Paper PO37

Program Flow Design for Efficient CRT Programming

Sam Mao, Quintiles, Inc., Kansas City, MO
Sid Morris, Quintiles, Inc., Kansas City, MO

ABSTRACT

As CDISC Submission Data Model evolved to SDTM 1.01 (SDS 3.1), the perception/description of the data structure changed extensively in the submission data standards, though some general characteristics of the standards remain unchanged. These unchanged characteristics relate to fundamental data definition and presentation, such as definition of the variable attributes (type, length, format, and label); order of variable appearance within the dataset; and labeling of the dataset. When programming a CRT, the CRT domain may be generated by merging several source datasets, and the resulting CRT should subset variables to contain only the CRT domain defined variables. Programming these aspects of the standard/specification is not usually difficult, though it may be tedious and challenging to maintain if the CRT specifications for a given project change frequently. This paper presents a programming design for implementing the aforementioned aspects of CRT standard/specification dynamically and efficiently. With this flow design, CRT programming efforts can be focused on variable mapping and derivation.

INTRODUCTION

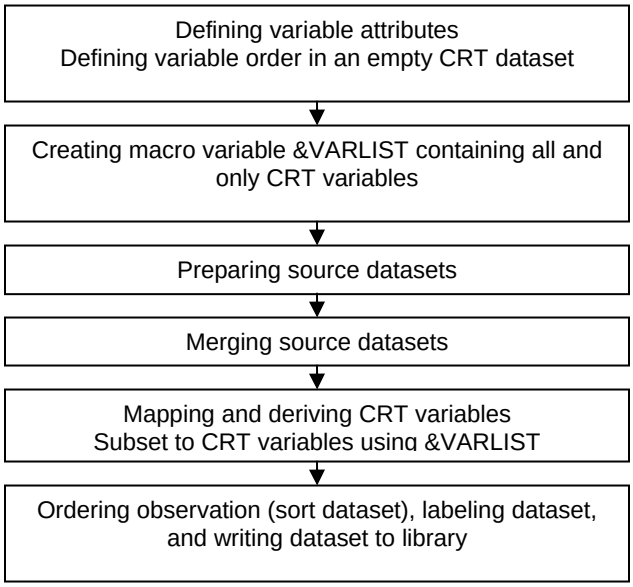
As CDISC (Clinical Data Interchange Standards Consortium) Submission Data Standard (SDS) evolved from SDS 1.x, 2.0, 3.0 to Study Data Tabulation Model (SDTM) 1.01 (SDS 3.1), the perception/ description of the data structure changed extensively in the submission data standards. These changes have resulted in considerable impact for the project team to develop their project Case Report Tabulation (CRT) specification. However, some general characteristics of the standards remain unchanged in so far as programming is concerned. These unchanged characteristics relate to fundamental data definition and presentation, such as definition of the variable attributes (type, length, format, and label); order of variable appearance within the dataset, and labeling of the dataset. This paper discusses the implementation of these aspects of the SDS/SDTM standards.

Ordering the observations in a dataset should also be an important attribute of the dataset affecting dataset presentation and review. CDISC does not recommend a set of variables for sorting each dataset, but does indicate that certain variables are included in the model to assist in sorting. Project CRT specifications will most likely contain sort information. This paper discusses when to sort the dataset.

Variables of each CRT domain may originate from multiple source datasets, which results in the necessity of merging these datasets. Additionally, CRT programming often requires that one map/derive CRT variables from source variables. Should one merge datasets and derive variables in the same data step, the program may succumb to a merge pitfall. An option to avoid this merge pitfall will be discussed. The resulting final CRT domain should contain a list of variables specified in the SDS/SDTM standard or project specific CRT specification (raw variables should be dropped). Specifying such a `keep` list of variables can be time consuming, and can be challenging to maintain when CRT specifications change frequently. Multiple version changes during CRT development are not uncommon especially while the whole industry is learning to comply with industry standards to generate CRTs. Multiple version changes also present challenges for programs to reflect current variable attribute definitions and variable ordering. This paper presents a programming design which can facilitate dynamic updates of variable attribute definitions; ordering of variables; and the list of `keep` variables. By automating these tasks, CRT programming efforts can remain focused on CRT variable mapping and derivations.

