Einlesen von Excel-Dateien mit SAS Base

Christoph Klein
Landesbank Baden-Württemberg
Am Hauptbahnhof 2
70173 Stuttgart
Christoph.Klein@LBBW.de

Steffen Melang
Landesbank Baden-Württemberg
Am Hauptbahnhof 2
70173 Stuttgart
Steffen.Melang@LBBW.de

Zusammenfassung

Immer noch erfolgen viele Datenlieferungen per Excel-Weitwurf. Um die Daten in SAS verwenden zu können, muss man sie zunächst einlesen. Ein Weg dazu führt in SAS Base über Proc XSL und die XML Libname Engine, da aktuelle Excel-Dateien XML-Dokumente sind. Wir zeigen, wie wir uns damit durch die Datenstruktur einer Excel-Datei navigieren und Folgendes auslesen: Workbook, Namen, Shared Strings sowie Zellkoordinaten und -inhalte.

Schlüsselwörter: Excel-Import, XML, PROC XSL

1 Einleitung

Betreibt ein Dienstleister oder Geschäftspartner kein Datenbanksystem, liefert er Daten oft als Excel-Datei, die man dann mit SAS einlesen muss. Dazu bietet SAS mehrere Möglichkeiten. Eine wenig bekannte Variante in SAS Base nutzt aus, dass Excel-Dateien XML-Dokumente sind. Wir erläutern Teile ihres Aufbaus und zeigen, wie wir sie mit XML-Werkzeugen von SAS einlesen, wenn man z.B. SAS/ACCESS to PC Files im Windows-Betriebssystem nicht zur Verfügung hat.

2 Einführung XML

Eine kurze Einführung in XML gibt [5]. Wir fassen diese Zusammenfassung nochmal kurz zusammen.

2.1 XML

XML steht für Extensible Markup Language. In einem XML-Schema wird die Struktur definiert, mit denen Daten in XML-Dateien gespeichert werden. Dies ist sehr flexibel möglich. Daher ist XML die Grundlage für sehr viele IT-Produkte und elektronische Dokumente. Wir geben ein kurzes, einfaches Beispiel einer XML-Datei ohne das XML-Schema anzugeben:

```
<?xml version="1.0">
<Charts>
     <Eintrag Platz="1">
          <Titel>
               Verdammt ich lieb' dich
          <Titel/>
          <Interpret>
               Matthias Reim
          <Interpret/>
     </Eintrag>
     <Eintrag Platz="2">
          <Titel>
               Nothing Compares 2 U
          <Titel/>
          <Interpret>
               Sinead O'Connor
          <Interpret/>
     </Eintrag>
</Charts>
```

Typisch sind wie bei HTML auch die öffnenden und schließenden Tags in spitzen Klammern. Sie können tief verschachtelt sein.

2.2 XML Libname Engine

Diese SAS-Engine kann über das libname-Statement angesprochen werden und ist eine Möglichkeit, XML-Dateien einzulesen und auch zu schreiben. Für komplexe Fälle benötigt SAS dazu eine XML-Map, die erläutert, wie aus den XML-Tags Tabellen erstellt werden sollen. Wir geben ein kurzes, einfaches Beispiel:

```
libname chartlib XML xmlfileref = charts xmlmap = "charts2lib.xml";

data hitparade;
     set chartlib.eintrag;
run;
```

Damit wird die Bibliothek chartlib angelegt, die die Fileref charts einliest und die XML-Map charts2lib nutzt, um die XML-Daten darin zu interpretieren. Wichtig ist die Angabe von XML, da damit die XML Libname Engine angesprochen wird.

