

# JAVA FORUM 2005 stuttgart

**MDA mit Open-Source-Software – Eine Bestandsaufnahme**

**Gerhard Wanner ([wanner@hft-stuttgart.de](mailto:wanner@hft-stuttgart.de))**

**Stefan Siegl ([s.siegl@novatec-gmbh.de](mailto:s.siegl@novatec-gmbh.de))**

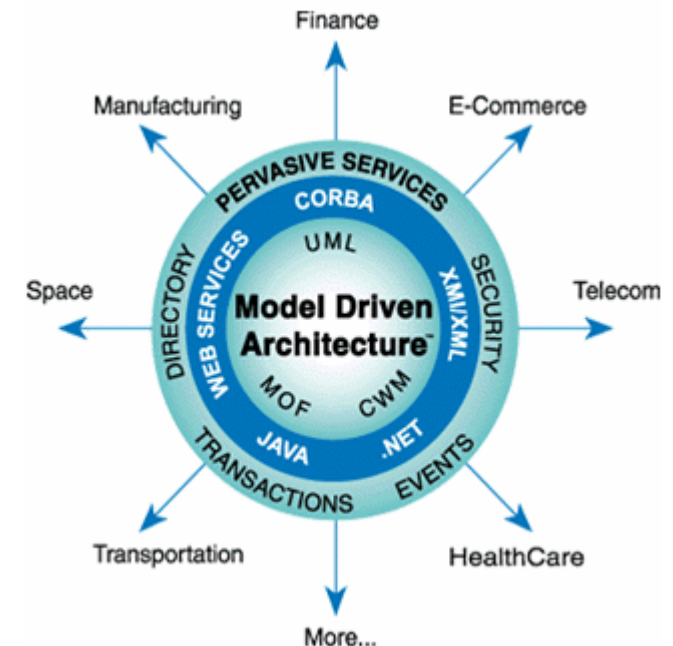
- **Model Driven Architecture (MDA)**
  - Einführung/Übersicht/Motivation
  - Unsere Anforderungen an MDA
- **Marktübersicht (Longlist)**
- **Untersuchte Werkzeuge (Shortlist)**
  - AndroMDA
  - Open Generator Framework
  - openMDX
- **Zusammenfassung/Ergebnisse.**

