PharmaSUG2011 - Paper CD02

Automating the Process of Preparing Data Definition Document for NDA Electronic Submission from Programming Specification in Word Format

Min Chen, Vertex Pharmaceuticals, Cambridge, MA
Xiangchen (Bob) Cui, Vertex Pharmaceuticals, Cambridge, MA
Scott Moseley, Vertex Pharmaceuticals, Cambridge, MA

ABSTRACT

When submitting clinical study data in electronic format to the FDA, Case Report Tabulations Data Definitions in an XML format (define.xml) are required for both CDISC SDTM and ADaM to provide the domain and variable metadata in order to improve the efficiency of the Regulatory Review process. The programming specification documentation is the key to prepare define.xml, for it provides most of, if not all, the information in define.xml. In pharmaceuticals industry, the programming specification is often created in Microsoft® Word format due to its popularity. Automating the process of preparing define.xml from Word specification is of great help to reduce the tedious manual work and achieve the high quality of the submission.

This paper introduces an effective way of automating define.xml preparation process from Word document. The key part is to read Word document to SAS®, and then filter the useful information from the programming specification. It was successfully used in our two Phase II and III studies for NDA submission.

INTRODUCTION

Word documentation is widely used in Pharmaceuticals industry. Although the programming specification created in Word format is intuitive, flexible, and easy to handle, it is not as easy to be applied in other process such as NDA Electronic Submission since SAS cannot directly import data from the Word documents. The manual work for preparing define.xml is time-consuming, labor-intensive and error-prone, and the consistency checking among SDTM/ADaM datasets, programming specification, and define.xml adds more workload to the process. It is highly desirable for SAS programmers to programmatically retrieve the information in the specification in MS Word format and convert it into a SAS dataset and/or Excel® files. The automation significantly reduces the time and resources for consistency checking, define.xml preparation, and frees the programmers from cumbersome manual work.

SAS programmers have already explored the methods to import data from Word/RTF to SAS without any manual pre-process. Gupta et al. (2010) developed a macro to convert the word readable RTF documents into SAS datasets by predefining variable formats in the Microsoft Access table. Zhou (2009) programmatically imported a Word document into SAS using Excel via DDE (Dynamic Data Exchange).

Inspired by these ideas, this paper describes a method which applies a revised DDE solution in a macro %readword to append the whole programming specification in Word Format to a Excel® file predefining the column length, automatically retrieves the useful information based on the format of the programming specification by a macro %retrieve, and creates Excel® files for the consistency checking and define.xml preparation by a macro %output.

This method guarantees the consistency between SDTM/ADaM datasets and the programming specification by consistency checking, and guarantees the consistency between the programming specification and define.xml in that define.xml is created from programming specification. It was successfully used in our two Phase II and III studies for NDA submission of ADaM. It can be applied to SDTM if only Word Format specification is available.

Figure 1 shows the overview of the process flow.

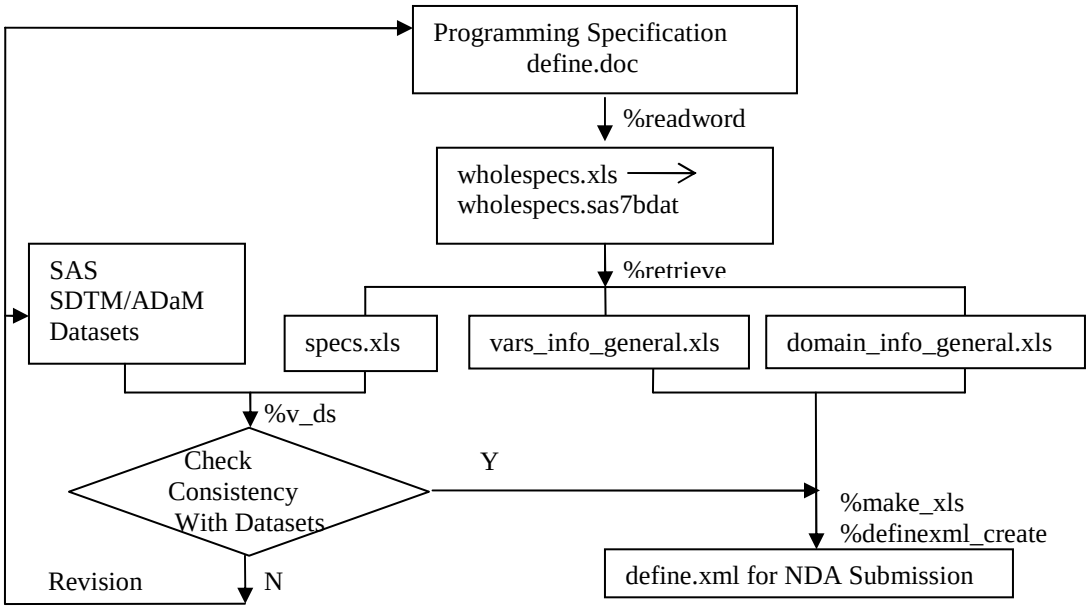
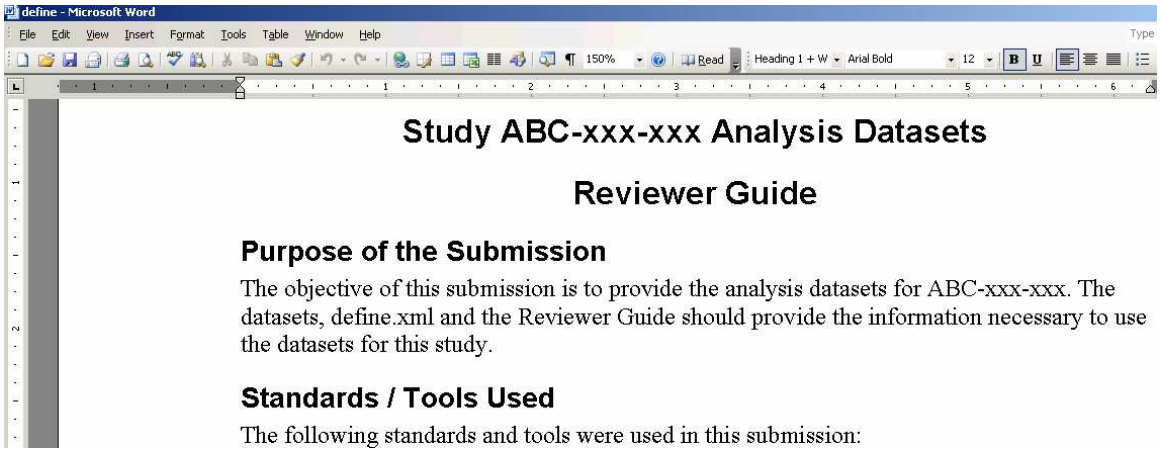


