Making Ideas a Reality

MDA basierte Entwicklung von Java Applikationen

Michael Benkel

Chaos Report, Standish Group

- 1994: basiert auf 7500 IT Projekten
  16 % erfolgreich (on time, budget, feature)
  31 % abgebrochen
  53 % irgendwie beendet
- 1996: 27 % erfolgreich
- 1998: 26 % erfolgreich
- 2000: 26 % erfolgreich
- 2002: basiert auf 175000 IT Projekten
  16% erfolgreich
  31 % abgebrochen
  53 % irgendwie beendet

Status der SW Entwicklung

- Anteil der Software wächst in allen Bereichen
- Anforderungen an SW Systeme wachsen
  - Benutzerfreundlich
  - Hohe Verfügbarkeit
  - Gutes Zeitverhalten
- Mix aus unterschiedlichen Technologien

- Höhere Komplexität der Anwendungen

Did you know:

Java 1.4 has 26,000 interfaces and 4,000 new interfaces per year.

Quelle: http://www.maketechnologies.com/solutions/solutions2.html

Schlussfolgerung Nr. I

> **Modell**-zentriertes Arbeiten

<<UML>>

UML, ein Open Standard

- Standardisiert von der OMG
  - UML Konsortium
  - www.omg.org/uml
- Notation zur Beschreibung von Systemen
  - Einheitliche Notation für viele Einsatzgebiete
- 8 Diagrammformen zur Beschreibung unterschiedlicher Aspekte
- Erweiterungsmechanismen
- Gibt keinerlei Hinweise zum Prozess

Komplexität im Griff durch Modelle

- Abstraktionsgewinn durch die Verwendung von Modellen
  - Komplexe Sachverhalte möglichst einfach darstellen
- Verschiedene Sichten auf den Problembereich
  - System als Black Box
  - Statische und dynamische Aspekte
- Ein gutes Modell liefert
  - Besseres Verständnis der Aufgabenstellung
  - Bessere Kommunikation im Projektteam
  - ...

Vorteile von Modellen

Aussagen von Kent Beck:

- Bilder machen schlechtes Design sichtbar
- Zeichnen von Bildern ist 3 mal schneller als codieren

aber

- Bilder nicht aus Angst verwenden
- Nicht mehrere Formen derselben Information auf dem gleichen Stand halten

Ziel ist ein lauffähiges System

- Es muß Code erzeugt werden
  - C++, Ada, Java, XML, VB, Java Script,...
- Es muß getestet werden
  - Testfälle, Test Scripte, ...
- Es muß dokumentiert werden
  - HTML, XML, RTF, ...

>> Umsetzung der Modelle in die Implementierung

Umsetzung der Modelle in der Praxis

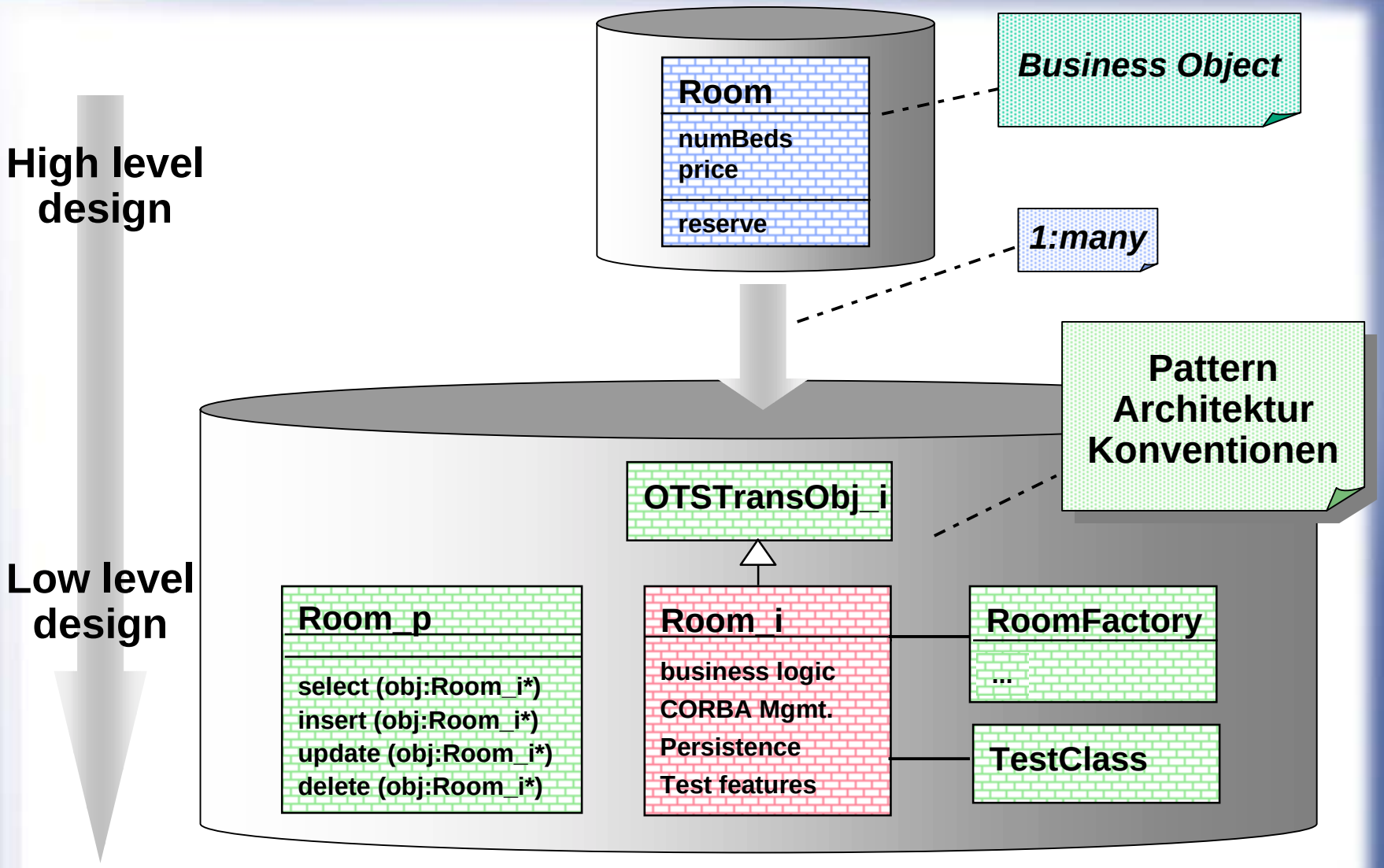
„ Generieren von Code? ... Viel Aufwand im Detaildesign ... “