## ■ Ständig wechselnde Techniken und neue Komponenteninfrastrukturen erschweren die Erstellung von Anwendungen

- MDA ist die Antwort der OMG auf diese Situation

## ■ MDA besteht aus 3 Schalen:

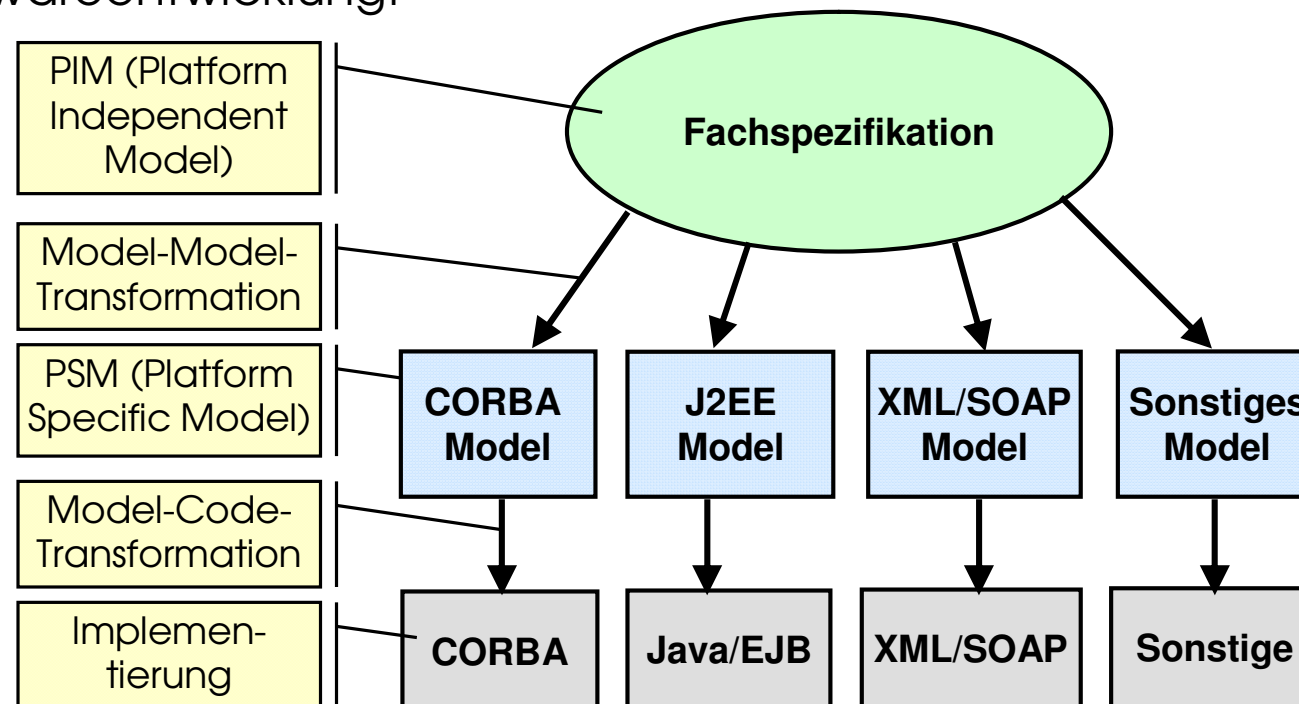
- Der MDA-Kern integriert die von der OMG spezifizierten Standards und beschreibt die Modellierung
- In der nächsten Schale der MDA befinden sich die für den Ablauf der modellierten Anwendungen nötigen Techniken oder Plattformen
- In der äußersten Schale der MDA befinden sich allgemeinere Dienste.



Das Schichtenmodell der OMG für das Zusammenspiel der verschiedenen Bestandteile (Quelle: [www.omg.org](http://www.omg.org))

- Die Grundidee von MDA ist, durch den Einsatz von Modellen so viel Code wie möglich automatisiert für die Plattform zu generieren

- MDA ist eine konsequente Weiterführung des Abstraktionsgedankens in der Softwareentwicklung.



## ■ Modelle

- Sichern der Wiederverwendbarkeit von Modellen durch Langlebigkeit
- Aktuelle Dokumentation des Systems
  - Diskussion auf verschiedenen Ebenen (PIM, PSM) möglich

## ■ Generierung

- Modellierung wesentlicher Architekturmerkmale; kein starrer Generator
- Konstante Codequalität für die generierten Teile (durch gute Templates)

## ■ Einsatzbereich

- Primär Einsatz bei Neuentwicklungen
- Nachträglicher Einsatz bei einer ohne MDA entwickelten Anwendung problematisch

## ■ Entwicklungsprozess

- Hohe Produktivität, wenn Architektur steht und Werkzeug etabliert
- Gute Integration bei Einsatz eines iterativen Vorgehensmodells, wenn vom Werkzeug unterstützt...

## ■ Modell

- Mindestens „marked PIM“ (a.k.a. annotated PIM)

## ■ Metamodell

- Verständlich und anpassbar; kein proprietäres XMI-Format
- Ideal: UML, MOF

## ■ Templatesprache

- Hohe Funktionalität und Übersichtlichkeit; z.B. kein JavaScript
- Flexibles Templatesystem
  - Freie Anpassbarkeit der Templates durch den Entwickler
- Direkte Verwendung der Modellelemente in der Templatesprache
- Ideal: Velocity, da weit verbreitet

## ■ Import-Möglichkeiten aus (Modellierungs-)Werkzeugen

- Gute Import-Möglichkeiten
- Ideal: XML, aber wegen verschiedener Versionen oft inkompatibel

## ■ Entwicklungsprozess

- Forward-Engineering mit geschützten Codebereichen
- Ideal: Unterstützung von Roundtrip-Engineering.

