Die Qual der Wahl: Standardmakros verteilt entwickeln - mit SAS Studio oder SAS Enterprise Guide?

Johannes Lang
HMS Analytical Software
Rohrbacher Str.26
69115 Heidelberg
johannes.lang@analytical-software.de

Zusammenfassung

Die webbasierte Programmierumgebung SAS Studio ist seit einigen Jahren als Alternative zum SAS Enterprise Guide verfügbar. Einige Vor- und Nachteile der jeweiligen Software-Pakete wurden bereits auf früheren Konferenzen vorgestellt.
Für die Nutzung in SAS-Projekten, bei denen z.B. Standardmakros entwickelt werden, ist jedoch folgende Frage besonders wichtig: Wie können mehrere Entwickler gemeinsam an einer Codebasis arbeiten? Themen, die hier eine maßgebliche Rolle spielen, sind beispielsweise Versionsverwaltung, Softwareverteilung oder Datenzugriffe über das Netzwerk.
Vor diesem Hintergrund soll zum einen gezeigt werden, welche Möglichkeiten SAS Studio bisher (nicht) bietet, und wie diese in der Praxis genutzt (bzw. nachgerüstet) werden können. Zum anderen wird die Herangehensweise der SAS-Enterprise-Guide-Software an die oben genannten Themen skizziert. Das Ziel: SAS-Entwicklern bei der Auswahl und Einrichtung ihrer Projektumgebung Hilfestellung geben zu können.

Schlüsselwörter: SAS Studio, SAS Enterprise Guide, Versionskontrolle, Git, Repository, Workspace Server

1 Einführung

Über die spezifischen Eigenschaften bzw. Vor- und Nachteile der beiden Entwicklungswerkzeuge SAS Studio und SAS Enterprise Guide wurde bereits einiges veröffentlicht (vgl. [4], [5], [6]). An dieser Stelle sollen einige Aspekte, die für die gemeinsame Entwicklung von Standardmakros wichtig sind, exemplarisch für beide Umgebungen betrachtet werden.

Dazu gehören das Aufsetzen einer Projekt-Entwicklungsumgebung und das Skizzieren eines praktikablen Workflows. Hierzu wird versucht, eine Einführung in das Arbeiten mit dem verteilten Versionskontrollsystem Git (vgl. [1]) zu geben. Dies ist nach Einschätzung des Autors entscheidend für die effektive Zusammenarbeit, da damit viele Arbeitsabläufe bereits auf Infrastruktur-Ebene unterstützt werden.

1.1 Zur Erinnerung: Steckbrief der beiden Werkzeuge

Der SAS Enterprise Guide (SEG) ist ein Windows-Client, der einzeln (d.h. ohne SAS Deployment Wizard) installiert werden kann. Er ermöglicht den Zugriff auf lokale oder entfernte SAS-Server. Als Gliederungselement dienen Prozessflüsse, die Arbeit wird in so genannten Projekten gespeichert.

SAS Studio hingegen ist eine Webapplikation, die im SAS Webapplikationsserver installiert und betrieben wird. Die Interaktion erfolgt über den Browser. Es sind grundsätzlich drei Varianten zu unterscheiden (vgl. [7], S.1ff):

- Single User: Diese Variante kann zusammen mit SAS Foundation lokal installiert werden. Dabei wird ein lokaler Webapplikationsserver mitkonfiguriert, der als Dienst gestartet werden muss. Man arbeitet unter dem Account des lokalen Betriebssystembenutzers.
- Basic: Hier läuft SAS Studio im Webapplikationsserver der SAS Plattform, und nicht auf dem Client-Rechner. Die Anmeldung erfolgt über einen Betriebssystembenutzer auf dem SAS Server.
- MidTier/Enterprise: Die Architektur ist entspricht grundsätzlich derjenigen bei der Basic-Variante, aber die Anmeldung erfolgt über einen SAS Metadaten-Benutzer. Dieser hat dann Zugriff auf alle zentral registrierten und berechtigten Workspace Server, und kann ggf. zwischen diesen umschalten.

