ModSoft

Modellbasierte Software-Entwicklung mit UML 2 im WS 2014/15

Einführung

Prof. Dr. Joachim Fischer
Dr. Markus Scheidgen
Dipl.-Inf. Andreas Blunk

fischer@informatik.hu-berlin.de

Teil I: Einführung

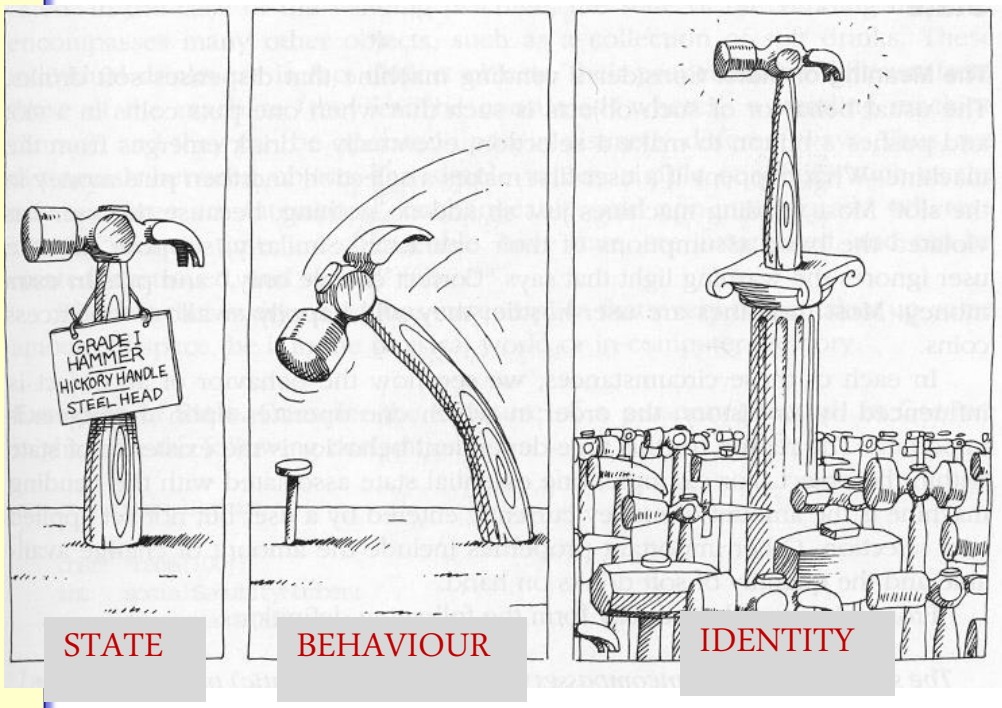
1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassen (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen / allg. Classifier
4. Identifikation verschiedenster Beziehungen zwischen Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

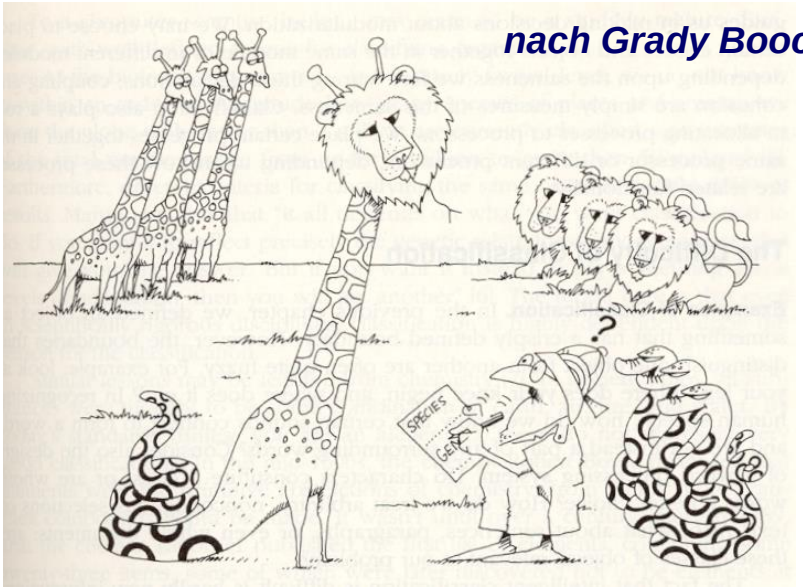
nach Grady Booch



Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen



nach Grady Booch

KLASSIFIKATION VON OBJEKTEN

Definition benötigter Objektklassen
statt
Definition einzelner Objekte

PRINZIP DER DATENKapslung

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

OBJEKTE AKTIVER Klassen

~ Vorgänge in RAUM und Zeit

… agieren eigenständig
und in Kooperation miteinander
bei Austausch
von Informationen /Daten
(synchron /asynchron)
statische oder dynamische Existenz

OBJEKTE Passiver Klassen

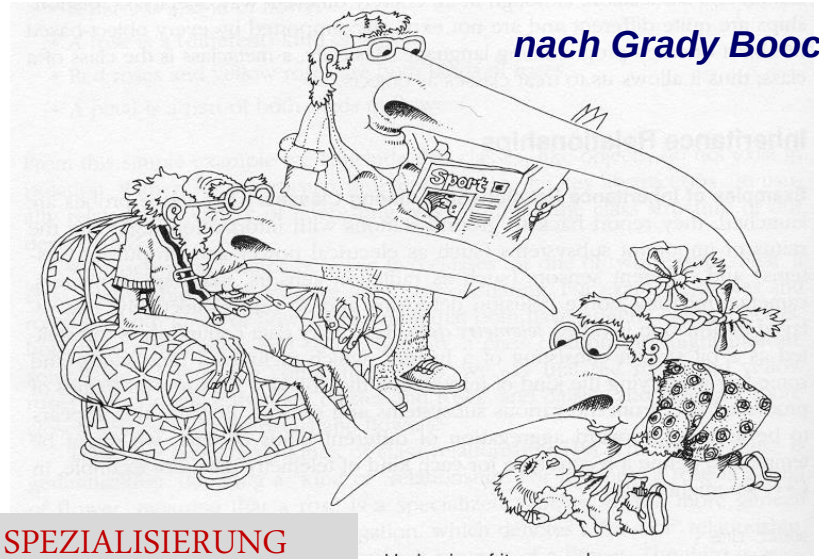
… werden von Objekten
aktiver Klassen benutzt

***Bsp**: ein Thread führt Operationen über Objekte von Java-Klassen aus*

Modellierungsparadigmen

Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - ...
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen



nach Grady Booch

SPEZIALISIERUNG

bedeutet
Wiederverwendung der gemeinsamen Basiskonzepte
- Attribute
- Verhalten / Methoden

