Structured Custom Metadata:

How Do We Include External Metadata in the DDI?

*Arofan Gregory*

*26 March 2015*

# I. Overview

This document attempts to describe the many places in DDI where, during the Moving Forward process, a need for the inclusion of external metadata within DDI constructs has arisen. Further, it explores some of the approaches for structuring and including this metadata, and highlights the strengths and weaknesses of several approaches.

DDI has always had the capability for including custom attributes: by using extensions to XML schema; by using specific customized name-value pairs which appear in some places in DDI-Lifecycle, etc. These features have been used extensively in actual DDI implementations, as there is often a need for this functionality.

This paper, however, addresses this question on a perhaps higher level. The use of this type of functionality, as being discussed for DDI 4, places it in a more prominent position, especially as regards the discussion around Qualitative data, previously unknown data types, the “datum-based” approach (useful for event history data), and in the discussions of data and metadata citation.

# II. Custom Attributes in DDI- Lifecycle Today

There are several places where DDI-Lifecycle today allows for the inclusion of custom attributes today. The constructs discussed here may not be an exhaustive list, but they serve to illustrate the extent to which this capability already exists.

## A. Controlled Vocabularies/External Codelists

Many places in DDI – for both Lifecycle and Codebook – allow for the use of external controlled vocabularies. This is done, in DDI 3.\*, using the CodeValueType, which is heavily used throughout the standard. This allows for the referencing of an external list, and an inclusion of a value from that list. It assumes that any validation is external to the DDI. It does not require the use of an external list, however: DDI elements of this type can take any simple string value if that is what the user wants. Further, any information about the structure of the external list (could be a flat list, a hierarchy, etc.) is unknown and unspecified. It is assumed that the user’s system will handle any such information based on just the reference to the external list.

## B. User Attribute Pairs

Another common way of adding custom attributes to DDI is to use the UserAttributePair element. Here, we have two things provided: an Attribute Key and an Attribute Value. The key here is actually of CodeValueType, and so can point to some external structure such as a CV, which is recommended practice.

This structure shares the limitations of CodeValueType, as all structuring and validation of the supplied value is expected to be external to the DDI itself. Further, the use of this construct is always optional – there is no way that a specific type of custom attribute can be required for a specific DDI construct within the DDI itself.

## C. XML Schema Extensions

DDI-Lifecycle has always been an XML standard, and has a documented use of the capability of XML Schema to provide extensions. This feature of DDI has been heavily used, as it works well inside of XML processing systems (being a use of generic XML Schema functionality). Unlike the earlier approaches described, however, it is possible using this technique to describe much more complex structures. To do this, you simply add one or more extended elements to a DDI type and then describe the structure of the new element(s) using additional XML Schemas. Thus, any type of XML structure can be added to the DDI.

Additionally – and unlike earlier approaches mentioned – this technique allows you to specify whether a custom element is optional or mandatory, and it allows the custom structures to be validated using the XML parser which is validating the DDI.

Thus, we have a more complex but more powerful way of creating customized metadata within DDI. However, for DDI 4 this will not be a useful approach, as the model is no longer canonically described by XML Schema, but is model-based. Thus, this extension methodology would be much less useful, and should probably not be included as part of the DDI standard. (Specific implementations of DDI 4 could use this generic XML functionality, but it would not be the standard DDI extension methodology as it is for DDI – Lifecycle today.)

## D. Other Customizable Elements and Types in DDI

There are several other places in DDI where it is possible to provide custom attributes. One such place is in the DataItem construct in Physical Data Product: Proprietary. Here, we again see the use of the name-value pair approach found elsewhere in DDI as already described. Another place where external XML can be included using extensions is when attaching processing code to (CommandCode). Here, external XML schemas for things like MathML can be included.

It is not our purpose to enumerate every place where customized attributes or metadata can be included in the DDI, but to show how this has been done historically. The techniques already described are sufficient for that purpose.