PROGRAM FLOW DESIGN

The tasks involved in CRT programming are variable mapping/derivation; definition of variable attributes; variable ordering and subsetting; dataset sorting and labeling; and writing the CRT to a library. Some prefer to define variable attributes when each contributing source dataset is processed and each CRT variable is mapped/derived. Others may prefer to define variable attributes just before writing the CRT to a library when all variables are mapped/derived. The programming flow presented here defines variable attributes in an empty CRT dataset before any CRT variables have been derived. Further discussion of advantages of this design will follow in later sections.



DEFINING VARIABLE ATTRIBUTES AND ORDER:
Using a data step and SQL procedure (See Appendix I) to create an empty dataset VARATT is the foundation of the illustrated CRT programming design. The empty dataset contains all CRT variables, with their attributes and order defined. Defining attributes and order of all variables in one place allows for easy maintenance and it can facilitate the automatic update of variable attributes and order when CRT specifications change versions. The code in Appendix I, Section 1 is generated by a SAS macro that reads the CRT specifications from a MS Excel spreadsheet.

CREATING MACRO VARIABLE &VARLIST
Once the empty CRT dataset containing all CRT variables and no observations is created, a macro variable &VARLIST can be generated by retrieving variable names from dictionary.columns using proc SQL (See code in Appendix I, Section 2). Should the CRT specifications change (to add/drop variables), the list of variables in &VARLIST will be automatically updated with an update of the empty dataset.

PREPARING SOURCE DATASETS:
As noted in SDS 3.1, data that were collected on multiple CRF modules may go into one V3 domain. It is not uncommon to utilize variables from three or more source datasets when programming the final CRT dataset. For example, a study day variable such as DMDY (Study day of collection) in DM (demographic) dataset needs an event date (demographic date from demographic dataset) and a reference date (often first dose date from drug administration dataset) to derive. In this case, it is often seen that CRT variables are mapped/derived when preparing each source dataset from the raw dataset. With this approach, the attributes of mapped/derived variables may need to be defined separately in steps when the variable is being mapped /derived, thus automatic update of variable attributes can not be supported. Maintenance can also be difficult since the variable definition is not centralized in one place. The authors' approach is to prepare each source dataset (such as sorting, subsetting, variable renaming, etc.) first and then mapping/deriving the CRT variables after all source datasets and the variable attribute definition dataset (VARATT) have been merged together.

MERGING SOURCE DATASETS:
This step is to merge all source datasets together if the merge keys from each of the contributing datasets are the same. More than one merge step may be necessary if the merge keys differ.

The merge step(s) also include(s) merging the empty dataset VARATT (and VARATT should be the first dataset on the merge statement). In doing so, the attributes and order of all CRT variables defined in VARATT will be maintained in the merged dataset, though they will all have blank values at this point.

While it may be natural to perform mapping and derivation of CRT variables in the same data step as merging, our approach is to separate the merging step from the mapping/derivation step. Should one map/derive in the same step as the merge, they incur the risk of falling into a merge pitfall as referenced earlier in this paper (See Appendix III).

The authors' experience has shown that keeping the source data preparation, merging, and mapping/derivation in distinct steps is clean, and enables one to avoid a merge pitfall.

MAPPING AND DERIVING CRT VARIABLES:
In this step, all/majority of CRT variables can be mapped and derived. We populate the blank CRT variables created from the previous step of merging VARATT with the source datasets. If all CRT variables have been mapped/derived and there exists no need to keep non-CRT variables, then variables may be subset to include only the CRT specified variables using a "`keep &VARLIST`" statement. In utilizing the macro variable &VARLIST, the `keep` list is dynamic and thus maintenance is automated.

The end of this data step may result in no further mapping/derivation for many CRT datasets (e.g., DEMO, PE). However, additional variables in certain datasets may need to be derived in subsequent data steps/procedures, for example, deriving baseline and change from baseline in the VITAL and LAB datasets.