- **GMT (Generative Model Transformer – Eclipse Projekt)**
- **JAG (Java Application Generator)**
- **UMT-QVT**
- **AndroMDA**
- **Open Generator Framework**
- **openMDX.**

## ■ GMT (Generative Model Transformer)

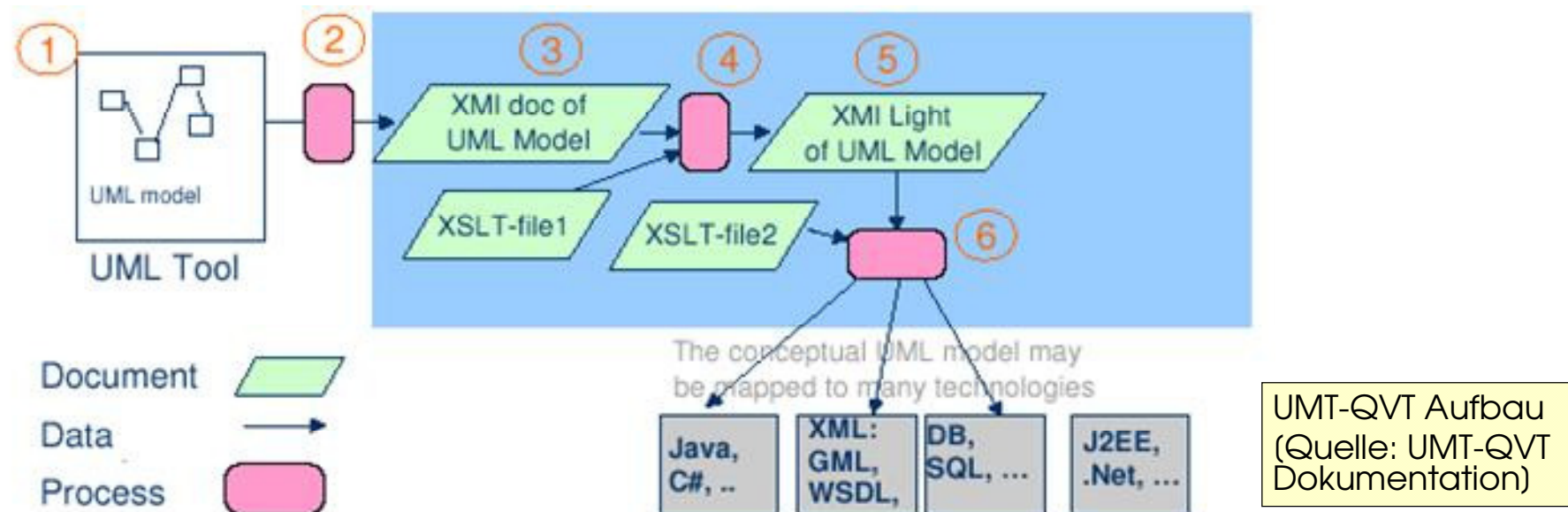
- Eclipse Projekt derzeit noch in der Anfangsphase
- Ziel: Basiswerkzeuge für MDA Ansätze bereitzustellen – geplant sind:
  - Kombination zweier XMI Modelle
  - Transformation XMI → XMI
  - Generierung XMI → Text
  - Oberfläche

## ■ JAG (Java Application Generator)

- Generiert J2EE Anwendungen basierend auf angereichertem UML Model
- Templatesprache: Velocity
- Die Möglichkeit eigene Templates zu definieren ist komplex.

## ■ UMT-QVT

- Hat nichts mit dem OMG Standard „Query, View, Transformation“ zu tun
- Zweistufiger Ansatz
- Modellierung in UML
- Intern wird XMI mittels XSLT in XMI Light (proprietär) überführt
- Roundtrip Engineering kompliziert (Überschreiben ja/nein)
- UMT Profile zur Definition der erlaubten Stereotypen.



