SDTM Domain Mapping with Structured Programming Methodology

Chengxin Li, Boehringer Ingelheim, Ridgefield, CT

Jing-Wei Gao, Boehringer Ingelheim, Ridgefield, CT

Nancy Bauer, Boehringer Ingelheim, Ridgefield, CT

ABSTRACT

This paper describes the implementation of SDTM domain mapping with structured programming methodology. Structured design and structured programming make our mapping codes easy to implement, understand, debug, and maintain. Example codes for domain mapping are provided, with which the readers can easily adapt to meet their mapping requirements.

INTRODUCTION

CDASH AND SDTM

Clinical Data Acquisition Standards Harmonization (CDASH) [1], Study Data Tabulation Model (SDTM) [2] are the key components of Clinical Data Interchange Standards Consortium (CDISC) [1].

CDASH, harmonized with SDTM, standardizes structure and the metadata for underlying clinical trial data collections with corresponding domains. It defines clinical data collection standards from the beginning of clinical trials set-up. The current version CDASH V1.1 provides 16 standard collection domains.

SDTM defines standard structures and model rules of clinical trial data tabulations for submission to regulatory authority such as FDA. The latest Study Data Tabulation Model Implementation Guide (SDTMIG) v3.1.2 [3] defines implementation rules for over 30 standard domains. These domains are categorized as special-purpose (DM, CO, SE, SV), interventions (CM, EX, SU), events (AE, DS, MH, DV, CE), findings (EG, IE, LB, PE, QS, SC, VS, DA, MB, MS, PC, PP, FA), trial design (TA, TE, TV, TI, TS), and relationship (SUPPQUAL, RELREC).

STRUCTURED DESIGN AND PROGRAMMING

Structured design is the methodology that divides a complex task into smaller conquered modules (procedures) and interrelate those modules (procedures) [4]. The flowcharts are used to dictate data flow. The structured design is implemented by structured programming [5].

In SAS®, the structured programming are represented by Macros for repeated blocks, DATA STEP SEQUENCES, IF-THEN-ELSE or CASE WHEN and DO-LOOP structures, etc.

DESIGN AND IMPLEMENTATION

The philosophy of structured design is from generic to specific, and dividing complex tasks into a series of small procedures.

Figure 1 shows the process flow we used for domain mapping. Figure 1 consists of three sections. Section I dictates the overall process flow. Clinical data are collected through data capture tool with CRF (Case Report Form), which

1 http://www.cdisc.org/

preferably complies with CDASH. Then the data are stored in Oracle. The SAS mapping program driven by Master file maps Oracle data (OCVIEWs) to SAS datasets--Domains. The Master file is an excel sheet which holds the domain components, standardizes meta data of SDTM variables and company specific variables, and provides mapping specifications. The meta data defines variable attributes by variable and domain. The SDTM variables are mapped to corresponding SDTM domain, while the company specific variables go to SUPP (supplemental) domain or FA (Finding About) domain or CO (comment) domain. Before each final domain is generated, the domain meta data are reviewed by data manager. After the final domain is created, an open source tool, OpenCDISC Validator (http://www.opencdisc.org/projects/validator), is used for domain compliance check.

