PharmaSUG 2011 - Paper AD14

Creating a define.xml file for ADaM and SDTM

John H. Adams, Boehringer Ingelheim Pharmaceutical, Inc., Ridgefield, CT

ABSTRACT

The use of Define.xml files is currently required for most FDA submissions. While the define.xml file process for SDTM only submissions is pretty stable now, many users in the pharmaceutical industry are still struggling with define.xml files that also cover ADaM submissions.

A define.xml is central to any electronic FDA submission. It is what a reviewer sees first and guides the reviewer through the objectives, analyses and data for the submission. You can think of the define file as a container of metadata (and table of contents) that describes all of the data and analysis that a submission contains. While it is a machine readable file, an accompanying style sheet allows the reviewer to display and read the file in any browser. Since the define.xml file has imbedded active links, the reviewer can easily drill down into the data and or supporting documents.

Define.xml files are dependant on two other issues, a schema and a style sheet. The schema, in essence, defines the type of data (and its hierarchical structure) that can be described in the file. The style sheet, on the other hand, describes how to display (or render) the data in a browser. You can't include data (elements or attributes) in the file that are not part of the schema. Logically, you also can't have the style sheet reference data (elements or attributes) that are not part of the schema.

While there are standard SDTM schema and style sheet available from CDISC, this is not the case for ADaM. The final drafts of these are still under discussion by the CDISC team. The CDISC pilot 1 project did create and used a modified schema / style sheet set. This paper describes a project for creating a metadata user interface and a program to create a viable SDTM/ADaM define.xml file, using that pilot 1 schema / style set.

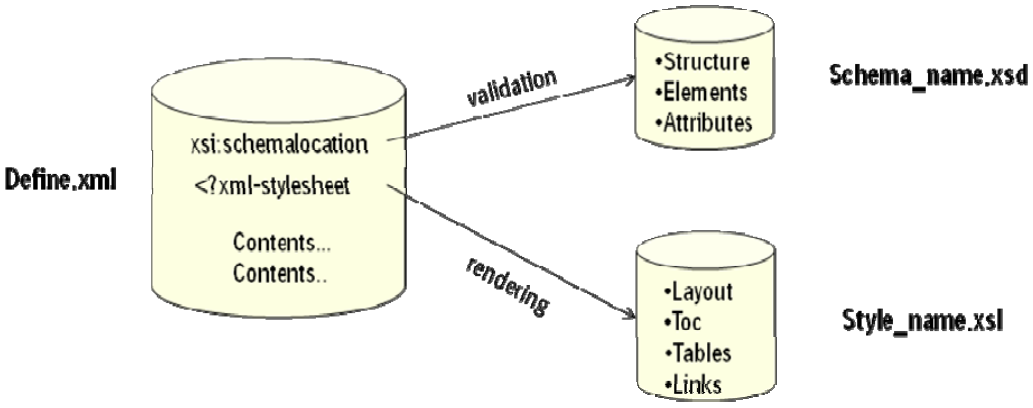
1 INTRODUCTION

A define.xml is central to any electronic FDA submission. It is what a reviewer sees first and guides the reviewer through the objectives, analyses and data for the submission. You can think of the define file as a container of metadata (and table of contents) that describes all of the data and analysis that a submission contains. While it is a machine readable file, an accompanying style sheet allows the reviewer to display and read the file in any browser. Since the define.xml file has imbedded active links, the reviewer can easily drill down into the data and or supporting documents.

A define.xml file is basically a markup language type file containing a bunch of data items, each of which is surrounded by tags, e.g. <NOTE> data </NOTE>. These are called elements. An element can have child-elements, values or attributes. For example, **< NOTE >** is a root element with several child-elements that have values. Here's a simple example:

```
<NOTE>
    <TO>KAREN</TO>
    <FROM>JOHN</FROM>
    <HEADING>REMINDER</HEADING>
    <BODY>PLEASE VALIDATE MACRO</BODY>
</NOTE>
```

Creating an xml file is quite easy since it is essentially a sequential ASCII type file. However, creating a valid define.xml file is much more difficult. The define.xml file must be properly constructed according to a specific CDISC schema, supplied along with the define.xml file. This schema defines the internal structure of allowable elements and their composition. Additionally, a style sheet must also be supplied. This style sheet defines the rendering or layout of the display for a define.xml file.



This paper will describe our project to create SDTM and ADaM compatible define.xml files, using the schema and style sheet from the CDISP Pilot 1 project. It will also provide a brief tutorial/primer on schema and style sheets.

1.1 A SCHEMA TUTORIAL

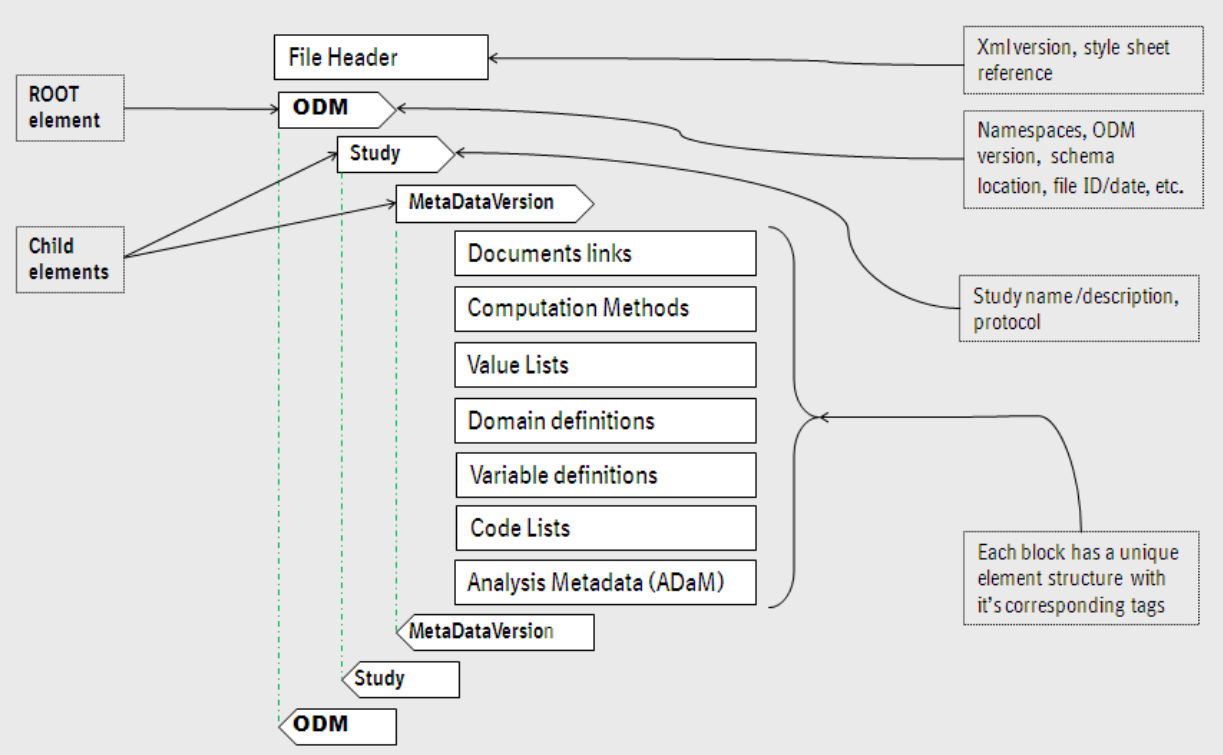
A schema for a define.xml file defines the:

1. the elements that can appear
2. the attributes that can appear for elements
3. which elements are child elements
4. the order (structure) of child elements
5. the number of child elements
6. whether an element is empty or can include text
7. the data types for elements and attributes
8. default and fixed values for elements and attributes

