PharmaSUG 2012 - Paper CC02

Automatic Version Control and Track Changes of CDISC ADaM Specifications for FDA Submission

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

ABSTRACT

CDISC ADaM specifications documentation is the core of the programming activities in FDA submission. It serves as the primary source for ADaM programming and validation, define.xml and reviewer guide, and also helps in establishing metadata traceability. It is preferable to create ADaM specifications in Word® Format to facilitate both reviewing and approving of specifications documentation by statisticians and/or validators. Since derivation rules may be complex and subject to constant change during the whole ADaM programming activities, it is desirable to automatically keep track of different versions of ADaM programming specifications. It is more beneficiary for sponsors to keep track of different versions when ADaM programming is outsourced to external vendors. This paper introduces a SAS macro to automatically detect and report any revisions between the new version and the old version of programming specifications. It helps to generate necessary documents for version control, establishes traceability and achieves high efficiency in FDA submission.

INTRODUCTION

The programming specification documentation is a key part of the ADaM derivation process as it is used for programming and validation of ADaM datasets, preparation of reviewer guide, and the generation of the major parts of define.xml. It is usually created in Word® format as Word documents are intuitive, flexible, and user friendly for statisticians to review and track changes.

In word processing, 'Track Changes' is an editing command that is commonly used to keep track of the changes made to the original document. It is also a useful tool for collaborating on a document, as it allows multiple users to make revisions without losing the context of the original document. The tracked change will be deleted if it is accepted or rejected. Since the programming specifications for ADaM datasets are frequently updated during the whole ADaM programming activities due to complex derivation rules, 'Track Changes' command in Word can no longer fulfill the needs for tracking the specifications changes. It is preferable to automatically accomplish version control by keeping records of the different versions, and automatically track any changes in the programming specifications from any previous versions in order to help statisticians and programmers to review the new specifications and facilitate the decision making for the revision.

In this paper a macro **%read_specs** is called to automatically store both the word file and a SAS dataset of specifications with time stamp in a study subfolder named as \history. The word file can be used to recover the old version if needed, and the SAS dataset of programming specification can be used as an input to automatically detect the changes of the programming specifications by a macro **%track_change.**

The macro **%track_change** is called to automatically detect any changes in current specifications with respect to any previous version of specifications as per user request. Reports of specification changes will be generated by the macro, including variable added, variable deleted, variable attributes revised, variable comments revised, variable origin and/or controlled terminology revised, and variable number of order revised in the new specifications. The reports of changes provide the developer and reviewers a convenient tool to track any changes from any old version of specifications, and help to finalize the programming specification at the very early stage in an efficient way. The automation of tracking changes can also be conducted at the post delivery stage. If there are any revisions in the programming specification after the delivery, reports for changes will serve as documentation for audit trail. It ensures the traceability and clear documentation for FDA submission.

AN INTRODUCTION OF PROGRAMMING SPECIFICATION FOR ADAM DATASET(S)

In order to facilitate statisticians to review the programming specification and track the change, the individual programming specifications for ADaM are created in MS Word® format, as shown in Display 1. The specification for each domain is composed of three parts: domain information table, variable information table, and an optional appendix or notes for a complex algorithm or derivation rules. In the domain information table, description of the domain will serve as the label of the ADaM dataset; in the variable information table, the variable name, label, type, length, and the format will define the variable attributes of the ADaM dataset. The information in the first two parts will be retrieved by the in-house Macros **%read_specs** during ADaM Programming, and a SAS data named ADXX_**VARS** will restore the variable information from the individual programming specification. Example SAS

datasets from an ADaM specification are shown in Display 2.

1.1.1 ADSL: Subject Level Analysis Dataset

Domain Information Table

Dataset	**ADSL**
Program Name	Adsl.sas
Description	Subject-Level Analysis Data
Unique identifier Variables	usubjid
General Class	Special Purpose
Structure	One record per subject
Input Datasets	DM, DS, EX, VS, DC, HC, AE
Notes	Includes all subjects enrolled.

