PhUSE 2016

Paper CD10

Linking Metadata from CDASH to ADaM

Author: João Gonçalves
Business & Decision Life Sciences, Brussels, Belgium

ABSTRACT

CDISC standards include instructions describing how variables are determined. These are used by industry professionals to do data management, implement computational algorithms that derive variable values and describe these algorithms for submission. This is meant to demonstrate traceability to reviewers, from collection to tabulation, and then from tabulation to analysis.

However, such human-readable traceability descriptions are only the first step of standardization. The move from textual descriptions of computations to formal specifications is a logical way forward, enabling a more automated way of working in the industry. This move implies enriching existing standards with metadata, explicitly linking computational algorithms to their source variables (both in the SDTM and ADaM cases), and working towards re-usable code that implements them. In this paper we present an example use case of such enriched metadata, demonstrating generation and traceability assessment all the way from CDASH to ADaM.

INTRODUCTION

CDISC standards describe how clinical data should be collected and submitted. These include, most importantly, the expected datasets, the variables present in those datasets and the used controlled terminology. They are published as specifications and Implementation Guides (IG), but also in more machine-friendly formats that describe the metadata in a more structured way such as Excel or Define-XML. These structured formats describe standard variable metadata provided by CDISC includes well-known attributes such as *Name*, *Label*, *Type*, *Order* and *Role*. However the two formats also show significant differences, for example regarding the presence of variables in datasets: as the Excel version contains the *Core* variable, directly drawn from the IG, the Define.xml has the *Mandatory* attribute.

Another difference of the Excel and Define.xml formats is the way that they describe each variable. While Define-XML has a semi-formal definition, described in the next Section, the Excel format simply relies on a textual description of the variable in the *CDISC Notes* column, directly drawn from the IG. This description guides industry professionals in a number of standards-related activities, namely data management, statistical programming, tabulation and analysis description for submission. In the cases where the variables are calculated instead of collected, this description typically defines how the variable is calculated, both for SDTM and ADaM.

Variable Name	Variable Label	Type	Codelist/ Controlled Terms	Core	CDISC Notes
AGE	Age	Num		Req	DM.AGE. If analysis needs require a derived age that does not match DM.AGE, then AAGE must be added
AGEU	Age Units	Char	(AGEU)	Req	DM.AGEU
AGEGRy	Pooled Age Group y	Char		Perm	Character description of a grouping or pooling of the subject's age for analysis purposes. For example, AGEGR1 might have values of "<18", "18-65", and ">65"; AGEGR2 might have values of "Less than 35 y old" and "At least 35 y old".

Table 1 - Example of standard ADaM metadata definition from the Implementation Guide

LINKING IN DEFINE-XML

The Define.xml is an essential deliverable describing tabulation data and analysis results. The origin of calculated variables refers to computational methods that can be found in the Computational Algorithms section of an SDTM Define.xml, or in the Analysis Derivations section of an ADaM Define.xml.

These computational methods are specified in the Define-XML format by using the *MethodDef* element. This element defines the *OID*, *Name*, the *Type* (e.g. "Computation"), and a *Description* that is often adapted or derived from the textual content of the *CDISC Notes*. Optionally, for complex algorithms that require detailed description, reference to an external document can be added via the *DocumentRef* element. Once defined, the *OID* of computational methods can be referenced by variables, via the *MethodOID* attribute of the *ItemRef* element. This means that the referencing variable uses that specific

