
Methoden des Software-Engineering

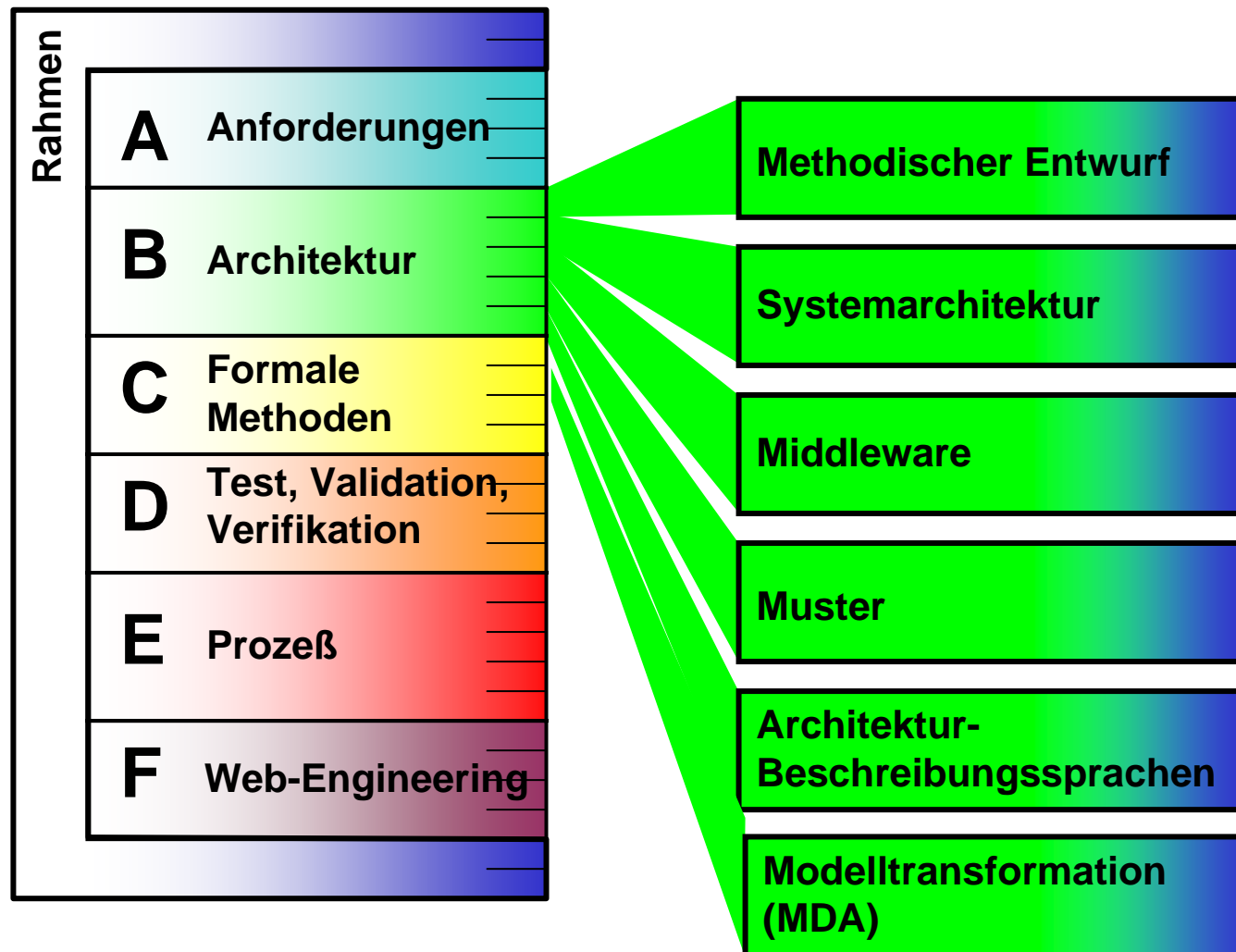
Software Architektur

Martin Wirsing

**Block B, Teil 6: Einführung in Modelltransformation
(Model-driven Architecture)**

WS 2006/07, LMU München

Gliederung Block B Teil 6: Modelltransformation

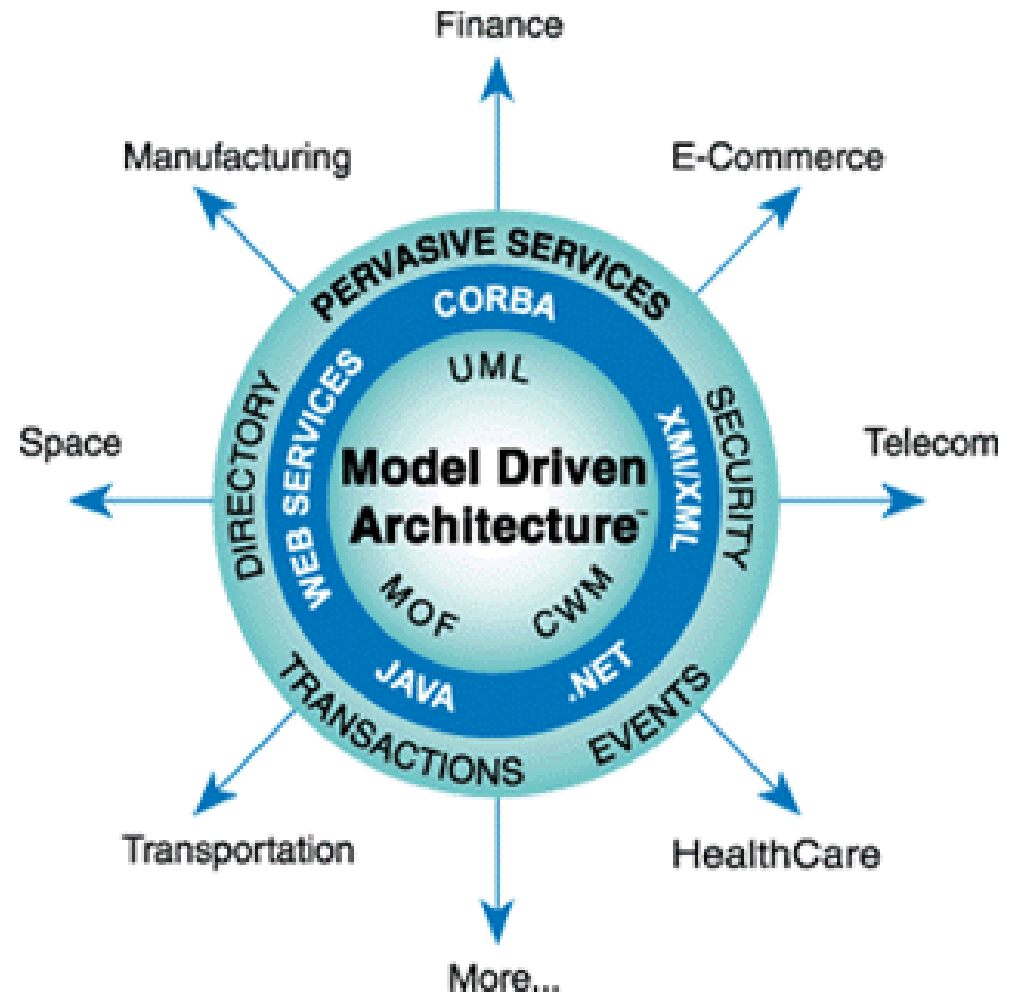


Ziele

- Model-driven Architecture kennen lernen
- Erweiterungsmechanismus von UML verstehen
- Grundkonzepte der Modelltransformation verstehen

MDA

- Fragestellung
 - Geschäftssoftware ist komplexe Software mit hohen Anforderungen an Durchsatz, Transaktionen, Sicherheit, Automatisierung, Anpassbarkeit
 - Es gibt immer mehr Speziallösungen an Middleware
 - CORBA
 - COM, DCOM, MTS
 - C#, .Net
 - Java, EJB
 - XML, SOAP
 - Was ist die nächste beste Middleware?



MDA

- **MDA = Model Driven Architecture**
 - auch: MD (Software/Application) Development, Model-Based Development/Management/Programming
 - Model Driven Engineering, Model Integrated Computing
- **Initiative der OMG (Warenzeichen)**
 - Object Management Group: CORBA, UML, . . .
 - offenes Firmenkonsortium (ca. 800 Firmen)
- **Ziel: Verbesserung des Softwareentwicklungsprozesses**
 - Interoperabilität
 - Portabilität
- **Ansatz:**
 - Verlagerung des Software-Entwicklungsprozesses von der Codeebene auf die Modellebene
 - Wiederverwendbarkeit von Modellen
 - Transformation von Modellen
 - Codeerzeugung aus Modellen