Modellierungsparadigmen

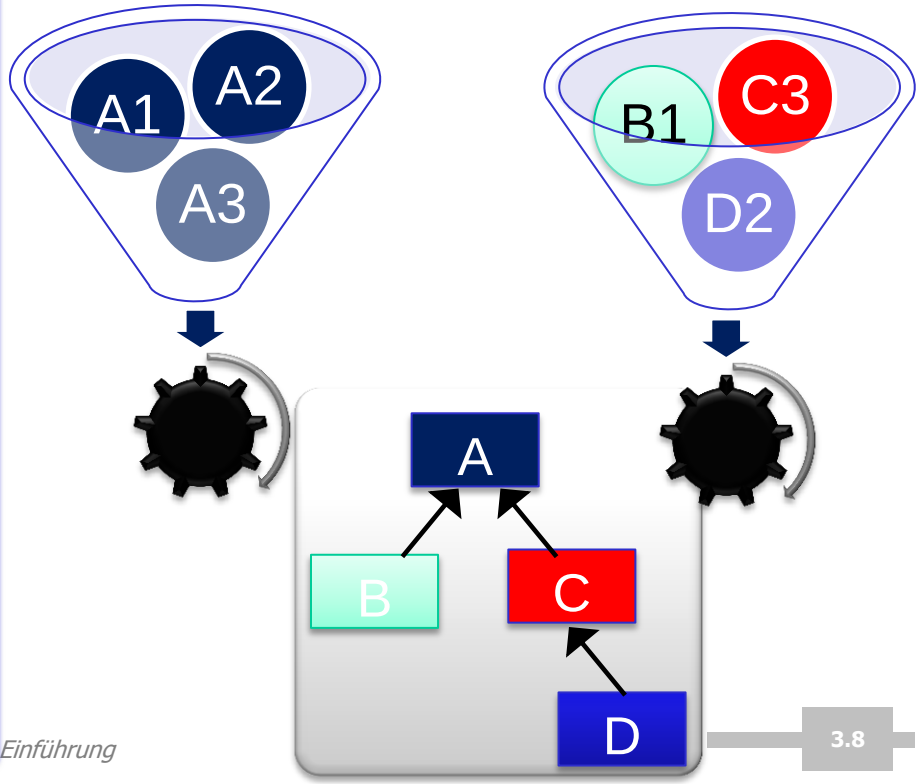
Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifizierer
6. Polymorphie von (getypten) Referenzen

Vielgestaltigkeit

überall dort, wo die Verarbeitung von Objekten einer (Basis-) Klasse definiert ist,

lässt sich auch eine Verarbeitung von Spezialisierungen organisieren



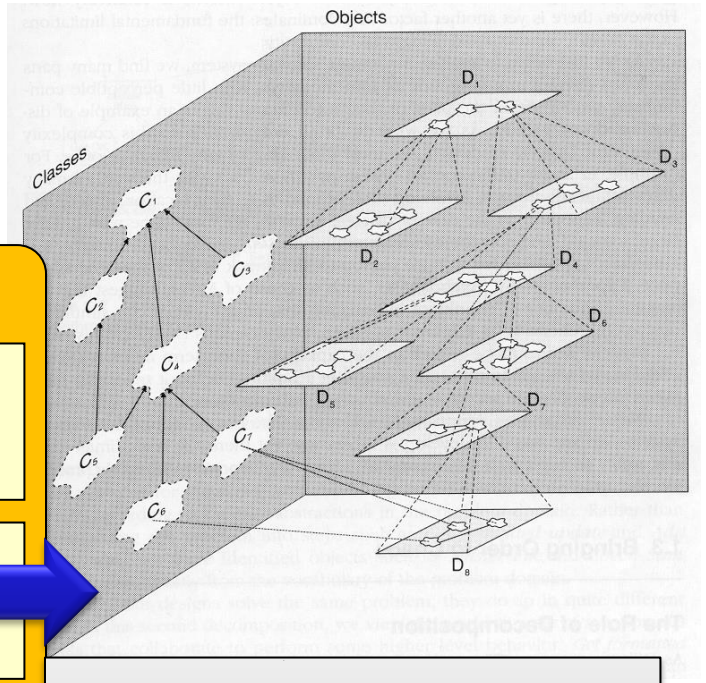
Modellierungsparadigmen

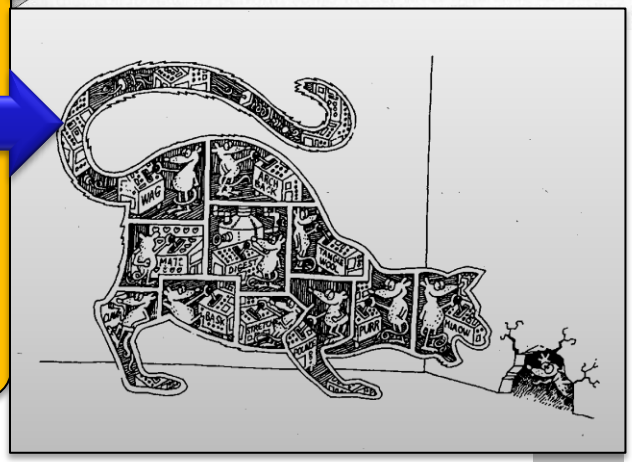
Objektorientiertes Abstraktionskonzepte

1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassifikation (Definition von Objekt
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zw. Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifiziere
6. Polymorphie von (getypten) Referenzen

zusätzlich …

- Gruppierung von Modellelementen
- Komposition/ Dekomposition
- Nebenläufigkeit/ Parallelität/ Synchronisation
- zeitdiskretes/ zeitkontinuierliches Verhalten



