Paper SM10

A Practical Approach to Automating SDTM Using a Metadata-Driven Method That Leverages CRF Specifications and SDTM Standards

Xiangchen (Bob) Cui, CRISPR Therapeutics AG, Boston, USA
Min Chen, CRISPR Therapeutics AG, Boston, USA
Jessie Wang, CRISPR Therapeutics AG, Boston, USA

ABSTRACT

Automated SDTM generation has several benefits, including efficiency, accuracy, compliance with regulatory requirements, and the speeding up of the data analysis process. However, due to the dissimilarity and varying complexity of different CRFs, SDTM domains, and eSource systems among different studies, development of a tool to automate SDTM has been a challenging task for sponsors, CROs, and EDC service providers.

We propose a new approach in automatic generation of SAS® code for SDTM. A SAS-based macro is developed based on CRF specifications from an EDC database and SDTM standards. Our approach is user-friendly with high transparency, easily scalable to multiple studies, and especially useful for relatively smaller sponsors and CROs, for there is no requirement to standardize CRFs and raw dataset variables' attributes (which is the best practice but can be too work-intensive) and no required expertise in other computer languages.

INTRODUCTION

SDTM automation streamlines the process of transforming raw clinical trial data into the standardized SDTM format. This process involves CRF annotations, SDTM specification writing, development of SAS code to generate SDTM data, validation, SDRG writing, and define.xml generation. SDRG and define.xml support regulatory submissions along with annotated CRFs and SDTM datasets in v5 Transport Format (XPORT) [1]. Automation offers several benefits, including efficiency, accuracy, compliance with regulatory requirements, and the speeding up of the data analysis process. However, it is not an easy task to achieve due to the complexity of different CRFs and eSource systems. Automation is a hybrid process comprising of applications or tools plus manual parts involving CRF annotations and SDTM specification writing. Standardization of raw data collection can dramatically reduce the time spent on these manual parts. However, the resources needed for standardization are not always feasible for smaller sponsors or CROs.

This paper introduces a new approach in the automatic generation of SAS code for SDTM automation that strikes a balance between high-level automation and resource investment. A SAS-based macro named ***%SDTM_Code_Generator*** was developed based on CRF specifications from an EDC database and SDTM programming standards [2,3]. We provide details on the rationale and logic flow for this macro, its inputs and outputs, how to build a master-annotation spreadsheet to support SDTM automation, how to efficiently scale it up for new studies, how to handle external data, how to effectively validate its output datasets, and how to deal with EDC database changes.

Based on our working experience from applying this new approach to two different types of oncology studies, this paper is titled "**A Practical Approach to Automating SDTM**" for the following reasons:

1. High quality delivery of SDTM datasets and operational efficiency through high-level automation
2. Flexibility for users to control the degree of automation and account for cost/timelines
3. Faster and solid validation due to not requiring double programming for all domains and a guarantee that all raw dataset variables are accounted for in SDTM programming
4. Scalability to multiple studies by leveraging existing CRF specifications, master-annotation spreadsheets, and macros
5. Transparency and user-friendliness as users can easily and directly review the inputs and outputs of the process so that they have very high confidence in the delivery of SDTM datasets
6. No requirement of huge efforts to standardize CRFs or raw dataset variable attributes
7. No requirement of expertise in other computer languages, such as Structured Query Language (SQL) for script creation

STANDARD SDTM PROGRAMMING WORKFLOW

Figure 1 below depicts the standard SDTM programming workflow. SDTM programmers start to develop SAS programs for the SDTM dataset generation ***only*** after CRF annotations and SDTM specifications are available. A SDTM programmer manually annotates each CRF either to set up the one-to-one mapping from each raw dataset variable specified in the CRF to its mapped SDTM domain variable or supplemental qualifier or to label it as "NOT SUBMITTED".

The annotated case report form (aCRF) then guides the programmers to develop SAS programs for SDTM dataset generation. One also manually completes each SDTM domain specification to document and select the required SDTM variables and/or supplemental qualifiers in the final SDTM dataset. This is a time-consuming process even with tools such as template SAS Programs [3] for SDTM mapping or CRF annotation tools [4], which guide the programmers in annotating each CRF based on the applicable standards. The amount of manual work required is high as well.

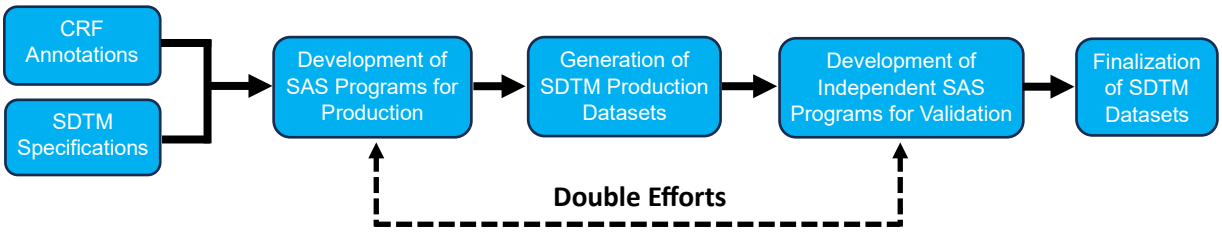

Figure 1. Standard SDTM Programming Workflow

In the past, we've written about our established SDTM programming process [2, 3], which complies with FDA Business Rules [5] and CDISC SDTMIG [6]. The utilization of template SAS Programs for SDTM Mapping has been our standard practice, and they have been successfully applied to multiple clinical studies, including several FDA submissions and their approvals. The present goal is to replace our standard SDTM mapping templates with a macro for SDTM automation.

INTRODUCTION TO OUR NEW SDTM PROGRAMMING WORKFLOW

Figure 2 below shows our new SDTM programming workflow. In contrast to the standard workflow depicted in Figure 1, a master-annotation spreadsheet is created from CRF annotations and CRF specifications. The master-annotation spreadsheet contains metadata and variable attributes for the raw datasets combined with annotations mapping raw dataset variables to specific SDTM domain variables. SDTM specifications contain information on SDTM standards along with variable inclusion/exclusion and derivation. The master-annotation and SDTM specifications are the inputs of our new macro, ***%SDTM_Code_Generator***, which automatically generates SDTM mapping SAS programs. In contrast to the traditional double programming validation shown in Figure 1, our new programming validation process consists of the following three steps: code reviewing, real data testing, and developing an independent mapping SAS program to validate a SDTM dataset for some complicated domains as needed per the team's decision.

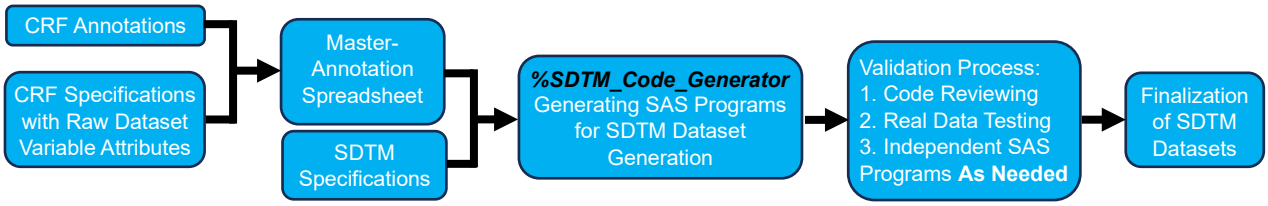

Figure 2. New SDTM Programming Workflow

RATIONALE FOR THE DEVELOPMENT OF A MACRO FOR SDTM AUTOMATION

Aside from trial design domains, there are typically over twenty SAS programs needed to read and map the data collected from CRFs for each study. From a quality assurance (QA) perspective, independently developed SAS programs for validation are designed to ensure the highest quality. However, that doubles the development work required. Some domains, such as LB or PR, may have hundreds of data blocks due to the numerous tests or procedures collected on CRF forms. The manual effort needed to ensure that all of these are correctly included in both production and validation programs is time-intensive and still prone to error. Table 1 shows the advantages and benefits of a macro for SDTM automation over the SDTM mapping template SAS programs. A huge amount of work is needed to update template SAS programs for EDC database changes or new studies while a macro can automatically adapt to some of those changes and save development time.

