Die Reise vom Mainframe nach Linux/Unix

Klaus Kepert
HMS Analytical Software GmbH
Grüne Meile 29
69115 Heidelberg
Klaus.kepert@analytical-software.de

Zusammenfassung

Auf IBM Mainframe Systemen sind in der Regel historisch gewachsene Strukturen über Jahrzehnte entstanden. Im Laufe der Zeit wurden Linux/Unix Systeme schneller, zuverlässiger, dynamischer und vor allem günstiger. Deshalb kann eine Migration von z/OS nach Linux/Unix interessant, aber auch abschreckend sein. Wenn man die Hürden und Herausforderungen kennt, wird das Wagnis zum Projekt.

Schlüsselwörter: SAS, Migration, z/OS, Linux, Automatisierung

1 Daten

Es ist zu erwarten, dass auf der Zielplattform Linux/Unix noch keine Daten verfügbar sind. Deshalb müssen diese vom Mainframe nach Linux/Unix übertragen werden. Das geht nicht einfach durch ein Kopieren, weil die Zeichensätze unterschiedlich sind.

1.1 Datentransport

SAS stellt ein paar Prozeduren bereit, die eine Übertragung über Betriebssystemgrenzen hinweg möglich macht.
Mit PROC CPORT werden die Daten in ein Doppelbyte Format verwandelt, das auf dem anderen Betriebssystem mit PROC CIMPORT wieder in das hier passende Format einer SAS Bibliothek zurück verwandelt werden.

Hinweis

Das DSN Format auf Mainframe muss FB80 sein.

Andere DSN Formate führen zu einer Endlosschleife, bis die Festplattenkapazität erreicht ist.

