XML mit SAS leicht gemacht

Andreas Adlichhammer
HMS Analytical Software
Rohrbacher Straße 26
Heidelberg
Andreas.adlichhammer@analytical.software.de

Zusammenfassung

Dank seiner weltweiten Verbreitung und Akzeptanz ist XML die Technologie von Heute für den strukturierten Austausch von Daten aller Art. Hunderte Standards aus Technik und Wissenschaft zeigen, wie Inhalte in XML verpackt werden können– von der Vektorgraphik bis hin zum Jahresabschluss. Wie SAS Sie dabei unterstützt diese Inhalte für Ihre Arbeit zu erschließen, beschreibt dieser Beitrag. Die wichtigsten SAS XML Technologien werden Ihnen dafür nicht nur in Theorie und Beispiel vorgestellt, sondern kritisch analysiert und gegenübergestellt.
.

Schlüsselwörter: SAS, XML, SXLE, XMLMap, Tagset, PROC XSL, XSLT

1 XML – eine Kurzvorstellung

Die Extensible Markup Language ist eine einfache und flexible Auszeichnungssprache. XML wurde 1998 vom Word Wide Web Consortium ursprünglich für die Erfassung und Verbreitung elektronischer Publikationen entwickelt [2]. Heute findet sich XML weltweit in den meisten IT-Produkten und in elektronischen Dokumenten aller Art wieder. XML ist damit de facto Standard in vielen IT Bereichen: Webserver und andere IT-Produkte speichern ihre Konfiguration in XML Dateien, große Firmen wie Microsoft oder Adobe nutzen XML als Basistechnologie für ihre Dokumente und Firmen senden Ihre Geschäftsberichte in XBRL an Aufsichtsbehörde.

In den meisten Anwendungsfällen versteckt sich XML vor dem Endanwender und wird in Dokumenten, Grafiken oder Programmoberflächen aufbereitet und präsentiert. Auf der Ebene der Programme und Inhalte muss man sich aber ein Grundverständnis der XML Technologie aneignen. Dies gilt für alle Programmiersprachen, von VBA über Java bis eben auch hin zu SAS.

1.1 XML Dokumente

Typisch für ein XML Dokument sind die eckigen Klammern. Innerhalb und um die eckigen Klammern sind die Bezeichner der strukturierenden Elemente oder der eigentliche Inhalt eingebettet. Ein XML Dokument muss darüber hinaus noch eine Vielzahl an Regeln einhalten um ein gültiges XML Dokument zu sein. Man spricht dann von einem „wohlgeformten“ XML Dokument. Neben der hier vorgestellten technischen Validität gibt es auch die inhaltliche und strukturelle Validität. Die inhaltliche und strukturelle Validität wird von verschiedenen Standards, oft in sogenannten XML Schemata, definiert. Dieser Beitrag geht auf diese Standards und Schemata nicht genauer ein.

Abbildung 1: Beispiel für ein XML Dokument und seine Bestandteile

Tabelle 1: Wichtige XML Bestandteile

XML Deklaration	`<?xml version="1.0"?>`
Computerprogramme können an der Deklaration erkennen, dass es sich um ein XML Dokument in der XML Version 1.0 handelt.	
Wurzel	`<Einkaufsliste>`
Jedes XML Dokument besitzt genau eine Wurzel (auch Wurzelelement, Wurzelknoten). Alle anderen Inhalte finden sich innerhalb des öffnenden und schließenden Wurzelelement (sie „Tags“)	