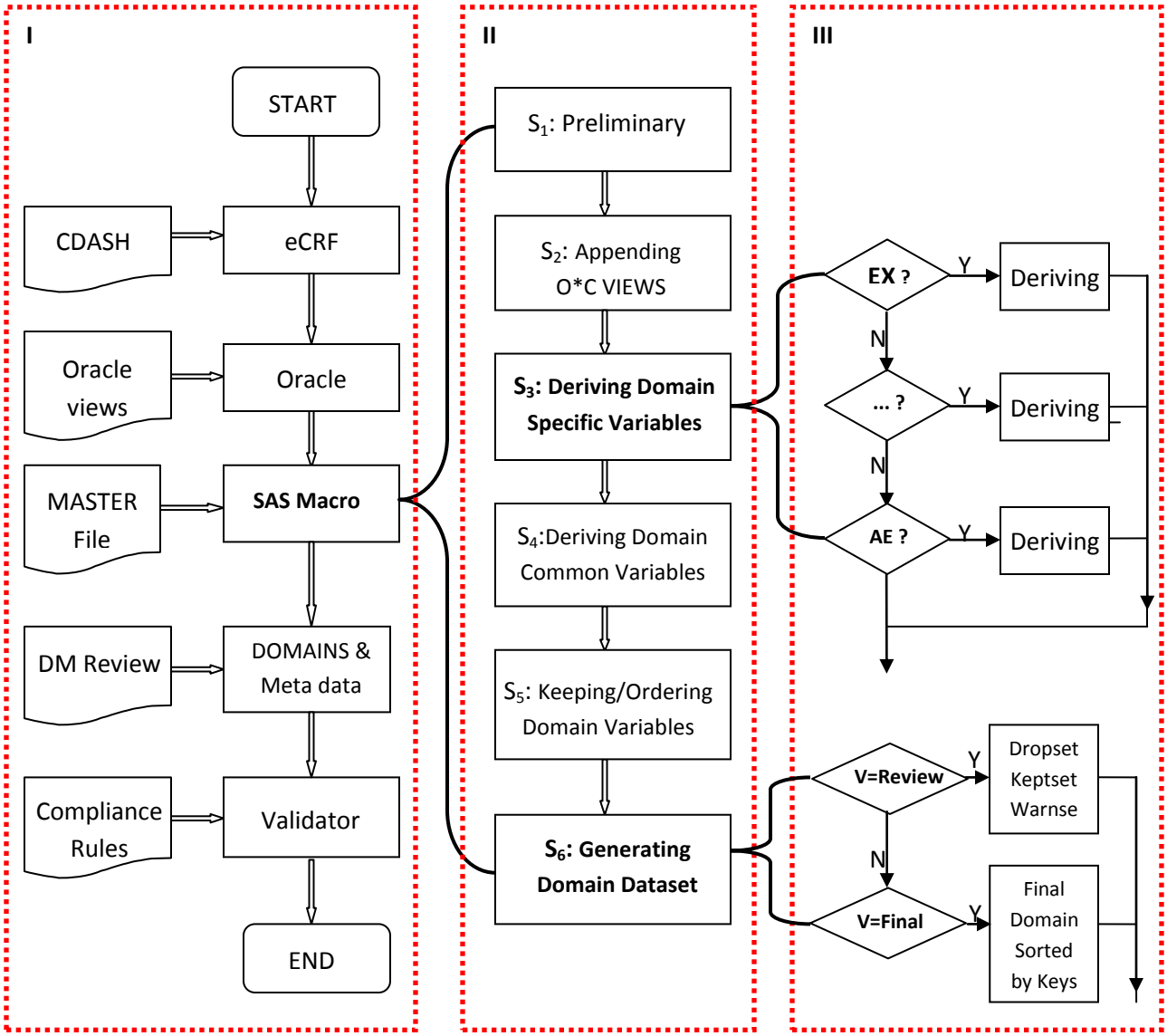

Figure 1 Domain Mapping Process Flow

The section II of figure 1 shows the framework of mapping. We define six steps for each individual domain mapping.

- Step 1 is preliminary step. It sets up the environment, defines macro variables, loads the Master file, etc.
- Step 2 is for appending Oracle views. One domain may have one or more Oracle views as inputs.
- Step 3 functions to derive domain specific variables such as Date/Time variables (DTC) with ISO8601 format and Study Day variables (--DY) with numeric type. Structure transpose from horizontal to vertical is also performed in this step like in VS domain.

- Step 4 is for deriving common variables dynamically for all the domains to reduce code redundancy. Those variables are DOMAIN, --SEQ, etc.

- Step 5 is where we conduct keeping and dropping variables. We work off from two sets, Set A and Set B. Set A contains all the variables of a specific domain defined in Master file; Set B contains all the variables derived and populated in Oracle View(s) for the same domain in Set A. The figure 2 VENN graph shows the set operation results between the set A and set B.

KEPTSET (brown)= Set A and Set B; KEPTSET is the set of all shared variables by Set A and B and will be populated in the final domain, or go to SUPP (supplemental) domain or FA (Finding About) domain or CO (comment) domain.

DROPSET (pink)=Set B and not Set A; DROPSET is the set of all the variables which will be dropped from the Oracle view(s), including CDASH specific variables and administration variables.

WARNSET (green)=Set A and not Set B; WARNSET is the set of all the variables defined in Master file but neither collected nor derived in the trial.

