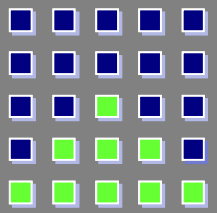


D E L T A
software
technology

The Generator Company

Automatisch fehlerfrei – eine Vision?

MDD&PL 2009 Leipzig, 24.3.2009

Rüdiger Schilling

Delta Software Technology GmbH

MDD, PL und GP vs. Wartung und Modernisierung

MDD, System Families & Generative Programming

- Primär Konzepte für Neuentwicklung

Fragen

- Lassen sich Modelle nachträglich ableiten?
- Welche Rolle spielen Systemfamlien/Produktlinien?
- Lässt sich GP in der Wartung einsetzen?
- Was machen wir mit Millionen 'alter' Cobol-Programme?

Die wichtigste Frage: **Warum**?

MDD, PL und GP vs. Wartung und Modernisierung

Software-(Neu)Entwicklung

- Größte Aufmerksamkeit in Publikationen und Forschung
- Viele Werkzeuge

Nur **10-20%** der Software-TCO

Wartung, Weiterentwicklung und Moderniserung

- Geringes Renommee **à** notwendiges Übel
- Kaum Forschung
- Wenig Werkzeuge

80-90% der Software-TCO

Mögliche Anwendungsfelder

- Massenänderung
  - Jahr 2000
  - UTF-16 Umstellung für EU-Harmonisierung
- Modernisierung
  - Plattformwechsel
  - Framework-Anpassung/Austausch
  - Architektur-Transformation

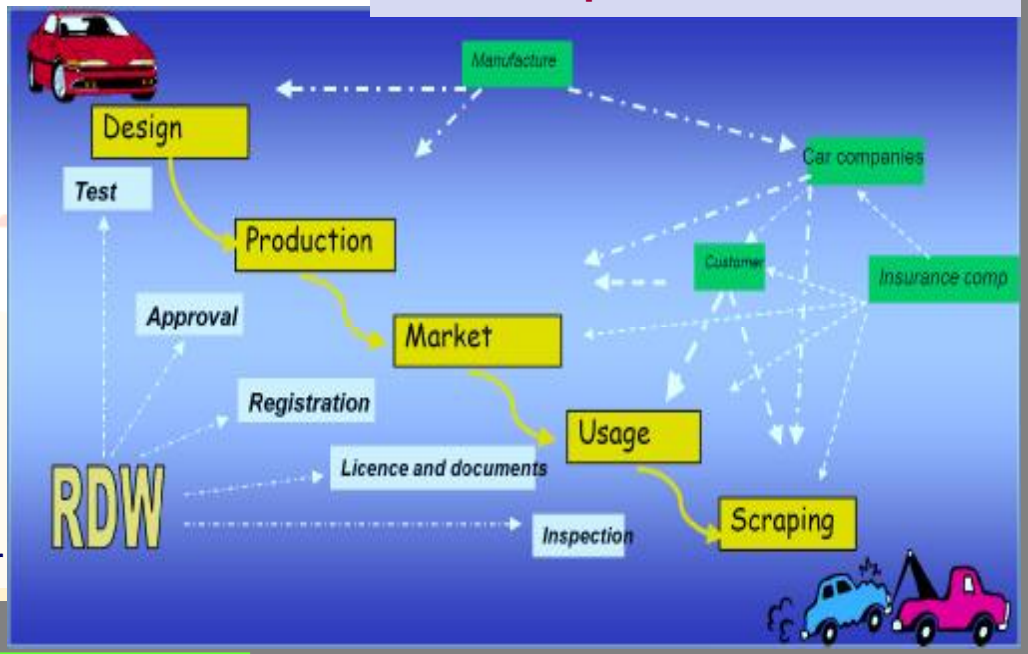
Beispiel aus der Praxis

Der Kunde: RDW (Niederlande)