## ■ Ursprung: UML2EJB Projekt – Seit März 2003 AndroMDA

## ■ Modellierung

- Zweistufiger Ansatz:  
„marked PIM“ /  
„annotated PIM“
- Basierend auf UML
- Metadaten Prozessierung  
mittels MDR

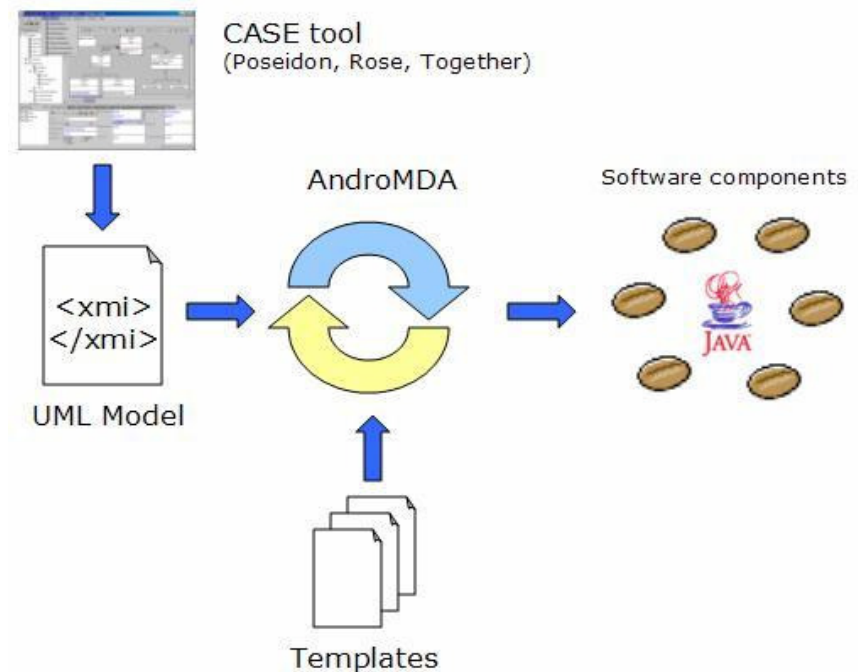
## ■ Modularer Aufbau

## ■ „Cartridges“

- Code Generierungskomponenten
- Default-Skriptsprache: Velocity
- Out of the box: Hibernate, EJB,  
Struts, Web Services, Spring

## ■ Metafacades

- Wrapper um Model Elemente.



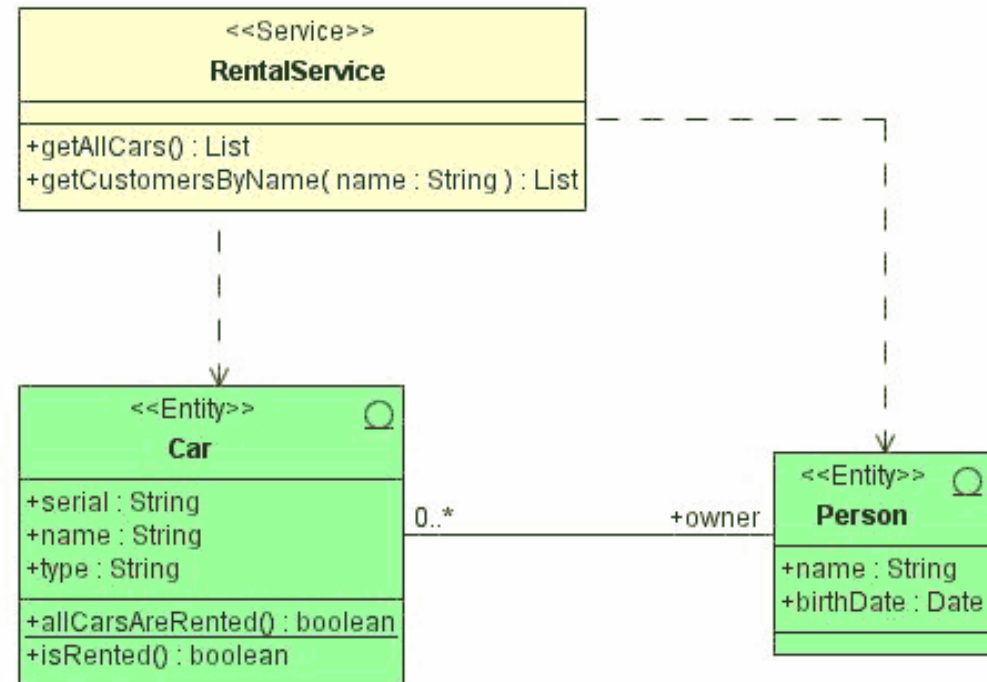
AndroMDA Übersicht  
(Quelle: [www.andromda.org](http://www.andromda.org))