2.3 Proc XSL

Die Prozedur ist seit SAS 9.3 im Status „Production“, davor war sie im Status „Preproduction“. Sie transformiert komplexe XML-Strukturen in andere Formate, z. B. einfache XML-Strukturen, die dann mit der XML Libname Engine eingelesen werden können. Dazu benutzt sie die Extensible Stylesheet Language, die ihr auch den Namen gab. Diese Sprache ist sehr mächtig und geht weit über den Umfang dieses Beitrags hinaus. Wir begnügen uns mit einem kurzen Beispiel:

```
proc xsl
     in   = "eingabe.xml"
     xsl  = "transformationsvorschrift.xslt"
     out  = "ausgabe.xml"
;
run;
```

Die drei Parameter in, xsl und out sind Pflichtangaben.

3 Aufbau einer Excel-Datei

Der Aufbau einer Excel-Datei mit der Endung „xlsx“ orientiert sich seit 2007 am ISO-Standard Office Open XML[1]. Dessen Dokumentation ist sehr umfangreich, wir konzentrieren uns auf wenige Punkte. Für Dateien mit der Endung „xlsm“ funktioniert unser Verfahren auch.
Die Daten liegen als XML vor und sind gezippt, zusammen mit weiteren Objekten. Ändert man die Endung in „zip“, erhält man unter Windows eine Übersicht über die Elemente der Datei. Abbildung 1 verdeutlicht dies anhand eines Beispiels. Im Verzeichnis /xl/ verbergen sich die Elemente der Excel-Datei; es sind vor allem XMLs, die man zum Beispiel mit dem Internet Explorer öffnen kann. Einige wichtige Element sind in Tabelle 1 aufgelistet.

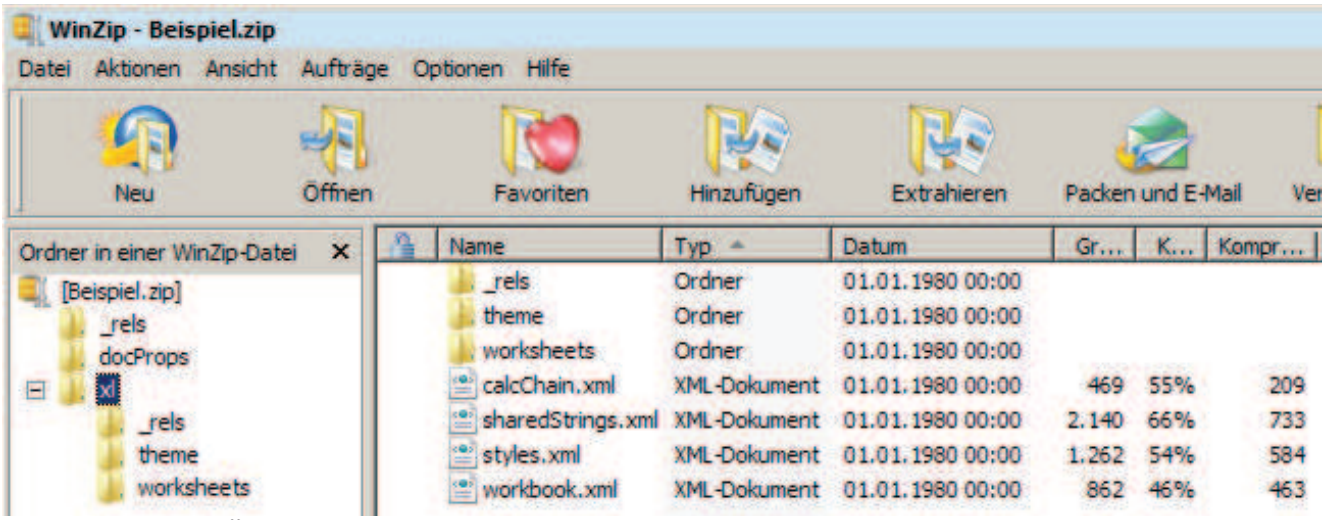

Abbildung 1: Öffnen einer Excel-Datei mit WinZip

Tabelle 1: wichtige Elemente einer Excel-Datei

Element	Erläuterung
Workbook	Die Datei /xl/workbook.xml ist die Kopfdatei. In ihr sind unter anderem alle Blätter und Namen[2] der Excel-Datei aufgelistet. Ein Blatt (sheet) ist nicht mit dem Dateinamen gespeichert, sondern mit einem Identifier (rID).

