**Wofür?**

* **Ziel:** **Zustände** eines Objekts/Systems und **erlaubte Übergänge** (mit Ereignis, Bedingung, Aktion) zeigen.
* **Nutzen:** Reihenfolgen- und Zustandsfehler vermeiden (z. B. Bezahlen vor Zielwahl), Verhalten eindeutig kommunizieren, Lebenszyklen prüfen.

A close-up of a sign

AI-generated content may be incorrect.A white rectangular sign with black text

AI-generated content may be incorrect.**Bausteine & Notation (Essentials)**

* **Startzustand:** ● (gefüllter Kreis)
* A white square with black text and a black arrow

  AI-generated content may be incorrect.**Endzustand:** ◎ (Doppelkugel)
* **Einfacher Zustand:** Abgerundetes Rechteck, Name z. B. Bereit
* **Zusammengesetzter/Superzustand:** Zustand **mit Unterzuständen** (hierarchisch)
* **Transition:** Pfeil Quellzustand → Zielzustand
  + A black line on a white background

    AI-generated content may be incorrect.**Ereignis**: löst Wechsel aus (extern/intern, z. B. Timer)
  + **Bedingung (Guard)**: in eckigen Klammern [ … ]
  + **Aktion**: nach Schrägstrich /aktion()
  + **Syntax:** Ereignis [Bedingung] / Aktion

**Einsatzbereiche**

* **Lebenszyklen** von Domänenobjekten (Bestellung, Ticket, Antrag)
* **UI-Zustände** (Maske geladen, validiert, gesperrt)
* **Geräte/Maschinen** (Automat, Drucker, IoT)

**Vorgehen in 6 Schritten (Checkliste)**

1. **Objekt/„Wer?“ wählen:** Für **ein** Objekt modellieren (z. B. Bestellung), nicht für den ganzen Prozess.
2. **Relevante Zustände sammeln:** Nur **beobachtbare, fachlich sinnvolle** Phasen.
3. **Start- & Endzustand** setzen (falls Lebenszyklus endet).
4. **Transitionen definieren:** Ereignis [Guard] / Aktion je Zustandswechsel.
5. **Superzustände** nutzen, wenn Unterphasen helfen (sonst weglassen).
6. **Review:** Zyklen, Sackgassen, nie erreichbare Zustände? Guards disjunkt?

**Qualitäts-Tipps (IHK-relevant)**

* **Ein Objekt pro Diagramm.** Prozesslogik gehört eher ins **Aktivitätsdiagramm**.
* **Guards knapp & eindeutig** ([Daten vollständig], nicht Roman).
* **Keine UI-Klickfolgen** als Zustände (Zustände = **fachliche** Lage).
* **Eintritt/Austritt-Aktionen** nur, wenn fachlich wichtig.
* **Minimieren:** Wenige, aussagekräftige Zustände > viele Mikrozustände.

**Durchgängiges Beispiel: Bestellprozess (Objekt: Bestellung)**

**Achtung:** Wir modellieren **den Lebenszyklus von Bestellung**, nicht den gesamten Checkout-Prozess.

A diagram of a company

AI-generated content may be incorrect.**Zustände:**

* Neu (Warenkorb gefüllt, noch nicht bestätigt)
* Checkout (Adresse/Zahlung erfasst)
* Aufgegeben
* InBearbeitung
* Versandt
* Abgeschlossen

**Start-/Ende:**

* **Startzustand** → Neu
* Abgeschlossen → **Endzustand**

**Transitionen (mit Ereignis/Bedingung/Aktion):**

* **Start → Neu**  
  createBestellung / init()
* **Neu → Checkout**  
  kasseAufrufen
* A diagram of a product

  AI-generated content may be incorrect.**Checkout → Aufgegeben**  
  bestaetigen [Daten vollständig] / sendeBestaetigung()
* **Aufgegeben → InBearbeitung**  
  auftragAnnehmen / sendeMail("Bestellung angenommen")
* **InBearbeitung → Versandt**  
  [verpackt ∧ Label erstellt] / sendeVersandmail()
* **Versandt → Abgeschlossen**  
  zugestellt

**Optionale Fehlpfade (alt-Varianten, kurz):**

* **Checkout → Neu**  
  abbrechen
* **Aufgegeben → Neu**  
  zahlungFehlgeschlagen / informiereKunde()

**Mögliche Ein-/Aus-Aktionen (sparsam):**

* entry/ setzeZeitstempel() in Aufgegeben
* exit/ archiviere() in Abgeschlossen

**Häufige Fehler & Gegenmittel**

* **Prozess statt Objekt modelliert:** → Auf **ein** Domänenobjekt fokussieren (z. B. Bestellung).
* **Zustände als Tätigkeiten** („Zahlung ausführen“) → **Tätigkeiten = Aktionen/Transitionen**, keine Zustände.
* **Fehlende Guards:** Bedingungen **immer** in [].
* **Unerreichbare Enden/Sackgassen:** Pfade prüfen, alternative Rücksprünge erlauben.