## ■ Out of the box

- Cartridges
- Automatisierter Build Prozess (Ant / Maven)
- Schema2XML
- Kickstart für neues Projekt

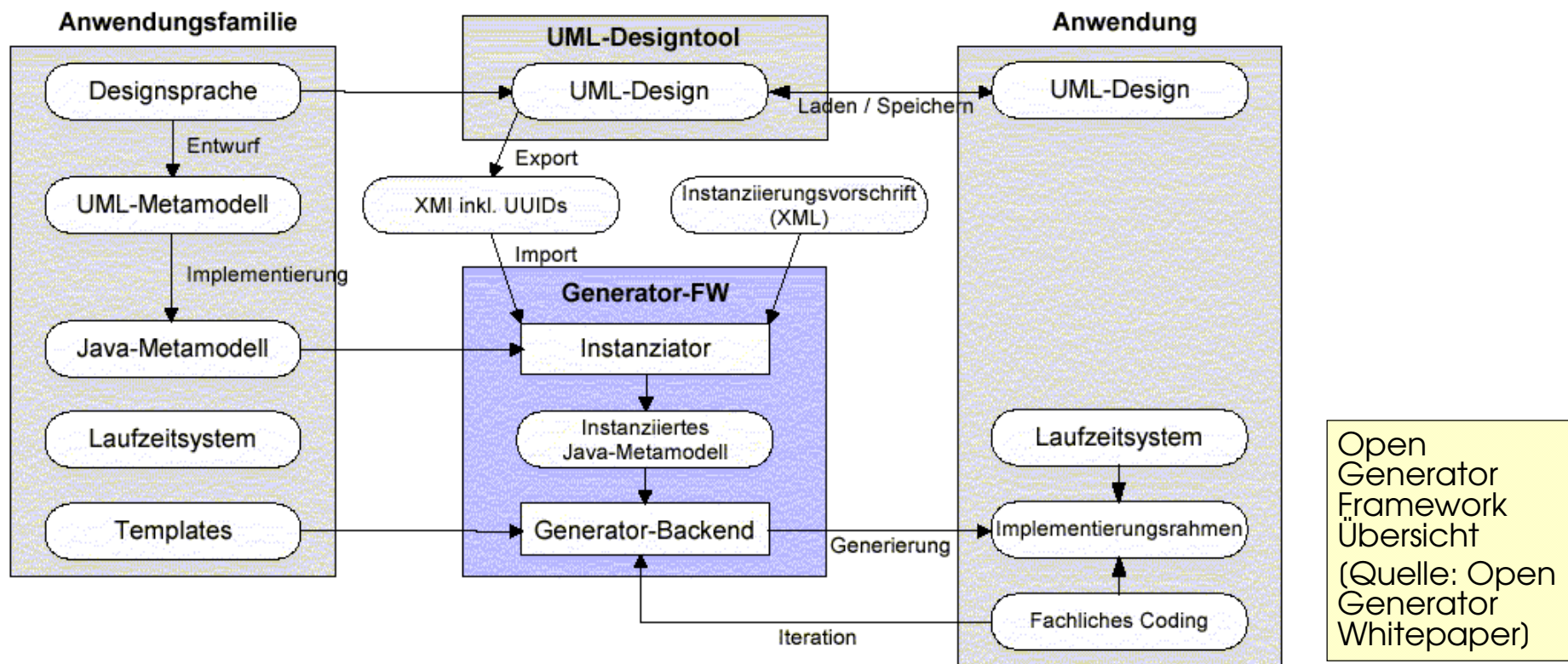
## ■ Roundtrip Engineering

- Keine Protected Regions (führt zu starken Abhängigkeiten).



