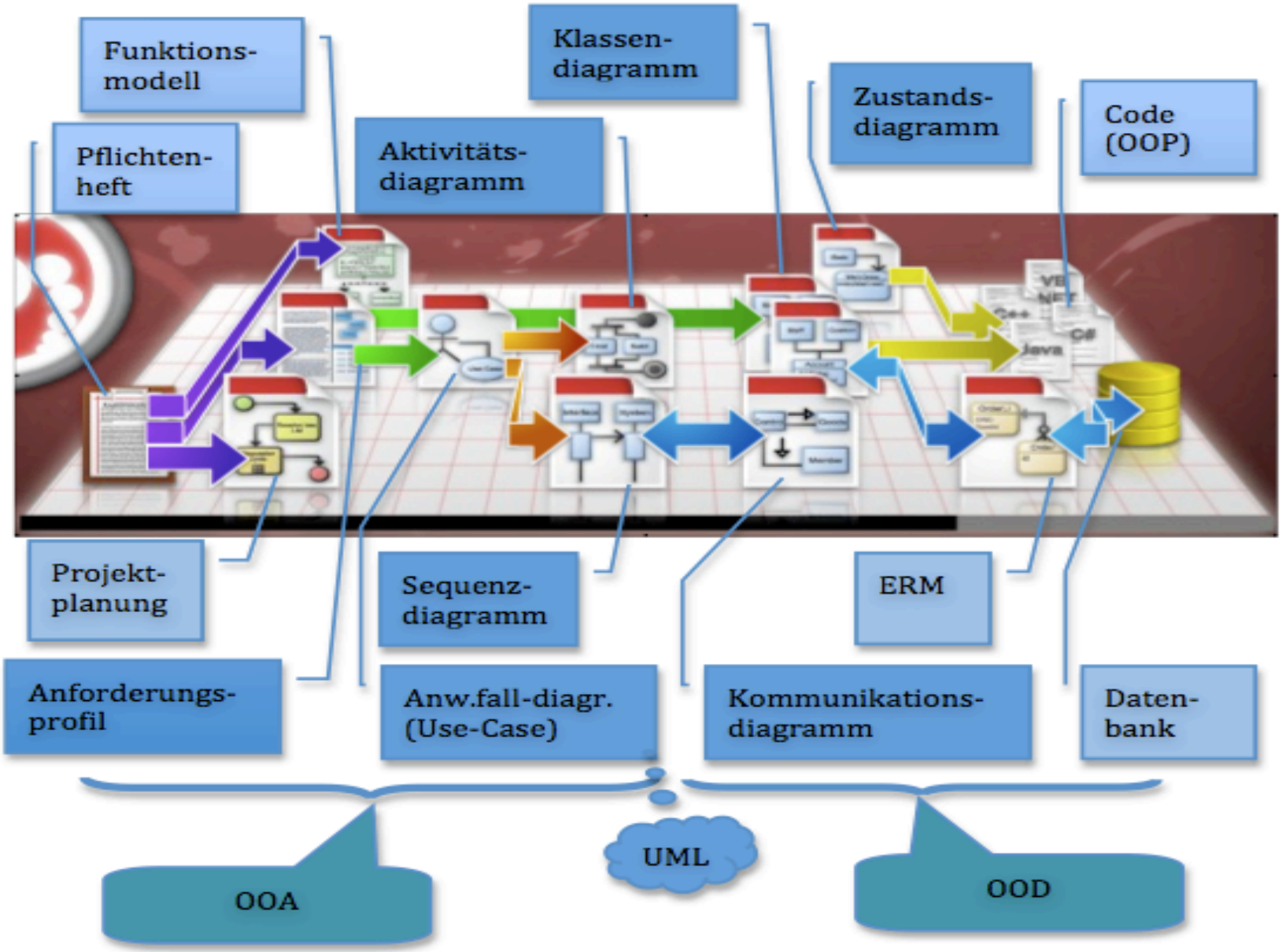
Objektorientiert Analyse (OOA) und Design (OOD)



Inhalt

- Einführung: Analyse – Design
- Funktionsmodell
- Use-Cases (UML2.5+)
- Beziehungen (UML2.5+)
- Sequenzdiagramme (UML2.5+)
- Aktivitätsdiagramme (UML2.5+)

Unser Vorgehen

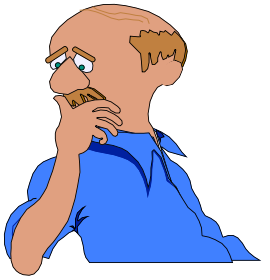
1. Kundenauftrag
2. Fach-Analyse
   Analyse der Prozesse, des Fachbereiches
   → Anforderungsanalyse (UC) → Pflichtenheft → Projektplanung (M306)

3. **OO-Analyse**
   Anforderungsprofil und Funktionsmodell (evtl. GUI, Testpläne etc.)
   → Anwendungsfall verstehen (Use-Case)
   → Kritische Sequenzen analysieren
   → Aktivitäten analysieren
   → Kunde und fachliche Anforderungen stehen im Vordergrund

4. **OO-Design** (=Entwurf)
   → Verfeinerung Modell, Struktogramme, Klassendiagramm, ERM, etc.
   → Vorbereitung für Entwicklung
   → Interne Muster & Schnittstellen stehen im Vordergrund

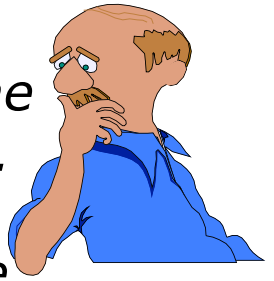
5. Implementierung zu zweit (pair programming, siehe Kap. 13.4)
6. Testen

Auftrag: „Entwickeln Sie eine Liftsimulation“



Auftrag: „Entwickeln Sie eine Liftsimulation“

- **Kunde:**
  In unserem Parkhaus mit drei Liften (nebeneinander) soll eine Steuerung die Personen noch oben und unten transportieren.
  Die Personen können auf jedem Stockwerk über die Anforderungstasten „Nach oben“ und/oder „Nach unten“ eine Kabine anfordern.
  Die Kabinen werden durch die Anforderung auf dem Stockwerk angehalten und lassen die Personen durch die geöffnete Tür ein. Die Steuerung optimiert automatisch und wählt dazu eine der drei Kabinen aus.
  Die eingetretenen Person(en) wählen Zielstockwerke an und werden dorthin transportiert.



- Sie sind für die Applikations-Logik zuständig. Was machen Sie nun?

- **Ziel: Entwurf einer Anwendung von Grund auf Werk–zeuge für Analyse und Design anwenden**
- **Vorbereitung auf die Projektarbeit LB2**

