

# Softwaretechnik

## Modellierung von Sequenz- diagrammen mithilfe der UML

Prof. Dr. Bodo Kraft

# Übersicht UML-Diagramme

## Sequenzdiagramm

### Diagramme der UML 2

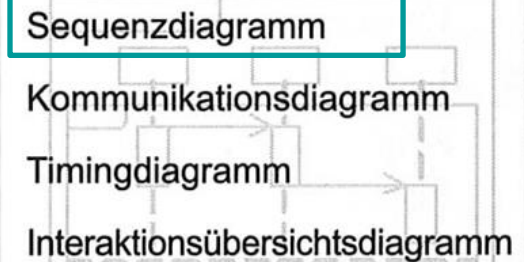
#### Strukturdiagramme



#### Verhaltensdiagramme



#### Interaktionsdiagramme



Quelle: UML 2 glasklar, Chris Rupp

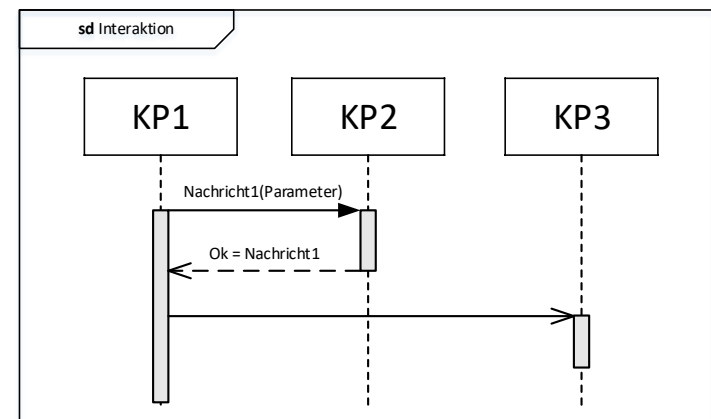
# Motivation

## Sequenzdiagramm

- Zeigen den Informationsaustausch zwischen Kommunikationspartnern (innerhalb eines Systems oder zwischen mehreren Systemen)
- Interaktionen erfolgen via Nachrichtenaustausch
- Zeigen Ablaufdetails in einem konkreten Szenario (vgl. UseCase-Diagramme)

Liefern Antwort auf die Frage:

*„Wie läuft die Kommunikation in meinem System ab?“*



# Definition

## Sequenzdiagramm

---

Sequenzdiagramme sind **Interaktions- und Verhaltensdiagramme**.

## Was genau versteht man unter einer Interaktion?

*„Immer dann, wenn zwei oder mehrere Einheiten, die ein eigenes Verhalten realisieren, miteinander kommunizieren, spricht man von Interaktion.*

*Eine Interaktion definiert damit eine spezielle Art von Verhalten.“*

*[Rupp, UML2 glasklar]*

Eine Interaktion verbindet mehrere (i.d.R.) Objekte und regelt den Kommunikationsfluss mithilfe von **ausgetauschten Nachrichten**.