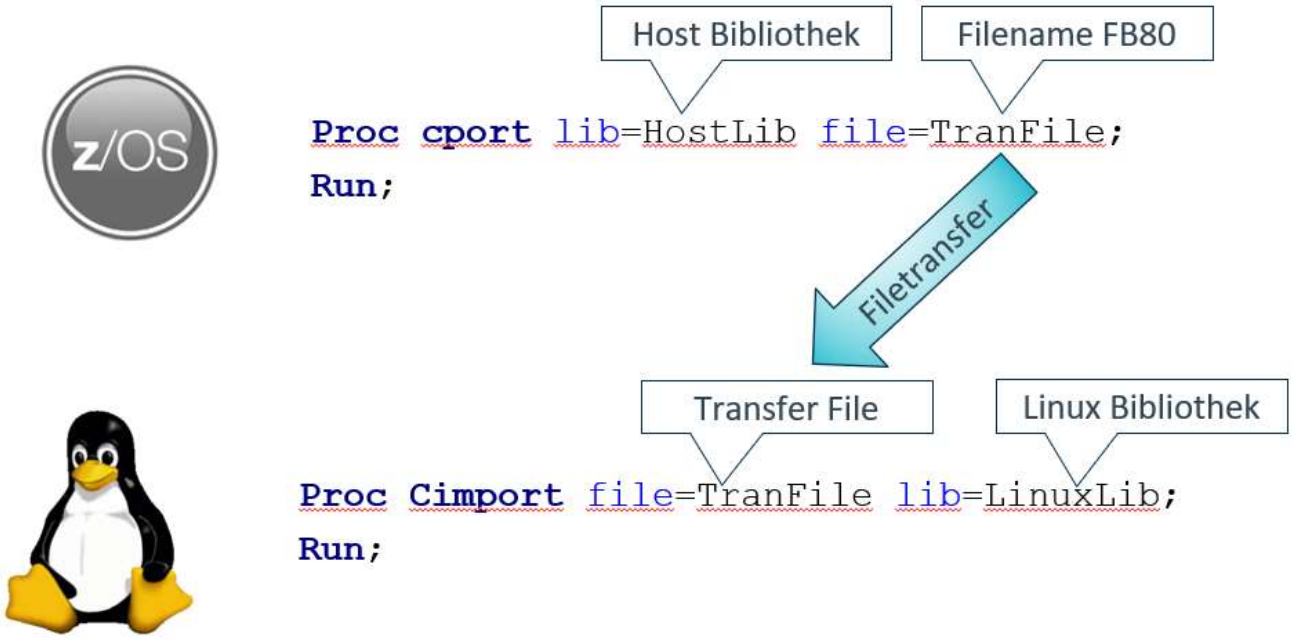

Abbildung 1: Datentransport

1.2 Micro Abweichungen

SAS kennt nur Gleitkommazahlen als Datentyp für numerische Variablen. Es gibt keine spezifischen Typen für Zahlen mit fester Anzahl Nachkommastellen. Daher werden nicht-ganze Dezimalzahlen mit einer winzigen Unschärfe abgebildet. Solange man die Rechnerarchitektur nicht wechselt, spielen diese Unschärfen keine Rolle.
Die CPU interne Darstellung von Gleitkommazahlen unterscheidet sich zwischen Host und Linux. D.h., beide Systeme haben numerische Unschärfen, es sind aber nicht die gleichen. Dadurch kann es passieren, dass sich bei scheinbar identischem Input unterschiedliche Ergebnisse mit unsichtbaren Abweichungen ergeben.

Kreditkonten
Abweichungen in der Inanspruchnahme

Herkunft	ID	Inanspruchnahme	Nachkommastellen Inanspruchnahme
Linux	2481302109	11.589,44	.439999999995052000000000
Host	2481302109	11.589,44	.439999999998690000000000
Abweichung	2481302109	0,00	.000000000003637978807092

12. Nachkommastelle

Abbildung 2: Micro Abweichungen

Eine Abweichung in der 12. Nachkommastelle erscheint zunächst harmlos. Wenn über einige Millionen Datensätze aufsummiert wird, können diese Effekte aber doch Auswirkungen zeigen, weil die Abweichungen nicht +/- symmetrisch sind.
Tückisch an Micro Abweichungen ist, dass sie in den Daten augenscheinlich unsichtbar sind.
Ein weiterer Effekt kann auftreten, wenn eine IF Abfrage auf ein numerisches Feld im SAS Programm enthalten ist.
Die Anweisung **if Inanspruchnahme > 10000;** kann dazu führen, dass ein einzelner Datensatz unter tausenden auf die Bedingung unterschiedlich reagiert.

1.3 Sortierung

SAS sortiert Daten auf Mainframe und Linux/Unix in unterschiedlicher Reihenfolge, wenn es sich um Text Schlüssel handelt.

Mainframe EBCDIC	sortiert erst Buchstaben dann Ziffern
Linux/Unix ANSI	sortiert erst Ziffern dann Buchstaben.

Abbildung 3: Sortierreihenfolge

Dadurch erscheinen die Daten im Vergleich unterschiedlich.

Die Lösung ist, die Ergebnisse vor dem Vergleich in gleicher Reihenfolge zu sortieren.

```
proc sort data=mydata SORTSEQ=EBCDIC;
```

1.4 COBOL Daten einlesen

SAS kann auch auf Linux/Unix mit COBOL Daten umgehen, wenn man zwei Dinge beachtet.

1. Der Filename Anweisung muss ein Record Format und eine Record Länge gegeben werden.
2. Die Informate müssen auf die S370 Formate umgestellt werden. Für jedes Informat auf Mainframe existiert ein passendes S370 Format auf Linux/Unix.

```
filename hist "/home/HMS/&HLQ..COBOL.ROHDATEN" recfm=f  lrecl=22 ;

Data x;
  infile hist;
  input  kto      s370fpd6.
         ewb      s370fpd8.2
         betrag   s370fpd8.2
         ;
Run;

Filename hist;
```

Abbildung 4: Codebeispiel für COBOL Daten

2 SAS Programme

Die SAS Programme können über ein Transfer Programm wie z.B. WinSCP kopiert werden. Der Zeichensatz wird dabei korrekt umgewandelt, wenn das richtige Coding eingestellt ist. In der Regel erkennen das die Transfer Programme selbst. Achten Sie auf die Umlaute.

2.1 Genau hinschauen

Prüfen Sie zunächst, welche Programme nicht mehr verwendet werden. Diese Mühe kann sich lohnen, weil damit der Migrationsaufwand reduziert wird.

2.2 SAS Optionen

Prüfen Sie einmalig alle Optionen. Diese können unterschiedlich sein.

