RDF Decisions and Open Issues: Dagstuhl Sprint, 16-20 October, 2017

# Overview

This document summarizes the objectives, goals, decisions and open issues surrounding the range of RDF-related topics at the 16-20 October 2017 Dagstuhl sprint. There were several different groups which addressed different aspects of the DDI RDF binding for Moving Forward, covering topics ranging from specific technical issues in the existing binding to the realization of round-trip capability between bindings, and how the DDI constructs relate to similar constructs in other popular RDF ontologies/vocabularies.

This exercise was conducted in the context of a need to produce useful outputs in the upcoming June 2018 “Prototype” release (covering Data Description, Data Capture, and Codebook portions of the model). In many cases, specific decisions were made regarding not only what the DDI Alliance would produce in the fullness of time, but what should or should not be included in the upcoming release. The release also placed significant emphasis on having a fully functional production system in place for the prototype release – this topic is discussed in other documents coming out of the sprint.

This document – rather than being organized around the many groups and sub-groups which addressed a set of related topics, is organized around the discussion which took place. Not every topic included in the initial agenda was covered (although most were), and some topics emerged during discussions which subsequently became important, although had not necessarily been anticipated.

This document concludes with a list of remaining open issues/discussion topics, and the requirements for a process moving forward for the resolution of these issues in a timely manner (especially as regards the prototype release).

# Goals and Proposed Agenda Topics

There was a detailed document listing topics for the sprint, incorporating preceding work, notably at the recent Cologne sprint. The background from that document reads:

*DDI-Views has always had the goal of being expressed in multiple binding including but not limited to*

*XML Schema and RDFS/OWL. Knowledge on these ontology description languages within the current*

*DDI community is limited and external assistance is needed to provide the basic work on the creation of*

*the RDFS/OWL binding of the UML model. The RDFS/OWL binding must meet the following criteria:*

* *Support the ability to round-trip metadata content between the various bindings, especially*

*with XML Schema*

* *Use existing RDF vocabularies when available, limiting the creation of DDI superfluous vocabulary items*

Specific sections of that document raised several topics, organized under the following headings:

* General Questions
* Namespace Considerations
* Mapping
* Selected Vocabularies and Their Use
	+ Criteria for Selection
	+ Recommended Vocabularies and Priority
	+ Other Possible Vocabularies
	+ Syntax Rules
	+ Validation Rules
	+ Use Cases
* Future Proofing

In each of these areas, documentation was intended to be produced. As we summarize the discussions conducted at the sprint, it will become clear that this range of topics was substantively covered, if not in a point-by-point fashion.

# Discussions and Decisions

## Technical Issues with the Existing RDF Binding Transformation

The existing transformation uses an RDF-S/OWL vocabulary definition from the DDI Moving Forward model. The transformation currently used is one that is largely unchanged from when it was originally created several years ago. When the current model was submitted to this transformation, the results were not satisfactory – RDF tools would produce fatal errors and warnings on import. One group was formed to explore the output of the existing technical transformation, and to identify how such errors could be avoided, whether these involved changes to the model or to the transformation process.

In resolving these errors, some issues were solved by making changes to the model (mainly consistency issues already addressed by existing modelling guidelines which had been violated) while others were resulting from specific technical features of the target binding. Among these, perhaps the most important were name collisions resulting from properties and relationships having the same name (producing an illegal OWL expression). These were addressed by making changes in the model, which further pointed to some improvements in the binding design, better leveraging the reuse of the constructs being generated off the model in the OWL expression.

Steps for implementing the improved style of OWL in the binding were defined, with the expectation that these would result in a binding which was considerably easier to read and understand by implementers. The degree of effort required would be fairly minor, and will be implemented before the production of the prototype deliverables is finalized.

## Survey of Earlier Work

RDF has been an area of focus for several years, first around the production of specific, DDI-related vocabularies (XKOS and DISCO) and then within the context of the DDI Moving Forward project. A survey of the documents resulting from earlier efforts was conducted, covering both earlier discussions at Dagstuhl workshops and discussions held in other places.