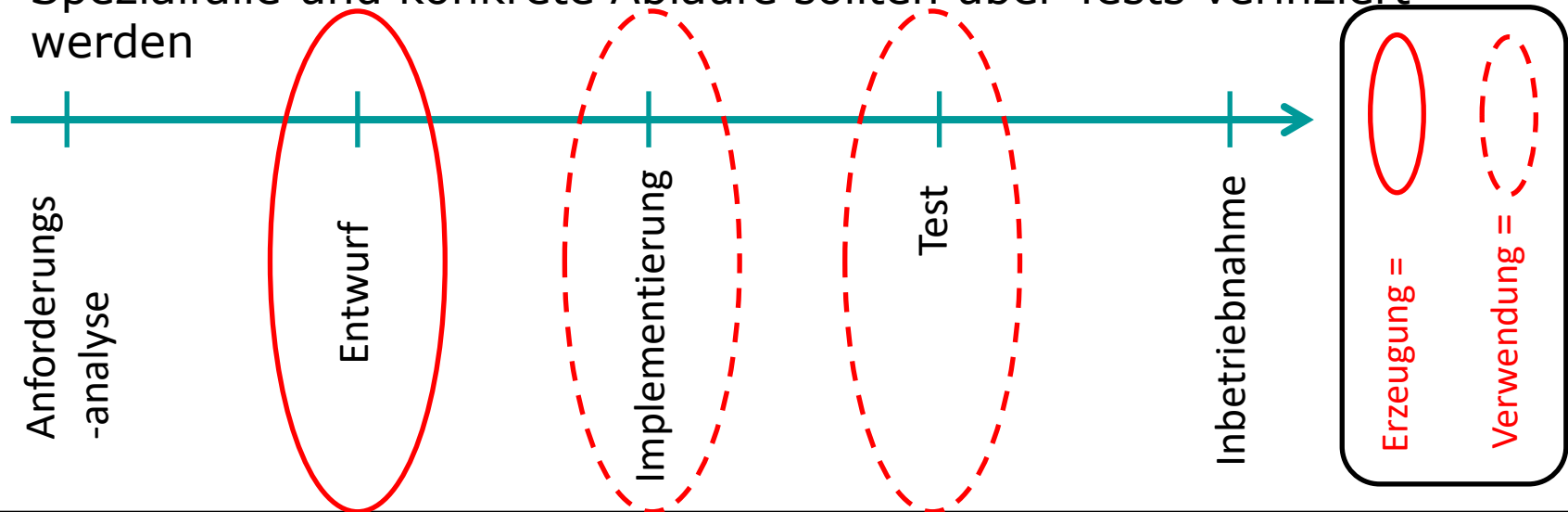
Die beteiligten Objekte treten paarweise in den Rollen **Sender** und **Empfänger** auf.