Xml schemas are based on ODM and CDISC standards, but they are extensible. You might ask why we would want to use a schema. Well, a schema makes it easy to:

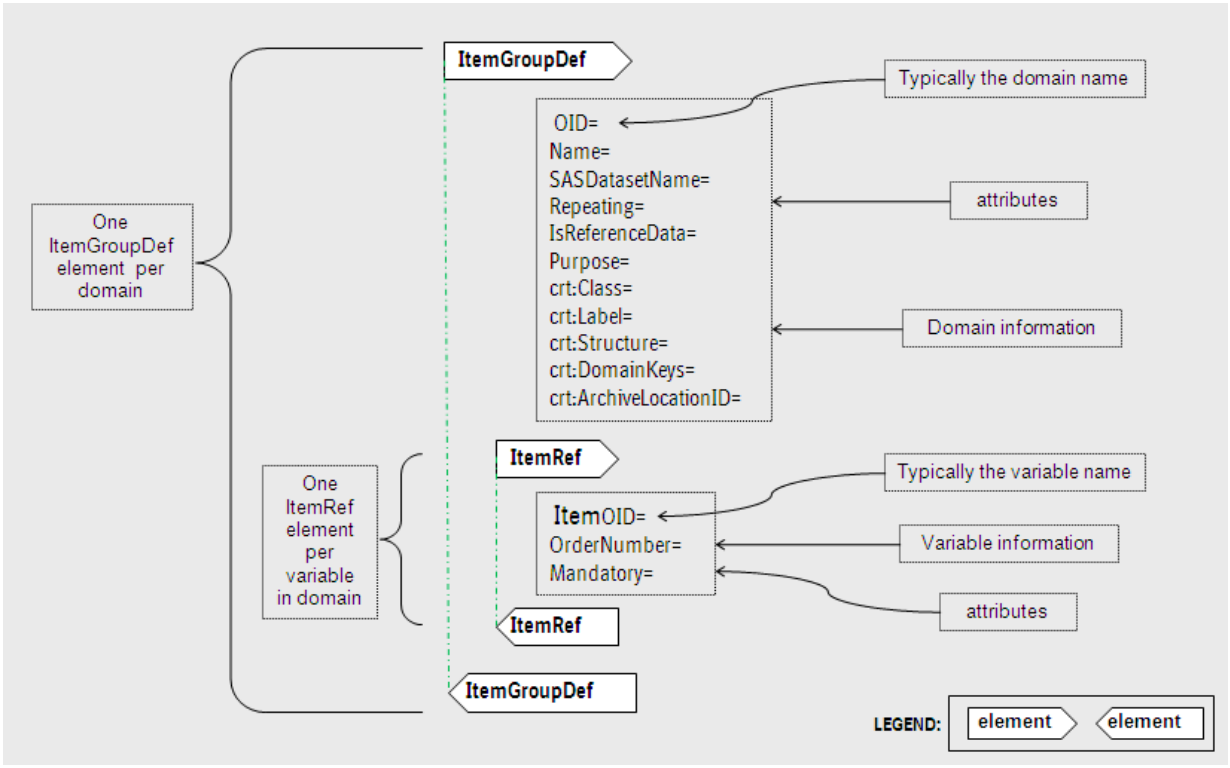
1. describe allowable file content
2. validate the correctness of data
3. work with data from databases
4. define data aspects (restrictions on data)
5. define data patterns (data formats)
6. convert data to different data types

The following diagram shows the general CDISC (pilot 1) schema structure that was used for this application. It is capable of carrying both SDTM and ADaM data.

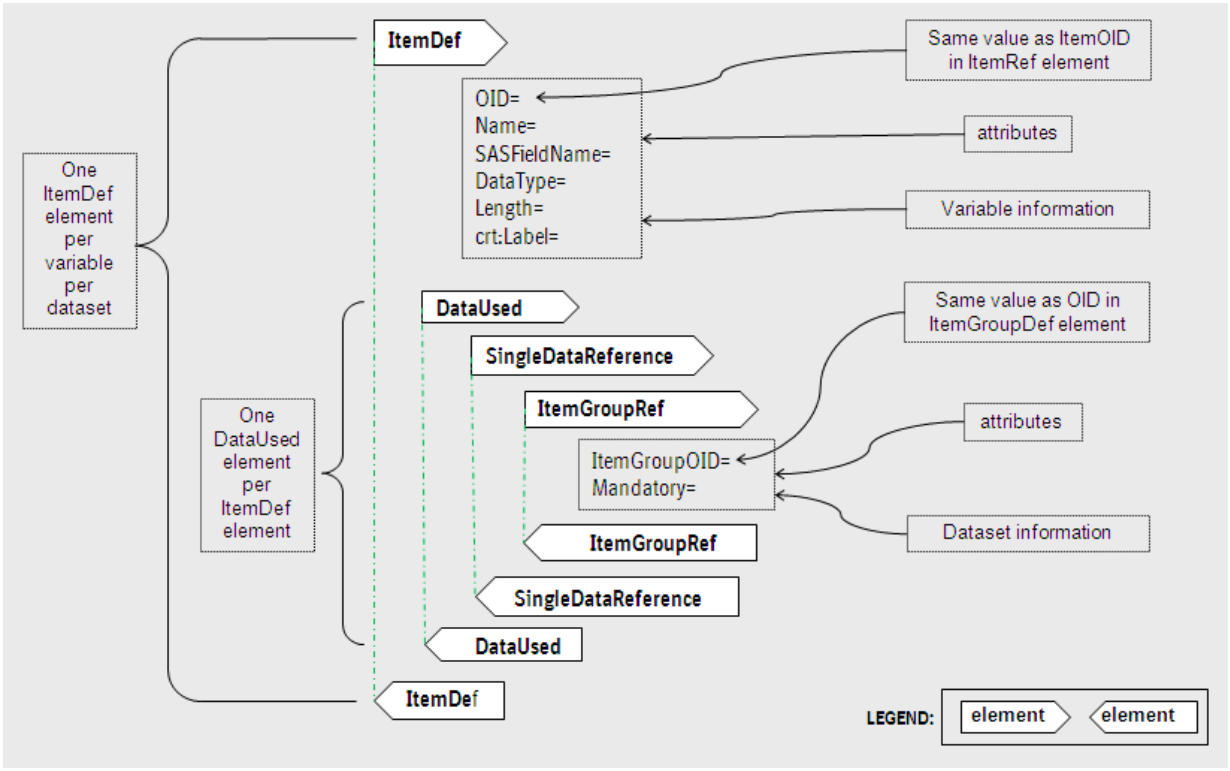


Three of the most important substructures, i.e. those for the Domain definitions, Variable definitions and Analysis results, are shown in more details below. Others are not shown in this paper.

