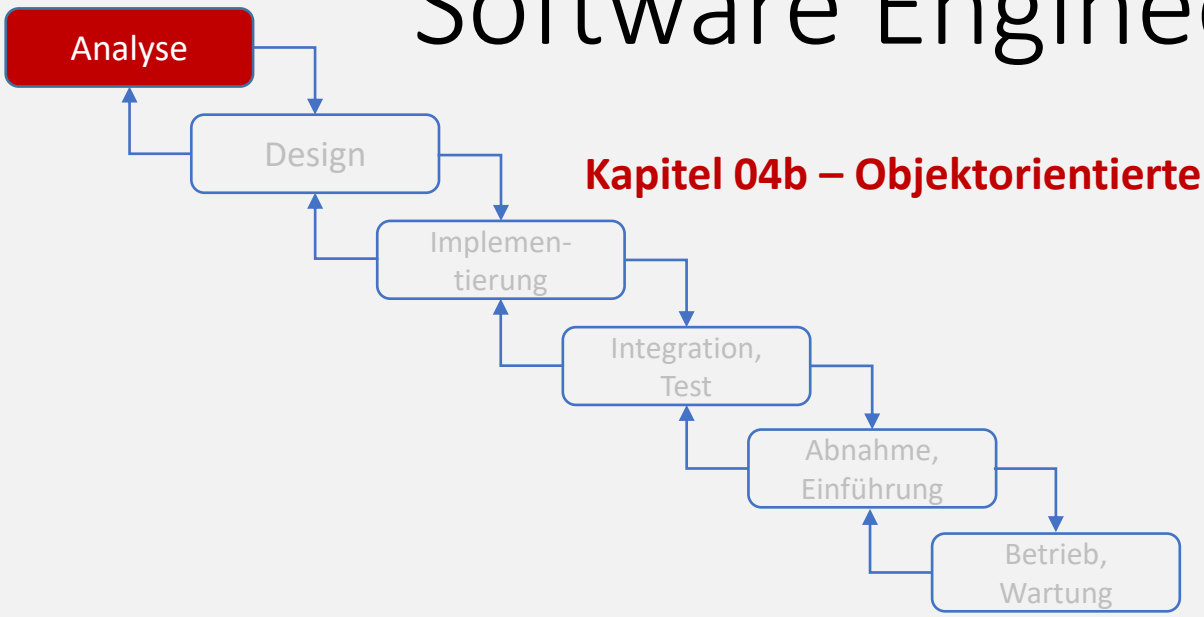
Software Engineering



Kapitel 04b – Objektorientierte Analyse (OOA)

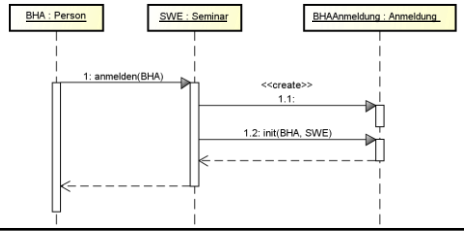
Sequenzdiagramme

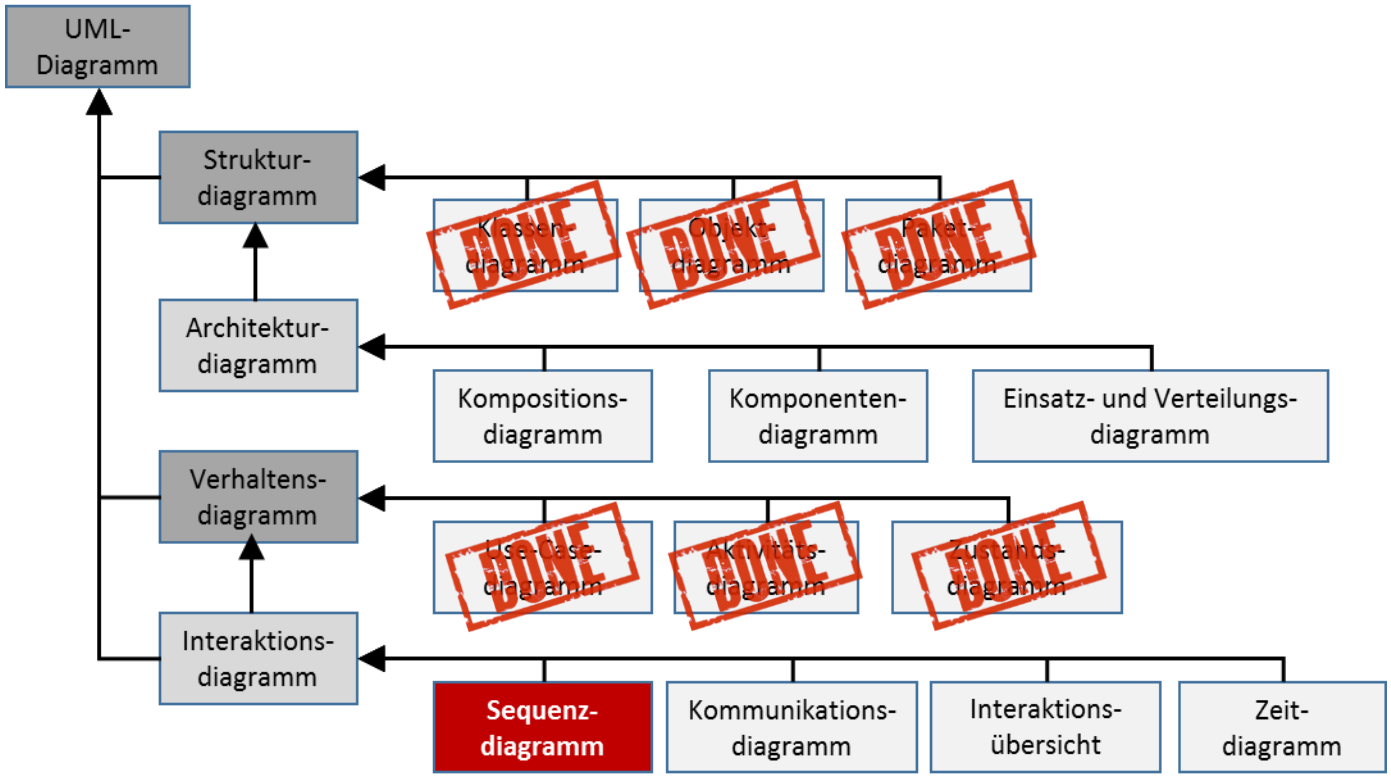
OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Entwicklung eines dynamischen Modells (4 Schritte)

1. Szenarios erstellen:
   - Jeden Geschäftsprozess durch Menge von Szenarios beschreiben
   - **Ergebnis: Sequenzdiagramm, Kollaborationsdiagramm (Alternativ/ergänzend auch Aktivitätsdiagramme)**
2. Zustandsautomaten erstellen:
   - Für jede Klasse prüfen: muss/kann nicht-trivialer Lebenszyklus erstellt werden
   - **Ergebnis: Zustandsdiagramm**
3. Operationen eintragen:
   - **Ergebnis: Klassendiagramm**
4. Operationen beschreiben:
   - Überlegen, ob eine Beschreibung notwendig ist. Falls „ja“ auch über Komplexitätsgrad Gedanken machen
   - **Ergebnis: Klassendiagramm, fachliche Beschreibung der Operationen, Zustandsautomaten, Aktivitätsdiagramme**

