Architecture Working Group

> AMDraftUpdate

User
UserPreferences
Site
Page
Actions

- AttachFile
- DeletePage
- LikePages
- LocalSiteMap
- SpellCheck

Search

Title:
Text:

DCMI Abstract Model - DRAFT Update

IMPORTANT: This document and the associated UML diagrams are draft. They are under discussion on the dc-architecture@jiscmail.ac.uk mailing list. Comments on this work should be sent to that list.

1. Introduction

This document specifies an abstract model for DCMI metadata [DCMI]. The primary purpose of this document is to provide a reference model against which particular DC encoding guidelines can be compared. To function well, a reference model needs to be independent of any particular encoding syntax. Such a reference model allows us to gain a better understanding of the kinds of descriptions that we are trying to encode and facilitates the development of better mappings and translations between different syntaxes.

This document is primarily aimed at the developers of software applications that support Dublin Core metadata, people involved in developing new syntax encoding guidelines for Dublin Core metadata and those people developing metadata application profiles based on the Dublin Core.

2. DCMI abstract model

The abstract model of the *resources* being described by DCMI metadata *descriptions* is as follows:

- Each *described resource* may have one or more *property/value pairs*.
- Each *property/value pair* is made up of one *property* and one *value*.
- Each *value* is a *resource* (the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*).
- Each *resource* may be an instance of one or more *classes*.
- Each *resource* may be a member of one or more *vocabulary encoding schemes*.

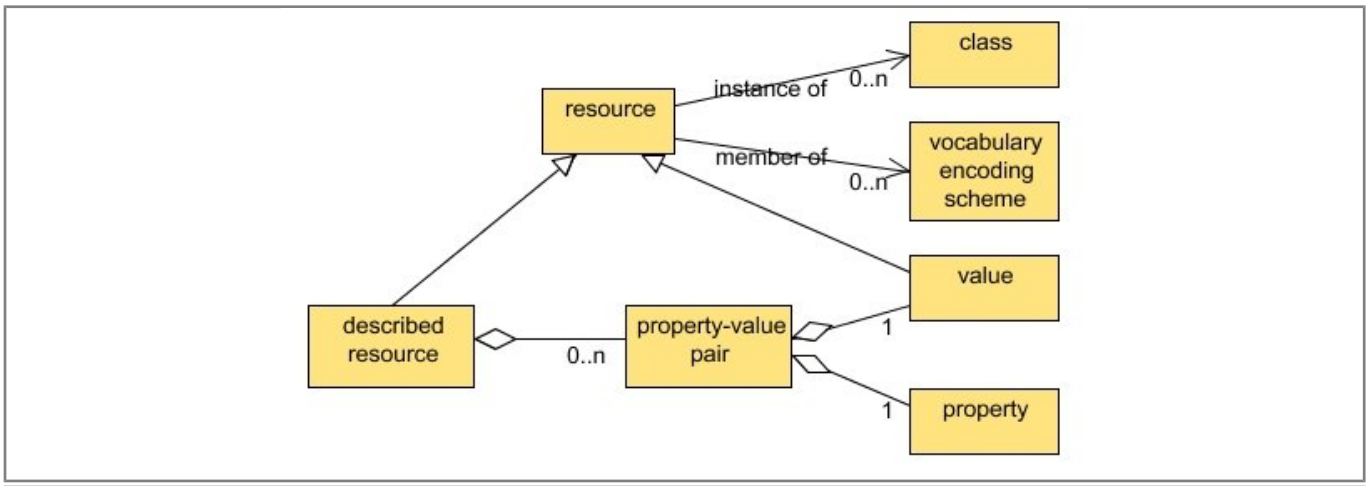

Figure 1 - the DCMI resource model

The abstract model of the *property* and *value* vocabularies used in DCMI metadata *descriptions* is as follows:

- Each *property* may be related to one or more *classes* by a *has domain* relationship. Where it is stated that a *property* has such a relationship with a *class* and a *described resource* is related to a *value* by that *property*, it follows that the *described resource* is an instance of that *class*.
- Each *property* may be related to one or more *classes* by a *has range* relationship. Where it is stated that a *property* has such a relationship with a *class* and a *described resource* is related to a *value* by that *property*, it follows that the *value* is an instance of that *class*.
- Each *value* may be an *instance of* one or more *classes*.
- Each *value* may be a *member of* one or more *vocabulary encoding schemes*.
- Each *property* and *class* has some declared *semantics*.
- Each *class* may be related to one or more other *classes* by a *sub-class of* (refines) relationship (where the two *classes* share some *semantics* such that all *resources* that are instances of the sub-class are also instances of the related *class*).
- Each *property* may be related to one or more other *properties* by a *sub-property of* (refines) relationship. Where it is stated that such a relationship exists, the two *properties* share some *semantics* such that whenever a *resource* is related to a *value* by the sub-*property*, it follows that the *resource* is also related to that same *value* by the *property*.

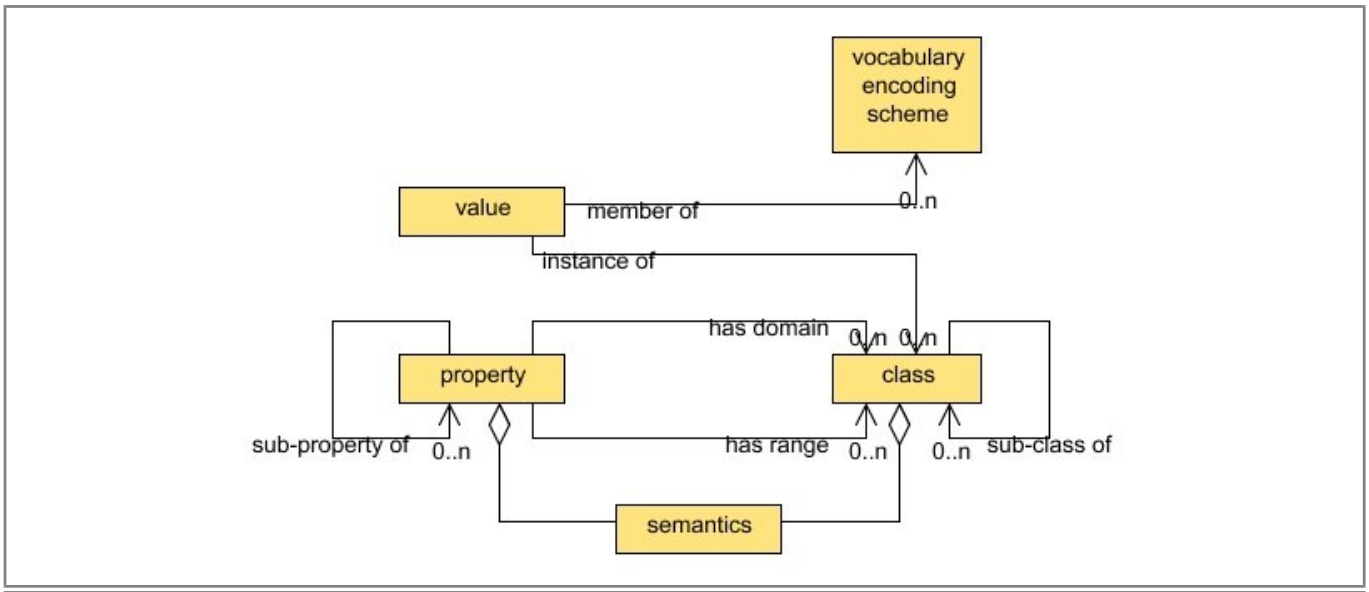

Figure 2 - the DCMI vocabulary model

The abstract model of DCMI metadata *descriptions* is as follows:

- A *description* is made up of one or more *statements* (about one, and only one, *described resource*) and zero or one *resource URI* (a URI that identifies the *described resource*).
- Each *statement* instantiates a *property/value pair* and is made up of a *property URI* (a URI that identifies a *property*), zero or one *value URI* (a URI that identifies the *value* of the *property*), zero or one *vocabulary encoding scheme URI* (a URI that identifies the *vocabulary encoding scheme* of which the *value* is a member), zero or one *value class URI* (a URI that identifies the *class* of the *value*) and zero or more *value representations* of the *value*.
- The *value representation* may take the form of a *value string* or a *rich representation*.

