1.3 Objektorientierte Analyse & Entwurfsverfahren

1.3.1 UML Klassendiagramm, Objektdiagramm
1.3.2 UML Vererbung
1.3.3 UML Assoziationen
1.3.4 UML Use Case Analyse

BIBI 3.JG Wi
4(4)h

Schuljahr 2013/2014
BAT, KAUF

1.3.1 UML Klassendiagramm

Zur Darstellung von „*Struktur*“ in UML Diagrammen werden u.a. folgende Komponenten benutzt:

- Use Case (Anwendungsfall)
- Klasse
- Interface
- Objekt

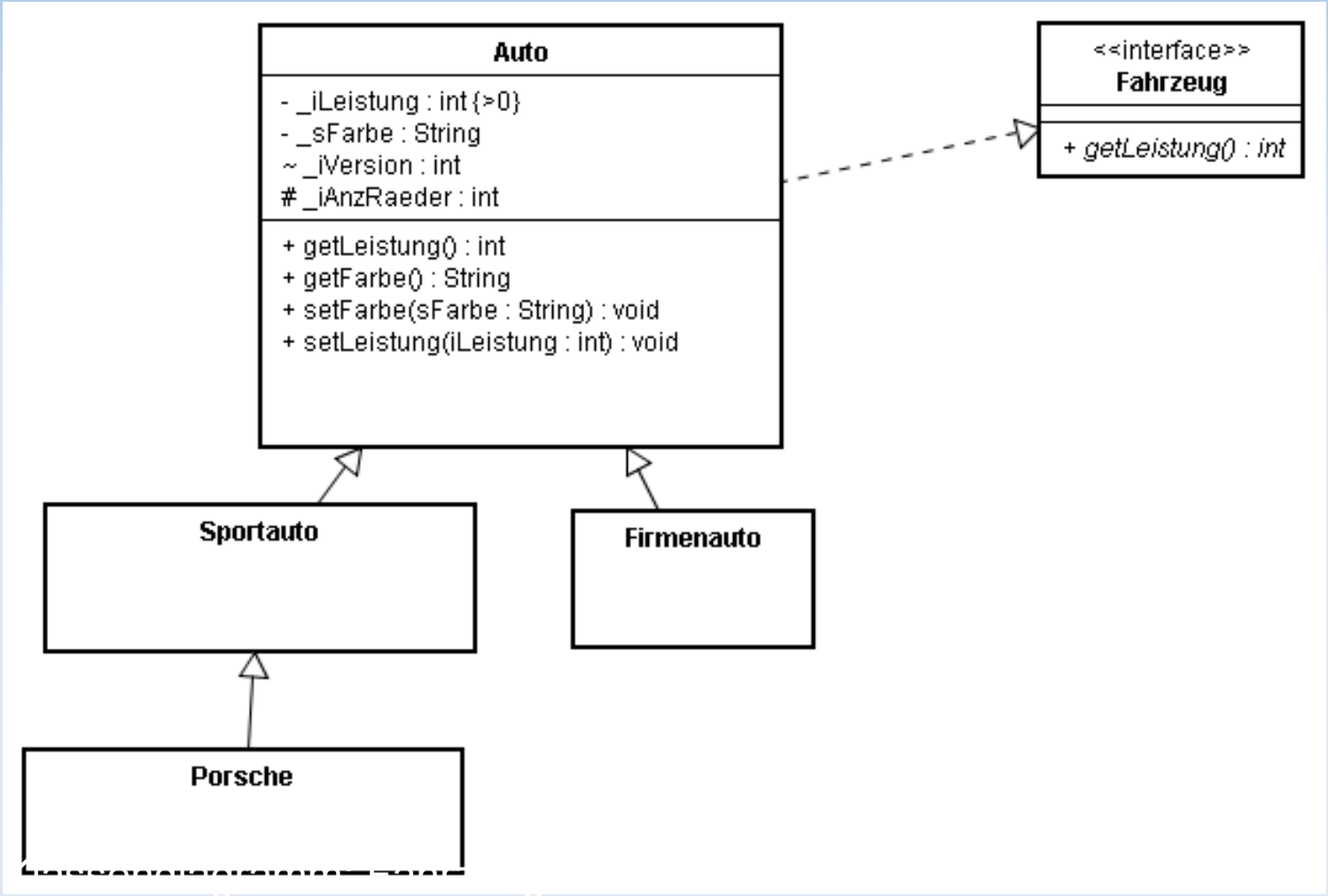
- Man spricht auch von *UML Strukturdiagrammen* im Gegensatz zu *UML Verhaltensdiagrammen.*

[http://de.wikipedia.org/wiki/Unified_Modeling_Language]

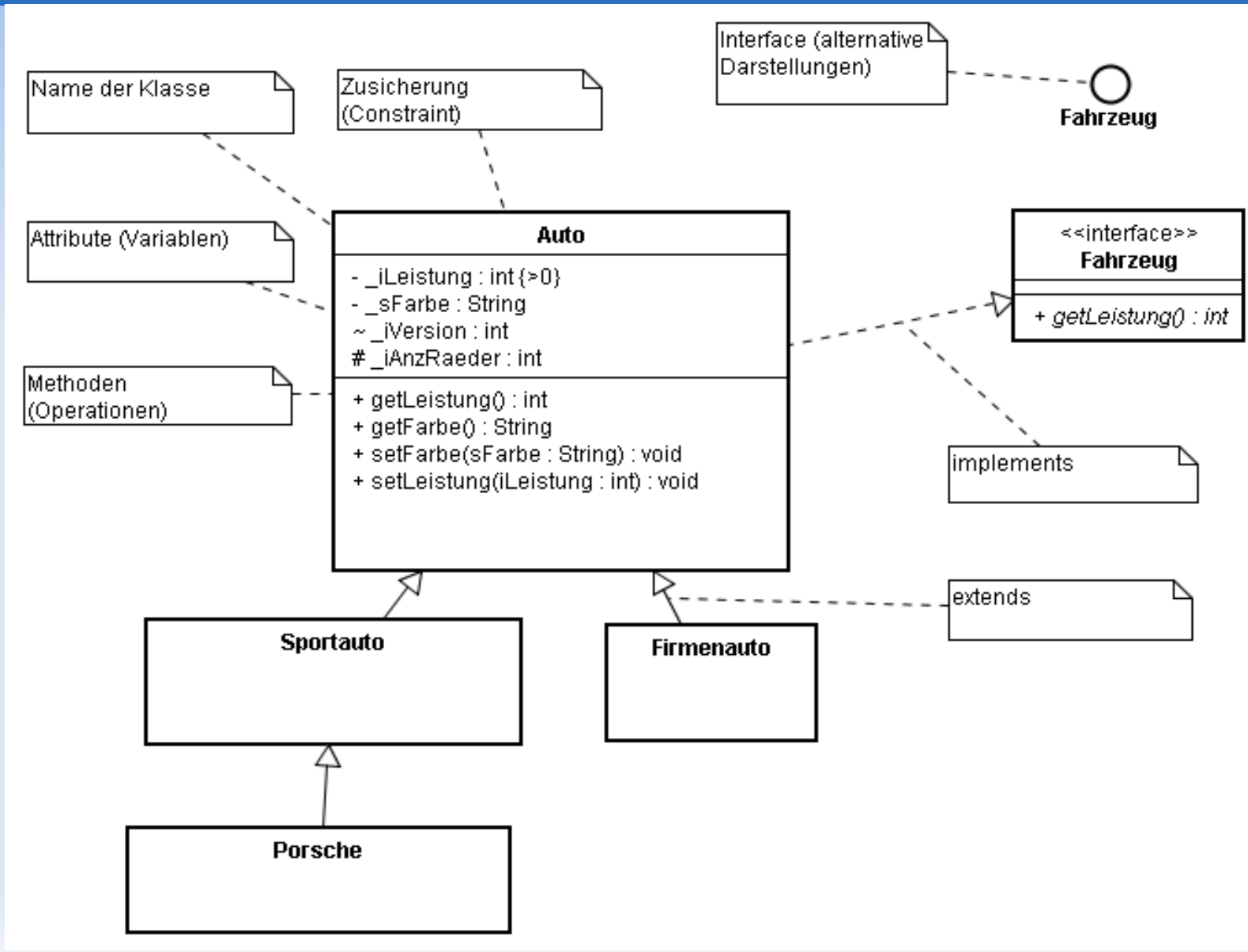
1.3.1 UML Klassendiagramm

- Klassendiagramme
  - beschreiben Klassen durch Angaben von:
    - Methoden (Operationen)
    - Attribute (Zustände, Struktur)
    - Beziehungen (Assoziationen / Relationen) zwischen Klassen
    - Zusicherungen (Randbedingungen)
  - Verwendung in Entwurfsphase des SEP
  - Zentrales Diagramm in UML
  - *Statisches* Diagramm

