

München, 24. November 2009

Bestandssysteme risikolos modernisieren

Dr. Markus Pizka

- 1** Bestandssoftware: Alt \neq Legacy
- 2** Warum modernisieren – reale Herausforderungen?
- 3** Wie kosteneffizient und risikolos modernisieren?
- 4** Erfahrungsberichte

Vortrag: Bestandssysteme risikolos modernisieren

COBOL lebt:

200 billion lines of COBOL are in use today (65% of the total software). This represents a **\$2 trillion dollar total investment**. - Gartner Group

5 billion lines of new COBOL are developed **every year**. - Gartner Group.

COBOL applications **process over 80% of all daily business transactions** and mainframe platforms store 70% of the all the data

Quelle: Future of COBOL, LegacyJ Corporation. 2003.
<http://www.legacyj.com/cobol/FutureOfCobol.pdf>

===>

"Alt" NOT = "Legacy"

Bestandssysteme sind alt, da sie erfolgreich sind.

- Hoher Wert für Unternehmen
- Unterstützen unternehmenskritische Prozesse
- Fachliches, technisches Wissen im Code dokumentiert

D. Parnas 94: **Software alert** ...all successful products

- mangelnde Bewegung, unkundige Behandlung

- Gewichtszunahme, Sinkende Leistung
- Abnehmende Zuverlässigkeit

- Alterung aufhalten, Nachdokumentieren
- Restrukturieren, Amputieren

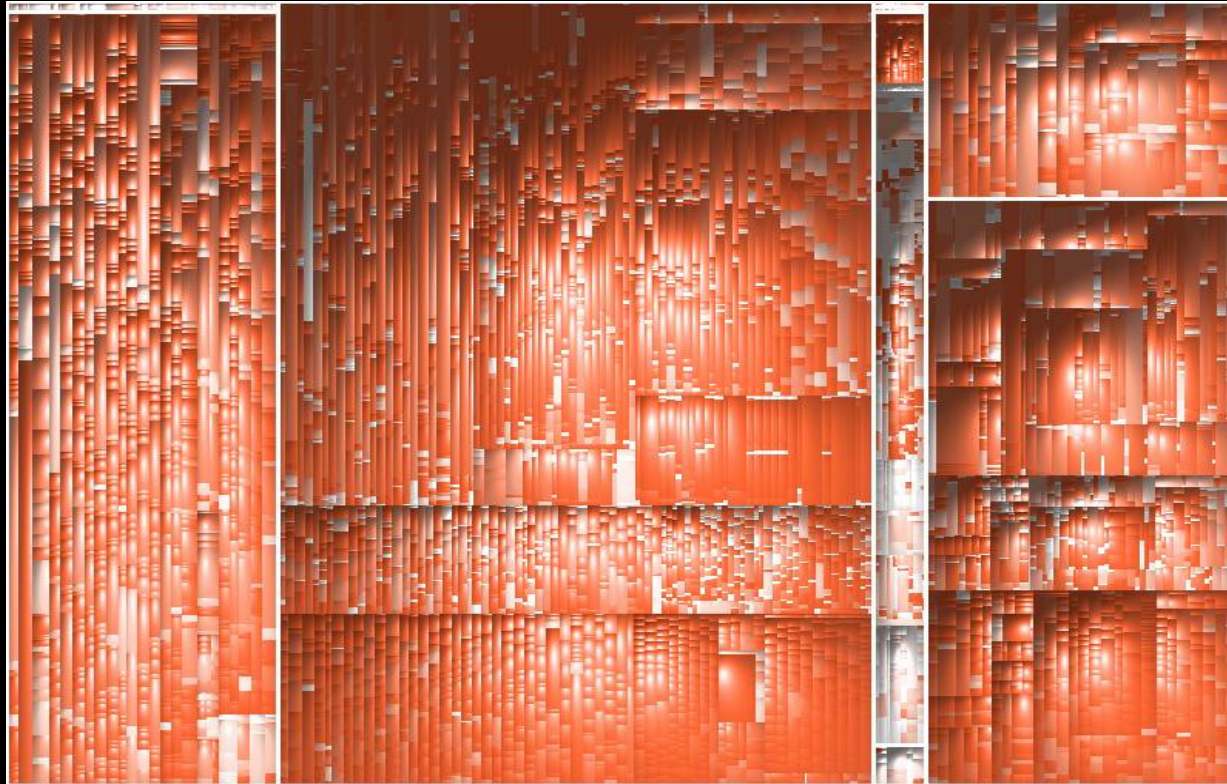
Mas, Pizka: Web-basierte und andere junge Legacy-Systeme

