PHUSE EU Connect 2021

Paper SI03

Advanced Excel Specification for Automatic SDTM Generation

Ilya Mishchenko, Atlant Clinical Ltd., Moscow, Russia
Aleksandra Gorbunova, Atlant Clinical Ltd., Moscow, Russia
Evgeny Dmitriev, Atlant Clinical Ltd., Moscow, Russia
Timur Zagidullin, Atlant Clinical Ltd., Moscow, Russia

ABSTRACT

Being one of the most important parts of clinical trials submission to regulatory authorities SDTM gives significant advantage to all stakeholders via providing a predictable and clear structure for data accumulation and further analysis. Comparing to ADaM or TFLs, SDTM seems to be a less sophisticated part of deliverables with defined non-extensible types of domains and variables. Also SDTM requires just few and straight-forward derivation types from raw data.

We propose a way for automation of SDTM generation by extending standard Excel specification to a structure acceptable for both human and machine interpretation and implementation.

We describe the self-sufficient Excel-document architecture for automatic generation of SDTM domains with additional variables and sheets for SAS-code instructions and external metadata. We also present our attempt at classifying the whole set of SDTM domains by unified programming groups and applying standard SAS arrays and user-defined macro-arrays for domains building.

INTRODUCTION

Clinical trials have been steadily growing in complexity and accuracy, so for any clinical trial started after December 17, 2016 FDA requires the use of CDISC and SDTM standards when preparing datasets for submission. Building SDTM datasets that pass all quality control checks is a very arduous task solely due to large volume of efforts and human-hours needed to complete it.

As a result, numerous attempts have been made to delegate some of the processes for SDTM datasets generation to computers and algorithms: from utilizing special meta data approaches [1, 2] to applicating artificial intelligence and voice recognition [3].

This paper presents another way, namely combining SDTM specification and SAS program needed for SDTM datasets generation. To achieve this goal, we embed specially organized SAS-code instructions directly into Excel specification which allow us to automate thoroughly the subsequent process of SAS program creation.

We describe the methodology and specifics of our approach through the following steps.

1. We start with a brief overview of information-flow process in the life circle of a clinical study project.
2. Thereafter we consider SDTM generation block by the means of standard and extended Excel specifications in more details. We describe standard Excel specification structure with separate lists of required domains, variables within each of the domains as well as set of predefined CRF data-independent domains.
3. Based on this we present our approach to automation of SDTM generating process with the help of embedding SAS-language instructions directly in the specification to obtain both human and machine interpretable document. This may be done by organizing required instructions in additional columns containing variable definition, data step numbers for each variable creation with upper and lower borders for each data step.
4. We also propose our attempt to classify all diversity of SDTM generating instructions onto three distinct types: straightforward derivations, combining required variables into SAS-arrays and organizing standard SAS-loops over them and, lastly, involving user-defined macro-arrays of datasets and required parameters and performing macro-loops calculation for their creation.
5. Finally, we discuss the advantages of our approach as compare to similar works performed in the last few years and present its further generalization to ADaM and TFL generation.

METHODS

INFORMATION-FLOW PIPELINE IN THE PROJECT LIFE CYCLE

Typical project workflow (Fig. 1) is considered as a basis for further discussion.