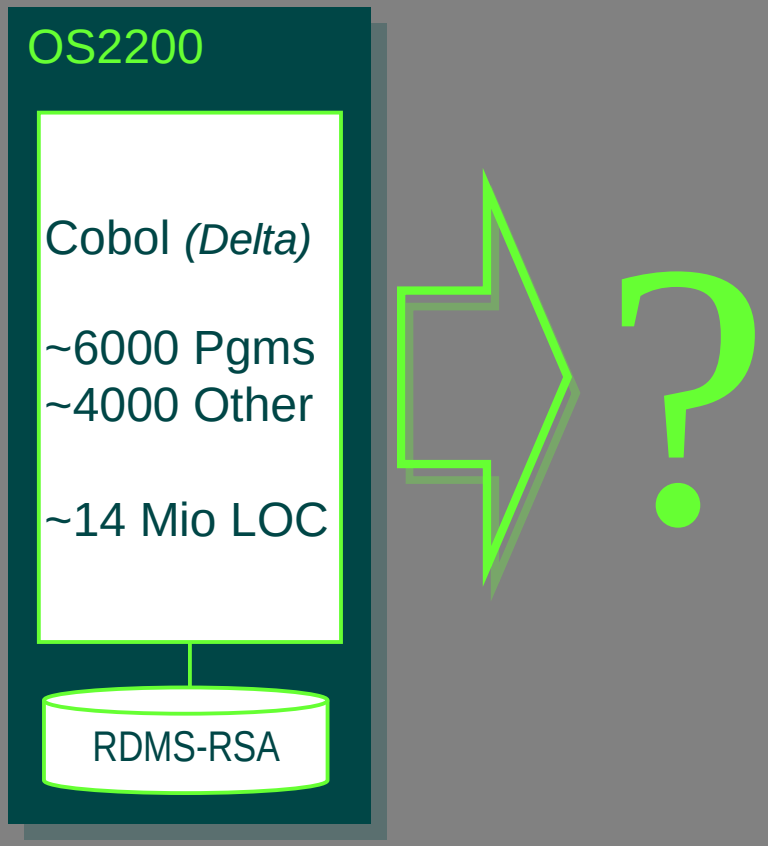
≈ Zulassungsbehörde + TÜV + …

9.600.000	active vehicles
9.700.000	driver licenses
6.000.000	transactions per year
3.500	telephone inquiries/day
7.200.000	APK-inspections/year
750	car inspections/day
23.000	type approval & test reports/year
60.000	exemptions for special transports per year
150.000.000	inquiries by police/year

Combination of **Technical Services Department** of the National Transport Inspectorate and the State Department for **Registration of Motor Vehicles** and Trailers, part of the **Public Works Department**



Die Aufgabe:
Ein einfaches Migrationsprojekt?



