PhUSE 2015

Paper AD07

Automated generation of program templates with metadata based specification documents for integrated analysis

Thomas Wollseifen, inVentiv Health Clinical, Eltville, Germany

ABSTRACT

Delivering ADaM (analysis data model) datasets as part of integrated analyses is an important part of the process of preparing clinical study data submissions to medical authorities (e.g. FDA/EMA).
Conforming with statistical specifications such as Statistical Analysis Plans, Data Handling Documents and ADaM Implementation Guidelines, ADaM datasets are usually generated via SAS derivation and transfer programs that incorporate metadata, codelists and other standards. The program that includes all derivations to map the study data to the integrated data structure in the ADaM analysis dataset is usually created by hand for each domain (ADSL, ADAE, ADDS, etc.).

The process of writing transfer files could be automated using a specification file based on the metadata of the integrated database, the study data structure, and derivation rules.

This paper discusses an approach for creating a specification file which includes all derivations and data mapping rules. This specification includes metadata and describes rules in the form of text descriptions as wells as SAS program segments or templates. A macro subsequently creates SAS program templates for each domain based on the derivations in the specification. These program templates can then be adapted or extended as needed.

One of the advantages of this semi-automatic process is that the generated program files comply with industry and company standards and also accord with good programming practice. Moreover, the process of study data transfer can be more automated and made more transparent.

INTRODUCTION TO DATA INTEGRATION PROCESS

Integrating and analyzing data of multiple studies across a compound can be a challenging task. The Clinical Data Interchange Standards Consortium (CDISC) has developed different clinical data standards to facilitate efficient transfer, access, review and analysis of clinical trial data. These standards include the Operational Data Model (ODM), the Study Data Tabulation Model (SDTM) and the Analysis Data Model (ADaM).
In Fig. 1 the data flow of collected data of a clinical study from the CRF (case report form) or EDC (electronic data capture), laboratory data and other device data is displayed. From the captured source data, data management creates an Operational Acquisition Database (OAD) for later use. The OAD is then transformed into SDTM datasets.
Derived variables and records additional subgroups and flags are then created in ADaM data. ADaM data is then used by the statistical analysts to prepare Tables, Listings and Figures (TLFs) for a clinical report. For meta analyses SDTM or ADaM datasets could also be mapped into an integrated database (IDB) of a specific compound from which integrated summaries of safety (ISS) and integrated summaries of efficacy (ISE) are prepared for a submission.

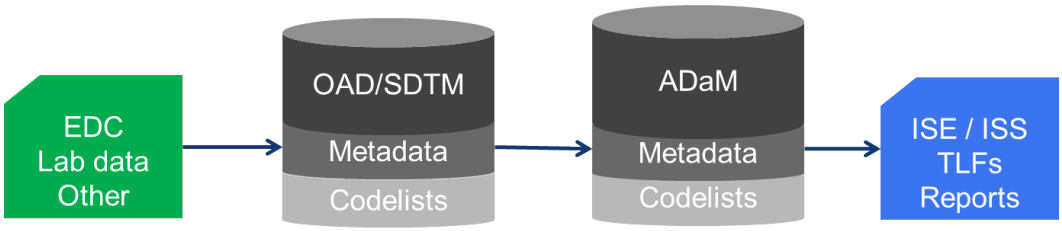

Fig. 1: Data flow from clinical study to database and resulting reports

This paper presents a semi-automatic method to map or integrate data of different studies of a compound into an integrated database. The data source can be any data standard, also company standards. A mapping tool will be

presented that will read the structure of a study database and find mapping solutions which fit to the metadata of an integrated database.
The following figure Fig. 2 shows a typical process of mapping data of different clinical studies into one structure – an integrated database (IDB). The mapping programs are usually written manually based on specifications (Clinical Study Protocols/Statistical Analysis Plans (SAP) / Data handling documents). To analyse the data of an integrated database, the data should be in an ADaM format – one step away from analysis. All derived variables should be already in the integrated database. The integrated database usually contains different ADaM domains (ADSL, ADAE, ADEX, ADCM, etc.).