Option NLSCOMPATMODE
Sie interpretiert den SAS Code auf eine sehr alte Weise. So sind z.B. die geschweiften Klammern auf Mainframe noch nicht existent und werden im SAS Code mit Ä und Ü maskiert.
Dieser Code funktioniert auf Linux/Unix nicht mehr.

S=72
Sie begrenzt die Länge einer Code Zeile auf 72 Zeichen. Diese Option macht auf Linux/Unix keinen Sinn mehr und muss im Code gelöscht werden.
Sie kann bewirken, dass SAS eigene Macros in der Zeile abgeschnitten werden und nicht mehr funktionieren. Sehr tückisch ist es, wenn ein Kommentar Ende Zeichen abgeschnitten und damit ignoriert wird.

Option DKRICOND=ERROR
Wenn eine Variable angesprochen wird, die in den Daten nicht existiert, z.B. aber in einer DROP Anweisung vorkommt, entscheidet diese Option, ob mit einem ERROR oder einem WARNING darauf reagiert wird. Die Default-Werte unterscheiden sich, so dass auf Linux/Unix Programme abbrechen, die auf Mainframe nur ein WARNING hatten.

-encoding WLATIN1
Der Zeichensatz muss auf Linux/Unix angepasst werden. Der Standard Zeichensatz WLATIN9 ist nicht ausreichend, weil einzelne Zeichen fehlen, so z.B. das € - Zeichen.

2.3 Alte Prozeduren

Auf Mainframe können historisch gewachsen Prozeduren wie z.B. PROC DB2EXT oder PROC DBLOAD für den Zugriff auf DB2 verwendet werden, die veraltet sind.
Diese Prozeduren laufen auf Linux/Unix nicht mehr und müssen durch PROC SQL ersetzt werden.

2.4 Feldlänge

Eine numerische Variable kann auf Mainframe die Länge 2 Byte haben

```
Length Sort 2;
Attrib Sort length=2;
```

Auf Linux/Unix ist die minimale Länge 3 Byte.

```
Length Sort 3;
Attrib Sort length=3;
```

2.5 Libname / Filename

Libname und Filename Statements greifen auf das Betriebssystem zu und sind deshalb syntaktisch unterschiedlich.

z/OS: `Libname KunDat "&HLQ..KUNDEN.SAS"  DISP=SHR;`

Linux/Unix `Libname KunDat "&HLQ./KUNDEN/SAS" Access=Readonly;`

Hier könnte auch ein Macro helfen, das die Umwandlung automatisiert in allen Programmen gleich macht.

```
%ConvertDSN("&HLQ..KUNDEN.SAS",DISP=SHR);
```

2.6 Hex-Werte

Falls im SAS Code eine Umwandlung von Hex-Codes notwendig ist, müssen diese geändert werden. Im Internet können Tabellen von Hex Codes für alle Betriebssysteme abgefragt werden.

Mainframe
```
kd=translate(Hex_Key,' ', '0D'x,' ', '25'x,'Ö', '66'x,'ö', '62'x);
```

Linux/Unix
```
kd=translate(Hex_Key,' ', '0D'x,' ', '0A'x,'Ö', 'C3'x,'ö', 'C2'x);
```

3 JCL (Job Control Language)

In der JCL wird auf Mainframe die Laufzeit Umgebung definiert.

Es wird festgelegt, ob überhaupt ein SAS Programm zur Ausführung kommt (EXEC SAS), welches Programm (SYSIN) und mit welchen Datenzugriffen (DD Statement).

Libname und Filename Statements können hier definiert sein, so dass im SAS Code kein Libname/Filename Statement gebraucht wird.

Da die Syntax relativ klar ist, kann die JCL von einem Scanner Programm eingelesen werden, das alle Programmaufrufe mit allen Datenzugriffen identifiziert.

Hinweis

Libname und Filename lassen sich in der JCL nicht unterscheiden.

Für die DD-Statements müssen auf Linux/Unix Libname/Filename Statements ins SAS-Programm eingebaut werden. Da diese sich im DD-Statement nicht unterscheiden, muss versucht werden, am Dateinamen (DSN) zu erkennen, worum es sich handelt. Wenn das

nicht geht, muss im SAS Code geprüft werden, ob ein DD Statement als Libname oder Filename angesprochen wird.

In einer JCL kann auch Logik enthalten sein, z.B. die Berechnung des Buchungstages. Diese muss in SAS oder in einem Shell-Script nachgebaut werden.