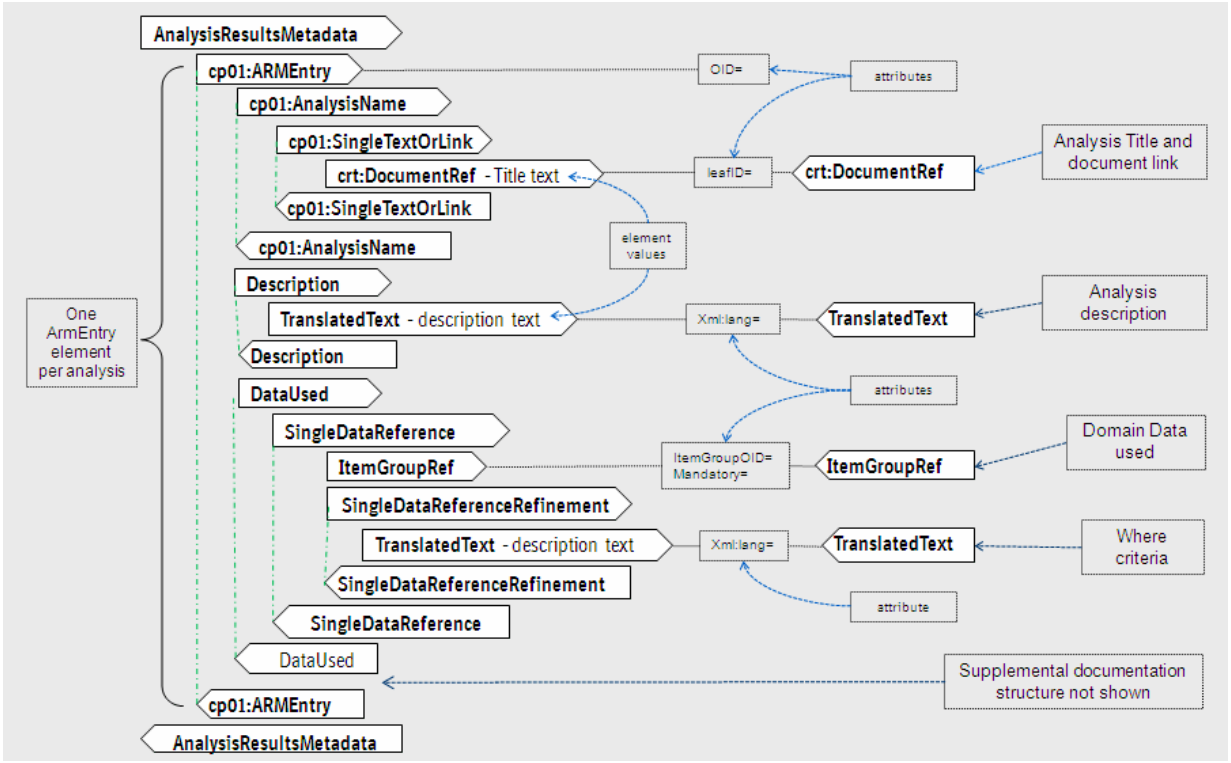
The **ItemGoupDef** structure below defines the domains that are included in the submission:



The **ItemDef** structure below defines all variables in each domain:

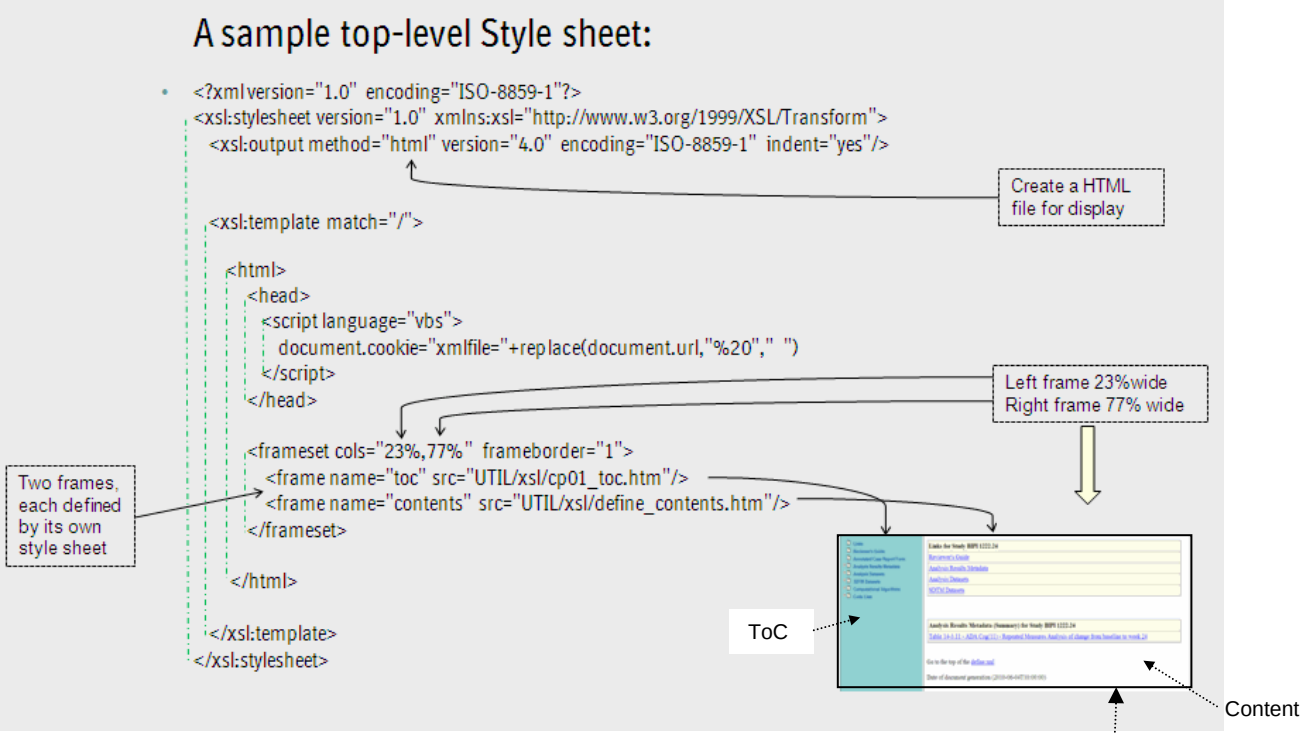


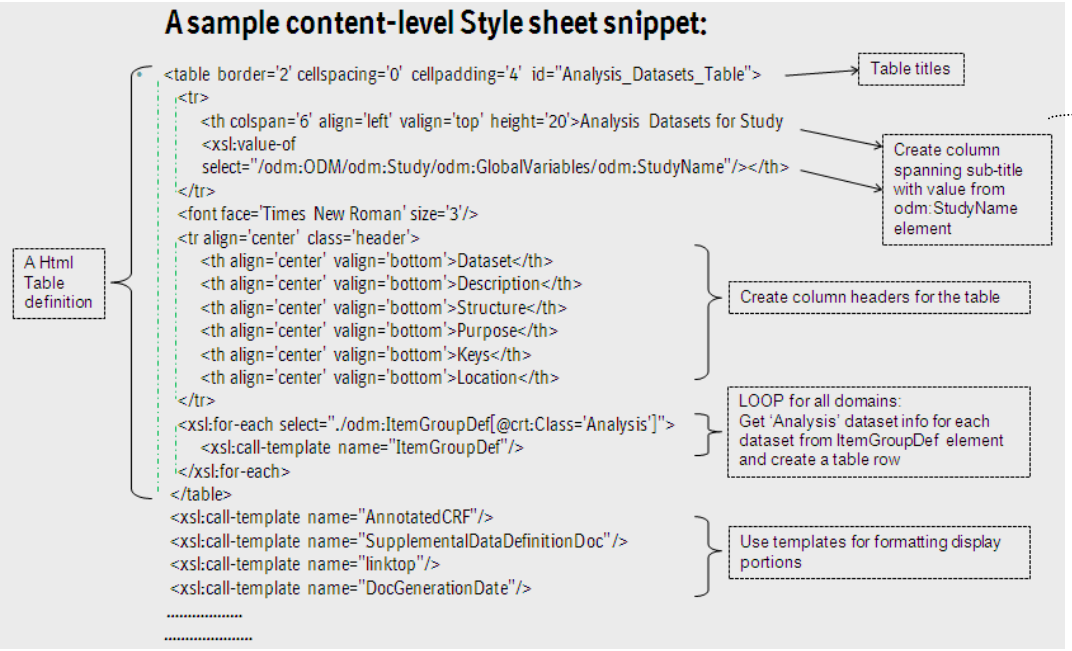
The **AnalysisResultsMetadata** structure (partial) below defines all analysis' in the submission:



1.2 STYLESHEET TUTORIAL

Style sheets are written in XML syntax and are stored as XSL files. The style sheet is used to transform an XML document into another type of document, like HTML, that is recognized by a browser. All major browsers support XML and XSL type files. With a style sheet you can rearrange and sort elements, perform tests, make decisions about which elements to hide and display, etc. So a linked style sheet for a define.xml file defines the layout of the desired display, i.e. how the browser should display / render elements from the define.xml file. Of course, style sheets are also extensible.



