Das gedruckte Werk ist unter ISBN xxx.xxx.xxx im Shaker-Verlag erschienen.

SAS Enterprise Guide erweitern mit Add-Ins Anleitung zum Einstieg

Andreas Mangold
HMS Analytical Software GmbH

Rohrbacher Str. 26
69115 Heidelberg
andreas.mangold@hms-heidelberg.de

Marcus Pilz
J.-W.-Goethe-Universität
Frankfurt am Main
Kiesstraße 29
60486 Frankfurt am Main
marcus.pilz@gmx.de

Zusammenfassung

Zunächst geht es um die verschiedenen Einsatzgebiete für SAS Enterprise Guide-Add-Ins. Dabei werden entsprechende Beispiele vorgestellt, die von SAS, den Referenten oder von anderen entwickelt wurden. Im zweiten Teil des Beitrags wird erläutert, wie SEG-Add-Ins mit Microsoft Visual Basic entwickelt und in SEG eingebunden werden können. Es geht uns nicht darum, die vorhandene Dokumentation (siehe Referenzen) zu ersetzen, sondern die Zusammenhänge zu beschreiben und mit einer Vorlage den Start in die Add-In-Programmierung zu erleichtern. Im Anhang findet sich eine Liste aller uns derzeit bekannten SEG-Add-Ins mit Bezugsquelle.

Keywords: SAS Enterprise Guide, Add-Ins, Integrated Object Model, IOM, Visual Basic.

1 Einführung und Übersicht

1.1 Einführung

SAS Enterprise Guide (kurz „SEG“) bietet SAS-Anwendern erstmals eine deutschsprachige grafische Benutzeroberfläche zur einfachen Bedienung der analytischen Softwaremodule von SAS und erzeugt komfortabel Ergebnisse in HTML-, Word- und PDF-Dateien. SAS hat dadurch ein Gesicht, das es auch für Anwender interessant macht, die individuell Auswertungen und Analysen erstellen wollen, deren primäre Aufgabe aber nicht das Programmieren ist.

Die Stärken von SAS bleiben erhalten: Man kann auf alle Server zugreifen, auf denen SAS mit Integration Technologies installiert ist und auf den lokalen PC mit SAS. Da SEG im Hintergrund SAS-Programme generiert und auch das Ausführungsprotokoll speichert, sind die Abläufe wiederholbar und dokumentiert.

SAS hat angekündigt, dass ab der Version 9 von SAS die Version 3.0 von SEG den SAS Display Manager ersetzen wird und damit die zentrale Benutzeroberfläche für Auswertungen und Analysen wird.

Nicht alle SAS-Prozeduren können derzeit jedoch von der komfortablen Benutzeroberfläche von SEG aufgerufen werden. Häufig wird SAS von den Anwenderunternehmen und -organisationen auch erweitert um eigene Makropakete und sogar eigene SAS-Prozeduren, oder es gibt Standardabläufe, die zu parametrieren sind. Für solche Aufgaben gibt es in SEG zwar das Codefenster, dafür benötigen die Anwender dann aber wieder weitergehende Programmierkenntnisse.

Als Lösung bieten sich Add-Ins für SEG an (im Folgenden „SEG-Add-Ins“ genannt). Add-Ins sind Anwendungsroutinen, die sich nahtlos in SEG integrieren lassen. Sie unterscheiden sich für den Anwender nicht von den in SEG eingebauten Anwendungsroutinen, die immer einen Dialog für die Parametrierung sowie einen Generator für ein SAS-Programm besitzen. SAS hat Enterprise Guide eine Schnittstelle für die Einbindung solcher Add-Ins als Microsoft COM-Objekte mitgegeben, die man zum Beispiel mit Microsoft Visual Basic entwickeln kann.

1.2 Übersicht

Kenntnisse als Anwender von SAS Enterprise Guide werden in diesem Papier vorausgesetzt. Für den Abschnitt 3 werden auch Kenntnisse in Microsoft Visual Basic vorausgesetzt.

In Abschnitt 2 geht es zunächst um die verschiedenen Einsatzgebiete für SEG-Add-Ins. Dabei wollen wir auch entsprechende Beispiele vorstellen, die von SAS, den Referenten oder von anderen entwickelt wurden.

Im zweiten Teil des Papers (Abschnitt 3) erläutern wir, wie SEG-Add-Ins mit Microsoft Visual Basic entwickelt und in SEG eingebunden werden können. Es geht uns nicht darum, die vorhandene Dokumentation [1] zu ersetzen, sondern die Zusammenhänge zu beschreiben und mit einer Vorlage den Start in die Add-In-Programmierung zu erleichtern

In Abschnitt 4 sind einige Referenzen zu finden. Einfache Verweise im Text in der Form „[X]“ verweisen auf eine Referenz im Abschnitt 4. Verweise in der Form „(3.3)“ verweisen auf einen Abschnitt des vorliegenden Beitrags.

Der Anhang listet alle uns derzeit bekannten SEG-Add-Ins mit Bezugsquelle auf.

Alle Informationen und Beispiele in diesem Beitrag beziehen sich auf die Version 2.0 von SAS Enterprise Guide.

Nicht alle hier vorgestellten Add-Ins und Beispielprogramme wurden von den Autoren entwickelt und getestet, wir übernehmen daher keinerlei Gewährleistung für die hier vorgestellten Programme und Informationen.

2 Einsatzmöglichkeiten und Beispiele

Wo ist der Einsatz von SEG-Add-Ins sinnvoll? Hier werden fünf Bereiche vorgestellt und jeweils Beispiele genannt.

Jedes SEG-Add-In präsentiert sich dem Anwender als „Anwendungsroutine“, genauso wie die in SEG eingebauten Anwendungsroutinen. Wenn der Anwender eine Anwendungsroutine aufruft, entweder über die „Liste der Anwendungsroutinen“ oder über das Menü, erhält er einen Dialog, der normalerweise in Form eines Assistenten gestaltet ist, in dem er in mehreren Schritten die Anwendungsroutine parametrieren kann. Wenn der Anwender auf „Ausführen“ klickt, wird im Hintergrund SAS-Code erzeugt und dieser ausgeführt.

Ein großer Vorteil der SEG-Add-Ins ist, dass sie genauso nahtlos in SEG integriert sind wie die von SAS mitgelieferten Anwendungsroutinen.

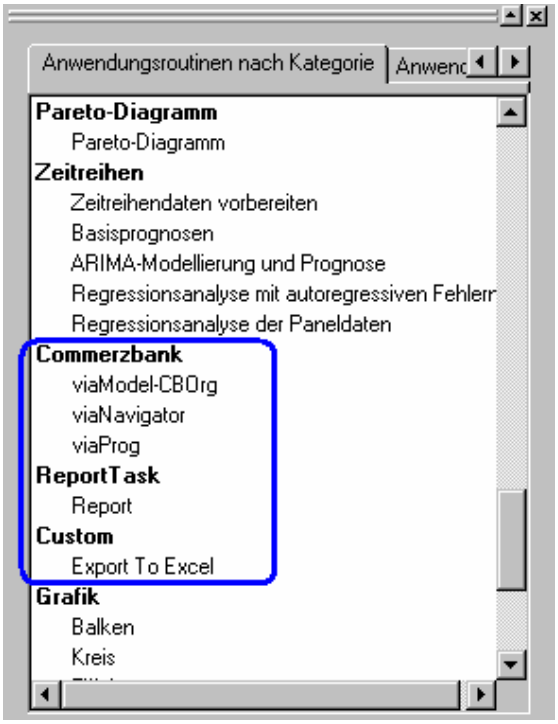

Abbildung 1: SEG-Add-Ins fügen sich nahtlos in die Liste der Anwendungsroutinen ein

2.1 Unterstützung bei organisationsspezifischen Abläufen

SEG bietet für jeden etwas, der Analysen und Auswertungen machen möchte. In jedem Unternehmen gibt es jedoch spezielle wiederkehrende Aufgaben und spezielle Vorschriften für deren Bearbeitung. In SEG gibt es dafür das Codefenster, in dem man bestehenden oder von SEG erzeugten Code übernehmen, modifizieren und ausführen kann. Dafür werden allerdings Programmierkenntnisse benötigt.

Mit SEG-Add-Ins können solche Abläufe standardisiert und Anwender, die nur wenig Programmierkenntnisse haben, unterstützt werden. Beispiele sind in den folgenden Abschnitten zu finden. Ein Anwendungsbeispiel findet sich auch in [9].