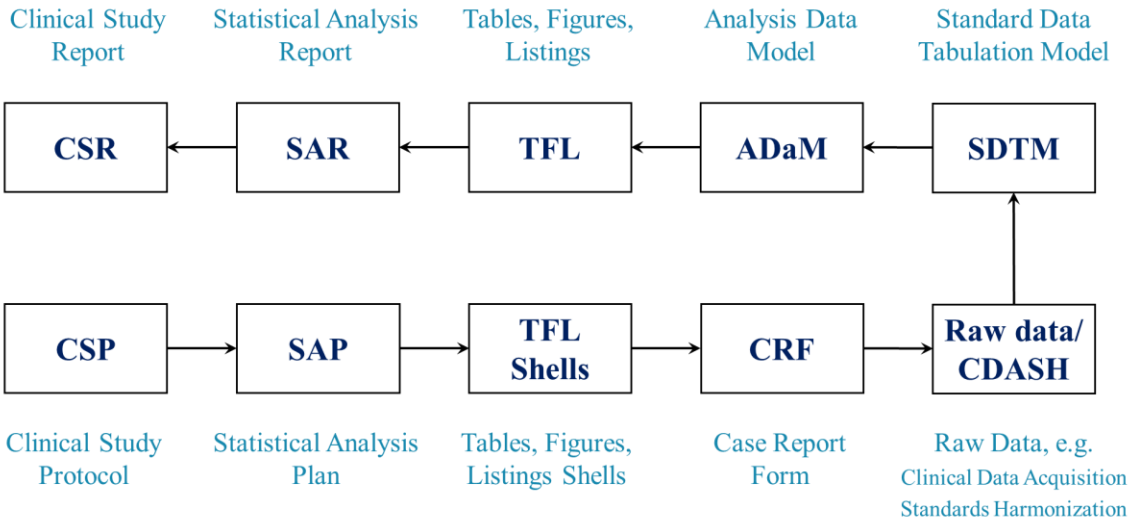

Figure 1. General framework of information-flow pipeline in the project life cycle

As usual the protocol goes first, which serves as starting point for SAP and TFL shells creation. After that CRF is designed and study data is collected in a database. Using this raw data SDTM and ADAM datasets is consequently generated. Next step is programming of TFL followed by SAR and CSR writing. Here we concentrate on creation SDTM from raw data with the help of automatic approach.

Fig. 2 shows the scheme of the proposed process. First of all, we create SDTM specification in Excel format according to SDTM IG and project-specific CRF design. In a standard way the next step is a manual programming of domains based on created specification. But in our approach we additionally write SAS-instructions for every variable directly in the Excel specification. After that we read this Excel file via SAS and combine this instruction on one SAS program. If some domains are too complicated to describe each variable in a few SAS instructions in Excel file we can write the code directly in a final SAS program. We call this process as “manual tuning”. Finally, we use raw data as input for this program to generate SDTM datasets.

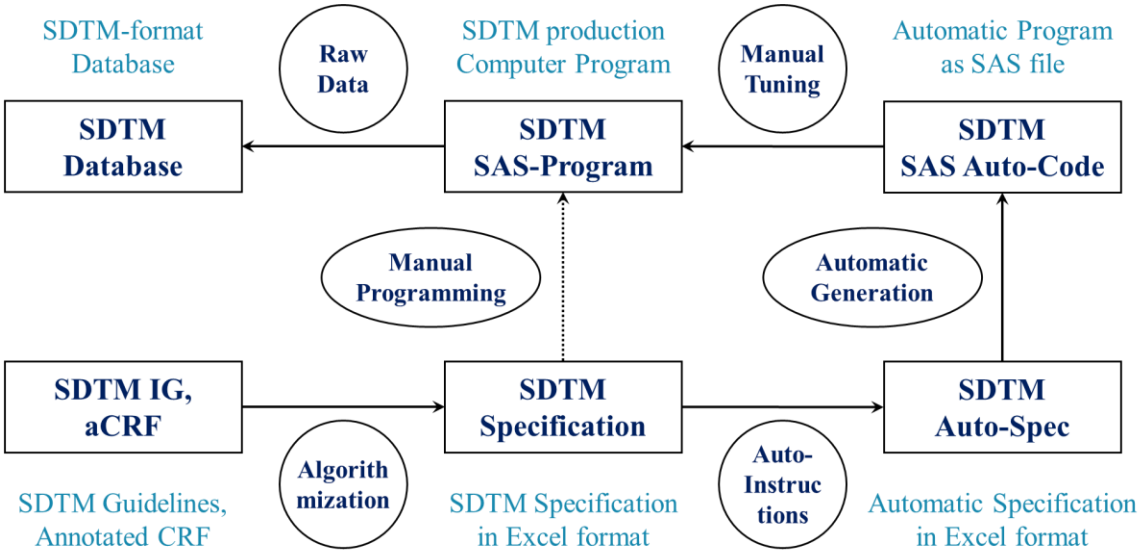

Figure 2. SDTM-generating “junction” via standard vs. extended Excel specification

SDTM-GENERATING VIA STANDARD EXCEL SPECIFICATION

Now let us look at the first step, namely constructing of an Excel specification. Our Excel file contains several sheets, and first of them is a sheet with a list of domains which should be created in a study. There we have columns for sequential number for each of the domains, their codes and names and in addition to standard specification we add one special column which defines the order in which the domains should be programmed. In many cases particular order of their production is not important, though there are several exceptions when some of the domains should be ready before the other ones. For example, DM variables is often used for calculation variables in other domains, so DM has number 1 in this sequence. Also, any parent domain should be finished before starting correspondent supplementary one.

