Usability over reusability

The problem with over-use of Identifiable and similar issues

Author: Ørnulf Risnes, 27 May 2016 (DDI-Sprint Knutholmen, Norway)

See also [DMT-80](https://ddi-alliance.atlassian.net/browse/DMT-80).

# Background

Through the production of Codebook instances in YAML during the sprint, it became evident that an extremely high number of elements are AnnotatedIdentifiable. That problem is a *subset* of the problem I’m writing about here, and the Annotation-related problem is documented in Issue DMT-77 (<https://ddi-alliance.atlassian.net/projects/DMT/issues/DMT-77>).

But the problem I want to talk about here goes beyond annotations and is about the fact that identifiable is overused (IMO) and the consequences this has for programming with DDI and comprehensibility of DDI - and therefore also for the adoption of DDI.

The conclusion of this article is that the identification-requirement should be relaxed substantially, so that it doesn’t get in the way of documenting data for those who can’t or don’t need the identification regime in DDI. This goes into the general discussion of “usability VS reusability” and is strongly in favor of the “usability”-end of the spectrum. To me, having usable, machine-*readable* metadata with a strong conceptual model behind it, is what NSD (and CESSDA) needs for the time being and for many years to come. Machine-*actionable* metadata would be a noble goal, but very difficult to pull off, and not a priority at this point. Machine readable metadata can be read and understood, and to a large extent drive processes, albeit not to the extent envisioned in the “machine actionable” visions laid out for DDI 3.

This text contains examples of how DDI could work if the identity requirement is relaxed.

# Compatibility with Moving Forward vision

I firmly believe that the reasoning in this document is compatible and aligned with the goals stated when the Moving Forward process was initiated[[1]](#footnote-1):

*Goals for the new model-based specification include:*

* *Extend data life cycle coverage*
* *Broaden the focus to new research domains*
* *Develop a robust and persistent conceptual model, with extension possibilities and implementations in different technologies*
* *Make the specification simpler and easier to understand and use, with better documentation*

The summary page does not mention machine actionability explicitly.

# Coupling

The identifiers create coupling within and between DDI instances. For some users, who manage their metadata *as DDI*, this may be useful and desirable.

For users who manage metadata in other kinds of systems (with their own normalization and identification logic), and for users who simply don’t manage their metadata in sophisticated ways, this coupling represents a problem.

This became very evident to me when I was creating DDI 4 codebook examples by hand (see APPENDIX 1 below for the example). On one side, I was able to express precise properties and relationships by using the DDI 4 conceptual model. On the other hand I was forced to assign a globally unique identifier to most objects/elements, and to maintain references between them.

Such references represent coupling, and require me to create a sophisticated id-integrity-management system at hand to tie everything together correctly. I want to document my data with information that I already have (and would like to disseminate) - and I don’t want to build an id-management system to be able to do that. We manage normalization and integrity in internal systems, where policies and models change all the time, and which is different from the DDI-way of thinking about identifiers. Aligning all this with the DDI-way of thinking about identifiers and references adds a substantial burden that is not likely to pay off in a long long time.

# Doing one thing well

The task of documenting data (and related activities) is very very different from the task of managing identifiers and references. DDI tries to support both – and my statement here is that this prevents progress and adoption at this stage of DDI’s life. The domain expertise represented in the DDI community is very much in favour of the content side (i.e. data documentation), whereas integrity management is a completely different domain more related to the computer science domain.

Agreement on a common and sufficiently rich and consistent conceptual model with machine readable representations will have enormous benefits for data producers, distributors and consumers - regardless of machine actionability and fine-grained identity management.

# Example1 - Study[[2]](#footnote-2) Identifier as a namespace for localIds

In a Codebook use-case, the only object NSD would care assigning an institution-unique identifier to, would be the Study itself. Everything below the Study would only need local/relative identifiers that you need to be able to distinguish objects of the same type (e.g. InstanceVariables) from one another within the study.

This would be essentially the same as we had in DDI-Codebook 2.1, where variable ids were unique *within* the namespace of the containing Study.

# Example 2 - Collapse of the Variable Cascade

This is a quite different example than the one above, and has perhaps deeper implications.

The use case is documentation of variables in a Codebook scenario.

The DDI 4 conceptual model has three main classes:

* ConceptualVariable
  + RepresentedVariable
    - InstanceVariable

Now, Codebook talks about instance-variables. For organizations that maintain a registry of reusable RepresentedVariables and ConceptualVariables, it could sometimes (depending on the use-case) make sense to be explicit about the cascade also in a Codebook situation.