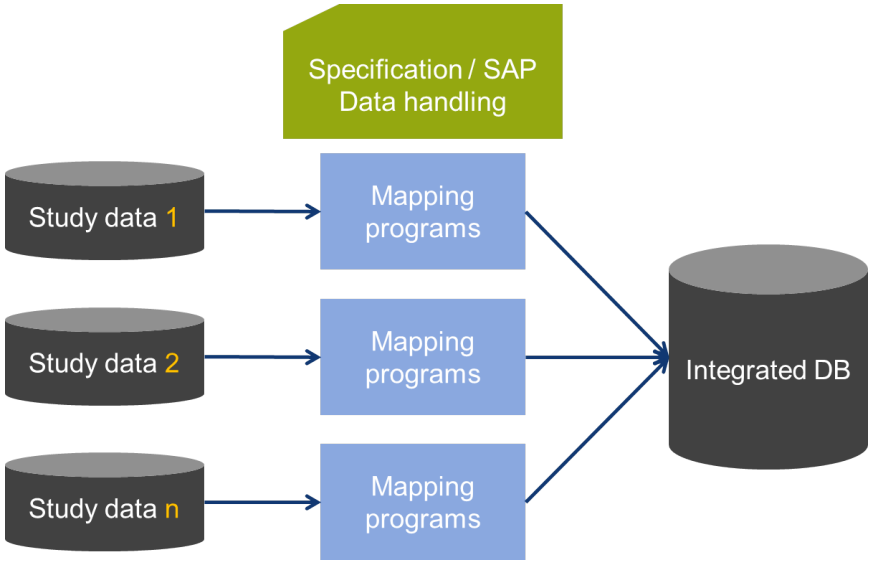

Fig. 2: Mapping of multiple clinical studies into one integrated database

In the following paragraph the process of semi-automatic specification generation based on study data and metadata of the IDB is explained.

HOW TO MAP CLINICAL STUDY DATA INTO AN INTEGRATED DATABASE

A basic idea of the mapping one structure into another structure is a specification which includes all rules and definitions of how to map a structure **A** to a structure **B**.
In clinical data, especially in SDTM or ADaM data structures, specific domains with defined structures exist. In an integrated database of a compound the metadata describes the structure of the database. Usually, the metadata or the definition of the database contains rules like imputations, definitions of the visit structure, derivations of specific variables, or definitions for summarizing records (e.g. compliance or summaries of drugs taken). These derivations or rules could be different for other compounds.

Let us now discuss a semi-automatic generation of a specification based on the metadata of the IDB, the clinical study and general rules for a specific compound as described above. The following Fig. 3 shows the overview of the specification generation and mapping process of SDTM data into ADaM of the IDB by SAS macros.
Later, the relevant SAS macros used to create a specification and the mappings programs (template programs) will be described. The semi-automatic approach we will observe more closely has three steps:

1) Create a specification: **%m_create_specification**
2) Fill missing derivations **manually** based on the SAP/data handling/Derivation rules
   The specification is also reviewed and approved by a second programmer / Statistician
3) Create mapping programs **%m_create_mapping**

First the SAS macro **%m_create_specification** creates an Excel file. This is the specification for the mapping. Of course, derivation rules and definitions from the SAP or data handling document will either be inserted manually or taken from a special definition dataset which includes general rules / derivations in the current project.
After the specification Excel file is generated, additional manipulations or derivations need to be added. This process is usually a manual process. A statistical analyst inserts necessary derivations. These derivations will be a description of the derivation and SAS code or SAS macro calls. A statistician or a statistical analyst responsible for the IDB usually approves the specification file to be used in the following mapping process.

In our case as third step a SAS macro **%m_create_mapping** reads the specification file and creates program files (template programs) for all domains of the clinical study to be mapped to the IDB. The templates follow company programming guidelines and adhere to program templates (e.g. program header with correct description of the program and all necessary information) used in the company. All derivations given in the specification Excel file are transcribed into SAS code. Comments will be automatically inserted before a derivation / macro call.

Indentation is also inserted into the mapping programs by the %m_create_mapping macro.

If additional changes or specific modifications of the derivation / mapping are necessary, the program templates could be changed later by a statistical analyst. Before the mapping programs will run in the production / final environment a validation step is necessary. Each automatically generated mapping program will be tested and run in a test environment – and finally validated.

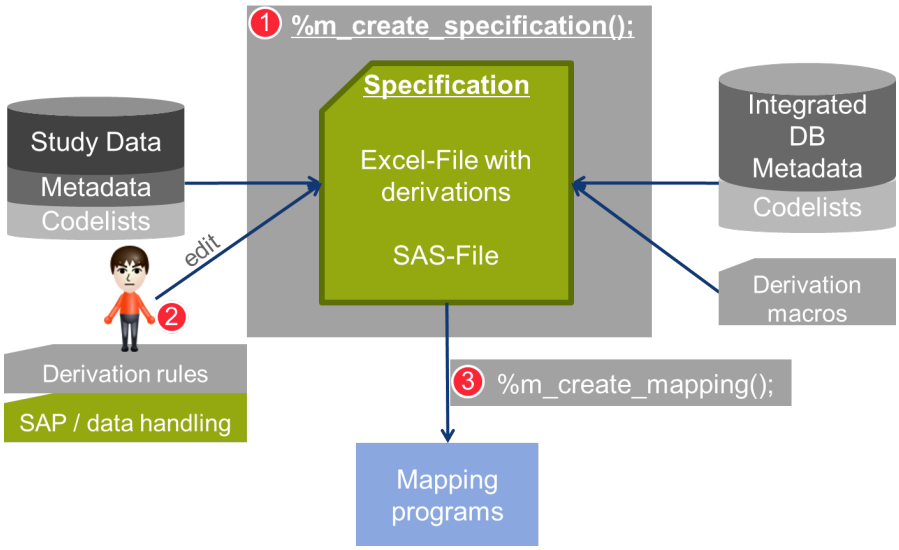

Fig. 3: Overview - Creation of mapping programs based on a semi-automatic specification document