The next sheet defines all variable for all required domains. The values for first several columns arise from implementation guide: they are variable name, label, type and so on, while the others depend on the study protocol. For example, one of these columns indicates weather the protocol, other SDTM domain or a raw dataset is used as an origin for a variable. Also, we specify a source variable within this origin and comments how the variable should be calculated. E.g., we can simply use values of source variable as they are, that is direct mapping type, or perform some procedures such as renaming, formatting or more complicated derivations.

Besides that, we create separate sheets in the Excel file for those domains which do not depend on data in CRF or other data sources updated during the study. For example, trial domains are constructed according to the protocol and can be created before the start of patient recruitment. Another example is RELREC domain which requires information only from the specification to identify relationships between SDTM datasets. We exclude these domains from further consideration because they may be import directly from Excel file by a SAS program and then export in SAS or Excel format together with other domains which will be produced in a more complicated way. We describe that way on the next sections.

RESULTS

EXTENSION OF THE STANDARD SPECIFICATION WITH SAS-INSTRUCTIONS

To transfer to a machine-reading document we enriched variable sheet of the standard specification with an additional column containing local SAS instructions for each variable creation (Table 1).

Table 1. Example of extension of Variables sheet in standard specification with Code Substitutions column

Seq. For Order	Domain	Variable Name	Variable Label	Type	Code Substitutions
30	AE	AESMIE	Other Medically Important Serious Event	Char	AESMIE = ifc (AESER2Y = "Other Medically Important Event", "Y", "");
31	AE	AESTDTC	Start Date/Time of Adverse Event	Char	AESTDTC = AESTDAT_DTS;
32	AE	AEENDTC	End Date/Time of Adverse Event	Char	AEENDTC = AEENDAT_DTS;
33	AE	AESTDY	Study Day of Start of Adverse Event	Num	AESTDY = %DateInt (RFICDTN, datepart(AESTDAT));
34	AE	AEENDY	Study Day of End of Adverse Event	Num	AEENDY = %DateInt (RFICDTN, datepart(AEENDAT));

On that we try to stick to the standard right-to-left assignment form of variable derivation where left part presents required variable while right part contains its definition. For this purpose such novel SAS functions as *ifc* (), *whichc* () and *choosec* () for character parameters or *ifn* (), *whichn* () and *choosen* () for their numerical analogies turns out to be very helpful.

Another instrument to achieve code brevity may be extracting universal formulas as macro-substitutions and organizing them in a separate Macro sheet as it is done for date interval calculation on the example below (Table 2).

Table 2. Example of user-define macro-substitution in a separate Macro sheet

Macro Functions

```
                        /* Date Interval */
%macro DateInt (StDate, EnDate);   /* Start Date, End Date */
  &EnDate. - &StDate. + (&EnDate. >= &StDate.)
%mend;
```

Meanwhile local code instructions are not enough for fully automatic processing. We also need to decide within which data step a particular variable should be created and explicitly specify upper and lower border instructions for this data step. This usually may be done in universal *data-merge-keep* form with subsequent *proc sort* and *run* statements (Table 3).

Table 3. Example of extension of Variables sheet in standard specification with Data Step and Code Surroundings columns

Seq. For Order	Domain	Variable Name	Variable Label	Type	Data Step	Code Surroundings
					data	Libr.&&SDTMName&d..1;
					merge	Libr.&&SDTMName&d..0
						Libr.in_ae;
					keep	%LoopSDTM_InVar_n (&d.,
						%nrstr(&&SDTMVarName&d._&n..))
						AESTDAT AEENDAT
			Reported Term for the Adverse			/* For Further Calculations */
						VMEDDRA AEDLT AEIM;
						/* For SUPPAE Domain */
5	AE	AETERM	Event	Char	1	/* First One-String File merge, no by-statement */
			End Date/Time of Adverse			proc sort; by &InIDVar. AEDECOD AESTDTC;
32	AE	AEENDTC	Event	Char	1	run;

Moreover, we use agreed notation for required domains and variables within current domain so that they may be organized with the help of user-defined macro-loops for further automation of the whole process. This domains and variables macro-arrays which depends on single and double macro-indexes respectively are read directly via specification analysis.

