Structured Re-usable Documentation for DDI 4

Prepared by: Oliver Hopt, Jon Johnson, Amber Leahey

28 October 2016

##

##

Introduction 2

Section 1: Establishing a structured reusable documentation approach 2

Missing documentation 3

Section 2: Structured re-usable documentation workflows 4

Model production workflow 4

Figure 1 - Documentation production model workflow - High-Level (HL) 4

Building a topic-mapping approach at the class-level for documentation slicing 5

Figure 2 - Documentation inclusions at the Class-Level (CL) 5

List of items for improvement 6

Section 3: Documentation for different audiences 6

Figure 3 - Audience documentation needs / types 7

Section 4: Use Case Builder Tool 7

Figure 4 - Use Case Builder Tool Overview Workflow 9

Figure 5: Example ‘Data Capture’ 10

Appendix 1: Issues reported to JIRA (Documentation WG) 11

##

## Introduction

The documentation of DDI can be used for different purposes and for different audiences. The DDI 4 model documentation is currently structured in such a way that makes the re-use of documentation components difficult. A key goal of the working group is to identify structures and workflows that are better optimized for documentation re-use to support different purposes and audiences.

Changes to the documentation process could be made using a structured approach. As new classes in the model are finalized, it would be required that this structure be completely populated to support the creation and re-use of documentation, beginning at the class-level, providing re-usable documentation at the functional-view level and for use cases. A class will thus be accompanied by rich documentation as well as the standard property and relationship sets that can be re-use across the model and tools to support implementation.

The new documentation structure aims to meet two requirements:

1. to support re-use of documentation for different outputs based on the model;
2. enabling changes to the modelling process to capture better documentation

At a high-level, documentation is provided throughout the model and this passes through a workflow in the production of DDI. As the documentation evolves the manner in which it is presented for difference purposes and audiences, but this needs to be able to be managed in a sustainable way that aligns to the evolution of changes in the model.

At the class-level, the capture of additional types of documentation to support re-use greatly enhances the modelling process, reduces inefficiencies, increases understanding, opens up the possibility of validity checking, and, enables the re-use of documentation for different use cases and audiences.

Audiences are particularly important for defining how documentation can be re-used, as certain audiences require certain types of documentation. Early consideration of these various audiences will reduce documentation management and maintenance burden, improve workflows across the model development, and build understanding both in the project and to support for instance, training efforts.

Finally a set of identified structured documentation approaches, workflows, and guidelines are provided which supports the creation and re-use of rich documentation across the model.

## Section 1: Establishing a structured reusable documentation approach

In general, documentation should ultimately aim to improve people's understanding of DDI and how to use it. Structured documentation is required for the systematic re-use of information about how to document data in the DDI 4 model. Researchers and data documenters should be able to use a particular DDI View and document their data without needing to know or understand the underlying model. In order to adopt this kind of approach to documentation, we need to identify the important elements of documentation creation, the audiences, considerations and constraints that may determine use of documentation, and ideally how this will be used or made visible to users.

**An overview approach for establishing structured re-usable documentation:**

1. Identify audiences

2. ‘Things’ to describe (model, classes, views)

3. How to do the ‘things’ (technical people, non-technical people)

4. How to surface the ‘things’ to different audiences

Using a structured documentation approach, it’s possible to address the documentation needs of some identified set of audiences, such as ourselves (DDI Moving Forward), deciders (managers, administrators), implementers (developers), and data documenters (data managers, researchers, etc.). Documentation requirements for the different audiences, includes identifying high-level and class-level documentation required for supporting migration by current users of DDI (Codebook and Lifecycle), linkages to glossaries, vocabularies, and definitions, and the requirements for defining relationships to existing documentation.

The following provides an overview of the documentation structure at the class-level. Moving forward all class items would be populated with the required documentation to support the goal of structured and re-usable documentation across the model. At the functional view level, documentation will take granular class items and provide users with what is recommended for use based on that specific functional-view.

**Structured documentation at the Class-Level (CL)**

|  |  |
| --- | --- |
| **Structured Documentation**  | **Class-Level (CL) (Required items)** |
| Formal Definition | **Y**  |
| Purpose | **Y**  |
| Relationship to existing documentation  | **Y**  |
| Other specifications / domains | **Y**  |
| Links to existing glossaries, vocabularies, definitions, etc. | **Y** |
| Technical examples e.g. links to use cases | **Y** |

Implementing a structured documentation approach considers the production workflows that can support documentation re-use. Therefore to ensure that documentation is improved and structured, as documentation is captured at the varying levels, an evaluation of the different documentation production workflows is necessary.

