Standardisierte EXCEL-Berichte mit SAS-DDE

Benedikt Muschik
Dibera GmbH
Kolde-Ring 21
48126 Münster
benedikt.muschik@dibera.de

Zusammenfassung

In Unternehmen werden Berichte von Fachabteilungen meist in Form einer formatierten EXCEL-Datei mit dynamischen, interaktiven Excel-Diagrammen, Formeln und VBA-Macros gewünscht, welche die Richtlinien des Corporate-Designs im Unternehmen erfüllen sollen. SAS-Programmierer möchten die Berichte im Sinne der Zentralen Datenhaltung automatisiert aus SAS erzeugen.
Diese EXCEL-Sheets können mit Hilfe der seit langem existenten Schnittstelle „SAS-DDE“ direkt aus SAS befüllt werden. Hierbei werden bis auf SAS-Base unter Windows und EXCEL keine weiteren Hilfsmittel benötigt. Durch diese „alte Methode“ können 100% SAS-Funktionalität mit 100% EXCEL-Funktionalität kombiniert werden.
Im Beitrag wird eine Lösung gezeigt, welche im Rahmen der Basel2-Prüfung zum Einsatz gekommen ist. Die Aufträge für die verschiedenen Berichte werden durch den Fachbereich in eine Steuertabelle eingegeben, in der die Berichte beliebig skaliert werden können. Danach wird ein SAS-Programm gestartet, das als Ergebnis ein EXCEL-Workbook erstellt, welches die in der Steuertabelle angeforderten Berichte enthält.
Somit können alle Funktionen, die im EXCEL bisher durch Formeln oder Makros abgebildet werden, durch eine SAS basierte Lösung ersetzt werden.

Schlüsselwörter: SAS-DDE, MS-Excel

1 Motivation

Im Rahmen des Basel2-Reportings an die BaFin sind in regelmäßigen Abständen Berichte zu erstellen. Zur Erstellung dieser Berichte müssen oft komplizierte Statistiken berechnet werden, für deren Erzeugung SAS sehr gut geeignet ist. Die BaFin prüft, ob die Berichte nach gewissen Vorgaben an die Datenhaltung und Reproduzierbarkeit erzeugt wurden. Ein solches „prüfungsfestes“ Umfeld kann man (nach meinen Erfahrungen) durch den Einsatz von MS-Office Produkten –zur Datenhaltung und Berechnungnicht herstellen.
Die Fachabteilungen, welche die Reports für die BaFin erstellen, wollen die Ergebnisse der Berechnungen jedoch oft in EXCEL sehen, um sie von dort mühelos in die WORD-Berichte an die BaFin oder in andere Unternehmenspräsentationen einzufügen. Hierbei sind für den Fachbereich oft auch interaktive EXCEL-Grafiken wünschenswert, welche in Präsentationen leicht kommentiert bzw. modifiziert werden können.

2 Systemvoraussetzung

Für die Nutzung von SAS-DDE müssen SAS und EXCEL unter WINDOWS vorhanden sein. Die im Folgenden beschriebenen Programme funktionieren nur, wenn SAS und EXCEL auf einem Rechner unter WINDOWS vorhanden sind. Es sind KEINE kostenpflichtigen Zusatzmodule von SAS erforderlich!

3 Einstieg in die SAS-DDE Programmierung

Einen sehr guten Einstieg in die DDE-Programmierung bieten die Arbeiten von Vyvermann [3,4]!

3.1 Dynamic Data Exchange (DDE)

Die folgenden Erklärungen zu DDE wurden aus Wikipedia übernommen:
„Dynamic Data Exchange (engl., Abk. DDE) bedeutet dynamischer Datenaustausch. Es handelt sich um ein Protokoll für den Datenaustausch zwischen verschiedenen Anwendungsprogrammen, also eine Interprozesskommunikation nach dem Client-Server-Modell."

3.1.1 Details

Dieses Protokoll ist lokal und innerhalb von Netzwerken in den Betriebssystemen Windows (ab Version 2.0) und OS/2 verfügbar. Damit Daten ausgetauscht oder Befehle abgesetzt werden können, müssen beide Anwendungsprogramme gleichzeitig laufen. Falls nötig muss der DDE-Client den benötigten Server-Prozess starten. Der Datenaustausch erfolgt grundsätzlich in kompletten Dateneinheiten in Windows-Zwischenablage-Formaten (also auch Binärdaten); kontinuierliche Datenströme werden üblicherweise nicht unterstützt. Da zwei sich kennende Programme neue Zwischenablageformate definieren können, ist auch der Austausch von großen arbiträren Datenmengen, etwa Matrizen in MATLAB, kein Problem. Als Besonderheit bietet DDE ein „Advise" genanntes Hot-Tracking (Datenänderungsbenachrichtigung) an, mit dem der Client automatisch über Veränderungen des Server-Datenbestands informiert wird.
Typisch für DDE und auch festgelegt ist die dreistufige Adressierung von Datenelementen, eingeteilt Server (Dienst), Topic (Thema) und Item (Element). Eine tiefergehende Adressierung, etwa bestimmte Zeilen und Spalten einer Tabellenkalkulationsseite, muss durch ein wahlfreies, nicht-aufzählbares Item erfolgen. Implementiert ist DDE durch Nachrichten über die Windows-typischen Thread-Warteschlangen sowie mittels gemeinsam genutztem Speicher für die eigentlichen Daten.
Die Leistungsfähigkeit von DDE liegt grob bei 1000 Übertragungen pro Sekunde, wenn die Kommunikation über einen GUI-Thread erfolgt, und kann bei Verwendung gesonderter Threads deutlich höher liegen. Die mit Windows 3.1 eingeführten DDE-Funktionen, zunächst in der DDEML.DLL verpackt, sind praktisch nur Wrapper, die die Verwendung vereinfachen, aber keinen Geschwindigkeitsvorteil bringen.

