PharmaSUG 2014 - Paper DS19

SAS® as a Tool to Manage Growing SDTM+ Repository for Medical Device Studies

Julia Yang, Medtronic Inc. Mounds View, MN

ABSTRACT

When we have a substantial number of medical device studies in many different therapeutic areas, it is desirable to have a common data repository to facilitate clinical data management, analysis and reports. Modified Study Data Tabulation Model plus (SDTM+) becomes the infrastructure for our clinical data. SDTM+, with some adaptations, followed the SDTM and the SDTM for Medical Devices (SDTM-MD) by Clinical Data Interchange Standards Consortium (CDISC).

There are many challenges associated with mapping multiple studies in many different therapeutic areas into one set of SDTM+ domains. These challenges include: 1) ensuring consistency across all studies; 2) incorporating new medical devices added periodically, quite possibly indefinitely; 3) making the SDTM+ database both scalable and stable; and 4) making sure the database is “self-explanatory” and source-traceable so new users do not need to refer to multiple documents.

This paper summarizes what we learned so far. It discusses SAS macros developed to help map data to the SDTM+ , monitor SDTM+ consistency and check SDTM+ data integrity.

Key Words: SAS, SAS Macro, CDISC, SDTM, SDTM-MD, Medical Device Data

INTRODUCTION

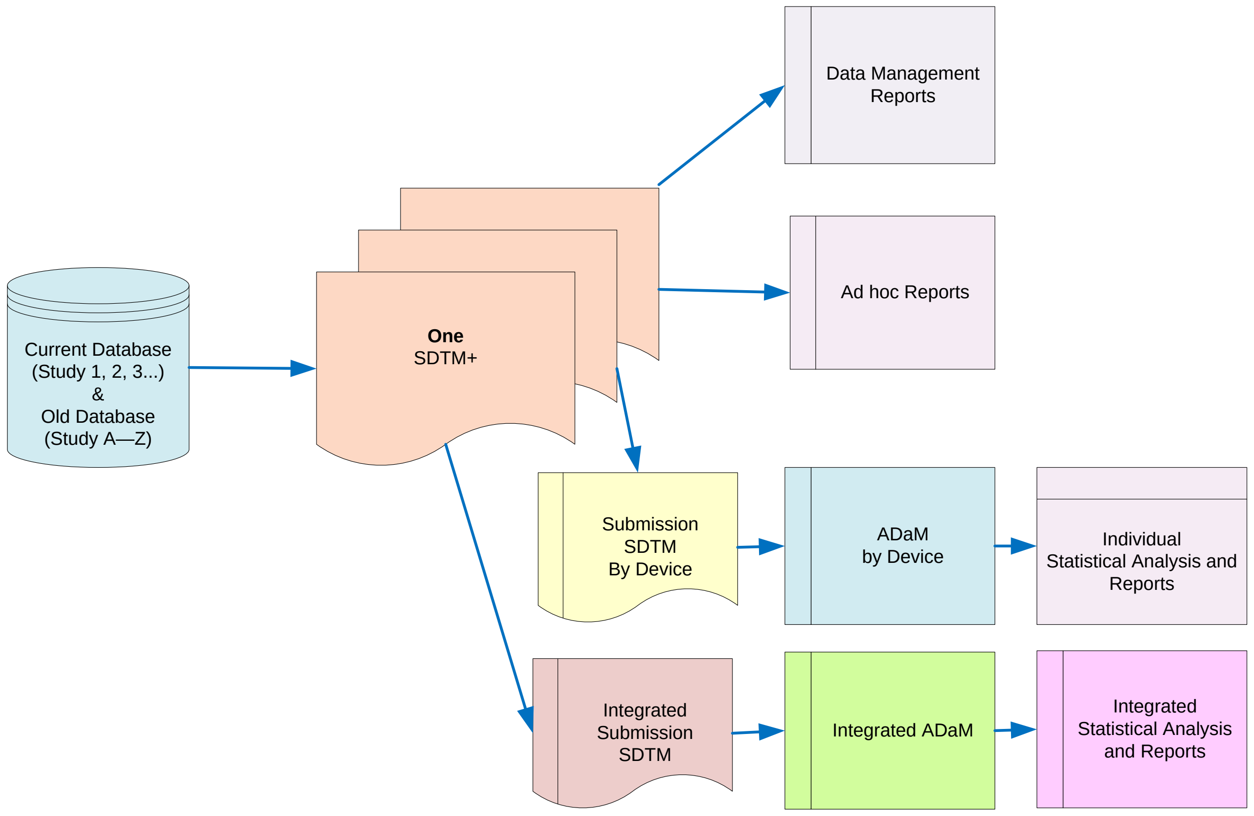

Data Management Reports
Ad hoc Reports
Current Database (Study 1, 2, 3...) & Old Database (Study A—Z)
One SDTM+
Submission SDTM By Device
ADaM by Device
Individual Statistical Analysis and Reports
Integrated Submission SDTM
Integrated ADaM
Integrated Statistical Analysis and Reports

Figure 1 Data Transformations from Many Clinical Data Sources

When there are many medical devices involved in clinical studies, we can use SDTM (Study Data Tabulation Model) standard from the CDISC (Clinical Data Interchange Standards Consortium) as the standard for clinical data repository and as a starting point for future data management reports, ad hoc reports, individual and integrated statistical analysis and reports.

As shown in Figure 1, we mapped all medical device clinical study electronic case report forms (eCRFs) data into one SDTM+ database. The "+" symbol indicates a modification of SDTM to effectively manage the data and ultimately facilitate regulatory submissions. This SDTM+ has become the data repository for the clinical source data and not merely the data tabulations for regulatory submission. It allows controlled flexibility for all raw data fields and data management questions. The plan is to create one "uniform" SDTM+ database. Doing so will ensure that the SDTM standards drive the standards and efficiencies in data collection, data management and data usage prior to any regulatory submissions. Importantly, SDTM+ will need to be straightforwardly transformed into submission ready CDISC SDTM when needed.

SAS has been used as a tool in managing consistency in data transformation from source to SDTM+. This paper includes the efficiency and effectiveness leveraged by SAS. Hopefully, it will stimulate a thoughtful discussion of SAS in the implementation of SDTM.