computational method to be calculated. The XML code snippet below depicts, from bottom to top, the definition of the ADSL.AGEGR1 computational method via *MethodDef*, the definition of variable ADSL.AGEGR1 via *ItemDef*, its inclusion in the ADSL dataset via *ItemRef*, and the association of the variable and computational method via he *MethodOID* attribute of *ItemRef*.

```
<ItemGroupDef OID="IG.ADSL" Name="ADSL" SASDatasetName="ADSL" Repeating="No" IsReferenceData="No"
Purpose="Analysis" def:Structure="one record per subject" def:Class="SUBJECT LEVEL ANALYSIS DATASET"
def:CommentOID="COM.ADSL" def:ArchiveLocationID="LF.ADSL">
[...]
  <ItemRef ItemOID="IT.ADSL.AGEGR1" OrderNumber="17" Mandatory="No" MethodOID="MT.ADSL.AGEGR1"/>
[...]
  <ItemDef OID="IT.ADSL.AGEGR1" Name="AGEGR1" SASFieldName="AGEGR1" DataType="text" Length="20">
    <Description>
      <TranslatedText xml:lang="en">Pooled Age Group 1</TranslatedText>
    </Description>
    <CodeListRef CodeListOID="CL.AGEGR1L"/>
    <def:Origin Type="Derived"/>
  </ItemDef>

[...]

<MethodDef OID="MT.ADSL.AGEGR1" Name="MT.ADSL.AGEGR1" Type="Computation">
  <Description>
    <TranslatedText xml:lang="en">AGEGR1 = "&lt;18" if AGE &lt;18. AGEGR1 = "&gt;=18-&lt;25 " if AGE 18-24.
AGEGR1 = "&gt;=25-&lt;65 " if AGE 25-64. AGEGR1 = "&gt;65" if AGE &gt;65.</TranslatedText>
  </Description>
</MethodDef>
```

The ADaM Define.xml, beyond describing analysis datasets, variables and derivations, is meant to have Analysis Results Metadata – a description of tables, figures and listings (TFLs) present in the Clinical Study Report, each specified by a *ResultDisplay* element. A single *ResultDisplay* element can contain one or more *AnalysisResult* elements, which describe the data displayed. This description includes the specification of what dataset, variable and "where clause" is considered via the *AnalysisDataset*, *AnalysisVariable* and *WhereClauseRef* elements, respectively. These elements use an *OID*-based reference, analogous to the computational method case, to reference the source datasets, variables and where clauses. Furthermore, the actual code used to create the TFL based on the specified analysis variables is also referenced through the *ProgrammingCode* element.

Unlike for Analysis Results, and despite being a machine-readable format, the description of SDTM and ADaM computational methods in Define-XML, via the *MethodDef* element, remains mostly natural language, often mixed with pseudo-code. The machine-readable link between resulting variables and their computational methods, specified via *MethodOID*, has no counterpart on the source side – there is no machine-readable link between computational methods and the source variables or parameters that serve as input for them. Furthermore, while computational methods descriptions tend to follow a number of well-accepted practices, they remain what they are: informal natural language descriptions of mathematical or statistical calculations, and thus, more art than science.

SOURCE VARIABLES METADATA

The inexistence of a formal specification of source variables used for standard calculations is currently mitigated by the common industry practice of extending standard metadata with *mapping specifications*. Mapping specifications are structured documents - typically Excel sheets - that enhance existing study metadata with additional information, most relevantly the source variables used each derivation or computation, additional computation details, and fixed values to be used from the protocol or Statistical Analysis Plan (SAP).

Mapping specifications are slightly different for the SDTM and ADaM cases, as the possible origins and sources for calculation of SDTM and ADaM variables are distinct. In the SDTM case variables can be copied or computed from collected data (CDASH or non-standard formats), computed based on other SDTM variables, or set with a fixed value based on metadata standards rules or Protocol definitions. Analogously, ADaM variables can be copied or computed from SDTM, computed based on other ADaM variables, or set with a fixed value based on SAP definitions.

In order to support the specification of variable calculated (or copied) from other variables, mapping sheets typically have a few "source data" columns that define the source domain prefix(es) or dataset(s), as well as the source variable name(s). A "fixed value" column supports the cases in which the variable value is extracted from the study documents, namely Protocol and SAP. The additional computational details and comments are usually expressed in natural language, similarly to CDISC Notes in standard metadata.