Was kann MDA bringen?

- **Höherer Abstraktionsgrad**
 - Portabilität
 - Interoperabilität
 - Wiederverwendbarkeit

- **Modelle und Modelltransformation**
 - Produktivität
 - Dokumentation und Wartung
 - Spezialisierung

Höherer Abstraktionsgrad

- **Portabilität und Wiederverwendbarkeit**
 - Entwicklung abstrahiert von Zielplattform
 - Technologieabbildung in wiederverwendbaren Transformationen
 - Neue Technologie => neue Transformation
- **Interoperabilität**
 - Systeme sind plattformübergreifend
 - Informationsübertragung zwischen Plattformen durch *Brücken*
 - Nebenprodukt von Modelltransformation

Modelle und Modelltransformation

- **Produktivität**
 - Jede Phase der Entwicklung leistet direkten Beitrag zum Produkt, nicht nur die Implementierung.
- **Dokumentation und Wartung**
 - Änderungen durch Änderung der Modelle
 - Modelle sind Dokumentation) Konsistenz
 - Trennung von Verantwortlichkeit
 - Handhabbarkeit von Technologiewandel
- **Spezialisierung**
 - Geschäftsprozesse
 - Technologien

Ein Modellbegriff (nach Herbert Stachowiak, 1973)

- **Repräsentation**
 - Ein Modell ist Repräsentation eines Original-Objekts.
 - **Abstraktion**
 - Ein Modell muss nicht alle Eigenschaften des Original-Objekts erfassen.
 - **Pragmatismus**
 - Ein Modell ist immer zweckorientiert.
- ⇒ Modellierung erzeugt eine Repräsentation, die nur die für einen bestimmten Zweck relevanten Eigenschaften beinhaltet.