2.1.1 SAS-Makros parametrieren

Häufig wurden organisationsspezifische Abläufe schon bisher in Form von SAS-Makros bereitgestellt, die die komplette Datenverwaltung und die Auswertung erledigen und dabei verschiedene SAS-Prozeduren und DATA-Schritte aufrufen. Der Anwender muss solche Makros in seine SAS-Entwicklungsumgebung einbinden, mit Parametern versorgen, aufrufen und die Ergebnisse verwalten. Diese Aufgaben können mit Hilfe von SEG-Add-Ins vereinfacht werden.
Makros einbinden: erfolgt über die Einbindung entsprechender SAS-Optionen (SASAUTOS etc.) im SEG-Administrator: In den Server-Eigenschaften können Startanweisungen für den SAS-Server mitgegeben werden, am Besten eine %include-Anweisung. Wenn nur wenige Makros eingebunden werden sollen, ist es stattdessen auch möglich, den Code des jeweiligen Makros direkt in den vom Add-In erzeugten Code einzufügen. Das SEG-Add-In „Optimize“ (siehe 2.5.1) arbeitet nach diesem Verfahren.
Parametrieren und aufrufen: Hierfür können Add-Ins verwendet werden, in denen der Anwender die Parameter angibt und die dann den Makro-Aufruf erzeugen, siehe Beispiel im Abschnitt 2.1.2.
Ergebnisse verwalten: Dafür eignet sich das SEG-Projekt, in dem man sowohl die Aufrufe als auch die erzeugten Auswertungsergebnisse speichern kann. Projekte können sogar automatisch aktualisiert werden, siehe 2.1.4.

2.1.2 Beispiel: Organisationseinheiten ermitteln

Für eine Bank haben wir ein Add-In für ein Makro entwickelt, mit dem man zu einem gegebenen Kunden die Organisationseinheiten ermitteln kann, die für den Kunden zuständig sind. Die Anwendungsroutine erzeugt einfachen DATA-Schritt-Code und verwendet dabei ein unternehmensspezifisches SAS-Makro namens %CBORG.

Abbildung 2 zeigt, wie man die Variablen, die die Kundennummer enthalten, den entsprechenden Spaltenrollen zuweist. In Abbildung 3 kann man auswählen, welche Ebenen der Organisationshierarchie berücksichtigt werden sollen und wohin das Ergebnis geschrieben werden soll. Abbildung 4 zeigt den erzeugten SAS-Code.

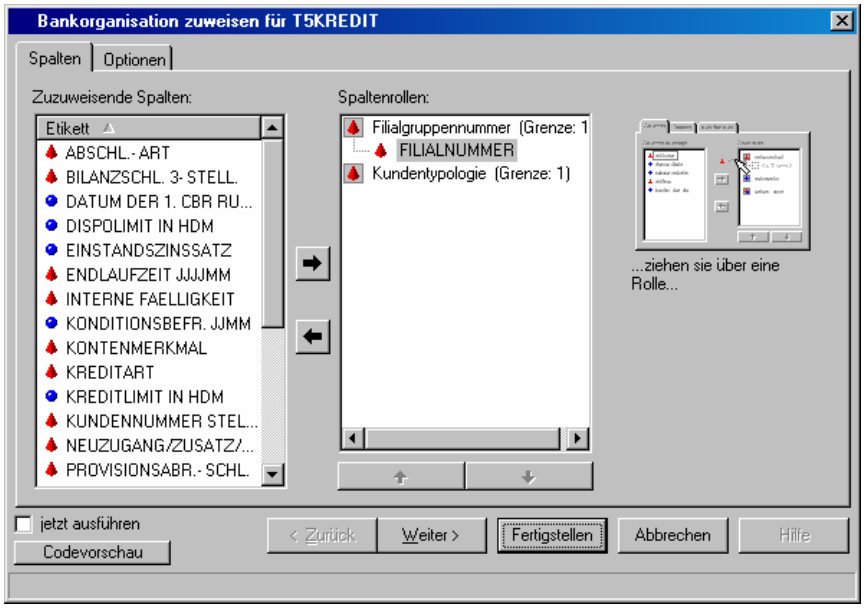

Abbildung 2: Organisationseinheiten ermitteln – Spaltenrollen festlegen

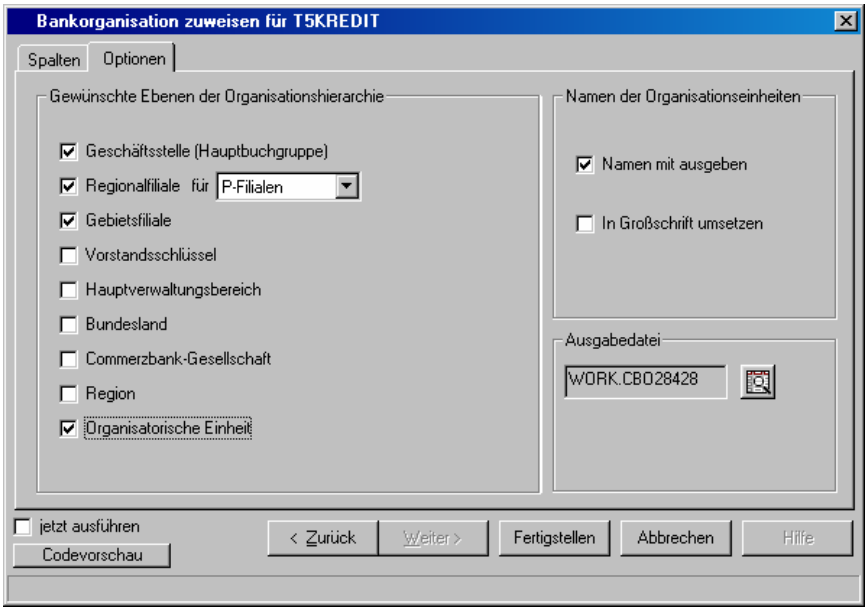

Abbildung 3: Organisationseinheiten ermitteln – Optionen einstellen

```
%LET _EGTASKLABEL = %NRBQUOTE(viaModel-CBOrg);
/*----------------------------------------------------------
   Erzeugt von SAS Enterprise Guide
   Am Montag, 16. Februar 2004   um 14:03:42
   Anwendungsroutine: CBOrg
   Server           : Local
   Datei            : BEISPIEL.T5KREDIT
----------------------------------------------------------*/

/*----------------------------------------------------
   Ausgabedatei bzw. View löschen,
   bevor die neue Ausgabedatei erzeugt wird.
*/

PROC SQL;
   DROP VIEW WORK.CBO28428;
   DROP TABLE WORK.CBO28428;
QUIT;

/*----------------------------------------------------------
   Ein DATA-Schritt erzeugt die Ausgabedatei.
   und erstellt dabei die gewünschten Variablen
   der Organisationshierarchie der Commerzbank.
*/

DATA WORK.CBO28428(LABEL="T5KREDIT mit Bankorganisation");
   SET BEISPIEL.T5KREDIT;
%CBORG (HBGRUPPE,JA,FILNR,,,FILTYP='0P');
%CBORG (REGFIL,JA,FILNR,,,FILTYP='0P');
%CBORG (GEBIET,JA,FILNR,,,FILTYP='0P');
%CBORG (ORGEINH,JA,FILNR,,,FILTYP='0P');
RUN;

%LET _EGTASKLABEL =;
```

Abbildung 4: Organisationseinheiten ermitteln – erzeugter Code

2.1.3 Idee für ein SEG-Add-In „Makrostarter“

Wenn in einem Anwenderunternehmen bereits viele Makros vorliegen, liegt es nahe, ein Add-In zu entwickeln, dem man für jedes vorhandene Makro nur noch mitteilen muss, wo es liegt, wie es heißt, was es tut und welche Parameter es benötigt. In einer Steuerdatei werden alle Makros, die auf diese Weise gestartet werden sollen, dokumentiert und das Add-In ermöglicht die Auswahl eines Makros und zeigt Dokumentation und entsprechende Parameterdialoge an.

Vorteil dieser Vorgehensweise wäre, dass man nicht mehr für jedes Makro ein eigenes Add-In entwickeln muss, sondern dass große Bestände von bereits vorhandenen SAS-Programmen auf diese Weise erschlossen werden können.

2.1.4 Exkurs: OLE-Automation

Im Zusammenhang mit organisationsspezifischen Abläufen ist noch interessant, dass man SEG-Projektdateien auch fernsteuern kann mit Hilfe der Technologie „OLE-Automation“. Diese Technologie kann dazu verwendet werden, Zeit gesteuert oder in Abhängigkeit von externen Ereignissen Anwendungsroutinen in SEG-Projektdateien automatisch auszuführen und die Ergebnisse abzulegen.

Dies ist jedoch nicht Thema des vorliegenden Vortrags, Informationen siehe [7] sowie in der Hilfe von SEG unter dem Stichwort „Automation“.

2.2 Zusätzliche SAS-Prozeduren als Anwendungsroutinen

SEG enthält bereits mehr als 60 verschiedene Anwendungsroutinen für eine große Zahl von SAS-Prozeduren aus den Bereichen Datenverwaltung, Reporting, Analyse und Grafik. Nicht alle SAS-Prozeduren können jedoch über Anwendungsroutinen aufgerufen werden. SEG-Add-Ins ermöglichen es hier, weitere SAS-Prozeduren für Anwender zu erschließen, die nicht programmieren wollen.
Im Folgenden werden einige Beispiele vorgestellt. Es gibt aber sicher noch wesentlich mehr SAS-Prozeduren, die häufig angewandt werden und bei denen man sich trotzdem nicht alle Optionen merken möchte.