- Phänomen **nicht primär** alters- / technologieabhängig

===>

Reale Mängel: Redundanz

Bis zu 90% des Programmtextes Copy + Paste



1 MLOC statt 10 MLOC: 1,5 Mio EUR statt 15 Mio EUR p.a.

==>

F3=Verlassen F8=Vorwärts

Strukturelle Mängel: IF - AND - OR - Ketten

```
IF ZVDTA-BIS-CJMT (IDFA9) NN
OR ZNDFA-MT (IDFA9) NUMERIC
AND ZNDFA-MT (IDFA9) > 96
OR (ZNDFA-SA (IDFA9) = ,40'
OR ZNDFA-SA (IDFA9) = '69')
AND ZNDFA-VA (IDFA9) > 96
OR ZNDFA-MT (IDFA9) NUMERIC
AND ZNDFA-BIS-CJ (IDFA9) NOT = WA-VORJAHR-CJ
AND ZNDFA-BIS-CJ (IDFA9) NOT = WA-ABU-CJ
OR ZNDFA-SA (IDFA9) = ,70'
AND ZNDFA-VA (IDFA9) = '15'
AND ZNDFA-ST (IDFA9) = '15'
AND ZNDFA-WERT-ZT (IDFA9) NUMERIC
AND ZNDFA-WERT-ZT (IDFA9) = ZERO
AND ZNDFA-SICHER (IDFA9) NUMERIC
AND ZNDFA-SICHER (IDFA9) = ZERO
OR ZNDFA-S-STELLE (IDFA9) = ,RTG'
AND ZNDFA-SA (IDFA9) = ,70'
AND ZNDFA-SICHER (IDFA9) NUMERIC
AND ZNDFA-SICHER (IDFA9) = ZERO
AND (ZNDFA-VA (IDFA9) = ,25'
OR ZNDFA-VA (IDFA9) = '25'
OR ZNDFA-VA (IDFA9) = ,35'
OR ZNDFA-VA (IDFA9) = ,36'
```

Bei 24 Zeilen Umblättern notwendig

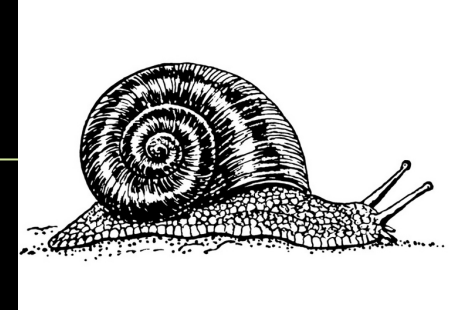
```
OR ZNDFA-VA (IDFA9) = ,45'
OR ZNDFA-VA (IDFA9) = ,46')
OR (ZNDFA-S-STELLE (IDFA9) = ,JUZ'
OR ZNDFA-S-STELLE (IDFA9) = ,BWE'
OR ZNDFA-S-STELLE (IDFA9) = ,KLP')
AND ZNDFA-SA (IDFA9) = ,90'
AND ZNDFA-WERT-ZT (IDFA9) NUMERIC
AND ZNDFA-WERT-ZT (IDFA9) = ZERO
AND ZNDFA-SICHER (IDFA9) NUMERIC
AND ZNDFA-SICHER (IDFA9) = ZERO
AND (ZNDFA-VA (IDFA9) = ,25'
OR ZNDFA-VA (IDFA9) = ,30'
OR ZNDFA-VA (IDFA9) = ,32'
OR ZNDFA-VA (IDFA9) = ,33'
OR ZNDFA-VA (IDFA9) = ,34'
OR ZNDFA-VA (IDFA9) = ,35'
OR ZNDFA-VA (IDFA9) = ,36'
OR ZNDFA-VA (IDFA9) = ,45'
OR ZNDFA-VA (IDFA9) = ,46')
MOVE ,O' TO ZNDFA-OK (IDFA9)
END-IF.
```



Unter welcher **fachlichen** Bedingung wird ZNDFA-OK mit ,O' belegt?

+50% Kosten in der Wartung

Ineffiziente Algorithmen



```
CUR = FIRST  
FOR Z = 1 TO ANZ  
WHILE (CUR != NULL && NOT LAST  
  IF CUR->NEXT  
    THEN CUR = CUR->NEXT;  
  ELSE  
    LAST = '1';  
END;
```

> 100.000 EUR Mehrkosten im Rechenzentrum p.a.