Problem

- Unisys OS2200 ist teuer und ohne Zukunft
- Große, kritische Anwendung

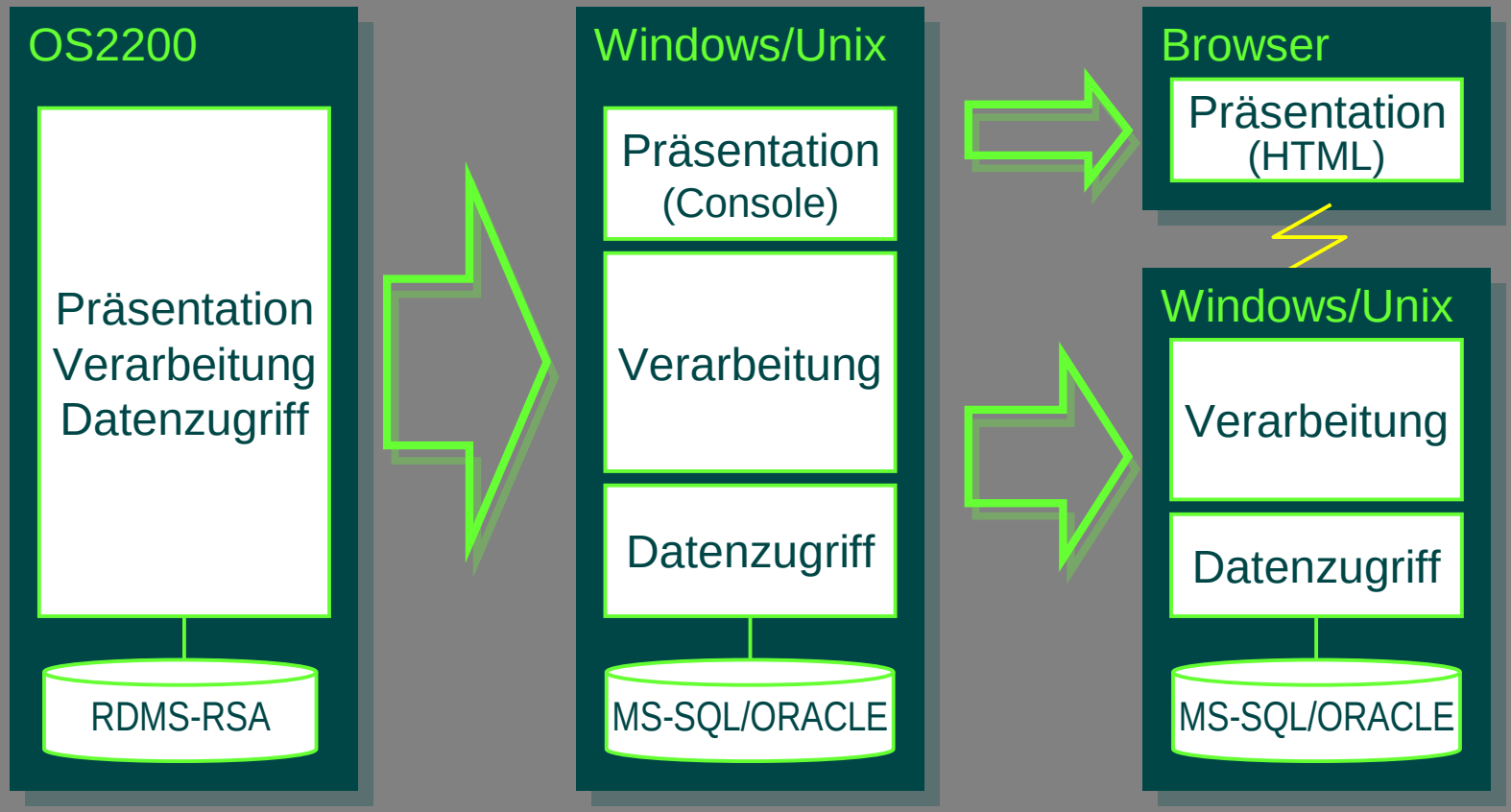
Aufgabe

- Wechsel zu neuer Plattform Windows (evtl. Unix)
- Risiko-Minimierung
- Langfristige Lösung

Ziel&Weg (lesson learned)

- Platform onafhankelijkheid *(Plattformunabhängigkeit)*

"Platform onafhankelijkheid"
= Architektur Transformation



Vorschlag und Auftrag

Unser Vorschlag

- Automatische Transformation
- Modellbasiert
- Plattformspezifische Element in Generatoren gekapselt
- Automationsgrad 80-90%

Forderungen

- Mindestens 95% Automation, besser 98%
- Keine Risiken für die Produktion akzeptabel

Details zur Aufgabe

Unisys OS2200

- 9-Bit Hardware (9 Bit/Byte, 36 Bit/Word)
- **à** Alle numerischen Definition müssen überprüft und ggf. angepasst werden
- **à** Folge: Strukturkonflikte

RDMS-RSA

- “Alte” SQL Call-Schnittstelle, Non-Standard

Monolithischer Code

- DBMS- und Dialog-Operationen nicht getrennt

Vorgehen

DBMS-Deklarationen und –Operationen

- In plattformunabhängige Modelle umwandeln
- Separate, vollständig generierte Persistenzadapter

