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

**Monday, March 24, 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

**Introduction**

Therese set the stage for the sprint week by reviewing the sprint methodology and process. The process is a flexible one, and the team decides on priorities in collaborative manner.

**Initial Agenda**

Therese reviewed the agenda of items going into the sprint:

* Creating a Kanban board to track the development process and surface dependencies
* Technical: Web Services orientation
* Technical: how we package, version, issue
* Content: New areas -- Library and Archive Management, Authorization and Access Control, Data Management Planning
* Standard query documents and service architecture (this is a lower priority, but we need to address it as it can influence design)
* Production process and production flows and tooling
* Design principles, modeling style, design and architecture – this encompasses extensions and views and how they relate; Discovery and Simple Codebook are views -- how are components they are built from built and organized? (High priority)
* Update on pulling 3.2 on Drupal – it is in spreadsheet format now and needs to be imported
* Identification – there is a document about identification in the world of XML schema and RDF that needs to be reviewed and commented on

It was pointed out that we should not model from scratch but use 3.2 and GSIM when possible. Also, it’s important to test what we have done.

**Architecture – Views and Extensions**

Views are uses of the model not the model itself. At design time it‘s important that all of the objects that are used later are available in the model with meaningful relationships. We need to distinguish between the simple case and extensions. But what is really simple and what is the extension point? That is not clear right now. It doesn’t make sense to add additional properties in the extended case; we need to have these properties in a “library” but restrict the set with a view.

The group split into smaller teams. Afterwards, Arofan reviewed his paper on DDI Design Principles. Insert Design principles document here

Ron showed examples of standalone reusable thematic packages, or building blocks:

1. Instance variables
2. Representation (e.g., codes, categories for variables, enumeration, data types
3. Concepts to be measured/operationalized
4. Questions
5. Development of a questionnaire (objects to put together to validate)
6. Administrative/registry data
7. Interviews/qualitative data
8. Observational data
9. Organizations
10. Individuals
11. Coverage

Are these too granular? GSIM has 4 packages and views on the model, but some consider this two few.

We have separated out representation because it is used in so many situations. Some of these things we are talking about are abstract types and there is some dissatisfaction with how they are represented currently. Instance variables have a high dependency on other types of variables.

We need an automated way of bringing out documentation with a view. We can have a small set of common usage views, and then people might build other views.

The content groups are working on standard views and anyone who wants extensions and their own views could use our same packages and tools.

An important principle is that we reuse objects that are in 3.2 and not start over to remodel when we already have items that are modeled well.

It was pointed out that the notion of the data lifecycle is good, but the problem was that the lifecycle assumed that you would do things in a certain order, and this is not the case.

The RDF Disco vocabulary is a view – what are the objects and relationships that went into that view? We need to decompose this. What are the biggest circles in the model drawing?

Instrument is an abstract type in a way, but we are currently missing content in DDI for other types of instruments. We need to make room for that in our model. What are non-questionnaire items and what do they look like? Likewise variable means it’s a certain type of data.

There could be several thematic things that we put together in a single package but don’t have to be used together because that is the role of the view.

In deciding what goes into a package, we need to look at the probabilities of change in those items. Building blocks should not be candidates for change in the near future – is this a possible design rule?

Perhaps we should think about using the term “building blocks” rather than “thematic package”. “Metadata management packages” is another possibility for a term.

The team looking at how to import DDI 3.2 into Drupal reported out. They had identified for every complex and simple object whether or not it was to be kept because performed a needed function. They used the key: K = Keep; R= review (questionable structure issues), including physical data product contents, representations and response domains; T= Toss, including schemes, groups and some of the major packaging structures (high level grouping, resource packages, local holdings package, logical product, data collection).

About a third were designated as Keep and there was a small number of Toss items. The rest are for review. 3.2 will be imported into a sandbox for now, and what is reviewed will move to another space.