CONCEPT OF THE SDTM+ REPOSITORY

The challenges of mapping medical device data to SDTM are greater than we expected. This section will focus on our SDTM+ design to illustrate the use of SAS as a tool to manage the growing SDTM+ repository for medical device data.

1. Source data traceable, a "self-explanatory" SDTM+ database

It is desirable for any database user to have a database that is traceable to its source – i.e., a database that tells a "complete story" of its own. That is, by reviewing the database, users should be able to understand where and how the SDTM+ is created from eCRF.

We can facilitate this traceability by adding a set of variable(s) that hold the unique Source System Identifier(s) at each record of SDTM+ domains. Most of the SDTM+ data at a record level comes from one source system identifier. Sometimes there is more than one source for a SDTM record. In that situation, each SDTM+ variable is given its own source system identifier.

This Source System Identifier establishes a one-to-one relationship between a variable and its source in the database. However, this Identifier alone does not facilitate the quick identification of the particular source table/field for a variable in a SDTM domain when there are a plethora of tables and multiple source variables in each table. There is a SAS macro that serves as a lookup tool to show users which eCRF fields/form those source system identifiers indicating. Here is a simple call that lists the source tables and variables from which the DT (Device Tracking and Disposition) domain is built. If the SDTMVar is not defined, then the macro only lists the source table names. When the SDTMVar is identified, as shown in Table 1, the macro searches the Data Definition Table (DDT) as well as the Source table to generate the lookup table. The macro output provides a useful tool for data review and validation.

```
%MSourceSystemIDLookUp(SDTM=DT, SDTMVar=DTERM);
```

Unique Subject Identifier	SDTM Domain Name	SDTM Variable Name	Source Table Name	Source Variable Name	Source System Identifier
0001	DT	DTTERM	AdverseEvent	Var_A	100011
0002	DT	DTTERM	AdverseEvent	Var_A	100022
0002	DT	DTTERM	Exit	Var_Z	999991

Table 1 Source System Identifier lookup table for DT Domain from % MSourceSystemIDLookUp

2. Source data traceable with assistance of Data Definition Table

Not all traceability can be built into SDTM+ domains. Therefore, an additional way to track data transformation is to use a Data Definition Table (DDT) in Excel format. Part of the DDT is shown at Table 2 . Notice the structure of the DDT has "controlled format" with required columns and column names in Excel for all SDTM+ domains. Later we will discuss how we check "controlled format" and compare different versions of DDTs with SAS macros.

The detailed SDTM+ mapping instructions are in the Mapping Comments column. All studies must follow a mapping definition so that variables in SDTM+ bear the "same variable name, same meaning" when we put all studies into one SDTM+. When SDTM+ becomes one integrated database for all studies, data management report, integrated ADaM (Analysis Data Model) data sets and analysis reports can be built with minimum effort.

Domain Prefix	Variable Name	Variable Label	Length	Format	Mapping Comments
DT	DTTERM	Reported Term for the Tracking Event	200	$200.	1. AdverseEvent.Var_A: Logic 1 etc... 2. Exit.Var_Z: Logic 2 etc...

Table 2 Example of SDTM+ Traceability: DDT Mapping Comments Column

DDT in Excel format rather than Word format is preferable. DDT is a form of metadata and holds the data structure and variable attributes of a SDTM+ domain. The advantage of Excel format is that it is readily readable by SAS. We have SAS macros to translate the Excel sheet into SAS and build SDTM+ domains.

For example, the macro call %*MImportExcel*(sdtmdataset=DT) will import the entire DDT content for DT domain as a data set into SAS. The %*MImportExcel* macro uses ODBC driver to retrieve the Excel data. Part of the macro code is shown below:

```
libname excel odbc required="Driver={Microsoft Excel Driver (*.xls, *.xlsx,
*.xlsm, *.xlsb)};

DBQ=&inpath.\&ExcelFileName..&filetype.";
```

We have noticed that Mapping Comments imported from Excel were truncated at 255 or 1024 characters. The problem can be addressed as follows: First, the number of rows scanned in determining the length of a column can be changed. SAS uses the Microsoft Access Connectivity Engine to read Excel files. The engine uses a TypeGuessRows registry key to sense how many rows to scan before determining the length of the column. By default it uses 8 rows to do so, but this can be changed to 0, resulting in checking the first 16384 rows in Excel. Second, the default length for the TEXTSIZE parameter can be increased from the default 1024 by adding TEXTSIZE=32767. Finally, the Excel column should always be selected and formatted as Text instead of General.

After *%MImportExcel* creates the SAS data set from DDT, the macro call *%MCreateSDTMShell*(sdtmdataset=DT) transforms metadata in DT DDT into SDTM+ domain shell. The latter macro faithfully retrieves variable sequence and variable attributes. These macros provide substantial efficiencies, avoiding the time and potential errors associated with tediously typing the metadata. Part of the *%MCreateSDTMShell* is shown here:

```
data &sdtmdataset.Shell;
       %let ddtindx=1;
       /*using variable sequence & attributes as dictated in DDT*/
       %do %while (%qscan(%bquote(&namelist.),&ddtindx.,@@) ^= );
              Attrib %unquote(%qscan(%bquote(&namelist.),&ddtindx.,@@))
              %unquote(length=%Qscan(%bquote(&lenlist.),&ddtindx.,@@) )
              %unquote(format=%Qscan(%bquote(&formatlist.),&ddtindx.,@@))

       %unquote(informat=%Qscan(%bquote(&formatlist.),&ddtindx.,@@) )
              %unquote(label="%Qscan(%bquote(&labellist.),&ddtindx.,@@)");
         %let ddtindx = %sysevalf(&ddtindx.+1);
       %end;
       /*initiating character and numeric variables*/
       %if &nexist %then %do;
              array num(*) _numeric_ ;
       %end;
       %if &cexist %then %do;
              array chr(*) _character_ ;
       %end;
run;
```

The %*MImportExcel* and %*MCreateSDTMShell* macros combined enable DDT as one source to manage SDTM+ data structure. If DDT is updated, SDTM+ data structure and attributes are updated automatically.