In the following paragraphs we will be looking on the different parts of the specification and mapping process

CREATE SPECIFICATION

First we will be examining the generation of the specification. The macro %m_create_specification will receive all necessary information: the project name, the study ID, the domains to be mapped, information about where to find the study data (library names), the metadata of the IDB and a path and filename of the resulting Excel file.

Fig. 4 shows the header of the macro %m_create_specification. Later all relevant steps of the macro are described

```
%macro m_create_specification(   project  =          /*Project name                          */
                               , studyid  =          /*Study ID                              */
                               , domain   =          /*Domains to map (ex. A->B)             */
                               , path     =          /*Path to output specification          */
                               , filename =          /*Excel filename                        */
                               , metalib  =          /*Metadata of IDB                       */
                               , inlib    =          /*Study data                            */
                               , adslbase =adsl      /*ADSL base (could be DM for SDTM->ADaM)  */
                              );
```

Fig. 4: Definition of macro %m_create_specification

The macro **%m_create_specification** creates an Excel specification to map study data according to the integrated database metadata.

In the macro definition a project name (e.g. a therapeutic area or a drug) can be specified. It is necessary to specify information about the IDB (library of the IDB), the study (study ID, library of the study data) and specific information about the output file (path, name). Important is also to specify which datasets of the study should be mapped to which data domain of the IDB.

The definition of the mapping adheres to the rule: A->B (map A to B, e.g. DM->ADSL# AE->ADAE). In the macro call different domains are separated by a '#' symbol.

The basic rule how a domain dataset is mapped is described below:

Domain= <domain_A> **->** <domain_B> **#** <domain_C> **->** <domain_D>

The following Fig. 5 shows an example of a %m_create_specification macro call, where study data in SDTM structure is mapped to ADaM.

```
/*****************************************************************************/
/* Example call SDTM --> ADAM */
/*****************************************************************************/
%m_create_specification(
                    studyid=99999
                  , domain=ae->adae#dm->adsl#ex->adex#lb->adlb
                  , path=&path
                  , filename=9999_specification.xls
                  , metalib=meta
                  , inlib=in
                  , adslbase=dm);
```

Fig. 5: Example call to create mapping specification from SDTM to ADaM

If a wrong domain dataset name is put into the macro call, or libraries do not exist, the macro %m_create_specification returns warnings for the relevant and critical macro parameters.

In the next paragraph we will have a look on the different steps of the macro %m_create_specification. It consists of the following main steps:

1. Check all study domains against metadata of IDB
2. Create specification Excel
    a. Write domain description (domains/key variables / sort order)
    b. Write pool metadata and study metadata + derivations to Excel

The macro reads the study data domains and checks the information of study structure against the metadata of the integrated database. It goes through a loop of all mappings given in the domain macro variable.

An integral part of the **%m_create_specification** macro is a check macro **%mcheck_variables_metadata**, which merges information form the study structure of each domain with the information of the IDB metadata domains and checks study metadata and data against the metadata of the IDB. This step is very important for the resulting specification, because all important structural elements of the study data and metadata are compared to the metadata of the IDB and the information is then used for specification.

In the check macro all relevant information of the domain metadata are checked and compared:

- Variable name
- Type of a variable
- Length of a variable
- Codelists/Formats
- Labels (this optional, because already slightly differences in writing could be detected and lead to differences)
- NotNull information (a variable that cannot be missing)

The resulting comparison is the most important dataset and it keeps the information which will be needed to create the Excel mapping specification.

The following Fig. 6 shows the call of the sub-macro **%mcheck_variables_metadata** in the loop through all domains assigned before.

```
%do d=1 %to &numdomains;
   …
  /*************************************************************************/
  /*check study data structure against metadata of the IDB */
  /*************************************************************************/
  %mcheck_variables_metadata(
                                Indata  =&inlib..&sourcedomain
                               , meta    = &metalib..&outdomain,
                               , inadsl   =&inlib..&adslbase
                               , outdata =&outdomain._check);
   …
%end;
```

Fig. 6: Check macro for study data against IDB metadata

After checking and gathering information about study metadata structure vs. pool metadata structure the %m_create_specification macro writes a **description sheet** to the Excel file with information about the domains used in the mapping, the key variables, the sort order within the domains, and source and target domains. An example of a description sheet within the specification file is given in Fig. 7 below.

