**Wofür?**

* **Ziel:** Wer (Akteure) interagiert **woraufhin** mit **welchen Systemfunktionen** (Use Cases) – **Systemgrenze** klar ziehen.
* A white stick figure with black text

  AI-generated content may be incorrect.A grey rectangular object with a black border

  AI-generated content may be incorrect.**Nutzen:** Saubere **Anforderungsanalyse**, gemeinsames Vokabular, Basis für **Abläufe (Aktivitäten)**, **Interaktionen (Sequenzen)** und **Tests**.

**Kernbausteine & Notation**

* **Systemgrenze**: Rechteck mit Systemname; **alle Use Cases darin**.
* A black line with a white background

  AI-generated content may be incorrect.**Akteur**: Strichmännchen außerhalb; **Rolle** (Person/System).
* **Use Case**: Ellipse, Name = **Ziel aus Nutzersicht** (Verb + Objekt).
* **Assoziation**: durchgezogene Linie **Akteur ↔ Use Case** (Interaktion).
* A black stick figure with arrows pointing to another stick figure

  AI-generated content may be incorrect.**Generalisierung**: durchgezogene Linie mit **offener Dreiecksspitze** (spezialisierter Akteur/Use Case erbt Beziehungen/Verhalten).
* **«include»**: **zwingendes** Einbeziehen (gestrichelte Linie mit Pfeil **zum** eingebundenen UC).
* A diagram of a diagram

  AI-generated content may be incorrect.A diagram of a diagram

  AI-generated content may be incorrect.**«extend»**: **optionale** Erweiterung unter **Bedingung** (gestrichelte Linie mit Pfeil **zum Basis-UC**; Bedingung textlich notieren).

**Wann welches Beziehungsmuster?**

* A diagram of a diagram

  AI-generated content may be incorrect.**Assoziation**: Wer nutzt was? (immer!)
* **include**: Wiederkehrende **Pflicht-Teilfunktion** (z. B. „Authentifizieren“).
* **extend**: **Optionaler** Zusatzpfad (z. B. „Fehlermeldung anzeigen“ **wenn** Zahlung fehlschlägt).
* **Generalisierung**:
  + **Akteure**: „Premium-Kunde“ **ist ein** „Kunde“ → erbt alle Use Cases.
  + **Use Cases**: Spezial-UC erbt Basis-Verhalten und **verfeinert** es.

**Gute Namen (IHK-tauglich)**

* **Use Case**: *Verb + Objekt* („Bestellung absenden“, „Zahlung durchführen“).
* Keine UI-Schritte („Button klicken“) und keine Technik („POST /api/pay“).

**Vorgehen (6 Schritte)**

1. **Systemgrenze** ziehen & benennen.
2. **Akteure** (primär/sekundär) identifizieren.
3. **Hauptziele** der Akteure als **Use Cases** formulieren.
4. **Akteure ↔ Use Cases** verbinden (Assoziationen).
5. **include/extend** nur bei echter Pflicht-Teilfunktion bzw. optionalem Pfad ergänzen (Bedingung beim extend notieren).
6. **Generalisierungen** für Rollen/Use Cases nur, wenn’s wirklich vereinfacht.

**Mini-Beispiel (Onlineshop Zahlung)**

* **Akteure:** Kunde, Zahlsystem (extern).
* **Use Cases:** „Zahlung durchführen“, «include» „Authentifizieren“, «extend» „Fehlermeldung anzeigen“ **[wenn Zahlung fehlschlägt]**.
* **Systemgrenze:** „Shop-Backend“.

**Häufige Fehler (und wie du sie vermeidest)**

* **Zu technisch/zu detailliert:** Use Cases bleiben **fachlich** (keine UI-Klicks, keine Endpunkte).
* **Verwechslung include/extend:**
  + include = **immer dabei**, Wiederverwendung.
  + extend = **optional**, Bedingung nennen.
* **Verwaiste Use Cases:** Jeder UC braucht **mindestens einen Akteur**.
* **Unklare Systemgrenze:** Alles, was **Systemleistung** ist, gehört **in** das Rechteck.

**Qualitäts-Check (Kurzliste)**

* Passt jeder Use-Case-Name zu einem **Akteursziel**?
* Sind **Pflichtteile** als include, **Optionen** als extend modelliert (mit Bedingung)?
* Sind **Akteursrollen** statt Personen modelliert (z. B. „Sachbearbeiter“ statt „Max“)?
* Ist die **Systemgrenze eindeutig**?