Für diesen Beitrag wurde insbesondere SAS Studio Enterprise evaluiert. Dort wo Aspekte für beide Varianten funktionieren, wird darauf hingewiesen.

2 Arbeiten mit Programmen und Daten

2.1 Prozessflussansicht in SAS Enterprise Guide

In der Prozessflussansicht des SEG können Programme eingebettet oder als Verknüpfungen eingefügt werden. Geöffnete Tabellen werden automatisch als Verknüpfungen im aktuellen Prozessfluss eingefügt. Das Arbeitsbereich-Layout kann geändert werden in geteilt bzw. gestapelt, so dass z.B. zwei Tabellen gleichzeitig angezeigt werden können.

Um mit größeren Datenmengen effizient arbeiten zu können, kann die Anzahl der initial zu ladenden Zeilen eingestellt werden (Extras / Optionen / Daten / Performance). Außerdem kann eine Tabelle in der Serverliste markiert und im Kontextmenü „Durchsuchen“ aufgerufen werden, ohne dass die Tabelle in den Prozessfluss eingefügt wird.

2.2 Programmeditor- bzw. Prozessflussansicht in SAS Studio

In SAS Studio kann zwischen zwei Ansichten umgeschaltet werden: Einer Programmeditor-Ansicht („SAS Programmer“) und einer Prozessflussansicht ("Visual Pro-

grammer"). Die Programmeditor-Ansicht entspricht konzeptionell eher der des SAS Display Management System (DMS), da hier pro Programm eine neue Registerkarte (Tabber) geöffnet wird. Im SEG wird die Editor-Ansicht nur beim Öffnen eines Programms innerhalb der Prozessfluss-Ansicht geöffnet. Allerdings ist der Funktionsumfang des Editors ähnlich groß wie im SEG (Auto-Vervollständigung, Code-Einrückung etc.). Verfügbare SAS-Tabellen können im linken Auswahlbereich aufgeklappt werden, um die Spalten anzuzeigen. Werden Objekte dort markiert und per *drag and drop* ins Editorfenster gezogen, dann werden die Texte (Librefs, Variablennamen) kopiert, wodurch Schreibarbeit gespart werden kann. Jede Tabelle wird in einem neuen Tabber geöffnet. Standardmäßig ist immer nur ein Tabber sichtbar, doch auch hier kann die Anzeige geteilt werden, indem ein Tabber per *drag and drop* nach rechts gezogen und dort in einem dynamisch grau markierten Bereich losgelassen wird (vgl. Abbildung 1). Aus Performance-Gründen werden in der aktuellen Version immer nur die ersten 100 Beobachtungen sofort geladen, der Rest erst beim Scrollen durch die Daten (vgl. [5], S.12).

