PhUSE 2017

Paper SI04

Analysis Results Metadata 1.0 for Define-XML 2.0 Benefits to Statistical Analysis

Lionel Debecq, Business & Decision Life Sciences, Brussels, Belgium
Roxane Debrus, Business & Decision Life Sciences, Brussels, Belgium

ABSTRACT

Since its release in 2014, ADaM Define-XML 2.0 has been enriched with “Analysis Results Metadata” (ARM). Define-XML tells the FDA what datasets, variables, controlled terms, and other specified metadata were used. However, for a study converted in the ADaM model, all the information necessary to the good understanding of the whole study are not present in the Define-XML. CDISC added the ARM extension to its current model to compensate this lack. ARM assists the reviewer to have an overview of the key statistical analysis criteria and the study benefits of the clinical study. With a practical example, we would like to demonstrate what we have learned during the implementation of ARM for Define-XML, and to show the additional functionalities and benefits.

INTRODUCTION

Critical analyses, links between results, documentations on the analyses performed are generally present in the Statistical Analysis Plan (SAP), the Clinical Study Report (CSR), even the programs sometimes, but not in the Define-XML itself. Analysis Result Metadata provides all the links and documentations specific to the analysis. It also provides information about the analysis method used and the reason the analysis was performed.

Another big advantage, from a regulatory submission point of view, is the traceability and reproducibility of the key results. The reviewer will have a better view on all factors necessary to understand the submission, and a better understanding of the study benefits. In this paper, we will introduce what is Define-XML, ARM in a Define-XML perspective, its theoretical implementation in a project, and the issues met during the process.

ORIGINAL PURPOSE

Define-XML was originally designed to be a machine-readable document for analysis processing, it has also the particularity to present all metadata to a reviewer. Define-XML is commonly used to show *traceability* between SDTM and ADaM datasets.
Traceability could be defined as *the ability to track a specific piece of data through the study to its origin*. Traceability is important for transparency and understanding. For both the Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM), a Define-XML will include different levels of metadata:

- Dataset level metadata
- Variable level metadata
- Value level metadata
- Code list metadata
- Computational algorithms
- Linked documentation
- And specifically for ADaM, result metadata

The results metadata are referred to as Analysis Results Metadata (ARM) and are provided as an extension to the Define-XML model. The version 1.0 of the specifications is published since January 2015.

All the results created from the ADaM datasets are not relevant or necessary for the good understanding of the study. Therefore, we faced two difficulties. One is the conversion from Define-XML 1.0 to Define-XML 2.0. There are big structural and content changes between version 1.0 and version 2.0 and no backward compatibility. (1) Second is extracting the relevant and necessary information from the sources (SAP, CSR, and Programs), and integrate those information in the Define-XML 2.0 to enhance (a) Readability and, (b) Traceability.

WHY ARM? THE IDEA BEHIND IT.

Hume, Aerts, Sarnikar, & Huser (2016) (2) described the origin of Define-XML. Define-XML is an extension to the Operational Data Model (ODM). Analysis Results Metadata is a new (optional) extension to ODM and specifically to address specific concerns and needs of ADaM in Define-XML. Those specific concerns and needs of ADaM have been addressed by the CDISC ADaM Metadata Sub-Team in 2015. There was a need for traceability for a particular analysis result, for specific ADaM data used to generate the results, and to provide information about the methods used, and the reason the analysis was performed. ARM is ADaM version 2.1 compliant. Those results metadata are provided to assist the reviewer by i.e. identifying the critical analyses, providing links between results, documentation, and datasets, and documenting the analyses performed.

This would suggest additional work for whoever is responsible to identify those key points of the analysis (such as tables, figures, individual p-values, etc.). In addition to metadata on analysis datasets, variables and associated information necessary to assist the reviewer, for a single clinical study or an integrated summary.

Analysis Results Metadata has been assessed by the Pharmaceutical and Medical Devices Agency (PMDA) around 2015 and presented in a review document in 2016. From their assessment, there is an importance of standardized analysis datasets and the relationship to the results. Most reviewers examine the submission materials (analysis results) first. Since that assessment, PMDA requests ADaM datasets and Analysis Results Metadata. (3)

While PMDA requires ARM with the ADaM Define-XML 2.0, the Food and Drug Administration (FDA) doesn't state the requirement of ARM (4). Nonetheless, Define.xml version 2.0 is the newest standard supported by the FDA, and available since 2013. The support for Define.xml version 1.0 will stop for studies that start 12 months after March 15, 2017 (5). It is in the best interest to be prepared for Define-XML 2.0, and for ARM as well.

SOLUTION

In 2015, BDLS decided to develop a metadata specifications converter, capable of converting most of the metadata issued for the version 1.0 of the Define-XML to version 2.0. This made it possible to deliver a Define-XML in a few clicks. Meaning less work to perform on the file, lesser time spent, and less error-prone.

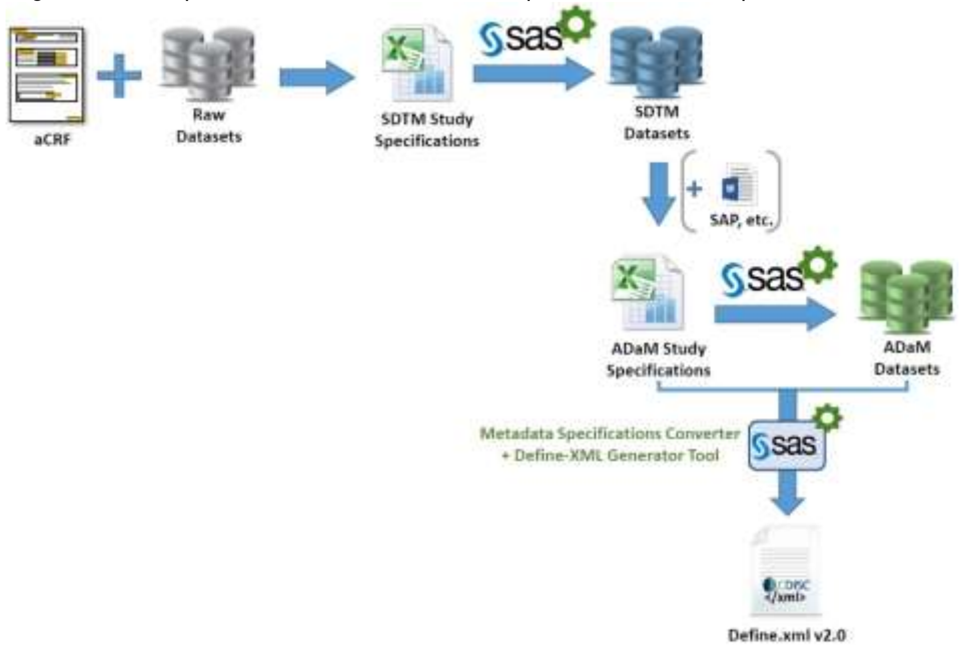

aCRF
Raw Datasets
SDTM Study Specifications
sas
SDTM Datasets
SAP, etc.
ADaM Study Specifications
sas
ADaM Datasets
Metadata Specifications Converter + Define-XML Generator Tool
sas
CDISC </xml>
Define.xml v2.0