UML: Sequenzdiagramm





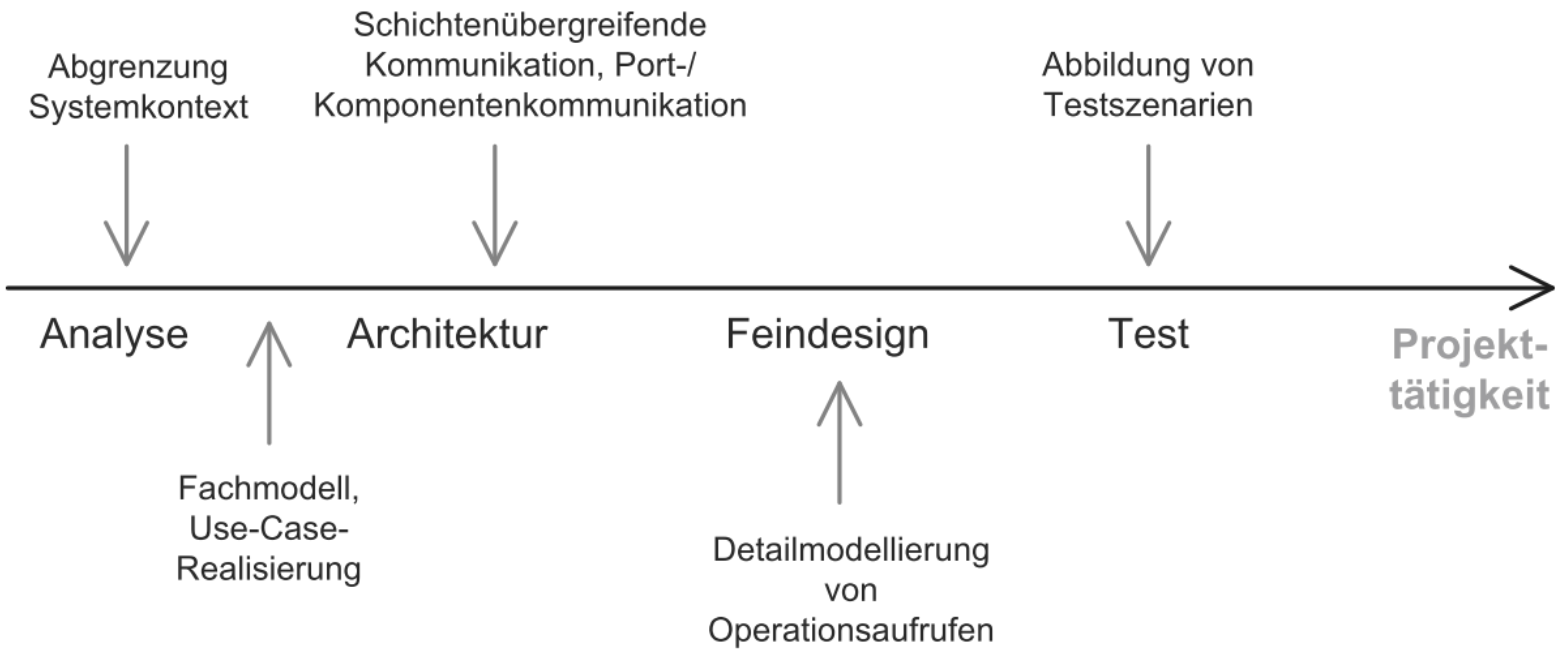
Sequenzdiagramme

Sequenzdiagramme liefern eine Antwort auf die Frage:

„Wie läuft die Kommunikation in meinem System ab?“

- Das Sequenzdiagramm ist das meist verwendete Interaktionsdiagramm
- Es ist außerdem das mächtigste Interaktionsdiagramm

Sequenzdiagramme: Anwendung im Projekt

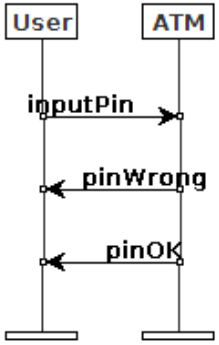


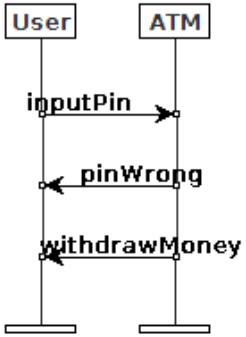
Fazit:

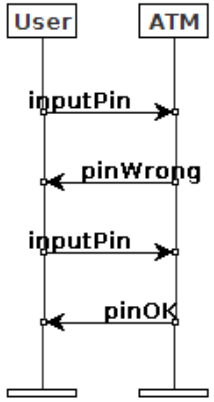
- Es lassen sich mit Sequenzdiagrammen Interaktionen auf verschiedensten Abstraktionsniveaus modellieren
- Sequenzdiagramme sind zu jedem Zeitpunkt im Projekt sinnvoll einsetzbar

Basis-Sequenzdiagramme/ Message Sequence Charts (MSCs)

- Einfache Sequenzdiagramme, die nur eine Kommunikationsstruktur eines Systems beschreiben
- Besitzen (in unserem Fall) sehr wenige Modellierungselemente (Welche?)
- Werden häufig für die Modellierung einfacher Szenarios/Systemläufe in der Analyse oder beim Testen (s. Kapitel “Testen”) verwendet





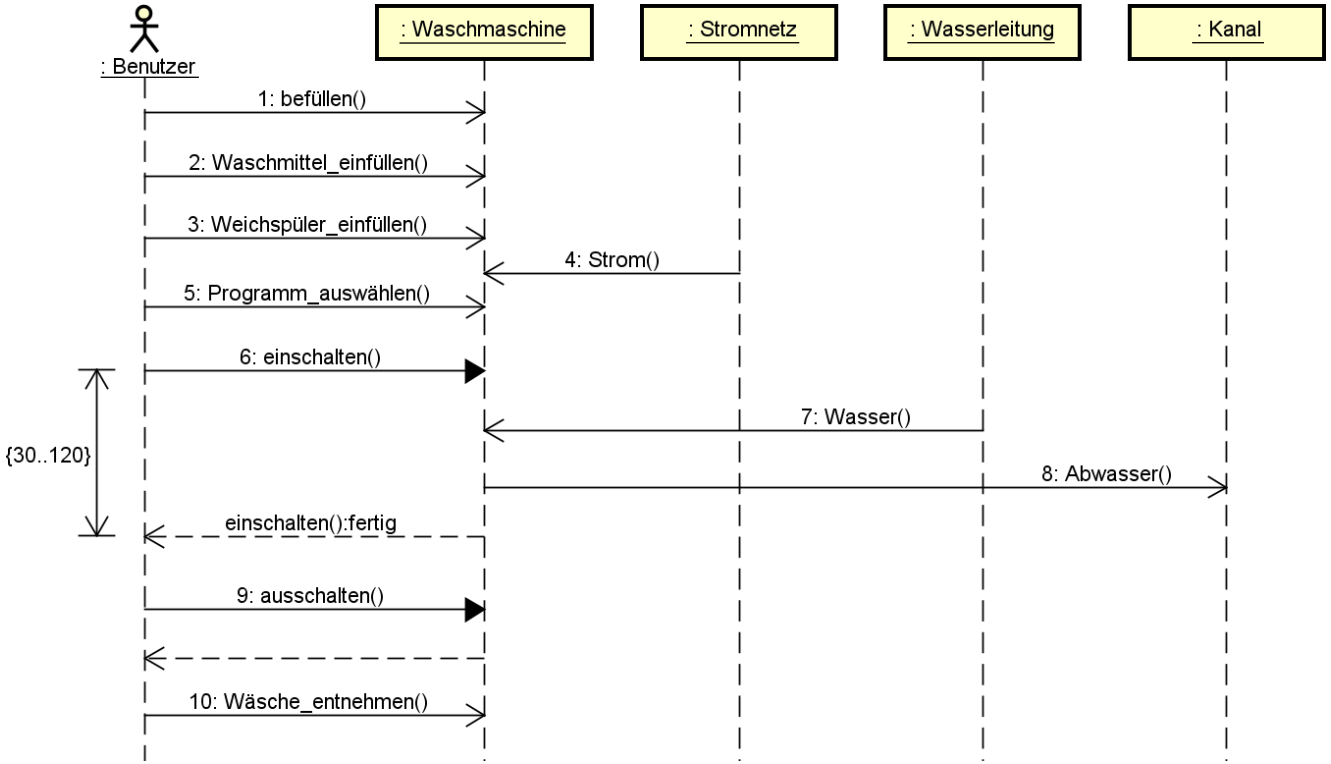


Sequenzdiagramme

- Sequenzdiagramme zeigen den Informationsaustausch zwischen beliebigen Kommunikationspartnern innerhalb oder zwischen Systemen
- Häufig sind Kommunikationspartner Objekte von Klassen
- Erlauben Modellierung von festen Reihenfolgen, zeitlichen und logischen Ablaufbedingungen, Schleifen und Nebenläufigkeit
- Kommunikationspartner werden durch Rechtecke mit Lebenslinien (gestrichelte Linie) gekennzeichnet
- Die Zeit schreitet von oben nach unten fort

Sequenzdiagramme: erstes Beispiel

(Eine spätere Konventionen ist verletzt)