# III. Where Structured Customized Metadata Would be Useful in DDI 4

There are several places within the emerging DDI 4 model where the inclusion of customized metadata has been considered. The model is constantly evolving, so it is not easy to describe specific objects which are impacted. The general areas and uses of this functionality, however, can be described. They fall into several areas:

(1) Describing qualitative data

(2) Describing unanticipated quantitative data structures

(3) Structuring citations

(4) Describing datum-based quantitative data sets (event history data, clinical records, etc.)

(5) For controlled vocabularies and minor DDI extensions as for DDI – Lifecycle

Each of these areas is described in more detail below.

## A. Qualitative Data

In earlier work on qualitative data description in DDI Moving Forward, the structures for attaching metadata were based on a set of known types of qualitative data structures (images, video, audio, texts, relational databases). Because qualitative research can be done on a huge variety of data types, this approach was questioned. If we have the ability to describe a previously unknown type of data, and attach metadata to it, then the DDI’s ability to describe qualitative research would be much enhanced.

## B. Unanticipated Quantitative Data Structures

Recent thinking here has largely arisen from the discussion around qualitative data. If we could develop the capability to describe new qualitative data types, then why should that capability not extend to quantitative data of previously unknown types? Proposals have been put forward for how this might be accomplished, notably by Larry Hoyle at the NADDI conference at the University of Kansas. This proposal has fed into the general discussion around this issue within DDI 4.

## C. Structuring Citations

In the discussion at the DDI 4 sprint at Dagstuhl in 2014, data citation was a major subject of discussion. What emerged was that, in some types of citations, custom attributes would be very useful. In the current DDI 4 model, almost everything is potentially citable, and some of the citation information will apply broadly to all citations. However, it is not easy to determine which specific attributes apply to the citation of which specific object – this problem may not be possible to solve in advance. Therefore, it would be very useful to be able to describe custom citation fields for the purposes of citation.

This is an area where the use of citation is evolving in the real world: data sets are not the only citable metadata. Because we do not know where the practice of citation will lead, having a very flexible mechanism for extending citations is desirable.

## D. Describing Datum-Based Data Structures

In recent discussions within the DDI 4 Moving Forward process, and in some recent implementations (notably the RAIRD project in Norway), a new type of data description has been proposed. This is termed the “datum-based” data structure, and it has many antecedents in the real world.

One of these is event history data (aka “spell data”). Another of these is clinical record data. For describing the latter, one standard which has been brought forward for consideration is the Open Electronic Health Records standard (Open EHR). Open HER has a very interesting approach to data description, because of the requirements of tis type of application. Very often, a single measurement will have a large amount of metadata associated with specifically that data point: this is known in Opoen EHR as the “context” of the data. To provide an example, if I am recording the blood pressure of a patient, there are many things I want to know about that measurement: was the patient standing or prone? Were they performing some activity? Etc. For each different type of measurement, however, there are different descriptors for the context, leading to a need for custom attributes.

In the datum-based approach to describing data structures, we see a similar case: each datum has a set of metadata which is attached to it, and in many cases, depending on the data, it would be useful to be able to describe additional, custom metadata fields.

## E. Describing Controlled Vocabularies and Minor Extensions

Currently, the DDI 4 model is using a technique very similar to earlier versions of DDI – Lifecycle for expressing controlled vocabularies, as described above: external name-value pairs, which can be taken from an external structure of some type (a codelist or classification). The proposed mechanism for other extensions is to allow sophisticated users to take the DDI model and to extend it on the level of the model itself, using the same tools as the Moving Forward modelers themselves. This would allow for the creation of extended XML schemas and RDF Vocabularies for customized user-defined Views, using the same syntax bindings and production tools as used by DDI 4 developers. This is potentially a very complicated activity, but it is also very powerful – more powerful, in fact, than using the XML Schema extension mechanism employed by DDI 3.\*.

One issue here is whether we wish to have multiple ways of achieving the same goal, and whether the existing name-value pair approach is sufficient, or optimal. A second issue is whether the complicated “model it yourself from our library” approach is generally feasible, and whether it would produce interoperable metadata sets.