Figure 1 Process for Define-XML 2.0 generation

By definition, Analysis Results Metadata would not necessarily exist at the time of the ADaM datasets creation from Raw Data, Analysis Data Sets (ADS), or SDTM Data, even if ARM is linked to them. ARM has been designed originally for the primary and secondary end-point results. The decision about the results to consider belongs to the sponsor. Thus, the definitive results components won't be known until the finalization. (9)

We wanted to be able to create a define.xml without ARM and, when available at a later stage generate a define.xml with ARM without too much work (or re-work) to the current study specifications.

There were 2 possible solutions: (a) ARM is part of the whole specifications, then, complete the metadata specifications with *Dummy Data* or leave the ARM part blank; (b) have a metadata specifications separated from the main specifications. Since the purpose is to minimize the work on the specifications, the solution (a) is not suitable. ARM is not designed to be completed from the start of ADaM conversion. The best practice is to complete an extra specification document when all the information are available from the sponsor.

GENERATION METHODOLOGY

The Define-XML is generated from a metadata repository for a study from the Study Specifications (mapping). To achieve the ARM integration, the generation process had to be slightly adapted, by addition of a conversion step.

Some assumptions:

- Two files will be used: one “main” metadata file and one ARM file
- Both metadata specifications are Define-XML 1.0 compatible
- Files are in Microsoft® Excel Open XML format (.*xlsx*). The tool is used as a common interface to complete efficiently the specifications.

Results metadata specifications version 1.0 are added to the metadata specifications through a separated document that would need reconciliation and treatment to enable the addition to the main metadata. The metadata needed to be converted to the appropriate model prior the generation of the Define-XML.

DEFINE-XML WITH ARM GENERATION

In early 2015, Business & Decision Life Sciences developed an in-house tool to make the conversion of a given Study Metadata Specifications to a particular version of Define-XML accessible and easier.

The tool operates from a Metadata Environment:

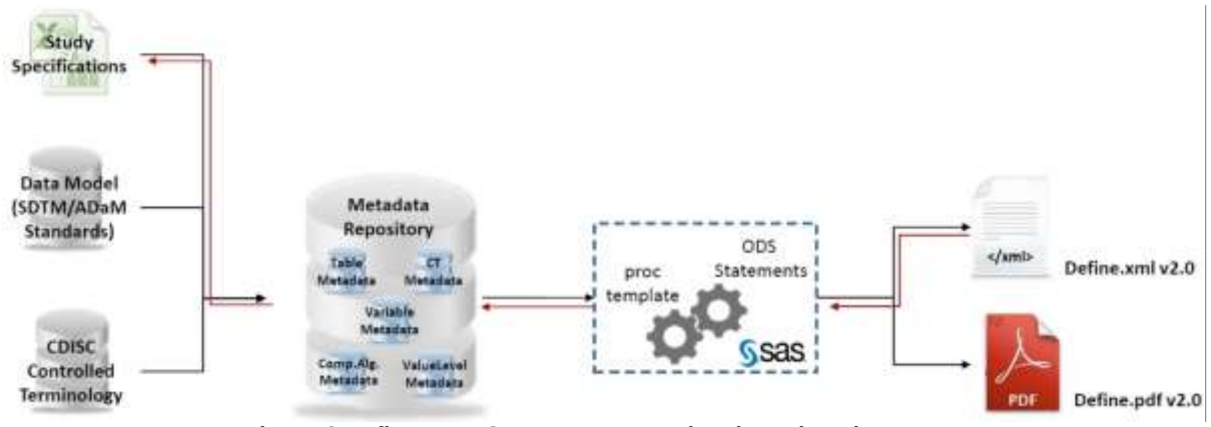


Figure 2 Define-XML Generator – Metadata based environment.

The tool has been developed as an easy way of generating a Define.xml from study specifications. It reads metadata specifications stored in Excel format, loads them in a metadata repository and applies SAS® ODS statements from template used to parse the metadata loaded in the repository and, generate the Define-XML files (Figure 2).

The tool is designed to be metadata oriented, and compared to the previous Define-XML generation method, the work needed to generate a Define-XML is less than it used to be, considering the work on ARM.

A large portion of the information in the metadata specifications can be ported or derived to the second version of the metadata specifications, compatible with Define-XML 2.0, with very few additional work to add the missing information (custom variables label, custom code lists label, and some specifics), that cannot be ported directly to the second version of the metadata. The SAS® process (SAS Logo with the gear symbol) mentioned in the figure consists of a series of Macros that remap the original study specification’s metadata variables used to make the version 2.0. The specifications in their first version are kept because they are easier to read, to complete and to understand for any user. But, to use them for define.xml, a remap is needed. In the remap process, we include the variables remap, all derivations due to structure change (i.e. ValueLevelMetadata) and internal consistency checks (i.e. Codelist in VariableMetadata exists in CodelistMetadata sheet, etc.).