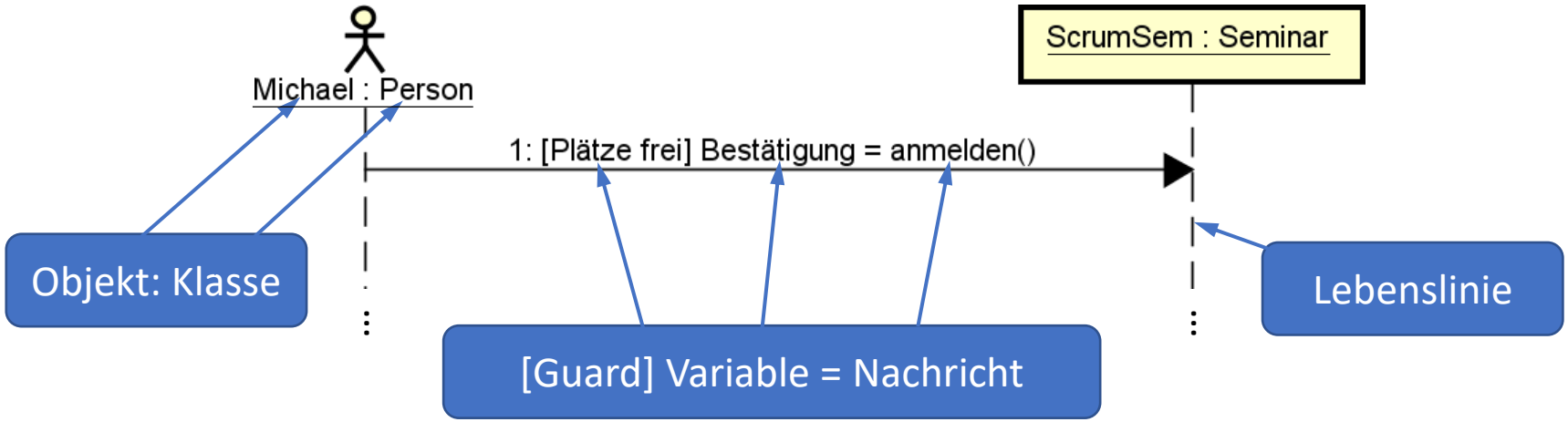


- Sequenzdiagramm mit fünf Prozessen
- Unterschiedlichen Arten von Nachrichtenaustausch (synchron, asynchron)
- Darstellen von Zeitdauern

[Quelle: UML 2 glasklar]

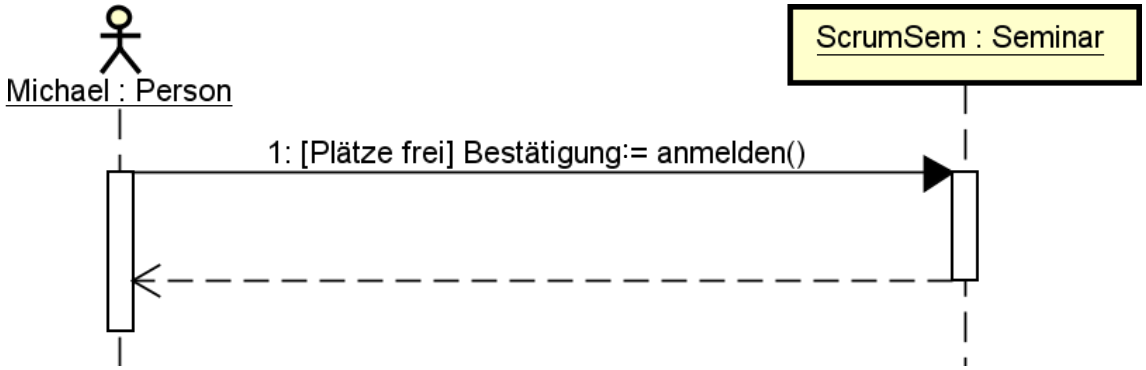
Sequenzdiagramme: Grundlegende Eigenschaften

- Objekte werden wie Klassen dargestellt aber mit einem unterstrichenen Namen: Typ
- Über Nachrichten, die zwischen den Objekten ausgetauscht werden, werden Methoden der Objekte aufgerufen
- Jedes Objekt hat eine gestrichelte senkrechte Lebenslinie
- Dynamische Objekte werden zum Zeitpunkt ihrer Erzeugung eingetragen
- Ihre Zerstörung wird durch ein Kreuz als Endpunkt der Lebenslinie markiert
- Statische Objekte stehen ganz oben im Diagramm



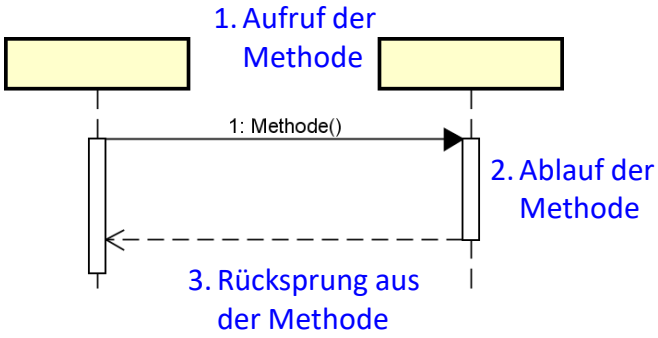
Sequenzdiagramme: Methodenaufrufe (1)

- **Kommunikation zwischen Objekten** erfolgt durch Nachrichten (Methodenaufrufe), die als gerichtete Pfeile mit Spitze und Beschriftung dargestellt werden:
  - Beschriftung: `['[‘Bedingung’]’][Variable ‘:=‘] NameBotschaft’(‘[Argumente]’)’`
  - Bedingung muss erfüllt sein, damit Nachricht gesendet wird
  - Variable nimmt das Ergebnis auf
  - Methode wird durch Namen und Argumente identifiziert

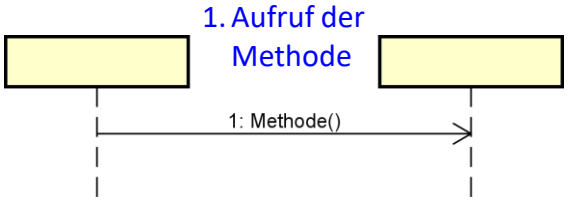


Sequenzdiagramme: Methodenaufrufe (2)

- **Synchroner Aufruf:**
  - Empfangendes Objekt verarbeitet die Nachricht und
  - Liefert anschließend eine Ergebnisantwort zurück



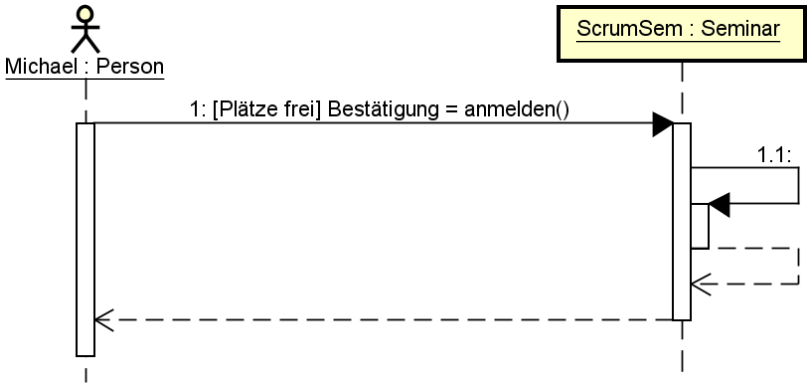
- **Asynchroner Aufruf:**
  - Quelle sendet Nachricht an Ziel
  - Quelle wartet nicht auf Antwort
  - Sondern fährt mit Verarbeitung fort



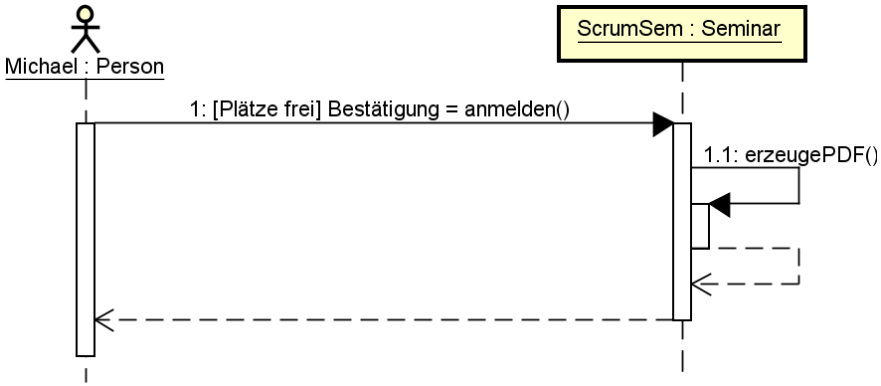
	Sende- und Empfangsereignis treten auf	Sendeereignis ist nicht bekannt	Empfangsereignis ist nicht bekannt
synchroner Operationsaufruf	→	●→	→●
asynchroner Signal-/Operationsaufruf	→	●→	→●
Antwortnachricht/ Rücksprung auf einen synchronen Operationsaufruf	- - ->	●- - ->	- - ->●