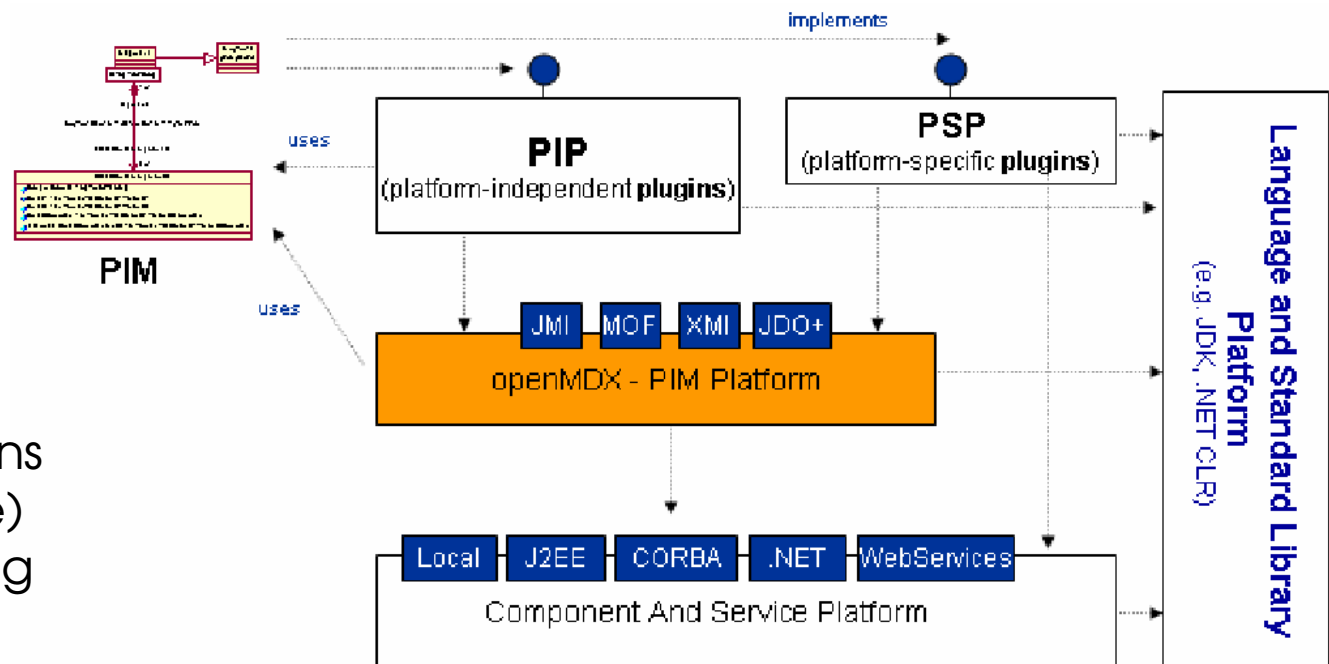
„marked PIM“ Beispiel  
(Quelle: [www.andromda.org](http://www.andromda.org))

- **Basiert auf dem b+m Generator Framework**
  - Seit September 2003 Open Source
- **MDSM (Model Driven Software Development) konform**



- **Zweistufiger Ansatz: „marked PIM“**
- **Frei definierbares Objektorientiertes Metamodel**
  - Metamodel unabhängig (nicht UML gebunden)
  - Verwendung verschiedener Inputformate für die gleiche Anwendung möglich
  - „Instantiators“ regeln die Überführung in eine in-memory Repräsentation
- **Roundtrip Engineering**
  - Durch Protected Regions unterstützt
- **Skriptsprache Xpand**
  - Proprietär – ähnelt XSLT
  - Polymorphismus auf Templateebene
- **„Generator-Plugin“**
  - Definieren das Templateverhalten, Instantiators und Invoker
- **„Invokers“**
  - Definieren Hilfsfunktionen für Templates.

- Frameworkgestützt - Kein generatives PIM-PSM Mapping
- Zusätzliche Abstraktion von Komponenten und Service Architekturen
- Modellierung
  - MOF
- Plugin-gestützt
  - Client – Supplier Prinzip
  - Generische Plugins (bsp. Persistence)
  - Zusammenfassung zu "Providern"
- Proof of concept
  - openCRX.



openMDX Übersicht  
(Quelle: openMDX Quickstart)

