed</VocabularyEncodingSchemeOccur-
rence>
        <ValueStringConstraint maxOccur="1">
          <SyntaxEncodingSchemeOccurrence>disallowed</SyntaxEncodingSchemeOccurrence>
          <LanguageOccurrence>disallowed</LanguageOccurrence>
        </ValueStringConstraint>
```

```
      </NonliteralConstraint>
    </StatementTemplate>

  </DescriptionTemplate>

  <DescriptionTemplate maxOccur="1" minOccur="1">
    <StatementTemplate maxOccur="1" type="literal">
      <Property>http://xmlns.com/foaf/0.1/name</Property>
      <LiteralConstraint>
        <SyntaxEncodingSchemeOccurrence>disallowed</SyntaxEncodingSchemeOccurrence>
        <LanguageOccurrence>disallowed</LanguageOccurrence>
      </LiteralConstraint>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

The above XML documents the Book DSP in a machine-processable way. The DSP describes a class of description sets matching the given constraints on the book and creator descriptions.

We will now see how such a format can be used.

Using DSPs

A Description Set Profile can be used for many different purposes, such as:

- as a formal representation of the constraints of a Dublin Core Application Profile
- as a syntax validation tool
- as configuration for databases
- as configuration for metadata editing tools

The DSP specification tries to be abstract enough to support such diverse requirements.

Formal documentation: The Wiki DSP generator

An example of where DSPs fills the purpose of formal documentation is the Wiki DSP generator used by the Dublin Core project and developed by one of the authors, Fredrik Enoksson. The software adds markup definitions to a wiki system (currently a MoinMoin installation) that generates a HTML-formatted display of the DSP, intermingled with human-readable text. Upon request, the software can generate an XML file.

The Wiki can then be used to host both the human-readable application profile

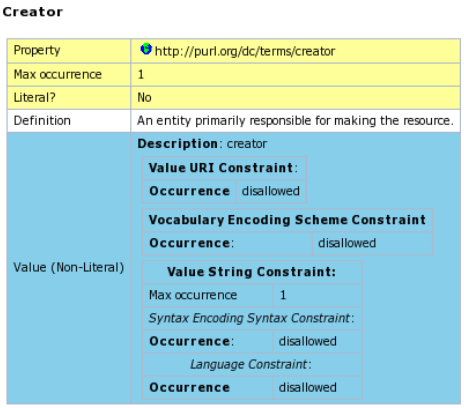

Figure 2: An HTML rendering of the DSP Wiki syntax

guidelines and the XML version, maintained in a single place. See Figure 2 for the HTML output for the Book DSP example.

The wiki syntax is defined in Enoksson (2007).

Syntax validation

Validating metadata using a DSP can be done directly by an implementation of the DSP model in a custom validation tool. A more promising approach, however, is to leverage the widespread tool support for validating existing concrete syntaxes and, in particular, for XML validation.

Given a concrete XML syntax for DCAM-based metadata, such as DC-XML (currently being defined by the DCMI), a DSP can be converted to a syntax-specific validating schema. In the XML case, there are multiple options, such as XML Schema, RelaxNG and SchemaTron, each supporting different complexity in constraints. The authors are currently experimenting with translations from a DSP to these schema languages.

Interesting to note is that the complexity of such a translation is dependent on multiple factors:

- The flexibility of the schema language. XML Schema has well-documented difficulties in expressing certain forms of constraints, that are simple to express in RelaxNG, etc.
- The options available in a DSP. If the model allows for too complex constraints, translating them into a schema language will prove difficult.
- The design of the XML serialization of DCAM metadata. A more regular and straightforward syntax is more easily constrained.

The above considerations affects the design of the DSP specification it is desirable that it be straightforward to implement. It also affects the design of Dublin Core syntaxes, especially DC-XML, which is currently under revision it is desirable that the syntax is straightforward to validate using DSPs.

Metadata editors

DSPs have successfully been used to configure metadata editors. The SHAME metadata editing framework (Palmér et al 2007) is a RDF-based solution for generating form-based RDF metadata editors. The DSP XML format is translated to the form specification format of SHAME, and then used to create an editor. See Figure 3 for an example editor generated from a definition of a Simple Dublin Core Application Profile .

Figure 3: The SHAME editor configured by a DSP

Conclusions

The definition of a formal model for Description Set Profiles marks an important milestone in the evolution of the Dublin Core Metadata Initiative, and is a validation of the DCMI Abstract Model as a foundation for defining application profiles. Still, the model has yet to be validated by wide deployment and implementation, and many important issues remain to be studied. A few initial proofs of the concepts have been realized using DSP for formal documentation, using DSPs to configure metadata editors, and using DSPs to generate XML Schemas for validation.

We expect that the next few years will show if DSPs solved the perceived problem or not. As part of the DC Singapore Framework for applications profiles, we hope that DSPs will serve the community's need for application profile definitions in support of quality control.