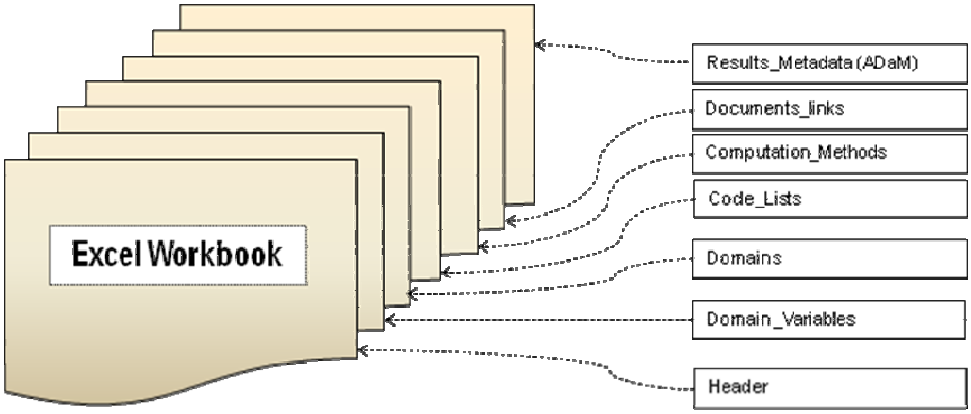


2 THE APPLICATION

2.1 User interface for Input of Metadata

The first hurdle to overcome was to design an easy user interface to capture the metadata needed for the define.xml file. It was decided to use an EXCEL workbook as input during the first phase of this project. A later phase would eliminate the workbook and pull the metadata automatically from other sources.

The EXCEL workbook was organized to have seven separate sheets (tabs) that logically contain the major types of data needed, as per the following diagram:



As you'll notice, the sheets do not reflect a one-to-one mapping to major schema elements, i.e. ItemGroupDef, ItemDef, etc., as some developers have done. In fact, the Domains_Variable sheet sources both the ItemGroupDef and the ItemDef elements. The major focus of the sheet design was, instead, on creating logical groupings of data that users understand. All sheets have additional help built in, i.e. drop-down selections, data checking, popup comments, etc. Let's look at samples of all sheets.

Following is a sample Header sheet:

A	B	E	F
Parameter_Description	**Parameter_Value**		
xmlVersion	1.0		
xmlLanguage	ISO-8859-1		
StyleSheetType	text/xsl		
StyleSheetPath	UTIL/XSL/cp01.xsl		
CDISC_ODM_ReferenceDocument	http://www.cdisc.org/ns/odm/v1.3		
W3_XML_SchemaDocument	http://www.w3.org/2001/XMLSchema-instance		
W3_ReferenceDocument	http://www.w3.org/1999/xlink		
CDISC_CRT_SchemaExtensions	http://www.cdisc.org/ns/crt/v3.1.1		
CDISC_SDTM_ModelReference	http://www.cdisc.org/ns/sdtm/v3.1.1		
CDISC_Pilot_CP01_Namespace	http://www.cdisc.org/ns/pilot/1.0		
CDISC_DEF_DocumentReference	http://www.cdisc.org/ns/def/v1.0		
SchemaLocation	http://www.cdisc.org/ns/odm/v1.3 UTIL/cp01.xsd		
ODM_Version	1.3		
FileOID	CDISC ADaM		
FileType	Snapshot		
CreationDate_time	2010-06-04T10:00:00		
AsOfDate_time	2010-05-26T02:39:00		
StudyOID	BIPI XXXX.yy		
StudyName	BIPI XXXX.yy		
StudyDescription	A phase III Randomised, double-blind, double-dummy, placebo-controlled, 4-way cross-over study to determine the 24-hour FEV1-time profiles of an orally inhaled drug		
ProtocolName	BIPI XXXX.yy		
MetaDataOID	BIPI XXXX.yy		
MetaDataVersionName	Study BIPI XXXX.yy , Data Definitions		
MetaDataDescription	Study BIPI XXXX.yy, Data Definitions		
SchemaVersionUsed	1.0.0		
CDISC_StandardName	CDISC ADaM		
CDISC_StandardVersion	2.0		

Header | Domains | Domain_Variables | Code_lists | Computation_Methods | Document_links | Results_metadata

Tabs represent sheets

The following figure shows the layout of the **domain sheet**:

Dataset Name	Dataset Description	Dataset Location	Dataset Structure	Key Variables of Dataset	Class of Dataset	Documentation	Repeating	Reference Data
ADSL	Baseline and Covariate data	adsl.xpt	One record pers subject	USUBJID	ADSL	Section X.4 of the ADS plan.	No	No
ADPFT	Pulmonary function testing data	adpft.xpt	One record per subject, Baseline type, Period, Analysis day, end point parameter, Planned time and analysis flag	USUBJID, BASETYPE, APERIOD, PADY, PARAMN, ATPTN, ANL01FN	BDS	Section X.5.1 of the ADS plan.	Yes	No

Describe structure of datset, i.e. one record per test, per visit, per subject

Comment appears when a cell is selected

Next, we see a sample of the **Domain_variables** sheet (partial) below:

Dataset Name	Parameter Identifier	Variable Name	Variable Label	Variable Type	Display Format	Code or Controlled Terms	Variable Derivation	Variable Source	Variable Length	Variable Role	Variable Comment	Variable Order
ADSL	ALL	USUBJID	Unique subject identifier	text	$15				15			1
ADSL	ALL	STUDYID	Study identifier	text	$15				9			2
ADSL	ALL	SUBJID	Subject identifier for the study	text	$10				10			3
ADSL	ALL	PTNO	Patient number	integer	10			patd.ptno	8			4
ADSL	ALL	SITEID	Study site identifier	text	$10				10			5
ADSL	ALL	INVNAME	Name of investigator	text	$20				20			6
ADSL	ALL	UCENTRE	Unique centre number	text	$20				15			7
ADSL	ALL	CENTRE	Centre	integer	10				8			8
ADSL	ALL	COUNTRY	Country	text	$7				7			9
ADSL	ALL	COUNTRYC	Country description	text	$40				40			10
ADSL	ALL	SEXN	Sex (N)	integer	1	code1			8			11
ADSL	ALL	SEX	Sex	text	$8				8			12
ADSL	ALL	RACEN	Race (N)	integer	1				8			13
ADSL	ALL	RACE	Race	text	$40				19			14
ADSL	ALL	AGE	Age	integer	3				8			15
ADSL	ALL	AGEU	Age units	text	$8				20			16
ADSL	ALL	AGEGR1	Age group 1	integer	1				8			17
ADSL	ALL	AGEGR1C	Age group 1 (C)	text	$10				10			18
ADPFT	ALL	USUBJID	Unique subject identifier	text	$15				16			1
ADPFT	ALL	STUDYID	Study identifier	text	$15				9			2
ADPFT	ALL	SUBJID	Subject identifier for the study	text	$10				10			3
ADPFT	ALL	PTNO	Patient number	integer	10				8			4
ADPFT	ALL	SITEID	Study site identifier	text	$10				10			5
ADPFT	ALL	UCENTRE	Unique centre number	text	$20				20			6
ADPFT	ALL	CENTRE	Centre	integer	10				8			7
ADPFT	ALL	TRTP	Planned treatment	text	$60				8			8
ADPFT	ALL	PTRSORT	Planned treatment sort variable	text	$20				20			9
ADPFT	ALL	BASETYPE	Baseline type	text	$40				40			10
ADPFT	ALL	PADY	Planned analysis relative day	integer	3		method1		8			17
ADPFT	ALL	PCUMADY	Planned cumulative analysis relative	integer	3		method2		8			18