Figure 1 Overview of Process Flow

CONVERT PROGRAMMING SPECIFICATION IN WORD® TO SAS® DATASET

AN INTRODUCTION OF WORD® PROGRAMMING SPECIFICATION AND EXCEL® FILE TO PRE-DEFINE THE COLUMN LENGTH

The programming specification in word® is served as reviewer guide in NDA submission, as shown in Fig. 2(a). Dataset specification, which is needed for consistency checking and define.xml preparation, is in the Section 4, as shown in Figure 2(b). The specification for each domain is composed of three parts: dataset description table, variable information table, and an optional appendix or notes for a complex algorithm or derivation rules. The first two parts will be filtered to prepare define.xml. The common variables across the datasets are recorded before all other variables in the variable information table.



(a) Reviewer Guide

4.1 Analysis Datasets Description

4.1.1 ADAE: Adverse Events

domain_info_general.xls

vars_info_general.xls

Dataset	ADAE
Program Name	adae.sas
Description	Adverse Events
Unique identifier Variables	studyid usubjid aedecod aestdtc
Sort Variables	usubjid aeseq aedecod aestdt
Structure	One record per adverse event per subject per intensity
Input Datasets	ADRAND, ADSL, ADTERM, ADVISW, DM, SUPPAE, AE
Notes	Each subject has at least one record. A subject without AE has one record with AEANY = N.

Map to Domain

Map to Description

Map to Keys

Map to Structure

Variable Name	Label	Type	Controlled Terminology	SDTM/ADaM Origin	Comments
Common variables					Subset of variables from ADSL merged by usubjid.
USUBJID	Unique Subject Identifier	Char 40		DM.usubjid	Unique subject identifier within the submission. (e.g. ABC-xxx-xxx-1001-1001004) Equals DM.usubjid
AESEQ	Sequence number	Num 8		AE.aeseq	A unique sequential number within the same USUBJID. Equals AE.aeseq
AESPID	Sponsor-Defined Identifier	Char 8		AE.aespid	Concatenation of CRF Page number and AE form number. Equals AE.aespid
AEORGNO	CRF ID Number of the Original AE	Char 8		AE	AE.aeorgno
AEANY	Any Adverse Events	Char 1	Y N	Derived	If subject in DM but not in AE, then AEANY='N'. otherwise AEANY='Y'.
AETERM	Reported Term for the Adverse Event	Char 200		AE.aeterm	Verbatim name of the event. Equals AE.aeterm

Notes:

1. The imputation rules for AESTDT and AEENDT:
   a. Initial imputation for AESTDT:

Not Needed for Neither Consistency Checking Nor define.xml Preparation

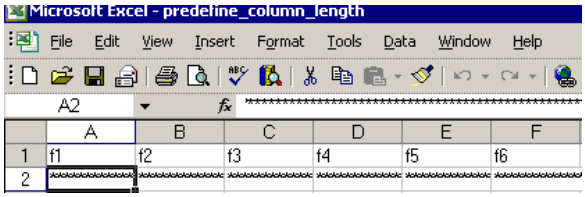
(b) Section 4 – Dataset Specification, the Information Needed