Formale Modelle

- **Modelle, die in einer formalen Sprache verfasst sind**
 - Textuell: definiert durch Grammatik, BNF, o.ä.
 - Grafisch: definiert durch *Metamodell*
 - Welche Modellierungselemente?
 - Welche Kombinationen?
 - Welche Modifikationen?
- **Modelle, die eine formale Semantik besitzen**
 - Beispiel: logische Formel \Rightarrow Wahrheitswert
 - Beispiel: kontextfreie Grammatik \Rightarrow Sprache
 - Beispiel: Programm \Rightarrow Programmausführung

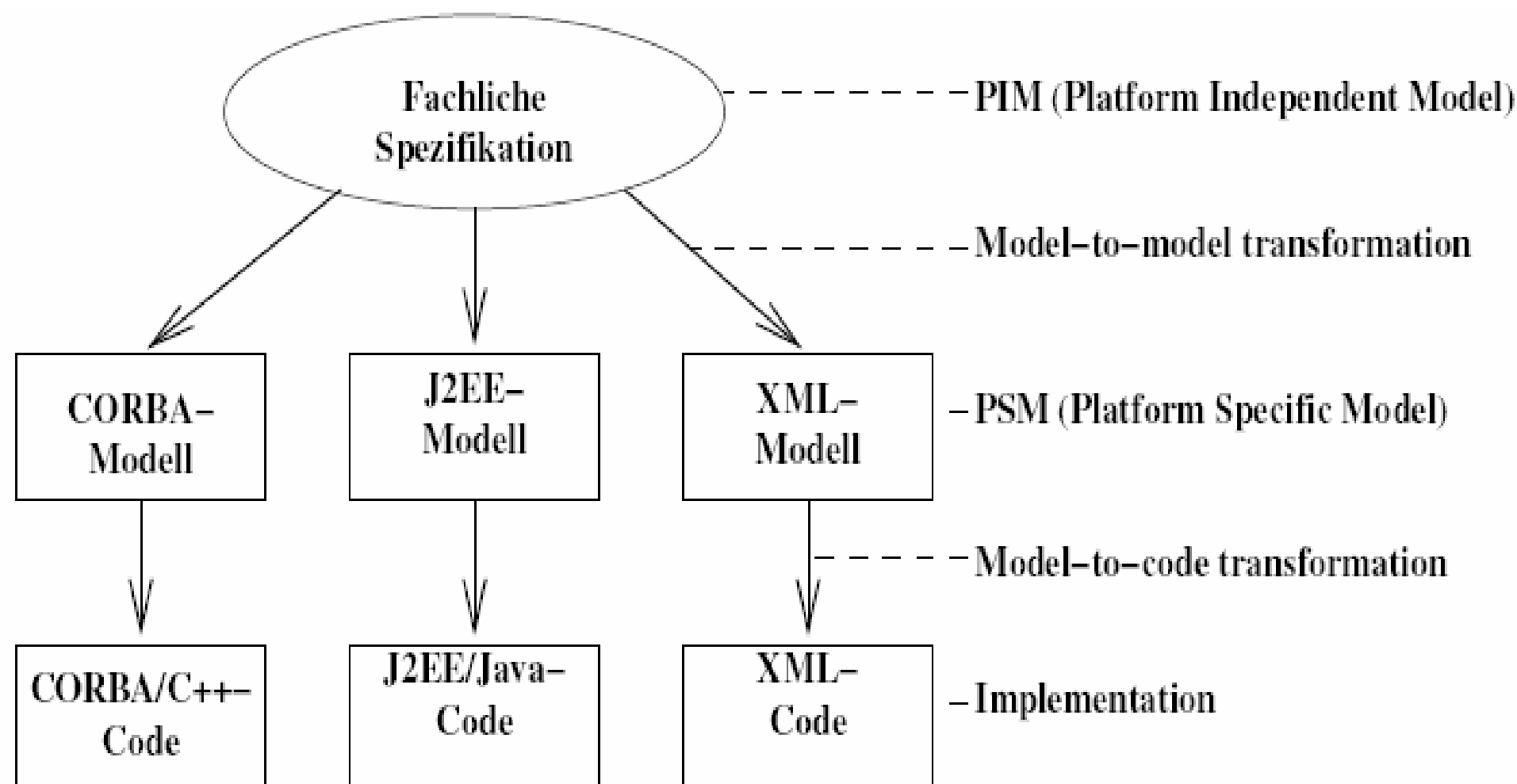
Warum formale Modelle?

