

SDTM Harmonization in the Absence of CDASH – A Modularized Approach to Domain Programming

Annie Guo, ICON Clinical Research, San Francisco, California

ABSTRACT

SDTM is the recommended format for submissions to the FDA. CDASH defines basic standards for the collection of clinical trial data. The two standards streamline the processes of converting CRF data to SDTM format. However, in the absence of CDASH, de-normalized database structure and lacking the use of controlled terms can cause SDTM programming time-consuming and error-prone. This paper presents a modularized approach to developing reusable code and facilitating programming. The outcome is reduced programming time and quality output SDTM data.

INTRODUCTION

Clinical database in alignment with CDASH standard has the following features.

- Database is normalized.
- Variable names are consistent among similar forms per study, or for the same forms across studies.
- Controlled terminology is part of the database.

When CDASH is not built into clinical data collection or not applicable to the modalities of interest, CRF database might not be conformant to CDISC standards. For example, proprietary questionnaires that are mapped to SDTM QS domain are presented in their validated manner. That means a questionnaire can have tens of data fields to collect individual reply choices and to program in SAS®. If there are separate forms for different visits, such as Columbia-Suicide Severity Rating Scale (C-SSRS) Baseline and Since Last Visit, the two forms are stored in different data sets, and the number of variables to be processed and mapped to SDTM QSORRES is multiplied. Furthermore, each reply choice from questionnaires is assigned a test name QSTEST, test code QSTESTCD, category QSCAT and sub-category QSSCAT if applicable. The values on those variables are usually adapted from pre-printed text on questionnaires, which can take a lot of typing in SAS programs and lead to mapping error or difficulties. On top of that, if the study collects data from several questionnaires, each questionnaire is processed individually, and the workload to program a single QS domain can be enormous.

This paper presents a modularized approach to facilitating SDTM domain mapping in the absence of CDASH. The following sections describe concept and techniques, followed by applications for QS domain and sample code.

