eFoundations

metadata, middleware, elearning

October 07, 2009

What is "Simple Dublin Core"?

Posted by PeteJ at 12:27 07 October 2009 in Metadata , Resource Discovery | Permalink

Over the last couple of weeks I've exchanged some thoughts, on Twitter and by email, with John Robertson of CETIS, on the topic of "Qualified Dublin Core", and as we ended up discussing a number of areas where it seems to me there is a good deal of confusion, I thought it might be worth my trying to distill them into a post here (well, it's actually turned into a couple of posts!).

I'm also participating in an effort by the DCMI Usage Board to modernise some of DCMI's core documentation, and I hope this can contribute to that work. However, at this point this is, I should emphasise, a personal view only, based on my own interpretation of historical developments, not all of which I was around to see at first hand, and should be treated accordingly.

The exchange began with a couple of posts from John on Twitter in which he expressed some frustration in getting to grips with the notion referred to as "Qualified Dublin Core", and its relationship to the concept of "DC Terms".

First, I think it's maybe worth taking a step back from the "Qualified DC" question, and looking at the other concepts John mentions in his first question: "the Dublin Core Metadata Element Set (DCMES)" and "Simple Dublin Core", and that's what I'll focus on in this post.

The Dublin Core Metadata Element Set (DCMES) is a collection of (what DCMI calls) "terms" - and it's a collection of "terms" of a single type, a collection of properties - each of which is identified by a URI beginning http://purl.org /dc/elements/1.1/; the URIs are "in that namespace". Historically, DCMI referred to this set of properties as "elements".

Although I'm not sure it is explicitly stated anywhere, I think there is a policy that - at least barring any quite fundamental changes of approach by DCMI - no new terms will be added to that collection of fifteen terms; it is a "closed" set, its membership is fixed in number.

A quick aside: for completeness, I should emphasise that those fifteen properties have not been "deprecated" by DCMI. Although, as I'll discuss in the next post, a new set of properties has been created in the "DC terms" set of terms, the DCMES properties are still available for use in just the same way as the other terms owned by DCMI. The DCMES document says:

> Implementers may freely choose to use these fifteen properties either in their legacy dc: variant (e.g., http://purl.org /dc/elements/1.1/creator) or in the dcterms: variant (e.g., http://purl.org/dc/terms/creator) depending on application requirements. The RDF schemas of the DCMI namespaces describe the subproperty relation of dcterms:creator to dc:creator for use by Semantic Web-aware applications. Over time, however, implementers are encouraged to use the semantically more precise dcterms: properties, as they more fully follow emerging notions of best practice for machine-processable metadata.

The intent behind labelling them as "legacy" is, as Tom Baker puts it, to "gently promote" the use of the more recently defined set of properties.

Perhaps the most significant characteristic of that set of terms is that it was created as a "functional" set, by which I mean that it was created with the notion that that set of fifteen properties could and would be used together in combination in the descriptions of resources. And I think this is reflected, for instance, in the fact that some of the "comments" provided for individual properties refer to other properties in that set (e.g. dc:subject/dc:coverage, dc:format/dc:type).

And there was particular emphasis placed on one "pattern" for the construction of descriptions using those fifteen properties, in which a description could contain statements referring only to those fifteen properties, all used with literal values, and any of those 15 properties could be referred to in multiple statements (or in none). In that pattern of usage, the fifteen properties were all "optional and repeatable", if you like. And that pattern is often referred to as "Simple Dublin Core".

Such a "pattern" is what today - if viewed from the perspective of the DCMI Abstract Model and the Singapore Framework - we would call a Description Set Profile (DSP).

So "Simple Dublin Core" might be conceptualised as a DSP designed, initially at least, for use within a very simple, general purpose DC Application Profile (DCAP), constructed to support some functions related to the discovery of a broad range of resources. That DSP specifies the following constraints:

- A description set must contain exactly one description (Description Template: Minimum occurrence constraint = 1; Maximum occurrence constraint = 1)
- That description may be of a resource of any type (Description Template: Resource class constraint: none (default))
- For each statement in that description:
    - The property URI must be drawn from a list of the fifteeen URIs of the DCMES properties (Statement Template: Property list constraint: (the 15 URIs))
    - There must be at least one such statement; there may be many (Statement Template: Minimum occurrence constraint = 1; Maximum occurrence constraint = unbounded)
    - A literal value surrogate is required (Statement Template: Type constraint = literal)
    - Within that literal value surrogate, the use of a syntax encoding sceme URI is not permitted (Statement Template/Literal Value: Syntax Encoding Scheme Constraint = disallowed)

And this DSP represents a "pattern" that is quite widely deployed, perhaps most notably in the context of systems supporting the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), which requires that an OAI-PMH repository expose records using an XML format called oai_dc, which is essentially a serialisation format for this DSP. (There may be an argument that the "Simple DC" pattern has been overemphasised at the expense of other patterns, and as a result people have poured their effort into using that pattern when a different one might have been more appropriate for the task at hand, but that's a separate discussion!)

It seems to me that, historically, the association between the DCMES as a set of terms on the one hand and that particular pattern of usage of those terms on the other was so close that, at least in informal accounts, the distinction between the two was barely made at all. People tended to (and still do) use the terms "Dublin Core Metadata Element Set" and "Simple Dublin Core" interchangeably. So, for example, in the introduction to the Usage Guide, one finds comments like "Simple Dublin Core comprises fifteen elements" and "The Dublin Core basic element set is outlined in Section 4. Each element is optional and may be repeated." I'd go as far as to say that many uses of the generic term "Dublin Core", informal ones at least, are actually references to this one particular pattern of usage. (I think the glossary of the Usage Guide does try to establish the difference, referring to "Simple Dublin Core" as "The fifteen Dublin Core elements used without qualifiers, that is without element refinement or encoding schemes.")