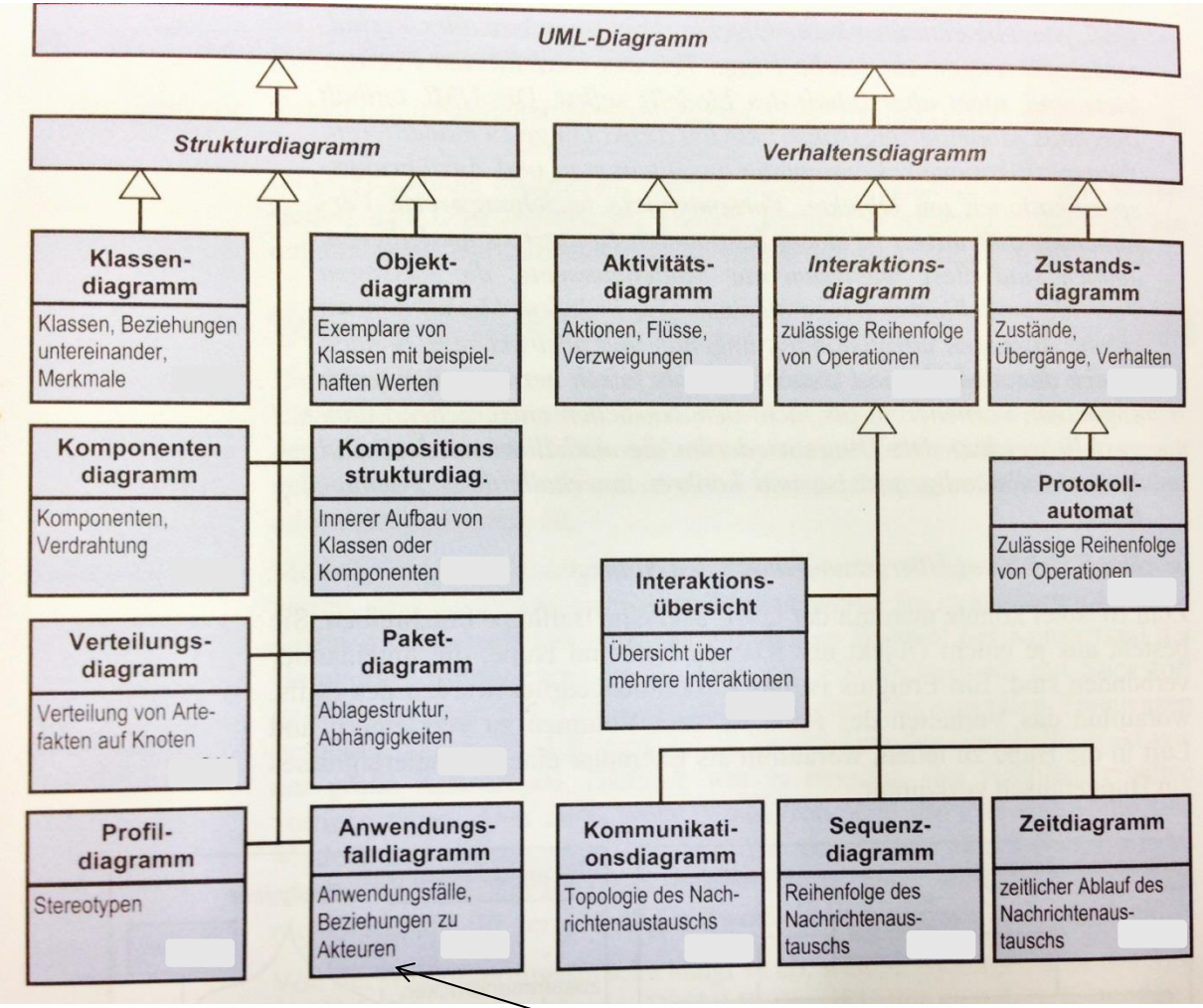


Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

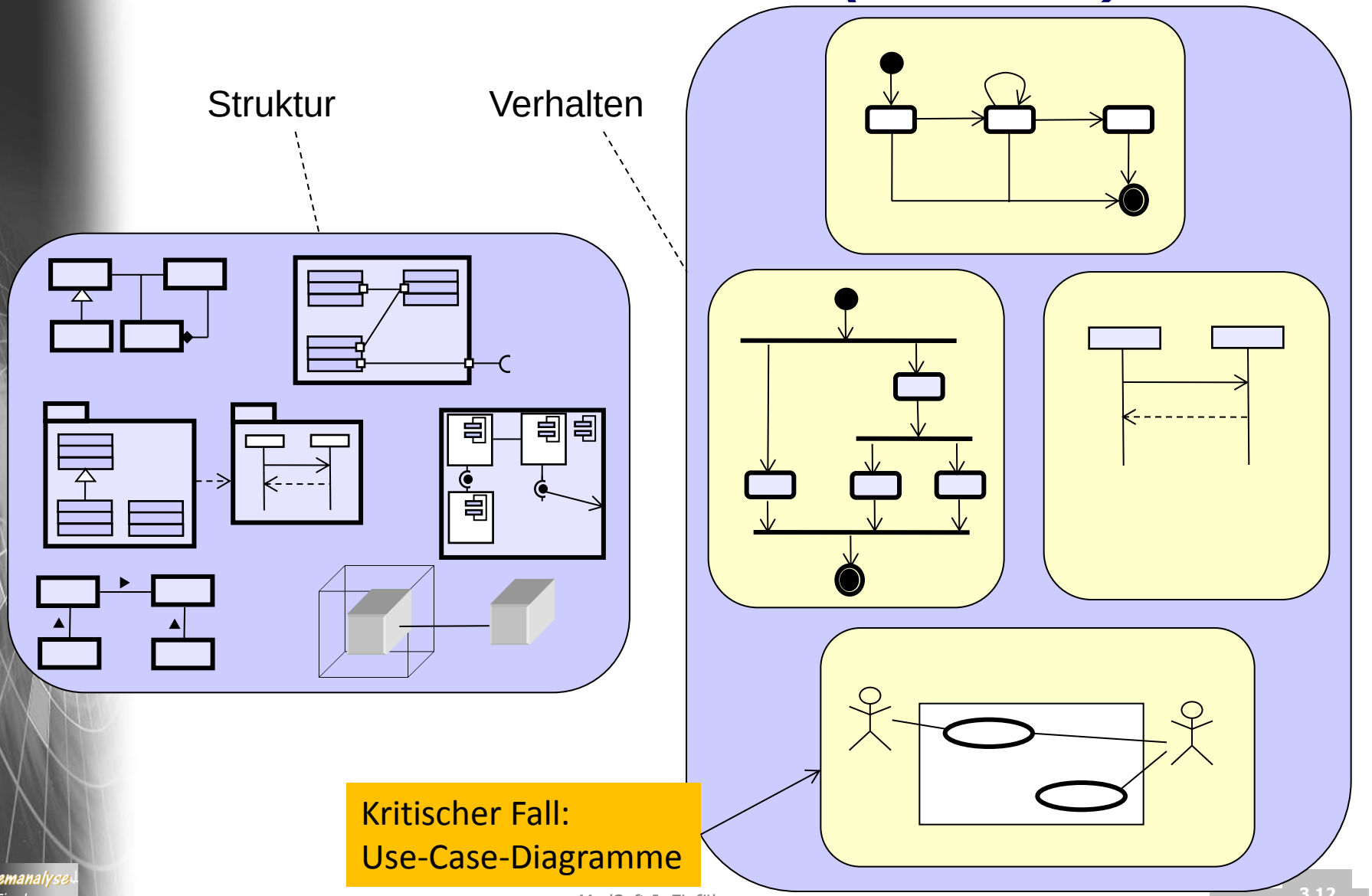
UML-Diagrammarten

UML erlaubt aber auch beliebigen Mix in einem Diagramm (Tool-abhängig)



ACHTUNG: Abweichung
UML-Standard: Anwendungsfalldiagramm als Verhaltensdiagramm)

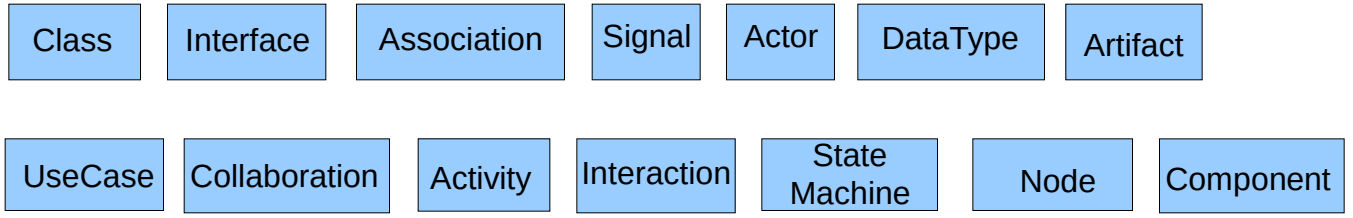
Diagrammarten zur Anordnung von Modellelementen (Entitäten)



Kritischer Fall:
Use-Case-Diagramme

Strukturelle Modellelemente: Überblick

- Bezeichnung: Substantive in Modellen
- bilden die Struktur eines Modells
  (meist in statischer aber auch dynamischer Art und Weise)
- generalisierende Zusammenfassung als *Classifier*