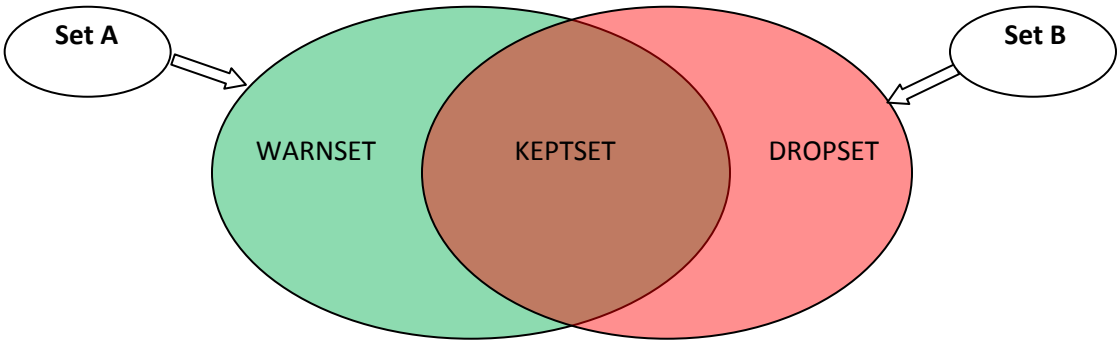

Figure 2 VENN Graph

- Step 6 is where the final domain generation occurs. Step 6 attaches domain label, snapshot, and the date of domain generation. The final domain is sorted by KEYs. Although SDTM IG v3.1.2 gives KYE recommendation at p18-19, we define our KEY set according to our business model.

The general rules for KEYs are to use REQ and EXP variables as KEY options. Three level keys are defined. Identification variable STUDYID and USUBJID as the first level keys; the date and timing variables (e.g., visitnum, --STDTC, --ENDTC) as the second level keys; topic or qualifier variables as the third level keys (--TERM, --DECOD, etc). In most cases, the third level keys are not going to take effects after sorting by the second level keys. SEQ will be generated accordingly by the sequences per subject after sorting by the keys.

The section III of figure 1 details step 3 and step 6 with the derived flowchart structures.

For data review purpose, we output meta data of dropset, keptset, and warnset for each domain into REVIEW library in step 6. After review, the final domain is output into SDTM library.

There also exist dependencies among domains (Oracle views). For instance, to derive --STDY and --ENDY variables, DM.RFSTDTC and DM.RFENDTC are needed. Therefore, DM domain (Oracle view) should be generated first. Figure 3 exhibits this kind of dependencies.

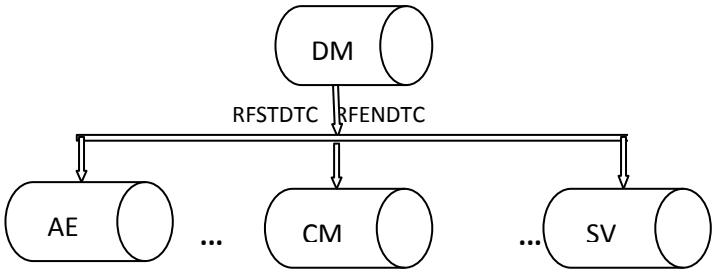

Figure 3 Domain Dependency

EXAMPLE CODES FOR MH DOMAIN MAPPING

We implemented the domain mapping with SAS MACRO. The macro has three parameters:
trgt: the target domain, which will be produced as an output;
src : the source Oracle view(s), which contributes to the target domain;
ver: either review or final. When ver=review, the kept variable set (KEPTSET), dropped variable set (DROPSET), warning variable set (WARNSET), and the interim domain will be output to REVIEW lib for data manager's review. After review, re-run mapping program with ver=final, and the final produced DOMAIN will be sent to SDTM lib.

Here gives MH as an example showing how the macro works and how the structured programming implemented. The macro call for mapping MH domain are:

```
%mapping(trgt=%str(MH), src=%str(ocview.DMMHTD ocview.MHBCON),ver=review)
%mapping(trgt=%str(MH), src=%str(ocview.DMMHTD ocview.MHBCON),ver=final)
```

The above calls indicate there are two Oracle views contributing to MH domain, DMMHTD and MHBCON. The generated target domain is MH with review version for data manager, and with final version for programmer.

```
/* Step 1: prelimitary. */
  PROC IMPORT OUT=&trgt._master
              DATAFILE= "v:\Master.xls"
              DBMS=EXCEL REPLACE;
       RANGE="&trgt.$";
       GETNAMES=YES;
       MIXED=NO;
       SCANTEXT=YES;
       USEDATE=YES;
       SCANTIME=YES;
  RUN;

  DATA &trgt._selectM;
    SET &trgt._master;
    KEEP MVarName MOrder MKey MType MLength MFormat MLabel;
  RUN;
```