DDE überwindet „Bitgrenzen“ relativ problemlos. So ist es möglich, dass ein 16-Bit- und ein 32-Bit-Windows-Programm kommunizieren, so auch ein 32-Bit- und ein 64-Bit-Programm. Eine Kommunikation zwischen 16-Bit und 64-Bit ist regulär nur mittels Netzwerk-DDE machbar, da die 64-Bit-Windows-Versionen keine 16-Bit-Unterstützung haben.

3.2 Aktueller Stand

DDE findet heute immer noch Verwendung, etwa zum Weiterleiten von Kommandozeilen an bereits laufende Prozesse, typischerweise an MDI-Anwendungen. So kann der Windows-Explorer bei geeignet eingestellten Verknüpfungen ein Dokument an einen bereits laufenden Bearbeitungsprozess „senden“, ohne eine unnötige Prozess-Kopie zu erstellen.
Auf Grund seiner Einfachheit ist Netzwerk-DDE bei Maschinen- und Fabriksteuerungen verbreitet. DDE wurde durch das OLE-Protokoll ergänzt und erweitert.” [1]

3.3 Notwendige SAS-Optionen, starten von EXCEL

```
options noxwait noxsync;
```

In SAS sollten die beiden Optionen noxwait und noxsync gesetzt werden. Diese beiden Optionen werden in Zusammenhang mit der X-Anweisung in SAS benötigt und sind in der SAS-Dokumentation näher erläutert:
„**NOXSYNC:** specifies that the operating system command execute asynchronously with your SAS session. That is, control is returned immediately to SAS and the command continues executing without interfering with your SAS session. With NOXSYNC in effect, you can execute an X command or X statement and return to your SAS session without closing the process spawned by the X command or X statement.“ [2]
„**NOXWAIT:** specifies that the command processor automatically returns to the SAS session after the specified command is executed. You do not have to type EXIT.“ [2]
Bevor Daten von SAS nach Excel übergeben werden können muss EXCEL gestartet werden. Der Start eines existierenden EXCEL-Workbooks über ein SAS-Programm kann beispielsweise über die folgende Anweisung vorgenommen werden:

```
X "C:\Daten\dde\dde.xlsm";
```

3.4 Export von SAS-Daten nach EXCEL

3.4.1 Filename-Anweisung zum Datenaustausch

Für den Export von SAS-Daten nach EXCEL benötigt man als erstes eine Filename-Anweisung:

```
filename <fileref> dde '<server app>|<topic>!<item>';
```

Die Filename-Anweisung beinhaltet nach der Filereferenz den String „dde“ und danach das DDE-Triplet. Das DDE-Triplet besteht aus dem Namen des Servers, also in diesem Beispiel Excel, dem sogenannten Topic, welcher sich für EXCEL aus dem Namen des

Workbooks und des Worksheets zusammensetzt und dem sogenannten Item, welches in diesem Beispiel die Zellbezüge in Excel wiedergibt.

3.4.2 Bestimmung des DDE – Triplets

Die Bestimmung des DDE-Triplets für die Filename-Anweisung ist relativ schwer. Deshalb gibt es hier von SAS eine Hilfestellung und man kann sich ein DDE-Triplet (im Beispiel für EXCEL) über SAS anzeigen lassen. Dabei geht man wie folgt vor:

1. Ausgabebereich in Excel markieren und „STRG+C“ drücken:

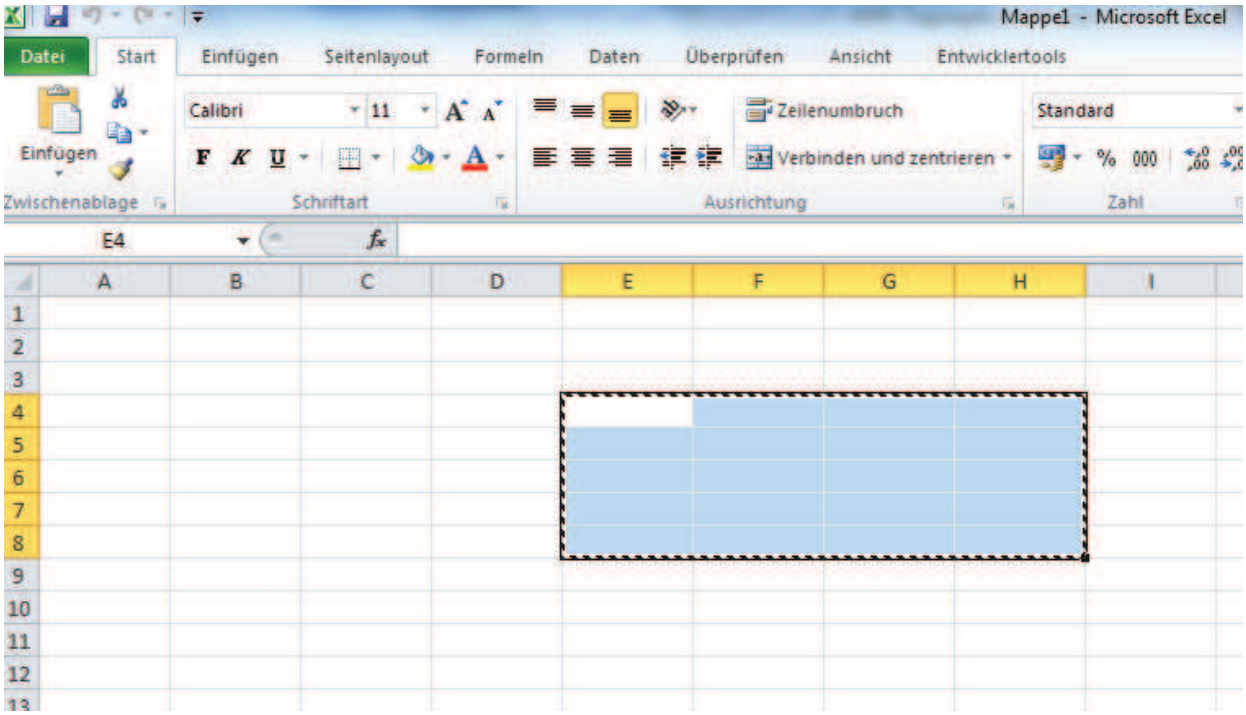


Abbildung 1: Ausgabebereich in Excel markieren

2. SAS öffnen und SOLUTIONS/ACCESSOIRES/DDE Triplet anwählen

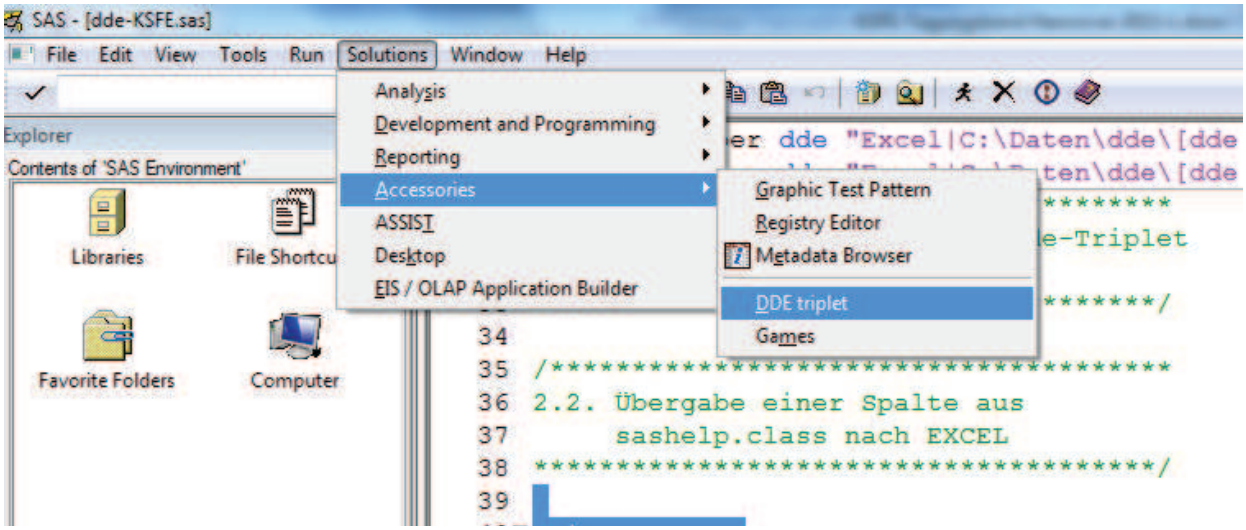


Abbildung 2: DDE-Triplets in SAS anfordern

3. danach öffnet sich eine BOX aus der das dde-Triplet herauskopiert werden kann

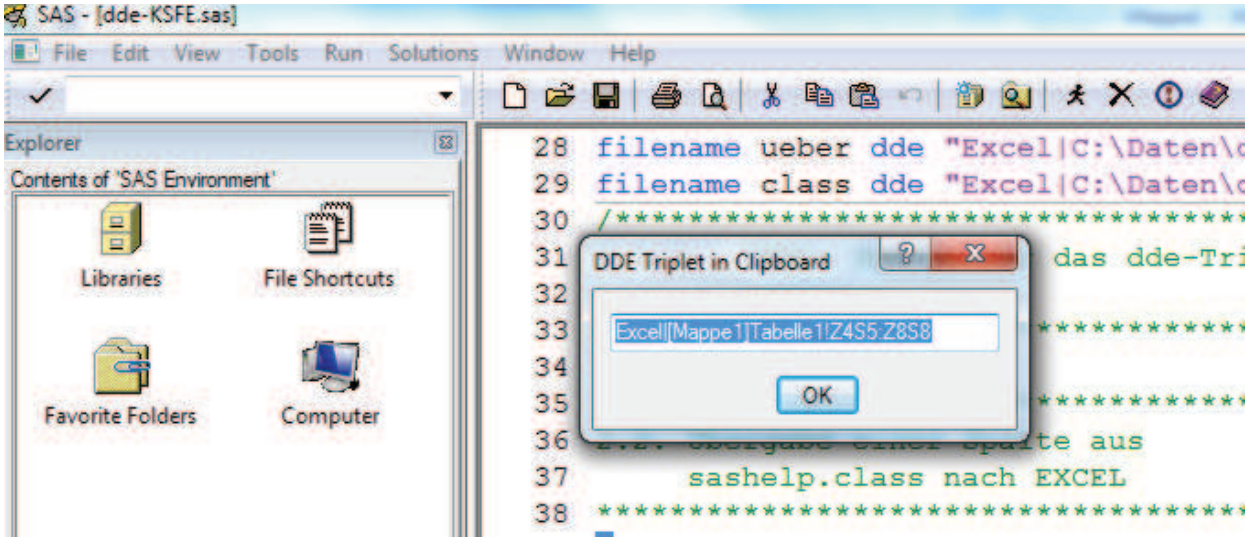


Abbildung 3: Auswahl DDE-Triplet

Die resultierende Filename-Anweisung für die DDE-Ausgabe nach Excel hätte also folgende Gestalt:

```
filename ueber dde "Excel|[Mappe1]Tabelle1!Z4S5:Z8S8";
```

Das EXCEL-Triplet kann sich hinsichtlich des ITEM's je nach „System-Sprache" des Rechners ändern. Falls der Rechner auf die englische Sprache eingestellt ist, ändert sich Z(eile) zu R(ow) und S(palte) zu C(olumn). Im Folgenden gehe ich von einer deutschen Spracheinstellung aus.

3.4.3 Export einer Spalte aus SAS nach EXCEL

```
filename ueber dde
"Excel|C:\Daten\dde\[dde.xlsm]Tabelle1!Z1S1:Z1S1";
filename class dde
"Excel|C:\Daten\dde\[dde.xlsm]Tabelle1!Z2S1:Z20S1";
```

Die Filereferenz ueber referenziert die obere linke Ecke (also Zelle A1) im Worksheet „Tabelle1" des Workbooks „C:\Daten\dde\dde.xlsm". Sie soll für die Ausgabe einer Überschrift genutzt werden. Die Filereferenz *class* referenziert auf die EXCEL-Range A2:A20 im gleichen Workbook und Worksheet. Sie soll für die Ausgabe von einer Spalte aus einer SAS-Tabelle genutzt werden. Über einen SAS-Data-Step kann nun in die Filereferenzen *ueber* und *class* geschrieben werden.

```
Data _NULL_;
set sashelp.class;
/* nach ueber wird ein konstanter String geschrieben!*/
If _N_= 1 then do;
     file ueber;
     put "Name";
end;
```

```
/* nach class wird der Inhalt der Variable name aus sashelp.class
geschrieben*/
file class;
put name; run;
```

3.4.4 Export einer mehrspaltigen Tabelle nach EXCEL

```
 filename ueber dde