Only used for SDTM

CodeList_ID defined in 'code_lists' sheet

Enter Codelist ID (must be defined in CODE_LISTS tab)

Computation_MethodID Defined in 'Computation_Methods' sheet

The referenced code lists from the above sample are defined in the sample **Code lists** sheet below:

CodeList ID	Codelist Name	Data Type	Code Value	Code Rank	Decode Text	Language
code1	gender	integer	1	1	Male	en
code1	gender	integer	2	2	Female	en
code2	endpoint	character	FEV	1	Forced expiratory volume in 1	en
code2	endpoint	character	FEV200	2	Trough (pre-dose) FEV1 [L]	en
code2	endpoint	character	FEV201	3	Trough (24h) FEV1 [L]	en
code2	endpoint	character	FEV211	4	FEV1 Peak (0-3h) [L]	en
code2	endpoint	character	FEV214	5	FEV1 Peak (0-12h) [L]	en
code2	endpoint	character	FEV215	6	FEV1 Peak(12-24h) [L]	en
code2	endpoint	character	FEV220	7	FEV1 AUC (0-3h) [L]	en
code2	endpoint	character	FEV223	8	FEV1 AUC (0-12h) [L]	en
code2	endpoint	character	FEV224	9	FEV1 AUC (0-24h) [L]	en
code2	endpoint	character	FEV225	10	FEV1 AUC (12-24h) [L]	en

The referenced computation methods from the above Domain_variables sample are defined in the sample **Computation_methods** sheet below:

A	B	C	D
Computation MethodID	**Computation Name**	**Computation Description**	**Language**
Method1	planday	if visit in (2,5,8,11) then pady = 1; if visit in (4,7,10,13) then pady = 43;	en

Method2	Cumday	if visit = 2 then pcumday = 1; if visit = 4 then pcumday = 43;
		if visit = 5 then pcumday = 57; if visit = 7 then pcumday = 99;
		if visit = 8 then pcumday = 113; if visit = 10 then pcumday = 155;
		if visit = 11 then pcumday = 169; if visit = 13 then pcumday = 211;
			en

Computation_MethodID referenced in 'Domain variables' sample sheet

Algorithm

All documents are defined in the sample **Document links** sheet below:

A	B	C	D
Document ID	**Document Title**	**Document Location**	**Document Bookmark**
blankcrf	Annotated Case Report Form	blankcrf.pdf	
suppdoc1	Supplemental document	suppdoc1.pdf	
ReviewerGuide	Reviewer's Guide	coverletter.pdf	#page=3
Report1	SAP Section 10.1.1	TSAP.pdf	#nameddest=section_10.1.1
Report2	Table 14-3.11	analreports1.pdf	#nameddest=Out_table_14-3.11
Report3	Table 14.6	analreports1.pdf	#nameddest=Out_table_14.6
Report4	Table 15.1	analreports2.pdf	#nameddest=Out_table_15.1

DisplayDocumentID

Document is located in same folders as define file, unless preceded by a path

Point to a location within a Document (optional)

And finally, a sample **Results_Metadata** sheet sample. This defines each analysis (one per row) below:

A	B	C	D	E	F	G	H	I	J	K	L	M
Display Identifier	**Display Document Link**	**Display Name**	**Result Identifier**	**Param**	**ParamCD**	**Analysis Variable**	**Reason**	**Dataset**	**Selection Criteria**	**Other Document Links**	**Documentation Text**	**Program Statements**
Table 14-3.11	Report2	ADA Cog(11) - Repeated Measures Analysis of change from baseline to week 24	ADAS-Tot	ADAS-Cog(11) Total Score	ACTOT11	CHG	Pre-specified in SAP	ADPFT	ITTFL='Y' and DTYPE NE 'LOCF and Paramcd='ACTOT11'	Report1, Report3	Adjusted means for the change fromBaseline at week 24 and pairwise comparisons between treatment groups	Proc Mixed; Class Usubjid sitegr1 avisitn trp; Model chg=trtp sitegr1 avisitn trtp* avisitn.....;Run;

Primary DisplayDocumentID as defined in Document_lnks

Other DisplayDocumentIDs as defined in Document_links sheet

2.2 The program

A SAS program was written to create the define.xml file. The program has 3 major sections:

1. The **input section** uses the XLS engine to read each spreadsheet into a dataset. Using the XLS engine in SAS 9.2 as opposed to using Proc Import has a number of advantages, as seen by the simplicity of this code:

```
%MACRO escapechars (char_str=);
  &char_str = TRANWRD (&char_str, '&', '&amp;');
  &char_str = TRANWRD (&char_str, "<", '&lt;');
  &char_str = TRANWRD (&char_str, ">", '&gt;');
  &char_str = TRANWRD (&char_str, "'", '&apos;');
  &char_str = TRANWRD (&char_str, '"', '&quot;');
  &char_str = TRANWRD (&char_str, '0a'x, '&#xA;');
%MEND;

libname xlsdata 'F:\CDISC\1222_Project\PharmaSug\datasets\analysis\User_defined_xml_input.xls'
          getnames=yes scantext=yes scan_textsize=yes version=2002;

data header;
  set xlsdata.'Header$'n;
run;

data Domains ;
  set xlsdata.'Domains$'n  ;
  length item_group_oid $50;
  item_group_oid=trim(dataset_name);
  if dataset_name ^= ' ';
  %escapechars(char_str=dataset_description);
run;
proc sort data=Domains; by dataset_name;  run;

data Domain_variables ;
  set xlsdata.'Domain_variables$'n;
  length item_oid item_group_oid $50;
  item_group_oid=trim(dataset_name);
  item_oid=trim(dataset_name)||'.'||trim(variable_name);
  %escapechars(char_str=variable_label);
run;
proc sort data=Domain_variables; by dataset_name variableorder;  run;

data Code  lists;
  set xlsdata.'Code_lists$'n;
  length codelist_oid $50;
  codelist_oid='Codelist.'||trim(codelist_id);
  %escapechars(char_str=decode_text);
  if codelist_id ^=' ';
run;
proc sort data=Code_lists;    by codelist_id;    run;

data Computation_methods ;
  set xlsdata.'Computation_methods$'n;
  length compmethod_oid $50;
    %escapechars(char_str=computation_description);
  compmethod_oid='Method.'||trim(Computation_methodid);
run;

data Document  links ;
  set xlsdata.'Document_links$'n;
run;

data Results_metadata;
  set xlsdata.'Results_metadata$'n;
    %escapechars(char_str=display_name);
    %escapechars(char_str=reason);
    %escapechars(char_str=documentation_text);
    %escapechars(char_str=selection_criteria);
    %escapechars(char_str=program_statements);
run;
```

Macro to replace non-allowable characters in XML elements

Read in each spreadsheet and do some pre-processing

2. The second section processes the input data and creates the actual text lines, stored in datasets, for the seven major elements in the define.xml file. The creation of these text lines is very schema dependant, so you must be very familiar with it in order to create these text lines..

In this application, the text lines represent the proper formats and structures of elements in the CDISC pilot 1 schema. Only the control variable setup and the coding for one of the major elements (ItemGroupdef) is shown here for illustration. Coding for the other major elements follows a similar methodology.