Screen-Definitionen und Dialog-Operationen

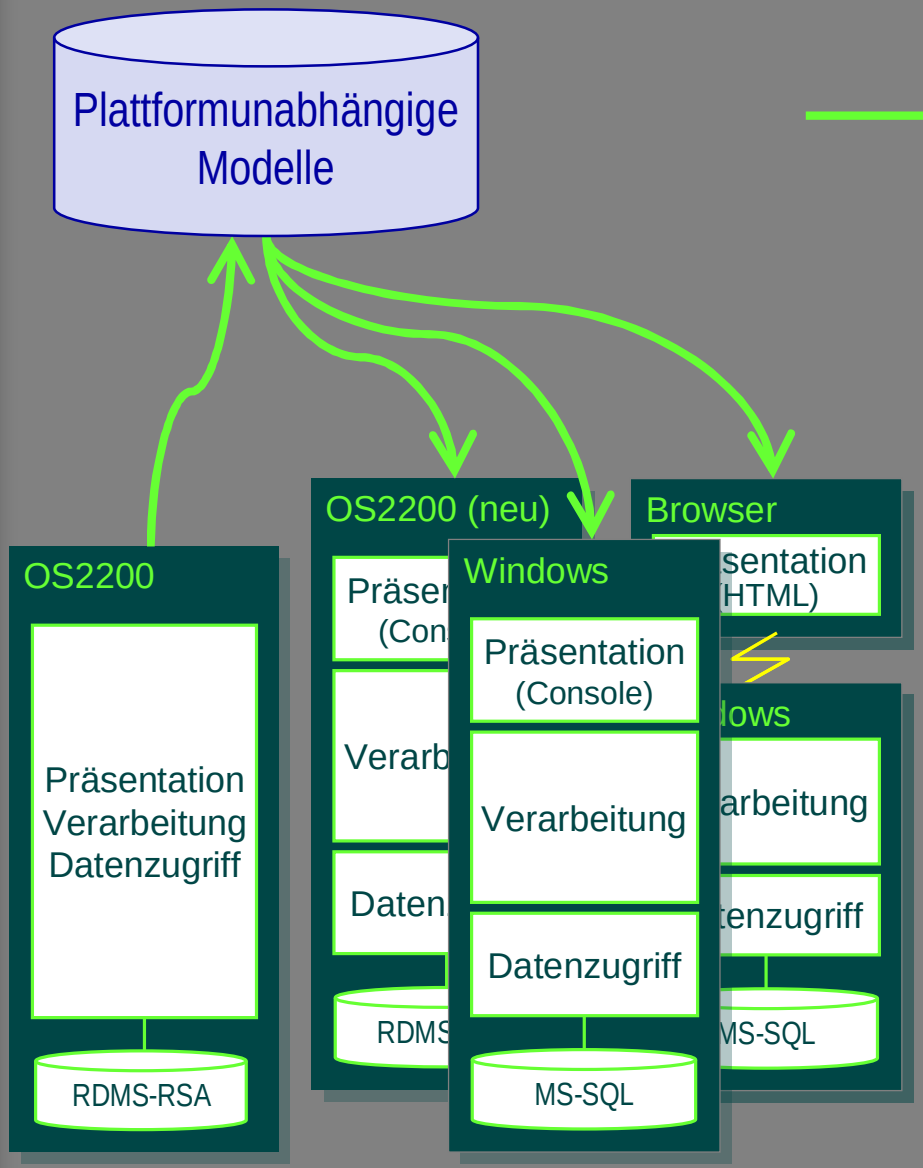
- In plattformunabhängige Definitionen umwandeln
- Separate, vollständig generierte Screenmodule
- Dialog-Operationen in plattformunabhängige „Macros“ gekapselt

Kritische Datendefinitionen

- Angepasst, falls möglich **à** komplexe statische Code-Analyse
- Sonst gekapselt und mit generierten „Fassaden“ versehen