DOMAIN	VARIABLE	LABEL	TYPE	LN	ROLE	CT	METHOD	ORIGIN	SOURCE DOMAIN	SOURCE VAR
ADSL	AGE	Age	num	8	Analysis			DM	DM	AGE
ADSL	AAGE	Analysis Age (Years)	num	8	Analysis		ADSL.AAGE	Derived	ADSL	AAGE
ADSL	AGEU	Age Units	char	5	Analysis	AGEU		DM	DM	AGEU
ADSL	AGEGR1	Age Group 1	char	20	Analysis	AGEGR1L	ADSL.AGEGR1	Derived	ADSL	AAGE
ADSL	AGEGR2	Age Group 2	char	20	Analysis	AGEGR2L	ADSL.AGEGR2	Derived	ADSL	AAGE

Table 2 - Mapping specification example with source variable

Reference	Computational Method
ADSL.AAGE	Equals to ADSL.AGE
ADSL.AGEGR1	Equals to '<18 years' when ADSL.AAGE < 18; Equals to '>=18-<25 years' when 18 =< ADSL.AAGE < 25; Equals to '>=25-<65 years' when 25 =< ADSL.AAGE < 65; Equals to '>=65 years' when ADSL.AAGE >= 65
ADSL.AGEGR2	Equals to '<41' when ADSL.AAGE < 41; Equals to '>=41 and <65' when 41 =< ADSL.AAGE < 65; Equals to '>=65' when ADSL.AAGE >= 65

Table 3 - Mapping specification example with Computational Methods

The popularity of this practice illustrates both the usefulness and feasibility of formal source variable definition. Support for this in Define-XML for mapping SDTM to ADaM could be accomplished via a method analogous to the one that exists for the analysis variable references in analysis results metadata. The key difference is that the OIDs referenced here originate from a different define file that would have to be referenced as a *leaf*. The XML code snippet below shows how it could look like.

```
<def:leaf ID="LF.SDTM-Define" xlink:href="../sdtm/define.xml">
  <def:title>SDTM Define.xml</def:title>
</def:leaf>

[...]

<ItemDef OID="IT.ADSL.AGEGR1" Name="AGEGR1" SASFieldName="AGEGR1" DataType="text" Length="20">
  <Description>
    <TranslatedText xml:lang="en">Pooled Age Group 1</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.AGEGR1"/>
  <def:Origin Type="Derived"/>
  <ext:SourceDatasets>
    <def:DocumentRef leafID="LF.SDTM-Define"/>
    <ext:SourceDataset ItemGroupOID="IG.DM" >
      <ext:AnalysisVariable ItemOID="IT.DM.AGE"/>
    </ext:SourceDataset>
  </ext:SourceDatasets>
</ItemDef>
```

The support of SDTM to CDASH or non-standard collection datasets links in Define-XML is slightly more complex than the SDTM to ADaM case as there is no define file for collected datasets, meaning the original collection datasets would probably have to be referenced.

AUTO-GENERATION PROOF OF CONCEPT

With Define-XML supporting machine-readable traceability from CDASH to Analysis Results Metadata, the following step would be the use of this metadata together with generic and formal computational methods to automate the generation of analysis datasets and TFLs. Mapping specifications are often used strictly in study-specific way, but many algorithms are the same across studies even if they may have some study-specific settings. We explored this in a proof of concept (PoC) developed at Business & Decision Life Sciences (BDLS).

Three common tables were selected, one depicting demographics, another vital signs and the third adverse events. Each of these require a set of variables from ADaM datasets, as well as some code to render the table. In turn, the supporting ADaM datasets – ADSL, ADVS and ADAE – require a set of SDTM variables from datasets and computational method code. The goal was to try and decouple as much as possible the variable and parameters metadata from the actual code, making it as re-usable as possible, as illustrated by Figure 1.