Methodisches Vorgehen

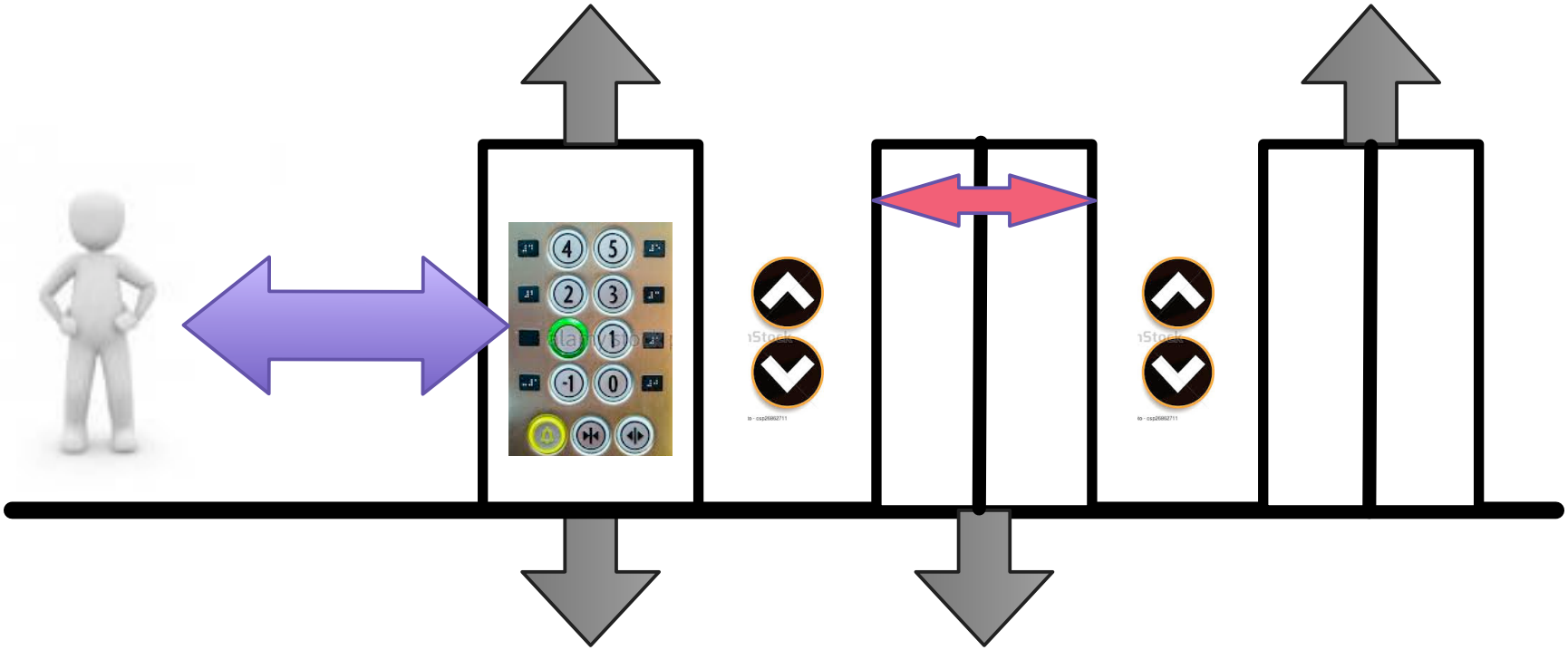
- Die ***Anforderungsanalyse*** kann Geschichten oder Abläufe umfassen, wie Leute die Anwendung einsetzen. Diese können als *Use Cases* (*Anwendungsfälle)* geschrieben werden. Use Cases sind kein objektorientiertes Artefakt, sondern einfach niedergeschriebene Abläufe. Sie sind ein beliebtes/gängiges Werkzeug der Anforderungsanalyse.
  Beispiel: „Die Person kann eine Liftkabine anfordern...“

- Bei der ***objektorientierten Analyse*** liegt die Betonung darauf, die Objekte in dem Problembereich zu finden und zu beschreiben.
  Beispiel: Bei einer Liftsteuerung sind z.B. wichtige Konzepte `Liftkabine`, `Anforderung`, `Transport` und `Person`.

- Beim ***objektorientiertem Design*** liegt die Betonung darauf, geeignete Softwareobjekte und ihr Zusammenwirken zu definieren.
  Beispiel: Ein Lift-Objekt (Software-Umsetzung) kann ein `zielStockwerk`-Attribut und eine `anfordern`-Methode haben.

** ANALYSE **
Funktionsmodell Liftsimulation

- Zweck: Bildliche Darstellung der Problemstellung (eher statisch, funktional)
- Fokus sinnvoll wählen: z.B. Übersicht, Beziehungen, EVA, ...
- Methode / Darstellung: frei wählbar, vorzugsweise Linien und Kästchen



** Analyse **
Use-Case (Anwendungsfall) Diagramm

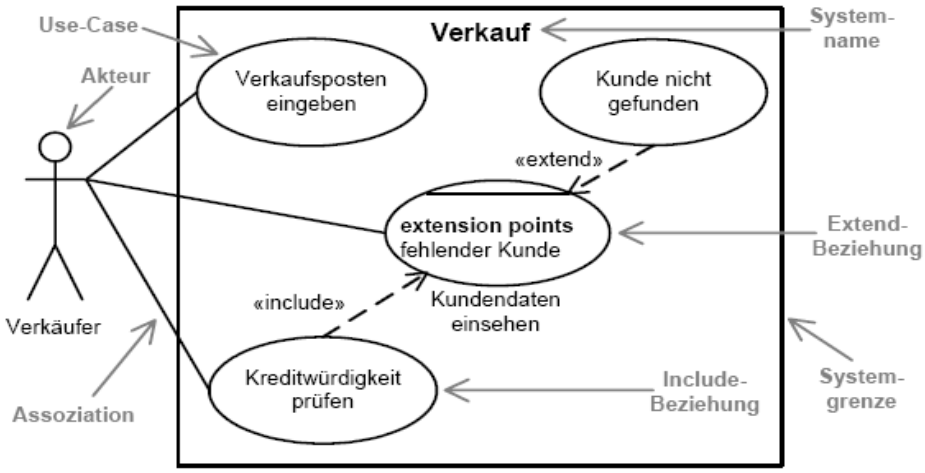
- Was soll mein geplantes System eigentlich leisten? Dokumentation einzelner Anwendungsfälle / Szenarios!
- Ein Use-Case stellt in einem geschlossenes System die Aktionen von und nach aussen dar, z.B. „Der Verkäufer gibt im Verkauf einen Verkaufsposten ein“.
- Das Use-Case soll das **Was** in den Mittelpunkt stellen und nicht das Wie! (Keine Implementationsdetails)
- Ideales Hilfsmittel, um mit den Stakeholdern (Anspruchsberechtigte) über ein System zu diskutieren. Ermöglicht eine Black-Box Sicht.

Use-Case (Anwendungsfall) Diagramm **Komponenten**

- Ein Use-Case besteht aus den Komponenten:
  - *(Sub-) System*
  - *Externer Kommunikations-partner: Akteure*
  - *Das Was, also das Use-Case*
  - *Beziehung zwischen*

    Akteur und Use-Case,

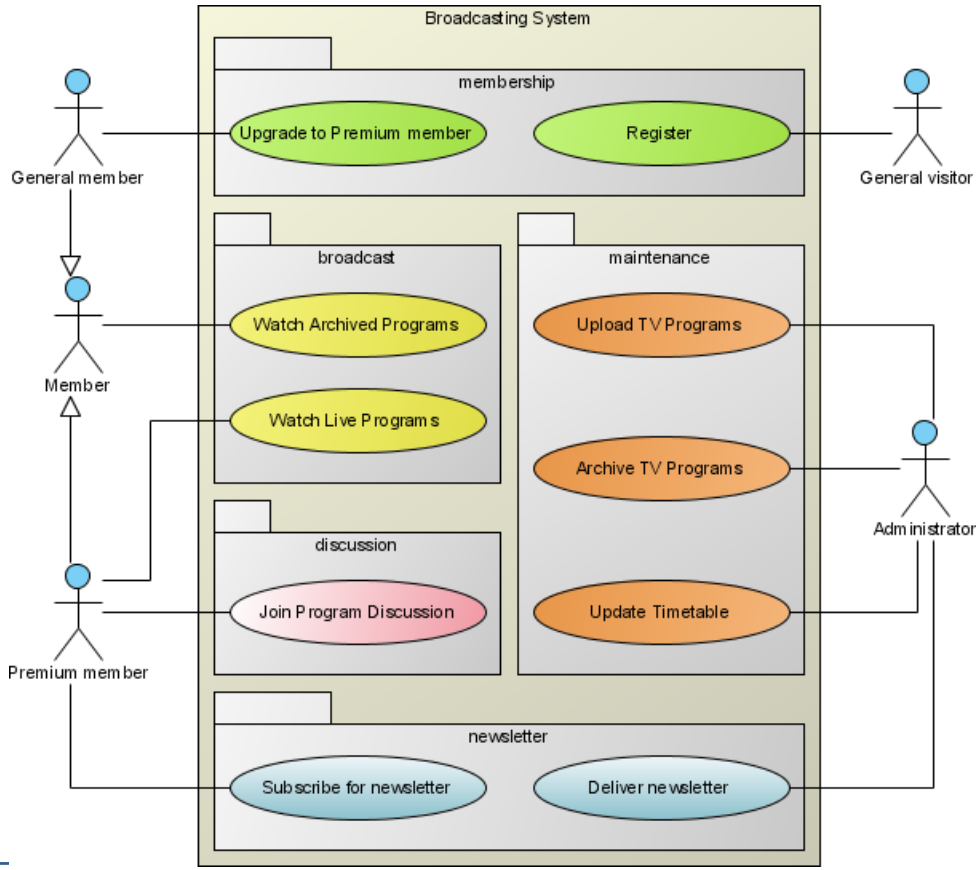
    bzw. untereinander.