„ Domain Modelle sind guter Startpunkt im Projekt “

„ Up-to-date Domain Modell?“
„So etwas haben wir nicht... “

„ Design Modelle enthalten viel Implementierungsdetails “

Evolution des Design Modells - Beispiel



Problem beim Einsatz von CASE Tools

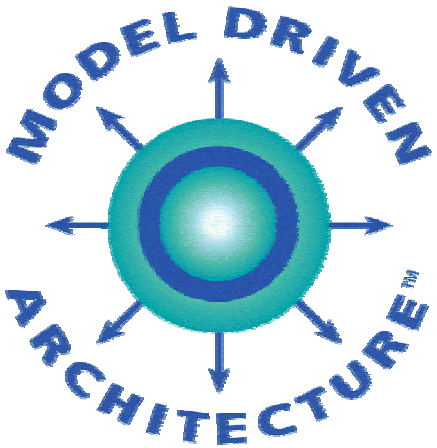
- Im Design wachsen Modelle oft dramatisch an
  - Aus Analyse Objekt werden mehrere Design Objekte
  - Architektur/Technologie führt zu redundanten Informationen

- Zwei Optionen für die Lösung
  - Klares, übersichtliches Design, keine Code Generierung möglich
  - Detailliertes, unübersichtliches Design, Code Generierung möglich

Schlussfolgerung Nr. II

> **Transformation von Modellen**

<<UML + MDA>>

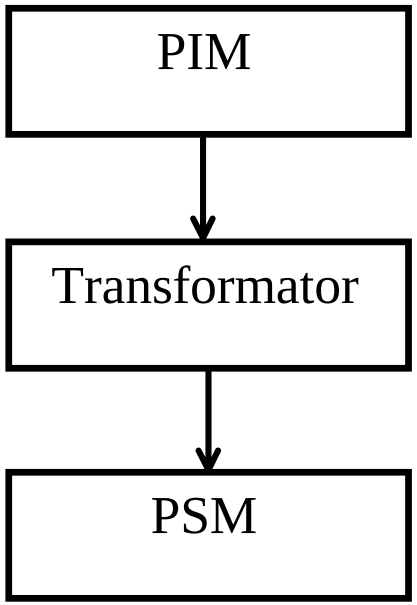


MDA im Überblick

- Neuestes Thema bei der OMG
  - www.omg.org/mda
- Fachliche und technische Aspekte trennen
  - Erstellen von Domain Modelle
  - Unabhängigkeit von der Zielplattform
  - Portability, increasing application re-use
- Modelle nehmen zentrale Stellung im Entwicklungsprozeß ein
- Transformation von Modellen

UML und MDA

- Erstellen von Platform Independant Model‘s (PIM)
  - Verwendung von UML Notation
  - Fokus liegt auf der Domäne
- Abbilden auf Platform Specific Model‘s (PSM)
  - Durch Transformatoren
    - Maximizing automation of the mapping step is the goal (MDA Paper, OMG)
  - Von Hand
- Source Code ist auch ein PSM



Fachliche und technische Aspekte

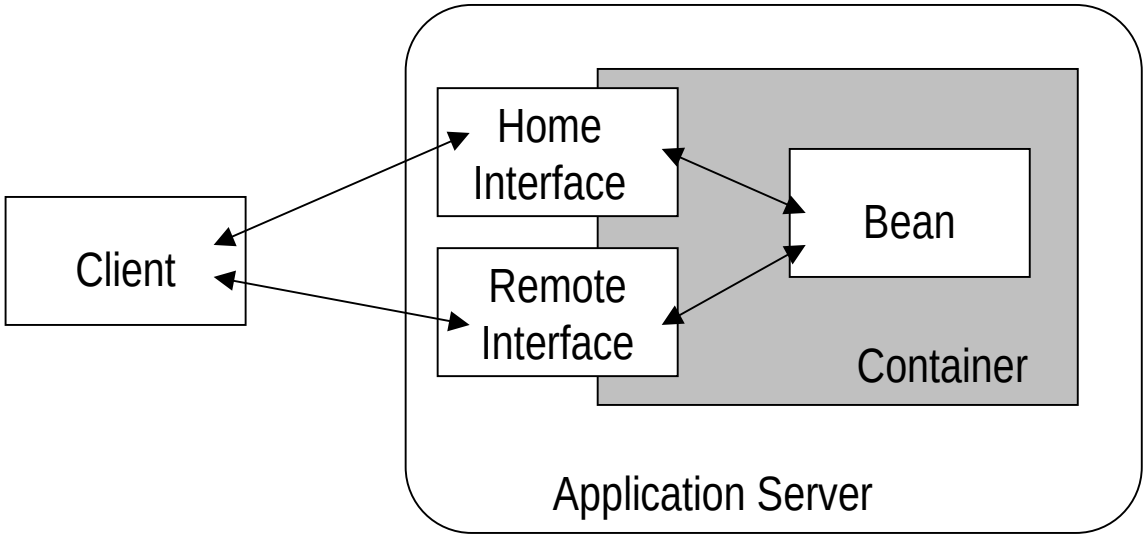
- Fachliche Aspekte machen nur 40% bis 50% des Source Code aus (Kühne 1999)
- Rest ist „Infrastruktur“:
  - Middleware (COM, CORBA, EJB, etc)
  - Persistenz und Datenbankzugriffe
  - Fehlerbehandlung, Implementierungskonventionen
- „Infrastruktur“ ist programmtechnisch kompliziert
- Fachliche und technische Aspekte haben unterschiedliche Release Zyklen

Wann funktioniert das?