# Zeitliche Einordnung in SW-Lifecycle

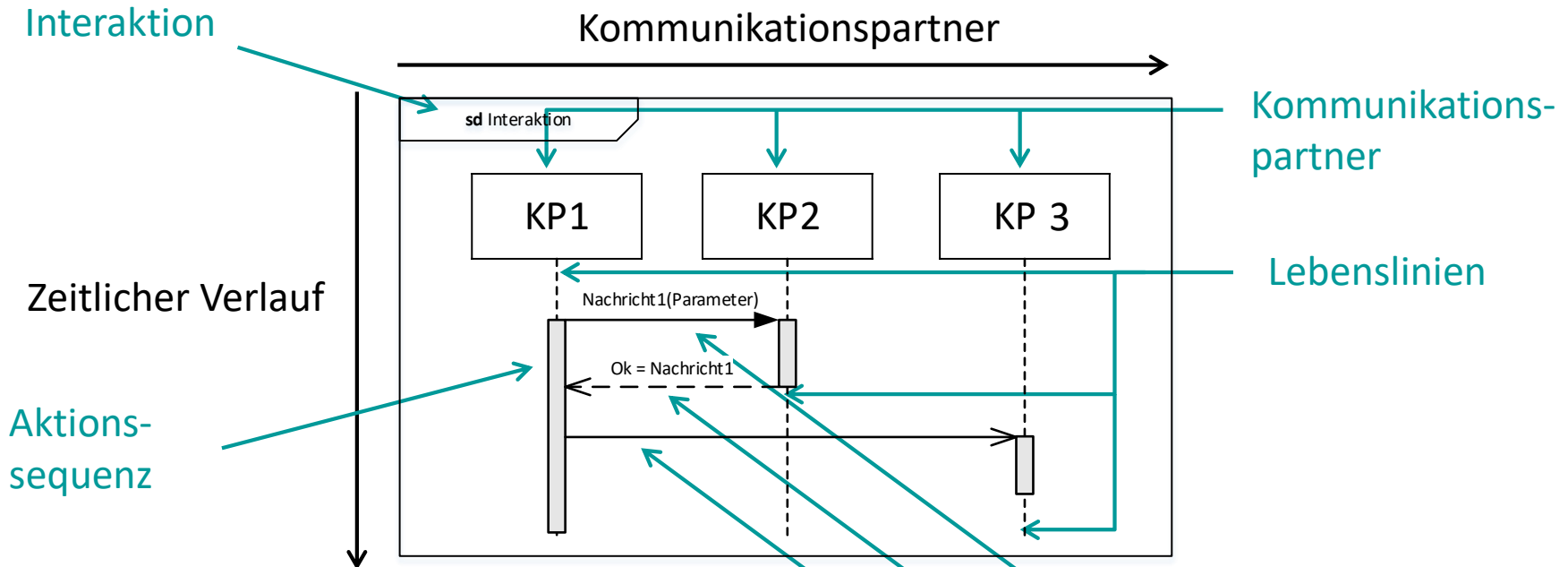
## Sequenzdiagramm

### Bei welchen Schritten des Software-Lifecycle sind Sequenzdiagramme(SD) relevant?

- SD verdeutlichen einzelne konkrete Abläufe des Programmes (Methodenaufrufe zwischen Klassen)
- Geeignet zur Darstellung von Sonder-/Spezialfällen
- SD eignen sich als Vorgabe für die Implementierung
- Spezialfälle und konkrete Abläufe sollten über Tests verifiziert werden



# Übersicht Elemente Sequenzdiagramm



Die wichtigsten Elemente sind:

1. Lebenslinien & Kommunikationspartner
2. Nachrichten
3. Interaktion
4. Aktionssequenzen/Aktivierungsbalken
5. Weitere Sprachmittel zur Flusskontrolle

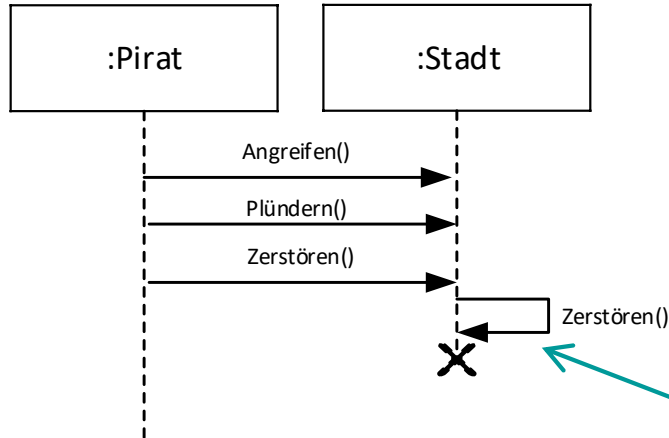
# Lebenslinien

## Syntaxelemente im Detail

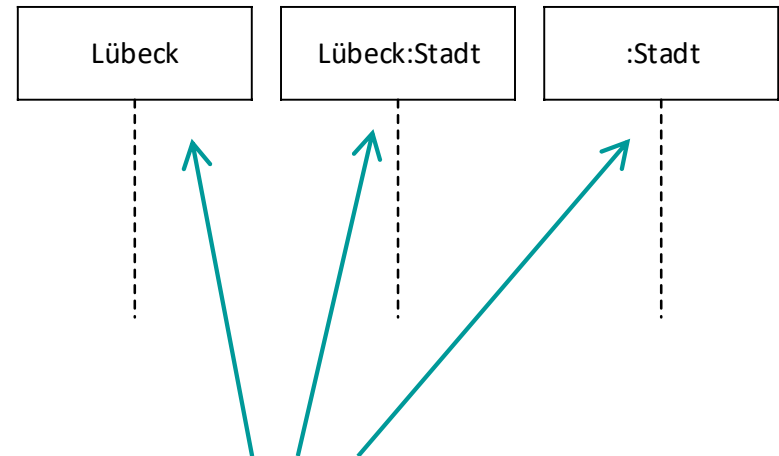
- Jede Lebenslinie repräsentiert **genau einen Teilnehmer** / eine Objektinstanz in der Interaktion.
- Verdeutlicht den zeitlichen Ablauf