Element	`<Artikel />`
Elemente bilden die Basisstruktur des Dokuments. Ein Element ist immer in zwei eckige Klammern eingeschlossen und kann, wie in obigem Beispiel, mit < geöffnet und mit /> geschlossen werden. In der Regel werden Elemente aber durch ein schließendes Elementtag wieder geschlossen. (siehe „Tags“)	
Attribut	`Name="Butter"`
Attribute stehen innerhalb der eckigen Klammern eines Elements und beziehen sich auf dieses Element. Attribute bestehen immer aus einem Attributnamen (Name) und dem Attributwert (Butter) in Hochkommata.	
Kommentar	`<!-- Standard ist immer 1 Stueck -->`
Kommentare werden mit <!-- eingeleitet und mit --> geschlossen. Kommentare werden von Programmen nicht ausgewertet.	
öffnendes Tag	`<Artikel>`
Alle Elemente und Wurzelelemente werden mit < und > geöffnet. Nachfolgende Elemente und Inhalte befinden sich innerhalb des Elements.	
schließendes Tag	`</Artikel>`
Alle Elemente und das Wurzelelement werden mit </ und > geschlossen. Ein schließendes Tag gehört dabei immer mit dem öffnenden Tag zusammen.	
Inhalt	`2`
Inhalt kann zwischen den Elementen im leeren Raum eingetragen werden und bezieht sich immer auf das unmittelbar einschließende Element. Es sind sowohl Zahlen als auch alphanumerische Einträge erlaubt.	
Sonderzeichen	`"Br&#246;tchen"`
Wie viele computernahe Sprachen müssen auch in XML besondere Zeichen umcodiert werden, damit während der Verarbeitung keine Probleme entstehen. Dies betrifft Umlaute aber auch z.B. eckige Klammen, falls diese im Attributnamen oder als Inhalt genutzt werden.	

Wichtige XML Regeln:

1. Es muss genau ein Wurzelelement geben
2. Alle öffnenden XML Tags müssen wieder geschlossen werden
3. Bevor ein Element geschlossen werden kann, müssen alle beinhalteten Elemente ebenfalls geschlossen sein.
4. Attributnamen müssen pro Element eindeutig sein
5. Attributwerte müssen in doppelte Hochkommata eingeschlossen sein

1.2 XML Path

Wenn Sie mit SAS und XML arbeiten werden Sie früher oder später über ein weiteres wichtiges Konzept aus dem XML Technologie Dschungel stolpern. Der XML Path Standard oder kurz XPath ist eine einheitliche Methode um Elemente oder Attribute in einem XML Baum zu referenzieren [3]. Ein XPath entspricht dabei etwa einem Verzeichnispfad in Ihrem Dateisystem, nur dass Sie damit nicht in Ihren Ordnern und Dateien navigieren sondern eben in den Elementen und Inhalten einer XML Datei. Ein XPath Ausdruck kann dabei aber mehrere Elemente gleichzeitig referenzieren und ganze Teilbäume (alles was von einem Element eingeschlossen wird) enthalten.

Beispiele für XPath Ausdrücke zu der XML Datei in Abbildung 1:

`/einkaufsliste/artikel`
Bezieht sich auf alle „Artikel“ Elemente

`/einkaufsliste/artikel@Name`
Bezieht sich auf den Wert des Attributs „Name“ in den „Artikel“ Elementen

`/einkaufsliste/artikel[@Name="Butter"]`
Bezieht sich auf alle „Artikel“ Elemente, deren Attribut „Name“ den Wert „Butter“ hat

Der XPath Standard ist sehr mächtig und wurde hier nur rudimentär beschrieben, insbesondere da SAS selbst den XPath Standard nur rudimentär unterstützt und nutzt. (Ausnahme: PROC XSL)

2 SAS XML Technologien

Die Herausforderung in der Arbeit mit XML und SAS besteht darin, die baumartige XML Struktur in eine tabellarische Form zu klopfen bzw. umgekehrt, eine zweidimensionale SAS Tabelle in eine hierarchische XML Struktur zu entfalten (Das gilt sicher nicht nur für SAS sondern für alle Bereiche in denen mit Tabellen gearbeitet wird). SAS hat hierfür eine Reihe von Technologien mit jeweils unterschiedlicher Zielsetzung und Anwendungskomplexität:

1. SXLE - SAS XML Library Engine
2. SAS XMLMap - SAS XML Mapping
3. ODS Tagsets - Templates des SAS Output Delivery Systems
4. PROC XSL - Prozedur für XML Transformationen

Die verschiedenen Technologien werden im Folgenden vorgestellt und kurz bewertet. Der Schwerpunkt liegt dabei auf den Methoden 1. und 2., dem Grundgerüst der SAS XML Verarbeitung. Die Methoden 3. Und 4. sind für fortgeschrittene Ansprüche.