Types of Changes	Specific Changes	SDTM Mapping Template SAS Programs	A Macro
eSource systems or EDC database changes	Raw dataset names, variable attributes, CRF annotations	Must make the corresponding updates/changes across over forty SAS programs from both production and validation by typing and/or copying, which is time-consuming and error-prone	No changes or minimal changes
New studies	New CRFs, domains, more changes, annotations, and specifications	Make the corresponding updates/changes to SAS programs	May need to update the macro correspondingly and update its input files if necessary

Table 1. Advantages of a Macro for SDTM Automation Over SDTM Mapping Template SAS Programs

INTRODUCTION TO THE MACRO'S SINGLE PARAMETER AND ITS OUTPUT FOR SDTM AUTOMATION

Table 2 below shows the macro's single parameter, its calls, and the outputs of the calls. Its single parameter is either a specific SDTM domain name or "ALL", and its call generates a SAS program for the specified domain or SAS programs for all domains, respectively. It requires that all CRF annotations (SDTM mapping), all raw dataset names, and their attributes (variable names, labels, and types) are stored in a single spreadsheet named as ***master-annotation***.xlsx. Further details for the master-annotation spreadsheet are included in a later section.

Macro Call	Output of the Macro Call
%SDTM_Code_Generator(domain_=*Domain Name*) For example, % SDTM_Code_Generator(domain_=DM);	A SAS program with the domain name (e.g., DM.sas)
%SDTM_Code_Generator(domain_=ALL)	All SAS programs for the domains specified in master-annotation.xls

Table 2. *%SDTM_Code_Generator* Macro Calls and Outputs

Of note, the SUBJECT VISITS (SV) domain is a special purpose domain that requires more complex derivations, many of which are different from ones of the original ***%SDTM_Code_Generator*** macro. To simplify the development of the macro and reduce the length of SAS code needed, we developed an additional macro named ***%SV_Code_Generator***, which leverages the output from the call of ***%SDTM_Code_Generator*** and extends it further. Please refer to [7] for more information.

HOW *%SDTM_CODE_GENERATOR* CREATES A SAS PROGRAM

Our macro generates SAS mapping code from its input files and writes that mapping code into a SAS dataset. Display 1 below is an example of that SAS dataset with 2 columns: ***lines*** and ***_order***. Using the code in Display 2 from ***%SDTM_Code_Generator***, we can output the contents of our final dataset into a SAS program file, CM.sas (Display 3). The lines of code contained in this output CM.sas file (Display 3) are identical to the contents of Display 1's ***lines*** column. This is done separately for each SDTM domain.

#	lines	_order
1	data try;	10
2	attrib &attrib.;	20
3	set CM(drop=studyid siteid);	30
4		901
5	STUDYID='Study-101'	902
6	DOMAIN = 'CM';	903
7	USUBJID = strip(STUDYID) || strip(substr(SUBJECT, 4));	904
8	CMSPID = strip(put(RECORDPOSITION, z3.));	905
9	if not missing(CMTRT) then CMTRT=strip(CMTRT);	906
10	CMDECOD = strip(CMTRT_PT);	907
11	CMCAT = 'PRIOR AND CONCOMITANT MEDICATIONS';	908
12	if not missing(CMINDC_STD) then CMINDC=strip(CMINDC_STD);	909
13	CMCLAS = coalescec(CMTRT_ATC4, CMTRT_ATC3, CMTRT_ATC2, CMTRT_ATC1);	910
14	CMCLASCD = coalescec(CMTRT_ATC4_CODE, CMTRT_ATC3_CODE, CMTRT_ATC2_CODE, CMTRT_ATC1_CODE);	911
15	if not missing(CMDOSE) then CMDOSE=CMDOSE;	912
16	if not missing(CMDOSU_STD) then CMDOSU=strip(CMDOSU_STD);	913
17	if not missing(CMDOSFRM_STD) then CMDOSFRM=strip(CMDOSFRM_STD);	914
18	if not missing(CMDOSFRQ_STD) then CMDOSFRQ=strip(CMDOSFRQ_STD);	915
19	if not missing(CMROUTE_STD) then CMROUTE=strip(CMROUTE_STD);	916
20	%map_dtc_date(_DATEVAR=CMSTDTC,_RAWDATE=CMSTDAT);	917
21	%map_dtc_time(_DATEVAR=CMSTDTC,_RAWTIME=CMSTTIM);	918
22	%map_dtc_date(_DATEVAR=CMENDTC,_RAWDATE=CMENDAT);	919
23	%map_dtc_time(_DATEVAR=CMENDTC,_RAWTIME=CMENTIM);	920
24	if CMONGO = 1 then CMENRTPT = 'ONGOING';	921
25	if not missing(CMENRTPT) then CMENTPT = 'END OF STUDY';	922
26	run;	922.5

Display 1. A SAS Dataset with Columns *lines* and *_order* Containing a Snippet of Code for SDTM CM.sas

```
%let outdir=.../&program_path.;
%let domain_=CM;
data _null_;
     set final;
     file "&outdir./_.&domain_..sas";
     put @1 lines $255.;
run;
```

Display 2. SAS Data _NULL_ Step to Output a SAS Program

```
21 data try;
22      attrib &attrib.;
23      set CM(drop=studyid siteid);
24
25          STUDYID = 'Study-101';
26          DOMAIN = 'CM';
27          USUBJID = strip(STUDYID) || strip(substr(SUBJECT, 4));
28          CMSPID = strip(put(RECORDPOSITION, z3.));
29          if not missing(CMTRT) then CMTRT=strip(CMTRT);
30          CMDECOD = strip(CMTRT_PT);
31          CMCAT = 'PRIOR AND CONCOMITANT MEDICATIONS';
32          if not missing(CMINDC_STD) then CMINDC=strip(CMINDC_STD);
33          CMCLAS = coalescec(CMTRT_ATC4, CMTRT_ATC3,  CMTRT_ATC2, CMTRT_ATC1);
34          CMCLASCD = coalescec(CMTRT_ATC4_CODE, CMTRT_ATC3_CODE,  CMTRT_ATC2_CODE, CMTRT_ATC1_CODE);
35          if not missing(CMDOSE) then CMDOSE=CMDOSE;
36          if not missing(CMDOSU_STD) then CMDOSU=strip(CMDOSU_STD);
37          if not missing(CMDOSFRM_STD) then CMDOSFRM=strip(CMDOSFRM_STD);
38          if not missing(CMDOSFRQ_STD) then CMDOSFRQ=strip(CMDOSFRQ_STD);
39          if not missing(CMROUTE_STD) then CMROUTE=strip(CMROUTE_STD);
40          %map_dtc_date(_DATEVAR=CMSTDTC,_RAWDATE=CMSTDAT);
41          %map_dtc_time(_DATEVAR=CMSTDTC,_RAWTIME=CMSTTIM);
42          %map_dtc_date(_DATEVAR=CMENDTC,_RAWDATE=CMENDAT);
43          %map_dtc_time(_DATEVAR=CMENDTC,_RAWTIME=CMENTIM);
44          if CMONGO = 1 then CMENRTPT = 'ONGOING';
45          if not missing(CMENRTPT) then CMENTPT = 'END OF STUDY';
46 run;
```

Display 3. A Snippet of the Output SAS Program for SDTM CM.sas

INTRODUCTION TO THE LOGIC FLOW OF *%SDTM_CODE_GENERATOR*

Figure 3 below shows the logic flow of ***%SDTM_Code_Generator*** alongside a typical SDTM programming logic flow with arrows connecting corresponding blocks. The former outputs a SDTM SAS program while the latter outputs a SDTM dataset.