Use-Case (Anwendungsfall) Diagramm
System - Subsysteme

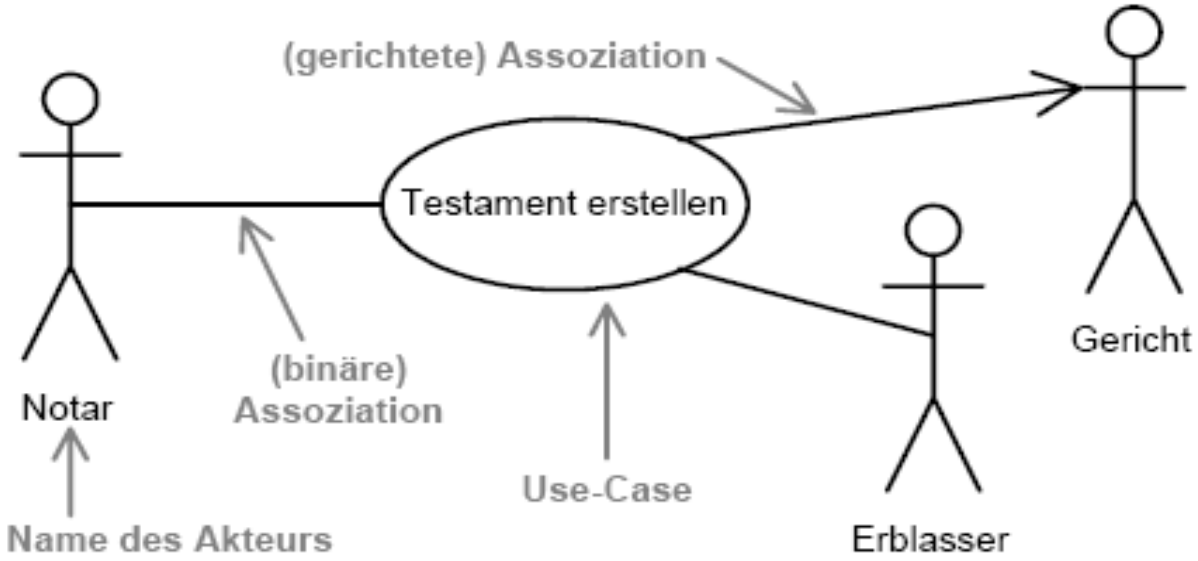
- Das System (z.B. Netflix) ist diejenige Einheit, die das Verhalten, welches durch die Use-Cases beschrieben wird, realisiert und anbietet.

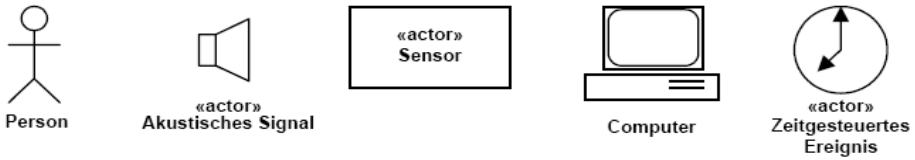
- Unter Umständen zergliedert sich das System, und einzelne Bestandteile realisieren Teilaufgaben; insgesamt jedoch muss das Verhalten „nach aussen“ ganzheitlich angeboten werden.

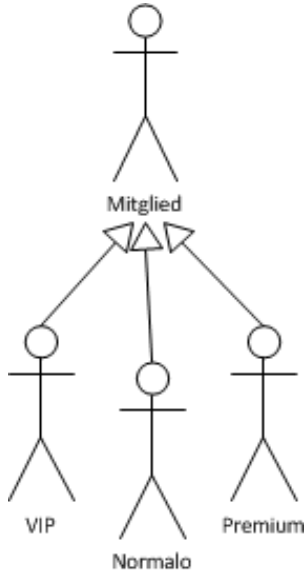


Use-Case (Anwendungsfall) Diagramm
Akteur

- Ein Akteur steht immer ausserhalb vom System und kann auch ein Sensor, ein anderes System usw. darstellen.







- Akteure können in einer Vererbungshierarchie stehen.

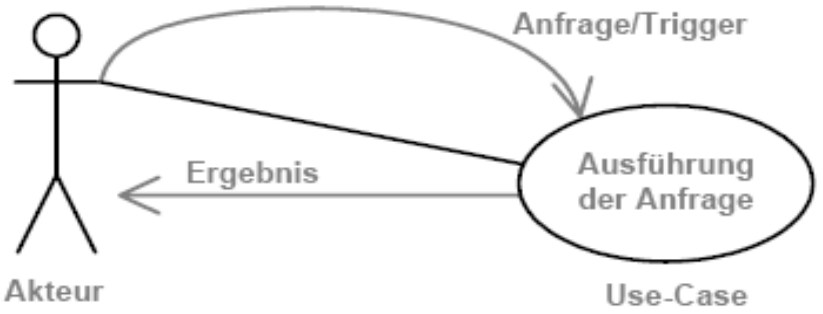
Use-Case (Anwendungsfall) Diagramm
Einsatz

Modellieren Sie Use-Cases, wenn Sie

- die funktionalen Dienstleistungen eines Systems oder einer Komponente „auf einen Blick“ zeigen wollen
- Ihr System aus der Nutzersicht in handhabbare, logische Teile zerlegen wollen
- die Aussenschnittstellen und Kommunikationspartner des Systems modellieren möchten
- komplexe Systeme leicht verständlich und auf hohem Abstraktionsniveau darstellen möchten
- planbare Einheiten, das heisst Inkremente, für Ihre Entwicklung benötigen

Use-Case (Anwendungsfall) Diagramm
Use-Case I

- Ein Use-Case bildet eine Art Hülle, die auch Sonder- und Fehlerfallaktionen einschliessen kann.
- Ein Use-Case wird stets von einem **Akteur ausgelöst** und führt zu einem fachlichen **Ergebnis** (Datei auf dem Medium abgelegt).
- Die Bezeichnung des Use-Case spiegelt **die Sicht des Akteurs** wieder (z.B. „Spiel starten“) und nicht die des Systems (müsste dann ja „Spiel startet“ heissen).
- Behalten Sie die Anzahl Use-Cases klein!