1.3.1 UML Klassendiagramm



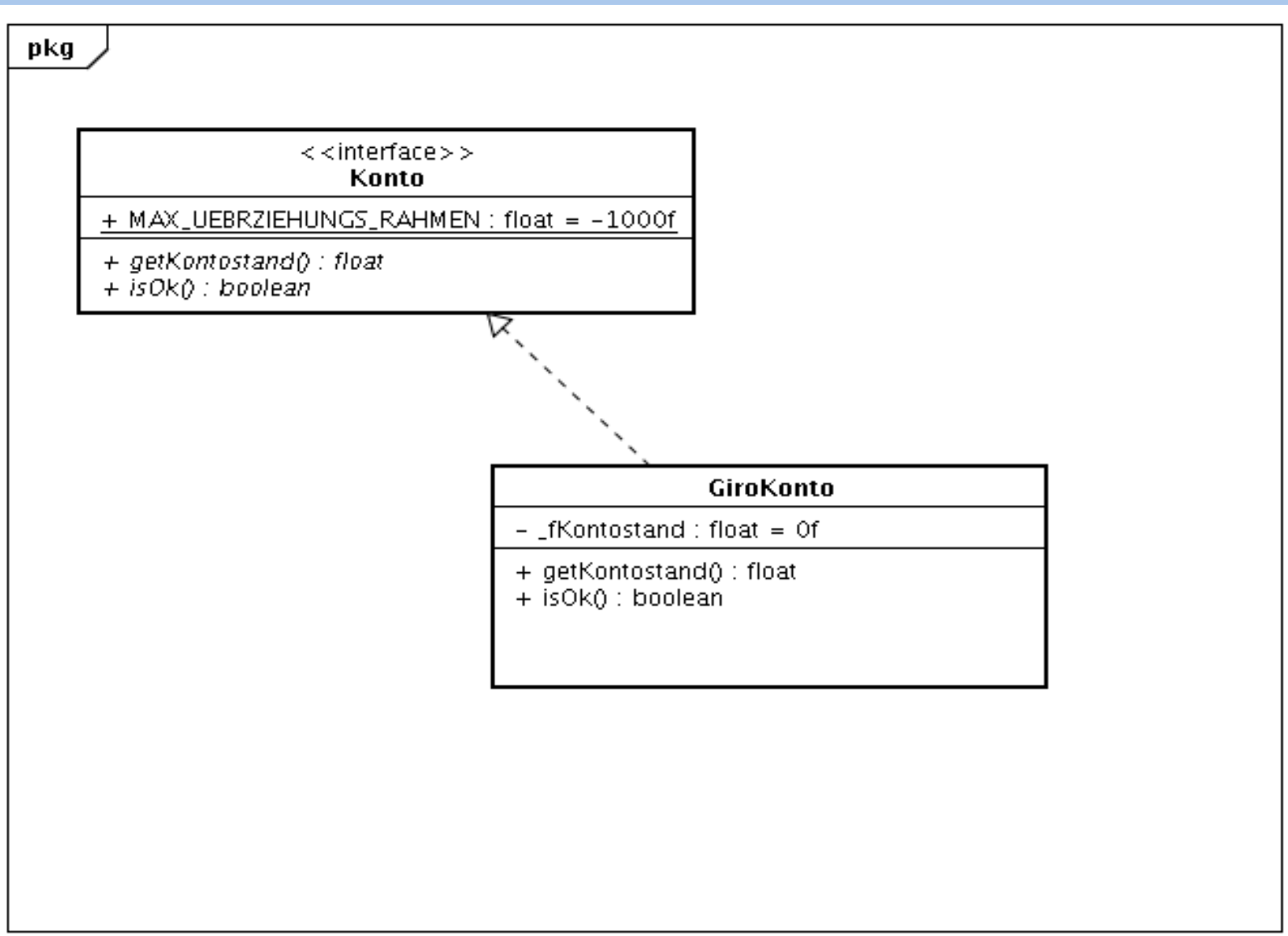
1.3.1 UML Klassendiagramm



1.3.1 UML Klassendiagramm

- Interfaces
  - Sind „spezielle Klassen“
  - legen Signaturen der Methoden fest
  - können Konstanten definieren
  - „garantieren“ die Kompatibilität zwischen Klassen, welche dasselbe Interface implementieren und nicht unbedingt voneinander erben

1.3.1 UML Klassendiagramm



1.3.1 UML Klassendiagramm

```
public interface Konto
{
    public static final float MAX_UEBRZIEHUNGS_RAHMEN = -1000f;

    /**
     * Liefert den Kontostand
     * @return    Kontostand in Euro
     **/
    public float getKontostand(); //abstrakte Signatur-Def.

    /**
     * Überprüft das Konto
     * @return    true, wenn Überziehungsrahmen
     *            eingehalten, sonst false
     **/
    public boolean isOk();

}
```

1.3.1 UML Klassendiagramm

```
public class GiroKonto implements Konto
{
    private float _fKontostand = 0f;

    // Implementierung der durch Konto definierten Methoden

    public float getKontostand()
    {
        return _fKontostand;
    }

    public boolean isOk()
    {
        return getKontostand() > MAX_UEBRZIEHUNGS_RAHMEN;
    }
}
```

1.3.1 UML Objektdiagramm

- Objektdiagramme
  - Gehört zu den *dynamischen* Diagrammen der UML
  - zeigt Ausprägungen der im Klassendiagramm modellierten Typen zu einem bestimmten Zeitpunkt (ggf. mit Werten für Variablen)

1.3.1 UML Objektdiagramm

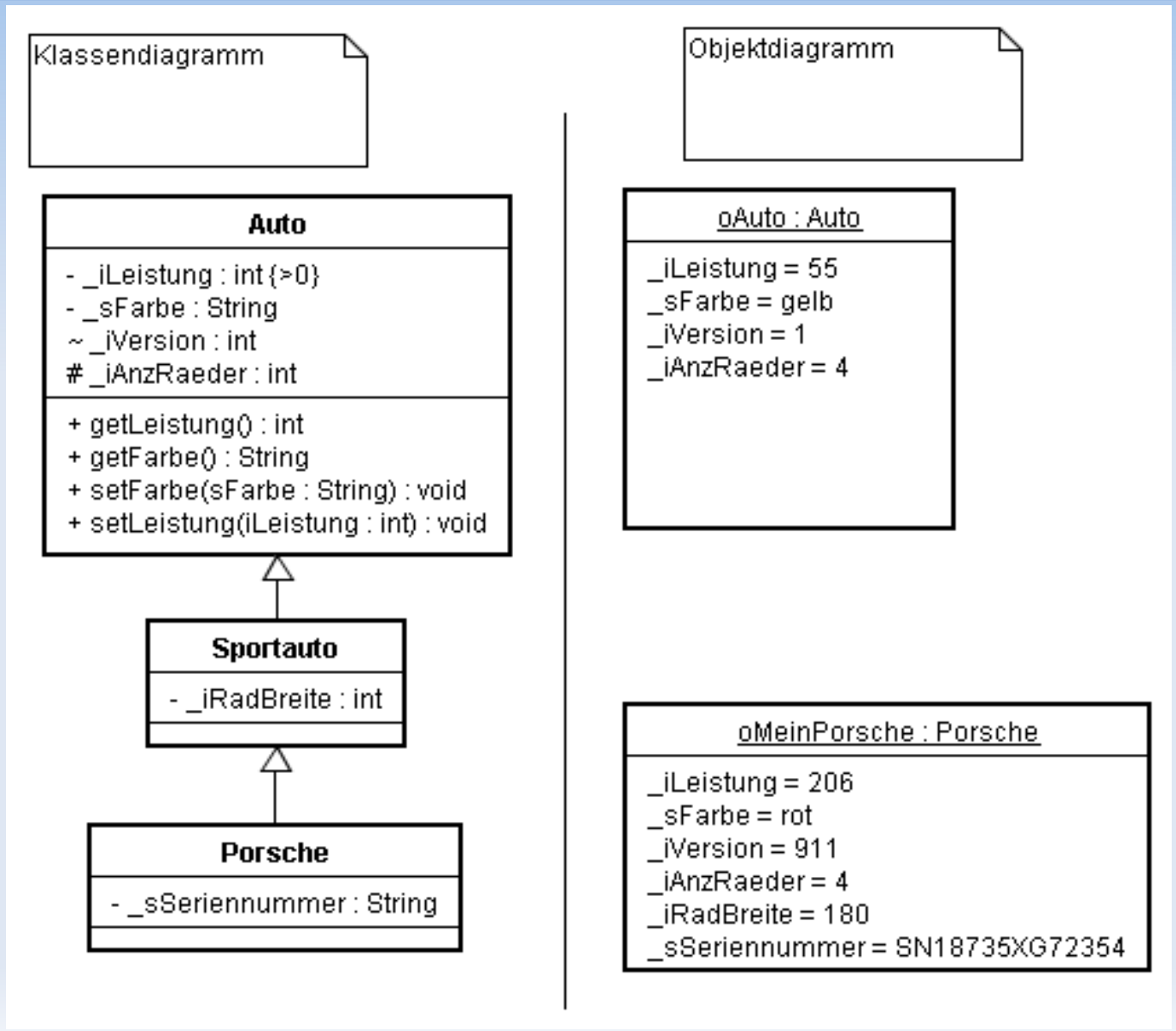
- Objektdiagramme
  - Bei der Modellierung kann das Objektdiagramm eingesetzt werden, um in einer konkreten „Momentaufnahme“ komplexere Strukturen zu betrachten.
    - Ein Objekt wird durch ein Rechteck repräsentiert.
    - Hinter dem Namen des Objektes folgt ein „:“ und danach der Klassenname
    - `Objekt:Klasse`

```
Auto oAuto = new Auto(); //Aufruf Konstruktor

oAuto.setFarbe(„gelb“);

oAuto.setLeistung(55);
```