For many use-cases, and for many institutions who don’t care about reusability or think about reusability in a different way, the variable cascade gets in the way when producing a codebook instance. RepresentedVariables and ConceptualVariables don’t exist and cannot be named/referenced. But we still would like to talk about the ValueDomain (from RepresentedVariable) and in some cases also the ConceptualDomain (from ConceptualVariable) in the codebook.

This is what the old codebooks did, and they did it reasonably well. Now we have a *better* conceptual model, but no way of collapsing the variable cascade properties into inline properties with the InstanceVariable we’re documenting.

## Extra Controversial Part

One could argue that the model doesn’t need the variable cascade at all, only InstanceVariables with properties brought over from today’s ConceptualVariable and RepresentedVariable. Reuse of ConceptualDomains, ValueDomains and other currently reusable properties and their integrity would then have to be maintained *outside* DDI (in backing systems) and brought into DDI instances. For purposes such as comparability and harmonisation one would then rely on post-processing techniques like data mining, NLP-processing and other techniques to infer similarity or differences between different representations.

This is very controversial because we then remove the RepresentedVariable and ConceptualVariable also from the *model.* This probably sounds risky and far out - but here’s the point:

If we can show that InstanceVariables should have ValueDomains and could have ConceptualDomains - it will be up to *users* of DDI to make out how they want to model reuse in the most optimal way for them. Different user institutions and software vendors would likely think differently about how to do this in ways that would suit *them*, and organize their own variants of the objects we call ConceptualVariable and RepresentedVariable - but the output on the end-user side will be the same, and the essence of the conceptual model we have developed will be materialized.

# Benefits

## Ease of production

The biggest benefit by restricting the identified object to be the study only, is that it becomes easy to produce codebooks and other types of metadata compilations, regardless of buy-in into the entire DDI identification scheme in your production systems. You only have to buy into the level of detail that you need for your use-case and your organization.

## Inline content increases comprehensibility

Another benefit is that inline content is encouraged over referenced content. This makes processing of codebooks and other types of compilations substantially easier. Comprehensibility is extremely important for the cost of developing and implementing DDI solutions. It is not a coincidence that formats such as JSON and YAML have caught on for data interchange on the web. Formats based upon those syntaxes are (usually) easy to read and understand - and comprehension is perhaps *the* most important foundation for quality and speed in implementation projects.

# Cons

## No normalization within DDI instances

With inline, unidentified content, duplication will occur since you cannot “normalize” content within a DDI instance. Many organizations manage their metadata and data in non-DDI or (preferably) DDI-inspired systems where normalization and reuse is handled internally and integrated with production, collection, ingest and curation processes.

## Impacts for modeling principles

I don’t really overlook the impact these suggestions could have for modeling. I can envision what will happen if we collapse the Variable Cascade, because we have done that in NSD-projects already (RAIRD, etc). But other modeling consequences are not clear to me at this point as I’m no modeler. But I believe the effect of relaxing the identifiable-requirement, and favouring usability before reusability would simplify the model a great deal down the line.

## Reusability concerns are moved out of the DDI

It is important to acknowledge that reuse is possible without the DDI supporting it directly. CVs, templates, global content registries (e.g. for classifications and other code-lists) can be maintained and content and structures may be brought in for re-use. This will not support a “global normalized space of reusable content” (which we all dream of) - but it will support reuse. More importantly, a more widespread adoption of DDI could stimulate this type of out-of-band reuse that in volume could exceed the in-band global integrity-idea substantially.

# APPENDIX 1 - YAML-based Codebook Example Instance

The following YAML-based example was produced with the XML-based example (done by Oliver Hopt) as background. It is not complete, and not necessarily well-formed or fully correct. But it should give an idea of what the codebook really could look like:

---

codebook:

DocumentInformation:

Id: "urn:ddi:di1"

Agency: "DDI Alliance"

Version: "1.0.0"

HasAnnotation:

Annotation:

Id: "urn:ddi:andi1"

Title:

String:

Language: "en-US"

Content: "Simple Codebook first test instance"

String

Language: "de-DE"

Content: "Simple Codebook erste Testinstanz"

Abstract:

String:

Language: "en-US"

Content: "some text"

String:

Language: "de-DE"

Content: "ein bischen Text"

Creator:

Agent:

String:

Language: "en-US"

Content: "Oliver Hopt"

Contributor:

Agent:

String:

Language: "en-US"

Content: "Ørnulf Risnes"

