The DDI 4 Model – Binding and Intermediate Models

*Minneapolis DDI Sprint, May 2015*

# I. Overview

The Production Framework team has identified the possibility of separating the process of binding the model into two stages: the Platform Independent Model (PIM), which is what we have in Drupal, and the Platform-Specific Model (PSM), which would be a form of the model which has been adjusted to suit the expressive capabilities of the target technology binding. It is the latter model which would be transformed into the target platform (XML Schema, etc.)

This document takes one problematical part of the DDI 4 model when it comes to binding, and uses it as an example to explore this approach.

# II. The PIM

The diagram below shows how an “interface” (the Collections pattern) would be expressed in the PIM in the context of a Concept System.



Note that it would be possible to have the OrderRelations be turned into ComplexDataTypes if this was desireable (it could produce a more terse XML).

# III. The PSM

The same model would be collapsed to hide much of the chain of inheritance, by removing all abstract classes from the XML schema. In such a case, the PSM would appear as below.



From this, using current binding rules, the XML format could be derived.

# IV. The XML

## Simple list of Concepts

Consider a list of three concepts: age, gender, and language.

### XML Option 1: Verbose ordering

The first example shows what the XML binding could like if it directly mirrors the platform specific model.

<ConceptSystem>

 <ConceptReference ref="ageConcept" />

 <ConceptReference ref="genderConcept" />

 <ConceptReference ref="languageConcept" />

 <Sequence>

 <Precedes ref="ageConcept" />

 <Succeeds ref="genderConcept" />

 </Sequence>

 <Sequence>

 <Precedes ref="genderConcept" />

 <Succeeds ref="languageConcept" />

 </Sequence>

</ConceptSystem>

<Concept id="ageConcept" />

<Concept id="genderConcept" />

<Concept id="languageConcept" />

### XML Option 2: Simple ordering

In the XML binding, we could use XML's ordering to represent the order of items in the collection.

<ConceptSystem>

 <ConceptReference ref="ageConcept" />

 <ConceptReference ref="genderConcept" />

 <ConceptReference ref="languageConcept" />

</ConceptSystem>

<Concept id="ageConcept" />

<Concept id="genderConcept" />

<Concept id="languageConcept" />

## Hierarchical Concepts

Consider three concepts: color, red, and blue. Red and blue are subtype or children of color.

### XML

For a concept hierarchy, we explicitly state the relationship of one concept to another.

<ConceptSystem>

 <ConceptReference ref="colorConcept" />

 <ConceptReference ref="redConcept" />

 <ConceptReference ref="blueConcept" />

 <ParentChild>

 <Parent ref="colorConcept" />

 <Child ref="redConcept" />

 </ParentChild>

 <ParentChild>

 <Parent ref="colorConcept" />

 <Child ref="blueConcept" />

 </ParentChild>

</ConceptSystem>

<Concept id="colorConcept" />

<Concept id="redConcept" />

<Concept id="blueConcept" />