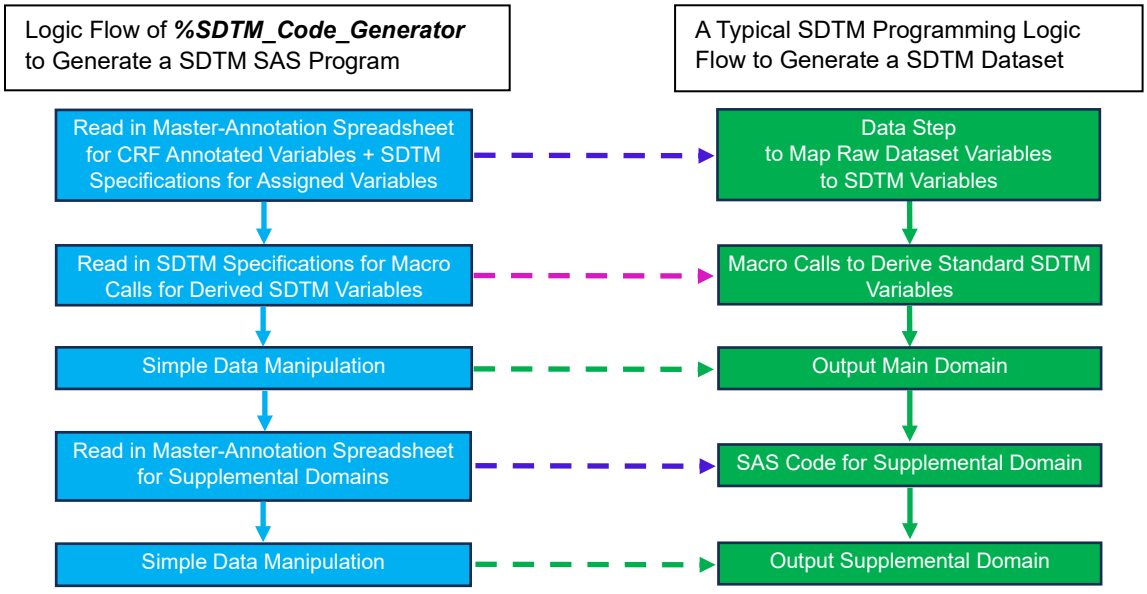

Figure 3. The Logic Flow from *%SDTM_Code_Generator* vs. Typical SDTM Programming

Before developing SAS code for each SDTM domain, ***%SDTM_Code_Generator*** first reads in and subsets the master-annotation spreadsheet for the dedicated SDTM domain. Secondly, it reads the ***Origin*** and our newly added ***Derivation/Assigned*** columns from SDTM specifications, which contain information on whether variables are directly mapped, assigned, or derived. Table 3 below shows the source of inputs for the macro's automation.

Origin of SDTM Variables	Input of a SAS-Based Macro	Source	Case
CRF	Master-Annotation Spreadsheet	Combination of CRF Specifications and CRF Annotations	All mapping variables
Protocol	SDTM Specifications	Derivation/Assigned column	Assignment, e.g., STUDYID='Study-101'
Assigned	SDTM Specifications	Derivation/Assigned column	Assignment, e.g., DOMAIN = 'CM' or DM.ARMCD from the call of %get_trt
Derived	SDTM Specifications	Derivation/Assigned column	A line of code or macro call(s)

Table 3. Sources of Inputs for *%SDTM_Code_Generator*

INTRODUCTION TO OUR STANDARD SDTM SPECIFICATIONS

Our standard SDTM specification [2] is based on CDISC's standard. See Table 4 below for an example of our DM domain specification with a sample of variables. The Variable, Label, Type, Controlled Terminology, and Core columns come directly from the SDTMIG v3.4. Per the SDTMIG [6], the sources of SDTM variables are categorized by the origin of the data source in the Define-XML document file as "CRF", "Protocol", "Assigned", or "Derived". To support our SDTM automation, we've enhanced each SDTM specification with a new column, ***Derivation/Assigned***, to store either a simple line of SAS assignment code or a SAS macro name. This allows us to further customize code for the variables without needing to add extra code to ***%SDTM_Code_Generator***.

When *Origin* is "CRF Page xx", most SDTM variables (e.g., RFICDTC) can be directly mapped from raw dataset variables. When *Origin* is "Protocol", "Assigned", or "Derived", the *Derivation/Assigned* column can be utilized to further customize SAS macro code. In the case when one line of code is sufficient (e.g., STUDYID, USUBJID, etc.), we write the code directly in the *Derivation/Assigned* column, and the macro reads that in. However, some other standard SDTM variables require more lines of code and/or data steps to derive from other variables either within the same raw dataset or across multiple raw datasets. For coding efficiency, the derivation of these variables is generalized and grouped into a utility macro, and the name of that specific utility macro is included in the SDTM specifications (e.g., ***%get_trt***).

Variable	Label	Type	Length	Controlled Terminology	Origin	Core	Derivation/Assigned
STUDYID	Study Identifier	Char	20		Protocol	Req	STUDYID='Study-101';
DOMAIN	Domain Abbreviation	Char	2	DOMAIN	Assigned	Req	DOMAIN='DM';
USUBJID	Unique Subject Identifier	Char	40		Derived	Req	USUBJID=strip(STUDYID)|| strip(substr(SUBJECT,4));
SUBJID	Subject Identifier for the Study	Char	20		CRF Page 267	Req	SUBJID=strip(substr(SUBJECT,5));
RFSTDTC	Subject Reference Start Date/Time	Char	20	ISO 8601	Derived	Exp	%get_rfstdtc
RFENDTC	Subject Reference End Date/Time	Char	20	ISO 8601	Derived	Exp	%get_rfendtc
RFXSTDTC	Date/Time of First Study Treatment	Char	20	ISO 8601	Derived	Exp	%get_rfxstdtc
RFXENDTC	Date/Time of Last Study Treatment	Char	20	ISO 8601	Derived	Exp	%get_rfxendtc
RFICDTC	Date/Time of Informed Consent	Char	20	ISO 8601	CRF Page 5	Exp	
RFPENDTC	Date/Time of End of Participation	Char	20	ISO 8601	Derived	Exp	%get_rfpendtc
RACE	Race	Char	50	RACE	CRF Page 7	Exp	%map_race
ETHNIC	Ethnicity	Char	40	ETHNIC	CRF Page 7	Perm	
ARMCD	Planned Arm Code	Char	20		Assigned	Exp	%get_trt
ARM	Description of Planned Arm	Char	200		Assigned	Exp	%get_trt
ACTARMCD	Actual Arm Code	Char	20		Assigned	Exp	%get_trt
ACTARM	Description of Actual Arm	Char	200		Assigned	Exp	%get_trt
ARMNRS	Reason Arm and/or Actual Arm is Null	Char	80		Assigned	Exp	%get_trt
ACTARMUD	Description of Unplanned Actual Arm	Char	200		Assigned	Exp	%get_trt

Table 4. DM Domain Specification With a Sample of Variables

INTRODUCTION TO OUR SAS UTILITY MACROS

Table 5 lists five of our SAS utility macros dedicated to SDTM variables: --DTC, RACE, RACE1, …, RACE5, --SEQ, and --DY. A centralized SAS dataset named ***macrocalls*** stores all macro calls located in the macro library, except for %map_dtc_date and %map_dtc_time, which are directly called by ***%SDTM_Code_Generator***. Table 6 shows examples of the records in the macrocalls dataset for domains AE, DM, and LB.