"Excel|C:\Daten\dde\[dde.xlsm]Tabelle1!Z1S1:Z1S5";
filename class dde
"Excel|C:\Daten\dde\[dde.xlsm]Tabelle1!Z2S1:Z20S5" notab;
```

Die Tabelle *sashelp.class* hat fünf Spalten. Für die Ausgabe der Spaltenüberschriften und der Daten werden die Filereferenzen *ueber* und *class* auf jeweils fünf Spalten erweitert (S5). In der Filename-Anweisung (der Filereferenz *class*), die für die SAS-Daten genutzt werden soll, wurde jetzt noch zusätzlich die Option notab eingefügt. Diese Option ist bei der Übergabe mehrerer Spalten zwingend anzugeben. Salopp gesprochen werden bei der Kommunikation zwischen SAS und EXCEL via DDE Leerzeichen in Tabulatorzeichen umgewandelt. Das ist schlecht, weil der Tabulator in EXCEL als Trennzeichen zwischen zwei Zellen genutzt wird. Deshalb wird diese Eigenschaft über die ***notab*** Option ausgeschaltet. Die Tabulatoren werden später über Anweisungen im Programm immer dann gesetzt, wenn sie benötigt werden.
Im folgenden Beispiel wird das Tabulator-Zeichen in die Makrovariable „tab“ gespeichert und danach immer dann genutzt, wenn in einer EXCEL-Zeile von einer Zelle in die nächste gewechselt werden soll:

```
%let tab="09"x;
Data _NULL_;
set sashelp.class;
file ueber;
put "name" &tab "sex" &tab "Age" &tab "height" &tab "weight";
file class;
put Name &tab Sex &tab Age &tab Height &tab Weight; run;
```

3.5 EXCEL Befehle bzw. VBA-Macros aus SAS heraus starten

Um aus SAS heraus EXCEL-Befehle oder VBA-Macros anzustoßen, benötigt man zunächst eine Filereferenz an das EXCEL-System:

```
filename sas2xl dde "EXCEL|SYSTEM";
```

3.5.1 VBA-Macro aus SAS heraus starten

Durch Nutzung der Filereferenz *sas2xl* können jetzt existierende VBA-Macros gestartet werden:

```
Data _NULL_;
file sas2xl;
/*******************************************
2.4.2. Aufruf des existierenden VBA-Makros "dde"
*******************************************/
put '[run("dde")]';
run;
```

