Ein Translator für die COBOL-Java-Migration

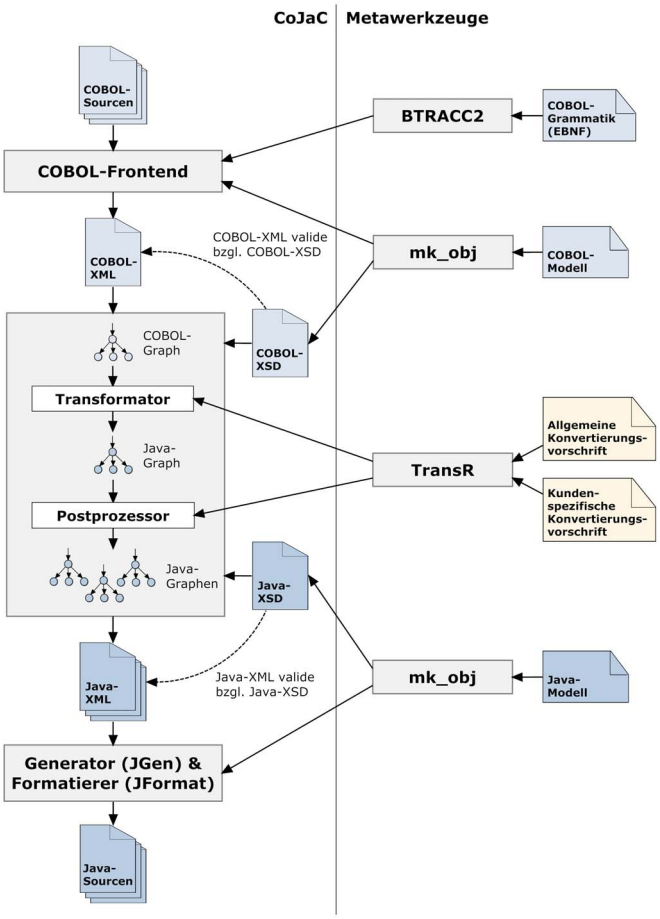
Uwe Erdmenger, Denis Uhlig
pro et con Innovative Informatikanwendungen GmbH, Dittesstraße 15, 09126 Chemnitz
uwe.erdmenger@proetcon.de, denis.uhlig@proetcon.de

Abstract

In [1] wurden von pro et con bereits erste Ansätze der Migration von COBOL nach Java dargestellt. Nach dem erfolgreichen Abschluss des Projektes *SOAMIG*[1] sollen in diesem Beitrag die Ergebnisse im Bereich der COBOL-Migration vorgestellt werden. Neben einem Überblick über das entstandene Werkzeug *CoJaC* (**CO**BOL to **Ja**va **C**onverter) sollen dabei die Migration der Schnittstellen eines COBOL-Programms und eine differenzierte Betrachtung des generierten Java-Codes bezüglich seiner Wartbarkeit und Qualität im Vordergrund stehen.

1 CoJaC - COBOL to Java Converter

CoJaC besteht aus einer Reihe von Werkzeugen (*toolchain*), welche in ihrer Gesamtheit die Konvertierung von COBOL-Programmen in Java-Klassen ermöglichen. Alle Werkzeuge sind Entwicklungen von *pro et con*. Es handelt sich dabei um Komponenten der hauseigenen *Software Reengineering Architecture* (SRA [2]), welche für diese Funktionalität orchestriert wurden. Die folgende Abbildung zeigt einen Überblick des gesamten Werkzeugs:



Die Ein- und Ausgaben des Werkzeugs sind mit COBOL bzw. Java vorgegeben. Als Schnittstellenformat der einzelnen Verarbeitungsschritte kommt bei *CoJaC* XML zum Einsatz. Alle Werkzeuge der SRA nutzen ein einheitliches Modell zur Abbildung von COBOL respektive Java. Die Modellierung erfolgt in einem eigenen Format, welches vom Metatool *mk_obj* verarbeitet und den anderen Werkzeugen in entsprechender Form zur Verfügung gestellt wird. Bei *CoJaC* kommt das Modell als XML-Schema und Definition der internen Darstellung (C++) zum Einsatz.