===>

F3=Verlassen F8=Vorwärts

Workarounds

```
// If the dialog was closed with „cancel“  
// the selection disappears. We don't know  
why!
```

```
// solution: re-initialize the selection  
if (!allowed) {  
    final int level = this.getModel().getLevel();
```

```
// since the reinit is a change for the  
// controller we detach the listener  
// to avoid recursion
```

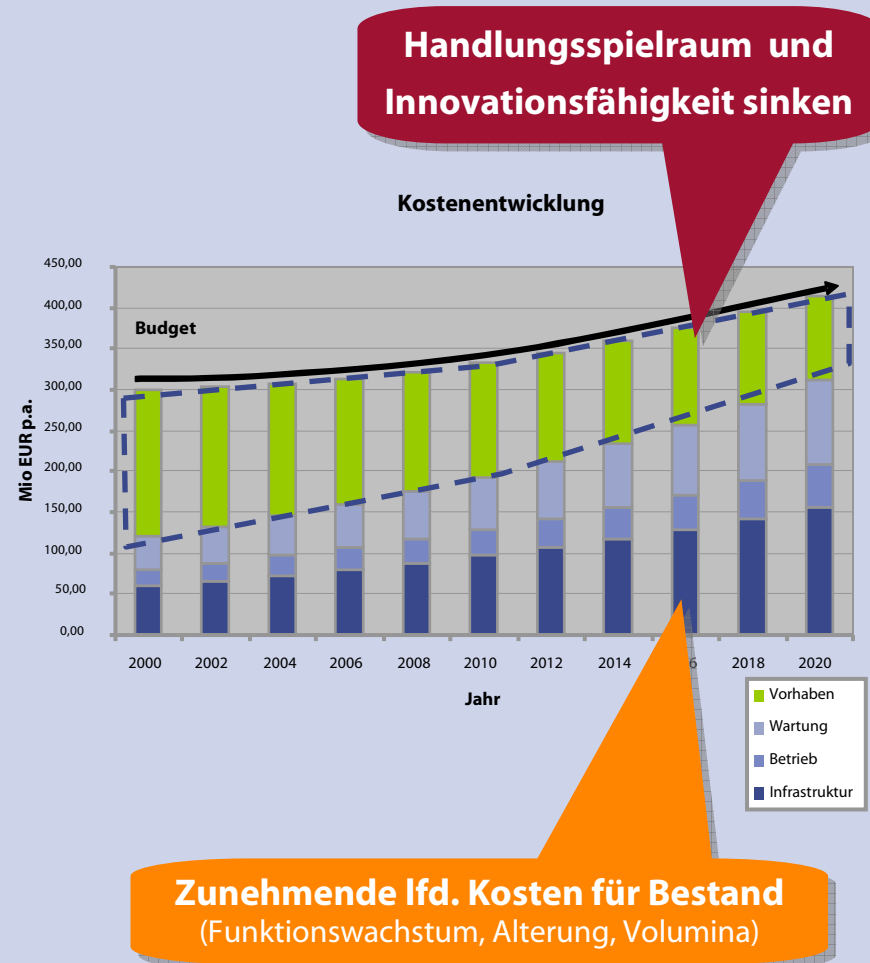


Konsequenzen

Enorme laufende Mehrkosten und erhebliche Verzögerung ... ohne Mehrwert!



- **Geschäftsschädigend!**
- **Verlust der Wettbewerbsfähigkeit**
- **Indien, China, etc. schläft nicht!**



Zukunftstechnologie COBOL?

- Geringere Produktivität
- Nicht wettbewerbsfähige Infrastruktur
 - Bibliotheken, Frameworks
 - Datenstrukturen (z.B. HashMap)
 - Werkzeugunterstützung (z.B. Refactoring)
- Besonders häufig strukturelle Mängel
 - COBOL: ø 39%
 - Java: ø 15%
- Abnehmende Verfügbarkeit von Experten

Sprache	Source Lines of Code / Function Point
COBOL	175
PL/I	126
Natural	100
C++	80
Java	80

Quelle: Davis Consulting

Nur wer handelt kann am Markt bestehen.

IT Effizienz steigern, Freiheitsgrade schaffen,
Unternehmenserfolg sichern

Wer stehen bleibt, verliert!



Status Quo

- Sinkender Marktanteil
- Steigende Fehlerhäufigkeit
- Hohe Kosten in Betrieb und Wartung
- Budget für Weiterentwicklung fehlt
- Prozessabdeckung mangelhaft
- Lange Änderungs-Zyklen
- Abhängigkeiten und Kopfmonopole

Effizienz steigern

Optimiert

- Marktführer
- Min. Fehlerrate durch innovative QS
- Effizienz durch Produktqualität
- Konzentration auf Innovation
- „IT enables Business“
- Agilität
- Flexible, verteilte Entwicklung

Aber, zum Scheitern verurteilt ...