#### Missing documentation

Identifying items for inclusion and re-use allows the development and modelling team to generate a list of items or documentations still required, at the class-level. Such a list would support the creation of a structured set list of required documentation still in need of creation before publication of the model. This list will provide reviewers with a mechanism to review documentation before publishing the specification for use by others, and ensure documentation is complete.

## Section 2: Structured re-usable documentation workflows

### Model production workflow

Drupal and bitbucket, the documentation tools used for the creation of the DDI 4 model, supports structured and versioned documentation for classes, views, and packages. Using the high-level overview (model production workflow), the goal is to produce as much as possible *lossless* class-level documentation and incorporate structured documentation in the production of the DDI outputs (XML, RDF, etc.). The export and inclusion of all class-level elements into the XMI from Drupal ensures richer documentation of the model and uses of it.

#### Figure 1 - Documentation production model workflow - High-Level (HL) DDI-Views Production Flow - Page 1.png

*The above diagram represents the documentation model production workflow (the transfer, exchange, and reuse of documentation in the DDI 4 model).*

In order to facilitate the production and re-use of structured documentation for different purposes and different audiences, we currently need a mechanism for documenting recommended and excluded class-level documentation to be re-used in the creation of functional views, use cases, and audience specific documentation. This would support the re-use and slicing of specific documentation required for different purposes to be presented.

### Building a topic-mapping approach at the class-level for documentation slicing

The approach taken to enable documentation views or slicing is to structure items at a granular level. At the class-level, structured items include the specific fields associated with the class-level documentation, each of which can be identified for inclusion in the model specification, and, specified for re-use by a particular audience / or purpose. Enabling this through the use of field level unique identifiers to produce a topic based index of item association using “tags” would be an approach to building a mechanism for documentation slicing to support different audiences. Mandatory fields may include necessary items all users may require, similarly tagging fields as technical or technical-lite may also facilitate the slicing of documentation for implementers, developers, data documenters, researchers, and deciders. The ability to re-use documentation requires granular itemization and classification, once this is in place, it will be easy to re-use documentation across the model for different purposes and audiences.

The following presents a set of recommended documentation inclusions to support re-use of documentation for different purposes and audiences.

#### Figure 2 - Documentation inclusions at the Class-Level (CL)

In green – include under Definition field in the XMI (model)

**M – Mandatory**

**T - Technical**

**TL - Technical Lite**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Output | Index  | XMI fields | Type |
| Name | Y | Y | Y | M |
| Package | Y | Y | Y | T |
| Extends | Y | Y | Y | T |
| Is Extendable | N | N | N |   |
| Is Property | N | N | N |   |
| Status | N | N | N |   |
| Stage | N | N | N |   |
| Version\* | Incorporate Drupal versioning - Y | N | ? | M |
| DDI 3.2 | Y | N | N | TL |
| GSIM | Y | N | N | TL |
| RDF Mapping | Y | N | N | TL |
| Contributor | N | N | N |   |
| Definition | Y | Y | Y | M |
| Example | Y | N | N | M |
| Synonyms | Y | Y | N | M |
| Explanatory Notes | Y | Y | N | M |
| Overage Image | N | N |   |   |
| Properties\* | Y | N | Y | M |
| Relationships\* | Y | N | Y | M |
| PackageDescriptionName | Y | N/A | Y | M |
| FunctionalView | Y | N/A | Y | M |

#### List of items for improvement

Building on this approach, there is now currently a list of items for improvement (in JIRA), in terms of types and kinds of documentation, that can be added to the class-level. These items include adding explicit qualifiers to the DDI 3.2 mapping field to support meaningful mapping when possible, and to offer options for when no mapping equivalencies exist or if the modeller does not know if equivalencies exist. This would also support the ability to generate a structured “burndown” list when identifying elements in DDI 3.2 to be included or left to include in the DDI 4 model.

Similarly, at the moment there is no way to identify whether a particular class supports structured DDI vocabularies, which should be incorporated into the model and indicated appropriately in the documentation to support re-use. Adding usage information about a external controlled vocabularies, including DDI CVs (where applicable) should be considered.

At the time, there currently is inconsistent and incomplete documentation throughout the model, and thus there requires some training and support for users and developers to enable the creation of structured documentation to support testing workflows, validation, documentation re-use, and developing feedback mechanism for improvement of documentation across the model.