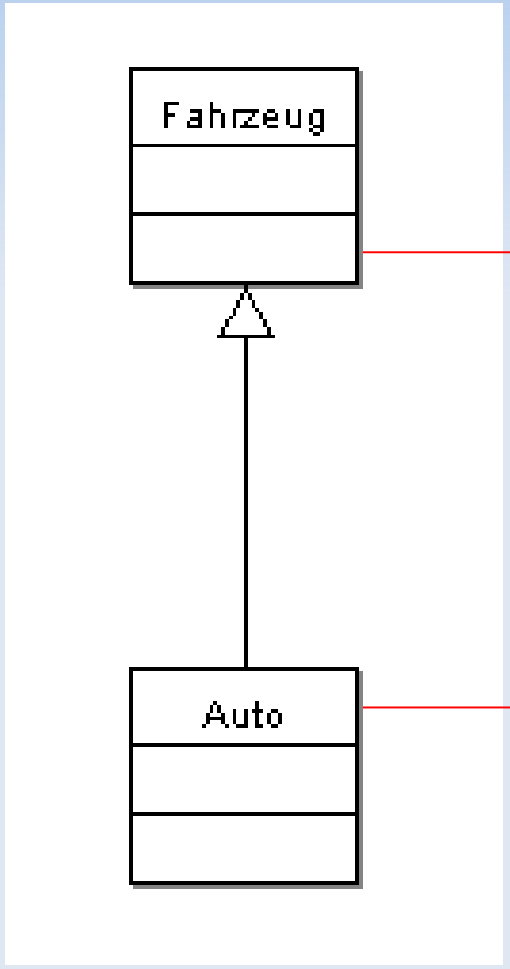
1.3.1 UML Objektdiagramm



1.3.2 UML Vererbung

- Entspricht einer „is-a“ Beziehung

Generalisierung

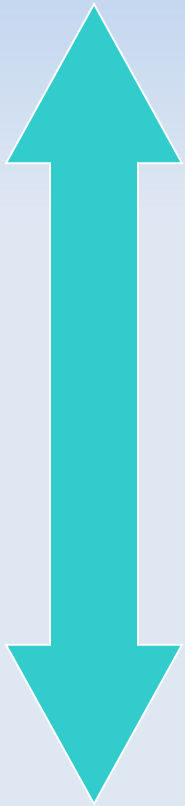


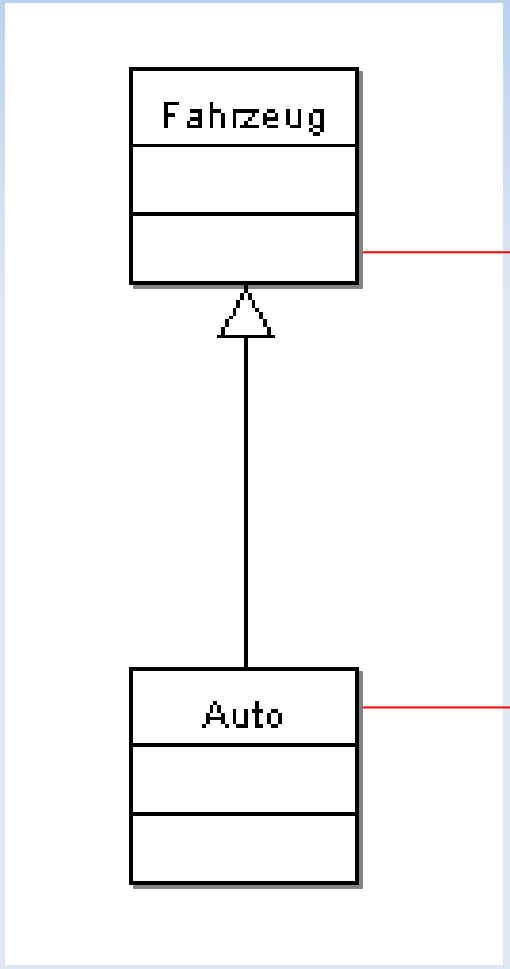
Spezialisierung

1.3.2 UML Vererbung

- „is-a“ Beziehung

Generalisierung





Vaterklasse/Basisklasse/
Superklasse

Sohnklasse/abgeleitete
Klasse

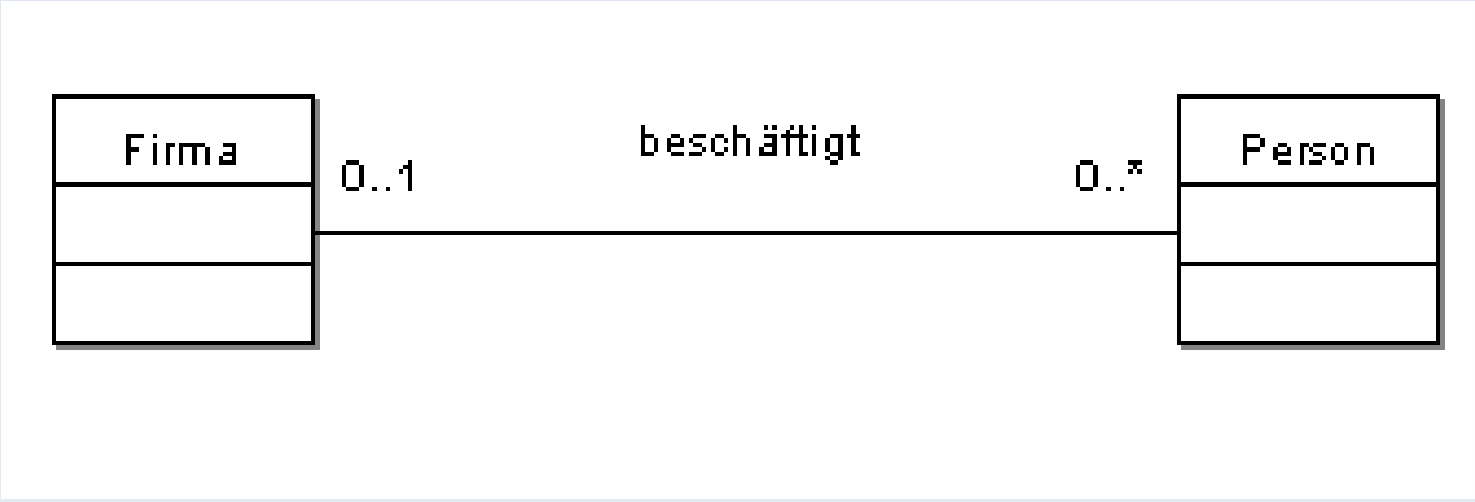
Spezialisierung

1.3.3 UML Assoziationen

- Klassen und Objekte stehen i.A. zu anderen Klassen/Objekten in Beziehung (Betrachtung d. Kontext)
- ➔Assoziationen im Klassendiagramm/Objektdiagramm
  - Analogie zu *ER-Modellierung bei DB*
  - Eine Assoziation stellt den Zusammenhang zwischen Klassen dar und beschreibt damit die allgemeine Bedeutung und Struktur verschiedenster Typen von „Verbindungen“ zwischen Objekten.
  - Assoziationen sind der Mechanismus, der es den Objekten erlaubt untereinander zu kommunizieren.
  - Sie beschreiben die Verbindungen zwischen verschiedenen Klassen (die Verbindung zwischen den eigentlichen Objekten werden als Objektverbindungen oder als Link bezeichnet).