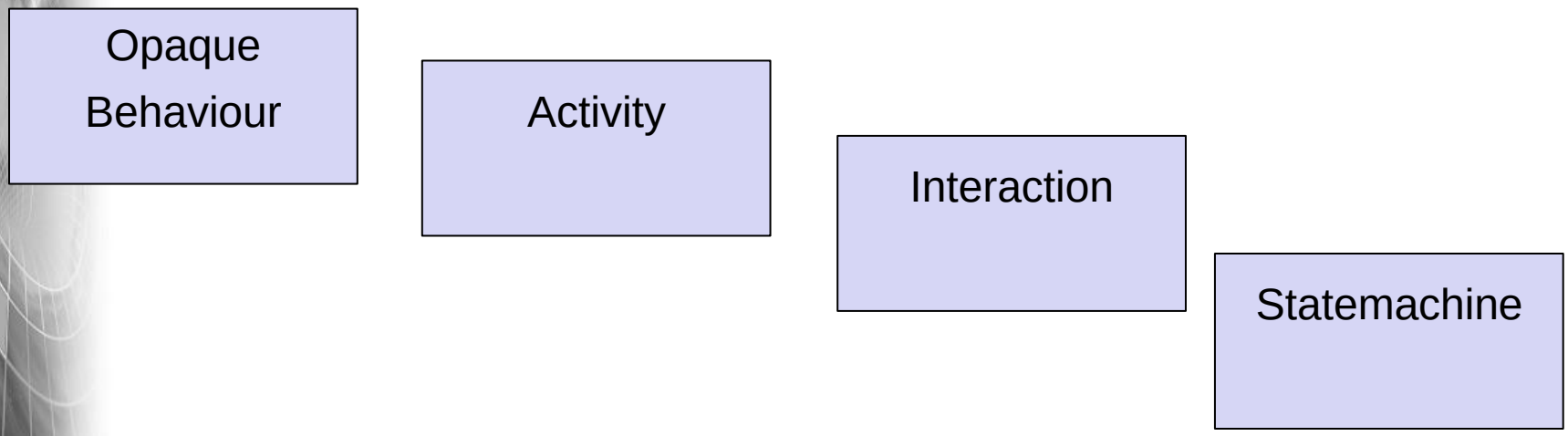


z.T. mit weiteren Spezialisierungen

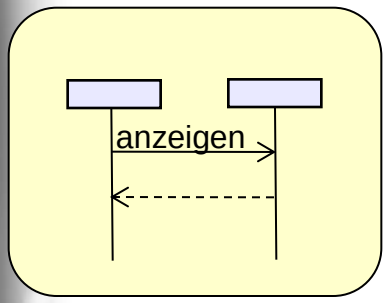
dafür nun einige Beispiele (Syntax, Semantik) …

Verhaltensorientierte Modellelemente

- bestimmen das Verhalten eines Modells in Raum und Zeit
- Zusammenfassung als abstraktes Konzept *Behaviour*
- Hauptformen

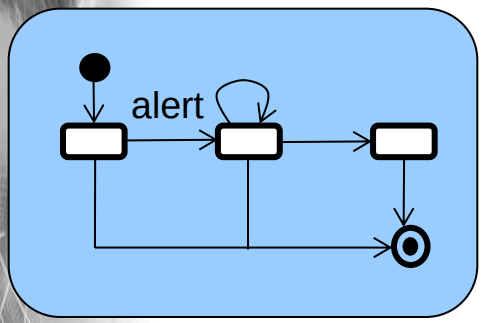


Interaktion, Zustandsautomat



Interaktion (Interaction)

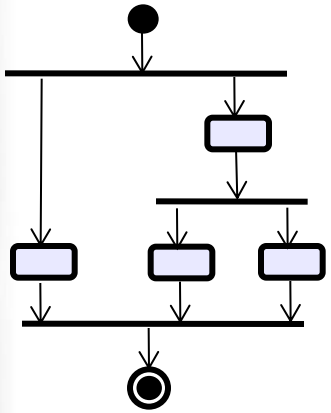
- beschreibt gerichtete Interaktion zwischen Objekten oder Rollen eines best. Kontextes zur Erbringung einer best. Funktionalität
- besteht aus Nachrichten, Aktionen und Konnektoren
- Pfeilart und Linienausführung bewirken unterschiedliche Semantiken



Zustandsautomat (State Machine, State-Chart)

- beschreibt Abfolge der Zustände, die ein Objekt oder eine Interaktion während seiner Existenz als Reaktion auf Ereignisse durchläuft, wobei Ereignisreaktion ebenfalls dargestellt wird
- gestatten die Beschreibung von Verhalten von Klassen oder Kollaborationen
- Zustandsautomaten bestehen aus weiteren Entitäten: Zustände (inkl. Start, Stop), Übergänge, Ereignisse

Aktivität



Aktivität (Activity)

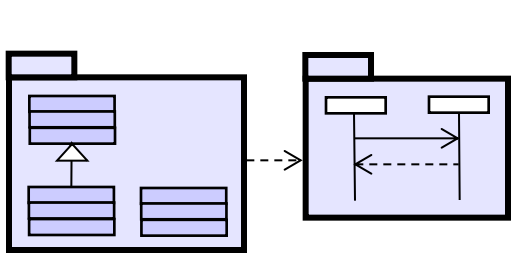
- beschreibt gerichtete Abfolge von Schritten eines Verarbeitungsvorganges
- Schritte einer Aktivität heißen Aktionen (Actions)
- bei der Interaktion liegt der Schwerpunkt der Verhaltensbeschreibung auf einer Reihe von Objekten, die interagieren

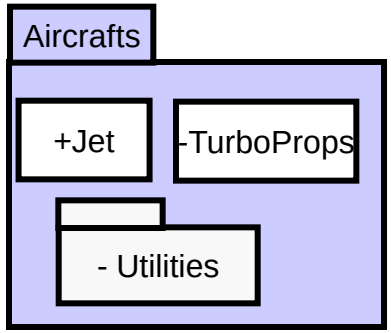
  beim Zustandsautomaten beim Lebenslauf einer einzelnen Entität !!!
- bei der Aktivität liegt der Schwerpunkt bei den Übergängen von Schritt zu Schritt - **ohne** zu beachten, welches Objekt den jeweiligen Schritt realisiert

Gruppierungs-Elemente

Organisatorische Einheiten von UML-Modellen:

Paket (Package),
aber auch Paket-Variationen
(Rahmenwerken, Modelle, Subsysteme, System)

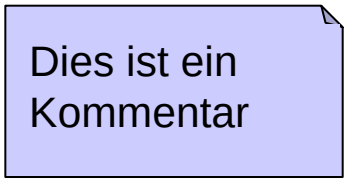




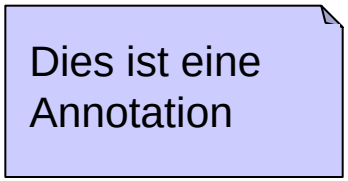
- Zusammenfassung von Struktur-, Verhaltens-, Anmerkungs- und anderen Gruppierungs-Entitäten ist möglich
- es gibt Regeln zur Abhängigkeit und zur Nutzung von Paketen
- Gruppierungs-Entitäten sind rein konzeptueller Natur
  (existieren nicht zur Laufzeit)

4. Anmerkungs-Elemente

beschreibende Anteile eines UML-Modells



beliebiger Kommentar



Informale oder formale Textbeschreibung von Modelleinschränkungen und –zusätzen (Constraints)

UML-Standard schließt OCL ein (Object Constraint Language)

- können beliebigen Modellelementen zugeordnet werden
- Achtung: Anmerkungen werden von Tools auch über einblendbare Textboxen realisiert

UML-Praxistauglichkeit

- **20% der UML-Konzepte** erfüllen 80% der Praxisanforderungen
- Vorlesung versucht sich bei der Schwerpunktsetzung daran zu orientieren

- Standard besteht aus
  a) Superstructure (statische und dynamische Modellelemente)
  b) OCL (als Referenz auf einen separaten Standard)
  c) Diagramm-Austausch-Definition

Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Diagrammrepräsentationen in UML
8. Struktur des UML-Standards (2.5)

Specification Simplification

This specification has been extensively re-written from its previous version to make it easier to read by removing redundancy and increasing clarity. In particular, the following major changes have been made since UML 2.4.1:

- The **UML Infrastructure** no longer forms part of the UML specification. The entire UML specification is constituted in this document.
- **Package Merge** is not used within the specification. Every metaclass is specified completely in one clause.
- The specification is **organized to reduce forward references** as much as possible. This means that topics such as Templates which are pervasive in their effects appear early in the specification.
- Every clause has a **section of documentation generated from the metamodel** that contains all of the metaclasses with their properties, and all of the meta-associations with their properties. All cross-references in this generated documentation include hyperlinks to their targets.
- **The compliance levels** L0, L1, L2, and L3 have been eliminated, because they were not found to be useful in practice. A tool either complies with the whole of UML or it does not. A tool may partially comply with UML by implementing a subset of its metamodel, notation, and semantics, in which case the vendor should declare which subset it implements.