2.1 XML Library Engine

Die SAS XML Library Engine oder kurz SXLE bildet das Fundament der XML Verarbeitung mit SAS. Mit SXLE können XML Dateien schnell und einfach eingelesen und ausgegeben werden. Die Programmlogik ist minimal und dank des Engine-Konzepts weitgehend transparent für den Entwickler. Die über das Libname Statement Eingebundenen XML Dateien werden wie eine normale SAS Datei verwendet.

Beispiel 1: XML Dateien mit SXLE ausgeben – SAS Code

```
libname XMLBib xml "C:\xml\schueler.xml";

data XMLBib.schueler;

  set sashelp.class (obs=2 keep=name age);

run;
```

In Beispiel 1 wird die SAS Tabelle `sashelp.class` vorgefiltert. Das Ergebnis wird in die XML Datei `schueler.xml` geschrieben. Das Libname Statement unterscheidet sich nur an zwei Stellen von herkömmlichen Libname Statements:

1. Die Engine `xml` wird definiert

2. Die Pfadangabe `"C:\xml\schueler.xml"` verweist auf eine Datei, nicht auf einen Ordner wie man es von SAS Bibliotheken kennt.

Beispiel 1: XML Dateien mit SXLE ausgeben – XML Ergebnis

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
   <SCHUELER>                          <!-- Zeile 1 Anfang -->
      <Name> Alfred </Name>            <!-- Spalte 1       -->
      <Age> 14 </Age>                  <!-- Spalte 2       -->
   </SCHUELER>                         <!-- Zeile 2 Ende   -->
   <SCHUELER>
      <Name> Alice </Name>
      <Age> 13 </Age>
   </SCHUELER>
</TABLE>
```

Die resultierende XML Datei ist sehr einfach aufgebaut. Das Wurzelelement `<TABLE>` umschließt die einzelnen Datenzeilen `<SCHULER>` die wiederum die einzelnen Spalten enthalten. Alle XML Dateien die mit der SXLE ausgegeben werden unterliegen diesem Schema: Wurzel enthält Zeilenelemente, Zeilenelemente enthalten Spaltenwerte.

Beispiel 2: XML Dateien mit SXLE einlesen – SAS Code

```
libname XMLBib xml "C:\xml\schueler.xml";

data work.schueler;

  set XMLBib.schueler;

run;
```

Der Programmcode um Dateien einzulesen unterscheidet sich, bis auf die verdrehten Rollen im Datastep, nicht vom Ausgabecode. Wichtig: Der XML Inhalt muss so aufgebaut sein wie in Beispiel 1 beschrieben. XML Dateien die von der Struktur abweichen, z.B. tiefer verschachtelt sind, können mit den SXLE Standardeinstellungen nicht verarbeitet werden.

2.2 XMLMap

XMLMaps erweitern die SXLE und ermöglichen es, auch komplexer verschachtelte XML Dateien zu verarbeiten. Basis hierfür ist die sogenannte XMLMap, die beschreibt wie XML Elemente auf SAS Spaltennamen abzubilden sind. SAS bietet Ihnen mit dem frei verfügbaren XML Mapper ein graphisches Tool um XMLMaps zu generieren. Der XML Mapper wird hier nicht vorgestellt.

Beipspiel 3: XML Dateien mit XMLMaps einlesen – SAS Code