3.5.2 EXCEL-Befehle aus SAS heraus starten

Aus SAS heraus kann man ebenfalls EXCEL5-Befehle absetzen. Der Sprachumfang dieser EXCEL5-Befehle ist in der Datei macrofun.hlp beschrieben. Diese Datei kann man sich im Downloadbereich von Microsoft herunterladen. Die EXCEL5-Sprache ist sehr alt und wahrscheinlich gibt es auch keinen Support mehr dafür. Mir selbst ist es nicht gelungen, alle in macrofun.hlp beschriebenen Befehle zu nutzen. Die Befehle selbst sind schwer zu lesen und damit auch schwer zu warten. Hier ein Beispiel zur Formatierung des Bereiches Z2S1:Z2S5 (als Zeile 2 von Spalte 1 bis Spalte 5):

```
Data _null_;
file sas2xl;
    put '[select("' "Z2S1:Z2S5" '")]';
     put '[patterns(1,,37,true)]';
     put '[edit.color(37,14,76,151)]';
run;
```

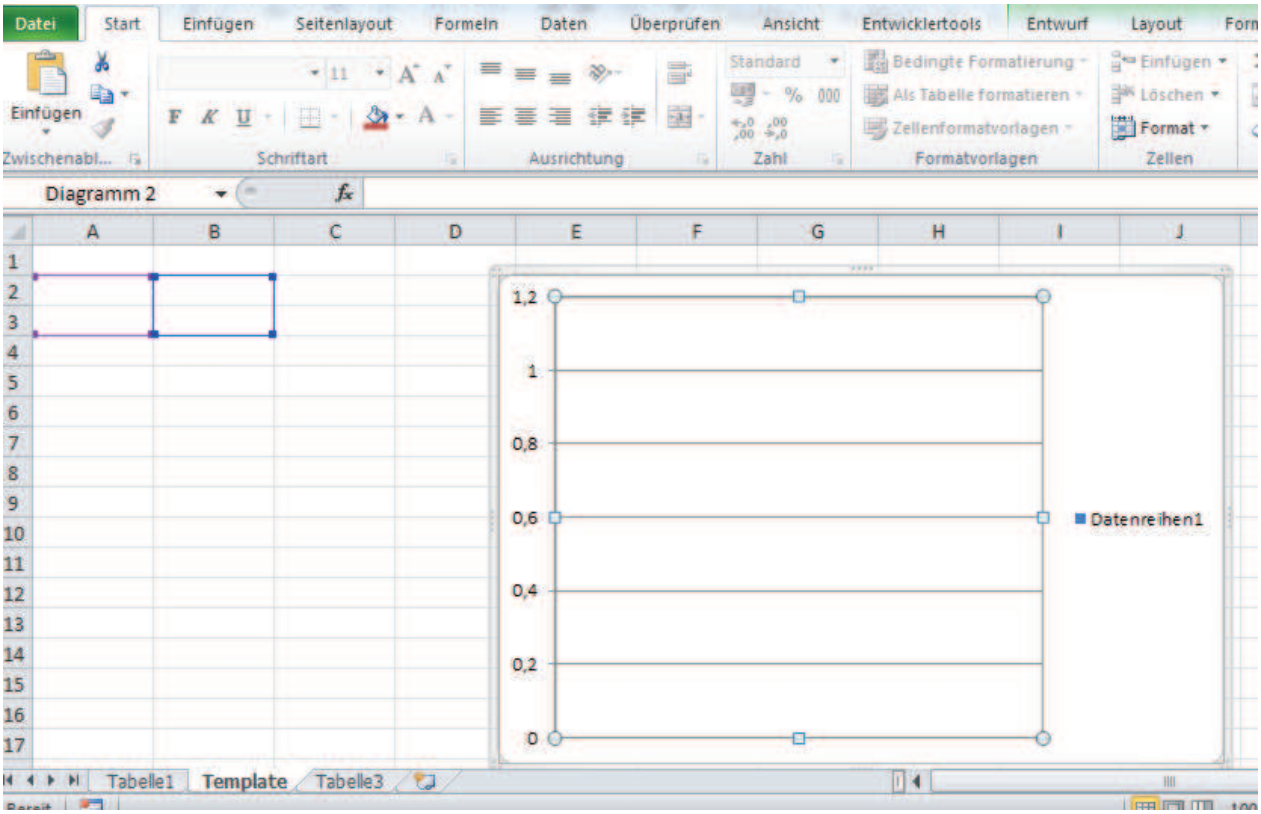
Ohne ausgiebige Kommentare oder wiederholtes Studium der Macrofun.hlp kann ich mir schwer merken, was die einzelnen Übergabeparameter bedeuten. Aus den genannten Gründen nutze ich die EXCEL 5 Befehle nur in seltenen Fällen.
Einen Befehl den ich gern nutze ist der „Save.As“-Befehl, der das Speichern des erstellten EXCEL-Workbooks unter einem beliebigen Namen ermöglicht:

```
data _NULL_;
file sas2xl;
put '[Save.As("C:\Daten\dde\neu.xls")]';
run;
```

3.6 Formatierung der ausgegebenen Daten in EXCEL

Die Formatierung der von SAS nach EXCEL übergebenen Daten habe ich in zwei verschiedenen Methoden vorgenommen. In der ersten Variante habe ich die Daten nach EXCEL ausgegeben und die erzeugten Daten dann in einem Schritt per VBA-Macros formatiert. EXCEL-Grafiken wurden hier im Anschluss an die Datenausgabe ebenfalls per VBA-Macro erzeugt.
In der zweiten Variante habe ich formatierte EXCEL-Worksheets erzeugt und ohne Daten in einem Bereich für Templates gespeichert. Diese Templates können dann zur

Laufzeit kopiert, umbenannt und an die Größe der ausgegebenen SAS-Tabelle angepasst werden. Erst danach werden sie aus SAS heraus mit Daten befüllt.
Dabei wird das Kopieren, Umbenennen und das Anpassen des Ausgabebereiches des Templates über kleine VBA-Macros erledigt.



