Überblick UML (Unified Modeling Language)

- Objektorientierte Analyse, objektorientierter Entwurf
- Vorläuferväter: Grady Booch, Jim Rumbaugh, Ingvar Jacobson

Modellierung von Daten/Funktionen (statisch)	*Modellierung von Prozessen (dynamisch)*
Strukturdiagramme: • **Klassendiagramm** • **Objektdiagramm** • Komponentendiagramm • Kompositionsdiagramm • Paketdiagramm • Verteilungsdiagramm	Verhaltensdiagramme: • **Anwendungsfalldiagramm (use case)** • Aktivitätsdiagramm (activity) • Zustandsdiagramm Interaktionsdiagramme: • Sequenzdiagramm • Kommunikations-/Kollaborationsdiagramm • Zeitverlaufsdiagramm • Interaktionsübersichtsdiagramm

Objektorientierung

Objekte

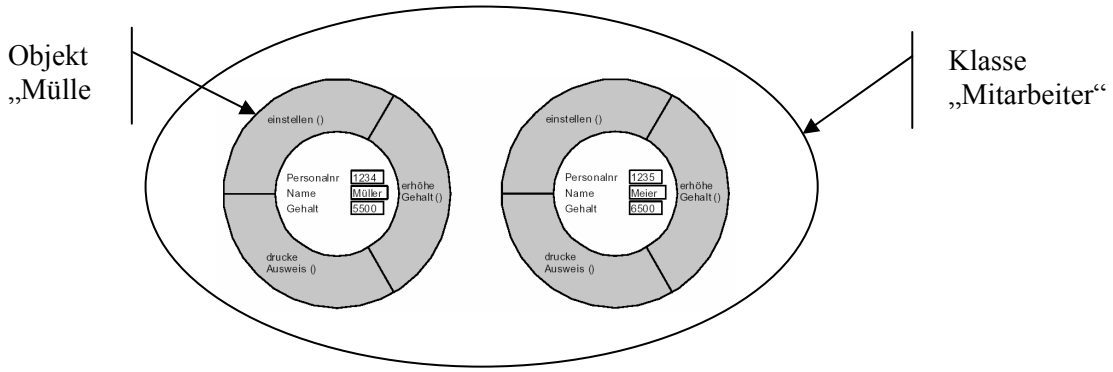
- Objekte: „Dinge“ der realen Welt, die von Interesse ist.
  - Arten
    - Gegenständlich: Auto, Produkt, Kunde, Büro
    - Ungegenständlich: Geschäftsvorfall, Reservation, Organisationseinheit
  - Charakteristika
    - besitzt bestimmten *Zustand*.
    - reagiert mit definiertem *Verhalten* auf seine Umgebung.
    - besitzt Identität, die es von allen anderen Objekten unterscheidet.
    - kann *Beziehungen* mit einem/mehreren Objekten eingehen.
- Objekte fassen Zustand und Verhalten zusammen
  - Zustand
    - Attribute bzw. deren aktuellen Wert (inhärente Merkmale des Objekts; veränderbar)
    - Jeweiligen Verbindungen zu anderen Objekten
  - Verhalten
    - wird durch Menge von Operationen beschrieben
    - Änderung/Abfrage eines Zustands ist nur über Operationen möglich
    - Bei UML ist Verhalten immer ereignisgesteuert oder diskret.

Klassen

- Gleichartige Objekte werden zu Klassen zusammengefasst
- Klasse definiert für eine Menge von Objekten..
  - Struktur (Attribute)
  - Verhalten (mögliche Operationen)
  - Beziehungen

  ⇒ Ein Objekt ist eine *Instanz* einer Klasse.

  ⇒ Objekte verhalten sich zu Klassen wie Entitäten zu Entitätstypen.



Klassen- und Objektdiagramme

Attribute und Verhalten

Objektdiagramm *Klassendiagramm*

	Müller: Mitarbeiter	Müller: Mitarbeiter		Mitarbeiter
Bezeichner			Bezeichner	
Struktur	Personalnr = 1234 Name = Müller Gehalt = 5500	Personalnr = 1235 Name = Meier Gehalt = 6500	Struktur	Personalnr Name Gehalt
Verhalten	Einstellen() ErhöheGehalt() DruckeAusweis()	Einstellen() ErhöheGehalt() DruckeAusweis()	Verhalten	Einstellen() ErhöheGehalt() DruckeAusweis()

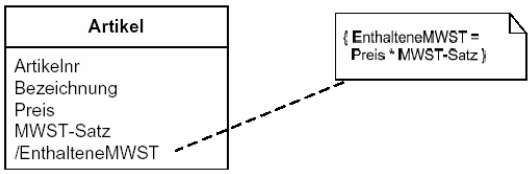
- Optional: Angabe von Attributtypen
  ⇒ Darauf wird bei „implementierungsunabhängiger“ (z. B. Art der DB) Modellierung verzichtet.