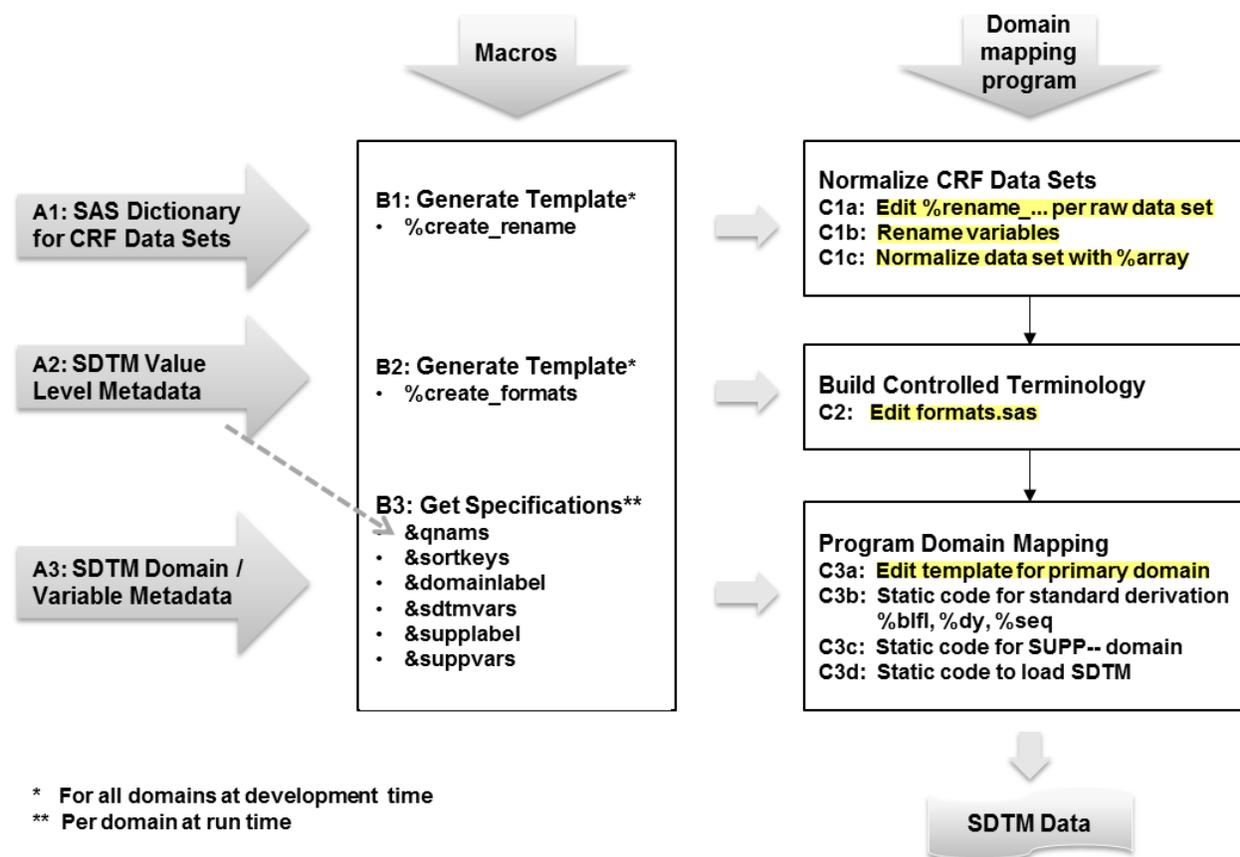
CONCEPT AND TECHNIQUES

The concept and techniques of SDTM harmonization without CDASH are as follows.

- Transform clinical database to CDASH-like structure to maximize reusable code. This is implemented by imposing naming convention with the RENAME statement, and normalizing CRF database with DATA step and ARRAY statement. Flowchart 1, **A1** → **B1** → **C1a**, **C1b** and **C1c** illustrates this.
- Carry over controlled terminology from external sources to domain mapping programs, to skip manually typing the text in SAS programs. This is implemented with dynamic programming to generate custom formats that follow the syntax of the FORMAT procedure, for example, `proc format; value $qstest "QSTESTCD_value" = "QSTEST_value";` and for use in domain programming as `qstest = put(qstestcd, $qstest.)` Flowchart 1, **A2** → **B2** → **C2** illustrates this.

After the CRF data sets are re-structured in a consistent manner and controlled terms are organized as custom formats, mapping CRF data to SDTM variables is done via template programming with minimal editing at development time. Final step to complete SDTM transformation is to execute reusable code, such as deriving --SEQ variable and creating SUPP-- data set, and load the data to permanent SAS data set. This is illustrated in Flowchart 1, **A3** → **B3** → **C3a**, **C3b**, **C3c** and **C3d**. Note that **C3b**, **C3c** and **C3d** are static code and do not require editing.

The numbering in Flowchart 1, **A1**, **B1**, etc., is referenced where the techniques are described in later sections of this paper.



Flowchart 1. Modularized Approach to SDTM Domain Mapping, Manual Steps in Yellow Mark

APPLICATIONS FOR QS DOMAIN

Take C-SSRS questionnaires as an example. Table 1 is partial aCRF showing the first few lines. There are 39 questions on C-SSRS Baseline, and 34 questions on C-SSRS Since Last Visit. Data are stored in two SAS data sets, `css1` and `css2`, respectively. Variable names in the two data sets are different, that is, $39 + 34 = 73$ variables in total. Comparing the two forms identifies only 40 unique questions.

Tables 2 and 3 show the SDTM QS/SUPPQS domain level and variable level metadata, respectively. They cover the attributes of SAS data sets and variable, and the variables used to sort records per SAS data set.

Table 4 lists the value level metadata for QS/SUPPQS. It includes the controlled terminology for the 40 unique questions, --TESTCD / QNAM, --TEST/QLABEL, --CAT, --SCAT and Orig/QORIG.

In reference to the steps in Flowchart 1, QS domain mapping program consists of the following modules. They are described in next sections.

- CDASH transformation – naming convention (**A1** → **B1** → **C1a** and **C1b** in Flowchart 1)
- CDASH transformation – normalization (**C1c** in Flowchart 1)
- Controlled terminology – custom formats (**A2** → **B2** → **C2** in Flowchart 1)
- Domain mapping – template programming for primary domain (**A3** → **B3** → **C3a** in Flowchart 1)
- Domain mapping – static code for standard derivations (**C3b** in Flowchart 1)
- Domain mapping – static code for SUPP-- mapping (**C3c** in Flowchart 1)
- Domain mapping – static code to load SDTM (**C3d** in Flowchart 1)