Use-Case (Anwendungsfall) Diagramm
Beispiel Party I

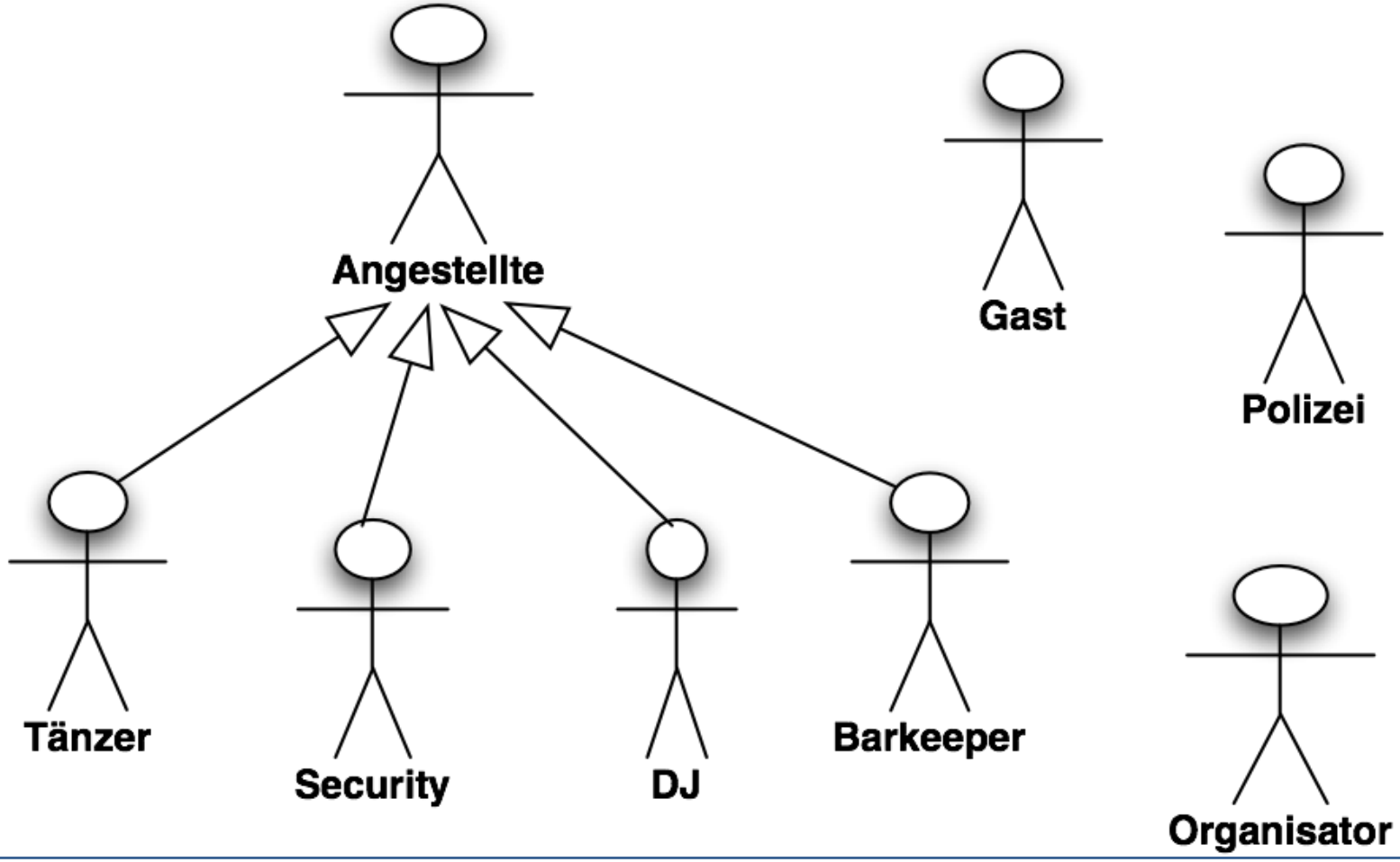
Sie geben eine Party.

- Welche Akteure gibt es?

Use-Case (Anwendungsfall) Diagramm
Beispiel Party II

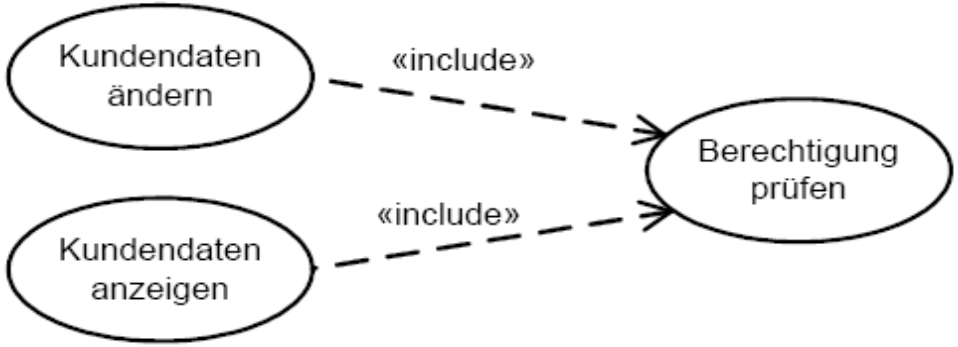
Sie geben eine Party.

- Welche Akteure gibt es?

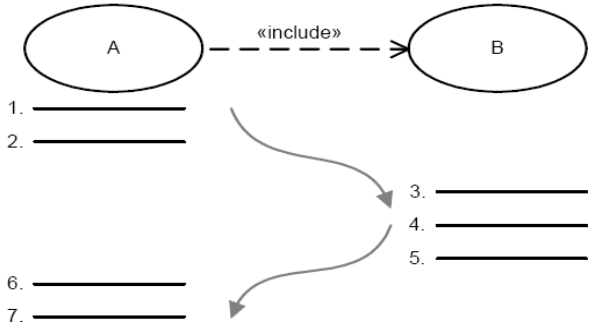


Use-Case (Anwendungsfall) Diagramm
Use-Case II: <<include>> (uses)

- Die «include»-Beziehung visualisiert, dass ein Use-Case (Kundendaten ändern) das Verhalten eines anderen Use-Case (Berechtigung prüfen) importiert (benutzt).

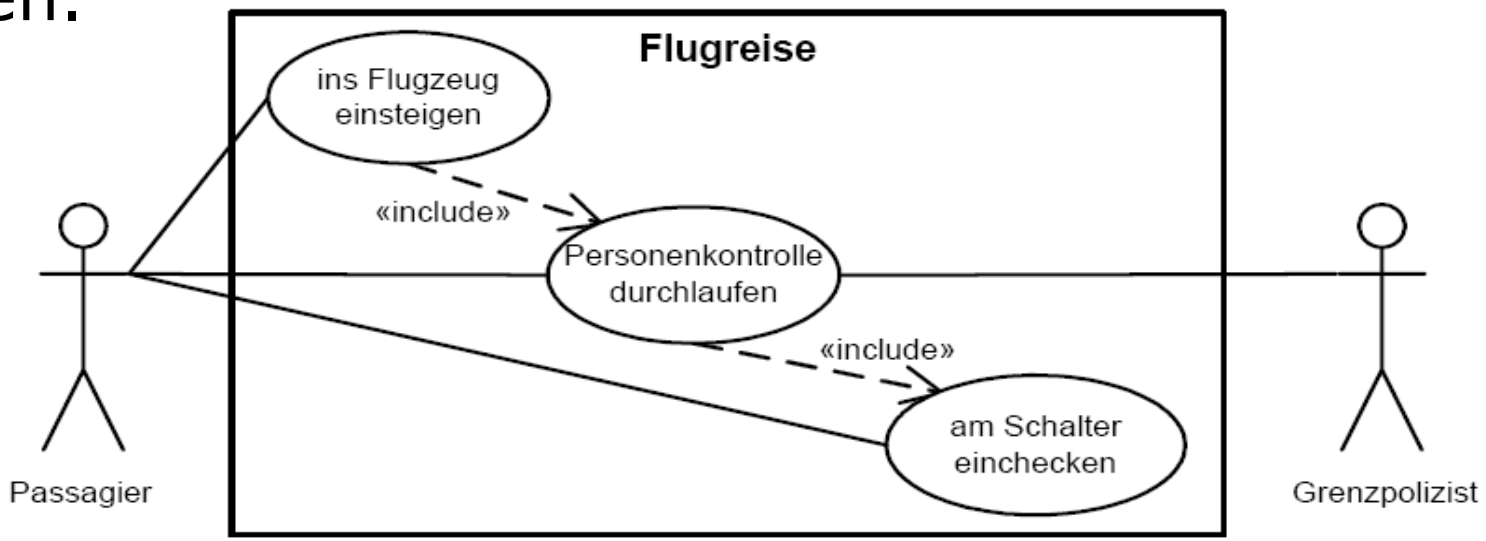


- Der Ablauf ist somit nicht mehr klar ersichtlich:



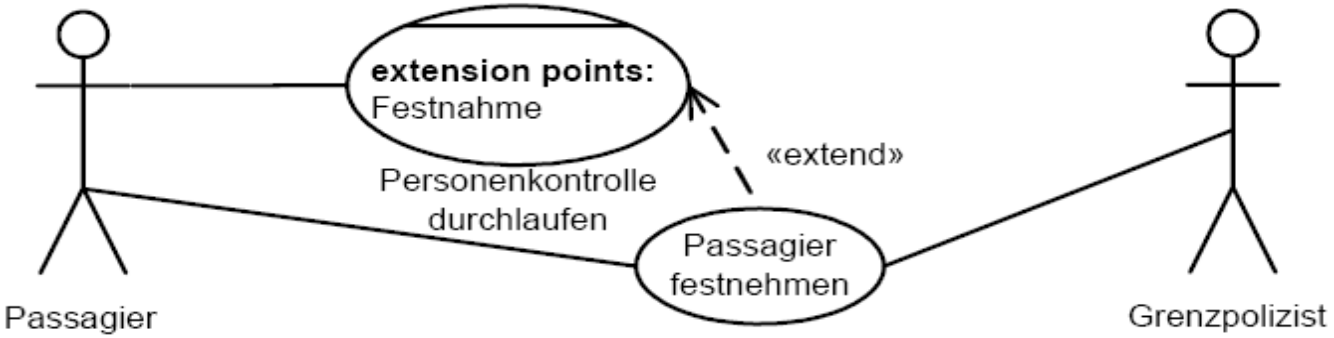
Use-Case (Anwendungsfall) Diagramm
Use-Case III: <<include>>

- In dem Use-Case einer *Flugreise* wird der Use-Case ins *Flugzeug einsteigen* hierarchisch zerlegt. Dadurch wird ausgedrückt, dass der Use-Case *am Schalter einchecken* komplett im Use-Case *Personenkontrolle durchlaufen* enthalten ist. Dieser wiederum wird vollständig im Use-Case ins *Flugzeug einsteigen* durchlaufen.



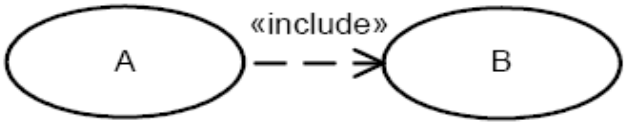
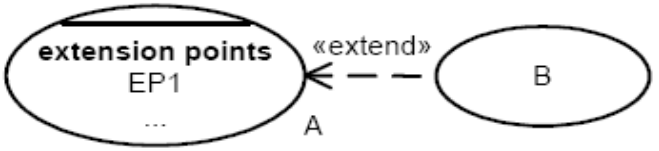
Use-Case (Anwendungsfall) Diagramm
Use-Case IV: <<extend>>

- Die «extend»-Beziehung zeigt an, dass das Verhalten eines Use-Case (Personenkontrolle durchlaufen) durch einen anderen Use-Case (Passagier festnehmen) erweitert werden kann (im Falle einer Festnahme), aber nicht muss.



- Den Zeitpunkt, an dem ein Verhalten eines Use-Case erweitert werden kann, bezeichnet man als *Erweiterungspunkt* (engl. extension point). Ein Use-Case darf mehrere Erweiterungspunkte besitzen. Sie werden innerhalb der Use-Case-Ellipse dargestellt und müssen benannt sein. Die Bezeichnung des Use-Cases verschiebt sich unter die Ellipse.
- Die Vorbedingung zur Ausführung des Erweiterungspunktes kann hinter den Doppelpunkt oder als Kommentarfeld zum «extend»- Pfeil geschrieben werden.

Use-Case (Anwendungsfall) Diagramm
Use-Case V

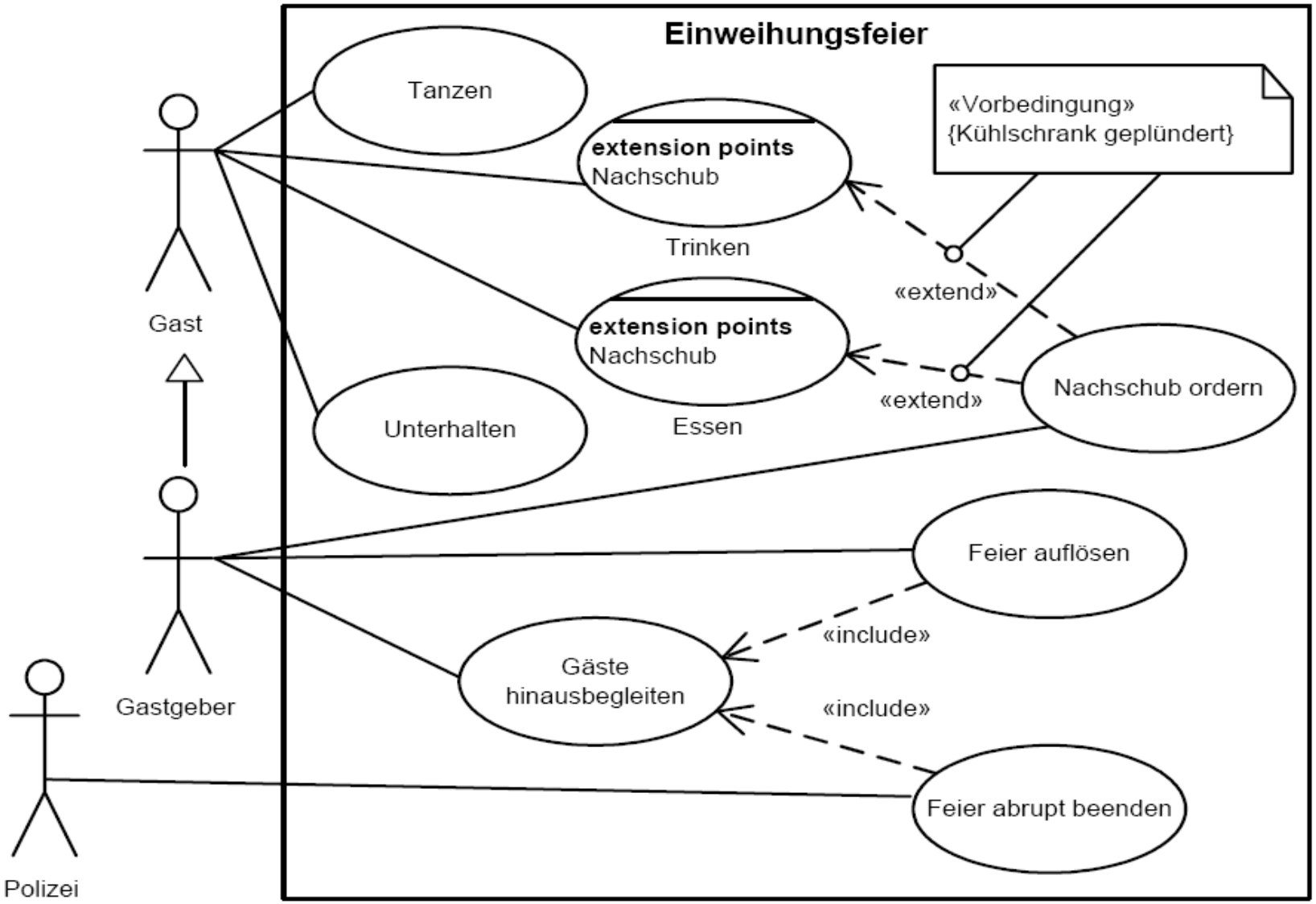
	«include»-Beziehung	«extend»-Beziehung
Notation		
Bedeutung	Ablauf von A schließt immer Ablauf von B ein.	Ablauf von A kann, muss aber nicht durch Ablauf von B erweitert werden.
Wann wird die Beziehung genutzt?	Ablauf von B kann in verschiedenen Use-Cases genutzt werden. Hierarchische, funktionale Zerlegung.	A besitzt neben Normalverhalten auslagerbare Sonderfälle.
Bedeutung für die Modellierung	A ist meist unvollständig und wird erst durch Inklusion B zu einem vollständigen Ablauf. B ist häufig künstlich zur Redundanzvermeidung gebildet.	A ist meist vollständig und kann durch B optional erweitert werden. B ist meist in sich vollständig.
Abhängigkeiten	A muss B bei der Modellierung berücksichtigen. B wird **unabhängig von A** modelliert, um die Nutzung durch weitere Use-Cases sicherzustellen (Wiederverwendbarkeit), B muss in sich **nicht vollständig** sein („B weiß nicht, durch wen er inkludiert wird“).	A muss durch Angabe von Erweiterungspunkten auf die Erweiterung durch B vorbereitet werden. B wird **in sich vollständig** und **unabhängig** von A modelliert („B weiß nicht, wen er erweitert“).

