Using the DC Abstract Model to Support Application Profile Developers

Sarah Pulis
La Trobe University
Tel: +61 4 17 574 502
mailto:sarah.pulis@latrobe.edu.au

Liddy Nevile
University of Tsukuba/
La Trobe University
Tel: +61 419 312 902
liddy@sunriseresearch.org

Abstract:

In this paper, we consider the use of the DCMI Abstract Model as a potential vehicle for communication between developers of metadata application profiles and system developers. We note that it is not intended to be a developer's tool and consider what would make it so while maintaining its semantic integrity so designers of Application Profiles could also use it. We propose a UML compliant model of the DCAM as a first step towards the final development of a UML meta-model that will allow for the use of valid UML by those developing DC conformant Application Profiles. A number of characteristics of the DCAM are analysed and refinements suggested to enable a UML-conformant version of the DCAM which is required to build a new UML metamodel for DC Application profile developers.

Keywords
UML, DCAM, DCMT, Dublin Core, Application Profile, Meta-model.

1 Introduction

This paper is based on real world experience in which an experienced systems architect and a metadata Application Profile developer tried to work together to construct an Application Profile that was both easily implemented and Dublin Core conformant. The systems architect wanted only information expressible in conformant Unified Modeling Language (UML (i)) that he used to model the system and instruct his developers.

The Metadata Application Profile (MAP) developer was familiar with Dublin Core Metadata Terms (DCMT) and wanted to satisfy locally specific needs and support the importation of metadata from other schemas. She wanted to use qualified DCMT to exploit some of the features of the Semantic Web enabled by the Resource Description Framework (RDF (ii)).

The first problem encountered was that although UML is in common use for expressing systems for implementation in standard XML, it could not be used in this case because the data would need to be in RDF/XML which is not supported well by standard UML (as shown below). The second problem was that, while it is possible to use the extension features of UML to adapt the UML meta-model so that UML could be used, its development was not possible within the scope of the project.

In this paper, we report on the first stage of our work in the development of a UML meta-model. It will be used to extend UML in the way proposed, so future MAP developers can express in UML, a DC Application Profile conformant to RDF, so the programmers can use RDF/XML. The UML models developed using this meta-model will support but not require the use of RDF/XML: they will also work for developers who work with standard XML and other object oriented languages. Our first stage was the step from the first DCMI Abstract Model (DCAM (iii)) to what we think of as a UML-conformant version of that model and call the DCAM-UML.

To achieve our goals, we analysed the documentation and graphical representations of the DCAM, and then we built a UML model with what was available. In some cases, this meant some interpretation or decisions that may not be accepted by DCMI. We make them explicit to enable clear decision-making. We intend to use our model, once it is acceptable to the DCMI, to develop the UML meta-model so that UML can be used as described above.

2 Dublin Core Abstract Model

The DCAM consists of two models: the DCMI resource model and the DCMI description model. The models distinguish between the resource being described and the metadata description of the resource. Both models are described in text and graphically, using the notation and some of the common constructs of UML class diagrams (such as classes, and generalization and association relation-ships). It should be noted that its authors, Powell et al., state 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”. Software developers are, however, explicitly stated to be one of the three target audiences for the DCAM, the other two being developers of syntax encoding guidelines and of application profiles.

The first version of the DCAM was a significant achievement, following the earlier representation of the DCMT as a grammar by Thomas Baker (iv), and the DCAM is an important base for ongoing work.

3 Unified Modeling Language

The Unified Modeling Language (UML) is used to state explicitly, and in detail, the decisions made in a system. These decisions apply to all processes in the development and deployment of a system, from analysis and design to implementation and testing. The UML is not a visual programming language but a modeling language tailored to but not restricted to representing object-oriented systems. A set of complete UML diagrams can be mapped into a specific programming language, such as C++ or Java, or a table in a relational database.

UML facilitates the documentation of the architecture of a system while the system is being developed as well as when it is deployed. UML is distinct from any method, methodology or software development process. It provides building blocks for system modeling, not the instructions or method of their use.