The following code block creates a number of macro control variables. These are used during the processing of the seven major elements / blocks.

```
proc sql noprint;
  select count(parameter_description)              into: header_cnt                from header;
  select count(distinct dataset_name)              into: domain_cnt                from domains;
  select distinct dataset_name                     into: domain_names   separated by '~' from domains;
  select count(distinct dataset_name)              into: dsname_cnt                from domain_variables ;
  select count(variable_name)                      into: Tot_var_cnt               from domain_variables;
  select count(distinct code_or_controlled_terms)  into: CT_cnt                    from domain_variables;
  select count(Variable_derivation)                into: comp_cnt                  from domain_variables;
  select distinct dataset_name                     into: dsname_names   separated by '~' from domain_variables;
  select DS_var_cnt                                into: DS_var_cnt        separated by '~'
        from (select count(variable_name) as DS_var_cnt , dataset_name  from domain_variables group dataset_name);
  select distinct dataset_name                     into: ds_names1         separated by '~'
        from domain_variables order by dataset_name;
  select distinct code_or_controlled_terms         into: CT_names          separated by '~' from domain_variables;
  select distinct Variable_derivation              into: comp_IDs          separated by '~' from domain_variables;
  select count(distinct codelist_name)             into: codelst_cnt              from code_lists;
  select distinct codelist_name                    into: codelst_IDs       separated by '~' from code_lists;
  select distinct language                         into: language1                from code_lists;
  select count(distinct Computation_methodid)      into: meth_cnt                 from computation_methods;
  select distinct Computation_methodid             into: meth_IDs          separated by '~' from computation_methods;
  select distinct language                         into: language2                from computation_methods;
  select count(distinct Document_id)               into: doc_cnt                  from document_links;
  select distinct Document_id                      into: doc_IDs           separated by '~' from document_links;
  select count(distinct Display_identifier)        into: display_cnt              from Results_metadata;
  select distinct Display_identifier               into: display_IDs              from Results_metadata;
  select Display_document_link                     into: analdoc_ID        separated by '~' from Results_metadata;
  select other_document_links                      into: analdoc_lst       separated by '~' from Results_metadata;
  select distinct dataset                          into: ds_names2         separated by ' '  from Results_metadata;
quit;
```

Next, there are separate code blocks for creating the element structures of major elements. The following code is for creating the ItemGroupDef structure. As you can see below, each 'LINE' text is constructed complete with the appropriate tags, sub elements, attributes and values.

```
data xml  ItemgroupDef(keep=line drop=errmsg);
   length line $5000 errmsg $200;
 if &domain_cnt = 0 then do;
    errmsg = 'ERR' || 'OR: - (ItemGroupDef) dataset count in DOMAINS = 0';
    put errmsg;
 end;
 else if &dsname_cnt = 0 then do;
    errmsg = 'ERR' || 'OR: - (ItemGroupDef) dataset count in DOMAIN_VARIABLES = 0';
    put errmsg;
 end;
 else if &domain_cnt ^= &dsname_cnt then do;
    errmsg = 'ERR' ||'OR: - (ItemGroupDef) dataset count in DOMAINS(' ||trim(left(put(&domain_cnt,best8.))) ||
           ') not equal to that in DOMAIN_VARIABLES(' || trim(left(put(&domain_cnt,best8.))) || ')';
    put errmsg;
 end;
 else if &domain_cnt > 0 then do i=1 to lastdom;                    /* start loop through domains */
   set domains point=i nobs=lastdom;
    if dataset_name ^= ' ' then do;
      dataset_type='SDTM';                                          /*Default value*/
      line = ' ';                                                                                  output;
      line = '<!-- ********************************************************************************** -->';  output;
      line = '<!-- Defining ItemGroupDef for dataset ' || trim(dataset_name) || '                  -->';  output;
      line = '<!-- ********************************************************************************** -->';  output;
      line = '<ItemGroupDef OID="' || trim(dataset_name) || "" ;                                    output;
      line = '  Name="' || trim(dataset_name) || "" ;                                              output;
      line = '  SASDatasetName="' || trim(dataset_name) || "" ;                                    output;
      if  Repeating = ' ' then Repeating = ' ';
      line = '  Repeating="' || trim(repeating) || "";                                             output;
      if  Reference_data = ' ' then Reference_data = 'No';
      line = '  IsReferenceData="' || trim(Reference_data) || "";                                  output;

      if trim(left(upcase(Class_of_Dataset))) in('ADSL' 'BDS' 'OTHER') then do;      /*ADaM dataset*/
        dataset_type='ADAM';
        line = '  Purpose="Analysis"' ;                                                          output;
        line = '  crt:Class="Analysis"' ;                                                        output;
      end;
      else do;                                                                                   /*SDTM type dataset*/
        line = '  Purpose="Tabulation"' ;                                                        output;
        line = '  crt:Class="' || trim(left(Class_of_Dataset)) ||"" ;                            output;
      end;

      %escapechars(char_str=dataset_description);
      line = '  crt:Label="' || trim(dataset_description) || "";                               output;
      line = '  crt:Structure="' || trim(dataset_structure) || "";                             output;
      line = '  crt:DomainKeys="' || trim(Key_Variables_of_dataset) || "";                     output;
      line = '  crt:ArchiveLocationID="Location.' || trim(dataset_location) || "";             output;
      line = '>';                                                                            output;
      line = '   ';                                                                          output;

    if &dsname_cnt > 0 then do j=1 to lastvar;                  /* start loop through vars for dataset*/
      set domain_variables(rename=(dataset_name=dom_name)) point=j nobs=lastvar;
     if dom_name = dataset_name then do;
        line = '<!-- ItemRef for dataset (' || trim(dataset_name) || ') Variable ' || trim(Variable_name) || '       -->';  output;
        line = '  <ItemRef ItemOID="' || trim(Variable_name) || "";                            output;
        line = '    OrderNumber="' || trim(left(VariableOrder)) || "";                         output;
      if Value_Required = ' ' then Value_Required = 'No';
        line = '    Mandatory="' || trim(Value_Required) || "";                                output;
      if dataset_type='SDTM' and Variable_Role ^= ' ' then do;          /*define role for SDTM only*/
        line = '    Role="' || trim(Variable_Role) || "";                                      output;
      end;
      if Variable_derivation ^= ' ' then do;
       /* check if Variable_derivation variable is a linked method*/
        if index(upcase("&meth_IDs"),trim(left(upcase(Variable_derivation))))>0 then do;
          line = '    MethodOID="' || "MethList." || trim(left(upcase(Variable_derivation))) || "";   output;
```

Loop through each domain

Loop through variable for each domain

(From previous page)

Loop through variable for each domain

Loop through each domain

```
      end;
    end;
      line = '  />';                                                                    output;

  end;
end;    /* end loop through vars for dataset*/

      line = ' <crt:leaf ID="Location.' || trim(dataset_Name) || '"  xlink:href="' || trim(dataset_location) || '">';    output;
      line = ' <crt:title>' || trim(dataset_location) || '</crt:title>';                output;
      line = ' </crt:leaf >';                                                           output;

      line = '</ItemGroupDef>';                                                         output;
  end;
end;       /* end loop through datasets */

 Stop;
run;
```

Coding for the other major elements follows a similar methodology, i.e. creating the actual text lines in a dataset, as the above example, but won't be shown in this paper.

