

Work Less – Do More: An Automated Approach to Define-XML Validation

Kai Wanke, OCS Life Sciences

Coding Tips and Tricks – CT02

11 Nov 2019, PHUSE EU Connect, Amsterdam

Challenges In Define-XML Creation

- Define-XML creation can be challenging:

- Long and complex

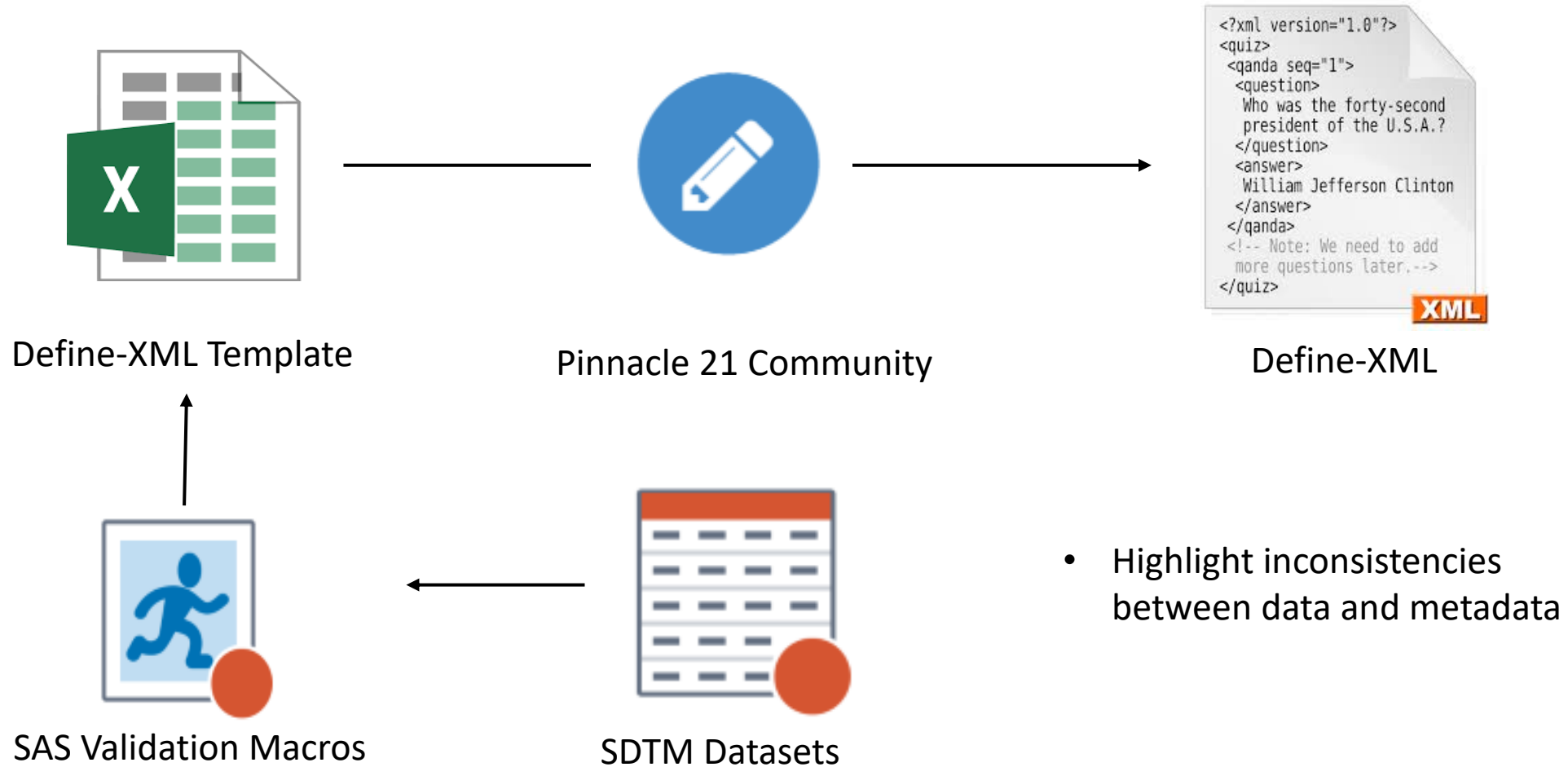
- Some aspects require interpretation (e.g. origin or value-level information)