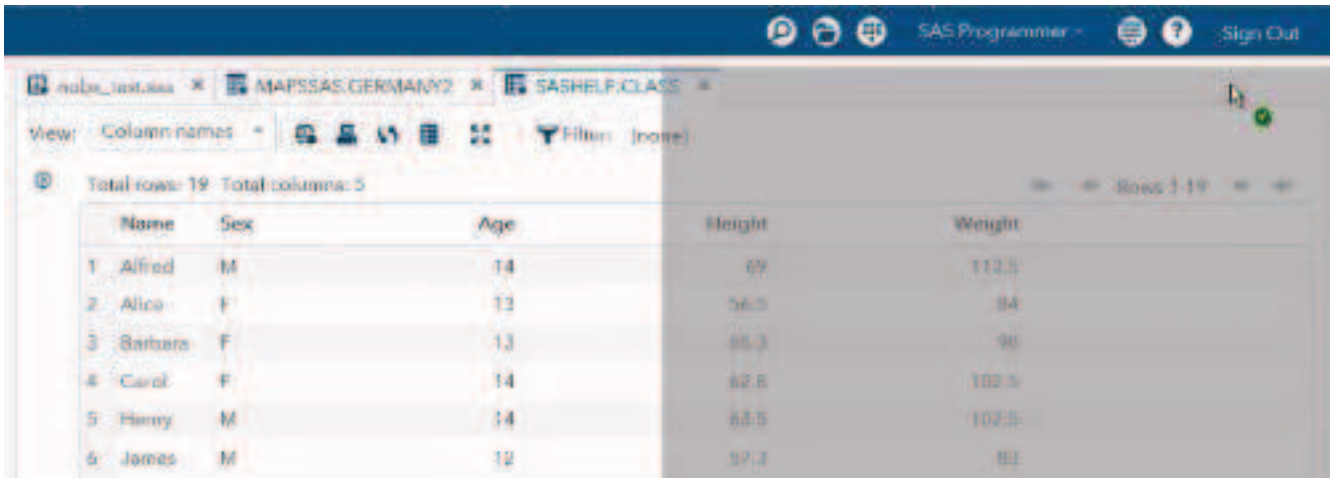

Abbildung 1: Drag and Drop von Tabbern in SAS Studio

Die Prozessfluss-Ansicht in SAS Studio unterscheidet sich grundsätzlich von der in SEG, da hier keine Verknüpfungen erstellt werden, sondern echte Knoten-Objekte, welche Ein- und Ausgabeschnittstellen (engl. *control ports*) haben. Tabellen werden beim Öffnen daher auch nicht als Verknüpfung eingefügt. Nur bei explizitem Import wird ein Prozessfluss-Knoten erzeugt. Auch kann keine manuelle Verbindung zwischen Code- und Tabellen-Knoten wie im SEG gezogen werden, um eine Ausführungsreihenfolge sicherzustellen.

Ergebnisse (z.B. HTML-Berichte) werden nur in der Editoransicht sofort geöffnet (in einem neuen Tabber). In der Prozessflussansicht wird der ausgeführte Knoten als erfolgreich ausgeführt markiert, zur Anzeige des Ergebnisses muss aber auf den Knoten doppelgeklickt werden.

2.3 Interaktive Ausführung von SASUnit-Tests

Wird mit automatisierten Tests gearbeitet (was grundsätzlich zu empfehlen ist), dann ist es mit der aktuellen SASUnit-Version möglich, diese in beiden Oberflächen interaktiv auszuführen, und das Ergebnis als HTML-Bericht direkt in der Entwicklungsumgebung anzuzeigen. Details dazu sind im Beitrag von Landwich in diesem Band nachzulesen.

3 Arbeiten mit einem verteilten Versionskontrollsystem

Nachdem nun kurz die Arbeitsweise mit den jeweiligen Werkzeugen vorgestellt wurde, geht es jetzt um das Wesentliche: Wie arbeiten mehrere Entwickler zusammen an derselben Codebasis?

3.1 Einführung

3.1.1 Lokale Repositories ermöglichen Trennung von Sicherung (*commit*) und Veröffentlichung (*push*)

Bei einem zentralen Versionskontrollsystem wie Subversion (SVN) bedeutet jeder Commit auch die Veröffentlichung der Codeänderungen, da es nur ein zentrales Repository gibt. Dies ist oftmals nicht beabsichtigt, da z.B. ein Feature noch nicht fertig implementiert oder getestet ist. Dies kann zu späten bzw. seltenen Commits führen, was die Zusammenarbeit erschwert, da die Kollegen ggf. so lange auf die Änderung bzw. Freigabe der gesperrten Datei warten müssen. Außerdem findet meist so lange keine Sicherung der Änderungen statt, wenn diese nur auf dem lokalen Rechner existieren.
Bei einem verteilten Versionskontrollsystem wie Git (vgl. [1]) können mehrere lokale Repositories von einem zentralen Repository abhängen. Jeder Entwickler kann sich problemlos sein eigenes Repository „klonen“, in dem er seine Änderungen durchführt. Damit können Zwischenstände durch lokale Commits immer versioniert werden, ohne dass Kollegen dadurch beeinträchtigt werden. Dem Entwickler steht eine so genannte *staging area* zur Verfügung, in die er Änderungen schiebt, die er später committen möchte. Auch hier können Änderungen zurückgehalten werden, was aber in der Regel weder sinnvoll noch notwendig ist.

