## **Zweck & Nutzen**

* **Wozu?** Zeitlich geordnete **Interaktionen** (Nachrichten) zwischen **Akteuren/Objekten/Komponenten** sichtbar machen.
* **Wert:** Verständlichkeit des **Systemverhaltens**, saubere **Schnittstellenabstimmung**, Frühwarnsystem gegen **Fehler in Reihenfolge, Blockierung, Parallelität**.

**Bausteine & Notation (mit Semantik)**

A line drawn on a white background

AI-generated content may be incorrect.A line drawn on a white background

AI-generated content may be incorrect.A dotted line with a circle

AI-generated content may be incorrect.A black dotted line on a white background

AI-generated content may be incorrect.**Lifelines (Lebenslinien)**

* **Darstellen:** Akteur, UI/Boundary, Control/Service, Entity/DB (als Objekte/Instanzen).
* **Notation:** Name oben (optional objekt:Klasse), darunter **gestrichelte vertikale Linie**.
* **Typische Rollen (hilfreich in der Praxis):**
  + **Actor** (extern, Mensch/System)
  + **Boundary** (UI/Adapter/Controller)
  + **Control/Service** (Geschäftslogik)
  + **Entity** (Domänenobjekt/DB-Gateway/Repository)

**Nachrichten (Calls/Signals)**

* A black line on a white background

  AI-generated content may be incorrect.**Synchron (blocking):** durchgezogener Pfeil **mit gefüllter Spitze** → Sender wartet (klassischer Request/Response).
* A black line on a white background

  AI-generated content may be incorrect.**Asynchron (non-blocking):** durchgezogener Pfeil **mit offener Spitze** → Sender wartet nicht (Events/Queues).
* A black line on a white background

  AI-generated content may be incorrect.**Rückgabe:** **gestrichelter Pfeil** (optional; wird oft weggelassen, wenn Ende des Aktivierungsbalkens die Rückkehr impliziert).
* **Selbstnachricht:** Pfeil, der auf dieselbe Lifeline zurückführt (Rekursion/innere Schritte).

**Praxisregel:** Geschäftslogik-Abfragen meist **synchron**; Events/Messaging/„fire & forget“ **asynchron**.

A black rectangular object with a white background

AI-generated content may be incorrect.**Aktivierungsbalken**

* **Bedeutung:** Zeitraum, in dem eine Instanz **aktiv arbeitet** (Operation läuft).
* **Lesetipp:** Verschachtelte Aktivierungen zeigen **verschachtelte Aufrufe** (Call-Stacks).

**Erzeugung/Vernichtung**

* **Erzeugung:** Nachricht auf eine **neue** Lifeline (oft mit Stereotyp «create»); Aktivierung startet direkt.
* **Vernichtung:** Lifeline endet mit **X** (Stereotyp «destroy»). Nur verwenden, wenn Lebensende im Szenario wichtig ist.

A computer screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.**2.5 Combined Fragments (Kontrollstrukturrahmen)**

* **alt** (Alternative): mehrere **wechselseitig ausschließende** Fälle, jede mit Bedingung in eckigen Klammern [Bedingung].
* **opt** (Optional): **ein** bedingter Zweig (wie alt mit nur einem Operand).
* **loop** (Schleife): Wiederholung **solange** Bedingung gilt ([i < n], [mehr Ergebnisse?]).
* **par** (Parallelität): Teilflüsse laufen **gleichzeitig**.
* **ref** (Referenz): Verweis auf **anderes Sequenzdiagramm** (gut für Wiederverwendung/Komplexitätsreduktion).
* **region/critical** (seltener): kritischer Abschnitt / nicht unterbrechen.

A diagram of a program

AI-generated content may be incorrect.A computer screen shot of a computer

AI-generated content may be incorrect.**Guards knapp & disjunkt** halten: [ok] vs. [nicht ok], keine Überschneidungen.

**Vorgehen – Template in 7+1 Schritten**

1. **Szenario wählen:** Ein **konkreter** Pfad (z. B. Happy Path *oder* ein bestimmter Fehlerfall).
2. **Lifelines festlegen:** Actor, Boundary, Control, Entity/DB (nur die, die wirklich beteiligt sind).
3. **Hauptfluss skizzieren:** 5–9 Kernaufrufe in logischer Reihenfolge (links→rechts).
4. **Nachrichtentypen wählen:** sync vs. async je nach fachlicher Blockierung/Entkopplung.
5. **Aktivierungen setzen:** Wo wird gearbeitet? (Dauer visualisieren hilft beim Lesen.)
6. **Kontrollstrukturrahmen ergänzen:** alt/opt/loop/par mit **klaren** Bedingungen.
7. **Lebenszyklus optional:** create/destroy nur, wenn relevant (z. B. kurzlebige Worker).  
   **+ Review:** Dupplikate entfernen, **ein Szenario pro Diagramm**, klare Namen (Verb + Objekt).

**Namensstil für Nachrichten:** verbObjekt(...) z. B. pruefeVerfuegbarkeit(id), ladeKunde(kundennr).  
**Keine Technikdetails** (keine Ports/URLs/SQL), Fokus = **Fachablauf/Schnittstellen**.

**Qualitäts-Check (Schnellliste)**

* **Klarer Pfad:** Ein Diagramm = ein Szenario.
* **Rollen sauber:** Actor/Boundary/Control/Entity korrekt.
* **Guards eindeutig:** in alt/opt jeweils **eine** klare Bedingung.
* **Rückgaben sparsam:** Wo informationshaltig, sonst weglassen.
* **Sync/Async bewusst:** Blockieren nur, wenn fachlich nötig; Events entkoppeln.
* **Lesbarkeit:** max. ~9 Hauptnachrichten pro Diagramm; weitere Details auslagern per **ref**.

**Häufige Fehler (und Gegenmittel)**

* **Alles in ein Diagramm stopfen:** → Split in **Happy Path** + **Fehlerfälle** + **ref** auf Subdialoge.
* **UI-Klickfolge statt fachlicher Logik:** → Nachrichten benennen als **fachliche Operationen**.
* **Fehlende Bedingungen in alt/opt:** → Immer [Bedingung] in die Kopfzeile des Fragments.
* **Falscher Nachrichtentyp:** → Prüfe, ob die aufrufende Seite **warten** muss (sonst async).
* **Unklare Verantwortungen:** → Boundary vermittelt, Control entscheidet, Entity liefert Daten.