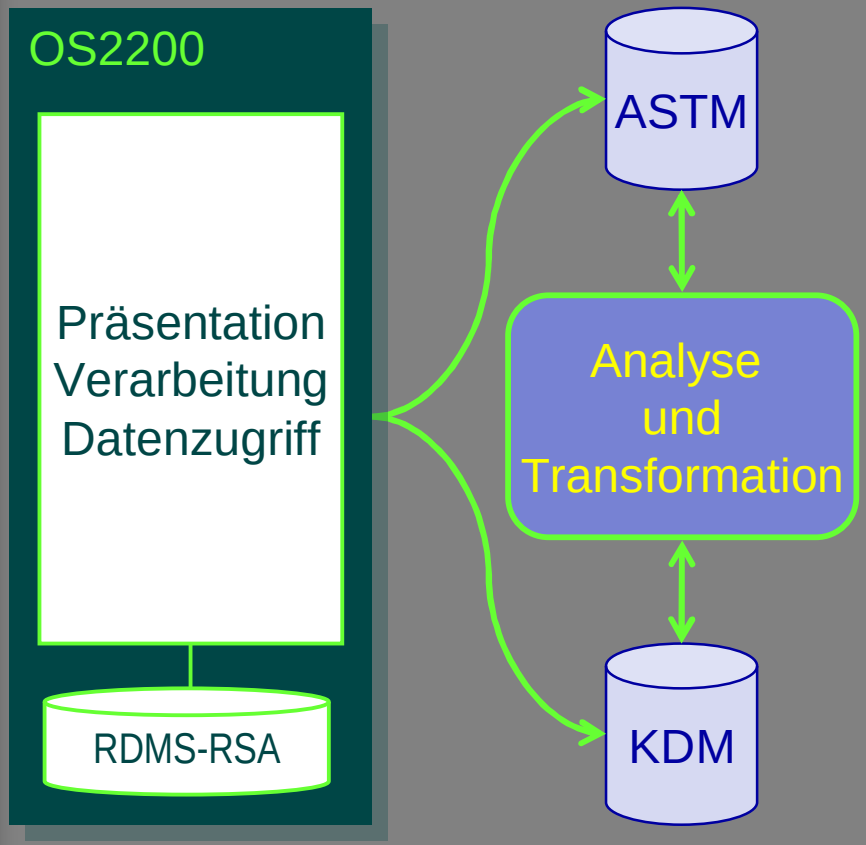
Inline Migration

Inline Migration



1. Die Anwendung wird in eine plattformunabhängige Form überführt
   - Und in dieser Form gewartet
2. Per Generierung, für die alte Plattform
   - „Inline Migration“
3. und für neue Plattformen
   - Ohne weiteren Eingriff
   - „Single Source – Multiple Target“