Variable Name	Variable Label	Type	Length	Controlled Terms or Formats	Origin	Role	Comments	Core
STUDYID	Study Identifier	Char	20		DM.studyid	Identifier	Constant Value: "ABC-ZZZ-XXX"	Req
USUBJID	Unique Subject Identifier	Char	40		DM.usubjid	Identifier	Equivalent to studyid || "-" || strip(siteid) || "-" || strip(subjid)	Req
SUBJID	Subject Identifier for the Study	Char	20		DM.subjid	Identifier	(e.g. 102130)	Req
SITEID	Study Site Identifier	Char	8		DM.siteid	Record Qualifier	DM.SITEID	Req
AGE	Age	Num	8		DM.age	Record Qualifier	Equals to DM.age	Req
AGEGR1	Pooled Age Group 1	Char	20		Derived	Record Qualifier	<=45, if age <= 45 >45 and <=65, if 45 < age <= 65 >65, if age > 65 Note: Decode variable for AGEGPN.	Perm
AGEGR1N	Pooled Age Group 1 (N)	Num	8	AGEGR1N (AGEGR1): (1) 1 = <=45 (2) 2 = >45 and <=65 (3) 3 = >65	Derived	Synonym Qualifier	Category derived if age non-missing. Equals 1, if age <= 45 2, if 45 < age <= 65 3, if age > 65	Perm

Controlled Terms or Value Level Metadata

Variable Information Table

Display 1. ADaM Programming Specifications in Word Format

Library | ADSL_VARS

Freeze | Hide | Show... | Format | Filter... | Font... | Find

Table | View

	DOMAIN	VARNUM	VARIABLE	LABEL	TYPE	DATATYPE	LENGTH	ORIGIN	TERM	CODELIST	ROLE	COMMENT	CORE	MANDATORY	pairedvar
1	ADSL	1	STUDYID	Study Identifier	Char	text	20	DM.studyid			Identifier	Constant Val...	Req	Yes	
2	ADSL	2	USUBJID	Unique Subject Identifier	Char	text	40	DM.usubjid			Identifier	Equivalent t...	Req	Yes	
3	ADSL	3	SUBJID	Subject Identifier for the Study	Char	text	20	DM.subjid			Identifier	(e.g. 102130)	Req	Yes	
4	ADSL	4	SITEID	Study Site Identifier	Char	text	8	DM.siteid			Record Qualifier	DM.SITEID	Req	Yes	
5	ADSL	5	AGE	Age	Num	float	8	DM.age			Record Qualifier	Equals to D...	Req	Yes	
6	ADSL	6	AGEGR1	Pooled Age Group 1	Char	text	20	Derived			Record Qualifier	<=45, if age ...	Perm	No	
7	ADSL	7	AGEGR1N	Pooled Age Group 1 (N)	Num	float	8	Derived	AGEGR1N	(1) 1 = <=45 (2) 2 = >45...	Synonym Qualifier	Category der...	Perm	No	AGEGR1
8	ADSL	8	AGEU	Age Units	Char	text	8	DM.ageu	YEAR	(1) YEARS	Variable Qualifier	AGEU = DM...	Req	Yes	
9	ADSL	9	SEX	Sex	Char	text	2	DM.sex			Record Qualifier	DM.SEX	Req	Yes	
10	ADSL	10	SEXN	Sex (N)	Num	float	8	Derived	SEXN	(1) 1 = M (2) 2 = F	Synonym Qualifier	Equals, 1, if ...	Req	Yes	SEX

Display 2. ADaM Programming Specifications in SAS Format

A MACRO FOR RETRIEVING THE USEFUL INFORMATION FROM SPECIFICATION AND VERSION CONTROL

The macro **%read_spec** reads the information from the individual domain programming specification in CSV format, retrieves the useful domain information and variable information based on the standard structure of the given specification, outputs SAS datasets containing ADaM specs information, and stores both word version and SAS dataset of the programming specification with time stamp in a study subfolder, named as \history.