- However, the metamodel itself remains unchanged from UML 2.4.1 **superstructure**, with a few exceptions:
- The **metamodel** has been partitioned into packages, corresponding to the clause structure of this specification. All of these packages are owned by a **top-level package named UML**; they are also imported into UML so that metaclasses may be referred to by their unqualified name in UML.
- Many **OCL constraints** have been corrected or added where they were absent. In order to do this, some names of association-owned properties and the corresponding associations have been changed in order to avoid ambiguity in OCL expressions.

Zentrale UML-Modellelement-Typen

aus diesen Konzepten wird die gesamte UML aufgebaut

1. ***Classifiers***.

Classifier ist eine abstrakte Basis-Klasse im Metamodell von UML
und gemeinsame Basisklasse von Klasse, Anwendungsfall, Komponente, …
(weitere 34 Konzepte)

d
hat
tself may
ctural

features in sub clause 9.4.3.)

2. ***Events***.
   An event describes a set of possible occurrences. An *occurrence* is something that happens that has some consequence with regard to the system.
3. ***Behaviors***.
   A behavior describes a set of possible executions. An *execution* is a performance of a set of actions (potentially over some period of time) that may generate and respond to occurrences of events, including accessing and changing the state of objects.

   (As described in sub clause 13.2, behaviors are themselves modeled in UML as kinds of classifiers, so that executions are essentially modeled as objects. However, for the purposes of the present discussion, it is clearer to consider behaviors and executions to be in a separate semantic category than classifiers and objects.)

Quelle: OMG Unified Modeling Language TM (OMG UML), *Version 2.5*

4 Ebenen einer Sprachdefinition nach MOF

Ebene der Konzeptwelt,
die die oo-Defintion von Sprachen im Sinne von MOF beschreibt

M3

Ebene der Sprachdefinition von UML
(oder einer anderen Sprache X)

M2

Ebene der UML-Anwendung (oder einer anderen Sprache X)

(UML-Programm1, UML-Programm2, …
oder X-Programm1, X-Programm2, …)

M1

Ebene der Konzeptwelt,
die die Laufzeit-Repräsentation eines UML-Programms bzw. Die Realität
oder -X-Programms **beschreibt**

M0

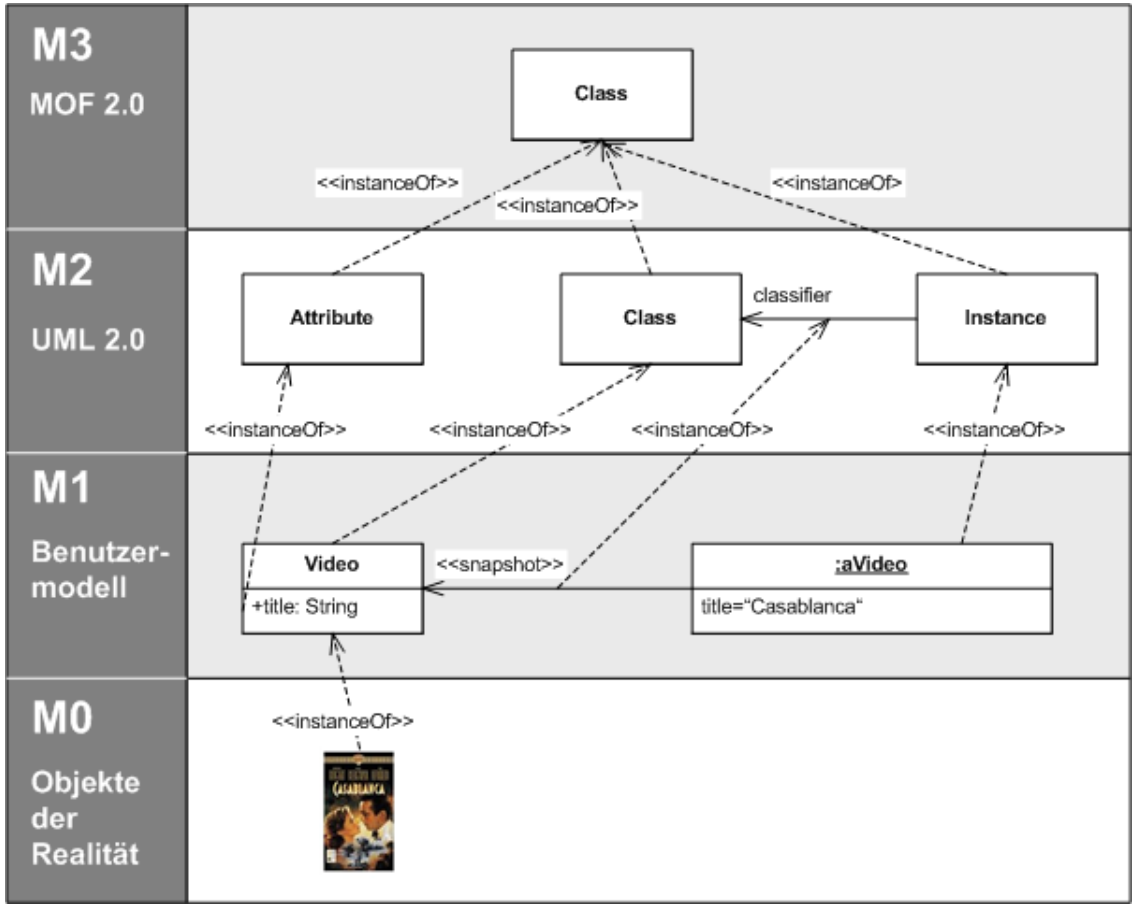
unter Nutzung von Konzepten der höheren Ebene