Sie besteht aus 2 Komponenten

- Kopf (Rechteck mit Name des K.P.)
- Vertikale, gestrichelte Linie



**Pirat lebt weiter,  
Stadt existiert nicht mehr**



Für die Benennung gibt es 3 Darstellungsarten:

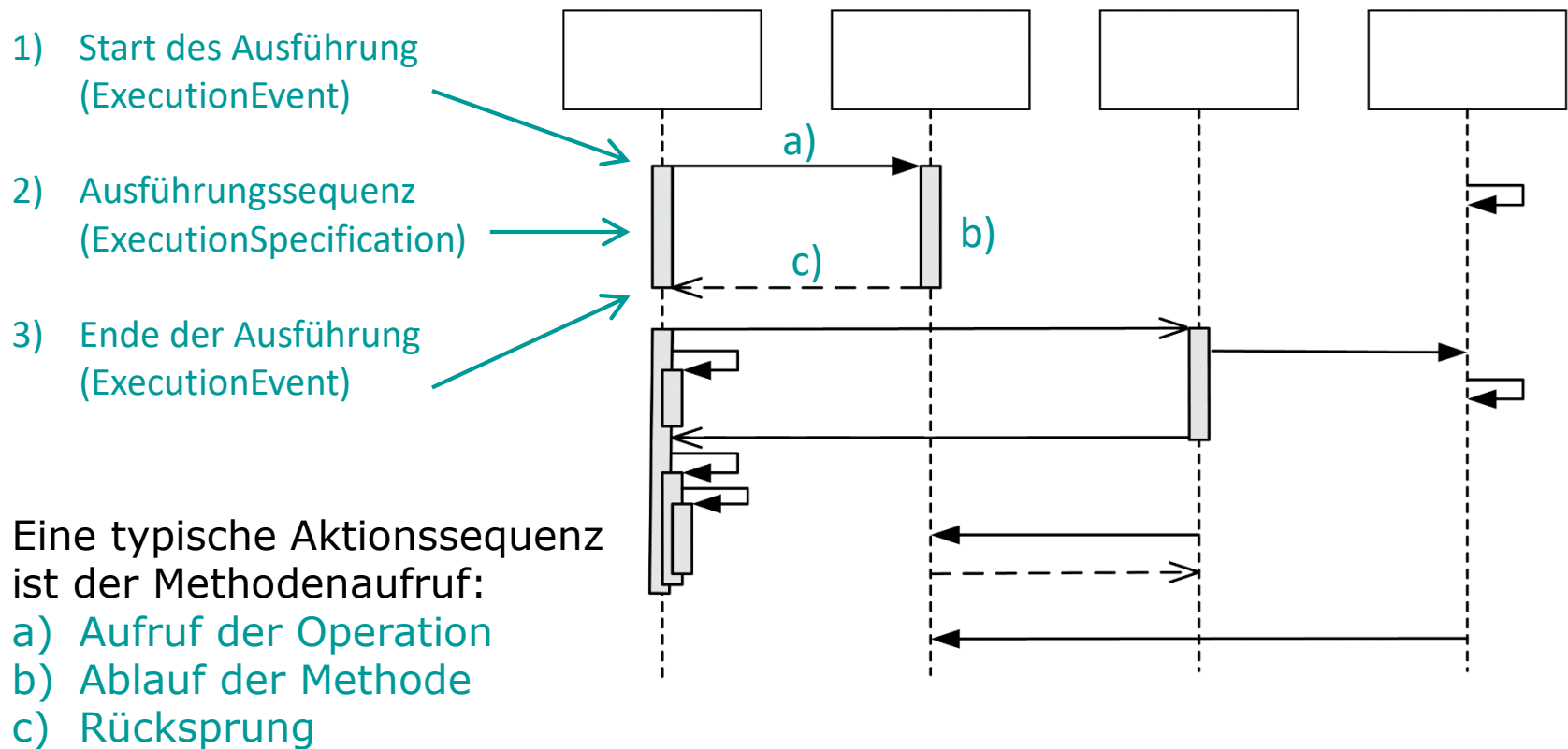
- **Expliziter Name (ohne Typ)**
- **Expliziter Name (mit Typ)**
- **Anonyme Instanz (mit Typ)**

