A Short Paper – Principles for Structuring Data-oriented Interoperability Specifications

This is a "position paper" for the September 24th discussion meeting, "The Future of Interoperability Standards – Technical Approaches", organised by JISC CETIS in association with the ICOPER project. http://wiki.cetis.ac.uk/Future_of_Interoperability_Standards_September_2010

Adam Cooper, JISC CETIS, 2010-09-07

1. Introduction 2
2. Challenges 2
3. The Principles – Structural Aspects 3
3.1. Define a Conceptual Model 3
3.2. Organise Components in Layers 4
3.2.1. The Bottom Layer 4
3.2.2. Constraints I - Associating Classes and Properties 4
3.2.3. Constraints II - Assembly and Application Profiles 4
3.2.4. Community-specific Profiles 4
3.2.1. A Note on Behaviour 5
3.3. Separate Encoding 5
3.4. Externalise Common Term Values 5
3.5. Avoid Message or Process Semantics 5
3.6. Allow Referential Use 5
4. The Principles – Technical Aspects 6
4.1. Architecture of the Web 6
4.2. Classes (Types) 6
4.3. Properties 6
4.4. Term Values (aka Vocabularies) 7
4.5. Global Identifiers 7
4.6. Constraints 7
4.7. Data Types 7
4.8. Re-use (from elsewhere) 8
4.9. Graceful Degradation 8
5. The Principles – Common Patterns 8
6. Reconciling Existing Practice 9

7. Examples of Existing Practice and References 9
7.1. OAGi 9
7.2. DCMI Singapore Framework and Abstract Model 9
7.3. CEN MLO-AD 10
7.4. LEAP2A 10
8. Acknowledgements 10
9. Copyright and Licence 10

1. Introduction

This paper suggests, at a high level, a set of guiding principles for specifications designed for exchanging data (processes and services are a separate concern). It tries to take a view that is one-step removed from current practices and to identify some general principles that can be applied across a range of practices.

It has been created for discussion and as a contribution to debate about how we should approach standard design and documentation in the future, improving on weaknesses and avoiding identifiable challenges. It is inspired by a number of examples of emerging practice and lessons learned, and largely drawn from people other than the author too numerous to acknowledge, over the last decade of work in learning technology specifications/standards.

It is assumed that the specifications are intended for general adoption rather than for implementation of point solutions. i.e. that they are "standards" in loose terms.

This paper is intentionally limited to principles relating to the technical expression and does not cover some very important matters of process such as "conceptual calibration"[1] before creating the specification and "many eyes"[2] during its creation.

2. Challenges

These guiding principles are intended to mitigate against the effects of several inter-related challenges (but cannot not guarantee success):

- We live in a complex adaptive system[3]. It is not possible to identify entirely isolated sub-systems and change and un-anticipated application are ubiquitous.

[1] This term is borrowed from Ambjörn Naeve to indicate a process where a shared conceptual model is constructed without intending to refer to a specific process. See for example http://kmr.nada.kth.se/papers/SemanticWeb/HSW.pdf

[2] This term is common in the Open Source community, the idea being that quality is improved as more people with varied experience engage with the source code.

[3] For more discussion on this point, see http://blogs.cetis.ac.uk/adam/2010/04/14/key-challenges-in-the-design-of-learning-technology-standards-observations-and-proposals/

- In practice we have to draw boundaries over relatively closed and relatively temporally-stable regions in order to achieve interoperability but this is always a compromise.
- While activity will generally have to focus on solutions to an immediate problem in a sub-domain, the specifications that that are created must be capable of being applied to the whole education domain and beyond.
- Universal consensus is impractical and we must accommodate conceptual and operational differences while optimising use of common factors.
- No organisation will ever manage specifications for the whole domain, nor have a monopoly or power of mandate. A melange will emerge from a networked system of actors.
- Semantic interoperability is recognised as being essential for practical interoperability. This implies that the meaning of terms should be the starting point in the design of process (although not necessarily so in all supporting documentation).