- Each *value string* is a simple, human-readable string that is a representation of the *resource* that is the *value* of the *property*.
- Each *value string* may have an associated *syntax encoding scheme URI* that identifies a *syntax encoding scheme*.
- Each *value string* may have an associated *value string language* that is an ISO language tag (e.g. en-GB).
- Each *rich representation* is some *marked-up text*, an image, a video, some audio, etc. or some combination thereof that is a representation of the *resource* that is the *value* of the *property*.

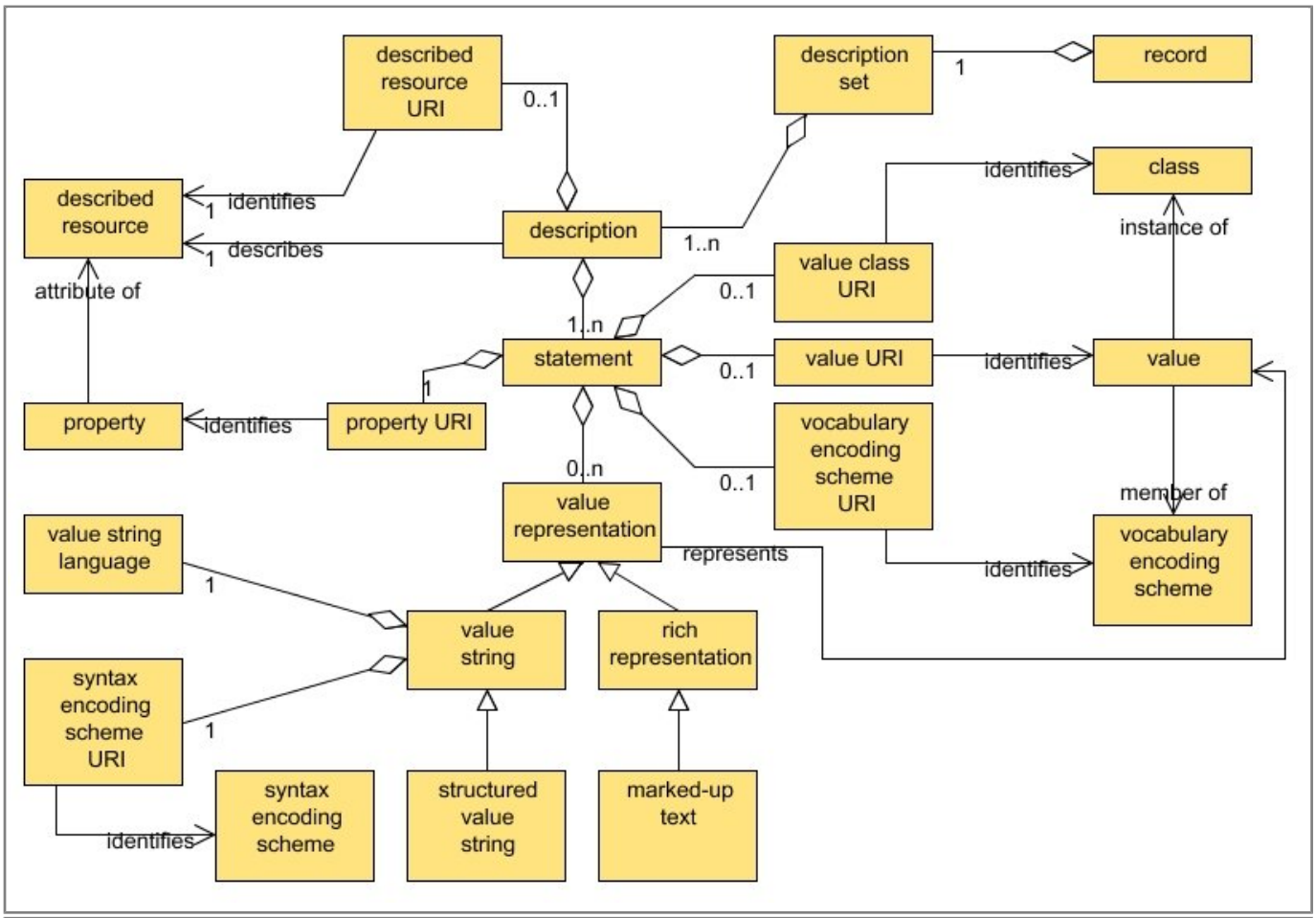

Figure 3 - the DCMI description model

The italicized words and phrases used above are defined in the terminology section below. A number of things about the model are worth noting:

- Each *value* may be the *described resource* in a separate related *description*.
- A related *description* describes a related *resource* and is therefore not part of the *description* - for example, a related *description* may provide metadata about the person that is the creator of the *described resource*.
- *Syntax encoding schemes* are also known as 'datatypes' in some contexts.
- In DCMI metadata *descriptions*, the *class* of the *described resource* is typically indicated by the *value* of the DC Type *property*.

The DCMI abstract model is represented as UML class diagrams [UML] in figures 1, 2 and 3.

Readers that are not familiar with UML class diagrams should note that lines ending in a block-arrow should be read as 'is' or 'is a' (for example, "a *value* is a *resource*") and that lines starting with a block-diamond should be read as 'contains a' or 'has a' (for example, "a *statement* contains a *property URI*"). Other relationships are labeled appropriately. The entities represented by the clear boxes are not mentioned explicitly in the textual description of the abstract model above but are discussed in Appendix A. Note that the UML modeling used here shows the abstract model but is not intended to form a suitable basis for the development of DCMI software applications.

3. Descriptions, description sets and records

The abstract model described above indicates that each DCMI metadata *description* describes one, and only one, *resource*. This is commonly referred to as the one-to-one principle.

However, real-world metadata applications tend to be based on loosely grouped sets of *descriptions* (where the *described*

