PharmaSUG 2013 - Paper IB04

Map Metadata – Going Beyond the Obvious/Connecting the Dots

Gregory Steffens, Novartis Pharmaceuticals, East Hanover, NJ, USA
Praveen Garg, ICON Development Solutions, Hanover, MD, USA

ABSTRACT

Much attention has been paid to the design and use of metadata to store data standards, study data specifications and creating the industry-standard metadata, i.e. the define.xml file. But not as much attention has been focused on the design and use of metadata to describe the transformations and derivations of data. We must expand beyond the focus on the stopping points of data flow and concentrate on the data movement from one stopping point to another. This paper describes the need for an industry standard for map metadata and presents a design that has been implemented with great success. The metadata design principles (described in earlier papers by Gregory Steffens and others) apply to map metadata, such as - 1.) Map metadata should not assume any one data standard, 2.) Well-designed map metadata supports meta-programming. Mapping information is often just put into free-form text fields, but putting this information into structured metadata enables meta-programming of data flows. Map metadata has many advantages, besides automating data flow; such as data flow transparency; standardized ways to exchange specifications about how data flows from one structure to another, as in SDTM to ADaM to IDB; and creating a target metadatabase from a source metadatabase and map metadata, such as creating a study specification from a data standard. Map metadata defines the relationship between the source and target database at dataset, variable, row and value level. Map metadata, along with source and target metadata, can enable automation of the dataflow and create transparent define files, with transformation logic as well as database attribute descriptions. This is the next evolution of metadata and of meta-programming, leading to a true Data Transformation Engine (DTE). Automation at this level is essential to meet the efficiency and quality goals in today's environment, which requires us to do more work with less staff and to improve quality at the same time.

INTRODUCTION

The pharmaceutical industry is a data-analysis-driven industry and it has gone through a transformation phase in last few years. Patent expiration has led to lot of pressure with reduced budgets and spending. Clinical Research related activities have not been unaffected by these spending cuts and there is more focus on doing more with less. There is lot more focus on data standards; data analysis standards and data presentation standards. Another transformation which has happened is an increase in outsourcing data and reporting related functions. These changes have led to lot more interaction and communication of technical details between pharmaceutical companies and its partners in the clinical research space. CDISC has played a key role in developing data standards for planning (protocol and study design); Data collection (CDASH and LAB); data tabulation (SDTM and SEND); statistical analysis (ADaM) and a metadata standard (define.xml). Many Pharmaceutical companies and CROs have adopted these data standards, as part of the clinical data flow, to support standard process around data creation and review. This trend has been supported by the FDA approval of these standards (SDTM in particular for data tabulation as part of submissions). Initially these industry and corporate standards were stored in .pdf or MSword or unstructured excel documents. As such they were accessible only to people reading the documents rather than to software programs, and did not include a standardized set of attributes. More of us are using metadata now although not standardized metadata. We need now to look to the next step of storing data flow (mapping) in metadata instead of unstructured, free-form text. Storing mapping information in standardized metadata sets adds more value to the effort put into defining these standards because this information is now available to computer programs. The move to store data standards in structured format which can be read by computer programs has picked up over last few years. This has helped in data requirements sharing across different organizations.

The objective of this paper is to expand beyond the definition of data standards and specifications in metadata format and discuss how to define the mapped transformations and derivations between 2 databases in the clinical data flow in metadata format. Similar to the early implementation of defining data standards in pdf or MsWord and unstructured excel documents; data flow details like derivations, transformations, merging, etc. is typically defined in an unstructured format accessible only to human readers rather than software programs. The focus of this paper is to define how this mapping information can be defined in a structured metadata format and what type of efficiencies can be gained using this approach. This new kind of metadata is called map metadata throughout this paper (MAPMD), to distinguish it from database metadata (DBMD). Map metadata contains the definition of linkages between the 2 database metadatabases, hence map metadata is used in conjunction with database metadata that describes the source and target databases.

DATABASE METADATA

The following table describes the 5 database metadata sets that describe any data standard or project-specific database specification. This is database metadata, DBMD.