Mehrere lokale Commits können dann bei Bedarf neu geordnet werden, um z.B. fachlich zusammenhängende Änderungen zu bündeln, die über mehrere Tage committet wurden. Schließlich kann der neue Stand in das zentrale Repository veröffentlicht werden (*push*). Wird ein Git-Server eingesetzt, dann können als Zwischenstufe auch Benachrichtigungen ausgelöst werden, z.B. Bitte um Code-Review und Freigabe (*pull request* von Entwickler A zu Entwickler B, der dann autorisiert ist, die *push*-Operation durchzuführen). Damit unterstützt die Infrastruktur bereits ein Vier-Augen-Prinzip zur Erhöhung der Software-Qualität. Änderungen im zentralen Repository können ins lokale Repository heruntergeladen (*fetch*) und dann schrittweise mit dem eigenen Stand zusammengeführt (*merge*) werden.

3.1.2 Verbessertes Branching und Merging

Im Unterschied zu Subversion gibt es bei Git keinen Entwicklungs-Stamm (*trunk*), sondern nur Entwicklungszweige (*branches*). Initial besteht jedes Git-Repository aus einem Hauptzweig (*master branch*), von dem dann je nach Arbeitsablauf mehrere Zweige abgeleitet werden. Hier liegt eine große Stärke, aber auch die große Komplexität innerhalb von Git, wenn es darum geht, unterschiedliche Entwicklungszweige wieder zusammenzuführen. Hier kommt z.B. Subversion oft an seine Grenzen, weil die Historie der Zweige nicht berücksichtigt wird, sondern nur die aktuell vorliegenden Versionen übereinander gelegt werden. Git kann zwei Versionsstände schrittweise (*commit* für *commit*) zusammenführen, wobei der Entwickler jeden Schritt kontrollieren und ggf. korrigieren kann.

3.2 Nutzung des SEG Plugins zur Änderungsverwaltung

Seit Version 7.1 bietet der SEG eine Integrationsmöglichkeit für Git, sofern das Repository lokal zugänglich ist. Hierzu muss in den Einstellungen des aktuellen SEG-Projektes die Option aktiviert werden, dass beim Einfügen von Programmen relative Pfade verwendet werden sollen (vgl. [2]). Extern versionierte Programme dürfen dann nur über den Datei/Öffnen-Dialog vom Arbeitsplatz geöffnet werden, damit die automatische Erkennung der Git-Metadaten (Ordner .git im Repository-Wurzelverzeichnis) funktioniert. Beim Öffnen über die Serverliste (auch bei lokalem SAS-Server) funktioniert die Erkennung leider nicht.

Einige wenige Git-Features sind über den SEG-Dialog möglich: Änderungen anzeigen, Commit, Historie anzeigen, Zurücksetzen. Daher würde der Autor einen separaten Git-Client empfehlen.

3.3 Nutzung im SAS Studio Kontext

SAS Studio bietet bisher keinerlei Unterstützung für die Nutzung eines Versionskontrollsystems. Für das Ein- und Auschecken etc. muss daher ein separater Client genutzt werden. Für Git sind viele Clients verfügbar, die ohne großen Aufwand installiert und genutzt werden können, um Repositories zu verwalten (siehe [3]). Git selbst bringt auch eine grafische Benutzeroberfläche mit, die für einfache Operationen (wie im SEG) ausreichend ist.