resources are typically related in some way), known here as *description sets*. For example, a *description set* might comprise *descriptions* of both a painting and the artist. Furthermore, it is often the case that a *description set* will also contain a *description* about the *description set* itself (sometimes referred to as 'admin metadata' or 'meta-metadata').

Description sets are instantiated, for the purposes of exchange between software applications, in the form of metadata *records*, according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS].

This document defines a *description set* and a DCMI metadata *record* as follows:

- A *description set* is a set of one or more *descriptions* about one or more *resources*.
- A DCMI metadata *record* is a *description set* that is instantiated according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)

4. Values

A DCMI metadata *value* is the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*. For example, the *value* of the DC Creator *property* is a person, organization or service - a physical entity. The *value* of the DC Date *property* is a point (or range) in time - a conceptual entity. The *value* of the DC Coverage *property* may be a geographic region or country - a physical entity. The *value* of the DC Subject *property* may be a concept - a conceptual entity - or a physical object or person - a physical entity. Each of these entities is a *resource*.

The *value* may be identified using a *value URI*; the *value* may be represented by one or more *value strings* and/or *rich representations*; the *value* may have some *related descriptions* - but the *value* is a *resource*.

5. Dumb-down

The notions of 'simple DC' and 'qualified DC' are widely used within DCMI documentation and discussion fora. This document does not present a definitive view of what these phrases mean because their usage is somewhat variable. However, in general terms, the phrase 'simple DC' is used to refer to DC metadata that only makes use of *properties* in the in the Dublin Core Metadata Element Set [DCMES], that does not make any use of *encoding schemes*, and in which each *statement* only contains a *value string*. The phrase 'qualified DC' is used to refer to metadata that makes use of all the features of the abstract model described here.

