

- Glossary meeting
 - <e:/u/folders/pittsburgh/dc2010/BRING/2010-10-21.dcuser-dcglossary-meeting.pdf>
 - <e:/u/folders/pittsburgh/dc2010/BRING/Glossary-notes.txt>
 - <http://dublincore.org/dcmirdataskgroup/apDesigns?action=print>
- DCAM2 meeting
 - <http://www.w3.org/2001/sw/wiki/JointMeeting2010>
 - <http://dublincore.org/architecturewiki/DcamInContext?action=print>
 - <e:/u/folders/DC/dcalog/2010/2010-10-12.xg-owl-and-application-profiles.txt>

Glossary and User Guide Task Groups - face-to-face meeting

Date: 2010-10-21, Thursday, 14:00-15:30
In program: <http://www.asis.org/Conferences/DC2010/program-sessions.html#userdoc>
Expected: Tom, Mary, Stefanie, Marcia, Pete (remote)

1. 14:00-14:30 - User Guide (Stefanie)

Discuss (see meeting PDF)
-- <http://colab.mpg.de/mediawiki/UsingDC>
-- <http://colab.mpg.de/mediawiki/CreatingMetadata>
-- <http://colab.mpg.de/mediawiki/PublishingMetadata>
-- Pete comments on User Guide

Note: guidance on specific properties in the User Guide may be discussed in the Libraries Task Group meeting on Wednesday, see <http://www.asis.org/Conferences/DC2010/program-sessions.html#librarytaskgroup>

2. 14:30-15:00 - Glossary and FAQ (Tom and Mary)

Discuss glossary entries (see meeting PDF)
DCMI Metadata Terms
Dublin Core
Dumb-Down
Namespace Policy
One to One Principle
Open World Mindset
RDF
Resource Discovery
Simple Dublin Core

Discuss FAQ entries (see meeting PDF)
On reusing XML elements

3. 15:00-15:30 - Issues and Next Steps

General principles
-- One list, with both current and obsolete technology.
-- Legacy or archaic terminology can be marked as such.

Issues
-- Cross-references and redundancies among User Guide, Glossary, FAQ
-- Form of publication (wiki document?)
-- Next steps, work plan

UsingDC

From MPDLMediaWiki

Contents

- 1 Purpose and Scope of this Guide
- 2 What is Metadata?
- 3 What is Linked Data?
- 4 Levels of Interoperability
- 5 What is Dublin Core?
- 6 Dublin Core and Linked Data
- 7 Dublin Core namespaces / URIs
- 8 Dublin Core Properties

Purpose and Scope of this Guide

"The Dublin Core" (aka the Dublin Core Metadata Element Set), created in 1995, is a set of fifteen generic elements for describing resources. These are: Creator, Contributor, Publisher, Title, Date, Language, Format, Subject, Description, Identifier, Relation, Source, Type, Coverage, and Rights. Today the Dublin Core is a formal standard ^{[1][2][3]}, used in countless implementations, and one of the top metadata vocabularies on the Web.

"Dublin Core metadata" is about more than fifteen elements. It is best described as a *style of metadata* that has evolved from efforts to put the fifteen elements into the context of a coherent approach to metadata on the World Wide Web generally. Since the late 1990s, the Dublin Core style has evolved in the context of a Dublin Core Metadata Initiative (DCMI) -- now incorporated as a non-profit organization hosted at the National Library Board of Singapore -- in tandem with a generic approach to metadata developed in the World Wide Web Consortium under the banner "Semantic Web". The Semantic Web approach achieved a breakthrough in 2006 with the Linked Data movement, which uses DCMI Metadata Terms as one of its key vocabularies. "DCMI Metadata Terms" is a larger set that includes the fifteen elements along with several dozen related properties and classes. "Dublin Core application profile" is the key expression of the Dublin Core style. An application profile uses DCMI metadata terms together with terms from other, more specialized vocabularies to describe a specific type of information -- and it does this in the framework of the Semantic Web.

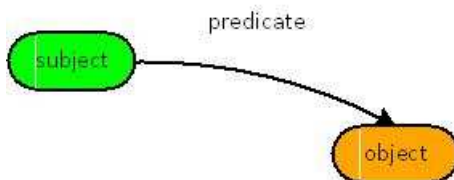
These guidelines provide an entry point for users of Dublin Core -- i.e., for users of DCMI Metadata Terms in the "Dublin Core style". For catalogers, it will show how to create metadata "descriptions" for information resources such as documents, images, data, etc. Implementers will it support publishing Dublin Core Metadata as Linked Data. The guidelines will neither show how to create or Dublin Core Application Profiles -- for this see the Guidelines for Dublin Core Application Profiles (<http://dublincore.org/documents/profile-guidelines/>) -- nor how to express Dublin Core Terms in different syntaxes -- for this see the section "Syntax Guidelines" of DCMI Specifications (<http://dublincore.org/specifications/>) .

What is Metadata?

Metadata has been with us since people made lists on clay tablets and scrolls. The term "meta" comes from the Greek for "alongside" or "with". Over time, "meta" was also used to denote something transcendental, or beyond nature. "Meta-data", then, is "data about data", such as the contents of catalogs, inventories, etc. Since the 1990s, "metadata" most commonly denotes machine-readable descriptions of things, most commonly in the context of the Web. The structured descriptions of metadata help find relevant resources in the undifferentiated masses of data available online. Anything of interest can be described with metadata, from book collections to football leagues and stuff you want to sell. Describing different types of resources requires different types of metadata and metadata standards.

What is Linked Data?

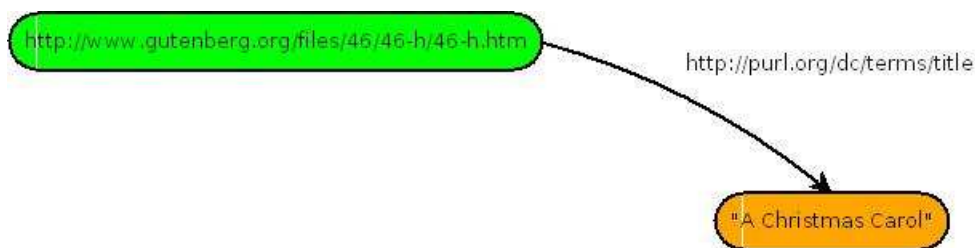
Linked Data is a method of exposing, sharing, and connecting data on the Semantic Web using URIs and RDF (see <http://linkeddata.org/>). Metadata are the backbone of this method, making statements about data and how they relate to each other. In Linked Data these statements have to be expressed in RDF triples, which break statements in three parts:



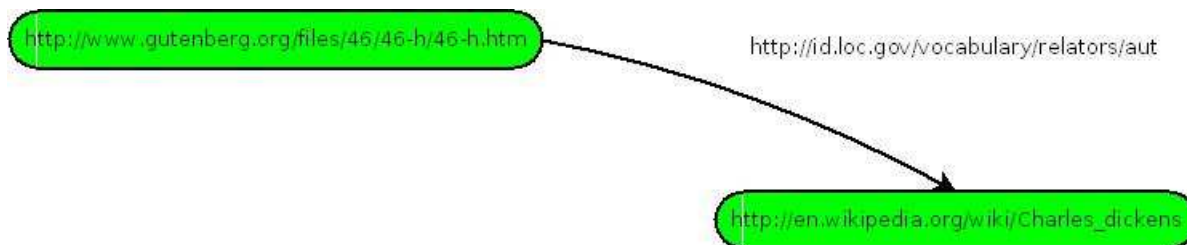
- the subject - the part that identifies the thing the statement describes,
- the predicate - the part that identifies a property of the described thing.
- the object - the part that identifies the value this property has when describing this thing.

(<http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#statements>)

Another "must" when publishing metadata in Linked Data is the usage of URIs. In Linked Data you need URIs referencing to things by identifying, localizing and interlinking them. Considering this a triple graph describing Charles Dickens "A Christmas Carol" might look this way:



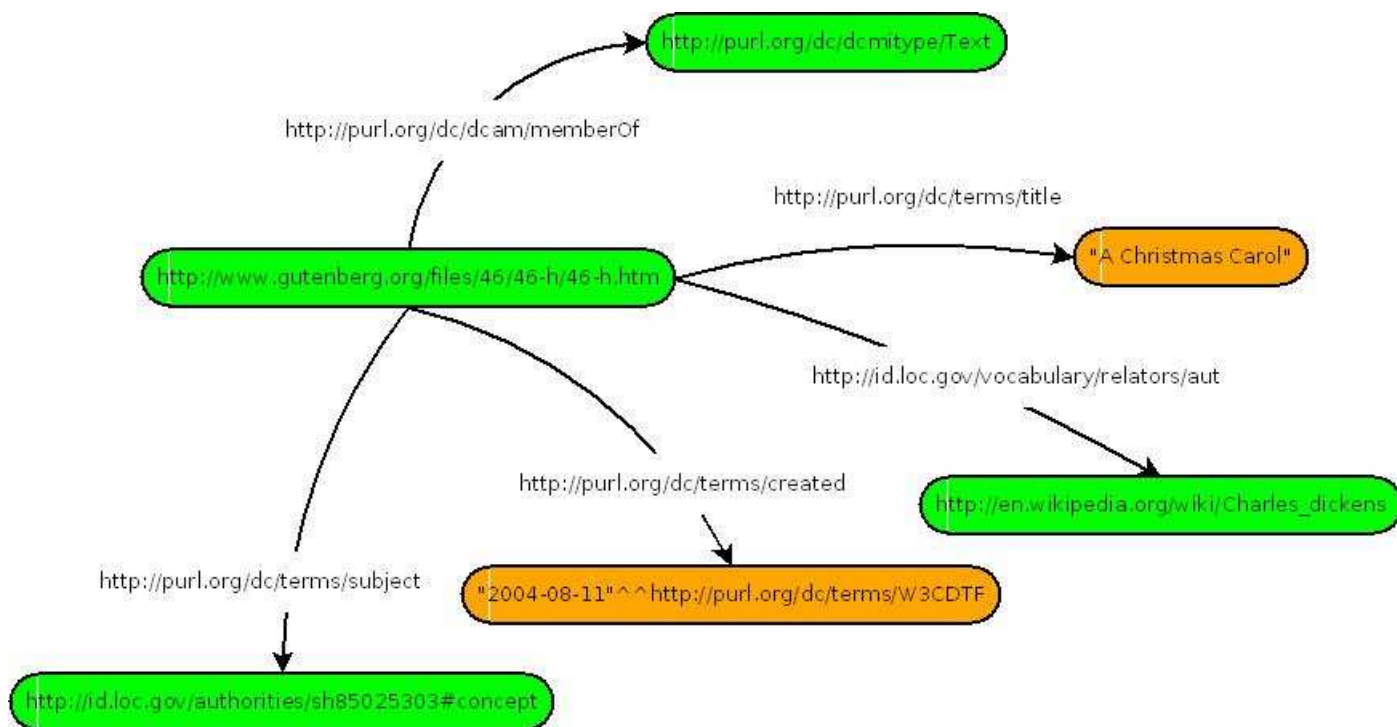
Here the value "A Christmas Carol" is a simple string or literal value. Another sort of values used in a triple are non-literal values, which means you use a URI that references to another description - the description of a thing that is the object of your statement.



Based on the above said a metadata description in Linked Data consists of:

- **a reference to a described resource**, i.e. to the thing that the metadata is *about*, the *subject* of the metadata. This reference takes the form of a Uniform Resource Identifier (URI) or of an unnamed placeholder that is inferred by context (e.g., that metadata embedded in a document is about the enclosing document).
- **references to properties**: typed relationships between the described resource and various bits of descriptive information, or between the described resource and another resource. Examples of properties, which are also known as *predicates*, include the fifteen elements of the Dublin Core.
- **values**: bits of descriptive information (string *literals*) or references to other entities (resources), such as people or concepts, which are related to the described resource via the properties.
- **references to classes**, i.e. to types, or categories, of things, such as the category *books* or the category *people*.
- **references to syntax encoding schemes (RDF datatypes)**, i.e. specifications of how value strings map refer to things in the world, such as *2010-09-24*, which uses the YYYY-MM-DD pattern specified in ISO 8601 to represent the date 22 September 2010.
- **references to vocabulary encoding schemes (VES)**, enumerated sets of resources of which the things referenced as values are members, the way a subject heading belongs to the VES *Library of Congress Subject Headings*.
- **language tags** indicating the language of words used in literal string values.

In a RDF graph this might look this way:



In this example the **described resource** is a web page referenced by the URI `http://www.gutenberg.org/files/46/46-h/46-h.htm`. We may reference to a **class** - in our example its the class "text" of the Dublin Core Type Vocabulary - using the property `memberOf`. Further **properties** of the resource are "title", "author", "created" and "subject". We used literal - "A Christmas Carol" and "2004-08-11" - and non-literal **values** - `<http://en.wikipedia.org/wiki/Charles_dickens>` and `<http://id.loc.gov/authorities/sh85025303#concept>`. Literal values may be constrained by **datatypes** (in our example values describing dates have to be conform with W3CDTF), non-literal values by **vocabulary encoding schemes** (in our example values used describing the topic of a resource have to be concepts of the Library of Congress Subject Headings). A **language tag** can be used to describe the language of a literal value (in our example the value of the title is English).

Levels of Interoperability

Metadata is most helpful when used to navigate the *information jungle* of the Web -- to find, identify, use resources -- or to share and exchange structured information. However there is a tension between requirements of applications and of people:

- people prefer customized descriptions which reflect their understanding of the world
- applications need interoperability between descriptions in order to process them efficiently.

Metadata *vocabularies* help bridge this gap. *Vocabularies* define properties and classes that can be used to describe resources in a coherent way within or between communities. A vocabulary provides the shared basis for exchanging descriptions within groups of people. They support the interoperability of different metadata applications, and they support the ability of applications to change data with systems with no or minimal loss of information. The Dublin Core distinguishes four levels of interoperability (<http://dublincore.org/documents/interoperability-levels/>) :

- **Level 1:** Shared term definitions: At this level, a community uses the same classes and properties, for example DCMI Metadata Terms (<http://dublincore.org/documents/dcmi-terms/>) .
- **Level 2:** Formal semantic interoperability: At this level, the description of resources is based on, or automatically mappable to, the shared formal model for metadata provided by W3C's Resource Description Framework (RDF), the basis of Linked Data.
- **Level 3:** Description Set syntactic interoperability. At this level, resource descriptions are based on a shared notion of RDF-based metadata *records* (based, specifically, on the DCMI Abstract Model (<http://dublincore.org/documents/abstract-model/>)).
- **Level 4:** Description Set Profile interoperability. At this level, a community uses not only the same classes and properties, based on the same underlying RDF model, but also uses, or *constrains* those classes and properties based on an agreed model of the things being described and on shared usage patterns. The Description Set Profile is the key component of what the Dublin Core community calls an *application profile*.

As of Fall 2010, the DCMI approach to defining *records* and *constraints* -- Interoperability Levels 3 and 4 -- is under review in light of the rapid evolution of alternative approaches to documenting metadata usage patterns in application-profile-like constructs for ensuring the consistency and quality of Linked Data.

At the other extreme, Level 1 interoperability is so open-ended that it quickly leads to a proliferation of custom-built solutions incompatible with each other, such as metadata expressed in document formats that require customized software to read and data models that cannot easily be mapped to generic, interoperable representations such as those expressed in RDF.

This User Guide therefore focuses on Level 2 interoperability -- the sweet spot between ad-hoc implementations and shared models of records and constraints, which remain the object of experimentation and research. Given the state of play in 2010, implementers can design their metadata for compatibility Linked Data target with the confidence that this will ensure its formal compatibility with an explosively growing Cloud of Linked Data (<http://lod-cloud.net/>) .

Users can explore the Dublin Core approach to shared record constraints (Levels 3 and 4) in the DCMI documents "Singapore Framework for Dublin Core Application Profiles" (<http://dublincore.org/documents/singapore-framework/>) , the draft "Description Set Profiles: A constraint language for Dublin Core Application Profiles" (<http://dublincore.org/documents/dc-dsp/>) , and the user-oriented "Guidelines for Dublin Core Application Profiles" (<http://dublincore.org/documents/profile-guidelines/>) . A well-developed example of an application profile developed according to these principles may be found in the "Scholarly Works Application Profile" (<http://dublincore.org/scholarwiki/SWAPDSP>) .

What is Dublin Core?

In the mid 1990s Dublin Core started with the idea of "core metadata" for simple and generic resource descriptions. An international, cross-disciplinary group of professionals from librarianship, computer science, text encoding and museum community, and other related fields of scholarship and practice developed such a core standard – the fifteen Dublin Core elements. But this was just the first steps – since then the World Wide Web has changed in some ways and has broken new ground on the way to a semantic web. Dublin Core followed this path developing further standards for metadata based on the World Wide Web Consortium's work on a generic data model for metadata, the Resource Description Framework (RDF). So **Dublin Core metadata standards** today are:

- The DCMI Abstract Model (<http://dublincore.org/documents/abstract-model/>) (DCAM) – an RDF based syntax independent model supporting mappings and cross-syntax translations.
- The Singapore Framework for Dublin Core Application Profiles (<http://dublincore.org/documents/singapore-framework/>) – a framework which defines the components that are necessary for documenting a Dublin Core compatible Application Profile.
- **The Dublin Core Metadata Vocabularies**
 - DCMI Metadata Terms (<http://dublincore.org/documents/dcmi-terms/>) in the `/terms/` Namespace
 - DCMI Type Vocabulary (<http://dublincore.org/documents/dcmi-type-vocabulary/>)
 - Metadata terms related to the DCMI Abstract Model (<http://dublincore.org/documents/abstract-model/>)
 - Dublin Core Metadata Element Set, Version 1.1 (<http://dublincore.org/documents/dces/>)

Dublin Core and Linked Data

Since the emergence of the Semantic Web and Linked Data approaches, implementers face a wider range of choices in designing applications. Traditional approaches based on metadata records and descriptive tags embedded in Web pages remain effective alternatives within closed, controlled implementation environments, while Linked Data approaches are designed to provide metadata in a generic form that is easily reusable by other applications for "mashing up" your data with related data published by others. Linked Data has given new meaning to old ideas, such as embedded metadata, which are being reinvented with new Web technologies and tools to solve practical problems of resource discovery and navigation. "The Dublin Core" (and DCMI Metadata Terms) provides a solid basis for bridging these traditional and modern approaches, lookin at DC terms as a "small language for making a particular class of statements about resources". In this language terms are arranged into a simple pattern of statements and DCMI metadata properties are used as predicates in subject-predicate-object triples.

```
ex:myPicture dc:title "Milking the goats" ;  
             dc:creator <http://d-nb.info/gnd/131724126> .
```

In this example there are two statements about the picture:

- it has a title, which is „Milking the goats
- it has a creator, represented by the URI <http://d-nb.info/gnd/131724126>

The object or value we use in the title statement is a simple string called a literal value. The value used in the creator statement is a non-literal value, a URI that leads to another metadata description, where the person who is the creator is described in a more detailed way.

We call the value used here a non-literal value. We want to say more about the creator and we need another description to do so. There are two statements about this person in our example:

- it's a link to the homepage of her workplace
- and her name

```
<http://d-nb.info/gnd/131724126> foaf:name "Rühle, Stefanie" ;  
foaf:workplaceHomepage <http://www.sub.uni-goettingen.de/> .
```

Though the statements we make here are not using Dublin Core properties they are non the less DC compatible, because in the namespace these terms are explicitly declared to be properties - one of the four metadata terms DCAM knows - and have been assigned a proper URI. This means we may use other properties than Dublin Core Terms as long as these terms are Dublin Core compatible (see above). This way Dublin Core provides the grammar for a "metadata pidgin for digital tourists", a grammar that allows to merge terms from different vocabularies of different communities in one language and may be used to display both simple and complex issues. With this grammar Dublin Core provides a mechanism for extending the Dublin Core term set for additional resource discovery needs expecting that other communities will create and administer additional metadata sets, specialized for the need of their community.

Dublin Core namespaces / URIs

! work in progress!

Dublin Core Properties

These guidelines list all properties defined in the following two namespaces of DCMI Metadata Terms (<http://dublincore.org/documents/dcmi-terms/>) :

- <http://purl.org/dc/elements/1.1/> (referred to here using the short prefix "dc:")
- <http://purl.org/dc/terms/> ("dcterms:").

and illustrate their usage by examples. Examples are offered for two points of view: for the "cataloger" creating metadata descriptions, typically with help from a software interface, and for the "technician" responsible for publishing the data created as linked data.

- **CreatingMetadata**: describes how to create content for DCMI Metadata listing each property by name, abbreviated URI, and definition, and groups together related properties -- i.e. properties that can be described with similar usage guidelines and illustrated with similar examples.
- **PublishingMetadata**: describes how to use DCMI Metadata as linked data listing the properties by namespaces.
 - The terms namespace
 - used with literal values
 - used with non-literal values
 - The legacy namespace

Retrieved from "<http://colab.mpd.l.mpg.de/mediawiki/UsingDC>"

Category: KIM

-
- This page was last modified 15:01:40, 2010-10-11.
 - Content is available under Creative Commons Attribution 2.0.

CreatingMetadata

From MPDLMediaWiki

Go to UsingDC | PublishingMetadata

How to create content for DCMI Metadata

Contents

About the examples

We are presenting the examples in tables. To interpret these tables please consider that:

- yellow rows are standing for statements describing a resource,
- white rows are standing for statements describing a resource that is the value of another statement but is not described by another record,
- bold strings are standing for properties,
- italicized strings are standing for values,
- red strings are standing for values linking to records.

Titles

Title

Title is a property that refers to the name or names by which a resource is formally known.

Alternative

Alternative is a property that refers to a name or names of a resource used as a substitute or alternative to the formal title. These are secondary titles, abbreviations, translations of a title, etc.

Guidelines for the creation of title content

For the title use the name given to the resource.

title	<i>Alvar Aalto Chair No. 66</i>
--------------	---------------------------------

as linked data

In most databases title is one of the main criteria to identify search results. So **if there is no formal title resp. name** you should formulate an adequate one by yourself.

title	<i>Data from a survey about the usage of metadata</i>
--------------	---

as linked data

If there is **more than one title resp. name** you should repeat the title property

title	<i>Autumn Leaves</i>
title	<i>The Dead Leaves</i>

as linked data

or use the alternative property.

Typically alternative is used for **secondary titles**,

title	<i>Passion for Pulses</i>
alternative	<i>A Feast of Beans, Peas and Lentils from Around the World</i>

as linked data

or for **abbreviations**.

title	<i>American Meteorological Association Newsletter</i>
alternative	<i>AMA Newsletter</i>

as linked data

If the title resp. name is expressed in **different languages** you should use language tags.

title	<i>La Joconde</i>	fre
title	<i>Mona Lisa</i>	eng
title	<i>La Gioconda</i>	ita

as linked data

the same is true for alternative

title	<i>EU Stability Programm of Belgium</i>	eng
alternative	<i>Council Opinion on the Updated Stability Programm of Belgium 2009 - 2010</i>	eng
alternative	<i>Stellungnahme des Rates zum aktualisierten Stabilitätsprogramm Belgiens für 2009 - 2012</i>	ger

as linked data

It is recommended to describe titles with plain text (as done in the examples above). But sometimes it is necessary to **create a relationship between the described resource and a more detailed title description**.

This should be used to refer to **a title with different transliterations**,

title		
	in greek	<i>Οιδίππους Τύραννος</i>
	in latin	<i>Oidipous Tyrannos</i>

as linked data

or to refer to a **title authority**.

title	<i>Nibelungenlied Handschrift B</i>
title	<i>nibelungenlied</i>

title	<i>Nibelungenlied Handschrift C</i>
title	<i>nibelungenlied</i>

identifier	<i>nibelungenlied</i>
label	<i>Der Nibelunge Not</i>

alternative label	<i>Nibelungenlied</i>
alternative label	<i>Nibelungenklage</i>
alternative label	<i>Nibelungensage</i>
description	<i>The Nibelungenlied exists of 39 aventiuren created between 1180 and 1210</i>

as linked data

Relationships between Resource and Agents

Using agent properties in Dublin Core

Persons, organizations and services can relate to resources in various ways. DCMI defined only a few common properties to describe the relationship between resources and agents. So when necessary other vocabularies should be used describing these relationships more detailed. In these cases we recommend to use the marcrelator codes (<http://id.loc.gov/vocabulary/relators.html>) . How to use them is described in

- MARC Relator terms and Dublin Core (<http://dublincore.org/usage/documents/relators/>)
- MARC Relator properties in Dublin Core metadata (<http://www.ukoln.ac.uk/metadata/dcmi/marcrel-ex/>)

Contributor

The contributor property represents a relationship between the resource and a person, an organization, or a service making a contribution to a resource.

Creator

The creator property represents a relationship between the resource and a person, an organization, or a service primarily responsible for making the content of a resource.

Publisher

The publisher property represents a relationship between the resource and a person, an organization, or a service responsible for making the resource available and provide access to the resource.

RightsHolder

This property represents a relationship between the resource and a person or an organization owning or managing rights over this resource.

Guidelines for the creation of content for agents

If you know there is a **URI standing for a person or organization** you should use it. For further information you should handle the person or organization like another resource.

Example with an **organization**:

rights holder	gnd	<i>39454-3</i>
	name	<i>Bundesarchiv Koblenz</i>
	homepage	<i>http://www.bundesarchiv.de/index.html.de</i>

as linked data

Example with a **person**:

contributor	gnd	<i>135066719</i>
--------------------	------------	------------------

	family name	<i>Elliott</i>
	given name	<i>Missy</i>
	nick name	<i>Missy E</i>

as linked data

Regardless of the existence of a URI **personal names** should be grouped in family name resp. surname as one part of the name and forename resp. given name as the other part .

You should handle the person name like **another resource**,

creator		
	family name	<i>Shakespeare</i>
	given name	<i>William</i>

as linked data

or you could devide these names **by comma**.

creator	<i>Shakespeare, William</i>
----------------	-----------------------------

as linked data

When you **in doubt about family name and given name** give the name as it appears.

contributor	<i>Snoop Dogg</i>
--------------------	-------------------

as linked data

If there is **more than one contributor/creator/publisher/rightsHolder**, each should be listed separately,

like another resource,

publisher	gnd	<i>2125990-2</i>
	name	<i>Rossijskaja Gosudarstvennaja Biblioteka</i>
	homepage	<i>http://www.rsl.ru/</i>
publisher		
	name	<i>Knižnaja Palata</i>

as linked data

or with plain text.

creator	<i>Hubble Telescope</i>
publisher	<i>University of Nowhere</i>
publisher	<i>All Your Data Inc.</i>

as linked data

Type

The type property refers to a description of the nature or genre of the content of a resource (e.g. a stylistic category, a function or an aggregation level). To describe the physical or digital manifestation, use format.

Guidelines for the creation of type content

We recommend to **select a value from a controlled vocabulary** (e. g. DCMI Type Vocabulary (<http://dublincore.org/documents/dcmi-type-vocabulary/>)).

type	dctype	
		<i>Still Image</i>

as linked data

But you may also use **plain text**.

type	<i>Conference</i>
-------------	-------------------

as linked data

If **no formal controlled vocabulary** exists, you could create a domain specific one.

type	<i>conference</i>
-------------	-------------------

identifier	<i>conference</i>	
label	<i>Conference</i>	eng
label	<i>Tagung</i>	ger
label	<i>съезд</i>	rus

as linked data

Is the resource composed of **multiple components of different types** the property should be repeated.

type	dctype	
		<i>Interactive Resource</i>
type	dctype	
		<i>Text</i>

as linked data

Different communities use a variety of type vocabularies. To ensure interoperability you can use terms from different vocabularies side by side - e.g. a type of the DCMI Type vocabulary in addition to a **non-controlled or domain specific type term**.

type		
		<i>PC Game</i>
type	dctype	
		<i>Software</i>

as linked data

Format

The property format refers to the file format, the physical medium (e.g. the data storage medium), or the dimension (the size or duration) of a resource. The information can be relevant to determine the equipment needed to display or operate a resource (e.g. if the described resource has format pdf you need a pdf reader to use it). To specify the different categories of format you should use extent and/or. To reference to the nature or genre of the content use type.

Guidelines for the creation of format content

For the description of the **file format** we recommend to use a controlled vocabulary - e.g. the list of Internet Media Types (<http://purl.org/NET/mediatypes/>) (MIME).

format	mime	<i>jpeg</i>
---------------	------	-------------

as linked data

You should repeat the properties when **more than one type of value exists**.

format	mime	<i>jpeg</i>
format		<i>40 x 512 pixels</i>

as linked data

Extent

This property refers to the size (e.g. bytes, pages, inches, etc.) or duration (e.g. hours, minutes, days, etc.) of a resource.

Guidelines for the creation of extent content

Typically the value used for the description of the extent consists of a **numeric value and a caption** to specify it. You may use a text string to present it

extent	
	<i>21 minutes</i>

as linked data

or use controlled values for the caption.

extent		
	minutes	<i>21</i>

as linked data

Medium

This property refers to the physical carrier of the resource and may only be used if the resource is of physical nature (e.g. a painting, a sculpture, etc.)

Guidelines for the creation of medium content

Note that the **media types must not be used with the property medium** because medium describes only physical objects.

We recommend to **use a controlled vocabulary**. If no formal controlled vocabulary exist you should nonetheless handle the media type like another resource.

medium	
	<i>oil on wood</i>
	<i>oil</i>
	<i>wood</i>

as linked data

Language

This properties refers to the language of the intellectual content of the resource.

Guidelines for the creation of language content

For the identification of languages please follow RFC 4646 (<http://www.ietf.org/rfc/rfc4646.txt>) . Best practice would be to **select a value from the three letter language tags of ISO 639** (e.g. <http://www.sil.org/iso639-3/codes.asp>).

title	<i>A great deliverance</i>	
language	ISO 639-3	<i>eng</i>

as linked data

or

title	<i>A great deliverance</i>	
language		
	RFC 4646	<i>eng</i>

as linked data

If the content is in **more than one language**, the property should be repeated.

title	<i>Charlie Wilson's War</i>	
language	ISO 639-3	<i>eng</i>
language	ISO 639-3	<i>hun</i>
language	ISO 639-3	<i>tur</i>

as linked data

But if **every language version has it's own identifier**, they have to be treated like single resources.

Video1

title	<i>Medieval helpdesk with English subtitles</i>	
identifier	<i>http://www.youtube.com/watch?v=pQHX-SjgQvQ&feature=player_embedded</i>	
isVersionOf	<i>Video2</i>	
language	ISO 639-3	<i>nor</i>
language	ISO 639-3	<i>eng</i>

Video2

title	<i>Book help (better version)</i>	
identifier	<i>http://www.youtube.com/watch?v=UOorZQLsmuA&feature=related</i>	
isVersionOf	<i>Video1</i>	
language	ISO 639-3	<i>nor</i>

as linked data

You could also use plain text

title	<i>The Power of Orange Knickers</i>
language	<i>English</i>

as linked data

or create your own language vocabulary.

title	<i>The Power of Orange Knickers</i>
language	<i>english</i>

identifier	<i>english</i>
label	<i>English</i>
ISO 639-1	<i>en</i>
ISO 639-3	<i>eng</i>

as linked data

Identifiers

Identifier

An identifier is an unambiguous reference to a resource. Examples of formal identification systems include the Uniform Resource Identifier (URI) - including the Uniform Resource Locator (URL), the Digital Object Identifier (DOI) or the International Standard Book Number (ISBN).

BibliographicCitation

- dcterms:bibliographicCitation

BibliographicCitation is a bibliographic reference to a resource identifying the resource by bibliographic details. Typically the described resource is the child in a parent/child relationship where the bibliographicCitation is not describing the relationship but the location of the described resource within the parent resource (e.g. the bibliographic citation of an article consists of the name of a journal as well as the number of volume, issues and even page references). For the description of parent/child relations see hasPart or isPartOf. BibliographicCitation may only be used to describe bibliographic resources like books, articles or other documentary resource.

Guidelines for the creation of identifier content

Best practice is to **declare the identification system** from which an identifier is selected.

title	<i>What's a URI and why does it matter?</i>	
identifier	<i>http://www.ltg.ed.ac.uk/~ht/WhatAreURIs/</i>	URI

as linked data

Identifiers should be selected from formal identification systems as above but can also be **local identifiers** as long as there is a proper declaration of these.

title	<i>Small and medium sized companies in Kathmandu</i>	
identifier	<i>03KTM147</i>	local ID

as linked data

Note that the identifier of the description of a resource (e. g. of the metadata record) is not the same as the identifier of the resource itself.

metadata record		
created	2010	W3CDTF
identifier	013234098	database ID
my Video		
title	Medieval helpdesk with English subtitles	
created	2007	W3CDTF
identifier	http://www.youtube.com/watch?v=pQHx-SjgQvQ&feature=player_embedded	URI

as linked data

If no identifier from a formal identification system exist, the identifier can be generated by a **bibliographic citation**. The bibliographic citation can be created as **text citation**,

title	Prototyping Digital Library Technologies in zetoc
bibliographicCitation	Lecture Notes in Computer Science 2458, 309-323 (2002)

as linked data

or as **machine readable citations**.

title	Prototyping Digital Library Technologies in zetoc
bibliographicCitation	&ctx_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.jtitle=Lecture Notes in Computer Science&rft.volume=2458&rft.spage=309^^info:ofi/fmt:kev:mtx:ctx

as linked data

You can also **structure the bibliographic information**.

title	My first article about metadata	
identifier		
	journal title	My Favorite Journal
	volume	3
	issue	2
	start page	14
	date	2010

as linked data

For additional information on bibliographicCitation see "Guidelines for Encoding Bibliographic Citation Information in Dublin Core Metadata" (<http://dublincore.org/documents/dc-citation-guidelines/index.shtml>) .

Descriptions

Description

This property refers to the description of the content of a resource. The description is a potentially rich source of indexable terms and assist the users in their selection of an appropriate resource. To refine the character of a description use abstract or tableOfContent.

Abstract

This property is used when the description of a resource is a formal abstract.

TableOfContent

This property is used when the description of a resource is a structured list of the contents of a resource.

Guidelines for the creation of descriptions

A description may be a **free text account**,

title	<i>Bugs from New Zealand</i>
description	<i>A box of ten bugs collected in New Zealand between 1845 and 1846</i>

as linked data

an **abstract**,

title	<i>The Foundations of Programm Verification</i>
abstract	<i>This revised edition provides a precise mathematical background to several program verification techniques. It concentrates on those verification methods that have now become classic, such as the inductive assertions method of Floyd, the axiomatic method of Hoare, and Scott's fixpoint induction. The aim of the book is to present these different verification methods in a simple setting and to explain their mathematical background. In particular the problems of correctness and completeness of the different methods are discussed in some detail and many helpful examples are included.</i>

as linked data

a **table of contents**,

title	<i>Remains of Claire Klawitter</i>
table of content	<i>Diary 1822 - 1824 -- 20 pictures of Indian farmers -- 5 letters to Rudi Ratlos -- 1 map of North India</i>

as linked data

or a **reference to a description**.

title	<i>Thriller</i>
description	http://en.wikipedia.org/wiki/Thriller_(album)

as linked data

You can give some **information about the description you refer to**.

title	<i>Coriandrum sativum</i>	
description	http://en.wikipedia.org/wiki/Coriander	
	title	<i>Coriander</i>
	contributor	<i>Wikipedia</i>

as linked data

If **descriptions in different languages** exist, the property should be repeated with language tags.

title	<i>Delvig and Kjuhelbeker</i>	
abstract	<i>The book is a collection of works of two poets - contemporaries and friends of Pushkin - A. A. Delvig and V. K. Kjuhelbeker. It includes poems and prosa by Kjuhelbeker, parts of his diary, the poem Jurij and Xenia and reviews by Delvig. The attachment presents some retrospections to Delvig and Kjuhelbeker. A detailed biographic description tells us something about the life of the poets.</i>	eng

abstract	<i>Сборник впервые объединяет произведения двух поэтов - современников и друзей Пушкина - А. А. Дельвига и В. К. Кюхельбекера. Наряду со стихотворениями в книгу включены проза Кюхельбекера, фрагменты из его дневника, поэма "Юрий и Ксения", а также рецензии Дельвига. В "Приложении" печатаются воспоминания о Дельвиге и Кюхельбекере. Подробные биографические очерки рассказывают о жизненном пути поэтов.</i>	rus
-----------------	--	-----

as linked data

Subject

The property subject represents a relationship between a resource and another resource which is a topic of the first resource resp. describes the intellectual content of the first resource. If the topic of the resource has a spatial or temporal character, use coverage, spatial or temporal.

