Seite 13. UML-Sequenzdiagramm

UML-Sequenzdiagramm

Diese Seite erklärt das **UML-Sequenzdiagramm** so, dass du es schnell für IHK-Aufgaben und Programmierlogik anwenden kannst.

Ein UML-Sequenzdiagramm zeigt:

Welche Objekte oder Teilnehmer beteiligt sind
welche Nachrichten zwischen ihnen ausgetauscht werden
in welcher zeitlichen Reihenfolge die Kommunikation abläuft
welche Rückgaben erfolgen
welche Prozesse nacheinander oder parallel ablaufen

Wichtig:

Ein Sequenzdiagramm zeigt einen konkreten Ablauf.
Der Ablauf wird von oben nach unten gelesen.

Grafik 1: UML-Sequenzdiagramm – Grundlagen

Diese Grafik zeigt die wichtigsten Bestandteile:

Teilnehmer / Objekte
Lebenslinien
Aktivierungsbalken
Nachrichten
Rückgaben
Zeitachse

Wofür braucht man ein UML-Sequenzdiagramm?

Ein Sequenzdiagramm wird verwendet, um die Kommunikation zwischen Objekten oder Systemteilen darzustellen.

Typische Beispiele:

Login prüfen
Bestellung aufgeben
Produkt speichern
Datenbankabfrage ausführen
Benutzeroberfläche ruft Service auf
Service ruft Datenbank auf

Es zeigt also nicht nur, **was** passiert, sondern vor allem:

wer mit wem spricht
in welcher Reihenfolge
welche Antwort zurückkommt

Woran erkenne ich in einer Aufgabe, dass ein Sequenzdiagramm gemeint ist?

Typische Signalwörter:

Signalwort / Formulierung	Hinweis
Ablauf zwischen Objekten	Sequenzdiagramm wahrscheinlich
Nachricht	Kommunikation zwischen Teilnehmern
Aufruf	Methodenaufruf / Nachricht
Rückgabe	gestrichelte Antwortlinie
zeitliche Reihenfolge	von oben nach unten
Benutzer klickt ...	Start eines Ablaufs
GUI ruft Service auf	typische Sequenz
Service fragt Datenbank ab	typische Objektkommunikation
„stellen Sie den Nachrichtenaustausch dar“	Sequenzdiagramm

Grundidee

Ein Sequenzdiagramm zeigt einen konkreten Ablauf als Szenario.

Beispiel:

Benutzer gibt Login-Daten ein.
Die Oberfläche sendet die Daten an den LoginService.

Der LoginService fragt die Datenbank ab.

Die Datenbank gibt einen Benutzer zurück.

Der Service gibt den Loginstatus zurück.

Die Oberfläche zeigt eine Meldung an.

Daraus entsteht ein zeitlicher Ablauf zwischen:

Benutzer

loginGUI:LoginGUI

loginService:LoginService

userRepository:UserRepository

Teilnehmer / Objekte

Teilnehmer stehen im Sequenzdiagramm oben.

Beispiele:

Benutzer

loginGUI:LoginGUI

loginService:LoginService

userRepository:UserRepository

Wichtig:

Ein echter Benutzer kann als Akteur dargestellt werden.

Objekte werden häufig als instanz:Klasse geschrieben.

Beispiele:

loginGUI:LoginGUI

loginService:LoginService

userRepository:UserRepository

Lebenslinie

Unter jedem Teilnehmer verläuft eine Lebenslinie nach unten.

Darstellung:

```
Teilnehmer
   |
   |
   |
```

In UML wird die Lebenslinie meistens gestrichelt dargestellt.

Merksatz:

```
Lebenslinie = zeigt, dass ein Teilnehmer während des Ablaufs existiert.
```

Zeitliche Reihenfolge

Ein Sequenzdiagramm wird von oben nach unten gelesen.

```
oben = früher
unten = später
```

Merksatz:

```
Die Reihenfolge der Nachrichten ergibt sich aus ihrer Höhe im Diagramm.
```

Nachricht / Methodenaufruf

Eine Nachricht zeigt, dass ein Teilnehmer einen anderen Teilnehmer aufruft.

Beispiel:

```
loginGUI → loginService: pruefeLogin(name, passwort)
```

Typisch im Diagramm:

```
durchgezogene Linie mit Pfeilspitze
```

Beispiele:

```
loginDatenEingeben(name, passwort)
pruefeLogin(name, passwort)
findeBenutzer(name)
```

Rückgabe

Eine Rückgabe zeigt das Ergebnis eines Aufrufs.

Beispiel:

```
userRepository --> loginService: benutzer
```

Typisch:

```
gestrichelte Linie zurück
```

Beispiele:

```
benutzer
loginStatus
dashboardDaten
benutzerKontext
```

Wichtig:

```
Rückgaben sind oft optional, aber in Lern- und Prüfungsdiagrammen sehr hilfreich.
```

Aktivierungsbalken

Ein Aktivierungsbalken zeigt, dass ein Teilnehmer gerade aktiv ist.

