DELTA software technology
The Generator Company

www.D-S-T-G.com

Generative Solutions for the Modernization of your COBOL Applications

Delta Software Technology is a leader in advanced software generators for the integration and modernization of COBOL applications.

The Generator Company

OOP 2006

Scalable Software Systems and Solutions
18.01.2006

Architecture-Driven Modernization
Ziele – Konzepte – Nutzen

Rüdiger Schilling
Delta Software Technology

DELTA software technology
The Generator Company

Credits

- Dieser Vortrag basiert in Teilen auf Tutorien und Workshop-Unterlagen der ADM Taskforce
- Mein besonderer Dank gilt den Kollegen:
  - William M.Ulrich, Tactical Strategy Group
  - Nicolaus Mansurov, KlocWork
  - Philip Newcomb, The Software Revolution

DELTA software technology
The Generator Company

Architecture-Driven Modernization

- Modernisierung?
- Szenarien
- ADM Taskforce
- Standards
- ADM jetzt?

DELTA software technology
The Generator Company

Herausforderungen

Viele Wege führen nach ...?

In a Nutshell...
Current Business Requirements

- Reducing time-to-market for new products and services
- Shifting to a customer-driven philosophy
- Streamlining transaction flow across supply and distribution chains
- Creating flexible information systems to achieve business agility
- Adapting systems to changing business processes

William M. Ulrich – Tactical Strategy Group

Herausforderungen

- Wir erleben immer wieder neue geschäftliche Anforderungen und sich immer rascher ändernde Geschäftsprozesse
- Ebenso tauchen regelmäßig neue Technologien, Architekturkonzepte und Entwicklungsprinzipien auf
- Bei dem Versuch die Anforderungen zu befriedigen, drängen sich zwei Fragen auf:
  1. Welche der neuen Technologien und Konzepte sind am besten geeignet?
  2. Was tun wir mit der vorhandenen Anwendung, kann sie weiter entwickelt werden oder ist sie nur im Wege?
- Legacy – wertvolles Erbe oder Altlast?

Legacy als wertvolles Erbe

CIO Insight – 12/2002

- Was sind die wichtigsten Gründe dafür, die Legacy-Systeme in Ihrem Unternehmen zu erhalten?
  - Sie **unterstützen** immer noch die **Geschäftsprozesse** 54.4%
  - Sie sind immer noch **zuverlässig** 49.7%
  - Das **Personal** für die Unterstützung ist **verfügbar** 44.3%
  - Immer noch **kostengünstiger** als die Alternativen 41.6%
  - Es gibt kein Budget für irgendwelche Änderungen 36.9%
  - Sie **unterstützen** immer noch die **strategischen Ziele** 36.9%

Einerseits ...
Legacy-Systeme sind nach wie vor relevant und zuverlässig

DELTA software technology
The Generator Company

Legacy als Altlast

CIO Insight – 12/2002

- Falls Sie derzeit eine Migration Ihres Legacy-Systems durchführen oder planen, was sind die wichtigsten Gründe?
  - **Neue** strategische **Ziele** 65.2%
  - Das Legacy-System **unterstützt** die Geschäftprozesse **nicht ausreichend** 59.9%
  - Das Legacy-System **unterstützt** die aktuellen **strategischen Ziele nicht** 56.1%
  - **Neue Systeme** sind **kostengünstiger** 48.5%
  - Legacy-Systeme bieten **kaum Interoperabilität** 41.7%

... und andererseits
Es sind Migrationen und Transformation geplant

DELTA software technology
The Generator Company

Investitionen am Beispiel CICS

- Investition in Anwendungen
  - 16 000 Unternehmen weltweit (darunter 490 der Fortune 500 Companies) setzen CICS ein.
  - Es sind 30 Millionen CICS Terminals installiert.
  - Hiermit werden 20 Milliarden Transaktionen/Tag ausgeführt.
  - Mit CICS Transaktionen werden täglich 64 Billionen (10^{12}) $ transferiert oder abgerechnet.
- Eine Überschlagsrechnung mit den folgenden Annahmen:
  - 20 000 S/390 Servers haben durchschnittlich 1 Mill. Zeilen aktiven Anwendungscode (zwischen 200 000 und 50 Millionen pro Server), kumulativ 20 Milliarden LOC.
  - Produktivität von 2 000 LOC/Mannjahr, Investition von 10 Millionen Mannjahren.
  - 100 000 $/Mannjahr, Investition von 1 Billion $ (10^{12}) in S/390 Anwendungssoftware
  - Zum Vergleich, das USA 1999 GNP war 9 Billion $.

Prof. Wilhelm Spruth, Tübingen

DELTA software technology
The Generator Company

Cobol Legacy - Was die Auguren sagen

- Gartner Group:
  - 75% of all business data is processed in COBOL.
  - 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.
  - Existing legacy systems are predominantly written in COBOL.
  - 15% of all new applications (5 billion lines) through 2005 will be in COBOL.
  - "Integration with Legacies" is the number one concern of IT managers in 2003.
  - Over the next four years there will be a 13% decrease in their number [of COBOL developers] due to retirement and death.
- The Cobol Report:
  - CICS transaction volume (such as COBOL-based ATM transactions) grew from 20 billion per day in 1998 to 30 billion per day in 2002.
- Tactical Strategy Group:
  - Replacement costs for COBOL systems, estimated at $25 per line, are in the hundreds of billions of dollars.
- Giga Group:
  - There are over 90,000 COBOL programmers in North America in 2002.
  - The most highly paid programmers in the next ten years are going to be COBOL programmers who know the Internet.

DELTA software technology
The Generator Company

Widersprüche

- Einerseits:
  - Gehen laut CIO Insight Umfrage 48,5% der befragten CIOs davon aus, dass neue Systeme kostengünstiger sind
- Andererseits findet sich im Informatik Spektrum, April 2003:

> *Der Aufwand für Wartung und ständige Anpassung an sich ändernde Unternehmensbelange ist erheblich, mit einer überraschenden Beobachtung: Die Wartungskosten für Cobol-Programme sind erheblich geringer als für alle anderen Programme. Die Jahr-2000-Umstellungs-Kosten pro Function Point betrugen im Durchschnitt für alle Sprachen 45$; für Cobol-Programme nur 28$. Cobol ist aus diesem Grund auch für Neu-Anwendungen häufig erste Wahl. So wächst die existierende Menge an Cobol-Programmen, rund 180 Milliarden Code-Zeilen, jährlich um 5 Milliarden Code-Zeilen.*

- Was heißt das?
  - Werden neue Systeme wieder mit COBOL entwickelt?

DELTA software technology
The Generator Company

Probleme mit der Qualität

- Ariane 5 Absturz – durch Registerüberlauf
- Ausfall des Londoner Ambulanz-Systems bei Einführung
- Ausfall Leitrechner Hamburger Bahnhof bei Neueröffnung (1995)
- Hartz-IV-Software (2004)
- Deutsche Bahn – Komplettausfall des Buchungssystems durch Update (2005)

DELTA software technology
The Generator Company

Wie werden wir den Herausforderungen gerecht?

- Die üblichen Verdächtigen:
- Ansatz 1: Alles neu entwickeln
  - Mit neuer Architektur, neuer Technik, neuen Mitarbeitern
  - Wir machen jetzt alles mit ...
- Ansatz 2: Enterprise Application Integration
  - Broker, Hub and Spoke
  - Middleware als Königsweg
- Ansatz 3: Software-Pakete
  - „Standard“-Software statt eigener Entwicklung
  - Wir machen jetzt alles mit ... SAP

Ansatz 1: Alles neu entwickeln

- Kosten und Risiken werden fast immer unterschätzt
- 85% der IT-Projekte: zu spät oder gar nicht fertig*
- Nur 9% der IT-Projekte in Termin und Budget*
- ERP-Projekte: Jahre für die Implementierung,
  35% abgebrochen und selten vollständig eingesetzt*

  * Standish Group International
- 16.5 Milliarden $ werden jährlich ausgegeben für Systeme, die nie in Produktion gehen (Information Week)
- **Markt und Anwender fordern neue Funktionen – nicht neue Software**

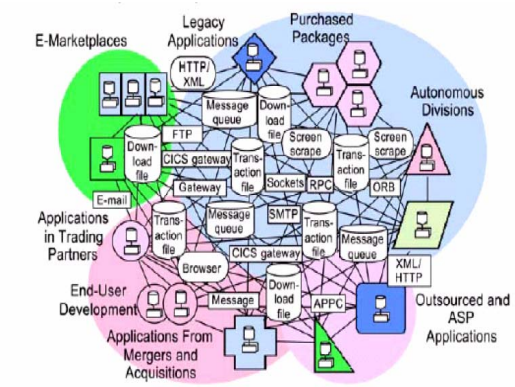
Wir machen jetzt alles mit ...

- Neue Technologien (Sprache, Middleware etc.)?
  - OMG EAI Workshop 2001:
    All efforts to replace N technologies by 1 new technology usually end up with N plus 1 technologies

    Greatest advantage of new technologies are usually that their disadvantages are not known yet
- Das SOA-Paradox
  - Ziel: Schnellere Befriedigung von Geschäftserfordernissen
  - Weg: Erst einmal die ganze Entwicklung blockieren, um neue Techniken einzuführen und die vorhandenen Anwendungen noch einmal zu realisieren

DELTA software technology
The Generator Company

Ansatz 2: Enterprise Application Integration?



