Datenmodelle und Datenmodellierung

Modellbildung in der Entwicklung

Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München

SS 2007

Inhalte

Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
   1. Datenmodellierung
      1. Grundlagen
      2. Datenstrukturen
      3. Datenrelationen
      4. Objektmodellierung
      5. Metamodellierung
   2. Zustandsmaschinen
   3. Abläufe und Prozesse
   4. Architekturen
   5. Schnittstellen
   6. Modellintegration
3. Anwendung
4. Modellgetriebene Entwicklung
5. Domänenspezifische Modellierung

Datenmodellierung

Datenmodellierung:

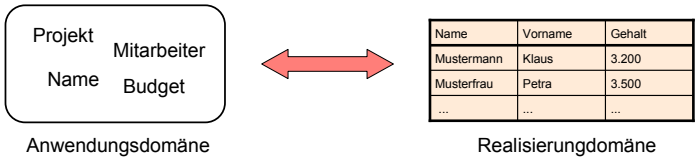
- Kontext: Informationssysteme
  - Betriebliche Informationssysteme
  - Eingebettete Systeme
  - verarbeiten Daten (strukturierte Informationen)
- Ziel: Beschreibung der für die Erstellung eines Informationssystems relevanten Daten einschließlich ihrer Strukturen und Beziehungen

Datenmodelle

Datenmodelle:

- Beschreiben informationelle Aspekte der Anwendung
  - Konzeptuelles Datenmodell
  - Technisches Datenmodell
- Charakterisieren die Struktur der Daten
  - Aufbau der Daten
  - Beziehungen zwischen Daten
- Charakterisieren die Verwendung der Daten
  - Definition Operationen auf Daten
  - Definition Nutzung von Daten

Bildung von Datenmodellen



Anwendungsdomäne

Name	Vorname	Gehalt
Mustermann	Klaus	3.200
Musterfrau	Petra	3.500
...	...	...

Realisierungdomäne

- Herausarbeiten der wesentlichen informationellen Merkmale, die den Konzepten der Problemdomäne entsprechen
  - Anwendungskonzepte (z.B. Mitarbeiter, Projekt)
  - Realisierungskonzepte (z.B. Liste, Menge)
- Zusammenfügen von Elementen zu komplexen Strukturen, die die Zusammenhänge der Problemdomäne wiedergeben
- Festlegung der zulässigen Operationen, die Aktionen und Ereignissen der Problemdomäne entsprechen.

Varianten der Datenmodellierung

Unterschiedliche Modellierungsansätze:

- Abhängig von
  - Modellierungsgegenstand („Was wird modelliert“)
  - Modelleinsatz („Wozu wird modelliert“)
  - Modellierungsansatz („Wie wird modelliert“)
- Variante: Datenstrukturmodelle
  - Motivation: Modellierung von Datentypen und ihrer Operationen
- Variante: Datenrelationsmodelle
  - Motivation: Modellierung von Daten und ihrer Beziehungen
- Variante: Objektorientierte Modelle
  - Motivation: Modellierung von Objektwelten

Komplexität: Aufbau vs. Nutzung

Unterschiedliche Komplexitätsquellen:

- Schwerpunkt: Aufbau der Daten
  - Komplexe/spezifische Struktur der Daten
  - Einfache/standardisierte Nutzung der Daten
  - Fokus: Strukturdefinition
  - Beispiel: Definition einer Datenbank
- Schwerpunkt: Nutzung der Daten
  - Einfache/standardisierte Struktur der Daten
  - Komplexe/spezifische Nutzung der Daten
  - Fokus: Operationsspezifikation
  - Beispiel: Definition einer Applikationsbibliothek

Grundkonzepte der Strukturdefinition

Allgemeine Konzepte zur Strukturdefinition:

- Klassifikation und Instantiierung:
  Festlegung der gemeinsamen und individuellen Aspekte einer Menge von Datenelementen
- Aggregation und Dekomposition:
  Festlegung des Aufbaus und der Zerlegung von Elementen einer Datenmenge
- Generalisierung und Spezialisierung:
  Festlegung von Über- und Untermengen von Mengen von Datenelementen
- Assoziation und Referenzierung:
  Festlegung von Beziehungen zwischen Mengen von Datenelementen