SORTING AND WRITING CRT TO LIBRARY:
It is often the case that a sort procedure sorts a CRT dataset, followed by a data step, which defines CRT variable attributes, adds a dataset label and writes the CRT dataset to a library. Since the variable attributes have been defined at the beginning of the program as reflected in our presented design, the task of sorting variables, labeling the dataset, and writing the dataset to a specific location is all done in the sort procedure at the end of our design and implementation. Such a design not only eliminates the need of an additional data step, but more importantly, the sort key variables also become the attributes of the final CRT dataset and will be available in output of `proc contents` procedure. In this way, the sort key is self-documented in the resulting final CRT dataset.

SUMMARY

The presented program flow design is characterized by 1) creating an empty CRT dataset at the beginning of the program, 2) separating the merge of source data from CRT variable derivation, and 3) writing the dataset to a library while sorting the dataset.

Creation of an empty CRT dataset can facilitate automatic updates of CRT variable attribute definitions and automatic updates of the variable `keep` list. Separating the merging of source data from CRT variable derivation makes programming less susceptible to errors. These design features enable the CRT programmer to become more efficient, and in doing so, their efforts can be focused on variable mapping and derivation.

ACKNOWLEDGMENTS

The authors would like to thank Anne Freeman and Tony Salehi for reviewing the manuscript and providing valuable comments.

CONTACT INFORMATION

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

Sam Mao
Quintiles, Inc.
P.O. Box 9708
Kansas City, MO 64134-0708
Work Phone: (816) 767-3783
Fax: (816) 767-7372
Email: sam.mao@quintiles.com

Sid Morris
Quintiles, Inc.
P.O. Box 9708
Kansas City, MO 64134-0708
Work Phone: (816) 767-6624
Fax: (816) 767-7372
Email: sid.morris@quintiles.com