```
filename XMLIn "C:\xml\schulverzeichnis.xml";
filename XMLMap "C:\xmlmap\schulverzeichnis.map";

libname XMLIn xml xmlmap=XMLMap;

data work.schueler;
   set XMLIn.schueler;
run;
```

Der Einlesecode unterscheidet sich von der SXLE Variante nur in zwei Punkten:

1. Es werden Filename Referenzen genutzt. Sowohl auf die XML-Datei als auch auf die verwendete XMLMap

2. Die Filename Referenzen werden im Libname Statement wieder verwendet. Über die Libname Option `xmlmap=` wird die XMLMap für die Verarbeitung eingebunden.

Beispiel 3: XML Dateien mit XMLMaps einlesen – XML Datei

```
<SCHULE>
   <KLASSE nummer="5A">
        <Schuelerliste>
            <Schueler name="Alfred">
                <Geschlecht> M </Geschlecht>
                <Alter> 14 </Alter>
            </Schueler>
            <Schueler name="Alice">
                <Geschlecht> F </Geschlecht>
                <Alter> 13 </Alter>
            </Schueler>
        </Schuelerliste>
    </KLASSE>
</SCHULE>
```

Beipspiel 3: XML Dateien mit XMLMaps einlesen – XMLMap

```
<SXLEMAP version="1.2">
   <TABLE name="SCHUELER">

         <TABLE-PATH syntax="XPATH">
          /SCHULE/KLASSE/Schuelerliste/Schueler
         </TABLE-PATH>
         <COLUMN name="NAME">
             <PATH>
               /SCHULE/KLASSE/Schuelerliste/Schueler@name
            </PATH>
            <TYPE>character</TYPE>
            <DATATYPE>STRING</DATATYPE>
            <LENGTH>8</LENGTH>
         </COLUMN>
         <COLUMN name="KLASSE" retain="YES">
           <PATH>
              /SCHULE/KLASSE@nummer
            </PATH>
            <TYPE>character</TYPE>
            <DATATYPE>STRING</DATATYPE>
            <LENGTH>2</LENGTH>
         </COLUMN>

  </TABLE>
</SXLEMAP>
```

Das obige Beispiel liest den Namen und die Klasse von Schüler aus einer XML Datei in eine SAS Tabelle. Besondere Herausforderung: Die Klasse ist jeweils nur einmal im XML Dokument aufgeführt, danach können mehrere Schüler folgen (Klasse 5A .. Schüler1 .. Schüler 2). XMLMaps nutzen die XPath Syntax, unterstützen aber nur einen kleinen Teil des Standards.

Tabelle 2: XMLMap Bestandteile

<SXLEMAP version="1.2">
Wurzelelement für XMLMaps mit Versionangabe
<TABLE name="SCHUELER">
Aus einer XML Datei können mehrere SAS Tabellen erzeugt werden. Für jede SAS Tabelle wird ein <TABLE> mit dem Namen der Tabelle spezifiert.
<TABLE-PATH syntax="XPATH"> **/SCHULE/KLASSE/Schuelerliste/Schueler**
Der Table-Path ist die Kernkomponente der XMLMap. Der Pfad entspricht der XPath Syntax in Abschnitt 1.2 und bestimmt für welche XML Elemente eine Ausgabe erzeugt werden soll. In diesem Beispiel soll für jedes Schueler Element eine Ausgabe erzeugt werden.
<COLUMN name="NAME">
Die Column Elemente spezifzieren die Ausgabespalten. Für jede Ausgabespalte muss ein Column Element vorhanden sein.
<PATH> **/SCHULE/KLASSE/Schuelerliste/Schueler@name**
Die Path Angabe innerhalb des Column Elements entspricht wieder der XPath Syntax. Sie zeigt an welcher XML Inhalt in eine Spalte ausgegeben werden soll. In diesem Fall wird der Wert des Attributs „name“ im Element „Schueler“ verwendet um die Spalte „NAME“ zu füllen.
<COLUMN name="KLASSE" retain="YES">
Wenn sich ein XML Element seltener wiederholt, als die Elemente des Table-Path, müssen diese mit dem Attribut/Wert Paar retain="YES" versehen werden.
<TYPE>character</TYPE> **<DATATYPE>STRING</DATATYPE>** **<LENGTH>2</LENGTH>**
Für jede Spalte müssen zusätzlich deren Typ in der XML-Datei, sowie der Datentyp und die Datenlänge in der SAS Datei angegeben werden

Beispiel 4: XML Dateien mit XMLMaps ausgeben – XMLMap