- Hard to maintain consistency

→ Extensive validation required

```
<ItemDef OID="IT.AE.AEACN" Name="AEACN" DataType="text" Length="30" SASFieldName="AEACN">
  <Description>
    <TranslatedText xml:lang="en">Action Taken with Study Treatment</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.ACNC"/>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankorf">
      <def:PDFPageRef PageRefs="21" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>
<ItemDef OID="IT.AE.AEBODSYS" Name="AEBODSYS" DataType="text" Length="52" SASFieldName="AEBODSYS">
  <Description>
    <TranslatedText xml:lang="en">Body System or Organ Class</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.AEDICT_F"/>
  <def:Origin Type="Assigned"/>
</ItemDef>
<ItemDef OID="IT.AE.AEDECOD" Name="AEDECOD" DataType="text" Length="18" SASFieldName="AEDECOD">
  <Description>
    <TranslatedText xml:lang="en">Dictionary-Derived Term</TranslatedText>
  </Description>
  <CodeListRef CodeListOID="CL.AEDICT_F"/>
  <def:Origin Type="Assigned"/>
</ItemDef>
<ItemDef OID="IT.AE.AEENDTC" Name="AEENDTC" DataType="date" SASFieldName="AEENDTC">
  <Description>
    <TranslatedText xml:lang="en">End Date/Time of Adverse Event</TranslatedText>
  </Description>
  <def:Origin Type="CRF">
    <def:DocumentRef leafID="LF.blankorf">
      <def:PDFPageRef PageRefs="21" Type="PhysicalRef"/>
    </def:DocumentRef>
  </def:Origin>
</ItemDef>
<ItemDef OID="IT.AE.AEENDY" Name="AEENDY" DataType="integer" Length="3" SASFieldName="AEENDY">
  <Description>
    <TranslatedText xml:lang="en">Study Day of End of Adverse Event</TranslatedText>
  </Description>
  <def:Origin Type="Derived"/>
</ItemDef>
<ItemDef OID="IT.AE.AEENRF" Name="AEENRF" DataType="text" Length="5" SASFieldName="AEENRF">
```

Automatic Generation of Define-XML



- 1) Summarize all variables and values in the study
- 2) Define variable types
- 3) Compare SDTM to Define-XML
- 4) Generate output

Step 1) Summarize all Variables in the Study

```
proc sql noprint;
```

```
    create table work.temp_sdtm01 as  
    select memname as dataset  
    from    dictionary.tables  
    where   lowercase(libname) = "sdtm";
```

```
quit;
```

```
proc sql noprint;
```

```
    select distinct quote(trim(memname))  
    into    :numvar_datasets separated by ","  
    from    dictionary.columns  
    where   lowercase(libname) = "sdtm"  
    and     type = "num";
```

```
quit;
```

- Use dictionary tables to select all variables in a study
- Distinguish between character and numeric variables

Step 1) Summarize all Variables in the Study

```
data _null_;
  set work.temp_sdtm01;

  call execute(cat(
    "data work.temp_char_",strip(dataset),";
      set sdtm.",strip(dataset),";
      array _char{*} _character_;
      length _dataset _variable _value _var_type $ 200;
      retain _var_type ","'char'",";
      _dataset = '",strip(dataset),'"; "
    ,"do i=1 to dim(_char);
      _variable = vname(_char{i});
      _value = _char{i};
      output;
    end; "
    ,"keep _dataset _variable _value _var_type;
      run;"
    ,"proc sort data = work.temp_char_",strip(dataset)," nodupkey;
      by _dataset _variable _value;
      run;")));

run;
```

- Call execute is used to write code to generate datasets containing all values of all variables for each SDTM dataset

 _dataset	 _variable	 _value	 _var_type
AE	AEACN	DOSE NOT CHANGED	char
AE	AEACN	DRUG INTERRUPTED	char
AE	AEACN	DRUG WITHDRAWN	char
AE	AEBODSYS	Cardiac disorders	char
AE	AEBODSYS	Gastrointestinal disorders	char

Step 2) Define Variable Types

```
data work.temp_type01;
  set work.temp_all01;

  if missing(_value) then _type = "missing";
  else if _var_type = "char" then _type = "text";
  else if find(_value, ".") and length(strip(_value)) > 1 then _type = "float";
  else _type = "integer";

  if _type = "missing" then do;
    if substr(_variable, length(_variable)-2, 3) = 'DTC' then do;
      temp_value = translate(_value, "xxxxxxxxxx", "0123456789");
      if find(temp_value, "xxxx-xx-xxTxx:xx") then _type = "datetime";
    end;
  end;
end;
drop temp_value;
run;
```

 _dataset	 _variable	 _value	 _var_type	 _type
AE	AEACN	DOSE NOT CHANGED	char	text
AE	AEACN	DRUG INTERRUPTED	char	text
AE	AEACN	DRUG WITHDRAWN	char	text
AE	AEBODSYS	Cardiac disorders	char	text
AE	AEBODSYS	Gastrointestinal disorders	char	text