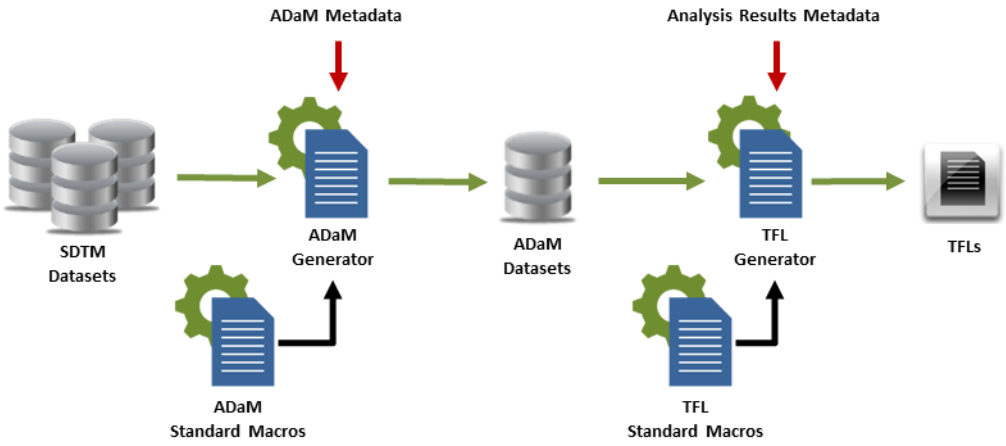

Figure 1 – Data flow diagram of the Auto-Generation Proof of Concept

The starting point for the PoC was set of study-specific mapping specifications and the SAS implementations of the computational methods defined there. These method implementations were separated in two parts: study metadata and generic method implementation. This implied abstracting away all study-related code from the method implementation, and move them to parameters.

For example, specific age group configurations were separated from the age calculation method. The code snippet below shows two different age group algorithms used in the same study defined at the metadata level. These two metadata-based definitions use the same underlying age-grouping macro, *MSDTM_AGEGRy*, to generate the values for both ADSL.AGEGR1 and ADSL.AGEGR2.

```
%MSDTM_AGEGRy(
        AGE_GROUP_NUMBER=1,
        AGE_GROUP='<18 years;>=18-<25 years;>=25-<65 years;',
        AGE_GROUP_LIST='<18;<25;<65;>65',
        AGE_GROUP_OTHER='>=65 years'
);

%MSDTM_AGEGRy(
        AGE_GROUP_NUMBER=2,
        AGE_GROUP='<41;>= 41 and < 65;',
        AGE_GROUP_LIST='<41;<65;',
        AGE_GROUP_OTHER='>= 65'
);
```

Similarly, the text, variables and where clauses were separated from the macro that creates a generic table, figure or listing. The code snippet below shows how a generic descriptive statistics table macro can be used to generate a demography descriptive statistics table specific for that study.

```
%TABLE_DESC_STATS(
        STUDY='XX YY mg (STUDY_ID_001)',
        NAME='Table 14.1.5.1',
        DESCRIPTION='Demographics - Intention-To-Treat population',
        VARIABLES='ADSL.AGE;ADSL.AGEGR2;ADSL.SEX;ADSL.RACE;ADSL.HEIGHT;ADSL.WEIGHT;ADSL.BMI;',
        GROUPS='ADSL.ACTARMCD=A;ADSL.ACTARMCD=F;',
        GROUP_LABEL_VARIABLE='ADSL.ACTARM',
        WHERE='ADSL.ITTFL=Y',
        VARIABLE_FOOTNOTES='ADSL.WEIGHT;Weight at baseline (Week 1). If weight is missing at baseline, weight
at screening is used;ADSL.BMI;Body Mass Index (BMI) (kg/m2) = Weight (kg) / [Height (m)]2;',
        FOOTNOTES='N = Number of subjects in specific group. n = Number of subjects. Calculation of
percentages based on N.'
);
```

The same separation exercise was done to all the computational methods implementations required to create the selected tables, creating what is depicted in Figure 1 as *standard macros*. These standard macros can be re-used across studies, enabling the automated generation of datasets and TFLs with simple parameterization changes. The PoC was especially successful regarding the generation of ADaM datasets. Although feasible, we found the TFL code more time-consuming to generalise, as there is more study-level metadata to be abstracted, as can be seen by the example.