# IV. Approaches to Structuring and Including Custom Metadata

There are several approaches for meeting some or all of the requirements we have discussed in the preceding section. We must consider several aspects of a solution when looking at each case: can the custom values supplied be validated? Can the structures be complex? Can cardinalities be expressed? Do the custom values have a meaningful semantic attached to them? There are many questions here.

When considering approaches to solving these problems, we are not dealing so much in absolutes, most of the time, but in optimizations. This paper attempts to categorize the approaches to solving the issues around custom metadata, according to the following understanding of requirements. These use the descriptions of the different cases given in the preceding section as their basis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Validation Needed?** | **Type of Structure** | **Cardinalities Needed?** | **Comments** |
| Qualitative Data | Yes | Complex | Yes |  |
| Unanticipated Quantitative Data | Yes | Complex | Yes |  |
| Structuring Citations | Would be nice | Potentially complex | Would be nice | It would be possible to solve this case using name-value pairs, but that is probably not optimal. |
| Datum-Based Data Structures | Yes | Complex | Yes | Again, this could be handled by name-value pairs but that is clearly not optimal. |
| CVs and Other Name-Value Pairs (DDI 3.\*) | No – can be external | Simple  | No | Existing practice does not require much for validation, etc., but it might be useful in some cases |
| Extension Mechanism | Yes | Complex | Yes | The use of XML Schema extensions and DDI 4 model extensions are both very complex to implement, and are mostly for sophisticated users. |

This is a very simplified view of the issues, but it gives us a basis for understanding at least some of the requirements for any proposed solution to this issue. Below, we consider some possible proposed approaches for handling custom attributes and values in model-based DDI 4.

## A. Name-Value Pair/CV Approach

One alternative is to continue doing what we have been doing in the past: using completely external structures which are referenced, and validated independent of the validation of DDI. This fits well into DDI 4 for many cases. For RDF syntax implementation, we do not have any inherent validation in any case: a value is what it is, if it exists. There is no standard validation mechanism provided by RDF technologies at this time. For an RDF expression of the model, then, we lose nothing by choosing this simple customization mechanism (although this is an issue for RDF technologies generally). For XML implementation, this approach – as for DDI 3.\* - external validation, using structural information which is not available through the DDI. It could be argued that this is sufficient for many of the cases in DDI, but would not be optimal for describing qualitative or new types of quantitative data.

It is possible to infer structures within a controlled vocabulary: many existing hierarchical classifications can be expressed using flat lists, with a specified number of digits indicating the level of a value within the hierarchy. DDI today does not contain a description of that level information for completely external classifications, however (that is, classifications or codelists not described in DDI). Such information for describing hierarchical structures (as well as flat codelists) does exist in the DDI 4 model, but we have not historically used DDI code lists or classifications to provide the values for name-value pairs.

We could, however, add a construct for describing controlled vocabularies within the DDI model, recognizing that a CV used for “plumbing” purposes is not necessarily the application of statistical concepts. Our model should show this distinction, since the management of CVs and the management of statistical classifications and codelists are typically separate activities within an organization using DDI.

If we added such a model of CVs to DDI, we could then provide at least some validation of custom values without reference to some externally described structure. It would at least be possible to describe flat lists and hierarchical structures, and potentially even graphs, depending on how complex we wish to make the CV model.

The picture below illustrates this approach: we have a native DDI object describing a CV (right hand side) and some other DDI document making a reference to a node within that CV, so that an application can validate that the value being provided is correct.

This approach comes with the cost of requiring CVs to be described as DDI, which may be an additional burden on implementers. The native DDI description could, of course, be made optional, but this only lessens, and does not remove the additional complexity of implementation (the application needs to check to see if the native DDI description exists before giving up and relying on an external structure of some unspecified type).



There is still no easy way in this picture to determine the cardinality of the supplied value (is it required? Optional? Etc.) Any mechanism for doing this would be ungainly: if it is supplied from the point of reference, how does the application know, in the absence of a value, that one is required? If we specify this as a property of the CV, then we may be making our CV non-reusable. It is best to keep our CV and the *use of the CV* separate, but this requires a structural description of the construct which is using the CV, independent of the description of the CV itself.