*Ein Kommunikationspartner lebt bis ans Ende des Diagramms, es sei denn er wird **explizit zerstört/beendet**.  
→ Symbol: „X“*

# Aktionssequenzen/Aktivierungsbalken

## Syntaxelemente im Detail

Aktionssequenzen verdeutlichen den Kommunikationsfluss. Sie sind keine Pflicht, aber hilfreich. Ein Beispiel:





# Nachrichten (I) beschreiben Kommunikation

## Syntaxelemente im Detail





Die UML unterscheidet bei Nachrichten zwischen **asynchroner und synchroner Kommunikation**.

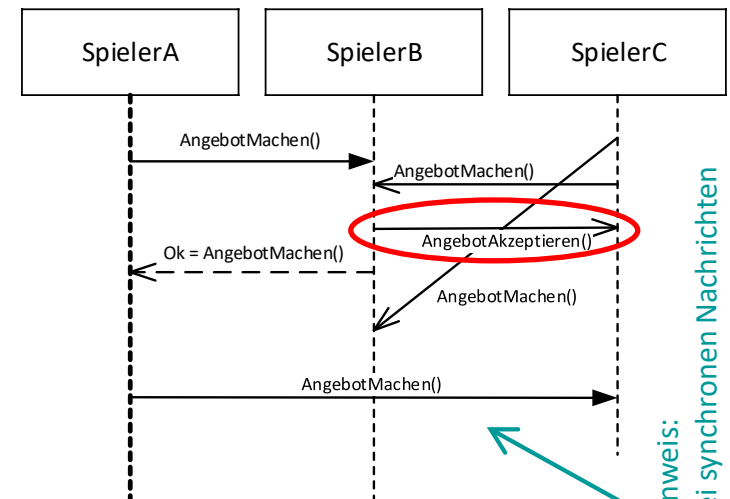
### Synchrone Kommunikation:

- (1) Sender verschickt synchrone Nachricht und blockiert.
- (2) Empfänger verarbeitet die Nachricht.
- (3) Empfänger liefert Antwortnachricht zurück.
- (4) Sender empfängt Antwortnachricht und löst Blockade auf.
- (5) Sender fährt mit Abarbeitung fort.

### Asynchrone Kommunikation:

- (1) Sender verschickt asynchrone Nachricht.
- (2) Sender fährt mit Abarbeitung fort (nebenläufig, keine Blockade)
- (3) Empfänger verarbeitet Nachricht und antwortet.
- (4) Senden/Empfang der Antwortnachricht meist als **asynchrone Nachricht** modelliert.

Nachrichtentyp	Symbol
Synchrone Nachricht	
Asynchrone Nachricht	
Antwortnachricht (auf synchrone Nachricht)	
Erzeugungsaufwurf	