- **Modelleditor**
 - Programm zur Manipulation von Modellen
 - benötigt formale Definition
- **Modelltransformation**
 - Überführung eines Modells in ein oder mehrere Zielmodelle
 - benötigt formale Definition ggf. formale Semantik
- **Modellverifikation**
 - Eigenschaften: Schnittstellen, Zeitverhalten, . . .
 - Verhältnis zwischen Modell und Original
 - benötigt formale Definition und formale Semantik

Arten von Modellen

- Positionierung im Softwareentwicklungsprozess:
 - Analyse, Definition, Entwurf, Implementierung
- Detaillierungsgrad
- Geschäftsmodell oder Softwaremodell
- Strukturell oder dynamisch:
 - Klassendiagramm, Aktivitätsdiagramm
- Plattformabhängig oder –unabhängig

Modelle in MDA



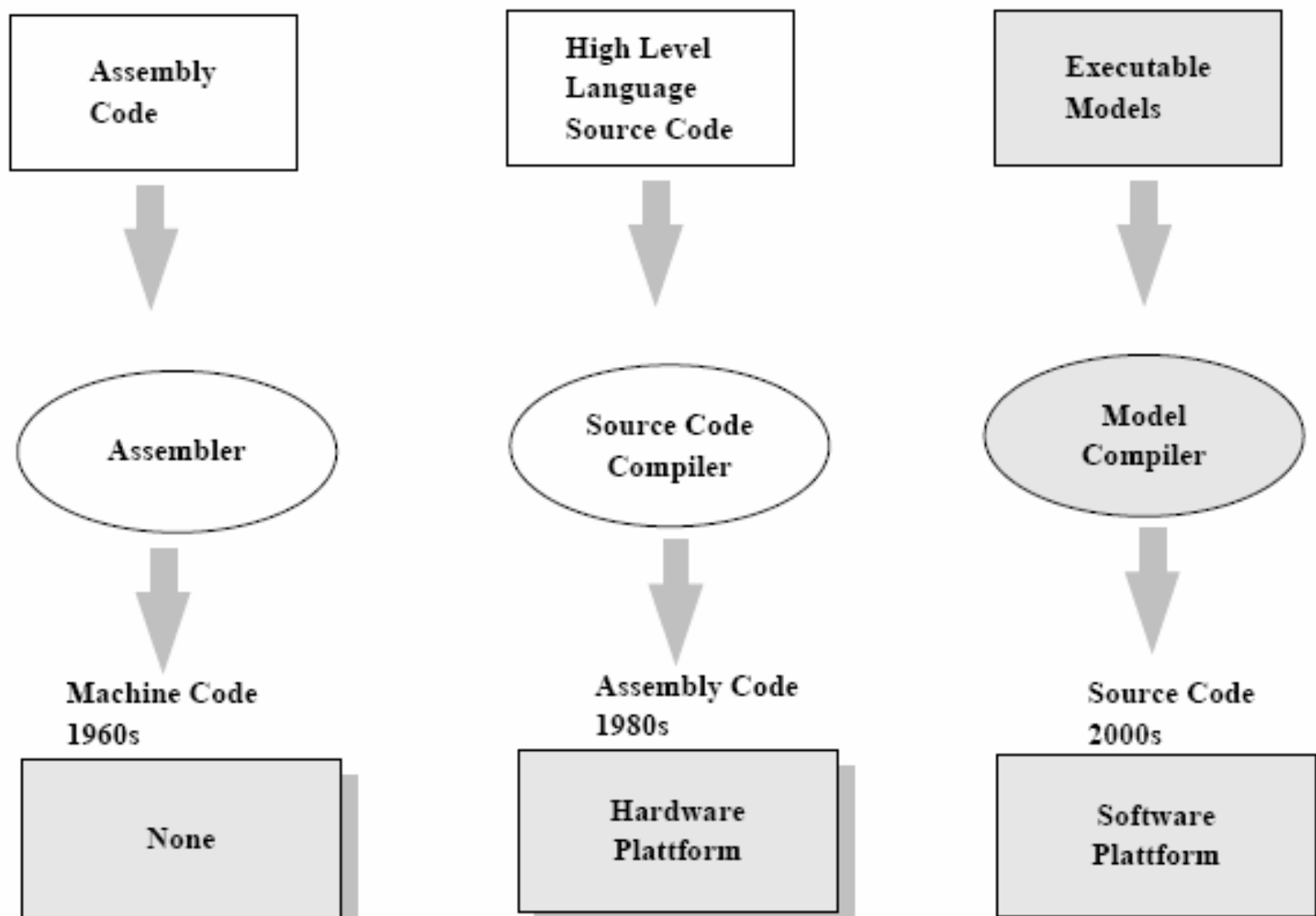
Modelle in MDA

- **Plattform-unabhängiges Modell (PIM) vs PlattformSpezifisches Modell (PSM)**
 - Relative Konzepte
 - Übergang fließend
 - Mehrere Modellebenen und Transformationsschritte möglich
 - Rücktransformation PSM => PIM kaum automatisierbar
- **Transformation**
 - Code ist ultimatives Modell (PSM)
 - Model-to-Code ist Spezialfall

