International Conference on Dublin Core and Metadata Applications

Tutorial 2: Dublin Core – Key Concepts

Pete Johnston, Eduserv Foundation
mailto:pete.johnston@eduserv.org.uk
What I’m going to talk about

• A conceptual model for Dublin Core metadata
  – DCMI Abstract Model (DCAM)

• The relationship of this model to another conceptual model for metadata
  – RDF Model

• (Briefly) How to represent instances of the DCAM conceptual model in concrete syntaxes
  – “Encoding Guidelines”

• N.B. May gloss over some of the detail!
What I’m not going to talk about

• The use of any specific sets of metadata terms
  – DCMI vocabularies or other metadata vocabularies
• Declaring/defining metadata terms
• “Simple Dublin Core” or “Qualified Dublin Core”
• Dublin Core Application Profiles
Why DCAM? Dublin Core in c2003

- Metadata vocabularies
  - … but what is a DC “element”?
- Syntax independence & encoding guidelines
  - … but what are we “encoding”?
- Grammatical Principles (including “1-to-1 rule”)
  - … fairly informal
- “Simple” and “Qualified” DC
  - … vocabularies?
  - … formats? (e.g. oai_dc)
  - … constraints on use of vocabularies? On which vocabularies?
- DC application profiles
  - … “(re)using” terms? But what “terms” can we “(re)use”?
- Relationship between DC & Resource Description Framework
The DCMI Abstract Model (DCAM)
DCMI Abstract Model

- Work by DCMI Architecture WG from mid-2003, initiated by Andy Powell
- Second Version, DCMI Recommendation, 2007-06-04
- Describes
  - Components and constructs that make up an information structure ("DC description set")
  - How that information structure is to be interpreted
- Made up of three related "information models"
  - Resource model
  - Description set model
  - Vocabulary model
DCMI Abstract Model

- DCAM describes DC description set…
- … but does **not** describe how to represent DC description set in concrete form
  - DCMI-defined “Encoding guidelines”
  - Formats defined by others, e.g. Eprints DC-XML

- DCAM specifies use of various **types** of metadata term…
- …but does **not** specify the use of any fixed set of terms
  - DCMI-owned metadata vocabularies
  - Vocabularies owned/defined by other agencies
**DCAM Resource Model**

- The “view of the world” on which DC metadata is based
- Concerned with description of **resources**
  - digital objects, physical objects, imaginary things, concepts…
  - anything of interest can be a resource
- When describing a resource we make assertions about its relationships with other resources
  - DCAM views world in terms of binary relationships
  - “Resource X is-related-in-some-way-to Resource Y”
    “Resource Y is-related-in-some-way-to Resource Z”
- **DCAM** uses the terminology
  - **Described resource**
  - **Property** = type of relationship
  - **Value** = other resource
Literals and “non-literals”

• The world of resources is further divided into
  – Literals (“Strings”)
    • “self-contained”
    • appear directly in DC metadata
    • “terminals” in DC metadata
    • can not be further “described”
  – Other “non-literal” resources (“Things”)
    • referred to in DC metadata
    • can be further described
      – in your metadata or in someone else’s metadata elsewhere
  • “Things” can be described, “strings” can’t
_literals and “non-literals”

- So, the designer of a DC application has to
  - Construct (or adopt) a model of the part of the world of interest
  - Types of resource, types of relationship between resources
- And they have to decide
  - Do I model this as a “Thing” - because
    - I need to describe it?
    - Or allow others to describe it?
    - Or take advantage of fact that others have described it?
  - Do I model this as a “String” (literal) – because
    - it is “self describing”? 
    - I don’t need to describe it?
    - Or allow others to describe it?
    - Or make use of others’ descriptions of it?
- Choice depends on requirements of application
## Relational Model

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>publisher</th>
<th>pub date</th>
</tr>
</thead>
</table>
## Relational Model

- Each row represents a resource being described

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>publisher</th>
<th>pub date</th>
</tr>
</thead>
</table>
### Relational Model

- Each column represents an attribute of those resources

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>publisher</th>
<th>pub date</th>
</tr>
</thead>
</table>
## Relational Model

- Intersection gives value of attribute for resource described

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>publisher</th>
<th>pub date</th>
</tr>
</thead>
</table>
From DCAM perspective

- Each attribute for resource described = Property + Value

<table>
<thead>
<tr>
<th>Described Resource</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (Thing)</td>
<td>has-title</td>
<td>“RDF Concepts &amp; Abstract Syntax” (String)</td>
</tr>
</tbody>
</table>
## Relational Model

- More attributes of document?