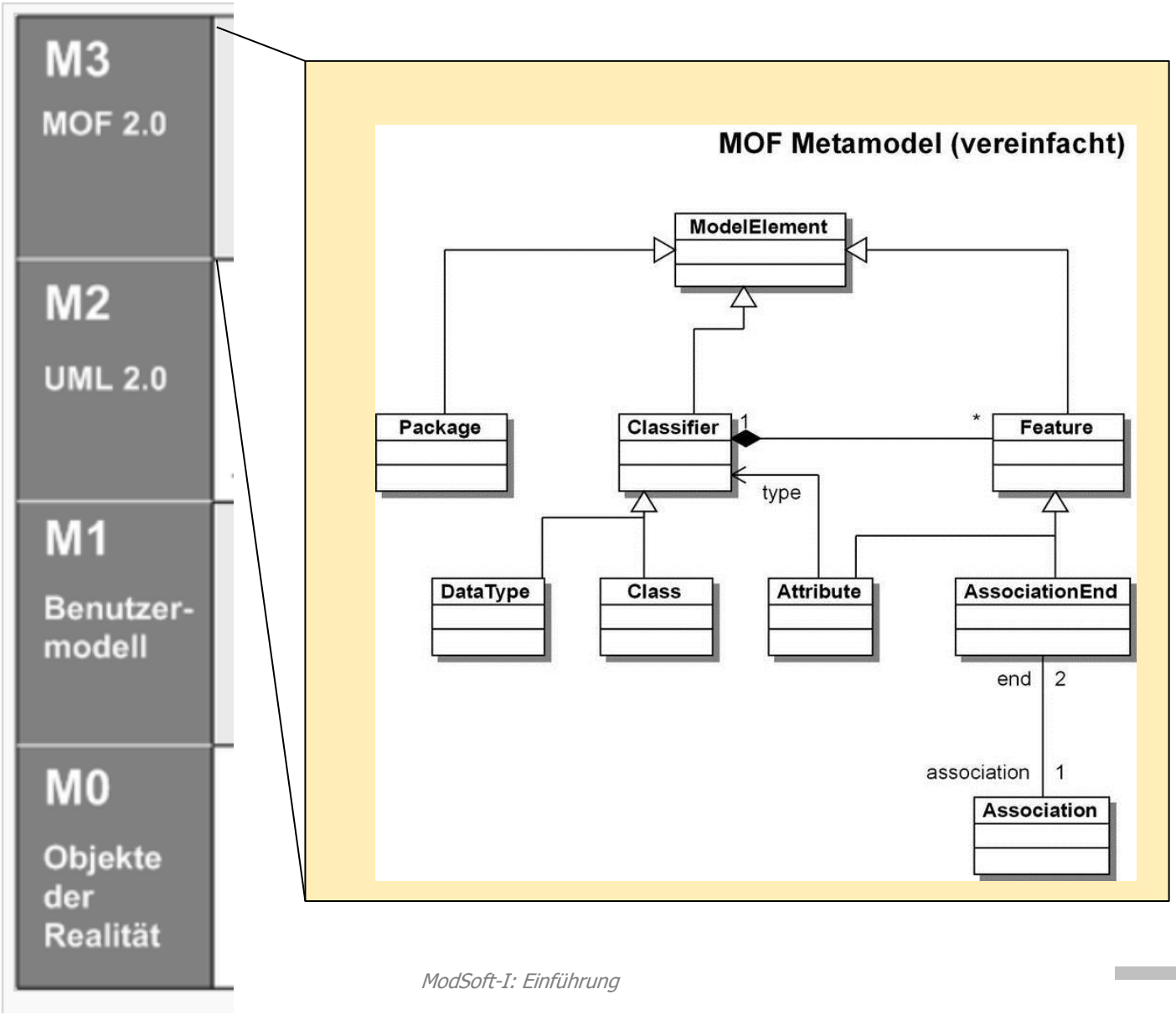
MOF

… stuft ihre Konzepte in vier Beschreibungsebenen (Meta-Ebenen) ein:

- **M0-Ebene** **Konkret**. Ausgeprägte Daten.
- **M1-Ebene** **Modelle**.
  Zum Beispiel physikalische oder logische Daten- oder Prozessmodelle oder konkrete Ausprägungen von UML- bzw. Objekt-Modellen, welche die Daten der M0-Ebene definieren.
- **M2-Ebene** **Meta-Modelle**.
  Definieren, wie die Modelle aufgebaut und strukturiert sind. Zum Beispiel definieren Sprachelemente wie Klassen, Assoziationen und Attribute der UML 2.0, wie konkrete UML-Modelle aufgebaut sein können.
- **M3-Ebene** **Meta-Meta-Modelle** (bzw. MOF-Ebene).
  Abstrakte Ebene, die zur Definition der M2-Ebene herangezogen wird. Die Definition der M3-Ebene erfolgt mit den Mitteln der M3-Ebene selbst, dies stellt den Abschluss einer sonst unendlichen Metaisierung dar.

MOF-Klassen: Stark vereinfachte Darstellung




M3
MOF 2.0
M2
UML 2.0
M1
Benutzer-
modell
M0
Objekte
der
Realität
MOF Metamodel (vereinfacht)
ModelElement
Package
Classifier
1
*
Feature
type
DataType
Class
Attribute
AssociationEnd
end
2
association
1
Association

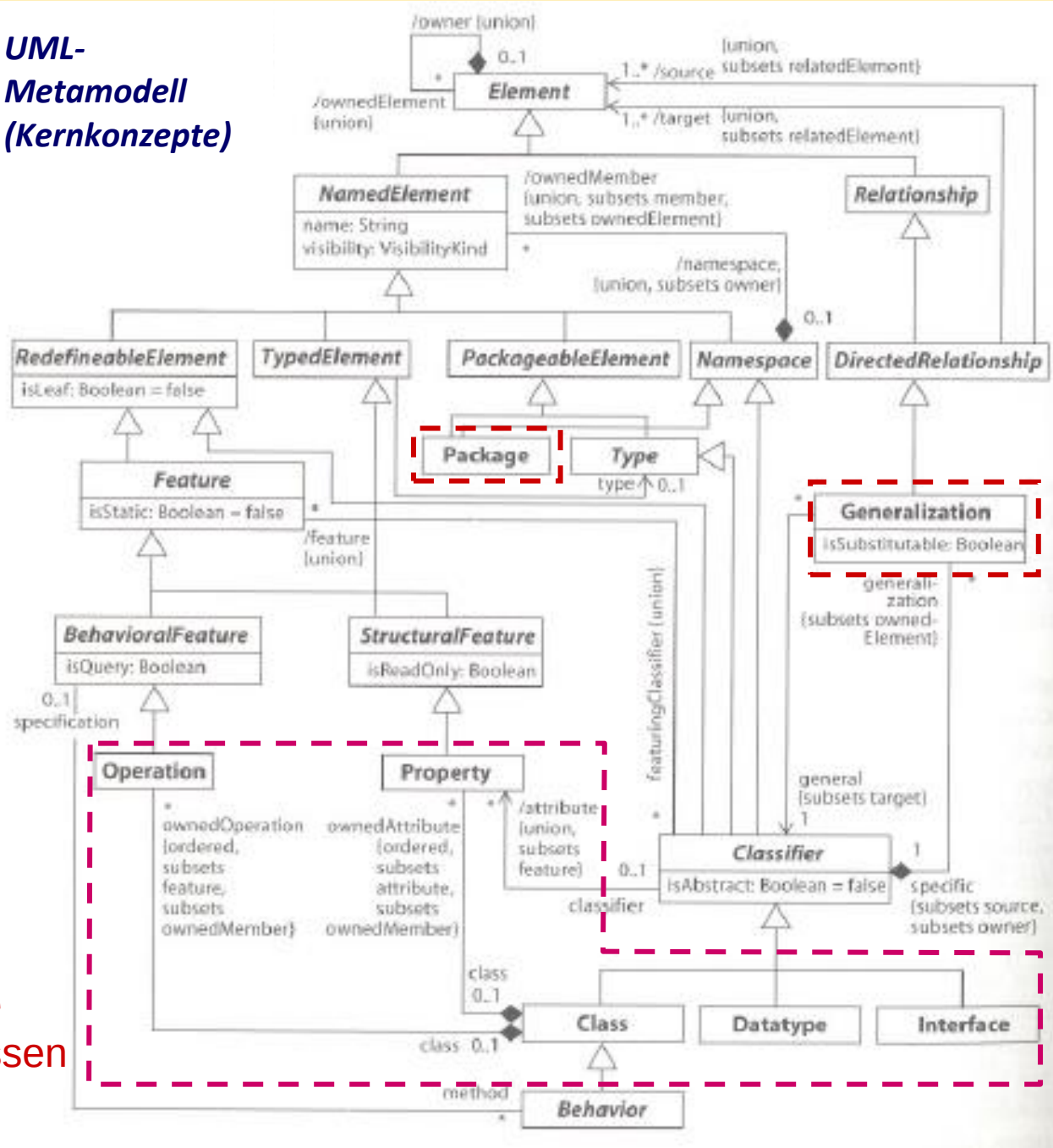
M3
MOF 2.0

M2
UML 2.0

M1
Benutzer-modell

M0
Objekte der Realität

UML-Metamodell (Kernkonzepte)



Konkrete Metaklassen

EMOF (Essential MOF)

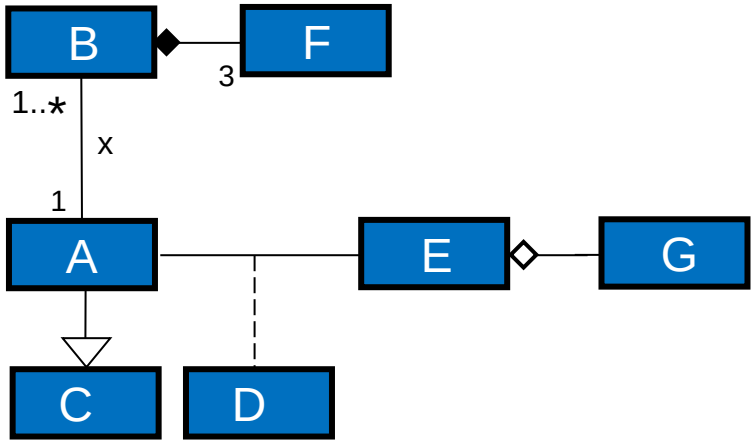
- ist eine Untermenge von MOF 2.0.
- Sie dient dazu, einfache Metamodelle auf einfache Weise – d. h. ohne die gesamte MOF verstehen zu müssen – erstellen zu können.
- EMOF ist auch weitestgehend kompatibel zum verbreiteten Metamodell Ecore,
- das im Eclipse Modeling Framework eingesetzt wird.