3.1 Class Diagrams

Class diagrams are the most commonly used diagrams in the UML diagram set. Class diagrams are used to model static components of the system, in particular, the vocabulary database schema of a system, and simple collaborations. Using class diagrams, components can be shown graphically and mapped to object-oriented programming languages. Complex class diagrams can be formally sub-divided into packages.

3.2 Class Diagrams and RDF

In UML class diagrams, a property (association or attribute) must be defined relative to a classifier. A similar principle applies to attributes.

A property is defined in RDF as a first-class entity. It need not have a range or domain. An RDF property can be defined relative to zero or more classes (domains).

Baclawski et al. (v) removed the domain and range restriction from a UML property so that a property could be a first-class entity. They used the UML extension mechanism and a meta-model layer of UML that defines the UML itself. The adjustment at the meta-model layer has effect at the modeling layer. In fact, they also demonstrated how UML is extended.

4 Analysis of the DCAM

We look now at some specific characteristics of the DCAM. We are most interested in those characteristics that make it difficult to represent the DCAM in conformant UML.

4.1 Semantics

In the DCAM, the term ‘semantics’ refers to both informal and formal semantics. Formal semantics include those shown in the DCMI resource model, such as sub-property and sub-class and others such as the definition or label of a property or class. Informal semantics include the ‘meaning’ of a property or class, which can never be formally expressed.

Apparently, the DCMI definition of ‘semantics’ was intentionally left open to include any and all semantic descriptions (vi). We have a problem with this.

The structure of the DCMI resource model allows only the name of an object, not its ‘formal’ semantics, to be stored. We blame this problem on the definition of semantics. From a software modeling point of view, a single class cannot represent any and all semantic descriptions.

4.2 Property and sub-property

In DCMI resource model description, “each *property* may be related to one or more other *properties* by a refines (sub-property) relationship.” In the DCAM UML-type diagram, a sub-property is modelled as a class not as a relationship between two properties as indicated in the text description. According to the diagram, a sub-property can be related to zero or more properties and vice versa. This is known to be a problem (vii).

4.3 Classes and vocabulary encoding scheme

In the DCMI resource model description, Powell et al. say "where the *resource* is a *value*, the *class* is referred to as a *vocabulary encoding scheme"*. A generalisation relationship is shown in the DCAM description model between what in UML would be defined as the subclass vocabulary encoding scheme and the superclass class. This relationship enables a sub-class to be related to a vocabulary encoding scheme. This means that in some cases, a value *is* a resource that inherits the relationships of resources (i.e. the relationship between resource and class and resource and property/value pair).

Although it is understandable that a class such as *texts* may have a sub-class or be a sub-class of another class, it is unusual for a vocabulary encoding scheme to be associated with sub-classes. Nilsson, a co-author of DCAM, confirmed that this decision was intentional although not all approved of the structure (viii).

4.4 UML Attributes

The DCAM is made up of relationships and classes that do not include any attributes. Classes must have attribute(s) to be able to store information or values when instantiated. This problem is similar to the problem in Section 4.1 and occurs throughout the DCMI description model.

4.5 Metadata Statements

In the DCMI description model, a metadata statement consists of one property URI, zero or one value URIs, zero or one vocabulary encoding schemes and zero or more value representations. This allows a metadata statement to consist of a property URI with no value URI or representation. Similarly, a statement can consist of a property URI and a vocabulary-encoding scheme without having a value representation or a value URI.

This appears to contradict Baker's grammar that indicates a Dublin Core statement consists of a property and a value but Nilsson confirmed it and said it is based on the structure of a metadata statement (ix). A metadata statement does not include a value but rather includes an identifier or representation of a value, that is, a value URI or a value representation. Nilsson argues that even if a value does not have a value URI or a value representation, the value still exists.

If a metadata statement consists of only one property URI and a value representation, there is no way to associate the statement with the value, using the DCMI description model alone or combined with the DCMI resource model. (Alistair Miles noted this on the DCMI Architecture wiki. (x))

4.6 Value String Language and Syntax Encoding Scheme

In the DCMI description model, a value string can consist of zero or one value string languages and zero to one syntax encoding scheme URIs. This structure is supported by the text description but elsewhere it is said the value string language and syntax encoding scheme URI should be mutually exclusive (xi). The diagram should show that a value string can have a value string language *or* a syntax encoding scheme URI, but not both.