	domain	sort_order	keyvar	source	target
2	ADSL	STUDYID USUBJID	STUDYID USUBJID	DM	ADSL
3	ADAE	STUDYID USUBJID ASEQ	STUDYID USUBJID ASEQ	AE	ADAE
4	ADEX	STUDYID USUBJID ASEQ	STUDYID USUBJID ASEQ	EX	ADEX
5	ADLB	STUDYID USUBJID PARAMCD	STUDYID USUBJID ASEQ	LB	ADLB
6	ADVS	STUDYID USUBJID PARAMCD	STUDYID USUBJID ASEQ	VS	ADVS
7	ADCM	STUDYID USUBJID CMSEQ	STUDYID USUBJID CMSEQ	CM	ADCM
8	ADEGM	STUDYID USUBJID PARAMCD	STUDYID USUBJID ASEQ	EG	ADEGM
9	ADMH	STUDYID USUBJID MHSEQ	STUDYID USUBJID ASEQ	MH	ADMH
10	ADTA	STUDYID TSPARMCD	STUDYID ARMCD TAETORD	TA	ADTA
11	ADTE	STUDYID ELEMENT	STUDYID ETCD ELEMENT	TE	ADTE
12	ADTS	STUDYID TSPARMCD	STUDYID TSPARMCD	TS	ADTS
13	ADTV	STUDYID VISITNUM	STUDYID VISITNUM	TV	ADTV

Fig. 7: Domain description sheet

Looping through all domains the macro %m_create_specification creates a sheet which includes the information collected by the check macro for each domain.
We will look on the structure of a domain sheet in the next paragraph.

Depending on the result of the check macro and matches between the study metadata structure and the IDB metadata structure an Excel spreadsheet is created.

The spreadsheet consists of:

- Description sheet (Fig. 7) with basic information (domain names, sort order, key variables, source domains, target domains)
- Sheets for each domains of the integrated database assigned to the mapping (Fig. 8)

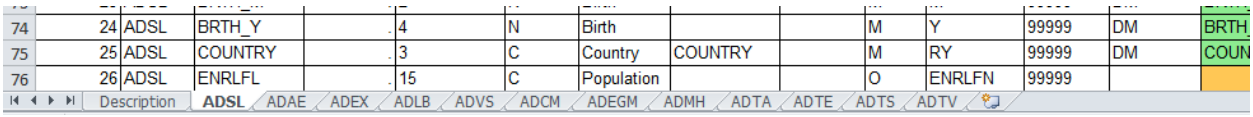

Fig. 8: Sheets for description and different domains

In the domain sheet all information about the mapping of study data A to pool data B is collected. It includes information about the target (IDB) and the source (study data) and about the derivations.
The sheets for each domain basically consists of three elements: the target structure, the source structure and the derivation rules.

- Target (IDB):
  - Varseq (sequence ID of the variable in the IDB)
  - Domain
  - Variable name

  - Keyvar (key variable ID number in metadata)
  - Output format
  - Type of the variable
  - Label of the variable
  - Codelist assigned
  - NotNull information
  - Mandatory (mandatory/optional)
  - Description in the IDB

- Source (Study metadata)
  - Study ID
  - Domain source
  - Variable name
  - Label
  - Type of the variable
  - Format in study
  - Length of the variable
- Derivation / Information about comparison/check
  - Code/Decode variable
  - Information about missings/inconsistencies in data
  - Derivation
  - Derivation type (data step/macro)

Below, in Fig. 9 three elements of the domain sheets are displayed in different colors:

- Green = Target (IDB metadata structure)
- Orange = Source (Study structure)
- Gray = Derivation (mapping description, SAS code/macro calls)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Project: 99999 - Pool Metadata											Source Study Metadata									Derivation
2	Varseq	Domain Target	sasname	Keyvar	Outform	Type	Label	Codelst	NotNull	Mand	Descript	Studyid	Domain Source	Var_study	Label Source	Type study	Format study	Length	Comment	Comment2	Derivation
42	-1		SAFETYLT									10000	DM	SAFETYLT	Safety	C	$	2	[not in metadata]		
43	-1		SAFETYN									10000	DM	SAFETYN	Population	N	NY	8	[not in metadata]		
44	-1		SAFSUR									10000	DM	SAFSUR	Population	C	$	2	[not in metadata]		
45	-1		SAFSURN									10000	DM	SAFSURN	Population	N	NY	8	[not in metadata]		
46	-1		SFETYLTN									10000	DM	SFETYLTN	Safety	N	NY	8	[not in metadata]		
47	1	ADSL	ADSNAME		8	C	Name	X_ADSNM		M		10000									
48	2	ADSL	STUDYID	1	10	C	Identifier		Y	M	D	10000	DM	STUDYID	Identifier	C	$	10			ADSL.STUDYID
49	3	ADSL	USUBJID	2	20	C	Subject		Y	M	D	10000	DM	USUBJID	Subject	C	$	20			ADSL.USUBJID
50	4	ADSL	AGE		3	N	Age			M	DM.AGE	10000	DM	AGE	Age	N		8			ADSL.AGE
51	5	ADSL	AGEGR1		15	C	1			O	AGEGR1N	10000							[AGEGR1N]		AGEGR1=put(AGEGR1N, Z_AGEGRP.);
52	6	ADSL	AGEGR1N		3	N	1 (N)	Z_AGEGRP		O		10000									
53	7	ADSL	AGEGR2		15	C	2			O	AGEGR2N	10000							[AGEGR2N]		AGEGR2=put(AGEGR2N, Z_AGEGRP.);
54	8	ADSL	AGEGR2N		3	N	2 (N)	Z_AGEGRP		O		10000									
55	9	ADSL	AGEGR3		15	C	3			O	AGEGR3N	10000							[AGEGR3N]		AGEGR3=put(AGEGR3N, Z_AGEGRP.);
56	10	ADSL	AGEGR3N		3	N	3 (N)	Z_AGEGRP		O		10000									
57	11	ADSL	AGEGR4		15	C	4			O	AGEGR4N	10000							[AGEGR4N]		AGEGR4=put(AGEGR4N, Z_AGEGRP.);
58	12	ADSL	AGEGR4N		3	N	4 (N)	Z_AGEGRP		O		10000									
59	13	ADSL	AGEU		6	C	Age Units	AGEU		M	DM.AGEU	10000	DM	AGEU	Age Units	C	$AGEU	6			ADSL.AGEU
60	14	ADSL	ARM		200	C	of Planned			M	ARMCD	10000	DM	ARM	of Planned	C	$	200	[ARMCD]		ADSL.ARM
61	15	ADSL	ARMCD		20	C	Arm Code	ZARMCD		M	DM.ARMCD	10000	DM	ARMCD	Arm Code	C	$ZARMCD	20			ADSL.ARMCD
62	16	ADSL	ASR		8	C	ace			M		10000									
63	17	ADSL	BASEBMI		8.3	N	Body Mass			O		10000									
64	18	ADSL	BASEHEIG		8.3	N	Height (cm)			O		10000									
65	19	ADSL	BASEWEIG		8.3	N	Weight (kg)			O		10000									
66	20	ADSL	BRTHDT		DATE9.	N	Birth Date			M		10000									
67	21	ADSL	BRTHDTL		9	C	Birth Date			M	for Listings	10000									
68	22	ADSL	BRTH_D		2	N	Birth			M	D	10000	DM	BRTH_D	Day of Birth	N		8		[includes only missings]	ADSL.BRTH_D
69	23	ADSL	BRTH_M		2	N	Birth			M	M	10000	DM	BRTH_M	Birth	N		8		[includes only missings]	ADSL.BRTH_M
70	24	ADSL	BRTH_Y		4	N	Birth			M	Y	10000	DM	BRTH_Y	Year of Birth	N		8			ADSL.BRTH_Y
71	25	ADSL	COUNTRY		3	C	Country	COUNTRY		M	RY	10000	DM	COUNTRY	Country	C	$COUNTRY	3			ADSL.COUNTRY
72	26	ADSL	ENRLFL		15	C	Population			O	ENRLFN	10000							[ENRLFN]		ENRLFL=put(ENRLFN, NY.);

Fig. 9: Specification Excel file created by %m_create_specification based on study data/metadata and pool metadata

In the generated specification file, a column about the study variables is color-coded to show whether the metadata structure of the pool match or do not match the single study metadata information. The figure above (Fig. 9) shows an example of a mapping from SDTM to ADaM. Red cells describe variables which are not in the pool metadata but in the single study data. Green cells indicate a match, which means the variable appears in both the pool metadata and the study metadata. When the color is orange these variables need to be derived and are not available in study data. Derivations can be entered in a specific derivation column manually.

It is also possible to include SAS code, macro calls. Often variables which appear in the study data and have the same meaning as IDB variables but with a different name could be renamed as: *"= variable name study"*. The renaming should be in the right row of the target variable name of the IDB. Typically, code and decode pairs occur in the IDB. If the code will be derived automatically, the put statement as *variable_decode=put(variable_code, codelist.)* is already created by the macro and included in the derivation column.

Information about missings is also taken from the check macro and included in a comment column of the specification.

In some cases a specific order of the derivations needs to be followed in the program flow. This could be considered in an order column where the order of the derivation steps could be changed. A derivation method describes how a particular variable is derived (data step/macro).
After including all derivations into the domain sheets a review step should be performed by a second analyst or a statistician.
In the following paragraph the prepared specification document containing all necessary information about the mapping from A to B will be used to create mapping programs automatically.