- *CMOF* (*Complete MOF*) umfasst dagegen den ganzen Sprachumfang.

Teil II: Strukturmodellierung

1. Klassen und Assoziationen (zweiter Eindruck)
2. Klassen und Verhaltensbeschreibungen (erster Eindruck)
3. Strukturelle Modellelemente/Entitäten (Überblick)
4. Klassendiagramm

Klassendiagramme



- A steht mit B in einer Beziehung namens x
  jedem A-Objekt sind 1 bis beliebig viele B-Objekte zugeordnet,
  jedem B-Objekt ist genau ein A-Objekt zugeordnet
- A ist eine Spezialisierung von C
- D ist eine Assoziationsklasse, die die Beziehung zwischen A- und E-Objekten beschreibt
  jeder A-E-Link ist durch ein D-Objekt charakterisiert,
  dieses hat zwei Attribute: A-Referenz, E-Referenz
- Jedes E-Objekt besitzt ein G-Objekt (Aggregation)
  G-Objekt ist Teil von A (könnte aber gleichzeitig Teil von etwas anderem sein)
- Jedes B-Objekt besitzt 3 F-Objekte als (Komposition)
  F-Objekt kann (max.) nur Teil vom B-Objekt sein

Klassen und Objekte

passive Klasse

aktive Klasse

Kunde

Kunde

name
titel
geburtstag
geschlecht

Kunde

name: **String**
titel: **String**
geburtstag: **Datum**
geschlecht: **Geschlecht**

alter(): **Integer**

Ticket
{abstract}

KommEinheit

partner: **KommEinheit**

«**signal**»
SigA(**integer**),
SigB(**float**)

op (**int**, **char**)

- beliebige **Detailierungsgrade** in der Darstellung einer Klasse möglich
  Tools sichern, dass in einem Namensraum keine Widersprüche zwischen Darstellungen einer Klasse auftreten

- Sichtbarkeit kann individuell eingeschränkt werden
  + public, # protected, - private, ~ package

Attribut-Belegung (Zustand zu einem Zeitpunkt) des Kunden-Objektes K1

K1:Kunde

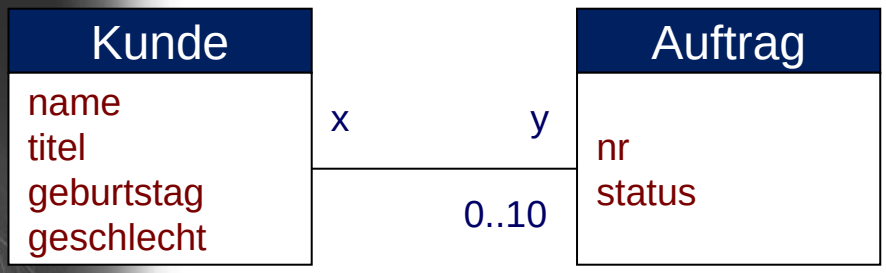
name = “Mustermann”
titel = “Dipl-Inf”
geburtstag= 10-09-1992
geschlecht= weiblich

:Kunde

anonymes Objekt

Dualität von Attributen und Assoziationsenden

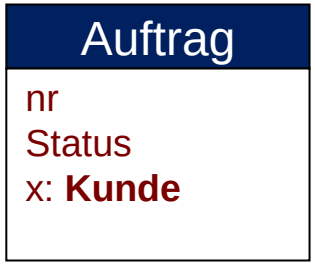
Sowohl **Attribut** als auch **Assoziationsende** sind im **Metamodell** Instanzen der Metaklasse **Property**



Lesart:
jedem Kunde-Objekt ist eine Kollektion y von Aufträgen zugeordnet

jedem Auftrag-Objekt ist genau ein Kunde-Objekt x zugeordnet

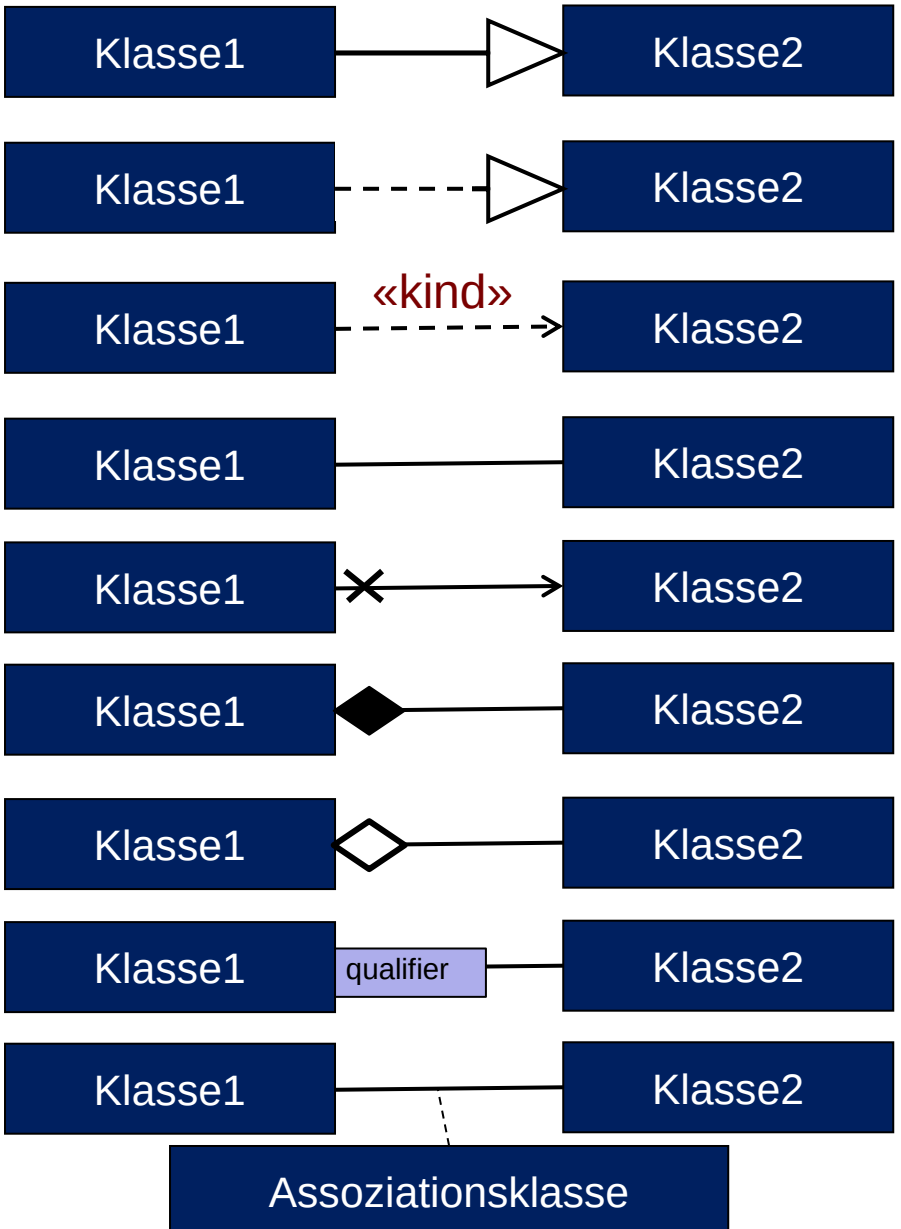
- *Das Assoziationsende x (vom Typ Kunde) dient der Navigation und entspricht implizit einem Attribut der Klasse Auftrag*
- *Das Assoziationsende y (vom Typ Menge über Auftrag) entspricht implizit einem Attribut von Kunde*