SDTM Variable	Macro Name	Description	SDTM Domains	Example Macro Call
--DTC	map_dtc_date and map_dtc_time	Derive –DTC variables when there are partial dates	All Domains except for DM and SV	%map_dtc_date(_DATEVAR=AESTDTC, _RAWDATE=AESTDAT); %map_dtc_time(_DATEVAR=AESTDTC, _RAWTIME=AESTTIM);
RACE, RACE1, …, RACE5	map_race	Derive RACE variables for DM and SUPPDM	DM, SUPPDM	%map_race(_NUMFL=Y,_VAR=RACE1 RACE2 RACE3 RACE4 RACE5 RACE6);

SDTM Variable	Macro Name	Description	SDTM Domains	Example Macro Call
--SEQ	get_seq	Derive --SEQ variables based on provided key variables	All Domains except for DM and SV	%get_seq(_DOMAIN=LB, _SORTKEYS=STUDYID USUBJID LBCAT LBTESTCD VISITNUM LBDTC);
--DY	get_dy	Derive --DY variables based on provided --DTC variables	All Domains	%get_dy(_DATEVAR=LBDTC, _DAYVAR=LBDY);

Table 5. Examples of SAS Utility Macros for SDTM Automation

MACRO	MORD	DOMAIN	VARIABLE	MCALL
%get_seq	100	AE	AESEQ	%get_seq(_DOMAIN=AE,_SORTKEYS=STUDYID USUBJID AESTDTC AEDECOD AESPID);
%get_dy	102	AE	AEENDY	%get_dy(_DATEVAR=AEENDTC,_DAYVAR=AEENDY);
%get_dy	102	AE	AESTDY	%get_dy(_DATEVAR=AESTDTC,_DAYVAR=AESTDY);
%get_aetrtem	111	AE	AETRTEM	%get_aetrtem();
%map_race	2	DM	RACE	%map_race(_NUMFL=Y,_VAR=RACE1 RACE2 RACE3 RACE4 RACE5 RACE6);
%get_rfstdtc	105	DM	RFSTDTC	%get_rfstdtc(_DATA=EX1 EX2 EX3,_DATEVAR=EX1STDAT EX2STDAT EX3STDAT, _SUBJVAR=SUBJECT,_TIMEVAR=EX1STTIM EX2STTIM EX3STTIM);
%get_rfendtc	106	DM	RFENDTC	%get_rfendtc(_DATA=EX1 EX2 EX3,_DATEVAR=EX1ENDAT EX2ENDAT EX3ENDAT, _SUBJVAR=SUBJECT,_TIMEVAR=EX1ENTIM EX2ENTIM EX3ENTIM);
%get_rfxstdtc	107	DM	RFXSTDTC	%get_rfxstdtc(_ASSIGN=RFSTDTC,_DATA=,_DATEVAR=,_SUBJVAR=,_TIMEVAR=);
%get_rfxendtc	108	DM	RFXENDTC	%get_rfxendtc(_ASSIGN=RFENDTC,_DATA=,_DATEVAR=,_SUBJVAR=,_TIMEVAR=);
%get_rfpendtc	109	DM	RFPENDTC	%get_rfpendtc(_CUTOFFDT=&cutoffdt.,_DATEVAR=EOSDAT);
%get_trt	110	DM	ARMCD	%get_trt(_DRGCRIT=not missing(EX3STDAT),_DRGDATA=EX3,_SFCRIT=ENRSF_STD='N', _SFDATA=EN,_SUBJVAR=SUBJECT);
%get_seq	100	LB	LBSEQ	%get_seq(_DOMAIN=LB,_SORTKEYS=STUDYID USUBJID LBCAT LBTESTCD VISITNUM LBDTC);
%get_dy	102	LB	LBDY	%get_dy(_DATEVAR=LBDTC,_DAYVAR=LBDY);
%get_lobxfl	103	LB	LBLOBXFL	%get_lobxfl(_DATEVAR=LBDTC,_DAYVAR=LBDY,_DOMAIN=LB,_LASTVAR=LBTESTCD, _RESVAR=LBSTRESC,_SORTVARS=USUBJID LBCAT LBTESTCD LBDTC);
%get_blfl	104	LB	LBBLFL	%get_blfl(_DATEVAR=LBDTC,_DAYVAR=LBDY,_DOMAIN=LB,_LASTVAR=LBTESTCD, _RESVAR=LBSTRESC,_SORTVARS=USUBJID LBCAT LBTESTCD LBDTC);

Table 6. Examples of Records From SAS Dataset *Macrocalls* for SDTM Domains: AE, DM, and LB

%SDTM_Code_Generator merges the ***macrocalls*** data with the ***Derived/Assigned*** column in SDTM specifications by domain, variable name, and macro name. Then it generates the macro calls for each domain's SDTM mapping program using the ***MCALL*** variable. Display 4 shows the generated macro calls for DM.sas along with the programming comments.

```
51    ********  Programming Note: SDTM Variable: RFSTDTC Needs the Derivation by the Macro Call: %get_rfstdtc;
52    %get_rfstdtc(_DATA=EX1 EX2 EX3,_DATEVAR=EX1STDAT EX2STDAT EX3STDAT,_SUBJVAR=SUBJECT,_TIMEVAR=EX1STTIM EX2STTIM EX3STTIM);
53    ********  Programming Note: SDTM Variable: RFENDTC Needs the Derivation by the Macro Call: %get_rfendtc;
54    %get_rfendtc(_DATA=EX1 EX2 EX3,_DATEVAR=EX1ENDAT EX2ENDAT EX3ENDAT,_SUBJVAR=SUBJECT,_TIMEVAR=EX1ENTIM EX2ENTIM EX3ENTIM);
55    ********  Programming Note: SDTM Variable: RFXSTDTC Needs the Derivation by the Macro Call: %get_rfxstdt;
56    %get_rfxstdtc(_ASSIGN=RFSTDTC,_DATA=,_DATEVAR=,_SUBJVAR=,_TIMEVAR=);
57    ********  Programming Note: SDTM Variable: RFXENDTC Needs the Derivation by the Macro Call: %get_rfxendt;
58    %get_rfxendtc(_ASSIGN=RFENDTC,_DATA=,_DATEVAR=,_SUBJVAR=,_TIMEVAR=);
59    ********  Programming Note: SDTM Variable: RFPENDTC Needs the Derivation by the Macro Call: %get_rfpendt;
60    %get_rfpendtc(_CUTOFFDT=&cutoffdt.,_DATEVAR=EOSDAT);
61    ********  Programming Note: SDTM Variable: ARMCD Needs the Derivation by the Macro Call: %get_trt;
62    %get_trt(_DRGCRIT=not missing(EX3STDAT),_DRGDATA=EX3,_SFCRIT=ENRSF_STD ='N',_SFDATA=EN,_SUBJVAR=SUBJECT);
```