Prozess in 4 Schritten



1. Schritt

Überführung der Anwendung in neutrale Formate

ASTM* – Abstract Syntax Tree Model

KDM* - Knowledge Discovery Model

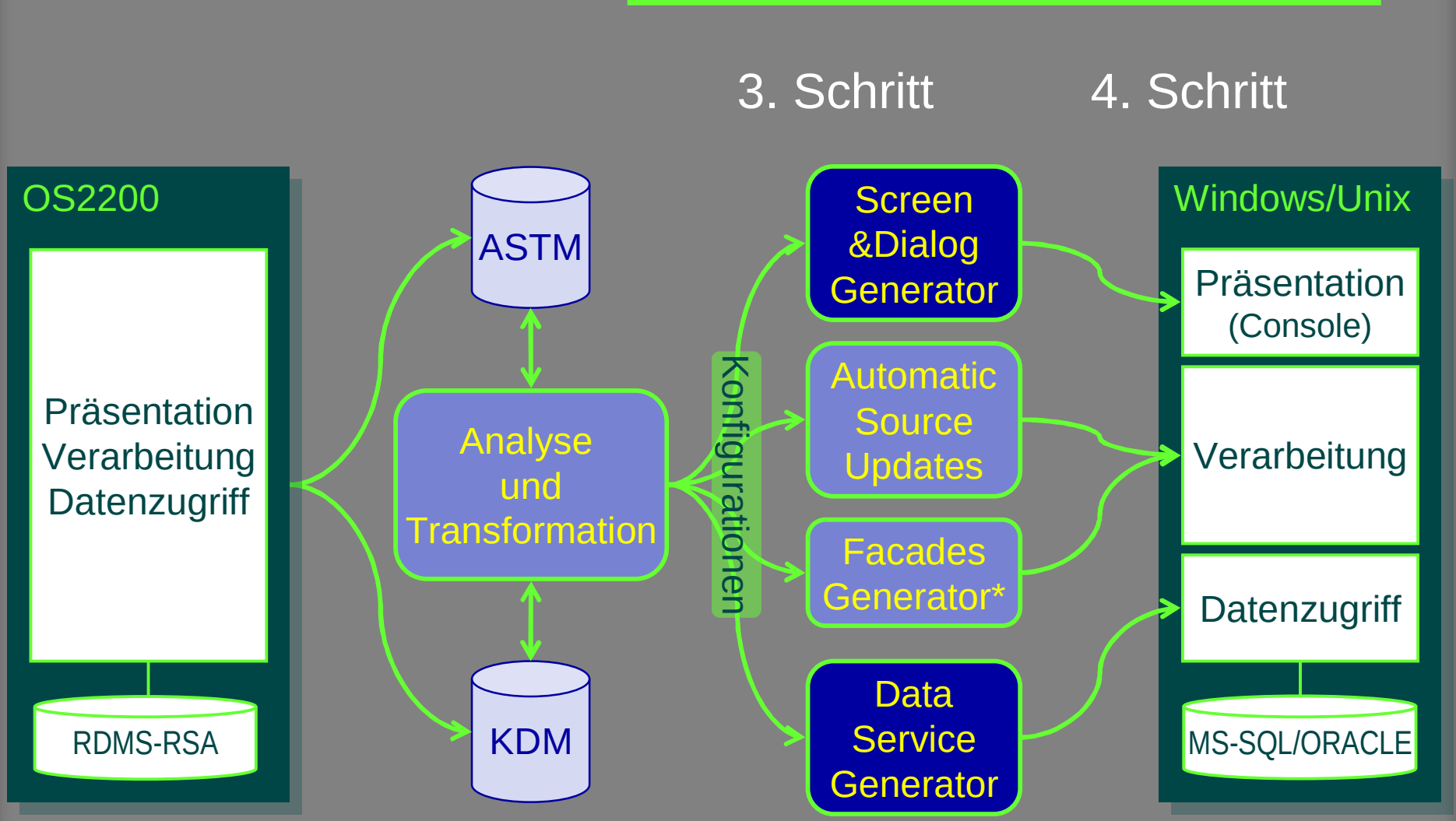
2. Schritt

Analyse und Manipulation in den Modellen

**ASTM und KDM*

Konzepte aus dem OMG Standard ADM (Architecture Driven Modernization)

Prozess in 4 Schritten



DELTA software technology

The Generator Company

Delta-Produkte

Amelio® Transformation Platform

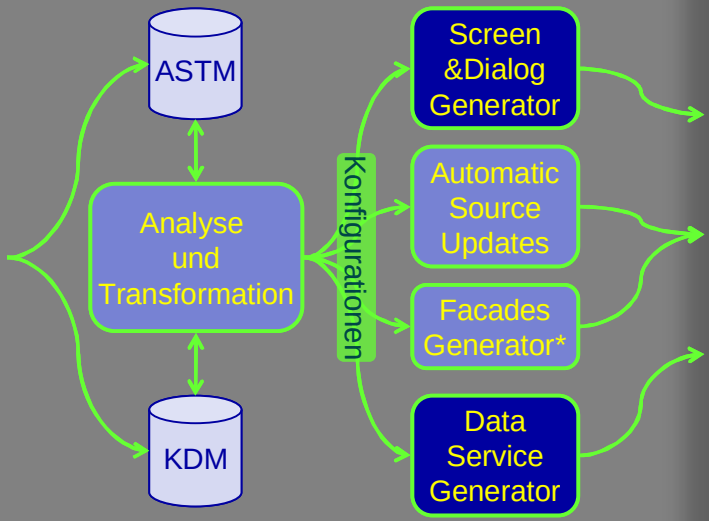
- Basis der Transformationsfabrik
- Discovery, Analyse, Transformation, Updates

SCORE® Data Architecture Integration

- Daten Services und Persistenzadapter modellieren und generieren
- Realisiert mit:

HyperSenses

- Generator-Entwicklungssystem
- Für individuelle Generatoren

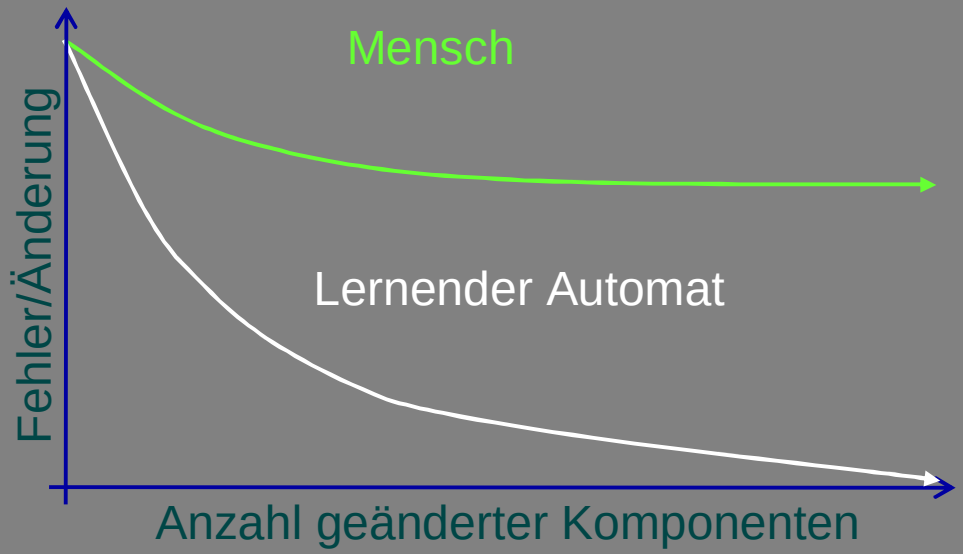


Lerneffekte (1)