Kardinalität als Eigenschaften von Attributen/Assoziationsenden

falls Assoziationsenden nicht benannt worden sind (also wenn Angabe y fehlt)

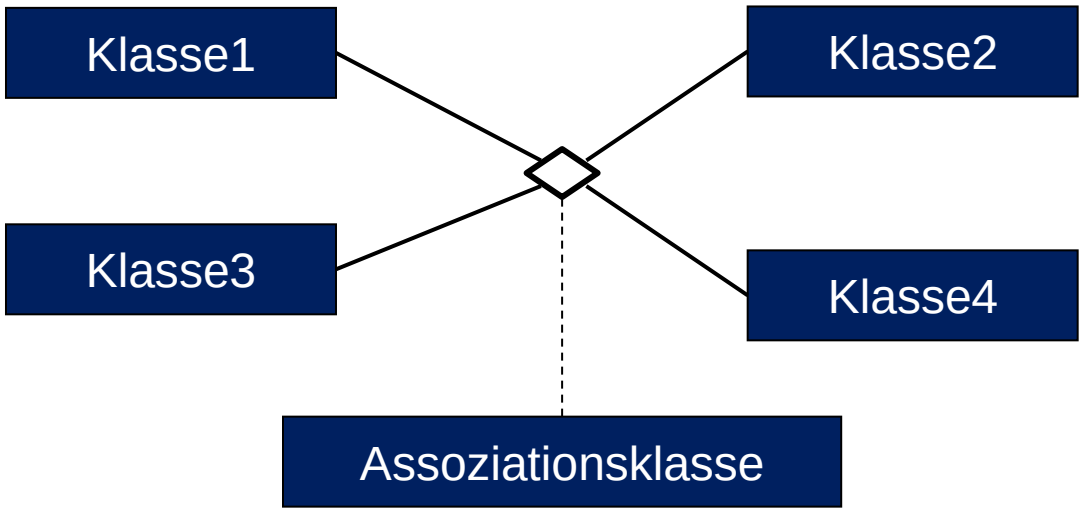
Beziehungen *zwischen Klassen bzw. Objektmengen*



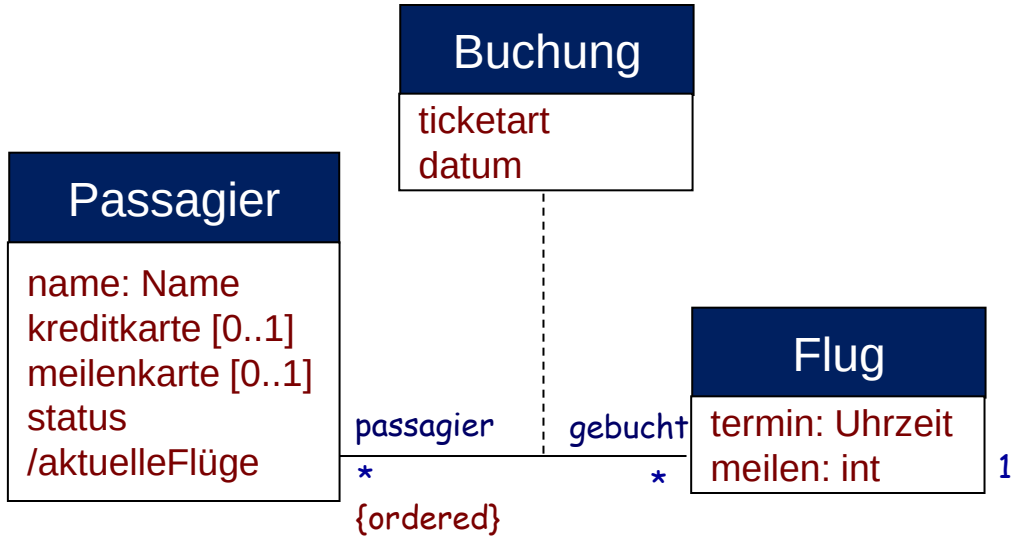
Assoziation

- beschreibt als Relation die gemeinsame Struktur und Semantik von Mengen von Objektverbindungen der beteiligten Klassen
- Objektverbindungen (Links) sind Instanzen einer Assoziation (eineAusprägung von Classifier)
- Eine Assoziation kann einen Namen haben
  (beschreibt worin oder weshalb eine Beziehung besteht)
- Die Enden tragen immer einen Namen
  (beschreiben die Rollen, die die Objekte der Klassen am jeweiligen Ende spielen)
  Ist kein Name angegeben, wird der kleingeschriebene Klassenname angenommen
- Für jede Rolle kann eine Reihe zusätzlicher Angaben gemacht werden:
  - Multiplizität (Standardwert=1),
  - Sichtbarkeit,
  - Eigenschaften
- Dualität von Assoziationsende und Attribut (jedes Rollenende könnte ein Attribut der gegenüberliegende Klasse oder Attribut einer evtl. Assoziationsklasse darstellen)

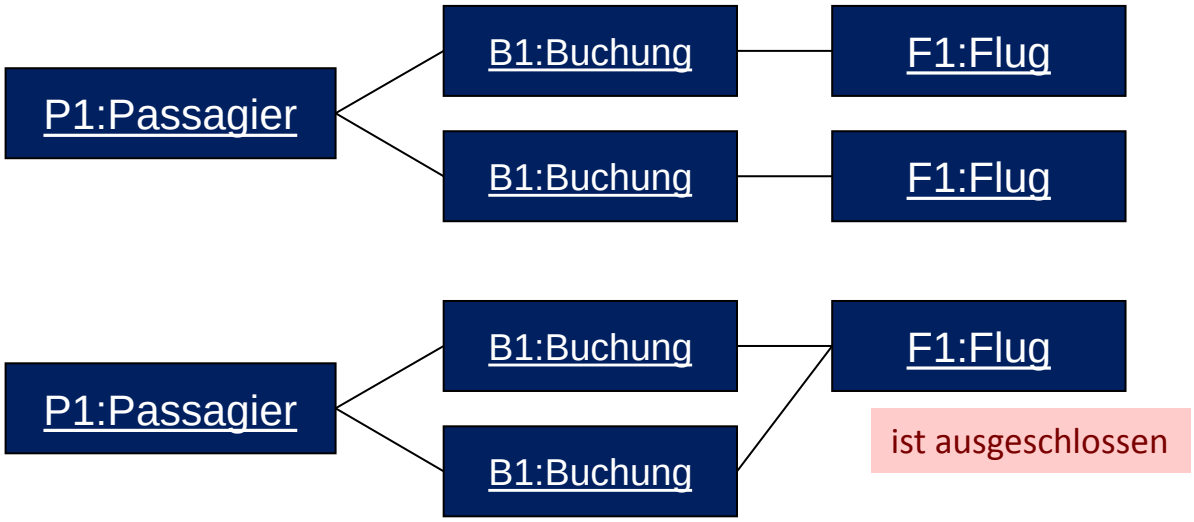
Mehrstellige Assoziation



Assoziationsklasse



jedes Verbindungspaar ist einmalig



ist ausgeschlossen