FAQ
Sprachtransformation von COBOL nach Java

1. Funktioniert das überhaupt automatisch? Habe ich da nicht zu viel Handarbeit?

Wir haben in mehreren Migrationsprojekten bewiesen, dass das funktioniert (z. B. COBOL nach Java bei ITZBund und SüdLeasing). Bei COBOL nach Java können derzeit große Teile des COBOL-Codes (ca. 90–95 %) automatisch konvertiert werden. Das Werkzeug CoJaC (COBOL to Java Converter) erzeugt dabei semantisch äquivalenten Java-Code. Die restlichen 5–10 % werden z. B. durch geeignete Kommentare deutlich ausgewiesen, was eine manuelle Konvertierung erleichtert.

2. Entsprechen die konvertierten Java-Programme einem objektorientierten Entwurf?

Die COBOL-Programme wurden in einer prozeduralen Sprache implementiert. Eine Neuaufteilung der Funktionalität auf einzelne Klassen und Methoden im Sinne des objektorientierten Entwurfs ist automatisiert nicht möglich. Das ist auch nicht unbedingt wünschenswert, schließlich müssen die Entwickler den Code auch wiedererkennen. Eine zukünftige, objektorientierte Weiterentwicklung der generierten Java-Applikation kann natürlich erfolgen.

3. Ist der entstehende Code wartbar und zur Weiterentwicklung geeignet?

Ja, definitiv. Der Grundaufbau des generierten Java-Codes ist analog zum COBOL-Code. Dadurch steigt der Wiedererkennungseffekt. Auch Kommentare werden an die korrekten Stellen des Zielcodes eingefügt. Die Entwickler des Altsystems finden sich (Java-Kenntnisse vorausgesetzt) schnell zurecht. Es gibt ausreichend Referenzprojekte für die automatische Sprachtransformation von COBOL nach Java, die praktisch beweisen, dass der generierte Code wartbar und performant ist. Die für Java verfügbaren, modernen IDEs (z. B. IntelliJ und Eclipse) erlauben dann eine komfortable Weiterentwicklung.

Bedenken werden oft von Mitbewerben geäußert, welche eigene COBOL-Entwicklungsumgebungen vermarkten und deshalb kein Interesse zeigen, „von COBOL weg“ zu migrieren.

4. Welche COBOL-Anweisungen sind nicht vollständig konvertierbar?

Nicht vollständig konvertierbar sind solche Anweisungen, für die Java nicht die entsprechenden sprachlichen Mittel zur Verfügung stellt. Das betrifft vor allem die sogenannten „compiler-directing statements“. Diese werden erst bei der Übersetzung des COBOL-Programmes ausgeführt, sozusagen in einer Präprozessor-Phase. Java stellt aber keinen Präprozessor zur Verfügung. Somit sind diese Anweisungen nicht mit den üblichen Mechanismen konvertierbar. Das bedeutet jedoch nicht, dass z. B. keine Copybooks konvertiert werden können. CoJaC beinhaltet einen ausgeklügelten Mechanismus zur Konvertierung von Copybooks in separate Java-Klassen.

pro et con
Innovative Informatikanwendungen GmbH
Reichenhainer Straße 29a
09126 Chemnitz

Telefon: +49 371 270951-0
Telefax: +49 371 270951-29

Internet: www.proetcon.de
E-Mail: info@proetcon.de

5. Wie sieht es mit GO TO aus? In Java stehen ja keine Sprunganweisungen zur Verfügung.

„`GO TO`“-Anweisungen sind komplett migrierbar. Dabei wird zunächst versucht, sie durch Java-typische Anweisungen zu ersetzen. Sprünge an das Ende einer Section werden z. B. durch eine `return`-Anweisung in Java übersetzt. Ist das nicht möglich, übernimmt das Laufzeitsystem die Ablaufsteuerung. Im Programm wird dann das `GO TO` durch den Aufruf einer `skip()`–Methode realisiert.

6. Entstehen bei der Migration nicht viele „Java-untypische“ Konstrukte, so dass es sich um ein COBOL-Programm in Java-Notation handelt?

Die Abbildungsvorschriften von COBOL-Konstrukten in Java-Konstrukte wurden bewusst so gewählt, dass der entstehende Java-Code Java-typisch ist. Es werden z. B. für die verschiedenen Datentypen Java-Klassen in einem Laufzeitsystem zur Verfügung gestellt, die alle notwendigen Informationen, wie Länge etc., kapseln und Methoden zu deren Verwaltung anbieten. Wo es möglich ist, werden direkt die nativen Java-Anweisungen verwendet, z. B. für Schleifen (`while`, `for`) oder `if`-Anweisungen.

7. Erkennen meine Programmierer den Quellcode wieder? Wie groß sind die strukturellen Änderungen?

Die prinzipielle Struktur eines Programmes bleibt identisch. Aus einem Programm wird eine Klasse, aus Datenstrukturen die (privaten) Datenfelder dieser Klasse und aus einzelnen Sections der `PROCEDURE DIVISION` werden die Klassen-Methoden. Dabei bleibt die Reihenfolge im Quelltext bestehen.