The macro call is shown as follows.

```
%macro read_specs(indir=,specsnm=,outdir=,newdtnm=,runorder=);
```

Where

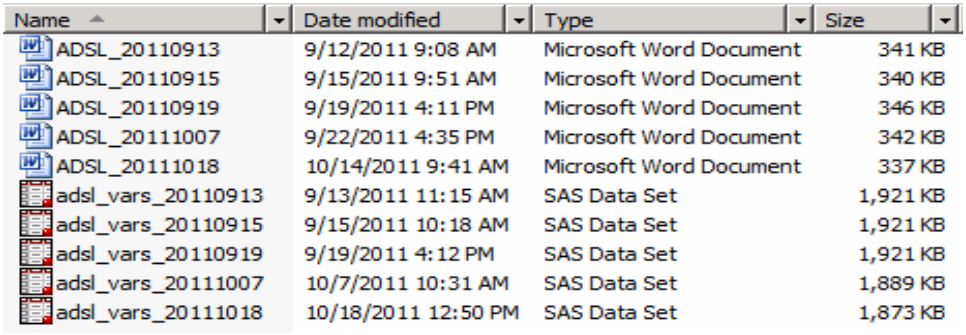
INDIR: Full Path for ADaM programming specification.
SPECSNM: Name of ADaM programming specification.
OUTDIR: Full Path for output SAS dataset which contains the attributes of ADaM variables.
NEWDTNM: A valid SAS dataset name for SAS dataset containing current specs information.

RUNORDER: A valid numeral, defining the order for a specific domain to run (in the final run).

With the macro call, the SAS datasets containing ADaM specs information are output to a study folder defined by macro variable OUTDIR. A Word file and a SAS dataset of the programming specification with time stamp at the running are output to a subfolder \history under the study folder &OUTDIR, and the code for version control is shown as follows:

```
*** Get date of programming running as time stamp ***;
data _null_;
     day = compress(put(today(),yymmdd10.),'-');
     call symput('day',strip(day));
run;
*** Output a SAS dataset of the specs with time stamp to a subfolder \history ***;
data history.&newdtnm._&day.;
     set __vars;
run;
*** Copy a Word Version with time stamp to a subfolder \history ***;
options noxwait NOXSYNC;
%let specdoc = %scan(&specsnm.,1);
X "%str(copy %"&indir.&specdoc..doc%" %"&outdir.history\&specdoc._&day..doc%")";
```

The word files of the specifications with different time stamps are stored in \history subfolder for version control purpose. The SAS dataset of specification with a time stamp can serve as an input to automatically capture the changes of the programming specifications**.** If needed, time can be added into the time stamp of word documents in addition to date.



Name	Date modified	Type	Size
ADSL_20110913	9/12/2011 9:08 AM	Microsoft Word Document	341 KB
ADSL_20110915	9/15/2011 9:51 AM	Microsoft Word Document	340 KB
ADSL_20110919	9/19/2011 4:11 PM	Microsoft Word Document	346 KB
ADSL_20111007	9/22/2011 4:35 PM	Microsoft Word Document	342 KB
ADSL_20111018	10/14/2011 9:41 AM	Microsoft Word Document	337 KB
adsl_vars_20110913	9/13/2011 11:15 AM	SAS Data Set	1,921 KB
adsl_vars_20110915	9/15/2011 10:18 AM	SAS Data Set	1,921 KB
adsl_vars_20110919	9/19/2011 4:12 PM	SAS Data Set	1,921 KB
adsl_vars_20111007	10/7/2011 10:31 AM	SAS Data Set	1,889 KB
adsl_vars_20111018	10/18/2011 12:50 PM	SAS Data Set	1,873 KB

Display 3. An Example of Version Control for ADSL Specification Document

TRACKING CHANGES

A macro **%track_change** will be invoked when tracking changes function is activated. The macro users can assign any SAS dataset in \history folder as an old version of specifications, to be compared with the current version of specifications. The reports on specifications revisions will be automatically output in RTF formats.