Another handy utility macro is %*MdomainVarlist*(sdtmdataset=). This macro creates a list of variables for a specific SDTM+ domain. As a result, only variables defined in DDT for a particular SDTM+ domain are kept while all other intermediate variables created or used during data transformation steps are dropped.

During the SDTM+ life cycle, all DDTs are loaded to a version control system to keep track of changes by authorized personnel. DDT+ is signed off with an electronic signature before SDTM+ domains are validated and released to production mode.

3. Usability, another focus of SDTM+

All dates in SDTM+ have a set of 3 forms: 1) CDISC date in ISO 8601 formats (--DTC); 2) date in character format as collected with eCRF; and 3) the date in SAS numeric Date/Time format. CDISC date format is for submission tabulation. Character date as collected is a reference for many users, especially a field often checked by data management reports. Date in SAS numeric format is used for calculation and sorting purposes.

Here is a sample call of %*mdtr2dtc_dt* macro, which creates three forms of date variables for a character input date:

```
%mdtr2dtc_dt(studystartdata=source.studystartdate, ⓿
             studystartdate=x_rfstdt, ❶
             inputdata=aesnm, outputdata=aesnm2, ❷
             inputdtr=aestdat_snm_dtr, ❸
             outputdat=x_aestdat, ❹
             outputdt=x_aestdt, ❺
             outputdtc=aestdtc, ❻
             datebeforestudystart=);❼
```

⓿ Specify the intermediate data set holding Subject Reference Start Date in numeric format

❶ Specify Subject Reference Start Date in numeric format

❷ Obviously the input and output data set names of the macro

❸ Specify input character date variable in source data set, later is renamed to variable provided at ❹

❹ Provides output date variable name in character format as collected in source data set

❺ Provides output date variable name in SAS numeric Date/Time format

❻ Provides output date variable name in CDISC ISO 8601 format

❼ When collected date is partial, this step will decide if the macro will impute SAS numeric date according to defined imputation guidelines. Depending on whether the imputed date is before or after Subject Reference Start Date, this option determines whether a calculated date is reasonable and should be accepted, or logically wrong and should be set to missing. For example, if we define adverse events as those occurring after Subject Reference Start Date, then any imputed event date before Subject Reference Start Date is reset to a special missing value, .C, meaning the imputed date conflicts with the consent date.

Another important function of %*MDTR2DTC_DT* is to handle partial dates according to CDISC defined rules. The resulting –DTC will take expected ISO 8601 Date/Time precision.

4. Common identifier variables

CDISC/SDTM has a set of common identifiers, such as Study Identifier, Domain Abbreviation and Unique Subject Identifier. Our SDTM+ has additional common Identifiers used internally. Those common identifiers are included in every domain in every study. A utility SAS macro *%mcommonvars* will import logic definition to create those common identifiers consistently in all domains and for all studies.

Call example to create a set of common variables in DX (Device Exposure) domain:

```
data DX;
set dxshell
    source.DeviceTRTData(in=a );
    if a;
```

```
        %mcommonvars(domain=ae);
    run;
```

5. The SDTM+ in summary

- One SDTM+ database hosts all studies. It should be a scalable yet stable data repository
- All expected and permissible variables will be included in the SDTM+ model to minimize physical variations across studies
- Additional internal variables added to allow storage of source data fields and/or convenience of frequent data usage
- At submission of SDTM:
    - Additional internal variables will be removed
    - Permissible variables that do not have values for any record will be dropped

MONITERING SOURCE DATABASE UPDATES

As what we call first dimensional growth, the SDTM+ database can host as many medical device studies as needed. What we refer to as second dimensional growth is the ability of SDTM+ to facilitate updates to a source data base consistent with study protocol updates.

To reduce the risk of accidentally altering a validated SAS program for a particular SDTM+ domain in a study, SDTM+ domains are developed by domain and by study – i.e. one DDT for one SAS program to create one SDTM+ domain in a study. This way, when study 1 is updated, only a few programs for study 1 should require revisions. Other programs and other studies will remain untouched.

Finally, a master program will assemble domains for all studies into one SDTM+ database. It is also worth mentioning that DDTs, SAS programs/macros and SDTM+ domains are version controlled to secure programming and SDTM+ database integrity.

The %MCompareOldNewMetaData macro compares previous source database with the new source database on a list of attributes that are critical to clinical data analysts, such as View Name, Variable Name, Variable Type (string, number or code value), Variable Max Length, eCRF form name, Full Question Text, Question number, SAS Label, Type (Pull down, Text , Radio ,Check box or Date). The macro outputs will help identify whether there is a change in any of these attributes that will require actions in SDTM+ development or revisions.

The core of this macro is the COMPARE PROCEDURE. The following is a fragment of the code:

```
proc compare data=&oldlib.ProcContentsByStudy

                   compare=&oldlib.ProcContentsByStudy

                   out=DiffByTableVar ⓿

                   outnoequal❶ outbase❷ outcomp❸ outdif❹ noprint❺;

        by TableName Varname; ❻

run;
```

In the program above,

⓿ OUT= specifies the name of the output data set.

❶ OUTNOEQUAL specifies only the unequal observations are included in output data set.

❷ OUTBASE provides the observations from the base data set

❸ OUTCOMP provides the observations from the comparison data set

❹ OUTDIF gives the difference between the two observations

An X shows that the characters do not match. A period "." indicates that the characters do match.

The difference is shown, if numeric variables do not match. E indicates that the numeric variables do match.

❺ NOPRINT suppresses the printing of the PROC COMPARE output

❻ and ❼ Give user options to compare by view/table name and variable name or by Item Reference ID that is a relative stable across database upgrades.

This macro compares database in year 2012 and after database update in year 2013.

```
%MCompareOldNewMetaData(oldlib=A_Y12, newlib=A_Y13);
```

The comparison results give the user a highlight on changes on database upgrade. The macro call creates the following data sets:

- Views/Variables only in new database (data set Only A_Y13)
- Views/Variables only in previous database (data set Only A_Y12)
- Views/Variables in both new and previous database but have differences in attributes (data set Diff)

ENSURING SDTM+ CONSISTENCY ACROSS ALL STUDIES AND CHRONOLOGICALLY

1. Controlled and harmonized DDTs used for SDTM+ development

Figure 2 illustrates the proposed controlled process to add new studies or to update existing studies in the SDTM+ database. It should be understood that new data fields are mapped using the same definitions as those previously mapped in existing DDTs.