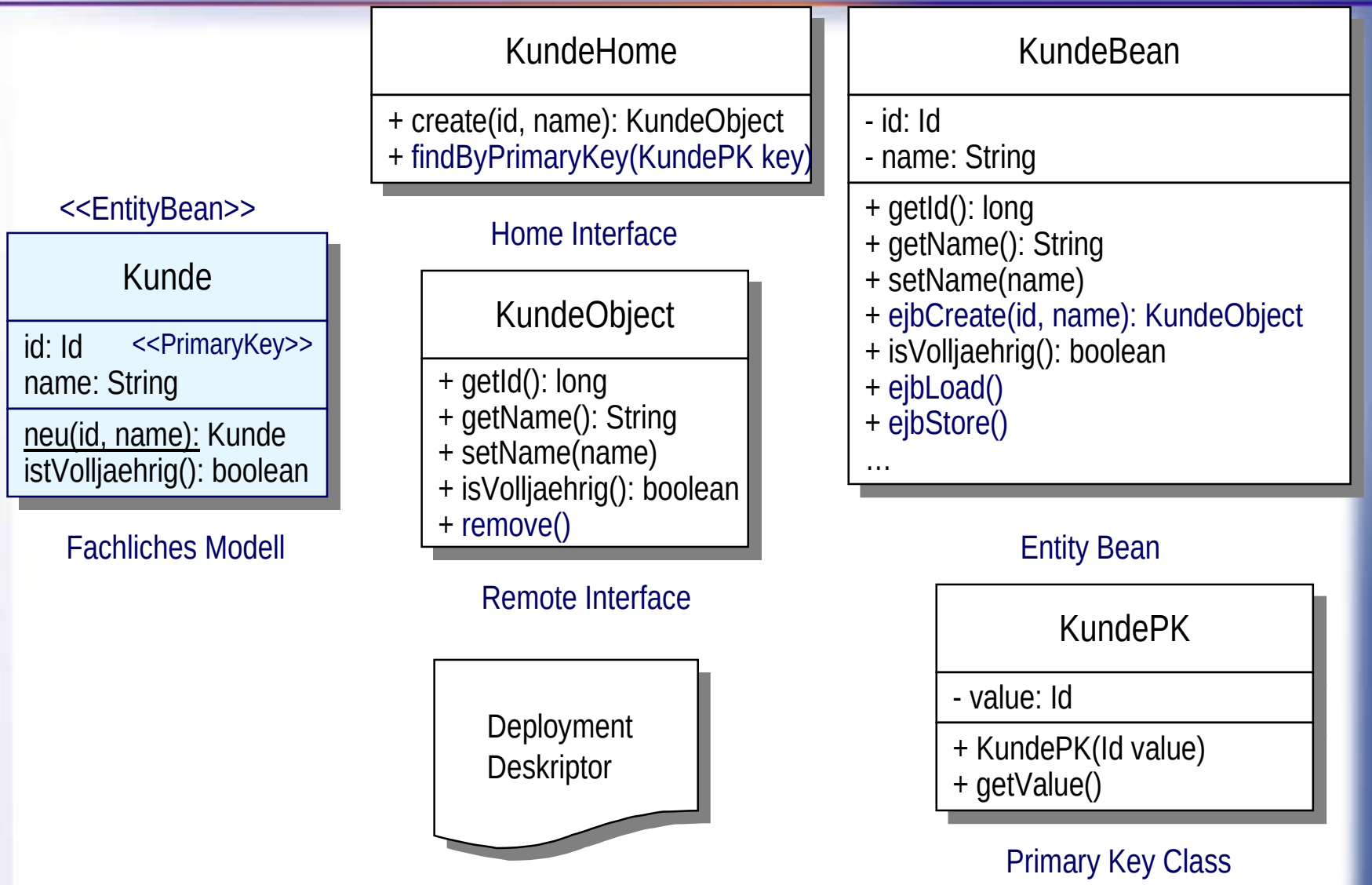
- Viele Geschäftsobjekte, die (rein technisch gesehen) nach den gleichen Prinzipien implementiert werden
  - Einhaltung von Programmierrichtlinien
  - Einsatz vordefinierter Design Pattern
  - Verwendung einer technischen Architektur
    - EJB, .Net
    - Corba Components
  - Verwendung eines speziellen Mappings
    - OO nach C
    - OO nach RDBMS

Beispiel EJB

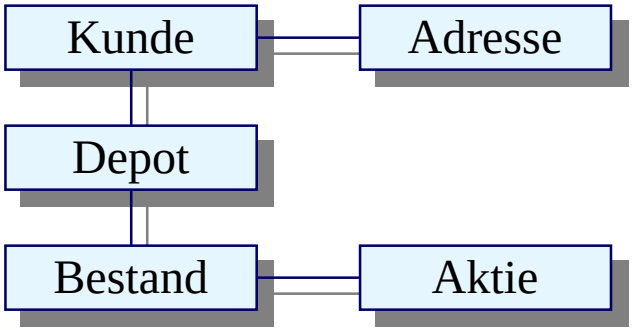
- Business Objekt lebt in einem Container
- Definierte Schnittstellen nach außen