3.6.1 Verwendung eines EXCEL-Grafik-Templates

Abbildung 4: Beispiel eines Tabellenblatt-Templates

In der obigen Abbildung sehen Sie ein Tabellenblatt „Template“, welches als EXCEL-Template für eine Grafik genutzt werden soll. Die Grafik selbst ist mit dem Range A2:B3 verknüpft.

1) In einem ersten Schritt wird das Template durch VBA-Macros kopiert und die Kopie umbenannt:

```
data _null_;
file sas2xl;
 put '[run("CopyTemplat")]';
 put '[run("renam")]';
run;
```

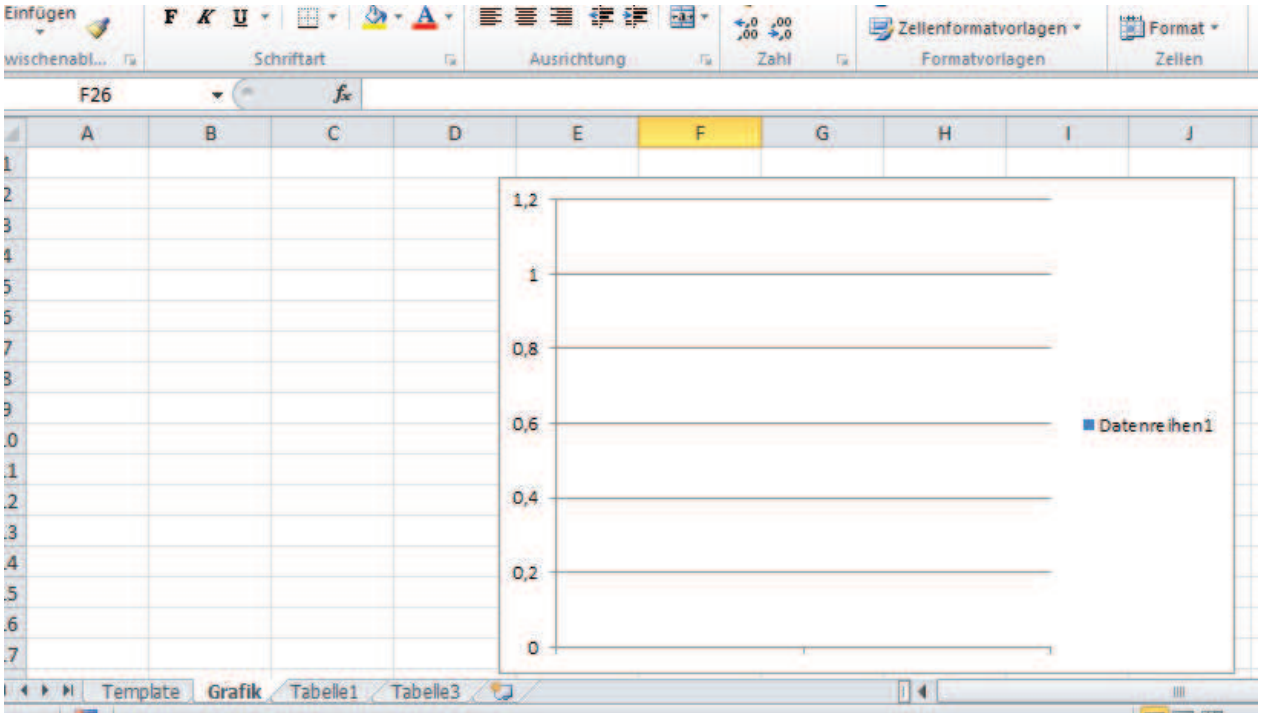


Abbildung 5: Kopie des Tabellenblattes

Die durch das Makro erstellte Kopie heißt im Beispiel Grafik

2) In einem zweiten Schritt wird der mit der Grafik verknüpfte Bereich durch VBA-Macros an die Größe der ausgegebenen SAS-Datei angepasst. Im Beispiel sollen jetzt drei Zeilen und zwei Spalten nach EXCEL übergeben werden. Deshalb wird der Ausgabebereich um eine Zeile erweitert:

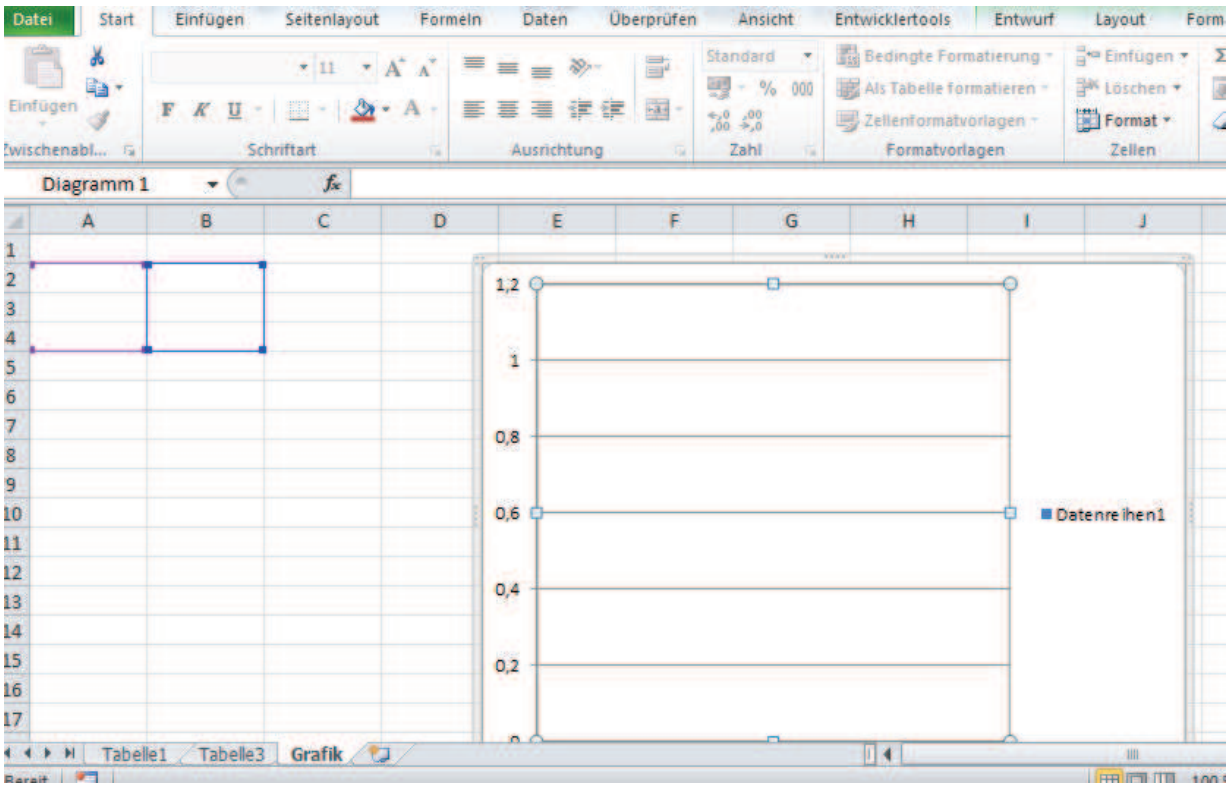


Abbildung 6: Vergrößerung des Ausgabebereichs auf 3 Zeilen

```
Data _null_;
file sas2xl;
do i=1 to 1;
 put '[run("addrow")]';
 put '[run("copyrow")]';
end;
run;
```

Der Ausgabebereich in EXCEL wurde nun auf den Range A2:B3 erweitert.

3) Im letzten Schritt können nun die Daten in den Ausgabebereich geschrieben werden:

```
filename class dde "Excel|C:\Daten\dde\[dde.xlsm]Grafik!Z2S1:Z4S2"
notab;

Data _NULL_;
set sashelp.class;
file class;
if _N_>2 then sex="C";
if _N_<4;
format weight numx12.2;
put sex &tab weight;
run;
```

