PharmaSUG China 2022 – Paper AT-105

Real-time SDTM

Chunpeng Zhao, Mijun Hu, Jie Liu, Jiapeng He, Liming Xie, Aide Zhou and Bingting Tao, BeiGene

ABSTRACT

In SDTM transformation, there is time-consuming and much-repeated work from trial to trial and person to person. In response, we have turned to SDTM automation to streamline the work, create welcomed efficiencies, and make real-time SDTM possible.

INTRODUCTION

The rationality of real-time SDTM is that SDTM transformation program is not data driven. There is not data-driven work in SDTM transformation program, except handling rare data issues. To make real-time SDTM happen, we need to make SDTM transformation specifications and SDTM transformation code ready before First Patient First Visit. Then raw data can be transformed into SDTM immediately once raw data is available. The paper will introduce how to achieve the goal making everything ready before First Patient First Visit.

DIGITALIZATION & MODULARIZATION

The SDTM transformation program is not data-driven, but Case Report Form (CRF) driven. A SDTM aCRF can be automatically decomposed and modularized into CRF-field-digital-capsules (CRF-FDCs). A CRF-FDC is identified by CRF Form Name and Field OID (OID: Object Identifier). It also contains Field Label, Field Options, Page Number, Field Label's Coordinates on the page, and the field's SDTM annotations and their relative coordinates to the Field Label. The field's SDTM annotations are further decomposed automatically into SDTM Domain Name, SDTM Variable Names, and Transformation algorithms from CRF Fields to SDTM Variables.

All information of a field's SDTM annotation and SDTM transformation are modularized into a CRF-FDC. Using these CRF-FDCs, the CRF field's SDTM transformation logic automatically generates SAS code. See Chunpeng Zhao etc., 2021 for more details.

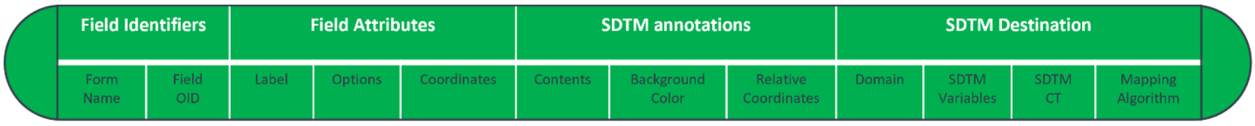

Figure 1. CRF-field-digital-capsule (CRF-FDC) Model

RECYCLING & INTEGRATION

All CRF-FDCs are uploaded to a shared cloud folder automatically when we build up CRF-FDCs in each study. A subsequent study CRF is mainly integrated by existing CRF fields which are defined in other various trials, with only limited additional, newly designed CRF fields added. For a subsequent study, only the newly designed CRF-FDCs need to be created.

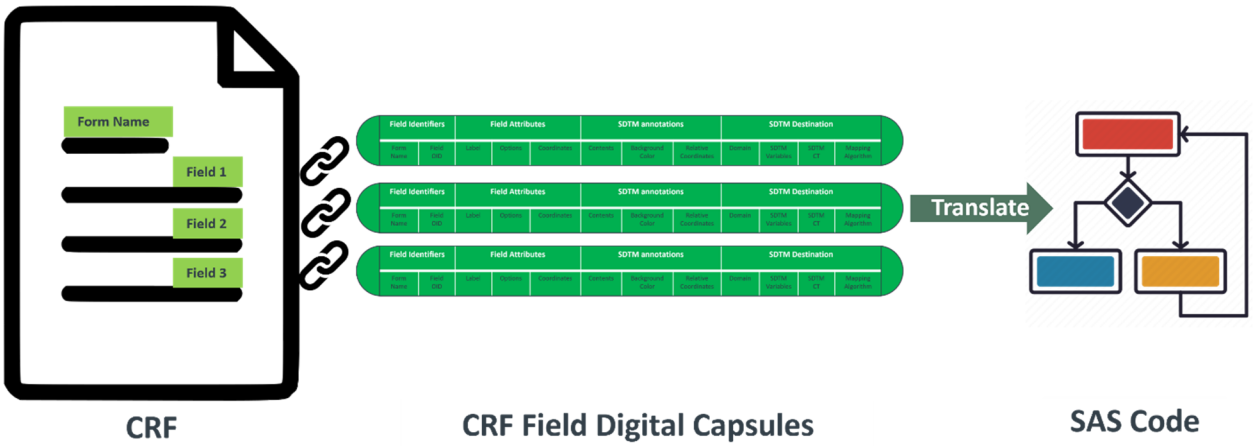

Figure 2. CRF-driven Coding-free SDTM Transformation Process

CENTRALIZATION & FRONT-LOADING

All studies' newly designed CRF-FDCs are created centrally by the SDTM Annotation Centralization Group at the CRF design stage. CRF data SDTM transformation specifications and SAS code are ready before the first patient enters the trial. Once the first CRF data record is available, it is immediately transformed into the SDTM.