Grundkonzepte der Operationsdefinition

Allgemeine Konzepte zur Operationsdefinition

- Modularisierung:
  Trennung zwischen der Definition und der Nutzung einer Rechenstruktur
- Kapselung:
  Verbergen des internen Aufbaus einer Rechenstruktur an der Nutzungsschnittstelle
- Parametrierung und Polymorphie:
  Generalisierung von Rechnenstrukturen durch Rechenstrukturparameter

Variante: Datenstrukturen

Datenstrukturen:

- Anwendungskontext: Modellierung von technischen Datenmodellen (Rechenstrukturen)
- Anwendungsschwerpunkt: Charakterisierung von wiederverwendbaren Datentypen und ihrer Operationen
- Modellierungsansatz: Abstrakte Datentypen
- Konzepte: Signatur, Sorte, Operation, Semantik

Abstrakter Datentyp

Abstrakter Datentyp:

- Notation: Algebraische Spezifikation
  - Basis John Guttag, Abstract Data Type, 1977
  - Div. Formalisierungen, u.a. Ehrig/Mahr, 1985
- Zweck: Implementierungsunabhängige Spezifikation von Datenwerten und zugehörigen Operationen
  - Modularisierung
  - Kapselung
  - Parametrierung und Polymorphie
- Anwendungkontext: Spezifikation von (technischen) Datentypen

Abstrakter Datentyp

Abstrakter Datentyp (ADT):

- Zweck: Festlegung des Wertebereichs einer Datenstruktur und zulässiger Operationen durch eine Spezifikation (Σ, E)
- Signatur Σ: Syntaktische Festlegung (S, OP) von Wertebereichen und Operationen
  - S: Menge von Sortenbezeichnern
    - Eingeführte Sorten
    - Verwendete Basissorten
  - OP: Menge von Operationsbezeichnern
    - Konstruktoren
    - Selektoren
- Axiomensystem E: Semantische Festlegung von
  - Menge von Axiomen/Gleichungen über (S,OP)-Termen
  - Beziehungen zwischen Operationsanwendungen

Abstrakter Datentyp Multimenge

```
spec MultiSet is
sorts MSet, T, Bool
ops  emptySet: → MSet
       insert: T × MSet → MSet
       delete: T × MSet → MSet
       isEmpty: MSet → Bool
       isElement: MSet × T → Bool
axioms
     isEmpty(emptySet) = True
     insert(x,insert(y,s)) =
           insert(y,insert(x,s))
     isElement(x,emptySet) = False
     eq(x,y) ⇒ isElement(x,insert(y,s)) =
                    True
     neq(x,y) ⇒
           isElement(x,(insert(y,s)) =
           isElement(x,s)
     delete(x,emptySet) = emptySet
     eq(x,y) ⇒ delete(x,insert(y,s)) = s
     neq(x,y) ⇒ delete(x,(insert(y,s)) =
           delete(x,s)
```


Abstraktion und Kapselung

```
spec MultiSet is
sorts MSet, T, Bool
ops    emptySet: → MSet
         insert: T × MSet → MSet
         delete: T × MSet → MSet
         isEmpty: MSet → Bool
         isElement: MSet × T →
       Bool
axioms
       isEmpty(emptySet) = True
       insert(x,insert(y,s)) =
             insert(y,insert(x,s))
```

```
isElement(x,emptySet) = False
eq(x,y) ⇒
isElement(x,insert(y,s)) =
          True
neq(x,y) ⇒

isElement(x,(insert(y,s)) =
          isElement(x,s)
 delete(x,emptySet) =
emptySet
eq(x,y) ⇒ delete(x,insert(y,s))
= s
neq(x,y) ⇒
delete(x,(insert(y,s)) =
          delete(x,s)
```

Abstraktion und Kapselung:

- Abstrakte Datentypen spezifizieren die Schnittstelle (Sorten und Operationen)
- Abstrakte Datentypen verbergen die Implementierung durch die Nutzungsspezifikation (Axiome)