Gartner Forecast Newsletter, 2001

- Herstellergetriebene Integrationslösungen erzeugen immer neue Legacy-Schichten
- EAI Architekturen erhöhen die Komplexität von Anwendungs-systemen offensichtlich ...
- ... und können am Ende ins Chaos führen

DELTA software technology
The Generator Company

Aber diese Ansätze sind zu kurz gegriffen

- Vollständige Ablösung der Anwendungen:
  - Zu teuer, zu zeitaufwändig und zu riskant
- Enterprise Application Integration
  - Löst das Problem der eigentlichen Architekturbeschränkungen nicht
- Standard Software* – kein Allheilmittel:
  - 35% der Projekte werden vorzeitig abgebrochen
  - Typischerweise liefern sie nur 40% der versprochenen Funktionalität
  - Nur 10% der Projekte in Termin und Budget

  **serverworldmagazine, Juni 2001*
- Outsourcing: Verschiebt die Probleme, ohne sie zu lösen

Die Alternative

Modernization Alternative – *Takes Over Where Existing IT Strategies Fall Short*

- Augments or displaces existing IT options
- Enables upgrade, conversion, consolidation, migration and related projects
- Applies a measured, phased approach to meeting IT challenges
- Utilizes scenarios to ensure a business-driven approach
- Builds upon solid foundation of production systems

William M. Ulrich – Tactical Strategy Group

Weil die üblichen Ansätze nicht reichen,

ist die Alternative „Modernisierung“ das einzig Sinnvolle.

Was aber ist Modernisierung?

Architecture-Driven Modernization

- ☑ Modernisierung?
- ➢ Szenarien
- ADM Taskforce
- Standards
- ADM jetzt?

DELTA software technology
The Generator Company

Modernisierung: Drei Basisaktivitäten

- Beurteilung
  - Analyse und Offenlegung der System- und Geschäftsartefakte, Architekturen, Daten und Prozessabläufe, Systemstrukturen und Verhaltensweisen
- Stabilisierung und Standardisierung
  - Aufgaben, die strukturieren, rationalisieren, neu orientieren, modularisieren oder auf eine andere Art und Weise bestehende Systeme umgestalten
- Transformation
  - Extraktion von Datendefinitionen, Daten und Geschäftsregeln, zusammen mit der Wiederverwendung der vorhandenen Systemartefakte beim Aufbau neuer Zielarchitekturen

http://adm.omg.org/

DELTA software technology
The Generator Company

Szenario I
Management des Anwendungsportfolios

- Ziel
  - Bestehende Anwendungen verstehen und dokumentieren
- Situation
  - Keine oder unzureichende Dokumentation der Anwendungssysteme und ihrer Architektur
  - Detaillierte Informationen werden benötigt zur Erfüllung von Revisionsanforderungen und gesetzlichen Bestimmungen (z.B. Sarbanes-Oxley, Basel II)
  - IT-Systeme sollen größeren Änderungen unterzogen oder outgesourcet werden
  - Modernisierungsprojekte brauchen Basisinformationen für Planung und Strategie

http://adm.omg.org/

Szenario II
Verbesserung von Anwendungen

- Ziel
  - Die Robustheit, Integrität, Qualität, Konsistenz und/oder Performance von Anwendungen soll erhöht werden
- Situation
  - Zunehmender Rückstau von Updates wegen Qualitätsproblemen
  - Erhöhte Fehlerraten und Probleme mit der Zuverlässigkeit
  - Unzufriedene Benutzer wegen langer Entwicklungszeiten
  - Zu hohe Kosten für die Systempflege
  - Als Vorbereitung für ein Outsourcing oder die Rückintegration
  - Punktuelle Änderungen, z.B. Feldlängen (Y2K, Kontonr. etc.)

Szenario III
Sprachkonvertierung

- Ziel
  - Übersetzung von einer Sprache in eine andere
  - Kann auch Sprachversionen und „Dialekte" betreffen
- Situation
  - Eine Sprache ist obsolete, nicht mehr vom Hersteller unterstützt oder wird von den verfügbaren Entwicklern nicht mehr verstanden
  - Neue geschäftliche Anforderungen können mit der alten Sprache nicht mehr befriedigt werden
  - Ein System muss auf eine neue Plattform gebracht werden, auf der eine Sprache oder Sprachversion nicht mehr verfügbar ist
  - Eine Basis soll geschaffen werden, um die aktuelle Anwendung in eine neue strategische Architektur migrieren zu können

Szenario IV
Plattform-Migration

- Ziel
  - Modifikation und Anpassung von Anwendungen, so dass sie auf neuen Hardware/Software-Plattformen genutzt werden können
- Situation
  - Die Hardware und/oder das Betriebssystem wird nicht mehr (ausreichend) unterstützt
  - „Right Sizing“ indem ein System auf eine verteilte Umgebung oder zurück zum Mainframe gebracht wird
  - Die aktuelle Plattform unterstützt das akzeptierte Betriebssystem nicht
  - Ein Plattformwechsel wird nötig, um zu einer gemeinsamen Plattform mehrerer Systeme zu kommen

Szenario V
Nicht-invasive Anwendungsintegration

- Ziel
  - Im Wesentlichen geht es um die Möglichkeit, Legacy-Anwendungen mit modernen Frontends zu versehen*
  - * *Diese Definition beschränkt sich auf die Benutzerschnittstellen, tatsächlich gibt es einen fließenden Übergang z.B. zum Service-Enablement (SOA-Transformation), das u.U. ebenfalls nicht-invasiv oder minimal-invasiv realisiert werden kann*
- Situation
  - Die Endbenutzer möchten die alternden Frontends durch Web-basierte ersetzen
  - Funktionalität, Datenstrukturen und Schnittstellen des Kernsystems sollen im Prinzip unverändert bleiben
  - Die Integration soll der erste Schritt für z.B. die folgende SOA-Transformation sein

Szenario VI
Service-orientierte Architekturtransformation

- Ziel
  - Restrukturierung vorhandener Anwendungen
- Situation
  - Geschäftsfunktionen eingebettet in monolithischen Batch- und Online-Anwendungen sollen als Services zur Verfügung gestellt werden
  - Komplexe Benutzerschnittstellen und Datenzugriffe komplizieren die Isolation der Geschäftsfunktionen
  - Bestehende Frontends basieren auf segmentierten, inkonsistenten und redundanten Backend-Systemen oder auf Batch-Updates

Szenario VII
Migration von Datenarchitekturen

- Ziel
  - Migration nicht-relationaler Datei- und DB-Systeme zu relationalen Datenarchitekturen
- Situation
  - Es gibt zunehmend Probleme mit der Konsistenz, Integrität und Erreichbarkeit der Daten
  - Vorhandene ISAM-, hierarchische und Netzwerkstrukturen eignen sich nicht für verteilte, moderne Systeme
  - Die Benutzer benötigen flexiblere Sichten auf die Geschäftsdaten
  - Ältere Datei- und Datenbanksysteme sind technisch überholt und werden nicht mehr unterstützt

Szenario VIII
Architekturkonsolidierung

- Ziel
  - Redundante Anwendungssysteme sollen zusammengeführt und integriert werden
- Situation
  - Nach einer Fusion oder Firmenübernahme gibt es redundante Anwendungen, die zusammengeführt werden müssen
  - Redundante Geschäftsbereiche sollen zu einer Einheit zusammengefasst werden
  - Mehrere Systeme enthalten große Segmente überlappender Funktionen
  - Unzureichende oder fehlende gemeinsame Datennutzung verschiedener Systeme vermindern die Verfügbarkeit und führen zu inkonsistenten Ergebnissen

Szenario IX
Einsatz von Data Warehouses

- Ziel
  - Data-Warehouses enthalten Daten für Extraktion und Analyse, basieren auf einem gemeinsamen Repository – kein Update
- Situation
  - Geschäftsfunktionen benötigen einen konsolidierten Zugriff auf bestimmte Daten (z.B. Kundeninformationen)
  - Benutzer benötigen Daten über Organisations- und Anwendungs-grenzen hinweg
  - Die entsprechenden Daten sind verteilt über mehrere Systeme, was Zugriff und Auswertung erschwert
  - Zeit oder Budget reichen nicht für eine „richtige“ Integration und Konsolidierung der Datensysteme

DELTA software technology
The Generator Company

Szenario X
Auswahl und Einsatz von Anwendungspaketen

- Ziel
  - Unterstützung von Analyse, Auswahl und Einsatz von Anwendungspaketen
- Situation
  - Es soll die Möglichkeit des Einsatzes eines Anwendungspakets untersucht werden
  - Ein Anwendungspaket ist bereits gekauft und soll angepasst werden
  - Eine weitergehende Dokumentation eines Pakets wird gebraucht
  - Es soll untersucht werden, wie ein Paket mit einem existierenden System verbunden oder integriert werden kann

DELTA software technology
The Generator Company

Szenario XI
Wiederverwendbare Software-Assets

- Ziel
  - Wiederverwendbare Anwendungsfunktionen identifizieren und aufbereiten
- Situation
  - Ein Unternehmen hat den Nutzen und Wert der Wiederverwendung und der komponentenbasierten Entwicklung erkannt
  - Es gibt eine umfangreiche Anwendungsbasis, die wertvolle Funktionen enthält, die in wiederverwendbare Bausteine und Komponenten überführt werden sollen

DELTA software technology
The Generator Company

Szenario XII
Modellgetriebene Architekturtransformation

- Ziel
  - Überführung eines nicht-modellbasierten Anwendungssystems in Richtung einer modellgetriebenen Architektur (MDA)
- Situation
  - Ein Unternehmen hat sich darauf festgelegt, Anwendungen modellbasiert zu entwickeln und zu pflegen
  - MDA-Konzepte und -Werkzeuge sollen eingeführt werden bzw. sind bereits im Einsatz
  - Die aktuelle Daten- und Anwendungsarchitektur ist überholt
  - Benutzer benötigen Funktionen, die in der gegenwärtigen Architektur nur schwer eingeführt werden können

DELTA software technology
The Generator Company

Architecture-Driven Modernization

- ☑ Modernisierung?
- ☑ Szenarien
- ➢ ADM Taskforce
- Standards
- ADM jetzt?