We can see that the use of a name-value pair approach could be very similar to what we have today in DDI 3.\*, or we could provide some additional level of native DDI description of a CVs structure. This does not meet all the requirements for all of our use cases, but presents some options for some of our use cases. It also has the benefit of being relatively simple from an implementation standpoint.

An initial attempt at modeling the DDI description of a CV is provided here. It is a simple implementation of the Collections pattern already being used within the DDI 4 model.



Collection, OrderRelation, and Member are the abstract classes found within the model today. We introduce a type of Collection – the Controlled Vocabulary – and a type of Member – the Vocabulary Entry. Additionally, we have two OrderRelations: a Parent-Child relationship and a Sequence relationship.

This structure is simpler than the use of Categories and Codes, and is sufficient to support validation of CVs. We recommend that its use be entirely optional, however, so as not to place additional burden on DDI implementers.

## B. Meta-Model Approach

A second way of meeting the requirements described above is to have a metamodel within the DDI 4 model for describing customized metadata. This approach is used by SDMX in its description of Reference Metadata Structures and Reports, and a similar construct was introduced into GSIM in version 1.1 (Referential Metadata). This aspect of SDMX has been heavily implemented, and can be used to describe virtually any metadata set structured as a flat list or a hierarchy. It can be used for validation purposes, as it supplies cardinality and data type information as well as structural information. SDMX Reference Metadata Reports can be thus validated against Reference Metadata Structures.

Because this approach is less familiar to most members of the DDI community, I will describe this approach in more detail than was given for the use of name-value pairs, above.

When we introduce a meta-model for customized metadata within the DDI model, we are providing a description of the metadata in a standard way, native to DDI itself. To give an example, if I want to describe a specific type of qualitative data, I might have an object called QualitativeDataStructure. This object would give a standard, known mechanism for describing qualitative data sets sufficient for the purposes of describing them elsewhere in the DDI.

If I want to address a segment within a qualitative object, I would explain how to do that within my structure description:

Qualitative Data Structure (Type = “Interview Transcript”)

 Field: Title (1..1)

 Field: Paragraph (0..n)

 Field: Speaker (1..1)

 Field: Line (0..n) – data type = “string”; locator type = “range”

Here, indentation indicates containership: an Interview Transcript consists of a Title followed by one or more paragraphs. Each of these contains a Speaker field plus one or more lines, each of which is made up of one or more characters.

If I provide identifiers for each of the fields in the instance of my Interview Transcript, then I can attach metadata to each of these objects (or potentially a range of objects, as for paragraphs, lines, and characters).

Now that I understand the structure of an Interview Transcript, I can now describe the instance of the transcript in a separate part of the DDI. I could have the following set of objects in my model:

Qualitative Data Instance (type=”Interview Transcript”)

 Qualitative Value: Title = “Interview with Achim Wackerow, 2 June 2015”

 Qualitative Value: Paragrapgh (ID = “P1”)

 Qualitative Value: Speaker (Value = “Interviewer”)

 Qualitative Value: Line (ID=”L1”)

 Content: “When I was seven I lived in…”

We know from the type of the transcript which Qualitative Data Structure to use when validating our instance: the one describing a Qualitative Structure. We also know which Qualitative Values to expect, as these correspond to the Fields described in the Qualitative Data Structure. There would then be a third component in this example, which would be annotations to attach metadata for the purposes of qualitative analysis. These would be values attached to the qualitative instances by making references to the segments: for this type of data, the structural description could contain information about how the segments are addressed, as well. This part of our example is specific to the Qualitative Case, however, so we will leave it out for now. (Think QuDEX as a model for attaching metadata – I assume we will have something similar in DDI 4).

There are some issues with using a metamodel approach: this is a fairly complex way of describing things. Further, it requires applications to support the metamodel, rather than a straightforward model as for the name-value pair approach. Essentially, the application constructs the model and then implements it at run time. This is not always hard to do, but it is not the way most developers will implement systems. They tend to expect a model.