2.2.1 PROC SORT

Sortieren kann man zwar in SEG auch ohne Add-In, aber nur im Rahmen einer Abfrage. Dabei wird PROC SQL erzeugt. Wenn man mit PROC SORT sortieren möchte, zum Beispiel weil man spezielle Sortieroptionen braucht (etwa „FORCE“), kann man das von SAS zur Verfügung gestellte SEG-Add-In „Sort Data“ verwenden.

Wir haben das Add-In ins Deutsche übersetzt und Funktionalität ergänzt, siehe Abschnitt 3. Die folgenden beiden Abbildungen zeigen die beiden Dialoge dieses Add-Ins. Alle Eigenschaften, die man von SEG-Anwendungsroutinen gewohnt ist, sind hier enthalten:

- Zuweisung von Spalten der ausgewählten Datei zu Spaltenrollen,
- Spalteneigenschaften (hier „Sort Properties“),
- Optionen der SAS-Prozedur als Steuerelemente im Dialog,
- Auswahl der Ausgabedatei mit Hilfe eines Standarddialogs,
- Codevorschau.

Es gibt in der Version 2.0 von SEG aber auch noch kleine Beschränkungen für die Funktionalität von Add-Ins:

- Die Codevorschau ist ein modales Fenster, d.h. man kann nicht in den Dialog klicken, während die Codevorschau angezeigt wird,
- Die Option „jetzt ausführen“ ist zwar sichtbar, aber nicht aktiv, der erzeugte SAS-Code wird immer sofort ausgeführt.

Der Visual-Basic-Code für dieses Add-In wird von SAS mitgeliefert und eignet sich gut als Vorlage für selbst entwickelte Add-Ins, siehe [1].

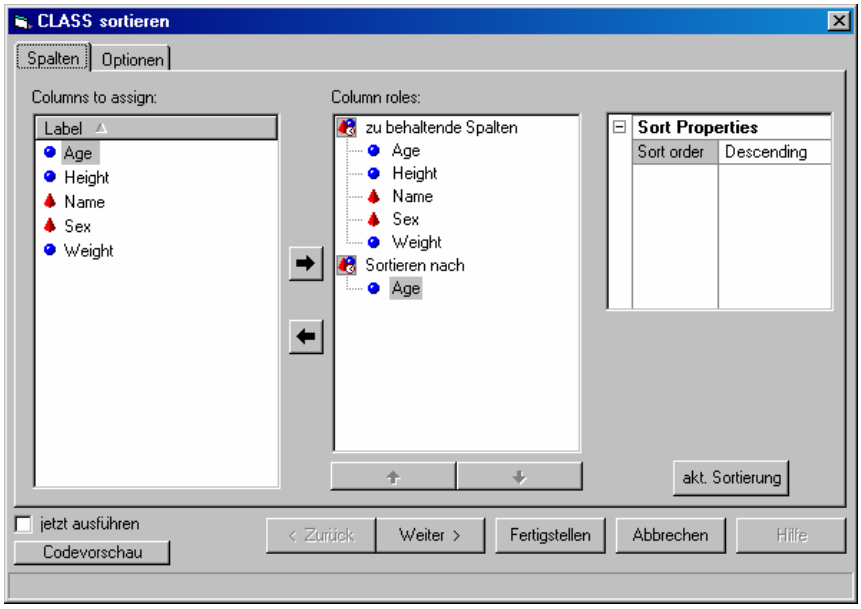

Abbildung 5: Anwendungsroutine für PROC SORT – Spaltenauswahl

Abbildung 6: Anwendungsroutine für PROC SORT – Optionen festlegen

2.2.2 PROC REPORT

Reports kann man mit SEG mit PROC PRINT („Listenbericht“) und mit PROC TABULATE („Zusammenfassungstabellen“) erstellen. Wer erweiterte Listenberichte mit PROC REPORT erstellen möchte, benötigt ein Add-In. Ein Add-In wird von SAS verteilt [3], ein anderes kann man bei einer Consulting-Firma kaufen [10].

2.2.3 PROC HPF

PROC HPF (High Performance Forecasting) ist eine SAS-Prozedur für die Modellselektion bei Zeitreihenanalysen und für die automatische und schnelle Analyse großer Mengen von Zeitreihen. Im SUGI-Paper [4] wird ein Add-In für diese Prozedur beschrieben.

2.3 Anbindung an Datenbanken und Data-Warehouse Umgebungen

SEG bietet die Anbindung an viele verschiedene Datenformate, Datenbanksysteme und Rechnerplattformen. Dies ist gut brauchbar, wenn der

Anwender weiß, wo er seine Daten finden kann. In sehr umfangreichen Datenbeständen, wie zum Beispiel Data-Warehouse-Umgebungen, ist jedoch die Möglichkeit zur Suche im Datenkatalog (auch Metadaten genannt) notwendig. Wer noch nicht SAS9 anwendet, wo ein Metadatenserver integriert ist, kann Add-Ins verwenden.

2.3.1 Suche in externen Metadaten

Wenn die Datenbestände sehr groß sind, wie in Data-Warehouse-Umgebungen, können nicht alle Datenbestände mittels Librefs der aktuellen Sitzung zugeordnet werden, denn das wäre sehr unübersichtlich.

Für die Commerzbank haben wir ein Add-In „viaNavigator“ entwickelt, das einen externen Datenkatalog auf dem Großrechner durchsucht, die Berechtigungen des angemeldeten Anwenders berücksichtigt und für die ausgewählten Daten einen Libref erzeugt, so dass die entsprechenden Daten an die SEG-Sitzung angebunden werden. Abbildung 7 zeigt ein Beispiel, weitere Informationen sind in [8] zu finden.

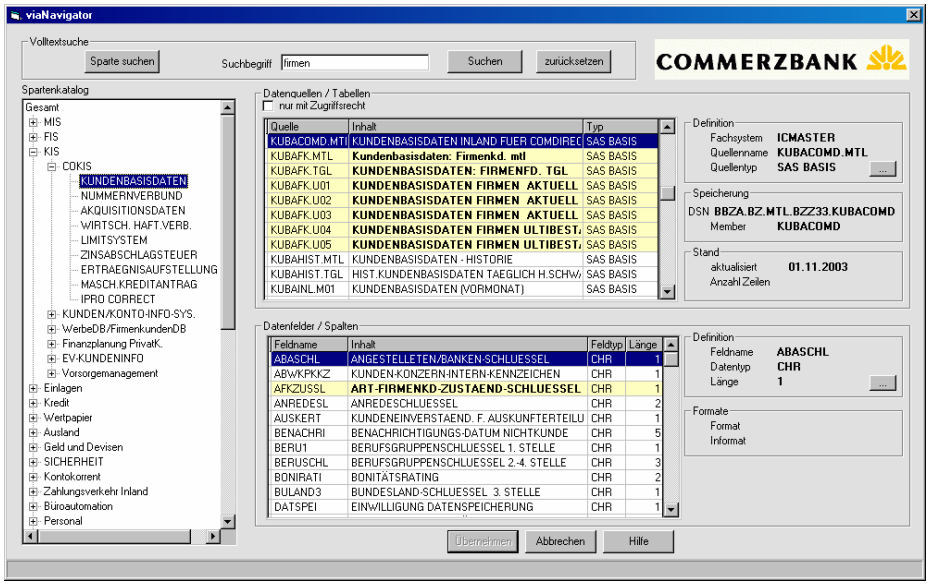

Abbildung 7: Datenkatalog durchsuchen mit viaNavigator

2.3.2 Suche in organisationsspezifischen Datenbanken

Das Abfragewerkzeug in SEG ist außerordentlich flexibel und komfortabel. Allerdings ist es notwendig, dass der Anwender das Datenmodell versteht und anwenden kann und z.B. weiß, welche Tabellen er wie verbinden muss. Denn in der aktuellen Version von SEG kann das Abfragewerkzeug noch nicht erkennen, welche Beziehungen zwischen den Tabellen einer Datenbank bestehen.

Eine Möglichkeit, dies zu lösen, ist es, mit Datenbankviews zu arbeiten, die die am häufigsten verwendeten Datensichten zur Verfügung stellen. Hier bietet es sich aber auch an, mit Add-Ins zu arbeiten, wenn immer wiederkehrende Selektionen komfortabel parametriert werden sollen.

2.3.3 Suche im SAS-Datenkatalog

SAS hat ein Add-In entwickelt (siehe [2]), mit dem auf allen angebundenen Servern alle Bibliotheken nach bestimmten Schlüsselbegriffen durchsucht werden können. Gesucht wird dabei nicht in den Daten, sondern nur in den Namen und Beschreibungen von Tabellen und Spalten. Dadurch kann man Daten finden, sofern sie über einen Libref an die aktuelle SEG-Sitzung angebunden sind. Wahlweise liefert das Add-In das Ergebnis als Bericht oder als SAS-Datei.