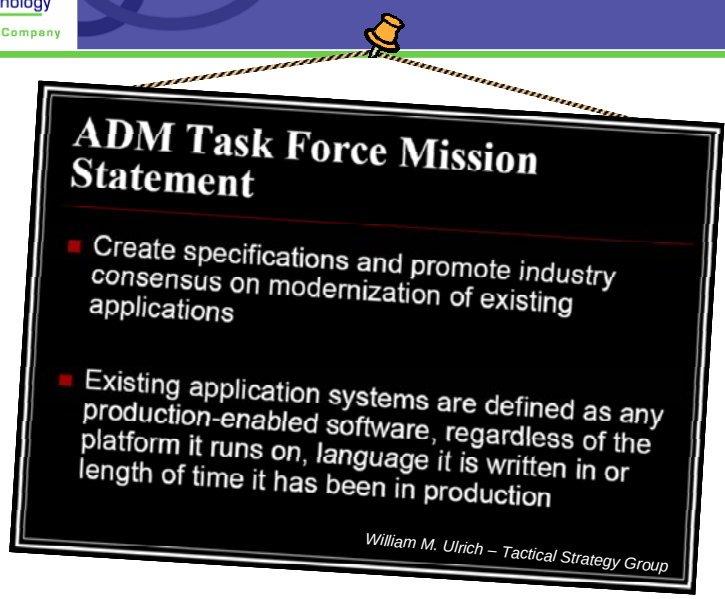
DELTA software technology
The Generator Company

Wikipedia: Die Abkürzung ADM steht für …

- … **Atomic Demolition Means**, Kernwaffen mit geringer Sprengkraft
- … **Algorithmische Diskrete Mathematik**
- … **Arbeitskre** **Markt-** und Sozialforschungsinstitute
- … **Archivo D** sina
- … **Arbeitsg**
- … eine Def **isner**, siehe ADM-M
- … **Atmospheric Dynamics Missio** der ESA, siehe ADM-Aeolus
- … die Amtrack-Abkürzung für den Bahnhof von **Ardmore** (Oklahoma)

Architecture Driven Modernization
→
Neue OMG Standards, die MDA-Verfahren und -Standards auf existierende Systeme ausweiten

DELTA software technology
The Generator Company



DELTA software technology
The Generator Company

OMG-Standards

- OMG Task-Force “Architecture-Driven Modernization”
  - ADMTF, gegründet 2003
- Die ersten beiden Standards:
  - Knowledge Discovery Meta-model (KDM)
  - Abstract Syntax Tree Meta-model (ASTM)
- Die Roadmap skizziert sieben Abschnitte der Standardentwicklung
- Wichtige Unternehmen der Branche nehmen teil:
  - IBM, EDS, Fujitsu, HP, Unisys ...
  - Delta Software Technology

DELTA software technology
The Generator Company

ADM Roadmap (1)

- 1. Knowledge Discovery Meta-Model Package (KDM)
  - Führt ein erstes Meta-Modell ein
  - Erlaubt Werkzeugen für die Modernisierung Anwendungs-Metadaten auszutauschen, quer über Anwendungen, Sprachen und Plattformen
  - Bietet eine umfassende Übersicht von Anwendungsstrukturen und -daten, aber nicht unterhalb der prozeduralen Ebene (s. ASTM)
- 2. Abstract Syntax Tree Meta-Model Package (ASTM)
  - Ergänzt die Software-Darstellung unterhalb der prozeduralen Ebene
  - Erlaubt die vollständige Repräsentation der Anwendungen
  - Unterstützt den Austausch granularer Metadaten für die Übersetzung auf der Sprachebene
  - Vereinigt alle syntaktischen Sprachmodelle in einem gemeinsamen Metamodell

DELTA software technology
The Generator Company

ADM Roadmap (2)

- 3. Analysis Package (AP) – (Initialisierung ‘05)
  - Unterstützt die Auswertung der strukturellen Metadaten (KDM, ASTM), um detaillierte Verhaltens- und Struktur-Metadaten zu gewinnen
  - Z.B. Design Pattern, Business Rules oder andere Aspekte, die nicht offensichtliche Bestandteile eines Systems sind, sondern semantisch abgeleitet werden müssen/können
- 4. Metrics Package (MP) – (Initialisierung ‘06)
  - Ableitung von Metriken, die technische, funktionale und architektonische Eigenschaften beschreiben
  - Zur Unterstützung von Planung, Aufwandsschätzung, ROI-Analyse und Risikobewertung

DELTA software technology
The Generator Company

ADM Roadmap (3)

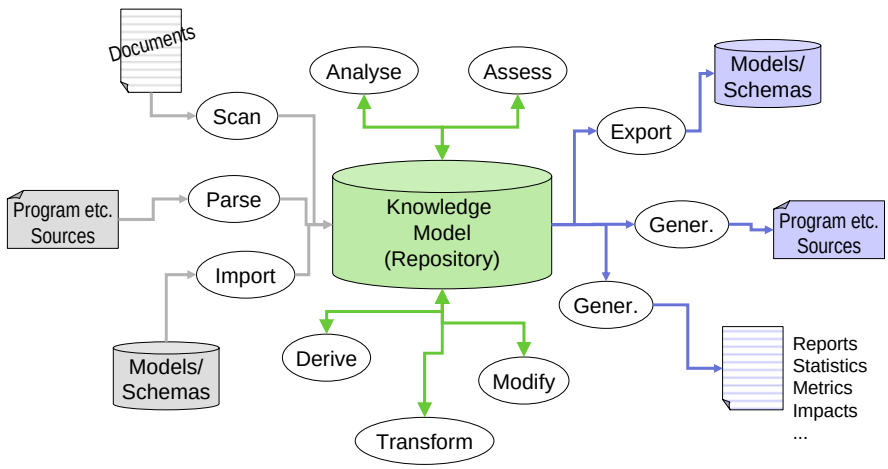
- 5. Visualization Package (VP) – (Initialisierung ‘07)
  - Zur (graphischen) Visualisierung der verschiedenen Aspekte der ADM-Modelle
  - Z.B Graphiken, Diagramme oder Metriktabellen
- 6. Refactoring Package – (Initialisierung ‘08)
  - Beschreibt Wege wie ADM-Modelle genutzt werden zur (werkzeuggestützten) Überarbeitung von Anwendungssystemen
- 7. Target Mapping & Transformation (TMT) Package – (‘09)
  - Beschreibt Abbildungen zwischen ADM-Modellen und Ziel-Modellen, z.B. UML
  - Von ADM zu MDA

Architecture-Driven Modernization

- ☑ Modernisierung?
- ☑ Szenarien
- ☑ ADM Taskforce
- ➢ Standards
- ADM jetzt?

Knowledge Discovery Meta-Model (KDM)

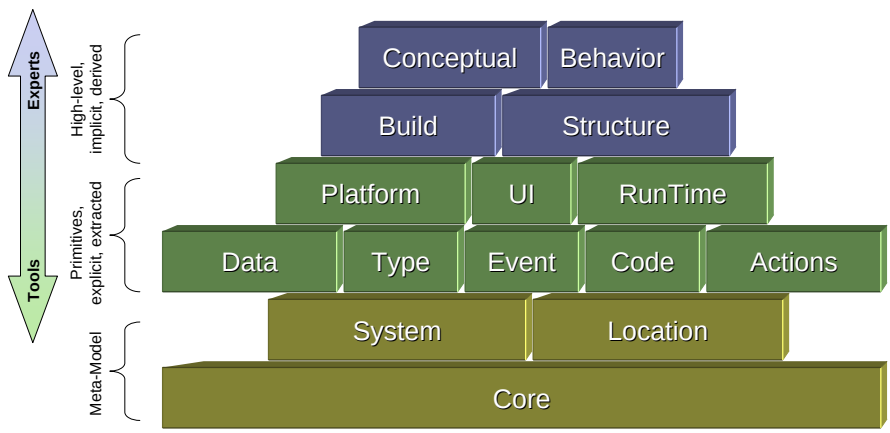
Ein Modell als Basis aller Modernisierungs-Aktivitäten