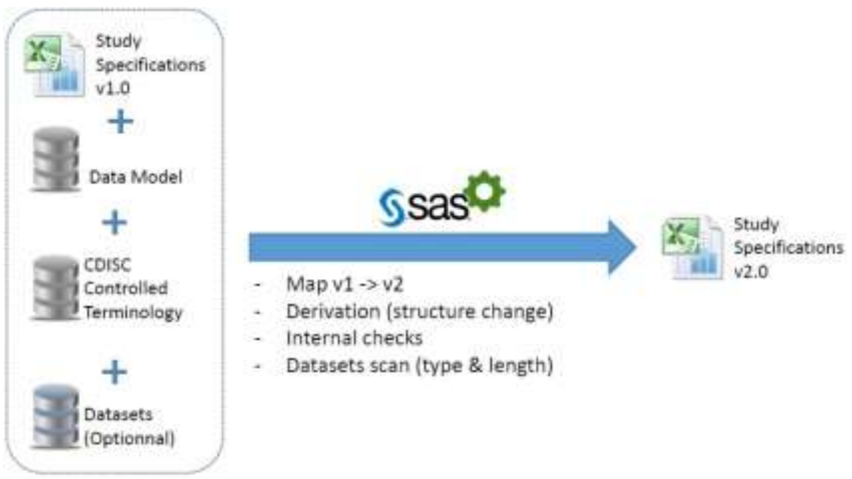

Study Specifications v1.0
+
Data Model
+
CDISC Controlled Terminology
+
Datasets (Optionnal)
sas
- Map v1 -> v2
- Derivation (structure change)
- Internal checks
- Datasets scan (type & length)
Study Specifications v2.0

Figure 3 High Level Process Conversion for Study Specifications V1.0 to Study Specifications V2.0.

Datasets are optional, target datasets are not always available at time of Define.xml generation.

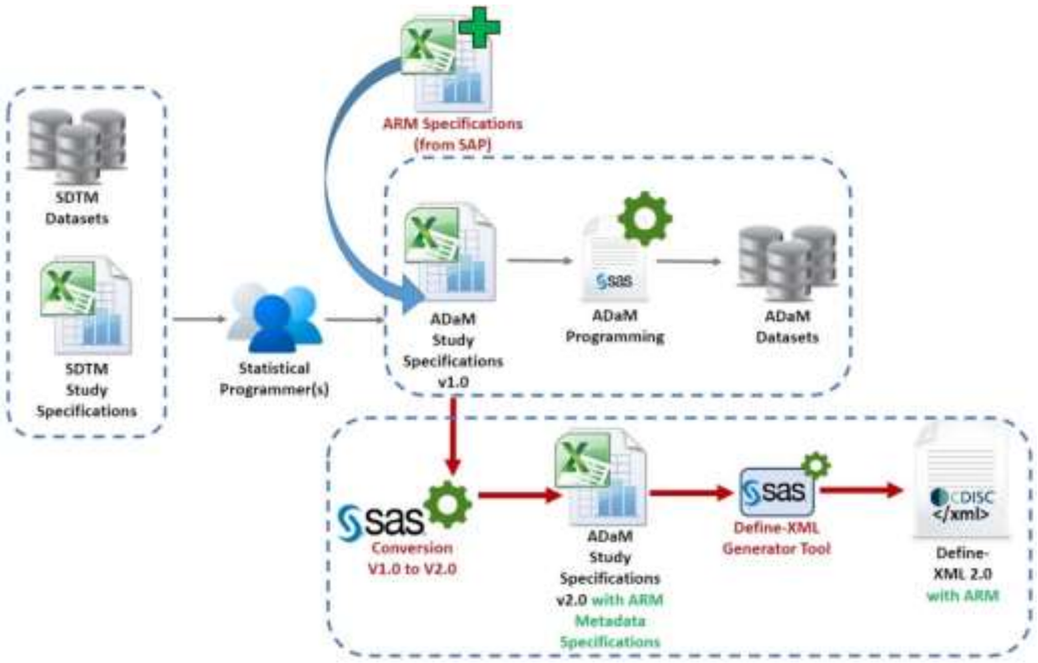

ARM Specifications (from SAP)
SDTM Datasets
SDTM Study Specifications
Statistical Programmer(s)
ADaM Study Specifications v1.0
ADaM Programming
ADaM Datasets
sas Conversion V1.0 to V2.0
ADaM Study Specifications v2.0 with ARM Metadata Specifications
Define-XML Generator Tool
CDISC </xml>
Define-XML 2.0 with ARM

Figure 4 General Conversion process. From left to right, SDTM to ADaM+ARM to Define-XML 2.0 with ARM

The Figure 4 represents the overall process from SDTM to ADaM and the Define.xml with ARM, with the notion of ARM being optional to the document generation. This method is working in two-steps:

I. Conversion of Study Specifications (with ARM)
II. Generation of Define-XML (with ARM)

This method in two-steps allows us to create a valid Define-XML 2.0 (without ARM), and when ARM is ready to be included, create a Define-XML 2.0 with ARM without too much re-work. This process have the advantage to be time saving (the conversion from one version to another without datasets is approximately taking a minute on laptop, less than 10 minutes with datasets length and type scan), i.e. by already preparing other deliverables while the analysis key points might not be fully known yet.

This method decreases the quantity of user manipulations compared to the previous method (each bullet-point is a user activity):

Define-XML version 1.0	***Define-XML version 2.0***
• *Study Specifications import,* • *Creation of the Metadata from the study specifications imported,* • *Import of the metadata in a Metadata Based Environment, SAS Data Integration Studio,* • *Define-XML v1.0 generation*	• Study Specifications (with ARM) Conversion • Import in the Define-XML generator tool for Define-XML v2.0 generation

The generation is done more direct. The user is not lost in multiple obscure steps with a lot of program to run. There are only two main steps in this approach for Define-XML (with ARM), which saves time.

METADATA SPECIFICATIONS FOR ARM

We tried to make the completion of the metadata specifications for ARM as simple as possible, while respecting model constraints. Thus, we had to set additional completion rules to avoid wrong data, the constraints are available on the "Analysis Results Metadata Specification Version 1.0 for Define-XML Version 2" document provided by CDISC (1). These simple rules allows us to generate Metadata Specifications compliant with Define-XML 2.0.