The above codes load MH sheet of Master file and keep required variables (variable name, variable order, variable key, variable type, variable length, variable format, and variable label) into SAS dataset MH_selectM. It may also subset key variables from MH_selectM by proc contents and proc sql into macro variable &key.

```
/* step2: appending ocviews. */
    DATA &trgt.1;
      SET &src;
      BY USUBJID;
    RUN;
```

The &src is resolved to string “ocview.DMMHTD ocview.MHBCON”. Afte step 2, ocview.DMMHTD and ocview.MHBCON are appended to MH1 dataset. MH1 is transferred to step 3.

```
/* step3: domain specific variable derivations */

  %if %upcase(&trgt)=MH %then %do;

    PROC CONTENTS data=&trgt.1 out=MHvar (keep=name varnum) noprint;
    RUN;
    PROC SQL noprint;
      SELECT name into: MHvar separated by ' '
      FROM MHvar;
    QUIT;

    %if %index(&MHVAR, &trgt.DAT)>0 %then %do;
    PROC SQL;
      CREATE table &trgt.2(drop=STDATE) as
      SELECT a.*, input(a.&trgt.DAT, yymmdd9.) as STDATE format=date9.,
             CASE
               WHEN length(a.&trgt.STDAT)=8 and calculated STDATE>=b.RFSTDT
                 THEN (calculated STDATE)-b.RFSTDT+1
               WHEN length(a.&trgt.STDAT)=8 and calculated STDATE< b.RFSTDT
                 THEN (calculated STDATE)-b.RFSTDT
             END AS &trgt.DY length=8 label="Study Day of History Collection" format=8.
      FROM &trgt.1 a, &SDTM..DM b
      WHERE a.USUBJID=b.USUBJID;
    QUIT;
    %end;
    %else %do;
      %put "&trgt.DAT not collected, thus &trgt.DY not derived. ";
      DATA &trgt.2; /* no needs to derive MHDY then. */
        SET &trgt.1;
      RUN;
    %end;
  %end;
```

Step 3 derives MH specific variables. Here gives how to derive MHDY variable. MHDY is derived only if MHDAT (CDASH specific variable) is collected. The above codes first get variable list in MH1 and put those variables to &MHvar. If MHDAT is existing in &Mhvar, then derive MHDY. During deriving MHDY, Reference Start Date in DM (DM.RFSTDT) is needed. The MH2 dataset is generated and output to step 4.

```
/* step4: deriving common variables (Domain, --seq) */
  PROC SORT data=&trgt.2;
    BY &KEY;
  RUN;

  DATA &trgt.3;
    SET &trgt.2;
    BY USUBJID;
    DOMAIN=upcase(symget('trgt'));
    RETAIN seq;
    IF FIRST.USUBJID THEN SEQ=0;
    SEQ=(seq+1);
    &trgt.seq=seq;
    DROP seq;
  RUN;
```

This step is simple. First sort MH2 by MH domain keys, then derive common variables like DOMAIN and MHSEQ. The generated MH3 is delivered to step 5.

```
/* step 5: keep BI-CDISC variables */

  /*get the var list to be processed (ocview+derived)--target set */
  PROC CONTENTS data=&trgt.3 out=&trgt._target noprint;
```

```
  RUN;

  /* get the kept var set, droped var set, warning var set */
  DATA &trgt._keptset(KEEP=MVarName MOrder MType MLength Length MFormat Format MLabel
Label MScope)
       &trgt._dropset(keep=MVarName Label)
       &trgt._warnset(keep=MVarName MOrder MType MLength MFormat MLabel MScope);
    MERGE &trgt._selectM (in=a) &trgt._target (in=b);
    BY mvarname;
    if (a and b) then output &trgt._keptset;
    if (not a and b) then output &trgt._dropset;
    if (a and not b) then output &trgt._warnset;
  RUN;


  PROC SORT data=&trgt._keptset SORTSEQ = UCA (NUMERIC_COLLATION=ON);
    BY morder;
  RUN;
```