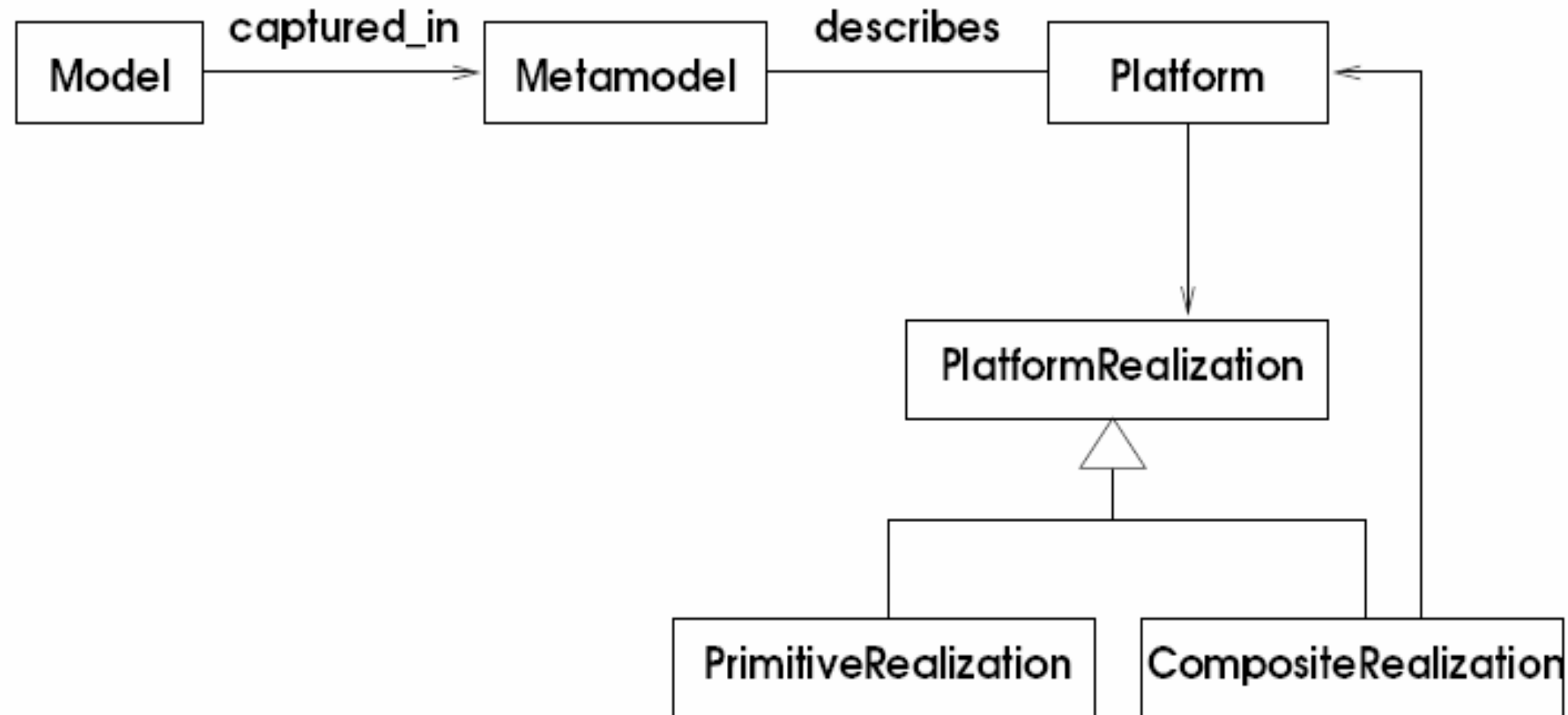
Plattform

- **Plattform**
 - kann auf verschiedenen Abstraktionsebenen existieren
 - besitzt typischerweise
 - eine Programmierschnittstelle, API
 - Virtuelle Maschine
 - Und stellt verschiedene Dienste zur Verfügung
- **Beispiele**
 - Hardwareplattform
 - Betriebssystem (Softwareplattform)
 - Java VM (Softwareplattform)
 - EJB (Komponentenplattform)
 - CORBA, Webservices, . . . (Middleware)
 - Anwendungsarchitektur, DSL (Domain Specific Language)