Guidelines for describing the subject of a resource

To express the topic of a resource we recommend to **use a URI representing another resource describing this topic**,

title	<i>Inviato alla Biennale : Venezia, 1949 - 2009</i>	
subject	http://www.labiennale.org/en/Home.html	
	title	<i>La Biennale di Venezia</i>

as linked data

or a URI representing the value of a controlled vocabulary,

- like **entries from authority file systems** (e.g. the Library of Congress Subject Headings (<http://id.loc.gov/authorities>) or the authority files of the German National Library (<http://d-nb.info/gnd/>) , etc.),

title	<i>My Winter Wonderland</i>	
subject	http://id.loc.gov/authorities/sh88004323#concept	
	label	<i>Cross-country skiing--Skating</i>

as linked data

- like **entries from classification systems** ((e.g. the Dewey Decimal Classification (<http://www.oclc.org/dewey/>) , or Linnaean taxonomy, etc.).

title	<i>Transports in Kazakhstan 2000 - 2010</i>
subject	<i>W03_7</i>
subject	<i>G06_3</i>

label	<i>W03.7</i>
name	<i>Freight Transport</i>
broader	<i>W03</i>
narrower	<i>W03.72</i>
narrower	<i>W03.75</i>

label	<i>G06_3</i>
name	<i>Kazakhstan</i>

broader	<i>G06</i>
narrower	<i>G06_31</i>
narrower	<i>G06_35</i>

as linked data

You may also use plain text. But if you need **more than one entry to describe the content** you should repeat the property.

title	<i>How to get an aircraft</i>
subject	<i>aircraft</i>
subject	<i>leasing</i>

as linked data

If you want to use a keyword or keyphrase in **different languages**, you should use language tags.

title	<i>KONSTITUCJA RZECZYPOSPOLITEJ POLSKIEJ</i>	
subject		
	<i>Rzeczpospolita Polska</i>	pol
	<i>Republic of Poland</i>	eng

as linked data

If the **subject is a person or organization** you should use names from formal name authorities (e.g. from the Library of Congress Name Authority Headings [1] (<http://id.loc.gov/authorities>) , or from the Virtuell International Authority File [2] (<http://www.viaf.org/>)).

title	<i>Candle in the wind</i>	
subject	gnd	<i>118583549</i>
	family name	<i>Monroe</i>
	given name	<i>Marilyn</i>
	born	<i>1926</i>
	died	<i>1962</i>

as linked data

Coverage

Coverage

The property coverage describes a relationship between a resource and another resource which represents the extent or scope of the content of the first resource. This includes the spatial locations (a place name or geographic co-ordinates), temporal periods (a period label, a date or a date range), or jurisdictions (states, counties, or other administrative entities). If you want to make a distinction between the temporal or spatial character of the content use temporal or spatial.

Temporal

This property describes the relationship between a resource and another resource which represent the temporal characteristics of the intellectual content of the first resource expressed by period labels or date encoding. If you want to describe date of the lifecycle of a resource use the date properties.

Spatial

This property describes the relationship between a resource and another resource which represents spatial characteristics of the intellectual content of the first resource expressed by geographic names, latitude/longitude, or other established georeferencing.

Guidelines for describing the coverage, spatial or temporal character of a resource

To describe the **temporal** characteristic of a resource

you may use plain text,

title	<i>Transports in Kazakhstan 2000 - 2010</i>
coverage	<i>2000 - 2010</i>

as linked data

or **structure your entry** using dates,

title	<i>Transports in Kazakhstan 2000 - 2010</i>		
temporal			
	start	<i>2000</i>	<i>W3CDTF</i>
	end	<i>2010</i>	<i>W3CDTF</i>

as linked data

or **period labels**.

title	<i>Analysis of rocks collected in Perth</i>	
temporal		
	start	<i>Cambrian period</i>
	scheme	<i>Geological timescale</i>
	name	<i>Phanerozoic Eon</i>

as linked data

Further information on encoding temporal characteristics you will find in the DCMI Period Encoding Scheme (<http://dublincore.org/documents/dcmi-period/>) .

To describe the **spatial** character of a resource, you could use plain text,

title	<i>Analysis of rocks collected in Perth</i>
coverage	<i>Perth, W. A.</i>

as linked data

or express it by **georeferencing**,

title	<i>Analysis of rocks collected in Perth</i>	
spatial		
	east	<i>115.85717</i>
	north	<i>-31.95301</i>
	name	<i>Perth, W. A.</i>

as linked data

or reference to a formal encoding.

title	<i>The growth of trees in the subtropical highlands</i>	
spatial		
	label	<i>Cwb</i>
	source	<i>Köppen-Geiger Climate Classification</i>
	main Climates	<i>warm temperate</i>
	precipitation	<i>winter dry</i>
	temperature	<i>warmest month averaging below 22°C</i>

as linked data

Further information on encoding spatial characteristics you will find in the DCMI Box Encoding Scheme (<http://dublincore.org/documents/dcmi-box/>) and the DCMI Point Encoding Scheme (<http://dublincore.org/documents/dcmi-point/>) .

Dates

Date

The property date refers to a description of any dates or ranges in the lifecycle of a resource and is typically associated with the creation or availability. If the distinction between different sorts of date is necessary, the following subproperties should be used. If a date is describing the content of a resource the properties coverage or temporal have to be used.

Created

This property refers to a description of the date or range of the creation of a resource. According to the one-to-one principle this has to be the creation date of the resource being described and not the creation date of any other resource from which the described resource derives (e.g. a former version or a superior resource). So a resource is created only once, every other date of creation belongs to another resource that has to be described on its own.

Issued

This property refers to a description of the date of the formal issuance resp. publication of a resource. A resource is issued only once, every other issuance belongs to another resource that has to be described on its own. If the issuance of a resource is not formal the property "available" should be used.

Available

This property refers to a description of the date a resource did become or will become available. A resource becomes available only once, every other availability belongs to another resource that has to be described on its own. If the availability of a resource starts with the formal issuance resp. publication use "issued".

Modified

This property refers to a description of the date a resource was changed. You may record every date a resource was modified by repeating this property or record only one date (this should be the last one).

Valid

This property refers to a description of the date or range a resource is, was or will be valid. This property should be used if a resource is only valid resp. relevant until a particular date.

DateCopyrighted

This property refers to a description of the date or range of the copyright of the resource.

DateSubmitted

This property refers to a description of the date a resource was submitted (e.g. a thesis at a university department, an article at the editorial board of a journal, etc.).

DateAccepted

This property refers to a description of the date a resource was accepted (e.g. a thesis by a university department, an article by the editorial board of a journal, etc.)

Guidelines for the creation of content for dates

For the structure of date properties we recommend the usage of the **W3CDTF profile of ISO 8601** [W3CDTF]. It allows to sort search results by date and facilitates the merging of metadata of different applications.

You should use this encoding for **a point in time**

created	2003-04-10	W3CDTF
----------------	------------	--------

as linked data

but must not use it with **a range**,

valid	2007-05-06/2007-07-15
--------------	-----------------------

as linked data

or when the date is located **before the common area**.

created	-500	gYear
----------------	------	-------

as linked data

If you need to **structure range data**, make a distinction of start and end date.

date			
	start	2007-05-06"	W3CDTF
	end	2007-07-15	W3CDTF

as linked data

If the **complete date is unknown** you should use

month and year

available	2006-07	W3CDTF
------------------	---------	--------

as linked data

or only the year

issued	2009	W3CDTF
---------------	------	--------

as linked data

If **more than one date of the same type** (e.g. modified) is recorded, the property must be repeated.

modified	2009-12-22	W3CDTF
modified	2010-01-08	W3CDTF
modified	2010-02-15	W3CDTF

as linked data

Since a resource has only one date of creation, issuance, availability and/or copyright you may repeat the properties created, issued, available and dateCopyrighted only if you want to **provide the same date in another structure**.

created	1752	W3CDTF
created	<i>probably after 1752</i>	

as linked data

If the described date is **only approximately known**, you may use plain text,

created	<i>aprox. 500 B.C.</i>
----------------	------------------------

as linked data

or describe it.

date		
	year	500
	qualifier	<i>approx.</i>
	epoch	<i>B.C.</i>

as linked data

Another date of creation, issuance, availability and/or copyright however belongs to another resource that has to be described on its own. The relation of both resources may be described by one of the relation properties or by source.

title	<i>Population estimates in Scandinavia</i>		
created	2004	W3CDTF	
source			
	title	<i>World health report 2002 statistical annex</i>	
	created	2002	W3CDTF

as linked data

Source and Relations

Relation

Relation represents the relationship between the described resource and another resource, that is related to the described resource in some way. Such relationship may be expressed reciprocally but this is not required and depends on the sort of relation. If the relation shall be specified more precicely use one of the following properties.

Source

Source describes the relationship between the described resource and another resource, from which the described resource is derived in whole or in part (e.g. data of a climate centre are the source of a forecast, a book or journal is the source of a scan, etc.)

IsPartOf

This property describes the relationship between the described resource and another resource of which the described resource is a physical or logical part (e.g. a painting as part of a collection, an article as part of a journal, etc.). The described resource is like a "child" in a hierarchical or "parent/child" relationship. For the reciprocal statement use hasPart.

HasPart

This property describes the relationship between the described resource and another resource which is a physical or logical part of the described resource (e.g. the described resource is a collection of paintings, or a journal with different articles, etc.). The described resource is like the "parent" in a hierarchical or "parent/child" relationship. For the reciprocal statement use isPartOf.

IsVersionOf

This property describes the relationship between the described resource and another resource, that is a former version, edition or adaptation of the described resource (e.g. the described resource is the revision of a book, or another recording of a song, etc.). Another version implies changes in the content of a resource. For resources with different formats use isFormatOf. For the reciprocal statement use hasVersion.

HasVersion

This property describes the relationship between the described property and another property, that is a later version, edition or adaptation of the described resource (e.g. the described resource is the older version of a revised book, or of a song, etc.). Another version implies changes in the content of a resource. For resources with different formats use hasFormat. For the reciprocal statement use isVersionOf.

IsFormatOf

This property describes the relationship between the described resource and another resource, that is a former version of the described resource with the same intellectual content but presented in another format (e.g. the described resource is the microfilm version of a printed book, or the pdf version of a doc document). For intellectual changes between resources use isVersionOf. For the reciprocal statement use hasFormat.

HasFormat

This property describes the relationship between the described resource and another resource, that is a later version of the described resource with the same intellectual content but presented in another format (e.g. the described resource is a printed book that is also available as a microfilm, or a doc document that is also available as pdf). For intellectual changes between resources use hasVersion. For the reciprocal statement use isFormatOf.

Replaces

This property describes the relationship between the described resource and another resource, that has been supplanted, displaced or superseded by the described resource. It is used for the valid version in chain of versions (e.g. the described resource is the the last draft of a contract, or the current version of guidelines). For the reciprocal statement use isReplacedBy.

IsReplacedBy

This property describes the relationship between the described resource and another resource, that supplants, displaces or supersedes the described resource. It is used, when in chain of versions only one version is valid (e.g. the described resource is one of the former drafts of a contract, or a former version of guidelines). For the reciprocal statement use replaces.

Requires

This property describes the relationship between the described resource and another resource supporting the function, delivery or coherence of the content of the described resource (e.g. the described resource is an application that can be used only with a particular software, or hardware). For the reciprocal statement use isRequiredBy.

IsRequiredBy

The described resource is necessary for the function, delivery or coherence of the content of the resource the property references to (e.g. the described resource is a software or hardware necessary to use a particular application). For the reciprocal statement use requires.

References

This property describes the relationship between the described resource and another resource that is cited, referenced, or otherwise pointed to by the described resource (e.g. the described resource is an article citing a book, or an interview pointing to a play). For the reciprocal statement use isReferencedBy.

IsReferencedBy

This property describes the relationship between a resource and another resource that points to the described resource by citation, acknowledgement, etc (e.g. the described resource is a book cited in an article, or a play pointed to in an interview, etc.). For the reciprocal statement use references.

ConformsTo

This property describes the relationship between a resource and an established standard, to which the described resource conforms (e.g. a metadata record that conforms to the RDA standard, or a pipe that conforms to ISO 3183, etc.)

Guidelines for the creation of content for relations and source

You may refer to the related resource by plain text or by a URI representing the related resource. If you use plain text, you should **use a formal citation**.

issued	2009	W3CDTF
isFormatOf		
	<i>Eike von Repgow: Sachsenspiegel, Aufss neue vbersehen mit Summarijs vnd Additionen ...; Leipzig 1561/1563</i>	

as linked data

However recommended best practice is to **use an identifier instead of text**,

conformsTo	http://www.w3.org/2001/XMLSchema	-
-------------------	---	---

as linked data

or to **describe the related resources like another resource**.

references			
	creator	<i>Black, Carl</i>	
	contributor	<i>White, Stuart</i>	
	title	<i>Black and White</i>	
	date	1988	W3CDTF

as linked data

If there is **more than one relation of the same sort** you have to repeat the property:

requires	
	<i>audio</i>
	<i>video</i>

as linked data

If **both resources of a relationship are described** the relation could be expressed reciprocally whereupon reciprocity could be generated automatically

identifier	<i>mySong1</i>	
title	<i>Candle in the wind</i>	
issued	<i>1973</i>	W3CDTF
description	<i>Portayal of the life of Marilyn Monroe</i>	
hasVersion	<i>mySong2</i>	

as linked data

identifier	<i>mySong2</i>	
title	<i>Candle in the wind</i>	
alternative	<i>Goodbye England's Rose</i>	
issued	<i>1997</i>	W3CDTF
description	<i>Tribut to the dead princess of Wales</i>	
isVersionOf	<i>mySong1</i>	

as linked data

Rights

Rights

The rights property represents the relationship between a resource and information about rights held in and over this resource. This includes information like access rights, Intellectual Property Rights (IPR), copyrights, references to legal documents describing how to use a resource, etc. To specify rights more precicely use `accessRights` or `license`.

AccessRights

This property represents the relationship between a resource and information about who can access a resource or an indication of its security status. Access rights provides information about restrictions to view, search or use a resource based on attributes of the resource itself or the category of user.

License

This property represents the relationship between a resource and a legal document giving official permission to do something with the resource (e.g. an otherwise free resource may not be used for reproduction within commercial applications). Examples of such licenses you will find at <http://creativecommons.org/>.

Guidelines for the creation of rights content

A rights statement may be **a text**,

title	<i>Data from my last evaluation</i>
accessRights	
	<i>My colleagues only</i>

as linked data

or a **URI** referencing to formal rights information,

title	<i>You and me</i>
rights	http://creativecommons.org/licenses/by/3.0/legalcode

as linked data

or a **combination of both**.

title	<i>GeoNetwork - Geographic Metadata Catalog</i>
license	http://www.gnu.org/licenses/gpl.html
	<i>GNU General Public License</i>

as linked data

If there are no formal rights statements to use you may also **create your own rights statement**.

title	<i>Diaries of Juanita Ramirez</i>
rights	<i>accessConditions</i>

identifier	<i>accessConditions</i>
title	<i>Access to my stuff</i>
description	<i>Resources under this right can only be read, searched and used by members of the myProject</i>

as linked data

Special properties for the description of education material

Audience

The property audience represents the relationship between a resource and the class of persons for whom the resource is intended or useful (e.g. the resource is a textbook for psychologists, etc.). To specify an audience more precisely use mediator or educationLevel.

Mediator

This property represents the relationship between a resource and the class of persons who mediate access to the resource and for whom it is intended or useful. This might be teachers, parents etc. (e.g. teachers are mediators for a resource intended to be used in elementary school lessons)

EducationLevel

This property refers to information about the progress of an audience through the educational or training context, for which the described resource is intended (e.g. the resource is an English workbook for students of the 4th - 5th grade).

InstructionalMethod

This property refers to the process used to engender knowledge, attitudes and skills, that the described resource is designed to support. Typically it includes ways of presenting instructional materials or conducting instructional activities, patterns of learner-to-learner and learner-to-instructor interactions, and mechanisms by which group and individual levels of learning are measured. Instructional methods include all aspects of the instruction and learning processes from planning and implementation through evaluation and feedback.

Guidelines for the creation of content for properties describing education material

You should use formal or informal **controlled vocabularies**. Though none are registered by DCMI, implementors are encouraged to develop local lists of values, and to use them consistently.

title	<i>Advances Physics</i>
audience	
	<i>elementary school pupils</i>

as linked data

mediator	
	<i>schoolteacher</i>

as linked data

educationLevel	
	<i>3rd - 4th grade</i>

as linked data

InstructionalMethod	
	<i>experimental learning</i>

as linked data

Special properties for the description of collections

AccrualMethod

This property refers to the method by which items are added to a collection.

AccrualPeriodicity

This property refers to the frequency with which items are added to a collection.

AccrualPolicy

This property refers to the policy governing the addition of items to a collection.

Guidelines for the creation of content for properties describing collections

Resources described by these properties **have to be collections**. We recommend to use values of formal or informal **controlled vocabularies**.

title	<i>Pottery in Scandinavia in the second half of 19th century</i>
accrualMethod	
	<i>purchase</i>

as linked data

accrualPeriodicity	
	<i>irregular</i>

as linked data

accrualPolicy	
	<i>Objects of this collection have to be Scandinavian ceramics from 1940s to 1999s.</i>

as linked data

Provenance

This property refers to a description of changes in ownership and custody. The statement should include any changes of the resource that are significant for its authenticity, integrity and interpretation.

Guidelines for the creation of content for provenance

The description of the provenance of a resource includes all changes made to a resource.

The provenance can be **described by text**,

title	<i>Luxor Obelisk</i>
provenance	
	<i>Originally located at the entrance to the Luxor temple the obelisk came to Paris in 1836 as a gift by Muhammad Ali Pasha.</i>

as linked data

or **described like another resource**.

title	<i>The flea circus</i>	
provenance		
	ownedBy	<i>1829 - 1833; Jim Button</i>
	ownedBy	<i>1833 - 1915; My Library</i>
	ownedBy	<i>since 1915; Flea Academy</i>

as linked data

Retrieved from "<http://colab.mpdl.mpg.de/mediawiki/CreatingMetadata>"

- This page was last modified 13:07:40, 2010-10-11.
- Content is available under Creative Commons Attribution 2.0.

PublishingMetadata

From MPDLMediaWiki

[Go to UsingDC](#) | [CreatingMetadata](#)

How to use DCMI Metadata as linked data

Contents

- 1 About the linked data examples
- 2 Properties of the terms namespace
 - 2.1 Properties of the terms namespace used only with literal values
 - 2.1.1 dcterms:alternative
 - 2.1.2 dcterms:available
 - 2.1.3 dcterms:bibliographicCitation
 - 2.1.4 dcterms:created
 - 2.1.5 dcterms:date
 - 2.1.6 dcterms:dateAccepted
 - 2.1.7 dcterms:dateCopyrighted
 - 2.1.8 dcterms:dateSubmitted
 - 2.1.9 dcterms:identifier
 - 2.1.10 dcterms:issued
 - 2.1.11 dcterms:modified
 - 2.1.12 dcterms:title
 - 2.1.13 dcterms:valid
 - 2.2 Properties of the terms namespace used only with non-literal values
 - 2.2.1 dcterms:accrualMethod
 - 2.2.2 dcterms:accrualPeriodicity
 - 2.2.3 dcterms:accrualPolicy
 - 2.2.4 dcterms:accessRights
 - 2.2.5 dcterms:audience
 - 2.2.6 dcterms:conformsTo
 - 2.2.7 dcterms:contributor
 - 2.2.8 dcterms:coverage
 - 2.2.9 dcterms:creator
 - 2.2.10 dcterms:educationLevel
 - 2.2.11 dcterms:extent
 - 2.2.12 dcterms:format
 - 2.2.13 dcterms:hasFormat
 - 2.2.14 dcterms:hasPart
 - 2.2.15 dcterms:hasVersion
 - 2.2.16 dcterms:instructionalMethod
 - 2.2.17 dcterms:isFormatOf
 - 2.2.18 dcterms:isPartOf
 - 2.2.19 dcterms:isReferencedBy
 - 2.2.20 dcterms:isReplacedBy
 - 2.2.21 dcterms:isRequiredBy
 - 2.2.22 dcterms:isVersionOf
 - 2.2.23 dcterms:language
 - 2.2.24 dcterms:license
 - 2.2.25 dcterms:mediator
 - 2.2.26 dcterms:medium
 - 2.2.27 dcterms:provenance
 - 2.2.28 dcterms:publisher
 - 2.2.29 dcterms:references
 - 2.2.30 dcterms:relation
 - 2.2.31 dcterms:replaces

- 2.2.32 dcterms:requires
 - 2.2.33 dcterms:rights
 - 2.2.34 dcterms:rightsHolder
 - 2.2.35 dcterms:source
 - 2.2.36 dcterms:spatial
 - 2.2.37 dcterms:subject
 - 2.2.38 dcterms:temporal
 - 2.2.39 dcterms:type
- 2.3 Properties of the terms namespace, that may be used with literal or non-literal values
 - 2.3.1 dcterms:abstract
 - 2.3.2 dcterms:description
 - 2.3.3 dcterms:tableOfContents
- 3 Legacy namespace
 - 3.1 Properties of the legacy namespace
 - 3.1.1 dc:contributor
 - 3.1.2 dc:coverage
 - 3.1.3 dc:creator
 - 3.1.4 dc:date
 - 3.1.5 dc:description
 - 3.1.6 dc:format
 - 3.1.7 dc:identifier
 - 3.1.8 dc:language
 - 3.1.9 dc:publisher
 - 3.1.10 dc:subject
 - 3.1.11 dc:rights
 - 3.1.12 dc:title
 - 3.1.13 dc:type

About the linked data examples

To present these examples in a concise form we use the Turtle syntax for RDF (<http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/>). The examples therefore appear in code lines such as:

```
@prefix ex: <http://www.example.com/>.
ex:aResource ex:aProperty "An RDF Literal" .
```

or

```
@prefix ex: <http://www.example.com/>.
ex:aResource ex:aProperty ex:anotherResource .
ex:anotherProperty "An RDF Literal"@en .
```

Within the examples we use different namespaces. The following prefixes are used to specify the namespaces we use for our examples:

For the **Dublin Core** namespaces we use:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> (DCMI legacy namespace used for the 15 core properties)
@prefix dcterms: <http://purl.org/dc/terms/> (DCMI terms namespace of the DCMI terms - properties, classes and datatypes)
@prefix dctype: <http://purl.org/dc/dcmitype/> (DCMI type namespace of the DCMI classes of types)
```

Further namespaces used are:

```
@prefix ex: http://www.example.org/ (an exemplary namespace)
@prefix xsd: http://www.w3.org/2001/XMLSchema# (namespace of the XML Schema language)
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# (namespace of the rdf vocabulary)
@prefix rdfs: http://www.w3schools.com/RDF/rdf-schema.xml (namespace of the rdf schema vocabulary)
@prefix foaf: http://xmlns.com/foaf/0.1/ (namespace of the "Friend of a Friend" vocabulary)
@prefix gnd: http://d-nb.info/gnd/ (namespace of the authority files of the German National Library)
@prefix mime: http://purl.org/NET/mediatypes/ (namespace for MIME types vocabulary)
@prefix skos: http://www.w3.org/2004/02/skos/core# (namespace of the SKOS vocabulary)
```

Properties of the terms namespace

The main difference between the legacy namespace and the terms namespace is the definition of domains and ranges for most terms of the last. The definition of a domain governs the entities for which a property may be used. The definition of a range governs the usage of literal and non-literal values in context with a property.

Properties of the terms namespace used only with literal values

dcterms:alternative

The range for dcterms:alternative is rdfs:Literal. So you can use dcterms:alternative only with literal values

```
ex:myBook dc:terms: title "Passion for Pulses" ;  
          dcterms:alternative "A Feast of Beans, Peas and Lentils from Around the World" .
```

```
ex:myStuff dcterms:title "American Meteorological Association Newsletter" ;  
          dcterms:alternative "AMA Newsletter" .
```

```
ex:myDocument dcterms:title "EU Stability Programm of Belgium"@eng ;  
          dcterms:alternative "Council Opinion on the Updated  
          Stability Programm of Belgium 2009 -  
          2010"@eng ,  
          "Stellungnahme des Rates zum  
          aktualisierten Stabilitätsprogramm  
          Belgiens für 2009 - 2012"@ger .
```

dcterms:available

The range defined for dcterms:available is the class of rdfs:Literal. Values used with this property therefore have to be literal values.

```
ex:myMusic dcterms:available "2006-07"^^dcterms:W3CDTF .
```

dcterms:bibliographicCitation

The range defined for dcterms:bibliographicCitation is the class of rdfs:Literal. Values used with dcterms:bibliographicCitation have to be instances of this class. Therefore this property can only be used with literal values.

Moreover a domain of the class dcterms:BibliographicResource is declared for dcterms:bibliographicCitation. Thus dcterms:bibliographicCitation may only be used describing bibliographic resources.

```
ex:myArticle dcterms:title "Prototyping Digital Library Technologies in zetoc" ;  
          dcterms:bibliographicCitation "Lecture Notes in Computer Science 2458, 309-323 (2002)" .
```

```
ex:myArticle dcterms:title "Prototyping Digital Library Technologies in zetoc" ;  
          dcterms:bibliographicCitation "&ctx_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:journal  
&rft.jtitle=Lecture Notes in Computer Science&rft.volume=2458&rft.spage=309"^^info:ofi/fmt:kev:mtx:ctx .
```

dcterms:created

The range defined for dcterms:created is the class of rdfs:Literal. Values used with this property therefore have to be literal values.

```
ex:myPicture dcterms:created "2003-04-10"^^dcterms:W3CDTF .
```

```
ex:mySculpture dcterms:created "-500"^^xsd:gYear
```

```
ex:myScript dcterms:created "1752"^^dcterms:W3CDTF ,  
          "probably after 1752" .
```



```
ex:mySculpture dcterms:created "approx. 500 B.C."
```

```
ex:myData dcterms:title "Population estimates in Scandinavia" ;  
          dcterms:created "2004-07-19"^^dcterms:W3CDTF ;  
          dcterms:source ex:oldData .  
  
ex:oldData dcterms:title "World health report 2002 statistical annex" ;  
          dcterms:created "2002-09-28"^^dcterms:W3CDTF .
```

dcterms:date

The range defined for dcterms:date is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

dcterms:dateAccepted

The range defined for dcterms:dateAccepted is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

dcterms:dateCopyrighted

The range defined for dcterms:dateCopyrighted is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

dcterms:dateSubmitted

The range defined for dcterms:dateSubmitted is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

dcterms:identifier

The range defined for dcterms:identifier is the class of `rdfs:Literal`. Values used with dcterms:identifier have to be instances of this class. Therefore this property can only be used with literal values.

```
ex:myWebsite dcterms:title "What's a URI and why does it matter?" ;  
             dcterms:identifier "http://www.ltg.ed.ac.uk/~ht/WhatAreURIs/"^^dcterms:URI .
```

```
ex:myFile dcterms:title "Small and medium sized companies in Kathmandu" ;  
          dcterms:identifier "03KTM147"^^ex:mylocalID .
```

```
ex:myVideo dcterms:title "Medieval helpdesk with English subtitles" ;  
           dcterms:created "2007"^^dcterms:W3CDTF ;  
           dcterms:identifier "http://www.youtube.com/watch?v=pQHx-SjgQvQ&feature=player_embedded"^^dcterms:URI .  
  
ex:myMetadata dcterms:created "2010"^^dcterms:W3CDTF ;  
              dcterms:identifier "013234098"^^ex:myDatabaseIdentifier .
```

dcterms:issued

The range defined for dcterms:issued is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

```
ex:myBook dcterms:issued "2009"^^dcterms:W3CDTF .
```

dcterms:modified

The range defined for dcterms:modified is the class of `rdfs:Literal`. Values used with this property therefore have to be literal values.

```
ex:mySoftware dcterms:modified "2009-12-22"^^dcterms:W3CDTF ,  
                               "2010-01-08"^^dcterms:W3CDTF ,  
                               "2010-02-15"^^dcterms:W3CDTF .
```

dcterms:title

The range for dcterms:title is `rdfs:Literal`. So you can use dcterms:title only with literal values.

```
ex:myFurniture dcterms:title "Alvar Aalto Chair No. 66" .
```

```
ex:mySong dcterms:title "Autumn Leaves",  
                        "The Dead Leaves" .
```

```
ex:myPainting dcterms:title "La Joconde"@fre ,  
                           "Mona Lisa"@eng ,  
                           "La Gioconda"@ita .
```

```
ex:myData dcterms:title "Data from a survey about the usage of metadata" .
```

dcterms:valid

The range defined for `dcterms:valid` is the class `rdfs:Literal`. Values used with this property therefore have to be literal values.

```
ex:myDraft dcterms:valid "2007-05-06/2007-07-15" .
```

Properties of the terms namespace used only with non-literal values

dcterms:accrualMethod

The range of `dcterms:accrualMethod` is the class `dcterms:MethodOfAccrual`. All values used with `dcterms:accrualMethod` have to be instances of this class. Therefore the property may only be used with non-literal values.

The domain of `dcterms:accrualMethod` is the class `dcmitype:Collection`. So `dcterms:accrualMethod` may be used only for the description of collections.

```
ex:myCollection dcterms:title "Pottery in Scandinavia in the second half of 19th century" ;  
                dcterms:accrualMethod [ rdfs:label "purchase" ] .
```

dcterms:accrualPeriodicity

The range of `dcterms:accrualPeriodicity` is the class `dcterms:Frequency`. All values used with `dcterms:accrualPeriodicity` have to be instances of this class. Therefore the property may only be used with non-literal values.

The domain of `dcterms:accrualPeriodicity` is the class `dcmitype:Collection`. So `dcterms:accrualPeriodicity` may be used only for the description of collections.

```
ex:myCollection dcterms:title "Pottery in Scandinavia in the second half of 19th century" ;  
                dcterms:accrualPeriodicity [ rdfs:label "irregular" ] .
```

dcterms:accrualPolicy

The range of `dcterms:accrualPolicy` is the class `dcterms:Policy`. All values used with `dcterms:accrualPolicy` have to be instances of this class. Therefore the property may only be used with non-literal values.

The domain of `dcterms:accrualPolicy` is the class `dcmitype:Collection`. So `dcterms:accrualPolicy` may be used only for the description of collections.

```
ex:myCollection dcterms:title "Pottery in Scandinavia in the second half of 19th century" ;  
                dcterms:accrualPolicy [ rdfs:label "Objects of this collection have to be  
                                       Scandinavian ceramics from 1940s to 1999s." ] .
```

dcterms:accessRights

The range of `dcterms:accessRights` is the class `dcterms:RightsStatement`. Values used with this property have to be instances of the class `RightsStatement` and therefore non-literal values.

```
ex:myDatabase dcterms:title "Data from my last evaluation" ;
              dcterms:accessRights [ rdfs:label "My colleagues only" ] .
```

dcterms:audience

dcterms:audience has a range of the class dcterms:AgentClass. Values used with this property therefore have to be non-literal values.

```
ex:myTutorial dcterms:title "Advanced physic" ;
              dcterms:audience [ rdfs:label "elementary school pupils" ] .
```

dcterms:conformsTo

dcterms:conformsTo has a range of the class dcterms:Standard. Values used with this property therefore have to be non-literal values.

```
ex:myData dcterms:conformsTo <http://www.w3.org/2001/XMLSchema> .
```

dcterms:contributor

The range for dcterms:contributor is dcterms:Agent. All values used with this property have to be instances of the class [dcterms:Agent] .

dcterms:contributor must not be used with literal values.

You may use dcterms:contributor only with non-literal values.

```
ex:myMusic dcterms:contributor gnd:135066719 .
gnd:135066719 foaf:familyName "Elliott" ;
              foaf:givenName "Missy" ;
              foaf:nick "Missy E" .
```

dcterms:coverage

dcterms:coverage has a range of the class dcterms:LocationPeriodJurisdiction. All values used with dcterms:coverage have to be instances of this class. Therefore the property must only be used with non-literal values.

dcterms:creator

The range for dcterms:creator is dcterms:Agent. All values used with this property have to be instances of the class [dcterms:Agent] .

dcterms:creator must not be used with literal values. You may use it only with non-literal values.

```
ex:myBook dcterms:creator _shakespearesName .
_:shakespearesName foaf:familyName "Shakespeare" ;
                   foaf:givenName "William" .
```

dcterms:educationLevel

dcterms:educationLevel has a range of the class dcterms:AgentClass. Values used with this property therefore have to be non-literal values.

```
ex:myTutorial dcterms:title "Advanced physic" ;
              dcterms:educationLevel [ rdfs:label "3rd - 4th grade" ] ;
```

dcterms:extent

The range of dcterms:extent is the class dcterms:SizeOrDuration. All values used with dcterms:extent have to be instances of this class. Therefore the property may only be used with non-literal values.

```
ex:myVideo dcterms:extent [ rdf:value "21 minutes" ] .
```

```
ex:myVideo dcterms:extent [ rdf:value "PT21M"^^xsd:duration ] .
```

dcterms:format

The range of dcterms:format is the class dcterms:MediaTypeOrExtent. All values used with dcterms:format have to be instances of this class. Therefore the property may only be used with non-literal values.

```
ex:myPicture dcterms:format mime:jpeg .
```

dcterms:hasFormat

This property is intended to be used with non-literal values.

dcterms:hasPart

This property is intended to be used with non-literal values.

dcterms:hasVersion

This property is intended to be used with non-literal values.

```
ex:mySong1 dcterms:identifier "mySong1"
           dcterms:title "Candle in the wind" ;
           dcterms:issued "1973"^^dcterms:W3CDTF;
           dcterms:description "Portayal of the life of Marilyn Monroe" ;
           dcterms:hasVersion ex:mySong2
```

dcterms:instructionalMethod

dcterms:instructionalMethod has a range of the class dcterms:MethodOfInstruction. Values used with this property therefore have to be non-literal values.

```
ex:myTutorial dcterms:title "Advanced physic" ;
              dcterms:instructionalMethod [ rdfs:label "experimental learning" ] .
```

dcterms:isFormatOf

This property is intended to be used with non-literal values.

```
ex:myScan dcterms:issued "2009"^^dcterms:W3CDTF ;
           dcterms:isFormatOf [ rdfs:label "Eike von Repgow: Sachsenspiegel, Auffz neue vbersehen mit
                               Summarijs vnd Additionen ...; Leipzig 1561/1563" ] .
```

dcterms:isPartOf

This property is intended to be used with non-literal values.

```
ex:myPainting dcterms:title "Still Life" ;
              dc:creator "Mignon, Abraham" ;
              dcterms:isPartOf http://www.rijksmuseum.nl/meesterwerken
http://www.rijksmuseum.nl/meesterwerken dcterms:title "The Masterpieces special"
              dc:contributor "Rijksmuseum Amsterdam"
```

dcterms:isReferencedBy

This property is intended to be used with non-literal values.

dcterms:isReplacedBy

This property is intended to be used with non-literal values.

dcterms:isRequiredBy

This property is intended to be used with non-literal values.

dcterms:isVersionOf

This property is intended to be used with non-literal values.

```
ex:mySong2 dcterms:identifier "mySong2"
           dcterms:title "Candle in the wind" ;
           dcterms:alternative "Goodbye England's Rose" ;
           dcterms:issued "1997"^^dcterms:W3CDTF ;
           dcterms:description "Tribut to the dead princess of Wales" ;
           dcterms:isVersionOf ex:mySong1 .
```

dcterms:language

The range of dcterms:language is the class dcterms:LinguisticSystem. All values used with dcterms:language have to be instances of this class. Therefore the property may only be used with non-literal values.

```
ex:myBook dcterms:title "A great deliverance" ;
           dcterms:language [ rdf:value "eng"^^dcterms:RFC4646 ] .
```

or

```
ex:myBook dcterms:title "A great deliverance" ;
           dcterms:language <http://www.lexvo.org/page/iso639-3/eng>
```

```
ex:myVideo dcterms:title "Charlie Wilson's War" ;
            dcterms:language <http://www.lexvo.org/page/iso639-3/eng> ,
                             <http://www.lexvo.org/page/iso639-3/hun> ,
                             <http://www.lexvo.org/page/iso639-3/tur> .
```

```
ex:myVideo1 dcterms:title "Medieval helpdesk with English subtitles" ;
             dcterms:identifier "http://www.youtube.com/watch?v=pQHx-SjgQvQ&feature=player_embedded"^^dcterms:URI .
             dcterms:isVersionOf ex:myVideo2 ;
             dcterms:language <http://www.lexvo.org/page/iso639-3/nor> ,
                              <http://www.lexvo.org/page/iso639-3/eng> .
```

```
ex:myVideo2 dcterms:title "Book help (better version)" ;
             dcterms:identifier "http://www.youtube.com/watch?v=UOorZQLsmuA&feature=related"^^dcterms:URI .
             dcterms:hasVersion ex:myVideo1 ;
             dcterms:language <http://www.lexvo.org/page/iso639-3/nor> .
```

```
ex:mySong dcterms:title "The Power of Orange Knickers"
           dcterms:language _:eng

_:eng rdfs:Label "English"
     ex:639-1 "en"
     ex:639-2 "eng"
```

dcterms:license

The range of dcterms:license is the class dcterms:LicenseDocument. Values used with this property have to be instances of the class LicenseDocument and therefore non-literal values.