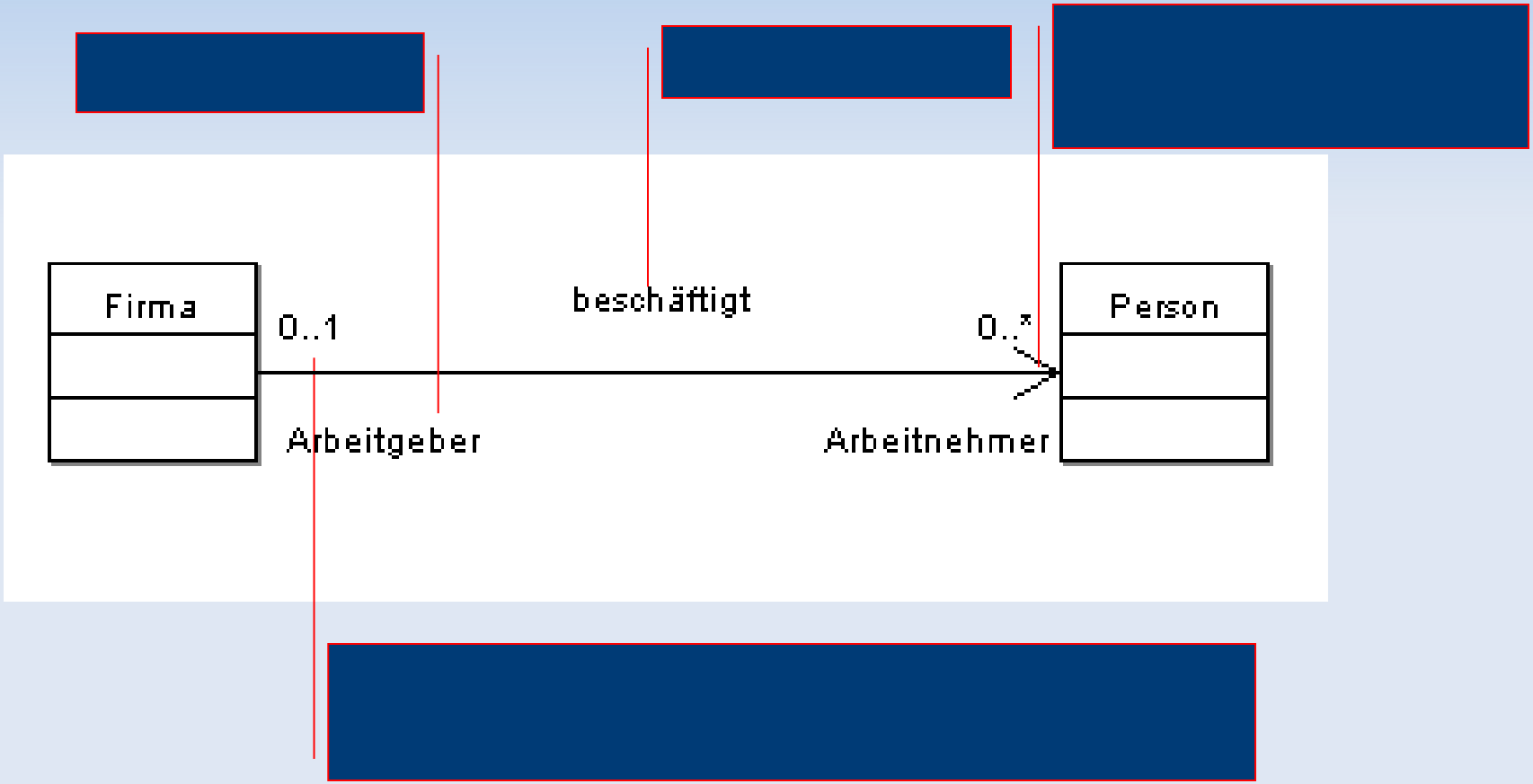
1.3.3.1 UML Assoziation

- „Assoziation ist eine Beziehung zwischen verschiedenen Objekten einer oder mehrerer Klassen“
  - Modellelement der UML
  - Analogien zu Beziehungen im ER-Modell
  - Meist binäre Assoziation (2 Enden)



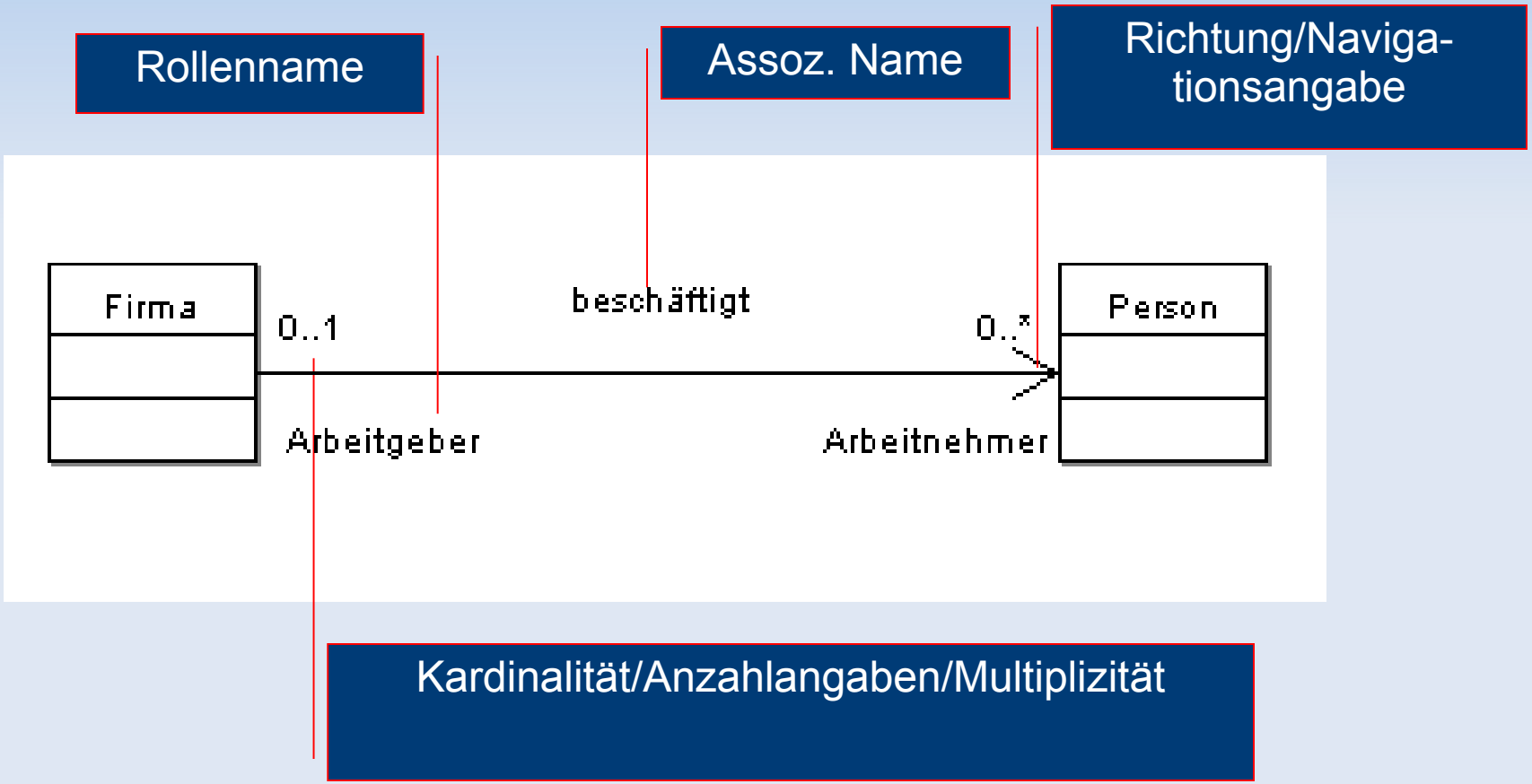
1.3.3.1 UML Assoziation

- Eigenschaften einer Assoziation



1.3.3.1 UML Assoziation

- Eigenschaften einer Assoziation



1.3.3.1 UML Assoziation

- Kardinalität/Multiplizität
  - Gibt an, mit wievielen Objekten einer gegenüberliegenden Klasse ein Objekt assoziiert sein kann
  - Über die Bandbreite 0..* wird das Minimum bzw. Maximum angegeben
  - Ist das Minimum = 0, ist die Beziehung optional!