Die Entwicklung eines SAS Studio Task für die Interaktion mit einem Versionskontrollsystem wäre zwar auch möglich, aber wenig zielführend, da es für diese Aufgabe bereits genügend brauchbare Lösungen gibt.

4 Konfiguration für die Zusammenarbeit im Projekt

In diesem Abschnitt sollen kurz die Möglichkeiten der beiden Werkzeuge beschrieben werden, auf gemeinsam genutzte Ressourcen zuzugreifen bzw. diese zwischen Entwicklern auszutauschen.

4.1 Einbinden von Serverordnern und projektspezifische Sitzungskonfiguration (SEG und SAS Studio)

Liegen die gemeinsam genutzten Ressourcen in einem freigegebenen Ordner, der vom SAS Server aus erreichbar ist, dann kann dieser an der Werkzeug-Oberfläche verfügbar gemacht werden. Dies wird in der Konfiguration des Workspace Servers eingetragen (beim Plattform-Szenario in der SAS Management Console). Hier gibt es folgende Möglichkeiten:

- *SAS User Root* (Default): Nur der zentral konfigurierte persönliche Ordnerpfad ist sichtbar.
- *System Root*: Alle Laufwerke auf dem SAS-Server sind sichtbar. Aus Sicherheitsgründen ist diese Einstellung nicht zu empfehlen.
- *Path*: Nur Ordner unterhalb des eingestellten Pfades sind sichtbar. Hier kann ein Wurzelpfad auf dem SAS-Server angegeben werden, unter dem die gemeinsam genutzten Ressourcen liegen.

Darüber hinaus kann die Initialisierung des Workspace Servers über eine Konfigurationsdatei (autoexec_usermods.sas) zentral gesteuert werden, z.B. um eine Projekt-Autoexec-Datei zu laden (dies geht auch bei SAS Studio Basic, siehe [4]). Um darin z.B. je nach angemeldetem Entwickler passende Pfade zu setzen, kann die automatische Makrovariable SYSUSERID verwendet werden.

In SAS Studio hat der Entwickler zusätzlich die Möglichkeit, Initialisierungscode für seine Sitzung in einem Codefenster zu hinterlegen, unabhängig vom verbundenen Workspace Server. Standardmäßig wird der erste gefundene Server initialisiert. Sind mehrere konfiguriert, dann kann (sofern diese Option nicht vom Administrator deaktiviert wurde) zwischen den Servern im Menü umgeschaltet werden. Die letzte Auswahl wird für den angemeldeten Benutzer gespeichert.

4.2 Nutzung von FTP-Serverordnern (SAS Studio)

In SAS Studio besteht die Möglichkeit, zusätzlich zu normalen Serverordnern auch FTP-Ordner über die Oberfläche einzubinden. Hierzu muss die Datenquelle einmalig im Menü registriert werden (mit Hinterlegung von Zugangsdaten). Dadurch können auch Internetdatenquellen angebunden werden.

4.3 Nutzung von Repositorys (SAS Studio)

SAS Studio hat einen eigenen Repository-Begriff eingeführt, der sich von dem der SAS Plattform (und von dem eines Versionskontrollsystems) unterscheidet. In der SAS Plattform ist ein Repository ein Metadaten-Container, der über den SAS Metadata Server erreichbar ist. Im Versionskontrollsystem Git ist ein Repository ein lokaler oder zentraler Container für versionierte Dateien.

Im SAS Studio Kontext jedoch ist ein Repository ein Software-Paket, das mit speziell formatierten XML-Metadaten über einen Webserver veröffentlicht werden kann. Der Zugriff erfolgt ausschließlich über eine HTTP-URL. Vorgesehen ist ein solches Repository nur für die SAS Studio spezifischen Snippets (SAS Codeschnipsel, nicht zwingenderweise Makros) und Tasks (Anwendungsroutinen mit eigener Oberfläche). Zwar können auch andere Dateien in den XML-Metadaten referenziert werden, dies ist jedoch nach Meinung des Autors kein sinnvolles Vorgehen zum Austausch von Standardmakros. Die Erstellung eigener Snippets und Tasks ist nicht Gegenstand dieses Beitrags und kann z.B. in [8] nachgelesen werden.