The process of translating qualified DC into simple DC is normally referred to as 'dumbing-down'. The process of dumbing-down can be separated into two parts: *property* dumb-down and *value* dumb-down. Furthermore, each of these processes can be approached in one of two ways. Informed dumb-down takes place where the software performing the dumb-down algorithm has knowledge built into it about the *property* relationships and *values* being used within a specific DCMI metadata application. Uninformed dumb-down takes place where the software performing the dumb-down algorithm has no prior knowledge about the *properties* and *values* being used.

Based on this analysis, it is possible to outline a 'dumb-down algorithm' matrix, shown below:

	Element dumb-down	**Value dumb-down**
Uninformed	Discard any *statement* in which the *property URI* identifies a *property* that isn't in the Dublin Core Metadata Element Set [DCMES].	Use *value URI* (if present) or *value string* as new *value string*. Discard any *related descriptions* and *rich representations*. Discard any *encoding scheme URIs*.
Informed	Recursively resolve sub-property relationships until a recognised *property* is reached and substitute the *property URI* of that *property* for the existing *property URI* in the *statement*. If no recognised *property* is reached, then discard the *statement*. (In many cases, this process stops when a *property* is reached that is not an element refinement.)	Use knowledge of any *rich representations*, *related descriptions* or the *value string* to create a new *value string*.

Note that software should make use of the DCMI term declarations represented in RDF schema language [DC-RDFS], the DC XML namespace URIs [DC-NAMESPACES] and the appropriate DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS] to automate the resolution of sub-property relationships.

In cases where software is dumbing-down a *description set* containing multiple *descriptions*, it may either generate several 'simpler' *descriptions* (one per *description* in the original *description set*) or a single 'simple' *description* (in which case it will have to determine which is the 'primary' *description* in the original *description set*). This is an application-specific decision.

6. Encoding guidelines

Particular encoding guidelines (HTML meta tags, XML, RDF/XML, etc.) [DCMI-ENCODINGS] do not need to encode all aspects of the abstract model described above. However, DCMI recommendations that provide encoding guidelines should refer to the DCMI abstract model and indicate which parts of the model are encoded and which are not. In particular, encoding guidelines should indicate the mechanism by which *resource URIs* and *value URIs* are encoded. Note that the abstract model does not indicate that a *value string* with an associated http://purl.org/dc/terms/URI *syntax encoding scheme* should be treated as a *value URI* or *resource URI*. Encoding guidelines should provide an explicit mechanism for encoding these features of the model. Encoding guidelines should also indicate whether any *rich representations* or *related descriptions* associated with a *statement* are embedded within the *record* or are encoded in a separate *record* and linked to it using a URI.

7. Terminology

This document uses the following terms:

class

> A *class* is a group containing members that have attributes, behaviours, relationships or semantics in common; a kind of category.

class URI

> A URI that identifies a class.

described resource

> A *resource* that is described by a *description*.

described resource URI

> A URI that identifies the *described resource*.

description

> A *description* is made up of one or more *statements* about one, and only one, *described resource*.

description set

> A *description set* is a set of one or more *descriptions*.

element

> Within DCMI, *element* is typically used as a synonym for *property*. However, it should be noted that the word element is also commonly used to refer to a structural markup component within an XML document.

element refinement

> An *element refinement* is a *property* of a *resource* that shares the meaning of a particular DCMI *property* but with narrower semantics. Since *element refinements* are *properties*, they can be used in metadata *descriptions* independently of the *properties* they refine. In DCMI practice, an *element refinement* refines just one parent DCMI *property*.

encoding scheme

> An *encoding scheme* is either a *vocabulary encoding scheme* or a *syntax encoding scheme*.

encoding scheme URI

> An *encoding scheme URI* is either a *vocabulary encoding scheme URI* or a *syntax encoding scheme URI*.