Upon Data Transfer Specification (DTS) review, all studies' non-CRF (aka eDT) data SDTM transformation specifications are created by the SDTM DTS Review Centralization Group. The similar Digital Capsule with CRF-FDCs is also created and shared per Field or per Test in eDT.

Study dynamic customized algorithms of SDTM derived data are also pre-defined.

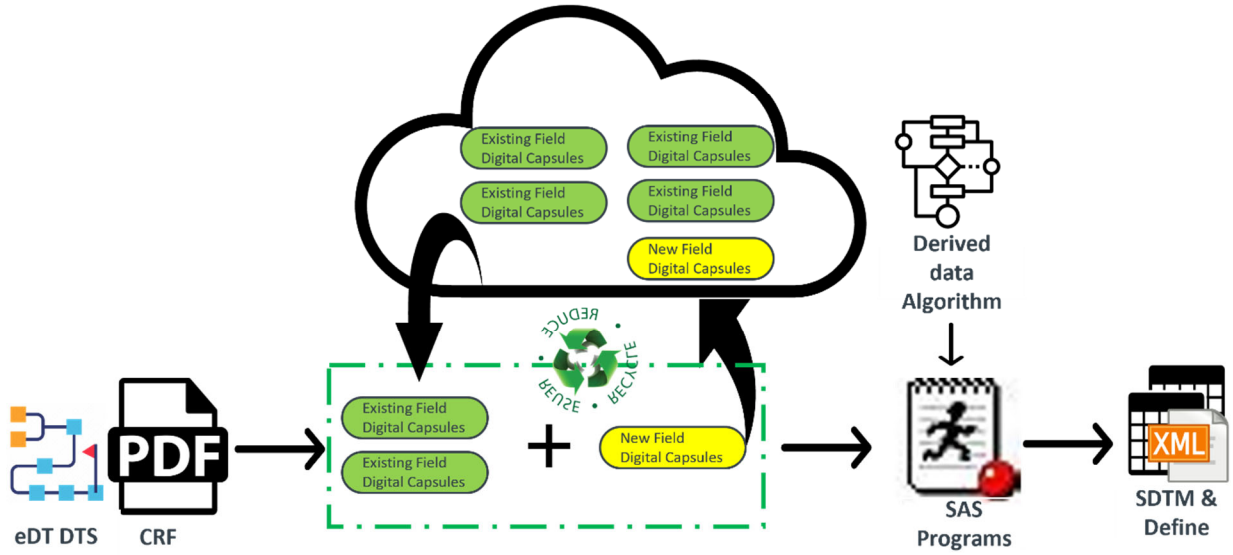

Figure 3. Flowchart of SDTM Transformation

META-PROGRAMMING

Meta-programming takes SAS code as data; it digitalizes and modularizes SAS code to make it machine-readable (i.e., to make SAS code itself recyclable and reproducible per CRF field). The direct outputs of meta-programming are not SDTM data sets, but metadata and SAS programs. CRF-FDCs are metadata in the SDTM. CRF-FDCs from various studies are automatically recycled and recombined based on a study CRF and converted to SAS code. Meta-programming creates common tools applicable to all studies. With the common tools, CRF-FDCs are copied and re-combined from other studies automatically based on current CRF design, and then converted to SAS programs that can create SDTM datasets. Meta-programming makes CRF collected data SDTM transformation coding-free.