[1] ISO/IEC 29500, niedergelegt in [1], [2], [3] und [4]
[2] Excel ermöglicht es, Zellen und Zellbereiche nicht nur über die Koordinaten R17 oder R23:T42 anzusprechen, sondern auch über individuelle Namen wie „Mein_Name_1“. Der Name wird in Excel links oben im Namenfeld angezeigt.

Relationships	Die Datei /xl/_rels/workbook.xml.rels ordnet einer rID den Dateinamen der XML-Datei des Blatts zu.
Shared Strings	Die Datei /xl/sharedStrings.xml enthält die alpha-numerischen Zellinhalte der Excel-Datei. Sie sind nicht in der XML-Datei des Blatts gespeichert, sondern separat, um Platz zu sparen, falls ein String mehrmals verwendet wird. Sie werden über ihre Nummer angesprochen, die sich aus ihrer Reihenfolge ergibt und bei 0 beginnt.
Zellkoordinaten	Beispiele: A1, BD15
Zellinhalte	Zahlen Texte Formeln …

Auszug aus dem Workbook einer Beispieldatei:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<workbook [...]>
     [...]
     <sheets>
          <sheet r:id="rId1" sheetId="3" name="Charts"/>
          <sheet r:id="rId2" sheetId="2" name="Bundesliga"/>
     </sheets>
     <definedNames>
          <definedName name="Differenz">
               Bundesliga!$I$3:$I$20
          </definedName>
          <definedName name="Interpret">
               Charts!$C$2:$C$11
          </definedName>
          <definedName name="Titel">
               #Ref!
          </definedName>
     </definedNames>
     [...]
</workbook>
```

Im Beispiel kann der Name "Titel" nicht aufgelöst werden, weil die Zelle bzw. der Bereich gelöscht wurde.

Auszug aus den Relationships einer Beispieldatei:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<Relationships […]>
     [...]
     <Relationship Target="worksheets/sheet2.xml" […] Id="rId2"/>
     <Relationship Target="worksheets/sheet1.xml" […] Id="rId1"/>
     [...]
</Relationships>
```

Unter Target ist der Pfad zu den XML-Dateien des Blattes zur rID vermerkt.

Auszug aus den SharedStrings einer Beispieldatei:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<sst uniqueCount="56" count="62" […]>
     <si><t>Verdammt ich lieb' dich</t></si>
     <si><t>Matthias Reim</t></si>
     <si><t>Nothing Compares 2 U</t></si>
     <si><t>Sinead O'Connor</t></si>
</sst>
```

Auszug aus dem XML zu einem Blatt einer Beispieldatei:

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<worksheet [...]>
     <dimension ref="A2:L20"/>
     [...]
     <sheetData>
          [...]
          <row r="19" [...]>
               <c r="A19"><v>17</v></c>
               <c r="B19" t="s"><v>21</v></c>
               <c r="C19"><v>34</v></c>
               <c r="D19"><v>8</v></c>
               <c r="E19"><v>7</v></c>
               <c r="F19"><v>19</v></c>
               <c r="G19"><v>43</v>
               </c><c r="H19"><v>88</v></c>
               <c r="I19">
                    <f t="shared" si="0"/>
                    <v>-45</v>
               </c>
               <c r="J19"><v>23</v></c>
               <c r="K19"><v>45</v></c>
               <c r="L19" t="s"><v>27</v></c>
          </row>
          [...]
     </sheetData>
     [...]
</worksheet>
```