```
<OUTPUT>
   <HEADING>
       <ATTRIBUTE name="description" value="Schueler" />
   </HEADING>
   <TABLEREF name="SCHUELER" />
</OUTPUT>
```

Eine XMLMap kann auch genutzt werden um XML Dateien auszugeben. Erweitern Sie dazu Ihre XMLMap um ein OUTPUT Element. Das OUTPUT Element spezifiziert welches TABLE Element in der XMLMap für die Ausgabe herangezogen werden soll. In diesem Beispiel erfolgt die Ausgabe wie die Eingabe über die TABLE SCHUELER.

Pro XMLMap kann es nur genau ein OUTPUT Element geben. Es ist aber möglich mehre Tabellen einzulesen. Dazu müssen nur mehrere TABLE Elemente in derselben XMLMap spezifiziert werden.

Beispiel 4: XML Dateien mit XMLMaps ausgeben – SAS Code

```
libname XMLOut xml92 xmlmap=XMLMap;
```

Um eine Datei mit XMLMaps auszugeben muss das Libname Statement mit der SAS 9.2 Engine xml92 definiert werden.

2.3 ODS Tagsets

Das Output Delivery System von SAS ist ein sehr mächtiges Werkzeug um Ausgaben aller Art zu erzeugen. SAS selbst nutzt es um zum Beispiel HTML und PDF Ausgaben aus Reporting Funktionen zu generieren. Eine Spielart der ODS Ausgabe sind die sogenannten Tagsets. Mittels vordefinierten oder eigenen Tagsets können Sie XML Dateien mit SAS erstellen, aber nicht einlesen. Ein ODS Tagsets ist mächtiger als SXLE und XMLMap und kann XML Dateien erzeugen bei denen die anderen beiden Verfahren an Ihre Grenzen stoßen.

Beispiel 5: XML Datei mit eigenem ODS Tagset ausgeben – SAS Code

```
filename XMLout "C:\xml\class_with_custom_tagset.xml";

libname  XMLout xml xmltype=GENERIC tagset=tagsets.custom;

data XMLout.class;
   set sashelp.class;
run;
```

Eine Vorlage für ein eigenes Tagset finden Sie im SAS User's Guide für die SAS 9.2 XML LIBNAME Engine [1]. Wenn Sie ein eigenes Tagset erstellen geben Sie Ihrem Tagset einen Namen, im obigen Beispiel „custom". Das Tagset können Sie dann über die Libname Option `tagset=tagset.custom` an die XML-Datei binden.

Ein ODS Tagset von Grund auf zu erstellen ist keine leichte Sache. Falls Sie noch keine Erfahrung mit anderen ODS Templates haben, wird Ihnen der Einstieg wahrscheinlich schwer fallen.

2.4 PROC XSL

Mit SAS 9.2 wurde die Prozedur XSL eingeführt. Die XSL Prozedur ist derzeit noch im Status „Preproduction", d.h. wenn Sie die Prozedur einsetzen übernimmt SAS derzeit noch keine Verantwortung für eventuelle Fehler und Probleme.

Im Gegensatz zu den bisher vorgestellten Technologien nutzt PROC XSL offene Standards um ein oder mehrere XML Eingabedateien in ein oder mehrere Ausgabedateien umzuwandeln. Die Ausgabe kann jede Form annehmen, dieser Abschnitt konzentriert sich aber auf die Erstellung von anderen XML Dateien. Für die Arbeit mit SAS bietet es sich z.B. an komplexe XML Dateien zuerst mit PROC XSL in einfache XML Dateien umzuwandeln, die anschließend mit der SXLE eingelesen werden können.

Beispiel 6: XML Dateien mit PROC XSL umwandeln – SAS Code