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>publisher</th>
<th>pub date</th>
</tr>
</thead>
</table>
From DCAM perspective

- More attributes? Additional Properties + Values

<table>
<thead>
<tr>
<th>Described Resource</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (Thing)</td>
<td>has-title</td>
<td>“RDF Concepts &amp; Abstract Syntax” (String)</td>
</tr>
<tr>
<td></td>
<td>has-author</td>
<td>“Graham Klyne” (String)</td>
</tr>
<tr>
<td></td>
<td>has-publisher</td>
<td>“W3C” (String)</td>
</tr>
</tbody>
</table>
## Relational Model

<table>
<thead>
<tr>
<th>uri</th>
<th>title</th>
<th>author</th>
<th>pub id</th>
<th>pub date</th>
</tr>
</thead>
</table>

### Attributes of document & attributes of agent?

<table>
<thead>
<tr>
<th>pub id</th>
<th>short name</th>
<th>full name</th>
<th>location</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>000001</td>
<td>DCMI</td>
<td>Dublin Core Metadata Initiative</td>
<td>Singapore</td>
<td>…</td>
</tr>
<tr>
<td>000002</td>
<td>W3C</td>
<td>World Wide Web Consortium</td>
<td>Boston, MA, USA</td>
<td>…</td>
</tr>
</tbody>
</table>
### From DCAM perspective

- Two described resources, one is also a value

<table>
<thead>
<tr>
<th>Described Resource</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document (Thing)</td>
<td>has-title</td>
<td>“RDF Concepts &amp; Abstract Syntax” (String)</td>
</tr>
<tr>
<td>Document (Thing)</td>
<td>has-author</td>
<td>Agent (Thing)</td>
</tr>
<tr>
<td>Document (Thing)</td>
<td>has-publisher</td>
<td>“W3C” (String)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Described Resource</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent (Thing)</td>
<td>has-short-name</td>
<td>“W3C”</td>
</tr>
<tr>
<td>Agent (Thing)</td>
<td>has-name</td>
<td>“World Wide Web Consortium”</td>
</tr>
</tbody>
</table>
DCAM Description Set Model

- The structure of “DC metadata”
- Uses URIs to refer to resources described & to metadata terms

- A description set is made up of one or more descriptions, each of which describes one resource
- A description is made up of
  - Zero or one described resource URI
    - Identifies described resource
  - One or more statements
  - Exact one property URI
    - Identifies property
    - Exactly one value surrogate
- A value surrogate is either a literal value surrogate or a non-literal value surrogate

E.g. http://purl.org/dc/terms/subject
DCAM Description Set Model

• a literal value surrogate is made up of
  – exactly one value string
    • encodes value
• a non-literal value surrogate is made up of
  – zero or one value URIs
    • identifies value
  – zero or one vocabulary encoding scheme URI
    • identifies a set of which the value is a member
  – zero or more value strings
    • represents value
• a value string is either a plain value string or a typed value string
  – a plain value string may have an associated value string language
  – a typed value string is associated with a syntax encoding scheme URI
• Vocabulary Encoding Scheme
  – A named set to which a “Thing” belongs
• Syntax Encoding Scheme
  – A named set of rules for the “interpretation” of a set of “Strings”
DCAM Description Set Model

- a *non-literal value* may be described by another *description*

- Remember the document and publisher example
Example: Description of document, description of publisher

Description

Statement

Property URI: <http://purl.org/dc/terms/publisher>
Value URI: <http://example.org/org/DCMI>

Non-Literal Value Surrogate

Value URI: <http://example.org/mySH/h123>
Vocab Enc Scheme URI: <http://example.org/terms/mySH>
Value String: "Metadata" en
Value String: "Métadonnées" fr

Description

Statement

Property URI: <http://purl.org/dc/terms/subject>
Value URI: <http://example.org/mySH/h123>

Value URI: <http://example.org/terms/mySH>

Example: Description of document, description of publisher

Description

Statement

Property URI: <http://xmlns.com/foaf/0.1/name>
Value String: "Dublin Core Metadata Initiative" en

Property URI: <http://purl.org/dc/terms/created>
Value String: "1995" ^xsd:gyear
DCAM Vocabulary Model