Im Auszug ist die Reihe 19 des Blattes dargestellt. Im Feld A19 ist der Wert 17 eingetragen, im Feld B19 der 21. SharedString. In Feld I19 ist eine Formel hinterlegt, die den Wert -45 liefert; die Formel wurde in Zelle I4 definiert und nach unten kopiert. Im Feld I4 ist die Formel hinterlegt:

```
<c r="I4"><f ref="I4:I20" t="shared" si="0">G4-H4</f><v>30</v></c>.
```

4 Einlesen von Excel-Dateien mit PROC XSL

Um ein einzelnes Blatt einzulesen, schälen wir die Excel-Datei wie eine Zwiebel. Die Schritte dazu sind:

1. Sheets und Namen aus dem Workbook einlesen
2. Relationships einlesen und mit Sheets verknüpfen
3. SharedStrings einlesen
4. Blatt einlesen und SharedStrings darin ersetzen

Jeder Schritt davon verwendet eine Routine, die eine XML-Datei der Excel-Datei in eine SAS-Datei entpackt. Wir stellen alle Programmteile nun im Detail vor. Wir empfehlen, am Ende jedes Programmteils die temporären Bibliotheken und Filenames zu löschen.

4.1 Entpacken eines XMLs in eine SAS-Datei

Zuerst legen wir eine temporäre SAS-Datei ztmp_xsl an, die nur ein XML-Tag enthält, nämlich das Kommando zum Entpacken der XML-Datei. Dazu müssen bekannt sein: Pfad und Name der Excel-Datei in der Makrovariablen zipfile und der Name der XML-Datei in der Makrovariablen xmlfile.

```
filename ztmp_xsl temp;

data _null_;
     length line $200;
     file ztmp_xsl;

     line = '<?xml version="1.0" encoding="UTF-8"
               standalone="yes"?>';
     put line;

     line = cats('<filename>jar:file://', &zipfile., "!", &xmlfile.,
               "</filename>");
     put line;
run;
```

Diese SAS-Datei dient als Eingabe für Proc XSL, um das XML zu extrahieren. Dabei verwenden wir die Transformationsvorschrift trans4.xsl. Das Ergebnis liegt in der SAS-Datei &outfile.

```
proc xsl
   in     = ztmp_xsl
   xsl    = "trans4.xsl"
   out    = &outfile.;
run;
```

Die Transformationsvorschrift trans4.xsl gibt an, wie die XML-Datei in eine SAS-Datei umgewandelt wird:

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform
      version="1.0">
     <xsl:output method="xml"/>
     <xsl:template match="/">
          <xsl:variable name="filename" select="/filename"/>
          <xsl:copy-of select="doc($filename)/*"/>
     </xsl:template>
</xsl:stylesheet>
```

Zum Schluss löschen wir die temporäre SAS-Datei wieder:

```
filename ztmp_xsl clear;
```

Dieses Verfahren wird für alle XMLs der Excel-Datei gleich angewendet: Workbook, Relationships und die Blätter selbst. Unterschied ist, dass die Namen der XML-Dateien für Workbook und Relationships im Voraus bekannt sind, die der Blätter jedoch nicht.

4.2 Einlesen des Workbooks

Liegt das Workbook bereits als SAS-Datei tmp_wbk vor, so lesen wir mittels XML Libname Engine Blätter und Namen ein als Datasets tmp_worksheets und xtmp_definedName:

```
libname xcl_wbk XML xmlfileref = tmp_wbk xmlmap = "worksheets.xml";

data tmp_worksheets;
     set xcl_wbk.worksheets;
run;

data xtmp_definedName;
     set xcl_wbk.definedname;
     * Bei Bedarf sortiert man hier fehlerhafte Definitionen aus.;
run;
```

Die verwendete XML-Map worksheet.xml dient dem Einlesen der Blätter und Namen:

```
<SXLEMAP version="1.2">
     <TABLE name="Worksheets">
          <TABLE-PATH syntax="XPATH">
               /workbook/sheets/sheet
          </TABLE-PATH>

          <COLUMN name="name">
               <PATH>/workbook/sheets/sheet@name</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>
```