When new fields from eCRF cannot be mapped to existing SDTM+ domains and require more variables in existing SDTM+ domains, or an entirely new SDTM+ domain, the SAS team should seek assistance and approval from the SDTM+ standard team.

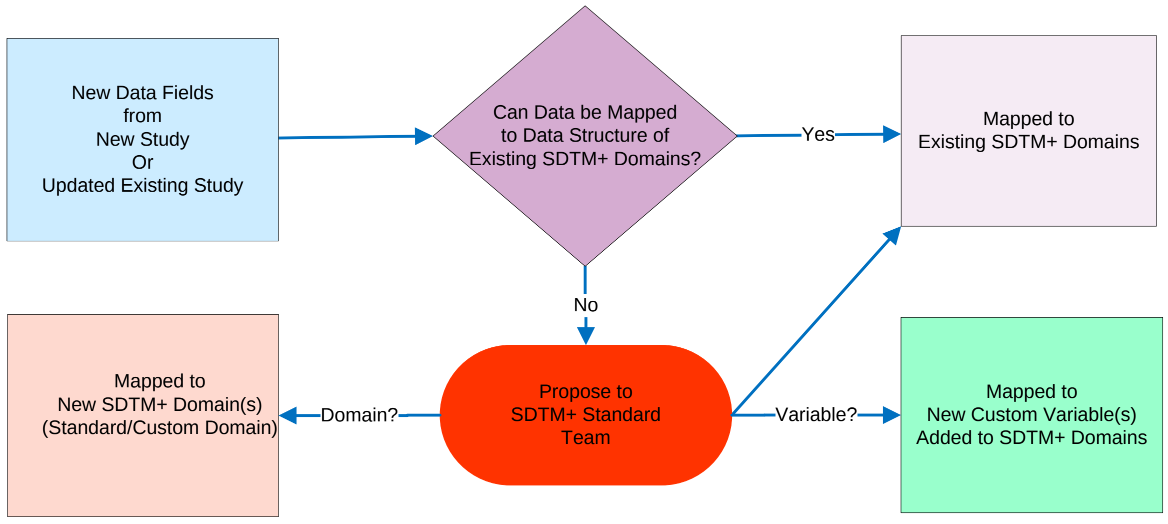
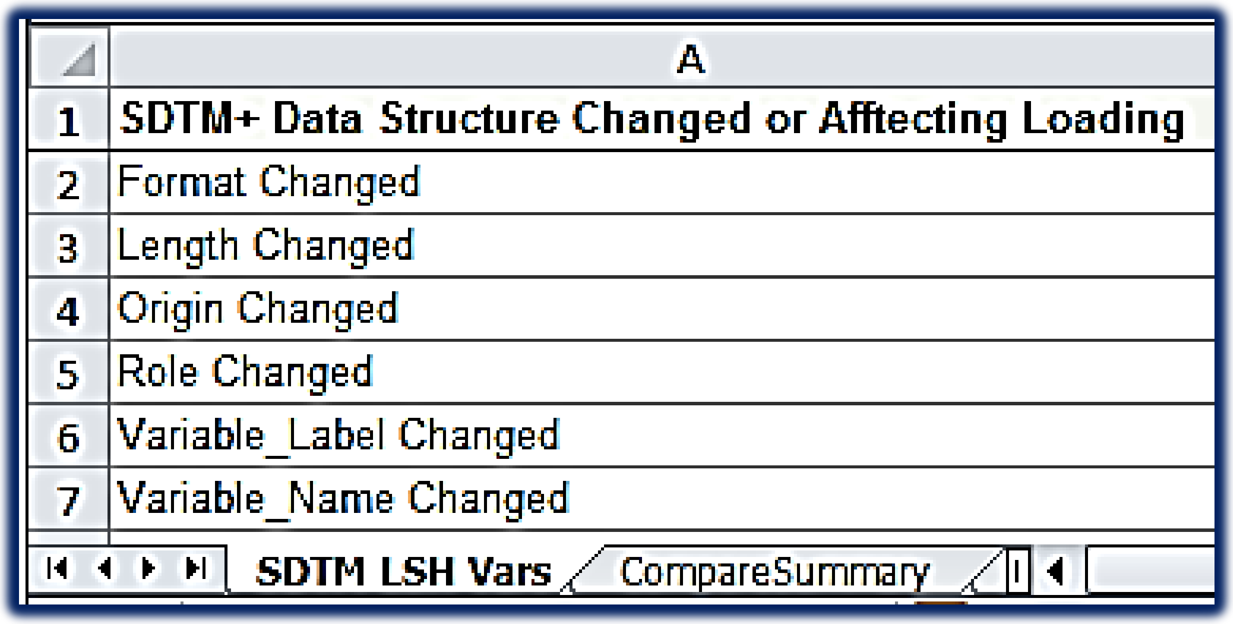

New Data Fields from New Study Or Updated Existing Study
Can Data be Mapped to Data Structure of Existing SDTM+ Domains?
Yes
Mapped to Existing SDTM+ Domains
No
Propose to SDTM+ Standard Team
Domain?
Mapped to New SDTM+ Domain(s) (Standard/Custom Domain)
Variable?
Mapped to New Custom Variable(s) Added to SDTM+ Domains

Figure 2 Controlled Process in Updating Study DDTs

2. Checking study DDTs consistency against base SDTM+ models

13 columns in DDT are required for loading data definitions in the production hosting system. If any of the attributes changed in DDT, an analysis should be performed to confirm that the changes will not affect the data integrity of studies already in SDTM+ production mode. If the changes do not affect existing studies, the data definitions are reloaded so new and updated study SDTM+ domains are created successfully in the host system.

The *%MCompareDDTs* macro compares new SDTM+ DDTs with existing base DDTs. Figure 3 is the 1st Excel tab in macro output and shows a few of the data attributes changed. Hopefully, the changes have been reviewed upstream from this step. It is assumed that the changes noted here are intended and approved by SDTM+ standard team. If that is the case, those changes require a data definition reload in SDTM+ hosting system.