5 Fazit

SAS Studio ist auf jeden Fall eine Option für künftige Entwicklungsprojekte, kann allerdings aufgrund seiner Architektur nicht wirklich mit einer Rich Client-Anwendung wie SAS Enterprise Guide konkurrieren.

Die Erstellung und Verteilung eigener Auswertungen (Tasks) in SAS Studio bietet eine interessante Möglichkeit, Programmlogik zwischen Benutzergruppen auszutauschen.
Für SAS Standardmakros ist der Repository-Ansatz von SAS Studio aber eher ungeeignet, da er sich auf die eigenen Tasks und Snippets (nur Programm-Bausteine, nicht zwingenderweise Makros) beschränkt.

Für die verteilte Zusammenarbeit ist ein verteiltes Versionskontrollsystem sehr förderlich, weil damit viele Arbeitsabläufe bereits auf Infrastrukturebene unterstützt werden. Der Standard ist hier Git, das auch zum Teil vom SEG aus nutzbar ist. Viele frei verfügbare Clients und Dokumentation erleichtert hier den Ein- bzw. Umstieg.
Die projektspezifische Konfiguration von SAS Workspace Servern kann für beide Werkzeuge genutzt werden, was den Entwicklern die Freiheit gibt, das Werkzeug ihrer Wahl im gemeinsamen Projekt zu nutzen. Damit sollte dem nächsten gemeinsamen SAS Entwicklungsprojekt nichts im Wege stehen.

Literatur

[1] S. Chacon, B. Straub (2014): Pro Git, 2.Auflage, Apress Media, California. https://git-scm.com/book/en/v2 [19.02.2018]

[2] J. Flynn et al. (2014): Check It Out! Versioning in SAS Enterprise Guide, in: Proceedings of SAS Global Forum 2014. http://support.sas.com/resources/papers/proceedings14/SAS179-2014.pdf [06.02.2018]

[3] Git Interfaces, frontends, and tools. https://git.wiki.kernel.org/index.php/Interfaces,_frontends,_and_tools [06.02.2018]

[4] P. Holland (2015): SAS Enterprise Guide or SAS Studio: Which is Best for You?, in: Proceedings of SAS Global Forum 2015. http://support.sas.com/resources/papers/proceedings15/2683-2015.pdf [19.02.2018]

[5] S. Muelling, J. Brower (2017): Choosing the Best Fit for Your Client/Server Architecture: SAS Studio versus SAS Enterprise Guide, in: Proceedings of SAS Global Forum 2017. http://support.sas.com/resources/papers/proceedings17/SAS0436-2017.pdf [06.02.2018]

[6] C. Ortseifen (2016): Einführung in bzw. Vorstellung von SAS Studio 3.4, in: Proceedings der 20. KSFE in Greifswald, Shaker-Verlag, Aachen. http://de.saswiki.org/images/9/96/20_KSFE_2016_Ortseifen_-_Einf%C3%BChrung_in_bzw_Vorstellung_von_SAS_Studio_3.4.pdf [07.02.2018]

[7] SAS Institute Inc. (2016): SAS Studio 3.6: Administrator's Guide. Cary, NC. http://documentation.sas.com/api/docsets/webeditorag/3.6/content/webeditorag.pdf [05.03.2018]

[8] SAS Institute Inc. (2016): SAS Studio 3.6: Developer's Guide to Writing Custom Tasks, Cary, NC. http://documentation.sas.com/api/collections/webeditorcdc/3.6/docsets/webeditordg/content/webeditordg.pdf [15.02.2018]