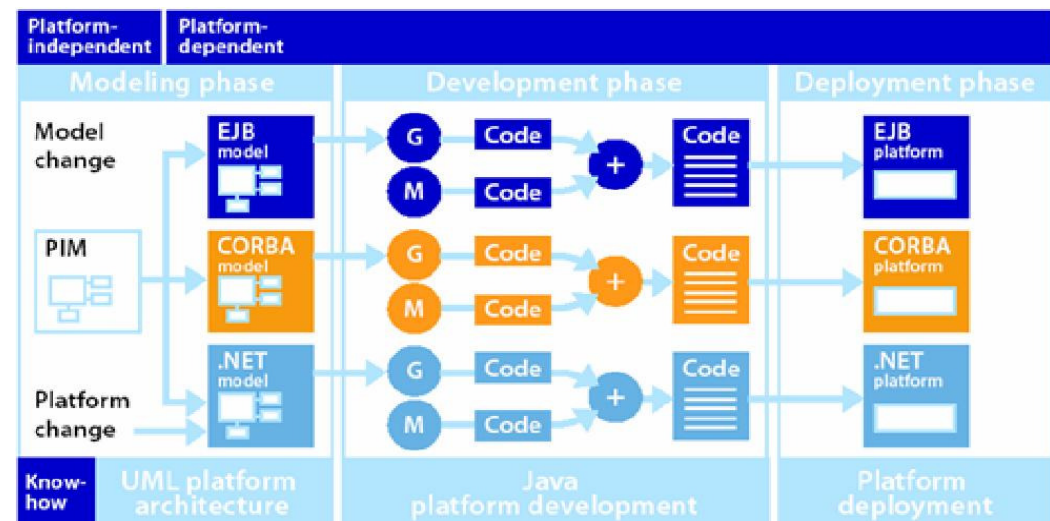
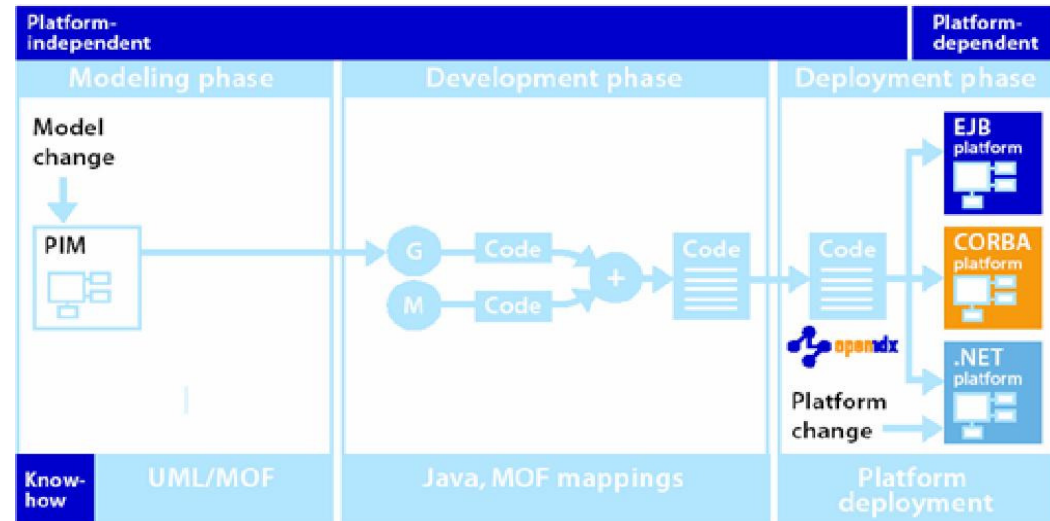
## Deployment

- Komplexer Deploymentdescriptor
- Lightweight openMDX Container falls Zielplattform nicht J2EE