The tool has been updated to:

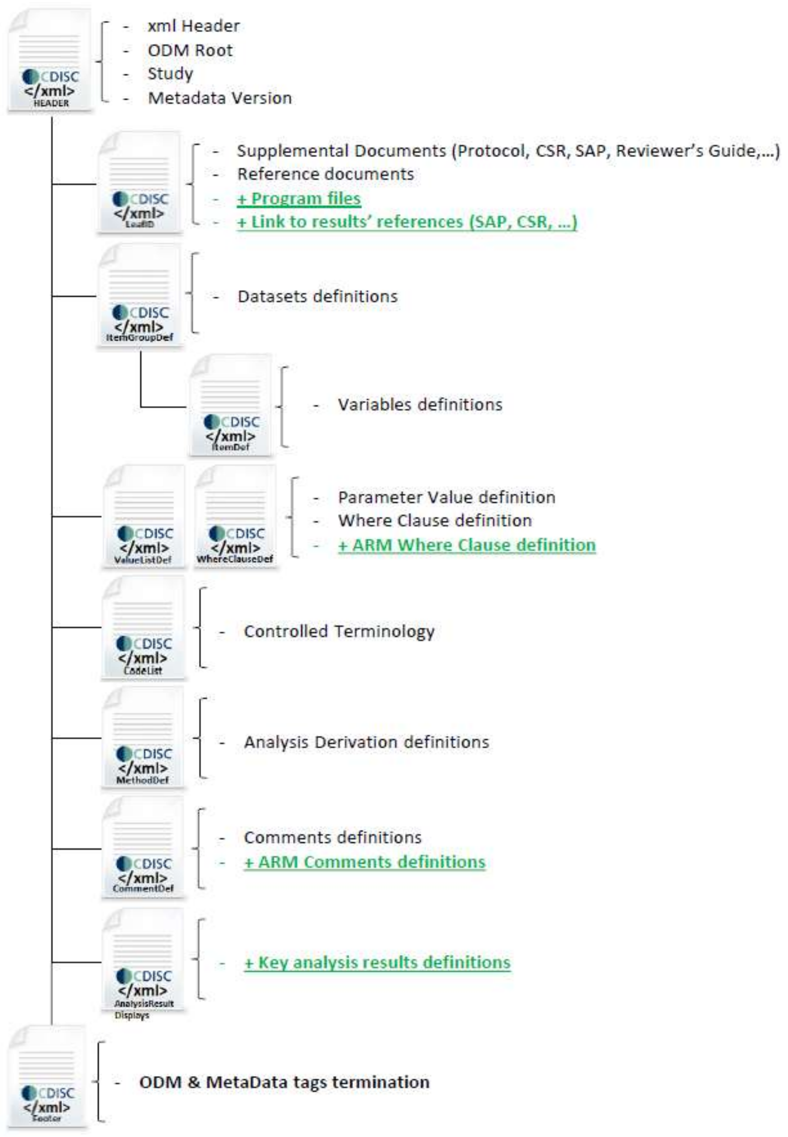
- Recognize when ARM is present in the file and when it is not; if the ARM sheet is present, import it.
- Create the ARM *ODS* template to generate the AnalysisResults xml.
- Insert the *WhereClause* elements of ARM to the main WhereClause of the Define-XML.
- Create the document elements (leafID).

The Stylesheet is available on the CDISC website.

The only remaining constraint was to add the Document and the ARM part in the right place inside the xml structure, following the model rules:

1. Repeated elements come first (documents, datasets, variables, etc.)
2. Common elements of ARM in Define.xml are added to the existing parts (documents, WhereClause, etc.)
3. ARM is added at the end of the XML File, following internally the first rule.

Using those simple rules from CDISC, we followed the structure of Figure 5:



CDISC </xml> HEADER
- xml Header
- ODM Root
- Study
- Metadata Version
CDISC </xml> LeafID
- Supplemental Documents (Protocol, CSR, SAP, Reviewer's Guide,...)
- Reference documents
- + Program files
- + Link to results' references (SAP, CSR, ...)
CDISC </xml> ItemGroupDef
- Datasets definitions
CDISC </xml> ItemDef
- Variables definitions
CDISC </xml> ValueListDef
CDISC </xml> WhereClauseDef
- Parameter Value definition
- Where Clause definition
- + ARM Where Clause definition
CDISC </xml> CodeList
- Controlled Terminology
CDISC </xml> MethodDef
- Analysis Derivation definitions
CDISC </xml> CommentDef
- Comments definitions
- + ARM Comments definitions
CDISC </xml> AnalysisResult Displays
- + Key analysis results definitions
CDISC </xml> Footer
- ODM & MetaData tags termination

Figure 5 CDISC XML Structure, template view.
The text in green are the sections added for ARM.

THE DIFFICULTY TO MAP ARM

The conversion rules have been programmed following the structure rules established by CDISC (*see previous section*) (1) (10). Figure 6 represents a tree, showing the organisation of the metadata from ARM linked to the define.xml. From the results, one can easily track each component of the results metadata to its dataset's origin. On the opposite, tracking the data to the results is not possible. By design, there is no link in the "main" dataset metadata to a specific result.