```
PROC XSL IN="C:\xslt\schueler.xml"
         OUT="C:\xslt\personen.xml"
         XSL="C:\xslt\combine.xsl";
RUN;
```

PROC XSL erwartet drei Parameter: Eine XML Datei die transformiert werden soll (IN), eine Ausgabedatei für das Transformationsergebnis (OUT) und eine Datei mit den Transformationsvorschriften (XSL). XSL steht für Extensible Stylesheet Language. In einer XSL Datei sind Transformationen enthalten die dem XSLT Standard genügen (Extensible Stylesheet Language - Transformation).

Auf der nächsten Seite sehen ein Beispiel für eine XSL Datei. Die XSLT Syntax ist sehr mächtig und kann in diesem Beitrag nicht erklärt werden. Im Gegensatz zu ODS Tagsets finden Sie im Internet und in Publikationen ausführliche Beschreibungen und Praxisbeispiele. Desweiteren gibt es viele Werkzeuge die Sie bei der Erstellung von XSL Dateien unterstützen. Von den vier Beschriebene SAS Methoden zur XML Verarbeitung ist dies die mächtigste Variante, wenn auch mit dem Nachteil nicht direkt in eine SAS Tabelle laden oder aus einer SAS Tabelle lesen zu können.

Beispiel 6: XML Dateien mit PROC XSL umwandeln – XSL Datei

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="ISO-8859-1" />

  <xsl:template match="/">
     <xsl:element name="TABLE">

        <xsl:for-each select="KLASSE/schueler">

          <xsl:element name="person">
             <xsl:element name="name">
                <xsl:value-of select="Name"/>
             </xsl:element>

             <xsl:element name="alter">
               <xsl:value-of select="Alter"/>
            </xsl:element>
          </xsl:element>

        </xsl:for-each>