MAPPING PROGRAM GENERATION

The final step of this approach is the preparation of the mapping programs. These programs also called "*template programs*" because in a post-processing step the program could use these automatically generated programs as basis (template) for changes or further derivations which were made after the semi-automatic specification was created. Of course changes of the derivation could also be included in the specification and the mapping programs could be generated again. According to validation SOPs usually a second programmer reviews the programs or double programming is performed. In case of a double program, a manually created program is prepared by a second programmer. The specification file will not be used as stand-alone system to replace current SOPs or programming guidelines. It is a further way to simplify the process of mapping data from A to B with automatic elements and support of a specification file created by the system described in the first part of this paper.

In the next paragraph we will look on the details how the mapping file is generated from the Excel specification by a macro **%m_create_mapping**. The following Fig. 10 shows the header of the mapping program generation macro.

```
%macro m_create_mapping (  project=          /*Project name                                          */
                         , studyid=          /*Study ID                                              */
                         , domain=           /*Domains (separated by '#', e.g ADSL#ADEX#ADLB#ADTS#ADTV)  */
                         , path=             /*Path to output folder of the progranms                */
                         , userid=           /*User ID                                               */
                         , specification=    /*Specification filename                                */
                         , spec_path=        /*Path to specification                                 */
                         , filenamepattern=  /*Pattern of the filnename                               */
                );
```

Fig. 10: Program header mapping macro %m_create_mapping

For the call of the macro %m_create_mapping the following parameters need to be specified:

- Project name (compound)
- Study ID
- Domains (different domains sperated by '**#**')
- Path to the output folder
- User ID
- Specification File
- Path to the specification
- File name pattern

The program template macro %m_create_mapping consists basically of a loop through all domains assigned in the macro call in the domain parameter. It is possible to call all domains given in the specification Excel file.

Loop through all <*domains*>

a. Read specification from Excel
b. Create header of template file
c. Insert derivations (Data Step / Macros) and comments (descriptions) into template file
d. Post-processing

End

The first step is to read the Excel spreadsheet for the specific domain in the loop. The macro reads the spreadsheet and generates a SAS dataset with the same information.
The following steps of the template macro create different parts of the program itself:
The header is created for example by a macro %template_header. In case changes of the header are necessary in case of a SOP change or program guideline change, this change could be easily implemented.
All derivations are then implemented in the template program by the macro %template_derivations. It mainly goes through the different rows of the derivation column and inserts SAS code into the program. Additionally, comments of the description of the derivation are added before each derivation step.

Macro calls in the derivation column create a macro call in the template program. The macro %template_derivations follows the order given for each derivation. Simple renaming of variables is usually done at the beginning of a mapping program.

One of the last steps of the macro is to add the metadata structure to the domain data along with final sort steps and cleaning steps. Generally, the structure of the template generation could be adapted to company and client needs.
The main loop of the macro %m_create_mapping is shown below in Fig. 11.

```
%varcount(list=&domain);
%do d=1 %to &var_num;


    %let _domain= %left(%qscan(&domain.,&d,"#"));


    %read_meta_spec(domain=&_domain, path=&spec_path., filename=&specification);
    %template_header(domain=&_domain, studyid=&studyid, project=&project,  path=&path, cwid=&cwid,
                     filenamepattern=&filenamepattern.);
    %template_set_ds(domain=&_domain);
    %template_set_keyvar(domain=&_domain);
    %template_in_data(domain=&_domain);
    %template_derivation(domain=&_domain);
    %template_sortorder(domain=&_domain);
    %template_drop_in_var(domain=&_domain);
    %template_add_dummy;
    %template_clean_lib;
    %template_eof;
%end;
```

Fig. 11: Macro %m_create_mapping main part

Now we will describe some of the sub-macros called from the macro %create_mapping. The first macro reads the specific domain sheet from the Excel file containing information about the metadata structure of the IDB (target), the study metadata (source) and the derivations. The macro %read_meta_spec creates a SAS dataset.
An important sub-macro (%template_header, see Fig. 12) creates a SAS mapping file with all necessary header information about the project, study ID, author, program name and path to output folder. The header should be adapted to specific company / client needs.
The header is covered in SAS comments. These comments are created within data _null_; and put statements.

```
%macro template_header(  domain=                          /*domain                                        */
                       , studyid=                         /*Study ID                                      */
                       , project=                         /*Project/Therapeutic area/Compound             */
                       , path=                            /*Path to output file                           */
                       , validation=double programming    /*Validation method <double/code review>        */
                       , userid=                          /*User ID / Name of the programmer              */
                       , filenamepattern=p-template-      /*Filename pattern                              */
                      );

filename _templ "&path/&filenamepattern.&domain..sas";

data _null_;
   file _templ old;
   put '/*-------------------------------------------------------------------------------------------*/';
   put '/* The Drug Company                                                                         */';
   put '/*-------------------------------------------------------------------------------------------*/';
   put "/* Studyname          : &studyid                                                             */";
   put "/* Project             : &project                                                             */";
   put '/* Program name       : %nrstr(%LET) prog = p-&targetdomain..sas;                            */';
   put "/* Purpose             : map dataset %upcase(&targetdomain.) to IDB                           */";
   put '/* Template            :                                                                      */';
   put '/* Validation method  : &validation                                                          */';
   put "/* Author name         : (&userid.) (%sysfunc(strip(&username.),char.))                       */";
   put "/* Date completed      : &sysdate9.                                                           */";
   put '/* Updated by          : (Name) - (Date):                                                     */';
   put '/*-------------------------------------------------------------------------------------------*/';
run;
%mend;
```

