Model Driven Architecture
Einführung

Prof. Dr. Peter Thiemann

Universität Freiburg

03.05.2006

Was ist 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 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 $\Rightarrow$ 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 $\Rightarrow$ 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.

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

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

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

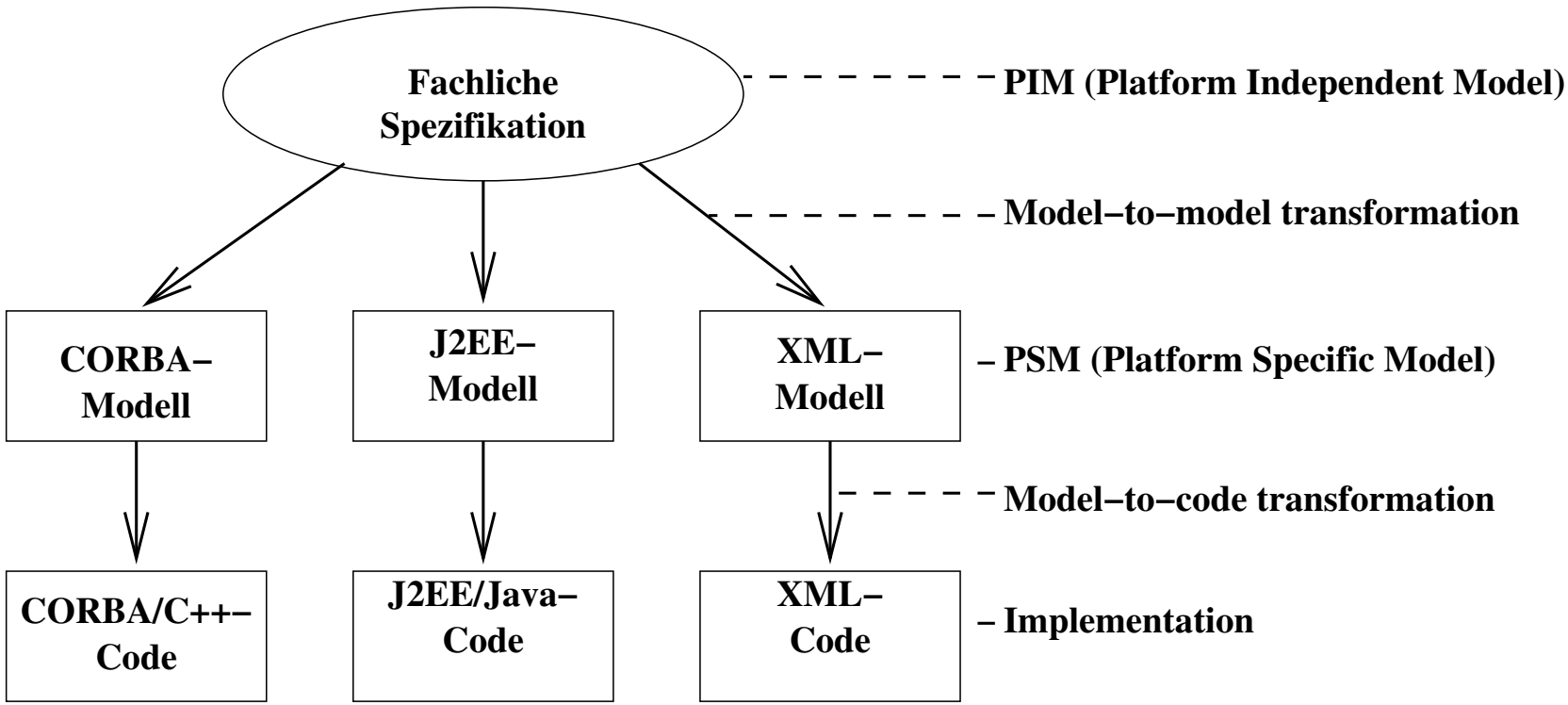
Modellverifikation

- 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/2

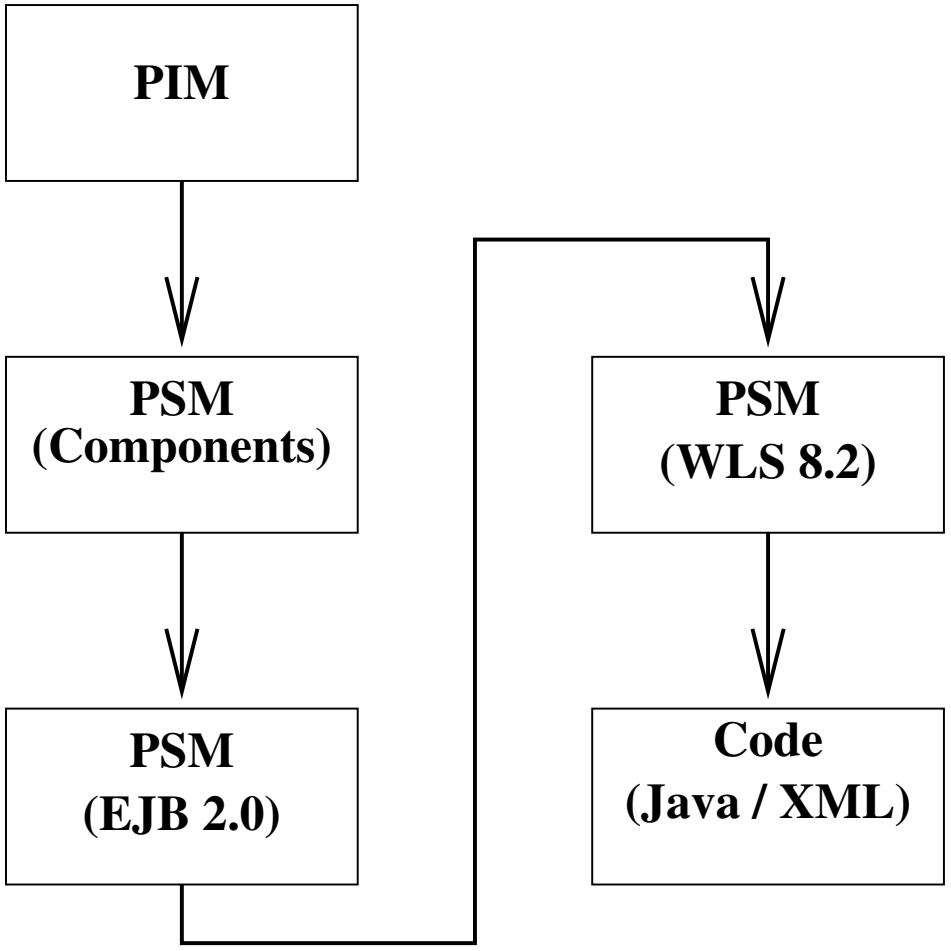
PIM vs PSM

- Relative Konzepte
- Übergang fließend
- Mehrere Modellebenen und Transformationsschritte möglich
- Rücktransformation PSM $\Rightarrow$ PIM kaum automatisierbar