     </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Im obigen Beispiel wird eine XML Datei erstellt, die dem einfachen Tabelle-Zeile-Spalte Konzept entspricht, das für eine Verarbeitung mit SXLE notwendig ist.

3 Vergleich der SAS XML Technologien

Die bisher vorgestellten SAS XML Technologien unterscheiden sich in vielen Faktoren von einander. In der Reihenfolge wie schwer es ist die Technik anzuwenden steht SXLE als einfachste Methode ganz vorne, gefolgt von den noch relativ einfachen XMLMaps. PROC XSL und die damit verbundene XSL Datei rangiert zwischen mittelschwer bis schwer, je nach Anwendungsfall. ODS Tagsets sind auch bei einfachen Anwendungen als schwer zu bewerten.

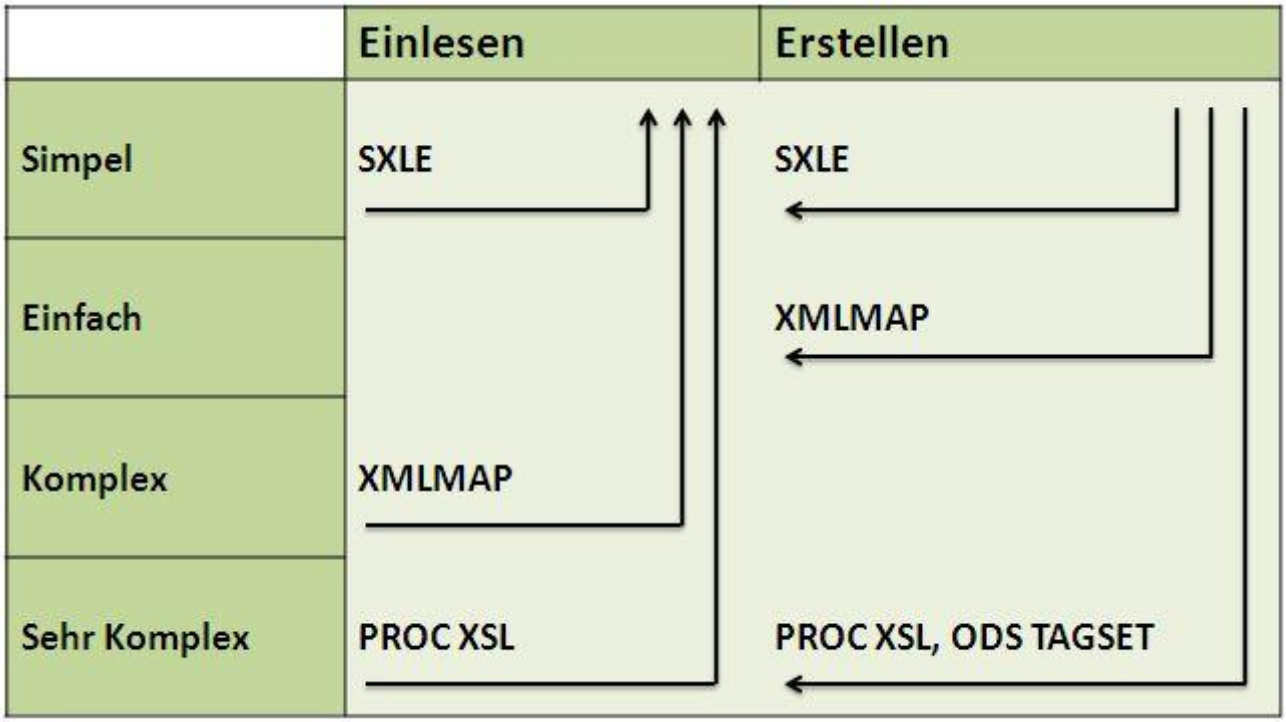

Abbildung 2: XML Komplexität. Was kann ich mit den Methoden verarbeiten

Neben der Anwendungskomplexität ist natürlich auch wichtig, welche XML Strukturen mit der jeweiligen Technik noch eingelesen oder erstellt werden können. Abbildung 2 enthält dazu eine Übersicht. Zum Beispiel können mit XMLMaps komplexe XML Dateien noch eingelesen aber nicht mehr ohne weiteres erstellt werden.

Auf der nächsten Seite finden Sie in Abbildung 3 die Ausgabeflexibilät der einzelnen Methoden beschrieben. Nicht jede Methode kann mehrere XML Dateien bzw. SAS Tabellen gleichzeitig verarbeiten oder ausgeben. Die Pfeile und deren Beschriftung zeigen, welche Technologien ein oder mehrere Tabellen bzw. XML Dateien in einander umwandeln können. Zum Beispiel ist es mit XMLMaps möglich einzelne Tabellen in einzelne XML Dateien umzuwandeln und umgekehrt. XMLMaps können darüber hinaus aus einzelnen XML Dateien mehrere SAS Tabellen erzeugen.

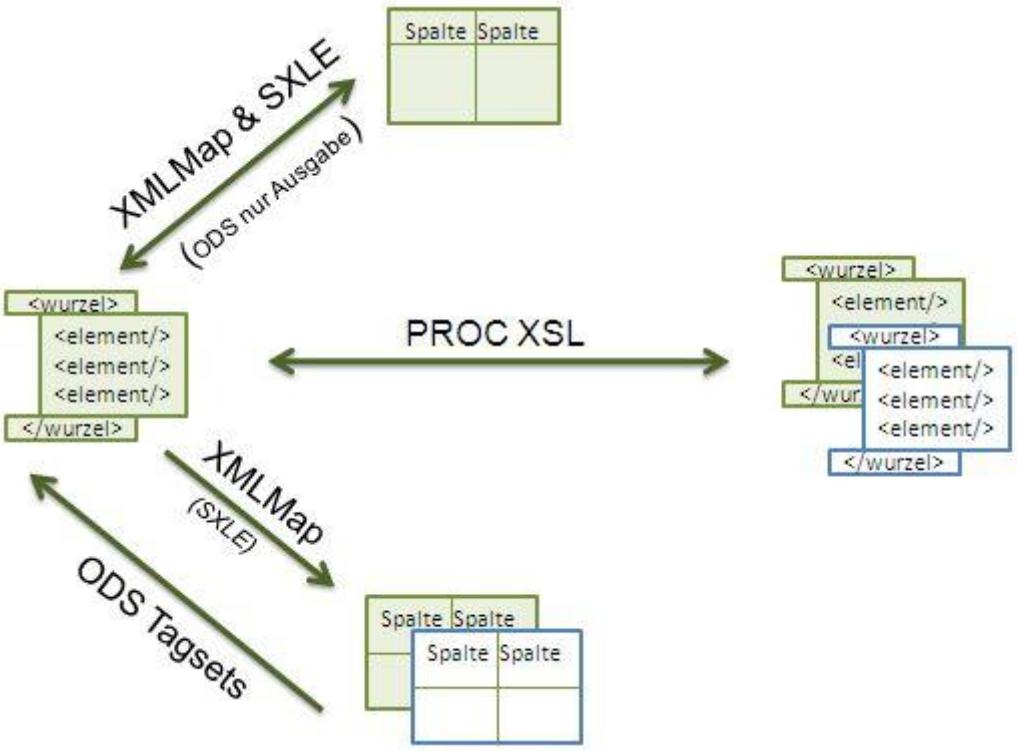

Abbildung 3: Flexibilität in der Anzahl von Ein-/Ausgabedateien und Tabellen