## Section 3: Documentation for different audiences

Documentation can be structured and made re-usable if the incorporation of documentation “type” tags are added to class items. This can ensure certain types of documentation are provided to specific audiences. The following identifies a set of audiences and their documentation needs.

#### Figure 3 - Audience documentation needs / types

|  |  |
| --- | --- |
| Audience | Types |
| Ourselves (Moving Forward) | everything |
| Deciders | M – High level |
| Developers | M + Technical |
| Documenters | M + T Lite |
| Researchers | M |

It has been identified that different audiences require different types of documentation. Indeed, researchers should not be expected to know or use quite technical documentation about DDI. In terms of documentation, often researchers struggle with technical data terminology, especially from the perspective of translating concepts across disciplines, research groups, etc. Additionally, as researchers are increasingly expected to document data, workflows, and processing steps, DDI documentation should aim to provide as much as possible support the re-use of documentation for a variety of purposes, including data reproducibility, auditing and compliance, referencing and identification, and reuse by others, which may include how concepts relate to survey questions and how variables are derived through a research processing workflow. Similarly, secondary users (researchers) require specific information about data provenance, question order, universe statements, etc. to support research reuse. As data are documented, it is easy to imagine how DDI and the underlying model that supports its capture is facilitating documentation exchange, and reuse, for a variety of purposes and audiences.

In an effort to improve understanding and ease of use it makes sense to adopt a structured documentation approach to facilitate the slicing of information to support different purposes and especially for the compiling of documentation to support different audience’s needs based on DDI.

Moving forward, the generation of documentation to support different audiences will be supported and can potentially lead to different DDI products to support its use across audiences. In the future, ideally a researcher will not have to know much about the technical model or structure of DDI, instead they will be able to capture and document data in a structured way, improving workflows, processing, and data description for a variety of purposes.

## Section 4: Use Case Builder Tool

Currently it is quite difficult to build structured and consistent use cases from the model. From a development and modelling perspective the model and documentation are not easily replicated or reusable, and as a result can lead to a delay in improvements due to to insufficient documentation and testing, inappropriate use, and conflicting use that should be reported and improved upon by those involved in modelling. There are few opportunities for automated testing, feedback loops, and use case building to support QC, further development, and use of the model.

Beyond the development and modelling associated with the DDI 4, use cases are useful sets of classes and dependencies that make up a particular use case. These can be granular, such as “capturing a blood pressure testing sample”, that may include concepts, questions, workflows, etc. Enabling the capturing of use cases will greatly improve the ability of researchers and data documenters to understand uses of the model and perhaps reuse when they are made available or published from a central repository. This can come much later, but for the time being there is a strong argument to make that use cases, and the ability to create structured, consistent use cases from the model is lacking and required.

Building use cases in a way that enabled reuse of the existing model’s documentation would greatly improve workflows for a variety of purposes and audiences. Currently in Drupal and BitBucket there exists a lot of documentation, and all of that can be reused for the purpose of creating use cases. Classes, for example, can be identified and selected, and properties and mappings inherited from Drupal to produce logical workflows and mappings within a particular use case. Use cases would have a set of components for people to work with and define, some of which can be inherited from classes or other documentation, but the capturing of a use case itself could be useful for others and therefore can become a published output itself for reuse.

What are the components of a Use Case?

1. Description
2. Requirements
3. Mappings
4. Code example(s)
5. Known limitations

Possible workflow for defining Use Cases with the model:

1. Use Case module accessed from online form
2. Selection of classes and mappings for use
3. Add explanatory information about the use case
4. Run and validate against the current model schema
5. Review outputs and provide feedback if issues reported
6. Publish to Use Case Repository

Use cases will require a space on the current documentation sites and this could potentially be in BitBucket, for centralization, and for technical model reuse reasons. The following is a list of technical components required for consideration and integration into a Use Case Builder Prototype.

Technical components to integrate into prototype:

1. Drupal
2. BitBucket
3. Validation and transformation actions
4. XML compiler and viewer
5. Storage of temporary use case generation
6. Error reporting and automated feedback loop (e-mail)
7. Repository for use cases

#### Figure 4 - Use Case Builder Tool Overview Workflow



##

#### Figure 5: Example ‘Data Capture’

##

## Appendix 1: Issues reported to JIRA (Documentation WG)

JIRA tasks, improvements, and new features added



Progress to date