Macro-loops over variables within current SDTM domain may be designed in the form of SAS *%do*-loops with appropriate masking of input parameter by *%nrstr* () function for macro-indexes freezing and subsequent unfreezing them in the body of *%do*-loop by *%unquote* () function (Table 4).

Table 4. Example of macro-loops over SDTM Variables in a separate Macro sheet

Macro Functions

```
                  /* Macro-Loop over SDTM Variables */
%macro LoopSDTM_Var_n (d, MaskStat);     /* Domain Number,
                                            Statement Masked by %nrstr() */
 %do n = 1 %to &&SDTMVarNum&d..;         /* From Specification Analysis */
  %unquote (&MaskStat.)
 %end;
%mend;
```

```
    /* Macro-Loop over SDTM Variables with Input Datasets ID-Variable */
%macro LoopSDTM_InVar_n (d, MaskStat);   /* Domain Number,
                                            Statement Masked by %nrstr() */
  &InIDVar.                              /* Input-ID  Variable */
  %LoopSDTM_Var_n (&d., &MaskStat.)
%mend;
```

STRAIGHTFORWARD DERIVATIONS

In this section we present our attempt to classify variable-deriving instructions for SDTM production into finite number of types.

The first and simplest type of such instructions is straightforward derivations in the case of one-to-one rows production which in turn subdivides into inheriting from a parent dataset, renaming and direct calculations (Table 5).

Table 5. Example of straightforward derivations (inheriting, renaming and direct calculations)

Seq. For Order	Domain	Variable Name	Variable Label	Type	Code Substitutions
5	AE	AETERM	Reported Term for the Adverse Event	Char	/* Automatically Filled-In */
7	AE	AELLT	Lowest Level Term	Char	AELLT = LLT_NAME;
19	AE	AESER	Serious Event	Char	AESER = put (AESER, $CTFmtYN.);

The latest one may be performed with the help of universal macro-substitutions or formats stored in separate sheets of designed specification (Table 6).

Table 6. Example of user-define format in a separate Formats sheet

Project Formats
proc format;
value $ CTFmtYN /* Yes/No */ "Yes" = "Y" "No" = "N" ;

SAS-ARRAYS AND DO-LOOPS

Another type of variable-deriving instructions arises through one-to-many rows generation, for example while constructing supplementary domains.

In this case it is natural to organize SAS-arrays of a fixed length for multiple qualifier names, labels and parent variables whose values will be stored in SUPP-domain (Table 7). After that we include them into a single *do*-loop ended by *output* instruction for multiplying records in the output domain (Table 7).

Table 7. Example of SAS-arrays of variables and standard do-loops calculations

Seq. For Order	Domain	Variable Name	Variable Label	Type	Data Step	Code Surroundings
						data Libr.&&SDTMName&d..1; merge Libr.&&SDTMName&d..0 Libr.AE_; keep %LoopSDTM_InVar_n (&d., %nrstr(&&SDTMVarName&d._&n..)); /* First One-String File merge, no by-statement */ %let NVal = 3;
4	SUPP AE	IDVAR	Identifying Variable	Char	1	/* Number of Assigned Variables Values */

Seq. For Order	Domain	Variable Name	Variable Label	Type	Code Substitutions
6	SUPP AE	QNAM	Qualifier Variable Name	Char	array NameVal [&NVal.] &CharVarFormat. _temporary_ ("AEDICT" "AEDLT" "AEIMMU");

						array LabelVal	[&NVal.] &CharVarFormat. _temporary_ (	
			Qualifier				"Dictionary Version"	
	SUPP		Variable				"Is this a suspected DLT?"	
7	AE	QLABEL	Label	Char			"Is adverse event immune related?");	
	SUPP		Data			array ValueVar	[&NVal.] &CharVarFormat.	
8	AE	QVAL	Value	Char			VMEDDRA AEDLT AEIM ;	

Seq. For Order	Domain	Variable Name	Variable Label	Type	Code Substitutions	Data Step	Code Surroundings
					do i = 1 to &NVal.;		
					QNAM = NameVal [i];		
					QLABEL = LabelVal [i];		
					QVAL = ValueVar [i];		
					QEVAL = ifc (QNAM in		
					("AEDLT", "AEIMMU"),		
					"INVESTIGATOR", "");		
	SUPP				output;		
10	AE	QEVAL	Evaluator	Char	end;	1	run;

Here as before, a separate Constants sheet may be appropriate for storing universal lengths and formats for character and numerical variables (Table 8).