References

Allinson, J., Johnston, P., Powell, A. (2007), A Dublin Core Application Profile for Scholarly Works, *Ariadne Issue 50*, January 2007. Retrieved Sep 1, 2007, from http://www.ariadne.ac.uk/issue50/allinson-et-al/

Baker, T. (2003), DCMI Usage Board Review of Application Profiles. Retrieved Sep 1, 2007, from http://dublincore.org/usage/documents/profiles/

Baker, T. (2005), Diverse Vocabularies in a Common Model: DCMI at ten years, Keynote speech, DC-2005, Madrid, Spain. Retrieved Sep 1, 2007, from http://dc2005.uc3m.es/program/presentations/2005-09-12.plenary.baker-keynote.ppt

Bearman, D., Miller, E., Rust, G., Trant, J. & Weibel, S. (1999), A Common Model to Support Interoperable Metadata, *D-Lib Magazine*, January 1999. Retrieved Sep 1, 2007, from http://www.dlib.org/dlib/january99/bearman/01bearman.html

Brickley, D. & Guha, R. V. (2004), RDF Vocabulary Description Language 1.0: RDF Schema, *W3C Recommendation 10 February 2004*. Retrieved Sep 1, 2007, from http://www.w3.org/TR/rdf-schema/

Carroll, J.J., Stickler, P. (2004), TriX: RDF Triples in XML, Technical Report HPL-2004-56, HP Labs. Retrieved Sep 1, 2007, from http://www.hpl.hp.com/techreports/2004/HPL-2004-56.pdf

Dublin Core Application Profile Guidelines (2003), CEN Workshop Agreement CWA 14855. Retrieved Sep 1, 2007, from ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa14855-00-2003-Nov.pdf

The Dublin Core Singapore Framework, DCMI. Retrieved Sep 1, 2007, from http://dublincore.org/architecturewiki/SingaporeFramework

Duval, E., Hodgins, W., Sutton, S. & Weibel, S. L. (2002), Metadata Principles and Practicalities, *D-Lib Magazine*, April 2002. Retrieved Sep 1, 2007, from http://www.dlib.org/dlib/april02/weibel/04weibel.html

Enoksson, F., ed. (2007), Wiki format for Description Set Profiles. Retrieved Sep 1, 2007, from http://dublincore.org/architecturewiki/DSPWikiSyntax

Friesen, N., Mason, J. & Ward, N. (2002), Building Educational Metadata Application Profiles, *Dublin Core - 2002 Proceedings: Metadata for e-Communities*: *Supporting Diversity and Convergence.* Retrieved Sep 1, 2007, from http://www.bncf.net/dc2002/program/ft/paper7.pdf

Godby, C. J., Smith, D. & Childress, E. (2003), Two Paths to Interoperable Metadata, *Proceedings of DC-2003: Supporting Communities of Discourse and Practice Metadata Research & Applications*, Seattle, Washington (USA). Retrieved Sep 1, 2007, from http://www.siderean.com/dc2003/103_paper-22.pdf

Guidelines for machine-processable representation of Dublin Core Application Profiles (2005), CEN Workshop Agreement CWA 15248. Retrieved Sep 1, 2007, from ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/MMI-DC/cwa15248-00-2005-Apr.pdf

Heery, R. & Patel, M. (2000), Application Profiles: mixing and matching metadata schemas, *Ariadne Issue 25*, September 2000. Retrieved Sep 1, 2007, from http://www.ariadne.ac.uk/issue25/app-profiles/

Heflin, J. (2004), OWL Web Ontology Language Use Cases and Requirements, *W3C Recommendation 10 February 2004*. Retrieved Sep 1, 2007, from http://www.w3.org/TR/webont-req/

Johnston, P., (2005a), XML, RDF, and DCAPs. Retrieved Sep 1, 2007, from http://www.ukoln.ac.uk/metadata/dcmi/dc-elem-prop/

Johnston, P., (2005b), Element Refinement in Dublin Core Metadata. Retrieved Sep 1, 2007, from http://dublincore.org/documents/dc-elem-refine/

Klyne, G. & Carroll, J. J. (2004), Resource Description Framework (RDF): Concepts and Abstract Syntax, *W3C Recommendation 10 February 2004*. Retrieved Sep 1, 2007, from http://www.w3.org/TR/rdf-concepts/

Lagoze, C. (1996), The Warwick Framework A Container Architecture for Diverse Sets of Metadata, D-Lib Magazine, July/August 1996. Retrieved Sep 1, 2007, from http://www.dlib.org/dlib/july96/lagoze/07lagoze.html

Lagoze, C., Sompel, H. Van de (2007), Compound Information Objects: The OAI-ORE Perspective. Retrieved Sep 1, 2007, from http://www.openarchives.org/ore/documents/CompoundObjects-200705.html

Manola, F. & Miller, E. (2004), RDF Primer, W3C Recommendation 10 February 2004. Retrieved Sep 1, 2007, from http://www.w3.org/TR/rdf-primer/

Nilsson, M., ed. (2007), DCMI Description Set Profile Specification. Retrieved Sep 1, 2007, from http://dublincore.org/architecturewiki/DescriptionSetProfile

Nilsson, M., Johnston, P., Naeve, A., Powell, A. (2007), The Future of Learning Object Metadata Interoperability, in Harman, K., Koohang A. (eds.) *Learning Objects: Standards, Metadata, Repositories, and LCMS* (pp 255-313), Informing Science press, ISBN 8392233751.

Palmér, M., Enoksson, F., Nilsson, M., Naeve, A. (2007), Annotation profiles: Configuring forms to edit RDF. *Proceedings of the international conference on Dublin Core and metadata applications 2007: Application Profiles: Theory and Practice*, Singapore, Aug 27 - 31 2007. Retrieved Sep 15, 2007, from http://www.dcmipubs.org/ojs/index.php/pubs/article/viewFile/27/2

Powell, A., Nilsson, M., Naeve, A., Johnston, P. (2007), DCMI Abstract Model, *DCMI Recommendation.* Retrieved Sep 1, 2007, from http://dublincore.org/documents/abstract-model/

Uschold, M. & Gruninger, M. (2002), Creating Semantically Integrated Communities on the World Wide Web, Invited Talk, Semantic Web Workshop, Co-located with WWW 2002, Honolulu, HI, May 7 2002. Retrieved Sep 1, 2007, from http://semanticweb2002.aifb.uni-karlsruhe.de/USCHOLD-Hawaii-InvitedTalk2002.pdf