While many of the concerns and decisions made during this earlier work were either rendered irrelevant or reversed by subsequent work, some issues emerged which are still important for the current work. It should be noted that earlier discussions were held in the context of a family of RDF technologies/standards which have themselves evolved over the course of the discussion within DDI. Some earlier decisions were simply no longer relevant, having been responses to limitations within the RDF arena which no longer exist.

Issues were identified during this work, most of which were disposed of:

* There had been a proposal to “flatten” the model to produce an RDF vocabulary with fewer classes in it. This decision was reversed in other discussions during the week.
* The original RDF binding removed abstract classes in the model from the resulting RDF vocabulary. This decision was also reversed in other discussions during the week.
* There was much discussion about (further) organization of the model constructs into a reusable core. Decision taken on this topic last week was that the current organization in the model was sufficient.
* The separation of administrative and “content” metadata was raised as an issue, but subsequent changes to the model rendered this issue irrelevant (this separation is now clear in the model).
* A distinction between persistent entities and the records about those entities was raised as possibly important. The decision was that this distinction was not useful.
* We implemented several design patterns in the model. These are larger than most “micro-patterns” found in other RDF vocabularies. The decision was that our larger patterns were sufficient to our needs.
* Issues regarding the round-tripping of identifiers between bindings had been raised – these issues were addressed by another group during the week.
* Different techniques for expressing similarity/equivalence with objects in other vocabularies were raised – these became part of work in other groups during the week.
* The issue of tracking the evolution of metadata throughout the data lifecycle was raised. The subsequent emergence of the PROV-O vocabulary promises to address this need for DDI (PROV-O did not exist when the original discussion took place).
* How named graphs or other RDF constructions could be used to do “packaging” of data and associated metadata was raised. This remains an open issue.
* The work process moving forward on RDF issues was raised (see below). This issue also remains open.

Specifics of this effort are summarized in a separate document.

**[MAKE REFERENCE TO THE DOCUMENT FROM THE SPRINT HERE]**

## JSON-RDF Binding

The generation of a JSON-RDF binding directly from the RDF-S/OWL was proposed, using Jena or a similar tool. The style of a JSON-LD binding was discussed, and the following example given:

*Here is an example shex schema: <https://github.com/shexSpec/shexTest/blob/master/schemas/_all.shex>*

*translates to this JSON(-LD): <https://github.com/shexSpec/shexTest/blob/master/schemas/_all.shex>*

*If you drop that json-ld into* [*https://json-ld.org/playground/*](https://json-ld.org/playground/) *and look at the [Framed] tab, you see that*

[[

{ "@context": "<http://www.w3.org/ns/shex.jsonld>",

 "type": "Schema",

 "shapes": [

 { "id": "<http://all.example/S1>",

 "type": "Shape",

 "expression": {

 "id": "<http://all.example/S1e>", "type": "EachOf",

 "expressions": [

 { "type": "TripleConstraint", "predicate": "<http://all.example/p1>" },

 { "type": "TripleConstraint", "predicate": "<http://all.example/p2>" },

 { "type": "TripleConstraint", "predicate": "<http://all.example/p3>" } ] },

 "extra": [

 "<http://all.example/p1>",

 "<http://all.example/p2>",

 "<http://all.example/p3>" ] },

]]

*turns into a generically-represented*

 [[

{ "@graph": [

 { "@id": "\_:b0",

 "@type": "<http://www.w3.org/ns/shex#Schema>",

 "<http://www.w3.org/ns/shex#shapes>": [

 { "@id": "<http://all.example/S1>",

 "@type": "<http://www.w3.org/ns/shex#Shape>",

 "<http://www.w3.org/ns/shex#expression>": { "@id": "<http://all.example/S1e>" },

 "<http://www.w3.org/ns/shex#extra>": [

 { "@id": "<http://all.example/p2>" },

 { "@id": "<http://all.example/p3>" },

 { "@id": "<http://all.example/p1>" } ] },

]]

*which is missing the definition for <*[*http://all.example/S1e*](http://all.example/S1e)*>*

*The former is what JSON users expect. They will write code to address e.g. shapes[0].expression.expressions[1] rather than something that shifts underneath then in different serializations.*

*Generating this type of model from the XMI would be possible.*

There were no decisions taken regarding when a JSON-LD serialization might be published, raising this as part of a more general issue concerning non-XML, non-RDF bindings.