Appendix I: Programming example illustrating the implementation of presented program flow design:

```
/*************************************************************************;
Program name:: demo.sas
Purpose     :: Generate CRT DM per CDISC SDTM 3.1 for PharmaSug 2005
               Presentation
Author      :: Sam Mao and Sid Morris
Program flow design:
          - Defining variable attributes and order in empty CRT dataset
          - Creating macro variable &VARLIST to contain all CRT variables and
            only CRT variables
          - Preparing source datasets
          - Merging source datasets
          - Mapping and deriving CRT variables from source variables,
            Subset to CRT variables using &VARLIST
          - Further work on highly derived CRT variables (such as baseline for
            LAB, VITAL)
          - Sorting dataset, labeling dataset and writing to library
*************************************************************************/

%inc "\\rootdir\ABCdrug\study123\setup.sas";
*************************************************************************;
* ① Defining variable attributes and order in empty CRT dataset          *;
*************************************************************************;
data varatt;
  attrib
        STUDYID  length=$10    label="Study Identifier"
        DOMAIN   length=$2     label="Domain Abbreviation"
        USUBJID  length=$20    label="Unique Subject Identifier"
        SUBJID   length=$10    label="Subject Identifier for the Study"
        RFSTDTC  length=$10    label="Subject Reference Start Date"
        RFENDTC  length=$10    label="Subject Reference End Date"
        SITEID   length=$10    label="Study Site Identifier"
        BRTHDTC  length=$10    label="Date of Birth"
        AGE      length=8      label="Age in AGEU at Reference Date"
        AGEU     length=$6     label="Age Units"
        SEX      length=$8     label="Sex"
        RACE     length=$60    label="Race"
        ARMCD    length=$1     label="Planned Arm Code"
        ARM      length=$50    label="Description of Planned Arm"
        COUNTRY  length=$50    label="Country"
        DMDTC    length=$10    label="Date of Collection"
        DMDY     length=8      label="Study day of collection";

  *** Initialize the variables and defining variable order in dataset ***;
  array num(*) _numeric_;
  array chr(*) _character_;
  ptno=.;
  if ptno=9999;
run;
```

```
*************************************************************************;
*② Create macro var VARLIST having all CRT variables as its value *;
*************************************************************************;
proc sql noprint;
  select a.name into :VARLIST
  separated by ' '
  from dictionary.columns a
  where libname='WORK' and memname='VARATT' and name ne 'PTNO';
quit;

*************************************************************************;
* ③ Preparing source datasets                                        **;
*************************************************************************;
*** Preparing demographic dataset ***;
proc sort data=rawdata.demo(rename=(sex=sexcrf race=racecrf)
          out=demo(keep=ptno protocol site sexcrf racecrf birthdt visitdt);
  by ptno;
run;

/*** Deriving fdosdtc, ldosdtc from drug administration dataset ***/
proc sort data=rawdata.drgadmin out=drgadmin;
  by ptno dsdate dstime;
run;

*** ... Further steps/procedures to get FDOSDTC and LDOSDTC ***;
*** ... Further steps/procedures to get FDOSDTC and LDOSDTC ***;

*** Preparing source dataset DRGADMIN to have FDOSDTC, LDOSDTC ready ***;
data drgadmin;
  set drgadmin;
  fdosdtc=put(fdosdt, yymmdd10.);
  ldosdtc=put(ldosdt, yymmdd10.);
  keep ptno fdosdtc ldosdtc;
run;

/*** preparing source dataset with treatment arm information ***/
*** Assuming study already unblinded, and treatment info is in rawdata.RANDO dataset
***;
proc sort data=rawdata.RANDO out=trtarm(keep=ptno trtcd trtdesc);
  by ptno trtcd trtdesc;
run;


*************************************************************************;
* ④ Merging all source datasets together                              *;
*************************************************************************;
data dm;
  merge varatt demo drgadmin trtarm;
  by ptno;
run;
```

```
*************************************************************************;
* ⑤ Mapping/Deriving CRT variables                                      *;
*************************************************************************;
data dm;
  keep &VARLIST;
  set dm;

  studyid=protocol;  *** protocol is a raw variable having study ID ***;
  domain='DM';
  usubjid=trim(studyid)||left(put(ptno, 10.));
  subjid=left(put(ptno, 10.);
  rfstdtc=fdosdtc;
  rfendtc=ldosdtc;
  siteid =left(put(site, 10.));
  ** convert numeric date BIRTHDT to char date BRTHDTC in ISO 8601 format **;
  brthdtc=put(birthdt, yymmdd10.);
  *** This is to show AGE derivation, not the true age algorithm ***;
  age=birthdt - input(rfstdtc, yymmdd10.);
  ageu='YEARS';

  select (sexcrf);
    when (1) sex='M';
    when (2) sex='F';
    otherwise;
  end;

  select (racecrf);
    when (1) race="White or Caucasian"                          ;
    when (2) race="Black or African American"                   ;
    when (3) race="Hispanic or Latino"                          ;
    when (4) race="Asian"                                       ;
    when (9) race="Other"                                       ;
    otherwise;
  end;

  armcd=trtcd; *** with values such as A, B, C ***;
  arm  =trtdesc;
  country="UNITED STATES";
  dmdtc=put(visitdt, yyddmm10.);

  *** the calculation can only be done after merging of two source datasets ***;
  dmdy=visitdt - input(rfstdtc, yyddmm10.);
run;


*************************************************************************;
* ⑥ Sorting dataset, labeling dataset, and writing dataset to library   *;
*************************************************************************;

proc sort data=dm out=crtdata.dm (label="Demographics");
  by usubjid;
run;
```