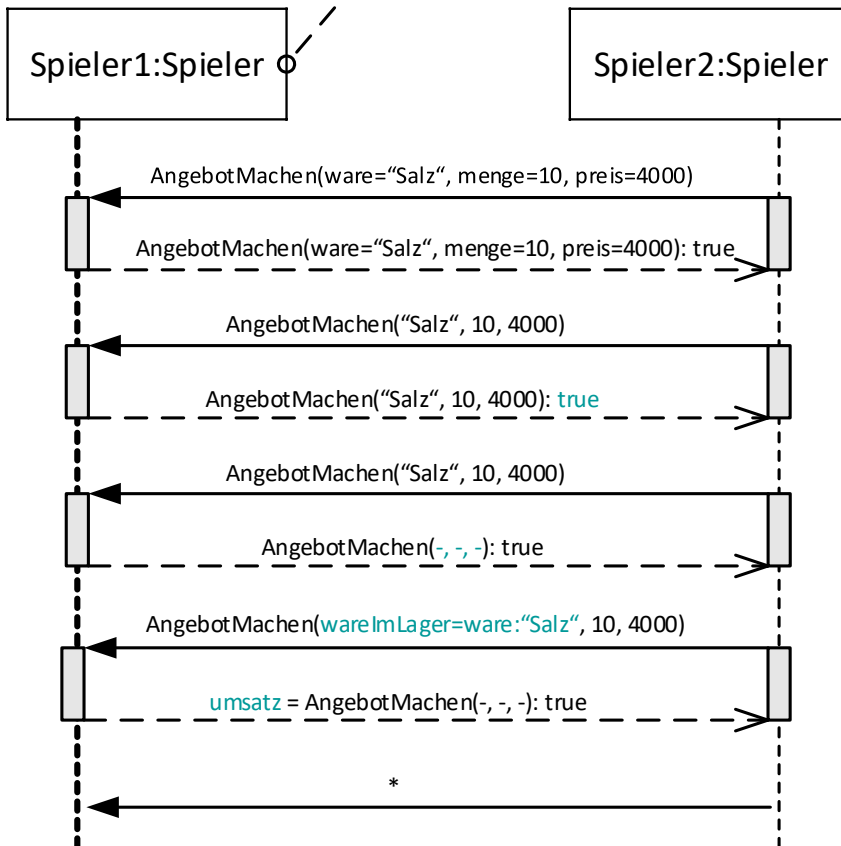
Hinweis:  
Bei synchronen Nachrichten  
muss die Antwortnachricht  
nicht explizit angegeben werden.

# Nachrichten (II) visualisieren Methodenaufrufe

## Syntaxelemente im Detail

```
- wareImLager: int = 0
- umsatz: int

-----
AngebotMachen(inout ware: String, in menge: int, in preis: int) : boolean
```



- Nachrichten können auch Signale darstellen.
  - Hinweis: **Signale sind immer asynchron!**
  - Ansonsten grafisch kein Unterschied
- Im Sequenzdiagramm können Nachrichten unterschiedlich benannt werden.

### Syntax für Benennung:

**<NachrichtenSignatur> ::=**

```
{[<Attribut> =] <NachrichtenName> [ ( [<Argument> [,
<Argument>]* ) ] [: <Rückgabewert>] ] | *
```

**<Argument> ::=**

```
{[<ParameterName> =] <ArgumentWert>}
| {[<Attribut>=<AusgabeParameterName> [: <ArgumentWert>]]}
| {-}
```

**Spezielle Symbole:**

- \* (generischer Name)
- (Argumente anonymisieren)

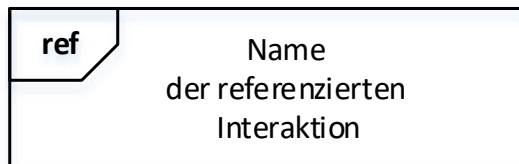
# Sprachmittel zur Strukturierung (I)