Study:

Id: "urn:ddi:std1"

Agency: "DDI Alliance"

Version: "1.0.0"

HasAnnotation:

Annotation:

Title:

String:

Language: "en-US"

Content: "survey of the DDI users"

Language: "de-DE"

Content: "Befragung der DDI Nutzer"

SubTitle:

String:

Language: "en-US"

Content: "Knutholmen edition"

Language: "de-DE"

Content: "Knutholmen Ausgabe"

Identifier:

IdentifierContent: "http://doi.org/10.1234/lalala"

isURI: "true"

ManagingAgency:

Content: "DOI"

# This can be a controlled vocabulary entry as well

Creator:

Agent:

String:

Language: "en-US"

Content: "Oliver Hopt"

Contributor:

Agent:

String:

Language: "en-US"

Content: "Ørnulf Risnes"

Role:

Language: "en-US"

Content: "Data collector"

ControlledVocabularyName: "List of roles"

Publisher:

Agent:

String:

Language: "en-US"

Content: "drive.google.com"

Role:

Language: "en-US"

Content: "Distributor"

Copyright:

String:

Language: "en-US"

Content: "DDI Alliance"

HasTopcialCoverage:

TopicalCoverage:

Id: "urn:ddi:topcov1"

Agency: "DDI Aliance"

Version: "1.0.0"

Keyword:

Language: "en-US"

Content: "user study"

Keyword:

Language: "en-US"

Content: "coneheadded"

Subject:

Language: "en-US"

Content: "DDI internal review"

HasTemporalCoverage:

TemporalCoverage:

Id: "urn:ddi:tempcov1"

Agency: "DDI Aliance"

Version: "1.0.0"

CoverageDate:

NonIsoDate:

DateContent: "Mai 2016"

TypeOfDate:

Language: "en-US"

Content: "time period"

HasSpatialCoverage:

SpatialCoverage:

Id: "urn:ddi:spacov1"

Agency: "DDI Aliance"

Version: "1.0.0"

SpatialAreaCode:

Language: "en-US"

ControlledVocabularyName: "ISO 3166-1"

Content: "NOR"

Description:

Content:

Language: "en-us"

IsPlainText: "false"

Content: " <p xmlns=\"http://www.w3.org/1999/xhtml\">Somewhere in the nowhere.</p><p xmlns=\"http://www.w3.org/1999/xhtml\">Surrounded by norwegian nature</p>"

UnitType:

Id: "urn:ddi:ut1"

Agency: "DDI Aliance"

Version: "1.0.0"

Name:

Content: "INDIVIDUALS"

Context:

ControlledVocabularyName: "THE\_OFFICIAL\_UNIT\_TYPE\_CV"

HasAnnotation:

Annotation:

String:

Language: "en-US"

Content: "Individuals"

Language: "de-DE"

Content: "Personen"

Universe:

Id: "urn:ddi:uni1"

Agency: "DDI Aliance"

Version: "1.0.0"

Name:

Content: "TEENAGERS"

Context:

ControlledVocabularyName: "THE\_OFFICIAL\_UNIVERSE\_CV"

HasAnnotation:

Annotation:

String:

Language: "en-US"

Content: "Teenagers"

Language: "de-DE"

Content: "Teenagers"

HasInstanceVariable:

InstanceVariable:

Id: "urn:ddi:var1"

Agency: "DDI Aliance"

Version: "1.0.0"

HasAnnotation:

Annotation:

Id: "urn:ddi:anivar1"

Title:

String:

Language: "en-US"

Content: "sam1"

Abstract:

String:

Language: "en-US"

Content: "It's in the article!"

BasedOnRepresentedVariable:

RepresentedVariable:

Name:

Content: "SAM1"

DisplayLabel:

DescriptiveText:

UnitType:

Id: "urn:ddi:ut1"

Agency: "DDI Aliance"

Version: "1.0.0"

Name:

Content: "INDIVIDUALS"

Context:

ControlledVocabularyName: "THE\_OFFICIAL\_UNIT\_TYPE\_CV"

HasAnnotation:

Annotation:

String:

Language: "en-US"

Content: "Individuals"

Language: "de-DE"

Content: "Personen"

1. <http://www.ddialliance.org/ddi-moving-forward-process-summary> [↑](#footnote-ref-1)
2. Note that there are other “containing entities” that could play the role where the set of variables belongs to something that is *not* a Study (e.g. an Administrative DataStore). Study is here just an example. [↑](#footnote-ref-2)