Figure 2 Programming Specifications in Word® Format

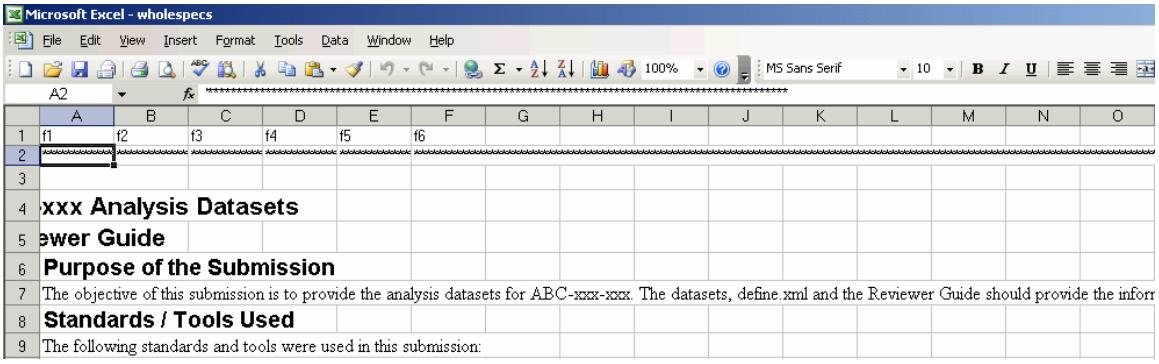
THE MACRO TO COPY WORD SPECIFICATION TO EXCEL FILE AND IMPORT IT TO SAS

The macro %readword will read the whole programming specification and convert it to SAS datasets. Since Column "comments" is very long, a lot of truncations occur during the import process. To avoid the truncation, an excel file with pre-defined length of the columns is generated before invoking %readword, as shown in Figure 3(a). The contents of define.doc will be appended into the corresponding columns of the pre-existing spreadsheet.

The final programming specification in Excel Format is shown in Figure 3(b).



(a) Excel File which Pre-Defines the Length of Each Column



	A	B	C	D	E	F
1	**Variable Name**	**Label**	**Type**	**Controlled Terminology**	**SDTM/ADaM Origin**	**Comments**
2						
11	AEORGNO	CRF ID Number of the Original AE	Char 8		AE	AE.aeorgno
12	AEANY	Any Adverse Events	Char 1	Y	Derived	If subject in DM but not in AE, then AEANY='N'. otherwise AEANY='Y'.
13				N		
14	AETERM	Reported Term for the Adverse Event	Char 200		AE.aeterm	Verbatim name of the event.
15						Equals AE.aeterm
37	AESEV	Severity/Intensity	Char 20	MILD	AE.aesev	AE Severity/Intensity
38				MODERATE		AE.aesev
39				SEVERE		

(b) Programming Specification in Excel Format with Pre-Defined Length for Each Column

Figure 3 Programming Specification in Excel Format

The codes of %readword are shown as follows:

```
%macro readword(indir=, outdir=, datin=, datout=);
options noxwait noxsync;
*** Copy all the contents from Word File ***;
%let rc=%sysfunc(system(start winword));
filename word DDE 'Winword|System';
%let fid=%sysfunc(fopen(word,S));
%if &fid eq 0 %then %do;
    options noxwait noxsync;
    x ' "D:\Program Files\Microsoft Office\OFFICE11\winword.exe"';
    data _null_;
         rc=sleep(6);
    run;
%end;

data _null_;
     file word;
     put '[FileOpen.Name="' "&indir.\&datin" '"]'; rc=sleep(20);
     put "[EditSelectAll]";
     put "[EditCopy]"; rc=sleep(10);
     put '[FileClose]';
     put '[AppMinimize]';
run;
data _null_; rc=sleep(1); run;

*** Append the contents from Word File to the end of Excel File with pre-defined column
length ***;
%let rc1=%sysfunc(system(start excel));
filename excel DDE 'Excel|System';
%let fid1=%sysfunc(fopen(excelchk,S));
%if &fid1 eq 0 %then %do;
    options noxwait noxsync;
    x ' "D:\Program Files\Microsoft Office\OFFICE11\excel.exe"';
    data _null_;
         rc1=sleep(10);
    run;
%end;

%let outfile=%trim(&outdir)\&datout.;
%put &outfile;
data _null_;
```

```
     file excel;
     put '[error(false)]';
     put '[Open(""&outdir."'\'"&excel_length."'")]'; rc=sleep(20);
     put '[select("r3c1")]';rc=sleep(10);
     put '[Paste]';
     put '[SAVE.AS("&outfile")]'; rc=sleep(30);
     put "[SAVE.AS(""&outfile"")]";
     put '[Select.Last.Cell()]';
     put '[Copy]';
     put '[quit]';
run;
data _null_;rc1=sleep(20);run;

%let rc2=%sysfunc(fclose(&rc1));
%let rc3=%sysfunc(fclose(&rc));

libname xlsin excel "&outdir.\&datout."
        mixed=yes scantext=no dbmax_text=32000 getnames=no;/* Row 1-32,767 */

data work.wholespecs;
  set xlsin."sheet1$"n
 (dbSasType=(
  f1   = char100
  f2   = char1000
  f3   = char100
  f4   = char1000
  f5   = char100
  f6   = char4000)
  );
  retain record 0;
  if missing(f1) and missing(f2) and missing(f3) and missing(f4) and
     missing(f5) and missing(f6) then delete;
  record + 1;
  format _character_;
run;
%mend;
%readword(indir = E:\esub\convert\define\analysis
        ,outdir = E:\esub\convert\define\analysis
        ,datin  = define.doc
        ,datout = wholespecs.xls);
```

In the macro %readword, **PUT statements** will send **WordBasic command** to Word, or send the **X4ML (Excel version 4 Macro Language) command** to Excel via DDE technique. WordBasic commands **FileOpen**, **EditSelectAll**, and **EditCopy** will open a Word document, select all the contents, and make a copy for it. X4ML command **error(false)** switches off the error-checking to keep Excel from displaying any message-boxes, and commands **open**, **select**, **paste**, and **save.as** will open an Excel document, move the cursor to the selected cell(s), paste the contents from clipboard to selected cell(s), and save the new Excel document as the customer's Excel file. Please refer to WordBasic Reference (Microsoft Press, 1995) and Excel 4.0 Macro Reference for further information about these commands.