3. The Principles – Structural Aspects

The above challenges indicate the necessity for interoperability specifications to be open in the sense that components of them may be adopted and adapted with minimal obstacles to the greatest variety of actors. NB: this is not meant to imply that there is no need for "locked down" specifications and standards, simply that lock-down should not be ubiquitous but should be one end of a spectrum of differential re-use.

Segmentation and "layering" of the specification is one way to compensate for the challenges and specifically to allow for differential re-use[4]. More than this: an approach is desired where definitions can be incremental and distributed in character. The highly structured approach proposed below takes this approach - which is aimed at those who would create interoperability specifications - but it is not optimally-accessible to implementers. For this reason, more developer-friendly guides and summaries that lack segmentation and include examples (etc) should generally be produced in practice.

In the ideal case, the specification should be structured along the following lines, interpreting them as inspiration rather than edict.

3.1. Define a Conceptual Model

This should be wholly independent of any paradigm such as Object Orientation, Relational Database, XML… It should also scrupulously avoid any pseudo-concepts that are matters of convenience for the specification writer; they must be meaningful to actors in the target context of the specification.

[4] The metaphor of "shearing layers", referred to in http://blogs.cetis.ac.uk/adam/2010/04/14/key-challenges-in-the-design-of-learning-technology-standards-observations-and-proposals/ is relevant to this view of differential re-use.

The method by which the conceptual model is developed is not in scope of this document but the early definition of a conceptual model and its status as an abstraction of what people do in practice[5] is of the utmost importance.

3.2. Organise Components in Layers

The following layers should be seen to build-up in a progression as they are ordered. It does not follow that all layers must always be present. The "Bottom Layer" is logically the most re-usable so should be practically the most re-usable (specification structure and licence to maximise re-use).

3.2.1. The Bottom Layer

The following should be independently defined:

- Classes
- Properties independent of the classes they may be applied to but including the classes of thing that a property refers to[6].
- Lists of permitted values for vocabulary terms (and potentially any relationship between these terms e.g. thesaurus rather than a simple collection)

3.2.2. Constraints I - Associating Classes and Properties

Although not generally considered to be constraints, statements about the properties used to describe a class are usefully considered to be constraints. Object-oriented design would have it differently, seeing properties as being intrinsic, but making these associations extrinsic offers us the chance of more reusable and durable definitions by separating the semantics from descriptive structure.

3.2.3. Constraints II - Assembly and Application Profiles

Classes, properties and term value lists, potentially sourced from various specifications, should be assembled to "realise" the connectedness of the conceptual model for a particular application.

The scope of application is becoming more constrained at this layer, a necessity for practical use, but the earlier layers exist independently of the constraints.

"Constraints II" might also include organisation into independently-usable modules or progressive aggregation to make more specialised units, although the benefits of making these independent specifications – i.e. more independent lifecycle – should be considered.

3.2.4. Community-specific Profiles

Application profiles (above) should, when feasible, be considered as independent from cultural, or community-specific, variation. For example, resource metadata for scholarly works management and discovery is an "application". The community of chemistry academics may have additional

[5] A conceptual model is not a wholly abstract creation; it is an abstraction of what is operated upon as "reality" by a group of people. This isn't the place to delve into epistemology.

[6] For example, consider the property "creator" where defining the subject to be a person or an agent (a more abstract class) gives us different meaning.

requirements but should share as much as possible with the application profile and layer-on their additions. The principle of "graceful degradation" (see below) particularly relevant when considering community-specific profiles; they should specialise and extend.

In practice, it may be difficult to distinguish application profiles from community/cultural variation.

3.2.1. A Note on Behaviour

In general, data specifications should be confined to exchanging information and avoid defining how it should be processed. This is not universal but depends on the kind of specification. For example, IMS QTI states that "modal feedback is shown to the candidate directly following response processing". This level of description is necessary consequent of what "modal feedback" is; no guideline is presented on how modal feedback actually occurs.