Display 4. SAS Code Generated by *%SDTM_Code_Generator* for the Macro Calls of the DM Domain

INTRODUCTION TO MEDIDATA'S ARCHITECT LOADER SPECIFICATION (ALS)

Medidata's Rave EDC (Electronic Data Capture) is widely used to build the EDC database for a clinical study. The Architect Loader Specification (ALS) is the document that Rave uses with metadata systems, and it provides information about how the database has been set up. One can duplicate the structure in another study database simply by customizing a pre-existing ALS and then importing the modified ALS into the new study database. Rave users can export an ALS directly from the Rave database. Table 7 shows an example of the *Forms* sheet from a study's ALS. The *OID* column shows the form names (EDC dataset names), and the *DraftFormName* column shows the label of each form. Table 8 shows an example of the *Fields* sheet from a study's ALS. The *FieldOID* column shows the variable names for the EOS form, along with their formats (*DataFormat*) and labels (*SASLabel*). Of note, the last column *VARIABLE TYPE* is added by the user and derived from column *DataFormat*. It is directly used for the derivation inside ***%SDTM_Code_Generator***.

OID	Ordinal	DraftFormName
SUBJ	1	Subject Registration
SV	2	Subject Visit
IC	3	Informed Consent
DM	4	Demographics
IE	5	Inclusion and Exclusion
DIA	7	Diagnosis
MH	8	Medical History
RADPRE	9	Prior Radiation Therapy
VS	18	Vital Signs
EG	22	12- Lead ECG - Single Timepoint
PK	25	Study Product PK

LBCHEM	35	Local Lab - Chemistry
UV	491	Unscheduled Subject Visit
EOS	494	End of Study

Table 7. An Example of the Forms Sheet From an ALS

FormOID	Ordinal	FieldOID	SASLabel / VARIABLE LABEL	DataFormat	VARIABLE TYPE
EOS	1	EOSDAT	End of Study Date	dd MMM yyyy	Date
EOS	2	EOSSTAT	Subject Disposition at the End of Study	$15	char
EOS	3	EOSREAS	Reason for End of Study	$40	char
EOS	4	EOSOTSP_O	Other, Specify	$200	char
EOS	5	EOSDEADT	Death Date	dd MMM yyyy	Date
EOS	6	PRCDTH	Primary Cause of Death	$20	char
EOS	7	EOSAESP_O	Adverse Event, Specify	$200	char
EOS	8	EOSNSRSP_O	Not Study Related, Specify	$200	char

Table 8. An Example of the *Fields* Sheet for the EOS Form From an ALS

An ALS is the repository for all raw dataset names and variable attributes for a study, which are some of the inputs for SDTM programming. CRF annotations set up the one-to-one mapping from each raw dataset variable in a CRF to its mapped SDTM domain variable or supplemental qualifier in each SDTM mapping program. If these could be directly used as the logic/mapping rules by a SAS macro, it would be more beneficial to the programming, and automation could be achieved. Hence, we store these annotations along with the metadata for raw datasets as described above in a single spreadsheet called "**master-annotation**", and this spreadsheet is "**semi-automatically**" developed per the availability of a study's ALS. This allows a SAS macro to simultaneously import all the mapping rules for all domains and utilize them in SDTM automation, instead of having multiple programmers individually annotate and develop programs for different SDTM domains.

INTRODUCTION TO OUR MASTER-ANNOTATION SPREADSHEET

As described above, we developed a spreadsheet file named **master-annotation.xlsx** as the repository of all raw dataset names and variable attributes (variable name, label, and type) as specified in the ALS along with CRF annotations mapping raw dataset variables to SDTM domains and their variables. Furthermore, extra columns are added to the file to aid the macro in automating SAS code generation. We start with variables (EDC DATASET NAME – VARIABLE LABEL) derived directly from the ALS as the foundation for the master-annotation as the ALS includes raw dataset names, variable names, labels, types, formats, and orders. Additional columns (SDTM DOMAIN – DECOD/TRT) are added in the master-annotation to facilitate mapping those raw dataset variables to the corresponding SDTM domains. These columns generally come from CRF Annotations. Extra columns (QLABEL – TRT ASSIGN) are designed to assist the automation for certain SDTM variables. Please see Table 9 for an example of how the RADPOST (Post-Treatment Radiation Therapy) form is annotated in the master-annotation and Table 10 for a summary of these key columns (variables) in the master-annotation.

From ALS / **From CRF Annotation**

EDC DATASET NAME	EDC DATASET LABEL	ORD.	VARIABLE TYPE	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE	CATEGORY	SUB CATE GORY	DECOD/TRT
RADPOST	Post-Treatment Radiation Therapy	1	char	RADPOSTYN_STD	Post-Treatment Radiation Therapy?	PR	[NOT SUBMITTED]	CONCURRENT RADIOTHERAPY		RADIOTHERAPY
RADPOST	Post-Treatment Radiation Therapy	2	Date	RADPOSTSTDAT	Date of First Dose	PR	PRSTDTC			
RADPOST	Post-Treatment Radiation Therapy	3	Date	RADPOSTENDAT	Date of Last Dose	PR	PRENDTC			
RADPOST	Post-Treatment Radiation Therapy	4	Numeric	RADPOSTTD	Total Dose	PR	PRDOSE			
RADPOST	Post-Treatment Radiation Therapy	5	char	RADPOSTTDU_STD	Total Dose Unit	PR	PRDOSU			
RADPOST	Post-Treatment Radiation Therapy	6	char	RADPOST_O	Other, Specify	PR	DOSSPEC in SUPPPR			
RADPOST	Post-Treatment Radiation Therapy	7	char	RADPOSTSR_STD	Site of Radiation	PR	PRLOC			
RADPOST	Post-Treatment Radiation Therapy	8	char	RADPOSTS_O	Other, Specify	PR	LOCSPEC in SUPPPR			
RADPOST	Post-Treatment Radiation Therapy	9	char	RADPOSTPU_STD	Purpose	PR	PURPOSE in SUPPPR			

Assisting the Macro to Automate the SAS Code Generation

EDC DATASET NAME	ORDER	QLABEL	QORIG	QEVAL	GRPID	TRT ASSIGN
RADPOST	1					
RADPOST	2					
RADPOST	3					
RADPOST	4					
RADPOST	5					
RADPOST	6	Total Dose, Other, Specify	CRF			
RADPOST	7					
RADPOST	8	Site of Radiation, Other, Specify	CRF			
RADPOST	9	Purpose	CRF			

Table 9. An Example of the Master-Annotation for the PR Domain (With Raw Dataset: RADPOST)