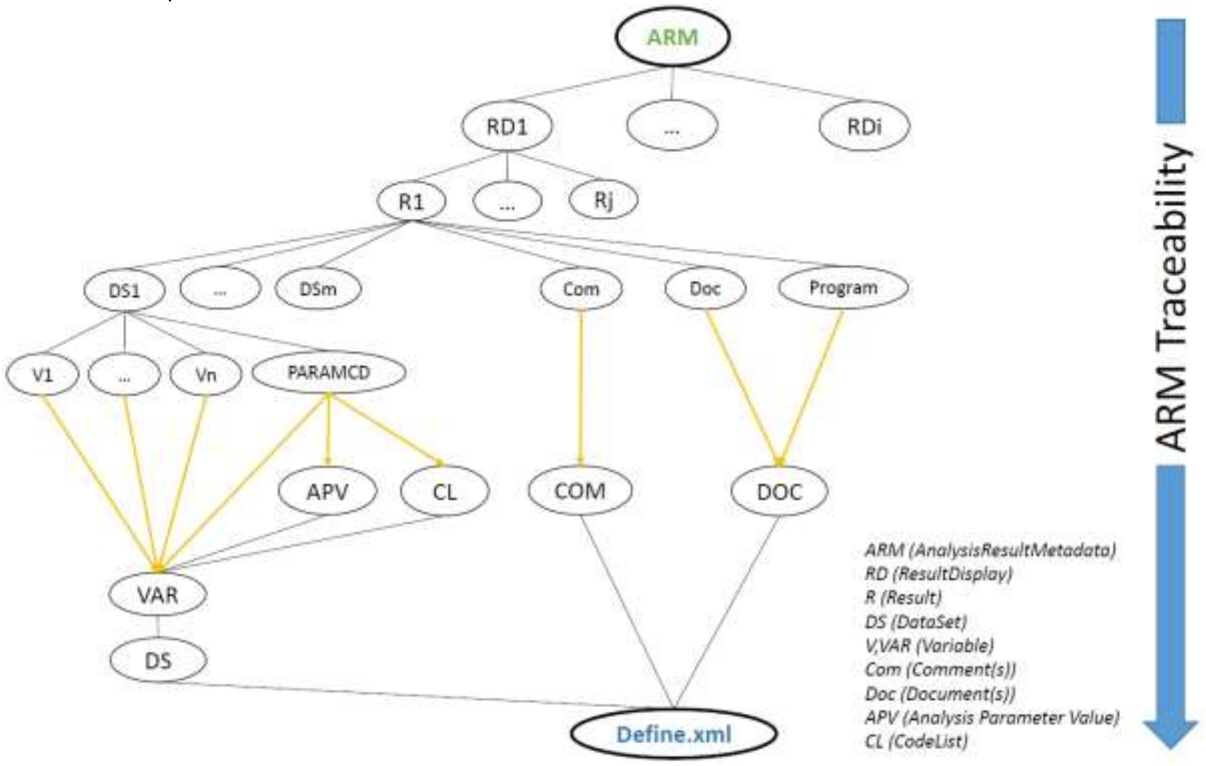
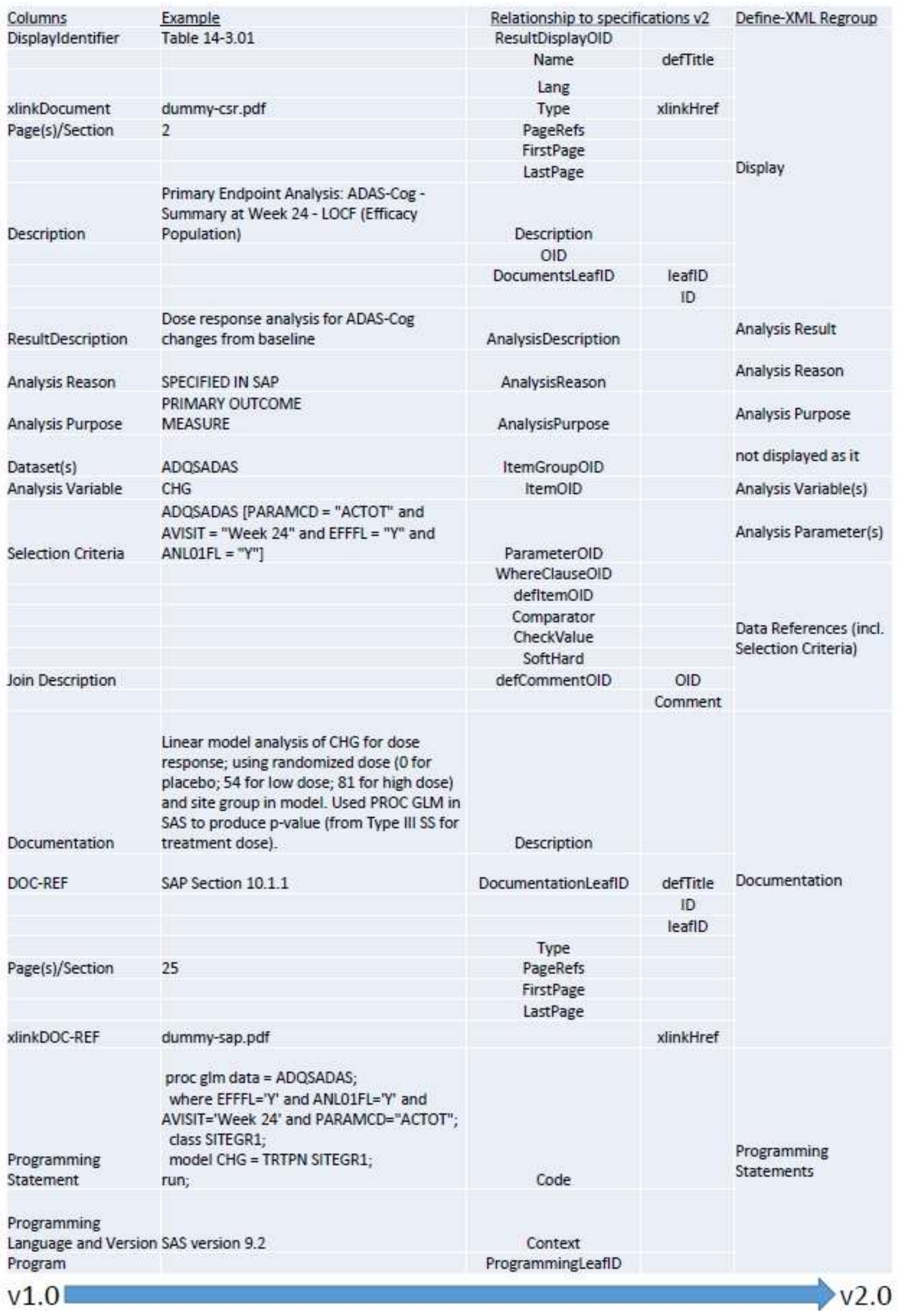


Figure 6 Tree for Traceability, from ARM to Define.xml Metadata.
The Yellow links are one way direction, and go from ARM to Define.xml.

Assuming that both the metadata specifications and the SAP (as well as CSR and programs) are final to minimize the extra-work. The statistical programmer (or the sponsor), completes the ARM Specifications from the available sources and the dataset contents. The Specifications are then converted to Metadata Specifications version 2.0 and fed to the generator, under the hypothesis that Define-XML is generated properly.
Each Result Display, a group of results, contains one or more sub-results. Which refers to one or more datasets, that contains one or zero parameter, one or zero analysis variable, and one or many selection criteria, constructed for one or many variables associated to its results.