```
          <COLUMN name="rid">
               <PATH>/workbook/sheets/sheet@r:id</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>
     </TABLE>

     <TABLE name="definedName">
          <TABLE-PATH syntax="XPATH">
               /workbook/definedNames/definedName
          </TABLE-PATH>

          <COLUMN name="name">
               <PATH>/workbook/definedNames/definedName@name</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>

          <COLUMN name="cell">
               <PATH>/workbook/definedNames/definedName</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>
     </TABLE>
</SXLEMAP>
```

Es wird der Aufbau der beiden Tabelle Worksheets und definedName definiert, beide haben zwei Spalten, die in den column-Tags beschrieben sind.

4.3 Einlesen der Relationships

Liegen die Relationships bereits als SAS-Datei tmp_rid vor, so lesen wir sie mittels XML Libname Engine ein als Dataset temp_rels:

```
libname xcl_rid XML xmlfileref = tmp_rid xmlmap = "rels.xml";

data tmp_rels;
     set xcl_rid.rels;
run;
```

Die verwendete XML-Map rels.xml dient dem Einlesen der Relationships:

```
<SXLEMAP version="1.2">
     <TABLE name="Rels">
          <TABLE-PATH syntax="XPATH">
                    /Relationships/Relationship
          </TABLE-PATH>
```

```
          <COLUMN name="Target">
               <PATH>/Relationships/Relationship@Target</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>

          <COLUMN name="rid">
               <PATH>/Relationships/Relationship@Id</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>
     </TABLE>
</SXLEMAP>
```

Danach verknüpfen wir in SAS über die rID das so erzeugte Dataset tmp_rels mit dem Workbook im Dataset tmp_worksheets (siehe 4.2). Dann kennen wir zu jedem Blatt den Dateinamen der XML-Datei und können diesen später verwenden.

4.4 Einlesen der SharedStrings

Liegen die SharedStrings bereits als SAS-Datei tmp_ss vor, so lesen wir sie mittels XML Libname Engine ein als Dataset ztmp_ss. Ihre Nummer ergibt sich aus ihrer Reihenfolge in der Datei und beginnt bei 0:

```
libname xcl_ss XML xmlfileref = tmp_ss xmlmap =
"shared_strings.xml";

data ztmp_ss;
     set xcl_ss.strings;
     * Nummer des sharedString in Variable _i_;
     _i_ = _n_ - 1;
run;
```

Die verwendete XML-Map shared_strings.xml dient dem Einlesen der SharedStrings:

```
<SXLEMAP version="1.2">
     <TABLE name="strings">
          <TABLE-PATH syntax="XPATH">/sst/si/t</TABLE-PATH>

          <COLUMN name="value">
               <PATH>/sst/si/t</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>2000</LENGTH>
          </COLUMN>
     </TABLE>
</SXLEMAP>
```

4.5 Einlesen eines Blattes

Liegt ein Blatt bereits als SAS-Datei tmp_sht vor, so wird es mittels XML Libname Engine eingelesen als Dataset ztmp_cells:

```
libname xcl_sht XML xmlfileref = tmp_sht xmlmap = "sheet.xml";

data ztmp_cells;
     set xcl_sht.cells (rename=(row=zeile));

     * SharedString sind durch den Typ 's' erkennbar;
     if type = 's' then id = input(value, 20.);

     spalte         = substr(cell, 1, findc(cell, , 'd') - 1);
     spalte_num     = input(spalte, excel_spalte.);
run;
```

Die verwendete XML-Map sheet.xml dient dem Einlesen des Sheets:

```
<SXLEMAP version="1.2">
     <TABLE name="cells">
          <TABLE-PATH syntax="XPATH">
               /worksheet/sheetData/row/c/v
          </TABLE-PATH>

          <COLUMN name="row" retain="yes">
               <PATH>/worksheet/sheetData/row@r</PATH>
               <TYPE>numeric</TYPE>
               <DATATYPE>integer</DATATYPE>
          </COLUMN>