THE MACRO TO RETRIEVE THE USEFUL INFORMATION FROM SPECIFICATION FOR CONSISTENCY CHECK AND DEFINE.XML PREPARATION

INTRODUCTION OF THE SPREADSHEETS NEEDED FOR CONSISTENCY CHECK

%v_ds, a macro developed in-house, will do the consistency checks required by Vertex for delivery qualities. The macro compares the programming specification and the SDTM/ADaM datasets, checks the availability of the individual dataset, individual programming specification, and each variable, and checks the consistency and the validity of the variable attributes. The programming specification read by the macro %v_ds is an Excel Workbook with multi-sheets as shown in Figure 4. Each sheet contains one domain.

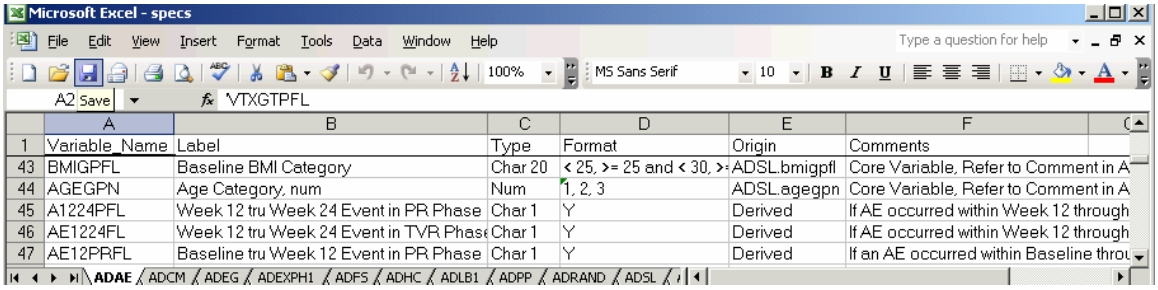
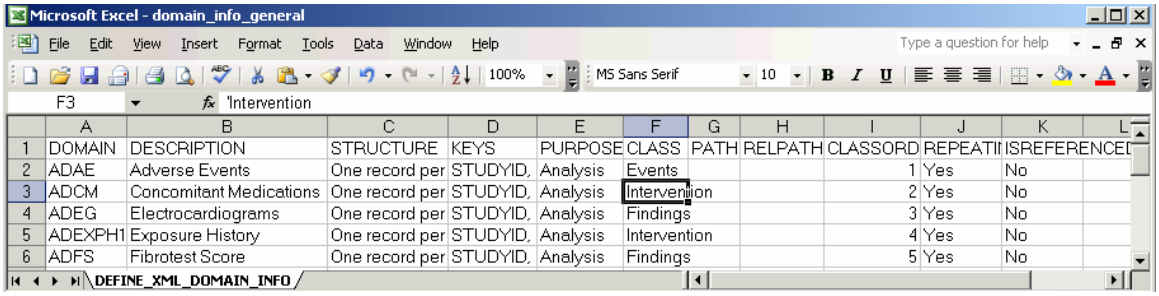


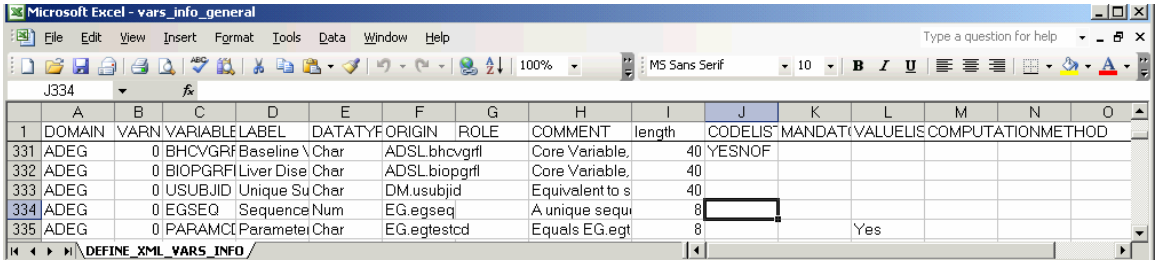
Figure 4 Dataset Specification in EXCEL Format for Consistency Checking

INTRODUCTION OF THE SPREADSHEETS NEEDED FOR DEFINE.XML PREPARATION

Metadata file define.xml will be created by invoking macros %make_xls and %definexml_create, developed in-house. These macros need information including Study Level Spreadsheet, Domain Level Spreadsheet (Dataset Metadata), Variable Level Spreadsheet (Variable Metadata), Value Level Spreadsheet (Value Level Metadata), Computational Algorithm Spreadsheet, and Controlled Terminology/Format Spreadsheet, in which Domain Level Spreadsheet and Variable Level Spreadsheet are needed for creating define.xml from dataset specification. They are shown in Fig 5.