has domain

> A *has domain* relationship between a *property* and a *class* indicates that if a *described resource* is related to a *value* by the *property*, then it follows that the *described resource* is an instance of that *class*.

has range

> A *has range* relationship between a *property* and a *class* indicates that if a *described resource* is related to a *value* by the *property*, then it follows that the *value* is an instance of that *class*.

instance of

An *instance of* relationship between a *resource* and a *class* indicates a *class* of which the *resource* is an instance.

marked-up text

A string that contains HTML, XML or other markup (for example TeX) and that is associated with the *value* of a *property*.

member of

A *member of* relationship between a *resource* and a *vocabulary encoding scheme* indicates a set of which the *resource* is a member.

property

A *property* is a specific aspect, characteristic, attribute, or relation used to describe *resources*.

property URI

A *property URI* is a URI that identifies a single *property*.

property/value pair

A *property/value pair* is the combination of a *property* and a *value*, used to describe a *resource*.

qualifier

Qualifier was the generic name used for the *terms* that are now usually referred to specifically as *element refinements* or *encoding schemes*.

record

A *record* is a *description set* that is instantiated according to one of the DCMI encoding guidelines (XHTML meta tags, XML, RDF/XML, etc.)

resource

A *resource* is anything that has identity. Familiar examples include an electronic document, an image, a service (e.g., "today's weather report for Los Angeles"), and a collection of other *resources*. Not all *resources* are network "retrievable"; e.g., human beings, corporations, concepts and bound books in a library can also be considered *resources*.

rich representation

Some marked-up text, an image, a video, some audio, etc. (or some combination thereof) that is associated with the *value* of a *property*.

statement

A *statement* instantiates a *property/value pair* and is made up of a *property URI* (a URI that identifies a *property*), zero or one *value URI* (a URI that identifies the *value* of the *property*), zero or one *vocabulary encoding scheme URI* (a URI that identifies the *vocabulary encoding scheme* of which the *value* is a member), zero or one *value class URI* (a URI that identifies the *class* of the *value*) and zero or more *value representations* of the *value*.

sub-class of

A *sub-class of* relationship between two *classes* indicates that the two *classes* share some *semantics* such that all *resources* that are instances of the sub-*class* are also instances of the related *class*).

sub-property of

A *sub-property* relationship between two *properties* indicates that the two *properties* share some *semantics* such that whenever a *resource* is related to a *value* by the sub-*property*, it follows that the *resource* is also related to that same *value* by the *property*.

syntax encoding scheme

A *syntax encoding scheme* indicates that the *value string* is formatted in accordance with a formal notation, such as "2000-01-01" as the standard expression of a date.

syntax encoding scheme URI

A *syntax encoding scheme URI* is a URI that identifies a *syntax encoding scheme*.

term

A *property* (i.e. *element* or *element refinement*), *class*, *vocabulary encoding scheme*, or *syntax encoding scheme*.

term URI

A URI that identifies a *term*.

value

A *value* is the physical or conceptual entity that is associated with a *property* when it is used to describe a *resource*.

value URI

A *value URI* is a URI that identifies the *value* of a *property*.

value representation

A *value representation* is a surrogate for (i.e. a representation of) the *value*.

value string

A *value string* is a simple string that represents the *value* of a *property*. In general, a *value string* should not contain any marked-up text.

value string language

The *value string language* is an ISO language tag that indicates the language of the *value string*.

value class URI

A URI that identifies a *class* of which the *value* is an instance.

vocabulary encoding scheme

A *vocabulary encoding scheme* is an enumerable set of which a *resource* is a member, such as the Library of Congress Subject Headings.

vocabulary encoding scheme URI

A *vocabulary encoding scheme URI* is a URI that identifies a *vocabulary encoding scheme*.

These links provide access to the three UML class diagrams shown above in a form suitable for loading into UMLet (the tool that was used to create them).

- dcam-resource-model.uxf
- dcam-vocabulary-model.uxf
- dcam-description-model.uxf