```
ex:mySoftware dcterms:title "GeoNetwork - Geographic Metadata Catalog" ;
              dcterms:license <http://www.gnu.org/licenses/gpl.html> .

<http://www.gnu.org/licenses/gpl.html> rdfs:label "GNU General Public License" .
```

dcterms:mediator

dcterms:mediator has a range of the class dcterms:AgentClass. Values used with this property therefore have to be non-literal values.

```
ex:myTutorial dcterms:title "Advanced physic" ;
              dcterms:mediator [ rdfs:label "schoolteacher" ] ;
```

dcterms:medium

The range of `dcterms:medium` is the class `dcterms:PhysicalMedium`. All values used with `dcterms:medium` have to be instances of this class. Therefore the property may only be used with non-literal values.

The domain of `dcterms:medium` is the class `dcterms:PhysicalResource`. So `dcterms:medium` may be used only for the description of physical resources.

```
ex:myPainting dcterms:medium _:oilOnWood
_:oilOnWood rdfs:label "oil on wood"
            rdfs:label "oil"
            rdfs:label "wood"
```

dcterms:provenance

The range of the provenance property is the class `dcterms:ProvenanceStatement`. So you may use this property only with non-literal values.

```
ex:myResource dcterms:title "Luxor Obelisk"
              dcterms:provenance [ rdfs:label "Originally located at the entrance to the Luxor temple the
                                   obelisk came to Paris in 1836 as a gift by Muhammad Ali Pasha." ] .
```

```
ex:myBook dcterms:title "The flea circus" ;
          dcterms:provenance _:thisProvenance
_:thisProvenance ex:ownedBy "1829 - 1833; Jim Button" ,
                           "1833 - 1915; My Library" ,
                           "since 1915; Flea Academy" .
```

dcterms:publisher

The range for `dcterms:publisher` is `dcterms:Agent`. All values used with this property have to be instances of the class `[dcterms:Agent]`. **dcterms:publisher must not be used with literal values.** You may use it only with non-literal values.

```
ex:myBook dcterms:publisher gnd: 2125990-2 ,
                              _:kniznajaPalata
gnd:2125990-2 foaf:name "Rossijskaja Gosudarstvennaja Biblioteka ;
                      foaf:homepage "http://www.rsl.ru/" .
_:kniznajaPalata foaf:name "Kniznaja Palata"
```

dcterms:references

This property is intended to be used with non-literal values.

```
ex:myArticle dcterms:references _:articlesReference .
_:articlesReference dc:creator "Black, Carl" ;
                   dc:contributor "White, Stuart" ;
                   dc:title "Black and White"
                   dc:date "1988"^^dcterms:W3CDTF
```

dcterms:relation

This property is intended to be used with non-literal values.

dcterms:replaces

This property is intended to be used with non-literal values.

dcterms:requires

This property is intended to be used with non-literal values.

```
ex:myGame dcterms:requires [ rdfs:label "audio" ] ,
                           [ rdfs:label "video" ] .
```

dcterms:rights

The range of dcterms:rights is the class dcterms:RightsStatement. Values used with this property have to be instances of the class RightsStatement and therefore non-literal values.

```
ex:myPicture dcterms:title "You and me" ;  
dcterms:rights <http://creativecommons.org/licenses/by/3.0/legalcode> .
```

```
ex:myDocuments dcterms:title "Diaries of Juanita Ramirez"  
dcterms:rights _:accessConditions  
_:accessConditions dcterms:title "Access to my stuff"  
dcterms:description "Resources under this right can only be read, searched and  
used by members of the myProject" .
```

dcterms:rightsHolder

The range for dcterms:rightsHolder is dcterms:Agent. All values used with this property have to be instances of the class [dcterms:Agent] .
dcterms:rightsHolder must not be used with literal values. You may use it only with non-literal values.

```
ex:myFilm dcterms:rightsHolder gnd:39454-3 .  
gnd:39454-3 foaf:name "Bundesarchiv Koblenz" ;  
foaf:homepage "http://www.bundesarchiv.de/index.html.de" .
```

dcterms:source

This property is intended to be used with non-literal values.

dcterms:spatial

dcterms:spatial has a range of the class dcterms:LocationPeriodJurisdiction. All values used with dcterms:spatial have to be instances of this class. Therefore the property must only be used with non-literal values.

```
ex:myData dcterms:title "Analysis of rocks collected in Perth" ;  
dcterms:spatial _:thisPlace .  
_:thisPlace ex:east "115.85717" ;  
ex:north "-31.95301" ;  
ex:name "Perth, W. A." .
```

```
ex:myData dcterms:title "The growth of trees in the subtropical highlands" ;  
dcterms:spatial _:Cwb .  
_:Cwb rdfs:label "Cwb"  
dc:source "Köppen-Geiger Climate Classification" ;  
ex:mainClimates "warm temperate" ;  
ex:precipitation "winter dry" ;  
ex:temperature "warmest month averaging below 22°C" .
```

dcterms:subject

This property is intended to be used with non-literal values.

```
ex:myBook dcterms:title "Inviato alla Biennale : Venezia, 1949 - 2009" ;  
dcterms:subject <http://www.labiennale.org/en/Home.html> .  
<http://www.labiennale.org/en/Home.html> dcterms:title "La Biennale di Venezia"
```

```
ex:myVideo dcterms:title "My Winter Wonderland" ;  
dcterms:subject <http://id.loc.gov/authorities/sh88004323#concept> .  
<http://id.loc.gov/authorities/sh88004323#concept> rdfs:label "Cross-country skiing--Skating" .
```

```

ex:myData dcterms:title: "Transports in Kazakhstan 2000 - 2010"
          dcterms:subject ex:W03_7 ,
          dcterms:subject ex:G06_3

ex:W03_7 rdf:type skos:Concept ;
          prefLabel "W03.7" ;
          skos:altLabel "Freight Transport" ;
          skos:broader ex:W03 ;
          skos:narrower ex:W03_72 ,
                        ex:W03_75 .

ex:G06_3 rdf:type skos:Concept ;
          prefLabel "G06_3" ;
          skos:altLabel "Kazakhstan" ;
          skos:broader ex:G06 ;
          skos:narrower ex:G06_31 ,
                        ex:G06_35 .

```

```

ex:myLaw dcterms:title "KONSTYTUCJA RZECZYPOSPOLITEJ POLSKIEJ"
          dcterms:subject _:polska

_:polska rdfs:label "Rzeczpospolita Polska"@pol ,
            "Republic of Poland"@eng .

```

```

ex:mySong dcterms:title "Candle in the wind" ;
          dcterms:subject gnd:118583549 .
gnd:118583549 foaf:familyName "Monroe" ;
              foaf:givenName "Marilyn" ;
              http://rdvocab.info/ElementsGr2/dateOfBirth "1926"^^dcterms:W3CDTF ;
              http://rdvocab.info/ElementsGr2/dateOfDeath "1962"^^dcterms:W3CDTF .

```

dcterms:temporal

dcterms:temporal has a range of the class dcterms:LocationPeriodJurisdiction. All values used with dcterms:temporal have to be instances of this class. Therefore the property must only be used with non-literal values.

```

ex:myData dcterms:title "Transports in Kazakhstan 2000 - 2010"
          dcterms:temporal _:thisPeriod .

_:thisPeriod ex:start "2000"^^dcterms:W3CDTF ;
              ex:end "2010"^^dcterms:W3CDTF .

```

```

ex:myData dcterms:title "Analysis of rocks collected in Perth" ;
          dcterms:temporal _:thisPeriod .

_:thisPeriod ex:start "Cambrian period" ;
              ex:scheme "Geological timescale" ;
              ex:name "Phanerozoic Eon" .

```

dcterms:type

The range of dcterms:type is rdfs:Class. Values used with dcterms:type may only be non-literal values.

```

ex:myPainting dcterms:type dctype:StillImage .

dctype:StillImage rdfs:label "Still Image" .

```

```

ex:myStuff dcterms:type dctype:InteractiveResource ,
              dctype:Text .

dctype:InteractiveResource rdfs:label "Interactive Resource" .

dctype:Text rdfs:label "Text" .

```

```

ex:myStuff dcterms:type _:PCGameType ,
              dctype:Software .

_:PCGameType rdfs:label "PC Game" .

dctype:Software rdfs:label "Software" .

```



```
ex:myEvent dcterms:type _:conference .  
  
_:conference rdfs:label "Conference"@eng ,  
              "Konferenz"@ger ,  
              "съезд"@rus .
```

Properties of the terms namespace, that may be used with literal or non-literal values

dcterms:abstract

There is no range defined for dcterms:abstract. Therefore you can use it either with literal values or with non-literal values.

```
ex:myBook dcterms:title "The Foundations of Programm Verification" ;  
          dcterms:abstract "This revised edition provides a precise mathematical background to  
                           several program verification techniques. It concentrates on those  
                           verification methods that have now become classic, such as the  
                           inductive assertions method of Floyd, the axiomatic method of Hoare,  
                           and Scott's fixpoint induction. The aim of the book is to present these  
                           different verification methods in a simple setting and to explain their  
                           mathematical background. In particular the problems of correctness and  
                           completeness of the different methods are discussed in some detail and  
                           many helpful examples are included." .
```

```
ex:myBook dcterms:title "Delvig and Kjuchelbeker"  
          dcterms:abstract "The book is a collection of works of two poets - contemporaries  
                           and friends of Pushkin - A. A. Delvig and V. K. Kjuchelbeker. It  
                           includes poems and prosa by Kjuchelbeker, parts of his diary, the  
                           poem Jurij and Xenia and reviews by Delvig. The attachment presents  
                           some retrospections to Delvig and Kjuchelbeker. A detailed biographic  
                           description tells us something about the life of the poets"@eng ,  
                           "Сборник впервые объединяет произведения двух поэтов - современников  
                           и друзей Пушкина - А. А. Дельвига и В. К. Кюхельбекера. Наряду со  
                           стихотворениями в книгу включены проза Кюхельбекера, фрагменты из его  
                           дневника, поэма Юрий и Ксения, а также рецензии Дельвига. В Приложении  
                           печатаются воспоминания о Дельвиге и Кюхельбекере. Подробные  
                           биографические очерки рассказывают о жизненном пути поэтов."@rus
```

dcterms:description

There is no range defined for dcterms:description. Therefore you can use it either with literal values

```
ex:myObjects dcterms:title "Bugs from New Zealand" ;  
             dcterms:description "A box of ten bugs collected in New Zealand between 1845 and 1846" .
```

or with non-literal values.

```
ex:myMusic dcterms:title "Thriller" ;  
           dcterms:description <http://en.wikipedia.org/wiki/Thriller_(album)> .
```

dcterms:tableOfContents

There is no range defined for dcterms:tableOfContents. Therefore you can use it either with non-literal values and with literal values.

```
ex:myFile dcterms:title "Remains of Claire Klawitter" ;  
          dcterms:tableOfContent "Diary 1822 - 1824 -- 20 pictures of Indian farmers  
                                -- 5 letters to Rudi Ratlos -- 1 map of North India" .
```

Legacy namespace

For properties of the legacy namespace neither domain nor range is defined. So all properties of the legacy namespace can be used either with literal values or with non-literal values.

Properties of the legacy namespace

dc:contributor

You may use `dc:contributor` either with literal values or with non-literal values.

```
ex:myMusic dc:contributor "Snoop Dogg" .
```

dc:coverage

You may use `dc:coverage` either with literal values or with non-literal values.

```
ex:myData dcterms:title "Transports in Kazakhstan 2000 - 2010"  
dc:coverage "2000 - 2010"
```

```
ex:myData dcterms:title "Analysis of rocks collected in Perth" ;  
dc:coverage "Perth, W. A." .
```

dc:creator

You may use `dc:creator` either with literal values or with non-literal values.

```
ex:myBook dc:creator "Shakespeare, William" .
```

dc:date

You may use `dc:date` either with literal or with non-literal values.

```
ex:myDraft dc:date _:thisValidity .  
_:thisValidity ex:start "2007-05-06"^^dcterms:W3CDTF ;  
ex:end "2007-07-15"^^dcterms:W3CDTF .
```

```
ex:mySculpture dc:date _:mySculptureDate .  
_:mySculptureDate ex:year "500" ;  
ex:qualifier "approx." ;  
ex:epoch "B.C." .
```

dc:description

You may use `dc:description` either with literal values,

```
ex:myHerbs dc:title "Coriandrum sativum"  
dc:description "Coriandrum sativum or Coriander is a herb used in Europe, North Africa  
and Asia. It belongs to the family of Apiaceae and ist growing to 20 inch."
```

or with non-literal values

```
ex:myHerbs dc:title "Coriandrum sativum"  
dc:description http://en.wikipedia.org/wiki/Coriander  
http://en.wikipedia.org/wiki/Coriander dc:title "Coriander"  
dc:contributor "Wikipedia"
```

dc:format

You may use `dc:format` either with literal values or with non-literal values.

```
ex:myPicture dc:format mime:jpeg ,  
"40 x 512 pixels" .
```

dc:identifier

You may use `dc:identifier` either with literal values

```
ex:myVideo dc:title "My first article about metadata"
            dc:identifier "My Favorite Journal 3 (2), 14-25 (2010)" .
```

or with non-literal values

```
ex:myVideo dc:title "My first article about metadata"
            dc:identifier _:myCitation .

_:myCitation ex:jtitle "My Favorite Journal"
              ex:volume "3"
              ex:issue "2"
              ex:spage "14"
              ex:date "2010" .
```

dc:language

You may use dc:language either with literal values or with non-literal values.

```
ex:mySong dc:title "The Power of Orange Knickers"
           dc:language "English"
```

```
ex:mySong dc:title "The Power of Orange Knickers"
           dc:language "eng"^^dcterms:RFC4646 .
```

dc:publisher

You may use dc:publisher either with literal values or with non-literal values.

```
ex:myData dc:creator "Hubble Telescope";
           dc:publisher "University of Nowhere" ,
                       "All Your Data Inc. .
```

dc:subject

There is no range defined for dc:subject. Therefore you can use it either with non-literal values and with literal values.

```
ex:myManual dc:title "How to get an aircraft"
             dc:subject "aircraft" ;
                       "leasing" .
```

dc:rights

You may use dc:rights either with literal values or with non-literal values.

```
ex:myVideo dc:Rights "May be used only by members of the myProject" .
```

```
ex:myBook dcterms:title "News from the South" ;
           dc:rights "http://creativecommons.org/licenses/by-nd/3.0/legalcode"^^dcterms:URI .
           dc:rights "Attribution-NoDerivs 3.0 Unported" .
```

dc:title

It is recommended to use dc:title with literal values. But there are important **uses with non-literal values** as well. To do so you have to use dc:title.

```
ex:myBook dc:title _:thisTitle .

_:thisTitle ex:greekAlphabet "Οιδίπους Τύραννος" ;
            ex:latinAlphabet "Oidipous Tyrannos" .
```

```
ex:myScript1 dcterms:title "Nibelungenlied Handschrift B" ;  
              dc:title _:nibelungenlied .  
  
ex:myScript2 dcterms:title "Nibelungenlied Handschrift C" ;  
              dc:title _:nibelungenlied  
  
_:nibelungenlied skos:prefLabel "Der Nibelunge Not" ;  
                  skos:altLabel "Nibelungenlied" ,  
                                "Nibelungenklage" ,  
                                "Nibelungensage" ;  
                  dcterms:description "The Nibelungenlied exists of 39 aventiuren created between 1180 and 1210" .
```

dc:type

The range of dcterms:type is rdfs:Class. Values used with dcterms:type may be either literal or non-literal values.

```
ex:myEvent dc:type "Conference" .
```

Retrieved from "<http://colab.mpgl.mpg.de/mediawiki/PublishingMetadata>"

Category: KIM

-
- This page was last modified 17:28:33, 2010-09-24.
 - Content is available under Creative Commons Attribution 2.0.

```
-----
Date:      Tue, 12 Oct 2010 17:27:48 +0100
From:      Pete Johnston <Pete.Johnston@EDUSERV.ORG.UK>
Subject:   Re: Using Dublin Core - next generation
To: To:    DC-GLOSSARY@JISCMAIL.AC.UK
```

Just a couple of quick points on the graphs:

1. I notice that you use a colour convention (yellow-ish v green) to distinguish between literal nodes and URI nodes. The usual convention for RDF graphs is to use rectangles for literals, and I think it might be a good idea to follow that?

2. The larger graph includes the triple

```
<http://www.gutenberg.org/files/46/46-h/46-h.htm> dcam:memberOf
<http://purl.org/dcmitype/Text> .
```

While this isn't "wrong", I think it is probably a slightly unusual example and might be confusing? i.e. `dcmitype:Text` is a class and would typically be referenced in an "is-a"/is-instance-of-class relationship (i.e. `rdf:type` or `dcterms:type`).

`dcam:memberOf` is typically used with things which are "sets" but aren't necessarily classes (i.e. "Vocabulary Encoding Schemes")

So I suggest

- replacing `http://purl.org/dc/dcam/memberOf` with `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

And if you do want to illustrate the use of a VES and the property `dcam:memberOf`, then add a new arc, hanging off the concept i.e. add

```
<http://id.loc.gov/authorities/sh85025303#concept>
<http://purl.org/dc/dcam/memberOf>
<http://id.loc.gov/authorities#conceptscheme>
```

3. I think the URI

```
http://en.wikipedia.org/wiki/Charles_dickens
```

identifies a document, i.e. if I do a GET on that URI, the server returns 200 and an HTML page.

But the relator properties are meant to be relations between things and agents, not things and documents - they don't have an `rdfs:range`, but I think that is the intent, anyway.

So I'd suggest using the DBpedia URI for the Person i.e. use