This is not the same as import/export behaviour for conformance of software.

3.3. Separate Encoding

Encoding (e.g. as XML Schema, RDF, JSON) must be introduced to make a specification usable in practice but encoding details should be cleanly separated from identification and definition of the classes, properties, term values, constraints, ... etc.

Encoding may precede or follow organisation into modules, according to the specific case. In some cases, a standard may not include an encoding or an independent part, with its own lifecycle, may be created.

3.4. Externalise Common Term Values

"Common Term Values" - often known as "vocabularies" although some communities use "vocabularies" more generally – are defined conceptual entities used as values for a property. For example, the EQF levels 1-8 are one set of values for the term qualification level.

Term values should always be separately defined (following "Organise Components in Layers") in a specification. Common value sets should be externalised into a separate specification to decouple the management lifecycles.

3.5. Avoid Message or Process Semantics

A data specification for entity "A" should not include information about a possible message about "A" or process involving "A". If a specification includes data and messaging components then these should be cleanly segmented. For example, information about a person should not include status codes relating to the stage of an application-enrolment process that the person is engaged in. Such application-enrolment status information should be kept separate until the "Constraints II - Assembly and Application Profiles" stage. A good conceptual model should avoid this kind of concept-pollution.

3.6. Allow Referential Use

The specification should be structured so that the maximum number of component parts can be adopted by reference – i.e. not by duplication - by future workers. This implies a thorough approach to version control and archival. It also implies that component parts of the specification are defined

a-contextually; modifying the meaning of entities by the structural contexts in which they occur should be avoided at all costs[7].

This is most easily achieved by assigning universal identifiers to all classes, properties, vocabularies, term values, constraints, modules (and other aggregations),…

4. The Principles – Technical Aspects

4.1. Architecture of the Web

Data models should be compatible with the architecture of the web. This does not mean that non-web encodings are either forbidden or impossible. These principles can be taken to extremes (arguably this is the case in some W3C specs) but a more relaxed interpretation is useful.

4.2. Classes (Types)

Classes – using the term generally, for types of abstract or concrete thing, and particularly not in its OO sense - should be defined and identified independent of the properties that may be used to describe them. Classes should be specified only for meaningful concepts and generalisation/specialisation relations should be defined using the subClassOf semantics of RDF Schema[8].

4.3. Properties

Properties should be defined independently of the classes that they may be used to describe. For example, "creator" property has diverse uses.

The "range"[9] concept of RDF Schema should be preferred for associating classes as the object of properties. This may be a normative or informative statement. When defining the range of a property, choose the most generalised class suitable in preference to a multiplicity of more specialised classes[10].

Generalisation/specialisation relations should be defined using the subPropertyOf semantics of RDF Schema[11]. NB: adoption of the semantics of "subPropertyOf" as defined in RDF Schema does not mean that RDF-based encodings should or must be used.

[7] E.g. "title" (in a given namespace) has a different meaning depending on whether it is associated with a person and a book.

[8] http://www.w3.org/TR/rdf-schema/#ch_subclassof (NB: this doesn't mean the spec has to be expressed using RDF(S), merely that I am making an explicit reference to an existing definition)

[9] http://www.w3.org/TR/rdf-schema/#ch_range (Use of the term "range" predates RDF Schema but, as for subClassOf, I am reusing a concept identified by its URI)

[10] E.g. the range of "creator" in Dublin Core is "Agent". Any sub-class of "Agent" is also an Agent so instances of them may be values of a "creator" property, for example if one was to define a "Person" as a sub-class.

[11] http://www.w3.org/TR/rdf-schema/#ch_subpropertyof

4.4. Term Values (aka Vocabularies)

Each term value should be uniquely identified and its usage defined. Whenever possible a single term value should have multiple language-specific labels and scope notes in preference to the creation of multiple term values. Culture-specific lists are an exception.