Transformation

- Code ist ultimatives Modell (PSM)
- Model-to-Code ist Spezialfall

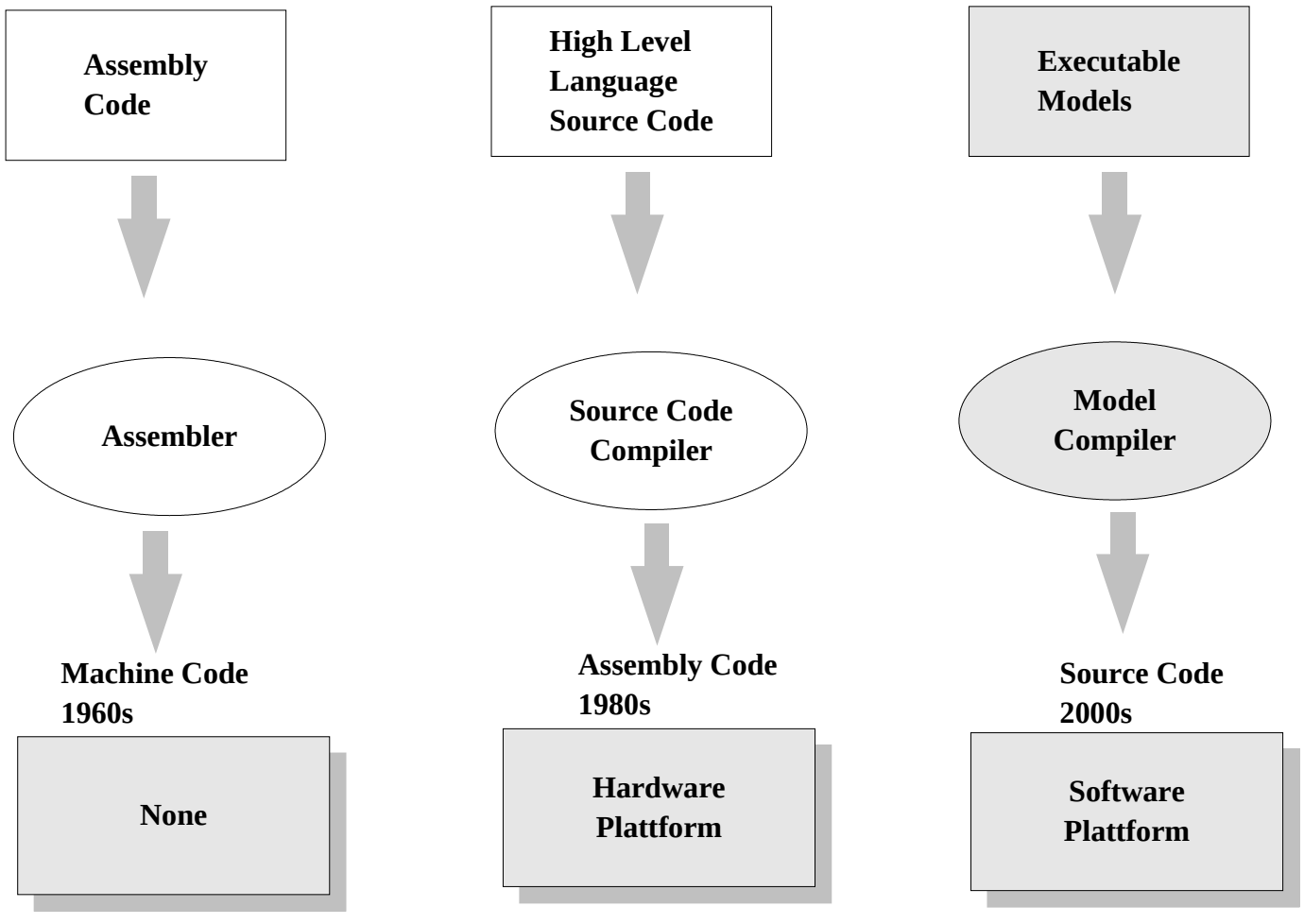
Modelle und Transformationen



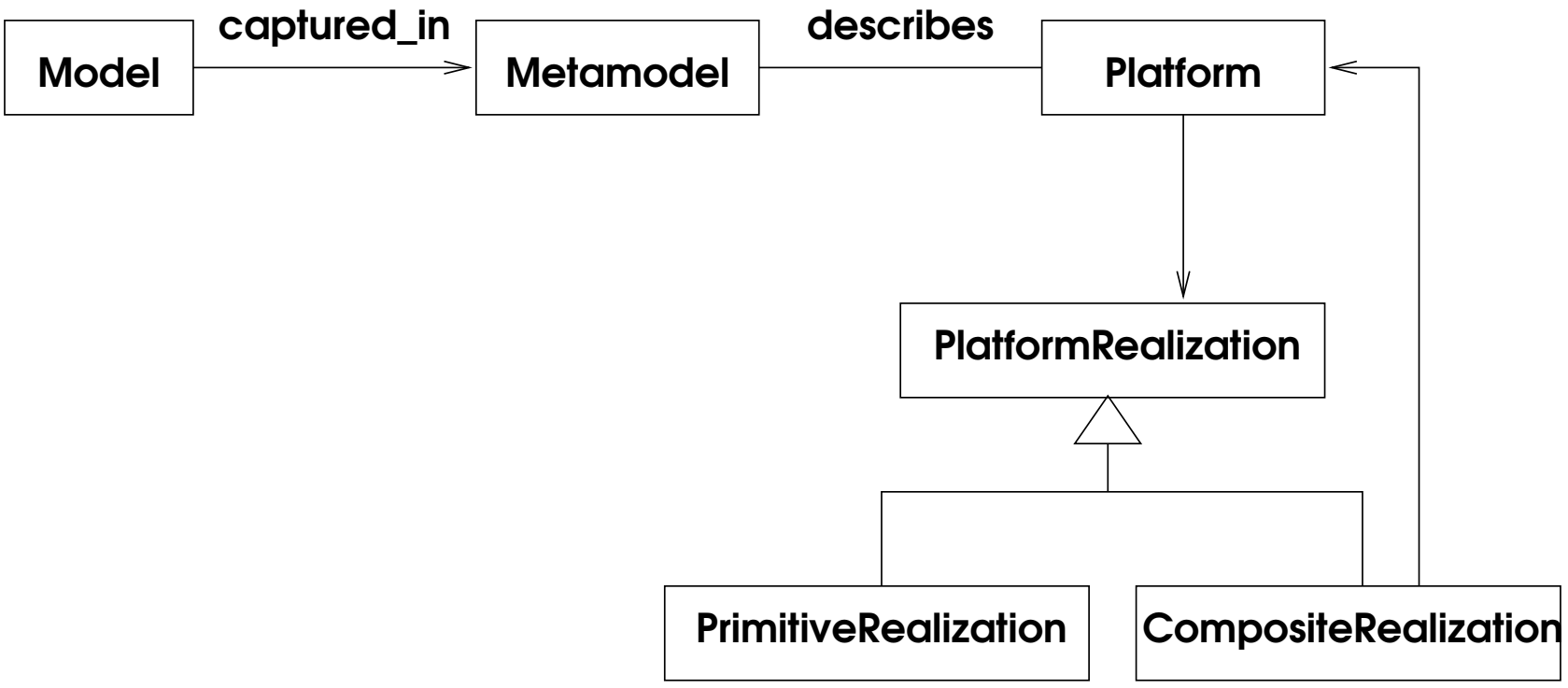
Plattform

- Programmierschnittstelle, API
- Virtuelle Maschine
- Stellt verschiedene Dienste zur Verfügung
- Beispiele
  - Hardwareplattform
  - Betriebssystem $\Rightarrow$ Softwareplattform
  - Java VM $\Rightarrow$ Softwareplattform
  - EJB $\Rightarrow$ Komponentenplattform
  - CORBA, Webservices, ...
  - Anwendungsarchitektur, DSL (Domain Specific Language)