4 Fazit

SAS macht es einem oft leicht mit XML zu arbeiten, aber nicht immer. Die einfachste Lösung ist sicher die SAS XML Libname Engine. Sie sollte immer genutzt werden wenn einfache XML Dateien erzeugt werden müssen - ohne weitere Ansprüche an den Aufbau der XML Datei. Der Anwendungsbereich ist aber eher überschaubar.

Relativ mächtig und mit ein bisschen Übung und Werkzeugunterstützung auch relativ einfach zu erstellen, sind die XMLMaps. Speziell beim einlesen von XML Dateien in mehrere Tabellen zeigen XMLMaps Ihre wahre Stärke. Leider kann es auch bei XMLMaps sehr schnell passieren, dass gängige XML Standards nicht abbildbar sind, insbesondere wenn XML erstellt werden soll. Ursächlich hierfür ist unter anderem die nur rudimentäre Unterstützung der XPath Technologie.

ODS Tagsets sind aufgrund Ihrer hohen Komplexität und geringen Flexibilität nur eingefleischten ODS Experten zu empfehlen oder solchen, die es noch werden wollen. Es gibt keine Werkzeugunterstützung und die Dokumentation des SAS Inhouse Standards ist eher schwach.

Für gehobene Ansprüche im Umgang mit XML empfiehlt sich ein Blick auf PROC XSL. XSLT ist zwar wie ODS Tagsets durchaus komplex, es gibt aber viele Werkzeuge und Literatur zu dem Thema. Zudem kann XSL Wissen auch in anderen Technologien angewandt werden. Der „Preproduction“ Status in SAS 9.2 und der doppelte Weg zum finalen SAS Dataset sind derzeit die größten Nachteile dieser Technologie.

Literatur

[1] SAS Institute: SAS® 9.2 XML LIBNAME Engine User's Guide, Second Edition. SAS® Publishing, Cary, 2010

[2] W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C 2008

[3] K. H. Goldberg: XML: Visual QuickStart Guide (2nd Edition), Peachpit Press