■ Big-Bang-Ablösung

- Neuerstellung des bestehenden Systems
- **Teuer, zeitaufwendig und riskant**



■ Wrapper-Lösung

- Erweiterung der Schnittstellen z.B. durch WebServices
- **Grundsätzliche Schwächen bleiben erhalten und potenzieren sich.**

■ (Voll-)automatische Transformation

- Vorhandenes System werkzeuggestützt in andere Sprache übersetzen
- **Mängel bleiben bestehen und unbrauchbare Ergebnisse**

- **Objektive Bewertung**
- **Verständnis: Modernisierung ist eine vertikale Aufgabe**
 - Geschäftsprozess → Anforderungen → Dokumente → Design → Code → Werkzeuge → Entwicklungsprozess und **Menschen**
- **Schrittweise Durchführung**
 - Gewinnbringende und risikolose Einzelschritten
 - Gesundes Alt-System ist Voraussetzung für erfolgreiche Migration
- **Fokus Wirtschaftlichkeit**
 - Jede Maßnahme hat einen ökonomischen Nutzen
 - Abschöpfbare Netto-Einsparung im festgelegten Planungszeitraum
- **Mut, Geduld und Beständigkeit!**

1. Transparenz schaffen (KPIs)

2. Vision entwickeln

- Realistisches, aber ambitioniertes Bild des Zielzustands!
- Durch Kennzahlen präzisiert



3. Handlungsspielraum ermitteln

- Finanzielle Mittel, Zeit und Mitarbeiter stehen zur Verfügung?

4. Realistische Schritte finden

- Im Rahmen des Handlungsspielraums
- Berücksichtigung alle möglichen Maßnahmen (ggf. temp. Lösungen, z.B. Brücke zur Alt-Daten)



Auswahl

Code

- Bereinigung
 - Unused Code
 - „Unwichtige“ Funktionalität
 - Kopien
 - Fehler
- Restrukturierung, Refactoring
- Optimierung
 - DB-Zugriff
 - Effiziente Algorithmen & DS
- Migration: Hardware, Sprache, ...

Dokumentation

- Redokumentation (Reverse Eng.)
- Ersetzung von Literalen
- Kommentierung

Reengineering der Organisation

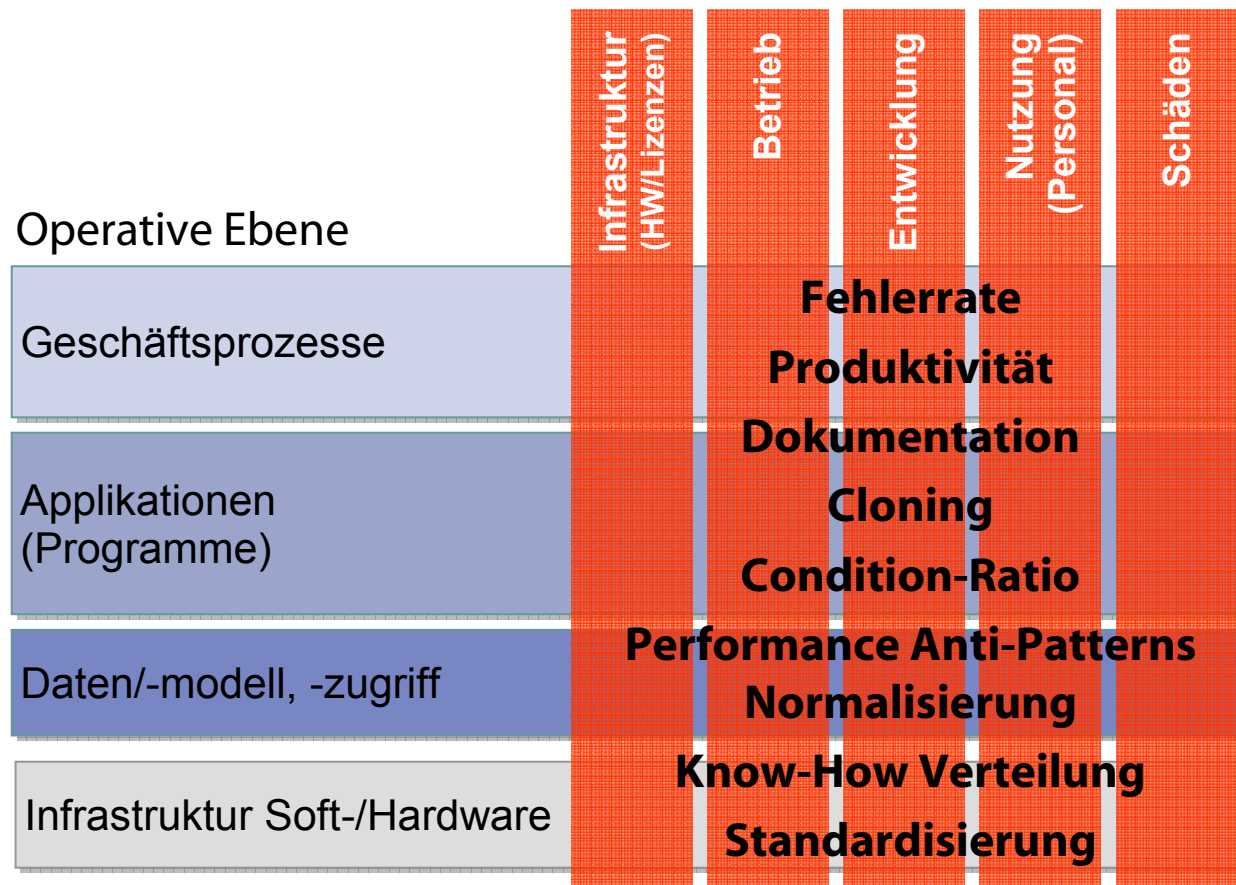
- Wissen verteilen (Rotation)
- Ausbildung
- Testautomatisierung
- Prozesseinführung