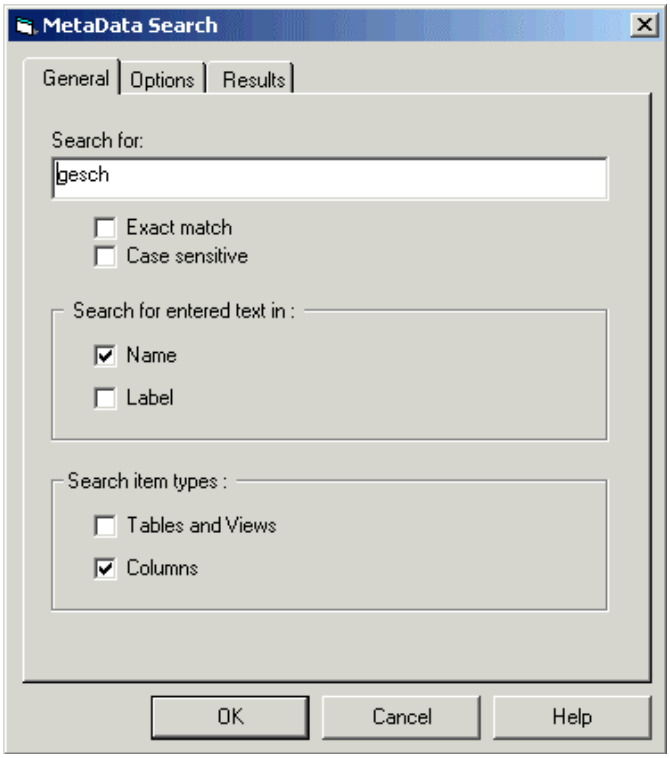

Abbildung 8: Suche im SAS-Datenkatalog

2.4 DATA-Schritt einbinden

Ab und zu haben wir von SAS-Anwendern gehört, SEG eigne sich nicht für ihre Arbeit, weil er keinen DATA-Schritt-Code erzeuge. Tatsächlich gibt es keine Anwendungsroutine „DATA-Schritt“, was sicher seinen Grund darin hat, dass der DATA-Schritt sehr flexibel ist und man nicht mit einem Dialog alle Anforderungen an die DATA-Schritt-Programmierung erfüllen kann. Auch sind die wichtigsten Aufgaben, die viele Anwender mit DATA-Schritt-Programmierung lösen, in SEG in der Routine „Abfrage“ mit SQL gelöst.

Wenn es Gründe gibt, die in einem spezifischen Fall gegen die Verwendung von PROC SQL sprechen (z.B. Performance oder die Verwendung spezieller Algorithmen), so muss man den DATA-Schritt von Hand codieren oder man kann, insbesondere in dem in Abschnitt 2.3.2 geschilderten Fall, ein entsprechendes Add-In entwickeln. Ein Beispiel für ein Add-In, das DATA-Schritt-Code erzeugt, ist auch in Abschnitt 2.1.2 zu finden.

2.5 Hilfsprogramme

Nützliche Hilfsprogramme vereinfachen die Arbeit. Auf die Metadatensuche wurde bereits in Abschnitt 2.3.3 hingewiesen.
Es steht auch ein Add-In für den Aufruf von PROC CONTENTS zur Verfügung (siehe [3]).

2.5.1 Speicherplatz sparen mit dem Optimize-Add-In

SAS hat auf der DISK 2003 in Bonn das Add-In „Optimize“ vorgestellt [2], das SAS-Dateien komprimieren kann. Es steht als Makro zur Verfügung, das von einem SEG-Add-In parametriert und aufgerufen werden kann.
Optimize bietet folgende Möglichkeiten:

- Abschneiden von Charactervariablen auf die maximal genutzte Länge. Dies ist besonders wichtig, wenn man Daten mit der Microsoft-Jet-Engine aus Exceldateien importiert, weil dabei die Spaltenlänge immer mindestens auf 255 gesetzt wird, was sehr viel Platz verbraucht.
- Einschränken der Präzision von numerischen Variablen, so dass weniger Platz benötigt wird.
- Komprimieren der Ausgabedatei mit der COMPRESS-Option, so dass der verbrauchte Platz noch weiter eingeschränkt wird.
- Report zur Dokumentation der Änderungen.

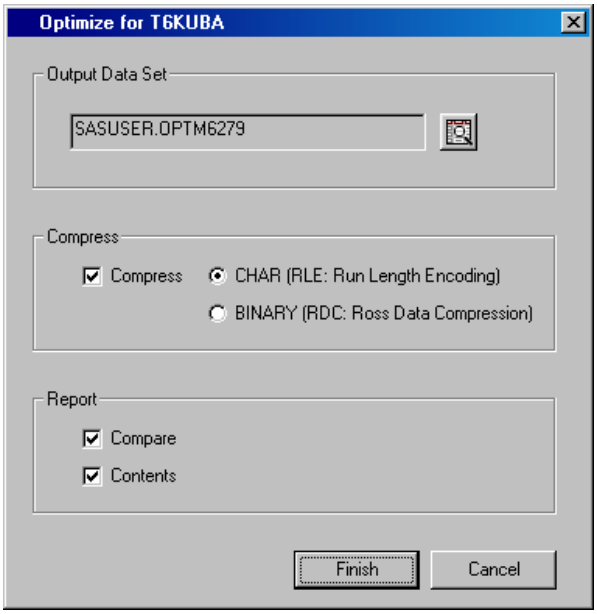

Abbildung 9: Komprimieren von Dateien mit SQUEEZE

3 Entwicklung von SEG-Add-Ins mit Microsoft Visual Basic

3.1 Übersicht

Im Folgenden geht es darum, einen schnellen Einstieg in die Entwicklung von SEG-Add-Ins zu geben und die Grundprinzipien zu erläutern. Die Dokumentation von SAS soll dadurch nicht ersetzt werden. Hingewiesen sei auf die Referenzen [1] und [4].
Die Autoren stellen auf Anfrage ein Beispielprojekt zur Verfügung, auf das im Folgenden Bezug genommen wird. Das Beispielprojekt dient als Kopiervorlage. Es handelt sich um das von SAS in [1] veröffentlichte Add-In „SortTask“, das von uns ins Deutsche übersetzt und um die Trace-Funktionalität (Ersatz für Debugger) und ein Beispiel für den Submit von SAS-Code erweitert wurde.

In Abschnitt 3.2 sind grundlegende Informationen für den Programmierer eines Add-Ins zu finden. 3.3 beschreibt das Zusammenspiel der verschiede-

nen Komponenten bei der Add-In-Entwicklung und ist als Ergänzung zur Dokumentation von SAS gedacht. Abschnitt 3.4 gibt einen Überblick der von SAS bereitgestellten Oberflächenelemente (Controls) und Abschnitt 3.5 behandelt das Thema der Registrierung von Add-Ins bei SEG.

3.2 Aufgaben des Visual-Basic-Programmierers

3.2.1 Demoprojekt in Betrieb nehmen

Im Folgenden erläutern wir, wie das Demoprojekt, das bei den Autoren erhältlich ist, in Betrieb genommen werden kann.

Voraussetzung ist eine Installation Microsoft Visual Basic Version 6.0 (im Folgenden „VB"), enthalten in Microsoft Visual Studio Version 6.0. Es sollte das Service-Pack 5 aufgespielt werden.
Die Entwicklung von Add-Ins ist auch mit anderen Entwicklungswerkzeugen möglich (zum Beispiel Microsoft Visual C++), mit Visual Basic ist es einfacher. Ab Version 3.0 von SEG wird die Entwicklung von Add-Ins auch mit dem .NET-Framework möglich sein, da die Version 3.0 mit diesem Framework entwickelt wird.
Um das Beispielprojekt in Betrieb zu nehmen, gehen Sie wie folgt vor:
Alle Dateien des Projekts in einen neuen Ordner kopieren, Projekt DemoTask.vbp öffnen und das Add-In erzeugen mit „Datei – DLL erstellen – DemoTask.dll". Danach kann das Add-In probehalber bei SEG registriert und ausgeführt werden (siehe Abschnitt 3.5).
Um nun ein eigenes Projekt daraus zu machen: Projekt schließen und die drei Dateien namens Demo… umbenennen.
Das Projekt (.vbp) öffnen. Durch das Umbenennen gibt es Fehlermeldungen. Im Projektfenster von VB müssen die umbenannten Dateien neu eingefügt werden (rechte Maustaste – „hinzufügen" – „Datei hinzufügen…"). Danach zeigt das Projektfenster zwei Formulare (der umbenannte Dialog und CodePreview), ein Modul (General) und zwei Klassenmodule (umbenanntes Class-File und SortOptions). Ändern Sie nun noch den Klassennamen des umbenannten Klassenmoduls und des umbenannten Dialogs (Doppelklick auf das Modul, dann Namen im Eigenschaftenfenster ändern).
Es sind jetzt alle notwendigen Komponenten eingebunden. Wenn Sie jedoch mit einem neuen Projekt starten und nicht die Demo-Vorlage verwenden, müssen Sie über „Projekt – Verweise" zumindest die Type Library SASAIOBJ.DLL aus dem Hauptverzeichnis der SEG-Installation einbinden. Um die Oberflächenelemente (siehe 3.4) zu verwenden, müssen

diese mit „Projekt – Komponenten“ zusätzlich eingebunden werden (fangen alle mit „SAS“ an).
Projekteigenschaften ändern: Projekttyp ist ActiveX-DLL, Projektname und Projektbeschreibung können angepasst werden. Auf der Karteikarte „Erstellen“ kann der Anwendungsname, Versionsnummer und Copyright etc. ergänzt werden.
An einigen Stellen muss nun der Code geändert werden: Alle Vorkommen von „Demo“ im Code ändern, wenn Sie die Objekte umbenannt haben. Zusätzlich kann der Logfilename geändert werden (im Klassenmodul in SEGAddIn_Connect). Achtung: das Verzeichnis, in dem das Logfile liegt muss beim Registrieren und beim Betrieb des Add-Ins in SEG vorhanden sein!
Beim Erstellen des Add-Ins mit „Datei – DLL erstellen“ kann der DLL ein neuer Name gegeben werden.