Column	Column Content		Origin	Manual?
EDC DATASET NAME	Raw dataset name		ALS	
EDC DATASET LABEL	Raw dataset label		ALS	
ORDER	The order of variables specified in CRFs, one of the key variables used to sort intermediate datasets generated by ***%SDTM_Code_Generator***		ALS	
VARIABLE TYPE	Variable type in raw dataset: Numeric, char, Date, Time, or Date & Time		ALS	Derived
VARIABLE NAME	Variable name in raw dataset		ALS	
VARIABLE LABEL	Variable label in raw dataset		ALS	
SDTM VARIABLE	SDTM variable name, SDTM variable name for a specific test, QNAM in supplemental domain, or not submitted		CRF Annotation	Y
CATEGORY	Text to assign –CAT. Applicable to domains: DS, EG, FA, LB, QS, TR, VS		CRF Annotation	Y
SUB CATEGORY	Text to assign --SCAT/FAOBJ. Applicable to domains: DS, EG, FA, LB, QS, TR		CRF Annotation	Y
DECOD/TRT	Text to assign --TRT/--TEST/DSTERM/DSDECOD. Applicable to domains: DS, FA, PR, TR		CRF Annotation	Y
QLABEL	Assign QLABEL in supplemental domains	Triplet to help map raw dataset variables in supplemental domains	User input (intended to assist ***%SDTM_Code_Generator*** with SAS code generation)	Y
QORIG	Assign QORIG in supplemental domains, with values: CRF, Derived, or Assigned.			
QEVAL	Assign QEVAL in supplemental domains, e.g., 'CLINICAL STUDY SPONSOR'			
GRPID	Column to define the group (block) within a raw dataset indicating that variables in the same group will be mapped to a specific intervention occurrence, event, measurement, or finding. Applicable to domains: DS, FA, PR, QS, TR, TU			
TRT ASSIGN	Column to aid automation and indicate extra coding is needed for the mapping of the variables. Applicable to all finding domains along with DS, FA, PR, and SV			

Table 10. Summary of the Key Variables in the Master-Annotation

The macro uses the variables above as its inputs to derive SDTM SAS programs. Table 11 shows an example of SDTM date variables --DTC, --STDTC, or –ENDTC along with raw dataset variables used to derive them. Display 5 shows SAS code from ***%SDTM_Code_Generator*** that is used to generate the SAS code for AE.AESTDTC and AE.AEENDTC in AE.sas. Of note, *SDTM VARIABLE* was renamed as *variable* for convenience inside the macro as shown on lines 4 and 6. Display 6 shows the output SAS code generated by Display 5 for AESTDTC and AEENDTC.

EDC DATASET NAME	EDC DATASET LABEL	ORDER	VAR. TYPE	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE
AE	Adverse Events	3	Date	AESTDAT	Start Date	AE	AESTDTC
AE	Adverse Events	4	Time	AESTTIM	Start Time	AE	AESTDTC
AE	Adverse Events	5	Date	AEENDAT	End Date	AE	AEENDTC
AE	Adverse Events	6	Time	AEENTIM	End Time	AE	AEENDTC
EOS	End of Study	1	Date	EOSDAT	End of Study Date	DS	DSSTDTC
EOS	End of Study	5	Date	EOSDEADT	Death Date	DM	DTHDTC
EX1	Lymphodepleting Chemotherapy: Fludarabine	8	Date	EX1STDAT	What was the treatment start date?	EX	EXSTDTC
EX1	Lymphodepleting Chemotherapy: Fludarabine	10	Date	EX1ENDAT	What was the treatment stop date?	EX	EXENDTC
LBCHEM	Local Lab - Chemistry	2	Date	LBDAT	Date of Collection	LB	LBDTC
VS	Adverse Events	2	Date	VSDAT	Date of Collection	VS	VSDTC

Table 11. A Sample of Raw Dataset Date/Time Variables and Their Mapped SDTM Variable Names From the Master-Annotation

```
1 if (strip(variable_type)='Date' and index(variable_label,'Date')) or
2    (strip(variable_type)='Time' and index(variable_label,'Time')) then do;
3    if substr(reverse(strip(variable_name)),1,3)='TAD' then
4       lines='        '||'%map_dtc_date(_DATEVAR='||strip(variable)||',_RAWDATE='||strip(variable_name)||');';
5    else if substr(reverse(strip(variable_name)),1,3)='MIT' then
6    lines='        '||'%map_dtc_time(_DATEVAR='||strip(variable)||',_RAWTIME='||strip(variable_name)||');';
7 end;
```

Display 5. SAS Code from *%SDTM_Code_Generator* Used to Generate SAS Code Inside AE.sas for AESTDTC and AEENDTC

```
1 %map_dtc_date(_DATEVAR=AESTDTC,_RAWDATE=AESTDAT);
2 %map_dtc_time(_DATEVAR=AESTDTC,_RAWTIME=AESTTIM);
3 %map_dtc_date(_DATEVAR=AEENDTC,_RAWDATE=AEENDAT);
4 %map_dtc_time(_DATEVAR=AEENDTC,_RAWTIME=AEENTIM);
```

Display 6. SAS Code to Map Raw Dataset Variables to AESTDTC and AEENDTC Inside AE.sas

From the example above, the macro uses the columns from the master-annotation for derivation, instead of needing to specify individual variable names from the raw datasets. This allows for the macro to be used in multiple studies, even if their EDC databases are built from different vendors. The vertical structure of raw dataset variable names and their attributes in a master-annotation provides the macro with an advantage over SDTM mapping templates. The macro uses a single column ***VARIABLE NAME*** to read each raw dataset variable name one by one to generate a SDTM SAS

program for a domain. When the raw dataset variable names change, there is very little impact on the macro as the changes are automatically reflected in the master-annotation's *VARIABLE NAME* column.

HOW TO HANDLE EXTERNAL DATASETS

A clinical trial usually has some external data, e.g., central safety lab, biomarkers, imaging data (MRI/CT, PET scan) from Central Imaging Services, etc. They are typically stored outside the EDC database, and their metadata are specified by Data Transfer Agreements (DTA) from different vendors. The finalization of DTAs and the first data transfer usually come much later than the first EDC raw data extract. The approach in this paper focuses on dealing with CRF data, not external data. The main reasons to exclude external data for SDTM automation are the timing of its availability (for both metadata and actual data) and simplifying the development of the macro to balance the level of automation with the cost of meeting timelines. When the external datasets are ready for inclusion, the team can decide if the new programming should be added to either the ***%SDTM_Code_Generator*** or the related individual SDTM SAS program. The decision requires balancing the generalization of the macro for future use with the spending of more time/resources in updating the macro and its potential impact of timelines.

HOW TO LEVERAGE THE EXISTING MASTER-ANNOTATION AND *%SDTM_CODE_GENERATOR* FOR A NEW STUDY

One can leverage the existing master-annotation as an automation template for new studies. We completed SDTM programming for two studies, and their EDC databases were both built by Medidata's Rave. Let us name them as Study-101 and Study-102, respectively. SDTM programming for Study-101 was first completed at the very early stage of the study. Hence, its master-annotation-101.xlsx and ***%SDTM_Code_Generator*** had been fully developed. Before starting to work on SDTM automation for Study-102, we compared its ALS with Study-101's and got the following five output files shown in Table 12.

Output File Name	Output File Label	Function
F1	Common variable names from common datasets	Identify discrepancies in variable attributes, which could potentially impact the macro for Study-102. e.g., raw dataset variable IE.IETESTCD is a character variable in Study-101 but numeric in Study-102.
F2	All variable names only included in Study-101	Identify variables potentially being omitted from Study-102. e.g., raw dataset variables UV.UVREAS and UV.UVREAS_O ("Reason for Unscheduled Visit" and "Other, Specify") were in Study-101 but not in Study-102.
F3	All variable names only included in Study-102	Identify variables that need new annotations. e.g., CM.CMDOSFRM, ICE.ICETOTAL ("ICE Total Score") were added to the CM and ICE forms in Study-102.
F4	All variable names only included in Study-101 among common datasets	Identify variables with different variable names from the same CRF. e.g., ICE.IAYN ("Was ICE Assessment performed?") from Study-101 vs. ICE. ICEPERF from Study-102.
F5	All variable names only included in Study-102 among common datasets	

Table 12. Five Outputs from the Comparison of ALSs between These Two Studies

Table 13 and Table 14 show the similarities (same CRF names and same variable names from the same CRF) and dissimilarities of CRFs for these two studies. Out of 80 CRFs in Study-101 and 67 CRFs in Study-102, there were 50 common CRFs between the two studies, such as Adverse Events, Concomitant Medications, Demographics, Enrollment, End of Study, Medical History, Vital Signs, etc. Not surprisingly, they are from standard safety domains.