| SUICIDAL IDEATION | | | |
|---|-------------------------------------|--------------------------|---|
| <i>Ask questions 1 and 2. If both are negative, proceed to "Suicidal Behavior" section. If the answer to question 2 is "yes", ask questions 3, 4 and 5. If the answer to question 1 and/or 2 is "yes", complete "Intensity of Ideation" section below.</i> | | | Lifetime: Time He/She Felt Most Suicidal |
| 1. Wish to be Dead Subject endorses thoughts about a wish to be dead or not alive anymore, or wish to fall asleep and not wake up. <i>Have you wished you were dead or wished you could go to sleep and not wake up?</i> | QSORRES for QSTESTCD=CSS0101 | <input type="checkbox"/> | <input type="checkbox"/> |
| If yes, describe: QSORRES for QSTESTCD=CSS0101A | | | |
| 2. Non-Specific Active Suicidal Thoughts General, non-specific thoughts of wanting to end one's life/commit suicide (e.g., "I've thought about killing myself") without thoughts of ways to kill oneself/associated methods, intent, or plan. <i>Have you actually had any thoughts of killing yourself?</i> | QSORRES for QSTESTCD=CSS0102 | <input type="checkbox"/> | <input type="checkbox"/> |
| If yes, describe: QSORRES for QSTESTCD=CSS0102A | | | |
| 3. Active Suicidal Ideation with Any Methods (Not Plan) without Intent to Act Subject endorses thoughts of suicide and has thought of at least one method during the assessment period. This is different than a specific plan with time, place or method details worked out (e.g., thought of method to kill self but not a specific plan). Includes person who would say, "I thought about taking an overdose but I never made a specific plan as to when, where or how I would actually do it...and I would never go through with it." <i>Have you been thinking about how you might do this?</i> | QSORRES for QSTESTCD=CSS0103 | <input type="checkbox"/> | <input type="checkbox"/> |
| If yes, describe: QSORRES for QSTESTCD=CSS0103A | | | |
| 4. Active Suicidal Ideation with Some Intent to Act, without Specific Plan | | | |

Table 1. C-SSRS aCRF

| Domain | Label | Sort |
|--------|------------------------------|---|
| QS | Questionnaire - QS | STUDYID, USUBJID, QSCAT, QSSCAT, QSTESTCD, VISITNUM |
| SUPPQS | Supplemental Qualifiers - QS | STUDYID, RDOMAIN, USUBJID, IDVAR, IDVARVAL, QNAM |

Table 2. SDTM Domain Level Metadata for QS and SUPPQS

| Domain | Variable | Label | Type | Length | Terms |
|--------|----------|--|----------|--------|----------|
| QS | STUDYID | Study Identifier | text | 17 | PHARMABC |
| QS | DOMAIN | Domain Abbreviation | text | 2 | QS |
| QS | USUBJID | Unique Subject Identifier | text | 25 | |
| QS | QSSEQ | Sequence Number | integer | 8 | |
| QS | QSSPID | Sponsor-Defined Identifier | text | 4 | |
| QS | QSTESTCD | Question Short Name | text | 8 | QSTESTCD |
| QS | QSTEST | Question Name | text | 40 | QSTEST |
| QS | QSCAT | Category of Question | text | 40 | QSCAT |
| QS | QSSCAT | Subcategory for Question | text | 40 | QSSCAT |
| QS | QSORRES | Finding in Original Units | text | 200 | |
| QS | QSSTRESC | Character Result/Finding in Std Format | text | 200 | |
| QS | QSSTRESN | Numeric Finding in Standard Units | integer | 8 | |
| QS | QSSTAT | Completion Status | text | 8 | ND |
| QS | QSREASND | Reason Not Performed | text | 200 | |
| QS | QSBFL | Baseline Flag | text | 2 | NY |
| QS | VISITNUM | Visit Number | integer | 8 | VISITNUM |
| QS | VISIT | Visit Name | text | 60 | VISIT |
| QS | QSDTC | Date/Time of Finding | datetime | 19 | ISO8601 |
| QS | QSDY | Study Day of Finding | integer | 8 | |
| SUPPQS | STUDYID | Study Identifier | text | 17 | |
| SUPPQS | RDOMAIN | Related Domain Abbreviation | text | 2 | PHARMABC |
| SUPPQS | USUBJID | Unique Subject Identifier | text | 25 | QS |
| SUPPQS | IDVAR | Identifying Variable | text | 8 | QSSEQ |
| SUPPQS | IDVARVAL | Identifying Variable Value | text | 40 | |
| SUPPQS | QNAM | Qualifier Variable Name | text | 8 | QNAM |
| SUPPQS | QLABEL | Qualifier Variable Label | text | 40 | QLABEL |
| SUPPQS | QVAL | Data Value | text | 200 | |
| SUPPQS | QORIG | Origin | text | 40 | QORIG |
| SUPPQS | QVAL | Evaluator | text | 200 | QVAL |

