Mit Java ans alte Eisen

Funktionstest von Host-Anwendungen

Kay F. Hildebrand, Tobias Voß

Immer noch bilden Anwendungen auf dem Mainframe die Basis oder einen wichtigen Bestandteil von vielen Geschäftsprozessen ab, gerade bei Banken und Versicherungen. Für den automatisierten Test von Batch-Programmen stellen wir einen Ansatz auf Basis von Open-Source-Komponenten vor, mit dem Mainframe-Anwendungen getestet werden können. In einem konkreten Projekt war dieses Vorgehen essenziell, da die Funktionstester über keinerlei Mainframe-Kenntnisse verfügten.

Der Dinosaurier Mainframe stirbt nicht so schnell aus wie prognostiziert [Als93]. Gerade bei Banken und Versicherungen sind die zentralen Geschäftsprozesse weiter in COBOL oder PL/I programmiert. Da der Mainframe viele Schnittstellen nach außen bietet, wird die tägliche Arbeit zunehmend auf PC-Anwendungen, wie die Eclipse-basierte *IDE Rational Developer for System z* [RDz], verlagert. In einem unserer Projekte standen wir vor der Herausforderung, eine Funktionstest-Infrastruktur für COBOL-Batch-Programme zu entwickeln, um Funktionstester ohne Mainframe-Kenntnisse zu unterstützen.

Testen von Batch-Jobs

Für den Funktionstest von Batch-Jobs gibt es abweichende Anforderungen im Vergleich zu Dialoganwendungen. Batch-Programme verarbeiten immer gleich strukturierte Eingaben, basierend auf einer fest definierten Schnittstelle von Eingabedateien oder Datenbanktabellen. Es sind keine Oberflächentests notwendig, aber gleichzeitig existiert auch keine Benutzerschnittstelle, die ein Tester ohne Host-Erfahrung bedienen könnte. Für eine Automatisierung der Testdurchführung sind keine besonderen Tools notwendig, da eine automatisierte Ausführung der normale Betriebsmodus für Batch-Programme ist.

Um Batch-Jobs einem Funktionstest zu unterziehen, sind damit drei Tätigkeiten notwendig. Erstens werden Testdaten gemäß der Eingabeschnittstelle erstellt. Diese Testdaten können sowohl in Dateien (sequenziell in spezifischen Mainframe-Formaten) als auch in Datenbanktabellen vorliegen. Zweitens müssen die Batch-Programme durch den Aufruf der bereitgestellten JCL (Job Control Language) ausgeführt werden. JCL gehört auf dem Mainframe zu jedem Batch. Drittens müssen die Ergebnisse ausgewertet werden, welche wiederum als Dateien oder in Datenbanktabellen vorliegen können.

Der gesamte Ablauf ist in der Architekturübersicht in Abbildung 1 schematisch dargestellt. Im Folgenden werden die Schritte einzeln beleuchtet.

Testdatensätze in Excel

Alle Testdaten werden in Excel-Tabellen erfasst. Durch die Verwendung eines bekannten Tools sind weder Einarbeitungszeit noch Kontakt mit Programmcode auf Seiten der Tester notwendig. Die Erstellung der Testdaten erfolgt nach dem Prinzip der Äquivalenzklassenanalyse für die einzelnen Felder der Eingabedateien [Spi12].

Darüber hinaus steht die gesamte Flexibilität von Excel-Tabellen zur Verfügung: Gruppierungen, farbliche Codierungen, Ein- und Ausblenden, automatisches Vervollständigen, Duplizieren von Spalten, Funktionen für Zufallszahlen, relative Datumsberechnungen und Verweise auf andere Zellen. Die Tester müssen sich lediglich an ein Grundgerüst innerhalb der Tabelle halten: Testfälle werden spaltenweise erfasst und beginnen ab einer festgelegten Zeile.