3.2.2 Projektaufbau

Um nun das Add-In nach den eigenen Vorstellungen zu modifizieren, ist folgendes zu beachten:
Im Klassenmodul (ehemals Demo.cls) sind alle Interfacemethoden implementiert, die notwendig sind, damit SEG mit dem Add-In kommunizieren kann, siehe Abschnitt 3.3.1. Man kann dazu auch Callback-Methoden sagen, da sie „von außen“ von SEG aufgerufen werden. Einige der Interfacemethoden werden in unserem Beispiel nicht benutzt und sind als Leerroutinen nur der Vollständigkeit halber vorhanden.
Im Formular (ehemals DemoTaskDialog.frm) ist der Dialog programmiert, es wird hierzu unter anderem auf die Add-In-Framework-Services zurückgegriffen, um auf SEG zuzugreifen, siehe Abschnitt 3.3.1. Das Klassenmodul setzt in der Methode SEGAddIn_Connect einige globale Variablen im Formular.
Das Formular CodePreview kann ohne Änderung verwendet werden.
Das Modul General.bas enthält Konstanten und Low-Level-Routinen, die in den anderen Modulen verwendet werden.
Das Klassenmodul SortOptions.cls ist nur speziell in diesem Beispielprojekt nötig, weil darin die Eigenschaften für jede Sortiervariable gehalten werden. Es kann als Vorlage dienen, wenn man für jede einer Rolle zugewiesene Variable weitere Eigenschaften festhalten möchte.
Ein direkter Zugriff auf das SAS Integrated Object Model (IOM) ist in unserem Beispiel nicht implementiert, aber möglich, siehe Methode SEGAppli-

cation.GetWorkspace. Stattdessen verwenden wir immer die Framework Services.

3.2.3 Erfahrungen bei der Entwicklung von SEG-Add-Ins

Das Registrieren des Add-Ins bei Windows und SEG muss man nur einmal machen, so lange man nichts an den Interfacemethoden von SEGAddIn und SEGTaskInfo ändert (was selten nötig ist). Wenn man die DLL neu erzeugen möchte, muss man SEG schließen, nach dem Erzeugen der DLL und dem Neustart von SEG steht das geänderte Add-In zur Verfügung.
Wenn Änderungen in den Interfacemethoden von SEGAddIn oder SEGTaskInfo vorgenommen werden, ist es notwendig, das Add-In neu zu registrieren.
Leider konnten wir den Debugger von VB nicht mit dem Add-In aktivieren. Für Hinweise sind wir sehr dankbar. Stattdessen schreiben wir ein Tracefile, das wir in der Methode SEGAddIn_Connect öffnen. Es ist wichtig, dass das Verzeichnis, in das diese Datei geschrieben wird, beim Registrieren und beim Betrieb des Add-Ins auch vorhanden ist.
Das asynchrone Hintergrund-Submit (siehe 3.3.6) ist recht kompliziert und wird hoffentlich in der nächsten Version durch ein synchrones ergänzt. Auch haben wir es bisher nicht geschafft, verlässlich an den SASLOG dieser Submits zu kommen. Für Hinweise sind wir sehr dankbar.
Beim Lesen von SAS-Tabellen mit SEGAccessor traten Probleme mit überschriebenen Speicherbereichen auf, die umgangen werden konnten (siehe 3.3.6).
Zur Laufzeit tritt manchmal das Problem auf, dass eine Komponente fehlt, siehe [6].
Es gibt derzeit die dokumentierte Beschränkung, dass in selbst entwickelten Add-Ins das Codefenster immer modal ist, d.h. man muss es schließen, um wieder an den Dialog zu kommen.
Es gibt derzeit die dokumentierte Beschränkung, dass die Option „Jetzt ausführen“ nicht deaktiviert werden kann. Es ist also nicht möglich, die Benutzereingaben in den Projektbaum zu übernehmen, ohne den erzeugten SAS-Code auszuführen.

3.3 Zusammenspiel der Komponenten

3.3.1 Übersicht

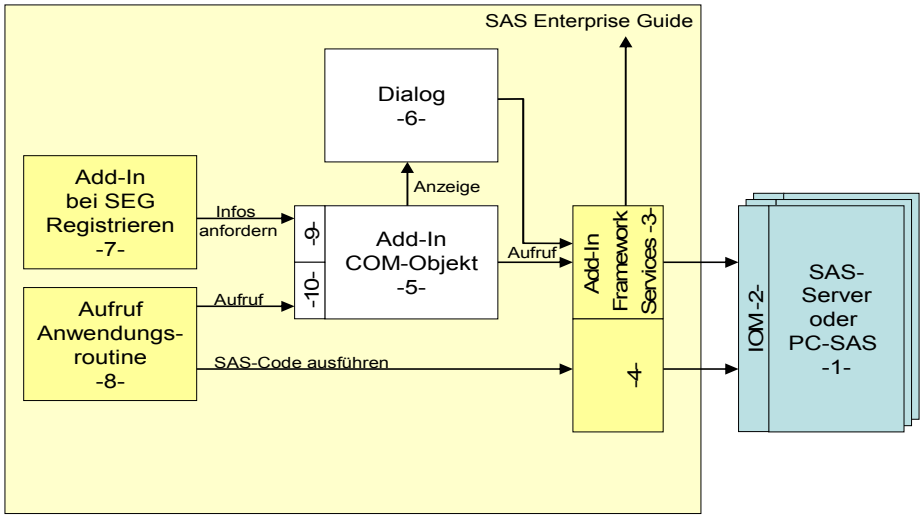

Abbildung 10: Übersicht über das Zusammenspiel der Add-In-Komponenten

Legende:
-1- und -2-: Serverkomponenten,
-3-, -4-, -7-, -8-: Komponenten von SAS Enterprise Guide,
-5-, -6-, -9-, -10-: Vom Add-In-Entwickler zu programmierende Komponenten. Entwickelt werden alle diese Komponenten in einem Visual-Basic-Projekt, zur Laufzeit stecken Sie alle in einer DLL.

Beschreibung der Komponenten:

1. SAS-Server: Bekanntlich können von SEG aus die lokale PC-SAS-Installation sowie beliebige SAS-Server angesprochen werden, sofern Integration Technologies installiert ist1. Es ist auch möglich, dass ein Add-In mit mehreren Servern kommuniziert. Auf welchem Server der erzeugte SAS-Code ausgeführt wird (-8-) wird jedoch

[1] Eine Lizenzierung von SAS Integration Technologies ist nur erforderlich, wenn der SAS Server nicht auf dem gleichen Rechner wie SEG läuft.

normalerweise von SEG bzw. vom Anwender und nicht vom Add-In festgelegt.

