COBOL

Der unterschätzte Dieselmotor der Wirtschaft

Michael H. Tucek
Tucek IT Services
michael@tucek.de

Agenda

- Ein paar Aussagen zu COBOL
- Eine kleiner Reality-Check zur Einschätzung der Sprache
- Was, bitteschön, ist „Legacy“

- Business Opportunities

COBOL ?? ---- Sie machen COBOL ????

COBOL is for morons. (Edsger Dijkstra)

The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence. (Edsger Dijkstra)

With respect to COBOL you can really do only one of two things: fight the disease or pretend that it does not exist. (Edsger Dijkstra)

Cobol has almost no fervent enthusiasts. As a programming tool, it has roughly the sex appeal of a wrench (Verrenkung). (Charles Petzold)

COBOL: (Synonymous with evil.) A weak, verbose, and flabby language used by card wallopers to do boring mindless things on dinosaur mainframes.

Java Forever…..oder:
Die Zukunft ist auch nicht mehr das, was sie mal war ;-))

Suns Programmiersprache ist out
Java wird zum neuen Cobol

02.01.2008 um 13:00 Uhr
Einst mit dem Siegeszug des Internet als die Programmiersprache der Zukunft gefeiert, gilt Java heute bereits veraltet. Ihr droht ein ähnliches Siechtum wie Cobol. (Computerwoche zitiert Infoworld)

Stimmen von der JAXX 2007 (aus einem Blog)

Zuerst ging es um das Thema AOP. Ca. die Hälfte der Anwesenden nutzt AOP. Die meisten aber nur dadurch, dass sie Spring einsetzen. Nur ein kleiner Teil benutzt AspectJ. Niemand benutzt AOP für was anderes als die üblichen rein technischen Beispiele Logging, Tracing, Transaktionshandling und Security.
Die meisten finden AOP eine interessante Technologie, hatten aber im Ansatz eine Vorstellung davon, wie man funktionale Aspekte definieren soll. Ein Teil der Teilnehmer fand AOP in Gesamtgröße a la AspectJ auch viel zu kompliziert.
Bei der Komplexität fanden einige, dass Java zu kompliziert wird. Dierk König (sinngemäß): "Für mich hat JDK 1.4 gereicht. Java 5 ist für mich zu kompliziert." Es gab eine ziemliche Einigkeit darüber, dass die Zukunft der Programmiersprachen in Einfachheit liegen müsste. Dabei gingen die Meinungen auseinander, ob man ein einfacheres Java braucht, eher was Dynamisches a la Ruby oder Groovy oder gar etwas Funktionales a la Scala. Dabei gab es ein erstaunliches Interesse an funktionaler Programmierung. Das rekursive Programmieren sei vielleicht am Anfang gewöhnungsbedürftig, aber das sei OO auch.
Außerdem bieten funktionale Programmierung den Vorteil, dass man das Programm transparent parallel ausführen kann.
Dann gab es noch eine Diskussion um Closures. Alle wollen Closures. Dierk sagt, dass noch nicht raus ist, ob und wenn ja in welcher Form Closures in Java 7 enthalten werden.

"Ils sont fou les informaticiens."

HOPL1, die Enzyklopädie der Computersprachen, listet seit 1968

8.512 Sprachen.

Das macht seit 1968 im Schnitt mehr als 200 neue Sprachen pro Jahr.

Asterix würde sagen: „Die spinnen, die Informatiker.“

Die ganz andere Wahrheit…….

Über die Wichtigkeit COBOL zu sein. (frei nach Oscar Wilde)

- 75% of all business data is processed in COBOL. - Gartner Group

- There are between 180 billion and 200 billion lines of COBOL code in use worldwide.
  This represents over 60 percent of the world's computer code. - Gartner Group

- Existing legacy systems are predominantly written in COBOL. 15% of all new applications (5 billion lines) through 2005 will be in COBOL. - Gartner Group

- CICS transaction volume (such as COBOL-based ATM transactions) grew from 20 billion per day in 1998 to 30 billion per day in 2002. - The Cobol Report