Knowledge Discovery Meta-Model (KDM)

- Jedes der Modernisierungs-Szenarien benötigt, erzeugt oder verändert Daten der betroffenen Komponenten
  - Welche Daten im Einzelnen gebraucht werden ist unterschiedlich
  - Grundsätzlich könnten ganz unterschiedliche Werkzeuge die Daten und in variabler Reihenfolge verwenden
- Zweck des Standards ist ein gemeinsames Metamodell für diese Daten zu definieren, wodurch insbesondere die Interoperabilität der Werkzeuge verschiedener Hersteller gewährleistet werden soll
  - Innerhalb der OMG werden Metamodelle auf der Basis von MOF (Meta Object Facility) spezifiziert
  - MOF ist der Standard zur Definition von Modellen und Metamodellen sowie für die Implementierung der entsprechenden Interfaces

Bausteine des KDM-Standards



Core Package

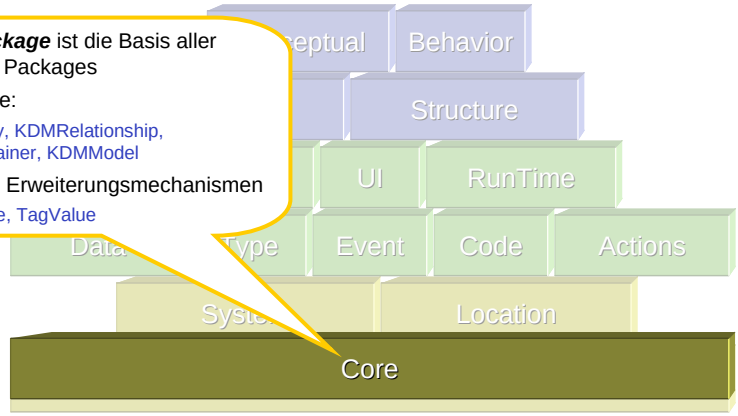
Das ***Core Package*** ist die Basis aller anderen KDM Packages

Meta-Konzepte:

KDMEntity, KDMRelationship, KDMContainer, KDMModel

„Light-Weight“ Erweiterungsmechanismen

Stereotype, TagValue



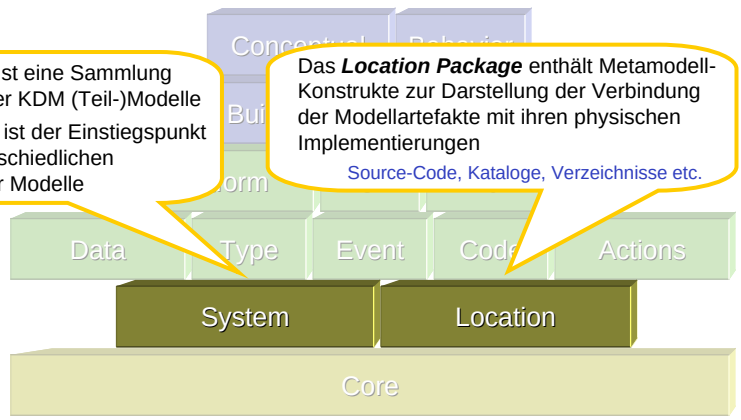
System & Location Package

Ein ***System*** ist eine Sammlung verschiedener KDM (Teil-)Modelle

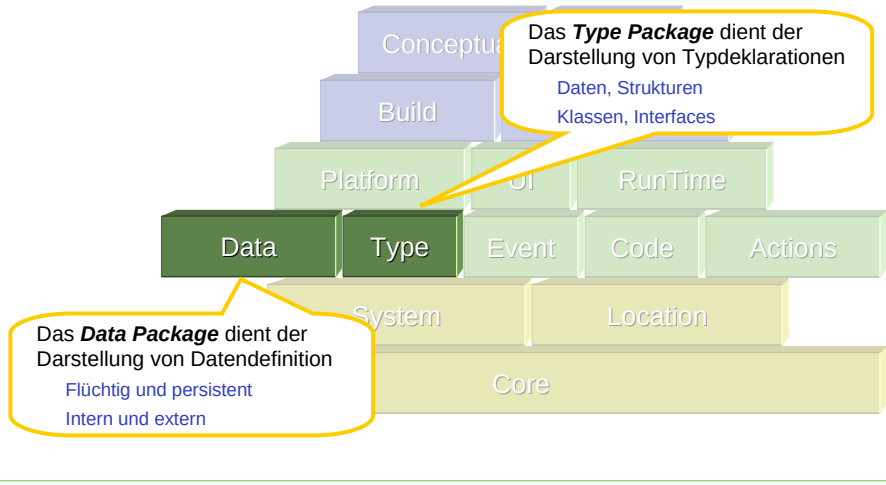
Das ***System*** ist der Einstiegspunkt zu den unterschiedlichen Aspekten der Modelle

Das ***Location Package*** enthält Metamodell-Konstrukte zur Darstellung der Verbindung der Modellartefakte mit ihren physischen Implementierungen

Source-Code, Kataloge, Verzeichnisse etc.



Data & Type Package



Das ***Type Package*** dient der Darstellung von Typdeklarationen

- Daten, Strukturen
- Klassen, Interfaces

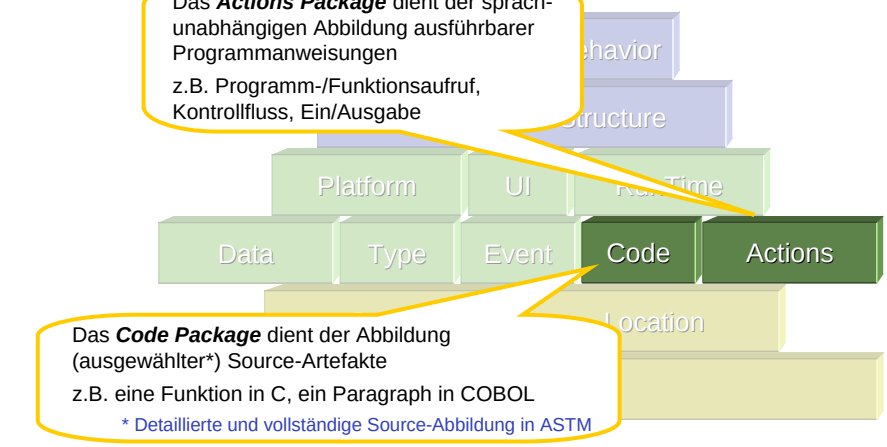
Das ***Data Package*** dient der Darstellung von Datendefinition

- Flüchtig und persistent
- Intern und extern

Code & Actions Package

Das ***Actions Package*** dient der sprach-unabhängigen Abbildung ausführbarer Programmanweisungen

z.B. Programm-/Funktionsaufruf, Kontrollfluss, Ein/Ausgabe



Das ***Code Package*** dient der Abbildung (ausgewählter*) Source-Artefakte

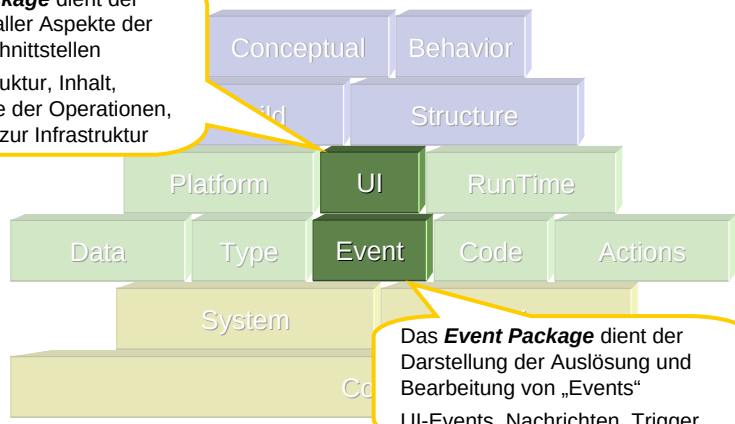
z.B. eine Funktion in C, ein Paragraph in COBOL

* Detaillierte und vollständige Source-Abbildung in ASTM

Event & UI Package

Das ***UI Package*** dient der Abbildung aller Aspekte der Benutzerschnittstellen

Aufbau, Struktur, Inhalt, Reihenfolge der Operationen, Beziehung zur Infrastruktur



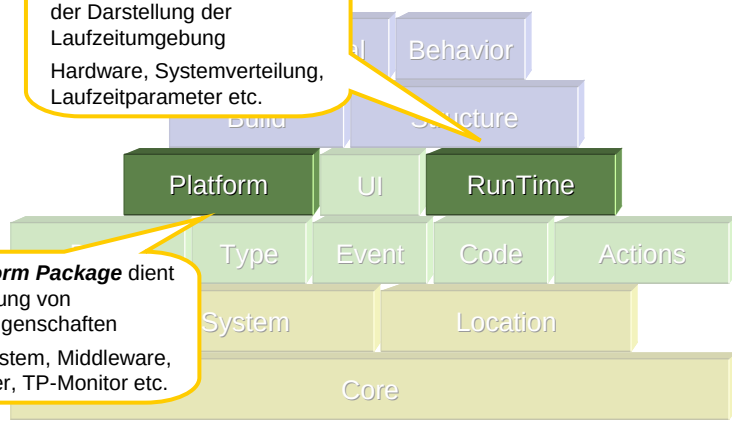
Das ***Event Package*** dient der Darstellung der Auslösung und Bearbeitung von „Events“