(a) Domain Level Spreadsheet (Dataset Metadata, domain_info_general.xls)



(b) Variable Level Spreadsheet (Variable Metadata, vars_info_general)

Figure 5 Spreadsheets Needed for Define.xml Preparation

RETRIEVE USEFUL INFORMATION BASED ON THE FORMAT OF THE INDIVIDUAL DOMAIN SPECIFICATION

The macro %retrieve is used to retrieve the domain information and variable information from the whole programming specification based on the format of the individual domain specification.

As shown in Figure 2, each domain specification begins with the section title, which includes the numbering 4.1.x, domain name, a colon sign, and the description of the domain. Domain information table goes after the title, beginning with 'Dataset' Row and ending with 'Notes' Row. Variable information table trails domain information table, beginning with 'Variable name'. After all these information, some domains may have detailed programming notes beginning with 'Notes:' or 'Appendix'.

Considering the above formats, two flags are generated for retrieving the useful information. Flag1 labels the domain information, and flag2 indicates all the records to be output, that is, domain information plus variable information.

The codes are listed as follows:

```
%macro retrieve(indir=,datin=);
data domain vars;
     set &indir..&datin.;
     retain flag1 flag2 0;
     if strip(upcase(f1)) = 'DATASET' then flag1 = 1;
     if strip(upcase(f1)) = 'VARIABLE NAME' then flag1 = 0;
     if strip(upcase(f1)) =: '4.1.' and missing(f2) then do;
```

```
         flag1 = 0;
         flag2 = 1;
      end;
      if strip(upcase(f1)) = 'NOTES:' or strip(upcase(f1)) =: 'APPENDIX' then
         flag2 = 0;
      if strip(upcase(f1)) =: '4.1.' and missing(f2) then do;
         if not missing(scan(compress(f1,' 0123456789.'),1,':')) then do;
            if index(f1,'PCOP') then f1 = 'PCOP';
            else f1 = scan(substr(f1,find(f1,'A')),1,':');
         end;
      end;
      if (flag1 and flag2) then output domain;
      if flag2 and not flag1 and strip(upcase(f1)) ne 'VARIABLE NAME' and
         strip(upcase(f5)) ne 'ORIGIN' then output vars;
      keep f1 - f6 flag1 flag2;
run;
%mend;
%retrieve(indir=work,datin=wholespecs);
```

TRICKS FOR OUTPUT

A macro %output is called to output all the needed spreadsheets for consistency checking and define.xml preparation. Some tips to be mentioned are shown as the follows:

1. The FDA guidance on electronic submissions requires that an analysis data set to be submitted in the form of a SAS transport file that may not exceed 400 MB in size. Datasets that exceed the FDA size limitation will be split into several subsets. If ADLB is split, the first split dataset ADLB1 will be shown in define.xml with the link of dataset ADLB1.xpt. The content of the Domain Column will be replaced with ADLB1, in the domain information and variable information spreadsheets.
2. Columns ‘Controlled Terminology’, ‘SDTM/ADaM’, and ‘Comments’ in dataset specification may contain several paragraphs with hard return sign in Word document. When read into the Excel file, each paragraph will occupy one row, as shown in Figure 3. The first paragraph for a variable is considered to be the row where Columns ‘Variable Name’, ‘Label’, and ‘Type’ are filled, and the subsequent paragraphs are the rows where these columns are blank. These paragraphs need to be combined together for the completed information.
3. Some characters, such as apostrophe sign and quote sign, in Word document will be read as special characters when they are converted into a SAS dataset. They need to be replaced by proper characters, as shown in the following codes.
4. Core Variables across the ADaM datasets are recorded in the first row of the variable information table of the dataset specification for readability. In define.xml, however, they need to be shown separately as single variables. The macro %output will retrieve the attributes and other information of these core variables in subject level dataset ADSL, and apply it to relative datasets.

The codes for output are shown as follows:

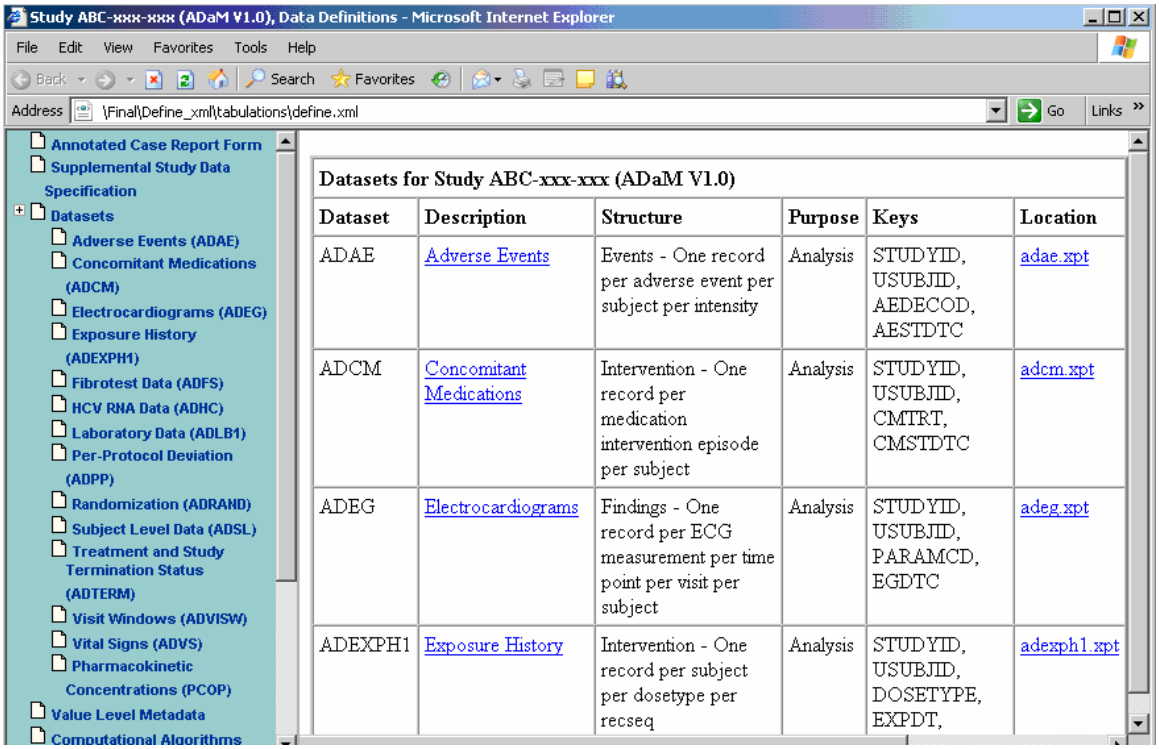
```
%macro output(outdir=,xls_domain=,xls_vars=,xls_specs=);
data domain;
     length DOMAIN $8 DESCRIPTION $40 STRUCTURE KEYS $200 PURPOSE $10
            CLASS PATH RELPATH $40 CLASSORD 8 REPEATING ISREFERENCEDATA $3;
      set domain;
      retain domain description structure keys purpose class path relpath
             repeating isreferencedata;
      retain classord 0;
      if propcase(f1) = 'Dataset' then classord + 1;
      if propcase(f1) = 'Dataset' then domain = strip(upcase(f2));
      if propcase(f1) = 'Description' then description = strip(f2);
      if propcase(f1) = 'Unique Identifier Variables' then
         keys = strip(upcase(f2));
      purpose = 'Analysis';
      if domain in ('ADHC','ADLB','ADFS','ADVS','ADEG','ADPP',
         'ADVISW','ADTERM','PCOP') then class = 'Findings';
      else if domain in ('ADAE','ADMH') then class = 'Events';
      else if domain in ('ADSL','ADRAND') then class = 'Special Purpose';
      else if domain in ('ADCM','ADEXPH') then class = 'Intervention';
      path = ' ';
      relpath = ' ';
      if propcase(f1) = 'Structure' then structure = strip(f2);
```

```
      if strip(lowcase(structure)) in ('one record per subject',
         'one record per subject randomized') then repeating = 'No';
      else repeating = 'Yes';
      isreferencedata = 'No';
      *** Since ADLB and ADEXPH are splitted into multiple datasets, ***;
      *** we use ADLB1 and ADEXPH1 in define.xml                      ***;
      if domain in ('ADLB','ADEXPH') then domain = strip(domain) || '1';
      *** Output the last record ***;
      if propcase(f1) = 'Notes';
      keep DOMAIN DESCRIPTION STRUCTURE KEYS PURPOSE CLASS PATH RELPATH
           CLASSORD REPEATING ISREFERENCEDATA;
run;

PROC EXPORT DATA= WORK.domain OUTFILE= "&outdir.\&xls_domain."
     DBMS=EXCEL REPLACE;
         SHEET="DEFINE_XML_DOMAIN_INFO";
RUN;

data specs;
     length domain $8 variable_name $16;
     set vars;
     retain domain ' ' variable_name ' ';
     if not missing(f1) and missing(f2) and missing(f3) and missing(f4) and
        missing(f5) and missing(f6) then do;
        domain = strip(upcase(f1));
        delete;
     end;
     if not missing(f1) then variable_name = strip(upcase(f1));
     *** Since ADLB and ADEXPH are splitted into multiple datasets, ***;
     *** we use ADLB1 and ADEXPH1 in define.xml                      ***;
      if domain in ('ADLB','ADEXPH') then domain = strip(domain) || '1';
run;

proc sort data = specs;
     by domain variable_name record;
run;

data specs;
     length Domain $8 Variable_Name $16 Label $50 Type $10
     Format $1000 Origin $100 Comments $4000;
     set specs(rename=(controlled_terminology=Format
                       SDTM_ADAM_Origin=Origin));
     by domain variable_name record;
     retain label type format Origin comments ' ';
     if first.variable_name then do;
        label = strip(f2);
        type = strip(f3);
        format = strip(f4);
        Origin = strip(f5);
        comments = strip(f6);
     end;
     else do;
        *** Combine multiple paragraphs for one variable ***;
        if missing(Format) and missing(f4) then ;
        else if missing(Format) and not missing(f4) then Format = strip(f4);
        else if substr(strip(reverse(Format)),1,1) in (',','.',';') and
             not missing(f4) then Format = strip(Format) || ' ' || strip(f4);
        else if not missing(f4) then
             Format = strip(Format) || ', ' || strip(f4);

        if missing(Origin) and missing(f5) then ;
        else if missing(Origin) and not missing(f5) then Origin = strip(f5);
        else if substr(strip(reverse(Origin)),1,1) in (',','.',';') and
             not missing(f5) then Origin = strip(Origin) || ' ' || strip(f5);
        else if not missing(f5) then
             Origin = strip(Origin) || ', ' || strip(f5);
```

