**DDI Sprint #3
Vancouver, BC, Canada**

**Thursday, March 27, 2014**

**Minutes**

Sprint participants:

Therese Lalor (Project Manager and Sprintmeister)

Jay Greenfield, Arofan Gregory, Larry Hoyle, Jon Johnson, Alan Li, Ron Nakao, Wendy Thomas, Joachim Wackerow, Mary Vardigan

**Working Documents**

Therese set up a space on the UNECE Confluence platform to hold internal documents from the sprint.

**Identification**

A document authored by Dan Smith on the topic of identification in DDI 4 was reviewed. If URIs can have any form how does this affect interoperability? This question will be posed to the author of the document. The Agency ID should have a specific form. A proposal to have a flexible prefix with a fixed suffix for Agency ID, Object ID, Version was put forward.

**Access Control**

This group recommended that the standard XACML (Extensible Access Control Markup Language) be hooked into DDI 4 to cover access and authorization. There is a profile of XACML that covers intellectual property as well. This adds a dimension that is also important, so the group recommends adding some or all of this content.

**Interacting with Other Standards**

How do we interact with other standards? We want to reference these things in our views. This is a modeling issue as DDI interacts with different standards in different ways. Do we need an interface to map to other standards? DISCO references four different standards. What is the mechanism for the references? This is how the Semantic Web works – the very loose structure of the Semantic Web enables this. It can be an advantage or a disadvantage. For the OWL representation there should be a configuration that could be expressed in OWL to say this DDI object is equal to an object in another standard. How is this translated to the XML schema level? The problem on the model level is that you would need to model other standards. Perhaps we could do something similar to what we do with the controlled vocabularies – we have a pointer to vocabulary and reference to an object in another standard. We have no control over the versioning of the other standards, but a way to solve this for loosely defined standards is: in DDI an equal or similar object is defined with mapping and then the external item can change but that doesn’t matter. It would be worth it to explore this and put a proposal forward this week for commenting later.

**Process**

This is a simple model that documents process. The process can apply to the archiving pipeline and you can create additional hooks to allow more plug and play.

* Views reuse components: define once (in the library); use many times
* Views are reusable models that users can pull off the shelf to perform a task
* The models support substitution / instantiation
* The models can plug into one another
	+ For example, the simple data processing model can be used as a step during archiving
	+ Our simple questionnaire model in Drupal plugs into the simple protocol execution model defined here

GSIM has process design and process method plus a rule. Jay mapped the rule into the OWL-S content.

How do you distinguish between a template for a process and an instance process? The model discussed is more of a template but you need to specify the choices that can be made and the choice that was made. Instantiation is not clear right now. If you have a set of processes that were built, you can plug that in as a substitution for this abstract class of data transformation.

What you need from process is an audit trail: what did you feed in and what happened to it? In the model there is a separate lifecycle event for paradata. You reference the template and say I did one of those but then show what happened to the dataset.

We should put this out for public view with a list of open issues.

**A Counter Proposal for Library Organization**

This proposal is an alternative to Jay’s using BFO, organizing the library in an intuitive way instead of the logical way. We want people to be able to find and reuse objects. The audience for this work is the set of content creators who will be reusing objects form the library to compose views.

Proposed packages/buckets/groups of building blocks/shelf in the library/organization in the library in UML:

* Instruments/Data Collection
* Universe, Concept, Category, Unit
* Representations, Codelists, Classifications
* Logical Data Structures (includes Quali)
* Physical Data Structures (includes Quali)
* Datasets (includes Quali)
* Variables, Data Elements, and Measures
* Agents
* Grouping and Comparison
* High level Study Info
* Process and processing
* Study Inception
* Archiving and Preservation
* Technical modeling stuff
* Access, Discovery, and Dissemination
* Geography

We should have a good organization in the library that supports discovery of the right object. In the BFO variables are separated from measures, which is counter-intuitive. However, in the above scheme, if one doesn’t know DDI, this doesn’t help.

It was pointed out that we need to put a semantic layer on top of the modeling, but it doesn’t matter how we name it. The only people interacting with the library are the tech people in this model. There are two groups of users for the library: the modelers and the customizers. This would be the granularity of the versions of objects. These are the release packages.

Organizing the library and supporting the people using things in the library are two tasks. We need to support them to find the right object so they do not invent new objects.

It’s a documentation issue mainly. Is there a way to order these things from different perspectives? These are not good for medical researchers, for example.

This should be indexed in the “Google” way so that people can find things.

We can have a distinction between meta-categories and what is more important – the namespaces. If they are a package in EA, they have a namespace. The organization should come from the modeling. This is a starting point to test against.

We need to add some example objects to each of the categories and add what is missing. A small group needs to get together around this. It needs to be optimized for versioning.

**Drupal Needs**

**Refining Drupal**

The XSD data types in XML are simple data types in UML. When they are output, they need to be wrapped correctly. The pull down list in Drupal (extension bases) should include data types and objects.

Drupal needs a new field to indicate that the object has a related GSIM object. We need the GSIM equivalent and the URL to the GSIM site.

New fields in the package level are needed: example, explanatory notes, status, stage, version.

Another priority is importing 3.2 into Drupal.

It would also be helpful to have tagging in Drupal.

