„Geht nicht – gibt's nicht" Hacking SAS & Excel

Alexander Jahnke
Siemens Healthcare Diagnostics
Products GmbH
Emil-von-Behring-Straße 76
35041 Marburg
Alexander.jahnke@siemens.com

Zusammenfassung

Am Beispiel eines Problems aus der Praxis wird der steinige Weg einer beidseitigen Kommunikation zwischen Excel und SAS beschrieben. Dabei verfolgt ein VBA-Makro im Dialog mit SAS über einen SAS-Share Server die Einträge und Änderungen in Excel-Dateien, wobei beide Programme ihre Statusmeldungen miteinander kommunizieren.
Was in der Problembeschreibung noch relativ trivial aussieht, erwies sich in der praktischen Umsetzung als problematisch und nicht mit Standardmitteln zu lösen.

Nach der Devise „Geht nicht – Gibt's nicht" wird im Rahmen des Vortrags aufgezeigt, wie man die Kommunikation (und andere Probleme) doch lösen kann, wenn man außerhalb der üblichen Bahnen denkt und seinen Werkzeugkasten erweitert.

Schlüsselwörter: Excel, SAS-Excel, Kommunikation

1 Die Aufgabe

Während der Produktion im Pharmabereich werden Messdaten aufgenommen und diese - zum Teil über mehrere Tage hinweg - in eine Excel-Datei eingetragen. Dabei ist nicht sichergestellt, ob bereits eingetragene Werte nachträglich verändert wurden (zufällig oder durch eine gewollte Korrektur). Mit SAS- und Excel-Mitteln sollte ein Tool geschaffen werden, das diese Funktionalität übernimmt. Dabei sollten bei jedem Speichervorgang die Inhalte mit der Vorversion verglichen werden und bei Änderungen soll dokumentiert werden, welche Zelle wie geändert wurde, von wem und warum.

2 Die Idee

Ein VBA-Makro in Excel soll die zu bearbeitende Datei vor unauditierter Dateneingabe schützen. Änderungen an der Datei sollen erst nach einem kontrollierten „*Aufschließen*" (und damit Start der Audit-Funktionalität) möglich sein. Dabei wird erfasst, wer an dieser Datei welche Eingaben gemacht hat und diese Eingaben bei jedem Speichervorgang gegen die Vorversion verglichen. Die Vorversionen werden als SAS-Dataset auf einem SAS-Share-Server zentral gespeichert. Jede so präparierte Excel-Datei ist genau einem Dataset auf dem Shareserver zugeordnet und wird über eine ID miteinander verknüpft. Um Missbrauch und Fehlbedienungen zu verhindern, ist die normale Speicherfunktion

in Excel durch das Makro ausgeschaltet. Ebenso ist es nicht möglich Excel zu schließen, ohne den auditierten Speichervorgang zu durchlaufen.

3 Probleme auf dem Weg

- Die zu auditierende Datei ist in Excel geöffnet und damit ist es nicht möglich mit SAS Access to PC Files die Daten zu lesen.
- In Excel müssen zusätzliche Menüpunkte angeboten werden, um die gesperrten Funktionalitäten wie Speichern zu ersetzen. Jeder dieser Menüpunkte muss eine andere Aktion in SAS starten.
- SAS muss eine Rückmeldung an Excel geben können: Hat das SAS-Programm korrekt funktioniert, was waren die Eingaben vom User, war die Bearbeitung fehlerhaft, lagen Änderungen vor oder anderes?
- Der Absturz von SAS während der Bearbeitung muss abgefangen werden.
- Eine Bearbeitung der Excel-Datei ohne das Excel-Steuer-Makro darf nicht möglich sein.
- Der User muss immer die aktuelle Version des Excel-Makros haben.