Table 8. Example of user-define constants in a separate Constants sheet

Project Constants
/* Numerical and Character Variable Lengths and Formats */
*%let NumVarLength = 8;
%let CharVarLength = 200;
%let NumVarFormat = Best.;
%let CharVarFormat = $&CharVarLength..;

MACRO-ARRAYS AND %DO-LOOPS

The third and most complex type of derivations arises through one-to-one or one-to-many rows production process when multiple initial datasets are combined into a single domain. In this case not only required parameters, but also initial datasets should be organized as a sequence of macro-constants and be treated in turn via user-defined macro-loops.

As a typical example we may consider LB domain production based on a number of datasets each containing data of laboratory analysis of a particular type (such as hematology, chemistry and so on). It is useful to list attributes of all analysis datasets (their codes, labels, and dataset names) as a separate sheet in the extended Excel specification (Table 9).

List of parameters for each analysis type may be stored in another sheet of the specification as a two-dimensional set (e.g., hematocrit, hemoglobin, etc. for hematology, albumin, alkaline phosphatase, etc. for chemistry and so on, see Table 9).

Tables 9. User-defined multi-dimensional arrays of macro-variables by an example of Laboratory Analysis and Parameters sheets

Analysis Number	Analysis Code	Analysis Name	Analysis Dataset
1	HEM	HEMATOLOGY	in_hem
2	CHEM	CHEMISTRY	in_chem
3	CO	COAGULATION	in_co

Analysis Number	Analysis Code	Parameter Number	Parameter Code	Parameter Name
				HEMATOLOGY
1	HEM	1	HCT	HEMATOCRIT
1	HEM	2	HGB	HEMOGLOBIN
1	HEM	3	MCV	ERY. MEAN CORPUSCULAR VOLUME
				CHEMISTRY
2	CHEM	12	ALB	ALBUMIN
2	CHEM	13	ALP	ALKALINE PHOSPHATASE
2	CHEM	14	ALT	ALANINE AMINOTRANSFERASE

To generate LB domain from several laboratory datasets we may construct a user-defined macro-loop iterating over all analysis types (Table 10). An argument of this macro-loop should include all instructions needed for data step execution. Moreover, it may also contain another nested macro-loop iterating over all parameters required for the current analysis type.

Tables 10. Nested user-defined macro-loops by an example of LB-domain calculation

Domain	Data Step	Code Surroundings
		/* Macro-Loop over All Laboratory Analysis Datasets */
		%MacroLoopAn_An (%nrstr (
		data Libr.&&SDTMName&d.._&An.;
		merge Libr.&&SDTMName&d..0
		Libr.&&AnDS&An..;
		keep %LoopSDTM_InVar_n (&d., %nrstr(&&SDTMVarName&d._&n..))
		LBSign; /* For SUPPLB Domain */
		/* First One-String File merge, no by-statement */
		/* Macro-Loop over All Laboratory Parameters within Current Analysis */
LB	1	%MacroLoopPar_k (&An., %nrstr (

Domain	Variable Name	Variable Label	Type	Code Substitutions
LB	LBTESTCD	Lab Test or Examination Short Name	Char	LBTESTCD = "&&ParCode&An._&k..";
LB	LBTEST	Lab Test or Examination Name	Char	LBTEST = "&&ParName&An._&k..";
LB	LBCAT	Category for Lab Test	Char	LBCAT = "&&AnName&An..";

Domain	Data Step	Code Surroundings
		output;
		))
		run;
		))
		/* Combining Prepared Laboratory Domains for All Analysis Together */
		data Libr.&&SDTMName&d..1;
		set %MacroLoopAn_An (%nrstr (
		Libr.&&SDTMName&d.._&An.)) ;
		proc sort; by &InIDVar. LBCAT LBTESTCD LBDTC;
LB	1	run;

So, the data step header in this representation may be quite complex. At the same time variable deriving instructions in many cases may be written as one-string assignment of newly creating variables by macro-variables stored in specification and indexed by analysis or parameter sequential numbers (Table 10).

If we wish to produce many rows in the output domain for each row in the initial dataset, we should specify *output* statement before the end of inner laboratory parameters macro-loop (Table 10).