Infrastruktur

- Arbeitsplätze, Geräte, Werkzeuge, ...

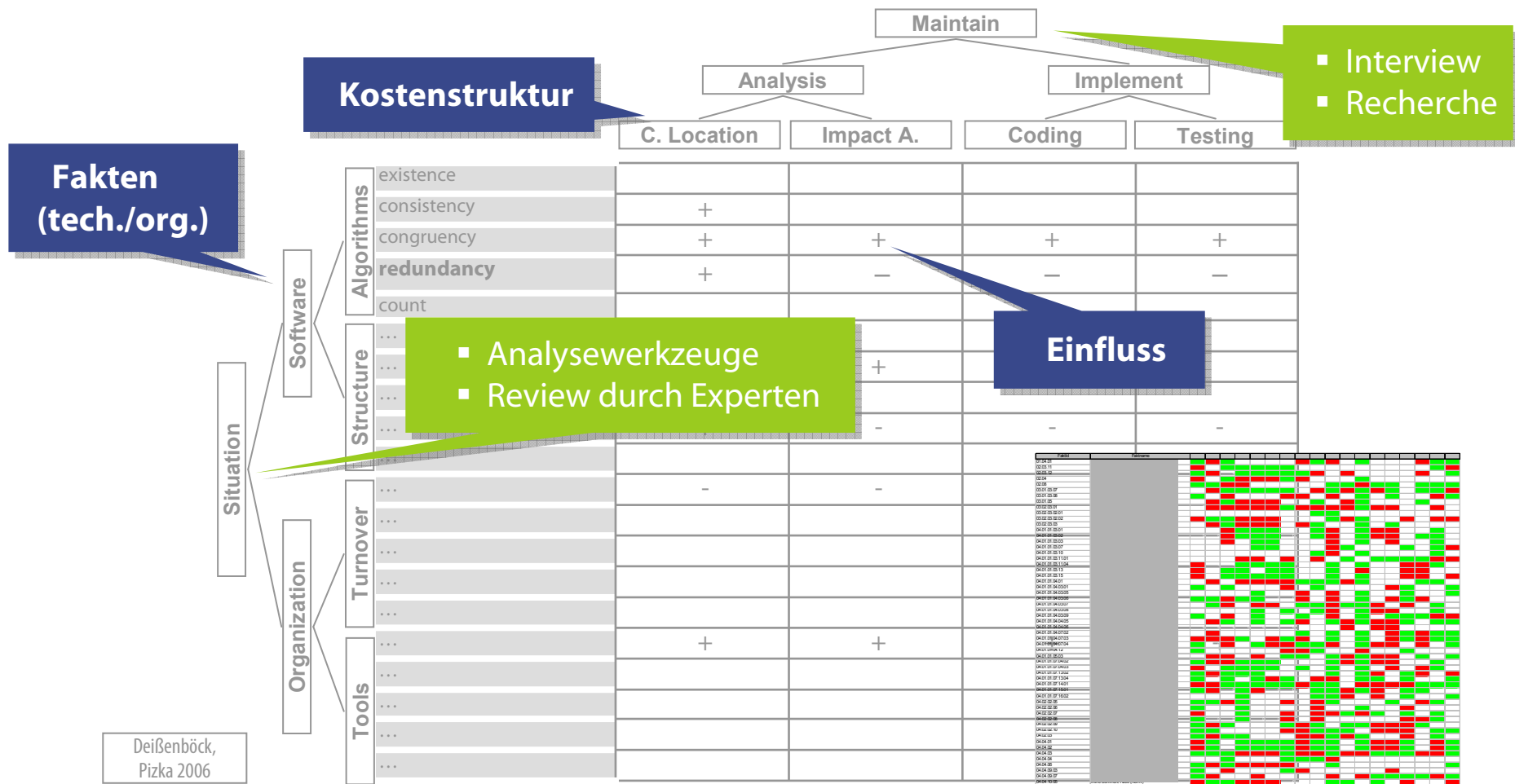
Transparenz schaffen

Kostenfaktor



Unser Qualitätsmodell

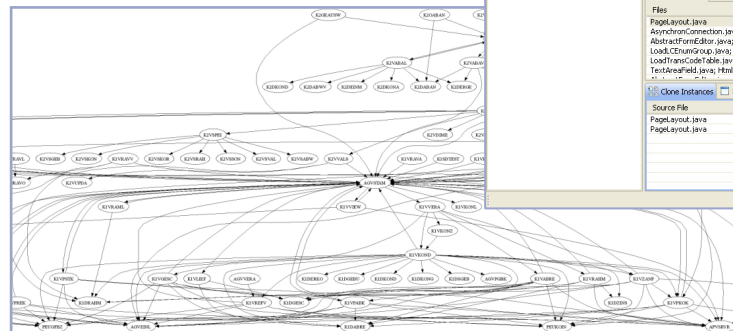
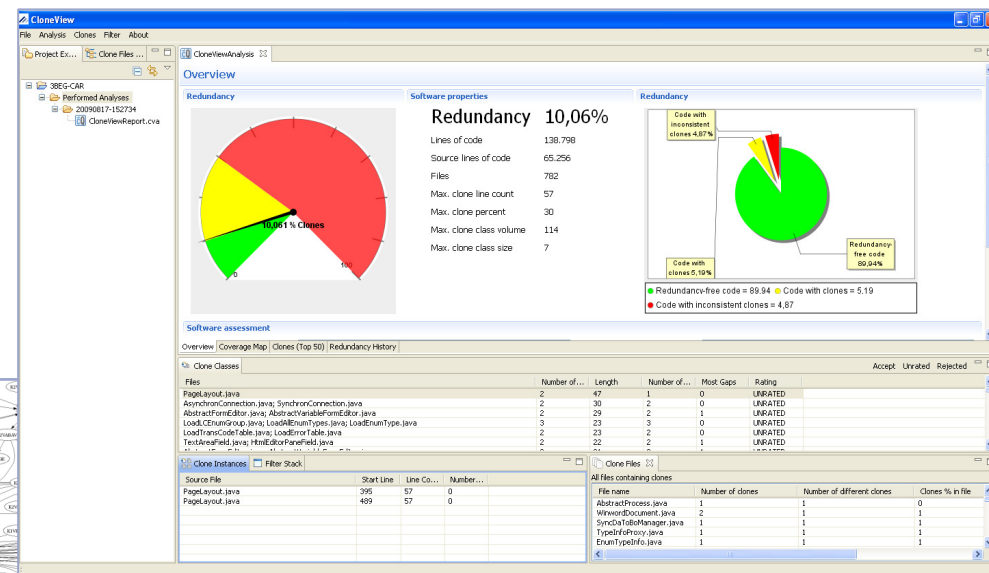
Von technisch/organisatorischer Qualität zu ökonomischer Realität



Deißenböck, Pizka 2006

Unsere Werkzeuge

- ConQAT (TUM)
 - Statische Analysen
- CloneView
 - Code Redundanz
 - Hot-Spots
- Call-Graph Gen
- Inspection Guideline
 - Anti Pattern Katalog



Potential - Beispiel