The macro call is shown as follows:

```
%macro track_specs(newdir=, olddir=, newdtnm=, predtnm=, outdir=);
```

Where

NEWDIR: Full Path for new ADaM programming specification.
OLDDIR: Full Path for old ADaM programming specification.
NEWDTNM: A valid SAS dataset name for SAS dataset containing current specs information.
PREDTNM: A valid SAS dataset name for SAS dataset containing old specs information.
OUTDIR: Full Path for reports of specifications changes.

The following is some notes about the logic of the macro.

1. Merge new specification with old one by variable, if the record comes from the new specification only, then the variable is a newly added one; if the record comes from the old specification only, then the variable is deleted from the old specification; if both specifications contain the variable, it will be further checked in next step.

```
data new_specs;
     length var_attrib $100.;
     set newspec.&newdtnm;
     if upcase(type)=:'CHAR' then var_attrib=trim(left(variable)) || " label='"
        || trim(left(label)) || "' length=$" ||trim(left(put(length,best3.)));
     else var_attrib=trim(left(variable)) || " label='" || trim(left(label))
        || "' length=" ||trim(left(put(length,best3.)));
run;
proc sort data=new_specs; by variable var_attrib; run;

data old_specs;
     length var_attrib $100.;
     set oldspec.&predtnm;
     if upcase(type)=:'CHAR' then var_attrib=trim(left(variable)) || " label='"
        || trim(left(label)) || "' length=$" ||trim(left(put(length,best3.)));
     else var_attrib=trim(left(variable)) || " label='" || trim(left(label))
        || "' length=" ||trim(left(put(length,best3.)));
run;
proc sort data=old_specs; by variable var_attrib; run;

data add_vars delete_vars both_;
     merge new_specs(in=new)
           old_specs(in=old rename=(var_attrib=o_var_attrib label=o_label
                                    type=o_type             length=o_length
                                    datatype=o_datatype     varnum=o_varnum
                                    origin=o_origin         term=o_term
                                    codelist=o_codelist     comment=o_comment));
     by variable;
     if new and not old then output add_vars;
     if old and not new then output delete_vars;
     if new and old then output both_;
run;
```

2. If the variable exists in both specifications, check whether there are any revisions on the variable attributes, comments, origin or controlled terminology, or the order of the variable in the specifications.

```
data changes;
     set both_;
     if upcase(var_attrib)^=upcase(o_var_attrib) then do; diff_attrib='Y';output;
end;
     if upcase(varnum)^=upcase(o_varnum) then do; diff_varnum='Y'; output; end;
     if upcase(origin)^=upcase(o_origin) then do; diff_origin='Y'; output; end;
     if upcase(term)^=upcase(o_term) then do; diff_term='Y'; output; end;
     if upcase(codelist)^=upcase(o_codelist) then do; diff_codelist='Y'; output;
end;
     if upcase(comment)^=upcase(o_comment) then do; diff_comment='Y'; output; end;
run;
```