Use-Case (Anwendungsfall) Diagramm
Beispiel Party I

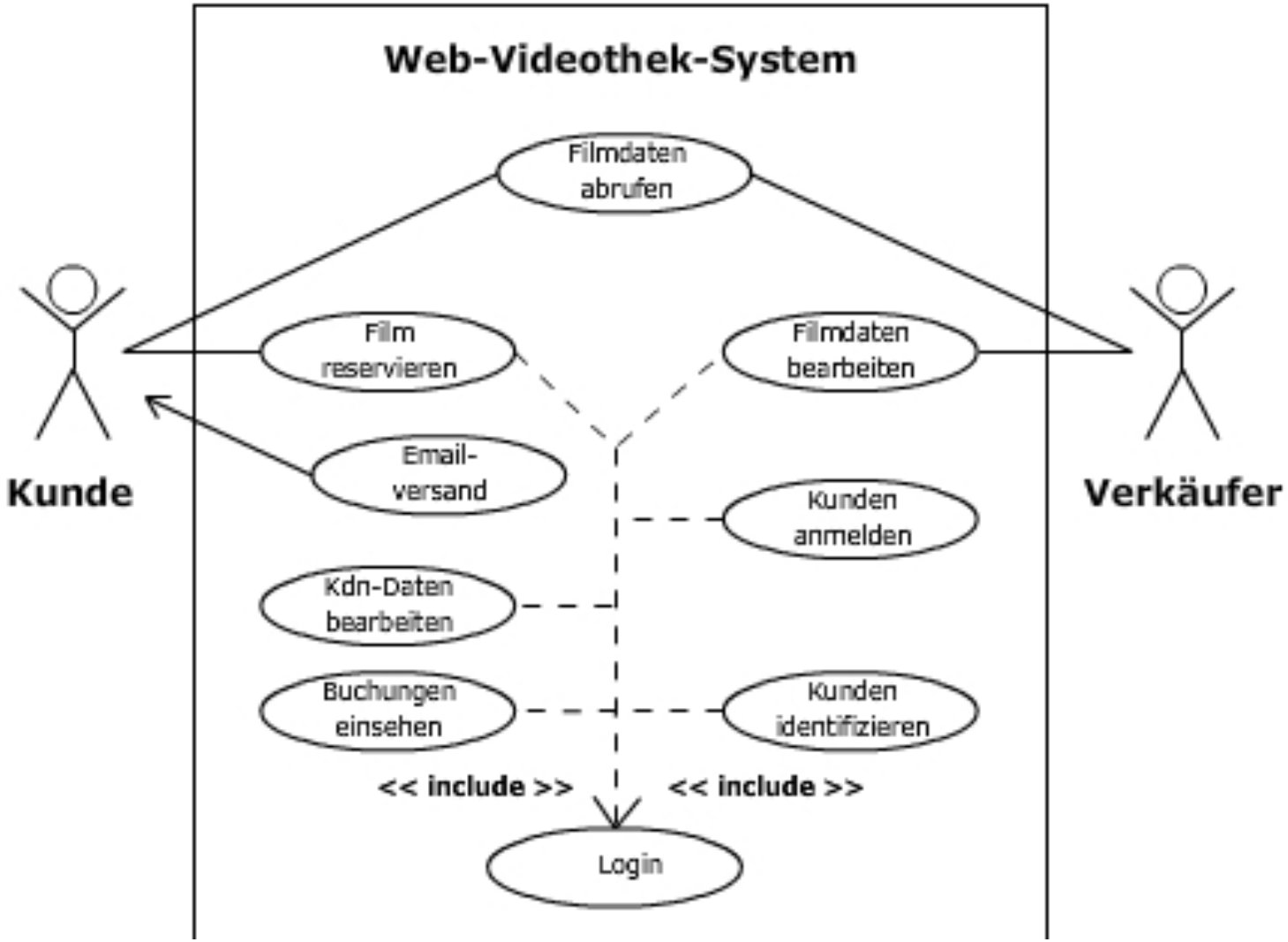
Sie geben eine Party.

- Welche Akteure gibt es?
- Welche Use-Cases gibt es zu den Akteuren Gast, Gastgeber und Polizei?

Use-Case (Anwendungsfall) Diagramm
Beispiel Party II



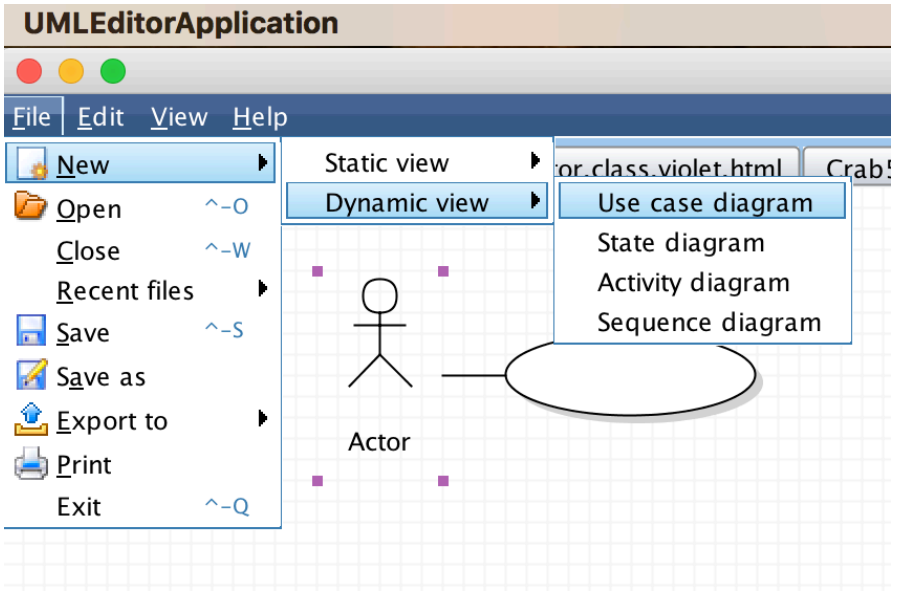
Use-Case (Anwendungsfall) Diagramm
Weiteres Beispiel