4.7 DCAM: 2 models or packages?

The DCAM consists of two logically separate models: the resource model and the description model. Although the DCAM does not make use of UML packages, it is possible that the elements (classes and relationships) in each model could be grouped together in a single model using the UML construct ‘package’. For this, the elements of the resource model would be contained within one package and the elements of the description model would be contained within another.

The problem is that while there is some supporting evidence for this approach, there is also some that supports thinking that each model is (more or less) a separate model that offers no more than a *different view* of the resource description. In UML, packages are formally structured to form a single model but the DCAM models are not.

5 DCAM-UML

In our UML version of the DCAM, we have made some minor changes to both diagrams. We adhere to the UML naming convention of starting a class name or new word with a upper case letter. We use a lower case letter to start an association name/role. We also correct inaccurate or missing cardinality within the UML diagram.

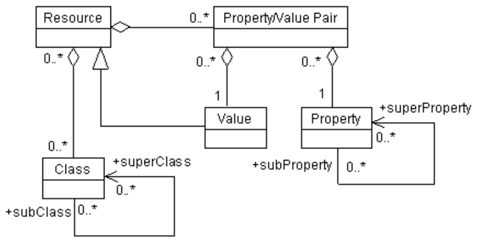

Figure 1: A UML conformant, modified version of the DCMI resource model.

5.1 Modified DCMI Resource Model

We removed (with no loss of information), the 3 classes of semantics, and refined class/property semantics that were redundant in the DCMI resource model. We replaced the class sub-property with a single association relationship between instances of Property. The cardinality is the same as for the association relationship between property and sub-property in the DCMI resource model. This avoids the difference between the text description of a sub-property as a relationship and its representation as a class in the DCMI resource model. The changes made to property and sub-property also apply to what used to be class and sub-class in the DCMI resource model.

5.2 Modified DCMI Description Model

Figure 2 is a UML conformant, modified version of the DCMI description model with fixes for some of the problems identified above.

For `ValueRepresentation` there is one attribute, `lexicalForm` with datatype `String`. The value string has become two classes: `PlainValueString`, associated with class `ValueStringLanguage` and an attribute language of type string, associated with a `SyntaxEncodingSchemeURI` that identifies a `SytanxEncodingScheme`. `ValueStringLanguage` also has one attribute `language` that has datatype `string`.

