Define.XML-centric dataset development

Leonid Tseytlin, CS Ltd, Kharkov, Ukraine
Purpose of Define.xml

• From the standard: “To support the interchange of dataset metadata for clinical research applications in a machine-readable format. An important use case for Define-XML is to support the submission of clinical trials data in CDISC SDTM, SEND or ADaM format to regulatory authorities”
Purpose of Define.XML

• Often created in the end of analysis
• But is it for interchange (or submission) only?
• Can it be used during programming?
Purpose of Define.XML

• Key point about Define.XML: “machine-readable”
• Can be read by your SAS program for useful applications
Define.XML-centric paradigm for dataset development

• Create Define.XML from your dataset specs
  – Our team uses in-house Excel template for specs and a custom conversion utility
  – Another possibility is Pinnacle21
  – Specs do not need to be final

• Create utility macros that use define.xml to help in various programming tasks
Declaring Variable Attributes

• How many times did you write a piece of code like this?

```r
data dm;
  attrib STUDYID length=16 label = "Study Identifier";
  attrib DOMAIN length=2 label = "Domain Abbreviation";
  attrib USUBJID length=23 label = "Unique Subject Identifier";
  attrib SUBJID length=6 label = "Subject Identifier for the Study";
etc.
```

• Probably way too many
Declaring Variable Attributes

• But all this information is already in define.xml
• We wrote a macro that reads these variable attributes from define.xml and translates them into appropriate SAS attrib statements

```sas
data dm;
  %DefineVarAttrib;
```
Verifying the Dataset Structure

• How do we make sure the dataset matches its specs?
  – QC, Double programming

• But can we catch some obvious mistakes, such as missing variable, extra variable, wrong data type, automatically?
Verifying the Dataset Structure

• We created a macro that compares the dataset against its specifications in define.xml. It verifies the following:
  – That the dataset has all the variables from the specs and does not have any extra variables
  – That all variables have the expected attributes (type, length, label, SAS format)
  – That all required variables are not missing
  – That variables with associated code lists have values compatible with these code lists
  – That date/time variables in ISO 8601 format are formatted properly
  – The trickiest part: that value-level metadata is adhered to.
Controlled Terminology: codes vs. decodes vs. ranks

• Disposition terms: (in DS.DSDECOD or perhaps ADSL.DCSREAS)

<table>
<thead>
<tr>
<th>ADVERSE EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LACK OF EFFICACY</td>
</tr>
<tr>
<td>PROTOCOL DEVIATION</td>
</tr>
<tr>
<td>WITHDRAWAL BY SUBJECT</td>
</tr>
</tbody>
</table>

• Presentation in a table:

<table>
<thead>
<tr>
<th>Adverse Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lack of Treatment Effect</td>
</tr>
<tr>
<td>Protocol Violation</td>
</tr>
<tr>
<td>Subject Withdrawn Consent</td>
</tr>
</tbody>
</table>

• Order in table is not alphabetic, e.g. follows CRF
Controlled Terminology: codes vs. decodes vs. ranks

Encode these mapping in a code list:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Coded Value</th>
<th>Decode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADVERSE EVENT</td>
<td>Adverse Event</td>
</tr>
<tr>
<td>4</td>
<td>LACK OF EFFICACY</td>
<td>Lack of Treatment Effect</td>
</tr>
<tr>
<td>2</td>
<td>PROTOCOL DEVIATION</td>
<td>Protocol Violation</td>
</tr>
<tr>
<td>3</td>
<td>WITHDRAWAL BY SUBJECT</td>
<td>Subject Withdrew Consent</td>
</tr>
</tbody>
</table>

<CodeList Name="NCOMPLT" DataType="text">
  <CodeListItem CodedValue="ADVERSE EVENT" def:Rank="1">
    <Decode>
      <TranslatedText xml:lang="en">Adverse Event</TranslatedText>
    </Decode>
  </CodeListItem>
</CodeList>
Controlled Terminology: codes vs. decodes vs. ranks

• We wrote a macro that translates such code list found in define.xml into a group of SAS formats and informats that map the standard terminology to the study-specific terms and sort orders

```sas
DSTERM_NO = INPUT(DSDECOD, NCOMPLT.);
DSTERM_DISP = PUT(DSDECOD, $NCOMPLT.);
```
Supplemental Qualifiers Datasets

• Supplemental qualifiers datasets are one of the less programming-friendly features of the venerable SDTM standard

• Hopefully, it is to be phased out and replaced with non-standard variables in SDTM IG 3.3
Supplemental Qualifiers Datasets

• It is not hard to merge a “supp” dataset to its main dataset, but the variable types are lost.

<table>
<thead>
<tr>
<th>QNAM</th>
<th>QVAL (char 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETRTEM</td>
<td>Y</td>
</tr>
<tr>
<td>NFUP</td>
<td>2</td>
</tr>
<tr>
<td>DESCR</td>
<td>A very long description</td>
</tr>
</tbody>
</table>

After merge all variables are char 200:

<table>
<thead>
<tr>
<th>AETRTEN</th>
<th>NFUP</th>
<th>DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>2</td>
<td>A very long description</td>
</tr>
</tbody>
</table>
Supplemental Qualifiers Datasets

• Can define type of each supplemental variable using a value list in define.xml

<table>
<thead>
<tr>
<th>QNAM</th>
<th>DataType</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>AETRTEM</td>
<td>text</td>
<td>1</td>
</tr>
<tr>
<td>NFUP</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>DESCR</td>
<td>text</td>
<td>200</td>
</tr>
</tbody>
</table>

• We wrote a macro that merges the supp-dataset enforcing types from the value list:

<table>
<thead>
<tr>
<th>AETRTEM</th>
<th>NFUP</th>
<th>DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>char 1</td>
<td>num</td>
<td>A very long description</td>
</tr>
</tbody>
</table>
Conclusion

• Define.xml can be created early in dataset programming process
• Can be used to facilitate programming of datasets to help ensure quality and speed up the development process
• Dataset metadata in a machine-readable format is very valuable for programming, opens a lot of possibilities
• A sample of which was presented
Thank you!

• Contact information:
Leonid Tseytlin
CS Ltd
42a Tobolskaya street
Kharkov 61072
Ukraine
Email: lz@csltd.com.ua