```
http://dbpedia.org/resource/Charles_Dickens
```

instead of `http://en.wikipedia.org/wiki/Charles_dickens`

Terms, DCMI Metadata Terms

The RDF Properties, Classes, Vocabulary Encoding Schemes, and Datatypes declared and maintained by the Dublin Core Metadata Initiative are collectively referred to as DCMI Metadata Terms [1]. Of these term types, only the notion of a Vocabulary Encoding Scheme is unique to DCMI; Properties, Classes, and Datatypes are exactly as defined in RDF (see glossary entry for RDF).

It hasn't always been this way. When Elements were introduced in a 1995 workshop, and Qualifiers in 1997, the W3C specification for RDF did not yet exist. The path from "native Dublin Core" term types to RDF only really ended with the publication of a revised DCMI Abstract Model in 2007 [2]. Starting with the Canberra Qualifiers of 1997 [3] -- Language, Scheme, and Type (Sub-Element) -- and continuing with the transitional Types A and B of 1998 [4], the path to RDF took the following turns:

- The Dublin Core Qualifiers published in 2000 were of two types -- Element Refinements and Encoding Schemes.
- By 2003, Elements began to be called Properties. The type Encoding Scheme was differentiated into Vocabulary Encoding Schemes and Syntax Encoding Schemes. [6]
- In the meantime, 2004 saw the publication of a major revision of RDF and laying the groundwork for modern Semantic Web implementations. [7]
- In the revised DCMI Abstract Model of 2007, Property replaced Element as the designation of choice and was explicitly equated with RDF Property. The term Element Refinement was dropped with the explanation that an Element Refinement was no more than a Property which happened to be a sub-property of another. Syntax Encoding Schemes were declared to be RDF Datatypes. [2]

- [1] <http://dublincore.org/documents/dcmi-terms/>
- [2] <http://dublincore.org/documents/2007/06/04/abstract-model/#sect-5>
- [3] <http://www.dlib.org/dlib/june97/metadata/06weibel.html>
- [4] <http://dublincore.org/documents/1998/10/07/datamodel/index.shtml>
- [5] <http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/>
- [6] <http://dublincore.org/usage/documents/2003/02/07/principles/>
- [7] <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>

Dublin Core, The Dublin Core, Dublin Core Metadata Element Set

"The Dublin Core", also known as the Dublin Core Metadata Element Set, is a set of fifteen "core" elements (properties) for describing resources. The elements are: Creator, Contributor, Publisher, Title, Date, Language, Format, Subject, Description, Identifier, Relation, Source, Type, Coverage, and Rights. The fifteen-element Dublin Core, originally drafted with thirteen elements in 1995, has been formally standardized as ISO 15836:2009, ANSI/NISO Z39.85, and the IETF RFC 5013 (@cites). Metadata based on the fifteen elements is used in countless implementations, and the Dublin Core is one of the top metadata vocabularies in use today, particularly for Semantic-Web and Linked-Data applications. The Dublin Core Metadata Element Set is part of a larger set of DCMI Metadata Terms (see related entry "DCMI Metadata Terms").

The Dublin Core was the first result of a process which engendered a wider "Dublin Core community" and led to the creation of a Dublin Core Metadata Initiative. (See entries for "Dublin Core Metadata Initiative" and "Dublin Core, used as an adjective").

Andy Powell

"Dublin Core (DC) is a metadata standard that provides a vocabulary for describing the "core" attributes of resources, often in the context of resource discovery and/or management."

Dublin Core, used as an adjective

Besides The Dublin Core, Dublin Core Metadata Element Set, and Dublin Core Metadata Initiative (DCMI), described in separate entries, "Dublin Core" is commonly used as an adjective in the following:

- "Dublin Core community". Participants in DCMI mailing lists, task groups, specialist communities, and annual conferences constitute a loosely defined Dublin Core community.
- "Dublin Core metadata". In the late 1990s, "Dublin Core metadata" referred to metadata based on the fifteen-element Dublin Core, has come to refer what might best be described as a "style", "form", or "flavor" of metadata -- a style that has evolved from efforts to put the fifteen elements into the context of a coherent approach to metadata on the World Wide Web generally. As Dublin Core metadata now denotes a type of metadata based on a generic model, it is often pointed out that Dublin Core metadata does not actually require use of The Dublin Core or of any other DCMI metadata terms. The definitive characteristic of the Dublin Core style is the "Dublin Core application profile".
- "Dublin Core application profiles". Analogously to "Dublin Core Metadata", the term "Dublin Core application profiles" referred in circa 2000 through 2004 to any metadata specification that "used" the Dublin Core, usually in combination with other metadata vocabularies. With the development of the DCMI Abstract Model in 2004 through 2007, the term "Dublin Core application profile" was narrowed in scope to refer specifically to application profiles based on the DCMI Abstract Model. Weak uptake of the DCMI Abstract Model, together with the emergence of alternative methods for documenting Linked Data patterns, may be loosening the dependence of the "DC application profile" notion on the DCMI Abstract Model per se in favor of a basis in Linked Data methods more generally.

Dublin Core Metadata Initiative (DCMI)

The Dublin Core Metadata Initiative (DCMI) is a public, not-for-profit organization, incorporated in Singapore, with a mission to develop interoperable metadata standards for a broad range of purposes and business models. DCMI is hosted by the National Library Board of Singapore and supported by institutional members and sponsors. From its start in 1995 with a workshop at OCLC in Dublin, Ohio, through

incorporation in Singapore in 2009, the Dublin Core Metadata Initiative was managed as an activity within OCLC's Office of Research. DCMI publishes metadata specifications, holds annual conferences, and hosts numerous discussion forums.

Dumb-Down Principle

The Dumb-Down Principle, which entered Dublin Core discourse in 1998 [1], has traditionally denoted a principled way of viewing a complex metadata description through the lens of a simpler, human-readable representation, typically Simple Dublin Core.

The meaning of "dumb-down" has evolved over the years:

- In 1998, the "dumb-down procedure" discussed by the Dublin Core Data Model Working Group involved ignoring contextual information such as datatypes and vocabulary encoding schemes and resolving URIs, if possible, to readable value strings.
- By 2000, the notion of "dumb-down" had been extended to include the resolution of "resource" placeholders in RDF triples to an intelligible text representation -- a "simple default name" by which the resource as a whole could be characterized. (This was done by resolving -- if necessary recursively -- triples using the predicate `rdf:value`.) Tools were supposed to "simply ignore any resources originating from intermediate structural nodes in the node-and-arc diagram, and follow the chain of `rdf:value` arcs until they terminate in a character string. The content of that character string is then returned as the value of the Dublin Core element."
- In 2005, the notion of "informed dumb-down" introduced the idea of using the sub-property relationships encoded in RDF schemas to infer statements using more-general properties from statements using more-specific properties [2].

As the notion of Simple Dublin Core becomes relatively less salient in the context of Linked Data, the dumb-down principle is increasingly understood as referring to the more general notion of partial interoperability among imperfectly aligned data sets in an open Web environment on the basis of Semantic Web principles. This notion is less about converting metadata into simpler forms and more about using formal definitions to infer additional information that can be used to align metadata descriptions based on different vocabularies. (See also the glossary entries for "Simple Dublin Core" and "Open World Mindset".)

- [1] <http://dublincore.org/archives/1998/1998-09-24.decisions.html>
- [2] <http://dublincore.org/documents/2005/03/07/abstract-model/>

Namespace, DCMI Namespace, DCMI Namespace Policy

The DCMI Namespace Policy, in 2001 among the first such policies of its kind to be articulated for any vocabulary, declares the principles by which DCMI Metadata Terms are identified and published as an RDF vocabulary.

The DCMI Namespace Policy defines a DCMI Namespace as "a collection of DCMI term URIs where each term is assigned a URI that starts with the same 'base URI'. The 'base URI' is known as the DCMI namespace URI." [1] The policy defines the following four base URIs:

- <http://purl.org/dc/elements/1.1/> for the original fifteen properties of the Dublin Core Metadata Element Set, Version 1.1
- <http://purl.org/dc/dcmitype/> for classes in the DCMI Type Vocabulary
- <http://purl.org/dc/dcam/> for terms used in the DCMI Abstract Model (of which there are currently two)
- <http://purl.org/dc/terms/> for all other DCMI properties, classes, vocabulary encoding schemes, and datatypes

The policy explains how a DCMI namespace URI is used together with a name, such as "extent", to form the URI "http://purl.org/dc/terms/extent".

Like other notoriously polysemous terms, the term "namespace" has been a source of confusion. The Namespace Policy clarifies as follows:

- Even though a DCMI namespace URI is sometimes used in XML formats -- arguably incorrectly -- as an XML namespace URI, a DCMI namespace is not the same as an XML namespace [2]. Technically, an XML namespace is a collection of two-part "expanded names", often abbreviated as "qualified names", or XML QNames, in which the namespace name is bound to a abbreviated prefix. An XML component with the same expanded name or QName can be used in multiple XML formats in the context of substantially different content models. In contrast, an RDF property with a given URI is designed to be interpreted a consistent way independently of the contexts in which it may appear. In RDF syntaxes such as RDF/XML and Turtle, prefixing mechanisms merely provide a way to abbreviate these URIs.
- The grouping of term URIs into DCMI Namespaces is orthogonal to the grouping of terms into sets designed to meet other functional needs, as in various types of vocabularies and formats.

The DCMI Namespace Policy declares guidelines for maintenance changes to DCMI terms: The correction of editorial errata (e.g., updated URLs pointing to documentation external to DCMI) result in no changes to DCMI term URIs, while semantic changes judged likely to have a substantial impact on machine processing will trigger the creation of a new term with a new URI.

In one of the historically earliest illustrations of Linked Data principles, DCMI namespace and term URIs have since 2001 dereferenced to machine-processable DCMI term declarations, so that "clicking on" a URI in a browser will retrieve a term representation in RDF.

- [1] <http://dublincore.org/documents/dcmi-namespace/>
- [2] <http://www.w3.org/TR/xml-names/>

One-to-One Principle (current)

The One-to-One Principle, first formulated in the earliest Dublin Core meeting, dictates that a metadata description should refer to just one resource. This principle was articulated in the recognition that most existing metadata records, in practice, combined descriptions of what might conceptually be seen as multiple distinct entities. For example, metadata records about books routinely included information such as the affiliation of an author (i.e., a property of the author), or the metadata record about a painting (e.g., "Mona Lisa") might include description of the photograph itself (a JPEG image).

The basic idea behind the One-to-One Principle became a fundamental feature of the RDF data model, which required that distinct resources be identified, distinguished, and described separately, as in the case of the original "Mona Lisa", created in 1506 by Leonardo da Vinci, and a photograph of "Mona Lisa" created in 2008 by John Smith. Accepting RDF as the foundational model for metadata allowed Dublin Core descriptions both to respect the One-to-One Principle and to transcend the limitations of "flat" single-resource descriptions.

What constitutes a "resource", hence a meaningful object of description, lies of course in the eye of the beholder. In a simple bibliography, a book may be described as a single resource, whereas a trained library cataloger might perceive the same book as the Expression of a Work, the particular version of which (Manifestation) is available in multiple copies (Items) -- in effect, as four separate resources requiring four separate, though related, metadata descriptions, potentially retrieved from four separate sources. The One-to-One Principle, therefore, is relative to the variety of subjective viewpoints in the world, where some people make distinctions while others do not, and others yet may draw the boundaries differently.

Open World Mindset (a very rough draft!!)

The modern Web environment has created possibilities for knowledge management that are fundamentally new with respect to pre-Web ("normal") information technology.

Traditional information technology environments are "closed worlds" inasmuch their data sources are carefully controlled; data formats are custom-defined for specific applications. Traditional databases, for example, require agreement on a schema as a precondition for storing and querying data. Closed world technologies work efficiently for well-defined, bounded systems.

Since the early 1990s, the Web has placed local sources of information into a new global context, giving rise to what Mike Bergman calls an "open world mindset". [1] The Web provides a context for integrating different sources of content.

@@@ Ideas from Mike Bergman [1] - would need to be summarized, condensed...

- * "By design, tolerance of incomplete information.
- * "Incremental, low-risk means to knowledge systems and management.
- * "Domains can be analyzed and inspected incrementally
- * "Systems designed to integrate information incrementally.
- * "Systems designed to incorporate new information incrementally.
- * "Schema can be incomplete and developed and refined incrementally
- * "Database design and management can be more agile, with schema evolving incrementally.
- * "Lower risk, lower cost, faster deployment, and more agile responsiveness.
- * "The process of describing an open, semantic Web world can proceed incrementally, sequentially asserting new statements or conditions.
- * "Start small, initial focus on a few applications with high returns.
- * "Designed to tolerate incomplete information. Partially known.
- * "Knowledge is never complete
- * "Knowledge is found in structured, semi-structured and unstructured forms.
- * "The data and the structures within these open world frameworks can be used and expressed in a piecemeal or incomplete manner
- * "We can readily combine data with partial characterizations with other data having complete characterizations
- * "Systems built with open world frameworks are flexible and robust; as new information or structure is gained, it can be incorporated without negating the information already resident, and
- * "Open world systems can readily bridge or embrace closed world subsystems.
- * "Reusable and extensible.
- * "Information can be combined about similar objects or individuals even though they have different or non-overlapping attributes.
- * "What is important to describe (the attributes) about certain information also varies by context and perspective.
- * "What distinguishes knowledge from information is that knowledge makes the connections between disparate pieces of relevant information. As these relationships accrete, the knowledge base grows. Again, RDF and the open world approach are essentially "connective" in nature.

"By contrast, systems based on the closed-world assumption are more brittle - new connections and relationships tend to break relational models.

"This makes CWA and its related assumptions a very poor choice when attempting to combine information from multiple sources, to deal with uncertainty or incompleteness in the world, or to try to integrate internal, proprietary information with external data.

"However, many of the new knowledge economy challenges are anything but defined and bounded. These applications all reside in the broad category of knowledge management (KM), and include such applications as data federation, data warehousing, enterprise information integration, business intelligence, competitive intelligence, knowledge representation, and so forth.

"Open world does not necessarily mean open data and it does not mean open source. Open world is simply a way to think

about the information we have and how we act on it. Open world technologies can be applied to internal, closed, proprietary data and structures.

[1] <http://www.mkbergman.com/852/the-open-world-assumption-elephant-in-the-room/>

Resource Description Framework (RDF)

RDF is "a standard model for data interchange on the Web" [1]. First standardized as a W3C Recommendation in 1999 [2], RDF was re-released as a revised W3C Recommendation in 2004 [3]. RDF underpins current approaches to Semantic Web and Linked Data [4].

The Dublin Core approach to metadata, which began two years before work started on RDF, has evolved in close interaction with the RDF community. DCMI Metadata Terms are defined as RDF Properties, Classes, and Datatypes (plus a DCMI-specific type of term, the Vocabulary Encoding Scheme), and DCMI properties are currently among the most widely used properties in Linked Data. (See glossary entry for DCMI Metadata Terms.)

Metaphorically, RDF may be seen as a "grammar" for a "language of data". Uniform Resource Identifiers (URIs) -- in essence, Web addresses used as unique identifiers for things real or conceptual -- constitute the "words" of that language. Like natural-language words, the words of Dublin Core -- i.e., their URIs -- belong to grammatical categories, where "properties" (e.g., "references" or "isReferencedBy") are a bit like verbs or "predicates"; "classes" are a bit like nouns; "vocabulary encoding schemes" a bit like proper nouns; and "datatypes" a bit like adjectives.

Aside from being "words" in this data language, URIs double as "footnotes" indicating ownership and maintenance responsibility for the words by way of ownership of the domain names under which the "words" (URIs) are coined, as recorded in the globally managed Domain Name Service (DNS). For example, the subdomain <http://purl.org/dc/> is "owned" (in the sense of "controlled") by the organization Dublin Core Metadata Initiative. Inasmuch URIs resolve to official representations of "words" (see glossary entry on Namespace), this globally managed space of unique identifiers functions as a continually updated "dictionary" of the RDF data language. For example, the URI <http://purl.org/dc/terms/title> resolves (by redirection) to an RDF schema ([1] at the time of writing) which says in a machine-understandable way that `dcterms:title` is an RDF property.

In its early years, the fifteen-element Dublin Core was likened to a "pidgin" -- a lexicon of generic predicates good enough for the sort of rough but serviceable communication one hears from intermediate-level speakers of foreign languages. As in natural languages, "sentences" make no sense without a shared sentence grammar which gives them meaning as complete thoughts, such as "Book A is a translation of Book B". The grammar of RDF statements follows a simple and consistent three-part grammar of "subject", "predicate", and "object". Statements all have the form: "Resource A is related to Resource B", where the "is related to" part is a predicate from an RDF vocabulary such as DCMI Metadata Terms.

Taken individually, an RDF statement does not say much, but when sentences are aggregated into "paragraphs" (RDF "graphs"), the statements can convey just about any kind of the information one might express using spreadsheets, databases, Web tags, or Web links. Just as pidgin vocabularies go only so far when expressing more specialized knowledge, a healthy ecosystem of RDF vocabularies needs to include both specialized lexicons -- e.g., for use by biologists, manufacturers, or library catalogers among themselves -- and generic lexicons for rough communication with the world in general.

RDF is a language designed by humans for processing by machines. RDF -- the grammar together with its vocabularies -- does not itself engineer a solution to the inherent problems of human communication any more than the English language guarantees world understanding. However, RDF does support the process of connecting dots -- "knowledge" -- by providing a shared basis for expressing information in a comprehensible way.

Just as English also provides a basis for communicating among non-native English speakers, RDF provides an idiom for coherently merging or linking data among systems which may not be based on RDF natively but can export or expose data on the basis of the RDF grammar using known RDF vocabularies.

RDF supports the longevity of information by expressing knowledge a generic form using a meaningful grammar -- assuming that society can find robust methods for preserving the parts of the Web where its dictionaries and resource identifiers are defined. Aside from supporting data interchange in the here and now, RDF provides a response to the ongoing and inevitable obsolescence of our computer applications and customized data formats.

- [1] <http://www.w3.org/RDF/>
- [2] <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- [3] <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- [4] <http://linkeddata.org/>
- [5] <http://dublincore.org/2010/10/11/dcterms.rdf>

Resource Discovery, Resource Description

As of 2010, the Dublin Core Metadata Initiative's mission includes the goal of "developing and maintaining international standards for describing resources".

It has not always been this way. The foundational workshop of March 1995 in Dublin, Ohio aimed at improving the "discovery" of electronic resources on a rapidly growing World-Wide Web. The approach taken was that of defining "elements" for simple "records" describing "document-like objects" [1]. Implicit was the traditional model of a search application indexing text fields (value strings) for retrieval.

What actually happened, as we know, was that Google developed a radically different, wildly successful, solution to the problem of resource discovery. The Dublin Core community's understanding of metadata's role subsequently evolved in several ways:

- The focus on "resource discovery" came to be seen as artificial -- what descriptive element might not be conceived as "useful for discovery"? -- and the objective was redefined as that of "resource description".
- The scope of Dublin Core metadata was broadened from "electronic resources" to encompass, in principle, any object that can be identified, whether electronic, real-world, or conceptual, and particularly including resources of the sort named in the DCMI Type Vocabulary [2], such as physical objects, software, services, and sound.
- Attention shifted from the "record" as an aggregate information object to individual metadata "statements" designed to be merged and recombined in an emerging Linked Data environment (see glossary entry "records or statements").
- Metadata came to be seen as valuable less for the value strings it carried than for its use of URIs -- controlled identifiers for subject headings, Web documents, people, and the like -- and for the role of properties (such as Dublin Core properties) in defining explicit links and cross-references between resources. URI-rich metadata came to be seen as the basis for richly interlinked browsing and data mashups.

[1] <http://www.dlib.org/dlib/July95/07weibel.html>

[2] <http://dublincore.org/documents/dcmi-type-vocabulary/>

[3] <http://dublincore.org/about/#mission>

Simple Dublin Core, Qualified Dublin Core

With the invention of Qualifiers in 1997 (see glossary entry Terms), a distinction was made between Simple and Qualified Dublin Core. Qualified Dublin Core referred to metadata that used the Dublin Core with DCMI qualifiers. Simple Dublin Core was typically taken to refer to:

- a description of just one resource,
- using only the fifteen properties of the Dublin Core Metadata Element Set,
- all optionally and repeatably,
- with string values,
- and without qualifiers.

This pattern was the product of a time when the Dublin Core Metadata Element Set was widely understood, even within the Dublin Core community, as the specification of a record format. At the time, it was also assumed that metadata should consist of textual values indexed for retrieval. This pattern was widely deployed after 2000 in 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, `oai_dc` [1], which may be seen as a serialization of Simple Dublin Core. Due to the popularity of such formats, many people still refer to the Dublin Core Metadata Element Set and Simple Dublin Core interchangeably.

Both Simple Dublin Core and Qualified Dublin Core are defined in XML schemas owned and maintained by DCMI [2] -- the latter following "Guidelines for implementing Dublin Core in XML" of 2003 [3]. While these formats have proven to be useful in numerous XML-based implementations over the years, such fixed formats show limitations in the heterogenous context of Linked Data. The availability of more flexible Semantic Web solutions cast Simple and Qualified Dublin Core into a different light:

- In a Semantic Web perspective, the Dublin Core Metadata Element Set -- i.e., the set of fifteen properties -- is understood as just one vocabulary, among many, available for use in data. The Simple Dublin Core pattern constrains the use of its properties in one particular way and can therefore be seen as an application profile that happens to be limited to properties from one DCMI namespace.
- The limitation of Simple Dublin Core to string values looks dated at a time when the Linked Data approach is emphasizing the use of URIs in order to make richly interlinked connections with other resources and metadata descriptions.
- In practice, implementers of Dublin-Core-based metadata more often create application profiles that use a sub-set of Dublin Core properties, or use Dublin Core properties together with properties from other vocabularies. In a Linked Data environment, and in accordance with an Open World Mindset, perfect agreement on fixed formats is not imperative, making the limitation to properties from one DCMI namespace seem a bit arbitrary. (See glossary entry on Open World Mindset.)
- Providing interoperable metadata as Linked Data, on the other hand, does not mean that data must be expressed natively using RDF. It may indeed be more practical to manage data in fixed XML formats on the backend and transform or convert the data for publication in a Linked Data context.

[1] http://www.openarchives.org/OAI/2.0/oai_dc.xsd

[2] <http://dublincore.org/schemas/xmls/>

[3] <http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/>

e:/u/folders/XPAD/Agenda.pdf
e:/u/folders/XPAD/UsingDC.pdf
e:/u/folders/XPAD/CreatingMetadata.pdf
e:/u/folders/XPAD/PublishingMetadata.pdf
e:/u/folders/XPAD/pete-comments-on-user-guide.pdf
e:/u/folders/XPAD/DCMI-Metadata-Terms.pdf
e:/u/folders/XPAD/Dublin-Core.pdf
e:/u/folders/XPAD/Dumb-Down.pdf
e:/u/folders/XPAD/Namespace-Policy.pdf
e:/u/folders/XPAD/One-to-One-Principle.pdf
e:/u/folders/XPAD/Open-World-Mindset.pdf
e:/u/folders/XPAD/RDF.pdf
e:/u/folders/XPAD/Resource-Discovery.pdf
e:/u/folders/XPAD/Simple-Dublin-Core.pdf
e:/u/folders/XPAD/index.txt
e:/u/folders/XPAD/FAQ-reusing-XML-elements.pdf

"In my metadata, I would like to use Dublin Core elements together with elements from the XML-based Standard XYZ in my metadata format. How can I do this?"

An XML format is a binding for an information structure constructed according to a specified model. XML formats define an ad-hoc "language" of elements and attributes nested within a hierarchical structure. The meaning of these components is determined solely by their placement in the tree structure of the given XML language and the interpretation that the developers of that language define in accompanying documentation. Prefixes bound to "namespace" URIs are used to avoid name collision between like-named elements, but there is no notion of XML elements themselves being identified by URIs.

In RDF, "elements" -- which in RDF are called "properties" -- are identified and referenced using URIs, and there is no notion of element containment or nesting. In contrast, there is a notion of URIs as components of "statements". The meaning of properties is considered to be global in scope, independently of any specific information structure, such as a particular record format. Statements using those properties are designed to be interpretable beyond the context of a specific information structure.

Terms defined as RDF properties are therefore not directly usable as elements in XML formats, and vice versa -- the two types are like apples and oranges.

In practice, the two can be used together if RDF properties (or classes or datatypes) are created which correspond to -- are based on or map to -- XML elements and attributes, or vice versa. In very simple cases, mappings may be so obvious as to seem trivial. But complex, highly nested XML formats can be intellectually challenging to interpret in RDF, especially if the tree structures are at all ambiguous about how particular elements or attributes explicitly relate to resources described.

In short: To "use" an XML element in RDF statements, a corresponding RDF property with a compatible meaning must be coined and assigned a URI -- but then it is no longer an XML element. To "use" an RDF property in an XML format, an RDF property can be taken as the inspiration for an XML element of comparable meaning -- but then it is no longer an RDF property. In other words, because XML elements and RDF properties are fundamentally different, they cannot be "used" together directly, but only via translation from one model into the other. Both uses may base themselves on DCMI Metadata Terms -- seen as a set of concepts with (human-readable) definitions -- but only as RDF properties are DCMI Metadata Terms expressed in a form directly usable for interoperability in a Linked Data context.

The differences between the RDF and XML models, and their consequences for interoperability, are elaborated in a 2005 discussion paper by Pete Johnston [1].

[1] <http://www.ukoln.ac.uk/metadata/dcml/dc-elem-prop/>

The history of Dublin Core holds a mirror to the transformation of blah blah.

Dublin Core is a product of the Web age, which ramped up in the early 1990s. The idea for a [simple metadata standard] to describe Web pages was floated in a hall conversation at the second "WWW" conference, in Chicago, in 1994.

Has matured with Web. All (or hopefully most) of the documents produced during... are still out there.

This glossary aims at...

We must acknowledge that the pace of change and innovation has been extraord. Some readers will be old enough to remember what they learned or perceived about DC in 1999, or 2005, and will be looking for hooks to guide them to modern concepts.

Others, perhaps the majority (definitely an increasing majority), will be coming at this fresh.

This glossary, updated in 2010, aims at striking a balance.

Not written for insiders who followed the development blow-by-blow. E.g., token, value qualifier, Warwick Framework.

Rather than aspiring to be a comprehensive glossary that one would consult upon coming across an unfamiliar term, we picture the glossary more as a short document, and therefore maintainable, written in an engaging style -- a sort of general introductory document one might assign for a metadata class or print off for reading on the train.

In light of its didactic nature, the editors see the glossary as an opportunity to coin new handles that capture and elucidate distinctions between "legacy" DCMI approaches and the current "semantic" or "linked-data" orientation. The editors therefore believe the glossary could serve as a useful focus for discussions in the DCMI community (Advisory Board, Oversight Committee, Usage Board, and general mailing lists) on clarifying DCMI's general message.

Joe: Scope of a Dublin Core Metadata Initiative Glossary.
As mentioned above, the scope of this glossary should include only those terms that are most important to describe Dublin Core style metadata. However, this is not the easiest thing to do. We currently have pre-DCAM Dublin Core and we have post-DCAM Dublin Core. We also have the problem of prescriptive versus descriptive functions of a glossary.

If we are to be selectively prescriptive and only include items post-DCAM, then we can create a small and simple document. I assume this would contain a scope statement at the beginning, outlining what was included in and excluded from the document, and a statement about its purpose - which was to prescribe definitions for terms used by DCMI post DCAM.

This leaves unresolved the question of where to put legacy terms, and their accompanying issues. This would require a larger document.

Descriptive list ("dictionary")

Prescriptive: approved and current terms ("glossary")

Joe: one word could appear in both dictionary and glossary, the former would have all definitions found or that were relevant or outdated, and the latter would have the approved definition. There could then be cross-references between the two.

All technology is transitional.

Not all of those "old world" concepts have been "recast" within the formal framework - though I'd like to think we are getting there, slowly.

Danbri message

* DC and RDF/SemWeb communities have overlapped since the beginning; chairs, editors and implementors...

* both DC and RDF families of specs are stronger because of this; and both continue to evolve

* neither DC or RDF impose any choice between 'simple' vs 'rich' metadata; instead we provide an environment where diverse kinds of linked information can be freely mixed and shared.

Re misconceptions, I think many who look at DC and see 'old fashioned' metadata are misunderstanding the (ill-documented) history of DC and of RDF; but they often also underestimate the importance of strings. Without textual strings in our data structures, we'd never find anything...

Marcia: Don't like "legacy" for things that are still valid, only older. Implies they are no longer valid. "Historical" doesn't have the same negative connotation as "legacy". Or "classic", as in classic / semantic.

Mary: We want people to think about what their metadata will look like outside their context.

Mary: Classic / Semantic / Legacy (warwick, lego).

Mary: Software designers need to understand triples. Others just need to input metadata - add VES, etc - and we confuse them.

Marcia: "Record" - really talking about Descriptions. What is the basic unit - Description? Record?

Andy in 2003

In general, you can do what you like within the confines of your own application. It is only when exporting metadata to external systems that you need to get the modelling correct.

In designing a metadata application, it is for many reasons desirable to use RDF properties that have already been declared somewhere. At a minimum, this is easier than doing the extra work involved in declaring one's own RDF properties. More importantly, the use of known properties provides a basis for semantic interoperability with metadata from other sources. Bear in mind that individuals who create vocabularies may change jobs and move on; research projects finish their work and eventually their servers disappear; and ownership of domain names may lapse, so that URIs which resolve today to an RDF schema might ten years from now resolve to shoe advertisements. It is best to use properties backed by organizations that have made a commitment to their maintenance.

RDF property semantics

RDF properties are usually provided with natural-language definitions. Designers of application profiles should take care to use the properties in ways that are compatible with these definitions. Designers may add technical constraints on use of properties (such as repeatability), or provide more narrow interpretations of definitions for particular purposes, but they should not contradict the meaning of the properties intended by their maintainers.

The intended meaning of a property is determined not just by natural-language definitions but also by formally declared relationships of the given property to other properties. Definitions typically specify a formal "domain" (the class of things that can be described by the property) and a "range" (a class of things that can be values). This additional information improves the utility of RDF properties by enabling inferences about the things they are used to describe. The property `foaf:img` (image), for example, has a domain of `foaf:Person` and a range of `foaf:Image`, so that when metadata-consuming applications find metadata using the property `foaf:img`, they can automatically infer that the thing being described with this property is a person and that the value being referred to by the property is an image. Properties may also be semantic refinements of other properties. The property `dcterms:abstract`, for example, is a sub-property of `dcterms:description`, meaning that anything which is said to have an abstract also may be said to have a description.

For the purposes of re-using properties in application profiles, it is especially important to check whether or not the properties are intended to be used with values that are literals. Properties that are intended to be used with values that are literals -- i.e., with values that by definition may consist of just one value string, optionally augmented with a language tag (in a "plain value string") or a datatype identifier (in a "typed value string") -- are said to have a "literal" range. Examples of properties with a "literal" range are `dcterms:date`, which is declared with a range of `rdfs:Literal`, and `foaf:firstName`, which is defined as being an `owl:DatatypeProperty`. The advantage of properties with a "literal" range is simplicity. The metadata carries -- and metadata-consuming applications expect -- just one plain or typed value string, making the metadata simple to encode and simple

to process.

Properties with anything other than a "literal" range are said to have a "non-literal" range. Examples of properties with a "non-literal" range include `dcterms:license`, with the range `dcterms:LicenseDocument`, and `foaf:holdsAccount`, with the range `foaf:OnlineAccount`. Where literal-range properties may be simpler to process, non-literal-range properties are more flexible and extensible. In descriptive metadata, literal values constitute "terminals" (in the sense of "end point"); the value string "Mary Jones" cannot itself be the starting point for any further description of the person Mary Jones. A non-literal value, in contrast, has hooks to which one may attach any number of additional pieces of information about the person Mary Jones, such as her email address, institutional affiliation, and date of birth. Potentially, non-literal values can be represented by any combination of the following:

- * A plain or typed value string (Value String in the DCMI Abstract Model) -- and not just one, but potentially several in parallel, as in the case of a title rendered in English, French, and Japanese.
- * A URI identifying the value resource (Value URI).
- * A URI identifying an enumerated set (or controlled vocabulary) of which the value is a member (Vocabulary Encoding Scheme URI).

Note that the difference between literal-range and non-literal-range properties is primarily a modeling issue. The type of property determines how the metadata will be encoded machine-processably for exchange and interpreted by applications that consume the metadata. End-users need not necessarily see the difference. When displayed in a search result, a value string looks the same regardless of whether it is directly a literal value or a value string attached to a non-literal value.

When using an existing property, the choice between a literal and non-literal range will usually be mandated by the official definition. If that mandated choice is not sufficient (e.g., the `dcterms:date` property has a literal range and a more complex value is needed), or if a property with the needed semantics cannot be found anywhere, then a new property (with a new URI) must be coined. Coining new RDF properties

By definition, Dublin Core application profiles "use" properties that have been defined somewhere -- i.e., somewhere outside of the profile itself. If no existing property can be found among any of the well-known vocabularies, then the designers of an application profile will need to declare one themselves.

Declaring a new property is in itself not a difficult task. One gives it a name, formulates a definition, decides whether it takes a literal or non-literal range, and coins a URI for the property under a namespace to which one has access (and not, for example, under `http://microsoft.com` or `http://amazon.de`). Services such as `http://purl.org`,

which is used for identifying DCMI properties, provide "persistent" URIs that can be redirected to documentation at more temporary locations. Guidance for creating and publishing RDF vocabularies can be found in "Cool URIs for the Semantic Web" [COOLURIS], the RDF Primer [RDF-PRIMER], and "Best Practice Recipes for Publishing RDF Vocabularies" [RECIPES]. Best-practice examples include DCMI Metadata Terms [DCMI-MT], Dublin Core Collection Description Terms [CTERMS], and Eprints Terms [ETERMS]. It is good practice for terms also to be published in RDF schemas; for examples, see the schemas associated with DCMI Metadata Terms [DCMI-MT] and Dublin Core Collection Description Terms [CTERMS].

Whether to assign a literal or a non-literal range is essentially a choice between simplicity and extensibility. Value strings alone may suffice for recording a date ("2008-10-31") or a title ("Gone with the Wind"), but for authors, one may need to record more than just a name. When in doubt, it is wise to assign a non-literal range. In the case of authors, for example, the non-literal range provides a hook for adding email address, affiliation, and date of birth or for using a URI to point to a description of the author somewhere outside one's own application. Because they support the use of URIs (i.e., the use of URIs "as URIs" and not just "as strings"), non-literal values are crucial in achieving the ideal of linked metadata -- descriptions that are cross-referenced using globally valid identifiers. Translating user-defined data requirements into design decisions

We can now return to the questions asked of data content experts about each potential property with regard to potential values.

Do you want to use free text? "Free text" (i.e., strings of characters) is called a Value String in the DCMI Abstract Model and can be used with properties of either a literal or non-literal range. Note that in some cases, there may be a requirement to use multiple value strings, in parallel, in a single statement, for example in the case of values that are represented in multiple languages. This can only be done in conjunction with non-literal-range properties.

Will the free text ever need to follow a pre-defined format? If so, then the Value String can be used with a Syntax Encoding Scheme (datatype).

Will single value strings suffice or is there a need (or potential need) for a more complex structure with multiple components? If anything more than a single value string is needed for the value, then the property used must have a non-literal range. If by chance you have found a property with the right natural-language definition but the wrong range -- for example, with the more limiting literal range -- you may need to coin your own property, with its own URI, using that definition with a non-literal range.

Might you ever want to use a URI to identify the value or point to a description of the value? The DCMI Abstract Model defines a Value URI as a syntactic construct separate from a Value String. Value URIs cannot be used to describe

literal values; they must be used with properties that have a non-literal range. It is of course possible to record a URI as a Value String -- a URI is, after all, a string -- but applications consuming the metadata on this basis will have no reliable way to distinguish that string from other strings in order to interpret it as an identifier.

Will you want to select valid values from a controlled list? If so, then the following are possible:

- * Simple lists of text strings may be informally documented as usage guidelines in a Description Set Profile.
- * Simple lists of text strings may be more formally defined as a Syntax Encoding Scheme (SES, or datatype), with a URI, making the list citable and available for use in many application profiles.
- * A list of strings may be interpreted as labels for a list of concepts. This is called a Vocabulary Encoding Scheme (VES). Note that in contrast to the individual text strings listed in an SES, the individual concepts of a VES may also, or alternatively, be identified using URIs.
- * If the list of values is already available somewhere and has already been identified (e.g., by DCMI) as a Syntax Encoding Scheme or Vocabulary Encoding Scheme, then use that URI with the proper modeling construct.
- * If the list of values is already available somewhere but has not yet been identified as an SES or VES, you may need to interpret which model it more closely fits. Sometimes either interpretation is defensible. Whichever way you decide, it is important that the URI you coin for the encoding scheme be clearly declared as one or the other in order to avoid any ambiguity in the metadata.
- * If you want to restrict the set of valid values to a fixed list (as opposed to allowing the use of unlisted values), this restriction can be declared and documented in a Description Set Profile.

In general, the use of formally defined values, such as controlled lists, adds precision to metadata and thus increases its suitability for automatic processing. The use of SES and VES, as appropriate, is an important step in this direction. Increasingly, however, URIs are being assigned to individual terms in controlled vocabularies using the RDF vocabulary Simple Knowledge Organization System [SKOS]. The concept "World Wide Web" in the Library of Congress Subject Headings, for example, has recently been assigned the URI <http://id.loc.gov/authorities/sh95000541#concept>. As controlled vocabularies become increasingly "SKOSified", it will become easier to use those vocabularies to find and integrate access to resources from multiple sources on the open Web. The VES construct can flexibly accommodate the transition from Value Strings alone, to Value Strings with VES URIs, and from there to Value URIs (with or without VES URIs).

01 URIs are footnotes of data. -> Preserve vocabs. -> Library role.
02 Language of triples. DC as pidgin, RDF as grammar. "Translating" into language of URIs.
03 Apps come and go, data remains. Data should be self-describing - not rely on "out-of-band".
CIO DCMI, 2006-2009 SWD SKOS, PhD, Well, MLS, GMD, Goett, AIT.

I'm Tom Baker. Aside from co-chairing this incubator group I am the CIO of Dublin Core Metadata Initiative Ltd and work on freelance projects, most recently with the Food and Agricultural Organization of the UN (resulting in a chapter for an upcoming book from Springer Verlag on linked enterprise data). From October 2006 to December 2009 I was co-chair of the W3C Semantic Web Deployment Working Group, which brought SKOS to Recommendation and shared responsibility with another working group for RDFa.

While finishing a PhD in anthropology at Stanford in the 1980s, I participated on The WELL, an early virtual community in the Bay area, and grew convinced that the Internet would change the world, in particular the library in which I loved to work. After working for awhile as a social scientist in Italy, I returned to the States, got an MLS at Rutgers, and went to the German National Research Center for Information Technology (GMD), where I worked on digital library projects and international networks, including the early Dublin Core effort. I continued this while living in Thailand for two years, where I taught at the Asian Institute of Technology, returning in 1999 to GMD (later Fraunhofer) then to the Goettingen State Library, returning to the States in July 2009.

My ambition has been to maintain Dublin Core as a best-practice RDF vocabulary for use in combination with other RDF vocabularies in application profiles. (Thanks to Tod Matola on this list for co-authoring a DCMI Namespace Policy in 2001.) In 2010, to my way of thinking, data without URIs is like scholarship without footnotes. Just as libraries have preserved the scholarly record, they should help keep the supporting vocabularies for data accessible for the long term. And they should have a keen interest in translating their legacy knowledge organization systems into the language of URIs.

URIs are just the words of the language of data; equally important is the grammar. As a grammar for expressing knowledge as coherent statements, RDF represents our best available hope for expressing knowledge independently of transitory application environments. Applications come and go, but the data remains -- or at least we hope it will.

we should not raise the bar by requiring the preservation of ad-hoc data formats and application environments
Dublin Core: The Road from Metadata Formats to Linked Data
August 25, 2010

Meeting Questions and Answers:

michelle asked: Does the URI for the book represent: a full-text online version of the book? the concept of the book?

URIs are the "words" of the Linked Data "language". URIs used as "predicates" (aka "properties", characterizing relationships between resources) are like verbs. URIs used as identifiers for resources that are the subject or object of statements are like nouns.

As in natural language, names can be given to anything -- including either a full-text online version of a book or the concept of a book. Users of the "FRBR" categories Work, Expression, Manifestation, Item (in the Functional Requirements for Bibliographic Records, a model used in the library world) might need to coin URIs for each of these entities, because in the library catalog context these distinctions are important. Others may not need, or even understand, such distinctions.

The book examples A and B might reasonably be interpreted to be about "manifestations" (in FRBR terms), even if the data does not say that. In an environment with full support of FRBR, related "works", "expressions", even individual "items" could be identified with URIs and described.

[1] http://www.ukoln.ac.uk/repositories/digirep/index/Scholarly_Works_Application_Profile

L. H. Kevil asked: How can you get the linked data from the Beeb?

In the BBC examples shown in the talk, the Linked Data corresponding to the two Web pages

http://www.bbc.co.uk/nature/species/Humboldt_Squid
<http://www.bbc.co.uk/nature/adaptations/Nocturnality>

can be seen at:

http://www.bbc.co.uk/nature/species/Humboldt_Squid.rdf
<http://www.bbc.co.uk/nature/adaptations/Nocturnality.rdf>

Some browsers, such as Firefox, can display this data nicely, or the URIs can be plugged into an RDF validator such as [1] for viewing as a graph.

More information is available at [2].

[1] <http://www.w3.org/RDF/Validator/>

[2] <http://www.bbc.co.uk/blogs/bbcbackstage/>

Amy Kirchhoff asked: Can anyone use these verbs/URIs or can only BBC use their BBC verbs?

The squid page uses eighteen properties (i.e., "verbs", predicates) defined by BBC. Seventeen of these are defined the BBC Wildlife Ontology [1]. Notice that [1] is licensed under a Creative Commons Attribution License, and everyone is invited to make free use of the work. For example, anyone is free to use the URI for the property "species name" [2] in their own metadata.

[1] <http://purl.org/ontology/wo/>

[2] <http://purl.org/ontology/wo/speciesName>

88RT93 asked: Is there a particular RDF graph visualizer you could recommend?

W3C provides an RDF Validation Service [1] where you can paste in (or point to) RDF data in XML as input and see triples and graphs as output. Google on "RDF validator" or "RDF visualizer" and you will find several more alternatives. Other tools, such as "rapper" [2], will freely convert RDF data between any two of several interchangeable formats, such as: RDF/XML, Turtle (a syntax more readable than RDF/XML), and RDF triples, and including (for output only) GraphViz DOT format [3], which can be used by general-purpose visualization software to generate graphs.

[1] <http://www.w3.org/RDF/Validator/>

[2] <http://rdflib.org/raptor>

[3] <http://en.wikipedia.org/wiki/Graphviz>

Lisa Conrad asked: Could you review very simply again the basic concept of the grammar of RDF? And concept of "triples"

For a general introduction, see [1].

Tom thinks of RDF as the grammar for a "language for Linked Data". "URIs" are the words of that language. The notions of "Property" and "Class", also expressed using URIs, are grammatical categories in that language, corresponding roughly to Verbs and Nouns. "RDF statements" are sentences in that language. RDF statements are about "individual" members of a class, a bit like "toaster" is a particular noun. By default, every individual is a member of the generic class Resource, but "toaster" might also be seen as a member of a more specific class such as "kitchen appliance".

The analogy to natural language is not perfect, but Tom finds it helpful as a first approximation.

[1] <http://www.w3.org/TR/rdf-primer/>

Laura Akerman Changes Question To: In Dublin Core (and RDF) the "atoms" of information are simple but what we need to do is often complex... for example we need to combine data from different vocabularies, or represent different aspects of a thing (e.g. FRBR work, expression, manifestation, etc.). Is your work with RDF informing any efforts to develop machine-actionable frameworks for expressing description sets and application profiles

Taken individually, the atoms that make up our bodies are simple too. However, when organized into DNA, cells, and organs, they constitute incredibly complex structures, even brains. By a very rough analogy, there is no inherent limit to the complexity of data structures that can be expressed using triples.

A DCMI working group is contributing to the development an RDF expression of RDA, and IFLA working groups are developing RDF expressions of FRBR. As Tom suggested on the call, a healthy

linguistic ecosystem for Linked Data needs both generic, pidgin-like vocabularies like the classic fifteen-element Dublin Core; specialized vocabularies such as the BBC Wildlife Ontology; and links between vocabularies to bridge the gap between specialists and the general public. This is pretty much the same as with natural languages in the real world.

DCMI's work on an Abstract Model [1] and Description Set Profile language of constraints [2] has been motivated precisely by the goal of specifying, as you put it, "machine-actionable frameworks for expressing description sets and application profiles".

[1] <http://dublincore.org/documents/abstract-model/>

[2] <http://dublincore.org/documents/dc-dsp/>

Donna Schenck-Hamlin asked: Is a URI a link to a file? If so, isn't there a danger the file might become unavailable?

We should mention here in passing that Web architecture has evolved mechanisms to respond to requests to "GET" a resource identified by a URI in ways that say either "the URI identifies an information resource, and here it is" or "the URI identifies something that is described in an information over there" (with an automatic redirect). In other words, the Dublin Core element "Contributor" and the person "Barack Obama" are not Web pages, but a user clicking on URIs identifying these things will be taken, using redirection, to a Web page describing those things.

It is "best practice" to make URIs resolvable to files, and when memory institutions coin those URIs, there is an expectation that they will remain resolvable to information about the URIs (i.e., to files) over the medium and long term. That said, "resolvability" to a file is not an absolute precondition to using a given URI as a unique identifier. We could still continue to use the word "toaster" even if, against all reasonable probability, the word were to disappear from our dictionaries. By analogy, the file describing the BBC Wildlife Ontology could disappear, but as long as the BBC Internet domain exists and the URI for "species name" is not re-purposed for something else, the URI itself could continue to function as a unique property name.

Ensuring the longevity and resolvability of URIs is, in Tom's opinion, one of the key challenges facing cultural memory institutions in the Web page.

Jennifer Riley asked: Can you explain the "placeholder" (RDF blank nodes) concept further?

In modeling terms, there is a subtle difference between saying:

This book has an author, 'Barack Obama'.

and:

This book has an author, whose name is 'Barack Obama'.

To us humans, these sentences are pretty much interchangeable.
In modeling terms, however, the second sentence is more explicit.
Using this pattern, for example, one might go on to say:

This book has an author, whose date of birth is '1961-08-04'.

The placeholder for the author, as the object of a statement, is either blank (if we do not have a URI identifying Obama the person), or -- if we do have one -- it can be a URI such as http://dbpedia.org/resource/Barack_Obama.

Bear in mind that this data expression will normally be formed by metadata software "under the hood". A human creating metadata using a Web form might type "Barack Obama", or choose "Barack Obama" from a pull-down menu, without needing to know that the underlying software is expressing that information using a URI or a blank node.

See also [1].

[1] http://en.wikipedia.org/wiki/Blank_node

Mary Parker asked: You mentioned the BBC uses linked data. Can you give an example of a library (besides Library of Congress) using linked data?

Makx mentioned a few on the call, notably the Royal Library of Sweden [1], a pioneer of Linked Data in the library world. But many others have projects underway, such as the Hungarian National Library and British Library [2]. Just yesterday I learned about a regional initiative of libraries in the German states of North Rhine-Westphalia and Rhineland-Palatinate in collaboration with Deutsche Nationalbibliothek. A W3C Library Linked Data Incubator Group [3] has been chartered for one year to collect information on such initiatives in the library world and to identify areas where further research is needed. (Full disclosure: Tom is a co-chair of that activity.)

Another particularly interesting initiative for libraries is the VIAF project (Virtual International Authority File), a joint project of OCLC, Library of Congress, and the national libraries of Germany and France, with participation of many more national libraries and other organizations (see [4]).

[1] <http://www.slideshare.net/brocadedarkness/libris-linked-library-data>

[2] <http://www.bl.uk/bibliographic/datafree.html>.

[3] <http://www.w3.org/2005/Incubator/lld/>

[4] <http://viaf.org/>

88RT93 Changes Question To: Do you know of any examples of libraries managing metadata as triples other than UK and Swedish national libraries?

I take this question to be whether any libraries using Linked Data manage that data using a triples store -- one of the options Makx presented. To be honest, we do not know, but how libraries manage their data internally is more of concern to their IT managers -- issues of efficiency, etc -- than to the consumers of their linked data. Consumers do not need

to understand the workflow in the kitchen in order to judge the quality of what they are served.

Jennifer Riley changes question to: Can you give an example of known limitations or problems with Linked Data as currently defined?

Well, it's a new language and there are not many "fluent speakers", so grammatical mistakes are an inevitable part of the collective learning curve. Also, conventions for publishing "application profiles" (in order to share patterns for describing things) and mechanisms for recognizing common patterns in data on the Web using search engines (in order to follow emerging practice) are still evolving.

Indeed, the language for linked data is itself still evolving. For example, triples do not declare their own source, so the provenance of metadata statements needs to be tracked locally in databases (as in "these statements are from BBC" and "these are from Library of Congress"). In the near-to-medium-term future, therefore, "triples" may evolve into "quads" -- four-part statements along the lines of:

Subject - Predicate - Object - SourceOfStatement

One issue that can be seen as a problem is that the Linked Data approach is so very flexible - anyone can define their own vocabulary. Now, if everyone declares their own vocabulary instead of using common, trusted, well-known vocabularies where possible, then in order to achieve interoperability, there will be an additional need to create (and maintain!) equivalence relations, such as `sameAs` or `similarTo`, between terms in as many different vocabularies. To avoid this, organizations like W3C and DCMI promote the use of well-known vocabularies like Dublin Core, FOAF, and SKOS.

Mary Parker Changes Question To: RDF dissolves metadata into a store of triple statements. Does RDF still have a conceptual place for descriptive records -- organized compilations of statements?

An excellent question! A key objective of the DCMI Abstract Model, first published in 2005 and updated in 2007, was to create an abstract syntax for organizing Statements into Descriptions (each of a single resource), grouped into Description Sets, in turn instantiated as metadata Records.

In the meantime, the RDF community has been developing the notion of a Named Graph -- a particular set of statements grouped together and given a name (i.e., URI).

The notion of a metadata Record is, and will remain, important for managing information in many institutional contexts. Supporting such constructs is seen as a high priority by the RDF community, so Tom thinks it is likely that a standardized approach to Named Graphs (or Description Sets) will emerge in the medium-term future.

OCLC Research asked: In the BBC example, how much of what we saw is

automatically generated? Don't they have to know where they are going for what, how much they grab, and how it will be displayed on the page? That all sounds kinda manual....

We do not know in detail how this is implemented at BBC; a high-level description can be found at [1]. It appears they use the "Muddy" software tool [2]. However, Tom suspects that pages are being generated by templates, perhaps built into a Content Management System, which automatically search the linked data for the latest information on particular topics -- in effect, "canned searches". At any rate, BBC cites efficiency as one of the key benefits of its approach. They write: "the system gives us a flexibility and a maintainability benefit: our web site becomes our API. Considering our feeds as an integral part of building a web site also means that they are very cheap to generate: they are just a different view of our data."

[1] <http://www.w3.org/2001/sw/sweo/public/UseCases/BBC/>

[2] <http://muddy.it/>

Michelle asked: The "open web" is free, but drives a huge economy based on advertising. Will something similar happen with Linked Data? Selling of rights to URIs?

Makx mentioned the International DOI Foundation as an example of an organization with a business model around the notion of managed URIs. As we emphasized in the talk, Linked Data is not necessarily Linked Open Data; it can also be Linked Enterprise Data, where the data may be hidden behind a corporate firewall. It seems probable that there will always be a visible, "open Web", and a hidden "deep Web", access to which is controlled or sold. We expect that there will be substantial amounts of public content on the Open Web from government, education, libraries, and research.

Heinz asked: Is there a relation to the "Semantic Web" idea?

Yes -- to the extent that many people now use the two terms interchangeably. Linked Data is based on Semantic Web principles. Before RDF was the language of Linked Data, it was (and remains) the language of the Semantic Web. In a way, Linked Data is the real-world fulfillment of the promise of the Semantic Web. The idea of Semantic Web was seen by many in circa 2000 as a sophisticated, highly engineered solution involving artificial intelligence -- e.g., letting one's smart phone find a dentist nearby and schedule an appointment automatically. It was a vision that sounded complicated and far in the future.

When the idea of Linked Data was launched in 2006, in contrast, the benefits were immediately visible in the form of data integrated across the boundaries of particular silos. This is the simpler, more concrete idea that took off, while the older, more ambitious vision of Semantic Web still remains a distant goal. The older vision was seen as focusing more on technology, whereas Linked Data shifted the emphasis to useful content. Technically speaking, however, Semantic Web -- RDF -- provided the foundation for Linked Data.

RDF as a grammar for a language of data. Sentences of that grammar have three parts - "resource A is related to resource B" or URIs are the words of that language. Has a unique identifier within a globally managed infrastructure, Domain Name Service. In context of the Web, those URIs are like footnotes for concepts (Dublin Core concept of date). Web is in that sense, functions like a dictionary for the words of this metadata language.

Properties (relationships) - think of those as being like verbs.
Classes - like nouns.

In a closed IT environment ("normal" for computing pre-Web): data formats are custom-defined for an application.

Web came along and provided a space that allows data sources to be integrated, gives rise to "open world mindset": the Web provides a context for integrating different sources of context. Systems designed to integrate information incrementally. Designed to tolerate incomplete information.

Vocabularies share this underlying grammar of data, can be used to make interchangeably to make statements about things.

URI is a name. Name anything you want. But that's the way language is. RDF as the language does not solve the larger issues that we have in human language. Language designed by humans for processing by machines.

Question whether anyone can use these verbs (e.g., BBC). Answer: these are declared in the public Web space. Organizations that do this encourage people.

We want to create a world where there are maybe five URIs for Title, but not 500, or 7 million. Hopefully not 10,000 URIs representing Obama - difficult to create coherence. Do you trust the organization that creates those URIs to be around for the foreseeable future.

Designing an application based on underlying grammatical principles. AP takes an implementer from functional requirements ("support navigation") - metadata to support those functions, to a domain model.

In a closed ("normal") IT environment, one would integrate data across silos via mappings between data structures (schemas).

If applications create good triples, then triples can be merged coherently because data links up.

Applications are very transitory things. We will not be using many of today's applications ten years from now. We will not be using, ten years from now, special data formats designed for those applications.

Triples based on known vocabularies such as Dublin Core make data self-descriptive. Merged data can be searched. Preserving vocabularies: when the data is self-descriptive, the underlying vocabularies for the predicates need to be preserved - and URIs for things like "Barack Obama".

Require shared grammar and shared words such as predicates of Dublin Core.

Metaphor in early language of Dublin Core: like a Pidgin. Can order beer. But lawyers and biophysicists will not get very far. In a healthy linguistic ecosystem, need both general vocabularies and specific vocabularies, and think how things written in specialized vocabularies can be simplified for a more general audience. There needs to be cooperation btw orgs like IFLA (assigning URIs to FRBR) and other members of that ecosystem - both the simplicity and the complexity. How can relationships between vocabularies be defined to help make such transformations more automatic.

The architecture is that each URI is owned. Unlike a word, a URI has an owner.

From tbaker@tbaker.de Sat Sep 25 13:16:00 2010
Date: Sat, 25 Sep 2010 13:16:00 -0400
From: Thomas Baker <tbaker@tbaker.de>
To: "ZENG, MARCIA" <mzeng@kent.edu>
Cc: "DC-GLOSSARY@JISCMail.AC.UK" <DC-GLOSSARY@JISCMail.AC.UK>
Subject: Re: "Dublin Core"
Message-ID: <20100925171600.GA2868@octavius>
References: <20100925143503.GA1996@octavius> <C8C38E63.1018A%
mzeng@kent.edu>
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
In-Reply-To: <C8C38E63.1018A%
mzeng@kent.edu>
User-Agent: Mutt/1.4.2.2i
Status: RO
Content-Length: 5668
Lines: 126

On Sat, Sep 25, 2010 at 11:32:51AM -0400, Marcia Zeng wrote:
> I like this whole description especially the way to make the
> distinguishes of The Dublin Core and Dublin Core metadata.
>
> 1. For the first part,
>
> ""The Dublin Core", also known as the Dublin Core Metadata
> Element Set, is a set of fifteen "core" elements (properties)
> for describing resources. "
> -- should the 'resources' be limited, e.g., 'document-like' or some other way that recently discussed?

I mention "document-like" in the (draft) entry on "resource discovery / resource description" [1]:

The foundational workshop of March 1995 in Dublin, Ohio aimed at improving the "discovery" of electronic resources on a rapidly growing World-Wide Web. The approach taken was that of defining "elements" for simple "records" describing "document-like objects".

Historically, the restriction on "document-like" objects was de-emphasized a long time ago. See, for example, the DCMI Type Vocabulary (which arguably provides a rough implicit scope to DCMI Metadata Terms), which has classes such as PhysicalObject, Service, Software, and Sound -- none of which is "document-like".

[1] <https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind1009&L=DC-GLOSSARY&P=5924>

> 2. In the DC community (and beyond) the terms elements, properties, and
> attributes have all be used to refer to the same thing. They came with
> different context of course. Maybe there is a need to explain where they
> are used in different modeling processes.

I try to address that in the proposed FAQ answer on
"re-using XML elements" [2]. I'm thinking there could be
cross-references between FAQ and Glossary entries. Or do
you think more needs to be said (somewhere)? Do you see DC
elements being called "attributes" alot, and should there be
a sentence or two about, say, UML?

Tom

[2] <https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind1009&L=DC-GLOSSARY&P=4021>

Marcia

On Sat, Sep 25, 2010 at 11:32:51AM -0400, Marcia Zeng wrote:

> I like this whole description especially the way to make the
> distinguishes of The Dublin Core and Dublin Core metadata.
>
> 1. For the first part,
>
> ""The Dublin Core", also known as the Dublin Core Metadata
> Element Set, is a set of fifteen "core" elements (properties)
> for describing resources. "
> -- should the 'resources' be limited, e.g., 'document-like' or some other=
way that recently discussed?

I mention "document-like" in the (draft) entry on "resource
discovery / resource description" [1]:

The foundational workshop of March 1995 in Dublin, Ohio
aimed at improving the "discovery" of electronic resources
on a rapidly growing World-Wide Web. The approach taken
was that of defining "elements" for simple "records"
describing "document-like objects".

Historically, the restriction on "document-like" objects was
de-emphasized a long time ago. See, for example, the DCMI
Type Vocabulary (which arguably provides a rough implicit scope
to DCMI Metadata Terms), which has classes such as PhysicalObject,
Service, Software, and Sound -- none of which is "document-like".

I also agree that the 'document-like' should be de-emphasized. This explan=
ation is very helpful. Maybe somewhere this will also be brought up to sho=
w the transferring of Dublin Core throughout the years.

It is just a fact that in the museum world or when describing an agent, som=
e core elements could not be applied (e.g., publisher, title, source, langu=
age, and original 'date'). Although each element is optional, it still lea=
ves the impression that DC is applicable to those that have become resource=
s through 'publication' (formally or informally). On the other hand, the p=
roperties that are needed for describing the resources that need to be inte=
preted (e.g., a museum object) according to other resources (published or =

orally conveyed) are not included in The Dublin Core or DC terms.

- > 2. In the DC community (and beyond) the terms elements, properties, and
- > attributes have all be used to refer to the same thing. They came with
- > different context of course. Maybe there is a need to explain where they
- > are used in different modeling processes.

I try to address that in the proposed FAQ answer on "re-using XML elements" [2]. I'm thinking there could be cross-references between FAQ and Glossary entries. Or do you think more needs to be said (somewhere)? Do you see DC elements being called "attributes" alot, and should there be a sentence or two about, say, UML?

Tom

[2] <https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=3Dind1009&L=3DDC-GLOSSAR=Y&P=3D4021>

I think somewhere there should be an entry or in the FAQ to align these terminologies together, and give the context of where they are used. In the library (MODS), museum (VRA Core 4), and archives (EAD) community (if I call them together as THEY), the element sets are all following XML terminology:

DCMI: element; > properties [of class] > relationship = types [of resource or between resource][1] > attributes [of resource/entity]

THEY: element and subelement; > DCMI: element
e.g., vra:title > dc:title

THEY: attributes [of element and subelement] > DCMI none
e.g., <vra:agentSet>
<vra:name type="..." vocab="..." refid="...">
e.g., <dateSet>
<date type="...">

It is clear that the term 'attribute' is the possible confusing point.

Marcia

Tom,

This email is VERY helpful. I lean more on the idea of linking to FAQ. What you have laid out can clear a lot what in people's minds.
More comments below.

- > The XML tree mindset and the (emergent) RDF graph mindset
- > were present in the DC community almost from the start,
- > causing the confusion and talking-past-each-other that still
- > persists today.

I like the way to explain these two as difference 'mindsets'. Maybe it could be the way to guide users of DC vocabularies through the understanding.

- > It was clear that people needed more precise, specialized
- > descriptions, but it was also clear that trying to specialize
- > in all directions at once would lead to an unmaintainable
- > muddle. Hence the emphasis on APs.

The emphasis on APs is critical for the outside communities. In the years after AP ideas came out there have been lot AP development in real applications. Maybe at the word 'resource' of your original definition, there will be an immediate link or footnote to guide people to the explanation (as you stated in the last paragraph above) and to the AP entry. The DCAP document has already provided a great landscape for AP developers - especially when used with the example of SWAP. I will borrow a term 'heavyweight' since they (e.g., SWAP) are seriously developing the model and AP and usually start from scratch. DCMI documentations seemed to have been written for these developers.

In relation to these, in contrast to the 'heavyweight' and 'developers', there are projects that may not add new terms (on top of the terms) and just want to bring cardinality and encoding requirement to terms, somewhat like 'lightweight' application profiles. (Maybe they should not be called as AP, though.)

- > I very much doubt that people in the library, museum, or
- > archives communities actually think like XML parsers.
- > It seems more the case that they think like humans (in
- > concepts and natural language) and are able to interpret the
- > visible results of applications using their concepts, and
- > that they merely care that the applications show them what
- > they expect. In other words, their concepts are reflected
- > in screen interfaces and other application outputs, not in
- > the actual data itself.

Can't agree more. I saw lots of small-to-medium size projects wanted to simply follow a practicable approach. For example, I often met participants who work in a county court, antique business, TV or radio stations, fashion institute, design schools, private foundation, etc. in addition to library, museum, and archives when I gave workshops organized by the New York Metropolitan Library Council. Usually they speak very different languages and do not know technologies like XML, RDF, namespace, etc. except spreadsheets or a turn-key system. (Well, in library community this could be true too.) All they care about is how to describe their resources, manage them, and associate theirs to others' descriptions. A web-based template or a turn-key system like ContentDM is very appealing to them because all technology stuff is not on the surface layer. These are not 'developers'; they are 'end-users'. I hope in the glossary and user guide they can be considered as well. The way used in previous "Using Dublin Core - The Elements "[1] seemed to be communicating with the end-users (I know its content needs to be revised).

[1] <http://dublincore.org/documents/usageguide/elements.shtml>

- > RDF, in contrast, uses terms with underlying grammatical
- > categories in the context of meaningful statements --
- > categories that are recognizably similar to the nouns and
- > verbs of natural language. RDF is a general-purpose language,
- > designed by humans for use by humans and machines.

These are, again, very helpful statements.

I think it could be useful to have a table aligning terminology, as you suggest, but that table would need to convey the notion that when XML elements are mapped to RDF properties, it is a mapping not of apples to other apples, but of apples to oranges -- a process of translation.

I do not, therefore, quite agree with:

> THEY: attributes [of element and subelement] =3D=3D=3D> DCMI none

because the attributes, if translated into RDF, would be expressed in the language properties and classes, which is also the "DCMI" idiom.

I tried to do the 'vocab' attribute with dcterms:aat which is fine. The r=efid, type, etc. seem to need to be 'translated', i.e., to turn the values =defined for that attribute to the predicates. A subelement also needs to be=merged with the its superordinate. Here are two statement sets:

E.g.,