Study Number	Number of CRFs	Number of Common CRFs	Number and Percentage of Unique CRFs	Total Number of Variables
Study-101	80	50 (62.5%)	30 (37.5%)	906
Study-102	67	50 (74.6%)	17 (25.4%)	678

Table 13. Tabulation of CRFs From Two Studies

Study	Total Number of Variables	Number and Percentage of Common Variables	Number and Percentage of Unique Variables
Study-101	906	410 (45%)	496 (55%)
Study-102	678	410 (60%)	268 (40%)

Table 14. Tabulation of CRF Variables From Two Studies

The **EDC DATASET NAME** (*FormOID*) and **VARIABLE NAME** (*FieldOID*) were combined as the key to merge the ALS of Study-102 with master-annotation-101.xlsx, bringing in the other columns (CRF annotations and other variables as specified in Table 10) of master-annotation-101.xlsx for CRFs and variable names that were common to these two studies. This newly augmented file was used as a starting point to complete master-annotation-102.xlsx, and users

only needed to fill in the other columns (e.g., CRF annotations) for new CRFs and variables that were unique to Study-102. Table 15 below shows an example of master-annotation-102.xlsx with the first column indicating the variables that need additional manual work to complete their master-annotation record.

Need New Annotation	EDC DATASET NAME	SDTM DOMAIN	EDC DATASET LABEL	Order	VARIABLE NAME	VARIABLE LABEL	SDTM VARIABLE
	EN	DS	Enrollment	1	ENRSF_STD	Was Subject Enrolled?	[NOT SUBMITTED]
	EN	DS	Enrollment	3	ENRDAT	Enrollment Date	DSSTDTC
	EN	DS	Enrollment	4	ENPHASE_STD	Study Phase	PHASEENR in SUPPDS
	EN	DS	Enrollment	5	ENPART_STD	Study Part	PARTENR in SUPPDS
Y	EN	DS	Enrollment	6	ENCOHRT_STD	Study Cohort	COHORT in SUPPDS
	EN	DS	Enrollment	8	ENSFDAT	Screen Fail Date	DSSTDTC
	EN	DS	Enrollment	9	ENRSP_STD	Screen Failure Reason	DSTERM
	EN	DS	Enrollment	10	ENRESCR_STD	Was Subject Re Screened?	SUBJRESC in SUPPDS
	ECHO	FA	Echocardiogram / MUGA	1	ECHOYN_STD	Was ECHO or MUGA performed?	[NOT SUBMITTED]
Y	ECHO	FA	Echocardiogram / MUGA	2	ECHOMETH_STD	If Yes, method of assessment performed?	ECHOMETH in SUPPFA
	ECHO	FA	Echocardiogram / MUGA	3	ECHODAT	Test Date	FADTC
	ECHO	FA	Echocardiogram / MUGA	4	ECHOORRES	Ejection Fraction	FAORRES when FATESTCD = LVEF
	ECHO	FA	Echocardiogram / MUGA	5	ECHOORESU_STD	Ejection Fraction Units	FAORRESU

Table 15. An Example of Master-Annotation-102.xlsx

Per Table 14, we had 410 variables that were in both Study-101 and Study-102 and 268 variables unique to Study-102 that needed manual work to complete master-annotation-102.xlsx. Thus, 60% of all variables for Study-102 were "borrowed" from Study-101, and only 40% of the variables required extra manual work for master-annotation completion. (The majority of that 40% was from the CRFs for efficacy data.) By utilizing the existing master-annotation for Study-101, huge time savings and high efficiency were achieved for Study-102! Higher standardization of CRFs could contribute to even more high-quality programming efficiency across studies!

Once the master-annotation and SDTM specifications are finalized by leveraging the method introduced in the previous section, ***%SDTM_Code_Generator*** can then be adapted to new studies for SDTM automation. The SAS code for ***%SDTM_Code_Generator*** should be carefully checked for mentions of the variables identified from the ALS comparison (Table 12), especially those from F1, F3, F4, and F5 which could potentially require some SAS coding updates. Variables from F2 should not have any impact. The macro's output files (i.e., SDTM mapping SAS programs) should also be carefully reviewed, and the programming validation process should be strictly followed. Our working experience is that there was almost no change to the macro for the safety domains (except for IEDTC in IE.sas due to the difference between two EDC database builds), but some changes had been made to the efficacy domains such as RS, TR, and TU.

INTRODUCTION TO THE SCALABILITY OF OUR NEW PRACTICAL APPROACH

So far, we have illustrated this new practical approach to automating SDTM. For the first study, one needs to develop the master-annotation spreadsheet (but can leverage existing CRF specifications) and ***%SDTM_Code_Generator*** from scratch. However, it is still a more efficient and less error-prone process than the SAS template programs suggested in Table 1. Once one has fully developed the master-annotation and macro for a clinical study, one can adapt them for new studies. The ease of adaptability depends on the similarity of CRF designs and specifications of new studies compared to the first study.

In the case where new studies use the same EDC vendor, we'd expect CRF form design and specifications to be relatively similar, especially for safety data. Thus, we can easily leverage and adapt the existing master-annotation and ***%SDTM_Code_Generator*** for those new studies. If the CRF design for efficacy data is different, then new programming may need to be added to the macro. In the case where new studies use a different EDC vendor, CRF form design should be somewhat similar but the actual CRF specifications would be different. More manual work would need to be done to update the SDTM automation tools, in particular the master-annotation spreadsheet, to account for the different CRF specifications. The user could still leverage CRF specifications and existing annotations but would need to manually fill in key columns from Table 10 where the origin is not "ALS". ***%SDTM_Code_Generator*** may also need to be updated to account for new or different efficacy data.

INTRODUCTION TO OUR VALIDATION PROCESS FOR SDTM PROGRAMMING

The development of ***%SDTM_Code_Generator*** starts only after CRF annotations and SDTM specifications pass the review and validation process as they are the inputs of the macro as shown in Figure 2. The traditional approach for SDTM dataset validation requires programmers to develop an independent mapping SAS program. This double programming requires more resources and time since it essentially doubles development efforts. Our new SDTM programming validation (Figure 4) consists of the following three steps: code reviewing, real data testing, and developing independent mapping SAS programs to validate relatively complicated SDTM datasets as needed per the team's decision. This validation process validates both the macro and each SDTM mapping SAS program.

When each SDTM mapping SAS program (e.g., DM.sas) is generated from the macro call, the macro developer and users work together to review the code to identify bugs before the testing phase until they make certain that the coding logic meets domain requirements. Once raw data are available, each SDTM mapping SAS program is tested by using

real data. Users walk through each data block to make sure that the execution is as expected and meets the requirement. This step also includes retesting after bug fixing, and this could repeat several times for the accumulating real data while the study is ongoing until the user ensures that the SAS program is thoroughly tested and meets the requirements of the domain. The team also identifies and decides which SDTM domains need traditional totally independent programming validation. For example, the TR (Tumor/Lesion Results) domain from one study included data from 13 CRFs, which had 190 variables in total, so we developed an independent SAS program to validate the TR domain.