Sets of values must be open for extension, replacement etc.

Reference to specific encodings of vocabularies (including "authority files") should be deferred to the encoding stage of the referring specification.

4.5. Global Identifiers

URIs are *de rigeur*. They should be assigned to all classes, properties and term values. They may also usefully be applied to sections of normative wording in the specification.

The URIs should be HTTP resolvable to information about the entity they identify.

4.6. Constraints

In general constraints should be avoided unless necessary and should be introduced as late as possible in the process of adding layers ("Organise Components in Layers").

The "domain"[12] concept of RDF Schema should be preferred for associating properties with the classes they describe but this should be avoided unless the property can only ever belong to that class.

Normative constraints should generally be expressed in natural language with reference to RFC2119 imperatives. They may additionally expressed in a formal language - such as OCL[13], OWL[14] or Schematron[15] – at the encoding level.

4.7. Data Types

Data type definitions should be adopted rather than defined *ab-initio*. The XML Schema data types[16] have become widely used to the point of being the preferred definitions. Data types should be generic/common unless there is good reason not to be so (e.g. use "int" and not "short"). If there is a widely used additional restriction on an XML Schema data type then that should be used (e.g. WC DTF[17] is a subset of permitted lexical space of XML Schema datetime).

[12] http://www.w3.org/TR/rdf-schema/#ch_domain Strictly speaking, domain is not a constraint but an assertion about the class the property is associated with.

[13] http://www.omg.org/technology/documents/formal/ocl.htm

[14] http://www.w3.org/TR/owl-features/

[15] http://www.schematron.com/

[16] http://www.w3.org/TR/xmlschema-2/

[17] http://www.w3.org/TR/NOTE-datetime

4.8. Re-use (from elsewhere)

Where possible, reuse components of other specifications by reference (i.e. rather than by duplication). Prefer the most widely-used components and be accurate (casual mis-use is pernicious). Re-use may simply be of a single class. This is often not easy with existing specifications, which is a problem. Dublin Core[18] is a prime candidate for reuse.

Widely used specifications that are essentially syntactic (in practice, even if not originally intended) are particularly useful as structures to map into: skeletons for encoding schemes. Atom, essentially a syntax for lists with minimal semantics, has been used in LEAP2A (reference later) to good effect. In addition to supporting "graceful degradation" (below), taking this approach allows developers to work with existing code and reduces the cognitive effort required to understand a specification.

4.9. Graceful Degradation

Graceful degradation is the process whereby either: software constructed with no knowledge of the interoperability specification can still identify some information; or, software constructed with knowledge of the specification but with lesser capability is programmed to accommodate information from applications with greater capability. These two cases approximately equate to syntactic degradation and semantic degradation respectively.

Syntactic degradation should be achieved by using common syntaxes (e.g. Atom, RDF/XML or XHTML) for the encoding. Semantic degradation should be achieved by specifying explicit generalisation/specialisation relationships for classes, properties and term values.

For term values that have structure rather than being simple collections, monolingual thesaurus relations should be used if possible[19].

5. The Principles – Common Patterns

Adopt existing conventions that are consistent with the guiding principles and have currency in your domain. These conventions might span naming, diagrammatic presentation, specification structure, data type subsets, lexical value-spaces, … The conventions might be the use of a common meta-model or "abstract framework".

Using an existing framework helps to make the specification more widely and easily intelligible and reduces the chance that incompatibilities between specifications are inadvertently introduced.

Coherence of LET standardisation could be greatly increased by definition and adoption of some common patterns and principles at an international or European scale.

[18] ISO 15836 represents an older version of DC than DCMI currently recommends. The DCMI recommendations should be preferred as they explicitly reference the legacy with "refines" semantics (see "Graceful Degradation").