4 Ungewöhnliche Lösungsansätze

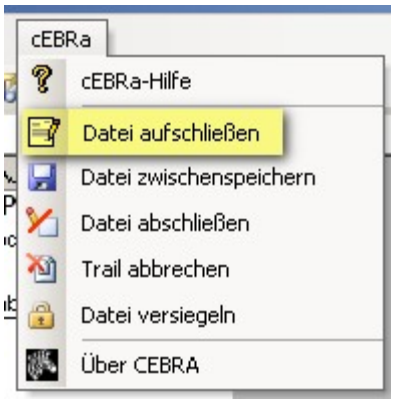
Ein normaler Lösungsansatz mit SAS- und Excel-Mitteln konnte diese Anforderungen und Probleme weder erfüllen noch lösen. An jedem Rechner stehen dem Programmierer aber eine Vielzahl weiterer technischer Möglichkeiten zur Verfügung, die einem auf den ersten Blick nicht in den Sinn kommen und die man für seine Zwecke umfunktionieren kann, um dann die gewünschte Lösung zu bekommen. Diese Technik wird auch als „Hacking“ bezeichnet:
„In einem übergreifenden Sinn umfasst ‚Hacker’ experimentierfreudige Personen, die mit ihren Fachkenntnissen eine Technologie beliebiger Art außerhalb ihrer normalen Zweckbestimmung oder ihres gewöhnlichen Gebrauchs benutzen.“
Quelle: Wikipedia

5 Die Lösungen

5.1 Exceldatei ist bereits geöffnet

Die Daten werden mittels DDE aus Excel gelesen, dabei können auch offene Dateien bearbeitet werden. Weiterhin ist es mittels DDE möglich die offene aktive Excel-Datei herauszufinden (wenn zum Beispiel mehrere Dateien in Excel geöffnet sind). Dazu wird ein neues Blatt erstellt und in dieses mit EXCEL4-Sprache ein Makro geschrieben, welches den Namen der aktuellen Datei in eine Zelle schreibt. Diese Zelle wird wieder von SAS ausgelesen. Auch diese Funktionalität wird komplett über DDE gestartet und gesteuert.

5.2 Verschiedene Befehle aus Excel



Jeder Eintrag im Excel-Menü muss eine bestimmte Funktionalität in SAS auslösen. Dabei ist SAS beim Klick auf den Menüpunkt noch nicht gestartet.
Klickt der Benutzer auf einen Menüpunkt, startet Excel ein einfaches MS-DOS Batchprogramm. Dieses Batch-File startet SAS und übergibt das eigentlich auszuführende SAS Programm als Parameter.
Der Aufruf sieht wie folgt aus (am Beispiel „Aufschließen“):

```
Programmaufruf: "<saspfad>sas.exe" -icon -autoexec „<autoexec-pfad>"
-config „<configpfad>" -log „<logpfad>"  -nosplash
-sysin  „<sasprogramm-aufschliessen>"
```

- **<saspfad>sas.exe**: Enthält den Pfad zur SAS-Installation
- **-icon:** startet SAS minimiert
- **-autoexec:** Pfad zur Autoexec-Datei
- **-config:** Pfad zur Config-Datei
- **-log:** Pfad zum Log-Speicherort
- **-nosplash:** unterdrückt das SAS-Logo zum Programmstart
- **-sysin:** Pfad zum eigentlichen SAS-Programm mit der Funktionalität „Aufschließen“

Im SAS Programm werden die Libraries für die Funktionaltät zugewiesen (zum Beispiel auf den Share-Server) und dann mit proc display ein SCL-Code aufgerufen. Innerhalb dieses Codes findet dann die tatsächliche Bearbeitung der Aufgabe statt.
(*Die Programmierung in SCL ist eine persönliche Vorliebe, keine Notwendigkeit.)*

5.3 Rückmeldung aus SAS

Eines der zentralen Probleme war eine funktionierende, beidseitige Kommunikation zwischen SAS und Excel zu schaffen. Es ist sehr einfach Daten aus Excel mit SAS zu lesen. Ungleich schwieriger ist es, Daten wieder von SAS an Excel zurückzusenden.
Der zuerst versuchte Pfad, die Statusdaten aus SAS ebenfalls per DDE wieder in ein verstecktes Blatt in Excel zu schreiben, erwies sich nicht als gangbar. Dazu hätte das Excel-Makro dauerhaft eine Zelle auf Inhaltsveränderung beobachten müssen - was machbar ist – allerdings ist es durch diese „Excel-Dauerbeobachtung“ nicht mehr möglich von SAS in die Zelle zu schreiben.
Um dieses Problem zu lösen, kommunizieren die Programme über einfache Text-Dateien in einem versteckten Verzeichnis. Dieses Verzeichnis wird von Excel auf Ände-

rungen überwacht. Wird dort eine passende Datei entdeckt, liest Excel die Informationen ein und löscht die Datei.
Um die Kommunikation möglichst schlank zu halten, werden die Informationen im Dateinamen verschlüsselt, die Datei selbst ist leer. Die Datei wird mit call system in SAS erzeugt. Da es in MS-DOS direkt nicht möglich ist, eine leere Textdatei anzulegen, muss ein kleiner Trick genutzt werden. Das Dos-Commando wird von SAS mit call system an Windows weitergegeben. Mit

```
type NUL>"<pfad>\Start_ok_4.ceb"
```