```
<agent>
<name type=3D"personal" vocab=3D"ULAN" refid=3D"50020307">Wright, Frank Ll=
yod</name>
  <dates type=3D"life">
    <earliestDate>1867</earliestDate>
    <latestDate>1959</latestDate>
    <culture>American</culture>
    <role vocab=3D"AAT" refid=3D"300024987">architects</role>
  </agent>

<location type=3D"site">
  <name type=3D"geographic" vocab=3D"TGN" refid=3D"2090809">Mill Run</name>
  .... [more names for the county, state, and country]
</location>
```

MODS that is based on MARC and EAD for the archives community are even more complicated than VRA Core 4. Although these may be beyond what the Glossary and User Guide groups need to deal with, the future mapping/transferring in DCMI interoperability level I and II may see these become relevant topics.

X. What is the difference between "dc:creator" and "dcterms:creator"?
More generally, how do properties in the "dc:" and "dcterms:" namespaces differ?

In the beginning (1995-1998), "the Dublin Core" consisted of fifteen "elements" [1]. When URIs were assigned a

[1] <http://www.ietf.org/rfc/rfc2413.txt>

In theory, of course, the prefix for /terms/ can locally be declared as anything, even "foobar:", with no loss of interoperability, but in practice, I agree that guidance is a good idea.

> I've argued that we should be using "dc" because it's short and because
> the old Dublin Core vocabulary is deprecated and we have an opportunity
> to set best practice via the RDFa specification documents - migrate
> everyone to the new vocabulary whether they're using "dc" or "dcterms".
> A number of people cut/paste Dublin Core RDFa markup from the RDFa spec

> documents.

The /1.1/ properties have not actually been "deprecated". The difference between the two is that /1.1/ properties have no domains or ranges, while /terms/ properties have ranges such as Agent. The way we put it is that the use of /terms/ properties is "gently encouraged" because of their higher precision.

When we assigned ranges, our understanding was that it was no longer considered best practice to declare properties without ranges. However, it is still sometimes argued that leaving ranges unspecified may actually be helpful in some circumstances. Europeana uses both namespaces and encourages the use of dc:creator with name strings. The use of dc:creator with names is so widespread, even in W3C documents, that we simply need to live with it -- though of course it's great if the new RDFa specs can nudge people towards better practice.

In existing practice, "dc:" is mapped to /1.1/ and "dcterms:" (or "dct") to /terms/. Our preference would be not to try to reverse this practice. For one thing, /1.1/ has only fifteen properties and /terms/ has fifty-five.

Using "dc:" for /terms/ would mean that the other forty properties, the twenty-two classes, eleven datatypes, and nine vocabulary encoding schemes would all be referenced as, for example, dc:dateCopyrighted, dc:RFC4646, or dc:RightsStatement. In the popular mind, "dc" means "Dublin Core", which means "fifteen elements", and trying to change that could be confusing.

> Other's say that we should only use "dc" with the old Dublin Core
> vocabulary, it is legacy and that we should be using "dcterms" and
> should never use "dc".
>
> What's Dublin Core's thinking on this - this decision will affect all of
> the RDFa specifications and it will also affect a number of
> specifications that are using Dublin Core within the music industry.

I see the logic of your position. As I see it, using "dc" for /terms/ and consistently using only /terms/ (and not /1.1/) properties in various specs would strongly encourage the use of the /terms/ vocabulary. On balance, I think this is a good thing! However, we should perhaps anticipate that some users will use the /terms/ properties incorrectly (by not understanding the nature of ranges).

In sum, our preference:

-- keep using the "dc:" prefix for /1.1/ properties, though not to promote their use with examples in the specifications

-- keep using the "dcterms:" prefix for /terms/ properties and promote their correct use with good examples for people to cut-and-paste.

<http://lists.w3.org/Archives/Public/public-esw-thes/2009Jun/0017.html>

On Fri, Jun 19, 2009 at 06:40:03PM +0100, Norman Gray wrote:

> The SKOS Primer illustrates annotating ConceptScheme instances using
> DCTerms properties. Is it a recommendation to use that rather than
> the DC elements properties from <[19]<http://purl.org/dc/elements/1.1/>
>?

[19] <http://purl.org/dc/elements/1.1/>

Yes. This point was raised and discussed in the SWD working group (e.g., [5]). As I sometimes put it, DCMI "gently promotes" the use of the dct: equivalents of the fifteen elements because of their formal ranges.

> This is possibly more of a DC question than a SKOS one, but this list
> is probably as good a place as any to ask the question, especially
> since I'm asking it in the context of trying to describe 'good
> practice' for developing vocabularies.
>
> DC elements have the advantage that they're probably more generally
> understood, and more things might be declared as rdfs:subClassOf DC
> elements.
>
> On the other hand, the DCTerms properties are more expressive, are
> declared as subclasses of the DC elements properties, and are newer
> (though I couldn't find an explicit statement that the older ones are
> deprecated, beyond the incidental description of the DC Elements
> properties as 'legacy' in the DC Terms documentation).
>
> My guess would be that the DC Terms properties are what I should
> recommend -- is there anything wrong with that?

DCMI has gotten a lot of positive feedback from the Semantic Web community on the "makeover" of DCMI properties with domains and ranges. A longer explanation, with historical context:

dc:title [01] and dc:subject [02] (and the other thirteen Dublin Core properties) were among the first RDF properties declared anywhere. They were declared as RDF properties before W3C standardized the notion of "range" in the RDF Schema specification.

As RDF matured, the DC properties became criticized in SW circles for being underspecified. DCMI wanted to assign ranges, but in doing so did not want to "break" existing legacy data, which used "subject" (for example) both with literal and non-literal values.

As described in [4, paragraphs starting "Formal domains..."], DCMI resolved this dilemma by creating fifteen properties in the /terms/ namespace in parallel to the corresponding terms in the /elements/1.1/ namespace, and declared the former as subproperties of the latter.

It is not actually incorrect to continue using dc:subject and dc:title -- a lot of Semantic Web data still does -- and since the range of those properties is unspecified, it is not actually incorrect to use (for example) dc:subject with a `_literal_` value or dc:title with a `_non-literal_` value. However, good Semantic Web practice is to use properties consistently in accordance with formal ranges, so implementers are encouraged to use the

more precisely defined dterms: properties.

Tom (wearing his DCMI hat)

[20] <http://dublincore.org/documents/dcmi-terms/#elements-title>

[21] <http://dublincore.org/documents/dcmi-terms/#elements-subject>

[22] <http://dublincore.org/documents/dcmi-terms/#H1>

[23] <http://dublincore.org/documents/dcmi-terms/#terms-subject>

[24] <http://lists.w3.org/Archives/Public/public-swd-wg/2009Jan/0000.html>

I mis-remembered Tom's terminology; my apologies. The phrase I mis-remembered was actually "legacy". This has similar associations but a bit weaker. On consideration I think I prefer "classic".

<http://lists.w3.org/Archives/Public/public-rdf-in-xhtml-tf/2007Jul/0147.html>

""Following this proposal,

```
<#paper> dc:creator "Mark"
```

(ie. dc1.1 -- danbri)

would remain valid, though the intention, over time, is to promote the use of the more precisely defined dterms: properties. It is worth considering whether the RDFa specs should continue to use the well-known legacy dc: properties or help promote the dterms: properties by using them in the examples.""

Do all uses of 1.1 have a natural re-expression in Terms-style modelling? Has anyone got modernisation code that can do this, eg. via SPARQL queries?

<http://dublincore.org/documents/dcmi-terms/>

describes those terms as "legacy", not "deprecated"

This issue came up on the public-esw-thes list, and the way I put it there [1] was:

As I sometimes put it, DCMI "gently promotes" the use of the dct: equivalents of the fifteen elements because of their formal ranges.

...

It is not actually incorrect to continue using dc:subject and dc:title -- alot of Semantic Web data still does -- and since the range of those properties is unspecified, it is not actually incorrect to use (for example) dc:subject with a `_literal_` value or dc:title with a `_non-literal_` value. However, good Semantic Web practice is to use properties consistently in accordance with formal ranges, so implementers are encouraged to use the more precisely defined dterms: properties.

Tom

[1] <http://lists.w3.org/Archives/Public/public-esw-thes/2009Jun/0017.html>

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.

Literal unconstrained range

> Or do I have to find or create a
> new "publisher" property with a literal range?

No, there's no need to do that - though you have to bear in mind that if/when your data is merged with other data "out there", because the dc:publisher property allows for both patterns of use, (unless you control/limit your sources to those you know use only one pattern) then when you process/query the data, you have to allow for both patterns, which makes things more complex.

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.

Example: two books published by an author with the name "Trafford". The limitation of this approach is that if there are two occurrences of this name in the data, consumers of the data (human or software) have now way to determine whether they refer to the same "Trafford" or to different people who happen to be called "Trafford". (And this is why "linked data" encourages the use of URIs and discourages the use of blank nodes). One does not need to have a URI for an author to use the dcterms:creator, but if you do use a URI, then you can be more precise in what you are referring to (as below). Now you do have a globally scoped, unambiguous means of referring to that specific organisation, and two separate use of that same URI will be treated as references to the same publisher.

The same comments as above for the publisher case apply. It's not the existence or use of the URI which is the issue; it's treating the value as a "thing" and structuring your RDF triples to reflect that. The important point: one can have non-literals without using URIs to refer to them.

What schema should one use '/dc/elements/1.1/' (DC) or '/dc/terms/' (DCTERMS)?

I understand that the all elements of DC are also part of DCTERMS.

So which scheme should be applied? Both, depending on the element?

How about if a WEB-page provides content in 2 different languages, say English and German? How can this be accomplished?

Also I am not quite sure how to apply the link relation for DCTERMS.license.

The idea is to express the copyright status AND also provide a URL to find the license text. This especially applies for my many pages displaying images.

Another confusion is the mixing of elements.

Which one of the 'Description' elements should be used in the example below? (notice the use of 2 languages). Which one should NOT be used?

1: <meta name="description" lang="en" xml:lang="en" content="WEB-Site of HALO-Photographs - specializing in ..., ...as fine art prints." />

2: <meta name="DCTERMS.description" lang="en" xml:lang="en" content="WEB-Site of HALO-Photographs - specializing in ..., ...as fine art prints." />

3: <meta name="DC.description" lang="de" xml:lang="de" content="WEB-Site von HALO-Photographs - Spezialist ..., ... Photodrucke." />

Also I would assume that there limitations in the field lengths varying for each element. But I was not able to find this documented.

So far I have applied the following to the 'Index.html' on my WEB-Site and would like to learn about what is correct and what is wrong?

I would appreciate if someone could advise me in properly implementing DCMI based on this example below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
<head profile="http://dublincore.org/documents/2008/08/04/dc-html/">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>HALO Photographs: Panoramic Images of Nature, Landscapes &#38; People in digital galleries</title>
<link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />

<meta name="DCTERMS.title" lang="en" xml:lang="en" content="HALO Photographs presents Collections of Panoramic
Images of Nature, Landscapes &#38; people in various digital galleries" />
<meta name="DCTERMS.alternative" lang="de" xml:lang="de" content="HALO Photographs pr&#228;sentierte Sammlungen
von Panoram-Bildern aus Natur und Landschaften sowie von Personen in verschiedenen Gallerien" />

<meta name="description" lang="en" xml:lang="en" content="WEB-Site of HALO-Photographs - specializing in digital
imaging and the creation of large Panoramas, images shown online available as fine art prints." />
<meta name="DCTERMS.description" lang="en" xml:lang="en" content="WEB-Site of HALO-Photographs - specializing in
digital imaging and the creation of large Panoramas, images shown online available as fine art prints." />
<meta name="DCTERMS.description" lang="de" xml:lang="de" content="WEB-Site von HALO-Photographs - Spezialist fuer
digitales Bildmaterial, insbesondere grosse Panoramas, gezeigte Aufnahmen als hochwertige Photodrucke erhaeltlich." />

<meta name="DCTERMS.abstract" lang="de" xml:lang="de" content="Sammlung von digitalen Aufnahmen. Anzeige von
Bilder in online Gallerien." />
<meta name="DCTERMS.abstract" lang="en" xml:lang="en" content="Collections of digital, photographic Images and Picture
```

Galleries presented in online." />

<meta name="language" content="en,de" />

<meta name="DCTERMS.language" content="en,de" />

<meta name="DCTERMS.format" scheme="DCTERMS.IMT" content="text/html" />

<meta name="DCTERMS.type" scheme="DCTERMS.DCMIType" content="Collection" />

<meta name="keywords" content="Engadin; Photographs; Images; Panorama; Silvaplana; Black Ice; Bodypainting; White Turf, Sils; Curling; Faszination Schwarzeis; Panoramas; ExifTool; gpsPhoto" />

<meta name="DCTERMS.subject" lang="en" xml:lang="en" content="Engadin; Photographs; Images; Panorama; Silvaplana; Black Ice; Bodypainting; Sils; Curling; Faszination Schwarzeis; Panoramas; ExifTool; gpsPhoto" />

<meta name="author" content="Hans Loepfe, Switzerland" />

<meta name="DCTERMS.publisher" content="HALO-Photographs" />

<meta name="DCTERMS.creator" content="Hans Loepfe, Switzerland" />

<meta name="copyright" content="Copyright 2004-2010 Hans Loepfe, HALO-Photographs, Switzerland" />

<meta name="DCTERMS.rights" content="Copyright 2004-2010 Hans Loepfe, HALO-Photographs, Switzerland" />

<meta name="DCTERMS.rightsHolder" content="Hans Loepfe, Switzerland" />

<meta name="DCTERMS.license" scheme="DCTERMS.URI" content="http://www.halo-photographs.com/info/obtain-license.html" />

<link rel="DCTERMS.license" href="http://www.halo-photographs.com/info/obtain-license.html" />

<meta name="DCTERMS.identifier" scheme="DCTERMS.URI" content="http://www.halo-photographs.com" />

<link rel="DCTERMS.identifier" href="http://www.halo-photographs.com" />

<meta name="date" content="2008-07-14" />

<meta name="DCTERMS.date" content="2008-07-14" scheme="DCTERMS.W3CDTF" />

<meta name="DCTERMS.modified" content="2010-06-30" scheme="DCTERMS.W3CDTF" />

<!--probably not very meaning full -->

<meta name="page-topic" content="Photography, Fotografie, Natur, Nature, Umwelt, Environment, Landscape, Fine-Art Photography" />

<meta name="page-type" content="Bericht, Reportage" />

<meta name="audience" content="global, all" />

</head>

<body>

</body>

</html>

Hans Loepfe
Bahnhofstrasse 22
CH - 8803 Rüschlikon
Schweiz / Suisse / Svizzera
Mobile: +41 (0)79 764 4422
www.HALO-Photographs.com
www.Black-Ice.com

From: dcmifb-bounces@dublincore.org [mailto:dcmifb-bounces@dublincore.org] On Behalf Of Makx Dekkers
Sent: Friday, July 09, 2010 11:17 AM
To: hans.loepfe@bluewin.ch; dcmifb@dublincore.org
Subject: RE: [DCMI Feedback] Web Site - What to use DC. or DCTERMS. ?

Dear Hans,

Thanks for your interest in the Dublin Core Metadata Initiative and Dublin Core metadata.

As to your question about the dc versus dcterms namespaces, it needs to be noted that the differences are mostly relevant for RDF implementations.

The main difference is that the dc: terms are less rigorously defined. For example the dcterms: terms have formal domains and ranges (see: http://www.w3.org/TR/rdf-schema/#ch_properties) assigned to them, while the dc: terms have not. In practice this means you can use a dc: term with any type of value, while the values of a dcterms: term are more restricted. As an example, the legal values of dcterms:creator (<http://dublincore.org/documents/dcmi-terms/#terms-creator>) are members of the class dcterms:Agent with the intention that the value is a URI that stands for the person or organization. In RDF:

`<dcterms:creator rdf:resource='http://dbpedia.org/resource/Max_Frisch' />` is correct whereas `<dcterms:creator>Max Frisch</dcterms:creator>` would be incorrect.

On the other hand, both `<dc:creator rdf:resource='http://dbpedia.org/resource/Max_Frisch' />` and `<dc:creator>Max Frisch</dc:creator>` are correct (although the receiving application would need to have a branch in its evaluation of the statement).

For XML and HTML implementations these differences may be less important, but even there I guess you would want to use the rule that dcterms:creator has a value that is a link, while dc:creator could have either a link or a text string as value. This is what is recommended in the DC-HTML spec at <http://dublincore.org/documents/dc-html/>. For example:

`<link rel='DCTERMS.creator' href='http://dbpedia.org/resource/Max_Frisch' />` for the dcterms variant and either `<link rel='DC.creator' href='http://dbpedia.org/resource/Max_Frisch' />` or

`<meta name=DC.creator content='Max Frisch' />` for the dc variant.

So my view is that if you want to be rigorous, and have URIs for your agents, subjects, etc., you're better off using the more strongly typed terms in the dcterms: namespace, if you need the flexibility (and are OK with moving the burden to the consuming application) the dc: namespace may be more useful. But you also need to note that the dc: namespace only contains the original 15 elements and nothing more.

As to your question on the license, I am not sure what your problem is. Does the licence not imply copyright status?

As to the use of any of the elements, this is something you need to determine for yourself, depending on the use you foresee for your data. Who are the consumers, and how do they index your data? My own opinion is that you should do multiple copies if you want to be flexible.

My apologies that I am not able to give you advice on the actual example that you provide. I have forwarded your message to the Usage Board in the hope they can help you with that, but due to holidays this may take a while.

2010-07-01

OK, maybe I should put this differently

I think there should really be a reaction from the Usage Board on this discussion between two people who may not be completely clear on how to do these things.

As it stands, the DC-General population may look at the result of this discussion as an endorsement (or at least silent approval) by DCMI for the use of two instances of dc:identifier pointing to two files with different formats of the same video material.

Is that what we want to convey?

Makx.

> c. Can one map from dc:title to dcterms:title and vice-versa?

As above, if I find

doc:D dcterms:title "abc"

I can infer

doc:D dc:title "abc"

Or to put it another way, one can map from dcterms:title to dc:title.

But not vice versa: I can't map from dc:title to dcterms:title. There's no basis for doing so in the definitions of the properties. The two properties aren't equivalents. And the reason for this is that they have different ranges: for dcterms:title, the range is the class of literals, but for dc:title the range is the class of all resources. In theory at least, dc:title could be used with a non-literal value, which the range of dcterms:title does not support.

The way the DCMI properties are described, using RDF Schema, it's actually pretty hard to introduce "contradictions" in any "formal" logical sense, but DCMI tries to make the intended use of the properties clear in its prose descriptions, and it would be going against those "informal" recommendations.)

Date: Sun, 5 Sep 2010 15:00:03 -0400
From: Thomas Baker <tbaker@tbaker.de>
To: DCite-L@listserv.ucop.edu
Cc: DCMI Usage Board <dc-usage@jiscmail.ac.uk>
Subject: Comments on Metadata Kernel for DataCite

Joan Starr,

Thank you for alerting the Dublin Core community to this work and for seeking review and feedback. I am very happy to see mappings to Dublin Core properties defined in Section 3, page 16, of the attached document (is the document also on the Web and citable by URL?).

Some general and specific comments on the Dublin Core mapping:

The cited document, DCMI Metadata Terms [1], does indeed define some of the elements -- to be precise, the fifteen elements of the traditional "Dublin Core" -- with URIs starting both with <http://purl.org/dc/terms/> and with <http://purl.org/dc/elements/1.1/>.

However, the contrasts "Dublin Core Simple" vs "Dublin Core Qualified", and "elements namespace" vs "terms namespace", are not supported by the language of [1]. The terminology of "simple" and "qualified" Dublin Core dates back to a time when Dublin Core was widely understood to be primarily as a data format (e.g., in XML) with simple and more complex variants, whereas DCMI Metadata Terms is defined as an RDF vocabulary.

What the dc: and dcterms: URIs identify are RDF properties -- descriptive concepts defined in a global space and with formal interpretations when used in Linked Data. The difference between the dc: properties and the dcterms: properties is that many of the latter have been given formal "ranges" -- i.e., they specify whether the properties are used to refer to a "string" (a character sequence, as with the date "2010-09-05" or title "Gone with the Wind"), or a "thing" (an entity, such as a Person, which may have a name, date of birth, and the like) when expressed as Linked Data.

A few suggestions on how to handle this in your mapping:

- Drop the references to "simple" and "qualified".
- Refer to the "elements" or "terms" simply as "properties" (which is unambiguous, hence preferred).
- Provide a sentence of context explaining what is being mapped to. As shown, I understand the mappings as mappings to RDF properties; if they were really intended to be schema-to-schema mappings, the schemas would need to be cited. At any rate, the URIs cited do not identify XML schema elements.
- For each DataCite element, map to either a dc: or dcterms: property, not to both. If the concept of formal RDF ranges is new to your group, my suggestion would be to map the DataCite elements consistently to dc: properties, where available, because they lack ranges (hence the mapped data is less likely to be "wrong" in a formal sense).

On the other hand, if keeping up with best-practice guidelines for Linked Data is a priority, it would be considered more "precise" to map to dcterms: properties -- but implementers of the mappings would need to take this precision into account. In short, it would be easier

(and not incorrect) to use dc: properties. The exception is "dc:extent", which does not exist.

- To judge by their names, several of the DataCite elements -- titleType, contributorType, resourceIdentifierType, relatedIdentifierType, relationType, and descriptionType -- imply a modeling function that would not map straightforwardly to Title, Contributor, Identifier, Relation, and Description, but rather to what DCMI Metadata Terms refers to as Syntax Encoding Schemes (RDF datatypes), Vocabulary Encoding Schemes, or other properties. How these are mapped would determine how they are interpreted in a Linked Data context. One would need to look at these "types" case-by-case to determine the precise mapping - something I cannot undertake in these brief comments.

If the group is not experienced with RDF and Linked Data modeling, my overall recommendation would be to start with a very simple mapping using dc: properties where available, and to add mappings to RDF datatypes and the like as the requirements and technical details for exposing data in a Linked Data context are clarified.

I hope this is helpful...

Best regards,
Tom Baker

[1] <http://dublincore.org/documents/dcmi-terms/>

Since 1998, when these fifteen elements entered into a standardization track, notions of best practice in the Semantic Web have evolved to include the assignment of formal domains and ranges in addition to definitions in natural language. Domains and ranges specify what kind of described resources and value resources are associated with a given property. Domains and ranges express the meanings implicit in natural-language definitions in an explicit form that is usable for the automatic processing of logical inferences. When a given property is encountered, an inferencing application may use information about the domains and ranges assigned to a property in order to make inferences about the resources described thereby.

Since January 2008, therefore, DCMI includes formal domains and ranges in the definitions of its properties. So as not to affect the conformance of existing implementations of "simple Dublin Core" in RDF, domains and ranges have not been specified for the fifteen properties of the dc: namespace (<http://purl.org/dc/elements/1.1/>). Rather, fifteen new properties with "names" identical to those of the Dublin Core Metadata Element Set Version 1.1 have been created in the dcterms: namespace (<http://purl.org/dc/terms/>). These fifteen new properties have been defined as subproperties of the corresponding properties of DCMES Version 1.1 and assigned domains and ranges as specified in the more comprehensive document "DCMI Metadata Terms" [DCTERMS].

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 dterms: 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 dterms:creator to dc:creator for use by Semantic Web-aware applications. Over time, however, implementers are encouraged to use the semantically more precise dterms: properties, as they more fully follow emerging notions of best practice for machine-processable metadata.

2010-02-25

On Wed, Feb 24, 2010 at 05:27:19PM +0000, Antoine Zimmermann wrote:

> Another slight problem of owl:equivalentProperty, though arguably
> minor, is that it is not part of the RDF/RDFS vocabulary. So, a pure
> RDFS reasoner or RDFS tool would simply ignore the two implicit
> rdfs:subPropertyOf. There is no reason to assume that all RDFS tool
> support the OWL vocabulary.

Antoine raises another point on which I would appreciate feedback on DCMI's work.

DCMI Metadata Terms [e.g., 1] are currently defined entirely using RDF and RDFS. Domains and ranges were assigned to most DCMI properties in 2007, as discussed in [2], but every DCMI property is still declared simply to be of type `rdf:Property` -- not of type `owl:DatatypeProperty`, `owl:ObjectProperty`, or `owl:InverseFunctionalProperty`, etc, as in FOAF [3].

DCMI metadata terms started to be coined in 1995, two years before RDF even began as a project, so much of DCMI's efforts have been about evolving with the times. Though we have certainly noticed the rising use of OWL for defining vocabularies, nobody has ever proposed that we revisit DCMI's RDF-based style for declaring terms. Indeed, Antoine's point above makes me wonder whether there might in fact be good reasons to continue along this current path - or, if we were to start using OWL vocabulary, to preserve compatibility with RDFS tools by using it only in addition to RDFS vocabulary.

I would be very interested to hear views from members of this list on this question. As always, DCMI tries to promote solutions that can straightforwardly be imitated by others, so the general question is whether it is still good practice to declare such a vocabulary using just RDF and RDFS, or whether the use of OWL significantly enhances its usability, and if so, in what ways.

I will be happy to pass any such views to DCMI lists but also cordially invite anyone to engage the DCMI community directly on these issues by posting to the dc-architecture mailing list [4].

Best regards,
Tom

- [1] <http://triplr.org/ntriples/purl.org/dc/terms/>
- [2] <http://dublincore.org/documents/2007/07/02/domain-range/>
- [3] <http://triplr.org/ntriples/xmlns.com/foaf/spec/>
- [4] <http://www.jiscmail.ac.uk/lists/dc-architecture.html>

Vatant

My 2 cents as a practitioner re-using frequently in customers' OWL ontologies elements from dcterms, I have each time to introduce their re-definition as OWL object or datatype properties. Actually this redinition is somehow automatic for object properties at least when introducing them through an ontology editor such as Protege I would definitely be more comfortable with such a definition being present in DCMI specification providing an OWL view of dcterms ontology.

In a nutshell, Aidan suggests:

- DC terms are so popularly instantiated by RDF publishers because they are lightly specified. That said, some lightweight OWL constructs could prove useful... owl:inverseOf springs to mind for properties of the form dct:replaces/dct:isReplacedBy.
- If DCMI leaves the safe cove of RDFS, some difficult/interesting questions would have to be answered.
 - > This is another hot-topic/can-of-worms, which relates to the
 - > expressiveness of Web vocabularies and compliance with the requirements
 - > of different applications and possible adopters.
 - >
 - > One point of note: adding OWL axioms to a vocabulary will not affect
 - > RDFS compliance. RDFS can be applied to arbitrary RDF graphs. Similarly,
 - > any reasoner with RDF-based semantics (RDFS/ OWL 2 RL *rule* engine)
 - > should usually be applicable over any RDF graph. Similarly, OWL is
 - > backwards compatible with a large part of the more interesting segment
 - > of RDFS.
 - >
 - > With respect to the DCMI metadata terms, the first question then is
 - > whether or not you need OWL? My intuition would be that you don't: DC
 - > terms are so popularly instantiated by RDF publishers because they are
 - > lightly specified. That said, some lightweight OWL constructs could
 - > prove useful... owl:inverseOf springs to mind for properties of the form
 - > dct:replaces/dct:isReplacedBy.
 - >
 - > More generally, once one steps outside of pure RDFS, the argument
 - > becomes more broad with respect to the myriad of restrictions present in
 - > the different DL-based species and profiles of OWL (OWL DL, OWL Lite,
 - > OWL 2 RL, OWL 2 EL, OWL 2 QL). Each has its own restrictions
 - > for validity with respect to a given ontology. Each has its own
 - > advocates and implementations and possible use-cases. Ouch.
 - >
 - > In any case, most Web vocabularies are either RDFS or OWL Full: they
 - > reasonably pick-and-choose the RDFS/OWL constructs they deem useful
 - > without too much sympathy for the DL-based restrictions (which,
 - > conversely, are not sympathetic to the creation of Web vocabularies).
 - >
 - > FOAF have had this problem for years I guess, with respect to trying to
 - > keep their vocabulary within OWL DL. Thus, they have included some OWL

> syntactic sugar to keep the vocabulary *nearly* OWL DL compliant.
 > However, FOAF have encountered an ultimatum: in OWL DL, a
 > datatype-property cannot be inverse-functional, and thus FOAF cannot say
 > that foaf:mbox_sha1sum is inverse-functional without dropping OWL DL
 > compliant. Since this definition was core to the earlier FOAF use-case
 > of "smushing" (in the heady days of rampant blank-nodes) they reasonably
 > decided to forgo OWL DL compliance. [I may stand corrected here Dan?]
 >
 > I know that Antoine has been informally thinking about how to neatly
 > allow Web vocabularies to link to different versions of the vocabulary
 > compliant with different profiles and tools which might fit into this
 > discussion. Perhaps Antoine might elaborate?
 >
 > Again, I cannot offer a concrete solution or way forward. All I know is
 > that if DCMI leaves the safe cove of RDFS -- and given the more formal
 > nature of DCMI which will probably not allow "best-effort" OWL Full
 > extension -- some difficult/interesting questions will have to be answered.
 >
 > Cheers,
 > Aidan
 >
 > > [1] <http://triplr.org/ntriples/purl.org/dc/terms/>
 > > [2] <http://dublincore.org/documents/2007/07/02/domain-range/>
 > > [3] <http://triplr.org/ntriples/xmlns.com/foaf/spec/>
 > > [4] <http://www.jiscmail.ac.uk/lists/dc-architecture.html>

Richard points out that a decision to type DC properties as
 Datatype- or ObjectProperties would rule out the use of DC
 properties for annotating ontologies and suggests that DCMI
 steer clear of the issue, as RDFS is sufficient, and OWL users
 will always be able to add local constraints for their
 ontologies, as they already do today.