	A
1	**SDTM+ Data Structure Changed or Afftecting Loading**
2	Format Changed
3	Length Changed
4	Origin Changed
5	Role Changed
6	Variable_Label Changed
7	Variable_Name Changed

Figure 3 Sample Output from *%MCompareDDTs* Macro: 1st Tab – Variables Attributes

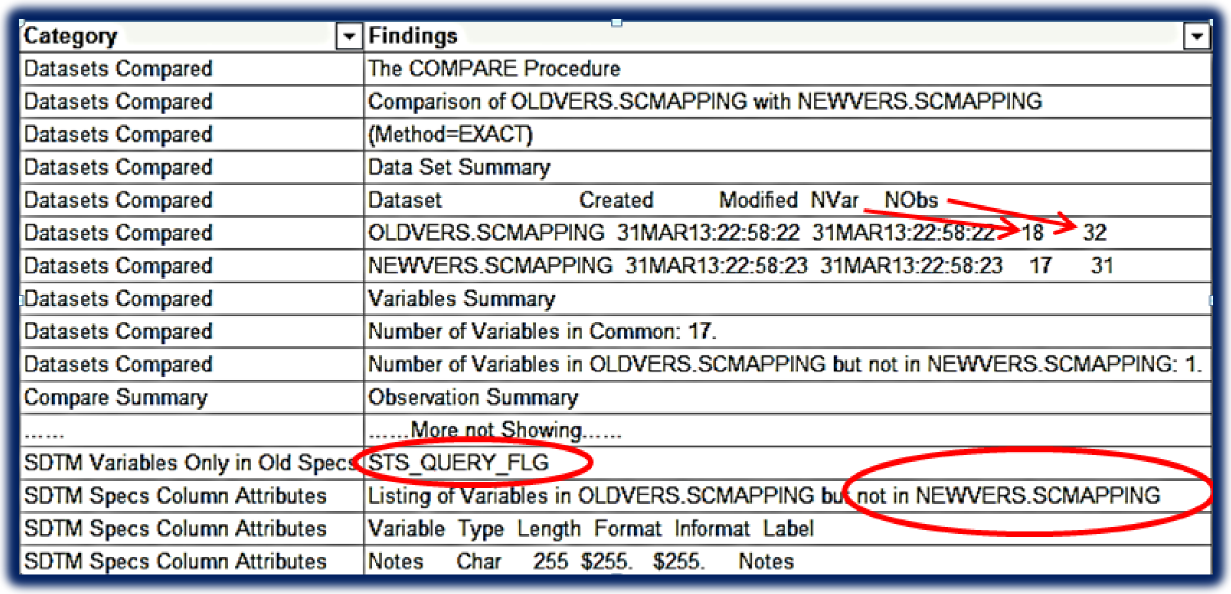
The 2nd tab showing in Figure 4 is the summary output of PROC COMPARE, testing to determine whether the DDT has more or fewer columns or rows as compared with the previous version.

```
    ods listing close;
    ods output
        comparedatasets     =     specscompare ⓿
        comparevariables    =     specscolsdiff ❶
        comparesummary      =     specscomparesummary; ❷
        proc compare data   =     oldvers.&currspecscmprs.mapping
            compare         =     newvers.&currspecscmprs.mapping
            out              =    comparediff&currspecscmprs. ❸
             outnoequal outbase outcomp;
        run;
    ods listing;
```

⓿, ❶ and ❷ creates 2nd tab of Excel output (Figure 4)

❸ OUTCOMP provides PROC COMPARE difference Listings in 3rd tab (Figure 5)

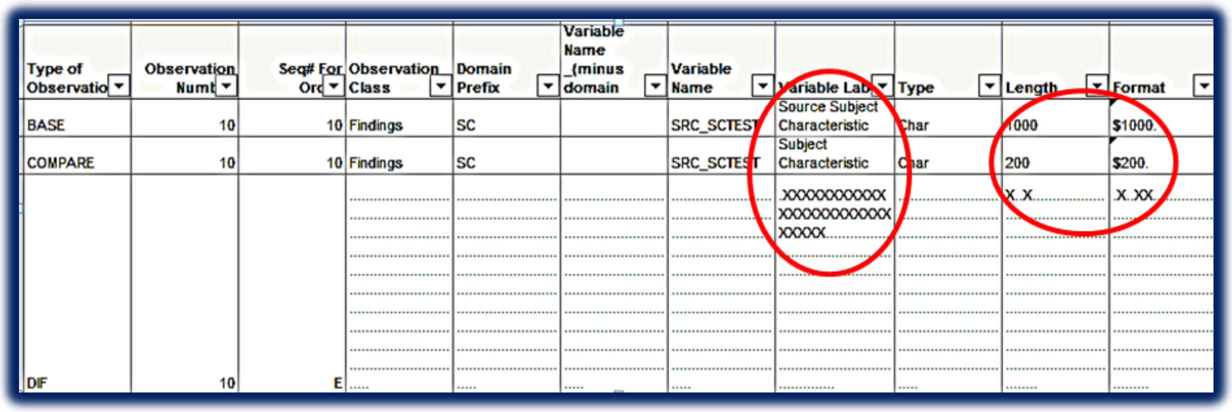
Figure 4 shows there are 18 columns in old version DDT and 17 columns in new version DDT. Indeed, a column 'Notes' is missing in the new version DDT. It is a controlled column name to capture domain level notes for SDTM+ mapping. However, the column is not required and does not need DDT correction. Old version DDT has an extra SDTM+ variable named 'STS_QUERY_FLG'. 'STS_QUERY_FLG' is one of the common identifiers and must be presented in all SDTM+ domains. 'STS_QUERY_FLG' finding requires DDT correction.