END-TO-END METADATA-DRIVEN PROCESS

By decoupling study metadata from the computational method implementation we obtain the standard code that can be referenced by method definitions at the *library-level*, i.e. independently from study, such as the *MSDTM_AGEGRy* and the *TABLE_DESC_STATS* macros. Enriching these with end-to-end machine-readable traceability metadata, as described in Section 3, would result in the creation of a metadata and code library that could be used, simply through study-specific configuration, to generate common SDTM datasets, ADaM datasets and TFLs, in an automated metadata-driven process. Consequently, the following step of BDLS's Auto-Generation PoC was to be able generate the code shown in the previous section based on such library-level metadata and some study-specific input as depicted in Figure 2.

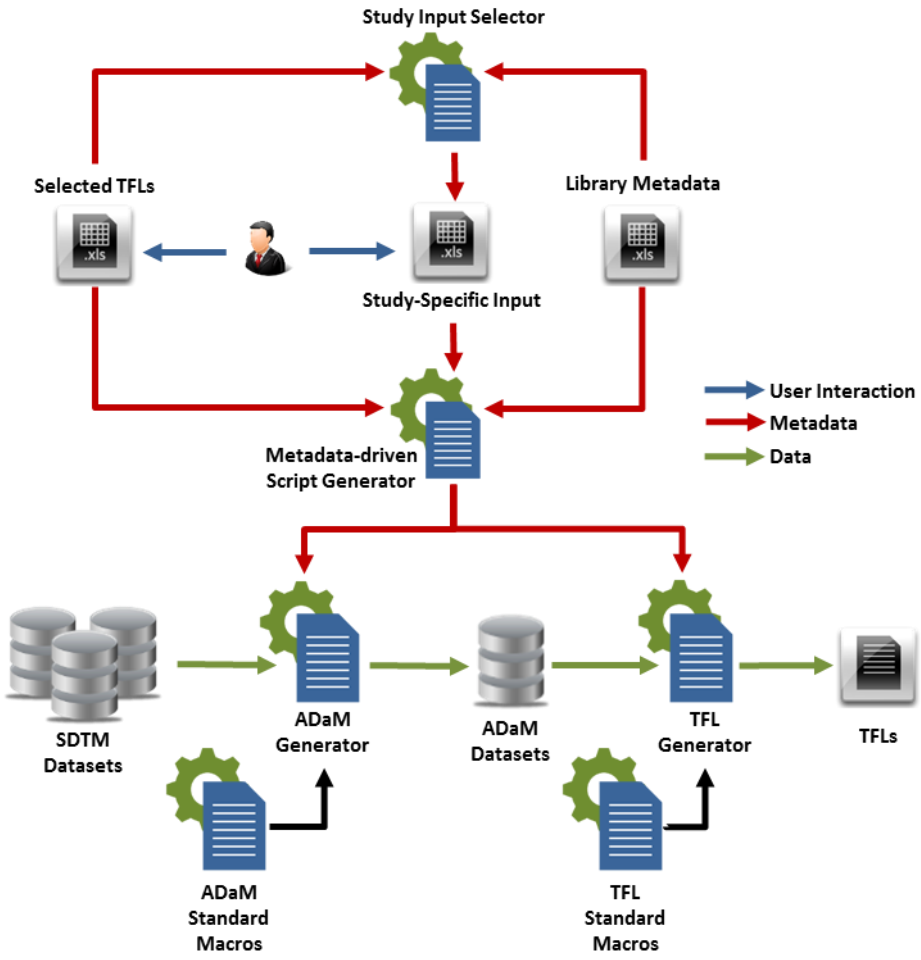

Figure 2 – Metadata-Driven Auto-Generation Proof of Concept Diagram

To this end, we described the generalized versions of the computational methods with metadata listing their source variables, domains and relevant value-level metadata, as well as their study-specific input requirements, as depicted in Table 4.

