Wege aus der Legacy-Falle

Viele Großrechner-Applikationen halten mit dem Veränderungstempo interner Unternehmensprozesse nicht Schritt. Sie müssen aufwendig modernisiert werden.

Von **Uwe Kaiser** und **Andreas Zilch***

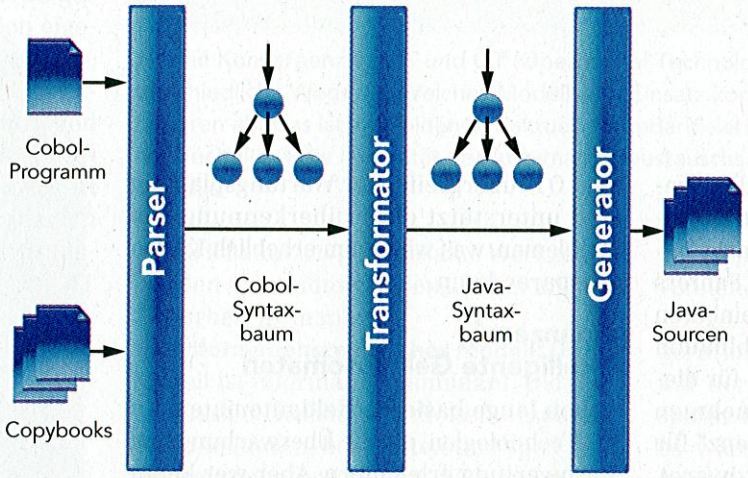
Für die Überarbeitung von Altapplikationen bieten sich drei Wege an: Eine komplette Neuentwicklung wäre die sauberste Lösung, braucht aber extrem viel Ressourcen und Zeit, zudem sind die Projektrisiken relativ hoch. Die Ablösung durch Standardsoftware ist womöglich empfehlenswert, wenn Standardkomponenten und -prozesse nicht durch individuelle Anpassungen verwässert werden. Die dritte Lösungsmöglichkeit ist die Softwaremigration. Auch diese Aufgabe ist komplex, zur Entlastung sollten daher Automatisierungstechniken und Software-Tools zum Einsatz kommen. Die Aufgabe dieser Werkzeuge ist die Konvertierung von Code. Die anspruchsvollste Aufgabe dabei ist es, Werkzeuge zur Programmkonvertierung, so genannte Translatoren, zu entwickeln.

Translatoren helfen

Die Arbeitsweise eines Translators ähnelt der eines Compilers, der Quellprogramme über verschiedene Zwischenstufen in ausführbaren Zielcode übersetzt: Ein Parser liest beispielsweise das Cobol-Programm einschließlich der dazugehörigen Copybooks ein und erzeugt daraus einen internen Syntaxbaum, der das vollständige Altprogramm repräsentiert. Er enthält auch alle Kommentare, um sie bei Bedarf in den Java-Zielcode einzufügen. Hier unterscheidet sich der Translator unter anderem vom Compiler, der Kommentare im Übersetzungsprozess verwirft. Im nächsten Konvertierungsschritt überführt ein Transformator den Cobol-Syntaxbaum in eine äquivalente Java-Ausführung. Die Basis dafür ist eine Abbildungsvorschrift, die definiert, wie einzelne Cobol-Konstrukte (etwa Definitionen und Anweisungen) in Java dargestellt werden. Eine solche Vorschrift zu erarbeiten ist eine der wichtigsten Aufgaben in der Translator-Entwicklung. Je genauer die Abbildung, desto größer ist die semantische Äquivalenz von Quell- und Zielcode, und umso einfacher lässt sich der Zielcode warten. Im letzten Konvertierungsschritt erzeugt der Generator aus dem Java-Syntaxbaum die Java-Sourcen.

So arbeitet ein Translator

In drei Schritten überführt ein Translator den Quell- in den Zielcode (hier am Beispiel einer Cobol-Java-Migration dargestellt).



Quelle: pro et con, Experton

Hoher Arbeitsaufwand

Erfahrungsgemäß beläuft sich der Entwicklungsaufwand für einen Translator unabhängig vom Quell- und Zielcode auf zirka drei bis vier Mannjahre, auch wenn Metawerkzeuge zur Verfügung stehen. Wurden Migrations-Tools nicht gemäß den wissenschaftlichen Grundlagen zur Compiler-Entwicklung entworfen, können der konvertierte Code und das Quellprogramm semantisch voneinander abweichen. Fachwissen im Compiler-Bau ist in jedem werkzeugbasierenden Migrationsprojekt erfolgsentscheidend.

Amadeus migriert BS2000

Ein Beispiel für eine gelungene Überführung ist das Projekt ARNO (Application Relocation to New Operating System) von Amadeus Germany, Anbieter von IT-Lösungen für die Reisebranche. Das Unternehmen hat seine BS2000-Anlagen durch Unix-Rechner abgelöst. Während des dreijährigen Projekts wurden Tausende von SPL-Programmen und SDF-Prozeduren mit mehreren Millionen Codezeilen automatisch nach C++ und Perl konvertiert. Zudem wurden das alte Filehandling-System in relationale Datenbanken migriert und der DCAM-Transaktionsmonitor von der Transactions-Processing-Plattform „openUTM" abgelöst.

Migrationswerkzeuge waren zu Projektbeginn teilweise vorhanden, einige wurden aber auch neu- oder weiterentwickelt. Das ist typisch für Projekte in individuellen Softwarelandschaften, wo Werkzeuge den jeweiligen Anforderungen angepasst werden müssen.

Erhebliche Einsparungen

Parallel dazu waren monatliche Programm-Updates erforderlich, um Kundenanforderungen zu erfüllen. Damit wurden Iterationen notwendig, da beispielsweise weiterentwickelte Sourcen wiederholt konvertiert werden mussten. „Der Testaufwand in einem solch komplexen Projekt wurde zu Beginn unterschätzt", räumt Werner Teppe, Senior Project Manager Development bei Amadeus Germany, ein. „Wir glaubten, ein umfangreiches Set manueller Tests sei ausreichend. Schnell zeigte sich, dass das nicht der Fall war." Daher habe man mit erheblichem Aufwand ein automatisches Test-Management aufgesetzt.

Bei jeder neuen Programmversion wurde ein definierter Testprozess iterativ durchlaufen, in dem auch die Konvertierungs-Tools selbst integriert worden seien, da diese sich auch ständig weiterentwickelten. Der Aufwand hat sich gelohnt: Die Kosten haben sich nach rund zwei Jahren durch eingesparte Hardware- und Lizenzkosten amortisiert. Zudem ebnete das Projekt den Weg für objektorientierte Programmierung und moderne Entwicklungsumgebungen. *(jha)*

***Uwe Kaiser** ist geschäftsführender Gesellschafter der pro et con GmbH. **Andreas Zilch** ist Lead Advisor bei der Experton Group.