Sequenzdiagramme: Methodenaufrufe (3)

- **Rekursiver Aufruf:**
  - Verarbeitendes Objekt führt rekursiven Aufruf an sich selbst durch

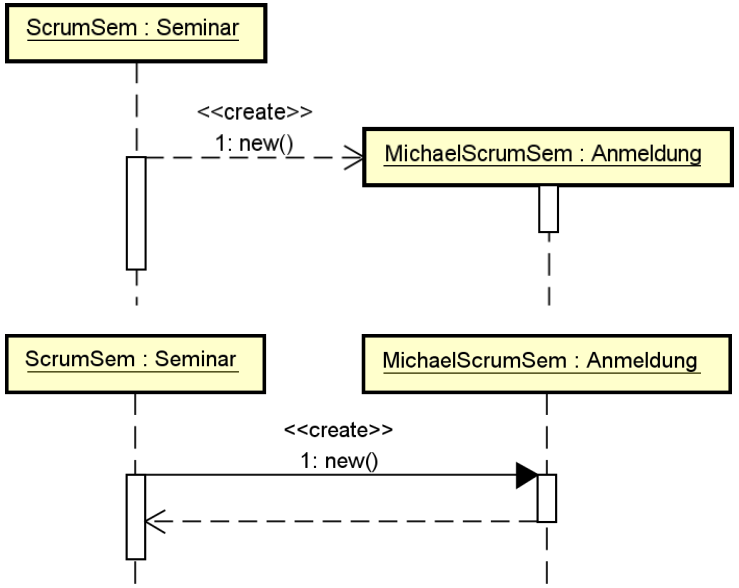


- **Lokaler Aufruf:**
  - Aufruf einer lokalen Methode

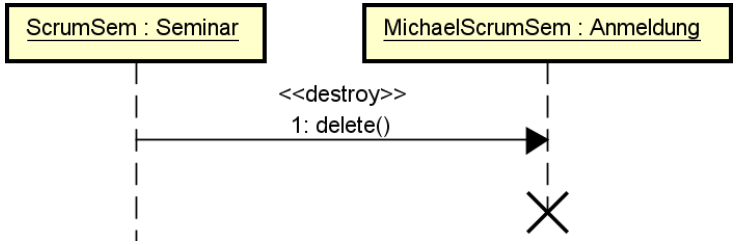


Sequenzdiagramme: Methodenaufrufe (4)

- **Objekterzeugung:**
  - Stellt die dynamische Erstellung (Instanziierung) eines Objektes dar
  - Zwei Darstellungsmöglichkeiten:

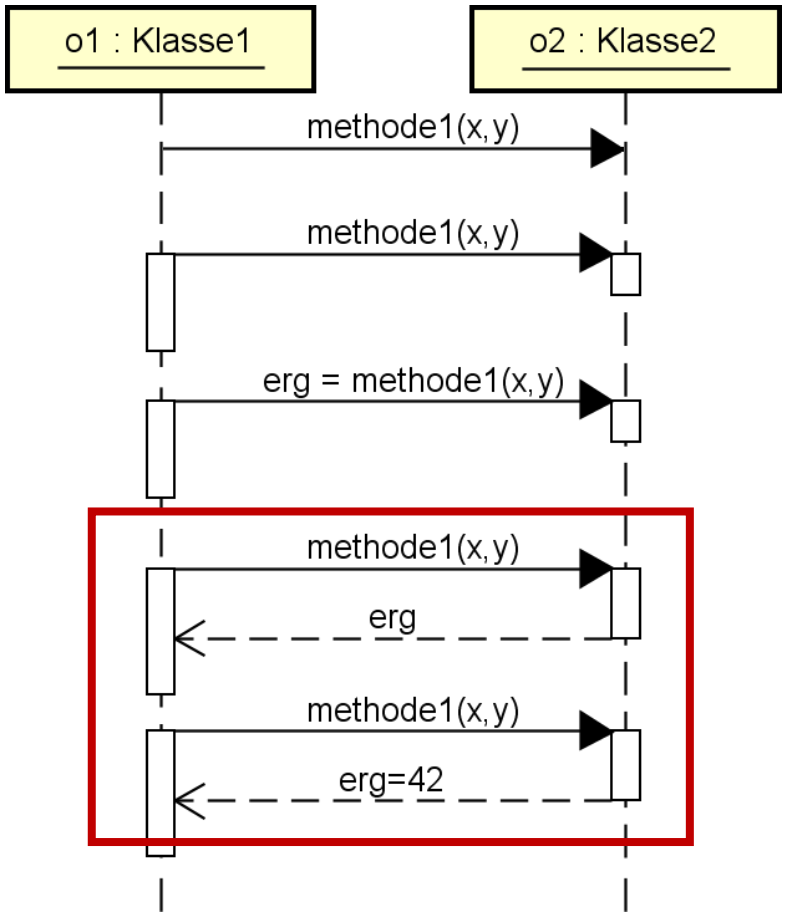


- **Objektzerstörung:**
  - Durch das "X" auf der Lebenslinie wird die Zerstörung der Instanz gekennzeichnet
  - Eine gesonderte Nachricht (wie im Beispiel) ist dabei nicht notwendig!



Sequenzdiagramme: Methodenaufrufe (5) unterschiedliche Notationen

- Rechts sind fünf verschiedene Darstellungsmöglichkeiten eines Methodenaufrufs aufgezeigt
- Wir nutzen eine der beiden untersten Varianten mit Return-Pfeilen
- Sequenzdiagrammen können helfen, die im Aktivitätsdiagramm beschriebenen Abläufe zu validieren
- Rückgabewerte werden häufig weggelassen, wenn nur Ablauf wichtig ist (z.B. bei MSCs)



Sequenzdiagramme: Beispiel “Seminarverwaltung”

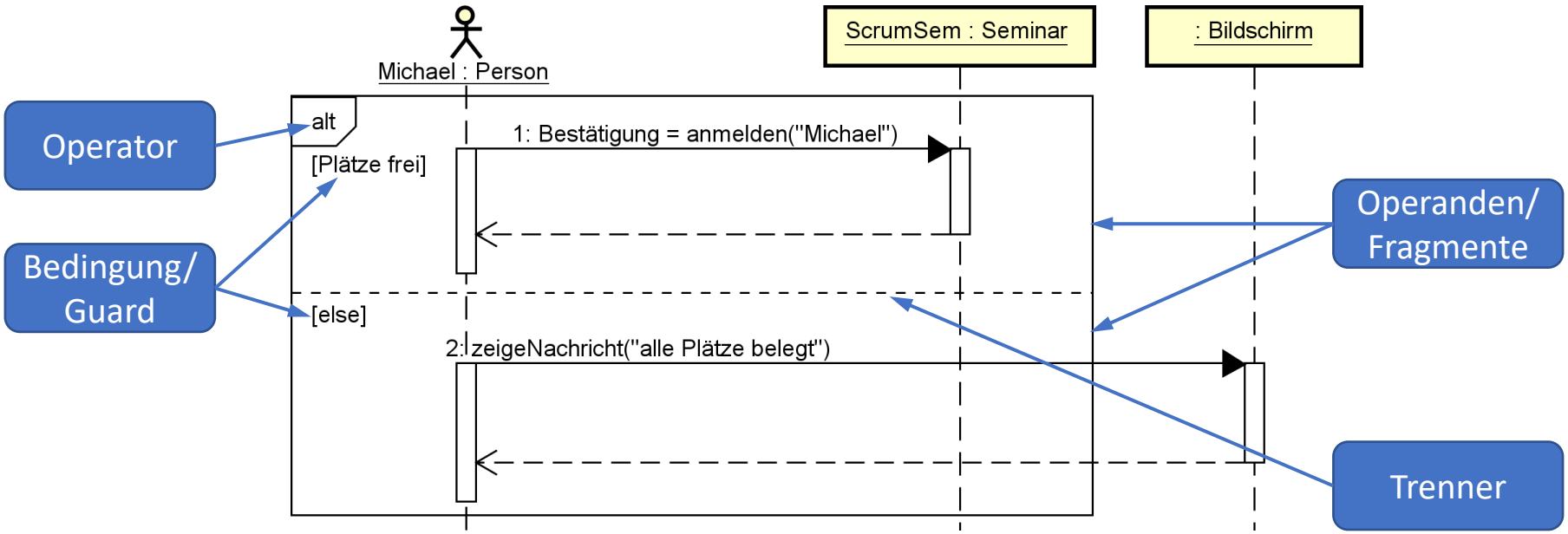
Erstellen Sie ein Sequenzdiagramm für eine Seminarverwaltung:

- Ein Kunde meldet sich an (Security-Komponente).
- Er sucht das Seminar “Informatik” in der Seminarverwaltung.
- Er bucht mit dem von der Securitykomponente bei der Anmeldung zurückgegebenen Kundenobjekt das Informatik-Seminar.
- Dazu erzeugt die Seminarkomponente eine Kundenbuchung und sendet anschließend mithilfe der Mailkomponente eine Bestätigung, in der die Kundendaten enthalten sind.
- Ist die Mail versandt, teilt die Mailkomponente das der Seminarkomponente mit.
- Anschließend gibt die Seminarkomponente als Antwort auf die Buchung noch eine kurze Buchungsbestätigung an den Kunden.