Reference	Description	Pseudo Code	Standard Macro	Study-Specific Input
ADSL.AAGE	Equals to ADSL.AGE	Equals to ADSL.AGE	MSDTM_AAGE	
ADSL.AGEGRy	Creation of the age groups. A subsequent number will be used as a suffix to distinguish between the different groupings.	Equals to %AGE_GROUP[n] when ADSL.AAGE < %AGE_GROUP_LIST[n], for the lowest n possible; Equals to %AGE_GROUP_OTHER if ADSL.AAGE > %AGE_GROUP_LIST[n], for the highest n possible	MSDTM_AGEGRy	%AGE_GROUP_NUMBER - value for the y suffix %AGE_GROUP - list or array containing the values to be assigned to the AGEGRy variable, except the one for the highest age group %AGE_GROUP_LIST - list or array containing upper age limits in the same order as %AGE_GROUP %AGE_GROUP_OTHER - the value to be assigned to the AGEGRy variable for the highest age group

Table 4 - Example of Library-level metadata description of a Computational Method

The existence of such a fully linked library such as this enabled a simple standard TFL selection and parameterization to calculate required study metadata at all levels – from TFL and Analysis Results to ADaM, from ADaM to SDTM and from SDTM to CDASH. This was accomplished in the PoC by the Study Input Selector script. This script takes the required TFLs

for a study from an Excel file, resolves the minimum required analysis results and ADaM variables for those TFLs, and pre-generates a study-specific input Excel file. This pre-generated file is then edited by the user with user specific configurations, as shown in Table 5 where the user configures two age-group configurations.

Reference	Param1	Param2	Param3	Param4
ADSL.AGEGRy	1	<18 years;>=18-<25 years;>=25-<65 years;'	'<18;<25;<65;'	'>=65 years'
ADSL.AGEGRy	2	< 41;>= 41 and < 65;'	'<41;<65;'	'>= 65'

Table 5 - Example of study-specific input for the PoC

A second script, the Metadata-driven Script Generator, then takes the study specific inputs and again the picked TLFs and the library-level metadata, to create SAS scripts capable of generating the required datasets or TFLs. It imports the referenced macros and composes the macro calls as outlined in the previous section.

CONCLUSION

This paper discussed two aspects of end-to-end metadata linking: how Define-XML could become capable of supporting this concept and the automation potential that it delivers. The metadata-driven generation of datasets and TFLs tested with the PoC promises significantly reductions in programming effort and associated manual errors, after an initial macro standardization investment. Furthermore, the code reusability potential would bring current pharma programming practices closer to generally accepted software best practices, where code is version controlled and continuously re-used and improved, instead of copied and adapted.

Perhaps most relevantly, this standardization effort enables the traceability currently only considered at study-level to become available end-to-end at a library-level. In the PoC we were able to generate SAS macros based on library-level metadata and study-specific input, which demonstrates the completeness of the metadata models considered, both at library and study levels. This traceable library-level metadata, together with some study input, allowed to determine the required downstream variables and their study input. If the approach is extended all the way to data collection, this would enable new advanced forms of study design, pre-populating study Case Report Forms based on the TFLs required in the Clinical Study Report.

The machine-readable representation of traceability metadata in Define-XML format would bring clear benefits to the industry, most obviously for introducing some automation in the traceability review process. However, as demonstrated by the PoC, this automation could be extended in the medium term to the analysis processes, and soon after to the study design process.

Future work includes validating the considered metadata model against a larger number of distinct types of TFLs. The metadata considered to describe TFLs at study level, Analysis Results Metadata, is now undergoing industry-wide validation and adoption. While in the PoC we devised a library-level counterpart, this model needs further testing and validation.

ACKNOWLEDGMENTS

This paper could not have been written without the contributions of several BDLS collaborators, namely the end-to-end traceability discussions with Peter Van Reusel, the ADaM expertise of Hermann Ziehl, and the SAS programming of Fernando Siqueira and Sonu Agarwal.

CONTACT INFORMATION

Contact the author at:
João Gonçalves
Business & Decision Life Sciences
St Lambertusstraat 141 Rue Saint-Lambert
1200 Brussel – Bruxelles
Work Phone: +32 2 774 11 00
Fax: +32 2 774 11 99
Email: joao.goncalves@businessdecision.be
Web: www.businessdecision-lifesciences.com