1.3.3.1 UML Assoziation

- Assoziationsname
  - Beschreibt die Beziehung näher (Verb)
- Rollennamen
  - Beschreiben genauer, welche Rolle die jeweiligen Objekte in der Beziehung einnehmen

1.3.3.1 UML Assoziation

- Verwendungs-/Aufrufungsbeziehung
  - „einfache Beziehung“
  - Objekte werden als lokale Variablen oder Methodenargumente verwendet

1.3.3.1 UML Assoziation

- Navigierbarkeit
  - Eine Assoziation bildet eine Art Brücke zwischen zwei Typen:
    - startet man bei der Instanz des einen beteiligten Typs kann man über eine Objektbeziehung zur Instanz des zweiten Typs navigieren.
    - Drei Arten, wie die Navigierbarkeit festgelegt werden kann

[http://de.wikipedia.org/wiki/Assoziation_(UML)]

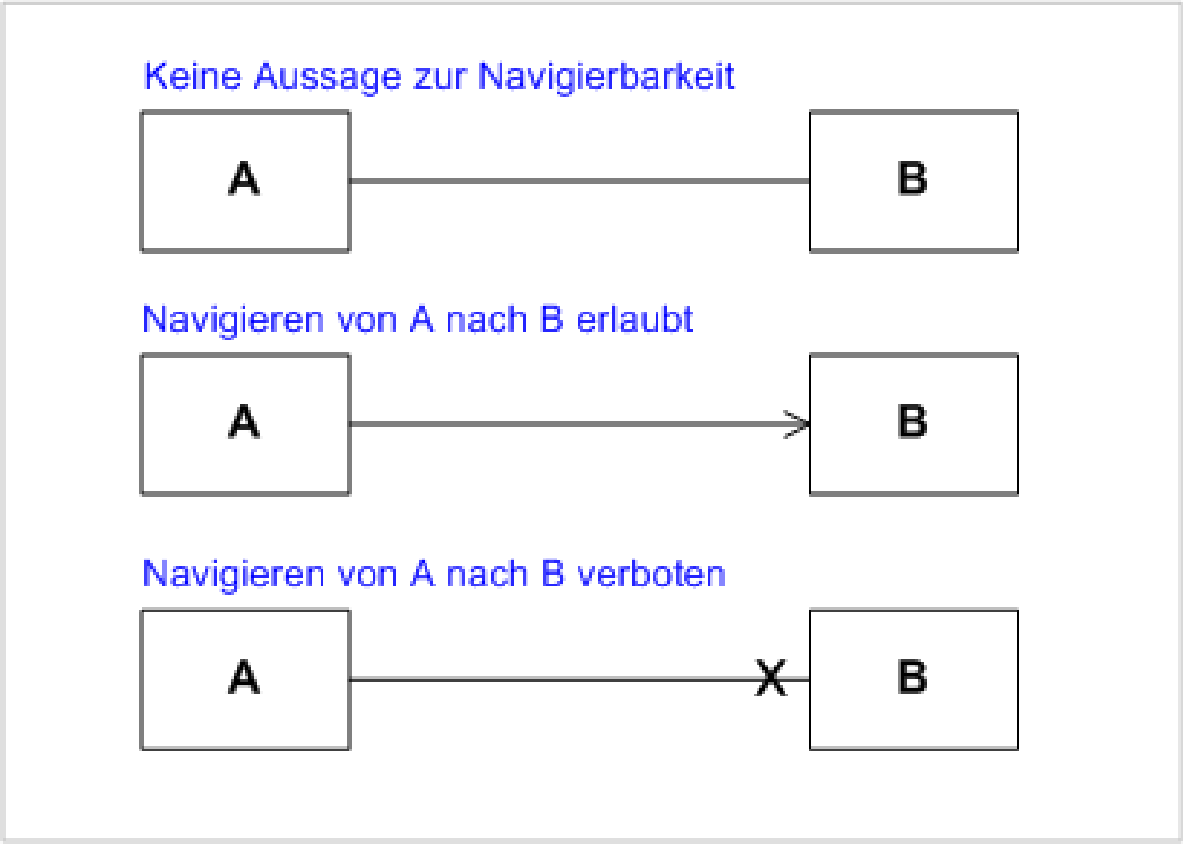
1.3.3.1 UML Assoziation

- Navigierbarkeit
  1. *Keine Aussage zur Navigierbarkeit*. Das Modell macht keine Aussage zur Navigierbarkeit. Sie ist unspezifiziert und soll erst zu einem späteren Zeitpunkt, zum Beispiel beim Softwaredesign, definiert werden.
  2. *Erlaubte Navigation*. Das Modell erlaubt die Navigation über das Assoziationsende.
  3. *Nicht erlaubte Navigation*. Das Modell verbietet die Navigation über das Assoziationsende.

[http://de.wikipedia.org/wiki/Assoziation_(UML)]

1.3.3.1 UML Assoziation

- Navigierbarkeit



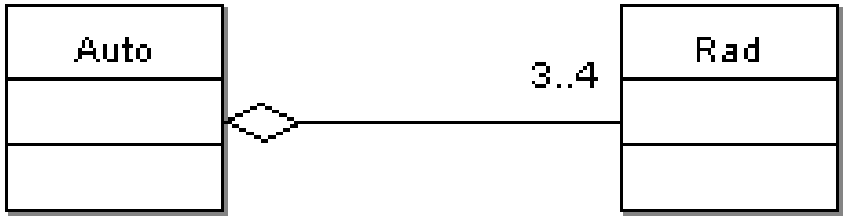
[http://de.wikipedia.org/wiki/Assoziation_(UML)]

1.3.3.2 UML Komposition

- „part-of“ Beziehung (Ganzes/Teil)
  - Komposition:
    - Die Einzelteile sind vom Aggregat (dem „Ganzen“) abhängig.
    - Wenn das Ganze gelöscht wird, werden immer auch alle Einzelteile gelöscht (Existenzabhängigkeit!).
    - Darstellung durch ausgefüllte Raute (beim Ganzen).

1.3.3.2 UML Komposition

- „part-of“ Beziehung (Ganzes/Teil)
  - Aggregation:
    - Zusammensetzung eines Objektes aus Einzelteilen. Das Ganze handelt stellvertretend für seine Teile.
    - Darstellung durch nicht ausgefüllte Raute (beim Ganzen)



1.3.4 UML Use Case Analyse

Was ist ein Use Case Diagramm?

Beschreibt das Zusammenwirken von *Personen* mit einem *System*.

Anwendungen die ein Benutzer mit dem System macht (=*Anwendungsfalldiagramm*)

Bedeutend in der Anforderungsanalyse:

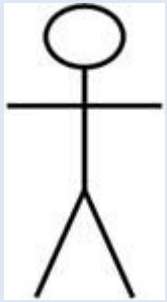
- Beschreibung der Funktionalität des zu entwickelnden Systems
- Ein Use Case führt immer zu einem erkennbaren Ergebnis
- Details des Systemverhaltens werden nicht betrachtet
- Beschreibung aus Benutzerperspektive
- Frage: Wer soll das System benutzen?
- Frage: Was soll das System für den User tun?

1.3.4 UML Use Case Analyse

Was ist ein Akteur?

Personen werden als *Akteure* bezeichnet.

Die Beschränkung der Akteure auf Menschen kann verallgemeinernd auch aufgehoben werden, und die Akteure können damit auch andere *Systeme* sein.

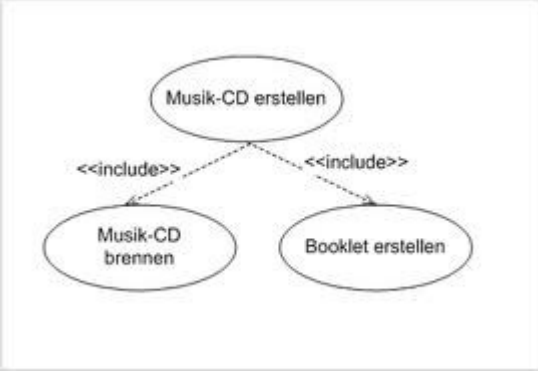


1.3.4 UML Use Case Analyse

Was sind Use Cases?

Unter einem Use-Case wird eine *typische Handlung (Aktionsfolge)* verstanden, die ein Benutzer mit einem *System* ausführt, z. B. "Aktienkauf".

==> Soll einen groben Überblick geben



1.3.4 UML Use Case Analyse

Was sind Use Cases?

Im Use-Case-Diagramm werden die Use-Cases als Ellipsen eingezeichnet

Verbindungen zwischen den Use-Cases und der Akteure werden durch Linien hergestellt

- Damit wird angezeigt, welche Akteure an dem entsprechenden Use-Case beteiligt sind.
- Es sind nur binäre Assoziationen erlaubt (das bedeutet, dass an einer Assoziation genau zwei Partner beteiligt sind)
- In seltenen Fällen nutzt man gerichtete Assoziationen, die darstellen, wer von beiden die Kommunikation anstößt.