## Namespaces

The discussion arrived at the following recommendation:

* Have a common namespace for all DDI classes,
* Have a mechanism to indicate the used classes for a specific view
* Generic namespace without an extension that resolves to a link with the physical ;
* Different bindings of that construct would be the same but with an extension. Make use of content negotiation with filenames, .html, .rdf, etc.
* This namespace scheme should be used for the prototype.

## Using Other Vocabularies and Inclusion in the DDI Model/Bindings

A list of RDF vocabularies which are potentially important to the DDI RDF bindings was provided as part of the agenda. This included the vocabularies mentioned in the following table, along with notes for specific cases.

Vocabularies to consider:

|  |  |
| --- | --- |
| **Vocabulary** | **Notes** |
| RDF-S/OWL |  |
| PROV-O | Restrict high-level PROV-O classes using OWL |
| SKOS/XKOS | For which purposes should these be used/reused. DDI controlled vocabularies are a good match. For classifications and code lists SKOS/XKOS should be useful. Things in our collection patterns may require some new extension – need to look at this. |
| DCAT | Provide a mapping for collections of data . We could extend or use DCAT classes directly. Look at ESTATs application profile for ideas of how to do this. |
| DataCube | Could be mapped against existing DDI NCube structures – also in Moving Forward model? What is the interplay with CSVW? |
| CSVW | Mapping needs exploration; CSVW is a little bit redundant with OWL/RDF-S – may not be useful if the redundant features are the only ones we would need in DDI. |
| PAV | Map at the field level. May need to make some decisions here where redundancies exist within other vocabularies of interest. |
| ORG |  |
| DC-O | OWL restriction classes should be used to create refinements to suit our purposes. |
| DISCO | Not used – will be deprecated when DDI Moving forward RDF is published. |

Alignment/compliance with OBO and SIO was considered. A decision was made to ignore these.

There was a discussion of whether external constructs should be included in the DDI model directly (even if refined) or whether such classes should be substituted into the RDF bindings, based on some configuration information. It is the case that OWL can be used to indicate where classes are similar or equivalent, assuming this information is known at the time the OWL binding is generated.

## Validation of RDF

It was noted that validation in the XML technology stack is different from the idea of validation in the RDF technology arena. For XML, the primary validation artefact is the XML Schema, which also serves to specify the structure of the XML binding. OWL does not perform the validation function in the same way, instead performing the specification function without the capability of validating specific RDF graphs structured according to it.

There is also, in the XML world, a standard “after-the-fact” validation language known as Schematron, which allows both XML schema-type validation, but also supports some more powerful validation by considering the combination of the values of different fields anywhere within the XML instance document.

The RDF community has developed two different standards for validating RDF graphs: ShEx and SHACL . ShEx has validation capabilities similar to those of XML Schema – SHACL validation is more similar to that provided by Schematron. The use of these validation standards is being considered for inclusion in the DDI work outputs for Moving Forward. It was decided that although having validation tools to accompany the RDF binding was a good idea which we will want to explore, it was out of scope for the impending Prototype release.

It was identified that artefacts such as SheX validations could be generated – like the XML Schemas – from the XMI which describes the model. Ideally, such validation tools would be created both for the entire library and for each functional view, just as for the OWL descriptions.

From the notes of this discussion:

***Summary:*** *Validation will not be part of the Prototype, too much work from now. Validation against PSM on basic things, like datatypes, is important. Possible to generate both languages without too much effort. Experiences from FHIR. Need a secondary validation to a subset of the whole DDI for the purposes of functional views, user groups, and tools.*

The idea of including some basic validation (at library level, but not for views) in the prototype was suggested.