3. This step is relatively simple, as most work has already been done in the 2nd step. The final define.xml file is created in this step by successively reading each of the seven data steps in proper order and writing each text line to the external file.

```
. filename xmlout 'F:\CDISC\1222_Project\PharmaSug\datasets\analysis\define.xml' lrecl=5000 ;

data _null_;

  set  xml_header xml_doclinks  xml_itemgroupdef xml_itemdef2 xml_codelist xml_methodlist xml_results_metadata

       end=done;
 file xmlout;
  put line;

 if not done then return;
 else do;
  line = ' ';                                                                                  put line;
  line = '<!-- ********************************************************************************************** -->';    put line;
  line = '<!--                Close the container elements                                   -->';    put line;
  line = '<!-- ********************************************************************************************** -->';    put line;
  line = '</MetaDataVersion>';                                                                 put line;
  line = '</Study>';                                                                           put line;
  line = '</ODM>';                                                                             put line;
 end;
run;

libname  xlsdata clear;
filename xmlout  clear;
```

Data sets with line text for all elements

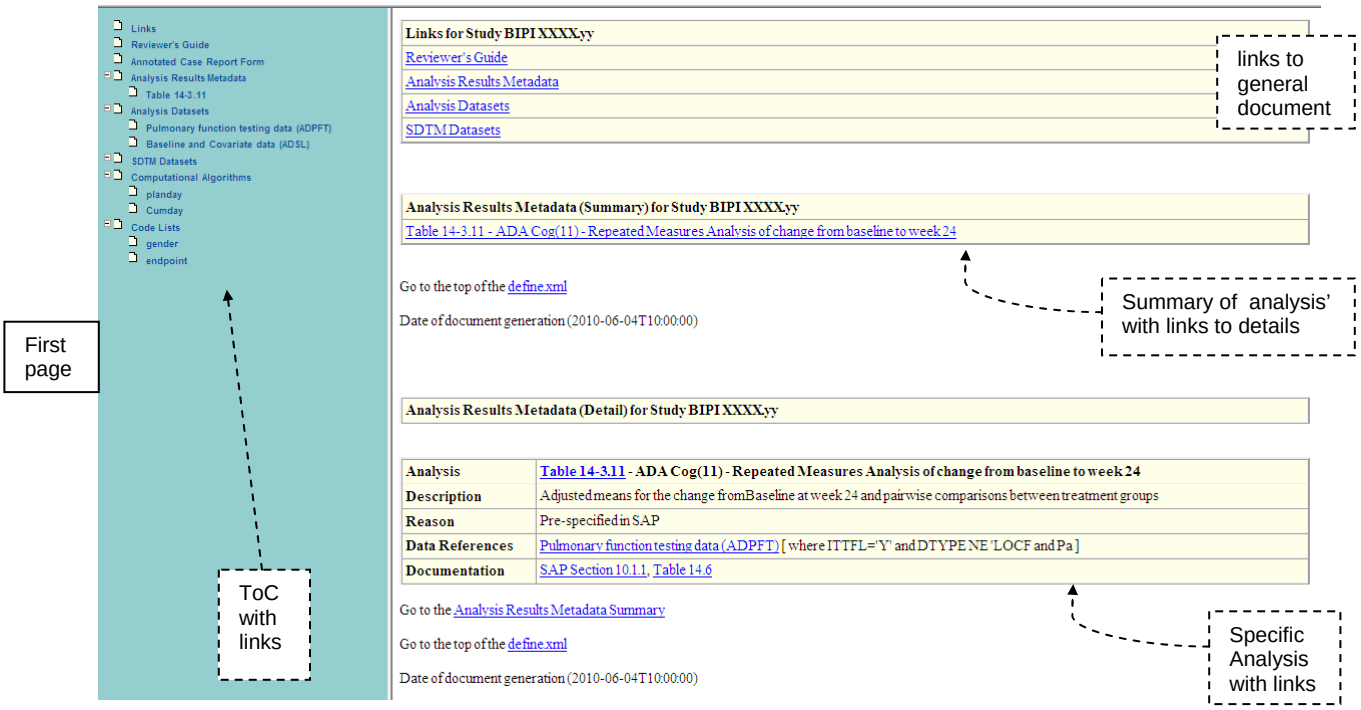
2.3 The Results

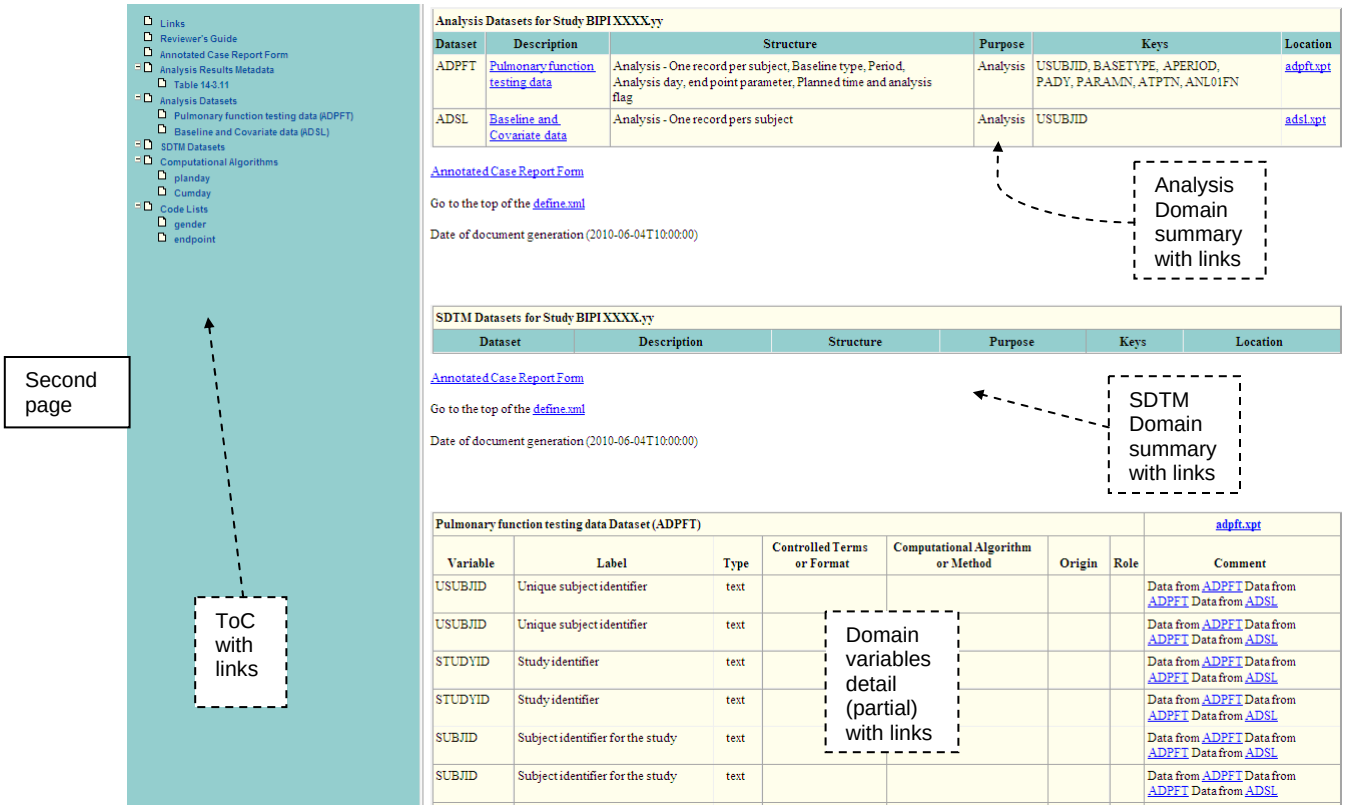
The program runs extremely fast (a few seconds) and produces a well formed XML file that is both SDTM and ADaM Implementation Guide (IG) compatible.