## Syntaxelemente im Detail

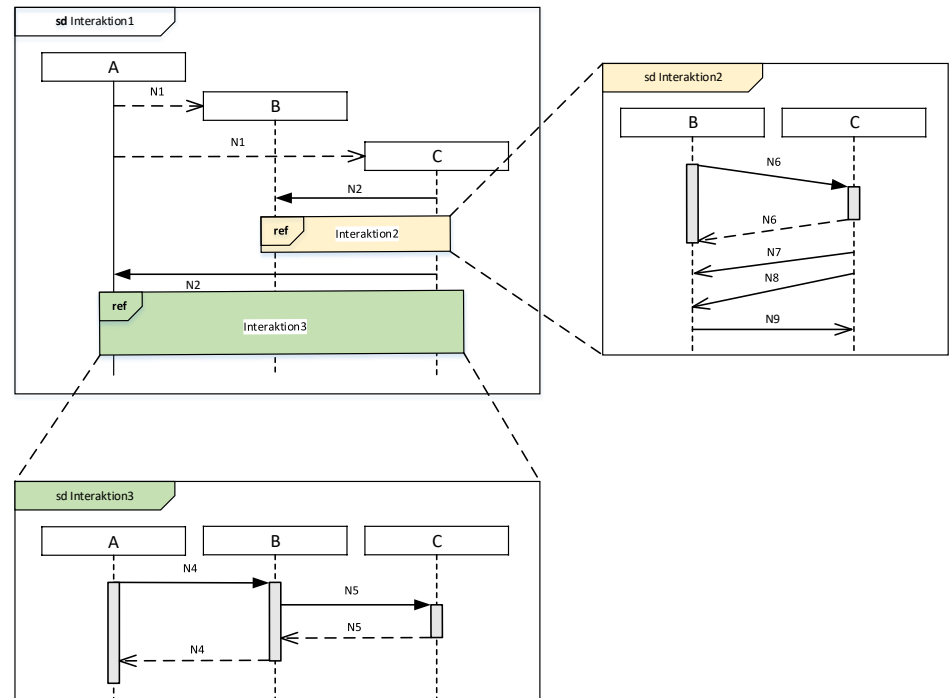
Auslagern von Teilen eines Sequenzdiagrammes mithilfe einer

## Interaktionsreferenz


Symbol:



Beispiel:



Hinweis:

Vergleiche -Symbol im Aktivitätsdiagramm.

# Sprachmittel zur Flusskontrolle (II)

## Syntaxelemente im Detail

Für komplexere Abläufe und bedingte Ausführungsfolgen erlaubt die UML2 weitere Sprachmittel:

Operator	Bedingung/ Parameter	Bedeutung
alt	[bed. 1] [bed. 2] [sonst]	Alternative; Ausführung einer von mehreren Möglichkeiten (Trennung jew. durch gestrichelte Linie)
loop	minint maxint [bed. ]	Teile des Kommunikationsablaufes wiederholen sich. Darstellung als Schleifendurchlauf, mindestens <b>minint</b> , maximal <b>maxint</b> Durchläufe werden erzwungen.
break	[bed. ]	Falls Bedingung erfüllt ist, wird die umgebende Interaktion beendet.
opt	[bed. ]	Optional ausführbare Sequenz.
par	-----	Parallele/nebenläufige Ausführung. Ausführung jeder Sequenz in eigenem Thread. (Trennung jew. durch gestrichelte Linie)