IBM Tools wie IEFBR14 und IDCAMS können in der Regel ignoriert werden. Auf Mainframe ist es üblich, Dateien (DSN) zu löschen und wieder anzulegen.
Diese Vorgehensweise ist auf Linux/Unix nicht mehr sinnvoll.

4 Automatisierung

Vermutlich sind hunderte oder sogar tausende Programme von einer Migration betroffen.
Man kann die Weiterentwicklung der Programme auf Mainframe nicht stoppen, solange die Migration läuft. Deshalb muss man klären, wie die Änderungen am Code auch in der Migrationsphase berücksichtigt werden.

4.1 Ohne Automatisierung

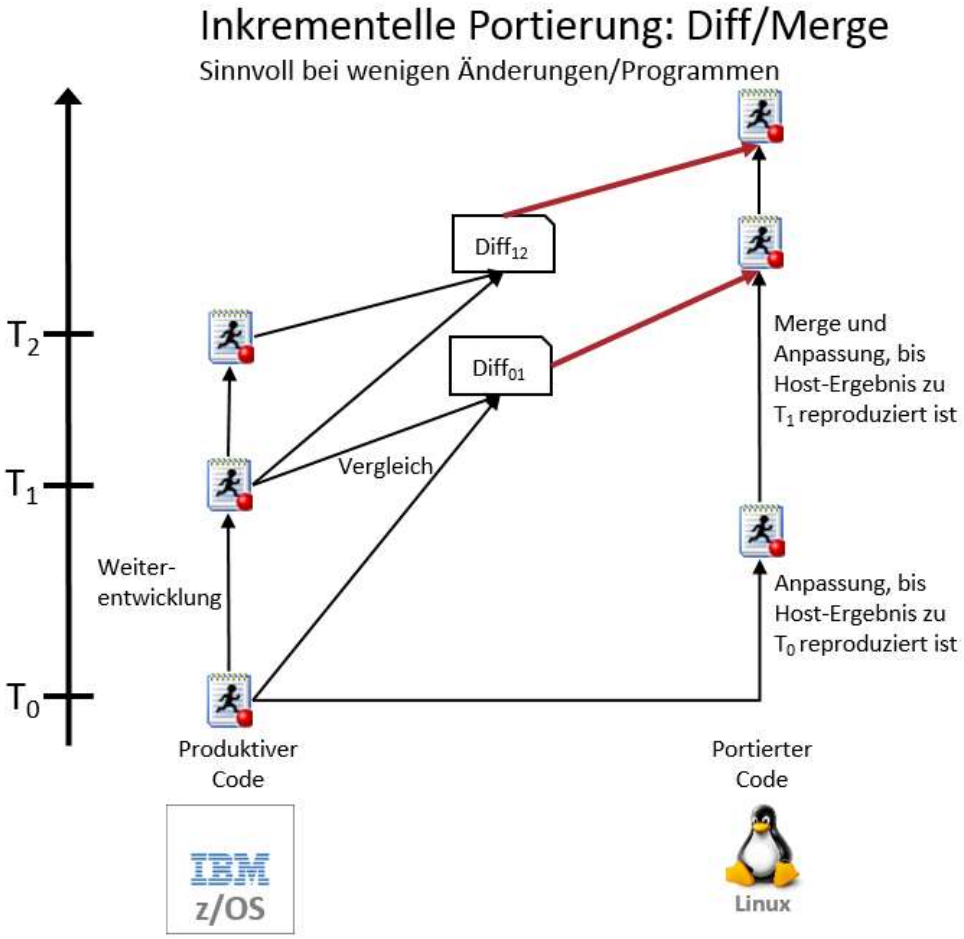

Abbildung 5: Abgleich der Programme von Hand

Jedes Programm muss vom Mainframe kopiert und mit dem auf Linux/Unix vorhandenen Programm mit einem Differenz Scanner (Differ) verglichen werden. Jede Änderung auf Mainframe muss von einem Programmierer übertragen werden.