Entwicklung des EJB Modells

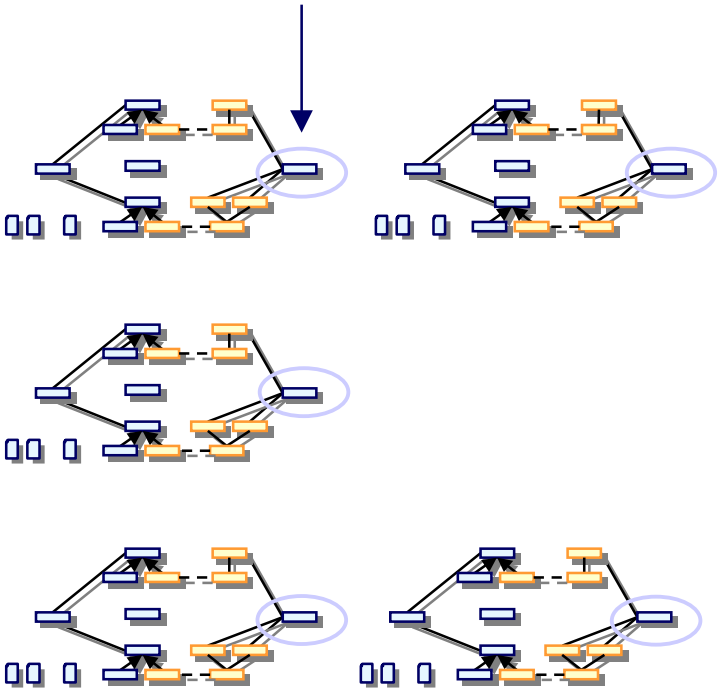


Besser, Konzentration auf das Wesentliche



Fachlich Entwerfen

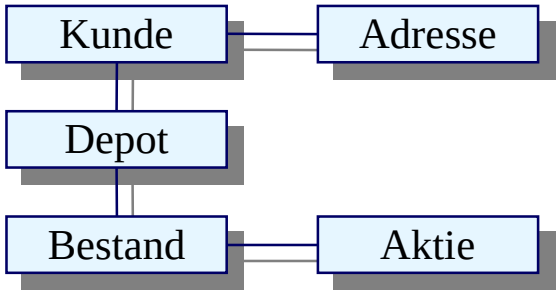
Manuelle Eingriffe an wenigen Stellen



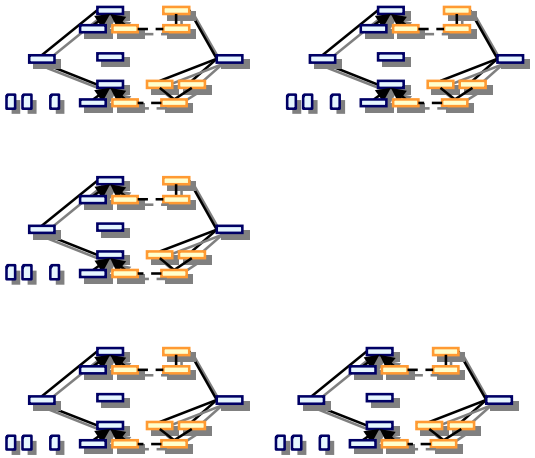
Code (weitestgehend) Generieren

Ein paar Zahlen - Kundenprojekt

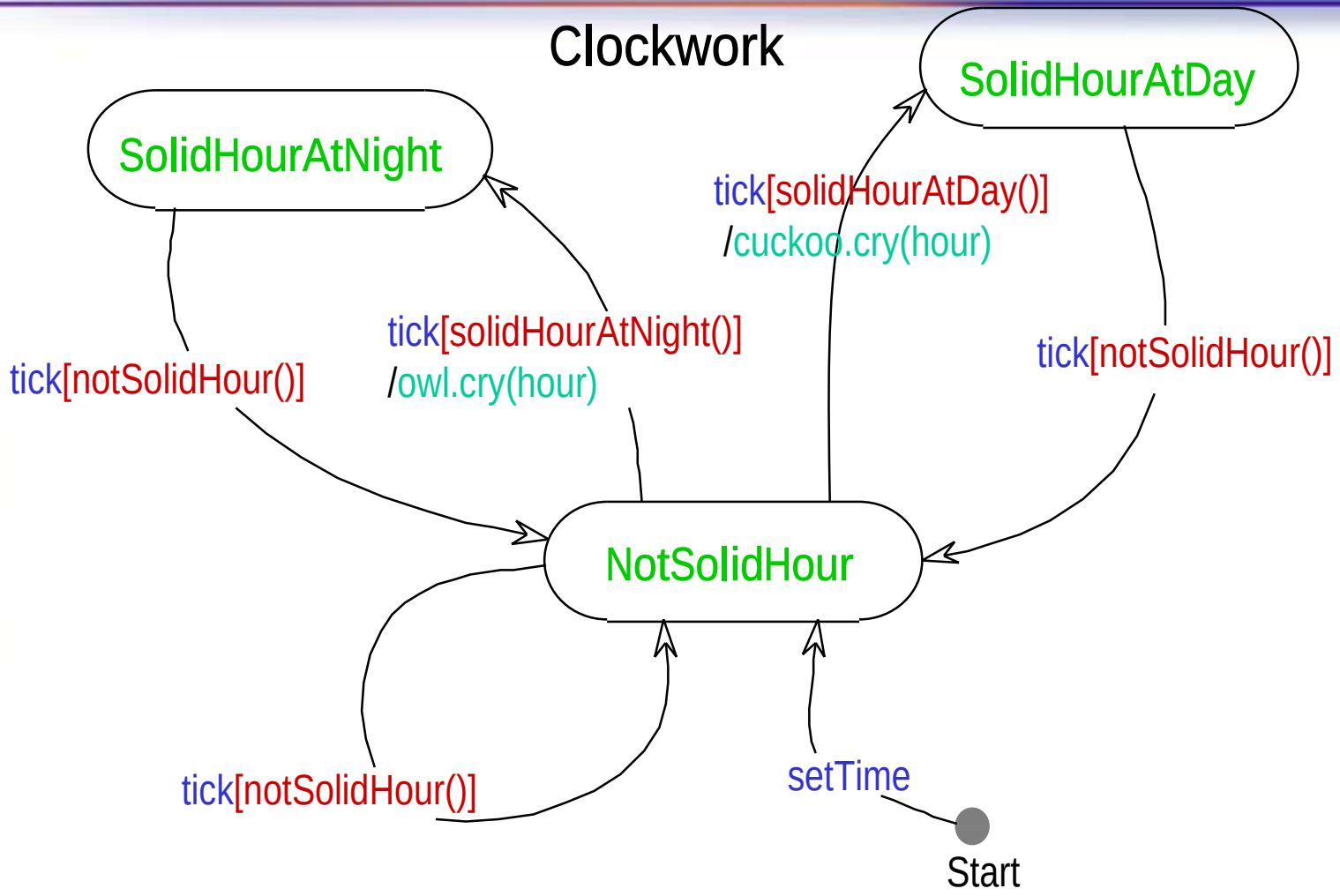
- Manuell modelliert:
  - 11 Klassen
  - 56 Attribute
  - 46 Operationen



- Translativ generiert:
  - 77 Klassen
  - 109 Dateien
  - 155 Attribute
  - 1164 Methoden



Beispiel Zustandsautomaten



Java Zustands-Modell Mapping

```
byte SolidHourAtNight = 0;                  // States are numbered
byte currentState = NotSolidHour;           // holds current state

public void takeEvent(byte ev)              // this function implements the
{                                                   // state machine
   switch(currentState) {
   case NotSolidHour:                       // State NotSolidHour
            switch(ev) {
            case tick:                      // When tick event comes do ...
                if ( solidHourAtDay() )     // Guard Condition
                      {
                        cuckoo.cry( hour );  // operation on transition
                        currentState = SolidHourAtDay;      // next state
                      }
                      break;
                      if ( /* other Guard */ ) ...
            }
   case /* other state */ ...
   }
}
```

Template für Zustands Definition in Java

```
byte [MState.name] = [getCount()];
byte currentState = [MState.name];
...
public void takeEvent(byte ev)
{
    switch(currentState) {
    [loop(MClass->MStatemachine->MState)]
        case [MState.name]:
            switch(ev) {
            [loop(MState->ToState->MTransition->TriggerEvent) ]
            case [TriggerEvent.name]:
            ...
            [end loop]
            }
    [end loop]
    }
}
```