Plattform: Beispiele



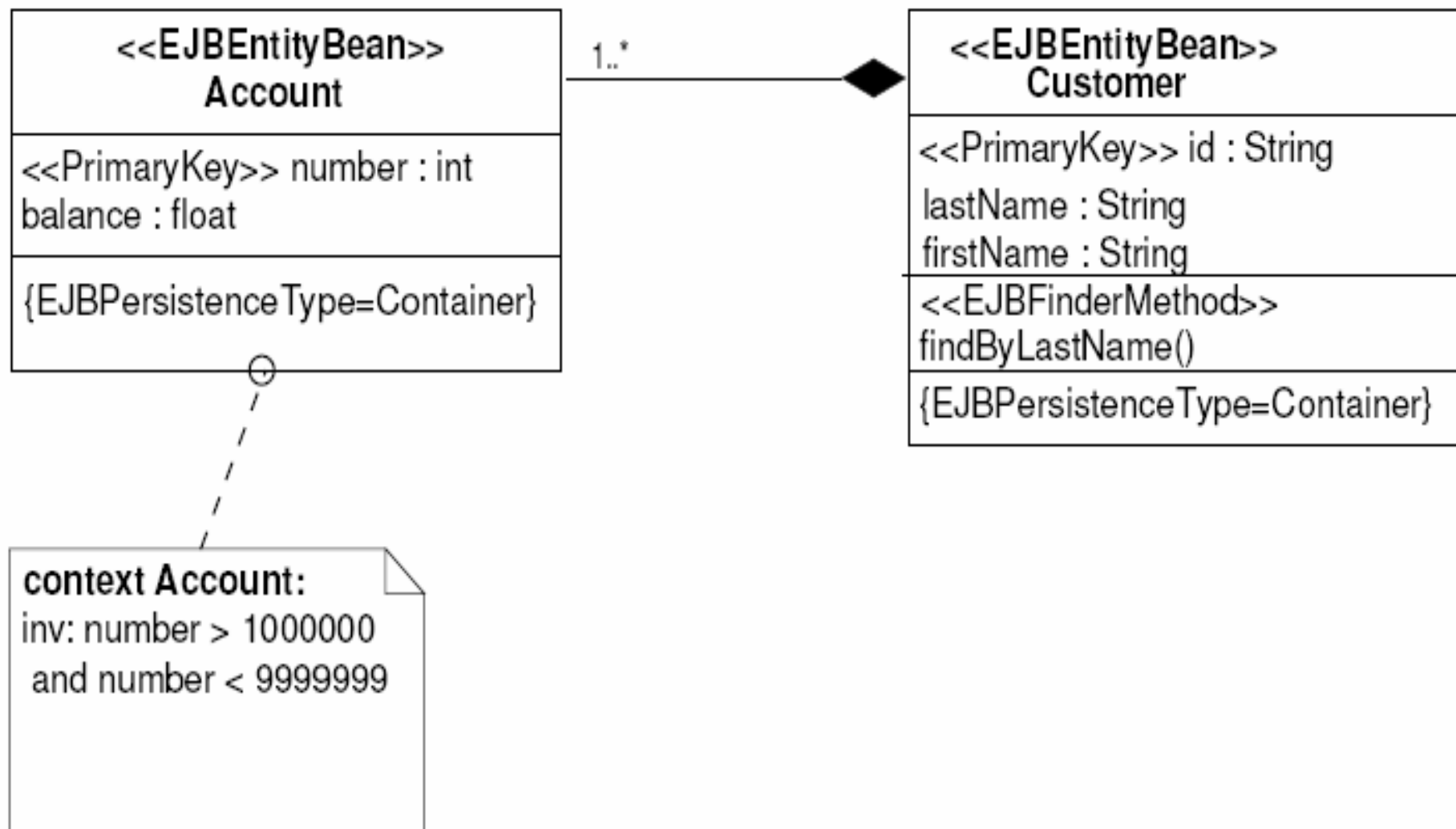
Plattform: Metamodell



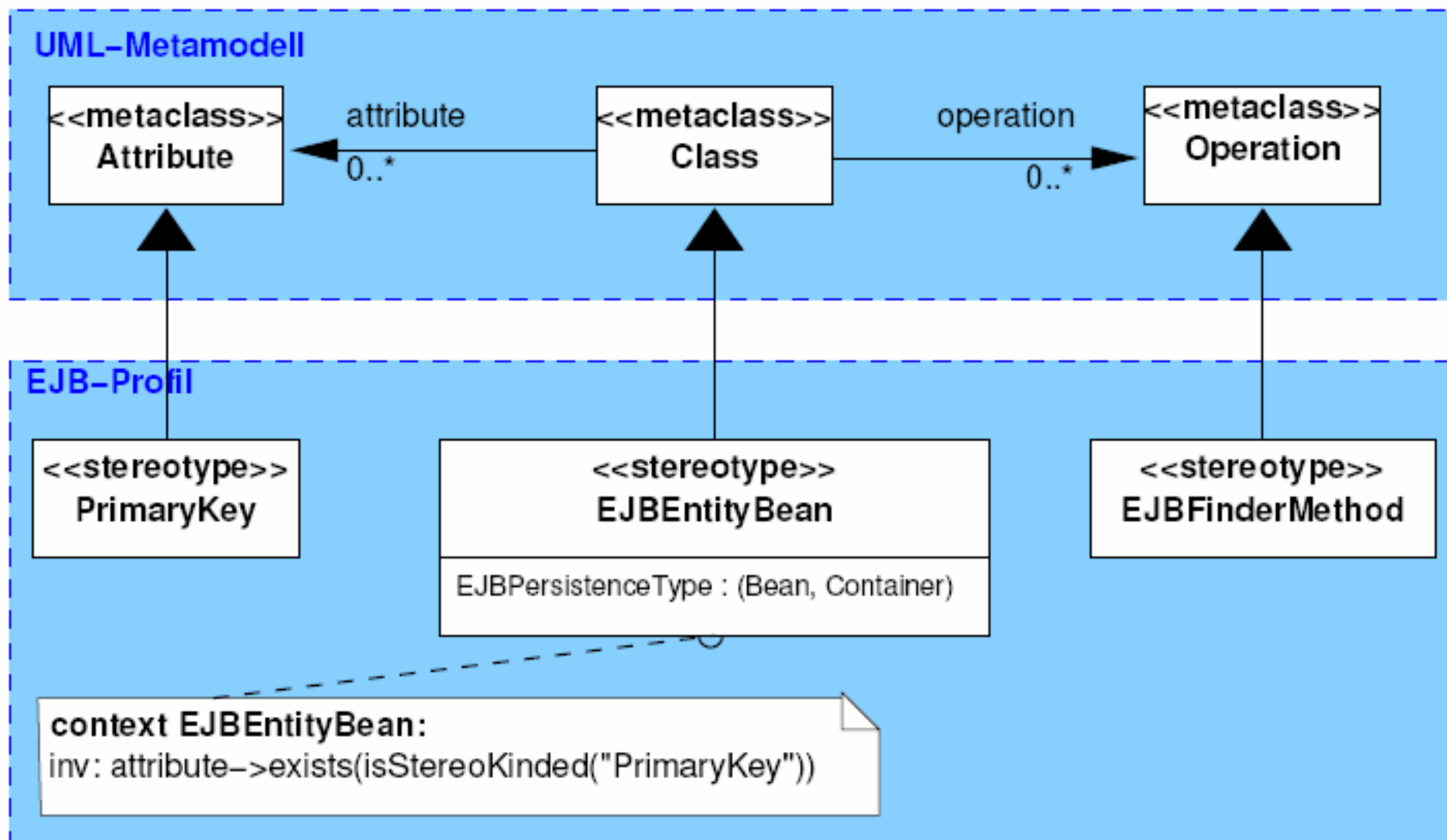
UML-Profil

- Ein UML-Profil ist ein Erweiterungsmechanismus von UML
- **Bausteine**
 - Klassen und Assoziationen
 - **Stereotypen** (verändern die Bedeutung eines Modellierungselements)
 - **Tagged Values** (definieren Attribute eines Modellierungselements)
 - **Constraints** (beschränken die möglichen Instanzen)
- Ein UML-Profil wird definiert als **Erweiterung des UML-Metamodells**
 - Metamodell definiert Modellierungselemente
 - Modell ist Instanz eines Metamodells
 - Metamodell definiert Vokabular zur Beschreibung einer Plattform

Beispiel: Verwendung eines UML-Profiles



Beispiel: Ein UML-Profil: Definition



Modelltransformationen

- Eine **Modelltransformation** ist eine Abbildung von Modell nach Modell.
- **Formale Definition** erforderlich für Automatisierung
- **Transformationssprachen**
 - **QVT (Query View Transformation)** Standardisierung in Arbeit, 23 Vorschläge
 - **UMLX**
 - Graphische Sprache zur Spezifikation von Transformationen von UML-Objektmodellen
 - **Graphtransformationen**
 - TIGER (TU Berlin, University of Leicester)
 - VIATRA2 (Universität Budapest)
- **Werkzeuge**
 - Transformationen basierend auf Metamodell
 - Codeerzeugung durch Muster
 - proprietäre Transformationssprachen (Skriptsprachen)
- Bislang fehlende Interoperabilität der Werkzeuge

Beispieltransformation in UMLX

UMLX:

- Transformationen basieren auf Pattern Matching eines Objektdiagramms
- Klassendiagramme werden als Objektdiagramme (des Metamodells) betrachtet
- Transformationen werden durch Erweiterung der Klassendiagrammnotation spezifiziert
 - Input
 - Output
 - Zu löschende Elemente
 - Neue Elemente