Menschen machen manchmal Fehler

Automaten machen den gleichen Fehler immer wieder

Mensch und Automat können aus Fehlern lernen



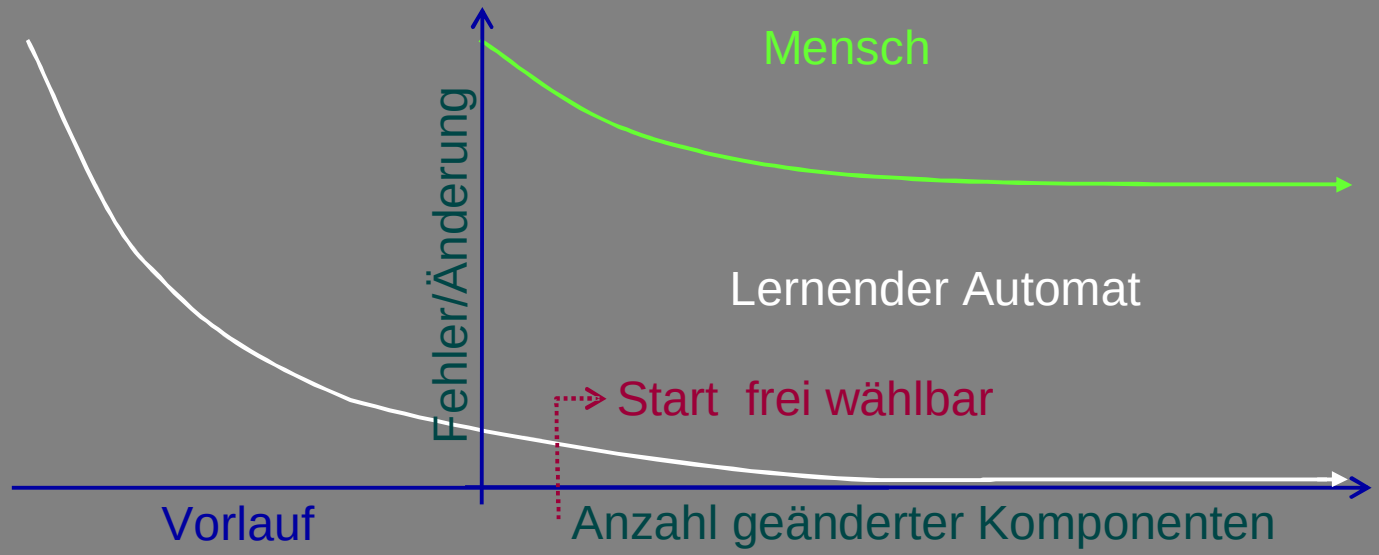
Bei Menschen ist der Lerneffekt geringer, je größer das Team ist. Flüchtigkeitsfehler u.ä. verringern sich nicht

Bei lernenden Automaten wird jeder erkannte Fehler vollständig beseitigt und tritt nie wieder auf

Automaten machen keine Flüchtigkeitsfehler

Lerneffekte (2)

Automatische Prozesse sind beliebig oft, mit wenig Aufwand und schnell wiederholbar



Daher:

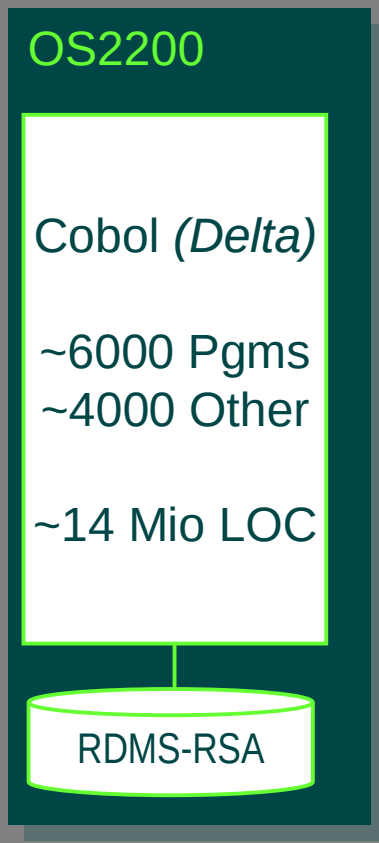
Tests der automatischen Änderungen können beliebig vorgezogen werden