UML nach Maß

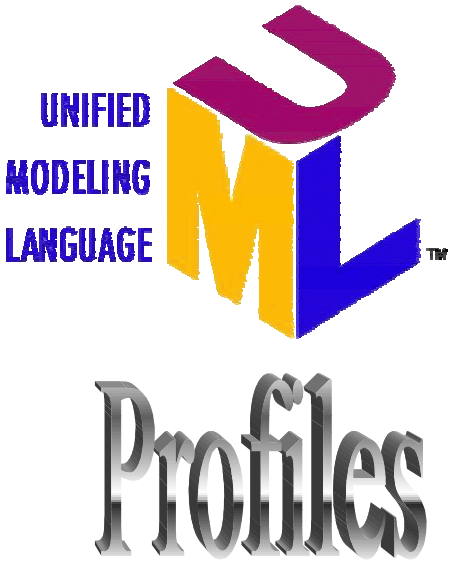
- Semantik der Modells ist abhängig von der Aufgabenstellung
  - Kaufmännische Applikationen
  - Embedded Systeme
  - Systems Engineering
- Beschreibung spezieller Sachverhalte
  - Ohne die Notation zu verändern

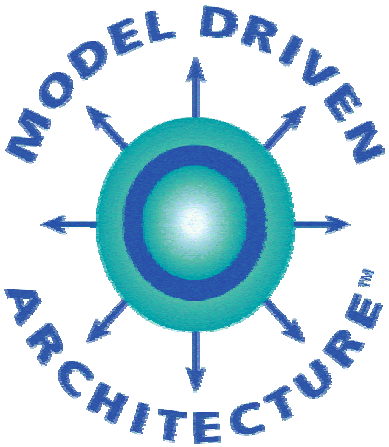
>> Verwendung von UML Profilen

Schlussfolgerung Nr. III

> **UML Profiles**

<<UML + MDA + Profiles>>





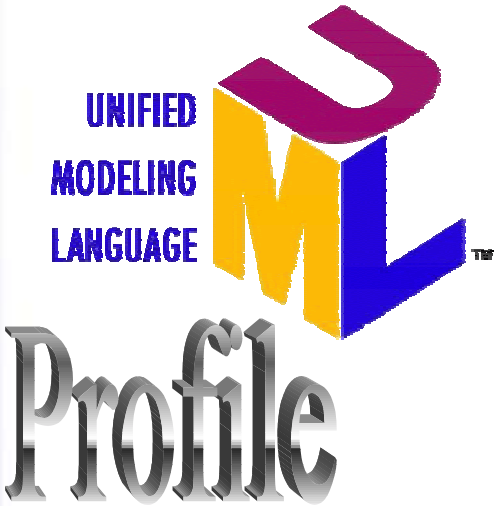
UML Profile

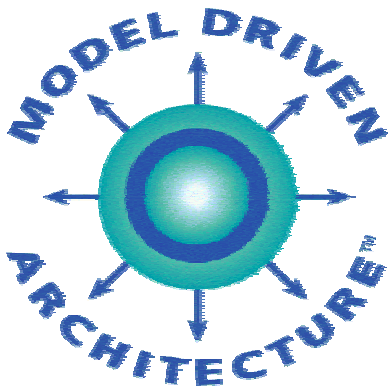
- Profile
  - Nutzen der Erweiterungsmechanismen der UML
  - Vordefinierte Stereotypes, Constraints, Tagged Values
- UML 2.0 beschreibt die Modellierung von Profilen
- Profile sind domainspezifisch
- Einfluss von Profilen
  - Auf die Modellierung
  - Auf die Transformation der Modelle

Schlussfolgerung Nr. IV

> Werkzeug-basiertes Arbeiten

<<UML + MDA + Profile + StP>>





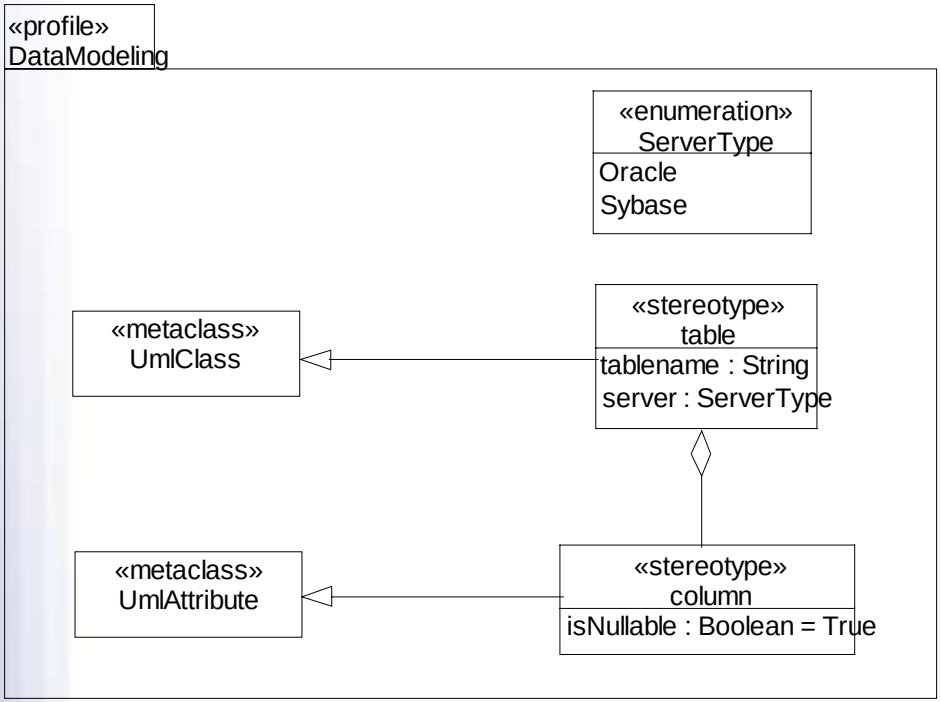
Software through Pictures (StP)