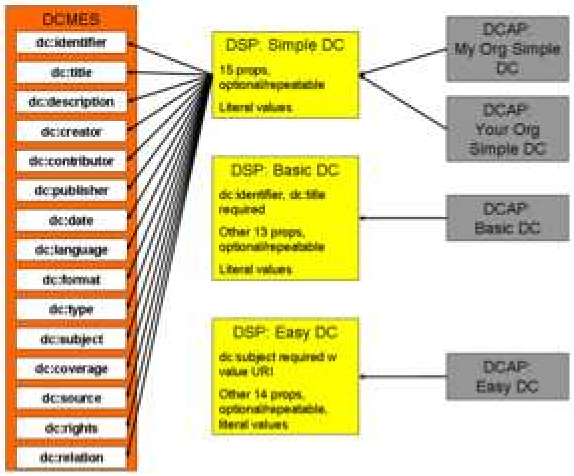
The failure to distinguish more clearly between a set of terms and one particular pattern of usage of those terms has caused a good deal of confusion, and I think this will becomes more apparent when we consider the (rather more complex) case of "Qualified Dublin Core", as I'll do in the next post, and it's an area which I'm hoping will be addressed as part of the Usage Board review of documentation.

If you look at the definitions of the DCMES properties, in the human-readable document, and especially in the RDF Schema descriptions provided in the "namespace document" http://purl.org/dc/elements/1.1/, with the possible exceptions of the "cross-references" I mentioned above, those definitions don't formally say anything about using those terms together as a set, or about "optionality/repeatability": they just define the terms; they are silent about any particular "pattern of usage" of those terms.

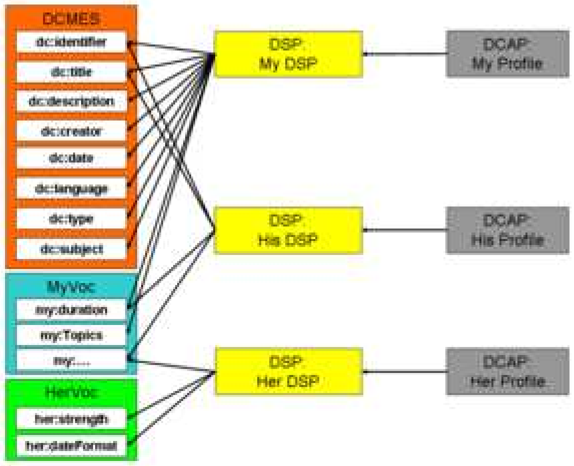
So, such patterns of usage of a collection of terms exist distinct from the collection of terms. And it is possible to define multiple patterns of usage. multiple DSPs, referring to that same set of 15 properties. In addition to the "all optional/repeatable" pattern, I might find myself dealing with some set of resources which all have identifiers and all have names, and operations on those identifiers and names are important to my application, so I could define a pattern/DSP ("PeteJ's Basic DC" DSP) where I say all my descriptions must contain at least one statement referring to the dc:identifier property and at least one statement referring to the dc:title property, and the other thirteen properties are optional/repeatable, still all with literal values. Another implementer might find themselves dealing with some stuff where everything has a topic drawn from some specified SKOS concept scheme, so they define a pattern ("Fred's Easy DC" DSP) which says all their descriptions must contain at least one statement referring to the dc:subject property and they require the use, not of a literal, but of a value URI from that specific set of URIs. So now we have three three different DC Application Profiles, incorporating three different patterns for constructing description sets (three different DSPs) each referring to the same set of 15 properties.

It's also worth noting that the "Simple DC" pattern of usage, a **single** set of structural constraints, could be deployed in **multiple** DC Application Profiles, supporting different applications and containing different human-readable guidelines. (I was going to point to the document *Using simple Dublin Core to describe eprints* as an actual example of this, but having read that document again, I think strictly speaking it probably introduces additional structural constraints (i.e. introduces a different DSP), e.g. it requires that statements using the dc:type property refer to values drawn from a bounded set of literal values.)

The graphic below is an attempt to represent what I see as those relationships between the DCMES vocabulary, DSPs and DCAPs:



Finally, it's worth emphasising that the 15 properties of the DCMES, or indeed any subset of them - there is no requirement that a DSP refer to all, or indeed any, of the properties of the DCMES - , may be referred to in other DSPs in combination with other terms from other vocabularies, owned either by DCMI or by other parties.



The point that DCMI's concept of an "application profile" is not based either on the use of the DCMES properties in particular or on the "Simple DC" pattern is an important one. Designing a DC application profile does **not** require taking either the DCMES or the "Simple DC" pattern as a starting point; **any** set of properties, classes, vocabulary encoding schemes and syntax encoding schemes, owned by any agency, can be referenced. But that is rather leading me into the next post, where I'll consider the similar (but rather more messy) picture that emerges once we start talking about "Qualified DC".

Technorati Tags: DCAM, Description Set Profile, DSP, Dublin Core, metadata, Simple Dublin Core

TrackBack

TrackBack URL for this entry:
http://www.typepad.com/services/trackback/6a00d8345203ba69e20120a5a4fdf0970b
Listed below are links to weblogs that reference What is "Simple Dublin Core"?:

Comments

Verify your Comment

Previewing your Comment