Category	Findings
Datasets Compared	The COMPARE Procedure
Datasets Compared	Comparison of OLDVERS.SCMAPPING with NEWVERS.SCMAPPING
Datasets Compared	(Method=EXACT)
Datasets Compared	Data Set Summary
Datasets Compared	Dataset Created Modified NVar NObs
Datasets Compared	OLDVERS.SCMAPPING 31MAR13:22:58:22 31MAR13:22:58:22 18 32
Datasets Compared	NEWVERS.SCMAPPING 31MAR13:22:58:23 31MAR13:22:58:23 17 31
Datasets Compared	Variables Summary
Datasets Compared	Number of Variables in Common: 17.
Datasets Compared	Number of Variables in OLDVERS.SCMAPPING but not in NEWVERS.SCMAPPING: 1.
Compare Summary	Observation Summary
......	......More not Showing......
SDTM Variables Only in Old Specs	STS_QUERY_FLG
SDTM Specs Column Attributes	Listing of Variables in OLDVERS.SCMAPPING but not in NEWVERS.SCMAPPING
SDTM Specs Column Attributes	Variable Type Length Format Informat Label
SDTM Specs Column Attributes	Notes Char 255 $255. $255. Notes

Figure 4 Sample Output from *%MCompareDDTs* Macro: 2nd Tab – Summary from PROC COMPARE

The example in Figure 5 shows Variable Label, Length and Format are different. Option OUTNOEQUAL is defined in macro, so only DDT rows having different values from old and new versions are listed here. As indicated by ‘X’. The finding is in agreement with 1st tab in that Variable Label, Length and Format are changed in new version as compared with old version. As mentioned, this requires data definition reload at SDTM+ host system.

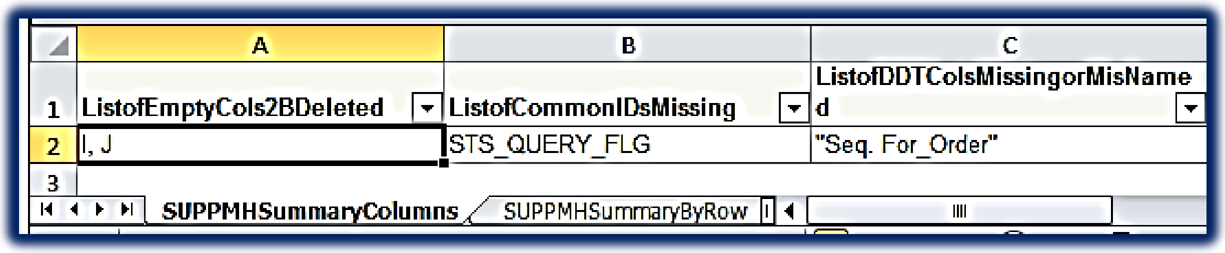


Type of Observatio	Observation Numb	Seq# For Ord	Observation Class	Domain Prefix	Variable Name _(minus domain	Variable Name	Variable Lab	Type	Length	Format
BASE	10	10	Findings	SC		SRC_SCTEST	Source Subject Characteristic	Char	1000	$1000.
COMPARE	10	10	Findings	SC		SRC_SCTEST	Subject Characteristic	Char	200	$200.
							.XXXXXXXXXXX XXXXXXXXXXXX XXXXX.		X.X	X.XX
DIF	10	E	.....	.....	.....	.....	....	.....	....	....

Figure 5 Sample Output from *%MCompareDDTs* Macro: 3rd Tab – PROC COMPARE Difference Listings

3. Checking DDT against the SDTM+ DDT guidelines

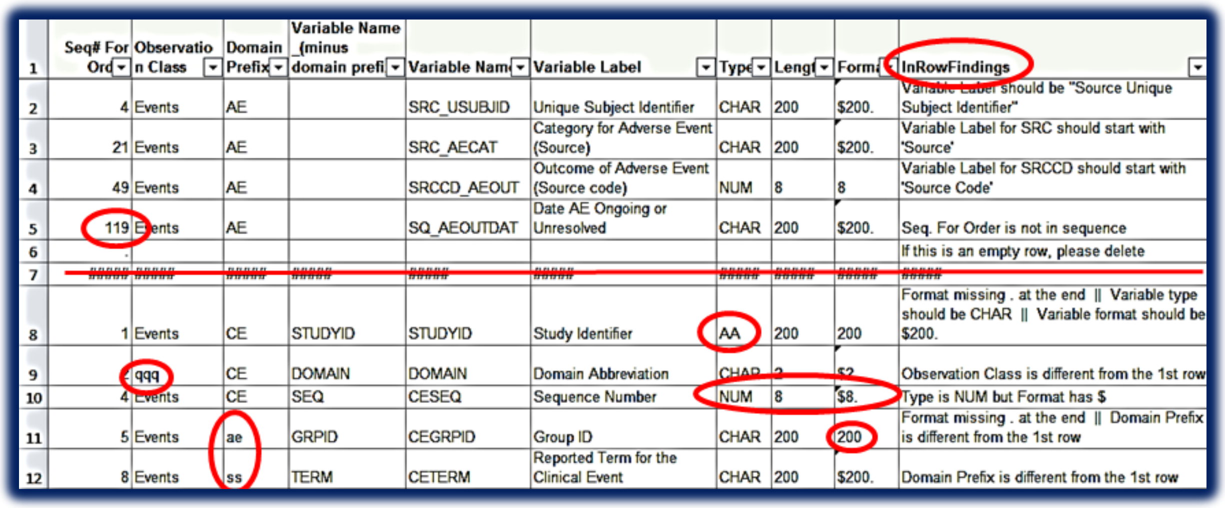
The *%MCheckDDT* macro checks DDT columns and rows according to the agreed upon Standard Guidelines. Figure 6 shows an example where there are extra empty columns I and J that should be deleted. SDTM+ common identifier STS_QUERY_FLG is missing. Also, one of the required columns Seq. for Order is missing in DDT.



	A	B	C
1	ListofEmptyCols2BDeleted	ListofCommonIDsMissing	ListofDDTColsMissingorMisNamed
2	I, J	STS_QUERY_FLG	"Seq. For_Order"
3			

Figure 6 Sample Output from *%MCheckDDT* Macro: 1st tab DDT column findings

Figure 7 demonstrates findings in two DDTs, one for the AE domain and another for the CE domain. The macro outputs identify departures from the guidelines, such as variable type, length and format typos, common identifiers missing and variables labeled differently than expected. The findings shown here are just for the purpose of testing macro functionality. The 'bad' cases like these have not been identified in actual practice.