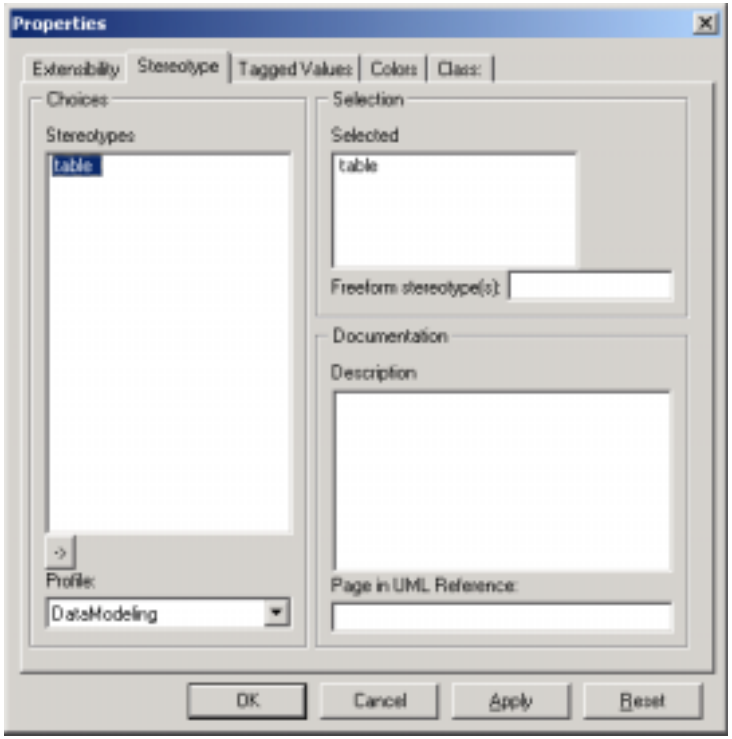
- Die Lösung für den MDA Ansatz
  - Eine der besten UML Implementierungen im Markt
  - Leistungsfähiger Transformator
- Support für UML Profile
- Vordefinierte Profile
  - C++, Java, Ada
  - EJB, CORBA
  - Datenbank Mapping
  - Abbildung von UML auf C

StP Profile Builder

- Basiert auf U2-Partners UML 2.0 proposal
- Modellierung von Profilen
  - Definition von Stereotypen und Tagged Values
  - Abhängigkeiten zwischen den Stereotypen
  - Modellelementen zuordnen
- Dokumentation der Profile
- Auswahl der Profile bei Projektstart
  - Aus vordefinierten Profilen
  - Aus selbstdefinierten Profilen