At the final step we should combine preliminary generated datasets for each analysis into single LB domain. This may be done with the help of the same macro-loop over laboratory analysis types (Table 10). In the contrary to possible unlimited complexity of the macro-loops argument the definition of macro-loops itself turns out to be quite plain. This may be performed by means of SAS *%do*-loops. The only special requirement here remains wrapping of the macro-argument with *%nrstr* () function for macro-indexes masking while its call, and subsequent unwrapping it with *%unquote* () function in the body of *%do*-construction (Table 11).

Table 11. User definition of macro-loops over Laboratory Analysis and Parameters in the Macro sheet

Macro Functions

```
                                /** LB Domain **/
                          /* Macro-Loop over All Analysis */
%macro MacroLoopAn_An (MaskStat);                    /* Statement Masked by %nrstr() */
 %do An = 1 %to &NAn.;          /* Forming Through Laboratory Analysis Specification */
  %unquote (&MaskStat.)
 %end;
%mend;
                /* Macro-Loop over Parameters for Specified Analysis */
%macro MacroLoopPar_k (An, MaskStat);     /* Analysis, Statement Masked by %nrstr() */
 %do k = 1 %to &&NPar&An..;  /* Forming Through Laboratory Parameters Specification */
  %unquote (&MaskStat.)
 %end;
%mend;
```

As we already pointed out, analysis and parameters indexes as well as their codes and names required for macro-loops construction and execution originate directly from respective sheets in the extended Excel specification. And total number of analysis as well as numbers of parameters for the current analysis may be easily calculated as counts of rows satisfying appropriate filtering conditions.

DISCUSSION

As we already mentioned above, FDA recommendations from 2016 on SDTM demand for trial report submissions has encouraged a lot of attempts on automation of SDTM generation process. Some of them also consider Excel specification as a core instrument for subsequent automatization [4, 5].

However, in published materials one can mostly find general ideas and several pointers to designed macro-procedures. In rare works which uncover underlying SAS code [5], we have seen instructions for generating the simplest examples of SDTM domains such as DM or AE. At the same time, particular mechanisms of production for such domains as SUPP__ or LB remain not evident. As a consequence, we could not find any suggestions on classification of all variety of SDTM generating instructions. That is what we aim to do in our work.

Now let us consider another question: for how far can we extend the approach proposed onto the other processes of statistical production?

First of all, it is definitely applicable for construction of some among ADaM datasets. The general flow remains the same: we add SAS-instructions into the specification in order to define which and how variables should be created and after that use them for automatic generation of designed domains. This can be easily perform for such domains as ADSL or ADAE which usually do not involve any complicated derivations. On the other hand, it may not always be useful to place some more sophisticated derivations which may be in need to create, for example, time-to-event (TTE) domains directly in an Excel specification. Those parts of deliverables can be handily programmed and debugged directly in original SAS software on the course of manual tunning stage.

Moreover, to some extent this approach can be generalized also for TFL production (Fig. 3). Though that process is obviously much more complex and efficacy outputs in most cases have to be manually programed, creating such tables as for demography or laboratory statistics can be formalized in quite an easy way. This can be done via creating of macro libraries with computational and reporting macro-procedures each aiming to produce one of predefined table

types, and then combining these units in a composite program for generation of all desired outputs. We treat this program as a roadmap for further development.