Parametrierung und Polymorphie

```
spec MultiSet is
sorts MSet, Int, Bool
ops    emptySet: → MSet
         insert: Int × MSet → MSet
         delete: Int × MSet →
       MSet
         isEmpty: MSet → Bool
         isElement: MSet × Int →
       Bool
axioms
       isEmpty(emptySet) = True
       insert(x,insert(y,s)) =
             insert(y,insert(x,s))
```

```
isElement(x,emptySet) = False
eq(x,y) ⇒
isElement(x,insert(y,s)) =
          True
neq(x,y) ⇒

isElement(x,(insert(y,s)) =
          isElement(x,s)
 delete(x,emptySet) =
emptySet
eq(x,y) ⇒ delete(x,insert(y,s))
= s
neq(x,y) ⇒
delete(x,(insert(y,s)) =
          delete(x,s)
```

Parametrierung:

- Abstrakte Datentypen nutzen Parametersorten für die Definition parametrierter Typen

Polymorphie:

- Abstrakte Datentypen unterstützen Polymorphie für die Instantiierung parametrierter Typen

Variante: Datenrelationen

Datenrelationen

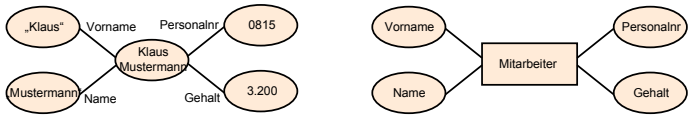
- Anwendungskontext: Modellierung von konzeptuellen Datenmodellen (fachliches Datenmodell)
- Anwendungsschwerpunkt: Charakterisierung der Zusammenhänge fachlicher Informationen
- Modellierungsansatz: Entity-Relationship-Modellierung
- Konzepte: Entität, Relation, Kardinalität, Aggregation, Attribut

Entity-Relationship-Modellierung

Entitiy-Relationship-Modellierung

- Notation: Entity-Relationship-Diagramme
  - Basis: P.P.S. Chen (vgl. P.P.S. Chen. "The Entity Relationship Model - Toward a Unified View of Data.)
  - Erweiterungen: Erweiteretes E/R-Modell, Strukturiertes E/R-Modell
- Zweck: Abstrakte Modellierung der Struktur der Informationsmodells
  - Klassifikation und Instantiierung
  - Aggregation und Dekomposition
  - Assoziation und Referenzierung
- Anwendungkontext: Logische Datenmodell im Datenbankentwurf

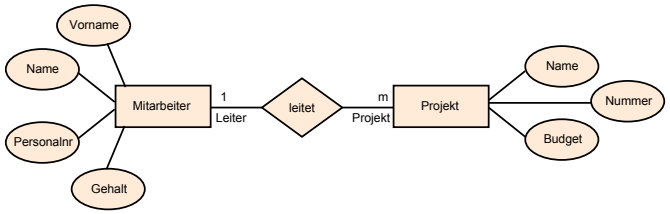
Entität



Entität: Unterscheidbare Einheit mit Eigenschaften

- **Identität**: Jedes Objekt ist eindeutig identifizierbar
- **Typ**: Jedes Objekt besitzt einen unveränderlichen Typ zur Charakterisierung seiner Eigenschaften
- **Eigenschaften**: Jede Entität besitzt Eigenschaften in Form von Attributen mit zugeordneten Werten

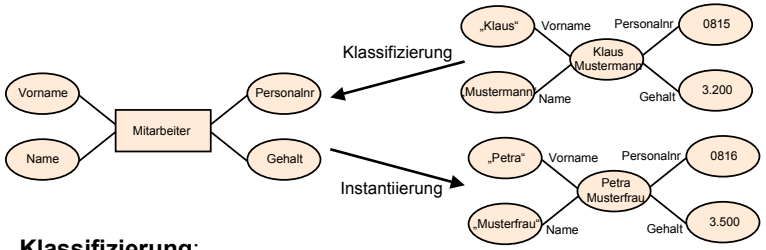
Relation



Relation: Bezeihung zwischen Entitäten

- **Rollen**: Eine Relation definiert die teilnehmenden Objekte
- **Multiplizitäten**: Eine Relation beschränkt die Anzahl zusammengehöriger Objekte (1:1, 1:n,m:1, m:n)