Metadata Set Name	**Purpose**	**Primary Key Metavariables**
TABLES	Describes data set level attributes	TABLE
COLUMNS	Describes variable level attributes	TABLE COLUMN
COLUMNS_PARAM	Describes row-level parameter related variable attributes in tall-thin data sets	TABLE COLUMN naming parameter variable PARAM parameter variable value PARAMREL variable name being described
VALUES	Creates controlled terminology lists, valid values lists and code lists that can be associated with a COLUMN or a PARAMREL, by name of the value list.	FORMAT (the name of the value list) START END
DESCRIPTIONS	A SAS® catalog of source entries that contain derivation descriptions, methods and comments that can be associated with the specification, a data set, a variable, of a parameter related variable.	DESCRIPTION (the name of the description)

Table 1. Database Metadata Structure

These metadata, along with metaprogramming (e.g. SAS macros) and a GUI (e.g. SAS/AF) application that access these metadata, supports the specification, creation, importation, exportation, comparison and validation of databases and format catalogs. The macros support functions such as:

- publishing the specification in several formats, including the define.xml and define.html formats
- implementing all the data set and variable attributes to the database with a simple macro call (variable name, type, length, label, format, etc.)
- listing discrepancies between the data specification and the database, including structure and valid values
- sorting the data by primary keys
- reordering variables in the PDV according to FDA preferences
- creation of user defined format catalogs
- creation of character decode variables from code variables or vice versa
- copy header variables to all the datasets in the database based on primary keys
- shorten the length of data set and variable names for transport file creation for eSubmission
- comparison of database structures to standards or to other studies, to assist in enforcing data standards and consistency

THE NEED FOR A MAP METADATA STANDARD

But the DBMD, as powerful as it is, has limitations. It cannot specify a variable to be renamed; or a transformation of a variable to a target data set with different key variables than the source data set; nor a target variable the requires one or more source variable with a derivation logic or formula applied. To expand beyond these limitations, we need MAPMD that specifies how each target variable is created from the source variables. We should not all design our own map metadata because then we will not be able to exchange the mapping information outside our companies. A CDISC map metadata standard design will greatly enhance what we have now and expand the scope of metadata.

Map metadata is an extension of the database metadata model that describes databases. This extension defines the relationships between two metadatabases. Database metadata describes databases; map metadata describes the relationships between metadatabases and their corresponding databases. Once these relationships are populated in map metadata, SAS macros, that constitute the data transformation engine, can read this and generate code that automates the transformations of the source database into the target database. Examples include the creation of target SDTM from the source CRF database, the creation of ADaM from SDTM and the creation of integrated databases from individual study databases.

There is automation that can be attained related to the transformation of the contents of a source database into a differently-structured target database. This includes data transformations from a simple variable rename to a complex rekeying of the data. The objective here is to automate data transformations with SAS macro code that does not make assumptions about the data it is transforming. This kind of automation does not make assumptions that a study data standard is complied with, the automation assumes only that the metadata describing the source and target data sets exists and that a new kind of metadata is created – map metadata.

Map metadata can also subset a DBMD, e.g. creating a study DBMD from a standards DBMD and MAPMD to the study or creating a therapeutic area specific data standard from a more general scope data standards DBMD. The sub-setting can be accomplished because the MAPMD contains a list of the subset of the source DBMD rows that are required for the target DBMD.

Map metadata can also enhance the content of a define file, by including the relationships between source and target DBMD as well as the code for mapped derivations. The map metadata includes, for example, the fact that height and weight are used to derive BMI and the formula for deriving BMI. This information can be put into the define file and not hidden in SAS code or data requirements document.

An industry-standard map metadata design is required so that we can exchange the critical information about the relationships between databases in structured metadata. We need to get the mapping information out of free-form text and non-standard mapping data sets. Creating an industry standard will allow pharmaceutical companies to communicate more information about data flow to programming staffs internally, in an outsourcing scenario and to regulatory agencies. The define.xml should be enhanced to include standard mapping information from a source define file to a target define file by mapping the OIDs of source objects to OIDs of target objects. Data transparency and data flow automation can be achieved only with standardized map metadata added to source and target DBMD.