8. Entstehen dabei nicht eine Menge Klone?

Copybooks werden in separate Klassen konvertiert. Auch bei mehrfacher Verwendung dieser Copybooks entsteht dabei nur eine Klasse. Klone entstehen lediglich bei Verwendung von `COPY` mit `REPLACING`-Klausel oder bei `COPY` in der `PROCEDURE DIVISION`. Wir verfügen mit JPackage über ein Werkzeug, mit dem so entstandene Klone wieder zusammengeführt werden können.

9. Ist es dann nicht besser, den Code gleich in Java neu zu entwickeln?

Neben den dabei anfallenden, wesentlich höheren Kosten sollten Sie bei dieser Entscheidung auch die Projektdauer beachten. Wenn über 90 % des Java-Codes automatisch generiert werden, ist ein Programm natürlich schneller umgesetzt als von Hand. „Code freezes“ und eventuelle Sperrzeiten bei der Weiterentwicklung/Wartung des Programmes werden damit minimiert. Unsere Erfahrungen besagen, dass sich eine Software-Migration zu einer Neuentwicklung im Verhältnis von 1:8 verhält, d. h., wenn Sie 5 Personenjahre für ein Migrationsprojekt unter Nutzung von Transformationswerkzeugen veranschlagen, dann benötigen Sie für eine Neuentwicklung ein- und desselben Programmsystems 40 Personenjahre.

pro et con
Innovative Informatikanwendungen GmbH
Reichenhainer Straße 29a
09126 Chemnitz

Telefon: +49 371 270951-0
Telefax: +49 371 270951-29

Internet: www.proetcon.de
E-Mail: info@proetcon.de

Für eine Neuentwicklung existieren zwei alternative Ansätze: Entweder, Sie schreiben das originale Programm 1:1 in Java um. Dann wird der Code dem automatisch migrierten Code ähnlich sein, sodass Sie auch automatisch migrieren können. Oder Sie extrahieren die Business-Logik des Programmes, erstellen daraus eine Spezifikation und entwickeln dann das Programm neu. Die Extraktion der Business-Logik ist jedoch ein sehr aufwendiger und fehleranfälliger Prozess.

10. Können sich Java-Entwickler, die den COBOL-Code vorher nicht kannten und COBOL nicht beherrschen, in die migrierten Programme einarbeiten?

Natürlich. Das ist ein weiterer Vorteil der Software-Migration. Die Entwickler, welche die konvertierten Programme zukünftig warten, müssen kein COBOL kennen. Eine Einarbeitung kann allein anhand des Java-Codes erfolgen.

11. Ich möchte aber meine COBOL-Programmierer behalten. Wie hoch ist der Schulungsaufwand beim Umstieg?

Nun, Java müssen sie natürlich lernen. Daran führt kein Weg vorbei. Dafür haben diese Programmierer aber den Vorteil, dass sie sich bereits mit dem fachlichen Hintergrund der Programme auskennen, wogegen sich Neueinsteiger diesen natürlich erst aneignen müssen.

12. Sind die konvertierten Programme nicht wesentlich langsamer und verbrauchen viel mehr Speicherplatz?

Auch das ist ein Argument, welches häufig von Kunden geäußert wird, welches sich in bisherigen Projekten aber nie bestätigt hat. Performance-Probleme existierten dort nur lokal begrenzt auf einzelne Programme und konnten durch geeignete Optimierungsmaßnahmen beseitigt werden. Heutige JVM arbeiten dank der fortgeschrittenen Technologie der „just-in-time-Compiler“ die Java-Programme nur wenig langsamer ab als der Prozessor die kompilierten COBOL-Programme.

13. Welche Erfahrung besitzt pro et con bei der Software-Migration?

Werkzeuge für die Software-Migration erfordern einen großen Entwicklungsaufwand. Zusätzlich benötigt man ein ganz spezielles Informatik-Know-how (Compilertechnik). Wir besitzen mehr als 25 Jahre Erfahrung in der Entwicklung von Werkzeugen für die Software-Migration. Die Basis bilden Forschungsarbeiten auf dem Gebiet der Compilertechnik an der Fakultät für Informatik der TU Chemnitz. Wir betreiben weiterhin Forschung und Entwicklung in Zusammenarbeit mit den Universitäten Koblenz-Landau und Oldenburg. Die Ergebnisse spiegeln sich direkt in unseren Technologien und Werkzeugen wider. Im Verlauf der letzten Jahre entstanden dabei verschiedenste Parser, Code-Generatoren, Formatierungs- und Metawerkzeuge, die alle in unserer pecBOX (pro et con – Toolbox für die Software-Migration) zusammengefasst sind und die in unseren Migrationsprojekten zum Einsatz kommen. Kein Migrationsprojekt gleicht dem anderen. Bei neuen Projekten werden notwendige, neue Migrationswerkzeuge aus den in der pecBOX befindlichen Komponenten zusammengestellt, wodurch sich deren Entwicklungszeit verkürzt. Durch diesen generischen Ansatz werden Migrationsprojekte preiswerter und die Projektlaufzeit verkürzt sich. Das bestätigen zahlreiche Referenzkunden (Amadeus Germany GmbH, MAN Truck & Bus SE, ITZBund, SüdLeasing GmbH, ...).