[19] It may often not be possible to use monolingual thesaurus relations but more complex approaches are unlikely to be adopted except by specialised communities. It may be more appropriate for culture-specific vocabularies to be left out of the specification entirely and national/regional profiles created instead.

6. Reconciling Existing Practice

As was initially-stated, this document is forward looking. It is fair to ask: what of the past?

Firstly, it is not necessary to throw all the work of the past away. That would be a poor principle. When (if) the limitations of an existing standard/specification becomes apparent, a refactoring according to the above principles should be investigated as a means of salvaging the good parts and reducing impact of the change.

Secondly, the author is optimistic that many existing specifications could be refactored (and maybe augmented) to follow the above principles, possibly with no breaking-changes at the level of the encoding. This is, however, not something that has been undertaken in practice and it is known that reconciliation of Dublin Core Metadata and IEEE LOM is difficult.

7. Examples of Existing Practice and References

7.1. OAGi

The Open Applications Group[20] is primarily relevant to B2B supply-chain interactions in the chemical, hi-tech and metals industries.

They do not follow most of the above "guiding principles" – there is an implicit assumption that OAGi are managing specification creation - but do have a well-developed "Business Object Document" architecture[21] that cleanly separates the nouns (the "business objects") from the verbs in the message. They also advocate an approach to building-up nouns from common components.

7.2. DCMI Singapore Framework and Abstract Model

The Singapore Framework[22] is an concise example of a "Common Framework" that includes many of the principles outlined above. This document differs primarily in scope and by being less specific to DCMI than the Singapore Framework. It is worth noting that, although the essence of the Singapore Framework is uncontroversial, the "Description Set Profile"[23] (a mandatory component) is not widely adopted and its adoption is not recommended.

The Dublin Core Abstract Model[24] (DCAM) includes significantly more detail than this document and provides a set of practices that could usefully be adopted beyond the scope of Dublin Core Metadata. DCAM is not, however, intended to be universally applicable and its details should be fully understood before adoption.

[20] http://www.oagi.org/

[21] http://www.oagi.org/oagi/downloads/ResourceDownloads/BOD_Architecture.pdf

[22] http://dublincore.org/documents/singapore-framework/ (version-independent URI)

[23] http://dublincore.org/documents/dc-dsp/ (version-independent URL)

[24] http://dublincore.org/documents/abstract-model/

7.3. CEN MLO-AD

Specifically the CEN Workshop Agreement 15903[25], the core of which is in process as an EN.

CWA 15903 demonstrates many of the "guiding principles" outlined above with a few variations: the "Learning Opportunity Object" class appears to have been introduced as a convenience rather than being a domain-concept; domain restrictions are applied at the level of the class definition rather than being separately layered.

CEN MLO omits controlled vocabularies as they are either region/country or education-sector specific. It also omits details of encoding. Both of these omissions make it more useful as a specification to be referenced, re-used...

7.4. LEAP2A

LEAP2A[26] demonstrates, among other things, reuse of the widely-used syntax of Atom and shows two aspects of graceful degradation:

1. The LEAP class (type) information is included as an attribute of atom:entry and may be ignored by an Atom parser, which will still recover the essential content.

2. Specialised relationships are explicitly defined in relation to more broad relationships (e.g. "is outome of" degrades to "supports").

An alternative encoding – LEAP2R, an RDF encoding of LEAP2 – is in development. This builds on the common, behind-the-scenes, conceptualisation of LEAP2 independent of the LEAP2A encoding.

8. Acknowledgements

With thanks to people who made specific comment on drafts: Tore Hoel, Erlend Øverby, Lorna Campbell and Wilbert Kraan.

ICOPER[27] is coordinated by Wirtschaftsuniversität Wien and is co-funded by the European Community eContentplus Programme.

9. Copyright and Licence

[25] ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/WS-LT/CWA15903-00-2008-Dec.pdf

[26] http://www.leapspecs.org/

[27] http://www.icoper.org/

[28] http://creativecommons.org/licenses/by/2.0/uk/