          <COLUMN name="cell">
               <PATH>/worksheet/sheetData/row/c@r</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>50</LENGTH>
          </COLUMN>

          <COLUMN name="value">
               <PATH>/worksheet/sheetData/row/c/v</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>200</LENGTH>
          </COLUMN>

          <COLUMN name="type">
               <PATH>/worksheet/sheetData/row/c@t</PATH>
               <TYPE>character</TYPE>
               <DATATYPE>STRING</DATATYPE>
               <LENGTH>20</LENGTH>
          </COLUMN>
```

```
     </TABLE>
</SXLEMAP>
```

Das Informat Excel_Spalte wandelt den Spaltennamen von Buchstaben wie A oder CF in Zahlen um. Zu seiner Definition übergeben wir das folgende Dataset an Proc Format:

```
data excel_spalte;
     fmtname   = "excel_spalte";
     type      = "I";
     label     = 1;

     do _i_ = '','A','B',[…],'Y','Z';
          do _j_ = 'A','B',[…],'Y','Z';
               start = cats(_i_, _j_);
               output;
               label + 1;
          end;
     end;
run;
```

Das Dataset ztmp_cells hat nun die Variablen Zeile, Spalte, Spalte_num, Value und Type. Von den Formeln wurde nur das Ergebnis eingelesen. Zum Schluss ersetzen wir die SharedStrings mit einer passenden SAS-Technik, verzichten jedoch auf die Darstellung im Beispiel. Damit liegt das Blatt als Dataset in Langform vor, also je Zelle eine Zeile mit den