2. IOM: Integrated Object Model, Aufrufschicht für den Aufruf von Services des SAS-Servers, das zugehörige SAS-Modul heißt „Integration Technologies“.
3. Add-In Framework Services: Wenn das Add-In spezielle Aufgaben zu erledigen hat und dazu auf SEG oder auf Daten auf dem SAS-Server zugreifen muss, so kann es die Methoden der COM-Schnittstellen SEGApplication, SEGData, SEGAccessor und SEGColumnInfo verwenden, zum Beispiel um die Spalten oder Datensätze der zu verarbeitenden Tabelle zu erhalten (siehe Abschnitt 3.3.5) oder um SAS-Code zu submittieren (3.3.6). Für den normalen Prozess der Codegenerierung und –ausführung ist dies jedoch nicht notwendig.
4. Inwieweit SEG selbst über die Framework-Services mit dem SAS-Server kommuniziert oder direkt auf IOM zugreift, ist nicht bekannt.
5. Add-In-COM-Objekt: Ein COM-Objekt ist eine DLL, die bestimmten Aufrufkonventionen (Interfaces) genügt, die im Windows-Betriebssystem registriert sind. Dadurch wird ein objektorientierter Zugriff möglich. Ein COM-Objekt, das als SEG-Add-In funktionieren soll, muss zwei Interfaces (-9-) für die Registrierung des Add-Ins in SEG implementieren und ein Interface (-10-) für den Aufruf des Add-Ins.
   Der Add-In-Programmierer muss die einzelnen vorgegebenen Methoden der Interfaces ausprogrammieren, den Dialog (-6-) aufrufen und bei Bedarf mit dem Add-In-Framework kommunizieren. Das Erzeugen der DLL wird von Microsoft Visual Basic erledigt. Das Registrieren der DLL im Betriebssystem und bei SEG muss derzeit vom Anwender des Add-Ins noch selbst erledigt werden (siehe 3.5).
6. Dialog: Hierbei handelt es sich um einen gewöhnlichen Windows-Dialog, der mit Visual Basic entwickelt werden kann. Alle verfügbaren Controls können verwendet werden. Für Standardaufgaben bei der Add-In-Entwicklung (z.B. Variablen zu Rollen hinzufügen) stellt SAS spezielle Controls zur Verfügung, siehe 3.4. Wie das Add-In-COM-Objekt selbst (-5-) kann auch der Dialog über die Add-In-Framework-Services mit SAS und mit SEG kommunizieren.
7. Add-In bei SEG Registrieren: Damit ein Add-In verwendet werden kann, muss es der Anwender unter „Extras – Anpassen“ bekannt

machen (siehe 3.5). Dabei fordert SEG bei dem Add-In bestimmte Informationen an, indem es die Interfacemethoden (-9-) aufruft.

8. Aufruf Anwendungsroutine: Dies ist ein komplexes Zusammenspiel zwischen Enterprise Guide und dem Add-In, wobei SEG das Add-In über die Interfacemethoden (-10-) aufruft. SEG übergibt dabei dem Add-In die notwendigen Informationen aus dem Projektbaum, das Add-In zeigt den Dialog (-6-) an, der Anwender macht seine Angaben, anschließend holt sich SEG aus dem Add-In den SAS-Code und übergibt ihn über IOM (-2-) an den SAS-Server (-1-). Näheres siehe Abschnitte 3.3.3 und 3.3.4.
9. Über die Methoden der Interfaces SEGAddIn und SEGTaskInfo liefert der Add-In-Programmierer Informationen zur Registrierung des Add-Ins an SEG, siehe 3.3.2.
10. Über die Methoden des Interfaces SEGTask kommuniziert SEG mit dem Add-In, siehe 3.3.3 und 3.3.4.

3.3.2 Methodenaufrufe bei der Registrierung des Add-Ins

Wenn der Anwender das Add-In bei SEG registriert (siehe 3.5), ruft SEG folgende Methoden in der angegebenen Reihenfolge auf. Der SEG-Add-In-Programmierer ist dafür zuständig, dass diese Methoden die richtigen Informationen zurückgeben, siehe rechte Seite der folgenden Liste. Die Methoden sind im Detail in [1] beschrieben.
Achtung: Diese Informationen speichert SEG nach der Registrierung intern. Wenn die Methoden geändert werden, muss das Add-In neu registriert werden!

SEGAddIn_Connect	Verbinden zum Add-In
SEGAddIn_GetName	Name des Add-Ins zurückgeben (zu finden unter Menüpunkt „Extras – Anpassen – Add-Ins“)
SEGAddIn_GetDescription	Beschreibung des Add-Ins zurückgeben
SEGTask_GetRequiresData	Zurückgeben, ob das Add-In eine Eingabetabelle braucht
SEGTaskInfo_GetTaskMajorVersion	Hauptversionsnummer (siehe Liste der Anwendungsroutinen, Anwendungsroutinen nach Name)
SEGTaskInfo_GetTaskMinorVersion	Unterversionsnummer (ebd.)
SEGTaskInfo_GetTaskType	Anwendungsroutine oder Codefenster

SEGTask_GetOptionsFlag In welche vorgegebene Kategorie der Liste der Anwendungsroutinen soll die Routine integriert werden?

SEGTaskInfo_GetTaskCategory Falls GetOptionsFlag = Custom: eigene Kategorie, d.h. Zwischenüberschrift in der Liste der Anwendungsroutinen festlegen, wird auch als Untermenü unter Menü „Analyse“ verwendet.

SEGTaskInfo_GetTaskName Name der Anwendungsroutine (kann sich vom Namen des Add-Ins unterscheiden, da ein Add-In mehrere Anwendungsroutinen enthalten kann).

SEGTaskInfo_GetTaskDescription Beschreibung der Anwendungsroutine (siehe Liste der Anwendungsroutinen, Anwendungsroutinen nach Name).

SEGTaskInfo_GetTaskProcsUsed Welche SAS-Prozeduren braucht die Routine (ebd.).

SEGTaskInfo_GetTaskValidation Welchen Validierungsstatus hat die Routine (ebd.).

SEGAddIn_Disconnect Verbindung zum Add-In trennen. Der Anwender erhält den Hinweis, dass die Registrierung des Add-Ins erst beim nächsten Aufruf wirksam wird.

3.3.3 Methodenaufrufe bei der Ausführung des Add-Ins

Wenn der Anwender eine Tabelle im Projektbaum auswählt und die Anwendungsroutine aufruft, die das Add-In bereitstellt, so ruft SEG folgende Methoden in der angegebenen Reihenfolge auf. Der SEG-Add-In-Programmierer ist dafür zuständig, dass diese Methoden die vorgesehenen Aufgaben, die auf der rechten Seite kurz beschrieben sind, erledigen. Die Methoden sind im Detail in [1] beschrieben.
Wenn ein bereits im Projektbaum befindlicher Aufruf der Anwendungsroutine reaktiviert wird (Doppelklick auf den Knoten im Baum), so ist der im Abschnitt 3.3.4 beschriebene Ablauf relevant.

SEGAddIn_Connect Initialisierung des Add-Ins: Dialogobjekt erzeugen, Variablen im Dialogobjekt initialisieren. Hier wird auch die Trace-Datei zum Debugging erstellt und geöffnet.

SEGDataSink_ActiveDataChanged (nur für Spezialfälle, von uns nicht verwendet)

SEGTask_Initialize Dialog initialisieren (Aufruf der Initialize-Methode des Dialogs), der Dialog muss alle Oberflächenelemente initialisieren, insbesondere auch die Variablen der gewählten Datei und ihre potenziellen Rollen anzeigen, siehe Abschnitt 3.4.1, und ggfs. die Ausgabedatei benennen, siehe 3.4.2.

SEGTask_GetPropertyList Muss eine Liste mit den Namen aller Properties zurückgeben, die benötigt werden, um den Zustand des Aufrufs im Projektbaum zu speichern, siehe unten SEGTask_GetProperty sowie SEGTask_SetProperty im Abschnitt 3.3.4.

SEGTask_Show Zeigt den Dialog an und wartet, bis der Anwender alle seine Eingaben gemacht hat. Meldet danach die Aktion des Anwenders (OK oder Abbrechen) an SEG zurück, damit SEG bei Bedarf fortfahren kann oder abbrechen.

SEGTask_GetProperty Diese Methode wird von SEG für jede Property, deren Name in SEGTask_GetPropertyList übergeben wurde, einmal aufgerufen, siehe oben. Dabei muss die Methode für jedes Property den Wert, den der Anwender gewählt hat, aus dem Dialog entnehmen und zurückgeben (z.B. welche Variable wurde welcher Rolle zugeordnet, welche Option wurde angekreuzt etc.). Diese Informationen speichert SEG hinter dem betreffenden Knoten im Projektbaum und gibt sie später über SEGTask_SetProperty (siehe Abschnitt 3.3.4) wieder zurück an das Add-In.

SEGTask_GetOutputDataCount Das Add-In muss SEG an dieser Stelle informieren, wie viele Ausgabetabellen von dem zu generierenden SAS-Code erstellt werden, soweit diese im Projektbaum sichtbar werden sollen. Für jede Ausgabetabelle wird später einmal SEGTask_GetOutputDataInfo aufgerufen. Hier geht es nur um SAS-Tabellen, nicht um ODS-Ausgabe.

SEGTask_GetSubmitCode Das Add-In muss hier den kompletten erzeugten SAS-Code als String zurückgeben, der anschließend von SEG ausgeführt wird. Der SAS-Code wird dabei anhand der Angaben generiert, die der Anwender im Dialog in SEGTask_Show gemacht hat.

SEGDataSink_GenerateOutputData (nur für Spezialfälle, von uns nicht verwendet)

SEGTask_GetOutputDataInfo — Für jede Ausgabetabelle, deren Anzahl mit SEGTask_GetSubmitCode mitgeteilt wurde, teilt das Add-In nun SEG mit, wo die Tabelle nach Ausführung des SAS-Codes zu finden ist und welche Bezeichnung im Projektbaum sichtbar sein soll.

SEGTask_Terminate — Hier können, wenn nötig, Aufräumarbeiten vorgenommen werden und Einstellungen gesichert werden.

SEGAddIn_Disconnect — Hier wird das Dialogobjekt freigegeben.