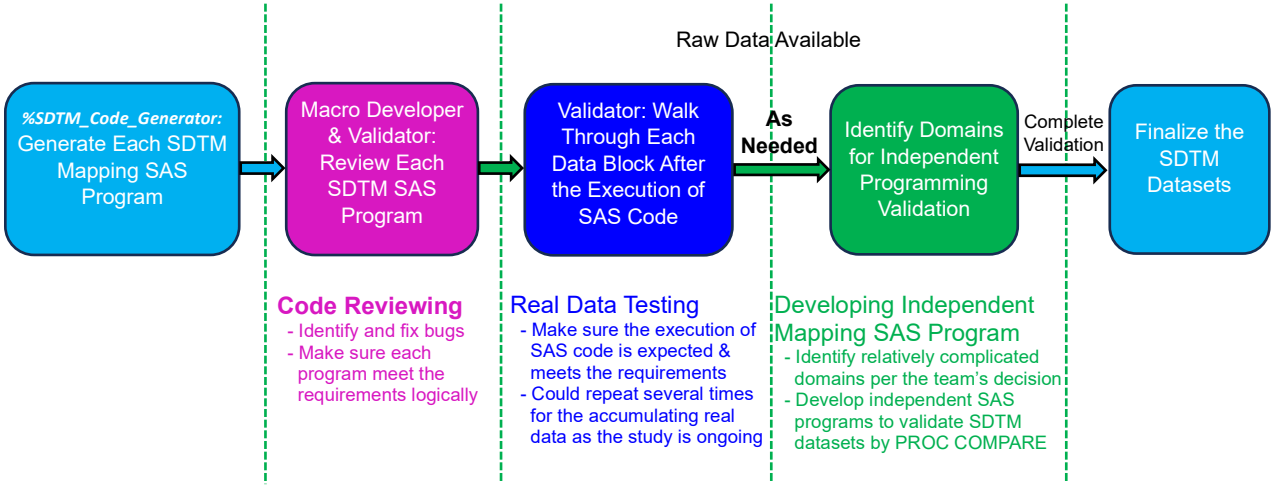

Figure 4. The Logic Flow of Our SDTM Programming Validation Process

Solid SDTM programming expertise and working experience from the macro developer and the users can shorten the development process and is the key to high quality delivery of SDTM datasets. This new approach to automating SDTM is user-friendly as users can directly review the output code (instead of facing a "black box") and test it with real data, ensuring that each SDTM dataset they produce is of the highest quality. Since only a fraction of SDTM datasets need independently developed SAS programs for programming validation, a lot of time and resources are saved compared to the traditional way of validating all SDTM datasets or the situation where a sponsor must have an in-house or outsourced SDTM programming team independently develop SDTM SAS programs to validate the automated SDTM datasets provided by vendors.

HOW TO GUARANTEE ALL RAW DATASET VARIABLES ARE MAPPED INTO SDTM

Another functionality of the macro ***%SDTM_Code_Generator*** is that it can automatically detect any raw dataset variables unmapped in SDTM. As mentioned previously, the SAS code generated by our macro is saved in a SAS dataset before it is output to a SDTM mapping SAS program. This SAS dataset contains all variables from the master-annotation specified in Table 10 in addition to the previously mentioned *lines* and *_order* variables. Each call of the macro merges this dataset with the master-annotation by *EDC DATASET NAME* and *VARIABLE NAME* for a specific domain. Any records that have non-missing values for *EDC DATASET NAME* and *VARIABLE NAME* but are missing *lines* are warning signs that those raw dataset variables may not have been mapped or included in the SDTM mapping SAS program. One exception would be the records where *SDTM VARIABLE* = "[NOT SUBMITTED]", which marks raw dataset variables that are intentionally not submitted. Table 16 shows an example of raw dataset variable AENOW from the AE form, whose omission was detected by the macro. However, as indicated by its *SDTM VARIABLE* column, AENOW was intentionally not mapped to any SDTM datasets.

EDC DATASET NAME	EDC DATASET LABEL	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE
AE	Adverse Events	AENOW	Form last updated	AE	[NOT SUBMITTED]

Table 16. Raw Dataset Variable(s) That Are Not Mapped to SDTM AE as Identified by the Macro

HOW TO HANDLE EDC DATABASE CHANGES

It is very typical for the EDC database to change due to a variety of reasons, such as protocol changes, EDC database build errors, etc. The newly updated ALS or CRF specifications are provided along with a document, such as a "**Database Change Request Form**". The simple solution is to use SAS programming to compare the new CRF specifications to the original one. The output file can help the team pinpoint the changes to examine the impact on SDTM programming. The worst-case scenario is to consider it as a new study. The previous sections present how to leverage the existing master-annotation and ***%SDTM_Code_Generator*** for a new study. We experienced a situation where one study's ALS was updated three months after the EDC database was in place. The comparison between these two ALS files showed that two safety lab tests had been added and one variable's label had been changed. We manually added these tests into master-annotation and reran the macro to generate LB.sas. After reviewing the

mapping sections for these two tests in the output LB.sas and confirming that they met our requirements, we finished our update process for SDTM LB programming.

CONCLUSION

This paper presented a new approach to automating SDTM using a metadata-driven method that leverages CRF specifications and SDTM standards. We compared the workflow between our new approach and the standard one. We introduced our master-annotation spreadsheet, which leverages CRF specifications from an EDC database (in particular, the Architect Loader Specification of Medidata's Rave EDC), and our macro ***%SDTM_Code_Generator***. We discussed our experience on two different types of oncology studies and demonstrated the practicality of our new approach through its efficiency, flexibility, transparency, and scalability. We are confident that the macro will become more mature as our new approach is applied to more studies down the road. The intent of this presentation is to share our ideas with readers to aid them in automating SDTM with much more efficiency and higher quality that is applicable across multiple clinical studies within an organization.

REFERENCES

[1] U.S. Department of Health and Human Services, Food and Drug Administration, Study Data Technical Conformance Guide: Technical Specifications Document. October 2023. Available at https://www.fda.gov/media/153632/download

[2] Xiangchen (Bob) Cui; Scott Moseley; Min Chen. A Cost-Effective SDTM Conversion for NDA Electronic Submission. Proceedings of the Pharmaceutical SAS® Users Group Conference, PharmaSUG 2011

[3] Xiangchen (Bob) Cui; Hao Guan, Min Chen, and Letan Lin. Automate the Process to Ensure the Compliance with FDA Business Rules in SDTM Programming for FDA Submission. DIA 2018 Global Annual Meeting Professional Poster; Proceedings of SAS Global Forum 2019; Proceedings of the Pharmaceutical SAS® Users Group Conference, PharmaSUG 2019

[4] Roman Radelicki; Swapna Pothula. End to End SDTM Automation: A Metadata Centric Approach. PhUSE US Connect 2019

[5] FDA Validator Rules. December 2022. Available at https://www.fda.gov/industry/fda-data-standards-advisory-board/study-data-standards-resources

[6] CDISC Study Data Tabulation Model and SDTMIG v3.4 at http://www.cdisc.org/sdtm

[7] A New Approach to Automating the Creation of Subject Visits (SV) Domain, under preparation for 2024 conference

ACKNOWLEDGMENTS

Appreciation goes to PK Morrow for her invaluable review and comments.

CONTACT INFORMATION

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

Author Name: Xiangchen (Bob) Cui, Ph.D.
Company: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 908-240-4086
E-mail: xiangchen.cui@crisprtx.com

Author Name: Min Chen
Company: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 857-928-4347
E-mail: min.chen@crisprtx.com

Author Name: Jessie Wang
Company: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 214-668-2107
E-mail: jessie.wang@crisprtx.com