Klassenattribute

- Nimmt für alle Objekte der Klasse den selben Wert an. (º Konstante)
- Festlegung auf Klassenebene
- Als Unterstrichenes Attribut gekennzeichnet

Abgeleitete Attribute

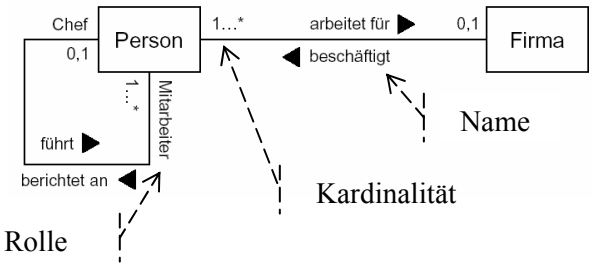
- Attribut, dessen Wert sich aus anderen Attributen errechnet.
- Kennzeichnung des Attributs durch Voranstellen von „/“.
  Berechnungsvorschrift (Restriktion, Constraint) kann angegeben werden.
- Methoden (anstelle als Ausprägung) dann sinnvoll, wenn abgeleitetes Attribut prozessual berechnet werden muss (über mehrere Schritte).



Unterschiede zu ER-Diagrammen

Objekte haben per se eine Identität, die sich voneinander unterscheidbar macht.
⇒ Keine künstlichen Identifikatoren nötig

Beziehungen (Assoziationen)

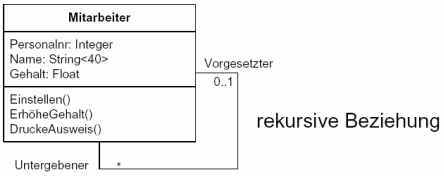


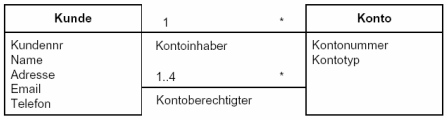
Einfache Beziehung
⇒ Angabe von Kardinalitäten obligatorisch.
⇒ Angabe von Name und Rolle optional.
⇒ Bezieh. zwischen Obj. gleichrangiger Klassen

Kardinalitäten

⇒ keine expliziten min|max Angaben!

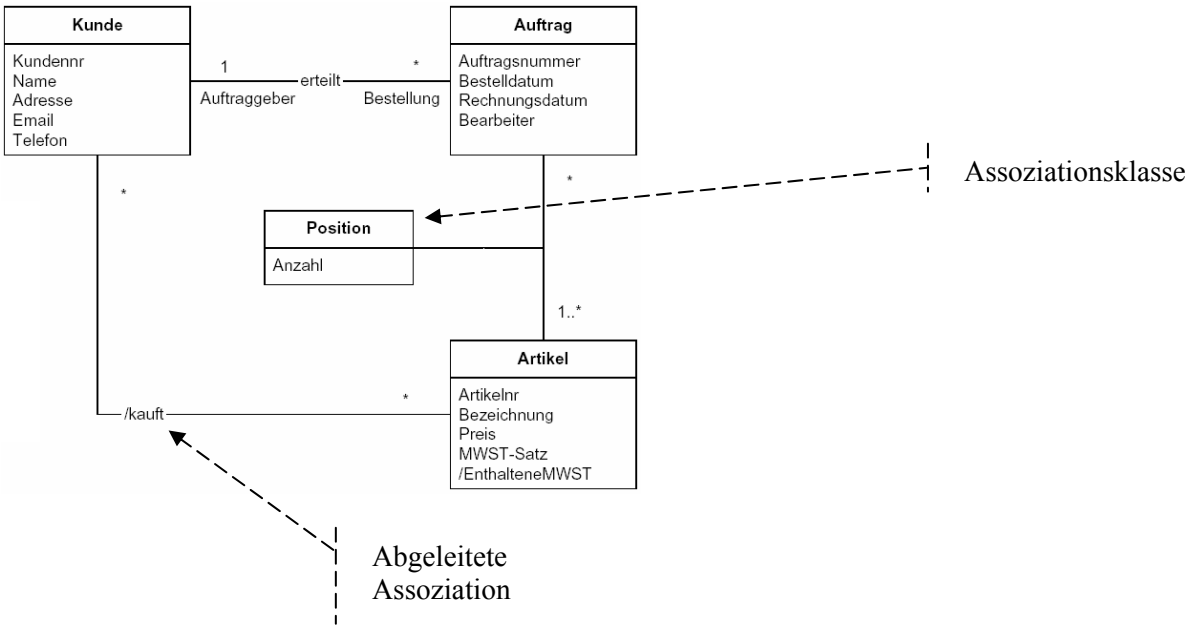
1	genau 1
0..1	0 oder 1
*	0 bis beliebig viele
1..*	1 bis beliebig viele
0..2	0, 1 oder 2
2	genau 2
2, 4, 6	2, 4, oder 6
1..5, 8, 10	1, 2, 3, 4, 5, 8 oder 10





Spezielle Assoziationen

- Assoziationsklassen: Assoziation kann über Attribute und Operationen verfügen.
- Abgeleitete Assoziation: Assoziation ergibt sich aus anderen Assoziationen.