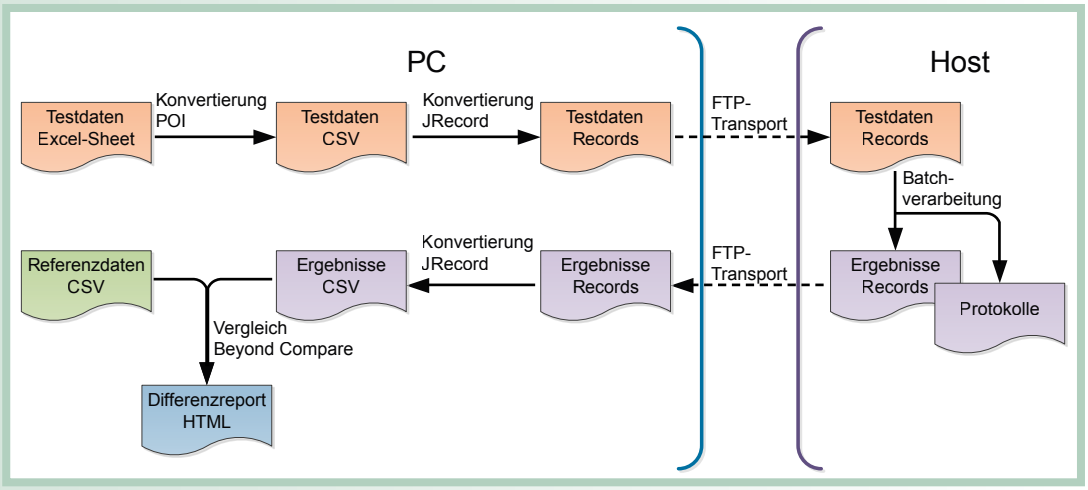

Abb. 1: Architekturübersicht

Erster Schritt zur Mainframe-Datei

Ein reiner CSV-Export reicht aufgrund einer unzureichenden Konvertierung der Datenformate nicht aus. Darüber hinaus werden Metainformationen in der Excel-Tabelle gepflegt (z. B. Testobjekt, Tester, Äquivalenzklassen, erwartetes Ergebnis), die beim Export herausgefiltert werden. Außerdem benötigen wir die Funktionalität, ein Datenfeld im Eingabeformat des Batches explizit als leer angeben zu können – auch hiermit ist der Excel-CSV-Export überfordert.

Wir verwenden Apache POI als Framework zum Lesen der Excel-Testdaten und zur Erstellung von CSV-Dateien, um die genannten Anforderungen für einen angepassten CSV-Export umzusetzen. Im Standard-Modus ist POI sehr speicherintensiv bei großen Excel-Tabellen, aber für ein reines Testtool ist dieser Aspekt unkritisch. Durch die Verwendung der Streaming-Variante kann der Speicherbedarf deutlich reduziert werden [POI].

Lost in Translation: EBCDIC vs. ASCII

Im nächsten Schritt werden die CSV-Dateien in das Mainframe-Format konvertiert. Das Mainframe-Betriebssystem z/OS hat eine eigene Zeichensatzcodierung namens EBCDIC (Extended Binary Coded Decimals Interchange Code) [Wiki]. Die Textfelder in den Testdaten müssen also von ASCII in den entsprechenden EBCDIC-Zeichensatz umcodiert werden. Auch für numerische Werte gibt es eigene Binärformate, unter anderem das sogenannte Packed-Decimal-Format für Festkommazahlen [Cob15], für die eine Konvertierung notwendig ist. Darüber hinaus werden unter z/OS häufig Dateien mit fester Länge verwendet, wobei auch die Längen der einzelnen Felder genau definiert sind.

Mit JRecord zum Fixed-Length-Record

Das Ergebnis der Extraktion der Testdaten aus Excel mittels POI – eine CSV-Datei – dient nun als Eingabe für JRecord, zusammen mit einer Beschreibung der CSV-Datei im XML-Format und einer Record-Beschreibung für die Zieldatei mittels einer COBOL-Schnittstellenbeschreibung – einem sogenannten Copybook. Dieses Copybook ist in Listing 1 für das reduzierte Beispiel eines Vertrags dargestellt, der je ein Attribut für die Vertragsnummer (numerisch, 10-stellig im Packed-Decimal-Format) und den Tarif (alphanumerisch, 10-stellig) besitzt. Die Beschreibung entspricht der COBOL-Syntax und wird in der Form auch in den Programmen verwendet, vergleichbar mit einer *struct* in C++.

```
* Copybook für Vertrag
  10 VERTRAG.
       15 VERTRAGSNUMMER  PIC S9(10) COMP-3.
       15 TARIF          PIC X(10).
```

Listing 1: COBOL-Schnittstellenbeschreibung für einen Vertrag

Für die Konvertierung der Testdaten in z/OS-Dateien verwenden wir die Open-Source-Bibliothek JRecord [JRec]. Diese wird aktiv entwickelt und ist äußerst hilfreich. Der Entwickler Bruce Martin hat schnellen Support per E-Mail geleistet und Probleme zügig behoben. Die Bibliothek ist wesentlich leistungsfähiger als der hier beschriebene Umfang. Zusätzlich zu der von uns verwendeten Funktionalität rund um Datensätze fester Länge verarbeitet die Bibliothek auch XML-Dateien, bietet ein Diff-Werkzeug und ein angegliedertes Reporting-Projekt.