1. Specifies the types of term used in DC metadata
   - Properties
   - Classes
   - Vocabulary Encoding Schemes
   - Syntax Encoding Schemes

2. Defines some relationship types that can exist between terms
   - between properties
     • Property P is subProperty Of Property Q
   - between properties and classes
     • Property P has Range Class C
     • Property P has Domain Class D
   - between classes
     • Class C is subclass of Class D

3. Provides rules for drawing logical conclusions based on these relationship types
Summary: key DCAM concepts

• Think in terms of simple relationships between two resources
• Resources & metadata terms identified by URIs
• Literal ("String") v Non-literal ("Thing")
  – “Things” can be described; “strings” can’t
• Use of description set structure
  – Description set
  – One or more Descriptions
  – One or more Statements
• Value surrogate
  – Literal value surrogate = simple structure
  – Non-literal value surrogate = more complex structure
    • Non-literal value as member of Vocabulary Encoding Scheme
    • Non-literal value represented by multiple Value Strings
Summary: key DCAM concepts

- DCAM specifies use of defined *types* of terms
  - Property
  - Class
  - Vocabulary Encoding Scheme
  - Syntax Encoding Scheme
- DCAM doesn’t require use of any specific set of terms
  - a description set might use **no** DCMI-owned metadata terms
- DCAM doesn’t specify use of any particular combinations of terms
The DCMI Abstract Model & the Resource Description Framework (RDF)
Resource Description Framework (RDF)

- Set of specifications from W3C
- A simple conceptual model...
- ...for making assertions about relationships between resources
  - “Thing-X is-related-in-some-way-to Thing-Y”
- Types of relationship are properties
- Assertions made in the form of triples
  - Subject, Predicate, Object
- Sets of triples often represented as node-arc-node patterns (graphs)
Resource Description Framework (RDF)

- RDF designed for use on Web
- Global context, global naming: URIs
  - URIs as names for things described (subject, object)
  - URIs as names of relationships between things (predicate)
- Also allows for
  - “unnamed” things: “blank nodes” (subject, object)
  - “literals” = text strings (object only)
- RDF defines rules
  - for merging sets of triples
    - e.g. for “joining together” your “descriptions” and my “descriptions” of the same set of resources
  - for drawing logical conclusions from some triples

http://ex.org/ThingX → http://ex.org/relatedTo → "literal"

http://ex.org/ThingX → http://ex.org/relatedTo → "literal"

http://ex.org/ThingX → http://ex.org/relatedTo → "literal"

**DCAM & RDF**

- A history of co-evolution
- DCAM grounded in concepts of RDF
  - assertions of binary relationships between resources, use of URIs
  - DCAM Vocabulary Model is RDF Schema
  - (rather informally!) shares RDF Semantics
  - basis for merging, inferencing
- DCAM doesn’t explicitly use “description model” of RDF (triple, graph)
- But description set can be seen as “specialised view” of RDF graph
  - adds some additional constructs
- Mapping provided by “Expressing DC metadata using RDF”, DCMI Recommendation, 2008-01-14
- In using the DCAM, you are also using RDF
- Dublin Core terms can be used in RDF without using DCAM
Encoding Dublin Core metadata
“Encoding” Dublin Core metadata

- DCAM description set model is syntax-independent
- For transfer between applications, description sets must be encoded as digital objects (records)
- “Encoding Guidelines” describe
  - how abstract information structure is **serialised/encoded** using a **metadata format**
  - how instances of a metadata format are **decoded/interpreted** in terms of abstract information structure
- Provider and consumer need shared rules for encoding/decoding
- DCAM description set as “interface”; concrete syntax as implementation

- N.B. This is just a very quick overview of DCAM-based encoding guidelines currently provided by DCMI
System A

Construct using DCAM & DSP

DC
Description Set

Encode using guidelines

XML Instance

<?xml version="1.0"?>
<dcds:descriptionSet>

System B

Interpret using DCAM

DC
Description Set

Decode using guidelines

XML Instance

<?xml version="1.0"?>
<dcds:descriptionSet>
“Encoding” Dublin Core metadata

- Multiple syntaxes available
  - Defined by DCMI
  - Defined by other parties
- Different syntaxes may be appropriate for different contexts
- “Encoding guidelines” specify
  - what subset of DCAM description model supported
  - how each supported feature of DCAM encoded as syntactic constructs
  - how syntactic constructs interpreted as DCAM features
- Terms are always referred to using URIs
  - Some syntaxes provide abbreviation mechanisms
“Encoding” Dublin Core metadata