Klassifikation und Instantiierung



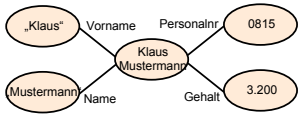
Klassifizierung:

- Ein Entitätstyp beschreibt eine Menge strukturell gleichartiger Entitäten

Instantiierung:

- Jede Entität ist (unveränderlich) Element eines Entitätstyps
- Entitäten eines Typs unterscheiden sich nur in ihren Werten

Aggregation und Dekomposition



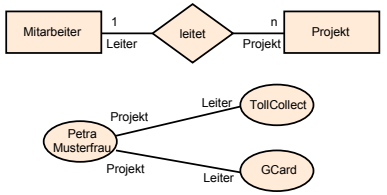
Aggregation:

- Eine Entität fasst Attribute zusammen

Dekomposition:

- Attribute erlauben Referenzierung der Attributwerte

Assoziation und Referenzierung



Beziehung:

- Relationen definieren Beziehungen zwischen Objekten
- Multiplizitäten beschränken Kardinalität der Beziehungen

Referenzierung:

- Rollennamen identifizieren die einem Objekt verwandten Objekte
- Rollenname entsprechen abgeleiteten Attributen

Variante: Objektmodellierung

Objektmodellierung

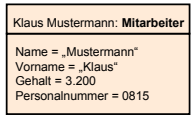
- Anwendungskontext: Modellierung von konzeptuellen und technischen Datenmodellen
- Anwendungsschwerpunkt: Modellierung der Zusammenhänge fachlicher Informationen sowie ihrer technischen Umsetzung
- Modellierungsansatz: Objektorientierte Modellierung
- Konzepte: Objekt, Klasse, Methode, Assoziation, Generalisierung, Spezialisierung, Komposition

Objektorientierte Modellierung

Objektorientierte Modellierung

- Notation: Kombination verschiedener Modellierungen
  - Object-Oriented Analysis, Coad-Yourdan, 1991
  - Object-Oriented Design, Booch, 1991
  - Object Modeling Technique, Rumbaugh, 1991
  - Unfired Modeling Language, Rumbaugh, Booch, Jacobson, 1991
- Zweck: Abstrakte Modellierung der Struktur der Informationsmodells
  - Klassifikation und Instantiierung
  - Generalisierung und Spezialisierung
  - Aggregation und Dekomposition
  - Assoziation und Referenzierung
  - Modularisierung
  - Kapselung
  - Polymorphie
- Anwendungkontext: Konzeptuelles Datenmodell in Analyse und Entwurf, technisches Datenmodell in Entwurf und Implementierung

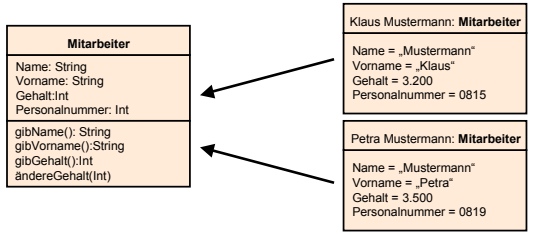
Objekt



Objekt: Unterscheidbare Einheit mit veränderbaren Eigenschaften

- **Identität**: Jedes Objekt ist eindeutig identifizierbar
- **Zustand**: Jedes Objekt besitzt (potentiell) veränderliche Eigenschaften in Form von (festen) Attributen
- **Verhalten**: Jedes Objekt besitzt (feste) Operationen zur Modifikation und Inspektion seines Zustandes

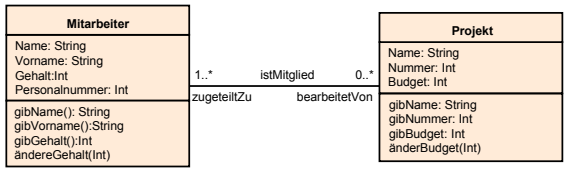
Klasse



Objektklasse: Charakterisierung gleichartiger Objekte

- **Attribute**: Eine Klasse definiert die gemeinsamen Attribute aller Objekte
- **Operationen**: Ein Klasse definiert die gemeinsamen Operationen aller Objekte
- **Typen**: Klassen bündeln Typen (Schnittstellendefinitionen)