In Listing 2 ist die Konvertierung eines Vertragssatzes beispielhaft dargestellt. Das Copybook aus Listing 1 (`VERTRAG.cpy`) wird geladen und die Zielcodierung `CP273` (EBCDIC, deutscher Zeichensatz) sowie eine feste Satzlänge (`Constants.IO_FIXED_LENGTH`) eingestellt. Über den von JRecord bereitgestellten `LineIOProvider` wird ein `Writer` instanziiert, der in eine Ausgabedatei `VERTRAG.bin` schreibt. Über das geladene Copybook-Layout wird eine neue Zeile (`Line`) erzeugt, die Attribute werden mit den gewünschten Werten belegt und der Satz wird geschrieben.

```
CobolCopybookLoader cbLoader = new CobolCopybookLoader();
ExternalRecord hostLayout = cbLoader.loadCopyBook("VERTRAG.cpy",
  CopybookLoader.SPLIT_NONE, 0, "CP273",
  Convert.FMT_MAINFRAME,0, null);
hostLayout.setFileStructure(Constants.IO_FIXED_LENGTH);
LayoutDetail oLayout = outLayout.asLayoutDetail();
AbstractLineWriter writer = LineIOProvider.getInstance()
  .getLineWriter(oLayout.getFileStructure());
writer.open("VERTRAG.bin");
outLine = new Line(oLayout);
outLine.getFieldValue("VERTRAGSNUMMER").set(1234567890);
outLine.getFieldValue("TARIF").set("Tarif1");
writer.write(outLine);
writer.close();
```

Listing 2: Konvertierung in Mainframe-Dateien mit JRecord

FTP für mehr als nur Transport

Die auf dem PC erstellten Eingabedateien werden über FTP binär auf den Mainframe übertragen. Das Betriebssystem z/OS liefert dafür standardmäßig einen FTP-Server, der von den Besonderheiten des Mainframe-Dateisystems abstrahiert, da zum Beispiel keine Verzeichnisse existieren.

Darüber hinaus bietet der FTP-Server die Möglichkeit, Batch-Jobs durch das Hochladen einer JCL anzustoßen, indem die hochgeladene Datei direkt an das Job Entry Subsystem übergeben und zur Ausführung gebracht wird. In Listing 3 ist dieses Vorgehen unter Verwendung der Apache Commons-Klasse `FTPClient` dargestellt. Über die `site`-Methode (entspricht dem FTP SITE-Kommando) wird dem FTP-Server die Einstellung `filetype=jes` mitgeteilt. Diese Einstellung ist spezifisch für den z/OS-FTP-Server und führt dazu, dass die hochgeladene Datei `Vertrag.jcl` als Job ausgeführt wird. Auf diese Weise lässt sich der Mainframe von einem Java-Programm steuern, das auf dem PC ausgeführt wird.

```
FTPClient ftpclient = new FTPClient();
ftpclient.connect(ftpServerAddress);
ftpclient.login(username, password);
ftpclient.site("filetype=jes");
FileInputStream inputStream = new FileInputStream("Vertrag.jcl");
ftpclient.storeFile("TEST.JCL(VERTRAG)",inputStream);
```

Listing 3: Ausführung von z/OS-Job über den Upload per FTP

Die Black Box „Batch“

Die benötigte JCL wird von den Entwicklern zusammen mit den Programmen bereitgestellt. Nach dem Starten der Jobs durch den Upload der JCL laufen die zu testenden Programme und schreiben Fehler- und Verarbeitungsprotokolle sowie Outputs.

Alles wird im Anschluss an die Verarbeitung per FTP wieder abgeholt. Im Hintergrund wird die Datenbank DB2 per Job mit den Ergebnissen beladen. Der gesamte Prozess ist für den Funktionstester transparent, sodass keine Kenntnisse der technischen Details des Ablaufs notwendig sind.

Aufbereitung der Ergebnisse

Der Rücktransport erfolgt ebenfalls per FTP im Binärmodus. Anschließend werden per JRecord die Ergebnisse zurückkonvertiert: Aus Records fester Länge in EBCDIC werden CSV-Dateien in ASCII. Das gilt für Ausgabedateien der Programme, Protokolle und Fehlerdateien.

Durch die asynchrone Batch-Ausführung ist die Festlegung eines Endekriteriums schwierig. Das Java-Programm muss wissen, wann die Ergebnisdateien auf dem Host vorliegen, damit sichergestellt ist, dass es auch alle Daten wieder zurück in die PC-Umgebung schaffen. Das Problem wurde mittels einer im Batch erstellten Signaldatei gelöst, auf die das Testtool aktiv wartet.

Ist alles erstellt und transportiert worden, sind die Ergebnisse wieder auf dem PC lesbar – und durch die Rückkonvertierung ins CSV-Format sogar in Excel zu öffnen.

Regressionstest mit Referenzen

Die Sollergebnisse für den Test sind im Vorfeld nicht festgelegt. Stattdessen werden die korrekten Ergebnisse nach der Prüfung als Referenzergebnisse und damit als Sollwerte für zukünftige Tests festgelegt. Hierfür ist ein einmaliger manueller Kontrollaufwand nötig. Bei späteren Änderungen ist dann ein Vergleich auf Basis der CSV-Ergebnisse möglich, zum Beispiel mit dem proprietären Beyond Compare oder anderen Diff-Werkzeugen.