Table 3. SDTM Variable Level Metadata for QS and SUPPQS

| Domain | CAT | SCAT | TESTCD/ QNAM | TEST/ QLABEL | Orig / QORIG |
|--------|-------------------------|-------------------|-----------------|--|-----------------|
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0101 | CSS01-Wish to be Dead | CRF |
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0101A | CSS01-Wish to be Dead, Describe | CRF |
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0102 | CSS01-Non-Specific Suicidal Thought | CRF |
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0102A | CSS01-Non-Specific Suicid Thought, Descr | CRF |
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0103 | CSS01-Suicidal Ideation-No Intent | CRF |
| QS | C-SSRS BASELINE | SUICIDAL IDEATION | CSS0103A | CSS01-Suicidal Ideation-No Intent, Descr | CRF |
| QS | C-SSRS BASELINE | ... | ... | ... | CRF |
| QS | C-SSRS BASELINE | ACTUAL ATTEMPTS | CSS0123C | CSS01-First Attempt Potential | CRF |
| QS | C-SSRS SINCE LAST VISIT | SUICIDAL IDEATION | CSS0201A | CSS02-Wish to be Dead, Describe | CRF |
| QS | C-SSRS SINCE LAST VISIT | SUICIDAL IDEATION | CSS0202 | CSS02-Non-Specific Suicidal Thought | CRF |
| QS | C-SSRS SINCE LAST VISIT | SUICIDAL IDEATION | CSS0202A | CSS02-Non-Specific Suicid Thought, Descr | CRF |
| QS | C-SSRS SINCE LAST VISIT | SUICIDAL IDEATION | CSS0203 | CSS02-Suicidal Ideation-No Intent | CRF |
| QS | C-SSRS SINCE LAST VISIT | SUICIDAL IDEATION | CSS0203A | CSS02-Suicidal Ideation-No Intent, Descr | CRF |
| QS | C-SSRS SINCE LAST VISIT | ... | ... | ... | CRF |
| QS | C-SSRS SINCE LAST VISIT | ACTUAL ATTEMPTS | CSS0222C | CSS02-Most Lethal Attempt Potential | CRF |
| QS | ... | ... | ... | ... | CRF |
| SUPPQS | | | CSSRSAP | Assessment Period | Assigned |
| SUPPQS | | | CSSRSVER | C-SSRS Version | Assigned |

Table 4. SDTM Value Level Metadata for QS and SUPPQS

CDASH TRANSFORMATION

Naming convention and normalized structure are two features of CDASH to streamline SDTM domain mapping.

NAMING CONVENTION

Naming convention benefits developing reusable code. For example, if test dates regardless of CRFs were all stored in a pair of character and numeric datetime variables, i.e., `dts` and `dtn`, the mapping from CRF to SDTM would be simply `--dtc = dts` in ISO8601 format, and `--dy` derived as `%dy (invar=dtn, outvar=--dy)`, where `%dy` is a custom macro.

Without CDASH and when each CRF data field has a unique variable name, we can use RENAME statement to manipulate CRF data and apply naming convention. How extensively the CRF variables are renamed is up to the discretion of programmers. In the case of C-SSRS questionnaires, one may wish to have consistent variable names for same questions from different forms. To semi-automate the RENAME statement, a custom macro `%create_rename (B1)` is developed. It pulls information from SAS Dictionary (A1) and generates RENAME template that is one macro per CRF data set (C1a).

A1: Get CRF data set names from SAS dictionary. Set the text to a global macro variable.

```
proc sql noprint;
  select distinct memname "," into :rawnms separated by ","
  from dictionary.tables
  where libname="RAWDATA"
  order by memname;
quit;
```

B1: Execute `%create_rename`. Loop thru each CRF data set to generate RENAME statement template.