3. Call a macro to check the number of observations in the datasets. If the datasets are not empty, output the reports for the differences between the old version and new version of specifications.

```
*** Macro to check the number of observations in the datasets. ***;
%macro get_cnt(datain=,m_cnt=new_cnt);
    %let dsid=%sysfunc(open(&datain));  ** Opens the data set to be read;
    %let &m_cnt=%sysfunc(attrn(&dsid,nobs));** Determine the number of records;
    %let rc=%sysfunc(close(&dsid));          ** Close the data set;
%mend;
%global add_vars_cnt delete_vars_cnt diff_attrib_cnt diff_comment_cnt otherchg_cnt
        diff_varnum_cnt;
*** 1. New variables added in the new version of specs. ***;
%get_cnt(datain=add_vars,m_cnt=add_vars_cnt);
%if &add_vars_cnt ^= 0 %then %do;
proc sort data=add_vars; by varnum variable var_attrib; run;
%end;
*** 2. Variables deleted in the new version of specs. ***;
%get_cnt(datain=delete_vars,m_cnt=delete_vars_cnt);
%if &delete_vars_cnt ^= 0 %then %do;
proc sort data=delete_vars; by o_varnum variable o_var_attrib; run;
%end;
*** 3. Attributes Changed ***;
proc sort data=changes(where=(diff_attrib='Y')) out=diff_attrib;
     by variable varnum;
run;
%get_cnt(datain=diff_attrib,m_cnt=diff_attrib_cnt);
%if &diff_attrib_cnt ^= 0 %then %do;
   data diff_attrib1(keep=variable varnum var_attrib)** Attributes from New Specs;
        diff_attrib2(keep=variable varnum o_var_attrib);*Attributes from Old Specs;
        set diff_attrib;
    run;
    data diff_attrib3;
         length index $12.;
         set diff_attrib1(in=a)diff_attrib2(in=b rename=(o_var_attrib=var_attrib));
         if a then index="New Specs";else if b then index="Old Specs";
    run;
    proc sort data=diff_attrib3; by varnum variable index; run;
%end;
*** 4. Comments Changed ***;
proc sort data=changes(where=(diff_comment='Y')) out=diff_comment;
     by variable varnum;
run;
%get_cnt(datain=diff_comment,m_cnt=diff_comment_cnt);
%if &diff_comment_cnt ^= 0 %then %do;
    data diff_comment1(keep=variable varnum comment)   ** Comments from New Specs;
         diff_comment2(keep=variable varnum o_comment); ** Comments from Old Specs;
         set diff_comment;
    run;
    data diff_comment3;
         length index $12.;
         set diff_comment1(in=a) diff_comment2(in=b rename=(o_comment=comment));
         if a then index="New Specs";else if b then index="Old Specs";
    run;
    proc sort data=diff_comment3; by varnum variable index; run;
%end;
*** 5. Origin and/or Format Changed ***;
proc sort data=changes out=otherchg; by variable varnum;
     where diff_origin='Y' or diff_term='Y' or diff_codelist='Y';
run;
%get_cnt(datain=otherchg,m_cnt=otherchg_cnt);
%if &otherchg_cnt ^= 0 %then %do;
   data otherchg1(keep=variable varnum origin term codelist)    ** From New Specs;
        otherchg2(keep=variable varnum o_origin o_term o_codelist);*From Old Specs;
        set otherchg;
    run;
    data otherchg3;
```

```
          length index $12.;
          set otherchg1(in=a)
   otherchg2(in=b rename=(o_origin=origin o_term=term
                          o_codelist=codelist));
      if a then index="New Specs"; else if b then index="Old Specs";
     run;
     proc sort data=otherchg3;by varnum variable index;run;
%end;
*** 6. Varnum Changed;
proc sort data=changes out=diff_varnum;by variable;
     where diff_varnum='Y';
run;
%get_cnt(datain=diff_varnum,m_cnt=diff_varnum_cnt);
%if &diff_varnum_cnt ^= 0 %then %do;
     data diff_varnum1(keep=variable varnum) diff_varnum2(keep=variable o_varnum);
          set diff_varnum;
     run;
     data diff_varnum3;
          length index $12.;
          set diff_varnum1(in=a) diff_varnum2(in=b rename=(o_varnum=varnum));
          if a then index="New Specs"; else if b then index="Old Specs";
     run;
     proc sort data=diff_varnum3; by variable index; run;
%end;
```

Tracking changes function is useful to capture any changes of the current version of ADaM specifications from any previous version, and facilitates reviewing the new ADaM specification. For example, tracking change function makes it possible to find out what contents in the ADaM specification are added, deleted, or revised. The Display 4 - 9 show the typical reports of specification changes when tracking change function is triggered.