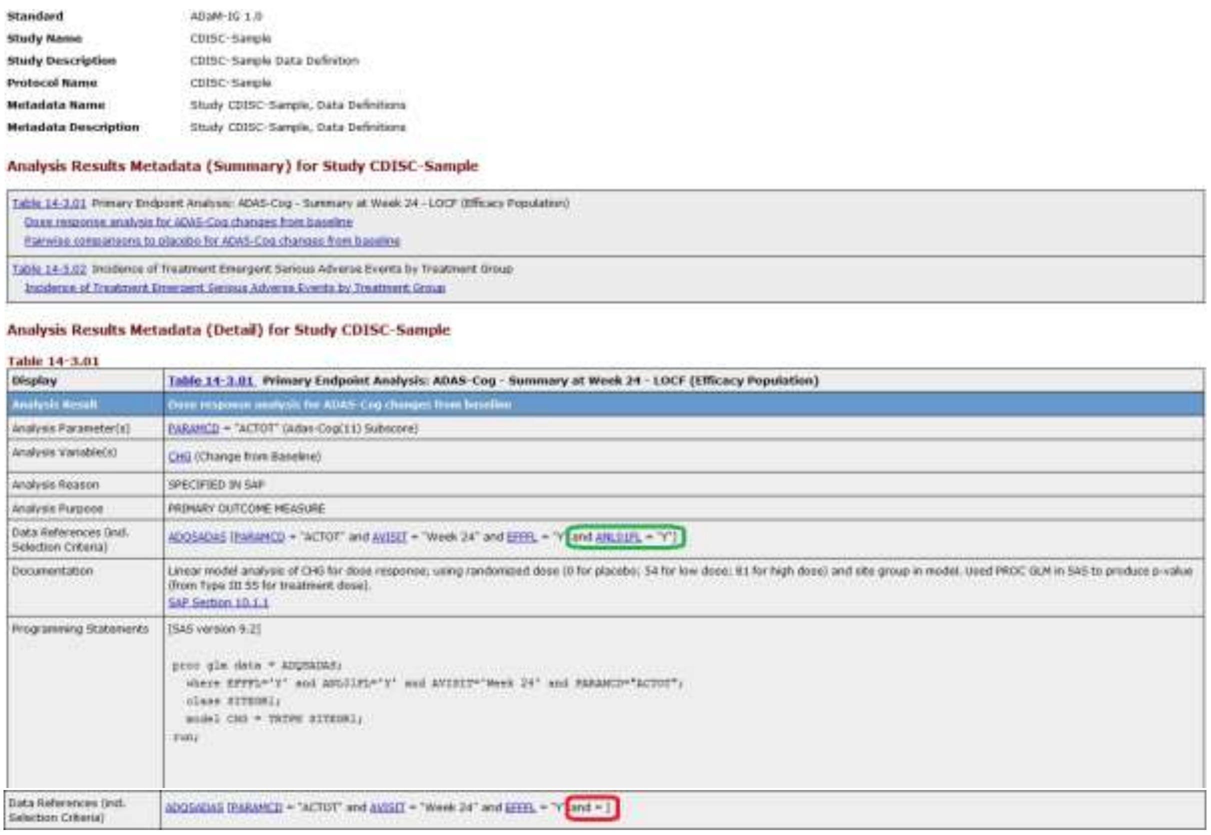
Out of the implementation specificities and issues, the major issue for a Define-XML is the loss of traceability. On Figure 6, a tree representing the links between the ARM elements and the Define-XML elements. From top to bottom, it is possible to retrace an element from the result to its root element in the main define.
When a link is broken, when a branch from the tree in Figure 6 is cut, the Define-XML is not able to link the element to its predecessor in the dataset metadata. In Figure 7, we represented a mapping of the Analysis Result Metadata in its first to its second version and an associated example (we represented a vertical mapping for the sake of readability, instead of the classic tabular presentation). In Figure 8, we have represented the result in the define.xml displayed by the stylesheet.

The examples showed below are all referencing the CDISC example available online for everybody.



Columns	Example	Relationship to specifications v2		Define-XML Regroup
DisplayIdentifier	Table 14-3.01	ResultDisplayOID		
		Name	defTitle	
		Lang		
xlinkDocument	dummy-csr.pdf	Type	xlinkHref	
Page(s)/Section	2	PageRefs		
		FirstPage		
		LastPage		Display
Description	Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)	Description		
		OID		
		DocumentsLeafID	leafID	
			ID	
ResultDescription	Dose response analysis for ADAS-Cog changes from baseline	AnalysisDescription		Analysis Result
Analysis Reason	SPECIFIED IN SAP	AnalysisReason		Analysis Reason
Analysis Purpose	PRIMARY OUTCOME MEASURE	AnalysisPurpose		Analysis Purpose
Dataset(s)	ADQSADAS	ItemGroupOID		not displayed as it
Analysis Variable	CHG	ItemOID		Analysis Variable(s)
Selection Criteria	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and ANL01FL = "Y"]	ParameterOID		Analysis Parameter(s)
		WhereClauseOID		
		defItemOID		
		Comparator		
		CheckValue		Data References (incl. Selection Criteria)
		SoftHard		
Join Description		defCommentOID	OID	
			Comment	
Documentation	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site group in model. Used PROC GLM in SAS to produce p-value (from Type III SS for treatment dose).	Description		
DOC-REF	SAP Section 10.1.1	DocumentationLeafID	defTitle	Documentation
			ID	
			leafID	
		Type		
Page(s)/Section	25	PageRefs		
		FirstPage		
		LastPage		
xlinkDOC-REF	dummy-sap.pdf		xlinkHref	
Programming Statement	proc glm data = ADQSADAS; where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT"; class SITEGR1; model CHG = TRTPN SITEGR1; run;	Code		Programming Statements
Programming Language and Version	SAS version 9.2	Context		
Program		ProgrammingLeafID		

Figure 7 Mapping data from ARM Specifications v1 Compatible to Define.xml v2, based on the example from CDISC Define.xml 2.0 with ARM. The relationship column links the version 1 to version 2 specifications. The second column of the relationship means that a link between two xml elements are created. The presence of more than one v2 element in the same relationship column means that the original element is used to create derived sub-element in v2 (i.e. Comment => [Comment ID, Comment] in comment metadata and Comment ID in the ARM).