UI-Events, Nachrichten, Trigger

Platform & RunTime Package

Das ***RunTime Package*** dient der Darstellung der Laufzeitumgebung

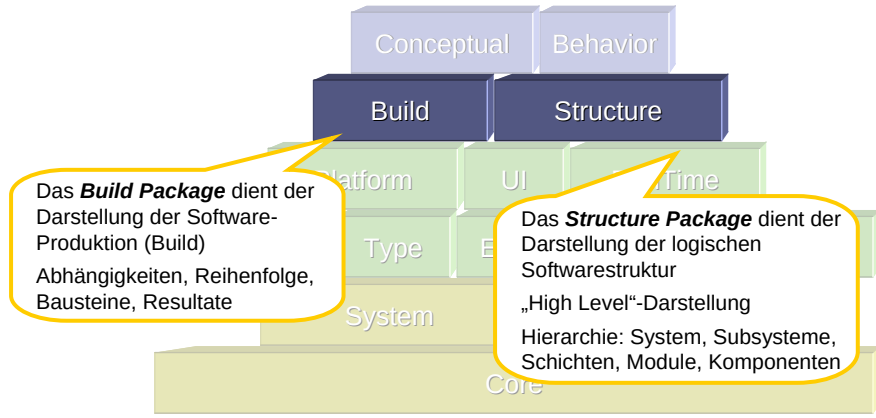
Hardware, Systemverteilung, Laufzeitparameter etc.



Das ***Platform Package*** dient der Abbildung von Plattformeigenschaften

Betriebssystem, Middleware, Appl.Server, TP-Monitor etc.

Build & Structure Package



Das ***Build Package*** dient der Darstellung der Software-Produktion (Build)

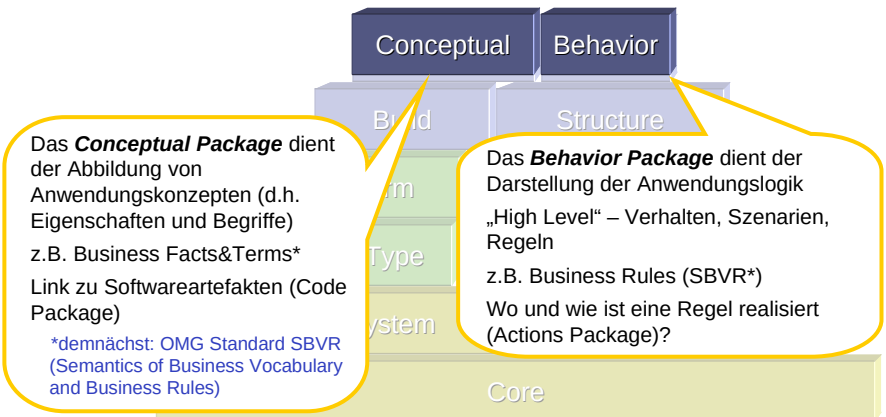
Abhängigkeiten, Reihenfolge, Bausteine, Resultate

Das ***Structure Package*** dient der Darstellung der logischen Softwarestruktur

„High Level"-Darstellung

Hierarchie: System, Subsysteme, Schichten, Module, Komponenten

Conceptual & Behavior Package



Das ***Conceptual Package*** dient der Abbildung von Anwendungskonzepten (d.h. Eigenschaften und Begriffe)

z.B. Business Facts&Terms*

Link zu Softwareartefakten (Code Package)

*demnächst: OMG Standard SBVR (Semantics of Business Vocabulary and Business Rules)

Das ***Behavior Package*** dient der Darstellung der Anwendungslogik

„High Level" – Verhalten, Szenarien, Regeln

z.B. Business Rules (SBVR*)

Wo und wie ist eine Regel realisiert (Actions Package)?

DELTA software technology
The Generator Company

Abstract Syntax Tree Meta-Model (ASTM)

- Es stellen sich folgende Fragen:
  - Woher kommen die Informationen über Programme (und ähnliche Artefakte) im KDM?
  - Wie werden die Details des Programm-Codes dargestellt?
  - Wie können Programme mit unterschiedlicher Syntax gemeinsam analysiert werden?
  - Wie können Programme einer Syntax in eine andere übersetzt werden?
- Um diese und andere Fragen beantworten zu können, wird eine geeignete, abstrakte Repräsentation benötigt
  - Das sind üblicherweise *Abstract Syntax Trees*
  - Davon gibt es schon viele verschiedene, z.B. in Compilern
  - Damit aber die o.g. Fragen beantwortet werden können, wird ein gemeinsames Metamodell benötigt:
    - *Abstract Syntax Tree Meta-Model*

DELTA software technology
The Generator Company

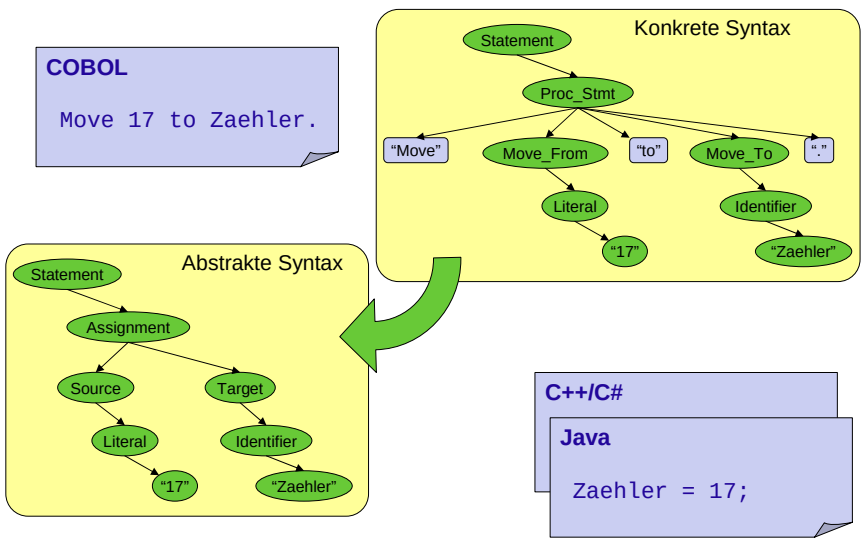
Was ist ein Abstract Syntax Tree?

- Ein AST ist ...
- ... eine formale Darstellung der Syntaxstruktur von Software
- ... besser geeignet für formale Analysen als die konkrete Syntax (*Surface Syntax*)
- ... ein präziserer Formalismus für eine detaillierte Informationsgewinnung als weniger formale Techniken (Scanner, Tokenizer, visuelle Untersuchung)

Was kann ein Abstract Syntax Tree?

- Ein AST kann …
- ... die Generierung (bzw. Rekonstruktion) konkreter Syntax von (Legacy-)Systemen aus der abstrakten Syntax erlauben
- ... angereichert werden durch Analyseergebnisse (Verknüpfung, Datenfluss, Kontrollfluss etc.)
- ... dazu benutzt werden, Software-Metriken und Dokumentationen zu gewinnen
- ... übersetzt werden in andere AST-Modelle, mittels geeigneter Transformationsregeln
- ... ausgewertet und manipuliert werden mit Sprachen zur Modellabfrage und -manipulation (z.B. OMGs QVT™, TSRIs JTGEN™/JRGen™ und Deltas ANGIE)

Source-Code und AST



GASTM, SASTM

- SASTM ➔ Specific AST Metamodell ➔ PSM (MDA)
  - Beispielsweise COBOL, C#, Java
- GASTM ➔ Generic AST Metamodell ➔ PIM (MDA)
  - Neutraler, sprachunabhängiger AST
- Transformation (z.B. C++ ⇔ Java)
  - C++-AST ⇔ Java-AST ☹
  - C++-AST ⇔ GAST ⇔ Java-AST ☺
  - MDA: PSM ⇔ PIM ⇔ PSM

ASTM – Der Standard

- Leider ist die OMG-Arbeitsgruppe (ADMTF) etwas hinter dem Zeitplan
- Es gibt noch keinen Standard und noch keinen endgültigen gemeinsamen Vorschlag
- Es gibt drei verschiedene Einreichungen, die zusammengefasst werden sollen:
  - Diskretes Modell – The Software Revolution (TSRI)
  - Kontinuierliches Modell – TCS Consulting
  - Interface-Modul Modell – Klocwork
- Ein gemeinsamer, überarbeiteter Vorschlag (joint revised submission) ist bis zum 24.1.2006 geplant

DELTA software technology
The Generator Company

Diskretes und Kontinuierliches ASTM

Diskretes ASTM (TSRI)	Kontinuierliches ASTM (TCS)
Das GASTM ist eine umfassende Untermenge aller gemeinsamen Sprachelemente (vieler Sprachen)	
Jedes SASTM basiert auf einer eigenen Menge von Sprachelementen für eine spezifische Sprachsyntax	Jedes SASTM ist eine Erweiterung oder Spezialisierung des GASTM
SASTM/GASTM-Abbildungsregeln stellen die funktionale Äquivalenz her	SASTM in Kombination mit GASTM komplettieren das AST-Modell für jede Sprache
Analysen werden vollständig auf der GASTM-Basis ausgeführt	Analysen werden auf der Basis der Summe von GASTM und SASTM ausgeführt
Mehrsprachige und sprachneutrale Abbildungen, Analysen und Transformationen mittels GASTM und SASTM/GASTM-Abbildung	Mehrsprachige und sprachneutrale Abbildungen, Analysen und Transformationen mittels Vereinigungsmodell (GASTM + beteiligte SASTMs)

DELTA software technology
The Generator Company

ASTM Interface Modell – Klocwork

- Interface-Definitionen statt spezifischer ASTM
  - Es wird unterstellt, dass diverse ASTs bereits existieren (z.B. in Compilern)
  - Über die Interfaces wird eine gemeinsame Sicht hergestellt
  - GASTM und SASTM sind somit virtuell
- Architektur ähnlich KDM
  - Core Package (Basis-Konstrukte)
  - Beschränkte Untermenge für GASTM
    - Scope, Statement, Expression, Identifier Package (Explizit)
    - Define-Use, Control, Data Flow (Implizit)
  - “Light-Weight“ Erweiterungsmechanismen

Architecture-Driven Modernization

- ☑ Modernisierung?
- ☑ Szenarien
- ☑ ADM Taskforce
- ☑ Standards
- ➢ ADM jetzt?