Vorteile dieser Methode sind:

- Semantische, komplexe Änderungen können verstanden und geprüft werden
- Die Synchronisation wird von Entwicklern gemacht

Nachteile dieser Methode sind:

- Hoher Zeit- und Kostenaufwand
- Hohe Anzahl Programme wird unbeherrschbar
- Wiederholungen sind mühsam und zeitintensiv

Der Versuch, jede Änderung an produktiven Programmen auch auf Linux nachzuziehen, ist nicht realistisch. Dabei werden Änderungen vergessen, übersehen, auf später verschoben oder anders umgesetzt. Auch würde sich der Entwicklungs- und Testaufwand erheblich erhöhen.

4.2 Mit Automatisierung

Die Programme werden vom Mainframe kopiert und ein Parser Programm baut die notwendigen Anpassungen für die Linux/Unix Umgebung ein. Programmänderungen auf dem Mainframe werden so mitgenommen. Der Vorgang läuft automatisch ab und kann beliebig wiederholt werden. Eine Liste der erwarteten Änderungen und was zu tun ist, führt den Parser.

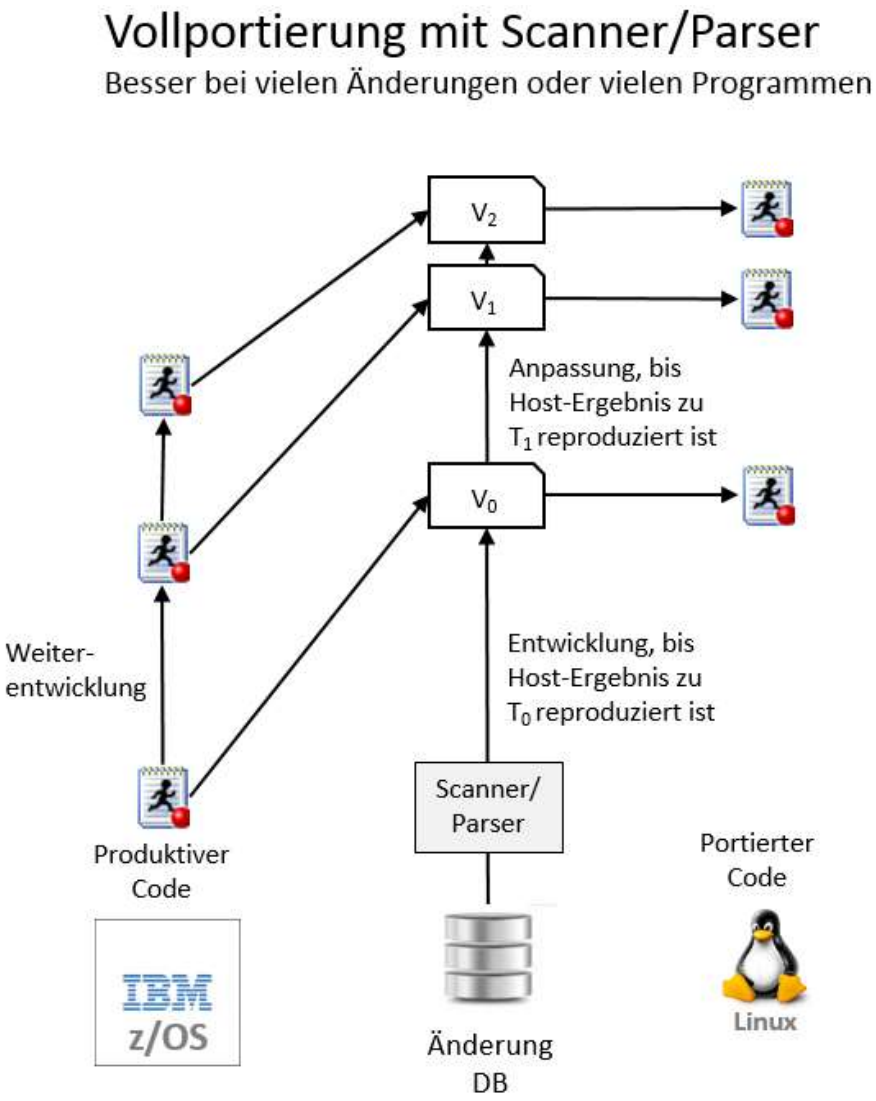

Abbildung 6: Automatisierter Abgleich

Beispiele für eine Änderung des Parsers:

- LIBNAME Anweisungen werden gesucht und durch Linux/Unix Libnames ersetzt.
- Die Anweisung OPTIONS wird gesucht und die Option S=72 wird entfernt.