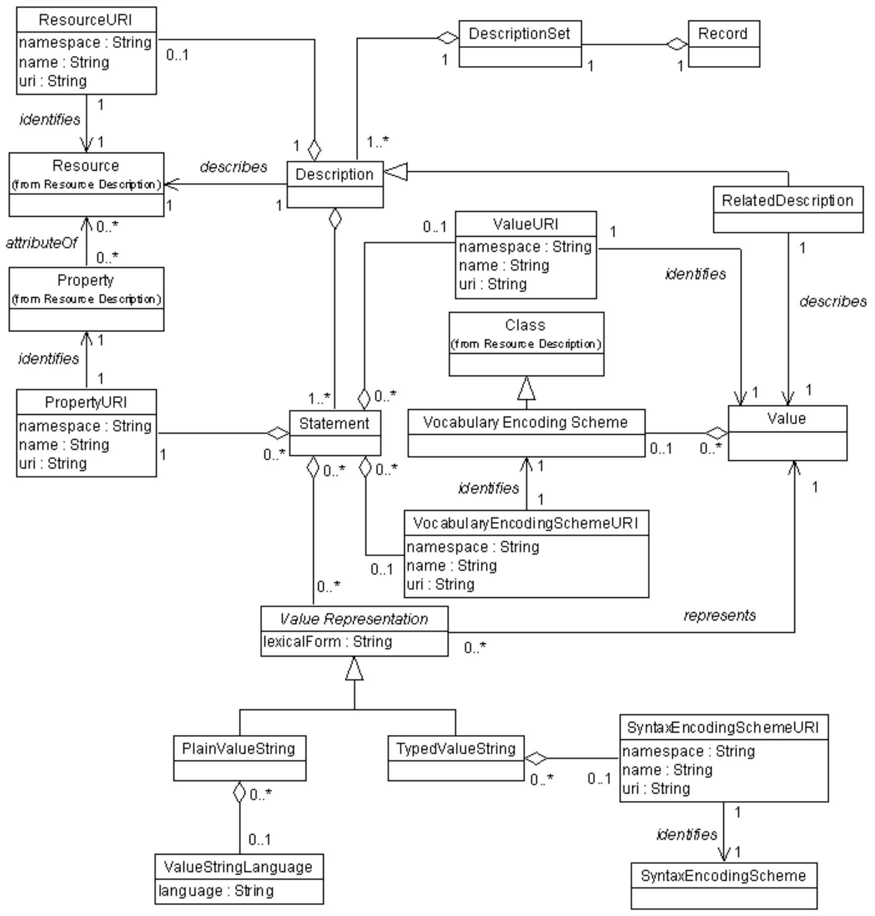

Figure 2: A UML conformant, modified version of the DCMI description model.

6 Outstanding Issues

There are a number of outstanding issues: a value may not be associated with a metadata statement; there is no way to represent semantics other than through the DCMI description model (e.g. propertyURI = rdfs:label, value representation = “Title”); there is still some redundancy between the DCMI resource model and DCMI description model, and a sub-class can still be associated with a vocabulary encoding scheme.

7 DCMI Meta-model

Having used UML to represent, as best we can, the current DCAM, we now propose a new UML model based on the DCAM. We use the convenience of UML packages.

7.1 DCMI Metadata Description Diagram

Figure 3 shows our proposed DCMI Metadata Description diagram as a modified version of the DCMI description model of the DCAM. It includes several major changes.

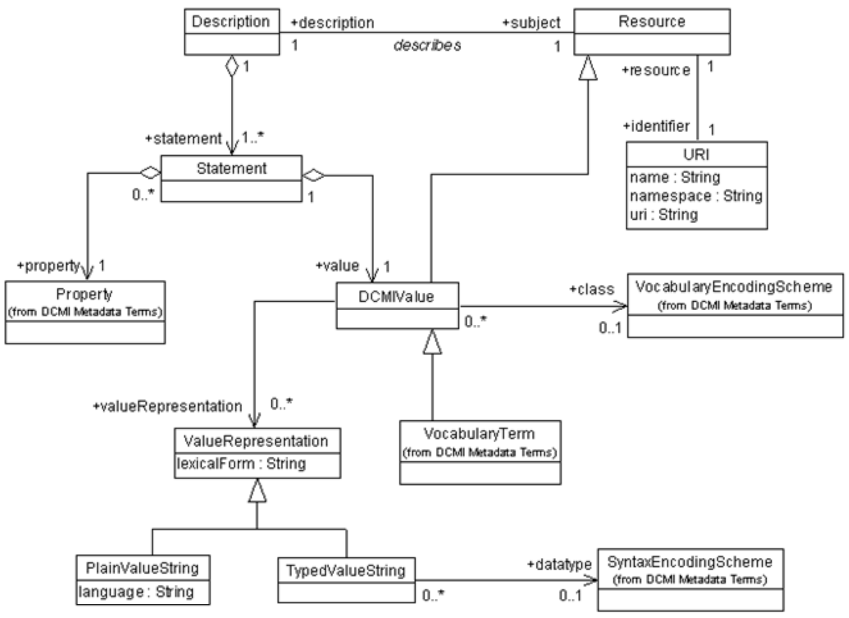

Figure 3: A DCMI Metadata Description diagram as a modified version of the DCMI description model of the DCAM.

We have removed the distinction between the resource being described and a metadata description of a resource. In our diagram, a `Statement` now consists of one `Property` and one `Value` (solving a problem explained above).

The association relationships between `Value` and `ValueRepresentation` and `Value` and `VocabularyEncodingScheme` are the same as in the DCAM. `Value`, a subclass of `Resource`, inherits the association relationship between `Resource` and `URI`, allowing a `Value` to be identified by a `URI` (equivalent to a value being identified by a value URI in the DCMI resource model).