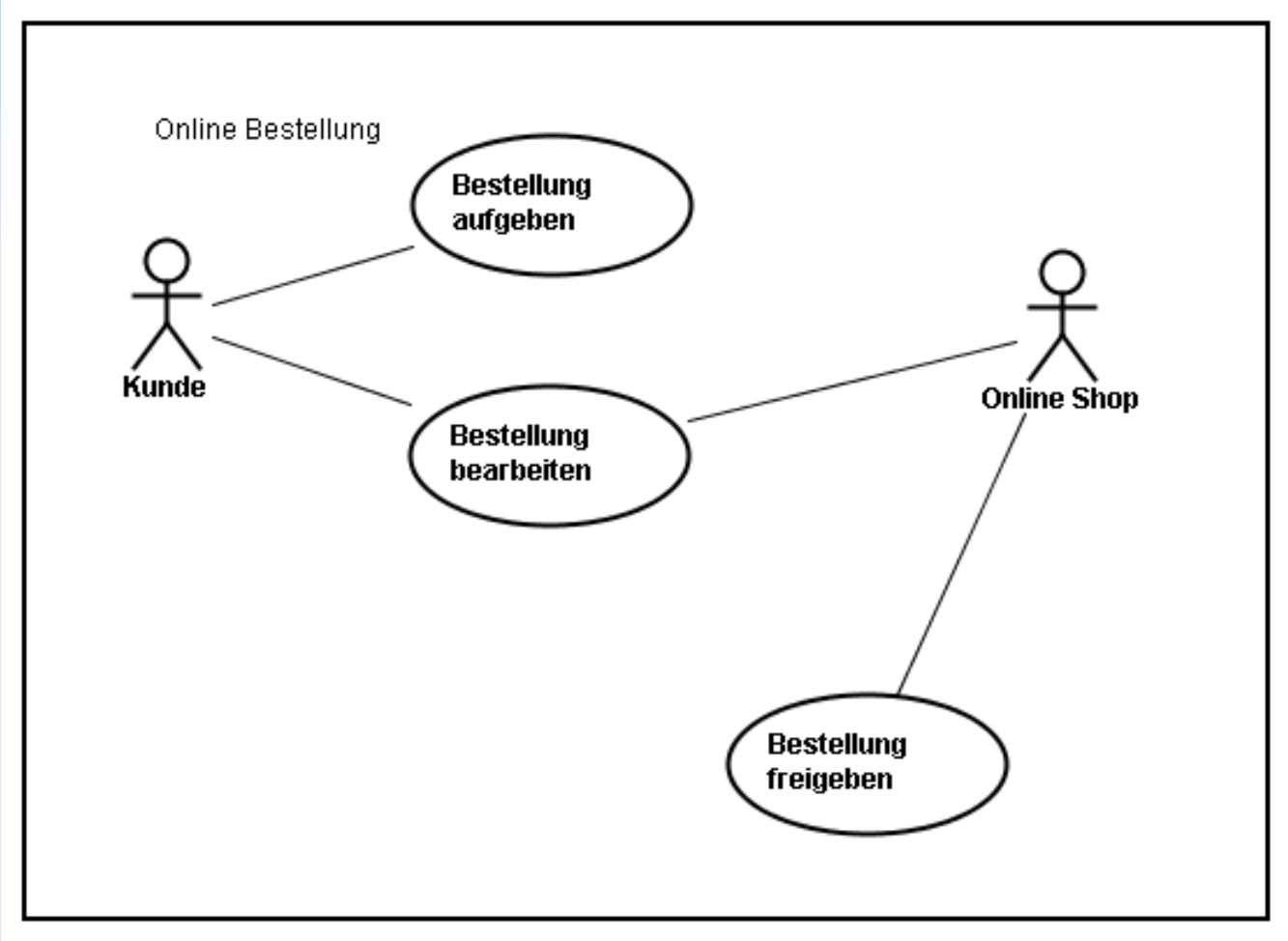
1.3.4 UML Use Case Analyse

Für wen ist ein Use Case Diagramm?

Es ist zur schnellen Übersicht für den *Auftraggeber/Kunden* oder den *Projektleiter* oder zum *schnellen Einschulen* eines zusätzlichen Projektbeteiligten.