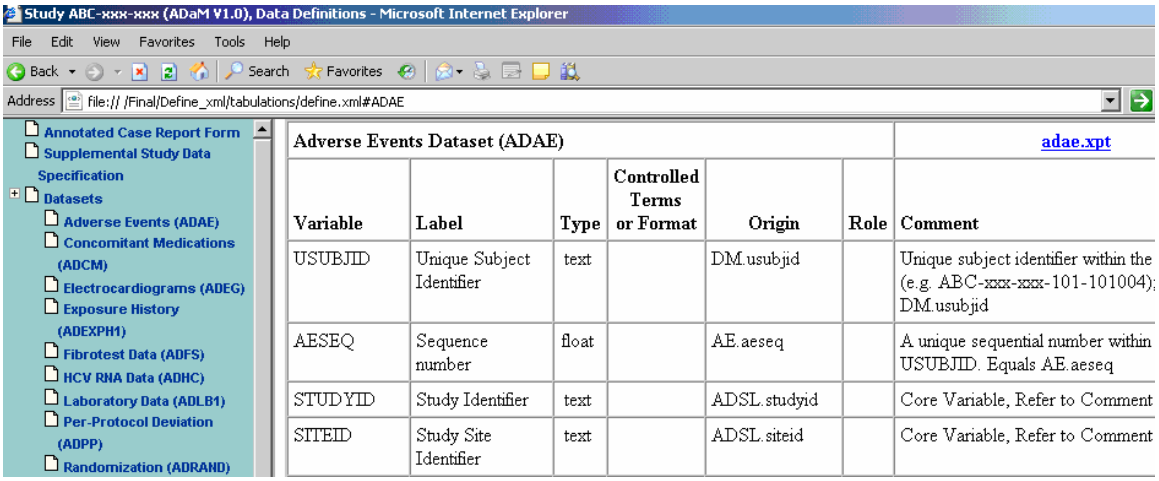
```
        if missing(comments) and missing(f6) then ;
        else if missing(comments) and not missing(f6) then
             comments = strip(f6);
        else if substr(strip(reverse(comments)),1,1) in (',','.',';',':','?')
             and not missing(f6) then
             comments = strip(comments) || ' ' || strip(f6);
        else if upcase(scan(strip(comments),-1)) = 'AND' and not missing(f6)
             then comments = strip(comments) || ' ' || strip(f6);
        else if upcase(scan(strip(comments),-1)) in ('EQUAL','EQUALS') and
             not missing(f6) then
             comments = strip(comments) || ', ' || strip(f6);
        else if not missing(f6) then
             comments = strip(comments) || '; ' || strip(f6);
     end;
     if last.variable_name;
     *** Replace the special characters read from Word ***;
     if last.variable_name then comments =
        translate(comments,"'","’","'","’",'"','”','"','”','"','”','-','–');
run;

*** Use the attributes in ADSL to retrieve attributes of common variables;
data commonvars(keep=Variable_Name Label Type Format Origin Comments);
     set specs(where=(domain='ADSL'));
     if upcase(variable_name) in (&corevars.);
     Comments = 'Core Variable, Refer to Comment in ADSL';
     Origin = 'ADSL.' || lowcase(strip(variable_name));
run;

proc sql ;
     create table vars2 as
       select a.*,b.domain
       from commonvars as a
       left join vars as b on 1
       where strip(propcase(b.variable_name)) = 'Common Variables'
       order by domain;
quit;

data specs;
     set vars2 specs(where=(propcase(variable_name) ne 'Common Variables'));
     by domain;
run;

proc sort data = specs out = domainame nodupkey;
     by domain;
run;

data _null_;
     set domainame end=eof;
     call symput('sheet' || strip(put(_n_,best.)),strip(domain));
     if eof then call symput('nsheet',strip(put(_n_,best.)));
run;
%let nsheet = &nsheet.;

%do i = 1 %to &nsheet.;
    data &&sheet&i.(drop=domain);
      set specs(keep=Domain Variable_Name Label Type Format Origin Comments);
      if domain = "&&sheet&i.";
    run;

    PROC EXPORT DATA= WORK.&&sheet&i. OUTFILE= "&outdir.\&xls_specs."
         DBMS=EXCEL REPLACE;
         SHEET="&&sheet&i.";
    RUN;
%end;

proc sort data = specs out = vars;
     by domain record;
```

```
run;

data vars(keep=DOMAIN VARNUM VARIABLE LABEL DATATYPE ORIGIN ROLE COMMENT
               length CODELIST MANDATORY VALUELIST COMPUTATIONMETHOD);
     length DOMAIN $8 VARNUM 8 VARIABLE $16 LABEL $40 DATATYPE $10 ORIGIN $100 ROLE $100
COMMENT $4000 length 8 CODELIST $100 MANDATORY VALUELIST COMPUTATIONMETHOD $10;
     set vars;
     retain varnum 0;
     variable = variable_name;
     datatype = scan(type,1);
     length = input(strip(scan(type,2)),??best.);
     if missing(length) and strip(upcase(datatype)) = 'NUM' then length = 8;
     role = ' ';
     *** Generate codelist, valuelist, and computation method automatically;
     if strip(upcase(variable)) = 'SEX' then codelist = 'SEX';
     if strip(upcase(variable)) = 'VMEDDRA' then codelist = 'MedDRA';
     if strip(upcase(variable)) = 'VWHODRUG' then codelist = 'WHODD';
     if strip(upcase(compress(format,' ,'))) in ('YN','NY') then
        codelist = 'YESNOF';
     if strip(upcase(compress(format,' ,'))) in ('Y') then
        codelist = 'YESF';
     mandatory = ' ';
     comment = comments;
     if index(variable,'PARAMCD')) then VALUELIST = 'Yes';
     if index(variable,'STDY') then COMPUTATIONMETHOD = 'STDY';
     if index(variable,'ENDY') then COMPUTATIONMETHOD = 'ENDY';
run;

PROC EXPORT DATA= WORK.vars OUTFILE= "&outdir.\&xls_var."
     DBMS=EXCEL REPLACE;
         SHEET="&sheet_var";
RUN;
%mend;
%output(outdir=&outdir., xls_domain=domain_info_general,
        xls_vars=vars_info_general, xls_specs=specs);
```