The left side of Output 1 shows the output file `macro_rename.sas` from `%create_rename`. It is composed of multiple macros, one per CRF data set, named `%rename_css1` and `%rename_css2` for example.

```

%macro create_rename;
  /** Loop thru each CRF data set **/
  %let m=1;
  %do %while (%scan("&rawnms",&m,",") ne );
    %let rawnm=%scan("&rawnms",&m,",");

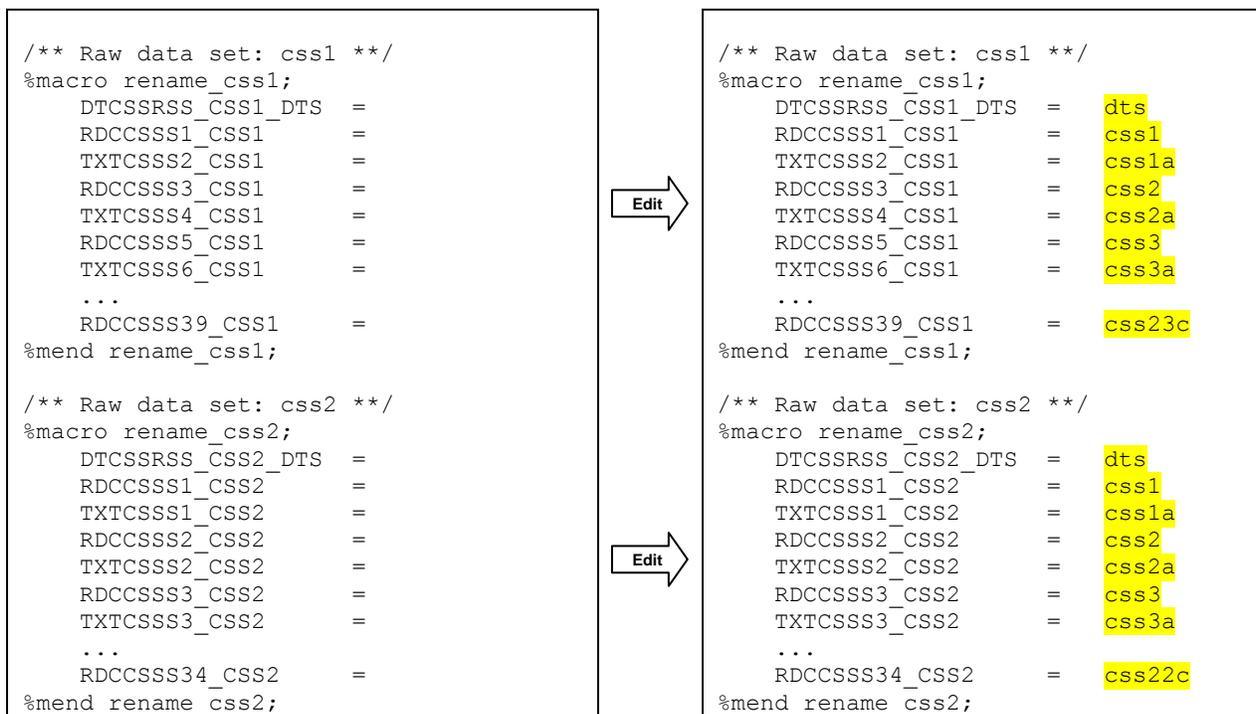
    /** Get variable names per CRF data set **/
    proc contents data=rawdata.&rawnm out=a(keep=name label);
    run;

    /** Output file macro_rename.sas **/
    filename outfile "macro_rename.sas";
    data _null_;
      file outfile notitles linesize=1000
      %if m ne 1 %then %do;
        mod
      %end; ;
      set a end=last;
      if _n_ = 1 then put
        /"/** CRF data set: &rawnm **/"
        /'%macro rename_' "&rawnm;";
      put "      " name $40. " = ";
      if last then put '%mend;';
    run;

    %let m=%eval(&m+1);
  %end;
%mend

```

C1a: The RENAME statement template is edited with new variable names. The right side of Output 1 shows an example. The text in yellow mark is edited new variable names. Note that the number part of the new variable names follow the overall sequence of the questions from C-SSRS questionnaires regardless of Baseline or Since Last Visit. This reduces the total number of CRF response variables for SDTM mapping to 40 from 73.



Output 1. Output macro_rename.sas, New Variable Names in Yellow Mark Added at Development Time

C1b: To incorporate the edited RENAME statement template to domain mapping program, CRF data sets are read in with SET statement and RENAME= option followed by %rename_css1 and %rename_css2. The following code is reusable except for editing the text in yellow mark.

```
data css;
  /** Assign variable length if multiple lengths **/
  /** length ...; **/
  set rawdata.css1(in=in1 rename=(%rename_css1))
    rawdata.css2(in=in2 rename=(%rename_css2));
  length memname $20;
  memname="";
  if in1 then memname="CSS1";
  else if in2 then memname="CSS2";
run;
```

Output data set `css` consists of 40 response variables instead of the original 73. Flag variable `memname` is added with value `css1` or `css2`, to tell the source CRF data set of each record. Caution: If response variable lengths from `css1` and `css2` are not all the same, use LENGTH statement to reset variable lengths and to prevent data truncation.

NORMALIZATION

Normalized data structure in clinical database is one record per response, and it is consistent with SDTM structure. For example, if all questionnaire response are stored on a variable named `orres`, the code to program QSORRES is just `QSORRES = ORRES`.

C1c: DATA step and ARRAY statement are used to transpose a data set from horizontal structure to vertical structure. The following code is a template. The text in yellow mark is edited at development time to plug in input data set name `css` and variable names `css1`, `css1a`, etc. The total number of elements in each array is 40, that is, the number of response variables mapped to QSORRES. Each record from `css` data set is transposed to 40 records, and the questionnaire responses in the output data set `css_norm` are stored in variable `orres`. The index variable `idx = 1 ~ 40` is to identify the input variables being transposed to create the output variable `orres`. When all the responses are blank, only one record is created with `idx = .`, `orres_nd = "NOT DONE"` and `orres = ""`. Output data set `css_norm` consists of the variables `subjectnumber` `visitindex` `memname` `idx` `orres` `orres_nd`, as shown in the inset of Output 1.

```
data css_norm(keep=subjectnumber visitindex memname idx orres orres_nd);
  set css;
  by subjectnumber visitindex memname;
  /** Initialize output variables **/
  length orres orres_nd $200 dummyc $1 idx 8;
  orres="";
  orres_nd="";
  dummyc="";
  idx=.;
  /** Output 1 record with orres_nd = NOT DONE **/
  /** when all responses are blank **/
  if css1="" and css1a="" and css2="" and ... and css23c="" then do;
    orres_nd='NOT DONE';
    output;
  end;
  /** Output 40 records when there is at least one non-blank response **/
  /** Normalize non-blank record **/
  else do;
    %array(** Output index variable idx=1 ~ 40 **/
      maxIdx=40,
      /** Define Array 1 **/
      /** Input elements = 40 response variables **/
      array1Columns=css1 css1a css2 ... css23c,
      /** Output variable = orres **/
      array1NewVar=orres)
  end;
run;
```