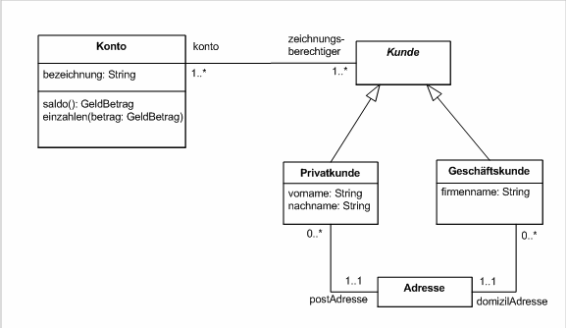


- **Aggregation**: Ganzes-Teil-Beziehung
  - Asymmetrisch: Das Teil existiert unabhängig vom Ganzen, aber das Ganze erfordert das Teil.
  - Transitiv: Ist A Teil von B und B Teil von C, dann ist auch A Teil von C.

- **Komposition**
  - bindet stärker als Aggregation
  - Ein Teilobjekt darf – zu einem Zeitpunkt – höchstens Bestandteil eines Aggregatobjekts sein.
    ⇒ Existenz des Teils hängt vom Ganzen ab.
    ⇒ Bsp. Verzeichnis löschen ⇨ dort vorhandene Dateien löschen

Vererbungen

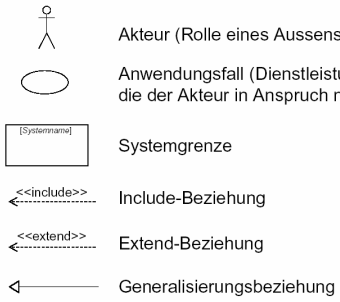
- Verknüpft Klassen miteinander (nicht Objekte!)
- Generalisierung ist partiell und nicht disjunkt.



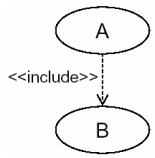
Anwendungsfalldiagramme (use cases)

- Überblick über das betrachtete System (inkl. Schnittstellen) auf hohem Abstraktionsniveau
- Anwendungsfalldiagramme werden aus Sicht der Akteure beschrieben.
- Akteure
  - Mensch oder anderes System ausserhalb des betrachteten Systems. (Rolle)
  - Steht meist in Beziehung zu einem Anwendungsfall.
  - Generalisierungen von Akteuren auch möglich.
- Anwendungsfall
  - Beschreibt Funktionalität des Systems.
  - Sagt nichts über das „Wie“ des Verhaltens aus. Dazu sind „Verhaltensbeschreibungen“ da.

Elemente

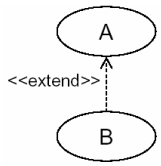


Include-Beziehung



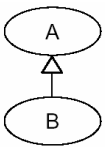
Importieren eines Anwendungsfalls: Das Verhalten des Anwendungsfalls schliesst zwingend das Verhalten des importierten Anwendungsfalls mit ein.

Exclude-Beziehung



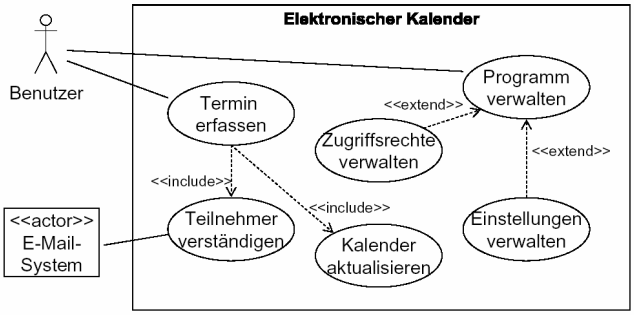
Erweitern eines Anwendungsfalls: Die Basisfunktionalität eines Anwendungsfalls wird mit spezifischen Anwendungsfällen erweitert, wobei deren Ausführung optional und in diesem Sinne unabhängig von der Basisfunktionalität ist.

Generalisierungsbeziehung



Der erbende Anwendungsfall erbt das Verhalten und die Beziehung des vererbenden Anwendungsfalls und kann diese überschreiben und ergänzen.
⇒ XOR-Generalisierung
⇒ Generalisierung ist total und disjunkt.

Beispiel



Vorgehen

1. Akteure identifizieren
2. Anwendungsfälle identifizieren
3. Akteure und Anwendungsfälle verbinden
4. Erweiterungen suchen und Diagramm verfeinern.