## Entwicklungsprozess

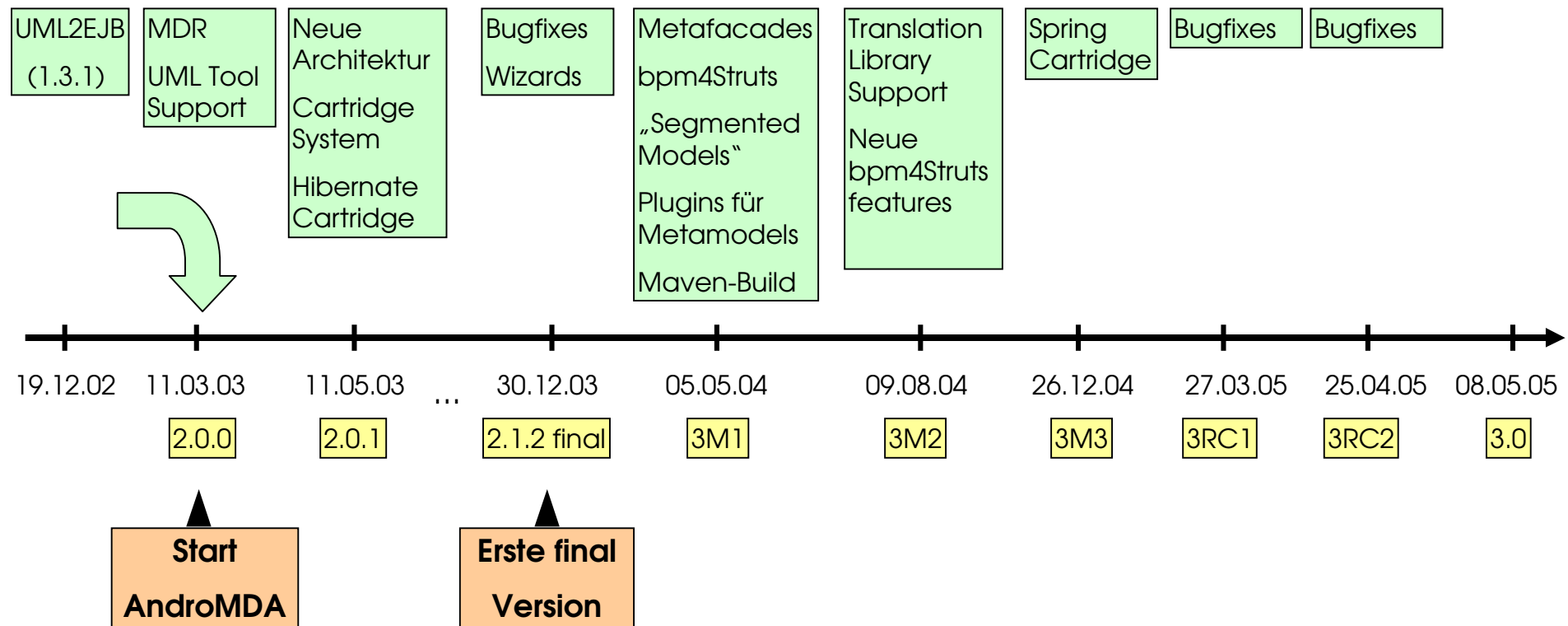
- Spätes Mapping auf die Plattform
  - Logik ist plattform-unabhängig
  - Einfacher Roundtrip (Kein Generator).

Gegenüberstellung Entwicklungsprozesse  
 Oben: OpenMDX  
 Unten: „normale“ MDA Tools  
 (Quelle: openMDX Quickstart)



# Historie der Entwicklung von Open Source MDA Tools

## ■ Am Beispiel AndroMDA:



## ■ Zusammenfassung

- (Fast) alle Open Source MDA Tools setzen auf einen zweistufigen Ansatz (Ausnahme: openMDX)
- Roundtrip Engineering meist recht schwer.
- Dauer für Ausführung des Generators in größeren Projekten recht hoch
- Integration von QVT (Query-View-Transformation) wird spannend

## ■ Ergebnisse

- **JAG** nur für kleine J2EE Anwendungen anwendbar
- **UMT-QVT** ist recht kompliziert (XMI Light, XSLT)
- **openMDX** bietet einen komplett unterschiedlichen Ansatz, der jedoch stark von der Ausführungseine abhängig ist
- **AndroMDA** und **Open Generator Framework** sind derzeit die besten Open Source MDA Werkzeuge
  - **AndroMDA** besticht durch out-of-the-box Funktionalität
  - **Open Generator Framework** kann v.a. durch seine Templatesprache glänzen.

# Abschluss

**JAVA**FORUM2005  
stuttgart

- Fragen?
- Vielen Dank!