1	Seq# For Ord	Observation Class	Domain Prefix	Variable Name _(minus domain prefi	Variable Nam	Variable Label	Type	Lengt	Form	InRowFindings
2	4	Events	AE		SRC_USUBJID	Unique Subject Identifier	CHAR	200	$200.	Variable Label should be "Source Unique Subject Identifier"
3	21	Events	AE		SRC_AECAT	Category for Adverse Event (Source)	CHAR	200	$200.	Variable Label for SRC should start with 'Source'
4	49	Events	AE		SRCCD_AEOUT	Outcome of Adverse Event (Source code)	NUM	8	8	Variable Label for SRCCD should start with 'Source Code'
5	119	Events	AE		SQ_AEOUTDAT	Date AE Ongoing or Unresolved	CHAR	200	$200.	Seq. For Order is not in sequence
6	.									If this is an empty row, please delete
7	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~	~~#####~~
8	1	Events	CE	STUDYID	STUDYID	Study Identifier	AA	200	200	Format missing . at the end || Variable type should be CHAR || Variable format should be $200.
9	2	qqq	CE	DOMAIN	DOMAIN	Domain Abbreviation	CHAR	2	$2.	Observation Class is different from the 1st row
10	4	Events	CE	SEQ	CESEQ	Sequence Number	NUM	8	$8.	Type is NUM but Format has $
11	5	Events	ae	GRPID	CEGRPID	Group ID	CHAR	200	200	Format missing . at the end || Domain Prefix is different from the 1st row
12	8	Events	ss	TERM	CETERM	Reported Term for the Clinical Event	CHAR	200	$200.	Domain Prefix is different from the 1st row

Figure 7 Sample Output from *%MCheckDDT* Macro: 2nd tab DDT Row findings

VALIDATE ALL SOURCE DATA FIELDS ARE MAPPED TO SDTM+

When all mapping specification is done at DTT, We want to know if all source fields have been mapped into SDTM+. The *%McheckvarsinSDTMspecs* macro is created to accomplish that task. The macro checks all current SDTM+ DDT and lists all source and SDTM+ target table/variable names and their relationship (Table 3). Source data that has not been mapped is highlighted in yellow and marked in the last column "Mapped to SDTM" as "No".

This output is also a quick reference tool for downstream SDTM+ consumers, to locate where the source data has been mapped to in SDTM+. For example, Variable DXTRT (Name of Device Exposure or Output) at domain DX is originated from source field DX1 at table DeviceExposure.

Source Table	Source Variable	SDTM Domain	SDTM Variable	Mapped to SDTM
DeviceExposure	DX1	DX	DXTRT	Yes
DeviceExposure	DX2	DX	DXSTDTC	Yes
DeviceExposure	DX3	DX	DXLAT	Yes
DeviceExposure	DX4			No
……	……	……	……	……
Device Events	DE1	DE	DESPID	Yes
Device Events	DE3	DE	DECAT	Yes
Device Events	DE4			No

Table 3 Source Tables/Variables and the relationship to Target SDTM Domains/Variables

MONITORING SDTM+ DOMAIN INTEGRITY WHEN SOURCE DATA REFRESHED DAILY

All SDTM+ domains are programmed independently by production and by validation programmers, following the same DDT mapping. Domain data sets from production and validation programming are compared using *%mcompareprodvaldata* macro.

We expect an exact match of PROC COMPARE outputs – i.e., no finding of discrepancy between production and validation SDTM+ domains. This will ensure that future program updates due to database updates will be built on 'clean' programs without any residual issue.

Here is part of the *%mcompareprodvaldata* macro:

```
%let anydiff=%MNumObs(CompareDiff); ⓿
proc sql noprint;
select distinct domain10
       into: faileddomains ❶
       separated by ', '
from CompareDiff;
quit;
%let faileddomains=&faileddomains.;

%local numdate;
%let numdate = %sysfunc(putn("&sysdate"d,YYMMDD10.)); ❷
%if &anydiff.=.  %then %do; ❸
       proc sql;
       insert into &DiffOutLib..CompareSDTMrunlog
              values("&numdate","&systime." "Passed"," ")
         ;
       quit;
%end;
%else %if &anydiff. >0 %then %do; ❹
       proc sql;
       insert into &DiffOutLib..CompareSDTMrunlog
              values("&numdate", "&systime.", "Failed","&faileddomains.")
         ;
       quit;

       data _null_;
           abort return; ❺
       run;
%end;
```

⓿ Count number of differences found between production and validation SDTM+ domains

❶ Get a list of failed domain names, if there is any

❷ Retrieve today's date

❸ If there is no difference noticed between production and validation SDTM+ domains, post Date/Time and mark "Passed"

❹ If there is difference noticed between production and validation SDTM+ domains, post Date/Time and mark "Failed" and list all failed domains

❺ Trigger host system to send a message to user, when the difference was found

The following macro call checks a list of SDTM+ domains &listofallsdtm in LIBNAME phase3p for production and LIBNAME phase3v for validation. Comparison is done by sorting with unique keys saved in dataset at LIBNAME target.

%mcompareprodvaldata (prodlib=phase3p, vallib=phase3v,

hashtextlib=target,diffoutlib=target,

indatalist= &listofallsdtm.);

The *%mcompareprodvaldata* macro is scheduled to run daily after the source database is refreshed and SDTM+ domains are updated. A daily running log is saved as a SAS data set, either indicating that all domains have passed validation or listing the domains that have failed. Table 4 shows that, on March 13th, the comparison failed with a few SDTM+ domains (DE, DI, DR and DU) not matching between production and validation. This should trigger the system to send a message to the user. After some programming fixes, the log shows the comparison passed in the following three days.

CompareDate	CompareTime	PassFail	DiffDomains
3/13/2013	17:37	Failed	DE, DI, DR, DU
3/14/2013	17:37	Passed	
3/15/2013	17:30	Passed	
3/16/2013	17:56	Passed	

Table 4 *%MCompareProdValData* Output: Comparesdtmrunlog Data Set

CONCLUSION

One SDTM+ database can be maintained consistently across all studies and throughout planned source database updates. As illustrated in Figure 8, during the development lifecycle, we monitor three checkpoints using SAS macros: source database updates, DDT updates for SDTM+ domains and SDTM+ domain updates. DDT utility macros discussed in this paper also help ensure the faithful use of metadata in DDT.