MAP METADATA DESIGN

One of the common mistakes in designing database metadata is mixing the information about database and map metadata in the same metadatabase, i.e. mixing up DBMD and MAPMD. This leads to multiple disadvantages and problems. One of the problems in combining DBMD and MAPMD is that you are narrowing the use of the metadata to one data flow. There is an assumption that data will always flow from collection to SDTM to ADaM, for example, and the scope of application of the metadata is thus unnecessarily narrowed. Using the triad of source DBMD, MAPMD and target DBMD allows the metadata and metaprogramming to be used with any data flow and greatly expands the scope of its application. Always expect the unexpected and support the unexpected data flow in your metadata and metaprogramming design. Another of the problems is difficulty to create, maintain and use metadata if the database standards and data flow is stored in 1 metadatabase. Let's take an example where database and map metadata standards are stored in 1 metadata. In this example, let's assume that we have 3 sets of databases in the dataflow (raw, SDTM and ADaM). Now to create these standards, you will need a SME who has knowledge of data collection, SDTM and ADaM standards and the transformation and derivations involved from raw → SDTM and SDTM → ADaM. This skillset is really difficult to find in 1 person. A benefit of defining the database and map metadata in different set of metadata is that you can create map metadata between different set of database metadata. You can break the metadata management into DM focused data standards and map metadata and Statistical analysis focused data standards and metadata. It makes it easier to create and maintain metadata without the complexities defined earlier.

The following table describes the map metadata structures that correspond to the above-described database metadata sets.

Map Metadata Set Name	**Purpose**	**Primary Key Metavariables**
TABLES_MAP	Defines the order source data sets should be run in when more than one source data set contributes to a single target data set.	SOURCE_TABLE TARGET_TABLE
COLUMNS_MAP	Maps 1 or more source variables to a target variable, as defined in source DBMD and target DBMD.	SOURCE_TABLE SOURCE_COLUMN TARGET_TABLE TARGET_COLUMN
COLUMNS_PARAM_MAP	Maps 1 or more source variables or 1 or more parameter-related (or value-level) variables to a target variable or parameter-related variable, as defined in source DBMD and target DBMD.	SOURCE_TABLE SOURCE_COLUMN SOURCE_PARAM SOURCE_PARMREL TARGET_TABLE TARGET_COLUMN TARGET_PARAM TARGET_PARMREL
VALUES_MAP	Maps codelist values from a source to a target value. All variables associated with this codelist mapping will have their values changed from the source value to the mapped target value.	SOURCE_FORMAT SOURCE_START SOURCE_END TARGET_FORMAT TARGET_START TARGET_END
INCLUDE_MAP	A SAS catalog containing software syntax that implements the formula to be used to create a derived variable. The DTE and map metadata brings together all the variables required for the derivation code to execute, by merging mapped data sets by shared primary keys defined in the DBMD.	The SAS catalog has named source entries and the names are referenced as a foreign key from tables_map columns_map and columns_param_map.
DESCRIPTIONS_MAP	A SAS catalog containing an optional description of the mapping in plain language.	The SAS catalog has named source entries and the names are referenced as a foreign key from tables_map, columns_map and columns_param_map.

Table 2. Map Metadata Structure

Map metadata makes the assumption that the source and target databases are described with the standard metadata, DBMD. Map metadata then supplies the definition of a relationship between each observation in the target metadatabase with any one or more observations in the source metadatabase. For example, any variable in the source can be mapped to the target. Variable and paramrel values can be mapped in the values_map, to generate all the code that changes the values of the source database to conform to the target database requirements. If you consider implementing source and target DBMD as define.xml files then the map metadata becomes pairs of OIDs that map an ItemDef in the source DBMD to an ItemDef in the target DBMD. The same OID mapping can be done for value level (what I call row-level) metadata and codelists.