Dieser Vergleich wird über ein Skript gesteuert, das ebenfalls aus Java heraus über die Klasse `ProcessBuilder` angestoßen wird. Aus Beyond Compare lassen sich verschiedene Report-Typen erzeugen. Bewährt hat sich ein HTML-Report, der alle Dateien eines Verzeichnisses mit ihrem Pendant aus den Referenzdaten vergleicht. Identische Dateien werden nur mit Namen aufgeführt, während für Abweichungen die jeweiligen Zeilen gegenübergestellt und die tatsächlichen Unterschiede farblich hervorgehoben werden. Der Vergleich arbeitet dabei so intelligent, dass eine neu hinzugekommene Zeile in den neuen Ergebnissen als solche erkannt wird und sich keine grundsätzliche Verschiebung für den Rest der Datei ergibt.

Ein Klick pro Batch

Wenn die Konfiguration vorgenommen worden ist, läuft der komplette Vorgang automatisch, bis hin zum Referenzvergleich. Alles wird per Java aus einem Programm heraus gesteuert. Java extrahiert die Daten aus Excel, konvertiert sie bis zum Host-tauglichen Format, transportiert sie, stößt die Verarbeitung an, sammelt Ergebnisse ein und erstellt einen Differenzreport.

Das Testtool folgt dem Prinzip *Convention over Configuration*. Mit entsprechenden Defaults kann man schnell neue Batches anbinden oder bestehende Tests um neue Ein- oder Ausgabedateien erweitern. Diese Anpassungen können durch die Tester vorgenommen werden, ohne dass der Programmcode des Testtools geändert werden muss.

Kleines Tool mit großer Wirkung

Der Ansatz trägt auch für abweichende Szenarien: Falls die Batches keine Eingabedateien, sondern Daten aus der Datenbank lesen, können diese Daten transparent für das Testtool in die auf dem Mainframe vorherrschende DB2-Datenbank importiert oder aus dieser exportiert werden. Die Testdaten werden dafür mit derselben Tool-Kette wie zuvor in eine Datei konvertiert, deren Struktur den Spalten der Tabelle entspricht. In den Batch wird dann JCL für den Aufruf des LOAD-Utility beziehungsweise UNLOAD-Utility integriert. Diese beiden DB2-Standard-Tools ermöglichen den Import beziehungsweise Export der Dateien in die DB2-Tabellen.

Der größte Vorteil des kleinen Tools: Die Tester müssen den Mainframe nicht kennen, sondern können mit der Standardsoftware Excel arbeiten. Das Arbeiten mit einer vertrauten Software steigert die Akzeptanz des Tools bei den Testern erheblich. Der Ansatz skaliert auch mit mehreren Testern und vielen Batch-Jobs hervorragend. Die Zahl der Testfälle ist nahezu unbegrenzt.

Regressionstests sind regelmäßig wiederkehrende Anforderungen, weswegen ein einfacher Vergleich mit vorherigen Testergebnissen besonders vorteilhaft ist. Die grafische Aufbereitung durch Beyond Compare ermöglicht einen übersichtlichen Report der Unterschiede zu einem vorherigen Testlauf, der schnell auswertbar ist.

Literatur und Links

[Als93] S. Alsop, Microsoft's Hermes: key network management system or myth?, Distributed Thinking, in: InfoWorld magazine, 22.2.1993, S. 4, s. `http://goo.gl/PGqoGf`

[Cob15] IBM Enterprise COBOL for z/OS Programming Guide Version 5.2.

[JRec] JRecord, `http://jrecord.sourceforge.net`

[POI] Apache POI HowTo, `http://poi.apache.org/spreadsheet/how-to.html#sxssf`

[RDz] IBM Rational Developer for z Systems, `http://www.ibm.com/developerworks/downloads/r/rdz/`

[Spi12] A. Spillner, T. Linz, Basiswissen Softwaretest, dpunkt.verlag, 2012

[Wiki] EBCDIC, Wikipedia, `http://de.wikipedia.org/wiki/Extended_Binary_Coded_Decimals_Interchange_Code`

Dr. Kay F. Hildebrand ist Business Analyst und Requirements Engineer bei der viadee IT-Unternehmensberatung. In seiner Beratungstätigkeit spezialisiert er sich auf Migrationsbegleitung und Geschäftsprozessmanagement.
E-Mail: kay.hildebrand@viadee.de

Tobias Voß ist Softwareentwickler und Projektleiter bei der viadee IT-Unternehmensberatung. Als Berater begleitet er Kunden bei der Umsetzung von individuellen Softwaresystemen auf der Basis von Java- oder Mainframe-Architekturen.
E-Mail: tobias.voss@viadee.de