3.3.4 Methodenaufrufe beim Reaktivieren einer Anwendungsroutine

Wenn ein bereits im Projektbaum befindlicher Aufruf der Anwendungsroutine reaktiviert wird (Doppelklick auf den Knoten im Baum), so ruft SEG folgende Methoden in der angegebenen Reihenfolge auf. Der SEG-Add-In-Programmierer ist dafür zuständig, dass diese Methoden die vorgesehenen Aufgaben, die auf der rechten Seite kurz beschrieben sind, erledigen. Die Methoden sind im Detail in [1] beschrieben.

SEGAddIn_Connect — (siehe 3.3.3)

SEGDataSink_ActiveDataChanged — (siehe 3.3.3)

SEGTask_Initialize — (siehe 3.3.3)

SEGTask_GetPropertyList — (siehe 3.3.3)

SEGTask_SetProperty — Diese Methode ist das Spiegelbild von SEGTask_GetProperty (siehe 3.3.3) und wird von SEG für jede Property, deren Name in SEGTask_GetPropertyList übergeben wurde, einmal aufgerufen. Dabei erhält die Methode für jedes Property von SEG den Wert mitgeteilt, den der Anwender beim letzten Aufruf der Routine gewählt hatte und muss diese Einstellung im Dialog reproduzieren.

SEGTask_Show — (weiterer Ablauf siehe 3.3.3)

3.3.5 Eingabedaten verarbeiten

In den meisten Fällen möchte man die Variablen einer Eingabetabelle im Variablenselektor (siehe 3.4.1) darstellen, damit der Anwender sie dort den Rollen zuordnen kann.
Hierzu kann man die Add-In-Framework-Services SEGData, SEGAccessor und SEGColumnInfo (siehe Abschnitt 3.3.1) wie folgt verwenden (Voraus-

setzung: Application als globale Variable vom Typ SEGApplication definiert und SASVasel als VariableSelector auf dem Dialog vorhanden, siehe auch Beispielprojekt):

```
'Variablen deklarieren
Dim Data As SEGData
Dim Accessor As SEGAccessor
Dim ColumnInfo() As SEGColumnInfo

'Tabelle zuordnen, die der Anwender ausgewählt hat
Set Data = Application.GetActiveData()
'Zugriffobjekt geben lassen u. Spalteninfo ermitteln
Set Accessor = Data.GetAccessor()
Accessor.GetColumnInfoList ColumnInfo()

'Rollen im Variablenselektor einrichten,
'ROLE_xxx sind Stringkonstanten mit den Rollennamen
SASVaSel.AddRole ROLE_COLUMNSTORETAIN, 0, 0,
False,False, _
                 "", "", "", "", "", 1
SASVaSel.AddRole ROLE_SORTBY, 0, 0, False, False, _
                 "", "", "", "", "", 1

'Alle Spalteninformationen in den Variablenselektor tun
lIndex = LBound(ColumnInfo)
lCount = UBound(ColumnInfo)
Do While lIndex <= lCount
   SASVaSel.AddVariable_V2 _
   ColumnInfo(lIndex).VariableName, _
   ColumnInfo(lIndex).DisplayName, _
   ColumnInfo(lIndex).DisplayName, _
   ColumnInfo(lIndex).Offset, _
   ColumnInfo(lIndex).VariableGroup, _
   ColumnInfo(lIndex).Length, _
   ColumnInfo(lIndex).Format, _
   ColumnInfo(lIndex).Informat, _
   ColumnInfo(lIndex).SortedBy
   lIndex = lIndex + 1
Loop
```

3.3.6 SAS-Code im Hintergrund submittieren

Normalerweise muss man vom Add-In aus SAS-Code nicht direkt submittieren, weil das von SEG übernommen wird, nachdem SEGTask_Code den generierten Code zurückgegeben hat. Für spezielle Anwendungen kann es jedoch nötig sein, während der Anwender im Dialog arbeitet im Hintergrund Informationen aus SAS abzufragen. Hierfür gibt es die Möglichkeit, SAS-Code zu submittieren.
Dies ist nicht ganz einfach, weil es derzeit nur vorgesehen ist, SAS-Code asynchron zu submittieren, d.h. der Anwender kann weiterarbeiten, während der SAS-Code ausgeführt wird.
Man kann mit Hilfe der Add-In-Framework-Services SEGApplication und SEGAccessor (siehe Abschnitt 3.3.1) wie folgt vorgehen, siehe auch unser Beispielprojekt, in dem PROC CONTENTS aufgerufen wird, um die aktuelle Sortierung der Eingabetabelle zu ermitteln, wenn der Anwender auf einen Button klickt.

- Im Click-Event des Buttons erzeugt man einen String, der den zu submittierenden SAS-Code enthält und übergibt ihn an SAS mit der Methode SEGApplication.SubmitCode. Dabei erhält man zur Identifikation des Submits ein „Submit-Cookie“ zurück, das man sich merken muss.
- Der Anwender kann jetzt im Dialog weiterarbeiten, die Submit-Routine läuft im Hintergrund.
- Irgendwann ist die Submit-Routine fertig und SEG ruft die Methode SEGSubmitSink_SubmitComplete auf und übergibt das Submit-Cookie, so dass das Add-In weiß, welches Submit jetzt beendet wurde.
- Jetzt kann man die Ergebnisse des Submits verarbeiten. In der Regel bedeutet dies, dass man auf eine SAS-Tabelle zugreift, die von dem Submit erzeugt wurde. Dafür verwendet man wieder den SEGAccessor, hier ein Beispiel. Wir hatten Probleme mit überschriebenen Speicherbereichen bei GetFormattedValue, daher das temporäre Umkopieren der Variableninhalte. SEGApplication ist wieder als globale Variable deklariert.

```
'Diese Methode wird aufgerufen von
'   SEGSubmitSink_SubmitComplete
Public Sub handleSubmitResults()

'Deklarationen für Zugriffsobjekte
```

```
Dim Data As SEGData
Dim Accessor As SEGAccessor
Dim Server As String

'Zugriff auf die Tabelle WORK.TEMP,
'   die durch Submit erzeugt wurde
Server = Application.GetActiveData().Server
Set Data = Application.CreateDataObject(Server, _
           "WORK", "TEMP", "")
Set Accessor = Data.GetAccessor()

'Deklarationen für temporäre Variablen
Dim temp As String * 1000
Dim optionName As String
Dim optionValue As String
Dim Line As Integer
Dim Missing As Boolean
Dim msg As String

'Wenn die Tabelle leer ist: Meldung ausgeben
If Accessor.Rows = 0 Then
   MsgBox "Gewählte Tabelle ist nicht von SAS sortiert”
_
          vbOKOnly, "aktuelle Sortierung"
   Exit Sub
End If

'Andernfalls Informationen aus der Tabelle anzeigen
msg = "Sortierung:" & vbCrLf
'Über alle Zeilen iterieren, Variablenwerte holen
For Line = 0 To Accessor.Rows - 1
   temp = Accessor.GetFormattedValue("LABEL1", _
          Line, Missing)
   optionName = Trim(temp)
   temp = Accessor.GetFormattedValue("CVALUE1", _
          Line, Missing)
   optionValue = Trim(temp)
   msg = msg & optionName & ": " & optionValue & vbCrLf
Next Line

MsgBox msg
End Sub
```

3.4 Der Control-Baukasten

Um leichter Dialoge bauen zu können, mit deren Hilfe die Anwender SEG-Anwendungsroutinen parametrieren können, liefert SAS einen Baukasten mit fertigen Controls, siehe Referenz [1]. Die wichtigsten sollen hier beschrieben werden.
Neben den unten beschriebenen Controls gibt es noch folgende Controls, die aber in den hier beschriebenen Szenarien keine Rolle spielen:

- Color Selector für die Auswahl von Farben für SAS/GRAPH
- Option Selector
- Format / Informat Selector.

3.4.1 Variable Selector (Variablen und Rollen)

Der Variablenselektor ermöglicht es, Variablen zu Rollen zuzuweisen, siehe beispielsweise Abbildung 5. Die beiden Listen „zuzuweisende Spalten“ und „Spaltenrollen“ gehören zum Variablenselektor. Ein Beispiel, wie der Selektor gefüllt werden kann, findet sich im Abschnitt 3.3.5.