The value-level metadata is recently addressed by the new version 2.0 of the define.xml schema. This adds a dimension of metadata that is necessary to the objectives of map metadata. For example, to map a weight variable in a short-wide data collection data set to a TESTCD/STRESN variable in a tall-thin data set, it is necessary to define the value-level metadata in the tall-thin data set so that there is an adequate target OID. This means that all the CDISC data standards need to be enhanced to include value-level metadata. Without this, transformations between short-wide and tall-thin data sets will not be able to be represented in MAPDB, since the source or target DBMD is incomplete. We need the virtual ItemDef OIDs to map to in the tall-thin data set. Controlled terminology that merely defines values of the TESTCD variable is inadequate.

AUTOMATION ENABLED BY MAP METADATA

Now let's talk about the automation which can be achieved from the use of DBMD and MPMD. The automation will be called DTE. The DTE macros determine the primary key variables in the source and target databases to determine how to copy the value of the source variable into the target database variable. The DTE assigns a transformation type based on the relationship between the source data set primary keys and the target data set primary keys. These categories are: same, superset, subset, supersub, wide2thin, thin2wide and none.

Primary Key Relationship Category	**Description of Category**
SAME	Source and target are keyed the same so a simple merge is done
SUPERSET	Target has a superset of key variables as compared to the source. A merge of the common key variables is done.
SUBSET	Target data set has a subset of the key variables as compared to the source. Each mapped target variable must create an array in the target.
SUPERSUB	The target data set has both more primary key variables and less than the source data set. A transformation is not supported here because the primary key relationship does not allow it.
WIDE2THIN	The target data set is tall-thin data structure and has one extra primary key as compared to the source data set. For example, a source vitals data set with one variable per vital sign and unit mapped to a target data set with the VSTESTCD primary key variable added. A short-wide to tall-thin data transformation is done.
THIN2WIDE	The target data set is short-wide data structure and has one less primary key as compared to the source data set. A tall-thin to short-wide data transformation is done.
NONE	The target data set has no common key variables with the source. A transformation is not done.

Table 3. Primary Key Relationships

Map metadata is far simpler in structure than the source and target metadata. Map metadata supports the definition of the relationships between metadatabases by including the primary key metavariable values of the source and target metadatabases. For example, if the metadatabase includes a metadata set named "columns" that includes primary key metavariables named "table" and "column" then the map metadata simply contains the variables "source_table", "source_column", "target_table", and "target_column". This simple 4-metavariable map metadata structure allows you to select any one observation in a source metadatabase and associate it with any one observation on the target metadatabase. There is complexity in the macro that reads the map metadata and generates the data transformation code, but populating map metadata is relatively simple.

The macros using map metadata support functions such as:

- publishing the specification in several formats, including the define.xml and define.html formats (with detailed data processing information)
- create study data requirements by programmatically copying subsets of the standard metadata to a study or existing study metadata to a new study.
- transform data from wide to thin and vice versa
- create integrated databases from individual study databases
- Transforms a database from one structure to another, with the DTE and map metadata

THE ADVANTAGES

The advantages of metadata approach are to attain the goal of a highly consistent level of quality with less resources and cost. That is, to do our work faster, better and cheaper. How would you like to create an SDTM – compliant database with code like this?

```
%dtmap(source_mdlib=m,source_prefix=raw_,target_mdlib=m,target_prefix=target_,
 maplib=me,inlib=raw,outlib=sdtm,suppqual_make=yes)
```