`VocabularyTerm` is a subclass of `Value` based on the function of a vocabulary term as a value of a property. `VocabularyTerm`, a subclass of `Value`, may be associated with a `VocabularyEncodingScheme`.

The class `Description` is still related to a resource by an association relationship, and `Resource` is related to `URI` (previously resource URI in the DCMI description model). Specific URIs (e.g. resource URI, or syntax encoding scheme URI) are replaced by a single class `URI` which represents any URI. Through inheritance, classes that are subclasses of `Resource` (e.g. `SyntaxEncodingScheme`) can be associated with a URI.

7.2 DCMI Metadata Terms Diagram

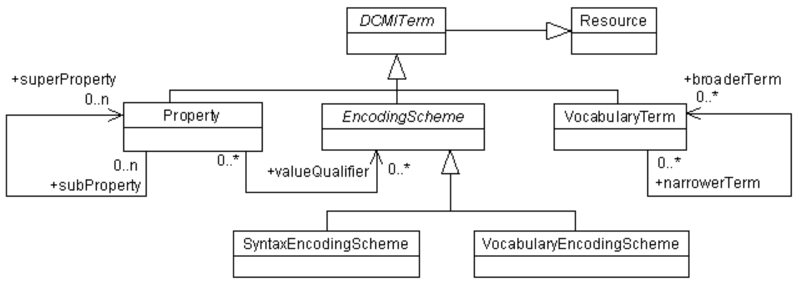

Figure 4: A DCMI Metadata Terms diagram as a modified version of the DCMI description model of the DCAM.

The hierarchy of classes in the DCMI Metadata Terms diagram represents the 'typology of DCMI metadata terms' (xii). DCMI metadata terms include elements, element refinements, encoding schemes and vocabulary terms (xiii).

The hierarchical relationship between DCMI metadata terms and specific types of terms is not included in either the text or the diagrams of the DCAM.

Our diagram uses generalisation relationships between terms. The class `DCMITerm` represents a DCMI metadata term. `DCMITerm` has three subclasses: `Property`, `EncodingScheme` and `VocabularyTerm`. `VocabularyEncodingScheme` and `SyntaxEncodingScheme` are subclasses of `EncodingScheme`. Our diagram includes association relationships to relate:

- two instances of `Property` with cardinality zero or more (either way);
- an instance of `Property` with an instance of `EncodingScheme` with cardinality zero or more (either way);
- two instances of `VocabularyTerm`, with cardinality zero or more (either way). This models hierarchical relationships between terms (e.g. *still image* is narrower than *image*).

We made some design decisions:

- `DCMITerm` inherits the association relationship of `DCMIResource` including the relationship between `DCMIResource` and `URI` from the generalisation relationship between the subclasses `DCMITerm` and `DCMIResource`. This means any subclass of `DCMITerm` (i.e. `Property`, `EncodingScheme`, `VocabularyTerm`, `VocabularyEncodingScheme` or `SyntaxEncodingScheme`) may be identified by a `URI`. As all these classes must be identified by a URI, such a constraint is added.
- The generalisation relationship between class `DCMIResource` and subclass `DCMITerm` enables DCMI metadata statements to be made about the subclasses of `DCMIResource` (see the DCMI Metadata Description diagram).

- `DCMITerm` and `EncodingScheme` are *abstract classes* (denoted by an italicised class name). An abstract class is a class that cannot have any direct instances.
- The association relationship between `Property` and `EncodingScheme` may seem to contradict that an encoding scheme applies to the value of a property but in the DCMT, an encoding *qualifies* a property, not a property value because here a property is not part of a statement.
- The relationship between a vocabulary term and encoding scheme is shown in the DCMI Metadata Description diagram. In this diagram, a `VocabularyTerm` is a subclass of `Value` that may be related to a `VocabularyEncodingScheme`. `VocabularyTerm` inherits the relationship between `Value` and `VocabularyEncodingScheme`, so an instance of `VocabularyTerm` may be related to an instance of `VocabularyEncodingScheme`.