Den Einstieg in die Verarbeitung bildet das COBOL-Frontend. Die COBOL-Programme werden analysiert und in das bereits beschriebene Zwischenformat überführt. Die entstandene XML-Datei ist die Eingabe des Modelltransformators. Dieser baut auf dem regelbasierten Metawerkzeug *TransR* auf. Der Transformator überführt den ihm übergebenen COBOL-Syntaxgraphen in eine Menge von Java-Graphen und stellt diese in Form von XML-Dateien bereit. Abschließend werden diese Ausgaben durch den Java-Generator *JGen* und den Formatierer *JFormat* in Java-Quellcode überführt.

2 Realisierung der eingebetteter Systeme

Neben der interaktiven Kommunikation zwischen Programmen haben externe Datenbestände im Allgemeinen eine große Bedeutung für produktive COBOL-Systeme. Dazu zählen unter anderem Dateien und Datenbanken. Bei der Migration eines Legacy-Systems müssen alle diese Schnittstellen bzw. deren Zugriffsoperationen migriert werden. *CoJaC* unterstützt in der aktuellen Version die Abbildung von COBOL-Servern in Java Webservices und die Migration von COBOL-Dateien und embedded SQL.

Bei der Abbildung der Kommunikationsschnittstellen der COBOL-Server als Services wird auf die Möglichkeiten der Java Webservices zurückgegriffen. Je ein COBOL-Server bildet einen Webservice. Für die Realisierung des Services kommt ein Bottom-Up-Verfahren zum Einsatz. Als erster Schritt des Verfahrens erfolgt eine Analyse der bisherigen Schnittstellen, welche zumeist über eine Middleware (z. B. TUXEDO) bedient wurden. Die Kommunikation findet über *Messages* in einem spezifischen Format statt. Das *Message*-Format wird in den COBOL-Servern definiert. Auf Basis dieser Informationen wird im zweiten Schritt der Verarbeitung eine Java-Klasse mit einer Schnittstellen-Methode generiert. Diese bedient das nach Java umgesetzte Meldungsformat. Die Methode und die Klasse werden über Annotationen (z. B. `@WebMethod`, `@WebService`) markiert. Anschließend werden vorhandene Werkzeuge wie JAXWS genutzt, um den eigentlichen Webservice zu erstellen und in Betrieb zu nehmen.

In COBOL gibt es Dateien verschiedener Speicherorganisation (sequentiell, index-sequentiell, relativ). Die Umset-

zung der Dateien aus COBOL sieht vor, die vorhandenen Dateien in Datenbanktabellen zu überführen. Die Informationen zu Dateinamen, Organisationsstrukturen, Schlüsselfeldern sowie die Satzstruktur werden in einem vorgelagerten Analyseschritt aus den COBOL-Programmen ermittelt. Auf Basis dieser Informationen werden SQL-Skripte zur Erstellung der Datenbanktabellen generiert, welche die COBOL-Dateien repräsentieren. Die Migration des COBOL-Programms orientiert sich an der von *CoJaC* vorgegebenen Strategie. Die in COBOL vorhandenen Befehle zur Interaktion mit Dateien werden als Aufrufe von Methoden nach Java übernommen.

Auch embedded SQL wird nicht direkt übernommen, da es in Java keinen echten Präprozessor gibt, welcher die eingebetteten SQL-Kommandos umsetzen würde. Aus diesem Grund werden die vorhandenen statischen SQL-Befehle in dynamisches SQL überführt. Da bei dieser Umsetzung viel organisatorischer Code notwendig ist (Datenbankverbindung herstellen, Exceptions behandeln, ...), werden, analog zu der Umsetzung der Dateien, Funktionen in einem Laufzeitsystem bereit gestellt.

3 Wartbarkeit und Codequalität

Generiertem Quellcode wird nachgesagt, nur schwer oder überhaupt nicht wartbar zu sein. Dass dieses Vorurteil nicht zutrifft, wurde jedoch schon mehrfach bewiesen. Als ein Beispiel seien die aus der Programmiersprache SPL automatisch generierten C++-Programme aus dem Projekt ARNO [3] benannt. Auch für das Werkzeug *CoJaC* wurde die Wartbarkeit und Lesbarkeit des Quellcodes als eine wesentliche Eigenschaft des Werkzeuges definiert. Die daraus entstehenden Vor- und auch Nachteile sollen in den folgenden Abschnitten diskutiert werden.