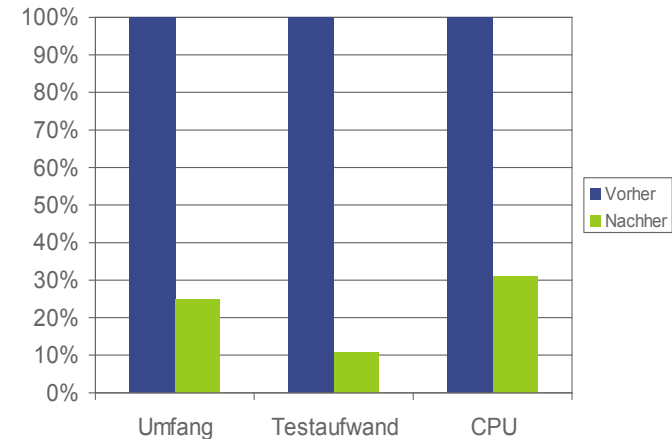


Position	Betrag p.a.	Potential	Begründung	Betrag p.a.
Entwicklung	250.000 EUR	- 30%	50% Redundanz, Struktur	75.000 EUR
Wartung	50.000 EUR	- 30%	s.o.	15.000 EUR
Betrieb (Personal)	150.000 EUR	- 50%	80% Berechtigungen	75.000 EUR
Betrieb (Ressourcen)	490.000 EUR	- 40%	Loop-Depth 8, Einzelsatzverarbeitung	196.000 EUR
Nutzung (Personal)	? EUR	- 25%	50 offene Anforderungen	? EUR
Schäden	> 1.000.000 EUR	- 20%	Bekannte Risiken	200.000 EUR
	1.940.000 EUR			561.000 EUR