7.3 DCMI Metadata Semantics Diagram

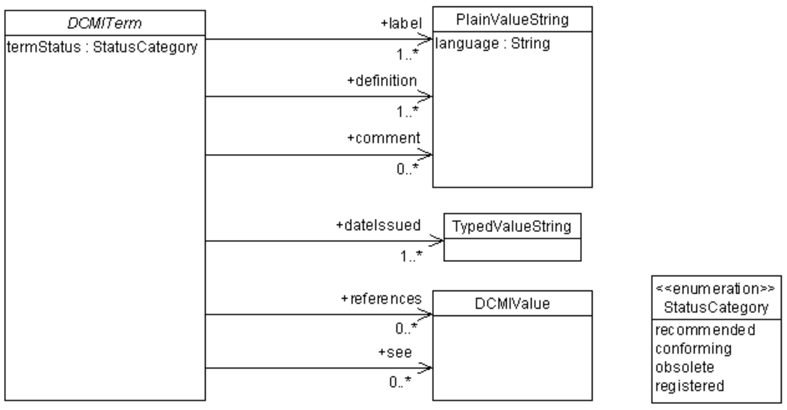

Figure 5: A DCMI Metadata Semantics diagram as a modified version of the DCMI description model of the DCAM.

The DCMI defines attributes (some mandatory) to describe Dublin Core Metadata Terms. We have treat them in three categories: identifier, relation and descriptor. Identifier attributes identify a metadata term; relation attributes link terms together, and descriptor attributes describe a metadata term in some way (e.g. with a human-readable label).

We use these attributes as semantics although only two of them are explicitly specified in the DCAM: refines, which links one property to another property, and broader/narrower than, which indicates that a class is more general or specific than another vocabulary term.

Identifier attributes (name and URI) are now the class `URI` and the attributes `name`, `namespace` and `URI`, in the DCMI Metadata Descriptions diagram, and relation attributes (refines, qualifies and broader/narrower than) are in the DCMI Metadata Terms diagram. The remaining description attributes are modeled in the DCMI Metadata Semantics diagram (Figure 5).

8 Conclusion

We responded to a problem in real life by working on how to make the DCAM useful to those systems architects who use UML. We propose some new versions of the DCAM as a step towards being able to develop a UML meta-model so that UML users will be able to work with conformant Dublin Core Application Profiles that use RDF. We look forward to some interesting discussions.

9 References

i Object Management Group (OMG). "The Unified Modeling Language", 2006. Available: <http://www.uml.org/>.

ii W3C, "Resource Description Framework", 2006. Available: <http://www.w3.org/RDF/>.

iii Powell, A, et al., "DCMI Abstract Model", March 2005. Available: <http://www.dublincore.org/documents/abstract-model/>.

iv Baker, T., "A Grammar of Dublin Core," October 2000. D-Lib Magazine, vol. 6, no. 10. Available: <http://dlib.anu.edu.au/dlib/october00/baker/10baker.html>.

v Baclawski, K., et al, "Extending UML to support ontology engineering for the semantic web," in *Proceedings of the Fourth International Conference on UML*, 2001. Available: <http://www.springerlink.com/media/h82chcawxg7qvj4hugdh/contributions/b/0/e/k/b0ekn07kqfy23hca.pdf>.

vi Nilsson, M, "Re: Interpretations of the DC Abstract Model", December 2005. Available: <http://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind0512&L=dc-architecture&T=0&X=1DCDC51CF29433904C&Y=sarah.pulis%40latrobe.edu.au&P=618>.

vii DCMI Architecture WG, "Issues with (and suggested changes to) the DCMI Abstract Model" 2005. Available: <http://dublincore.org/architecturewiki/AMIssues>.

viii Nilsson, M., (2006, January) Re: Interpretations of the DC Abstract Model. Available: <http://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind0601&L=dc-architecture&T=0&X=57B2E87E7AE56D7B04&Y=sarah.pulis%40latrobe.edu.au&P=679>.

ix (same as 6).

x (same as 7).

xi (same as 8).

xii DCMI Usage Board. "DCMI Grammatical Principles", 2003. Available: <http://www.dublincore.org/usage/documents/2003/11/18/principles/>.

xiii Powell, A. & Wagner, H, (eds.), "Namespace Policy for the Dublin Core Metadata Initiative", 2001. Available: <http://dublincore.org/documents/dcmi-namespace/index.shtml>.