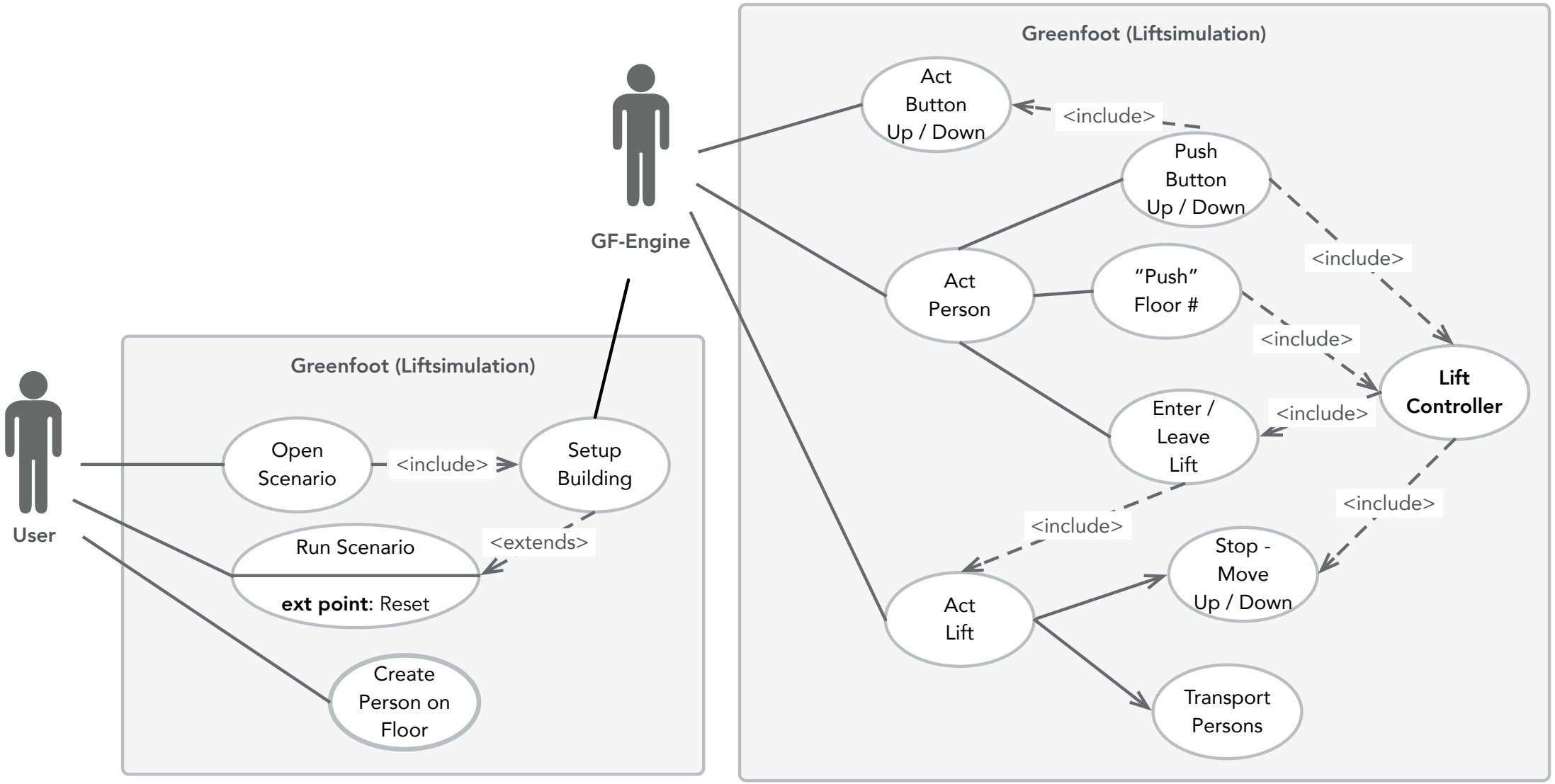
Use-Case (Anwendungsfall) Diagramm
Aufgabe Liftsimulation

Erstellen Sie ein UC-Diagramm für Liftsteuerung anhand der Abläufe:

- Welche Akteure gibt es (, die die Simulation aktivieren)?
- Welche (Haupt-)Anwendungsfälle werden ausgelöst?
- Ergänzen Sie mit «include»-Beziehungen und mind. einer «extend»-Beziehung!
- Ein UC-Diagramm sollte max. 10 UC's und max. 3 «UC-Ebenen» aufweisen… Ansonsten erstellen Sie ein neues UC-Diagramm!



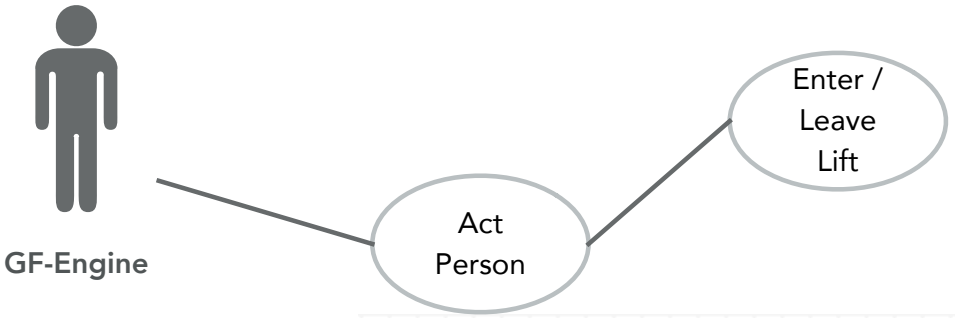
Use-Case (Anwendungsfall) Diagramm
Mögliche Lösung Liftsteuerung



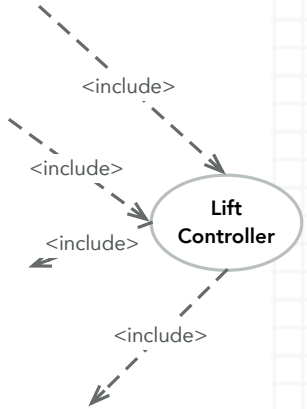
Use-Case (Anwendungsfall) Diagramm
Tipps

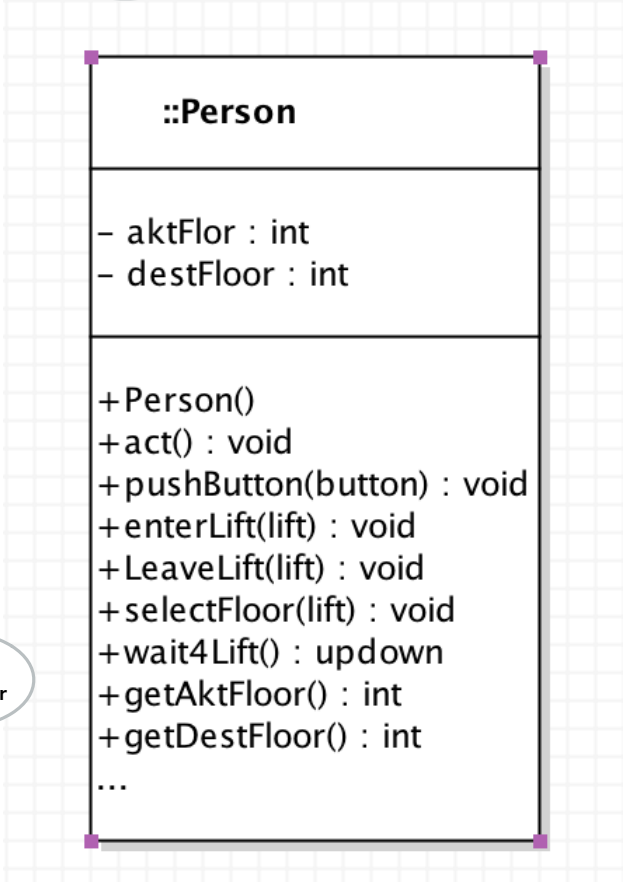
- Erstellen Sie Use-Cases aus der Sicht des Anwenders und nicht aus der Sicht des Entwicklers.
- Verwenden Sie Use-Cases nicht zur detaillierten Beschreibung von Operationen oder Funktionen.
- Modellieren Sie nicht zu viele Use-Cases (Empfehlung: maximal 10 Use-Cases pro Diagramm).
- Gehen Sie mit den erlaubten Beziehungen («include» und «extends») zwischen den Use-Cases sparsam um, denn sie reduzieren die intuitive Verständlichkeit für den Leser.
- Benennen Sie den Akteur immer in der Einzahl, vergeben Sie nie konkrete Namen wie Frau Meier (der Akteur ist eine Rolle).
- Spezifizieren Sie nicht-funktionale Anforderungen nicht mittels Use-Cases.
- Schreiben Sie den «extention Point» ins UC unter den Strich, die Beschreibung über den Strich. (UML 2.6)