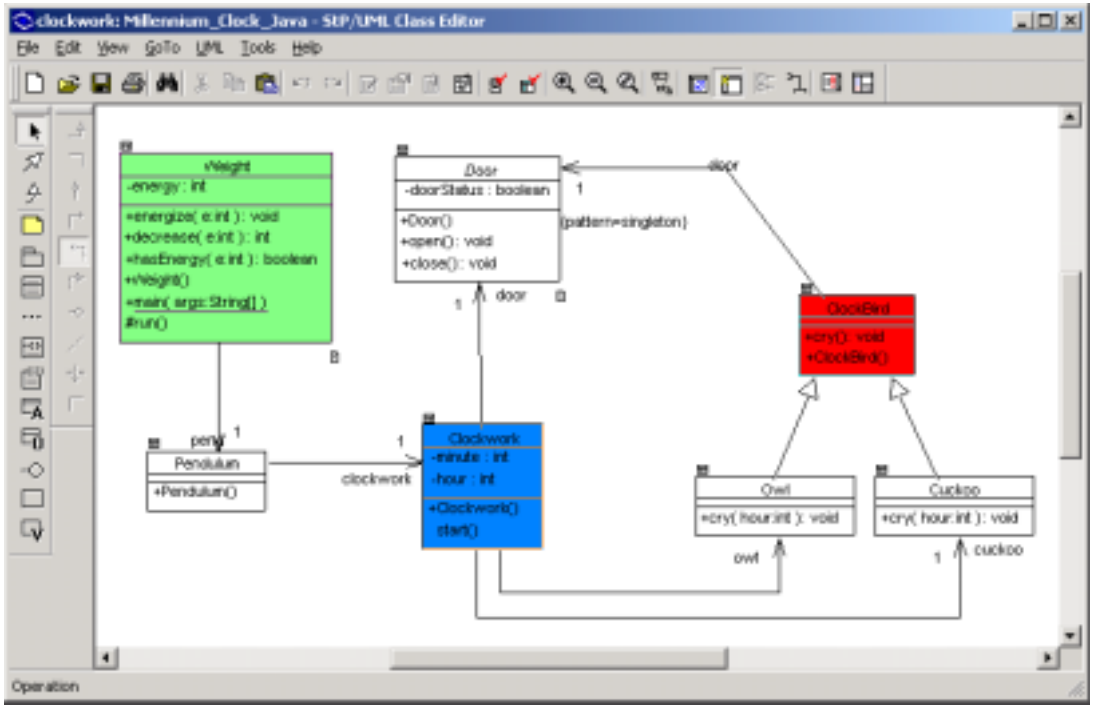
StP Profile Editor



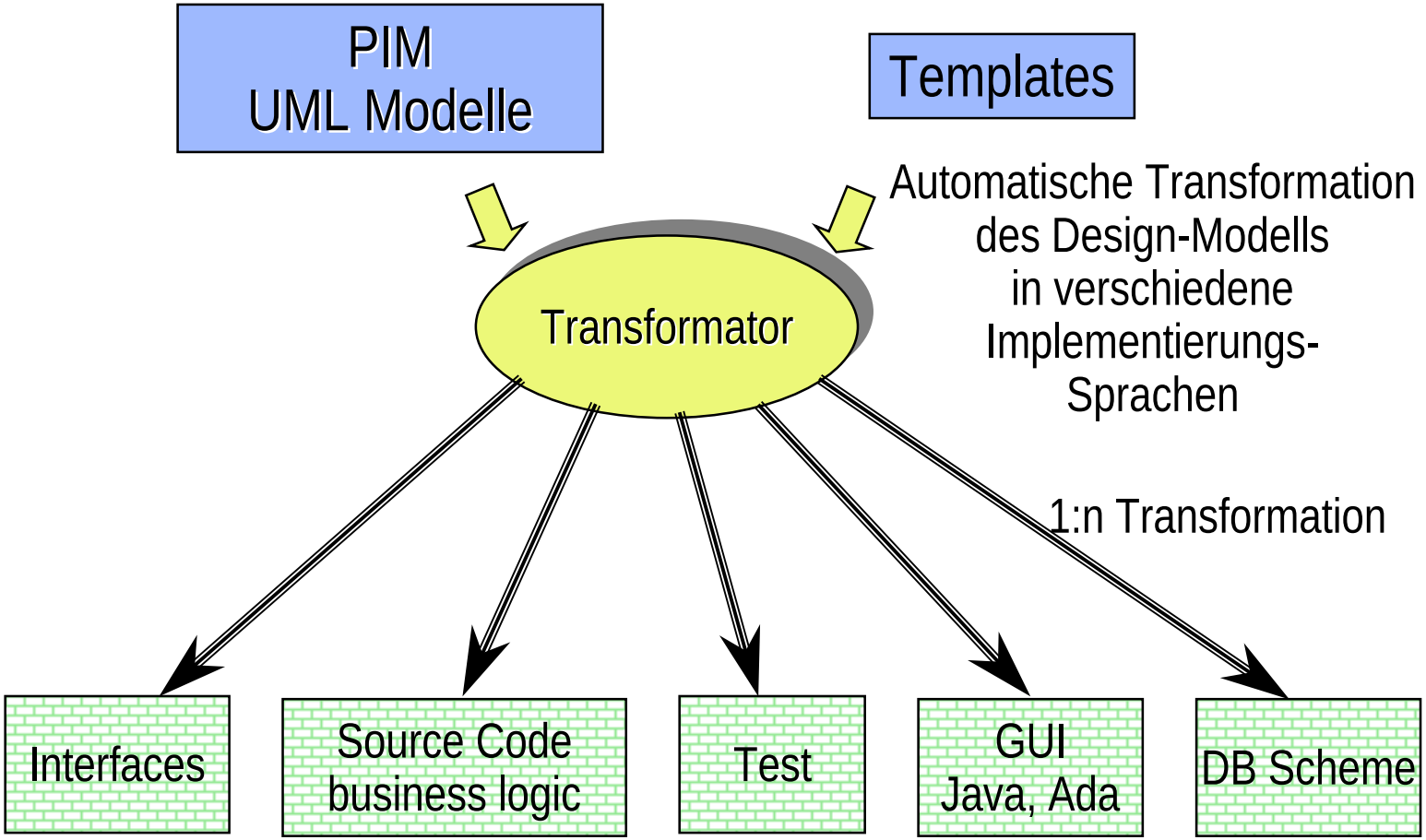


UML Profiles in Color

- Basiert auf ‚Java Modeling in Color with UML - Peter Coad‘
- Zuordnung von Farben zu Stereotypen



StP/ACD, Transformation von Modellen



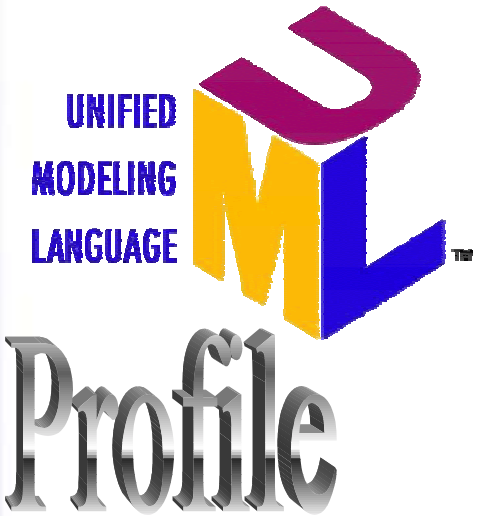
Was sollen Templates leisten?

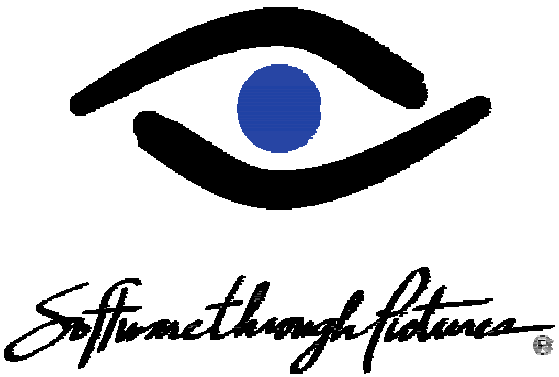
- Flexible Anpassungen an Zielarchitekturen
- Implementierungen von Profilen mit Bereitstellung von semantischen klar definierten UML Modellelementen
- Unterstützung verschiedener Zielsprachen
- Unterstützung von Klassendiagrammen und Zustandsautomaten
- Generierung von Header in Quellen, die gemäß des angewendeten Standards die notwendigen Informationen enthalten

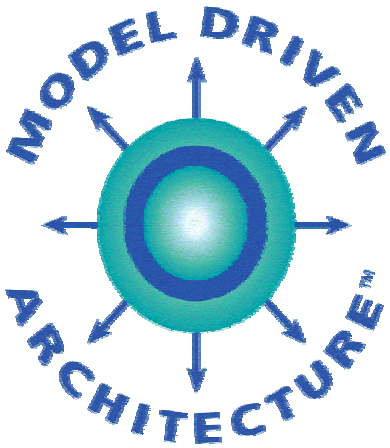
Schlussfolgerung Nr. V

> Umfassende Vorteile mit dieser Vorgehensweise

<<UML + MDA + Profile + StP>>







Vorteile von MDA laut OMG

- Erhöhung der Produktivität
- Modelle, Basis für die Wiederverwendung
- Bessere Qualität der Anwendung
- Schnellerer Einsatz neuer Technologien

Erhöhung der Produktivität

- In SW Projekten über mehr Mitarbeiter
- Ab einer Teamgröße von x läßt sich die Produktivität kaum steigern
  - Quelle: Tom De Marco, Der Termin

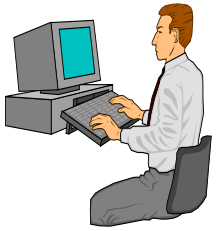
- Automatisierung ist die bessere Lösung
  - Erstellen von Templates durch Spezialisten
  - Know How wird durch Generator vervielfältigt

Wiederverwendung

- Beliebtes Schlagwort im OO Umfeld
- Wiederverwendung um
  - Kosten einzusparen
  - Schneller am Markt zu sein
- Fokus bei MDA:
  - Wiederverwendung von fachlichen Modellen
- In der Praxis:
  - Wiederverwendung von technischen Mustern