Plattformen im Beispiel



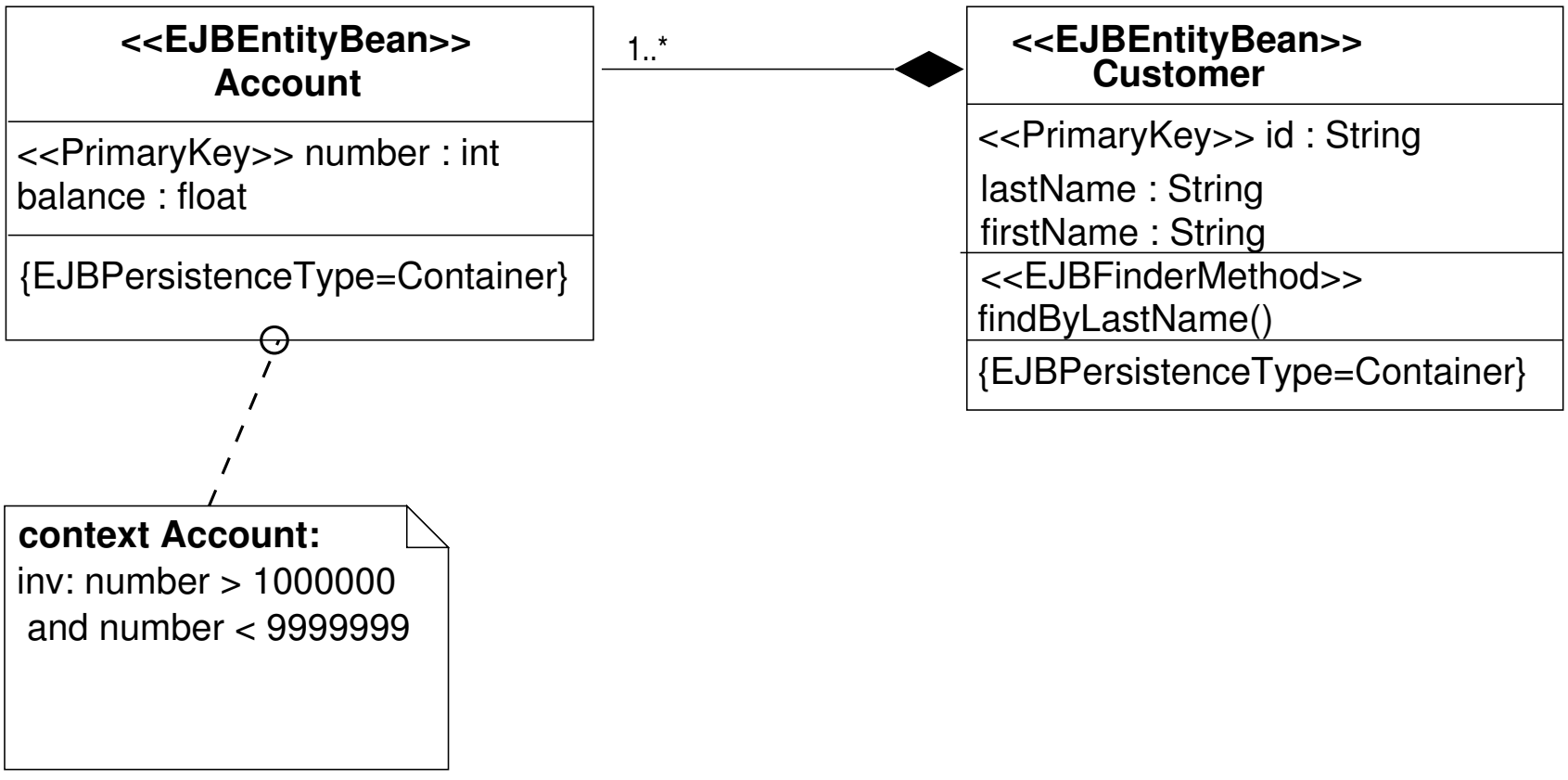
Plattformen, schematisch



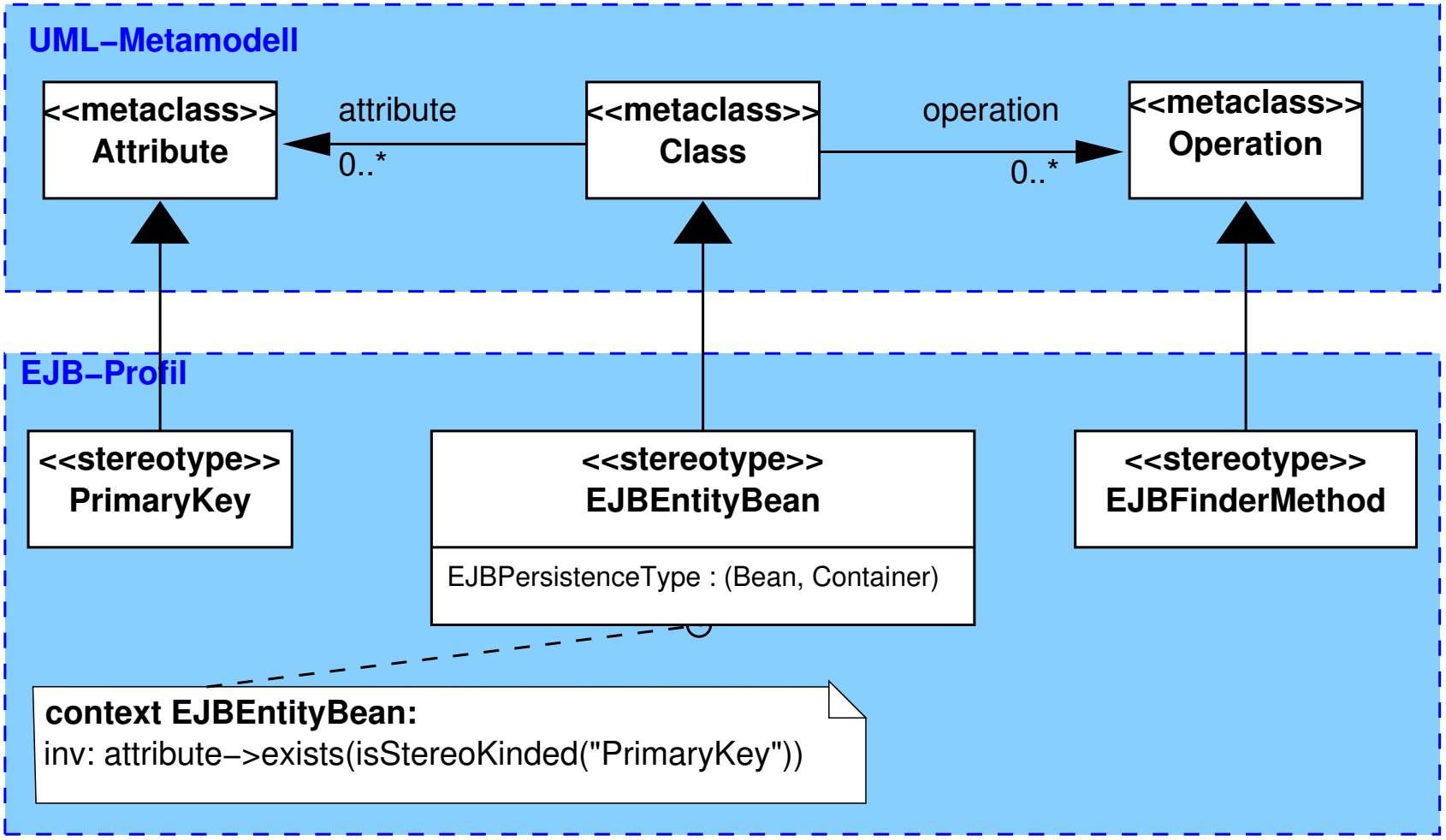
UML-Profile

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

Verwendung eines UML-Profils



Definition eines UML-Profils



Transformationen

- Abbildung von Modell nach Modell
- Formale Definition erforderlich für Automatisierung
- z.Z. keine standardisierte Transformationssprache
  QVT (Query View Transformation) in Arbeit, 23 Vorschläge
- Werkzeuge
  - Transformationen basierend auf Metamodell
  - Codeerzeugung durch Muster
  - proprietäre Transformationssprachen (Skriptsprachen)
- Bislang fehlende Interoperabilität der Werkzeuge

Nächste Schritte

- Grundkonzepte von UML2
  - Klassendiagramme
  - Aktivitätsdiagramme
  - OCL
- Metamodellierung