ADM-Prinzipien

- Nachdem an den Standards noch gearbeitet wird, kann es keine praktische Erfahrung damit geben
  - Aber, es gibt fundierte Erfahrungen mit den wesentlichen ADM-Prinzipien

1. Modellbasiert
   - Alle Informationen über die zu bearbeitenden Artefakte sind in einheitlichen Modellen hinterlegt; das gilt ebenso für abgeleitete Erkenntnisse
   - MOF und XML (XMI) als konzeptionelle und technische Grundlagen vermindern die Abhängigkeit von Werkzeugen und erlauben auch „fremde“ Produkte zu nutzen
   - Die Definition von Metamodellen macht den Einsatz von automatischen Discovery- und Analyse-Werkzeugen erst richtig sinnvoll

ADM-Prinzipien

2. Abstraktion und Plattformunabhängigkeit
   - Primärquellen dienen nur als Ausgangspunkt für die Population der Modelle, die allen weiteren Operationen als Basis dienen
   - Wo immer möglich, werden die gewonnenen Erkenntnisse abstrahiert und plattformunabhängig dargestellt
3. PSM-PIM-PSM-Transformation
   - In all den Fällen, bei denen im Rahmen der Modernisierung Software erzeugt oder modifiziert werden soll (Migration, Transformation, Integration), hat sich dieses Verfahren bewährt
   - Die mit diesem Verfahren verbundene deutliche Reduktion der Komplexität erhöht die Transparenz, führt zu weniger Fehlern und im Ergebnis zu höherer Effizienz
   - Nebenbei sind Plattformentscheidungen weniger kritisch, es wird eher möglich, die Wahl der endgültigen Plattform noch zu ändern

ADM-Prinzipien

4. Zielgerichtet und selektiv
   - „Mehr hilft mehr“ gilt sicher nicht für eine effektive Unterstützung von Modernisierungsmaßnahmen
   - Die Erfassung und Analyse der Metadaten muss so zielgerichtet wie möglich sein
5. Adaptive „Fabrik“
   - Modernisierungsaufgaben sind so vielfältig, dass es die eine Eierlegende-Wollmilchsau-Lösung gar nicht geben kann
   - Ebenso wenig kann es für jede einzelne Konstellation ein spezielles Werkzeug geben
   - Benötigt werden Werkzeugkomponenten, die mit einem gemeinsamen Modell als Basis, zu einer individuellen Modernisierungsfabrik montiert werden können

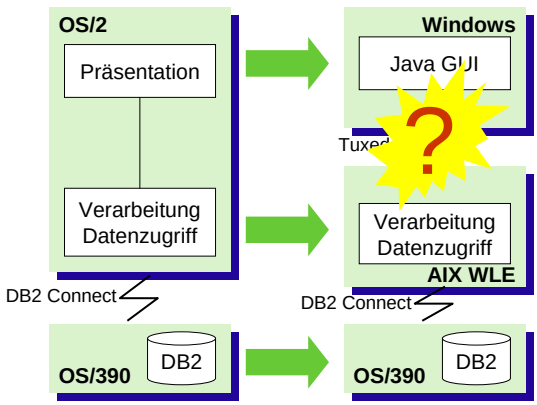
Modernisierung in der Praxis I

- Das erste Beispiel ist schon seit drei Jahren abgeschlossen, aber es enthält bereits wesentliche Elemente der ADM-Prinzipien
  - Es ist **modellbasiert**, wenn auch nicht KDM oder ASTM – die gab es damals noch nicht
  - Das verwendete Modell basiert auf **MOF** und **XML**
  - Das Prinzip der **Plattformunabhängigkeit** im Modell findet sich ebenfalls
  - Ebenso das **PSM-PIM-PSM** Prinzip
  - Das Prinzip der **Selektivität** gilt hier besonders, weil in diesem Fall sich fast alles auf die Schnittstellen und die Kommunikation konzentriert

Modernisierung in der Praxis I

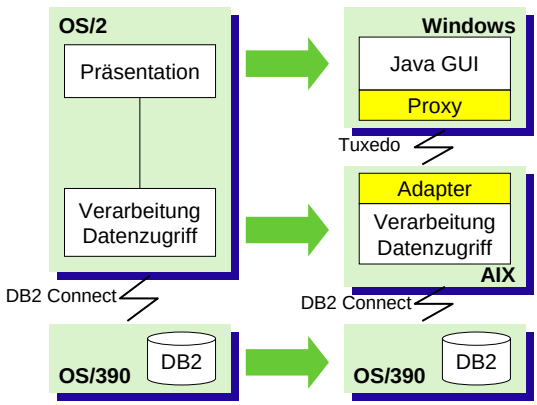
- Aufgabe ähnlich Szenarien IV (Plattform-Migration) und V (Nicht-invasive Anwendungsintegration)
- Ausgangspunkt:
  - COBOL-Anwendung mit Fat-Clients (MF Dialog System) auf der obsoleten Plattform OS/2
  - 3600 Programmmodule
  - 410 Screens
  - 5600 Copybooks (→ Schnittstellendefinitionen)
  - > 2000 Terminals
- Ziel
  - Echte 3-Tier-Architektur mit Thin-Clients (Java)
  - Application-Server auf Unix (BEA WebLogic Enterprise)
  - Kommunikation via Tuxedo

Vom Architektur-Mismatch ...



- Datenformate
- Schnittstellen
  - Klassen/Methoden vs. Call-Schnittstelle
- Granularität
  - COBOL-Strukturen > 120KB
- Transaktionslogik
- Client-Framework
- . . .

... zur adaptiven Architektur



- Proxy
  - Echte Java-Klassen und -Formate
  - Passend zum Framework
- Adapter
  - Mapping Call-Schnittstelle ⇔ Operationen/Parameter
  - State Management
  - Transaktionsanpassung
- Testtreiber, Log-Daten
- . . .

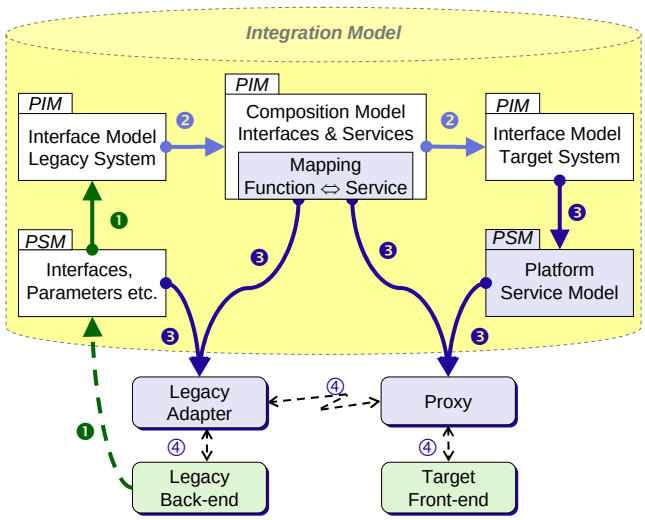
DELTA software technology
The Generator Company

Integration mit Dreisprung

- Discovery
  - Import der Programmschnittstellen, deren Parameterstrukturen in diesem Fall als COBOL-Copybooks vorlagen
  - Automatische Transformation in eine plattformunabhängige Modelldarstellung – MOF- und XML-basiert – **analog KDM**
  - Weitere Informationen wurden in diesem Fall nicht benötigt (**Selektivität**)
- Composition
  - Definition des Zielmodells: Interfaces, Operationen, Parameter
  - Definition des Mappings: Altes Modell ⇔ Neues Modell
  - ➔*Teilweise automatisch, der Rest mit Software-Unterstützung, deklarativ und plattformneutral*
- Production
  - Vollständige Generierung der Adapter und Proxies (COBOL und Java)
  - Vollständige Generierung der Mappingroutinen (COBOL)
  - Vollständige Generierung der Kommunikationsschicht (Tuxedo)
  - Einige wenige manuelle Eingriffe in den alten Modulen