Standard	ADaM-IG 1.0
Study Name	CDISC-Sample
Study Description	CDISC-Sample Data Definition
Protocol Name	CDISC-Sample
Metadata Name	Study CDISC-Sample, Data Definitions
Metadata Description	Study CDISC-Sample, Data Definitions

Analysis Results Metadata (Summary) for Study CDISC-Sample

Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)
Dose response analysis for ADAS-Cog changes from baseline
Pairwise comparisons to placebo for ADAS-Cog changes from baseline

Table 14-5.02 Incidence of Treatment Emergent Serious Adverse Events by Treatment Group
Incidence of Treatment Emergent Serious Adverse Events by Treatment Group

Analysis Results Metadata (Detail) for Study CDISC-Sample

Table 14-3.01

Display	**Table 14-3.01 Primary Endpoint Analysis: ADAS-Cog - Summary at Week 24 - LOCF (Efficacy Population)**
Analysis Result	Dose response analysis for ADAS-Cog changes from baseline
Analysis Parameter(s)	PARAMCD = "ACTOT" (Adas-Cog(11) Subscore)
Analysis Variable(s)	CHG (Change from Baseline)
Analysis Reason	SPECIFIED IN SAP
Analysis Purpose	PRIMARY OUTCOME MEASURE
Data References (incl. Selection Criteria)	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and ANL01FL = "Y"]
Documentation	Linear model analysis of CHG for dose response; using randomized dose (0 for placebo; 54 for low dose; 81 for high dose) and site group in model. Used PROC GLM in SAS to produce p-value (from Type III SS for treatment dose). SAP Section 10.1.1

Programming Statements	[SAS version 9.2]
	proc glm data = ADQSADAS;
	where EFFFL='Y' and ANL01FL='Y' and AVISIT='Week 24' and PARAMCD="ACTOT";
	class SITEGR1;
	model CHG = TRTPN SITEGR1;
	run;

Data References (incl. Selection Criteria)	ADQSADAS [PARAMCD = "ACTOT" and AVISIT = "Week 24" and EFFFL = "Y" and =]

Figure 8 ARM for Define.xml v2.0, CDISC example. In red, the traceability between a variable in a source dataset is broken. The variable mapped is not present in the source dataset. Therefore, the variable and its value are missing. In green, the representation when the variable can be tracked to its dataset origin.

The link to its dataset metadata cannot be retrieved, as circled in red on Figure 8. In green, the traceability is not broken, the element can be tracked to its origin in the datasets metadata.
The loss of traceability is due to mapping or completion issue, lack of consistency, mistakes, etc. Those can be minimized by stating the ARM specifications from the most recent version of the reference documents, and by cross-checking with the datasets metadata.

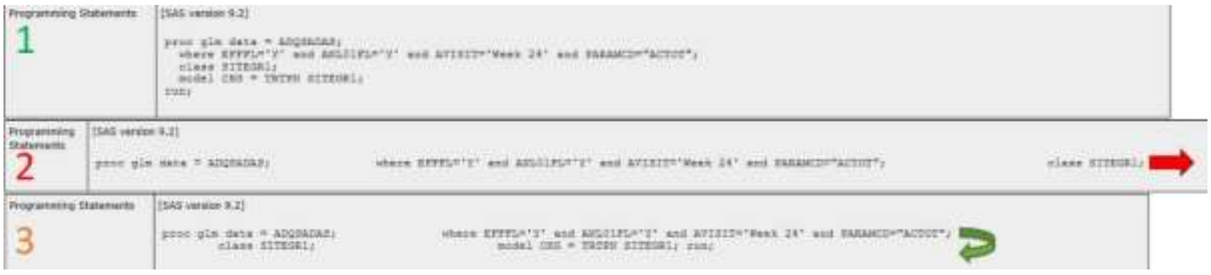
Issues related to the ARM Specifications:
Completion issues are very common. However, a rule is needed to complete some part of the ARM. All the constraints (required/optional, cardinality, order, and conditional completion) are coming from the ARM Specification document from CDISC (1). The completion rules are, of course, implementation dependant. We initially thought that it was straightforward enough to avoid any problem, and it is true for Descriptions, Comments, etc. Yet, during our "live" experience, mistakes were found on the Selection Criteria that could lead to wrong formatting and, loss of traceability. Hereunder, some examples of mistakes encountered.

We chose to keep our selection criteria format simple and close to the actual display in the xml document:

$DATASET_1$ [VAR_1 {Comparator} Value and … and VAR_n {Comparator} Value] ; … ;
$DATASET_n$ [VAR_1 {Comparator} Value and … and VAR_m {Comparator} Value].

With allowed comparators: *LT, LE, GT, GE, EQ, NE, IN, NOTIN,* <, <=, >, >=, =, ^=. Each "dataset group" separated by a semicolon, inside each group, each variable are separated by "and". Those instructions are then parsed by the program to create the *WhereClause* conditions list. Still, such expressions as *0 < AVISITN <= 52*, that is weighing the readability and might create wrong parsing or extra parsing work by the program, are unadvised; *(TRTA in ('A' 'B') as 'All Treatment' or TRTA in ('C') as 'All Placebo')* which is program oriented but not define readable is also unadvised*.* Usual mistakes are unbalanced quotation marks, unbalanced brackets for the group, usage of not allowed comparators, and usage of not allowed variable separator or group separator. This, despite the mapping recommendations.

The first enhancement we are foreseeing in the near future is the addition of *traceability checks* between ARM and datasets metadata into the converter.



```
[SAS version 9.2]

proc glm data = ADQSADAS;
  where EFFFL="Y" and ANL01FL="Y" and AVISIT="Week 24" and PARAMCD="ACTOT";
  class SITEGR1;
  model CHG = TRTPN SITEGR1;
run;
```

```
[SAS version 9.2]

proc glm data = ADQSADAS;              where EFFFL="Y" and ANL01FL="Y" and AVISIT="Week 24" and PARAMCD="ACTOT";              class SITEGR1;
```

```
[SAS version 9.2]

proc glm data = ADQSADAS;              where EFFFL="Y" and ANL01FL="Y" and AVISIT="Week 24" and PARAMCD="ACTOT";
        class SITEGR1;                          model CHG = TRTPN SITEGR1; run;
```

Figure 9 Example of display issue with too long lines in Programming Statements. 1 – Code snippet with good format, not too long. 2 – Code snippet badly formatted with no line-wrap, the text goes beyond the screen. 3 – Code snippet badly formatted but with auto line-wrap.

We also experienced some issues with the display, essentially in the programming statements (Figure 9). Usually, in the Define.xml, all texts are displayed independently of the carriage return, tabulations etc. All texts, but Code Statements.

Code Statements elements in ARM for Define-XML are displayed exactly as written. Programming Statements are code snippets, which means they are relatively small. Nevertheless, even small, those are usually out of the screen borders and we'll have to slide from left to right to read the complete line (red arrow, Figure 9). Which is not practical. To avoid this, one solution is to make the line's length from the code shorter, set smaller indentation spaces, add carriage return when needed. The second solution is to edit the stylesheet and add an auto line-wrap (green arrow, Figure 9).