Fig. 12: Macro %_template_header

In the next step the macro goes through all derivations available in the Excel file. The derivations are marked with derivation_flag='Y'. Also the orange color indicates that the variable of the domain needs to be derived, but the

derivation_flag tells the macro that the specific code in the derivation column will be included in the mapping template program. Before each derivation the macro also includes the information given in the label and comment column. This information is inserted as SAS comment. Again the derivation and the comments are included by data _null_ and put statements. An excerpt of the macro inserting derivations is given below in Fig. 13.

```
data _null_;
   set _deriv01;
   file _templ mod;
   put '/*******************************************************************************************/';
   put '/*' sasname '(' label '): ' descript '*/';
   put '/*******************************************************************************************/';
   put  derivation;
run;
```

Fig. 13: Macro excerpt - *put* derivations and descriptions

In Fig. 14 derivations for ADSL variables are given. For the user of the specification the green color indicates that these variables already exist in the source data and will not be derived. Sometimes a variable in the source data has the same name as in the target data but has a different meaning. In such a case, the source variable needs be changed although it exists: this could be done via the derivation flag column. Orange colors indicate that these variables are not available in the source data and should be derived. If a variable from the source does not exist in the target data, the variable is highlighted in red color. This could also mean that this variable needs to be renamed. Re-namings are not automatically detected while merging study data with the metadata of the IDB. A rename statement should be manually inserted in the specific row of the target variable.

L	M	N	O	P	Q	R	S	U	V	W	X
Source Study Metadata							Derivation				
Studyid	Domain Source	Var_study	Label Source	Type study	Format study	Length	Comment	Derivation	Derivation_order	Derivation_type	derivation_flag
99999						.			.		Y
99999	DM	STUDYID	Identifier	C	$	10		ADSL.STUDYID	.		
99999	DM	USUBJID	Subject	C	$	20		ADSL.USUBJID	.		
99999	DM	AGE	Age	N		8		ADSL.AGE	.		
99999						.	[AGEGR1N]	AGEGR1=put(AGEGR1N, Z_AGEGRP.);	2	[data step]	Y
99999						.		%_adsl_agegr1(age);	1	[macro]	Y
99999						.	[AGEGR2N]	AGEGR2=put(AGEGR2N, Z_AGEGRP.);	.		Y
99999						.		%_adsl_agegr2(age);		[macro]	Y
99999						.	[AGEGR3N]	AGEGR3=put(AGEGR3N, Z_AGEGRP.);	3		Y
99999						.		%_adsl_agegr3(age);	.	[macro]	Y
99999						.	[AGEGR4N]	AGEGR4=put(AGEGR4N, Z_AGEGRP.);	.		Y
99999						.		%_adsl_agegr4(age);	.	[macro]	Y
99999	DM	AGEU	Age Units	C	$AGEU	6		ADSL.AGEU	.		
99999	DM	ARM	of Planned	C	$	200	[ARMCD]	ADSL.ARM	.		
99999	DM	ARMCD	Arm Code	C	$ZARMCD	20		ADSL.ARMCD	.		
99999						.			.		Y
99999						.		%_adsl_bmi;	.	[macro]	Y
99999						.		%_adsl_heigth;	.	[macro]	Y
99999						.		%_adsl_weight;	.	[macro]	Y
99999						.		%_adsl_brthdt;	.	[macro]	Y
99999						.		%_adsl_brthdtl;	.	[macro]	Y

Fig. 13: Derivations with data step and macro calls for ADSL domain

Fig. 14 shows an excerpt of the resulting mapping template file with some basic derivations done in a data step.

```
/*******************************************************************************************/
/* Basic derivations given in specification */
/*******************************************************************************************/


data &dsn.;
   set &dsn.;
   /*******************************************************************************************/
   /*AGEGR1 (Age Group 1 ): decode of AGEGR1N */
   /*******************************************************************************************/
   AGEGR1=put(AGEGR1N, Z_AGEGRP.);
   /*******************************************************************************************/
   /*UASR (Unique Subject Identifier/Age/Sex/Race ):   */
   /*******************************************************************************************/
   uasr=cat(scan(uasre,1,'/'),'/',scan(uasre,2,'/'),'/',scan(uasre,3,'/'),'/',scan(uasre,4,'/'));
run;
```

Fig. 14: Resulting SAS code in template based on data step commands

Most of the more complicated derivations are generally done in macros. For the user of the specification file it is more convenient to insert macro calls than including complicated SAS code derivations. This also makes the program easier to read. Some of the macros are general macros used in the project and could be used also in other mappings of different clinical studies.