Appendix II: Example illustrating CRT program in alternative flow design, which makes automatic update of variable attributes difficult.

```
/*************************************************************************;
Program name:: demo_style2.sas
Purpose     :: Generate CRT DM per CDISC SDTM 3.1 for PharmaSug 2005
               Presentation
Author      :: Sam Mao and Sid Morris

Style 2:
          - Map/Derive CRT variables when processing each individual
            contributing source dataset;
          - Merge source datasets and further derive remaining CRT
            variables
          - Subset variables to keep CRT variables only (or drop unwanted
            variables)
          - Order variables in CRT datast
          - Order observations (sort dataset)
          - Define CRT variable attributes, label dataset
            and write to library
*************************************************************************/

%inc "\\rootdir\ABCdrug\study123\setup.sas";

*************************************************************************;
* ① Map/Derive CRT variables when processing each individual source dataset***;
* map/derive CRT variables when source variables are available              ***;
*************************************************************************;

/*** ①a: Processing demographic dataset ***/
proc sort data=rawdata.demo(rename=(sex=sexcrf race=racecrf)
          out=demo(keep=ptno protocol site sexcrf racecrf birthdt visitdt);
  by ptno;
run;

*** map/derive CRT variables contributed by rawdata.demo dataset ***;
data demo;
  *** CRT variable attribute may/may not be defined here ***;
  length brthdtc dmdtc $10 race $60 usubjid $20 sex $8;

  set demo;
  studyid=protocol;  *** protocol is a raw variable having study ID ***;
  domain='DM';
  usubjid=trim(studyid)||left(put(ptno, 10.));
  subjid=left(put(ptno, 10.);
  siteid =left(put(site, 10.));
  *** convert numeric date BIRTHDT to char date BRTHDTC***;
  brthdtc=put(birthdt, yymmdd10.);
  select (sexcrf);
    when (1) sex='M';
    when (2) sex='F';
    otherwise;
  end;

  select (racecrf);
    when (1) race="White or Caucasian"                          ;
    when (2) race="Black or African American"                   ;
    when (3) race="Hispanic or Latino"                          ;
    when (4) race="Asian"                                       ;
    when (9) race="Other"                                       ;
    otherwise;
  end;

  country="UNITED STATES";
  dmdtc=put(visitdt, yyddmm10.);
run;
```

```
/*** ①b Process rawdata.drgadmin data to derive RFSTDTC, RFENDTC ***/
*** First bring in the raw data and sort by date, time variables ***;
*** Assuming date and time variables are in numeric format ***;
proc sort data=rawdata.drgadmin out=drgadmin;
  by ptno dsdate dstime;
run;

*** ... Further steps/procedures to get RFSTDTC and RFENDTC ***;
*** ... Further steps/procedures to get RFSTDTC and RFENDTC ***;
*** ... Further steps/procedures to get RFSTDTC and RFENDTC ***;

*** This step is simply to show RFSTDTC, RFENDTC derived from above steps ***;
data drgadmin;
  *** CRT variable attribute may/may not defined here ***;
  length RFSTDTC RFENDTC $10;
  set drgadmin;
  RFSTDTC=put(fdosdt, yymmdd10.);
  RFENDTC=put(ldosdt, yymmdd10.);
  keep ptno RFSTDTC RFENDTC;
run;

/*** ①c Processing rawdata.RANDO to get treatment arm information ***/
*** Assuming study already unblinded, and arm info is in rawdata.RANDO ***;
proc sort data=rawdata.RANDO out=trtarm(keep=ptno trtcd trtdesc);
  by ptno trtcd trtdesc;
run;

***Map/derive treatment arm code, and arm ***;
data trtarm;
  *** CRT variable attribute may/may not defined here ***;
  length armcd $1 arm $50;
  set trtarm;
  armcd=trtcd;
  arm=trtdesc;
  keep ptno armcd arm;
run;

*****************************************************************************;
* ② Merge all source datasets, further map/derive remaining CRT variables   *;
*****************************************************************************;
data dm;
  *** CRT variable attribute may/may not defined here ***;
  length ageu $6;
  merge varatt demo drgadmin trtarm;
  by ptno;
  set dm;

  *** This is to show AGE derivation, not the true age algorithm ***;
  age=birthdt - input(rfstdtc, yymmdd10.);
  ageu='YEARS';

  *** the calculation can only be done after merging of two source datasets ***;
  dmdy=visitdt - input(rfstdtc, yyddmm10.);


*** The variable list need to be created manually ***;
*** and it can be challenging to maintain         ***;
  keep STUDYID DOMAIN USUBJID SUBJID RFSTDTC RFENDTC SITEID BRTHDTC
       AGE AGEU SEX RACE ARMCD ARM COUNTRY DMDTC DMDY;
run;
```

```
*******************************************************************************;
* ③ Order CRT variables so that variables appear in dataset as desired     ***;
*******************************************************************************;
data dm;
  retain STUDYID DOMAIN USUBJID SUBJID RFSTDTC RFENDTC SITEID BRTHDTC
         AGE AGEU SEX RACE ARMCD ARM COUNTRY DMDTC DMDY;
  set dm;
run;

*******************************************************************************;
* ④ Order observations (sort dataset)                                      ***;
*******************************************************************************;

*** order observations ***;
proc sort data=dm out=dm;
  by usubjid;
run;

*******************************************************************************;
* ⑤ Define variable attributes, label CRT dataset, write to library        ***;
*******************************************************************************;

data crtdata.dm (label="Demographics");
  attrib
        STUDYID  length=$10    label="Study Identifier"
        DOMAIN   length=$2     label="Domain Abbreviation"
        USUBJID  length=$20    label="Unique Subject Identifier"
        SUBJID   length=$10    label="Subject Identifier for the Study"
        RFSTDTC  length=$10    label="Subject Reference Start Date"
        RFENDTC  length=$10    label="Subject Reference End Date"
        SITEID   length=$10    label="Study Site Identifier"
        BRTHDTC  length=$10    label="Date of Birth"
        AGE      length=8      label="Age in AGEU at Reference Date"
        AGEU     length=$6     label="Age Units"
        SEX      length=$1     label="Sex"
        RACE     length=$60    label="Race"
        ARMCD    length=$1     label="Planned Arm Code"
        ARM      length=$50    label="Description of Planned Arm"
        COUNTRY  length=$50    label="Country"
        DMDTC    length=$10    label="Date of Collection"
        DMDY     length=8      label="Study Day of Collection"
        ;
  set dm;
run;
```