The above codes generate MH_keptset, MH_dropset, and MH_warnset for review purpose, and sorts the kept set in variable sequence order in Master file, i.e., in the MH variable sequences defined in SDTM IG v3.1.2.

```
  /* write the keptset, dropset, warnset into macro variables */
  PROC SQL noprint;
    SELECT STRIP(mvarname),COUNT(mvarname)
              INTO :kept_set separated by ', ',
                   :kept_no
    FROM &trgt._keptset;

    SELECT STRIP(mvarname),COUNT(mvarname)
              INTO :drop_set separated by ', ',
                   :drop_no
    FROM &trgt._dropset;

    SELECT STRIP(mvarname),COUNT(mvarname)
              INTO :warn_set separated by ', ',
                   :warn_no
    FROM &trgt._warnset;
  QUIT;


  /* generate the domain_set (&trgt.4) with the keptset */
  PROC SQL;
    CREATE table &trgt.4 as
    SELECT &kept_set
    FROM &trgt.3;
  QUIT;
```

Finally, the dataset MH4 is generated with required variables and the variables kept in proper order.

```
/* step 6: sort with BI-CDISC keys, generate datasets and domain either for review or
for final with label, date, and snapshot */

  %if %upcase(&ver)=REVIEW %then %do;
    DATA &review..&trgt._keptset (LABEL="The Kept Variables in Target Set for &trgt
Domain ");
      SET &trgt._keptset;
      LABEL MVarname='Variable Name';
      LABEL Morder='Variable Order in Master';
      LABEL Mlabel="Variable Label in Master";
      LABEL Label="Variable Label in OCVIEW";
      LABEL Mtype="Variable Type in Master";
      LABEL Mlength="Variable Length in Master";
      LABEL Length="Variable Length in OCVIEW";
```

```
      LABEL Mformat="Variable Format in Master";
      LABEL Format="Variable Format in OCVIEW";
      LABEL Mscope="Variable Scope in Master";
    RUN;

    DATA &review..&trgt._dropset (LABEL="The Dropped Variables in Target Set for &trgt
Domain ");
      SET &trgt._dropset;
      LABEL MVarname='Variable Name in OCVIEW';
      LABEL Label="Variable Label in OCVIEW";
    RUN;

    DATA &review..&trgt._warnset (LABEL="The Master Variables not in Target Set for
&trgt Domain ");
      SET &trgt._warnset;
      LABEL MVarname='Variable Name in Master';
      LABEL Morder='Variable Order in Master';
      LABEL Mlabel="Variable Label in Master";
      LABEL Mtype="Variable Type in Master";
      LABEL Mlength="Variable Length in Master";
      LABEL Mformat="Variable Format in Master";
      LABEL Mscope="Variable Scope in Master";
    RUN;

    PROC SORT data=&trgt.4 out=&review..&trgt noduplicate;
      BY &key;
    RUN;
  %end;
  %else %if %upcase(&ver)=FINAL %then %do;
    /* final domain generated */
    PROC SORT data=&trgt.4 out=&SDTM..&trgt noduplicate;
      BY &key;
    RUN;
  %end;
```

With parameter ver=REVIEW, the above codes in step 6 write KEPTSET, DROPSET, WARNSET, and MH domain as well into REVIEW lib for data manager's review. It may also add one extra variable (e.g., DIFF) in KEPTSET to indicate any inconsistency existed between variable attributes in Master and Oracle views.

After data review from data manager is confirmed, the parameter is update to ver=FINAL, and the final MH dataset is sent to SDTM lib.

CONCLUSION

In our experiences, with structured design and structured programming methodologies for SDTM domain mappings, our mapping codes bocome easy to implement, understand, debug, and maintain. Whenever adding one domain, the codes are easily extended only in step 3 for domain specific variable derivations.

The above listed code framework can be easily adapted to meet readers' mapping requirements.

REFERENCES

[1]CDISC CDASH Team. 18 January 2011. Clinical Data Acquisition Standards Harmonization (CDASH) v1.1.

[2]CDISC Submission Data Standards Team. April 28, 2005. Study Data Tabulation Model v1.1.

[3]CDISC Submission Data Standards Team. November 12, 2008. Study Data Tabulation Model Implementation Guide: Human Clinical Trials V3.1.2.

[4]Errol Pelchat, A Brief Introduction to Structured Design, Feb. 2004,

www.cs.cofc.edu/~bowring/classes/csci 360/presentations/structured design.ppt

[5]Structured Programming, http://en.wikipedia.org/wiki/Structured_programming

CONTACT INFORMATION

Author Name: Chengxin Li, Jing-Wei Gao, Nancy Bauer
Company: Boehringer Ingelheim
Address: 900 Ridgebury Road
City State Zip: Ridgefield, CT 06877
Email: chengxin.li@boehringer-ingelheim.com