Vom Use-Case (Anwendungsfall) Diagramm zum Klassendiagramm



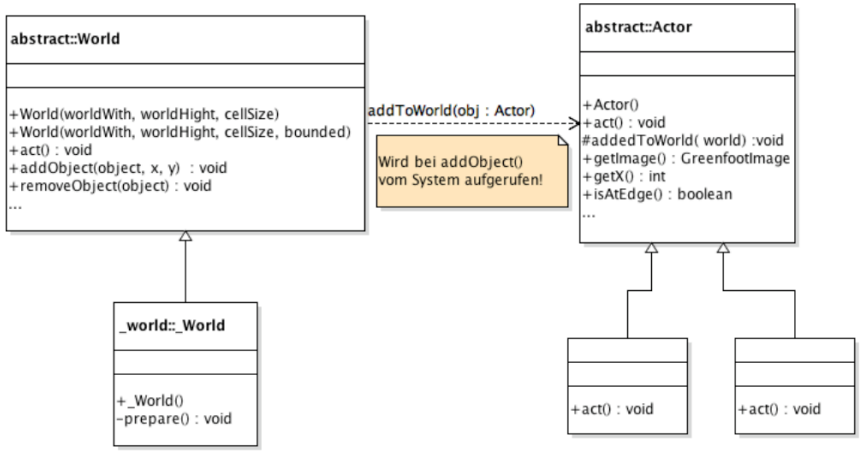
- Ein Use-Case in der ersten Ebene ***kann*** ein Indiz für eine mögliche Klasse sein.
- Ein UC in einer unteren Ebene ***kann*** ein Indiz für Methoden sein
- Ein vernetzter UC ***kann*** ein Indiz für eine Hilfsklassen sein





Aufgabe Klassendiagramm erstellen

- Laden Sie die UML Vorlage für Violet herunter.

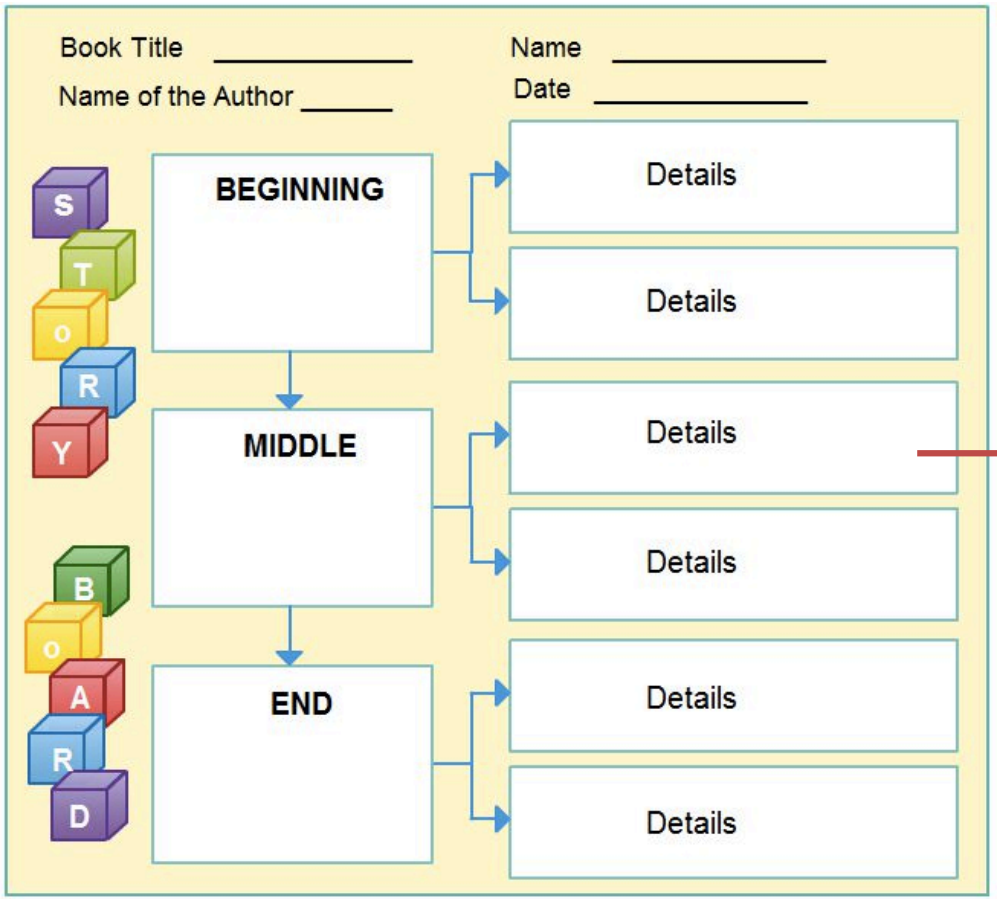


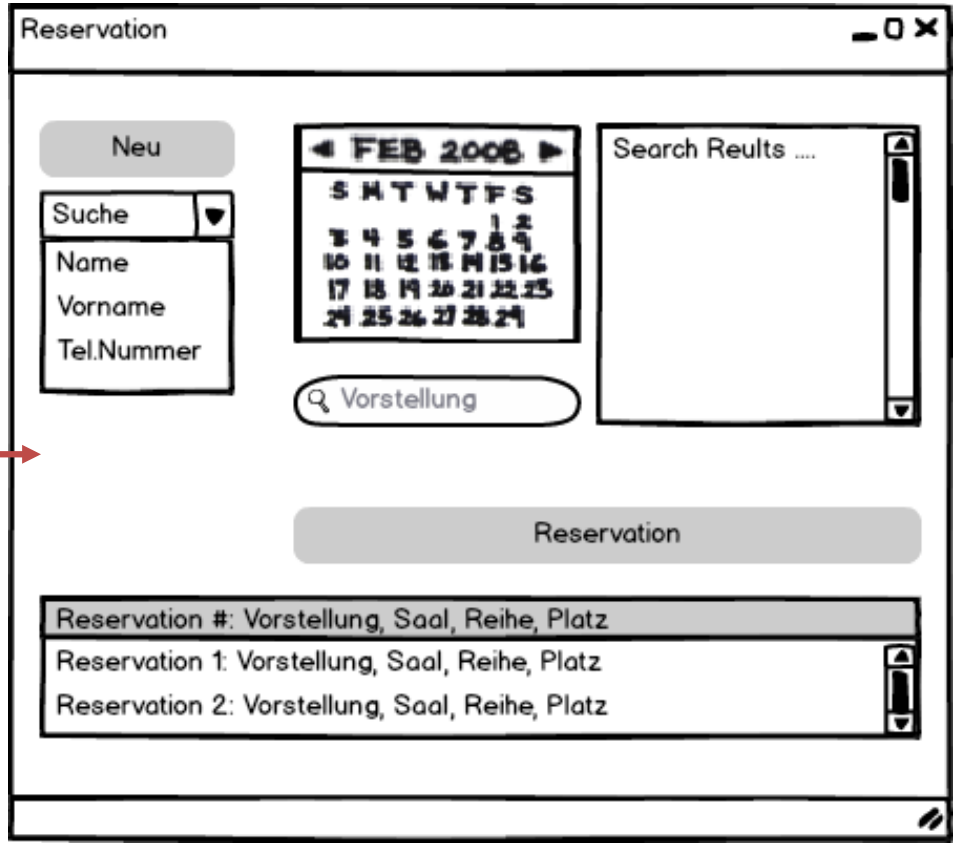
- Identifizieren Sie alle nötigen Klassen für die Simulation
- Bestimmen sie für 3 Klassen die nötigen Instanzvariablen und Methoden!
  (Siehe Beispiel Klasse Person auf vorheriger Folie)
- Legen Sie das UML auf dem BSCW ab (wird bewertet!)

** ANALYSE **

Storyboard & Mockups

- Zweck: Bildliche Darstellung eines typ. Ablaufs (eher dynamisch, exemplarisch)
- Sicht: exemplarische Benutzer-Anwendungsfälle, Voransicht GUI, ...
- Methode / Darstellung: Ablaufstruktur, ergänzt Use-Cases
  => www.creatly.com, www.draw.io, www.balsamiq.com





Zusammen-fassung OOA