An xsl stylesheet is formed from multiple templates, each one for a specific part to handle the xml layout and elements display. One of them serves for Cascading Style Sheets (CSS), a language that describes how each part of the xml and html should be displayed on screen. In the CSS, there is a "Code" section (Table 1), corresponding to the `<code>` tag in the xml file. That section has one particular property, white-space, which specifies how to handle the "blank" spaces (11). The white-space property needs to be set to:

Before	**After**

`.code{`
`    font-family:"Courier New", monospace, serif;`
`    font-size:1.2em;`
`    line-height:150%;`
`    white-space:pre;`
`    display:block;`
`    vertical-align:top;`
`    padding:5px;`
`}`
	`.code{`
	`    font-family:"Courier New", monospace, serif;`
	`    font-size:1.2em;`
	`    line-height:100%;`
	`    white-space:`**`pre-wrap`**`;`
	`    display:block;`
	`    vertical-align:top;`
	`    padding:5px;`
	`}`

Table 1 CSS modification in the xsl stylesheet of Define.xml v2.0 with ARM

This "pre-wrap" setting has the advantage to preserve the whitespaces, wrap the text when necessary, and on line breaks. The "pre" setting will wrap only on line break, which would give the case 2 as on the Figure 9.

PINNACLE 21 COMMUNITY VALIDATION

For each Define.xml 2.0 generated, it is well advised to validate both its structure and ensure the traceability to the data. Note that the community version before 2.2 (12) does not support ARM.

These checks consist in validating the structure, constraints, and consistency between data and metadata.

The checks severity are less restrictive for the PMDA than for the FDA. (13), considering the "Error" severity. However, the number of "Warning" and "Notice" severity are higher for PMDA (Table 2).

	P21 Severity	PMDA Severity
Error	108	61
Notice	1	13
Warning	14	31
(blank)	0	18
Grand Total	123	123

Table 2 Checks Severity count in Pinnacle 21 for Define.xml 2.0

CONCLUSION

In this paper, we tried to demonstrate the experience we had on the Define-XML with ARM. The most important notion with Define-XML and, particularly with Analysis Result Metadata is the Traceability.

The implementation we are using is one between many possible, but is well adapted to our own processes. Other implementations have been cited by Dilorio et al.(7), SAS (6) (8), or Santillan C. (9).

About the implementation. The first difficulty was the parsing of the information issued in the ARM specifications to a version compatible with the second version. The second difficulty was to create the arm elements in xml from template, and thus, generate a well formed xml file. For obvious reasons, this paper didn’t rely on any customer data. We chose on purpose to show examples from the CDISC data, for anybody to be able to reproduce the ARM to define.xml integration based on the mapping information provided.

During the implementation of ARM, we didn’t meet a lot of issues. Though, these “small” issues were important enough to make the Define-XML unclear and the data untraceable.

Most of the issues encountered are user related. The best way to avoid this would be a more intensive consistency check to identify that kind of issue as early as possible. Preferably at the study specification conversion level.

While the FDA is not officially asking for it, the ARM is already required by the PMDA. Indeed, ARM is a useful part of the deliverables. It links all together data, SAP, CSR, Programs, and of course, results. These results metadata have the ability to show what has been done on the data and link that to its root variables.

Implementing and using ARM was more direct than expected. We reduced the time spent on the Define-XML generation and we were able to make it easy to use. For the near future, BDLS is planning to incorporate the Study Specifications Converter to the Define-XML Generator. And if it not possible, make the user experience better by creating a user interface on top of the SAS Program. It is also planned to add internal consistency checks at converter level between datasets metadata and ARM.

REFERENCES

1. **CDISC ADaM Metadata Sub-Team.** *Analysis Results Metadata Specification Version 1.0 for Define-XML Version 2*. Clinical Data Interchange Standards Consortium, 2015.

2. *Current applications and future directions for the CDISC Operational Data Model standard: A methodological review*. **Hume, Sam, et al.** 2016, Journal of Biomedical Informatics, Vol. 60, pp. 352 - 362. 1532-0464.

3. **Ando, Yuko PhD.** *PMDA Update*. 2016.

4. **Kitahara, Takashi and Nakajima, Yuichi.** *Japanese Electronic Study Data Submission in CDISC Formats*. 2016.

5. *Electronic Study Data Submission; Data Standards; Support End Date for Case Report Tabulation Data Definition Specification Version 1.0*. **Food and Drug Administration.** 2016, Federal Register, pp. 14450-14451. 2016-05958.

6. **SAS.** CDISC Analysis Results Metadata 1.0 for Define-XML 2.0. 2016.

7. **Dilorio, Frank and Abolafia, Jeffrey .** *Results-Level Metadata: What, How, and Why*. PhUSE, 2015.

8. **Lex Jansen, SAS Institute Inc.** Creating Define-XML version 2 including Analysis Results Metadata with the SAS® Clinical Standards Toolkit. 2014.

9. **Santillan, Carla.** *Analysis Result Metadata... are we there yet*. 2016.

10. **CDISC Define-XML Team.** CDISC Define-XML Specification Version 2.0. 2013.

11. **W3 Schools.** CSS white-space Property. *w3schools.com.* 2017.

12. **Pinnacle 21.** Pinnacle 21 Downloads. *Pinnacle 21.* 2016.

13. —. Define-XML Validation Rules. *Pinnacle 21*. 2016.

ACKNOWLEDGEMENTS

I wish to thanks my colleagues who generously provided their insight and perspective, that greatly helped to improve the quality of this paper: Jean-Michel Bodart, Leroy Ovbude, Joke De Wever, Pascale Szternfeld, Nick De Donder, Mamdouh Ashmawy, Jamila Bhiri.

CONTACT INFORMATION

Contact the author at:

Lionel Debecq
Business & Decision Life Sciences
Rue Saint-Lambert 141
1200 Brussels (Belgium)
Phone: +32 478 52 19 91
Email: lionel.debecq@businessdecision.be
Web: http://www.businessdecision-lifesciences.com

Roxane Debrus
Business & Decision Life Sciences
Rue Saint-Lambert 141
1200 Brussels (Belgium)
Phone: +32 475 95 35 00
Email: roxane.debrus@businessdecision.be
Web: http://www.businessdecision-lifesciences.com