Step 3) Compare SDTM to Define-XML

```
proc sql;
```

```
create table work.temp_comp_vartype (drop = order) as  
select a.order  
      ,a.dataset  
      ,a.variable  
      ,a.data_type as xml_type  
      ,b.type_c as act_type
```

→ Type in template
→ Type in SDTM

- Merge the output of the macro with the Define-XML template
- This allows the comparison of expected and actual values

```
from work.xml_variables (where = (^missing(variable))) as a  
left join work.temp_type01 as b  
  
on a.dataset = b._dataset  
and a.variable = b._variable  
order by dataset, order;
```

```
run;
```

 Dataset	 Variable	 xml_type	 act_type
AE	AEBODSYS	float	text
AE	AESEV	text	text

Step 3) Compare SDTM to Define-XML

```
data work.check_vartype;
```

```
set work.temp_comp_vartype;
```

```
length type id result compare $ 200;
```

```
type = "Variable type";
```

```
id = cat("VAR_TYPE_",strip(put(_n_,z3.)));
```

```
if xml_type ^= act_type then do;
```

```
    compare = cat(
        "The variable ",strip(variable),
        " on dataset ",strip(dataset),
        " is defined as ",strip(xml_type)
        "when the expected type is
        ",strip(act_type),".");
```

```
    result = "Wrong";
```

```
end;
```

```
run;
```

- The data set checks if expected and actual values are equal
- It writes error messages if this is not the case

Dataset	Variable	xml_type	act_type
AE	AEBODSYS	float	text
AE	AESEV	text	text

result	compare
Wrong	The variable AEBODSYS on dataset AE is defined as float when the expected type is text.

Step 4) Generate Output

```
proc report data = work.check_vartype;
  columns dataset variable result;

  define dataset / display "Dataset";
  define variable / display "Variable";
  define result / display "Type";

  compute result;
    else if find(result, "Wrong") then do;
      call define(_col_, "style",
        "style=[background=red]");
    end;
    else if find(result, "Check") then do;
      call define(_col_, "style",
        "style=[background=yellow]");
    end;
  endcomp;
run;
```

- Proc Report generates is used to summarize all findings
- The report assigns a colour code to the error messages

→ The user obtains an easy-to-read summary of potential issues in their metadata

Step 4) Generate Output

Type of check	Dataset	Variable	ID	Message
Variable Type	AE	AEBODSYS	VAR_TYPE_004	The Variable AEBODSYS on dataset AE has type float but should be text.

Summary

- Define-XML creation can be tricky, but automation can make it **easier, faster** and **more consistent** across studies
- Benefits of the Define-XML validation macros:
 - **Modular** → can be adapted easily to any metadata requirements
 - **Little manual input**
 - Validate metadata using **objective** measures
 - They deal with **Dataset** - , **Variable** – and **Value** – level metadata
 - **Reduce time** and **effort** during the generation of Define-XML
- Complements our Define-XML **development** tool

Thank you!

Contact Details:

Kai Wanke

OCS Life Sciences

Email: kai.wanke@ocs-consulting.com

**Do you have more comments or questions?
Please visit us at the OCS Life Science Booth**




```
proc sort data = work.temp_type01;  
  by _dataset _variable _type;  
run;
```

```
data work.temp_type02;  
  set work.temp_type01;  
  by _dataset _variable _type;  
  if last._variable then output;  
run;
```

```
data work.temp_type03 (rename=(type_c = _type));  
  set work.temp_type02;  
  length type_c $ 200;  
  select(_type);  
    when(0)   type_c = "missing";  
    when(1)   type_c = "integer";  
    when(2)   type_c = "float";  
    when(3.3) type_c = "durationDatetime";  
    when(3.4) type_c = "partialTime";  
    when(3.5) type_c = "Time";  
    when(4)   type_c = "text";  
  
  drop _type;  
run;
```

Combine all Messages

```
/*Combine all variable checks*/
proc sql;
  create table work.table_variables as
  select g.order as order
        ,a.dataset
        ,a.variable
        ,a.result as type
        ,b.result as length
  from work.check_vartype as a
  left join work.check_varlen as b
    on a._dataset = b._dataset
    and a._variable = b._variable
  left join work.xml_variables as c
    on a._dataset = c.dataset
    and a._variable = c.variable
  order by dataset, c.order;
quit;
```

- There are separate macros for different aspects of the Define-XML (e.g. length, origin, aCRF pages)
- Each macro creates error messages that are being merged here