Variables in output data set

```
subjectnumber
visitindex
memname
idx
orres
orres_nd
```

Custom macro %array is developed to facilitate the data set transpose. The template described above shows the use of only one array, but this macro can process multiple arrays and output one new variable per array.

```

/** Custom macro to transpose horizontal data set to vertical structure */
%macro array(maxIdx=,
            array1NewVar=, array2NewVar=, ...,
            array1Columns=, array2Columns=, ...);

/** Define array1, array2, etc. */
%if (%quote(&array1Columns) ne) and (%quote(&maxIdx) ne) %then %do;
    array array1 ( &maxIdx ) &array1Columns ;
%end;

%if (%quote(&array2Columns) ne) and (%quote(&maxIdx) ne) %then %do;
    array array2 ( &maxIdx ) &array2Columns ;
%end;
...

/** Transpose each record to &maxIdx records */
do i = 1 to &maxIdx ;

    /** Populate output variables &array1NewVar, &array2NewVar, etc. */
    %if (%quote(&array1Columns) ne) %then %do;
        &array1NewVar = array1 (i);
    %end;

    %if (%quote(&array2Columns) ne) %then %do;
        &array2NewVar = array2 (i);
    %end;
    ...

    idx = i;
    output;
end;
%mend array;

```

CONTROLLED TERMINOLOGY

Controlled terms for value level metadata are usually saved in external files. Table 4 shows sample value level metadata for C-SSRS questionnaires (**A2**). It presents a nested structure. The topic variables --TESTCD / QNAM sit in the center, and each is associated with their synonym qualifier --TEST / QLABEL, grouping qualifiers --CAT and --SCAT for --TESTCD, and result qualifier QORIG for QNAM. For SDTM harmonization, the goal is to import the data to SAS and maximize auto-mapping the controlled terms to SDTM.

B2: Custom macro %create_formats is developed to generate PROC FORMAT template, as shown in Output 2.

C2: The PROC FORMAT template is almost complete except for lacking association with CRF data. This is fixed by editing the first VALUE statement, where each of --TESTCD value is linked to their corresponding questionnaire response in the normalized data set `css_norm` via the values on the index variable `idx`. The inset in Output 2, in yellow mark, shows edited VALUE statement.

```

proc format;

  value qscd
    /** C-SSRS BASELINE: IDX -> TESTCD **/
    = 'CSS0101'
    = 'CSS0101A'
    ...
    = 'CSS0123C'

    /** C-SSRS SINCE LAST VISIT: IDX -> TESTCD **/
    = 'CSS0201'
    = 'CSS0201A'
    ...
    = 'CSS0222C'
  ;

  value $qstest

    /** C-SSRS BASELINE: TESTCD -> TEST **/
    'CSS0101' = 'CSS01-Wish to be Dead'
    'CSS0101A' = 'CSS01-Wish to be Dead, Describe'
    ...
    'CSS0123C' = 'CSS01-First Attempt Potential'

    /** C-SSRS SINCE LAST VISIT: TESTCD -> TEST **/
    'CSS0201' = 'CSS01-Wish to be Dead'
    'CSS0201A' = 'CSS01-Wish to be Dead, Describe'
    ...
    'CSS0222C' = 'CSS02-Most Lethal Attempt Potential'
  ;

  value $qscat

    /** C-SSRS BASELINE: QSTESTCD -> QSCAT **/
    'CSS0101' = 'C-SSRS BASELINE'
    'CSS0101A' = 'C-SSRS BASELINE'
    ...
    'CSS0123C' = 'C-SSRS BASELINE'

    /** C-SSRS SINCE LAST VISIT: QSTESTCD -> QSCAT **/
    'CSS0201' = 'C-SSRS SINCE LAST VISIT'
    'CSS0201A' = 'C-SSRS SINCE LAST VISIT'
    ...
    'CSS0222C' = 'C-SSRS SINCE LAST VISIT'
  ;

  value $qsscatt

    /** C-SSRS BASELINE: QSTESTCD -> QSSCAT **/
    'CSS0101' = 'SUICIDAL IDEATION'
    'CSS0101A' = 'SUICIDAL IDEATION'
    ...
    'CSS0123C' = 'ACTUAL ATTEMPTS'

    /** C-SSRS SINCE LAST VISIT: QSTESTCD -> QSSCAT **/
    'CSS0201' = 'SUICIDAL IDEATION'
    'CSS0201A' = 'SUICIDAL IDEATION'
    ...
    'CSS0222C' = 'ACTUAL ATTEMPTS'
  ;