- Warning!
- Some of current DCMI “Encoding Guidelines” specs
  - Pre-date development of DCAM
  - Use earlier, simpler “DC abstract models”
  - Not compatible with RDF model
  - Not compatible with DCAM description set model
- Updating of specs currently (Sep 2008) in progress
DC-RDF

- RDF itself is conceptual model
  - Multiple concrete syntaxes available for RDF
    - RDF/XML, N3, Turtle, RDFa etc
- “Expressing DC metadata using RDF”, DCMI Recommendation, 2008-01-14
  - http://dublincore.org/documents/2008/01/14/dc-rdf/
  - Uses RDF abstract syntax
  - Supports full DCAM description model
  - Any concrete syntax for RDF can be used for DC metadata
- DC-RDF is stable, complete
System A

Construct using DCAM & DSP

DC
Description Set
Map using DC-RDF

RDF Graph

Encode using RDF Syntax spec

XML Instance

System B

Interpret using DCAM

DC
Description Set
Map using DC-RDF

RDF Graph

Decode using RDF Syntax spec

XML Instance

<?xml version="1.0"?>
<rdf:RDF>
<rdf:Description rdf:about="http://example.org/org/DCMI">
  <foaf:name xml:lang="en">Dublin Core Metadata Initiative</foaf:name>
  <dcterms:created rdf:datatype="http://www.w3.org/2001/XMLSchema#gyear">1995</dcterms:created>
</rdf:Description>
## DC-HTML

- “Expressing DC metadata using HTML/XHTML meta and link elements”, DCMI Recommendation, 2008-08-04
  
  
  - Supports subset of DCAM description model
  
  - DC metadata in HTML document describes that document
    - or at least document of which HTML page is representation
  
  - An HTML meta-data profile
  
  - GRDDL Profile Transformation to generate RDF/XML

- DC-HTML is stable, complete
DC-HTML

- Supports subset of description set model
  - One description
  - For non-literal value surrogate
    - Maximum of one value string
    - Value string must be plain value string
    - Value URI must be provided
    - Vocabulary encoding scheme URI not supported

- Do use the X/HTML profile attribute
- “Namespacing” using link/@rel=“schema.XX”
  - Don’t use “composite name” convention (dcterms.description.abstract)

- Statements w literal value surrogates, use <meta>
- Statements w non-literal value surrogates, use <link>
DC-DS-XML

- “Expressing DC Description Sets using XML (DC-DS-XML)”, Proposed Recommendation, 2008-09-01
  - Supports full DCAM description model
  - “TRiX”-like
    - Uses XML element names and XML attribute names corresponding to the names of the components of the description set
    - Instance data as XML element content and XML attribute values
    - URIs represented in full
      - Verbose, but easily processable
      - W3C XML Schema, RELAX NG Schema
      - GRDDL Namespace Transformation to generate RDF/XML
  - DC-DS-XML is still liable to change
DC metadata in XML

- DCMI Architecture Forum currently gathering requirements for other XML format(s)
  - e.g. use of schema validation to implement structural constraints of Description Set Profile
Summary: key points re encoding guidelines

- DCAM defines a conceptual model
- “Encoding guidelines” specify how to encode instances of that conceptual model
- Current DCMI specifications being updated
- DCAM conceptual model builds on/maps to RDF conceptual model
  - Several syntaxes exist for encoding RDF model
  - So any RDF syntax can also be used
- Some work ongoing!
Summary
Summary

• DCMI Abstract Model provides conceptual model for DC metadata
  – What is “DC metadata”?
  – What is the thing which is being “encoded”?
• “Encoding guidelines” define how to represent in concrete syntaxes
• DCAM model based on the RDF model
  – DC description set as “specialised view” of RDF graph
  – Use of DCAM description set implies use of RDF
  – DCAM Vocabulary Model = RDF Schema
• But N.B. DC metadata terms may be used in RDF graphs **without** use of description set concept
  – Different “levels” of “using Dublin Core”
Acknowledgements

• Thanks to Tom Baker, Mikael Nilsson & Andy Powell for comments and suggestions

• The relational database comparison draws on an example used in a presentation on RDF by Ian Davis (Talis)
International Conference on Dublin Core and Metadata Applications

Tutorial 2: Dublin Core – Key Concepts

Pete Johnston, Eduserv Foundation
mailto:pete.johnston@eduserv.org.uk