Sequenzdiagramme: Operatoren

- Für Sequenzdiagramme existieren vordefinierte Operatoren
- Diese erlauben Modellieren von Schleifen, Alternativen, Nebenläufigkeit, …

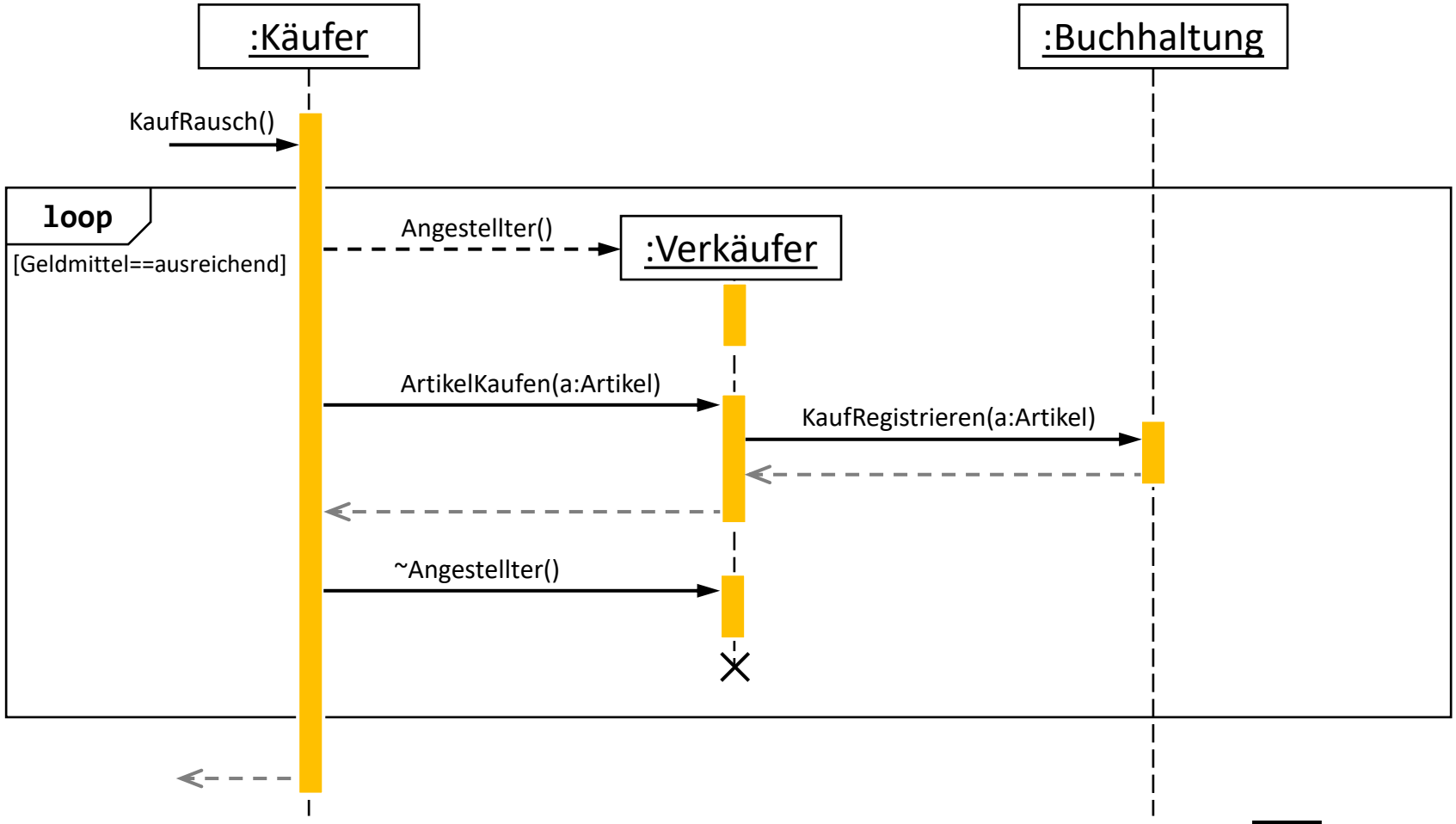
Beispiel für eine Alternative:



Sequenzdiagramme: Operatoren (unvollst.)

Operator	Bedeutung
`alt`	Alternative: Mindestens zwei alternative Fragmente. Nur das Fragment, dessen Bedingung erfüllt ist, wird ausgeführt
`opt`	Optional: Das Fragment, wird nur ausgeführt, wenn die Bedingung erfüllt ist
`par`	Parallel: Alle Fragmente werden parallel abgearbeitet
`loop`	Schleife: Das Fragment kann mehrere Male hintereinander ausgeführt werden. ein Guard spezifiziert die Durchlaufbedingung, (minInt,maxInt) die Mindest- bzw. Höchstzahl an Wiederholungen
`critical`	Kritische Region: Nur ein Thread ist in dieser Region erlaubt
`neg`	Negativ: Das Fragment zeigt eine nicht erlaubte Interaktion
`ref`	Verweis: Deutet auf eine Interaktion hin, die in einem anderen Diagramm definiert wurde. Der Rahmen wird gezeichnet, um die Lebenslinie entsprechend abzudecken. Parameter und Rückgabewert können angegeben werden
…	…

Sequenzdiagramme: Operatoren Beispiele



Sequenzdiagramme: Operatoren Beispiele

Mensch-ärgere-Dich-nicht:

- Wenn keine Figur im Spielfeld ist, kann der Spieler bis zu 3 mal würfeln, solange die Augenzahl nicht 6 ist

Welche Operatoren benötigen Sie?

alt	critical
opt	neg
par	ref
loop	…

Man benötigt die Operatoren:

- Option
- Schleife

Sequenzdiagramme: Operatoren Beispiele

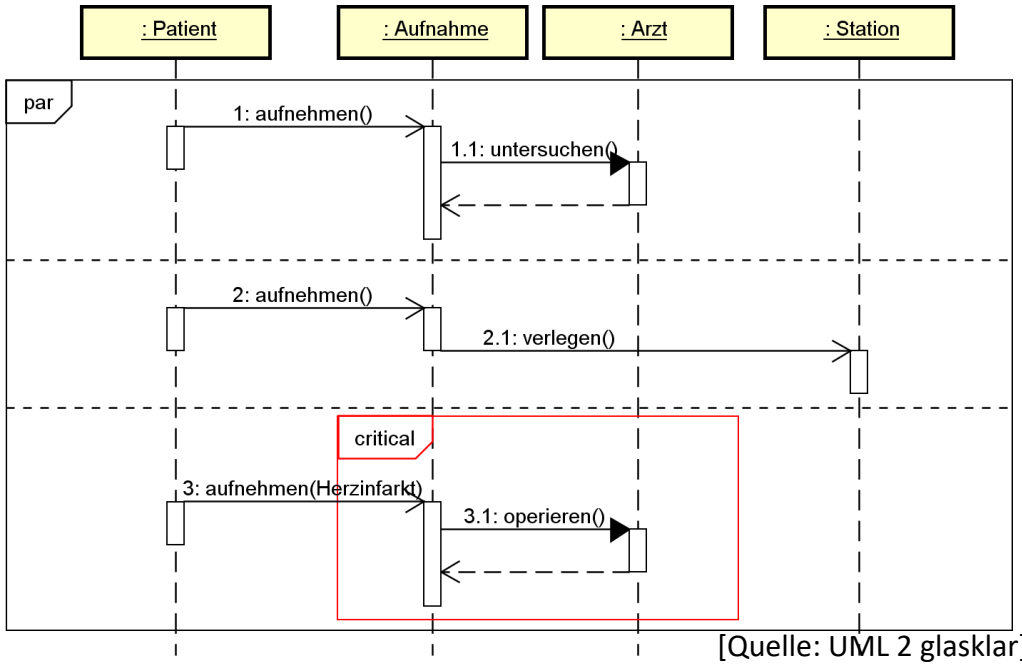
Unfallaufnahme:

- Patienten können über die Aufnahme aufgenommen werden und von einem Arzt untersucht werden. Das Ergebnis wird der Aufnahme mitgeteilt.
- Gleichzeitig können andere Patienten aber nach der Aufnahme auch direkt auf Station verlegt werden
- Kommt ein Patient mit Herzinfarkt, so wird alles stehen und liegen gelassen und der Patient direkt aufgenommen und operiert. Erst danach können wieder andere Patienten aufgenommen werden.