```

Edit

```

/** IDX -> TESTCD **/
value qscd1
/** css1 **/
1 = 'CSS0101'
2 = 'CSS0101A'
...
40 = 'CSS0123C'
. = 'QSALL'
;

value qscd2
/** css2 **/
1 = 'CSS0201'
2 = 'CSS0201A'
...
37 = 'CSS0222C'
. = 'QSALL'
;

```

Add 'QSALL' =
'Questionnaires Data'

```

value $qlabel
  /** ALL SUPP-- */
  'CSSRSAP' = 'Assessment Period'
  'CSSRSVER' = 'C-SSRS Version'
  ...
  ;

value $qorig
  /** ALL SUPP-- */
  'CSSRSAP' = 'Assigned'
  'CSSRSVER' = 'Assigned'
  ...
  ;

run;

```

Output 2. PROC FORMAT From %create_formats, Code in Yellow Mark Edited at Development Time

DOMAIN MAPPING

After CRF data sets are transformed to CDASH structure, and the controlled terms are organized as custom formats, the programming for domain mapping thru loading SDTM data is almost reusable with template programming and custom macros.

TEMPLATE PROGRAMMING

Template programming is made possible not only because of CDASH-like structured CRF data sets, but also by automatically assigning global macro variables (**B3**) with data from SDTM domain and variable metadata (**A3**).

B3: Output 3 shows global macro variables and their values coming from the metadata in Table 2 and Table 3 (**A3**).

```

/** Assign global macro variables per domain */
/** Macro variables for primary domain: &domainlabel, &sdtmvars, &sortkeys */
&domainlabel = Questionnaire - QS;
&sdtmvars = STUDYID DOMAIN USUBJID QSSEQ QSSPID QSTESTCD QSTEST QSCAT QSSCAT QSORRES
QSSTRESC QSSTRESN QSSTAT QSREASND QSBLFL VISITNUM VISIT QSDTC QSDY;
&sortkeys = STUDYID USUBJID QSCAT QSSCAT QSTESTCD VISITNUM;

/** Macro global variables for sup--: &supplabel, &suppvars, &qnams */
/** Nulled if SUPP-- is not applicable */
&supplabel = Supplemental Qualifiers - DS;
&suppvars = STUDYID RDOMAIN USUBJID IDVAR IDVARVAL QNAM QLABEL QVAL QORIG QEVAL;
&qnams = cssrsap cssrsver;

```

Output 3. SDTM Doman and Variable Metadata Assigned to Global Macro Variables

C3a: The following code shows a template to map QS domain. Only the text in yellow mark is edited. This template can be extended. For example, if standard result is collected on CRF, it can be added to the macro %array when normalizing CRF data sets `css1` and `css2`, and then incorporated into the template, as shown in the inset below.

```

/** Define global macro variable &domain */
%let domain = qs;

/** Merge common variables e.g. reference dates */
data all;
  merge css_norm(in=in1) rawdata.info(keep=subjectnumber rfstdtc);
  by subjectnumber;
  if in1;
run;

/** Edit domain mapping template */
data &domain ;

  /** Use shell data set to define data set and variable attributes */
  set libshell.&domain
    all;

  /** Set length $200 for SUPP QNAM source variables */
  length &qnam $200;

  /** Custom macro to map global variables STUDYID, DOMAIN, USUBJID */
  %header;

  /** Custom macro to map global variables VISIT and VISITNUM */
  %visit;

  /** Standard derivation next */
  /** %seq; */

  if memname in ("CSS1" "CSS2") then do;

    qsspid="";
    %num2char(invar=spid, outvar=qsspid);

    qstestcd="";
    if memname="CSS1" then qstestcd=put(idx,qscd1.);
    else if memname="CSS2" then qstestcd=put(idx,qscd2.);

    qstest="";
    if qstestcd ne "" then qstest=put(qstestcd,$qstest.);

    qscat="";
    if qstestcd ne "" then qscat=put(qstestcd,$qscat.);
    if qstestcd = "QSALL" then do;
      if memname = "CSS1" then QSCAT = "C-SSRS BASELINE";
      else if memname = "CSS2" then QSCAT = "C-SSRS SINCE LAST VISIT";
    end;

    qsscat="";
    if qstestcd ne "" then qsscat=put(qstestcd,$qsscat.);

    qsorres="";
    qsorres=strip(upcase(orres));

    qsstresc="";
    qsstresc=qsorres;
    /** Add additional mapping for QSSTRESC, if applicable */

    qsstresn=.;
    /** Verify if QSSTRESC is number only, then map to QSSTRESN */
    if %cknum(text=qsstresc) then qsstresn=input(qsstresc,best.);