1.3.4 UML Use Case Analyse

Beispiel

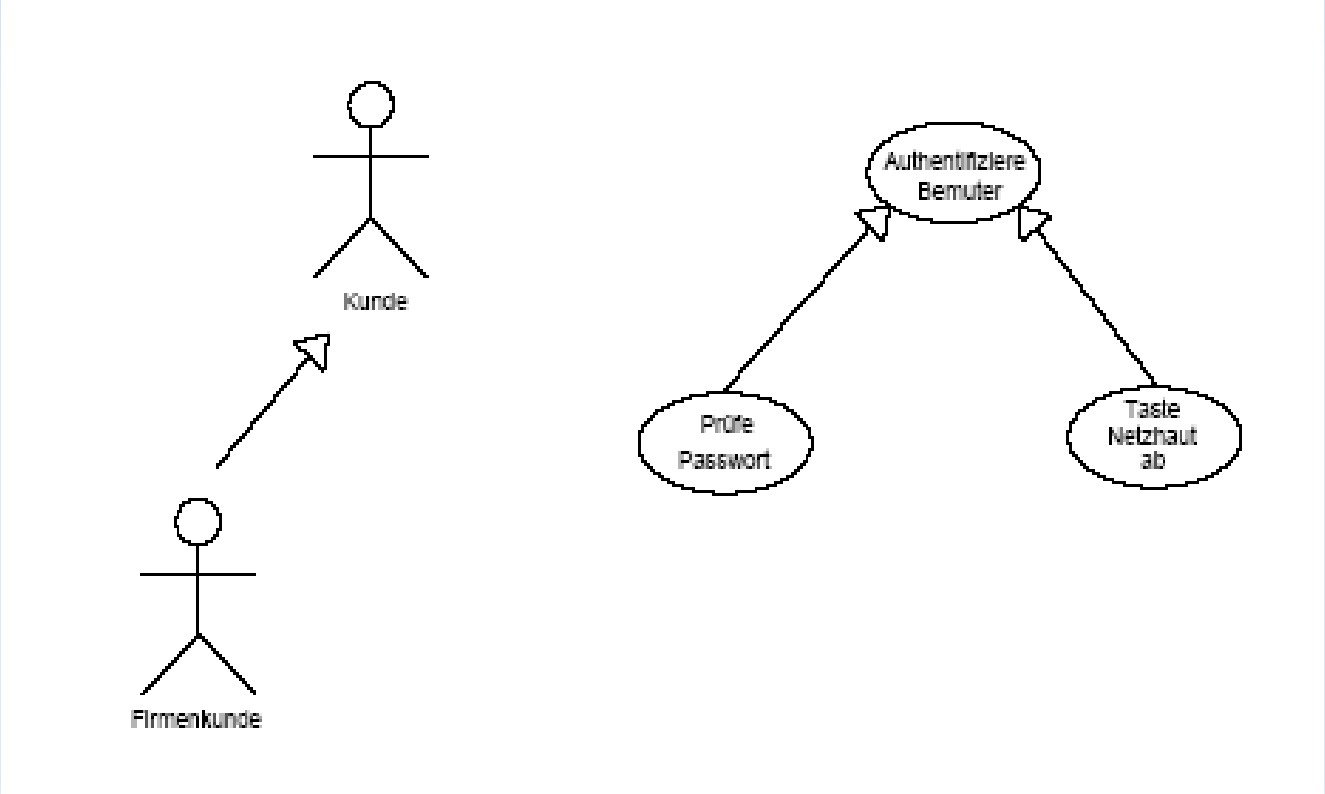


1.3.4 UML Use Case Analyse

Generalisierung / Spezialisierung

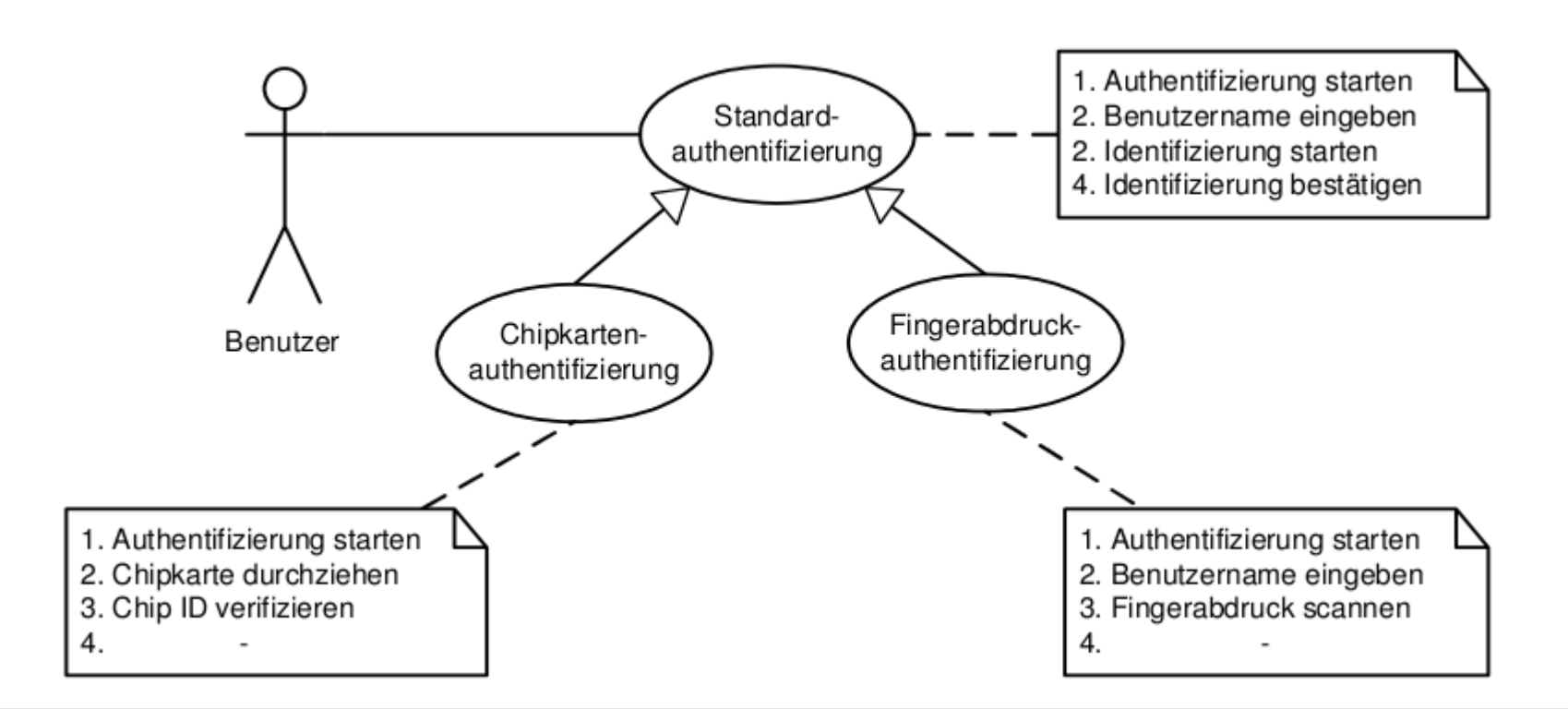
bei Akteuren & Use Case möglich

<<extend>>



1.3.4 UML Use Case Analyse

Beispiel



1.3.4 UML Use Case Analyse

Erweiterte Elemente

Abhängigkeit: - - - >

Änderung des unabhängigen Dinges (an der Pfeilspitze) bewirkt Änderung des abhängigen Dinges.

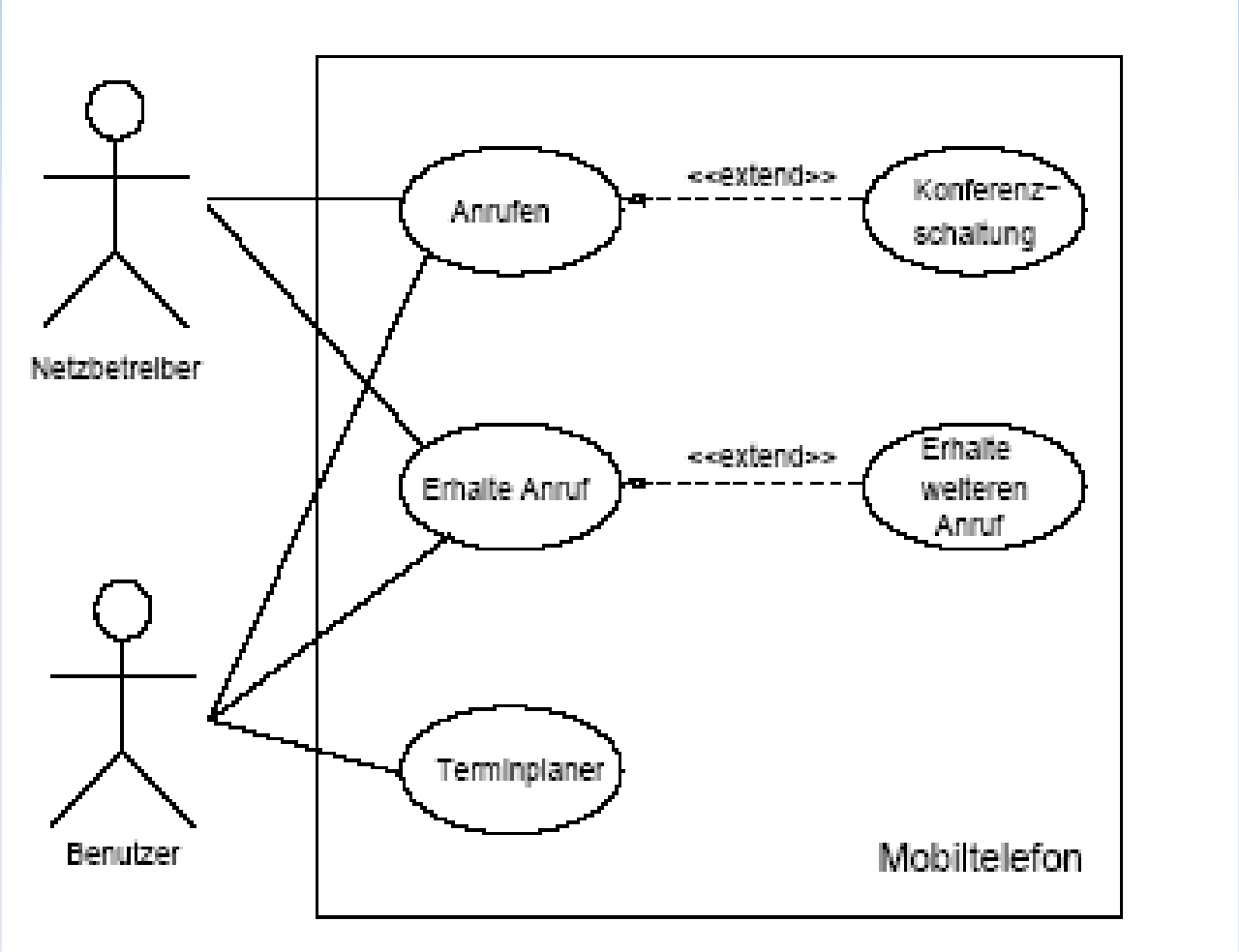
Stereotypen: << ... >>

z.B. <<include>> (bzw. <<uses>> in UML 1.2)

oder <<extend>> (Generalisierung/Spezialisierung)