The 3rd checkpoint on SDTM+ domain updates is also monitored when SDTM+ is in production lifecycle. The daily source data refresh triggers SDTM+ domain updates, which then starts the SAS macro checking and comparing on production and validation domains. SDTM+ domains remain in validated status, unless the SAS macro check failed and notified the user.

With assistance from SAS macros, managing the SDTM+ repository for medical device data becomes more efficient and accurate.

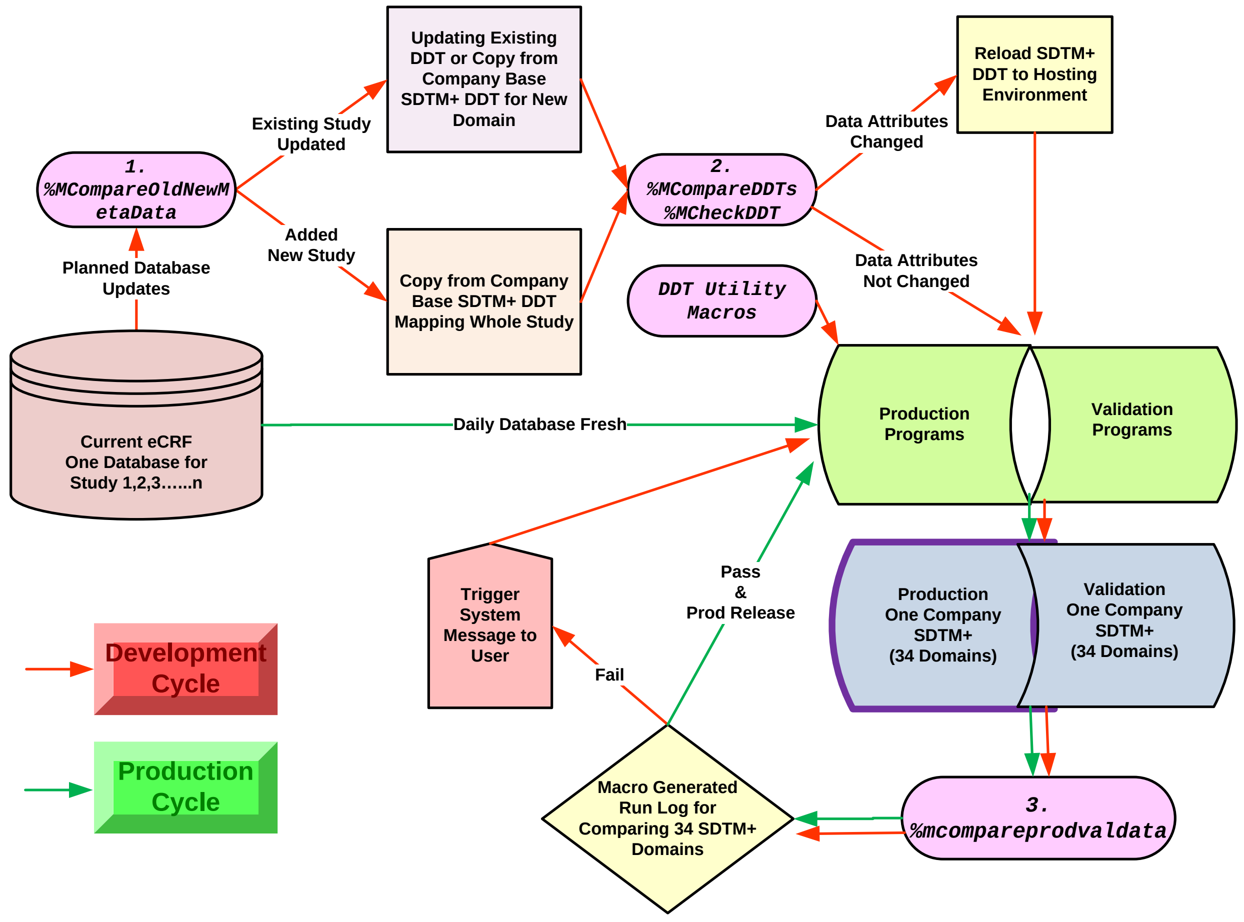

Updating Existing DDT or Copy from Company Base SDTM+ DDT for New Domain
Reload SDTM+ DDT to Hosting Environment
Existing Study Updated
Data Attributes Changed
1. %MCompareOldNewMetaData
2. %MCompareDDTs %MCheckDDT
Added New Study
Data Attributes Not Changed
Planned Database Updates
Copy from Company Base SDTM+ DDT Mapping Whole Study
DDT Utility Macros
Current eCRF One Database for Study 1,2,3……n
Daily Database Fresh
Production Programs
Validation Programs
Production One Company SDTM+ (34 Domains)
Validation One Company SDTM+ (34 Domains)
Trigger System Message to User
Pass & Prod Release
Fail
Development Cycle
Production Cycle
Macro Generated Run Log for Comparing 34 SDTM+ Domains
3. %mcompareprodvaldata

Figure 8 SAS Macros Used in Managing Live SDTM+ Data Repository

References

1. SAS Institute Inc. 2009. The COMPARE Procedure. *Base SAS® 9.2 Procedures Guide*. Cary, NC: SAS Institute Inc.
2. CDISC Study Data Tabulation Model (SDTM) v1.3 and Study Data Tabulation Model Implementation Guide (SDTMIG) v3.1.3 Available at http://www.cdisc.org/sdtm
3. Study Data Tabulation Model Implementation Guide for Medical Devices (SDTMIG-MD) v.1.0 Available at http://www.cdisc.org/stuff/contentmgr/files/0/3fa5f30f40ce5ecc7b3f91e558b55f73/misc/stdmig_md_v_1_0.pdf
4. Character strings can be truncated at 255 or 1024 characters when importing Excel files into SAS® http://support.sas.com/kb/46/472.html

DISCLAIMER

The views expressed in this paper are those of the author and do not necessarily represent those of the company the author affiliated to.

CONTACT INFORMATION

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

Name: Julia Yang

E-mail: julia.yang@medtronic.com