> >DCMI Metadata Terms [e.g., 1] are currently defined entirely
 > >using RDF and RDFS. Domains and ranges were assigned to most
 > >DCMI properties in 2007, as discussed in [2], but every DCMI
 > >property is still declared simply to be of type `rdf:Property` --
 > >not of type `owl:DatatypeProperty`, `owl:ObjectProperty`, or
 > >`owl:InverseFunctionalProperty`, etc, as in FOAF [3].
 >
 > The thing that worries me here is `owl:AnnotationProperty`. OWL (at
 > least in the DL flavour) strictly separates the ?instance? level from
 > the ?schema? level, and one has to choose whether a property is to be
 > used on the one level or on the other. Unlike FOAF properties, which
 > are about the ?instance? level (people etc), most DC properties can be
 > reasonably used both on the instance and on the schema level
 > (ontologies have creators and modification dates). A decision to type
 > DC properties as Datatype- or ObjectProperties would rule out the use
 > of DC properties for annotating ontologies.
 >
 > Personally I'd prefer if DC steered clear of this issue. The RDFS
 > definitions are sufficient for use on the web of data. And OWL users
 > can add additional local constraints for their particular ontologies,
 > as they already do today.

Another interesting perspective from Jonathan Rees from a
 relevant thread on the pedantic-web list [1]:

The DC terms are very popular, and in particular many users of OWL (and OWL-DL in particular) use them or adapt data sources that use them. The practice is generally to make a copy of DC and then edit it to turn it into an OWL or OWL-DL file. The popular ontology editor Protege even provides such a DC variant as part of its distribution.

I think users would be served better by having a common OWL-DL version of DC, whether provided by DCMI or by someone else. Protege's is close to being such (although it is based on dc: elements instead of dct: terms). One problem is the question of whether the properties should be annotation properties or object/data properties, which matters for DL. IIUC Protege takes the position that the dc: properties are all annotation properties, while Biba says that the dct: properties are object/data properties. I could fully sympathize if DCMI didn't want to get into the middle of this feud.

Matthew Horridge <matthew.horridge@CS.MAN.AC.UK>

OWL 2 supports annotation property domain and range and also allows an annotation property to be a sub-property of another annotation property (OWL 1 allowed none of this), so annotation properties are "more expressive" in OWL 2 than they were in OWL 1. The reason that I mention this, is that I believe annotation properties would be better than object and data properties for the translation from DCT into OWL. This is because the objects/values of an annotation property can be both resources (IRIs) or string literals. Whereas the values for a data property must be a string literal, and the value of an object property must be an OWL individual (resource), but data and object properties can't share the same names in OWL 2. I've seen mixed use of resources and literals for the dc:author property, for example, but this is only possible with annotation properties.

http://bloody-byte.net/rdf/dc_owl2dl/ - DC in OWL

Dublin Core in OWL 2

This site provides OWL 2 DL ontologies for terms of the Dublin Core Metadata Initiative. It is meant for applications and other ontologies which need OWL DL versions for reasoning or import rather than the existing RDFS schemas provided by the Dublin Core Metadata Initiative itself.

Various ontologies define the Dublin Core terms they use themselves rather than import something and they mostly define the properties as annotation properties. Others have provided OWL versions that also only use annotation properties which is problematic in some cases. Various new features of OWL 2 now allow alternative ways of modelling and this project tries to deliver coherent and consistent mappings of Dublin Core terms to OWL making use of those features.

Ontology URIs and files

Use the PURL URIs below as ontology URIs when you want to import the ontologies.

* Metadata Terms – annotation property version:

http://purl.org/NET/dc_owl2dl/terms (redirects and negotiates to Turtle version <dcterms.ttl> and RDF/XML version <dcterms.rdf>)

- * Metadata Terms – object and datatype property version:
http://purl.org/NET/dc_owl2dl/terms_od (redirects and negotiates to Turtle version <dcterms_od.ttl> and RDF/XML version <dcterms_od.rdf>)
- * Metadata Element Set, Version 1.1:
http://purl.org/NET/dc_owl2dl/elements (redirects and negotiates to Turtle version <dc.ttl> and RDF/XML version <dc.rdf>)
- * Abstract Model: http://purl.org/NET/dc_owl2dl/dcam (redirects and negotiates to Turtle version <dcam.ttl> and RDF/XML version <dcam.rdf>)
- * Type Vocabulary: http://purl.org/NET/dc_owl2dl/dcmitype (redirects and negotiates to Turtle version <dcmitype.ttl> and RDF/XML version <dcmitype.rdf>)

Annotation versus object/datatype

In modelling DC as OWL ontologies others have so far chosen to declare all properties as annotation properties. This is because Dublin Core doesn't always make clear statements if the values of the properties should be literals or non-literals (and for example for `|dcterms:title|` <<http://dublincore.org/documents/dcmi-terms/#terms-title>> it explicitly states that both literals and non-literals are allowed). And annotation properties allow for both because they have no semantic relevance for reasoning.

Because of this in OWL 1 DL you can't declare any domains, ranges and sub-properties for annotation properties. So if you want to use them as an integral part of your data model it doesn't work. Link an individual to another resource with `|dcterms:creator|` and you can't infer that it is a `|dcterms:Agent|` like Dublin Core states it is. Also you can't build on those properties, declaring your own refined properties as sub-properties of them.

In OWL 2 DL this is possible: you can declare domains, ranges and sub-properties for annotation properties. But quoting the spec: "These special axioms have no semantic meaning in the OWL 2 Direct Semantics, but carry the standard RDF semantics in the RDF-based Semantics (via their mapping to RDF vocabulary)." And, using this approach means that all properties you declare as sub-properties of annotation properties are annotation properties themselves.

Also new in OWL 2 DL is punning

<http://www.w3.org/TR/2009/WD-owl2-new-features-20090611/#F12:_Punning>.

You can use one and the same URI for entities of different kinds, thus treating for example a resource as both a class and an individual. The reasoning engine will just treat them as two different entities. This, although not explicitly mentioned, also works for ontology URIs. You can thus declare `|dcterms:creator|` as an object property and use it on ontologies, classes, properties and individuals alike and still get the full reasoning capabilities. You can even make regular sub-properties of it.

With punning there is still a restriction that one URI doesn't denote both a datatype property and an object property so for each property you have to decide which it should be. This is a bit difficult in some cases. Therefore two different ontologies are offered here for the Dublin Core Metadata Terms model (namespace `|http://purl.org/dc/terms/|`).

One uses annotation properties only. It imports the Element Set (namespace `|http://purl.org/dc/elements/1.1/|`) terms which are defined as annotation properties anyway and declares Metadata Terms properties

as sub-properties of those as declared in the specification. Using this ontology you have full flexibility in how you want to use the terms with the downsides of annotation properties as described above. This for example means that you can use DCSV <http://dublincore.org/documents/dcmi-dcsv/> for representing complex values as literals – or you can choose to model this with resources instead. While the DCMI itself recommends to do the latter (but doesn't restrict you to it), using this you will get better interoperability with legacy data.

Annotation properties seem a bit like cheating though. They do the trick of allowing both literal and non-literal values but they're not meant for that purpose. The second ontology offered therefore is not based on the Element Set ontology anymore but uses object and datatype properties mainly. This means a decision for one or the other had to be made sometimes (because the Metadata Terms specification is not always consistent in its definitions). Datatype properties were used for all date-related properties, for `|dcterms:identifier|` and for its sub-property `|dcterms:bibliographicCitation|`. Annotation properties were still used for the properties `|dcterms:description|`, `|dcterms:title|` and their sub-properties `|dcterms:abstract|`, `|dcterms:tableOfContents|` and `|dcterms:alternative|` because here the specification clearly states that both literals and non-literals make sense and because their character is annotation-like enough. All other properties are defined as object properties. The range of the property `|dcterms:type|` couldn't be defined as a class like in the annotation property version because this apparently is not valid for object properties in OWL 2 DL. It is recommended to use `|rdf:type|` <http://dublincore.org/documents/dc-rdf/#sect-5> in place of that property anyway.

Using this ontology you get the full reasoning capabilities and can use the properties as proper members of your data model, however you are restricted to one interpretation of the character of the properties and may get into interoperability issues.

Since the DCMI is recommending to use non-literal values in many cases (which is more in the gist of the modern Semantic Web) hopefully they will offer their own interpretation of the situation to make OWL reasoning more consistent in the future. Until then the ontologies offered here may be of help for you.

Licence

This page as well as all of the ontology files (but not the Dublin Core models themselves) are licenced under a Creative Commons licence <http://creativecommons.org/licenses/by/3.0/>.

> It really is all a bit confusing because I don't think there
> is any way of making these distinctions in RDFS. AFAIK, one
> really needs to get into typing things as `owl:DatatypeProperty`
> or `owl:ObjectProperty` to make this stuff clear in formal ways,
> but I seem to recall there were other reasons for not wanting
> to do that...

It has to do with some people wanting to type things as annotation properties instead, and we were advised to remain in the safe harbor of RDFS instead of coming down one way or the other...

Yes, I understand, but that does leave us with this perennial problem of having to explain (in some other way e.g. +examples) that we do have a preferred pattern of usage in mind.

I still wonder whether we could/should make an argument for making the distinction for the dcterms:* properties, and +leaving the dc:* properties alone.

Some Application Profile Design Patterns

These design patterns are based on the Dublin Core Description Set Profile (DSP). The DSP structure is a single description set that contains one or more descriptions, each of which contains one or more metadata statements. Both the description and the metadata statements can define constraints on usage.

Statement: A Simple String

The simplest metadata statement describes a property that takes a simple string value ("literal"), with no constraints.

```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14 C:\dcmi\dcmi-dsp.xsd">
  <DescriptionTemplate ID="resource" >
    <StatementTemplate ID="title" type="literal">
      <Property>http://RDVocab.info/Elements/title</Property>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

Mandatory and Repeatable

Both description templates and statement templates can be coded as mandatory/optional and repeatable/not repeatable. This is done using the values "minimum" and "maximum". The most commonly used values are:

- "minimum = 0" means that the element is optional (the element is not required) [DEFAULT]
- "minimum = 1" means that the element must appear at least once. (the element is required)
- "maximum = 1" means that the element is can only appear once (the element is not repeatable)
- "maximum = infinite" or "maximum = [any non-negative number here]" means that the element can appear more than once, up to the maximum number (the element is repeatable) [DEFAULT]

The following code defines a property ("title") that is required and not repeatable:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14 C:\dcmi\dcmi-dsp.xsd">
  <DescriptionTemplate ID="resource" >
    <StatementTemplate ID="title" type="literal" minoccurs="1" maxoccurs="1">
      <Property>http://RDVocab.info/Elements/title</Property>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

Using Controlled Lists

One way to assure consistency of metadata use is to require that values be taken from a controlled list. Controlled lists can be short ("yes" "no" "maybe") or they can be long (such as the Library of Congress Subject Headings, which number about 250,000).

To define a property as taking a member of a controlled list as its value using the terms of the DC Application Profile, the controlled list is called a "Vocabulary Encoding Scheme" and the list itself is represented with its identifier, a URI. Use of a list can be mandatory or optional.

This code illustrates a "subject" property that optionally can use terms from the LC Subject Heading list.


```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14">
  <DescriptionTemplate>
    <StatementTemplate type="nonliteral">
      <Property>http://purl.org/dc/terms/subject/</Property>
      <NonLiteralConstraint>
        <VocabularyEncodingSchemeOccurrence>optional
        </VocabularyEncodingSchemeOccurrence>
        <VocabularyEncodingSchemeURI>http://purl.org/dc/terms/LCSH
        </VocabularyEncodingSchemeURI>
      </NonLiteralConstraint>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

This code shows a subject property that requires the use of a term from the LCSH list:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14
C:\daisy\Business\dcmi\dcmi-dsp2.xsd">
  <DescriptionTemplate>
    <StatementTemplate type="nonliteral">
      <Property>http://purl.org/dc/terms/subject/</Property>
      <NonLiteralConstraint>
        <VocabularyEncodingSchemeOccurrence>
          mandatory
        </VocabularyEncodingSchemeOccurrence>
        <VocabularyEncodingSchemeURI>http://purl.org/dc/terms/LCSH
        </VocabularyEncodingSchemeURI>
      </NonLiteralConstraint>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

If you have a short list that you wish to embed in the DSP you can do that using the ["LiteralOption"](#). Each term is listed, and values must be taken from the list:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14
C:\daisy\Business\dcmi\dcmi-dsp2.xsd">
  <DescriptionTemplate>
    <StatementTemplate type="nonliteral">
      <Property>http://purl.org/dc/terms/subject/</Property>
      <NonLiteralConstraint>
        <ValueStringConstraint minOccurs="1" maxOccurs="1">
          <LiteralOption>red</LiteralOption>
          <LiteralOption>white</LiteralOption>
          <LiteralOption>blue</LiteralOption>
        </ValueStringConstraint>
      </NonLiteralConstraint>
    </StatementTemplate>
  </DescriptionTemplate>
</DescriptionSetTemplate>
```

You can support multiple languages with the [LiteralOption](#) by including the XML "lang" attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescriptionSetTemplate xmlns="http://dublincore.org/xml/dc-dsp/2008/01/14" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance" xsi:schemaLocation="http://dublincore.org/xml/dc-dsp/2008/01/14
C:\daisy\Business\dcmi\dcmi-dsp2.xsd">
  <DescriptionTemplate>
    <StatementTemplate type="nonliteral">
      <Property>http://purl.org/dc/terms/subject/</Property>
```

```
<NonLiteralConstraint>
  <ValueStringConstraint minOccurs="1" maxOccurs="1">
    <LiteralOption lang="en">red</LiteralOption>
    <LiteralOption lang="en">white</LiteralOption>
    <LiteralOption lang="en">blue</LiteralOption>
    <LiteralOption lang="it">rosso</LiteralOption>
    <LiteralOption lang="it">azzurro</LiteralOption>
    <LiteralOption lang="it">bianco
      <LiteralOption>
    </ValueStringConstraint>
  </NonLiteralConstraint>
</StatementTemplate>
</DescriptionTemplate>
</DescriptionSetTemplate>
```

Contents

- 1 "Application Profiles for Linked Data: models and requirements"
 - 1.1 Description
 - 1.2 Attendees (please add your name or write to tbaker@tbaker.de)
 - 1.2.1 Part 1 (14:00-15:30): Review of DCMI Abstract Model, brainstorming on requirements
 - 1.2.2 Part 2 (16:00-17:30): Emerging models and new uses for application profiles

"Application Profiles for Linked Data: models and requirements"

- Joint meeting, DCMI Architecture Forum and W3C Library Linked Data Incubator Group
- Chairs/Convenors: Tom Baker, Emmanuelle Bermès, Antoine Isaac
- Friday, 22 October 2010
 - Starts at 14:00 <http://www.timeanddate.com/worldclock/fixedtime.html?month=10&day=22&year=2010&hour=18&min=00&sec=0&p1=0>
- Part of [DC-2010](#) conference, 20-22 October 2010, Pittsburgh
 - Description in [conference program](#)
 - Note: [Registration at DC-2010](#) is required - early bird rates until September 10
- Preparation of this meeting is taking place on the following mailing lists:
 - [DCMI Architecture Forum](#) - open subscription
 - [Library Linked Data XG Community list](#) - open subscription
- [Remote participation](#)
 - http://www.w3.org/2005/Incubator/lld/wiki/Telecons#Connection_info

Description

This two-part meeting will look at current approaches to application profiles -- methods for documenting usage patterns in descriptive metadata to promote the design of compatible metadata applications and maximize the coherence of metadata in a Linked Data environment. Starting with a review of the approach based on the DCMI Abstract Model, including the Description Set Profile constraint language and Singapore Framework for Dublin Core Application Profiles, the meeting will also consider other emerging approaches to specifying and documenting metadata usage patterns. By taking a fresh look at requirements in a rapidly evolving environment, this meeting aims at identifying and prioritizing areas where future work may be needed.

Attendees (please add your name or write to tbaker@tbaker.de)

- Emmanuelle Bermès
- Tom Baker
- Bernard Vatant
- Alexander Haffner
- Kai Eckert
- Antoine Isaac
- Lars G. Svensson
- Jon Phipps
- Marcia Zeng
- Karen Coyle (not 100%; have meeting conflict)
- Jeff Young
- Gordon Dunsire (not 100%; have meeting conflict for Part 1)
- Michael Panzer

- Pete Johnston (remote)
- Stu Weibel
- Mark van Assem
- Peter Murray
- Ed Summers
- Ambjörn Naeve (remote)
- Mikael Nilsson (remote)
- Joe Tennis

Part 1 (14:00-15:30): Review of DCMI Abstract Model, brainstorming on requirements

- DCMI Abstract Model and Singapore Framework for Dublin Core application profiles - a review: Tom Baker and Pete Johnston (remotely)
 - Required reading:
 - [DCMI Abstract Model: past, present, future](#)
 - Background reading:
 - [DCMI Abstract Model](#)
 - [Singapore Framework for Dublin Core Application Profiles](#)
 - [Description Set Profiles: a constraint language for Dublin Core application profiles](#)
- Application profiles expressed in OWL2 and rules languages (RIF): Jeff Young and Michael Panzer
 - EPrints/SWAP (as an example) in UML/OWL/XML Schema, compared to expression as DC Application Profile
 - Background reading:
 - <http://dublincore.org/scholarwiki/SWAPDSP>
 - <http://dublincore.org/scholarwiki/SWAPDSP?action=DSP2XML>
 - <http://dublincore.org/documents/2008/10/06/dsp-wiki-syntax/>
 - <http://dublincore.org/documents/2008/10/06/dsp-wiki-syntax/DescriptionSetProfile-dist.zip>
 - <http://dublincore.org/documents/dc-dsp/#sect-7>
 - <http://www.ukoln.ac.uk/repositories/digirep/index/Model>

Part 2 (16:00-17:30): Emerging models and new uses for application profiles

- Application profiles for subject domains (FRSAD): Marcia Zeng, Gordon Dunsire, Maja Zumer
 - How formally (or informally) has this style of application profile been defined?
 - In what ways are application profiles for subject domains different from APs for descriptive metadata?
 - Lightning presentation on FRBR in OWL and ISBD in DCAP: some issues
- Requirements for application profiles and open questions (1h)
 - Do RDF and Linked Data need standard approaches to "application profiles"?
 - For expressing a metadata realm and communicating this to others for purposes of sharing or harmonization?
 - For providing a basis to validate metadata?
 - How formally does an application profile need to be defined?
 - Can we identify distinct scenarios for different types of application profiles?

A review of the DCMI Abstract Model with scenarios for its future

Tom Baker, Pete Johnston

Identifier: <http://dublincore.org/architecturewiki/DcamInContext>

Date : 2010-10-15



About this review


This paper:


- contextualizes the DCMI Abstract Model (DCAM) within the history of DCMI and Web standards;
- describes the DCAM approach with reference to Resource Description Framework (RDF);
- proposes alternative scenarios for the future development of DCAM;
- assesses the impact of alternative scenarios on specifications that depend on DCAM.

The paper has been produced for discussion on 22 October at a  [Joint Meeting of the DCMI Architecture Forum and the W3C Library Linked Data Incubator Group](#). To the extent possible, the meeting will try to determine a realistic way forward for DCAM.


A short history of Dublin Core



The Dublin Core community's first step was the definition of twelve (later fifteen) "elements" in 1995, supplemented in 1997 by the addition of a notion of "qualifiers" of those elements, the  ["Canberra qualifiers"](#) [3]. In July 2000, with the publication of  ["Dublin Core Qualifiers"](#), qualifiers were differentiated into "element refinements" and "element encoding schemes" [4].

This "typology of terms" formed the basis of a guide for DCMI Usage Board vocabulary maintenance decisions as  ["DCMI Grammatical Principles"](#) created in February 2003, which further differentiated "encoding schemes" into "vocabulary encoding schemes" and "syntax encoding schemes" [5].



In addition to these initiatives to define a "typology of terms", and specific sets of terms based on that typology, work was also done to specify the format-independent "abstract data structure", or "abstract syntax", within which references to those terms were made. This work is usefully summarised in the 2000 D-Lib article  ["A Grammar of Dublin Core"](#) [5a] which articulates a "grammar" of "statements" made up of:

- an implicit reference to the thing being described
- a reference to one of the 15 "properties" or "elements" of the DCMES
- a "property value" ("an appropriate literal")
- (optionally) references to one or more "qualifiers"

DCMI's first specification for a concrete syntax for Dublin Core metadata  [RFC2731 Encoding Dublin Core Metadata in HTML](#) [5b] (1999) was based (at least loosely/informally) on this model.

Starting in 1997, W3C's effort to define a Resource Description Framework (RDF) paralleled this work within DCMI, culminating in a first W3C Recommendation  ["RDF Model and Syntax Specification"](#) in February 1999 [6] and, following an extensive review process, a second W3C Recommendation  ["RDF Concepts and Syntax"](#) in February 2004 [7].

This led to the development within DCMI of two further sets of "encoding guidelines" for using Dublin Core terms in RDF, each of which specified the use of slightly different patterns/conventions:

-  [Expressing Simple Dublin Core in RDF/XML](#) [7a] (2002): specified the use of literal values only
-  [Expressing Qualified Dublin Core in RDF](#) [7b] (2002): a wider range of conventions, including use of Bag, Seq, Alt

The rationale for the DCMI Abstract Model

By the early 2000s, in addition to applications based on the specifications above, there were a growing number of "Dublin Core metadata" implementations which made no reference to any "abstract syntax", and the resulting picture was one of a landscape in which interoperability between applications was problematic. The RDF abstract syntax was recognized by parts of the Dublin Core community as a crucial development -- and the "grammar of Dublin Core" model was intended in part to popularize its notion of "statement-based" metadata -- but on the whole, RDF was seen by a large part of the Dublin Core community as a research project of dubious practical value. More specifically, RDF was seen less as a

fundamentally different way of conceptualizing metadata and more as an alternative XML format for metadata -- and one that compared unfavorably to apparently simpler and more readable XML formats.

Given the political difficulty of directly promoting the RDF abstract syntax as a common basis for metadata, work on the DCMI Abstract Model was undertaken in 2003 in an attempt:

- to clarify and formalize the "home-grown" model of metadata that had emerged from early Dublin Core workshops, and formed the basis of the DCMI Grammatical Principles; and
- (particularly in the second revision) to align that model with that of the RDF abstract syntax and RDF semantics.

This effort resulted in a first [DCMI Recommendation in March 2005](#) [1] and a second, revised [DCMI Recommendation in June 2007](#) [2]. The term "element" was de-emphasized in favor of "RDF property". "Element refinements" were defined as "RDF sub-properties". Syntax Encoding Schemes were defined as RDF datatypes. The generic term "encoding scheme" and the even more generic term "qualifier" were further de-emphasized [8].

The DCMI Abstract Model defines an "abstract syntax" based on a data structure it calls the "description set". The specification [Expressing Dublin Core metadata using the Resource Description Framework \(RDF\)](#) specification defines a mapping from the "description set" model to the RDF abstract syntax.

The revised DCMI Abstract Model of 2007 became the basis for revised concrete syntax specifications ([DC-HTML](#) [8a], [DC-DS-XML](#) [8b], [DC-TEXT](#) [8c]).

In 2009, Mikael Nilsson outlined a draft for an "RDF-based version" of the DCMI Abstract Model. The outline points towards an abstract model dramatically simplified with respect to the 2007 model. Entirely missing are two of the three component models of the 2007 model: the DCMI Resource Model (to be replaced by a simple reference to RDF) and the DCMI Vocabulary Model (to be replaced by a simple reference to the RDF Vocabulary Description Language, also known as RDF Schema). The intention of the "RDF-based version" was to define the abstract model entirely in terms of RDF while maintaining the "interface" to constructs defined in the third component of the 2007 model, the Description Set Model. As of 2010, this work remains at the stage of an early draft.

DC Application Profiles and Description Set Profiles

In 2000, the notion of an "application profile" was put forward and quickly became a central point of reference for the Dublin Core community. As originally proposed, an application profile was the specification of a particular pattern of "elements" and "encoding schemes" "used" in the context of a particular application or to describe a particular type of resource. In practice, this very general conceptualization was interpreted in a multiplicity of ways, resulting in a wide range of incompatible constructs.

The DCMI Abstract Model, with its formalization of an abstract syntax, provided the basis for a number of documents which sought to provide a formal specification of the "DC application profile" concept:

- A definition of a constraint language, ["Description Set Profiles"](#) [9]
- The ["Singapore Framework for Dublin Core Application Profiles"](#) (2007) [10];
- [application profile review criteria](#) used by the DCMI Usage Board [11]; and
- ["Interoperability Levels for Dublin Core Metadata"](#) [12]

A user-oriented document, ["Guidelines for Dublin Core Application Profiles"](#), was written to guide users through the process of designing and creating application profiles on the basis of the DCMI Abstract Model [16].

In addition to the XML syntax and RDF representations specified in the Description Set Profile document itself, a wiki syntax [16a] was also developed, together with a [MoinMoin](#) extension [16b] which generated both a tabular human-readable view and an XML representation of an application profile. The Description Set Profile was intended to be used for applications such as the automatic configuration of metadata editing tools and the generation of schemas for document validation.

Other approaches to the "structural constraints"/"validation" question have been explored within the Semantic Web community:

- In 2005, Dan Brickley put forward a proposal for conceptualizing [Dublin Core application profiles as "query profiles"](#) [13].
- In 2007, Alistair Miles proposed ["Son of Dublin Core"](#), a draft approach for encoding and validating "graph-based metadata" using a concrete XML syntax and language for expressing application-specific syntax constraints over a metadata graph [14].
- In 2009, at the Bristol Vocamp, Dave Reynolds noted [the use of OWL as a constraint language](#) [14a]: "there is no problem at all with creating tools which make a closed world and unique name assumption for the purposes of data validation. They aren't violating the OWL semantics, so long as they don't purport to be doing OWL consistency checking, they are doing a different job but a useful one."

Relationship of the DCMI Abstract Model to RDF

While the 2009 draft "RDF-based" revision of the DCMI Abstract Model was never developed beyond the outline stage, this discussion paper uses its basic ideas as a starting point.

Specifically, this paper ignores the Resource Model and Vocabulary Model defined in the 2007 Abstract Model and focuses exclusively on its centerpiece: the Description Set Model. As a guide to how the constructs of this model translate into RDF, this paper additionally follows the 2008 guidelines, ["Expressing Dublin Core metadata using the Resource Description Framework"](#) (referred to hereafter by its short name, "DC-RDF") [18].

In the 2007 Abstract Model, the Description Set Model specifies both a set of syntactic elements (things found in data) and a set of referents in the real world (things to which the syntactic elements may be interpreted to refer). If the DCMI Abstract Model is to be based on the RDF abstract syntax, we can limit our analysis here to the syntactic elements. These include grouping constructs (Description Set, Description, Statement, Non-Literal Value Surrogate, and Literal Value Surrogate) and slots for URIs and character strings (Described Resource URI, Property URI, Value URI, Vocabulary Encoding Scheme URI, Syntax Encoding Scheme URI, Value String Language, Plain Value String, and Typed Value String). One might think of these slots as components of the DCAM abstract syntax that can be tested. In the 2007 specification, these syntactic elements are described using UML, but they are more popularly depicted in the form of a nested metadata template, as in Figure 1 below.

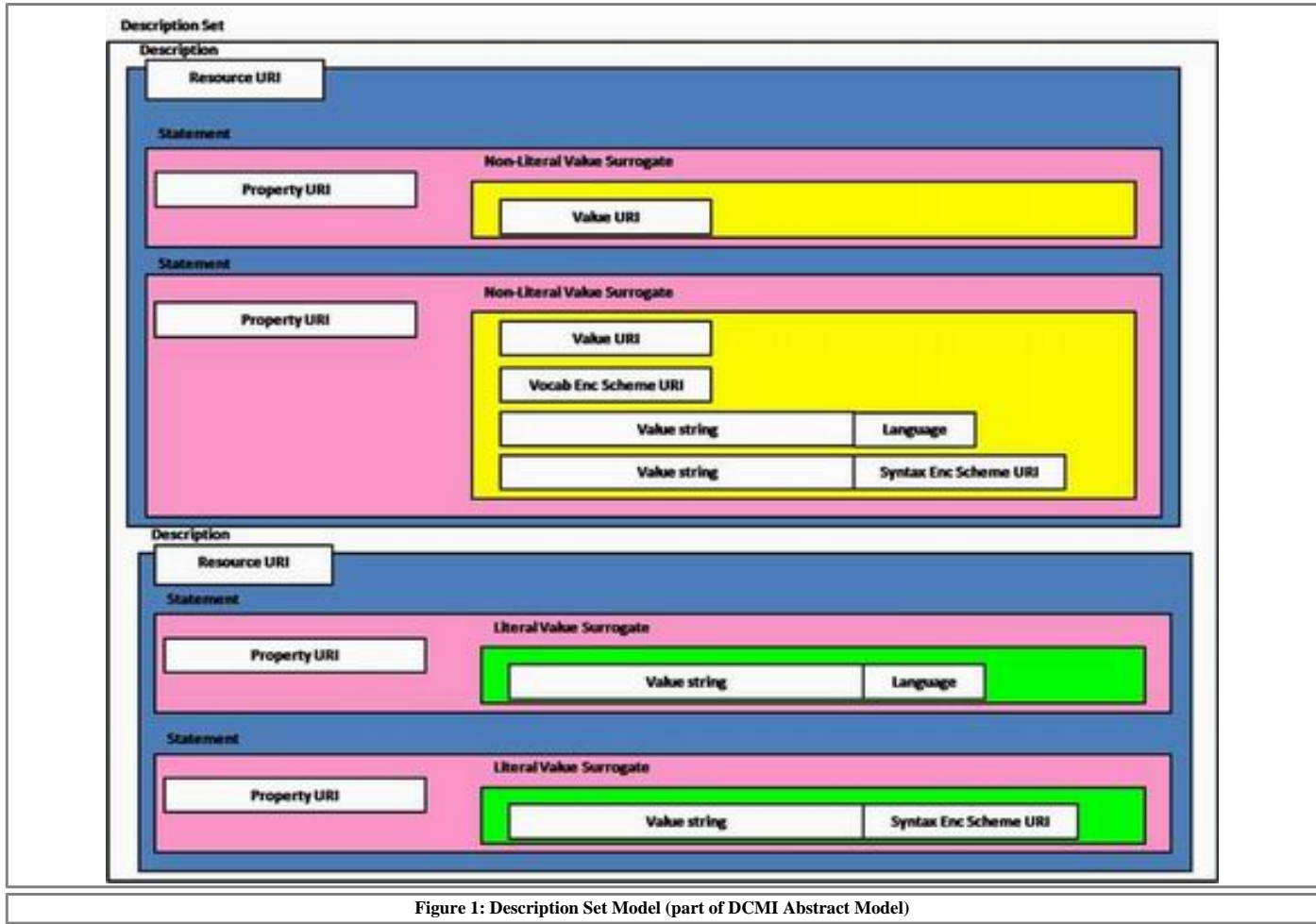
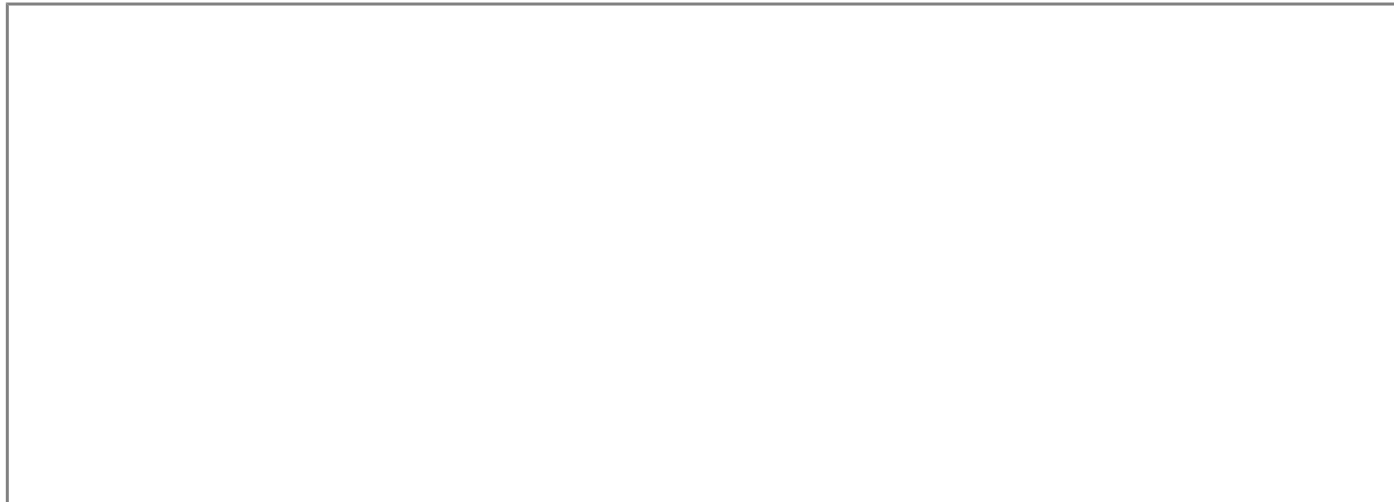
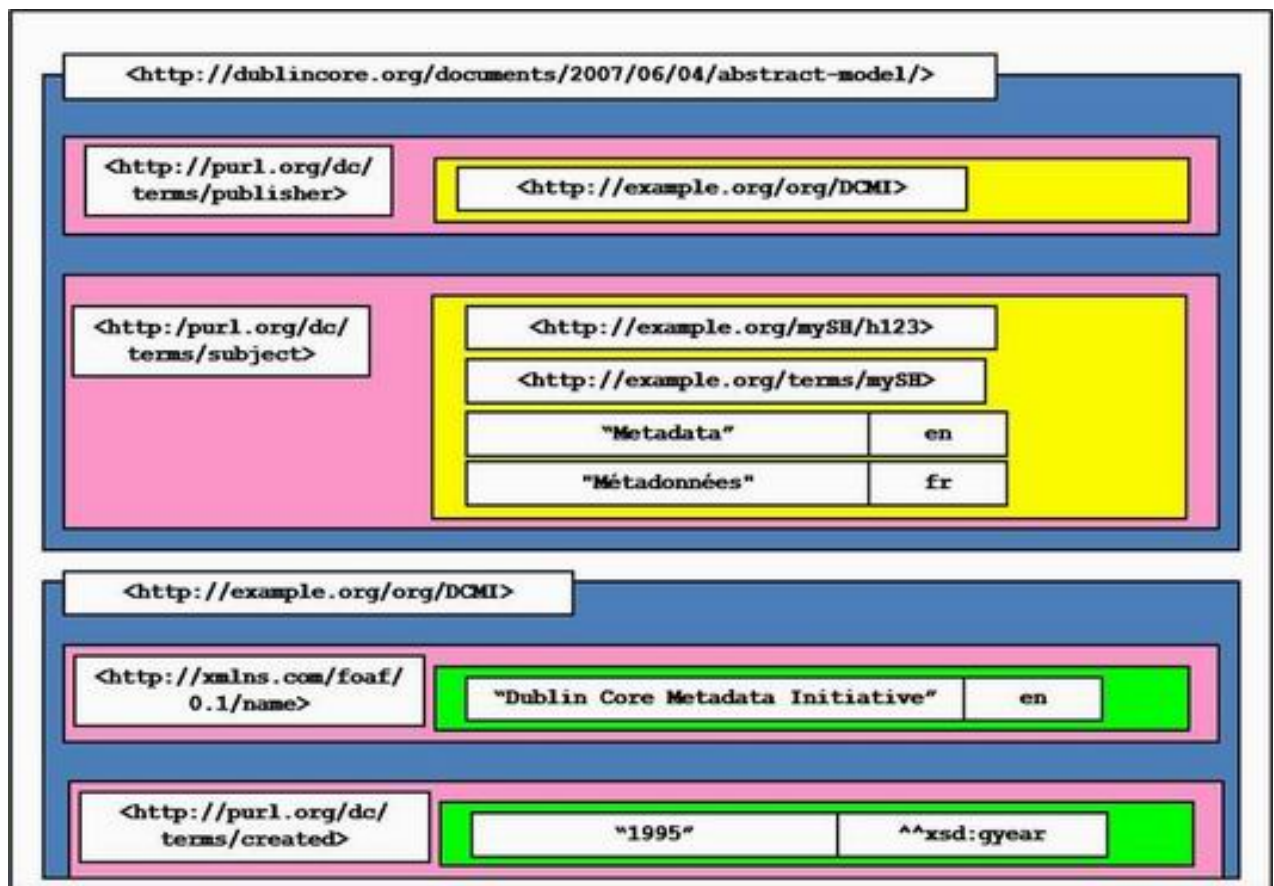


Figure 1: Description Set Model (part of DCMI Abstract Model)

As an illustration of how the syntactic elements of the Description Set Model are used, Figure 2 shows a set of example information values in the placeholders corresponding to those shown in Figure 1.