- Replacement costs for COBOL systems, estimated at $25 per line, are in the hundreds of billions of dollars. Tactical Strategy Group

- "Integration with Legacies" is the number one concern of IT managers in 2003. - Gartner Group

- There are over 90,000 COBOL programmers in North America in 2002. Over the next four years there will be a 13% decrease in their number due to retirement and death. - Gartner Group

Nach Einschätzung von Big Blue laufen gegenwärtig weltweit noch mehr als 200 Milliarden Codezeilen.
"*IBM Looks to Modernize COBOL," InfoWorld, May 11, 2004*

•The most highly paid programmers in the next ten years are going to be COBOL programmers who know the Internet. GIGA Group

Ergo:

" The truth is that ***modernized applications play a crucial role.***"

Giga Information Group
After the Bubble Bursts: *Legacy Application Renewal Options Draw Intense New Interest*

Was tun ?? Und wie ??

Modernisierung durch Migration ist das Zauberwort !

Es spart für den Kunden ZEIT und GELD !
Die erprobte und bewährte Funktionalität bleibt erhalten.

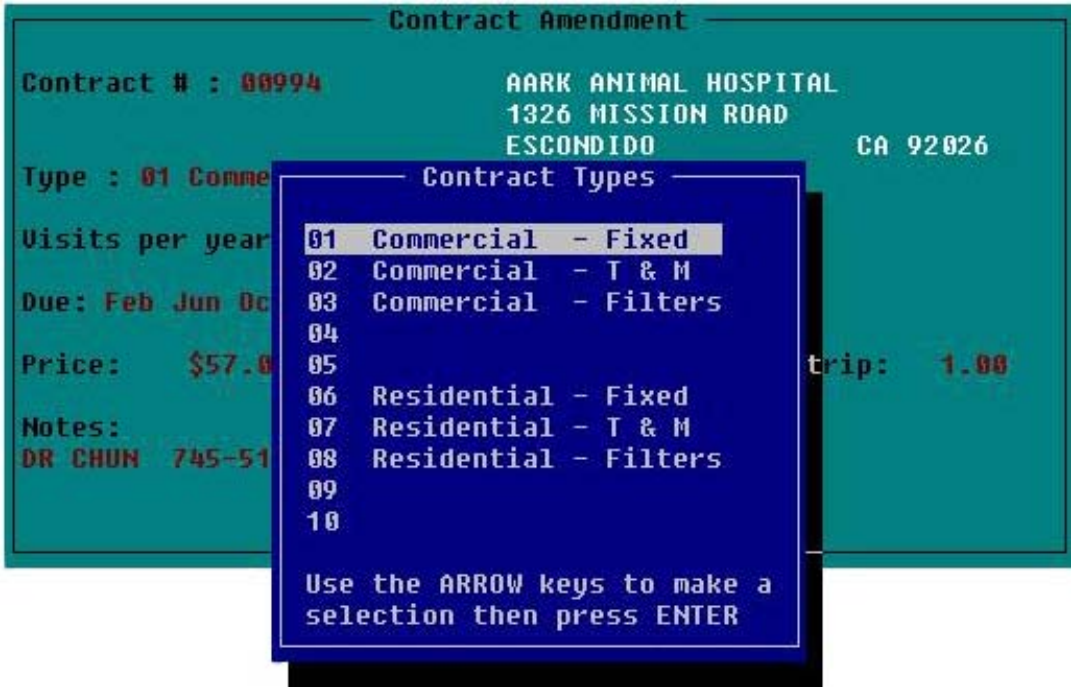
Und:

Modernes COBOL (!!) ermöglicht ALLES, was heute als „cool“ und „State of the art“ gilt.