META-PROGRAMMING EXAMPLE

Table 1 Sample of SDTM Metadata

CRFForm	FieldName	FieldLabel	SourceDataset	FieldValue	Dataset	VariableName	Type	VariableValue	Algorithm
Demographics	Form	Demographics	DM		DM	DOMAIN	Char	DM	Assigned
Demographics	BRTHDAT	Year of birth	DM		DM	BRTHDTC	Char		Copy
Demographics	AGE	Age	DM		DM	AGEU	Char	YEARS	Assigned
Demographics	AGE	Age	DM		DM	AGE	Num		Copy
Demographics	SEX	Sex	DM	Female	DM	SEX	Char	F	IfAssigned
Demographics	SEX	Sex	DM	Male	DM	SEX	Char	M	IfAssigned

```
/*Sample of Raw Dataset DM*/
data dm;
  infile datalines delimiter=',';
  input SUBJID $ BRTHDAT $ AGE $ SEX $;
  datalines;
001-001,1949, 73, Male
001-002,1963, 59, Female
001-003,1979, 43, Male
;
run;

/*Sample of SDTM Metadata. The content is as same as Table 1*/
data metadata;
       length CRFForm FieldLabel Algorithm $40;
       infile datalines delimiter=',';
  input CRFForm $ FieldName $ FieldLabel $ SourceDataset $ FieldValue $ Dataset $ VariableName $
Type $  VariableValue $ Algorithm $;
  datalines;
Demographics ,Form    ,Demographics   ,DM ,        ,DM ,DOMAIN  ,Char ,DM    ,Assigned
Demographics ,BRTHDAT ,Year of birth  ,DM ,        ,DM ,BRTHDTC ,Char ,      ,Copy
Demographics ,AGE     ,Age            ,DM ,        ,DM ,AGEU    ,Char ,YEARS ,Assigned
Demographics ,AGE     ,Age            ,DM ,        ,DM ,AGE     ,Num  ,      ,Copy
Demographics ,SEX     ,Sex            ,DM ,Female  ,DM ,SEX     ,Char ,F     ,IfAssigned
Demographics ,SEX     ,Sex            ,DM ,Male    ,DM ,SEX     ,Char ,M     ,IfAssigned
;
run;

/*Generate executable SAS Program based on SDTM Metadata */
data SASCode;
       length CodeString KeepVarLst $2000;
       retain KeepVarLst "SUBJID";
       set metadata end=LastObs;
       by FieldName notsorted;
       if _N_=1 then
              do;
                CodeString="data "||strip(Dataset)||";"; output;
                CodeString="  set "||strip(SourceDataset)||";"; output;
              end;

       if Algorithm="Assigned" then
              do;
                CodeString="  "||strip(VariableName)||"='"||strip(VariableValue)||"';"; output;
```

```
            end;
        else if Algorithm="Copy" and Type="Char" then
            do;
              CodeString="  "||strip(VariableName)||"="||strip(FieldName)||";"; output;
            end;
        else if Algorithm="Copy" and Type="Num" then
            do;
              CodeString="  "||strip(VariableName)||"=input(cats("||strip(FieldName)||"),
best.);"; output;
            end;
        else if Algorithm="IfAssigned" then
            do;
              if first.FieldName then
                do;
                  CodeString="  if "||strip(FieldName)||"='"||strip(FieldValue)||"' then";
output;
                end;
              else
                do;
                  CodeString="  else if "||strip(FieldName)||"='"||strip(FieldValue)||"'
then"; output;
                end;
              CodeString="    "||strip(VariableName)||"='"||strip(VariableValue)||"';";
output;
            end;

        KeepVarLst=tranwrd(strip(KeepVarLst), strip(VariableName), '')||" "||strip(VariableName);

        if LastObs then
          do;
            CodeString="  keep "||compbl(KeepVarLst)||";"; output;
            CodeString="run;"; output;
          end;
run;

data _null_;
  set SASCode;
  FILE "%sysfunc(pathname(work))/dm.sas" old;
  l=length(CodeString);
  put CodeString $varying2000.l;
run;

%include "%sysfunc(pathname(work))/dm.sas";
```

Table 2 Sample of Output DM Domain

DOMAIN	SUBJID	BRTHDTC	AGE	AGEU	SEX
DM	001-001	1949	73	YEARS	M
DM	001-002	1963	59	YEARS	F
DM	001-003	1979	43	YEARS	M

From above example, we can see there is nothing special on SAS code. Meta-programming is very related with mindset, but not with coding itself. There are two parts on meta-programming, metadata, and programming to create an executable program based on metadata.

The key to meta-programming is metadata. Metadata is a universal data model, which is applicable to all Case Report Forms and their corresponding SDTM Domains. Metadata includes all information how to convent CRF collected data to SDTM. And metadata is machine-readable. Machine-readable metadata have two meanings. First, metadata can be recycled and reused by coding. Second, executable SAS programs (e.g., dm.sas in above example) can be generated based on metadata through programming instead of manually, since all information which executable SAS programs (e.g., dm.sas in above example) need, can be machine-readable in metadata.

From metadata to executable SAS programs, it can be taken as a compilation process. In SDTM transformation, only one compiler is enough, if metadata model is stable. With the complier, SDTM

transformation will be coding-free. It means we create executable SAS programs on metadata, instead of typing SAS code directly by hands. There are three levels of programming philosophy. Creating programs by programming, i.e., writing programs to handle text streams, is the third level, because that is a universal interface (Erik van Baaren, The Unix philosophy).

CONCLUSION

SDTM transformation program is not data driven. With Digitalization & Modularization, Recycling & Integration, Centralization & Front-Loading and Meta-programming, we make SDTM transformation specifications and SDTM transformation code ready before First Patient First Visit. Once the first CRF data record is available, it is immediately transformed into the SDTM. Real-time SDTM can support real-time data monitoring and data cleaning.

REFERENCES

Chunpeng Zhao etc., 2021. “SDTM Annotated CRF Digitalization” https://www.lexjansen.com/pharmasug-cn/2021/DS/Pharmasug-China-2021-DS036.pdf

Chunpeng Zhao etc. 2021. “CRF Chinese Translation – Mask Plan” https://www.lexjansen.com/pharmasug-cn/2021/AD/Pharmasug-China-2021-AD035.pdf

Erik van Baaren, The Unix philosophy. “The Unix philosophy” https://python.land/the-unix-shell

CONTACT INFORMATION

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

Chunpeng Zhao
BeiGene
Chunpeng.zhao@beigene.com
Mijun Hu
BeiGene
mijun.hu@beigene.com
Jie Liu
BeiGene
jie1.liu@beigene.com
Jiapeng He
BeiGene
jiapeng.he@beigene.com
Aide Zhou
BeiGene
aide.zhou@beigene.com
Liming Xie
BeiGene
liming.xie@beigene.com
Bingting Tao
BeiGene
bingting.tao@beigene.com