How the elements of the Description Set Model relate to RDF is roughly visualized in Figure 3 and described in more detail in Appendix B below.

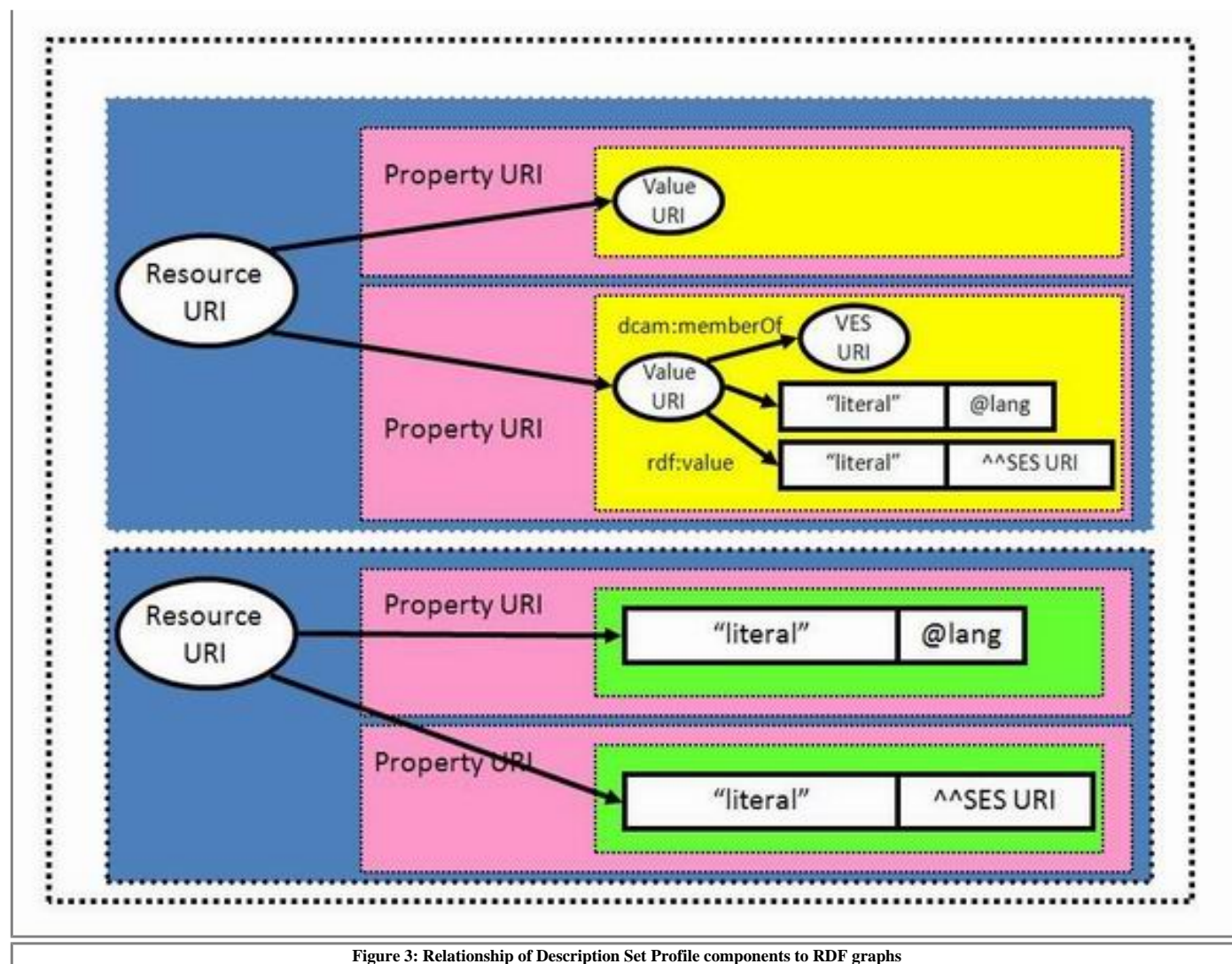


Figure 3: Relationship of Description Set Profile components to RDF graphs

The Description Set Profile constraint language

The March 2008 specification ["Description Set Profiles: a constraint language for Dublin Core application profiles"](#) (hereafter DC-DSP) [9] provides a language for specifying a set of constraints on the "description set" construct defined by the DCMI Abstract Model. The word "constraints" evokes the notion that the set of possible ways that the slots defined by the Description Set Model may be filled is infinite, and these infinite possibilities are being configured, or "constrained", in specific ways for specific content.

In the terminology of the DC-DSP specification, sets of constraints are expressed in "templates". Templates use constraints to specify some community- or application-specific rules for the contents of a description set: first, the descriptions, description by description (Description Templates); then within each description, statement by statement (Statement Templates); and within each statement, slot by slot (Constraints).

Conceptually, templates are like cookie cutters for mass-producing actual descriptions and statements in real instance metadata. Actual descriptions and statements in real instance metadata, in turn, are conceptualized as "matching" specific templates according to a matching algorithm.

Very broadly, DC-DSP provides a language for specifying things such as:


- Minimum and maximum allowable occurrences of actual descriptions matching a given Description Template within a Description Set.
- Whether descriptions matching a given Description Template may stand alone within a Description Set or whether their presence depends on the presence of descriptions matching another Description Template (example: "no stand-alone descriptions of authors in the absence of a description of the book they wrote").
- Minimum and maximum allowable occurrences of actual statements matching a given Statement Template within a Description.
- For slots designed to hold URIs -- such as Property URI, Value URI, and Value Encoding Scheme URI -- whether it is mandatory, optional, or disallowed that the given slot be filled in a given Statement Template, or a list of URIs that may be used in the given slot (as in: "the slot labeled Property URI must contain one of these URIs").
- For slots designed to hold character strings or language tags, whether it is mandatory, optional, or disallowed that the given slot be filled in a given Statement Template, or a list of character strings that may be used in the given slot.

DCAM in 2010

It is difficult to track the use of freely available specifications once they are released on the Web, but as of 2010, DCMI is not aware that any of the Abstract-Model-related specifications, with the possible exception of specific syntax guidelines, have been widely implemented.

Rather than building a bridge from more traditional metadata communities to the Semantic Web, the Abstract Model appears to have fallen between two stools -- its use of the "description set" abstraction perplexing to users more accustomed to metadata specifications defined in terms of a concrete syntax, and its added layer of Dublin-Core-specific terminology confusing to users comfortable with the RDF model.

Since 2006, however, the rapid success of Linked Data has given the notion of Semantic Web, based on Resource Description Framework (RDF), wider visibility and acceptance. If Linked Data is crossing the chasm to widespread deployment, and the conceptual model of RDF is reaching a wider community, is there still a need for the bridge that the DCMI Abstract Model was intended to be?

As of 2010, moreover, new Semantic Web specifications such as Simple Knowledge Organization System (SKOS) address issues that overlap with those of the DCMI Abstract Model.  [Discussions in the Semantic Web community](#) about a new version of RDF ("RDF 2") point towards further developments in the core Semantic Web standards, such as Named Graphs, that parallel some of the more innovative features of the DCMI Abstract Model [17]. To what extent can the DCMI Abstract Model already be expressed in terms of newly-mainstream Linked Data concepts? If aspects of the DCMI Abstract Model still cannot be expressed with more mainstream concepts, what are the prospects for being able to do so in the foreseeable future and, more to the point, what steps should be taken in the meantime to revise, deprecate, or replace the DCMI Abstract Model?

The question does not affect just the Abstract Model, but the suite of related specifications, syntax guidelines, and user documentation built on the Abstract Model. What requirements are reflected in this considerable body of work, which of these requirements now appear to be most important, and how should we best proceed to address those requirements?

Scenarios for the future of DCAM

Scenario 1. DCMI carries on developing DCAM as before

- DCMI carries on developing DCAM as before, incrementally improving the DCAM and Description Set Profile specifications, with a work plan for developing further concrete syntaxes based on DCAM.
- Questions:
 - Is there a demonstrated interest?
 - Who would edit the specs?
 - How would testing and review be managed?

Scenario 2a. DCMI develops a "DCAM 2" spec as the basis for new work

- DCMI develops a "DCAM 2" specification -- simplified and better aligned with RDF
- In Variant 2a, the improved DCAM 2 specification would be taken as the new basis
 - for the Description Set Profile language of structural constraints for application profiles
 - for a workplan to develop new and improve existing concrete syntaxes on the basis of "DCAM 2".
- Questions:
 - Is there a demonstrated interest in "DCAM 2"?
 - As in #1 above: Who would edit the specs? How would review and testing be managed?
 - What would be the impact of "DCAM 2" on specifications in the existing "DCAM family of specifications"?

Scenario 2b. DCMI develops "DCAM 2" as a transitional explanatory document

- DCMI develops "DCAM 2" as an explanation for how the legacy DCAM model relates to RDF
- In Variant 2b, "DCAM 2" would serve the purposes of:
 - Clarification, for the Dublin Core community generally and for users of DCAM in particular, of how DCAM relates to RDF and Linked Data
 - Role of a "transitional" specification, to be deprecated over time in favor of RDF
- No workplan for new concrete syntaxes would be undertaken.
- Questions
 - Is there interest in the clarification that a "DCAM 2" spec would provide?
 - Who would edit "DCAM 2"?
 - What should be done with the existing "DCAM family" of specifications? (Currently, most are DCMI Recommendations or DCMI Recommended Resources.)
 - What is an "application profile"? Is it based on DCAM/DSP or on the RDF abstract syntax?

Scenario 3. DCMI deprecates the DCAM abstract syntax and embraces RDF abstract syntax

- Rather than explain in any detail how the legacy DCAM model relates to RDF, DCMI simply depicts DCAM as a "product of its time" and henceforth promotes the RDF abstract syntax.
- Questions
 - Are there current users of DCAM who would be negatively impacted?

- What should be done with the existing "DCAM family" of specifications (e.g., in terms of status as DCMI Recommendations or Recommended Resources)?
- What explanation would DCMI provide, particularly with regard to application profiles -- hitherto a central aspect of the DCMI message? What is an "application profile" if it is not based on DCAM, DSP, and the Singapore Framework?

Scenario 4. DCMI does nothing - DCAM is simply left untouched

- DCMI does nothing to change the statuses of DCAM-related specifications.
- DCAM and DSP are in effect "frozen" and de-emphasized, with no particular explanation.
- Scenario 4 is the most "economical" in terms of (human) resources.
- Questions
 - If DCMI does not, in fact, stand behind specifications that continue to bear the status of DCMI Recommendations, what would be the cost to DCMI in terms of credibility?

General issues for discussion












DCAM abstract syntax versus the RDF abstract syntax

- Should DCAM dissolve into mainstream RDF? For example:
 - Are Descriptions and Description Sets expressible as Named Graphs?
 - Are there significant differences between Vocabulary Encoding Schemes and SKOS Concept Schemes?
 - Do aspects of the DCAM mapping to RDF need to be revisited (e.g., the `rdf:value` for value strings associated with object nodes, as opposed to `skos:prefLabel`, `rdfs:label`, `foaf:name`, `skos:notation`, or `dcterms:title`).

Application Profiles

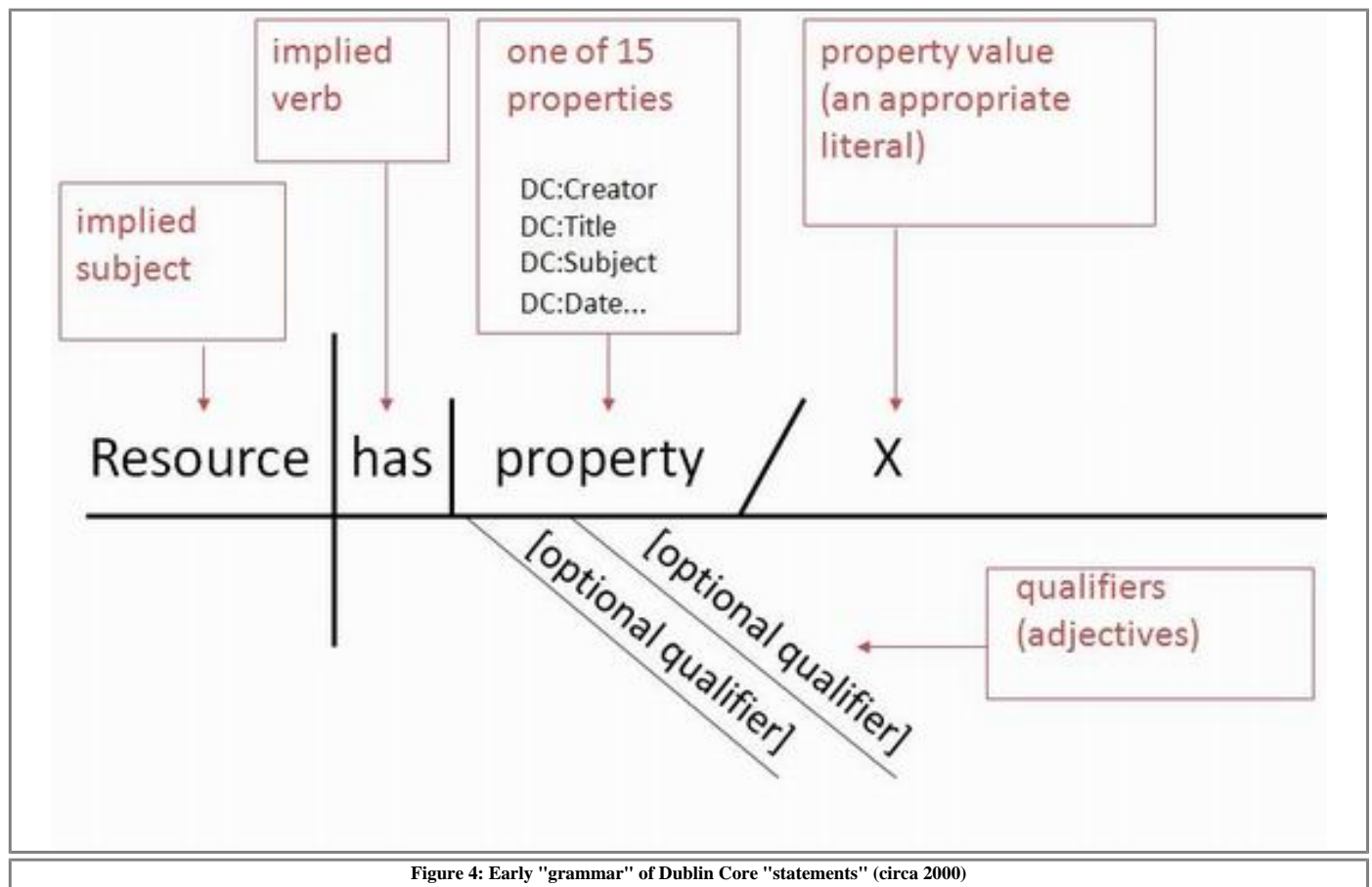
- Does RDF need a notion of Application Profiles?
- If so, what are the requirements?
- Do application profiles need to express constraints?
 - If not with DCAM, how should patterns of constraints at the level of RDF graphs be expressed?
 - Using syntax pattern checks (patterns "in the graph" rather than "in the world") along the lines of the Description Set Profile constraint language? Or might it be enough to use SPARQL query patterns?
 - Using OWL applied with closed-world Assumptions?
- Is it useful, as in the Singapore Framework, to distinguish strongly between declared vocabularies and declared vocabularies as used and constrained in data formats?
 - Should constraints be wired into the formal specifications of vocabularies?
 - Or should constraints be expressed as patterns matched to the data?

Appendix A. The DCAM family of specifications

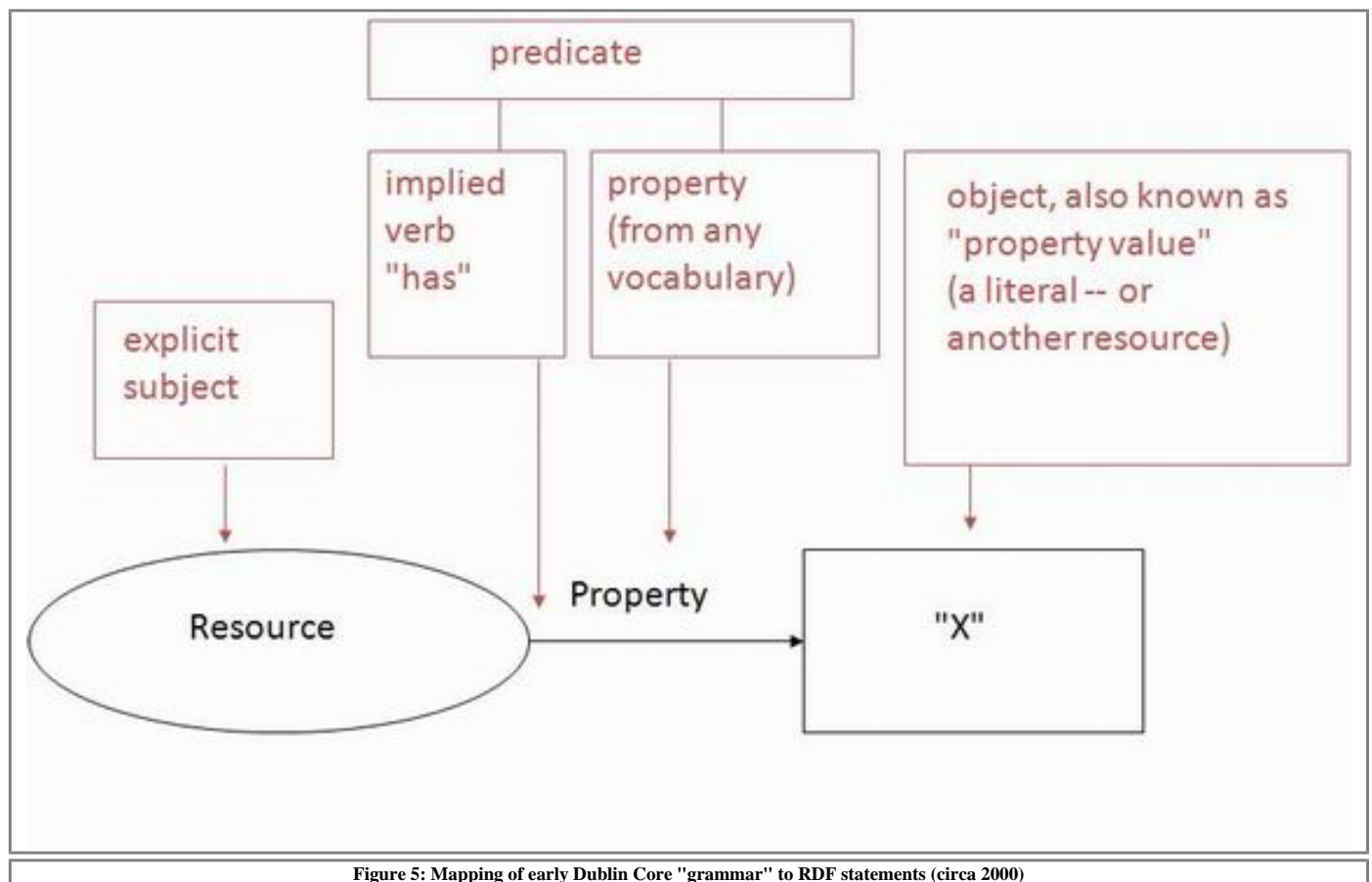
-  [DCMI Abstract Model](#)
-  [Description Set Profiles: A constraint language for Dublin Core Application Profiles](#)
-  [Singapore Framework for Dublin Core Application Profiles](#)
-  [DCMI Usage Board Criteria for the Review of Application Profiles](#)
-  [Interoperability Levels for Dublin Core Metadata](#)
-  [Guidelines for Dublin Core Application Profiles](#)
-  [DC-TEXT: A text syntax for Dublin Core metadata](#)
-  [Expressing Dublin Core metadata using the Resource Description Framework \(RDF\)](#)
-  [Expressing Dublin Core Description Sets using XML \(DC-DS-XML\)](#)
-  [Expressing Dublin Core metadata using HTML/XHTML meta and link elements \(DC-HTML\)](#)
-  [Guidelines for implementing Dublin Core in XML \(DC-XML\)](#)

Appendix B. Relationship of the Description Set Model to RDF

It is worth noting that the Dublin Core "grammar" for "statements" of circa 2000, pictured in Figure 4, was in part an early attempt to popularize the notion of metadata as being based on meaningful "statements" by way of analogy to commonly understood notions of natural-language grammar.



As part of the model, a rough mapping of the Dublin Core "grammar" to RDF "statements" was provided, as pictured in Figure 5.





This appendix examines how the syntactic elements of the Description Set Model, the central component of the DCMI Abstract Model of 2005-2007, relate to RDF -- both as it currently exists and as it is likely to evolve in the medium term. Note that the 2007 abstract model defines these elements both as syntactic elements and in terms of the "things in the world" to which these elements refer. For simplicity of exposition, this analysis focuses exclusively on the syntactic elements.

Grouping constructs in the Description Set Model

The grouping constructs in the Description Set Model are:

- Description Set: A set of one or more Descriptions.
- Description: A set of statements about one, and only one, resource. Syntactically, the Description consists of an optional Description Resource URI plus one or more Statements.
- Statement: A Property URI slot plus a set of slots grouped either in a Non-Literal Value Surrogate or a Literal Value Surrogate.
- Non-Literal Value Surrogate: A set of slots consisting of any combination of zero to one Value URIs, zero to one Vocabulary Encoding Scheme URIs, zero to many Value Strings, each of which may be a Plain Value String (with an optional Value String Language), or a Typed Value String (with a Syntax Encoding Scheme URI).
- Literal Value Surrogate: A slot or set of slots consisting of either a Plain Value String (with an optional Value String Language), or a Typed Value String (with a Syntax Encoding Scheme URI).

Slots in the Description Set Model

The slots in the Description Set Model map to constructs in the  [RDF abstract syntax](#) [19]. Following the  [DC-RDF guidelines](#) [18], these slots may be characterized in terms of the RDF abstract syntax as follows:

- Described Resource URI: In RDF terms, the URI in the Described Resource URI slot identifies the resource that is the subject of a Description. When a URI is present in this slot, that URI is the subject of triples about the described resource. In the absence of a Described Resource URI (i.e., the slot is empty), a blank node is the subject of triples about the described resource.
- Property URI: A URI identifying a property that is the predicate of an RDF triple about the described resource.
- Value URI: A URI identifying a resource that is the object of an RDF triple about the described resource. In the absence of a Value URI (i.e., the slot is empty), the object is a blank node.
- Vocabulary Encoding Scheme URI: A URI identifying an enumerated set of resources ("Vocabulary Encoding Scheme") to which the object of the RDF triple about the described resource belongs. This is expressed with a triple in which the subject is a reference (whether a blank node or URI) to the value resource, the predicate is the property "dcm:memberOf", and the object is the URI in the Vocabulary Encoding Scheme URI slot.
- Plain Value String: A character string with an optional language tag, an RDF plain literal. For a Value String within a Literal Value Surrogate, the plain literal is directly the object of an RDF triple about the described resource. For a Value String within a Non-Literal Value Surrogate, the plain literal is the object of an RDF triple in which the predicate is the property "rdf:value", and the subject is a reference (whether a blank node or URI) to the value resource.
- Value String Language: The language tag used together with the character string in the Plain Value String slot to form an RDF literal.
- Typed Value String: A character string associated with exactly one URI in the Syntax Encoding Scheme URI slot, an RDF typed literal. For a Value String within a Literal Value Surrogate, the typed literal is directly the object of an RDF triple about the described resource. For a Value String within a Non-Literal Value Surrogate, the typed literal is the object of an RDF triple in which the predicate is the property "rdf:value", and the subject is a reference (whether a blank node or URI) to the value resource.
- Syntax Encoding Scheme URI: A URI identifying an RDF datatype, which for legacy reasons is known in the DCMI Abstract Model as a "syntax encoding scheme".

References

- [1] <http://dublincore.org/documents/2005/03/07/abstract-model/>
- [2] <http://dublincore.org/documents/2007/06/04/abstract-model/>
- [3] <http://www.dlib.org/dlib/june97/metadata/06weibel.html>
- [4] <http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/>
- [5] <http://dublincore.org/usage/documents/2003/02/07/principles/>
- [5a] <http://www.dlib.org/dlib/october00/baker/10baker.html>
- [5b] <http://www.ietf.org/rfc/rfc2731.txt>
- [6] <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
- [7] <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
- [7a] <http://dublincore.org/documents/2002/07/31/dcmes-xml/>
- [7b] <http://dublincore.org/documents/2002/05/15/dcq-rdf-xml/>
- [8] <http://dublincore.org/documents/abstract-model/#app-a> (relationship to legacy terminology)
- [8a] <http://dublincore.org/documents/2008/08/04/dc-html/>
- [8b] <http://dublincore.org/documents/2008/09/01/dc-ds-xml/>

- [8c] <http://dublincore.org/documents/2007/12/03/dc-text/>
- [9] <http://dublincore.org/documents/dc-dsp/>
- [10] <http://dublincore.org/documents/singapore-framework/>
- [11] <http://dublincore.org/documents/2009/03/02/profile-review-criteria/>
- [12] <http://dublincore.org/documents/interoperability-levels/>
- [13] <https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind0509&L=DC-RDF-TASKFORCE&P=R2034&I=-3>
- [14] http://web.archive.org/web/20080214232032/http://isegserv.itd.rl.ac.uk/sodc/SODC-0_2/
- [14a] <http://www.amberdown.net/2009/10/owl2-for-rdf-vocabs/>
- [15] <http://dublincore.org/architecturewiki/DCAM-2.0>
- [16] <http://dublincore.org/usage/documents/profile-guidelines/>
- [16a] <http://dublincore.org/documents/2008/10/06/dsp-wiki-syntax/>
- [16b] <http://dublincore.org/documents/2008/10/06/dsp-wiki-syntax/DescriptionSetProfile-dist.zip>
- [17] <http://www.w3.org/2001/sw/wiki/RDF/NextStepWorkshop>
- [18] <http://dublincore.org/documents/dc-rdf/>
- [19] <http://www.w3.org/TR/rdf-concepts/>
- [20] <http://www.w3.org/2001/sw/wiki/RDF/NextStepWorkshop>
- [21] <http://www.w3.org/2010/06/rdf-work-items/table>
- [22] http://www.w3.org/2001/sw/wiki/index.php?title=RDF_Core_Work_Items&oldid=1990#Graph_Identification
- [23] <http://dublincore.org/dcmirdataskgroup/apDesigns>

From tbaker Mon Oct 11 10:26:20 2010
From: Mikael Nilsson <mikael@nilsson.name>
To: public-lld@w3.org
Content-Type: text/plain; charset="UTF-8"
Date: Mon, 11 Oct 2010 16:14:56 +0200
Message-ID: <1286806496.4536.39.camel@daneel>
Subject: Returning to OWL and application profiles

Hi all!

I'm in the middle of finalizing my thesis on metadata interoperability and harmonization, and I'm right now formulating a section on RDF and application profiles, so the discussion I saw here comes at an interesting time for me :-)

The issue from my point of view with using OWL for defining RDF application profiles, is that APs define domain-specific structural constraints while OWL adds semantics to existing classes.

I.e. if I produce an OWL-based AP saying that the cardinality of dc:title is exactly 1, for a specific class, and someone else produces an OWL-based AP saying that the cardinality is 2 for the same class, the result is a *contradiction*.

This differs substantially from the case with application profiles, where the cardinality is not seen as part of the semantics of a class, but rather part of a set of restrictions, external to and independent of the class. Multiple incompatible application profiles are perfectly normal.

Therefore, publishing an OWL ontology defining domain-specific semantics for certain classes or properties is just as bad practice as if someone produces an RDF Schema saying the range of dct:creator is myorg:Employee. This defines new semantics of dct:creator, something that is simply not true, and can cause involuntary contradictions.

The other issue is the open world assumption that I saw Pete mention, i.e. the fact that if an OWL ontology specifies a cardinality of 2 for dc:title, and only one is found, this results in the generation of a new dc:title statement, not in non-validity of the record.

Thus, we would need an alternative semantics for OWL to perform validation.

But that's exactly it - the semantics is "alternative" and on its face, the semantics of the published OWL file is something else entirely.

As a concrete example, if I serve an OWL file from my web server, using application/rdf+xml as suggested by the OWL specs [1], the interpretation will be as RDF triples using the RDF and OWL built-in semantics, thus resulting in the generation of new triples, potentially contradiction with other ontologies, and not in validation as expected.

What *would* work is using application profile specific classes for each separate OWL-based AP, and only constraining them, but that might appear a bit cludgy.

So, I'm a bit unsure regarding using non-standard semantics of OWL. I don't really see a clean solution at the moment.

/Mikael

[1] <http://www.w3.org/TR/owl-ref/#MIMEType>

Date: Mon, 11 Oct 2010 17:05:01 -0400
From: "Young,Jeff (OR)" <jyoung@oclc.org>
To: "Mikael Nilsson" <mikael@nilsson.name>,
<public-lld@w3.org>
Subject: RE: Returning to OWL and application profiles

Mikael,

I think it would be better to encourage the use of
owl:subClassOf and owl:subPropertyOf to resolve this scenario
instead. For example:

General-purpose class:
foo:Widget a owl:Class .

General-purpose property:
bar:title a owl:Property .

Limited-purpose domain:
baz:Widget owl:subClassOf foo:Widget.
baz:title owl:subPropertyOf bar:title .

Domain-specific cardinalities can be applied to terms in the
domain-specific ontology without creating general semantic
clashes. Other limited-purpose domains could do the same.

From: Mikael Nilsson <mikael@nilsson.name>
To: "Young,Jeff (OR)" <jyoung@oclc.org>
Cc: public-lld@w3.org
Date: Tue, 12 Oct 2010 00:09:18 +0200
Message-ID: <1286834958.4536.44.camel@daneel>
Subject: RE: Returning to OWL and application profiles

m=C3=A5n 2010-10-11 klockan 17:05 -0400 skrev Young,Jeff (OR):
> Mikael,

>
> I think it would be better to encourage the use of owl:subClassOf and
> owl:subPropertyOf to resolve this scenario instead. For example:

>
> General-purpose class:
> foo:Widget a owl:Class .

>
> General-purpose property:
> bar:title a owl:Property .

>
> Limited-purpose domain:
> baz:Widget owl:subClassOf foo:Widget.
> baz:title owl:subPropertyOf bar:title .

>
> Domain-specific cardinalities can be applied to terms in the

- > domain-specific ontology without creating general semantic clashes.
- > Other limited-purpose domains could do the same.

Yes, that sort of solves the contradiction issue.

But still, if a cardinality constraint is not met, an OWL reasoner is expected to produce additional statements or infer the equality of some of the values. This still does not amount to a "record validation" tool, unfortunately.

Date: Mon, 11 Oct 2010 21:14:51 -0400
Message-ID: <43b201cb69aa\$d7cc0656\$d71dae84@oa.oclc.org>
From: "Young,Jeff (OR)" <jyoung@oclc.org>
To: "Mikael Nilsson" <mikael@nilsson.name>
Cc: <public-lld@w3.org>
Subject: RE: Returning to OWL and application profiles

My sense is that OWL reasoning is an excellent abstract solution and don't think we should be overly concerned by the need for inferencing. I gave an example of a mechanical relationship between OWL and XML Schema to do "record validation". I suspect that RIF could do even better, but I admit more examples are needed.

Date: Tue, 12 Oct 2010 13:11:13 +0200
From: Antoine Isaac <aisaac@few.vu.nl>
MIME-Version: 1.0
To: Mikael Nilsson <mikael@nilsson.name>, public-lld <public-lld@w3.org>
Subject: Re: Returning to OWL and application profiles

Hi Mikael,

Thanks for starting this interesting thread :-)

- > I'm in the middle of finalizing my thesis on metadata interoperability
- > and harmonization, and I'm right now formulating a section on RDF and
- > application profiles, so the discussion I saw here comes at an
- > interesting time for me :-)
- >
- > The issue from my point of view with using OWL for defining RDF
- > application profiles, is that APs define domain-specific structural
- > constraints while OWL adds semantics to existing classes.
- >
- > I.e. if I produce an OWL-based AP saying that the cardinality of
- > dc:title is exactly 1, for a specific class, and someone else produces
- > an OWL-based AP saying that the cardinality is 2 for the same class, the
- > result is a *contradiction*.

Well, I have what I think is a quite traditional SW background, and I'd be tempted to turn the argument the other way round ;-)
If the instances of one class in an AP have one title and the instances of that class in another AP have two titles, then it is perhaps that these two APs are thinking of two different classes, really. Possibly they could be two subclasses of a common superclass, but two different classes, still. I think this is quite in line with Jeff's message suggesting you could do things like this `baz:Widget rdfs:subClassOf foo:Widget`. to make the commitment of your various APs a bit clearer.

I'm also a bit puzzled by your "APs define domain-specific structural constraints while OWL adds semantics to existing classes." Again, I am pretty new to the APs as practiced in the DC realm, but why wouldn't you consider classes (and APs built on top of them) from a (SW) semantic perspective? I understand that OWL and RDF will fail for arrangement/presentation of data (order of XML elements, e.g.), which is not really about semantics. But to me--and to many in the SW community--cardinality belongs to semantics of classes and properties.

- > This differs substantially from the case with application profiles,
- > where the cardinality is not seen as part of the semantics of a class,
- > but rather part of a set of restrictions, external to and independent of
- > the class. Multiple incompatible application profiles are perfectly
- > normal.
- >
- > Therefore, publishing an OWL ontology defining domain-specific semantics
- > for certain classes or properties is just as bad practice as if someone
- > produces an RDF Schema saying the range of dct:creator is
- > myorg:Employee. This defines new semantics of dct:creator, something
- > that is simply not true, and can cause involuntary contradictions.
- >
- > The other issue is the open world assumption that I saw Pete mention,
- > i.e. the fact that if an OWL ontology specifies a cardinality of 2 for
- > dc:title, and only one is found, this results in the generation of a new
- > dc:title statement, not in non-validity of the record.
- >
- > Thus, we would need an alternative semantics for OWL to perform
- > validation.

Point taken. But even then I can still play the devil's advocate. What in a pure XML world I decide to create something like

```
<dc:title>a title<dc:title>
<dc:title><dc:title>
?
```

By saying that the cardinality for dc:title is 2, it seems that you overlook some info on your model (or AP). That these titles should not be empty or unknown, for example. And taking it properly into account may require some more subtle axioms/specs, both in traditional XML and with Semantic Web languages. Note that it could be that OWL(2) is not enough for all your constraints. But there could be some complementary checking mechanisms (using SPARQL queries is an obvious candidate, but there could be others) which would help you perform validation without changing the semantics.

- > But that's exactly it - the semantics is "alternative" and on its face,
- > the semantics of the published OWL file is something else entirely.
- >
- > As a concrete example, if I serve an OWL file from my web server, using
- > application/rdf+xml as suggested by the OWL specs [1], the
- > interpretation will be as RDF triples using the RDF and OWL built-in
- > semantics, thus resulting in the generation of new triples, potentially
- > contradiction with other ontologies, and not in validation as expected.

Well, contradiction (aka inconsistency) is what is used for data validation in the RDF/OWL world. So if we can detect contradictions I wouldn't be unhappy, at least from the perspective of us able to validate some data :-)

- > What *would* work is using application profile specific classes for each
- > separate OWL-based AP, and only constraining them, but that might appear
- > a bit cludgy.