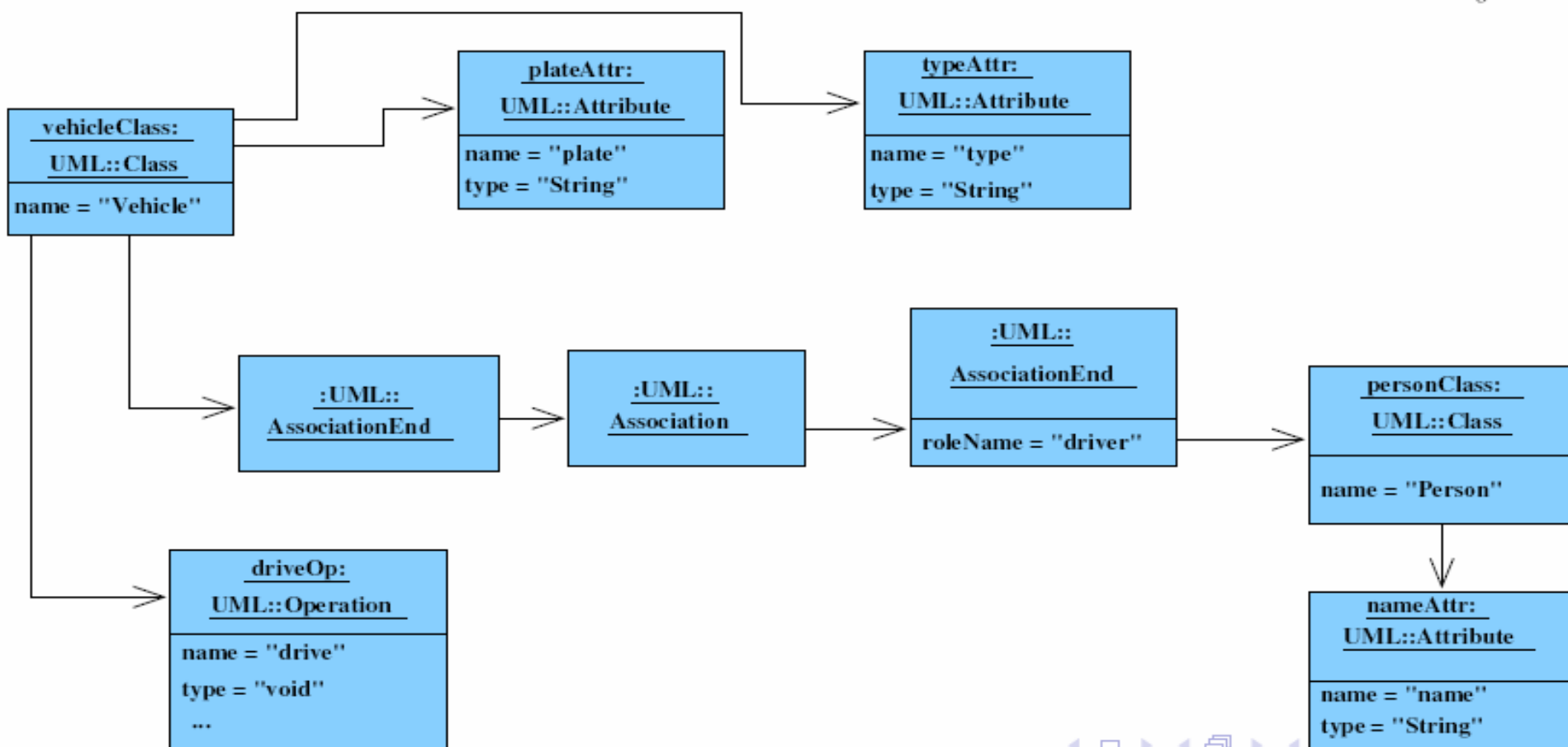
Beispieltransformation in UMLX: EJB

(Klassendiagramm als Objektdiagramm)



Class Diagram

Instance Diagram

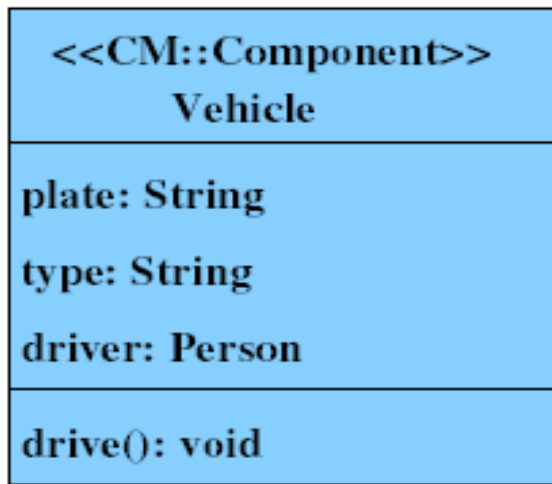


Beispieltransformation in UMLX: EJB

Anwendung auf einfaches Klassendiagramm



Eingabeklassendiagramm



Ergebnisklassendiagramm

Zusammenfassung

- Ziel der Model Driven Architecture (MDA) ist eine Verbesserung des Softwareentwicklungsprozesses durch Verlagerung der Entwicklungsaktivitäten von der Codeebene auf die Modellebene. Wichtigstes Hilfsmittel sind Modelltransformationen.
- Eine Modelltransformation ist eine Abbildung von Modell nach Modell.
- Durch Modelltransformation werden plattform-unabhängige Modelle in plattform-abhängige Modelle und Code transformiert.
- Heutige Transformationssprachen sind der angehende Standard QVT (Query View Transformation), UMLX und Graphtransformationssprachen wie TIGER oder VIATRA2.
- Eine Transformationsregel (für UML-Modelle) wird typischerweise auf UML-Metamodellebene formuliert und verwendet häufig UML-Profile – den Erweiterungsmechanismus von UML.

Literatur

- Anneke Kleppe, Jos Warmer, Wim Bast: *MDA Explained. The Model Driven Architecture: Practice and Promise*. Addison-Wesley Professional, 2003
- VIATRA2 (Visual Automated model TRAnsformations) framework <http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmt-home/subprojects/VIATRA2/index.html>, 2006.
- TIGER Homepages: TIGER Generator and Designer: <http://tfs.cs.tu-berlin.de/tigerprj/>, TIGER EMF Transformer: <http://tfs.cs.tu-berlin.de/emftrans/>, 2006.
- Edward D. Willink. UMLX : A graphical transformation language for MDA. In Arend Rensink, editor, *Model Driven Architecture: Foundations and Applications*, pages 13– 24. University of Twente, 2003.
- SmartQVT, (ein erstes QVT-Werkzeug) <http://universalis.elibel.tm.fr/qvt/>
- Eine Liste von MDA-Werkzeugen: http://www.modelbased.net/mda_tools.html