Appendix III: Example illustrating pitfall when data manipulation/derivation in the merge step.

Maximum vital sign change (CHNG) from pre-infusion by visit

SUBJID	VSTESTCD	VISITNO	MAXCHG
1001	DIABP	1	7.00
1001	DIABP	3	2.00

Vital sign dataset (VITAL)

SUBJID	VSTESTCD	VISITNO	VSPTMCD	VSPTM
1001	DIABP	1	120	2 hours after infusion
1001	DIABP	1	140	3 hours after infusion
1001	DIABP	1	160	4 hours after infusion
1001	DIABP	1	998	Physical Exam
1001	DIABP	3	120	2 hours after infusion
1001	DIABP	3	998	Physical Exam
1001	DIABP	3	140	3 hours after infusion
1001	DIABP	3	160	4 hours after infusion

```
*** The task: Populate MAXCHG to each vital sign record per visit        ***;
*** In the following merge, the merge key is: SUBJID VSTESTCD VISITNO    ***;
*** Data order in the same merge key group may be random, and unknown    ***;
```

Approach ①: Merging, and data manipulation/deriving variables in the **SAME DATA STEP**

```
data vital;
  merge examp.vital examp.chng;
  by subjid vstestcd visitno;
  if vsptmcd=998 then maxchg=.;
run;
```

Problem occurred - MAXCHG for some observations **incorrectly set to missing** (see shaded records)

SUBJID	VSTESTCD	VISITNO	VSPTMCD	VSPTM	MAXCHG
1001	DIABP	1	120	2 hours after infusion	7.00
1001	DIABP	1	140	3 hours after infusion	7.00
1001	DIABP	1	160	4 hours after infusion	7.00
1001	DIABP	1	998	Physical Exam	.
1001	DIABP	3	120	2 hours after infusion	2.00
1001	DIABP	3	998	Physical Exam	.
1001	DIABP	3	140	3 hours after infusion	.
1001	DIABP	3	160	4 hours after infusion	.

Approach ②: Merging data, and data manipulation/deriving other variables in **SEPARATE DATA STEPS**

```
data vital;
  merge examp.vital examp.chng;
  by subjid vstestcd visitno;
run;

data vital;
  set vital;
  *** . . . all other manipulation/derivation  . . . ***;
  if vsptmcd=998 then maxchg=.;
  *** . . . all other manipulation/derivation  . . . ***;
run;
```

Problem free - Value of variable MAXCHG is **set correctly for each observation**

SUBJID	VSTESTCD	VISITNO	VSPTMCD	VSPTM	MAXCHG
1001	DIABP	1	120	2 hours after infusion	7.00
1001	DIABP	1	140	3 hours after infusion	7.00
1001	DIABP	1	160	4 hours after infusion	7.00
1001	DIABP	1	998	Physical Exam	.
1001	DIABP	3	120	2 hours after infusion	2.00
1001	DIABP	3	998	Physical Exam	.
1001	DIABP	3	140	3 hours after infusion	2.00
1001	DIABP	3	160	4 hours after infusion	2.00