Or forcing you to make explicit the hidden assumptions of your profile...

It's really a situation where the cons for an OWL approach (and being awkward to manipulate is not the least default OWL has, sure) could be balanced by some strong pros, and we should not discard them too quickly!

From: Mikael Nilsson <mikael@nilsson.name>
To: Antoine Isaac <aisaac@few.vu.nl>
Cc: public-lld <public-lld@w3.org>
Date: Tue, 12 Oct 2010 13:50:19 +0200
Message-ID: <1286884219.4536.113.camel@daneel>
Subject: Re: Returning to OWL and application profiles

Hi Antoine,

tis 2010-10-12 klockan 13:11 +0200 skrev Antoine Isaac:

- > Hi Mikael,
- >
- > Thanks for starting this interesting thread :-)

Thanks for an interesting reply :)

- > I think this is quite in line with Jeff's message suggesting you could do things like this
- > baz:Widget rdfs:subClassOf foo:Widget.
- > to make the commitment of your various APs a bit clearer.

Yes, I do believe that conceptually, most APs can be described using the trick of the creation of subclasses, even though the subclass may well consist of the same individuals (!)

- > I understand that OWL and RDF will fail for
- > arrangement/presentation of data (order of XML elements,
- > e.g.), which is not really about semantics. But to me--and
- > to many in the SW community--cardinality belongs to semantics
- > of classes and properties.

Let me give a simple example. Let's assume I am designing a simple REST service for retrieving metadata records from a library repository (never mind the 303s etc).

I decide to return Turtle representations of the form

```
myrepo:book123 rdf:type lib:Book,  
                dct:title "Moby Dick",  
                dct:creator myrepo:author345 .
```

I can define this pattern using OWL restrictions on the lib:Book class.

Later on, I want to define an extended API for use by library partners.

This API returns records of the form

```
myrepo:book123 rdf:type lib:Book,  
    dct:title "Moby Dick",  
    dct:creator myrepo:author345,  
    lib:numCopies "5".
```

This record describes the exact same individual but using a different application profile.

To solve this with the subclass method I would need to define a subclass `lib:ExtendedBook` that captures the additional property, but has the same extension.

My point is that application profiles describe metadata *patterns*, independently of the semantics. None of the above properties are likely required (cardinality > 0) for the `Book` class in any case. And there may well be multiple such patterns for the same class or set of things.

In the DC world, the use of application profiles is often framed in this way - an application profile does not describe a class, it describes the metadata records emitted or accepted by a system (hence the term "application").

>> But that's exactly it - the semantics is "alternative" and on its face,
>> the semantics of the published OWL file is something else entirely.
>>
>> As a concrete example, if I serve an OWL file from my web server, using
>> `application/rdf+xml` as suggested by the OWL specs [1], the
>> interpretation will be as RDF triples using the RDF and OWL built-in
>> semantics, thus resulting in the generation of new triples, potentially
>> contradiction with other ontologies, and not in validation as expected.
>
>
> Well, contradiction (aka inconsistency) is what is used for data
> validation in the RDF/OWL world. So if we can detect contradictions I
> wouldn't be unhappy, at least from the perspective of us able to
> validate some data :-)

My point is that the two ontologies needed to describe the two uses of `lib:Book` above would be contradictory. One would be left wondering, "what is the semantics of the `lib:Book` class, is it as defined by ontology A or B?" and that's not good. The truth is that the semantics of `lib:Book` is less constrained (all the above properties optional), and the constraints appear only to describe certain *records*, not the `lib:Book` class itself.

> It's really a situation where the cons for an OWL approach (and being
> awkward to manipulate is not the least default OWL has, sure) could be
> balanced by some strong pros, and we should not discard them too
> quickly!

I agree that OWL has advantages, but we need to develop an understanding of what kinds of validation we can achieve.

Maybe we arrive at the conclusion that we need to accept a different set of use cases for OWL-based ontologies. I'm not very happy with that thought, however.

I therefore lean towards a solution based on true syntactical constraint

language on RDF graphs. Alistair Miles did some experiments in that direction, but they seem to have disappeared from the net.

It would be extremely interesting to try to figure out whether there are significant use cases for purely syntactic constraints in the LD domain.

Date: Tue, 12 Oct 2010 09:38:35 -0400
From: "Jon Phipps" <jonp@jesandco.org>
To: "Mikael Nilsson" <mikael@nilsson.name>,
"Antoine Isaac" <aisaac@few.vu.nl>
Cc: "public-ldd" <public-ldd@w3.org>
Subject: RE: Returning to OWL and application profiles

Hi Mikael, Antoine,

Mikael, the use case for syntactic constraints that occurs to me has more to do with treating an RDF graph as an expression of the metadata rather than the source.

It seems to me that our usual notions of data validation are turned on their head, given the inherent Open World assumption of the RDF data model and the fact that the model is more concerned with 'consistency' than validity. An AP is by definition domain-specific and is intended to communicate a domain-local understanding of the classes of resources and their properties. In that sense there would seem to be two useful metadata patterns that comprise an AP -- a set of domain-local validation patterns that looks at each statement with the XML-ish notion of pass/fail within the domain only, and a global-commitment consistency pattern that seeks to ensure that the data is both internally and globally consistent, i.e. does it 'make sense' out in the RDF-ish global web of data.

For instance it's perfectly reasonable for me to define dc:title as a property of mydomain:person and constrain its values to 'Mr., Ms., Dr.'. This clearly happens all the time, and I can document this effectively and produce a very nice schema that will invalidate a value of 'Miss'. But that same data ceases to 'make sense', given the defined semantics of dc:title, when it becomes part of the open world. The same is true of other structural and semantic constraints.

It's also useful to remember that an AP, as defined in the Singapore Framework, is primarily a set of documents that formally communicate domain-specific knowledge. The Description Set Profile is where that knowledge begins to be expressed as model-specific patterns. Personally I would very much like DC to investigate (as Alistair did) how the abstract patterns represented by the proposed DSP constraint language might translate into OWL2 ontologies for communicating domain semantics and XML Schema for domain-specific validation.

From: Kendall Clark <kendall@clarkparsia.com>
Date: Tue, 12 Oct 2010 09:47:12 -0400
Message-ID: <AANLkTi=F4YemtbDXBexYAE9VHR6Pset+n6TcNEGkux0@mail.gmail.com>
To: Jon Phipps <jonp@jesandco.org>
Cc: Mikael Nilsson <mikael@nilsson.name>, Antoine Isaac <aisaac@few.vu.nl>,

public-ldd <public-ldd@w3.org>

Subject: Re: Returning to OWL and application profiles

On Tue, Oct 12, 2010 at 9:38 AM, Jon Phipps <jonp@jesandco.org> wrote:

> It seems to me that our usual notions of data validation are
> turned on their head, given the inherent Open World assumption
> of the RDF data model and the fact that the model is more
> concerned with 'consistency' than validity

We've done quite a bit of work to give OWL2 a closed world semantic such that ontologies can be use to validate and process integrity constraints in RDF and Linked Data.

See these resources for more info:

<http://clarkparsia.com/pellet/icv/>

<http://weblog.clarkparsia.com/2010/04/14/pellet-icv-04-release-using-owl-integrity-constraints-to-validate-skos/>

<http://weblog.clarkparsia.com/2009/02/11/integrity-constraints-for-owl/>

We will submit the specification for this work to W3C as a Member Submission soon.

Cheers,
Kendall Clark

Date: Tue, 12 Oct 2010 16:08:22 +0200

From: Mark van Assem <mark@cs.vu.nl>

To: Mikael Nilsson <mikael@nilsson.name>

CC: Antoine Isaac <aisaac@few.vu.nl>, public-ldd <public-ldd@w3.org>

Subject: Re: Returning to OWL and application profiles

Hi Mikael,

I just defended my PhD thesis [1] last week and it contains a section 7.5 devoted to APs specified in OWL (also referring to your AP constraint language). I suggest you have a read :-)

What I propose is to create subclasses of a particular class such as lib:Book, e.g. my:Book and constrain property values on my:Book so that no inconsistency arises with his:Book (which may have entirely different constraints). We call this "collection-specific value ranges" (with collections referring to cultural heritage collections), but cardinality constraints is basically the same story.

The only thing that then has to be added is a way to classify every book as either my:Book or something else (the "something elses" representing wrongly specified books, i.e. violations of the AP constraints). This is probably abusing OWL a bit, but it can probably be done.

An alternative is to just accept OWL as a "syntax" for the AP constraints, and implement your own checker on top of that. This removes the need to develop your own language (and parser) which will contain almost the same syntactical elements anyway.

(As far as I can tell this is also what DC-APs are intended to do. I had discussions with Tom Baker about this, and he was part of the committee that accepted my thesis last week so apparently I didn't write rubbish :-)

Best,

Mark

[1]<http://www.cs.vu.nl/~mark/papers/thesis-mfjvanassem.pdf>

Date: Tue, 12 Oct 2010 17:09:12 +0200

Message-ID: <AANLkTik4G4gjRKBj07NTAwSGpFDACg9o+WHeciKjmU-v@mail.gmail.com>

From: Emmanuelle Bermes <manue.fig@gmail.com>

To: Mark van Assem <mark@cs.vu.nl>

Cc: Mikael Nilsson <mikael@nilsson.name>, Antoine Isaac <aisaac@few.vu.nl>, public-ldd <public-ldd@w3.org>

Subject: Re: Returning to OWL and application profiles

On Tue, Oct 12, 2010 at 4:08 PM, Mark van Assem <mark@cs.vu.nl> wrote:

> Hi Mikael,

>

> I just defended my PhD thesis [1] last week and it contains a section 7.5

> devoted to APs specified in OWL (also referring to your AP constraint

> language). I suggest you have a read :-)

>

> What I propose is to create subclasses of a particular class such as

> lib:Book, e.g. my:Book and constrain property values on my:Book so that n

> o

> inconsistency arises with his:Book (which may have entirely different

> constraints).

This approach seems very good from the modelling point of view, but I'd like to ask whether it is realistic in a pragmatic Linked Data world.

If I want to add specific constraints on, for instance, dct:creator, and I create my:creator, I will reach interoperability with others using dct:creator only through inferencing. This doesn't seem very straightforward to me. We are lacking actual use of Linked Data today, and I feel that adding more complexity in the data model is likely to create more barriers.

As librarians sitting on a big mass of data, our immediate need is to be able to make our data understandable in the global Linked Data world, and I'm not sure that encouraging us to systematically create our own classes and properties rather than reusing existing ones, for the sake of expressing patterns, is the right way to go. I see a risk to encourage the creation of a lot of redundant, slightly different but almost similar vocabularies (we already have 3 flavours of FRBR out there...)

What I like in the application profile approach is the idea that I can reuse *existing* classes & properties, and at the same time, express the pattern that my community should reuse for describing similar resources.

It's actually 2 separate needs :

- a need for visibility and interoperability, that can be fulfilled by existing vocabularies with their OWL semantics that are consistent globally
- a need for describing domain-specific patterns.

We call this "collection-specific value ranges" (with

> collections referring to cultural heritage collections), but cardinality

> constraints is basically the same story.

>

> The only thing that then has to be added is a way to classify every book as

> either my:Book or something else (the "something elses" representing wrongly
> specified books, i.e. violations of the AP constraints). This is probably
> abusing OWL a bit, but it can probably be done.
>
> An alternative is to just accept OWL as a "syntax" for the AP constraints,
> and implement your own checker on top of that. This removes the need to
> develop your own language (and parser) which will contain almost the same
> syntactical elements anyway.

I like the idea that the AP should be something that could be implemented following different syntaxes, maybe including OWL, but not excluding other approaches that wouldn't make it mandatory to declare local properties and classes systematically when additional semantics or constraints are needed.

Emmanuelle

Message-ID: <20101012101248.esjmtiou84cwok40@kcoyle.net>
Date: Tue, 12 Oct 2010 10:12:48 -0700
From: Karen Coyle <kcoyle@kcoyle.net>
To: Emmanuelle Bermes <manue.fig@gmail.com>
Cc: Mark van Assem <mark@cs.vu.nl>, Mikael Nilsson <mikael@nilsson.name>, Antoine Isaac <aisaac@few.vu.nl>, public-lld <public-lld@w3.org>
Subject: Re: Returning to OWL and application profiles

I agree with Emmanuelle's points below. Creating subclasses for every constraint sounds to me like it will fracture our information world. While machines may go merrily along creating inferences from classes and sub-classes, I think this greatly increases the complexity for humans who must design metadata and who will be creating metadata instances.

I also really appreciate Michael Panzer's explanation of the capabilities of OWL, and wonder if we shouldn't be looking for the "sweet spot" between modeling for broad information sharing and creating applications that make use of the semantic web in very specific ways. To use a library example, library applications may limit the number of authors to 3 per record, but that doesn't change the meaning of each individual author or his/her relationship to the work being defined when that data mingles with other data on the web, nor does it mean that someone re-using the data couldn't describe additional authors. This is a 1+ for Mikael's remark that the "A" in AP is "application" -- and there are some constraints that may be best left to applications.

kc

Message-ID: <4CB4A906.4080902@few.vu.nl>
Date: Tue, 12 Oct 2010 20:29:26 +0200
From: Antoine Isaac <aisaac@few.vu.nl>
MIME-Version: 1.0
To: Karen Coyle <kcoyle@kcoyle.net>
CC: Emmanuelle Bermes <manue.fig@gmail.com>, Mark van Assem <mark@cs.vu.nl>, Mikael Nilsson <mikael@nilsson.name>, public-lld <public-lld@w3.org>
Subject: Re: Returning to OWL and application profiles

Hello Karen,

- > To
- > use a library example, library applications may limit the number of
- > authors to 3 per record, but that doesn't change the meaning of each
- > individual author or his/her relationship to the work being defined when
- > that data mingles with other data on the web, nor does it mean that
- > someone re-using the data couldn't describe additional authors. This is
- > a 1+ for Mikael's remark that the "A" in AP is "application" -- and
- > there are some constraints that may be best left to applications.

Interesting example. But in fact if we went for OWL modelling in this AP sequence, there would be nothing that prevents the re-use of, say, dc:creator everywhere:

- 1. domain vocabulary coins dc:creator in its ontology
- 2. ontology for AP1 defines a new class of books with a cardinality restriction (3) for its usage of dc:creator
- 3. other applications can freely re-use the dc:creator statements of application 1. If they don't re-use the class defined in the ontology for AP1, then there is nothing against having dc:creator applied more than 3 times for their resources.

Note that in relation to your last sentence I've used "ontology" for both the domain vocabulary and the one for AP1. The SW stack assigns no intrinsic commitment wrt. a given application level for ontologies: ontologies are meant to define constraints or data production rules, irrespective of whether these will be used for an entire domain or a specific application.

 Message-ID: <20101012150217.e6bczz0r9cggckw0@kcoyle.net>
 Date: Tue, 12 Oct 2010 15:02:17 -0700
 From: Karen Coyle <kcoyle@kcoyle.net>
 To: Antoine Isaac <aisaac@few.vu.nl>
 Cc: public-ldd <public-ldd@w3.org>
 Subject: Re: Returning to OWL and application profiles

Quoting Antoine Isaac <aisaac@few.vu.nl>:

- >
- > Interesting example. But in fact if we went for OWL modelling in this
- > AP sequence, there would be nothing that prevents the re-use of, say,
- > dc:creator everywhere:
- > - 1. domain vocabulary coins dc:creator in its ontology
- > - 2. ontology for AP1 defines a new class of books with a cardinality
- > restriction (3) for its usage of dc:creator
- > - 3. other applications can freely re-use the dc:creator statements of
- > application 1. If they don't re-use the class defined in the ontology
- > for AP1, then there is nothing against having dc:creator applied more
- > than 3 times for their resources.

Yes, I agree with this. I do still worry that having every combination of constraints be a new class will become un-usable. So a usage restriction of 3 plus a value vocabulary v. usage restriction of 3 and no stated vocabulary; a restriction of 2 plus that same value vocabulary; a restriction of 5 and vocabulary or no vocabulary -- it

gets out of hand pretty easily. (Note: in one system I recall having to limit author names to 99 because there were records that exceeded that!). It's not a matter of whether it CAN be done, but whether doing it this way becomes a hindrance to metadata creation.

Message-ID: <4CB570A1.4070301@cs.vu.nl>

Date: Wed, 13 Oct 2010 10:41:05 +0200

From: Mark van Assem <mark@cs.vu.nl>

MIME-Version: 1.0

To: Karen Coyle <kcoyle@kcoyle.net>

CC: Antoine Isaac <aisaac@few.vu.nl>, public-ld <public-ld@w3.org>

Subject: Re: Returning to OWL and application profiles

> Yes, I agree with this. I do still worry that having every combination
> of constraints be a new class will become un-usable. So a usage
> restriction of 3 plus a value vocabulary v. usage restriction of 3 and
> no stated vocabulary; a restriction of 2 plus that same value
> vocabulary; a restriction of 5 and vocabulary or no vocabulary -- it
> gets out of hand pretty easily. (Note: in one system I recall having
> to limit author names to 99 because there were records that exceeded
> that!). It's not a matter of whether it CAN be done,

I don't get your point. The new class is just a placeholder, stating what the constraints on its superclass should be *within this particular application*. Such a placeholder is needed in any situation where you want to reuse existing classes and properties, but constrain them "locally". It's an application-specific version of the generic class, only to be used within that application.

Your example concerning having to limit author names is I think not related to the issue. If your database can have that many names, and that is allowed, then that's simply the way it is. That's a feature of your dataset that you apparently condone, so then there's no point in putting a restriction anyway.

Date: Tue, 12 Oct 2010 13:24:12 -0400

From: Thomas Baker <tbaker@tbaker.de>

To: Emmanuelle Bermes <manue.fig@gmail.com>

Cc: Mark van Assem <mark@cs.vu.nl>, Mikael Nilsson <mikael@nilsson.name>,

Antoine Isaac <aisaac@few.vu.nl>, public-ld <public-ld@w3.org>

Message-ID: <20101012172412.GA3444@octavius>

Subject: Re: Returning to OWL and application profiles

On Tue, Oct 12, 2010 at 05:09:12PM +0200, Emmanuelle Bermes wrote:

> We are lacking actual use of Linked Data today,
> and I feel that adding more complexity in the data model is likely to
> create more barriers.

There's the point. Is it really necessary to create more classes, more properties, and more triples in the data that is actually published?

> I like the idea that the AP should be something that could be
> implemented following different syntaxes, maybe including OWL, but not
> excluding other approaches that wouldn't make it mandatory to declare
> local properties and classes systematically when additional semantics
> or constraints are needed.

I also wonder about the training angle. I think we want to enable people to create application profiles pretty easily -- perhaps with menu-driven interfaces, and using their syntax of choice -- yet in a way that ensures that their data will play well as Linked Data.

Asking people to conceptualize their data in terms of class semantics seems to set the barrier really high. Explaining the simpler notion of a Dublin Core application profile has been challenging enough. Should the design of good metadata require knowledge of OWL? Can OWL semantics really be stuffed under the hood of a clever interface?

Date: Tue, 12 Oct 2010 20:13:03 +0200
From: Antoine Isaac <aisaac@few.vu.nl>
To: Thomas Baker <tbaker@tbaker.de>
CC: Emmanuelle Bermes <manue.fig@gmail.com>,
Mark van Assem <mark@cs.vu.nl>,
Mikael Nilsson <mikael@nilsson.name>, public-ldd <public-ldd@w3.org>
Subject: Re: Returning to OWL and application profiles

> On Tue, Oct 12, 2010 at 05:09:12PM +0200, Emmanuelle Bermes wrote:
>> We are lacking actual use of Linked Data today,
>> and I feel that adding more complexity in the data model is likely to
>> create more barriers.
>
> There's the point. Is it really necessary to create more
> classes, more properties, and more triples in the data that is
> actually published?

Yes. Please believe that I'd feel very concerned about proliferation of classes or properties, too. If that would be the price to pay then indeed this is not a very seducing solution. But I don't think anyone is suggesting that all properties should be duplicated in APs. As a matter of fact I believe that many examples in this thread could be dealt with by just introducing one AP-specific class, and putting axioms for it that just re-use the existing properties. Which is of uttermost importance: on the linked data environment being able to re-use the link types is much more important than re-using the classes (which is an aspect which I believe also emerged in our previous discussion on FRBR).

And of course again there is the possibility of trying to implement constraints via SPARQL. As long as we can relate to an established bit of the SW architecture that would be already quite positive, I think.

> Asking people to conceptualize their data in terms of class
> semantics seems to set the barrier really high. Explaining
> the simpler notion of a Dublin Core application profile has
> been challenging enough. Should the design of good metadata
> require knowledge of OWL? Can OWL semantics really be stuffed
> under the hood of a clever interface?

That would be the hope of course: I myself have sometimes trouble with OWL (well, I mean with the second version). In fact I wouldn't mind keeping the current formulation of the AP stuff. But it would be a big plus if we could map under

the hood this formulation to the SW tech stack, allowing interested implementors to benefit from the standardization status this stack has now gained.

Date: Tue, 12 Oct 2010 12:27:02 -0400
From: "Panzer,Michael" <panzerm@oclc.org>
To: "Mark van Assem" <mark@cs.vu.nl>,
"Mikael Nilsson" <mikael@nilsson.name>
Cc: "Antoine Isaac" <aisaac@few.vu.nl>,
"public-ldd" <public-ldd@w3.org>,
"Young,Jeff (OR)" <jyoung@oclc.org>,
"Emmanuelle Bermes" <manue.fig@gmail.com>
Subject: RE: Returning to OWL and application profiles

Hi Mark and Mikael,

this might be slightly off-topic, and I hope I am not beating a dead horse here, but I have major problems with talking about OWL as if it where a language allowing schema-like validation of data. It is not. I am not saying that this type of use cannot be achieved with OWL, but it is not trivial.

Statements in OWL are interpreted as facts that are used to infer new pieces of knowledge. Axioms in OWL support that goal as they are treated as "inference rules," not necessarily as constraints aimed at triggering inconsistencies. A trivial example would be the following.

My OWL AP says:

"Every published book has at least one author and exactly one publisher."

My data says:

1. "<MyBook> is a published book."
2. "<MyBook> was published by <Publisher1>."
3. "<MyBook> was published by <Publisher2>."

A possible reaction of a schema validation language would be:

"This data is not compliant to the AP. You state that <MyBook> is a published book, yet there is no author information, and it has two publishers! Your data is rejected because it is either bad or you lied about it."

This would be the reaction of an OWL reasoner:

"Thank you so much for this new information. I can now conclude that <Publisher1> and <Publisher2> refer to the same individual (no Unique Name Assumption, combined with a functional property) and that <MyBook> is connected to some individuals that are authors, but I don't know about them yet (Open World Assumption). I'll add the following facts to my knowledge base:

4. <Publisher1> is the same as <Publisher2>.
5. <MyBook> has some authors.

Please visit again soon."

> An alternative is to just accept OWL as a "syntax" for the AP
> constraints, and implement your own checker on top of that. This
removes
> the need to develop your own language (and parser) which will contain
> almost the same syntactical elements anyway.

Again, I don't say it can't be done. As suggested by Mark, one viable approach would be to interpret OWL axioms with integrity constraint semantics. You wouldn't have to implement your own checker, however. Pellet ICV [1] sounds really promising in that regard, as it generates SPARQL ASK queries to validate your RDF against OWL axioms.

SPIN [2,3] sounds like something similar to be explored as well.

>From a slightly different perspective, I think RIF [4] is what could fill the gap in a more conformant way. RIF has a defined path to OWL and is an integral part of the Semantic Web stack, but I haven't figured out how to use it efficiently yet (I have had little exposure to rules languages in the past, so I need more time to get up to speed there.)

Cheers
Michael

[1] <http://clarkparsia.com/pellet/icv/>
[2] http://www.proxmi.be/users/paul/weblog/aaad2/Integrity_constraints_in_SKOS_part_1_.html
[3] <http://www.spinrdf.org/>
[4] www.w3.org/2005/rules/

Date: Tue, 12 Oct 2010 19:59:24 +0200
From: Antoine Isaac <aisaac@few.vu.nl>
MIME-Version: 1.0
To: "Panzer,Michael" <panzerm@oclc.org>
CC: Mark van Assem <mark@cs.vu.nl>, Mikael Nilsson <mikael@nilsson.name>, public-ld <public-ld@w3.org>, "Young,Jeff (OR)" <jyoung@oclc.org>, Emmanuelle Bermes <manue.fig@gmail.com>
Subject: Re: Returning to OWL and application profiles

> this might be slightly off-topic, and I hope I am not beating a dead
> horse here, but I have major problems with talking about OWL as if it
> were a language allowing schema-like validation of data. It is not. I
> am not saying that this type of use cannot be achieved with OWL, but it
> is not trivial.
>
> Statements in OWL are interpreted as facts that are used to infer new
> pieces of knowledge. Axioms in OWL support that goal as they are treated
> as "inference rules," not necessarily as constraints aimed at triggering
> inconsistencies. A trivial example would be the following.

Well, your example is right, but this is just an example, and the first sentence of this paragraph is wrong: I can also exhibit OWL examples that are not meant at inferring new RDF facts but at checking the consistency of existing facts. The axioms about maximum cardinality, disjointness of classes, properties and individual are about that [1,2].

Granted, these axioms may fail at capturing all constraints you want to have in APs, and you may need some more close-world stuff like what Kendall is proposing. But that's another

issue :-)

[1] http://www.w3.org/TR/2009/REC-owl2-primer-20091027/#Property_Cardinality_Restrictions

[2] http://www.w3.org/TR/2009/REC-owl2-primer-20091027/#Class_Disjointness

Date: Tue, 12 Oct 2010 17:23:30 -0400
From: "Panzer,Michael" <panzerm@oclc.org>
To: "Antoine Isaac" <aisaac@few.vu.nl>
Cc: "Mark van Assem" <mark@cs.vu.nl>,
"Mikael Nilsson" <mikael@nilsson.name>,
"public-ldd" <public-ldd@w3.org>,
"Young,Jeff (OR)" <jyoung@oclc.org>,
"Emmanuelle Bermes" <manue.fig@gmail.com>
Subject: RE: Returning to OWL and application profiles

> > Statements in OWL are interpreted as facts that are used to infer new
> > pieces of knowledge. [...]
>
> Well, your example is right, but this is just an example, and the first
> sentence of this paragraph is wrong [...]

Please don't use a closed-world assumption to validate my statement! ;-)
Only because I tried to show (cum grano salis) one function of
statements in OWL doesn't mean that there aren't many others.

> I can also exhibit OWL examples that
> are not meant at inferring new RDF facts but at checking the consistency
> of existing facts. The axioms about maximum cardinality, disjointness of
> classes, properties and individual are about that [1,2].

I would still say (somewhat stubbornly) that the crux is not so much in
checking the consistency *of* existing facts, but checking the
consistency of new facts *with* existing facts, thus still about
inferring further information. (Unless either one of the original sets
of statements was already inconsistent, of course.)

You are of course right about the types of assertions you mention.
Looking back at my previous example, if the ontology had contained
information like:

N1: "<Publisher1> is a European publisher."
N2: "<Publisher2> is an Australian publisher."
N3: "No publisher can be a European and an Australian publisher at the
same time." (Disjointness)

Then the inference by the reasoner would have been: "Sorry. The
information you have provided is inconsistent with what I already know.
I would have needed to assume that <Publisher1> is <Publisher2> and that
this publisher is both European and Australian. I cannot do that."

But, and that was my point, (1) this requires the (careful) addition and
control of new assertions "higher up" in the ontology, and (2) the
outcome "inconsistent" here carries information. It doesn't necessarily
mean only "clean up your data."

The OWL 2 primer alludes to this can of worms very eloquently:

"In particular, there is no way to enforce that a certain piece of
information (like the social security number of a person) has to be

syntactically present. This should be kept in mind as OWL has some features that a user might misinterpret this way." [1]

That doesn't mean it can't be done, I think it means it can't be done superficially or "syntactically."

> Granted, these axioms may fail at capturing all constraints you want to
> have in APs, and you may need some more close-world stuff like what
> Kendall is proposing. But that's another issue :-)

Agreed. :-) The issue "OWL Full" vs. "OWL DL" comes to mind. For example, some integrity conditions of SKOS (S27: "skos:related is disjoint with the property skos:broaderTransitive." [2]) cannot be expressed in OWL 2 DL, as disjointness on transitive properties is not allowed (to avoid undecidability.)

Stuff like this is clearly needed for APs in some form, and, again, I still think most of it can be done with tools from the SW stack (OWL + assorted others).

Cheers
Michael

[1] http://www.w3.org/TR/owl2-primer/#What_is_OWL_2.3F
[2] <http://www.w3.org/TR/skos-reference/#L2422>

Date: Wed, 13 Oct 2010 10:54:54 +0200
From: Mark van Assem <mark@cs.vu.nl>
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; nl; rv:1.9.2.9) Gecko/20100915 Thunderbird/3.1.4
MIME-Version: 1.0
To: "Panzer,Michael" <panzerm@oclc.org>
CC: Mikael Nilsson <mikael@nilsson.name>,
Antoine Isaac <aisaac@few.vu.nl>,
public-ld <public-ld@w3.org>, "Young,Jeff (OR)" <jyoung@oclc.org>,
Emmanuelle Bermes <manue.fig@gmail.com>
Subject: Re: Returning to OWL and application profiles

I am very much aware of the open world assumption and the pains that OWL can cause because of it (with Guus Schreiber as PhD supervisor and Frank van Harmelen in the group, how could I not ;)

I know OWL was not meant to be used in this way. I merely wanted to point out that it CAN be used in that way without saying it's ideal.

The alternative, developing a whole new language with its own parsers and checkers, may not be very attractive either.

So the question becomes a trade-off between (ab)using OWL (maybe even with a "closed world switch in the reasoner?) to deliver this functionality, and the effort of creating and maintaining something entirely different for just this goal.

The most single great thing about the SemWeb - one model + shared tools for knowledge representation and querying - goes out the window if you choose the latter.

Date: Tue, 12 Oct 2010 14:27:28 -0400

From: Thomas Baker <tbaker@tbaker.de>
To: Mikael Nilsson <mikael@nilsson.name>
Cc: Antoine Isaac <aisaac@few.vu.nl>, public-ldd <public-ldd@w3.org>
Message-ID: <20101012182728.GA2652@octavius>
Subject: Re: Returning to OWL and application profiles

> I therefore lean towards a solution based on true syntactical constraint
> language on RDF graphs. Alistair Miles did some experiments in that
> direction, but they seem to have disappeared from the net.

There is a copy of the main page at [1]. Unfortunately, the Wayback Machine does not have copies of related pages, such as SODC-0_2/constraints/.

Tom

[1] http://web.archive.org/web/20080214232032/http://isegserv.itd.rl.ac.uk/sodc/SODC-0_2/

--

Tom Baker <tbaker@tbaker.de>

Date: Tue, 12 Oct 2010 14:47:45 -0400
From: Thomas Baker <tbaker@tbaker.de>
To: Alistair Miles <alimanfoo@gmail.com>
Cc: Mikael Nilsson <mikael@nilsson.name>, Antoine Isaac <aisaac@few.vu.nl>, Pete Johnston <Pete.Johnston@eduserv.org.uk>, Emmanuelle Bermes <emmanuelle.bermes@bnf.fr>
Subject: Son of DC revisited...
Message-ID: <20101012184745.GA3668@octavius>

On Tue, Oct 12, 2010 at 02:27:28PM -0400, Mikael wrote:
> > I therefore lean towards a solution based on true syntactical constraint
> > language on RDF graphs. Alistair Miles did some experiments in that
> > direction, but they seem to have disappeared from the net.
>
> There is a copy of the main page at [1]. Unfortunately, the Wayback Machine
> does not have copies of related pages, such as SODC-0_2/constraints/.
>
> Tom
>
> [1] http://web.archive.org/web/20080214232032/http://isegserv.itd.rl.ac.uk/sodc/SODC-0_2/

Hi Alistair,

There is an interesting discussion on the public-ldd list in preparation for a meeting next week at DC-2010 in Pittsburgh on Application Profiles -- and on how most elegantly and constructively to deprecate DCAM in favor of RDF abstract syntax. For example, see the agenda at [2], a thread which started yesterday at [3], and a recent Architecture call at [4].

The question is how best to express constraints, or rather under which circumstances which approach may be most appropriate, with approaches ranging from DC's syntax-oriented style to styles based more on modeling constraints in OWL.

In this context, there is interest in looking back at your Son of DC proposal from 2007. Those pages have disappeared from the Web, though I found a copy of the main page in the

Wayback Machine.

Do you happen to have a copy, or know where one is available?

Hope all is well with you after a year in your new position...!

Tom

[2] <http://www.w3.org/2001/sw/wiki/JointMeeting2010>

[3] <http://lists.w3.org/Archives/Public/public-lld/2010Oct/0035.html>

[4] <https://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=DC-ARCHITECTURE;9ee93529.1010>

From: Mikael Nilsson <mikael@nilsson.name>
To: Antoine Isaac <aisaac@few.vu.nl>
Cc: public-lld <public-lld@w3.org>
Date: Tue, 12 Oct 2010 21:52:28 +0200
Message-ID: <1286913148.4536.137.camel@daneel>
Subject: Re: Returning to OWL and application profiles

Many interesting replies here...

Based on the discussions here, what I'd like to see is a straightforward RDF graph constraint language, operating under one simple assumption:

* The RDF graph is tree-like, with a single root. This seems to cover most of the linked data examples as well as examples from Dublin Core

For example, an AP covering the case:

```
myrepo:im123 rdf:type dct:Image,  
    dct:creator [          # a single creator  
        rdf:type dct:Agent,      # always Agent  
        foaf:name "Ralph Johnson"] .    # single string name  
    dct:language [          # a single language  
        rdf:type dct:LinguisticSystem,  # always same type  
        rdfs:label "en" ] .    # en, fr or de  
    dct:subject <http://example.org/subjects/a> . # optional subject, a b or c.
```

could be described in XML like so:

```
<?xml version='1.0' encoding='UTF-8' ?>
```

```
<ap:profile xmlns:ap='http://dublincore.org/dc-ap/xml'  
  xmlns:dct='http://purl.org/dc/terms/'  
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'  
  xmlns:foaf='http://xmlns.com/foaf/0.1/'>
```

```
<ap:property="rdf:type" ap:uri="http://purl.org/dc/terms/Image" ap:minOccurs="1" ap:maxOccurs="1"/>  
<ap:property="dct:creator" ap:minOccurs="1">  
  <ap:property="rdf:type" ap:uri="http://purl.org/dc/terms/Agent" ap:minOccurs="1" ap:maxOccurs="1"/>  
  <ap:property="foaf:name" ap:nodeType="PlainLiteral" />  
</ap:property>  
<ap:property="dct:language" ap:minOccurs="1" ap:maxOccurs="1">  
  <ap:property="rdf:type" ap:uri="http://purl.org/dc/terms/LinguisticSystem" ap:minOccurs="1" ap:maxOccurs="1"/>  
  <ap:property="rdfs:label" ap:nodeType="PlainLiteral" ap:minOccurs="1" ap:maxOccurs="1">  
    <ap:oneOf>  
      <ap:alt>en</ap:alt>  
      <ap:alt>fr</ap:alt>  
      <ap:alt>de</ap:alt>
```

```

    </ap:oneOf>
  </rdfs:label>
</ap:property>
<ap:property="dct:subject" ap:nodeType="URI" ap:minOccurs="0" ap:maxOccurs="1">
  <ap:oneOf>
    <ap:alt>http://example.org/subjects/a</ap:alt>
    <ap:alt>http://example.org/subjects/b</ap:alt>
    <ap:alt>http://example.org/subjects/c</ap:alt>
  </ap:oneOf>
</ap:property>
</ap:profile>

```

And this in turn could be mapped to XML Schema for a normalized XML serialization, or alternatively implemented using SPARQL or whatever technology preferred.

The above is along the lines of Alistair's proof-of-concept Graph-ML.

Using the above method, the AP is completely decoupled from the semantics, and acts as pure pattern matcher.

One could imagine semantic extensions, for example by only allowing instances of a certain class as values of a property.

```

<ap:property="dct:subject" ap:nodeType="URI" ap:minOccurs="0" ap:maxOccurs="1">
  <ap:instanceOf ap:resource="http://example.org/subjectClass" />
</ap:property>

```

or even limiting values to resources matching a certain pattern,

```

<ap:property="dct:subject" ap:nodeType="URI" ap:minOccurs="0" ap:maxOccurs="1">
  <ap:match>
    <ap:property="skos:inScheme" ap:URI="http://example.org/subjectScheme"/>
  </ap:match>
</ap:property>

```

such a construct would then match external data (from a vocabulary), not triples in the metadata record.

Something like the above would be quite a valuable addition to the RDF set of specs, IMHO.