## Round-Tripping among Bindings

Many different syntax bindings of the DDI Moving Forward model have been discussed, based on the idea that there will be implementations other than the bindings/products identified by the DDI Alliance as work products. These include the following:

*Bindings:*

* XML
* RDF
* SQL
* Program libraries
	+ Java
	+ CSharp
	+ Python
	+ JavaScript
* JSON (-LD) (one fixed order?) - Notes: (1) generate JSON from the PIM not from RDF, then compare with RDF as a check; (2) RDF to JSON-LD might not be very useful.

*Validation tools:*

* JSG - originally JSON form of ShEx
* JSON schema (widely deployed)
* Java objects via JAXB-2 (from the XML Schema)
* Javascript (a well-trodden path)

When discussing the issue of “round-tripping”, criteria for equivalence must be established, and a mechanism for this must be identified. This was a topic of much discussion, and several approaches were considered (Canonical XML, class definition comparisons, isomorphic graphs, non-isomorphic equivalence, etc.)

The discussion was summarized in the notes from the team meeting :

*The Prototype should both consume and output XML and RDF via an abstract representation. The test cases would make sure that the round trip is possible and provide guidelines for implementers to avoid different dialects for DDI4.*

*Two approaches to using vocabularies in RDF*

* + *At model level*
	+ *In binding process*

*Serialization and parsing require knowledge of equivalence between serialized form and the abstract representation. When using other vocabularies in the RDF bindings we need explicit mappings between model classes and properties and external RDF names*

 *(e.g version ⇔ pav:version). This is a format binding transformation process which exists probably for every format binding.*

* *RDF naming conventions should be: object names initial uppercase, property names initial lowercase*
* *Need clear rules for case in XML bindings*

***Identification:***

 *DDI has used ISO 11179 for IDs. How do we do that in other bindings? In XML the version of an item includes properties and relationships. With RDF a SPARQL query might not return all of the properties and relationships for a given object. Could this be checked for being the same version as another RDF representation?*

*Have a media type for DDI4 RDF triples?*

# Outstanding Issues

Note that some very specific technical issues are now to be found in the JIRA tracker for the RDF Binding. Those issues will be handled by the production framework team. The issues listed here are such that they require discussion among a broader set of individuals within the DDI community.

1. How do we package RDF graphs? Named graphs? What guidance can we provide? What do we put into the specification? We have stringent requirements coming from official statistics regarding the need to publish easily visible “packages” of triples, to guarantee completeness of what is exposed to the public (ie, footnotes on tables regarding estimated values, etc.)
2. Will we produce a JSON-LD binding as envisioned/discussed? When would this be made available? Priorities? How does this relate to other bindings considered?
3. Do we adopt the namespace recommendations? For the prototype?
4. Do external constructs for RDF go into the PIM? Into the bindings only? Do we need a single approach, or do we consider this on a case-by-case basis as we work with specific external vocabularies?
5. What form of RDF validation (eg, ShEx, SHACL) do we include in DDI? How do we do this usefully, to leverage the library, views, and other things which might benefit from having such validation (eg, tools). Do we include anything in the prototype release?
6. Outstanding questions for round-tripping: can we make decisions on the points discussed regarding how to determine equivalence, how to address naming conventions, identifiers, etc. (See above.)

# Process Moving Forward – Requirements

While technical issues resulting from these discussions were entered into the Jira issue-tracking system, the ones listed above will need to be addressed by a broader group of individuals than those who typically handle Jira issues (the production framework team). Such a group does not currently exist as part of the DDI Moving Forward structure.

At the same time, there were many individuals with experience in modelling and implementing RDF at the Dagstuhl sprint.

The group which could usefully be convened to address the issues above should contain the relevant members of the Modeling Team, the TC, the production framework group, the DDI Alliance RDF expert, and invited experts from those present at Dagstuhl.

The process will need to be one which produces at least temporary decisions (that is, decisions which might later be reversed in light of subsequent experience) according to the timeline established for the upcoming prototype. This will means that these decisions – which could impact the model, the production tools (bindings), and deliverables for the prototype – may be need to be made very rapidly. In some cases, where the decisions will not affect what is included in the prototype, the decisions may be delayed until the group has time to do the necessary investigation to support a decision. Thus, a prioritization of these issues will be needed to identify where decisions are needed immediately, or can be postponed.