What is DDI 4 and what is its state?

Oliver Hopt, Wendy Thomas

This document is covering the intend of DDI 4 to clarify when a certain state of “ready” would be achieved.

As a major shift in the way in which a DDI product is developed, the DDI 4 effort has unfolded over several years in a series of sprints and online meetings. As noted, the effort involves developing not only the information model, but also the infrastructure for building that model, the transformation into a set of representations (initially XML schema and RDF/OWL), and the associated documentation.

# Technical

The main intent of DDI 4 was to move from a fixed implementation of the model in XML to a purely conceptual model defined in UML as a class diagram. UML class diagrams are a better representation of DDI compliant metadata, which has more of a graph character then the tree structure common to XML. The graph structure also makes several management layers, necessary to build a proper XML tree, obsolete. Therefore, the assumption is that it will be easier for users of DDI to understand the usage. So far, this is not proven.

At the same time, it is easier to extend a model structured as a UML class diagram than one modeled in a specific binding such as XML. Extensions include both those from broadening the standard from inside and the completion of the standard with additional things from users. One central point concerning this extensibility is, that in a tree, a broadening of elements would imply a reorganization of entities connection to the root. Seen as a graph, there will be just more relations.

From the DDI 4 UML that DDI 4 definition, all projected platform bindings should be generated automatically. To achieve a definition of DDI on a pure conceptual level, several steps have to be taken.

## Rules

A full usage of all UML features from class diagrams, the issues with implementing what comes from XML would be just shifted into UML. Therefore those elements were identified that are needed and would be applicable within all projected bindings. As the functional set of class diagrams is more or less applicable to the most high level programming languages, the set of rules is driven by the restrictions to meet XML and RDF.

The rules can be seen as completed as there is already a bunch of content implemented and no additional features are needed. They are to be found in <https://ddi-alliance.atlassian.net/wiki/display/DDI4/UML+Constructs>.

## Bindings

Planning to support multiple bindings of DDI allow us to take advantages of changes in the languages used to manage, preserve, and disseminate data and metadata. While XML is still used for many preservation and documentation purposes, RDF has been extensively used for web discovery and access systems. Initially the intent is to provide bindings in XML and RDF. Discussions regarding additional bindings are ongoing and the use of the UML class diagrams allows us to programmatically define additional bindings as needed.

The advantages of producing these binding in an automated process are, that:

1. Equal structures in the model will always be implemented in the same way in XML and RDF to improve consistency.
2. Changes can be checked really quick if they work in both bindings without the need to step away from the system where the documentation is written.
3. At the same time, any discrepancy between the documentation and the bindings is barred to improve quality control.
4. The general information flow between bindings can alway go through the conceptual model, so that specifics of certain bindings don’t have to be cleared for for each future pair of bindings.

### XML

The transformation from UML to XML schema is in a quite good shape. The current schema files produced are valid and they are able to validate an XML that is structured in the way that was projected in the following document: <https://bitbucket.org/ddi-alliance/ddi-views-production/raw/2896b40b68d2506dcfc7d7bb711a3fe4cfefcc20/transform/XML%20Binding%20from%20the%20DDI%20Model.docx>.

### RDF

The current state of the RDF binding is a bit unclear. The RDF/OWL document that is produces by the transformation is not valid. There are also some issues raised in a Dagstuhl workshop by Linked Data experts, that have to be resolved. A document outlining the role of an RDF consultant has been prepared. The procurement process for an RDF consultant is underway.

### Into the dark

There already is an initial approach for binding the model into programming languages. This is not officially driven by the DDI Alliance. Some of these might be officially supported in the future.

Additional official binding might come up in future to, as XML and RDF might not be the end of the development concerning exchange and publication of metadata. UML class diagramm on the other hand have proven to be very stable along with various paradigm changes in information technology.

## Production flow

The production flow was set up in Atlassian Cloud, but this has been discontinued in general. The replacement will be on Google Cloud. This setup already contains the productive environment for lion, the Drupal used for managing the model.

For executing actual builds, an installation of Apache Jenkins is used, which is running so far. The first task in Jenkins does not work yet but it is “just” a matter of configuration to perform the transformations to both bindings and to the input for the Sphinx based production of the documentation output.

The documentation of the production flow setup is cept up to date under the following link:

<https://ddi-alliance.atlassian.net/wiki/display/TOOLAD/Google+Cloud+Platform>.

# Conceptual

Besides the decision and solution of the way in which to model the next generation DDI, the setup for how to capture the model conceptually is settled.

## Definition (lion)

Close to the technical solution for the production, the entities are captured within a Drupal System extended with some functionality to draw UMLish graphs and export UML as an XML representation.

This Drupal has currently been moved from a server hosted by SND to the Google cloud.

## Abstract classes

Using abstract type definitions in XML schema is somehow different to the usage of abstract classes in class diagrams and programming languages. Therefore abstract classes are used within the model without bringing them into the XML binding. There their properties and relations are injected into the classes derived from the abstract classes.

## Patterns

Patterns are used as an abstract prototype model for those things in the actual model, that are semantically different but structurally similar. Therefore a pattern contains abstract classes that define what properties and associations certain things should have to meet this pattern. For associations, a pattern usually forms a closed graph.

To make use of a pattern for a concrete set of classes, they contain associations to the appropriate class in the pattern, which is name “realizes”. Within the current bindings, this information is not used neither is some kind of validation of consistent pattern usage performed.

# Content

## Patterns

## Packages

## Controlled vocabularies

## What is covered

### DDI 2.5

### DDI 3.x