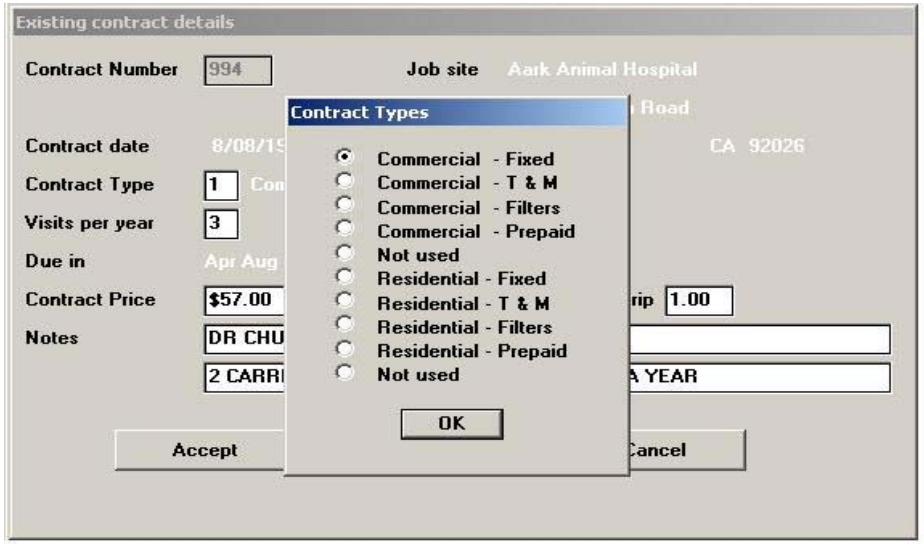
Migrationsthemen

- Das User-Interface
  Wird von den meisten Anwendern als altmodisch und „eine Schande“ betrachtet
- Datenbank-Anbindung / Business Intelligence
  Meist die einfachste Form der System/Daten-Integration
- Java Anbindung
  Wird allgemein als „modern“ betrachtet
- SOA
  Heute als Thema ein „Muss“ !!

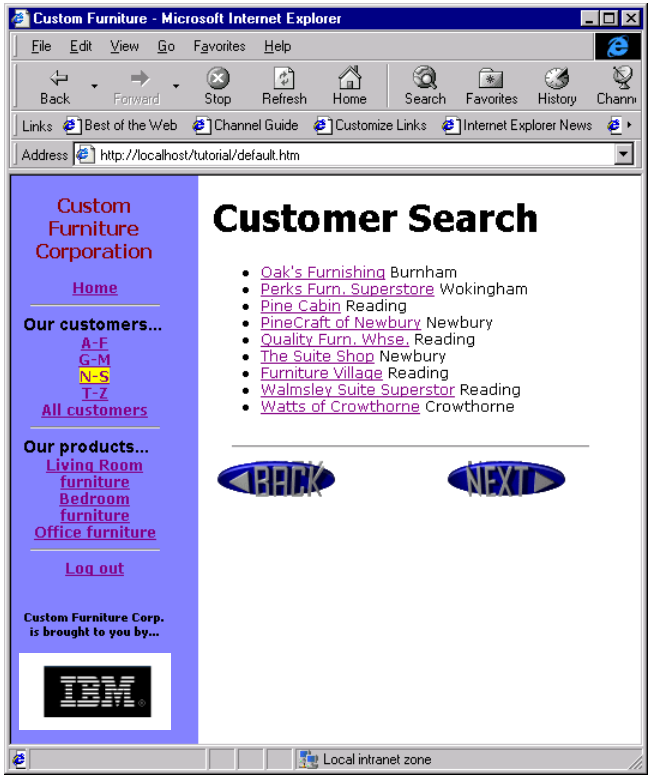
Vom „Es war einmal COBOL !“



Über aktuelle **(Windows-**) Oberflächen



Zum weltweiten Zugriff über Browser



Modernisierung des Frontends

Windows- oder Browser-Oberfläche:

Bei sehr vielen COBOL-Compilern als Compiler-Option enthalten

Aber:
Die Oberfläche sieht immer noch sehr „COBOL-like“ aus.

Alternativ ist ein grafisches Entwicklungswerkzeug einzusetzen, das heutzutage in der Bedienung einem Microsoft Visual Basic gleicht.

Um eine ansprechende Oberfläche zu erhalten ist manuelle Design-Arbeit erforderlich.

Modernisierung des Frontends

Externes Java-Frontend:

Moderne COBOL-Entwicklungswerkzeuge (bzw. deren Runtime-Umgebungen) bieten eine CALL-Schnittstelle zu Java.