In the following excerpt (Fig. 15) of the generated mapping template program comments and the related SAS macro calls are displayed.

```
/*****************************************************************************************/
/*AGEGR2N (Age Group 2 (N) ):   */
/*****************************************************************************************/
%_adsl_agegr2(age);
/*****************************************************************************************/
/*AGEGR3N (Age Group 3 (N) ):   */
/*****************************************************************************************/
%_adsl_agegr3(age);
/*****************************************************************************************/
/*AGEGR4N (Age Group 4 (N) ):   */
/*****************************************************************************************/
%_adsl_agegr4(age);
/*****************************************************************************************/
/*BASEBMI (Baseline Body Mass Index (kg/m2) ):   */
/*****************************************************************************************/
%_adsl_bmi;
/*****************************************************************************************/
/*BASEHEIG (Baseline Height (cm) ):   */
/*****************************************************************************************/
%_adsl_heigth;
/*****************************************************************************************/
/*BASEWEIG (Baseline Weight (kg) ):   */
/*****************************************************************************************/
%_adsl_weight;
/*****************************************************************************************/
/*BRTHDT (Birth Date ):   */
/*****************************************************************************************/
%_adsl_brthdt;
/*****************************************************************************************/
/*BRTHDTL (Birth Date ): Birth Date for Listings */
/*****************************************************************************************/
%_adsl_brthdtl;
```

Fig. 15: Resulting SAS macro calls in mapping template program

DISCUSSION OF THE RESULTS AND USABILITY

In some case studies the semi-automatic process of specification and mapping file generation could reduce the amount of time to map data of a clinical study into the integrated database. Usually at least two programmers work on the mapping of data. Also the time to prepare a mapping specification is significantly reduced because most of the general work is done by the macro **%m_create_specification**. It covers most of the tedious work to create a specification with all domains/sheets, variables. A check macro which merges the study structure with the metadata of the IDB also finds matches and indicates which variables need to be derived or possibly renamed. Project specific macros for important mappings which are used in different studies could be automatically integrated into the specification document. Finally, the statistician or a second analyst should review the specification file according to the requirements (SAP/data handling document). After inclusion of all missing derivations manually the final mapping file generation is an automatic process – the macro **%m_create_mapping** creates all program files for the different domains of the specification.

All derivations include automatically generated SAS comments describing the derivations. Another advantage is also that programs (sub-macros of the %m_create_mapping) can be modified to follow company programming SOPs and guidelines. The mapping programs have the adequate headers and enough comments for all derivations are included. The code is also easy to read. In case additional changes are necessary, this could either be included in the specification file or directly in the mapping program.

Moreover, the whole mapping process could be made more transparent and easy to follow and review. Only the description of the structure should follow specific rules (same definition of metadata).

Of course there exist already different approaches of automating the mapping process. An advantage of the described approach of this paper is that not only is the mapping done, but the specification is also generated in a

semi-automatic process. Different methods sometimes only map data of a specific structure, like SDTM to ADaM. With this method we could map any source structure in any target structure. You could map easily from A to B.

CONCLUSION

We have shown how to automate the process of mapping clinical study data into an integrated database. A result of this process is a semi-automatic generated specification. Additionally, as second result the mapping programs are fully generated automatically.

In a further stage additional data checks could be implemented and given the user more information about necessary changes or also feedback about issues in the data. Also the implementation of the define documents could be easily implemented based on the complete specification.

ACKNOWLEDGMENTS

I would like to thank Pam Giese (Principal Statistical Programmer, inVentiv Health) and Hans-Dieter Spies (Senior Statistical Programmer, inVentiv Health) for their great support and review of this manuscript.

CONTACT INFORMATION

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

Thomas Wollseifen
InVentiv Health Germany GmbH
Grosse Hub 10d
Eltville am Rhein / 65344
Work Phone: +49 (0) 6123 70437-10
Email: thomas.wollseifen@inventivhealth.com
Web: www.inventivhealth.com

About inVentiv Health

inVentiv Health is a top-tier professional services organization that accelerates the clinical and commercial success of biopharmaceutical companies worldwide. Our combined Clinical Research Organization (CRO) and Contract Commercial Organization (CCO) help clients improve their performance to deliver much-needed therapies to market. With 14,000 employees providing services to clients in 70 countries, inVentiv Health designs best practices, processes and systems to enable clients to successfully navigate an increasingly complex environment. For more information, visit inVentivHealth.com.