3.4.2 Data Set Selector (Ausgabedatei)

Der Data Set Selector vereinfacht es für den Anwender, eine Ausgabetabelle auszuwählen, weil damit der Standard-Dialog aufgerufen werden kann, den SEG für diesen Zweck bereithält. Ein Beispiel ist in Abbildung 6 rechts unten zu finden.
Es gibt einfache Methoden, wie man die Ausgabedatei vorbelegen kann, indem man einen eindeutigen Namen erzeugt. Dies wird normalerweise in der Initialize-Methode des Dialogs getan, die von SEGTask_Initialize aufgerufen wird (Application ist eine globale Variable vom Typ SEGApplication und SASDSSel ist ein Control auf dem Dialog):

```
'Informationen über die Eingabedatei holen,
'   um Server zu ermitteln
Dim Data As SEGData
Set Data = Application.GetActiveData()

'Wenn der Anwender nichts anderes angibt:
'Ausgabe nach WORK
SASDSSel.LibraryName = "WORK"
```

```
'eindeutigen Namen erzeugen
SASDSSel.ServerName = Data.Server
SASDSSel.DatasetName =
   Application.CreateMemberName(Data.Server, _
                                SASDSSel.LibraryName, "SORT")
```

3.4.3 Enhanced Editor (Quellcode-Editor)

Der Enhanced Editor ist ein kompletter Programmeditor mit Syntaxgesteuerter Farbgebung, wie er auch im Enterprise Guide selbst benutzt wird.
Abbildung 11 zeigt ein Beispiel aus [8] für den Einsatz des Enhanced Editors, der hier dazu verwendet wird, Programme, die auf dem Großrechner gespeichert sind, anzuzeigen und auszuführen.

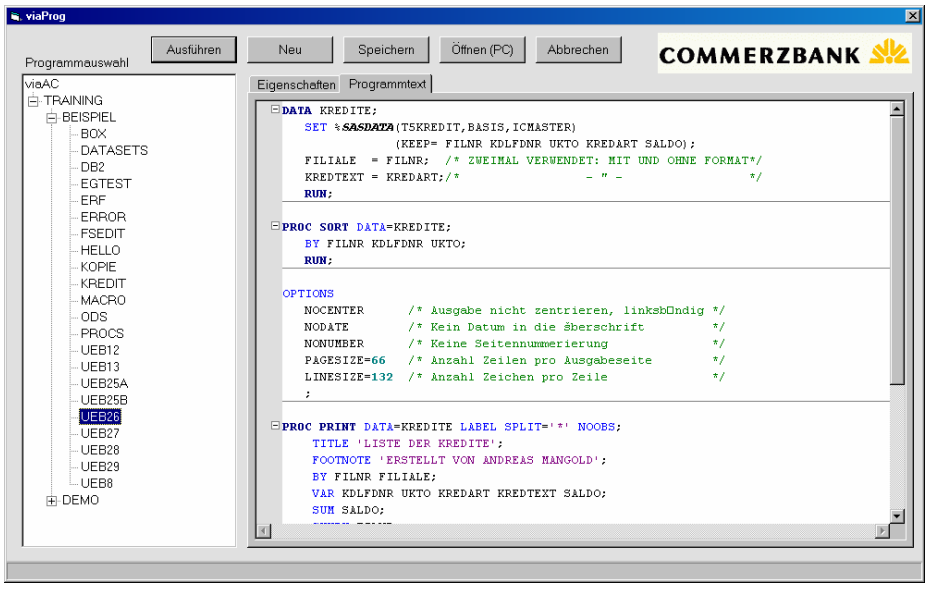

Abbildung 11: Einsatz des Enhanced Editors für Programme auf dem Großrechner

3.5 SEG-Add-Ins registrieren

SEG-Add-Ins sind DLLs, die zunächst mit REGSVR32.EXE bei Windows registriert werden müssen und anschließend bei SEG registriert werden müssen. Der gesamte Registrierungsprozess ist unter anderem in [2], [4] und [5] beschrieben und soll daher hier nicht weiter vertieft werden.
Die Version 3.0 von SEG verspricht, dass die Einbindung von Add-Ins einfacher wird.

4 Referenzen

[1] SAS Enterprise Guide – Adding Custom Tasks, SAS Institute, siehe http://www.sas.com/technologies/bi/query_reporting/guide/custom-tasks.html
Auf dieser Seite finden Sie Links zu den wichtigsten bei SAS öffentlich zugänglichen Ressourcen zu den Themen Entwickeln und Einbinden von SEG-Add-Ins. Auch sind hier einige Beispiele mit Quellcode zu finden, insbesondere auch „Sort Data“.

[2] SAS/Enterprise Guide®: Erweiterungen der Funktionalität über Add-Ins, Gregor Herrmann, SAS Training, Vortrag auf der SAS DISK 2003 in Bonn.
Die im Rahmen dieses Vortrags vorgestellten Add-Ins sind bei SAS auf Anfrage erhältlich.

[3] SAS Enterprise Guide Software – Advanced Workshop, Gregor Herrmann, SAS Training, Schulungsunterlagen und Begleit-CD.
Die im Rahmen dieser Schulung vorgestellten Add-Ins sind bei SAS auf Anfrage erhältlich.

[4] Developing Custom Analytic Tasks for SAS® Enterprise Guide®, Joe Carter, Stephen McDaniel, Mike Porter, SAS® Institute Inc., Cary, NC, siehe
http://www2.sas.com/proceedings/sugi28/051-28.pdf.
Kompakte Beschreibung des Aufbaus, der Entwicklung und der Einbindung eines SEG-Add-Ins für PROC HPF (High Performance Forecasting).

[5] SN-009580: How to include Add-In tasks into Enterprise Guide, SAS Institute, siehe
http://support.sas.com/techsup/unotes/SN/009/009580.html.
Kurzbeschreibung für das Einbinden von SEG-Add-Ins.

[6] SN-009447: Add-In tasks may not work when using Enterprise Guide, SAS Institute, siehe
http://support.sas.com/techsup/unotes/SN/009/009447.html.
Problembehebung für die Einbindung von SEG-Add-Ins.

[7] SAS Enterprise Guide – OLE Automation Server, SAS Institute, siehe http://www.sas.com/technologies/bi/query_reporting/guide/ole.html.
Auf dieser Seite finden Sie Links zu den wichtigsten bei SAS öffent-

lich zugänglichen Ressourcen zum Thema Steuern von SEG mit OLE-Automation.

[8] SAS Enterprise Guide vereinfacht die Programmentwicklung in der Commerzbank AG, Marcus Pilz, Andreas Mangold, siehe http://www.hms-heidelberg.de/extern/aktuell/DISK2003_SEG_web.htm.
Thema ist die Einbindung von SAS Enterprise Guide in eine komplexe Decision-Support-Umgebung auf dem Großrechner. Dabei werden SEG-Add-Ins verwendet.

[9] Global Overview and Goal of Add-In Buttons, National Office for Rural Measurement and Evaluation Systems, University of Arkansas, USA, siehe http://normes.uark.edu/eguide/custom.html.
Beispiel für ein Anwenderunternehmen, das SEG-Add-Ins einsetzt, um organisationsspezifische Aufgaben zu bewältigen. Interessant auch, dass die Add-Ins über eine eigene Installationsroutine verteilt werden. (Wir haben die auf dieser Seite angebotenen Add-Ins nicht getestet!)

[10] Enterprise Guide – Now and the Future – A Case Study, Philip R Holland, Holland Numerics Ltd., siehe http://www.hollandnumerics.co.uk/pdf/EG-Now_and_Future.pdf.
Das Papier gibt einen Überblick über das Zusammenspiel von Client- und Serverkomponenten im Rahmen von SEG, berichtet über Erfahrungen von Anwendern mit SEG und geht auch kurz auf Add-Ins ein. Auf derselben Website wird kommerziell auch ein SEG-Add-In für PROC REPORT angeboten, siehe http://www.hollandnumerics.co.uk/prodserv/hnl2.htm.

Anhang: Liste uns bekannter SEG-Add-Ins

Nr.	Name des Add-Ins	Registrierungsname	Beschreibung	Quelle (s. Abschn. 4)
1.	Optimize Data	optimize. data	Speicherlängen optimieren, vor allem nach Excel-Import	[3]
2.	Sort Data	SasSort. Sort	Tabellen sortieren mit PROC SORT aufrufen	[1]
3.	Export to Excel	XlsAddIn. XlsModule	Schreiben einer SAS-Datei in eine bereits existierende Excel-Arbeitsmappe unter Angabe des Bereichs	[3]
4.	Report	Report. ReportTask	Berichte erstellen mit PROC REPORT	[3]
5.	MetaData Search		Suche nach Texten in allen Tabellen- und Spaltennamen und beschreibungen	[2]
6.	Display Data Source Attributes	Metadata. Contents	PROC CONTENTS ausführen und Bericht anzeigen	[3]
7.	High-Performance Forecasting	SASHPF. Custom	High Performance Forecasting mit PROC HPF	[4]
8.			PROC SQL Update	[3] (nur im Schulungs-handbuch erwähnt
9.			Berichte erstellen mit PROC REPORT	[10]
10.	Demo	DemoTask. Demo	Vorlage für das Erstellen eigener Add-Ins, abgeleitet von Nr 2 (Sort Data)	dieses Paper
11.	viaModel-CBOrg		Organisationseinheiten einer Bank ermitteln (organisations-spezifisch)	[8]

Nr.	Name des Add-Ins	Registrierungsname	Beschreibung	Quelle (s. Abschn. 4)
12.	viaNavigator		Daten in einer Data-Warehouse-Umgebung suchen und einbinden	[8]
13.	viaProg		SAS-Programme vom Großrechner ausführen und verwalten	[8]
14.			School Improvement Reports (organisationsspezifisch)	[9]