DELTA software technology
The Generator Company

PSM→PIM→PSM für die Integration



Integration eines Legacy-Systems mit einer neuen Ziel-Plattform

❶ *Discovery*
i.e. Extract from Source

❷ *Composition*

❸ *Production*
i.e. Transformation with Generators

④ *Runtime*

Das Ergebnis

- Die Anwendung ist mittlerweile längst in Produktion und erfüllt alle Erwartungen
- Der Kostenrahmen und Termin des Gesamtprojekts wurden eingehalten ...
- ... obwohl die manuelle Umstellung der UIs (Java) sich als viel aufwendiger als geplant erwies
- Das wurde kompensiert durch automatisierte Transformation und Integration der COBOL-Module mit den SCORE®-Werkzeugen
- ***Laut Kunde ca. 5 mal schneller als eine manuelle Lösung***

Modernisierung in der Praxis II

- Das zweite Beispiel ist ganz aktuell, die Vorbereitung lief parallel zur Entwicklung der ersten ADM-Standards
- Es finden sich daher nicht nur die ADM-Prinzipien wieder, sondern auch vorläufige Implementierungen der Standards KDM und ASTM
- Hohe Anforderungen an Qualität und Sicherheit

  zusammen mit einigen technischen Besonderheiten

  erfordern eine ganz individuelle Modernisierungsfabrik

Modernisierung in der Praxis II

- Aufgabe Szenarios II (Verbesserung von Anwendungen), III (Sprachkonvertierung) und IV (Plattform-Migration)
  - Häufig anzutreffende Kombination bei größerem Plattform-Wechsel
- Ausgangspunkt:
  - COBOL-Anwendungen mit Online- und Batchprogrammen auf Unisys OS2200 mit RDMS (Call-Interface nicht ESQL)
  - Delta ADS-Generator (Batch), aber ohne Delta-Online und -Datenbankunterstützung, was Migration leichter gemacht hätte
  - 1500 Online-Programme
  - 3500 Batch-Programme
  - 1500 Screens
  - 800 Copybooks und Macros

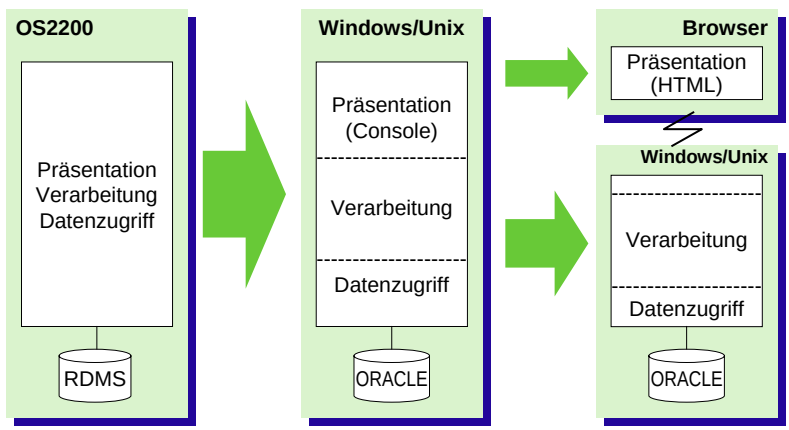
Modernisierung in der Praxis II

- Ziele – strategisch:
  - Kostenreduktion in der Produktion
  - Lösung von bisheriger (und zukünftiger) Herstellerbindung durch Plattformunabhängigkeit!
- Ziele – technisch:
  - Vollständige Ablösung der Unisys 2200-Plattform
  - Zielplattform Windows oder Unix mit ORACLE o.ä.
    - Die endgültige Plattform steht am Anfang noch nicht fest!
  - Ersatz aller OS2200-spezifischen Konstrukte (COBOL-Dialekt, DBMS, Screen)
  - Aufzeigen einer Web-Perspektive

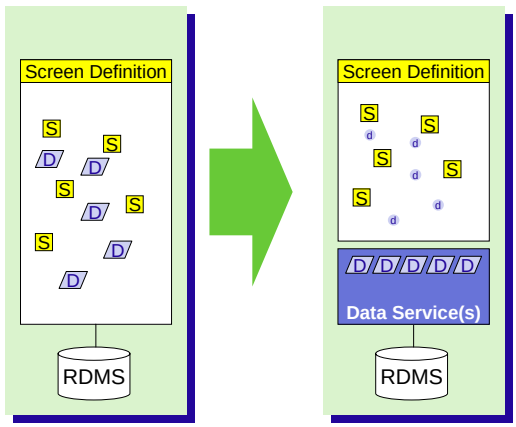
Modernisierung in der Praxis II

- Vorgaben für das Projekt:
  - Risikokontrolle und Risikominimierung – Unterbrechungen der Produktion oder Fehlfunktionen sind nicht tolerabel
  - Non-Blocking – Entwicklung und Wartung können nur für kurze Zeit und immer nur für Teile der Anwendungen unterbrochen und eingefroren werden
  - Automation, Reproduzierbarkeit und Rückverfolgbarkeit (Traceability)
  - Die Effizienz der Software-Entwicklung – heute unterstützt durch perfektionierte SE-Prozesse – darf nicht beeinträchtigt werden

Das Ziel



Auslagern des Datenzugriffs



S Screen/Dialog-Funktion

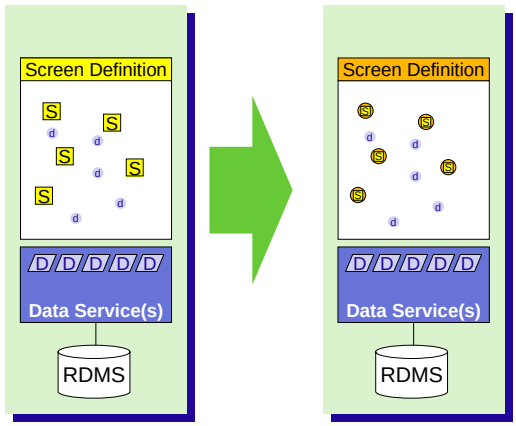
D DBMS-Funktion

d Data-Service-Aufruf

- Data-Service-Komponenten
  - Sammeln und Kombinieren von DBMS-Funktionen
  - Auslagern in separate *Data-Service*-Komponente(n)
  - DBMS-Funktion durch Service-Aufruf ersetzen
- Ziel
  - Isolation der Datenfunktionen
  - Erleichterung von Migration, Test und Verifikation
  - Notwendige Provisorien für den Übergang werden nicht verewigt

➔ Voraussetzung für *In-Place Migration*

Einkapseln der Benutzerschnittstelle



S UI-Funktion

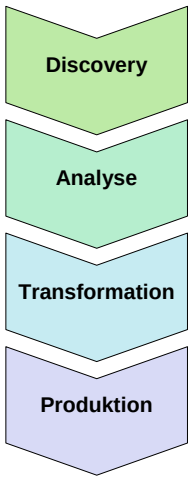
d Data-Service Aufruf

Ⓢ UI-Kapsel

- UI-Kapseln
  - UI-Funktionen können nicht ausgelagert werden, ohne Anwendungsfunktionen zu ändern
  - Sie werden in Macroaufrufe gekapselt (plattformunabhängig)
- Ziel
  - Isolation der UI-Funktionen
  - ähnlich DBMS-Funktionen

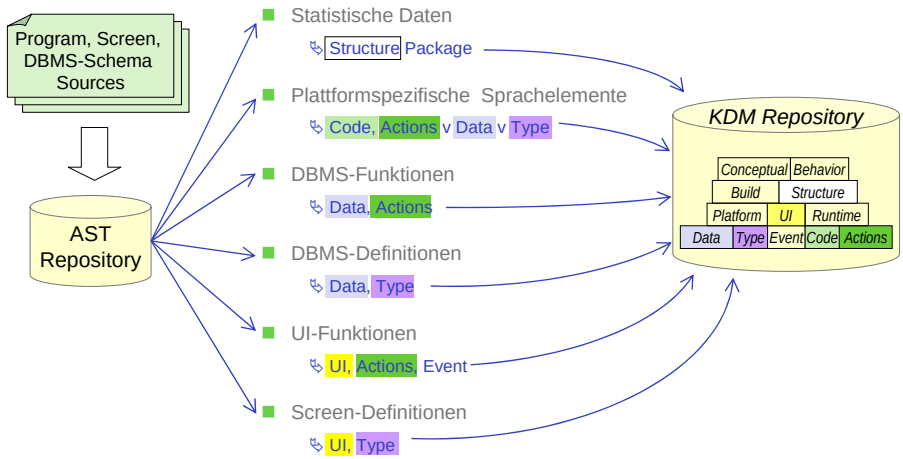
➔ Voraussetzung für *In-Place Migration*

Die Produktionsstraße der Modernisierungsfabrik



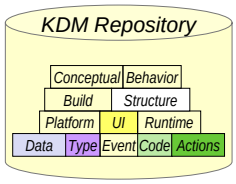
- Discovery
  - Importieren der Sourcen in ein AST-Repository
  - Übernehmen der relevanten Daten in das KDM-Repository
- Analyse
  - Statistische Auswertungen und Metriken
  - Analyse der Transformations-Kandidaten
- Transformation
  - Übersetzung plattformspezifischer Konstrukte in plattformunabhängige
- Produktion
  - Generierung der Ziel-Artefakte (Sourcen etc.)