Koordinaten der Zelle und ihrem Inhalt. Die weitere Verarbeitung hängt ganz davon, welche Daten benötigen werden. Zum Beispiel kann man auf Daten aus bestimmten Zellbereichen filtern; dynamisch kann man dazu vorhandene Namen verwenden.

5 Zusammenfassung und Ausblick

In den vorherigen Abschnitten haben wir gezeigt, wie wir ein einzelnes Blatt einlesen, indem wir die Excel-Datei wie eine Zwiebel schälen und uns so zum Inhalt vorarbeiten und diesen einlesen. Wenn nun die gelieferten Excel-Dateien immer gleich strukturiert sind, kann man dieses Vorgehen in Makros kapseln und immer wieder anwenden.
Insgesamt ist das vorgestellte Verfahren recht aufwändig; zuvor lohnt es sich, andere Möglichkeiten zu prüfen wie SAS/ACCESS to PC Files oder DDE oder ODBC.
Auf ähnliche Art und Weise kann man auch andere Office-Dateien[3] und sogar Enterprise Guide-Projekte einlesen, da sie ähnlich aufgebaut sind, aber natürlich andere XML-Schemata verwenden und außerdem auch weitere Dateitypen enthalten. Entpackt sieht ein Beispielprojekt so aus:

[3] Die Informationen liegen in der zip-Datei eines Word-Dokuments im Ordner /word/ und in der zip-Datei einer Powerpoint-Präsentation im Ordner /ppt/.

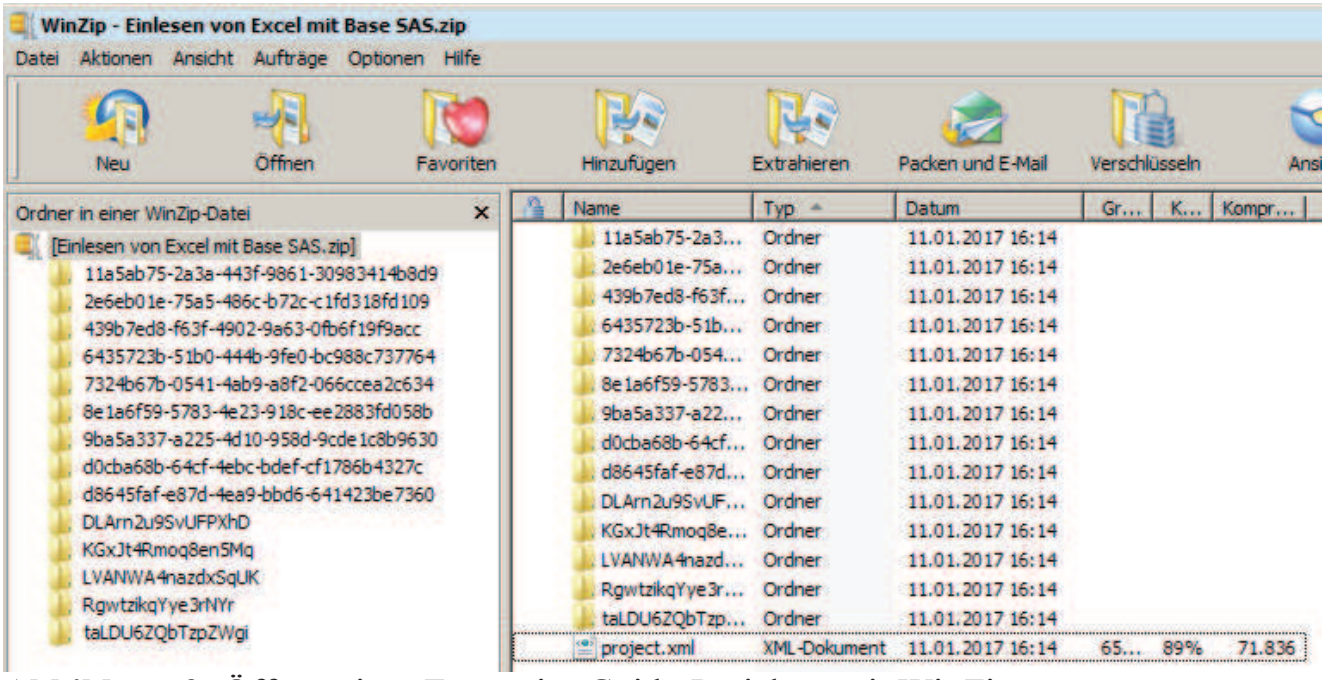

Abbildung 2: Öffnen eines Enterprise Guide-Projektes mit WinZip

In project.xml finden wir die Angaben zum Projekt. Zum Codeblock „Auswertung“ ist dort unter anderem folgendes vermerkt:

```
<Element>
     <Label>Auswertung</Label>
     <Type>TASK</Type>
     <Container>KkCYwgy5kbdI4rW6</Container>
     <ID>RgwtzikqYye3rNYr</ID>
     <CreatedOn>635392225061303747</CreatedOn>
     <ModifiedOn>636080787298726591</ModifiedOn>
     <ModifiedBy>Melang, Steffen</ModifiedBy>
     <ModifiedByEGID>ABCDE</ModifiedByEGID>
     <ModifiedByEGVer>5.100.0.14335</ModifiedByEGVer>
     <HasSerializationError>False</HasSerializationError>
</Element>
```

Der Tag ID verweist auf den Ordner, in dem der Code liegt; in dessen Unterordnern liegen das Log und die Ausgaben.

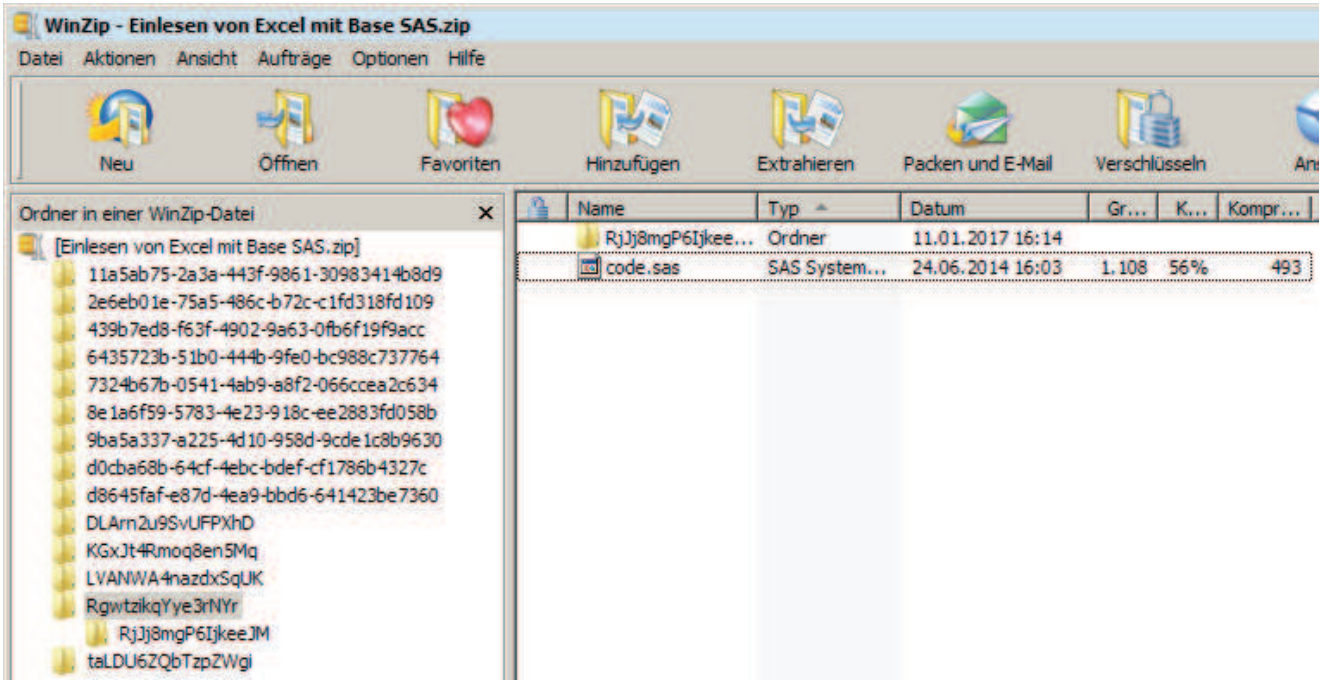

Abbildung 3: Öffnen eines Unterordners eines Enterprise Guide-Projektes mit WinZip

Literatur

[1] ISO 29500-1: Information technology — Document description and processing languages — Office Open XML File Formats — Part 1. International Organization for Standardization, Genf 2016. Verwendete Version heruntergeladen am 09.01.2017 von http://www.iso.org/iso/home/store/catalogue_ics.htm.

[2] ISO 29500-2: Information technology — Document description and processing languages — Office Open XML File Formats — Part 2. International Organization for Standardization, Genf 2012. Verwendete Version heruntergeladen am 09.01.2017 von http://www.iso.org/iso/home/store/catalogue_ics.htm.

[3] ISO 29500-3: Information technology — Document description and processing languages — Office Open XML File Formats — Part 3. International Organization for Standardization, Genf 2015. Verwendete Version heruntergeladen am 09.01.2017 von http://www.iso.org/iso/home/store/catalogue_ics.htm.

[4] ISO 29500-4: Information technology — Document description and processing languages — Office Open XML File Formats — Part 4. International Organization for Standardization, Genf 2016. Verwendete Version heruntergeladen am 09.01.2017 von http://www.iso.org/iso/home/store/catalogue_ics.htm.

[5] A. Adlichhammer: XML mit SAS leicht gemacht. Erschienen in C. Ortseifen, H. Ramroth, M. Weires, R. Minkenberg (Hrsg.): Proceedings der 15. KSFE Heidelberg. Shaker-Verlag, Aachen 2011.