Meist sind diese CALL‘s von beiden Seiten möglich.
Das heißt : COBOL ruft Java UND Java ruft COBOL. Natürlich mit Parameter- und Datenschnittstellen.

Nach heutigem Stand ausschließlich durch manuelle Arbeit zu erreichen.

Es gibt auch COBOL-Compiler, die Java-Code erzeugen und somit vollständig in Java eingebunden sind

Migration auf INFORMIX oder andere relationale Datenbanken

Die meisten existierenden Applikationen laufen bereits gegen relationale Datenbanken, wie Informix, DB2…….

Besonders im Mainframe-Bereich sind aber doch noch einige Anwendungen, die noch mit diesem oder jenem nicht-relationalen Datenhaltungs-System ausgestattet.
Meist sind dies ISAM-ähnliche oder CODASYL-Datenhaltungssysteme.

Hier ist tool-unterstützte, jedoch manuelle Arbeit erforderlich.
Die Daten-Zugriffe müssen isoliert und zu E/SQL – Statements umgewandelt werden.

Hierbei kann die betriebswirtschaftliche Programm-Funktionalität meist unverändert übernommen werden.

Die reine Daten-Migration ist meist zwar mittels Tools, aber teils erheblicher manueller Arbeit durchführbar

Einbindung in Java-Welten

Wie gehabt bei „Java-Frontend“

Moderne COBOL-Entwicklungswerkzeuge (bzw. deren Runtime-Umgebungen) bieten eine CALL-Schnittstelle zu Java.

Meist sind diese CALL‘s von beiden Seiten möglich.
Das heißt : COBOL ruft Java UND Java ruft COBOL. Natürlich mit Parameter- und Datenschnittstellen.

Es gibt auch COBOL-Compiler, die Java-Code erzeugen und somit vollständig in Java eingebunden sind

SOA – oder: universelle Wiederverwendbarkeit

Was ist „SOA“ eigentlich ?

Es ist die, nicht zwingend neue, Idee einmal entwickelte Funktionalitäten jederzeit und sprach- und umgebungsneutral allen anderen Applikation zur Verfügung zu stellen.

Services (also „Funktionalitäten“) stehen in einer Datenbank systemweit zur Verfügung und werden mittels eine „Brokers“ verteilt.

Dort sind sie beschrieben mit

NAME, Daten-Eingang, Daten-Ausgang.

Also einfach: Wie heißt der Service, was geht rein, was geht raus.

Schnittstellen werden mittels des Brokers angepasst.

Informationen über die Sprache, in der dieser Service geschrieben ist sind entbehrlich.

Dies kann natürlich auch COBOL sein.

In der Realität sieht es meist so aus, daß bestimmte, lebensnotwendige Berechnungen (z.B.: Risiko-Abschätzung bei Versicherungen, Druckberechnung bei Röhrenherstellern..) als erprobte und zuverlässige Funktionen zur Verfügung gestellt werden.

Diese Funktionen zu isolieren ist oft toolgestützt, aber in den seltensten Fällen ohne erhebliche manuelle Unterstützung nicht möglich.

Fazit

Es stellt sich die Frage, ob sich eine Neuentwicklung mit einer „anderen Sprache“ lohnt.
Im Idealfall hat man hinterher die selbe Funktionalität.

Und:
Die Geschichte hat gezeigt, dass angeblich moderne Sprachen und (pardon !) „Religionen“ dann doch recht kurzlebig waren.

„Never touch a running system“ ist zwar keinesfalls eine Lösung, aber solange COBOL revisionssicher ist - und auch von Nicht-Programmierern verstanden werden kann - sollte man vorhandene Groß-Systeme eher step-by-step modernisieren statt mit einem „Big-Bang-Ansatz“ ein extremes Risiko für das laufende Geschäft einzugehen.

Wie sieht die Zukunft von COBOL aus ???

„Vorhersagen sind schwierig,. insbesondere wenn sie die Zukunft betreffen“

Vielen Dank für Ihre Aufmerksamkeit !