Ein wesentliches Konzept von *CoJaC* ist die Abbildung eines COBOL-Programms in ein semantisch äquivalentes Java-Programm bzw. eine Java-Klasse. Das bringt natürlich mit sich, dass die bestehende Programmstruktur übernommen wird. Wenn im Legacy-Code eine nach aktuellen Gesichtspunkten ungünstige Struktur vorlag, wird diese mit migriert. Wenn die Struktur verändert werden soll, muss sie entweder schon im Legacy-System (Sanierung) oder im neuen Java-System (Refaktorierung) vorgenommen werden.

Die COBOL-Datenobjekte werden in Java zu Objekten von Klassen der Laufzeitbibliothek, welche aus den COBOL-Typen abgeleitet wurden. Diese Klassen bilden die Eigenschaften der entsprechenden COBOL-Datentypen ab. Sie wurden darüber hinaus mit den entsprechenden Java-Standard-Typen (z. B. `CharSequence`, `Number`) verknüpft. Dadurch ist für die Weiterentwicklung des migrierten Systems die Möglichkeit gegeben, auch nach Java-Standard zu entwickeln, ohne an das durch die Migration vorgegebene Typsystem gebunden zu sein.

Auch das Laufzeitsystem an sich ist eine wichtige Design-Entscheidung bei diesem Ansatz. Es erlaubt die Verwendung von kurzen und sprechenden Aufrufen in den generierten Klassen, da die Implementierung der Funktionalität „versteckt“ werden kann. Das hat zur Folge, dass der generierte Quelltext kurz und verständlich bleibt. Da die Aufrufe aber kein „reiner“ Java-Quellcode sind, wird eine gewisse Bindung an das Laufzeitsystem unabdingbar. Eine zukünftige Weiterentwicklung ohne die Laufzeitfunktionen ist aber dennoch möglich.

Der gewählte Ansatz stellt einen gesunden Kompromiss zwischen der Übernahme alter Strukturen und Funktionen in einen typschen Java-Quelltext dar. Dadurch wird der Quelltext für sowohl COBOL- als auch für Java-Entwickler les- und wartbar. Selbstverständlich kann *CoJaC* an spezielle Kundenwünsche flexibel angepasst werden.

4 Zusammenfassung

Als Ergebnis des Projektes SOAMIG kann die erfolgreiche Konvertierung von COBOL-Legacy-Programmen in Java Webservices betrachtet werden. In einer Fallstudie wurde die Einsetzbarkeit des Verfahrens bereits nachgewiesen. Darüber hinaus ist der Ansatz, Quellcode zu erzeugen, der les- und wartbar ist, tragbar und Erfahrungen zeigen, dass der generierte Quelltext ca. die zwei- bis dreifache Zeilenanzahl (*Lines of Code*) umfasst. Da aber im Verfahren bereits Ansätze zur Strukturierung enthalten und in der Java-Welt mannigfaltig vorhanden sind, kann diese Größe als vertretbar bezeichnet werden. Die Performance der generierten Programme entspricht der neu entwickelter Java-Programme.

Literaturverzeichnis

[1] Erdmenger, U.: Vom COBOL-Server zum Java-Webservice.
12. Workshop Software-Reengineering (WSR 2010), Bad Honnef 3.-5. Mai 2010
In: GI-Softwaretechnik-Trends, Band 30, Heft 2, ISSN 0720-8928, S. 64 und 65

[2] Erdmenger, U.; Kaiser,U.; Loos, A.; Uhlig, D.:
Methoden u. Werkzeuge für die Software Migration.
In: Gimnich, R.; Kaiser, U.; Quante, J.; Winter, A. (Eds.): 10th Workshop Software-Reengineering (WSR 2008), 5.-7. May 2008, Bad Honnef.
Lecture Notes in Informatics, (LNI)-Proceedings, Volume P-126, S. 83-97

[3] Erdmenger, U.: SPL-Sprachkonvertierung im Rahmen einer BS2000 Migration
8. Workshop Software-Reengineering (WSR 2006), Bad Honnef 3.-5. Mai 2006
In: GI-Softwaretechnik-Trends, Band 26, Heft 2, ISSN 0720-8928, S. 73 und 74

[1] Das vom Bundesministerium für Bildung und Forschung geförderte Projekt SOAMIG beschäftigt sich mit der Migration von Legacy-Software in serviceorientierte Architekturen. Es wurde von pro et con Innovative Informatikanwendungen GmbH, Amadeus Germany, dem Institut für Softwaretechnik der Universität Koblenz-Landau (IST) und OFFIS e. V. als Partner im Verbund realisiert.