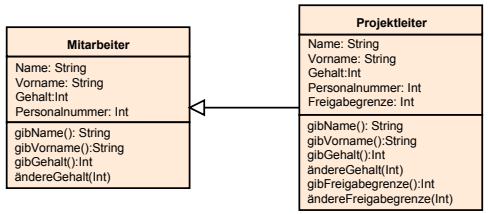
Assoziation



Assoziation: Zusammenhänge zwischen Objekten

- **Rollen**: Eine Assoziation definiert die teilnehmenden Objekte
- **Multiplizitäten**: Eine Assoziation beschränkt die Anzahl zusammengehöriger Objekte

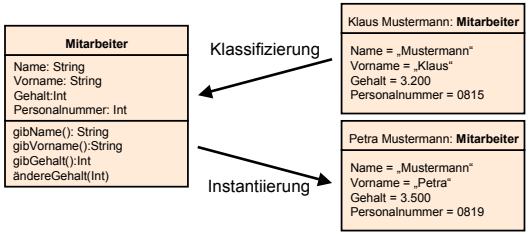
Vererbung



Vererbung: Wiederverwendung von Eigenschaften von Klassen

- **Oberklasse**: Die Vererbung überträgt die Eigenschaften der Oberklasse
- **Unterklasse**: Die Vererbung erweitert die Eigenschaften der Unterklasse

Klassifikation und Instantiierung



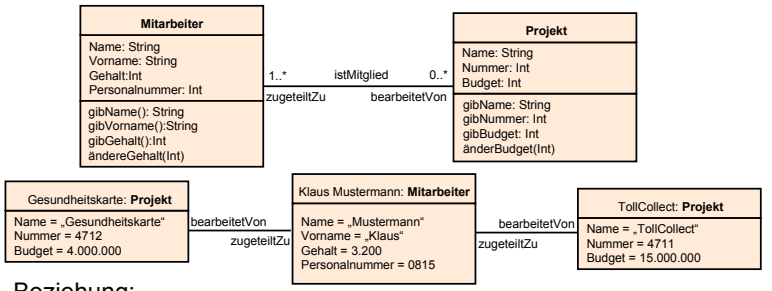
Klassifizierung:

- Ein Klasse beschreibt eine Menge funktionell und strukturell gleichartiger Objekte

Instantiierung:

- Jedes Objekt ist (unveränderlich) Element einer Klasse
- Objekte einer Klasse unterscheiden sich nur in ihrem Zustand

Beziehung und Referenzierung



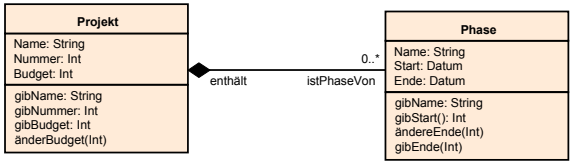
Beziehung:

- Assoziationen definieren Beziehungen zwischen Objekten
- Mutliplizitäten beschränken Kardinalität der Beziehungen

Referenzierung:

- Rollennamen identifizieren die einem Objekt verwandten Objekte
- Rollenname entsprechen abgeleiteten Attributen

Aggregation und Dekomposition



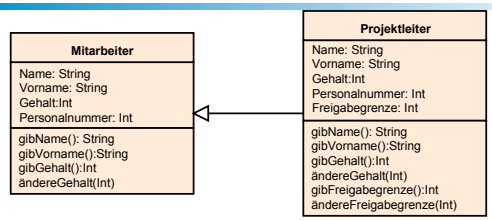
Aggregation/Komposition:

- Klassen nutzen Aggregation zur strukturellen Assoziation
- Klassen nutzen Komposition zur strukturellen Konstruktion

Dekomposition:

- Klassen nutzen Referenzierung zur Dekomposition

Generalisierung und Spezialisierung



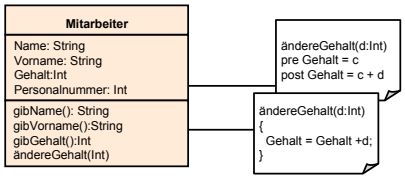
Spezialisierung:

- Spezialisierung verfeinert Klassen zu spezielleren Subklassen
- Superklassen können durch Subklassen substituiert werden

Generalisierung:

- Generalisierung vergröbert Klassen zu allgemeineren Superklasse
- Klasseninstanzen sind auch Superklasseninstanzen

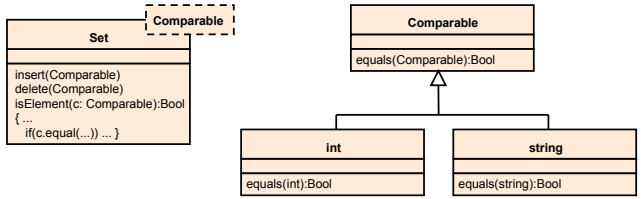
Abstraktion und Kapselung



Abstraktion und Kapselung:

- Klassen spezifizieren die Schnittstelle (Attribute und Methoden)
- Klassen verbergen die Implementierung der Nutzungsschnittstelle (Kontrakte)

Parametrierung und Polymorphie



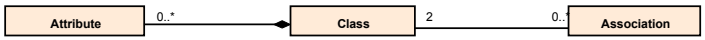
Parametrierung:

- Klassen nutzen Vererbung für die Definition parametrierter Klassen

Polymorphie:

- Klassen nutzen Vererbung zur Realisierung von Polymorphie
- Klassen nutzen Polymorphie für die Instantiierung parametrierter Klassen

Ausflug: Metamodellierung



Beobachtung:

- Modelle definieren konzeptuelle Strukturen zur Beschreibung der Anwendungsdomäne
- Die konzeptuellen Strukturen werden durch konzeptuelle Klassen und konzeptuelle Relationen beschrieben

Ansatz:

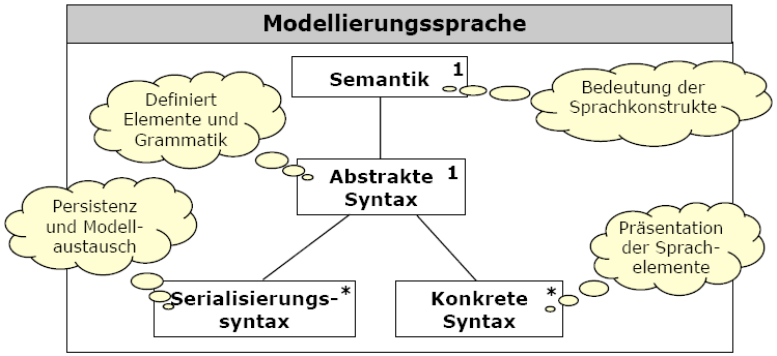
- Definition eines generischen konzeptuellen Modells zur Beschreibung der Strukturen spezifischer konzeptueller Modelle
- Definition von spezifischen Modellen als Instanzen des generischen Modells

Sinn und Zweck der Metamodellierung

- Vorteile von Metamodellen
  - Prägnante und präzise Definition der Sprachkonzepte
  - Einheitliches Austauschformat
  - Korrektheitsprüfung von Modellen
  - Verwaltung von Modellen in Repositories
  - Erweiterbarkeit der Sprache
- Verallgemeinerung auf höherer Abstraktionsebene durch das Meta-Metamodell
  - Sprachkonzepte für die Definition des Metamodells

Anatomie formaler Modellierungssprachen

- Obwohl Sprachen divergente Ausrichtungen und Anwendungsgebiete besitzen, haben sie doch einen gemeinsamen Sprachaufbau