When and only when consistency between the programming specification and the SDTM/ADaM datasets is achieved, can the macro %make_xls and %definexml_create define.xml be called to generate define.xml. Figure 6 (a) shows an example of TOC for define.xml. The detailed mapping rules, derivation rules, and origin information in the programming specification, in addition to variable attributes, are shown in Figure 6 (b).



(a) The Table of Contents (TOC)



(b) The Data Definition Table

Figure 6 define.xml

LESSON LEARNED

We developed this tool at the very late stage of NDA submission, that is, after the finalization of ADaM datasets. The structure of ADaM datasets is manually created based on the programming specification, and does not exactly reflect the attributes defined in the programming specification during the frequent revision in the developing period. The consistency checking between ADaM datasets and the Word programming specification requires extra work and is time consuming. Had we developed this tool earlier, say, before the creation of ADaM datasets, we would have retrieved the variable attributes from Word programming specification and created ADaM template with these attributes. It will further spare the resources, time, and workload for consistency checking between ADaM datasets and Word programming specification, and thus greatly improve the work efficiency.

CONCLUSION

This paper introduced a revised DDE solution to read programming specification in Word Format and to convert it into a SAS dataset, which is used to retrieve the useful information for consistency checking and define.xml preparation. This method automates the process to create define.xml from a file in Word Format, significantly reduces time and the resources needed

for preparation, and guarantees the high quality of the submission. We hope the methodology and the SAS codes provided in this paper can help you in saving your time and resources for NDA submission.

REFERENCES

1. CDISC Submission Data Standards Team. “Study Data Tabulation Model Implementation Guide: Human Clinical Trials”, August 2005.
http://www.cdisc.org/content1605

2. Ajay Gupta, Matthew Lesko. (2010). “Importing Data from RTF Output into SAS® utilizing Microsoft® Access/Word in an intriguing, efficient way”. PharmaSUG, May 2010.

3. Jay Zhou. (2009) ”Importing Data from Microsoft Word into SAS®”, PharmaSUG, June 2009.

4. Microsoft Press (1995), Word 95 WordBasic Reference.
http://www.microsoft.com/downloads/details.aspx?familyid=1a24b2a7-31ae-4b7c-a377-45a8e2c70ab2&displaylang=en.

5. Excel 4.0 Macro Reference. http://www.microsoft.com/downloads/en/details.aspx?FamilyID=00D31943-3AD1-4DF1-9F93-C19C7E84F01C&displaylang=en

ACKNOWLEDGEMENTS

Appreciation goes to Claude Fiset for his consistency checking tools, eSub Working Group for their define.xml macros, and Dean Gittleman and Kelly Blackburn for their review and comments.

CONTACT INFORMATION

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

Name: Min Chen**,** Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-7134
Fax: 617-460-8060
E-mail: min_chen@vrtx.com

Name: Xiangchen (Bob) Cui**,** Ph.D.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-6069
Fax: 617-460-8059
E-mail: xiangchen_cui@vrtx.com

Name: Scott Moseley, M.S.
Enterprise: Vertex Pharmaceuticals, Inc.
Address: 88 Sidney Street
City, State ZIP: Cambridge MA, 02139
Work Phone: 617-444-6162
Fax: 617-460-8059
E-mail: scott_moseley@vrtx.com