Beispiel:

```
loginService prüft gerade die Login-Daten.
userRepository sucht gerade den Benutzer.
```

Darstellung:

schmaler senkrechter Balken auf der Lebenslinie

Merksatz:

Aktivierungsbalken = Teilnehmer verarbeitet gerade etwas.

Grafik 2: Beispiel – Login prüfen

UML-Sequenzdiagramm – Beispiel Login prüfen

Diese Grafik zeigt einen Login-Ablauf.

Beteiligte Teilnehmer:

Benutzer
loginGUI:LoginGUI
loginService:LoginService
userRepository:UserRepository
dashboard:Dashboard

Ablauf:

1. Benutzer gibt Login-Daten ein.
2. LoginGUI ruft den LoginService auf.
3. LoginService fragt UserRepository ab.
4. UserRepository gibt Benutzer zurück.
5. LoginService gibt Loginstatus zurück.
6. Danach laufen zwei Prozesse parallel:
   Prozess A: Dashboard laden
   Prozess B: Benutzerkontext laden
7. Danach wird die Startseite angezeigt.

Parallele Prozesse

In der IHK-nahen Darstellung können parallele Abläufe durch **zwei parallele Linien** markiert werden.

In unserem Beispiel:

```
Prozess A: Dashboard laden
Prozess B: Benutzerkontext laden
```

Wichtig:

```
Parallel bedeutet:
Die Prozesse müssen nicht streng nacheinander ablaufen.
Sie können gleichzeitig oder unabhängig voneinander gestartet werden.
```

Merksatz:

```
Parallele Linien = paralleler Ablaufbereich
```

Beispiel als Textablauf

Aufgabe:

```
Ein Benutzer gibt seine Login-Daten ein.
Die Oberfläche sendet Name und Passwort an den LoginService.
Der LoginService sucht den Benutzer über das UserRepository.
Das UserRepository gibt den Benutzer zurück.
Der LoginService gibt den Loginstatus zurück.
Nach erfolgreichem Login werden Dashboarddaten und Benutzerkontext geladen.
Danach wird die Startseite angezeigt.
```

Daraus entsteht:

```
Benutzer → loginGUI: loginDatenEingeben(name, passwort)
loginGUI → loginService: pruefeLogin(name, passwort)
loginService → userRepository: findeBenutzer(name)
userRepository --> loginService: benutzer
loginService --> loginGUI: loginStatus

parallel:
loginGUI → dashboard: ladeDashboard()
dashboard --> loginGUI: dashboardDaten
```

```
loginGUI → loginService: ladeBenutzerKontext()
loginService --> loginGUI: benutzerKontext

loginGUI → Benutzer: startseiteAnzeigen()
```

Sequenzdiagramm vs. Aktivitätsdiagramm

Sequenzdiagramm	Aktivitätsdiagramm
zeigt Kommunikation zwischen Teilnehmern	zeigt Ablauf von Aktionen
Teilnehmer oben	Start/Aktion/Entscheidung
Lebenslinien	Kontrollfluss-Pfeile
Nachrichten zwischen Objekten	Prozessschritte
gut für Methodenaufrufe	gut für Geschäftsabläufe
Zeit von oben nach unten	Ablauf meist entlang der Pfeile

Merksatz:

```
Sequenzdiagramm = Wer ruft wen wann auf?
Aktivitätsdiagramm = Was passiert als nächstes?
```

Sequenzdiagramm vs. Klassendiagramm

Sequenzdiagramm	Klassendiagramm
dynamischer Ablauf	statische Struktur
zeigt Nachrichten	zeigt Klassen
zeigt Reihenfolge	zeigt Attribute und Methoden
konkretes Szenario	allgemeines Modell

Merksatz:

```
Klassendiagramm = Bauplan
Sequenzdiagramm = Ablauf zwischen den Bauteilen
```

Typische IHK-Fehler

Fehler	Warum problematisch?

Lebenslinien vergessen	Teilnehmer existieren im Ablauf nicht sichtbar
Pfeile in falscher Reihenfolge	Ablauf wird falsch gelesen
Rückgaben als normale Aufrufe gezeichnet	Ergebnis wirkt wie neuer Methodenaufruf
Teilnehmer und Klassen verwechselt	Sequenzdiagramm zeigt konkrete Beteiligte
zu viele technische Details	Diagramm wird unübersichtlich
Zeitachse falsch verstanden	oben passiert vor unten
parallele Prozesse nicht markiert	Gleichzeitigkeit wird nicht sichtbar
Pfeile überlagern sich	Diagramm wird schlecht lesbar

Vorgehensweise in der Prüfung

Wenn du ein Sequenzdiagramm erstellen sollst, gehe so vor:

Schritt	Frage
1	Welcher konkrete Ablauf wird beschrieben?
2	Welche Teilnehmer oder Objekte sind beteiligt?
3	Wer startet den Ablauf?
4	Welche Nachricht kommt zuerst?
5	Wer ruft wen auf?
6	Welche Rückgaben gibt es?
7	Gibt es parallele Abläufe?
8	Ist die Reihenfolge von oben nach unten korrekt?
9	Sind die Pfeile und Beschriftungen lesbar?