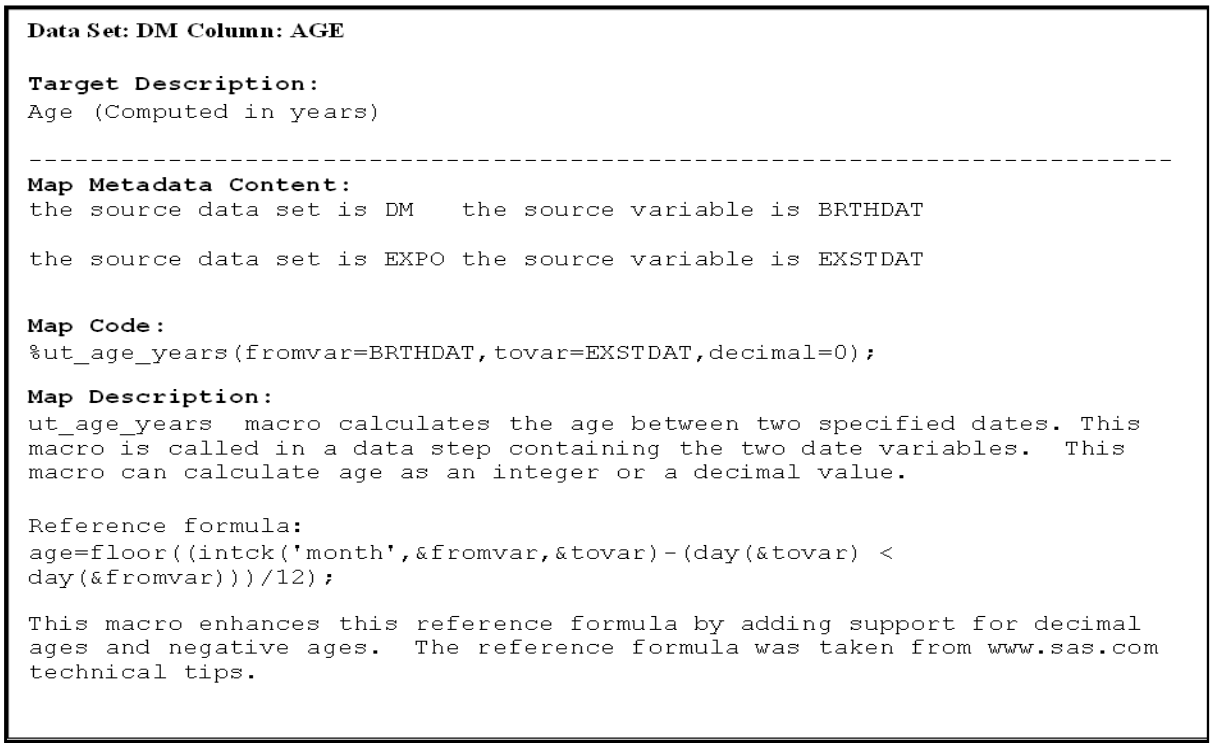
Or publish your data requirements with code like this?

```
%mdprint(mdlib=m,inprefix=sdtm_,maplib=m,mapprefix=raw_sdtm_,
html=yes,htmlfile=c:\metadata\target_define.html)
```

Or create the define.xml file with code like this?

```
%md2odm(mdlib=m,
 outxml=c:\metadata\SDTM.xml,
 StudyName=ICR_STDM_Standard 3.1.2,ProtocolName=SDTM,
 StandardName=SDTM 3.1.2,StudyDescription=SDTM Standard,
 defineVersion=1.0.0,odmversion=1.2,crt_prefix=def)
```

Or description in the define file for a derived variable like this?



```
Data Set: DM Column: AGE

Target Description:
Age (Computed in years)

---------------------------------------------------------------------------
Map Metadata Content:
the source data set is DM   the source variable is BRTHDAT

the source data set is EXPO the source variable is EXSTDAT


Map Code:
%ut_age_years(fromvar=BRTHDAT,tovar=EXSTDAT,decimal=0);

Map Description:
ut_age_years  macro calculates the age between two specified dates. This
macro is called in a data step containing the two date variables.  This
macro can calculate age as an integer or a decimal value.

Reference formula:
age=floor((intck('month',&fromvar,&tovar)-(day(&tovar) <
day(&fromvar)))/12);

This macro enhances this reference formula by adding support for decimal
ages and negative ages.  The reference formula was taken from www.sas.com
technical tips.
```

Figure 1. Detailed Description Example

Standards are the means to automation and automation is the means to improved efficiency and quality. But you may ask about the amount of work required to populate the metadata. The macro calls, above, look easy enough, but how difficult is it to populate the metadata and how does that workload compare to an environment without this automation? This is where good process is required to ensure that the quality and efficiency objectives are attained. Other points to consider are study-level validation, which is greatly reduced by automation, and a change in required skill sets to create quality deliverables.