Eine Weiterentwicklung auf Mainframe und zeitgleich auch auf Linux/Unix sollte unbedingt vermieden werden, weil die automatisierte Synchronisation zu kompliziert werden würde.

Vorteile dieser Methode sind:

- Sehr wenig Aufwand im Vergleich zur manuellen Methode
- Die Migration ist beliebig wiederholbar
- Die Anzahl der Programme beeinflusst nicht wesentlich den Aufwand
- Code Änderungen sind dokumentiert
- Der Parser kann sich weiter entwickeln

Nachteile dieser Methode sind:

- Die Entwicklung des Parsers ist eine Investition.
- Semantische, komplexe Änderungen können nicht automatisiert laufen
- Nur erwartete Änderungen werden berücksichtigt

Der produktive Code im z/OS kann sich durchaus fachlich umfangreich ändern.

Wird nach dieser fachlichen Anpassung ein Änderungskriterium nicht mehr gefunden, wird es vom Parser ignoriert.

Ist nach einer fachlichen Anpassung ein neuer Fall für den Parser enthalten, wird dies im Migrationsprozess entdeckt und ebenfalls automatisiert.

4.3 Automatisiertes Testen

SAS Unit ist ein Framework aus SAS Programmen mit der Aufgabe, SAS Programme im Test zu starten und vordefinierte Testfälle zu prüfen. Die Steuerung erfolgt über Steuertabellen, die die Informationen enthalten, welche Programme mit welchen Testfällen laufen sollen.

SAS Unit erzeugt HTML-Seiten als Ergebnis, die alle getesteten Fälle als erfolgreich oder fehlgeschlagen kennzeichnet.

Voraussetzung für einen automatisierten Testlauf ist, dass sowohl die Quelldaten als auch die daraus resultierenden Ergebnisse aus z/OS übertragen wurden. Ein Testfall gilt nur dann als erfolgreich, wenn das Ergebnis identisch ist.

Test Scenarios | SASUnit Examples - SASUnit

Main Page | Test Scenarios | Test Cases | Units under Test

No.	Test Scenario	Program	Last Run	Duration	Result
001	Test examples for assertExternal.sas	saspgm/assertexternal_example_test.sas	15FEB2016:09:57:28	0.9 s	☑
002	Test examples for assertImage.sas	saspgm/assertimage_example_test.sas	15FEB2016:09:57:29	2.4 s	☑
003	Test examples for assertText.sas	saspgm/asserttext_example_test.sas	15FEB2016:09:57:32	0.9 s	☑
004	Tests for boxplot.sas	saspgm/boxplot_test.sas	15FEB2016:09:57:33	3.8 s	☐
005	Tests for comparison.sas	saspgm/comparison_test.sas	15FEB2016:09:57:36	2.5 s	☑
006	Example for cross reference reports	saspgm/crossreference_test.sas	15FEB2016:09:57:39	0.9 s	☑
007	Tests for buidling a database	saspgm/database_test.sas	15FEB2016:09:57:40	1.3 s	☑
008	Tests for generate.sas	saspgm/generate_test.sas	15FEB2016:09:57:41	1.9 s	☑
009	Tests for getvars.sas	saspgm/getvars_test.sas	15FEB2016:09:57:43	0.8 s	☑
010	Tests for nobs.sas - has to fail!	saspgm/nobs_test.sas	15FEB2016:09:57:44	0.9 s	☒
011	Tests showing program documentation features	saspgm/programdocumentation_test.sas	15FEB2016:09:57:45	0.5 s	☑
012	Tests for regression.sas	saspgm/regression_test.sas	15FEB2016:09:57:45	1.4 s	☐

Abbildung 7: Beispiel für Unit Test Report

Beispiele für Unit Tests.

- Das Programm endet mit RC=0
- Das Programm enthält keine oder erwartete Warnings
- Die Ergebnistabellen sind identisch mit dem Mainframe

Es können leicht einige tausend Testfälle entstehen. Deshalb wurden die Routine Testfälle generiert.