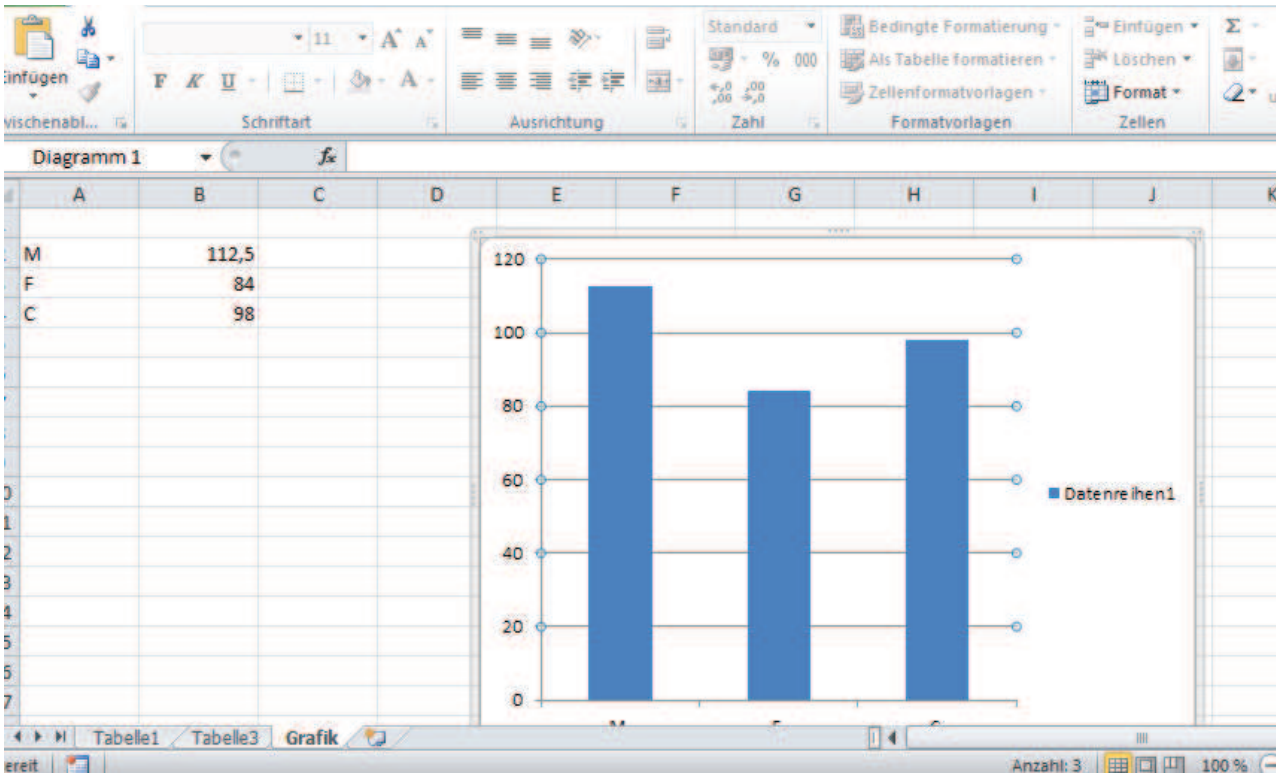


Abbildung 7: Einbezug der Daten in das Template

3.6.2 Fazit Formatierung von Berichten

Für die Formatierung von Berichten habe ich die beiden Varianten genutzt:
a) Formatierung der Daten über VBA-Macros und
b) Ausgabe in formatierte Templates
Die Variante b) ist meiner Meinung nach aus folgenden Gründen die bessere:

- es wird kein Programmcode für die Formatierung benötigt,
- es werden nur sehr wenige kleine VBA-Macros benötigt,
- ein neues Template kann sehr schnell von Hand erstellt werden, wohingegen eine neue Ausgabe, die per VBA-Macros formatiert werden soll, viel Zeit in Anspruch nimmt. Ein solches Template könnte auch von jemandem ohne Programmierkenntnis erstellt werden.
- Insgesamt ist die Wartung aufgrund des reduzierten Codings in Variante b) erheblich besser.

Etwas mehr Arbeit hat man bei Variante b) wenn sich später beispielsweise das Corporate Design ändert. In diesem Fall müssen alle Templates von Hand an das neue Design angepasst werden, während bei guter Programmierung in Variante a) nur ein paar Macros angepasst werden müssen. Allerdings ist eine Anpassung der Templates an ein neues Design zum großen Teil Fleißarbeit, die relativ schnell zu erledigen ist und bei der nicht allzu viel schief gehen kann.

4 Berichtsgenerator

Aufgrund der vorangegangenen Betrachtungen wurde der Berichtsgenerator aus folgenden Bausteinen zusammengesetzt:

- Excel Auftragstabelle
- Aufrufmakro in SAS
- EXCEL-Templates
- SAS-Programme
- geringe Anzahl kleinster VBA-Programme

In der Excel-Auftragstabelle muss der Anwender die von ihm geforderten Berichte einpflegen. In der Spalte Tabellenblatt steht der Name des Exceltabellenblatts, in das die Ausgabe erfolgt. Über die Spalte Analyse wird klar, welche SAS-Programme zur Kennzahlenberechnung und welches als Ausgabetemplate zu nutzen ist. Über die Spalte Variablenliste wird die Analysevariable(n) spezifiziert und über die Spalte SAS_Datei wird angegeben, welche SAS-Tabelle zur Analyse zu nutzen ist.

D2 | LGD_VALI.&VERF._Validierungsdaten(where=(Ausr_&zielvar ne 1))

	A	B	C	D
1	Tabellenblatt	Analyse	Variablenliste	SAS_Datei
2	DEFAULT_YEAR_g	ClVarHisto	class=Default_Year,sort=0	LGD_VALI.&VERF._Validierungsdaten
3	Status_g	ClVarHisto	class=Bearbeitungsstatus,sort=0	LGD_VALI.&VERF._Validierungsdaten
4	Status_Default_year_g	class_status	class=DEFAULT_YEAR	LGD_VALI.&VERF._Validierungsdaten
5	Continent_g	ClVarHisto	class=Continent,sort=0	LGD_VALI.&VERF._Validierungsdaten
6	Region_g	ClVarHisto	class=Region,sort=0	LGD_VALI.&VERF._Validierungsdaten
7	DEFAULT_YEAR_rep	Rep_graf	class=Default_Year,sort=0	LGD_VALI.&VERF._Validierungsdaten
8				

Abbildung 8: Beispiel für eine Excel-Auftragstabelle

Danach wird im Aufrufmakro eingestellt, welche EXCEL-Aufruftabelle einzulesen ist. Aus der Aufruftabelle werden danach die Aufrufe der benötigten SAS-Macros zur Erstellung der Berichte erzeugt. Am Ende werden die Berichte in die jeweiligen Tabellenblätter ausgegeben.

Literatur

[1] http://de.wikipedia.org/wiki/Dynamic_Data_Exchange

[2] http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#win-stmt-x.htm

[3] Creating Custom Excel Workbooks from Base SAS with Dynamic Data Exchange: A Complete Walkthrough http://www2.sas.com/proceedings/sugi27/p190-27.pdf

[4] Using Dynamic Data Exchange to Export Your SAS Data to MS Excel -Against All ODS, Part I - http://www2.sas.com/proceedings/sugi26/p011-26.pdf