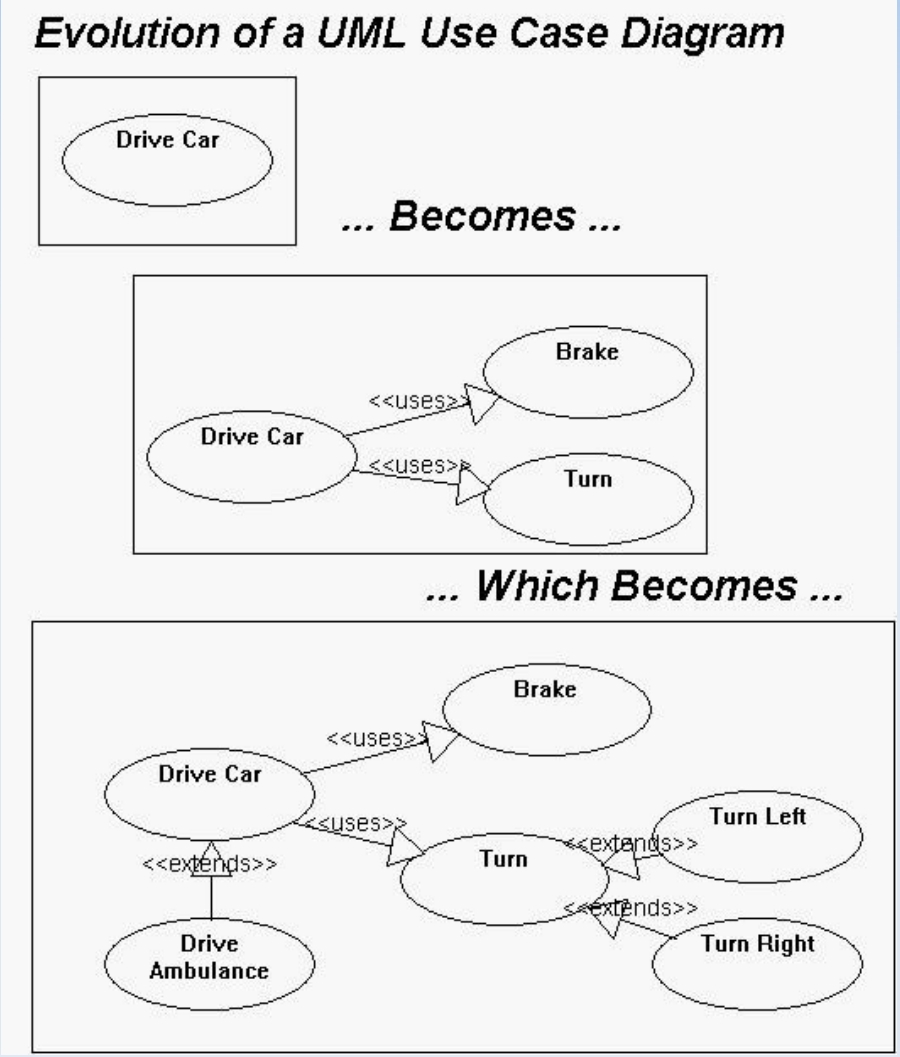
1.3.4 UML Use Case Analyse

Beispiel: Mobiltelefon



1.3.4 UML Use Case Analyse

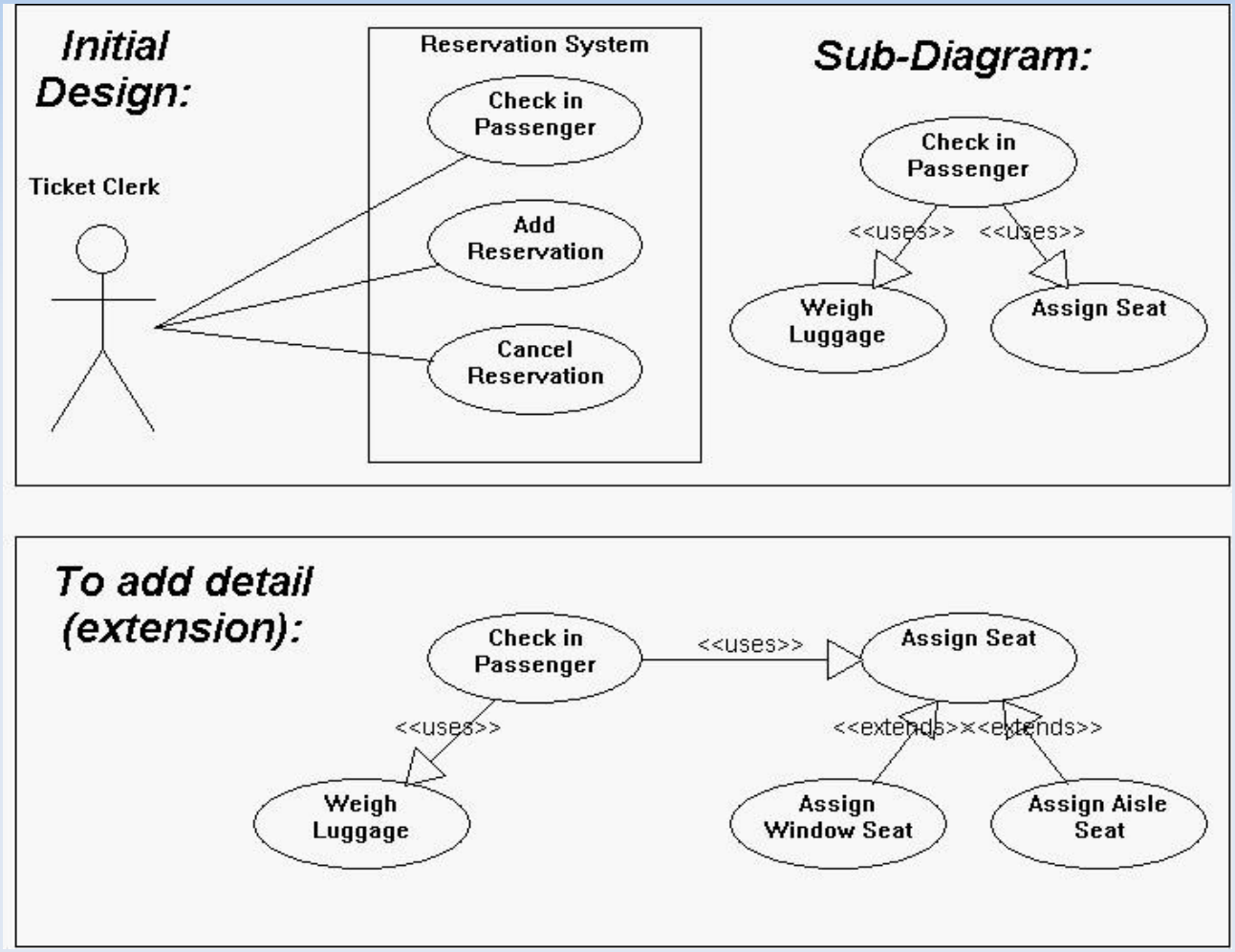
Beispiel: Evolution eines Use Cases



[http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html]

1.3.4 UML Use Case Analyse

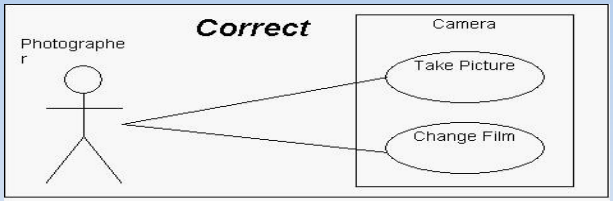
Beispiel: <<uses>> and <<extends>>

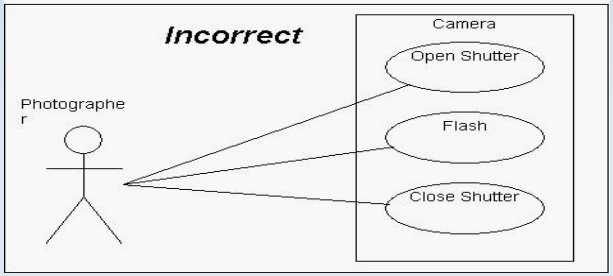


[http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html]

1.3.4 UML Use Case Analyse

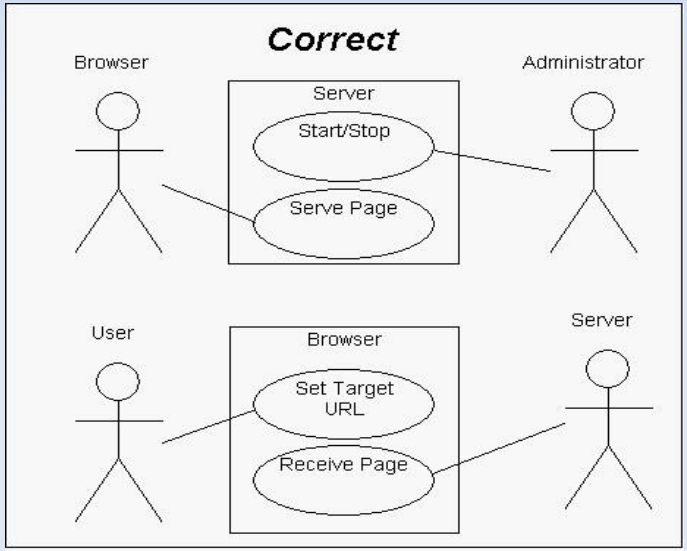
DO's and DON'Ts:

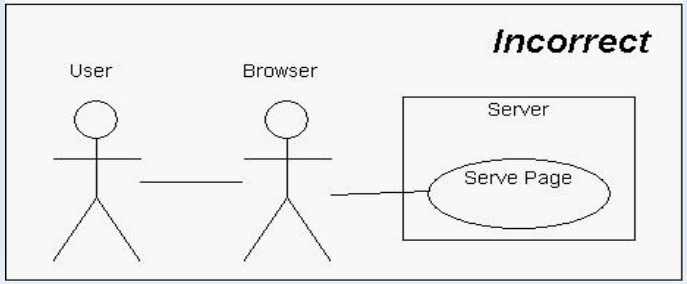




Fokus auf Verhalten, High-Level Beschreibung der möglichen Abläufe

Keine direkten Interaktionen zwischen Akteuren





[http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html]

1.3.4 UML Use Case Analyse

Schriftl. Beschreibung eines Use Case

- Name und Identifier
- Beschreibung, was im Anwendungsfall passiert
- Beteiligte Akteure
- Normalfall (normal flow)
- Alternativer Ablauf (alternative flow)
- Ausnahmen (exceptions)
- Vorbedingung / Preconditions / Auslöser
- Nachbedingung / Ergebnis