Wiederverwendung in der Praxis

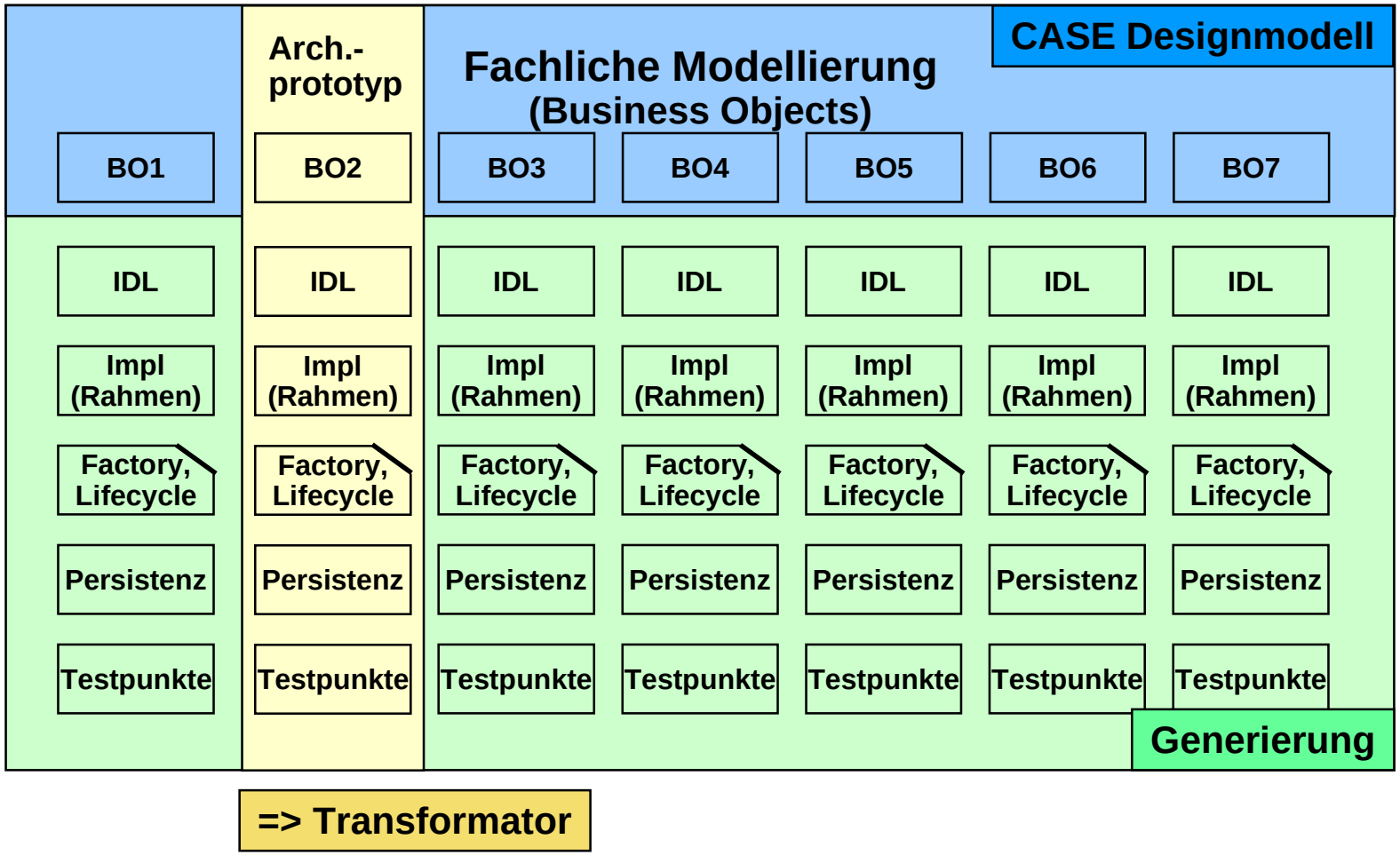
Kennen Sie die 'Kultur' des Copy&Paste?



Viele Projekte verwenden Copy&Paste!?

- Copy&Paste von technischen Konzepten, Pattern, ...
- Fehleranfällig, redundante Informationen, nicht wartbar

Automatisiertes ‘Copy&Paste‘



MDA und technische Architekturen

- Auswahl der technischen Architektur
  - .Net, J2EE, CORBA, ...
  - Bloß nichts falsch machen
- MDA, Trennung der fachlichen und technischen Aspekte
- Technologische Änderungen einfach möglich
  - Lerneffekt, neue Anforderungen
  - Haben keinen Einfluß auf die fachlichen Modelle
  - Risiko der falschen Technologiewahl ist geringer

Bessere Qualität der Anwendung

- Basis sind stabile Domain Modelle
  - Wesentlich kompaktere Modelle
  - Fokus für einen großen Teil der MA‘s
  - Re-Use von Domain Modellen
- Transformation in die Zielumgebung erfolgt größtenteils automatisch
  - Fehler lassen sich einfach beheben
  - Jede manuell implementierte Codezeile kann Fehler haben

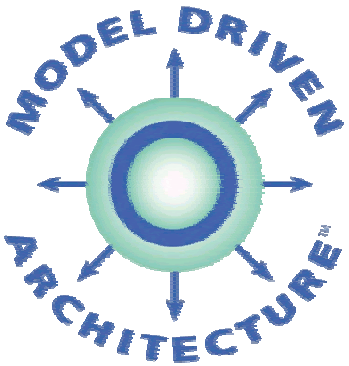
Eigentlich kein neues Konzept

- Translative Code Generation von Stephen Mellor
- Aspect Oriented Programming (AOP) des Xerox Palo Alto Research Center
- Wizzards in IDE‘s, aber wesentlich flexibler
- Wird seit Jahren erfolgreich eingesetzt

- Automatisierung in der Softwareindustrie
  - Produktivitätsgewinn
  - Qualitätsgewinn

Aonix und MDA

- Aonix ist seit 1992 OMG Member
- Aonix propagiert diese Vorgehensweise schon seit mehr als 4 Jahren
- Mit Software through Pictures haben wir ein Produkt für UML und MDA
  - StP/UML
  - StP/ACD, Architecture Component Development
- Praktische Erfahrung beim Einsatz von MDA
  - Referenzen aus verschiedensten Branchen

Zusammenfassung

- UML, der Standard für OO Projekte
- At a Minimum, MDA is a more sophisticated Way of using the UML
  (Paul Harmon)
- MDA hat sich in der Praxis bewährt
- OMG ist Befürworter von MDA