This is, however, a way of making a very broad set of descriptions interoperable: a system which understands the metamodel can handle \*any\* instance of it. And it is possible to create applications which only understand a single instance of the metamodel, and process only those data instances which are structured according to it.

Thus, I could have an application which understood any type of Qualitative Data Structure, or I could have an application which only understood Interview Transcripts. The second type of application would be easier to implement, but also more limited in capability.

If we can follow the SDMX case, we see both approaches being used, with a small number of “standard” instances of the meta-model being used. (Eurostat’s ESMS is the best example of a standard instance of the SDMX Reference Metadata Structure, which is used throughout Europe and in Mexico.)

An initial attempt at modeling this approach is shown below, both for the metamodel and for the instance of the metamodel which would be found in a DDI XML instance.

The metamodel structure provides a way of describing a list, a hierarchy, or a graph. It is very similar to the CV model, but adds an additional OrderRelation to provide RDF-like links between classes.



For the model of the instance corresponding to any given structure, we have a very similar set of objects. Obviously, the instance of the custom metadata the CMReport) would need to reference the custom metadata structure according to which it was constructed, for validation and processing purposes.



## C. Considerations

There is no requirement that we choose one way of solving each oif our use cases as presented in the table above. We might find that for some purposes a name-value approach is best, and for others we want to use a meta-model approach. Further, we have some choices to make about whether we have a single meta-model which gets reused or whether this functions in the model as a design pattern with different implementations.

Each use case should be decided on its merits. However, the more consistent and clear we are about when and where which approach is used, the better.

Note that the use of the library to produce custom views has not been described here, but can be found in the documentation from the Vancouver Sprint, 2014.

# V. A Proposal for Types of Data Description in DDI 4

The following proposal is presented after discussions with various of the working groups within DDI 4, and some implementers (notably Ornulf Rinses, who was the leading engineer for the RAIRD project). It is offered as the basis for further discussion, and as a means of identifying those within the Moving Forward project who need to be party to the discussion. This is one issue which is very cross-cutting in terms of its impact on the modeling work groups.

The idea is to provide, within DDI, several useful models for describing data, and to identify a strategy for implementing the different approaches for customization described in the preceding section.

To maximize the utility of DDI, it needs to offer descriptions of data which are aligned with common usage. In the past, it has provided two standard ways of describing data: as rectangular data files (similar to the way relational systems and analysis packages such as SPSS, Stata, R, and SAS structure data), and as dimensional structures (NCubes). Because NCubes are derived from rectangular input files, there is a way to link from one data description to the other (NCubes can reference the variables in rectangular structures). Both of these types of data structure description are useful – the expression in both of these by GSIM reinforces this point.

For DDI 4, we may wish to add some other types of standard data description. There is an idea that a “datum-based” form of data description could be very useful, as found in patient record databases and in many different systems working with event history/spell data. It should be pointed out that – as in the RAIRD project – this type of data structure is often mined to produce rectangular data files needed for analysis, and that these two types of data structure description should allow for expressing this link.

Additionally, we have proposals for describing qualitative data sets and other unanticipated quantitative data sets. It would be left to the implementer to decide which type of data description was best suited to a particular implementation, but all of these should be modeled in such a way that a data set description could be cast from one form to another easily within DDI.

This leaves us with five different styles of data description:

(1) Rectangular data files

(2) Dimensional data description (NCubes)

(3) Datum-based data structures

(4) Qualitative data structures

(5) Unanticipated quantitative data structures

For the first three, we propose that explicit models be created in DDI, using a data customization approach based on the name-value pair or meta-model techniques described above for describing the context of datum-based observations, and for customizing in other needed areas in rectangular data structures and NCubes.

For qualitative data structures and unanticipated quantitative data structures, a meta-model approach would be used, using the meta-model itself as a design pattern to be implemented in specific ways for each type. For citational metadata and other uses of custom metadata, either the name-value pair approach or a meta-model approach would be adopted, based on an analysis of requirements.