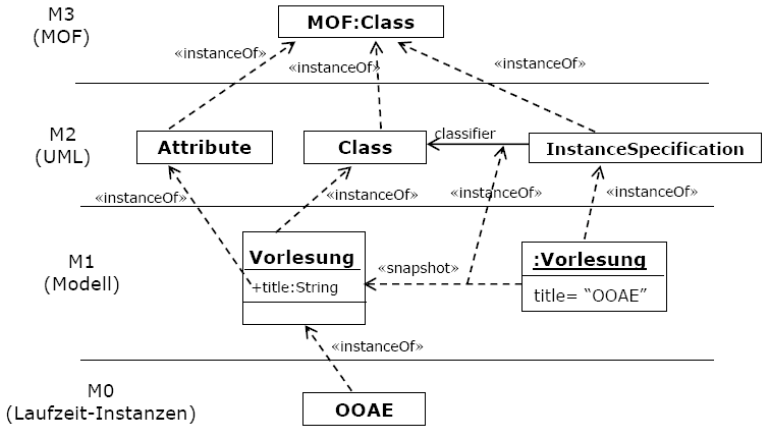
Anatomie formaler Modellierungssprachen

- Hauptbestandteile:
  - Abstrakte Syntax: Beschreibung der Sprachkonzepte und ihrer Kombination
  - Konkrete Syntax: Notation zur Darstellung der Sprachkonzepte und Modellekonstruktion
  - Semantik: Definition der Bedeutung der Sprachkonstrukte
  - Serialisierungssyntax: Zur persistenten Speicherung und Modellaustausch zwischen Werkzeugen
- Zusätzliche Eigenschaften:
  - Erweiterung der Sprache durch neue Sprachkonzepte, insb.domänen- oder technologiespezifische Erweiterungen
  - Abbildungen auf andere Sprachen, insb. semantisch äquivalente Transformation

Spracharchitektur von UML 2.0

- 4-Schichten Architektur
  - Typ/Instanz-Muster wird wiederholt angewandt -> M3-, M2-, M1-, M0-Ebene
  - MOF auf M3-Ebene -> “hard-wired” Meta-Metamodell
- Metamodell definiert “nur” die abstrakte Syntax
  - Mögl. Notationsform: MOF-Notation (~Klassendiagramm)
  - Einschränkungen definieren Wohlgeformtheit der Modelle
    - Sogenannte well-formedness Rules
- Konkrete Syntax wird informell definiert
  - Durch Notationstabellen
  - Eventuelle Notationsoptionen in natürlicher Sprache
- Semantik wird in natürlicher Sprache beschrieben
  - Zusätzlich werden Semantische Variationspunkte angegeben

Spracharchitektur von UML 2.0



Spracharchitektur von UML 2.0

Schicht	Beschreibung	Beispiele
Meta-Metamodell (M3)	Infrastruktur für Meta-modellierung. Definiert die Menge der Konstrukte, die zur Spezifikation von Metamodellen genutzt wird	*MOF:Class* *MOF:Attribute* *MOF:Association*
Metamodell (M2)	Metamodelle bestehen aus Instanzen von MOF-Konstrukten und definieren Modellierungs-sprachen	*UML:Class, UML:Association, UML:Attribute, UML:State, CWM:Table, CWM:Column*
Model (M1)	Instanz eines Metamodells um eine Domäne zu beschreiben	*Kundenverwaltung: Class Kunde, Class Bestellung, …*
Laufzeit-Instanzen (M0)	Konkrete Daten, Objekte	*Kunde Wimmer, Bestellung 123, …*

MOF – Meta Object Facility

- OMG-Standard für die Definition von Metamodellen
- Definiert eine Menge von Sprachelementen, um Modellierungssprachen zu spezifizieren
- Basiert auf objektorientierten Konzepten: Generalisierbare Klassen und Beziehungen zwischen diesen
- MOF ist selbst durch MOF beschrieben und unterteilt in:
  - EMOF (essential) - einfache Sprache für die Definition von Metamodellen: Klassen mit Attributen und Methoden
  - CMOF (complete) – erweitert EMOF, Unterstützt die Verwaltung von Metadaten und bietet erweiterte reflektive Dienste
- Abbildungsregeln für:
  - XML: XML Metadata Interchange - XMI
  - Java: Java Metadata Interfaces – JMI
  - CORBA: Interface Definition Language - IDL