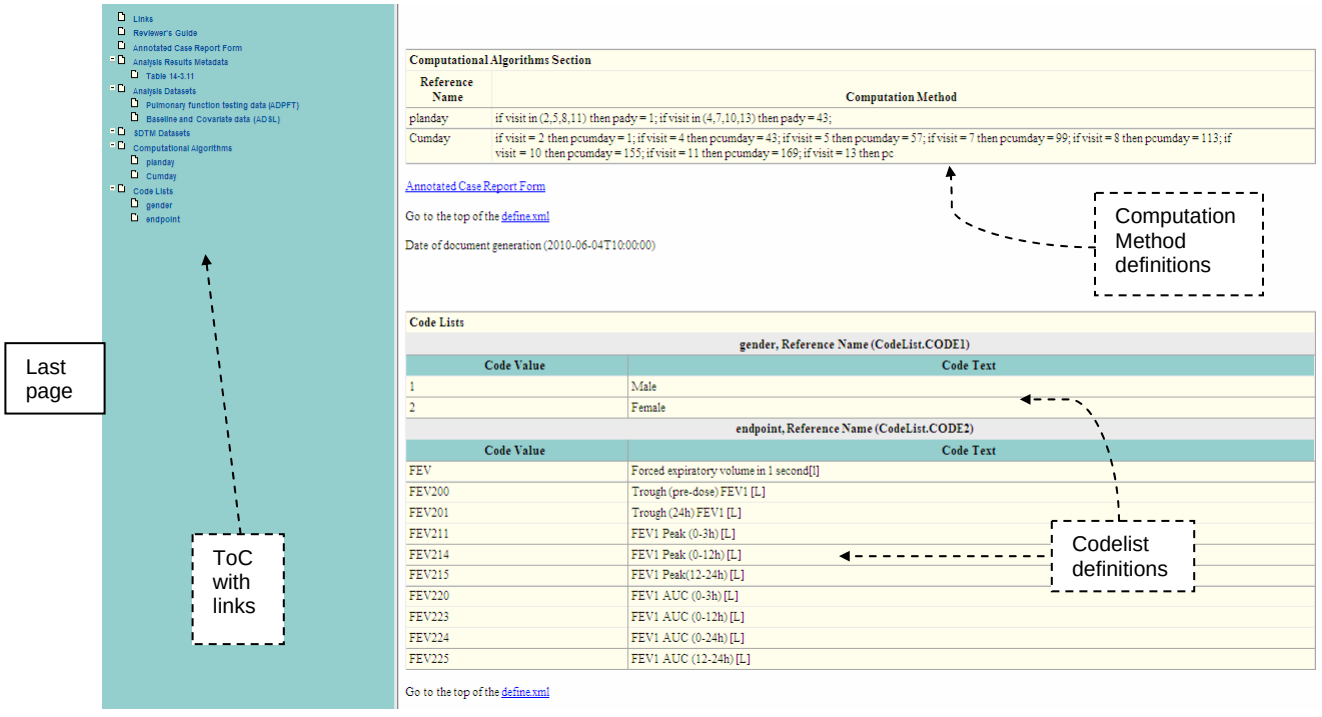
The log of a sample run shows that the external file was written:

```
NOTE: The file XMLOUT is:
      Filename=F:\CDISC\1222_Project\PharmaSug\datasets\analysis\define.xml
      RECFM=V,LRECL=5000
      Last Modified=16Feb2011:14:36:07
      Create Time=15Feb2011:11:19:04

NOTE: 4961 records were written to the file XMLOUT.
      The minimum record length was 1.
      The maximum record length was 311.

NOTE: There were 40 observations read from the data set WORK.XML_HEADER.
NOTE: There were 36 observations read from the data set WORK.XML_DOCLINKS.
NOTE: There were 1221 observations read from the data set WORK.XML_ITEMGROUPDEF.
NOTE: There were 3532 observations read from the data set WORK.XML_ITEMDEF2.
NOTE: There were 76 observations read from the data set WORK.XML_CODELIST.
NOTE: There were 16 observations read from the data set WORK.XML_METHODLIST.
NOTE: There were 33 observations read from the data set WORK.XML_RESULTS_METADATA.
NOTE: DATA statement used (Total process time):
      real time           0.10 seconds
      cpu time            0.07 seconds
```

Following are screen shots of the define.xml file as rendered by the Internet Explorer browser:







Links
Reviewer's Guide
Annotated Case Report Form
Analysis Results Metadata
Table 14-3.11
Analysis Datasets
Pulmonary function testing data (ADPFT)
Baseline and Covariate data (ADSL)
SDTM Datasets
Computational Algorithms
planday
Cumday
Code Lists
gender
endpoint

Analysis Datasets for Study BIPI XXXX.yy

Dataset	Description	Structure	Purpose	Keys	Location
ADPFT	Pulmonary function testing data	Analysis - One record per subject, Baseline type, Period, Analysis day, end point parameter, Planned time and analysis flag	Analysis	USUBJID, BASETYPE, APERIOD, PADY, PARAMN, ATPTN, ANL01FN	adpft.xpt
ADSL	Baseline and Covariate data	Analysis - One record pers subject	Analysis	USUBJID	adsl.xpt

Annotated Case Report Form

Go to the top of the define.xml

Date of document generation (2010-06-04T10:00:00)

SDTM Datasets for Study BIPI XXXX.yy

Dataset	Description	Structure	Purpose	Keys	Location

Annotated Case Report Form

Go to the top of the define.xml

Date of document generation (2010-06-04T10:00:00)

Pulmonary function testing data Dataset (ADPFT) — adpft.xpt

Variable	Label	Type	Controlled Terms or Format	Computational Algorithm or Method	Origin	Role	Comment
USUBJID	Unique subject identifier	text					Data from ADPFT Data from ADPFT Data from ADSL
USUBJID	Unique subject identifier	text					Data from ADPFT Data from ADPFT Data from ADSL
STUDYID	Study identifier	text					Data from ADPFT Data from ADPFT Data from ADSL
STUDYID	Study identifier	text					Data from ADPFT Data from ADPFT Data from ADSL
SUBJID	Subject identifier for the study	text					Data from ADPFT Data from ADPFT Data from ADSL
SUBJID	Subject identifier for the study	text					Data from ADPFT Data from ADPFT Data from ADSL

Computational Algorithms Section

Reference Name	Computation Method
planday	if visit in (2,5,8,11) then pady = 1; if visit in (4,7,10,13) then pady = 43;
Cumday	if visit = 2 then pcumday = 1; if visit = 4 then pcumday = 43; if visit = 5 then pcumday = 57; if visit = 7 then pcumday = 99; if visit = 8 then pcumday = 113; if visit = 10 then pcumday = 155; if visit = 11 then pcumday = 169; if visit = 13 then pc

Annotated Case Report Form

Go to the top of the define.xml

Date of document generation (2010-06-04T10:00:00)

Code Lists

gender, Reference Name (CodeList.CODE1)

Code Value	Code Text
1	Male
2	Female

endpoint, Reference Name (CodeList.CODE2)

Code Value	Code Text
FEV	Forced expiratory volume in 1 second[1]
FEV200	Trough (pre-dose) FEV1 [L]
FEV201	Trough (24h) FEV1 [L]
FEV211	FEV1 Peak (0-3h) [L]
FEV214	FEV1 Peak (0-12h) [L]
FEV215	FEV1 Peak(12-24h) [L]
FEV220	FEV1 AUC (0-3h) [L]
FEV223	FEV1 AUC (0-12h) [L]
FEV224	FEV1 AUC (0-24h) [L]
FEV225	FEV1 AUC (12-24h) [L]

Go to the top of the define.xml

3 Conclusions

While there are standard SDTM schema and style sheet available from CDISC, this is not the case for ADaM. The final drafts of these are still under discussion by the CDISC team.

The CDISC pilot 1 project did create and used a modified schema / style sheet set. The application described in this paper bridges the gap for the immediate future by using the CDISC pilot 1 schema and style sheet set. It provides us with a process for creating a viable SDTM/ADaM define.xml file right now. When a final schema and style sheet set becomes available from CDISC, the program can easily be adapted. Additionally, our plans call for more automation of the metadata user interface.

CONTACT INFORMATION

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

John H Adams
900 Ridgebury Road
Ridgefield, CT, 06877-0368
Work Phone: 203-778-7820
Fax: 203-837-4413
Email: john.adams@boehringer-ingelheim.com
adamsjh@mindspring.com