We would tag at the object level in the library. It makes less sense to come up with tags you would use to filter that would leave you in the same place you are in browsing. As long as there are not dependencies on them, you can use them to do the kind of search that you do in Amazon. Each set of tags can be grouped into faceted search.

**Workflows and Packaging**

A new proposal was made for “Terminology and Organization of Work.”

The term package is now being used for something very different than it was before. Then we introduced views replacing some types of packages but not all. Terms have become overloaded and confusing.

We need to agree on how we talk about the groups doing the work, what they are producing, and the things we will publish

We need to document this. We have:

Library = All objects

Packages = UML packages that are the shelves/buckets in the library.

Views = Subsets of objects in the library that perform a function like describing a simple codebook

Teams of people creating content

Views as a work product and sets of object in the library

Teams aligned to Drupal packages might now be called work teams working on work packages. The modelers would create UML packages in Drupal.

We need as list of which teams are producing which things.

A view produces an XML schema and an RDF expression. The library doesn’t get published but is the pool of objects.

We have to use “package” to mean a UML package but we should avoid using package in two ways. “Functional view” is better than just view.

**Report on Architecture**

Packages are hierarchical in EA so we can use hierarchical organization. We can use packages for everything and change the library accordingly. In terms of workflow there are impacts on how things are finalized. We can use packages for library packages and functional views.

A walk through the process:

Technically everything we are managing is packages.

Contents of the Functional View level

Deliverables are:

* A documented model including UML diagrams and narrative
* An EA file and an XMI file

An XML schema with

* XSD files with inline documentation
* HTML documentation – front page about functional view and field level

An OWL ontology with

* RDF expressions
* HTML documentation

Workflow

Step 1: The content team agree the scope of the work:

* The set of objects to be developed OR
* Requirements for a functional view

Step 2: The content team assesses the availability of needed objects

* In 3.2 import
* In library packages
* Needing to be created

Step 3: Needed new objects are created or existing 3.2 objects reviewed and edited

Step 4: Review and integration of new and edited objects by modeling team working in EA

* Objects are given a home in a library package and moved out of the working package in Drupal
* Working package is emptied
* QA is performed and objects are approved
* Drupal and EA are synched

Step 5: The content team creates a draft functional view

* This happens in a new package in Drupal
* All objects are pulled into new functional View package from library packages by reference
* This functionality will need to be added to Drupal

Step 6: Functional view level documentation is drafted and exported from Drupal as Docbook

* It is attached to the Functional View package in the XML

Step 7: Modeler Review of Functional Package

* The functional view package is exported and pulled into EA for review and integration by modelers
* Drupal is synched with EA

Step 8: Functional view model is drafted from diagrams in EA and documentation

Step 9: Functional view model is reviewed and approved

Step 11: XMI is exported from EA and combined with final Docbook

* All automated outputs are produced for review, approval, and publication

**Architecture – Enterprise Architect Model**

Highest level is Model

Workbench Package – In this we import all relevant DDI 3.2 objects (subpackage) and Draft View 1 (objects in simple instrument, for instance) and 2 (functional view with namespaces bringing together references of all objects you want to pull together) from Drupal.

Then modelers starting the work to build the Library. They look at all the things in the Workbench – the building blocks based on extended primitives and primitives. This is now a pool of things that could be used to build Functional Views.

Then Functional Views are built that encompass Final views.

All of this is published except for Workbench.

There is a simple workflow for this: take the list of needed 3.2 objects, transform to XMI, and hand it back to document.

The source for your object is 3.2 at first but then you are also creating new things.

As things are developed they go from EA back to Drupal where they are then available.

In this workflow there are two points at which Drupal and EA need to be synched. This seems complicated. Is there a way to manage more of this this in Drupal instead? As the proposal indicates, right now you have to synch to get packages and synch again to get views.

Modelers work in Drupal and then export that XMI into EA model. You need two environments. There are lag time issues and the round-trip issue.

One requirement at the time you create functional view is that you need to know the namespaces of the objects you are using. They need to be part of the library. The XMI that comes out to produce the syntax artifacts needs the namespaces.

The step forward is that we can create all sorts of packages in EA, which is a new understanding of EA and its functionality.

Regarding the round trip, there was the technical argument that feeding back from EA to Drupal would not work, but now this is resolved. The tool is not yet operational and we have a resource issue. The documentation could be fed back to the content people without synching to Drupal.

The content groups don’t have the skills to sift out what is needed, what needs to be extended, etc. These may be modeler tasks. The modeler should make sure that what is imported is good. Then it’s an integration process. For GSIM, integration was a huge part of the work.

People need to have responsibility for tools in this framework. We need two or three people who are responsible for these tools – to maintain them, etc.

Problems to address: (1) Synching and doing the modeling in Drupal. (2) Incorporation of docbook into schemas and OWL. (3) Tool to build functional view through references. (4) Roundtrip. (5) Production of HTML documentation.

Why do we need to feed back to Drupal? If a mature draft of the content people is built and imported into EA and then modeled and final view is created, this view could be fed back in the form of diagrams and generated documentation. This could be refined. However, the library keeps changing so anyone else doing development in Drupal needs a view of the current library.

**Updates**

**Interacting with other standards**

We will try an example with the newly identified access standard.

**Discovery**

In the Discover view, Coverage now has an aggregation of the three coverage types: topical, spatial, temporal.

**Library and Archive**

This has merged with Access and become Collection Management.