4.4 Übersicht der Automatisierung

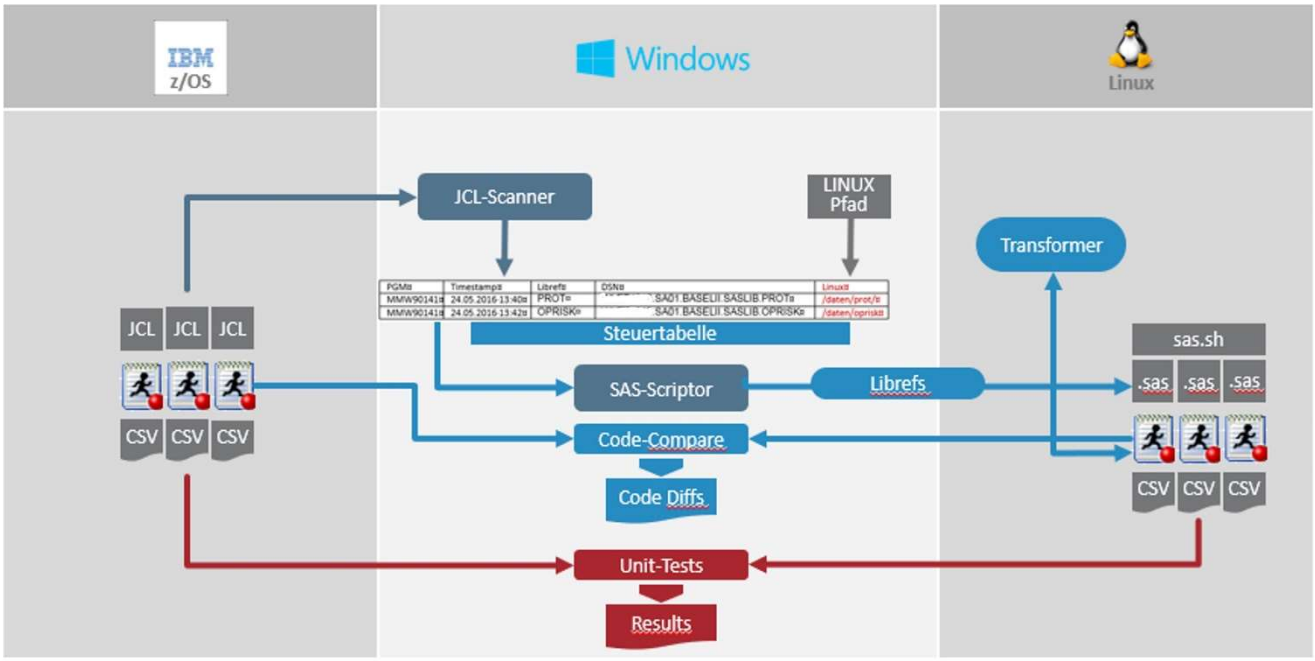

Abbildung 8: Übersicht der Automatisierung

In einer zentralen Steuertabelle werden alle Zusammenhänge abgebildet (Metadaten).

JCL-Scanner

Mit diesem Tool können alle JCLs gescannt, die SAS Aufrufe und deren Libnames/Filenames identifiziert und in der Steuertabelle abgelegt werden.

SAS Scriptor

Der SAS Scriptor generiert auf Basis der Steuertabelle die Libname/Filename Statements und trägt sie in die SAS Programme ein.

Transformer

Mit diesem Tool werden die Linux Anpassungen in die SAS Programme eingebaut. Welche das sind, ist in einer Steuertabelle abgelegt.

Code Compare

Die Mainframe und Linux/Unix Varianten der Programme werden miteinander verglichen und Unterschiede markiert. Dies dient hauptsächlich zur Dokumentation.

Fazit

Die Migration einer SAS Batch Umgebung von z/OS nach Linux/Unix ist zweifellos eine Herausforderung, insbesondere wenn sehr viele Programme betroffen sind. Eine Rechnung im Sinne von

Gesamtaufwand = X Programme * Aufwand in PT pro Programm

kann schnell einen erschreckend hohen Aufwand hervorbringen.

Die Automatisierung bedeutet eine Investition zu Beginn des Projektes, führt dann aber dazu, dass die Arbeitsabläufe einfacher, schneller und billiger erfolgen können.

Wenn man die Fallstricke kennt und weiß, wie man damit umgeht, ist eine Migration machbar und bezahlbar.