The advantage of writing less code to create the deliverables is clear. It is both inefficient and riskier to invent or even modify code for each study. The risk is to quality. Newly created or modified code has a higher risk of error and requires more validation as it is used in each study. Reusable code that is used in multiple studies vastly reduces the amount of new code that needs to be created. Reusable code should be validated independently of any specific project. This multi-use validation is more complex than the single-use code invented for a specific project, but after about three studies the validation effort comes out even and the return on investment really begins.

The reuse principle is applied to metadata too. The content of the metadata can provide information to multiple documents that are typically entered separately. Each kind of information should be entered once and used many times. This increases the value of the metadata content. Data requirements in metadata can be published in many documents and used in SAS programs instead of re-entering that information into multiple documents. We should move away from entering information into documents and towards entering that information into structured metadata and generating documents from the metadata. This facilitates the "enter once, use many" principle and ensure consistency across documents, since there is one source of the information in all those documents.

It is also easier to get a more consistent quality, even from junior staff, with automation. The automation is controlled and defined by the most experienced staff and the business rules are implemented in macros. Junior staff, calling these macros, generates better deliverables because these business rules are implemented in the macro and the learning curve is greatly reduced. Metadata data can be populated in a number of ways and by staff with different skill sets, as compared to a non-metadata environment. Without automation the skill set required to transform a database include SAS programming, database design and knowledge of clinical data. With automation new divisions of labor become available. Someone familiar with clinical data can populate metadata and a SAS programmer can use that metadata when creating, validating and transforming the data. The same principle is helpful during the creation of Integrated databases. The map from different database structures can be defined to Integrated database structure with the use of map metadata. The biggest advantage is the review of submission packages by Regulatory agencies. There is an increasing trend of FDA asking for SAS code used to create the databases. The main reason behind that is lack of transparency of data processing in the define files. With the map metadata approach, the information embedded in the SAS codes is transparent in the define files. If the review of SAS codes can be avoided through use of more informative define files then the review cycle for submissions may reduce leading to significant revenue increase for approved drug (each day saved in drug approval is worth millions of dollars in revenue).

The approach defined above has been implemented with great success in automation of SDTM submission packages. The process of creating the define files after SDTM creation is reversed. The creation of define file is instantaneous with the use of metadata and associated automation. This gives the programmers clear instructions and also ensures that the define file and programming instructions are same. SDTM creation effort is reduced significantly as all the transformations are performed by the automation. The effort is primarily spent in looking for the data anomalies leading to high quality data submission.

CONCLUSION

We have previously shown that innovative design of metadata; definition of database and map metadata in standard metadata structure and an associated SAS macro library can deliver automation of a large portion of data flows, during most phases of the database creation process, from specification through build and on to validation and delivery. The standardization of metadata and the automation that it enables, can be done in a way that is data standard neutral. This protects your investment, of the time to create the automation, from the inevitable changes in data standards. In fact, standardized metadata has a larger positive and more long-lasting impact, as compared to standardizing data. This impact has to be measured against of clearly defined objectives.

This paper shows that the next step, after establishing clinical data standards and database metadata, is to create an industry-level standard for map metadata to enable the representation of data flow in structured metadata. Moving mapping information out of unstructured text and into structured metadata will enable even more metaprogramming, that will further automate the creation and description of databases to support eSubmission, integrated or pooled databases and reporting.

REFERENCES

http://support.sas.com/resources/papers/proceedings09/171-2009.pdf
http://www.phusewiki.org/docs/2010/2010%20PAPERS/CD08%20Paper.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Gregory Steffens
Novartis Pharmaceuticals
Work Phone: 862-778-6550
Email: Gregory.steffens@novarits.com
LinkedIn: http://www.linkedin.com/pub/gregory-steffens/a/bb4/6ab/

Praveen Garg
ICON Development Solutions
7740 Milestone Pkwy,
Hanover MD 21076
Work Phone: 410-696-3218
Email: Praveen.Garg@iconplc.com
LinkedIn: http://www.linkedin.com/pub/praveen-garg/b/3b4/a84