wird eine leere Datei unter den Namen „Start_ok_4.ceb“ im angegebenen Pfad angelegt. Sie enthält den Status zur von Excel aufgerufenen Funktion „Aufschließen“ (Start) und übergibt, dass die Excel-Datei mit der internen ID 4 erfolgreich aufgeschlossen wurde (ok). Mit dieser Meldung arbeitet Excel weiter und gibt die Datei zur Bearbeitung für den Nutzer frei. Anderenfalls würde Excel den User über eventuelle Fehler informieren.

5.4 Absturz

Für die Kommunikation zwischen Excel und SAS werden vom Programm Zeitfenster vorgegeben. Damit soll verhindert werden, dass Excel „zu lange“ auf eine Antwort von SAS wartet, zum Beispiel wenn SAS abgestürzt ist oder nicht gestartet werden konnte. Da eine normale Kommunikation zwischen den Applikationen zwei Sekunden in Anspruch nimmt, wurde das Fenster auf acht Sekunden festgelegt.
Da es aber auch SAS-Aktionen gibt, die wiederum eine User-Eingabe erfordern (zum Beispiel einen Kommentar, warum eine Zelle geändert wurde), wären acht Sekunden deutlich zu kurz. In diesem Fall wird von SAS eine weitere Status-Meldung per MS-DOS geschrieben. Mit dem Status „wait_<ID-Nummer>.ceb“ erkennt das Excel-Makro, dass eine Benutzereingabe erforderlich ist und verlängert das Zeitfenster auf 60 Minuten.

5.5 Versionsmanagement

Da die Bearbeitung einer zu auditierenden Excel-Datei nur mit dem entsprechenden Programm erfolgen soll, ist es wichtig, dass der Nutzer immer mit der aktuellen Version arbeitet und nur eine kleine Hürde im gewohnten Arbeitsablauf darstellt.
Daher wurden diese neuen Funktionen in eine bereits bekannte und weitverbreitete SAS-Anwendung eingebaut. Beim Aufruf dieser Anwendung startet allerdings nicht direkt SAS, sondern ein weiteres Batch-File (*das Icon täuscht dem Nutzer vor, es würde sich um SAS handeln*). Dieses Batch-Programm kopiert bei jedem Aufruf, die aktuellen Version der Autoexec und der Config-Dateien vom Server auf den Client. Das dann startende SAS-SCL-Programm überprüft, ob es aktuellere Versionen einzelner SAS-Module auf dem Server gibt und kopiert diese ggf. auf den lokalen Client. Dabei wird auch das Excel-Steuer-Makro kopiert und aktuell gehalten.
Die bekannte SAS-Anwendung bietet für den User die allgemeine Übersicht der Excel-Bearbeitung und ist seine Steuerzentrale. Hier sieht er, welche Exceldateien er bearbei-

tet hat und welche noch offen sind. Aus der SAS-Anwendung kann der User seine zu bearbeitende Excel-Tabelle über einen „Öffnen“-Dialog anwählen. Dabei wird ihm das Excel-Makro unbemerkt untergeschoben:

```
rc=system("'C:\Program Files\<…>\Bin\Show Desktop.scf'");
```

Bei einem Windows-Betriebssystem minimiert der Aufruf von „Show Desktop“ alle offenen Fenster.

```
rc=system(„<pfad>\Excel_Macro.xls“)
```

startet das aktuelle Excel-Makro.

```
rc=system(„<pfad>\ausgewähtle_Excel_Datei.xls“)
```

startet die vom User ausgewählte Excel-Datei.
Da alle anderen Fenster durch „Show Desktop“ minimiert wurden, hat der User direkt die gewünschte Excel-Datei im Vordergrund und kann mit seiner Arbeit beginnen.

6 Fazit

Mit den schon fast vergessenen oder kaum bekannten Werkzeugen eines PCs lassen sich viele Funktionen, trotz anfänglicher Probleme realisieren. Dabei ist es wichtig, keinerlei Scheu vor dem „Missbrauch“ oder Umwidmung bestimmter Befehle und Funktionen zu haben, sondern einfach auszuprobieren, ob dieser neue Weg funktioniert.