Man benötigt die Operatoren:

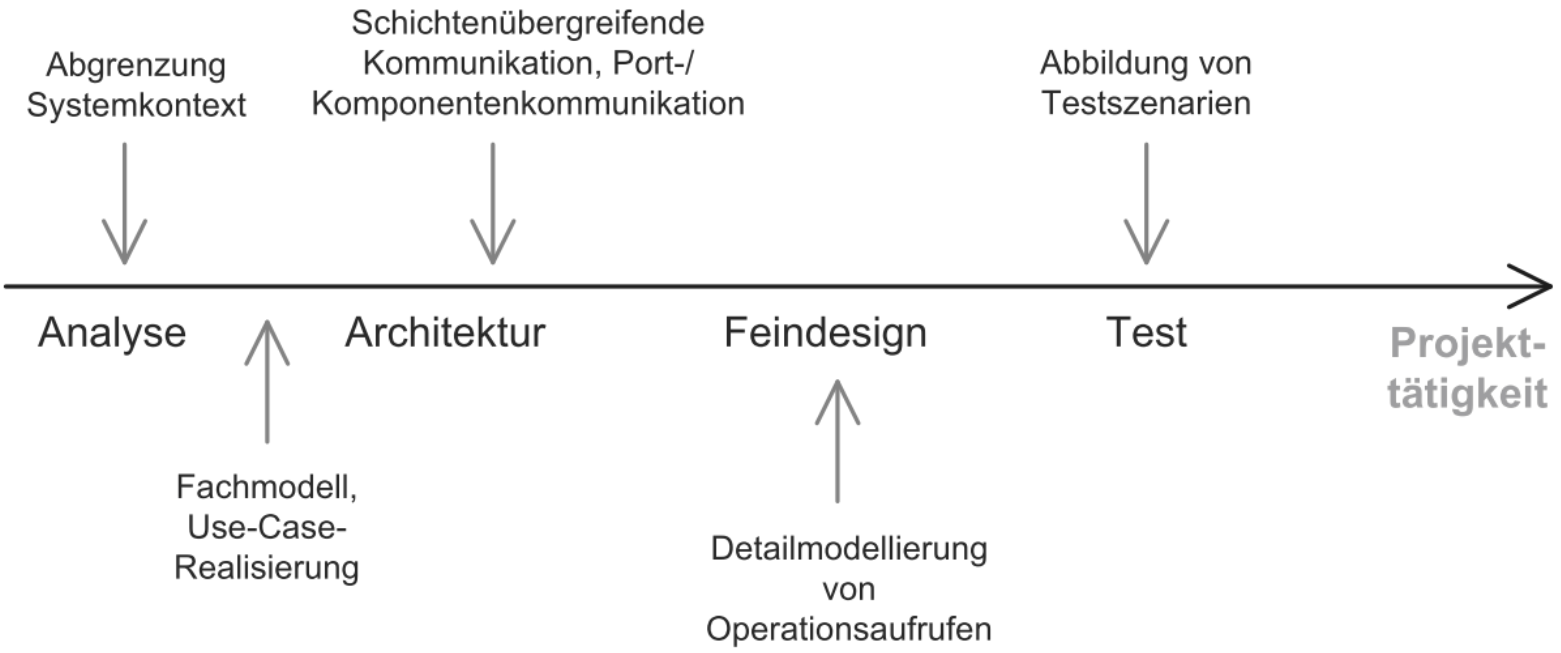
- Parallelität
- Kritischer Bereich

Was fällt auf?
Welche Besonderheiten gibt es?



[Quelle: UML 2 glasklar]

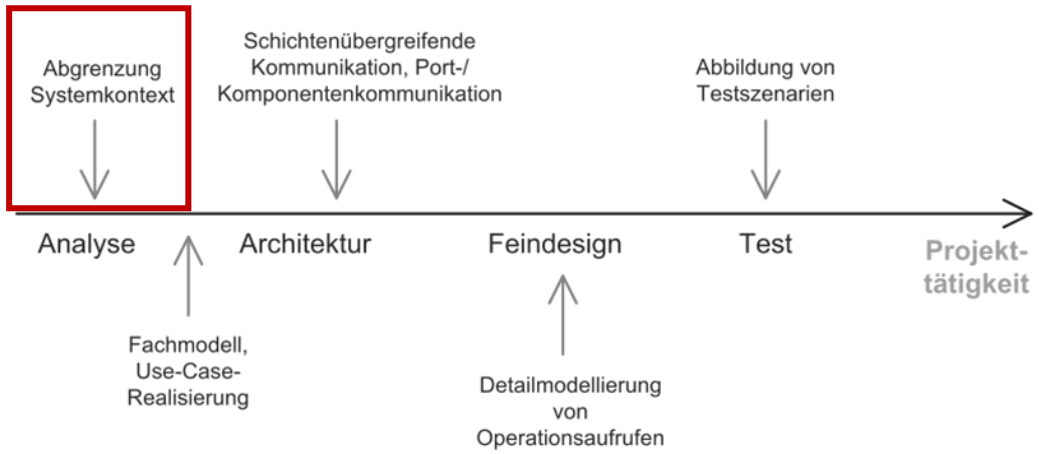
Sequenzdiagramme: Anwendung im Projekt



Fazit:

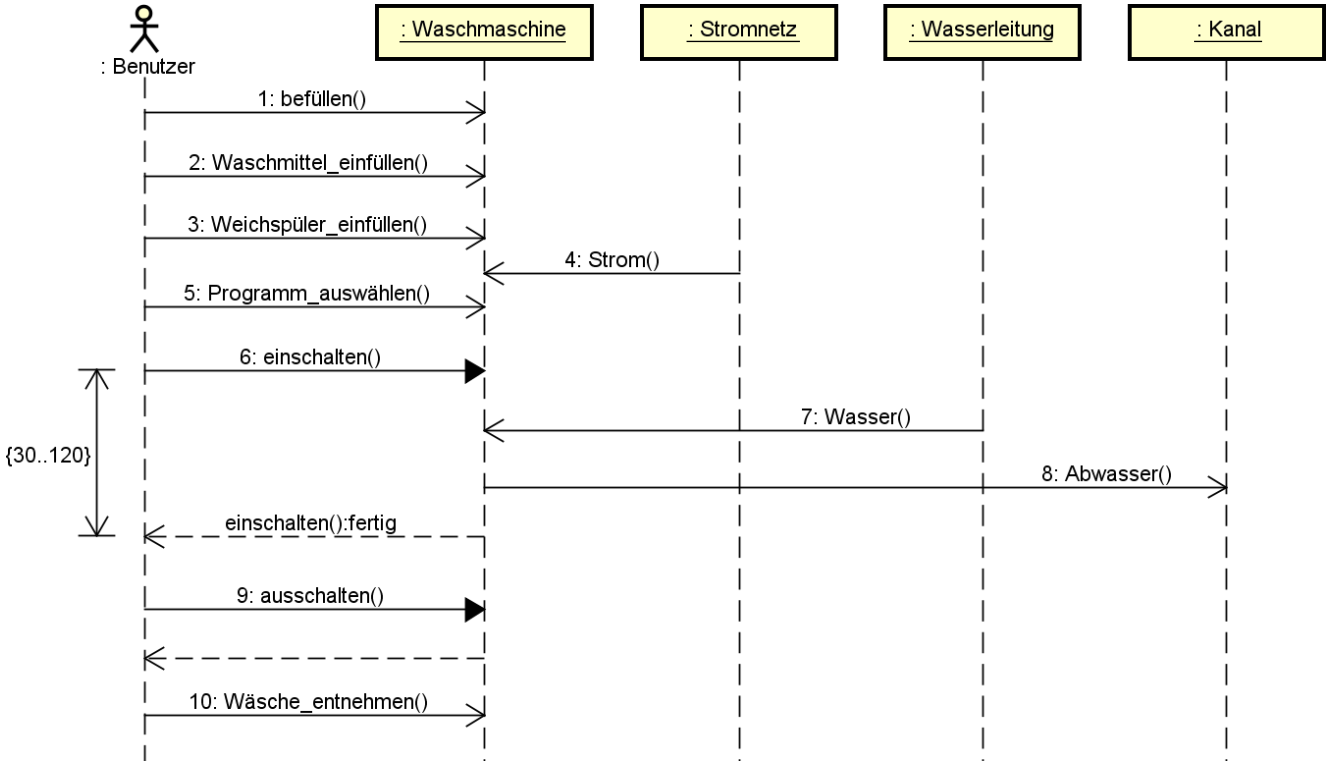
- Es lassen sich mit Sequenzdiagrammen Interaktionen auf verschiedensten Abstraktionsniveaus modellieren
- Sequenzdiagramme sind zu jedem Zeitpunkt im Projekt sinnvoll einsetzbar

