RDF Work Specification

DRAFT 2016-12-13

Oliver Hopt, Jon Johnson, Wendy Thomas, Joachim Wackerow, Benjamin Zapilko

# Background

DDI-Views has always had the goal of being expressed in multiple binding including but not limited to XML Schema and RDF. RDF knowledge within the current DDI community is limited and external assistance is needed to provide the basic work on the creation of the RDF binding of the UML model. The RDF binding must meet the following criteria:

* Support the ability to round-trip metadata content between the various bindings
* Use existing vocabularies when available, limiting the creation of DDI vocabulary items

(include reference to documents from previous sprints as appropriate)

(Find and reference document on Identification in XML and RDF)

# Requested Documents

## Document addressing general questions:

* Is there a value to having abstract classes in the RDF such as creating a graph view of the instances or for reasoning?
* Are there serious reasons to alter the workflow from the PIM to PSM to binding model structure? Current thinking is that it allows us to have a parallel process for each binding and that alterations to the configurations can be kept in sync more easily. External configuration programs are nice to have to separate the transformation logic, however it may be costly to have this separation. If configuration logic is held in the transformation file it must be well documented. Another consideration is how often changes need to be made as well as the level of transparency desired. Separate configuration files allow domain experts to manage configuration without deep technical knowledge.
* What is the advantage of modeling the selected RDF classes in the UML model in a separate package?
  + How is it used in publication of the RDF binding?
  + How is it handled in the XML or other forms of binding?
  + What is the management overhead in terms of change over time?
* How can inferences be supported? Should it just support the possibility of inference? For example, should we include the abstract classes to support inference?
  + For example, you can declare it in the OWL ontology as say a Label (non-executable) can write a SHACL or ShEx rule.
* What is the effort to build a SHACL or ShEx expression (validation syntax) for RDF? Is it based on the model or on the OWL/RDF-S syntax?
* We plan to generate JSON-LD at some point. What is the procedure to do this on the basis of the RDF ontology? What tools are available and recommended?
  + Is the a loss-less trip from JSON-LD to RDF possible?

## Document on Namespace Considerations:

* Should a namespace resolve to something concrete?
  + If resolvable, should content negotiation be used for different syntaxes and for what reason?
    - If yes, should it be resolvable only for RDF syntaxes or also for specifications in human readable text? (example, DISCO)
    - How is the relationship to the related XML Schema namespace maintained? Example: How should the Codebook namespace be resolved? Are there separate namespaces for RDF and XML?

## Document on Mapping:

The document should cover various approaches such as listed below, criteria for mapping, intended uses of mapping, etc.

* Pushing it from the configuration file to the UML
  + Reduces duplication between XMI and RDF configuration file
* By putting out overlap lists (PROV to FOAF for example)
  + Put out list of classes used in each vocabulary including overlap
  + Equivalencies are difficult to maintain and can have unintended consequences
  + Rely on external mapping rather than express internally
  + Everywhere you are using FOAF agent you can use DC Agent
    - If you have a new thing you have to say that it is a sub-class of both a FOAF Agent and DC Agent
* The approach needs to be property by property and relationship by relationship
* Use an iterative process of finding clear relationships
  + May have to edit the model to get a clean match but this cannot be at the expanse of requirements of the DDI community. The model should not be forced into being RDF centric any more than it should be XML centric.
  + Remaining items
    - Search for an appropriate vocabulary that meets the criteria
    - Create a primary DDI Vocabulary
      * Should be an extension (formal addition) or restriction (of the semantics) of an existing vocabulary if possible
      * As you go along you want to extend existing broader or looser concepts and relate any DDI vocabulary terms
* Discussion of use of relationship terms at the ontology level such as “sameAs”, “subclassOf”, “subpropertyOf”, etc.

## Document covering selected vocabularies and their use

### Criteria for selection

* Being used – the vocabulary is in active use
  + The existence of mapping to a vocabulary from other related vocabularies can be an indication of its prominence (http://lov.okfn.org/dataset/lov/vocabs)
* Strong Potential – the vocabulary appears to have a strong potential for use within its coverage area (filling a gap)
* Endorsed by W3C – a plus if other criteria are met
* Consider how the content is used by others such as Libraries scrapping Dublin Core objects to populate a catalog

### Recommended Vocabularies and Priority

* OWL/RDF-S, PROV-O, SKOS, XKOS, DCAT, DataCube, CSVW, PAV, ORG, DC (create a table associating these with specific package areas of the model include which communities outside of DDI might interface with this) (RDF-S is a subset of OWL minus many of the logical expressions of OWL, some RDF-S objects are easier to understand and are used within OWL, such as labels)
* RDF-S in preference to OWL. Other vocabularies should have an RDF-S or OWL origin if it is already in one of the other specific area vocabularies. Use the area specific vocabulary to retain the special semantics particularly in mapping sections of the model which have a specific related vocabulary.
* There is a requirement of ordering within collections
  + OrderList ontology – miniature ontology – just a sequence
  + Will need to make our own as there doesn’t seem to be a language to do that
  + DISCO used a SKOS construct – but we should use a native based ontology order
  + The goal is to have the complete Collection and Process Pattern in RDF – Do we need to rethink the inclusion of this type of detailed ordering in the RDF binding?
* FOAF is disallowed by the U.S. Federal Government

### Other possible vocabularies to explore (is there a reason to include or incorporate into DDI):

* SIO (Semanticscience Integration Ontology) http://semanticscience.org
* OBO Foundry http://www.obofoundry.org

### Syntax rules

* (Can we have the same syntax rules for the Model and all bindings?
  + Classes – Name starts upper case
  + Properties and Relationships – Name starts with lower case
  + Provide an overview table of current syntax rules [classes, properties, relationships | UML, XML, JAVA, RDF] “fill in the RDF”)

### Validation Rules

* Include cardinality

## Future proofing:

* Most robust should be the model
* Second would be the bindings
* PIM is the expression for the Model Library and Functional Views
  + The PIM will change based on additions to the model and addition of Functional Views
* The RDF Functional View provides the subset that would be relevant to a round-trip to an alternative binding
  + Also provides the Functional View specific documentation providing terminology and application information familiar to the target user group
* The syntax binding should remain stable except for providing for new classes in the model
  + Changes in the structure of RDF that would support desired functionality could drive changes to the RDF syntax binding