OASIS SET TC
Requirements Proposals

July 08
Stephen Green SystML
Requirements Proposals for the Semantic Support for Electronic Business Document Interoperability

Stephen Green
SystML
stephen.green@documentengineeringservices.com
What aspects of documents might vary or be common across documents? Let's use a convention and call these 'Dimensions of Variability', and number them D1, D2, etc

**D1.** Namespaces (may vary or may be the same)

**D2.** Models

**D3.** Core Components

**D4.** Core Components Harmonization Group (private, TBG17, organization, etc)
Dimensions of Variability Subsets

These four may be sufficient for variations in subsets and some other restricted customizations

D1. Namespaces (may vary or may be the same)
D2. Models
D3. Core Components
D4. Core Components Harmonization Group (private, TBG17, organization, etc)
Extensions and cross-standard interoperability cases require further dimensions of variability:

- **D1. Namespaces** (may vary or may be the same)
- **D2. Models**
- **D3. Core Components**
- **D4. Core Components Harmonization Group** (private, TBG17, organization, etc)
- **D5. Underlying syntax** (XML, ASN.1, EDI, etc)
- **D6. Variations in basic datatypes** (and codelists)
- **D7. Naming and Design Rules** (UBL, ATG2, etc)
As a dimension, context could add eight or more dimensions of its own

- **D1. Models**
- **D2. Core Components**
- **D3. Core Components Harmonization Group** (private, TBG17, organization, etc)
- **D4. Underlying syntax** (XML, ASN.1, EDI, etc)
- **D5. Namespaces** (may vary or may be the same)
- **D6. Variations in basic datatypes** (and codelists)
- **D7. Naming and Design Rules** (UBL, ATG2, etc)
- **D8. Context / Purpose** (D8.1, D8.2, etc)

Or Context could be seen as orthogonal
Dimensions of Variability

- There are also variations in context scheme (CCTS, UCM, etc)
  - D1. Namespaces (may vary or may be the same)
  - D2. Models
  - D3. Core Components
  - D4. Core Components Harmonization Group (private, TBG17, organization, etc)
  - D5. Underlying syntax (XML, ASN.1, EDI, etc)
  - D6. Variations in basic datatypes (and codelists)
  - D7. Naming and Design Rules (UBL, ATG2, etc)
  - D8. Context / Purpose (D8.1, D8.2, etc)
  - D9. Context Scheme
Non-Functional Requirements

- There are more general requirements common to most technologies which need to be considered
  - Quality Assurance
  - Conformance Criteria
  - Testability
  - Versioning Strategy
  - Change Management
  - Authority
  - Affordability
Technical Requirements

- The functional and non-functional requirements may all impact on considerations of technical requirements
  - Ontologies are a preferred means of achieving the functional requirements in supporting document interoperability across variations in several dimensions
  - OWL has been used to demonstrate feasibility in meeting these requirements in this way
  - OWL is an ontology designed with the semantic web and its theories in mind
Implementation Issues

Quality: Deprecation and Versioning
Since OWL is an ontology designed for the semantic web it follows a theory of openness which may not be suitable for some of the non-functional requirements of document interoperability. It needs to be measured against the matrix of Quality Assurance, Conformance Criteria, Testability, Versioning Strategy, Change Management, Authority, Affordability. Even ‘Affordability’ might be compromised if too much has to be added to OWL.
These requirements (such as Versioning) were considered for OWL

W3C Recommendation 10 February 2004
http://www.w3.org/TR/2004/REC-webont-req-20040210/

“3.2 Ontology evolution

An ontology may change during its lifetime. A data source should specify the version of an ontology to which it commits.

An important issue is whether or not documents that commit to one version of an ontology are compatible with those that commit to another. Both compatible and incompatible revisions should be allowed, but it should be possible to distinguish between the two.”
But the requirements were not met in OWL:
http://lists.w3.org/Archives/Public/www-webont-wg/2002Dec/0075.html
Regarding the measures for versioning put in place:

“Note, this approach does not address the problem described in Section 3.2 of the Requirements Document (under RDF(S) Support).

There, we gave an example where we wanted to "fix" an incorrect definition of Dolphin.

Solving this problem would require versioning capabilities that would change the model theoretic semantics, but at this time it is not clear what the correct approach would be. A later version of OWL may address this issue.”
This is despite the presence in OWL of identifiers which look like they are what we need:

```
priorVersion
backCompatibleWith
incompatibleWith
deprecatedClass
deprecatedProperty
```
OWL: How Wise?

So, if you want to say ‘a dolphin is a fish’ using OWL fine, OWL ontologies typically won’t stop you.

If you then want to say ‘all fishes have gills’ that is OK too.

Then if someone else comes and says to your knowledge base ‘a dolphin is a mammal’ it will be OK with that.

Only when you say a dolphin has no gills will it probably report a contradiction.
The problem really comes when someone asks your OWL knowledgebase ’what is a Dolphin’ when it will probably report both ’fish’ and ’mammal’ and even try telling you a dolphin has gills too. This cannot be corrected using OWL. Perhaps not without unloading the knowledgebase and reloading it without the erroneous ontology will it be able to say correctly that a dolphin is a mammal and accept that ’a dolphin is not a fish’ is not a contradiction.
The matter doesn’t just hinder versioning and quality assurance, it may hinder conformance and testing too – the bedrock of interoperability.

There needs to be consideration of what two implementations of any ontologies for documents might be needed when claiming conformance to any SET deliverables.

Maybe a set of Test Assertions and a list of them in a conformance clause can help.

Each ontology which is certified as correct can map to a Test Assertion or part of a Test Assertion and a version would list all these.
Alternatively it might be that OWL is fine for development and feasibility / prototyping. There may be other ontology languages which support the SET requirements adequately, even if they may cost more to implement. There may then be a way to convert from OWL to this other ontology language. OWL has advantages in being widely known and having possibilities for growing support. Knowledgebase/reasoner support does not yet seem to me to be a huge factor as yet – some knowledgebases still use non-web languages.
OASIS SET TC

Back to the TC . . . Thank you