Abgrenzung Systemkontext



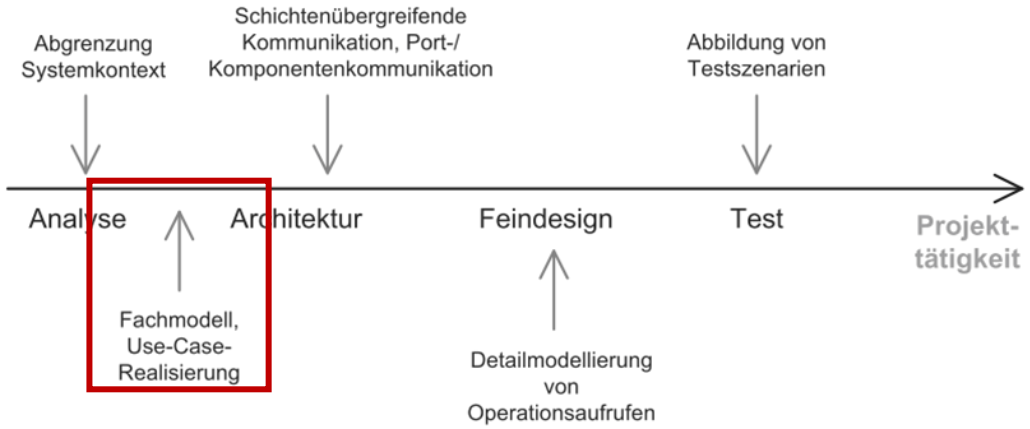
- System selbst wird als Blackbox betrachtet
- Sequenzdiagramme eigenen sich zur Kontextabgrenzung vor allem bei Systemen mit wenig unterschiedlichem und fest definiertem Interaktionsverhalten
- Zu viele Alternativen verhindern sinnvolle Darstellung
- Empfehlung: lieber mehrere Diagramme zeichnen (und evtl. über Interaktions-diagramm verknüpfen)

Abgrenzung Systemkontext: Beispiel



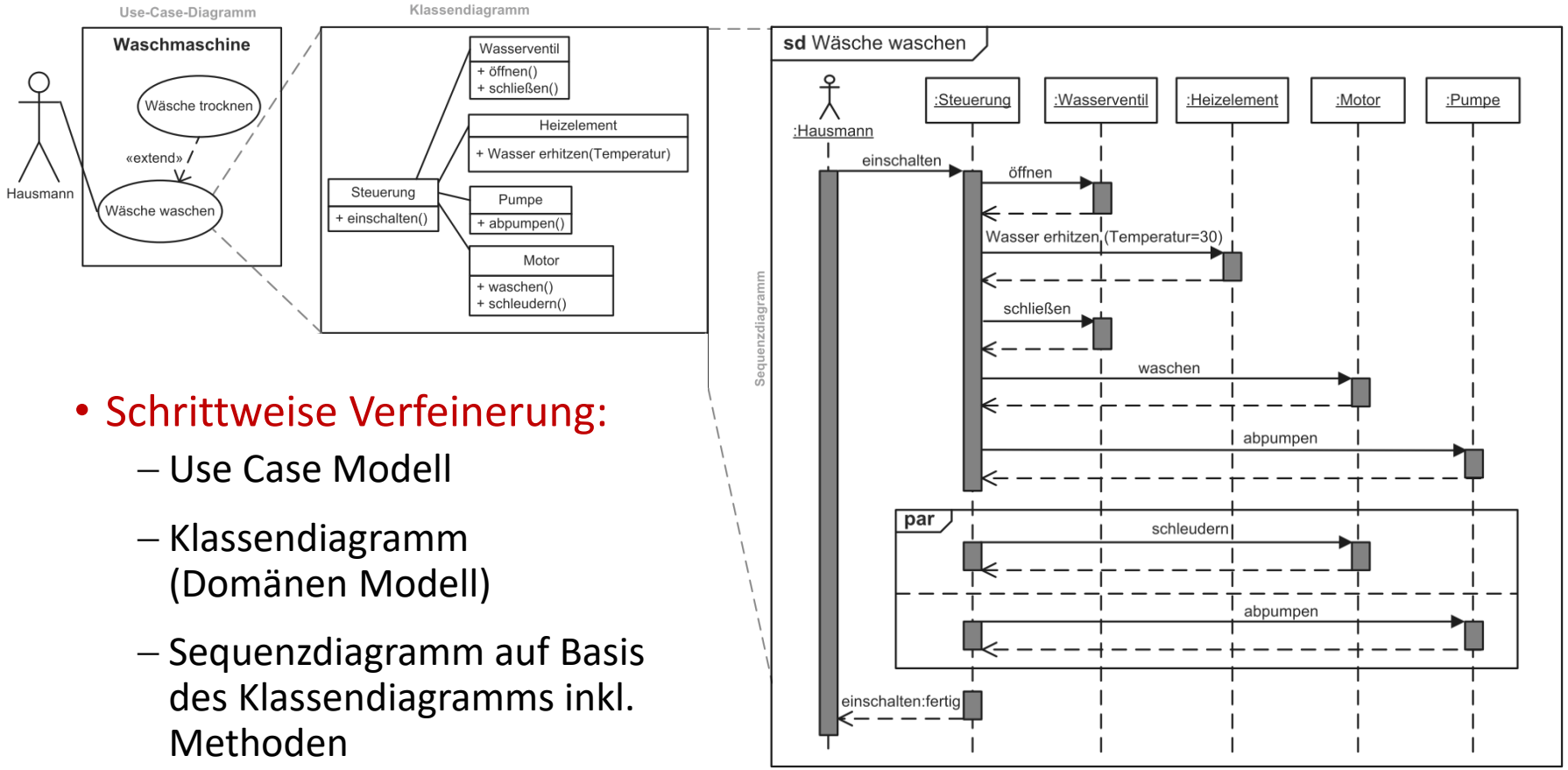
- Sequenzdiagramme zur Abgrenzung des Systemkontextes nutzbar
- Das System (Waschmaschine) im Kontext, mit dem es interagiert:
  - Benutzer, Stromnetz, Wasserleitung, Kanal, …

Use-Case-Modellierung



- **Ziel:** von elementaren Use Cases eines Systems zu detaillierterer Darstellung der zugrundeliegenden Abläufe gelangen (Hilfsmittel: Aktivitäts- oder Sequenzdiagramme)
- Sequenzdiagramme schlagen Brücke zwischen:
  - Strukturiertem oder ablauforientiertem Use-Case-Modell und
  - Objekt- oder datenorientierten Klassen-/Fach-/Analysemodell

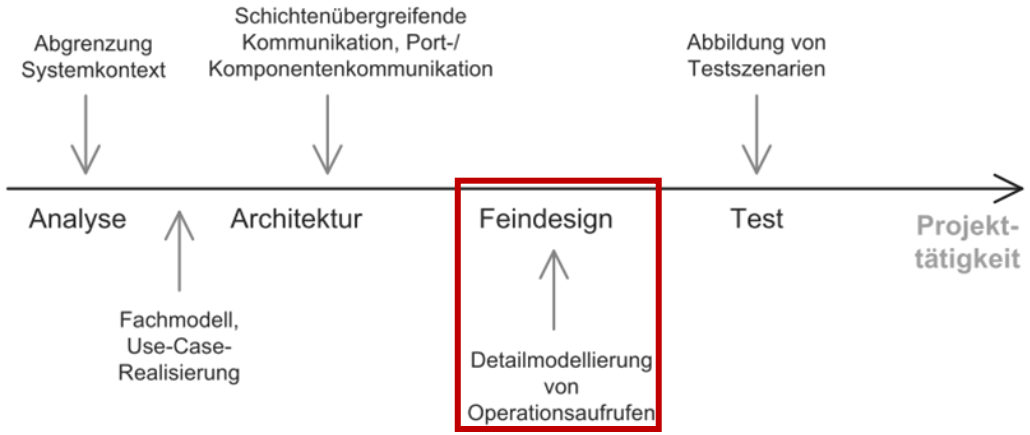
Use-Case-Modellierung: Beispiel



- Schrittweise Verfeinerung:
  - Use Case Modell
  - Klassendiagramm (Domänen Modell)
  - Sequenzdiagramm auf Basis des Klassendiagramms inkl. Methoden