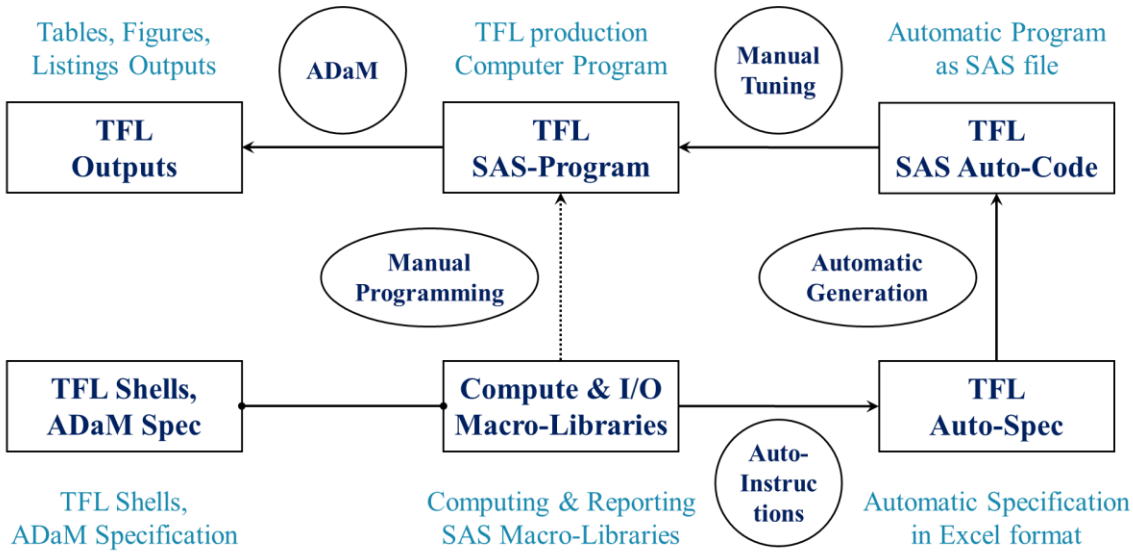

Figure 3. Further application of the proposed approach to TFL generation

CONCLUSIONS

Summing up, we have described the general framework of clinical study life circle and proposed one of the possible ways to automatic production of SDTM database. We have defined three types of SAS variable deriving instructions which in our approach are embedded directly into SDTM specification. Along with that we have discussed how to apply the proposed method for ADaM and TFL production.

REFERENCES

1. Roman Radelicki, Swapna Pothula. End to End SDTM Automation: A Metadata Centric Approach. PhUSE US Connect 2019. Paper SI01 https://www.lexjansen.com/phuse-us/2019/si/SI01.pdf
2. Pieter Blomme. From Source Data to SDTM: Enabling a Semi-automated Conversion. PhUSE US Connect 2019. Paper DH03 https://www.lexjansen.com/phuse-us/2019/dh/DH03.pdf
3. Akhil Vijayan, Anish George, Anoop Ambika. Automating SDTM – Get it prepared with your voice. GenPro Research https://genproresearch.com/publication-2/automating-sdtm-get-it-prepared-with-your-voice/
4. Hemalatha Elumalai. Streamline process: To Generate SDTM Program by Automation. PhUSE US Connect 2019. Paper PP02 https://www.lexjansen.com/phuse-us/2019/pp/PP02.pdf
5. Frederick Cieri, Rama Arja, Ramesh Karuppusamy. Programmatically mapping source variables to output SDTM (Study Data Tabulation Model) variables based upon entries in a standard specifications Excel® file workbook. PharmaSUG 2019. Paper BP-340 https://www.lexjansen.com/pharmasug/2019/BP/PharmaSUG-2019-BP-340.pdf

RECOMMENDED READING

1. Study Data Tabulation Model: Human Clinical Trials. Version 1.7 (2018-11-20)
2. Study Data Tabulation Model Implementation Guide: Human Clinical Trials. Version 3.3 (2018-02-20)

CONTACT INFORMATION

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

Author Name: Ilya Mishchenko
Company: Atlant Clinical Ltd.
Address: 12/1 Krivokolennyi Lane
City/Postcode: 101000, Moscow, Russia
Work Phone: +7 (495) 628 38 02
Email: ilya.mishchenko@atlantclinical.com
Web: www.atlantclinical.com

Author Name: Aleksandra Gorbunova
Company: Atlant Clinical Ltd.
Address: 12/1 Krivokolennyi Lane
City/Postcode: 101000, Moscow, Russia
Work Phone: +7 (495) 628 38 02
Email: aleksandra.gorbunova@atlantclinical.com
Web: www.atlantclinical.com

Author Name: Evgeny Dmitriev
Company: Atlant Clinical Ltd.
Address: 12/1 Krivokolennyi Lane
City/Postcode: 101000, Moscow, Russia
Work Phone: +7 (495) 628 38 02
Email: evgeny.dmitriev@atlant-clinical.com
Web: www.atlantclinical.com

Author Name: Timur Zagidullin
Company: Atlant Clinical Ltd.
Address: 12/1 Krivokolennyi Lane
City/Postcode: 101000, Moscow, Russia
Work Phone: +7 (495) 628 38 02
Email: timur.zagidullin@atlantclinical.com
Web: www.atlantclinical.com