Erfahrung: Wartbarkeit herstellen

■ Situation

- Geschäftskritisches System
- Nicht mehr wartbar
- Ziele:
 - Wartbarkeit herstellen
 - Betriebskosten -30%



■ Optimierung

1. Entfernung unused Code
2. Redundanzelimination
3. Restrukturierung
4. Schnittstellenbereinigung

Fehler beseitigt
Innovationsfähigkeit wiederhergestellt

Umfang - 75%
Testaufwand - 90 %
CPU-Kosten - 68 %

Amortisationszeitraum **14 Monate**

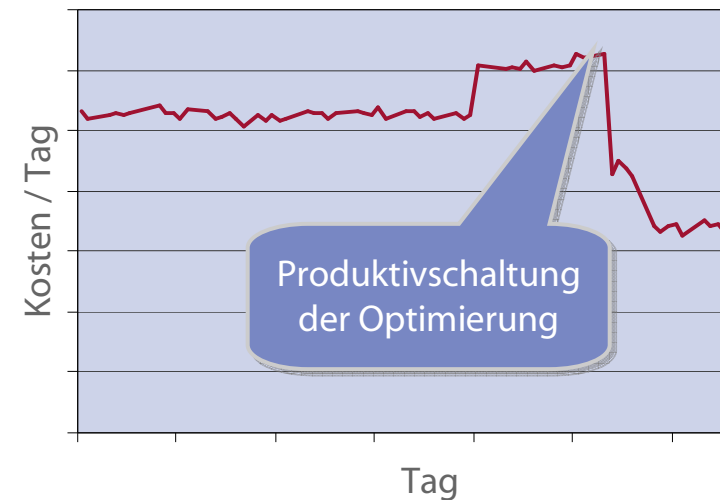
Erfahrung: Performance steigern

■ Situation

- Mio. DB-Zugriffe pro Ausführung
- Einzelsatzverarbeitung
- Triviale Algorithmen
- Nächtliche Batch-Verarbeitung
- Kostentreiber: CPU-Verbrauch

■ Optimierung:

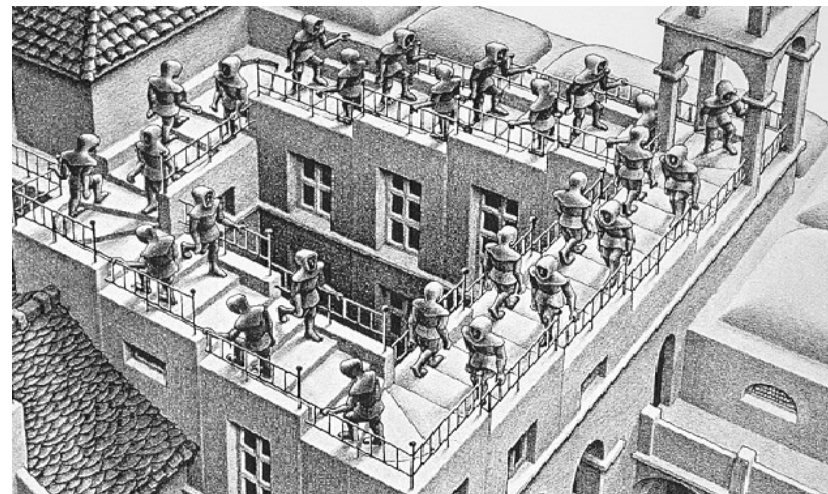
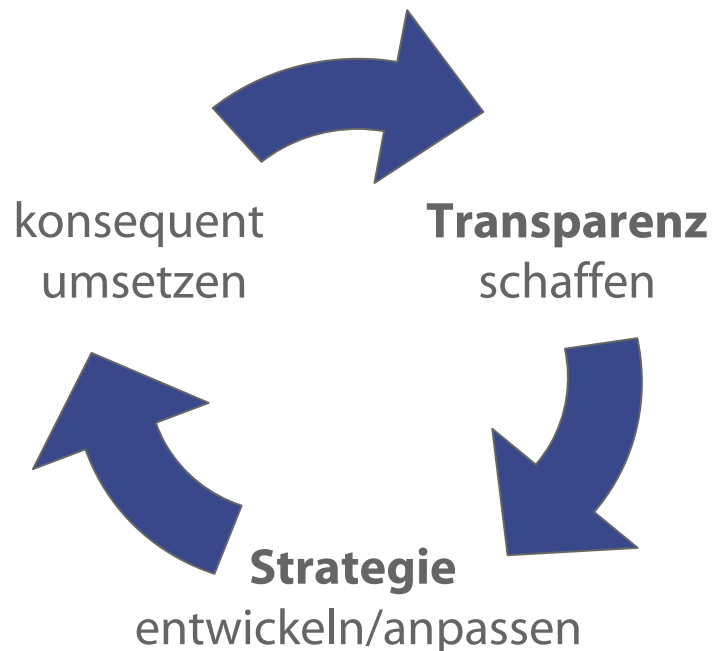
- Einführung Massendatenverarbeitung
- Vermeidung redundanter Zugriffe
- Abbildung von Logik in SQL
- Blockweise Verarbeitung
- Dynamisches SQL