Prüfungs-Merksätze

Teilnehmer stehen oben.
Lebenslinien laufen nach unten.
Zeit läuft von oben nach unten.
Nachrichten sind horizontale Pfeile.
Synchrone Aufrufe werden durchgezogen gezeichnet.
Rückgaben werden gestrichelt gezeichnet.
Aktivierungsbalken zeigen Verarbeitung.
Parallele Prozesse müssen erkennbar markiert werden.
Ein Sequenzdiagramm zeigt ein konkretes Szenario.

Mini-Beispiel 1

Aufgabe:

```
Ein Benutzer klickt auf Speichern.
Die Oberfläche sendet die Daten an den Controller.
Der Controller speichert die Daten in der Datenbank.
Die Datenbank gibt eine Bestätigung zurück.
```

Lösungsidee:

```
Benutzer → GUI: speichernKlicken()
GUI → Controller: speichern(daten)
Controller → Datenbank: insert(daten)
Datenbank --> Controller: ok
Controller --> GUI: gespeichert
GUI → Benutzer: bestätigungAnzeigen()
```

Mini-Beispiel 2

Aufgabe:

```
Ein Kunde gibt eine Bestellung auf.
Die Oberfläche sendet die Bestellung an den BestellService.
Der BestellService prüft den Lagerbestand.
Danach wird die Bestellung gespeichert.
```

Lösungsidee:

```
Kunde → ShopGUI: bestellungAufgeben()
ShopGUI → BestellService: bestellungPruefen()
BestellService → LagerService: bestandPruefen()
LagerService --> BestellService: bestandOK
BestellService → BestellungRepository: speichern()
BestellungRepository --> BestellService: gespeichert
BestellService --> ShopGUI: bestellungOK
```

Mini-Beispiel 3: Paralleler Ablauf

Aufgabe:

```
Nach dem Login werden Dashboarddaten und Benutzerdaten parallel geladen.
```

Lösungsidee:

```
parallel:
GUI → DashboardService: ladeDashboard()
DashboardService --> GUI: dashboardDaten

GUI → BenutzerService: ladeBenutzerDaten()
BenutzerService --> GUI: benutzerDaten
```

Merksatz:

```
Wenn zwei Abläufe unabhängig voneinander starten können,
kann ein paralleler Bereich sinnvoll sein.
```

Mini-Testfragen

1. Wofür verwendet man ein UML-Sequenzdiagramm?

Man verwendet es, um den zeitlichen Nachrichtenaustausch zwischen Teilnehmern oder Objekten darzustellen.

Merksatz:

```
Wer ruft wen wann auf?
```

2. Wie liest man ein Sequenzdiagramm?

Von oben nach unten.

```
oben = früher
unten = später
```

3. Was ist eine Lebenslinie?

Eine Lebenslinie ist die gestrichelte Linie unter einem Teilnehmer.

Sie zeigt, dass der Teilnehmer während des Ablaufs existiert.

4. Was zeigt ein Aktivierungsbalken?

Ein Aktivierungsbalken zeigt, dass ein Teilnehmer gerade aktiv ist oder etwas verarbeitet.

5. Wie werden Rückgaben typischerweise dargestellt?

Rückgaben werden meistens als gestrichelte Pfeile zurück dargestellt.

Beispiel:

```
userRepository --> loginService: benutzer
```

6. Was ist der Unterschied zwischen Nachricht und Rückgabe?

Eine Nachricht ruft eine Aktion oder Methode auf.

Eine Rückgabe liefert ein Ergebnis zurück.

```
Nachricht = Aufruf
Rückgabe = Ergebnis
```

7. Was bedeutet „Zeit läuft von oben nach unten“?

Nachrichten weiter oben passieren früher.

Nachrichten weiter unten passieren später.

8. Was zeigt ein paralleler Bereich?

Ein paralleler Bereich zeigt, dass mehrere Prozesse unabhängig oder gleichzeitig ablaufen können.

Beispiel:

```
Dashboard laden
Benutzerkontext laden
```

9. Warum sollte man Pfeile nicht überlagern?

Weil das Diagramm sonst schwer lesbar wird und die Reihenfolge oder Zuordnung der Nachrichten unklar werden kann.

10. Was ist ein typischer Fehler im Sequenzdiagramm?

Ein typischer Fehler ist, die zeitliche Reihenfolge falsch darzustellen.

Wichtig:

```
Der Ablauf wird von oben nach unten gelesen.
```

Nächste Seite

Danach kommt die eigene Trainer-Seite:

```
UML-Sequenzdiagramm-Trainer
```

Dort übst du:

```
Teilnehmer erkennen
Nachrichten richtig beschriften
Rückgaben unterscheiden
Reihenfolge beachten
parallele Prozesse erkennen
Sequenzdiagramme aus Textaufgaben ableiten
```

Revision #1
Created 28 May 2026 12:41:10 by Admin
Updated 28 May 2026 12:42:04 by Admin