Analysis: Functional Views in DDI 4

# Overview

This paper outlines the purpose and implementation of Functional Views as they currently exist in the Protype Review package. It then analyzes the central issues encountered in the implementation as it stands today, and describes a possible approach to realizing Functional Views in a useful way in the DDI 4 Core.

# The Original Idea

One of the original design goals of DDI4 was to produce “functional views” – subsets of the whole model with separate schemas designed for some specific purpose. Examples would be a statistical classification, sampling, and a codebook. The intention was to make DDI 4 simpler both for those learning about the model, and also for those directly implementing it (e.g., having a smaller XML Schema for implementation). This document explores the status of the implementation of functional views as of the prototype release of DDI4 in early 2019.

## Axioms

First there are a few axioms or facts about DDI4 that are relevant to functional views:

**The DDI4 Model:**

* DDI Lifecycle introduced reuse by reference, which DDI4 continues
* DDI4 classes have globally unique identifiers that allow them to be reused
* There are many connections among classes
	+ Many classes have multiple associations
* Viewed as a network the whole model does not have a lot of locally connected “neighborhoods.” There are not obvious functional views based on the association structure.

**Functional Views:**

* A functional view is a subset of the classes in the model
* Any such subset is likely to have multiple associations outside of the subset
* Bindings based on functional views will not be interoperable if they exclude the possibility of specific external associations or properties
* If functional views are to be interoperable classes must not change across functional views

## Implications of the Above Axioms

There are several implications of the axioms noted above, which limit what can be realistically implemented for the DDI 4 Core.

Classes in functional views are not modified: Associations to “external” classes are by DDI URN. These values must therefore be preserved by applications.

For almost any subset of the classes in the model (a functional view), there will be classes with associations to classes outside of the view. These references are represented by a DDI URN pointing to an object that is not from a class in the model. For an application designed around the functional view there will be no access to information about the properties or associations of that “external” object.

A functional view expressed in a syntax representation should allow for dealing with “external” associations – information should not be lost when moving among functional views.

As noted, if functional views are to be interoperable, a binding should not change any of the properties or associations of the classes it contains. The DDI URN for an “external” reference must be preserved.

A functional view is primarily for the purpose of signifying a set of classes important to some type of use of the model. This is a documentary purpose.

All of this leads to the conclusion that the original idea of creating simpler, independent sub-models of DDI4 was more a little more ambitious than achievable. This is not to say that functional views might serve no purpose. There is still utility in presenting a set of classes that function together to serve some purpose. Documentation and applications may still be written around the use of that subset of classes.

# Computing Views from the Association Structure

As an alternative to a more conceptual basis for defining functional views we have been using, it may be possible to use network analysis tools to identify sets of model classes that are more densely interconnected than the overall level of interconnection. These might be the basis for defining additional functional views or even packages. Functional views designed this way would have fewer “external” associations, and therefore tend to be more independent of other views.

# Validation

Applications dealing with functional views might typically need to validate instances of DDI 4. There are three levels of validation that might be done.

* Schema level validation can be done with a functional view schema
	+ Validation at this level can check that properties and associations of the objects of classes in the schema fit the rules.
* “Secondary” validation allows noting a reference to an external class via a DDI URN
	+ A DDI URN could be checked to see that it is well formed
* “Tertiary” validation requires the whole model and could potentially involve retrieving external DDI objects.
	+ Validation at this level could attempt to retrieve a referenced object and validate it against the schema for the whole model
	+ Alternatively, a manual kind of tertiary validation could be to confirm that the DDI URN is for a “well known” shared object. Suppose, for example that there was a published Likert scale CodeList. Validation would only need to confirm that the DDI URN was correct and would not need to validate the published object.

It should be noted that many technologies exist for validating instances in various syntaxes in both XML and RDF, before and after the fact. While XSD schemas are provided by the DDI Alliance for XML validation, other standards also exist. The two best-known are probably Schematron (now an ISO standard) and OASIS CAM. In the RDF world, ShEx and SHACL are the best-known ones. These different standards provide many different approaches for implementing validators at the levels described above.

# Implementation of Functional Views

In the current DDI4 model, functional views are described in the Lion Repository as a list of classes. Discussions are ongoing about other possible representations that model a functional view in the UML as a class of some sort. One alternative might be to model a functional view as an aggregation. We might also be explore modeling it as a structured collection, which might allow for specification of entry points.

These approaches will need to be further investigated, and will have an impact on the Modeling Guidelines as well as the DDI 4 Core deliverables (the model and syntax representations and their documentation).