Ohne Einfluss auf laufende Entwicklungen oder Produktion

Eine Fehlerrate von (nahezu) Null ist so erreichbar – bevor die eigentlichen Änderungen beginnen

Ergebnis in Zahlen

Input



Output

~ 10.000 Modifizierte Komponenten

~ 2.500 Persistenzadapter (Daten Services)

~ 1.500 Screen & Dialog Module

~ 1,3 Mio Modifikationen im Quellcode

~ 2,0 Mio Zusätzliche LOC generiert

Fehler bei Test und Einführung gemeldet

30 in modifizierten Komponenten

10 in neuen Komponenten

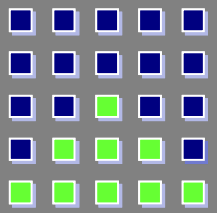
Fehlerrate

30 Fehler / 1,3 Mio Modifikationen

= 0,000024 **à** 0,0024%

10 Fehler / 2.0 Mio “neue” LOC

= 0,000005 **à** 0,0005%

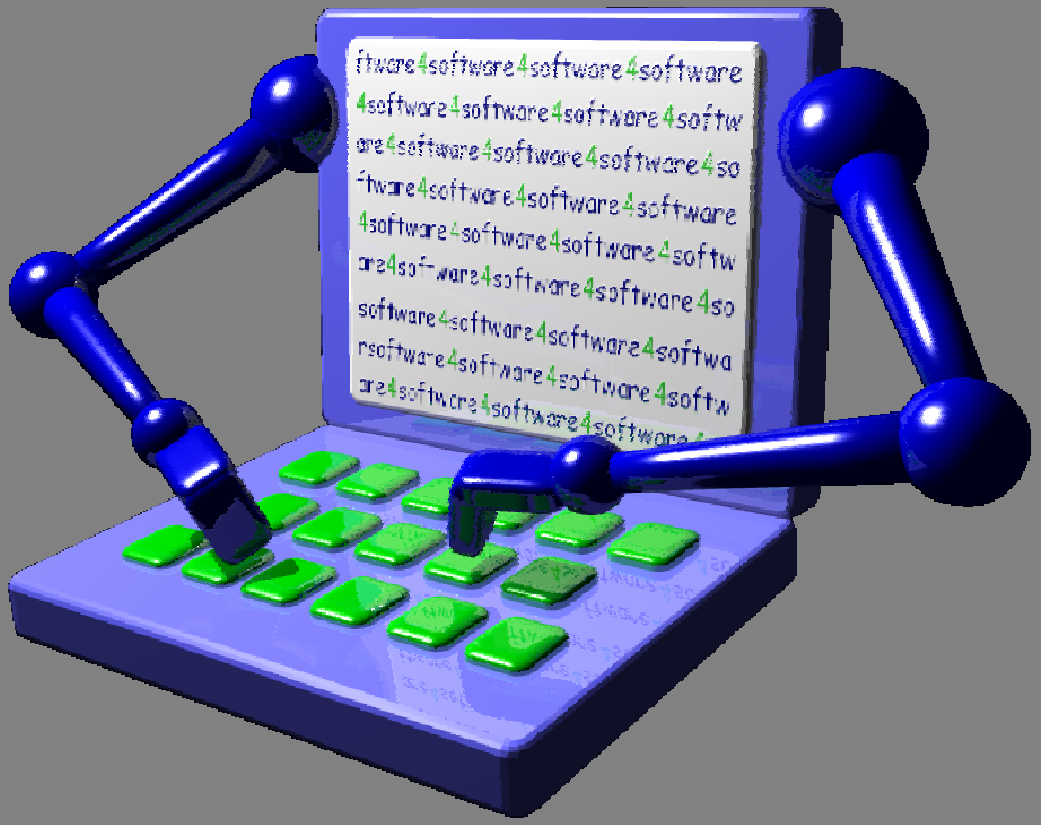


D E L T A
software
technology

The Generator Company

Automatisch und fehlerfrei – eine realistische Vision!

q.e.d.



www.software4software.com