[Quelle: UML 2 glasklar]

Detailmodellierung im Feindesign

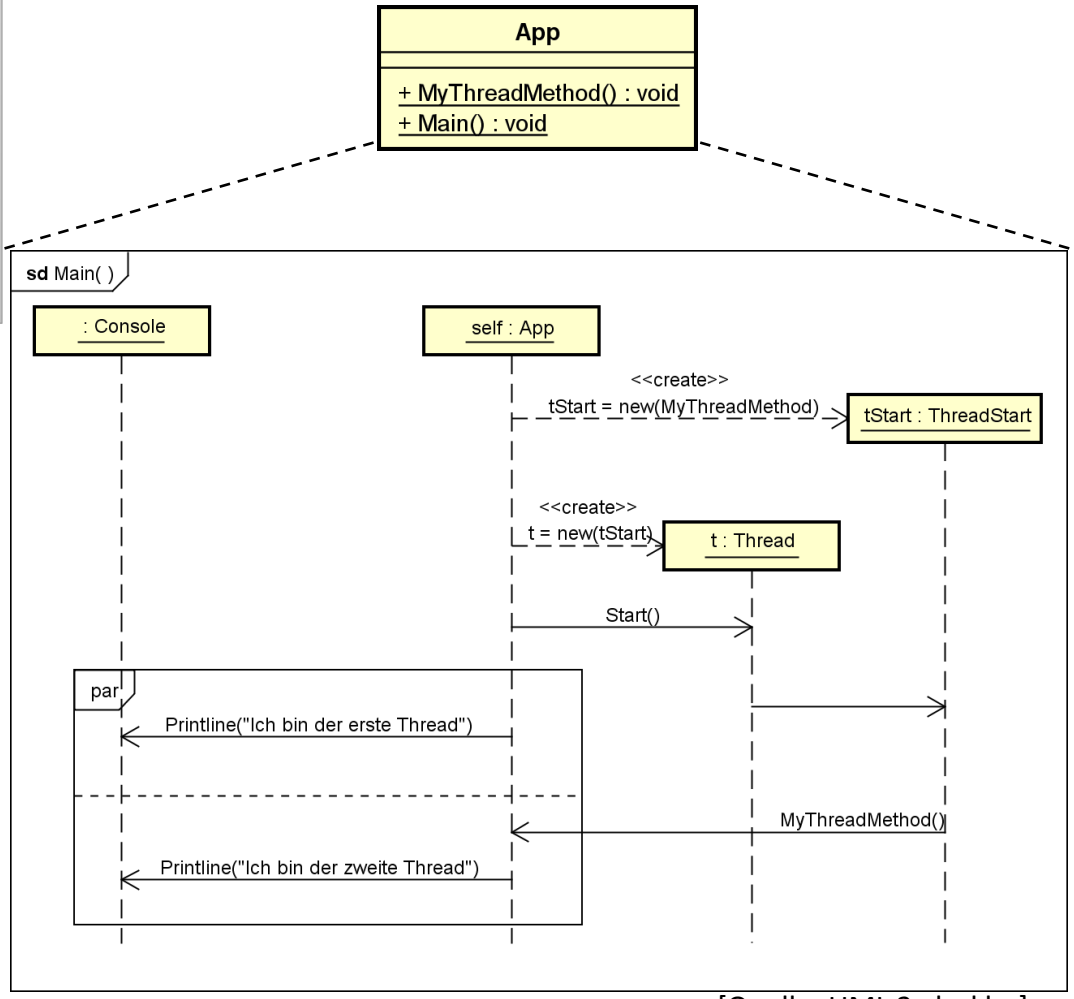


- Sequenzdiagramme bieten Notationselemente für alle relevanten Kontrollelemente höherer Programmiersprachen
- Daher ist mit ihnen eine detaillierte Modellierung von Feindesignmodellen möglich
- Mit entsprechenden Werkzeugen lässt sich daraus vollständig ablaufbarer Code generieren

Detailmodellierung im Feindesign: Beispiel

```
class App {
   public static void MyThreadMethod() {
      Console.Printline(“Ich bin der zweite Thread.”);
   }

   public static void Main() {
      ThreadStart tStart = new ThreadStart(MyThreadMethod);
      Thread t = new Thread(tStart);
      t.Start();
      Console.Printline(“Ich bin der erste Thread.”);
   }
}
```



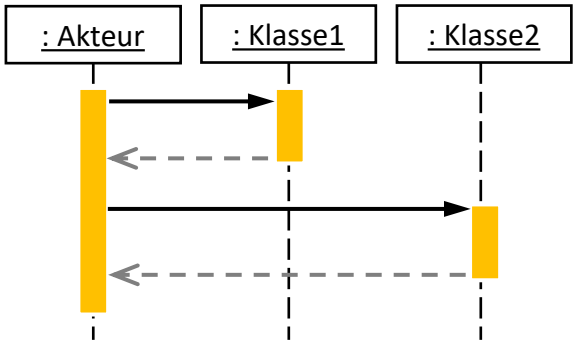
[Quelle: UML 2 glasklar]

Konstruktion von Sequenzdiagrammen

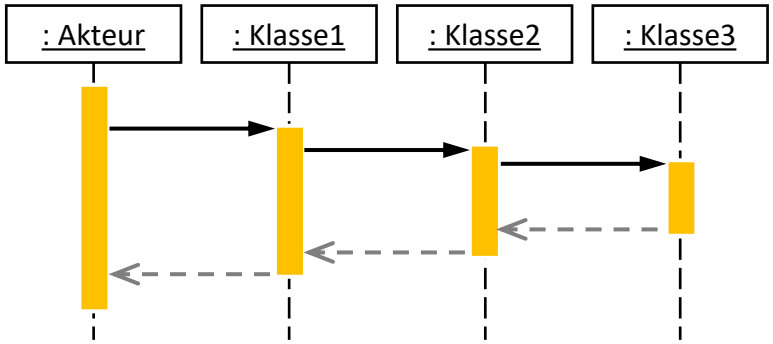
- **Grundidee:**
  - Identifikation aller relevanten Szenarien
  - Erstellen eines Sequenzdiagramms für jedes Szenario
- **Vorgehen:**
  1. Pro Use Case mehrere Szenarios entwickeln
     - Standardausführung und Alternativen berücksichtigen
     - Positive und negative Fälle unterscheiden (Verhalten das passieren soll oder nicht passieren darf)
  2. Entwicklung eines Sequenzdiagramms
     - Beteiligte Klassen finden
     - Aufgaben in Operationen zerlegen
     - Reihenfolge der Operationen prüfen
  3. Identifizierte Operationen den einzelnen Klassen zuordnen

Konstruktion von Sequenzdiagrammen für Use-Case-Modellierung

- **Grundidee:**
  - Identifikation aller relevanten Szenarien
  - Erstellen eines Sequenzdiagramms für jedes Szenario
- **Vorgehen:**
  4. Wie ist das Szenario zu strukturieren?
     - Zentrale Struktur (fork diagram) vs
     - Dezentrale Struktur (stair diagram)



zentral



dezentral

Sequenzdiagramme: Analytische Schritte

1. Sind Empfänger-Objekte erreichbar?
   - Assoziationen existieren (permanente Objektbeziehung)
   - Identität kann dynamisch ermittelt werden (temporäre Objektbeziehung)

2. Ist Sequenzdiagramm konsistent mit Klassendiagramm?
   - Alle Klassen sind auch im Klassendiagramm enthalten
   - Mit Ausnahme von Verwaltungsoperationen werden nur Operationen aus dem Klassendiagramm eingetragen

Sequenzdiagramme: Fehlerquellen

Fehlerquellen:

- Zu viele Details beschreiben
- Zu viele Operatoren nutzen

Stattdessen:

→ Auf die wichtigsten Szenarien beschränken

→ Nicht jeden Sonderfall beschreiben

→ Auf unnötige Details verzichten

→ Lieber mehrere einfache Sequenzdiagramme als wenige sehr komplext

Sequenzdiagramme: Typische Modellierungsfehler