```

/** If standard result
is collected on CRF,
map directly */
qsstresc=orres_c;

```

qsblfl="";
/** Standard derivation next **/
/** %blfl; **/
qsdtc="";
qsdtc=dts;

qsdy=.;
/** Standard derivation next **/
/** %dy; **/

qsstat="";
if orres_nd ne "" then do;
    qsstat="NOT DONE";
    qsreasnd="NOT DONE";
end;

/** SUPP QNAM source variables **/
cssrsap="";
if memname="CSS1" then cssassp="BASELINE";
else if memname="CSS2" then cssassp="SINCE LAST VISIT";

cssrsver="";
if memname="CSS1" then cssrsver="1/14/2009";
else if memname="CSS2" then cssrsver="1/14/2009";

/** Output valid records **/
if qsorres ne "" or qsstat ne "" then output;

/** End of if memname in ('CSS1' 'CSS2') then do; **/
end;
run;

```

Variable names `cssrsap` and `cssrsver`, same as QNAM, facilitate %supp (C3c).

STANDARD DERIVATION AND SUPPLEMENTAL QUALIFIER

The rest of the domain mapping program is static code and does not require editing. The following is an example.

C3b:

```

/** Derive --BLFL based on --DTC, --ORRES, --RFSTDTC and other applicable vars. **/
%blfl;

/** Derive --DY based on --DTC and RFSTDTC **/
%dy;

/** Sort data by &sortkeys and create --SEQ **/
%seq;

```

C3c:

```

/** Create SUPP-- data set **/
/** Comment out if no SUPP **/
%supp;

```

Take creating SUPP-- data set as an example. If the domain programming follows the steps described in the above sections, a custom macro %supp containing static code is all it takes to automate the mapping of SUPP-- data set. The following is partial code for %supp. It reads in the output data set &domain i.e. qs from the Template Programming section (C3a), where the values of QNAM are used as the source variable names for QVAL values. Normalize the data set via the TRANSPOSE procedure with VAR option = &qnams that is parsed to QNAM values (B3). The output data set is one-to-one record mapping to SUPPQS. Complete the mapping of controlled terms by using the custom format \$qlabel and \$qorig from Output 2 (C2).

```
%macro supp;
  proc transpose data=&domain out=supptrans;
    by studyid usubjid domain &domain.seq;
    var &qnams;
  run;

  data supp&domain (keep=&suppvars &domain.seq);
    set libshell.supp&domain
      supptrans;
    rdomain = domain;
    idvar = upcase(compress("&domain"||"SEQ"));
    idvarval = compress(put(&domain.seq,best.));
    qval = coll;
    qnam = upcase(compress(_name_));
    qlabel = put(qnam,$qlabel.);
    qorig = put(qnam,$qorig.);
    /** Keep valid SUPP records **/
    if qval ne "";
  run;
%mend;
```

LOAD SDTM

Final step is to load SDTM data sets. This is static code incorporating the macro variables from the Template Programming section (B3).

```
/** Load SDTM data **/
data &sdtmlib..%lowercase(&domain) (label="&domainlabel" keep=&sdtmvars);
  set &domain;
run;

data &sdtmlib..%lowercase(supp&domain) (label="&supplabel" keep=&suppvars);
  set supp&domain;
run;
```

CONCLUSION

This paper demonstrates a modularized approach to SDTM QS domain programming. It re-structures CRF database to imitate CDASH standards. Results are reusable code and structured programming, and reduced mapping errors.

REFERENCES

Clinical Data Interchange Standards Consortium, 2008. CDISC SDTM Implementation Guide, Version 3.1.2.

CDISC Controlled Terminology, 2012-12-21. Available at <http://www.cdisc.org/terminology>.

Columbia Suicide Severity Rating Scale annotated CRF. Available at <http://cdiscportal.digitalinfuzion.com/CDISC%20Therapeutic%20Area%20Documentation/Forms/AllItems.aspx>.

CONTACT INFORMATION

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

Name: Annie Guo
Enterprise: ICON Clinical Research
Address: San Francisco, CA, USA
Work Phone: 215-616-6597
E-mail: annie.guo@iconplc.com
Web: www.iconplc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.