CPU-Kosten -47 %
Amortisationszeitraum 7 Monate

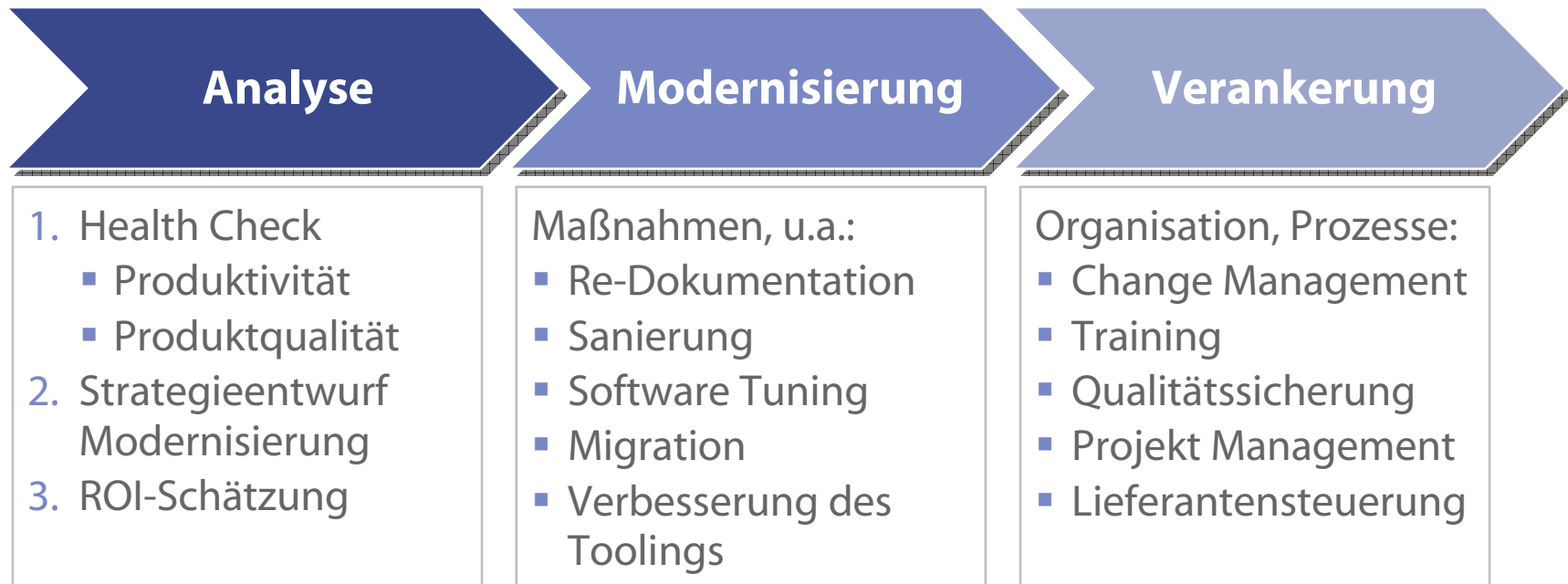
Erfolgreich Modernisieren

Software Modernisierung ist eine **kontinuierliche** Aufgabe – **ohne Ende**



Unsere Leistung: 20% – 80% Kosteneinsparung, ROI 6-18 Monate

Unsere Kunden: Automobil, Versicherungen, Banken, Energie



Rahmenbedingungen:

- Fokus: Software – ohne Produktbindung
- Übernahme von Verantwortung: Werkvertrag
- Erfolgsbasierte Bezahlung (Bonus-Malus-Regelung)



HABEN SIE FRAGEN?

Dr. Markus Pizka

pizka@itestra.de

+49 (179) 2108101