The following Variables Were Added in the New Version of Specs.!

Variable	Variable Attribute	Comment
ARMCD	ARMCD label='Arm' length=$8	DM.ARMCD

Display 4. An Example Report for Tracking Changes: Adding Variable(s)

The following Variables Were Deleted in the New Version of Specs.!

Variable	Variable Attribute	Comment
AGEC	AGEC label='Enrollment Age Category' length=$40	if . < age <= 45 then; agec = "AGE <= 45 years"; else if 45 < age <= 65 then; agec = "45 < AGE <= 65 years"; else if 65 < age then; agec = "AGE > 65 years";

Display 5. An Example Report for Tracking Changes: Deleting Variable(s)

The following Variables with Attributes Changed!

Variable	Variable Number	Source	Variable Attribute
COHORTN	29	New Specs	COHORTN label='Cohort (N)' length=8
		Old Specs	COHORTN label='Cohort Code' length=8

Display 6. An Example Report for Tracking Changes: Change of Variable Attributes

The following Variables with Comment Changed!

Variable	Variable Number	Source	Comment
RVRFN	104	New Specs	Equals to 1 if rvrfl="Y"; Equals to 0 if rvrfl="N"
		Old Specs	Equals to 1 if rvrfl="Y"; Equals to 0 if rvrfl="N"; Equals to 9 if rvrfl="U"; Equals to 99 if rvrfl="NA"

Display 7. An Example Report for Tracking Changes: Change of Comments

The following Variables with Origin and/or Controlled Terminology/Format Changed!

Variable	Variable Number	Source	Origin	Controlled Term or Format	Codelist
UNDW24FN	108	New Specs	Derived	YESNOFN	(1) 1 = Y (2) 0 = N
		Old Specs	Derived	NYNULLFN	(1) 1 = Y (2) 0 = N (3) 9 = U (4) 99 = NA

Display 8. An Example Report for Tracking Changes: Change of Origin or Controlled Terminology

The following Variables with Variable Number Changed!

Variable	Source	Variable Number
AGE	New Specs	5
	Old Specs	25
AGEU	New Specs	8
	Old Specs	28
ARM	New Specs	34
	Old Specs	15

Display 9. An Example Report for Tracking Changes: Change of Variable Order in the ADaM Dataset

POST-DELIVERY CHANGES AND TRACKING CHANGES

If there are any changes in the programming specifications after a Clinical Study Report (CSR) delivery, the tracking change function will be triggered, and the macro **%track_specs** can be used to generate reports to capture changes from the last version of specifications. The reports will serve as documentation for audit. For example, if comments column is updated after the delivery due to editorial change, only one report is generated: the report of change of comments, as shown in Display 7.

CONCLUSION

In summary, this paper introduces an efficient macro-based tool for automatic version control, and tracking changes of the ADaM programming specifications. The macro makes it possible for the developer of the programming specifications and the reviewers to track any changes of the specifications in an efficient way. The traceability can be achieved with the storage of the previous version of specifications. The documentation for audit can be automatically generated for post delivery changes. These features improve the submission quality in a cost-effective way. We hope this tool can help you in ADaM Programming for FDA submission.

REFERENCES

CDISC Analysis Data Model Team. “Analysis Data Model (ADaM) Implementation Guide”. December 2009. http://www.cdisc.org/adam

“How Does Track Changes in Microsoft Word Work”. http://www.shaunakelly.com/word/sharing/howtrackchangesworks.html

Xiangchen (BoB) Cui, Min Chen, and Tathabbai Pakalapati. “An Innovative ADaM Programming Tool for FDA Submission”, PharmaSUG, May 2012

ACKNOWLEDGEMENTS

Appreciation goes to Kelly Blackburn, Stacy Surensky and Tathabbai Pakalapati for their review and comments.

CONTACT INFORMATION

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

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: 617-460-8060
E-mail: xiangchen_cui@vrtx.com

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