Bestücken des KDM-Repository



DELTA software technology
The Generator Company

KDM-Analyse und -Transformation

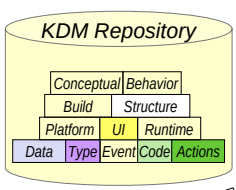


- Analyse plattformspezifischer Sprachelemente
  - *In: Code, In/Out:Actions v Data v Type*
- Analyse, Gruppierung und Transformation* von DBMS-Definitionen und -Funktionen
  - *In: Data, Type, Actions, Out: Data, Actions*
- Analyse, Gruppierung und Transformation* von Screen-Definitionen und UI-Funktionen
  - *In: Type, In/Out: UI, Actions, Event*
- Analyse- und Transformationsregeln werden zum Teil erst im Laufe des Projektes festgelegt bzw. weiterentwickelt

** Transformation: Übersetzung plattformspezifischer Konstrukte in plattformunabhängige*

DELTA software technology
The Generator Company

Produktion



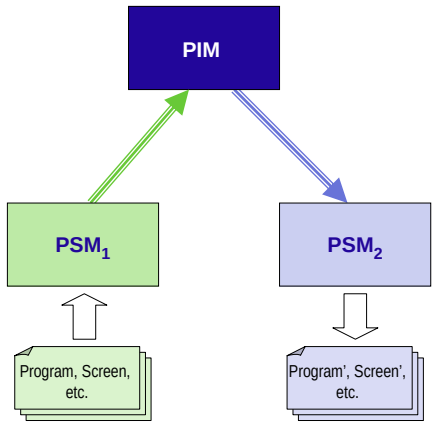
- Dokumentation von Analyse und Transformation
- Die Analyseergebnisse werden in die ASTs übertragen bzw. neue ASTs gebildet
- Aus KDM und ASTs werden modifizierte und neue Komponenten erzeugt (generiert)

Protokolle, Reports

AST Repository

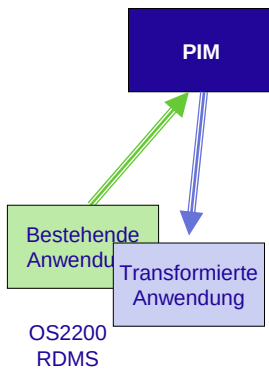
Program Sources, Screen Definitions, DBMS-Schema
HTML, ASP-Sources

PSM→PIM→PSM Transformation



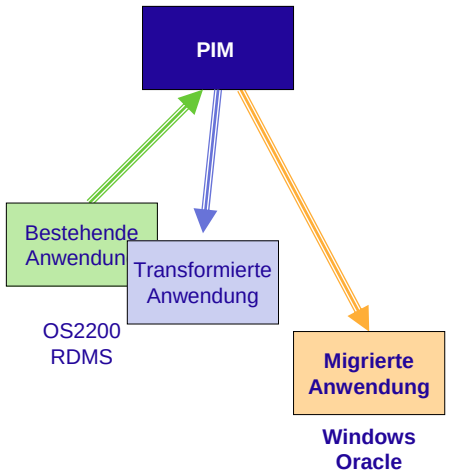
- PSM$_1$
  - Quellartefakte (Programm-Sourcen etc.), automatisch importiert
  - ASTM und KDM
- PIM
  - Ergebnisse aus Analyse und Transformation
  - Plattformneutrales Modell der transformationsrelevanten Elemente
  - KDM
- PSM$_2$
  - Zielartefakte
  - Plattformspezifische Übersetzung
  - ASTM und KDM

Schritt 1: “In-Place” Migration



- Nach Analyse und Transformation wird die Anwendung auf die Originalplattform “zurückgeneriert“
  - Mit unveränderten Schnittstellen und Funktionen,
  - aber neuer Architektur – Data Services und UI-Kapseln
  - Instrumentiert mit Testhilfen
  - Cluster beliebiger Größe
- Test im normalen Wartungsumfeld
  - Hohe Sicherheit durch die bewährte Testumgebung
  - Beliebig große/kleine Inkremente
  - Reversibel
- Kein “Big-Bang”, kein “No-Return”

Schritt 2: Endgültige Migration



- Endgültige Plattform – transformierte und getestete Programme werden noch einmal generiert, kompiliert etc.
- Automatisierte Regression-Tests, abgeleitet aus Schritt 1
- Kurze Durchlaufzeit
  - ➔ Produktionsunterbrechung auf das Minimum reduziert
- Hohe Sicherheit
  - ➔ Kein Risiko, keine Überraschungen

Ziele erreicht?

Vorgabe	Lösung
✓ Risikokontrolle und Risikominimierung – Unterbrechungen der Produktion oder Fehlfunktionen sind nicht tolerabel	Modellbasiertes Vorgehen bietet laufende Kontrolle der Aktionen, Impact-Analysen etc. 2-Schritt-Migration und automatisierte Tests reduzieren das Risiko auf das Minimum
✓ Non-Blocking – Entwicklung und Wartung können nur für kurze Zeit und immer nur für Teile der Anwendungen unterbrochen und eingefroren werden	*In-Place* Migration und Automation erlauben kurze Sperrzeiten für jeweils überschaubare Anwendungsausschnitte
✓ Automation, Reproduzierbarkeit und Rückverfolgbarkeit (Traceability)	Jede im Prozess relevante Information ist in den Modellen festgehalten und lässt sich auf die Quelle zurückführen – KDM-Prinzip Automation ist die Werkzeuggrundlage
✓ Die Effizienz der Software-Entwicklung – heute unterstützt durch perfektionierte SE-Prozesse – darf nicht beeinträchtigt werden	Durch die erweiterte Schichtenarchitektur wird die Effizienz sogar noch erhöht

Ziele Erreicht?

- Ziel – Kostenreduktion in der Produktion ✓
  - Dieses Ziel wird durch den Einsatz kostengünstiger Plattformen erreicht
  - Die Auswahl wird durch die Migration nicht beeinflusst
- Ziel – Lösung von bisheriger (und zukünftiger) Herstellerbindung durch Plattformunabhängigkeit ✓
  - Die Sprache (COBOL) betreffend werden alle OS2200-Elemente durch Standardelemente ersetzt
  - Die Daten- und Benutzerschnittstellen sind gekapselt und plattformunabhängig in Modellen abgelegt. Daraus werden – MDA folgend – plattformspezifische Komponenten generiert.
  - Falls das nicht mehr gewünscht ist, können die Generate auch ohne Werkzeug weiter gepflegt werden.

Architecture-Driven Modernization

- ☑ Modernisierung?
- ☑ Szenarien
- ☑ ADM Taskforce
- ☑ Standards
- ☑ ADM jetzt!

ADM jetzt!

Shift from "From Scratch" Development Philosophy to Phased Reuse

- Replace "throwaway" philosophy with "reuse" philosophy
- Shift from an "all or nothing / go for broke" approach to a phased deployment approach
- Seek lower risks, higher returns and faster delivery through phased delivery strategy

William M. Ulrich – Tactical Strategy Group

- **Modernisierung** ist zwar nicht die einzige, aber eine **unverzichtbare Alternative** bei der Realisierung zunehmend anspruchsvollerer **Business-Anforderungen**
- Die **Standards verhindern** kostspieligen **Wildwuchs**, **geben Leitlinien** und bringen **Interoperabilität** zwischen Werkzeugen verschiedener Hersteller

ADM jetzt!

- ADM bietet perfekte Strategien für die Modernisierung
- Das Instrumentarium für erfolgreiche Modernisierungsprojekte steht jetzt schon bereit – auch wenn der Standard noch in Arbeit ist:
  - Modelle – sind durch den Standard bereits definiert
  - Werkzeuge – aktuelle Werkzeuge implementieren jetzt schon die ADM-Konzepte
  - Methoden – die Basis dazu findet sich in den ADM-Prinzipien, weitere werden im Laufe der Zeit schrittweise dazu kommen
- Legen Sie los ...
  - ... die weitere Arbeit an diesen und anderen OMG-Standards wird nach und nach für eine noch bessere Unterstützung Ihrer Projekte, sowie für eine breitere Auswahl an Werkzeugen und Herstellern sorgen

DELTA software technology
The Generator Company

Modernisierungswerkzeuge von Delta

- SCORE® Transformation Factory
  - Modellbasiert: MOF, XML, den ADM-Standards folgend
  - Komponenten für Discovery, Analyse, Transformation und Generierung
  - Konfigurierbar und individuell erweiterbar
- SCORE® Adaptive Bridges
  - Generatives Service Enablement
  - Modellbasiert: MOF, XML, MDA, EDOC
  - Non-invasiv
  - Plattformunabhängig, technologieverbindend

DELTA software technology
The Generator Company

Architecture-Driven Modernization

Vielen Dank für Ihre Aufmerksamkeit

- Weitere Informationen
  - www.D-S-T-G.com

adm.omg.org
www.omg.org

www.systemtransformation.com
www.klocwork.com
www.softwarerevolution.com

Wiliam M. Ulrich: Legacy Systems: Transformation Strategies, Prentice Hall, 2002