# Sprachmittel zur Flusskontrolle (III)

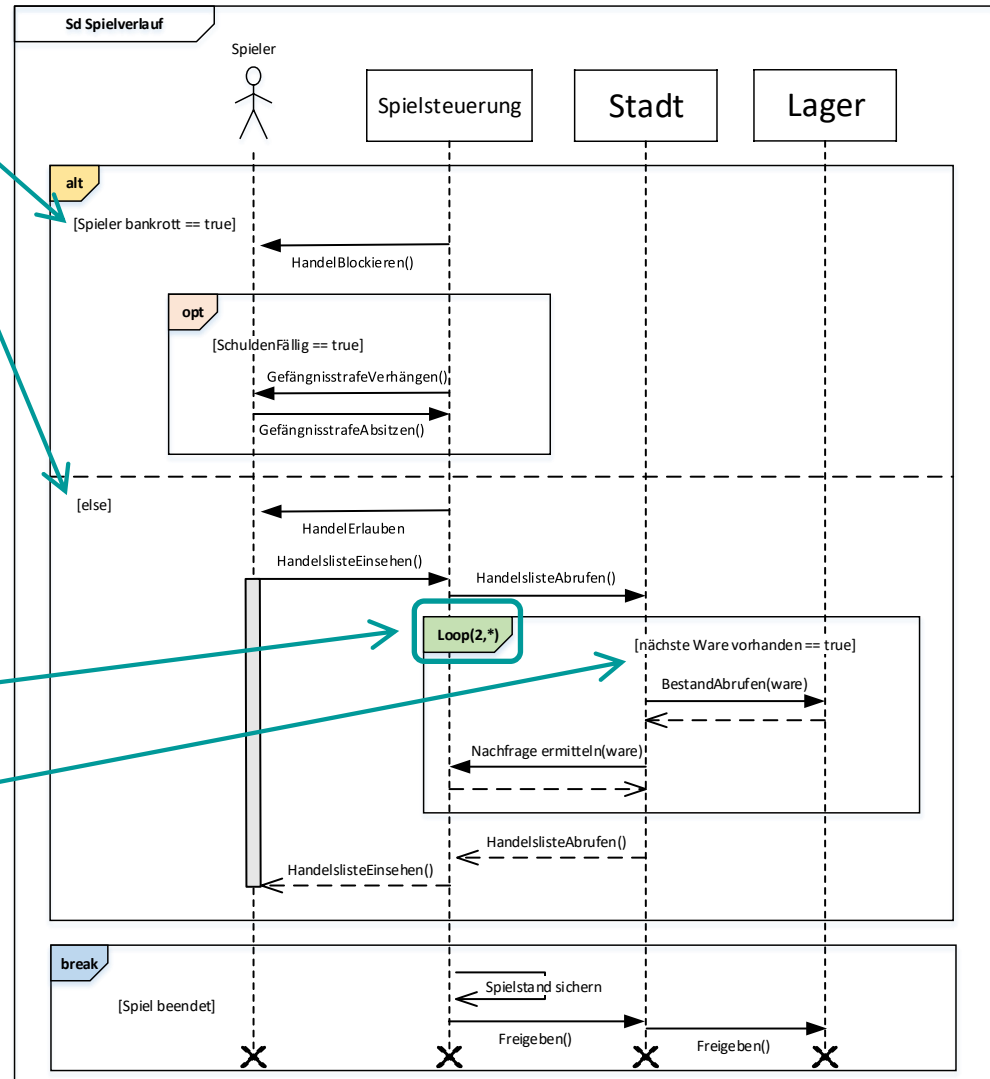
## Syntaxelemente im Detail

Alternativen müssen  
**disjunkt** sein.

Hinweis:  
Auch mehrere Fälle möglich  
(switch/case)

Schleifen:

- Minint = 2
- Maxint = \* (beliebig)
- Abbruchbedingung



# Handelsszenario bei Hanse

## Präsenzaufgabe

---

Erstellen Sie ein Sequenzdiagramm, das folgende Situation darstellt:

- Es wird das Beladen und Verschicken von Schiffen beschrieben.
- Der Spieler initiiert das Beladen eines bestimmten Schiffes mit 100 Einheiten der Ware "Pelz".
- Dies wird auch dem Schiff mitgeteilt, dessen Ladung sich verändert. Diese Änderung hat natürlich auch Auswirkung auf das Lager der Hauptstadt.
- Nun versendet der Spieler das Schiff. Es wird die passende Distanz errechnet und mit dem Ergebnis bei dem Schiff das Attribut "Runden bis Rückkehr" gesetzt.

Stellen sie die Objekte und die verschiedenen Nachrichten dar, die zwischen diesen ausgetauscht werden.

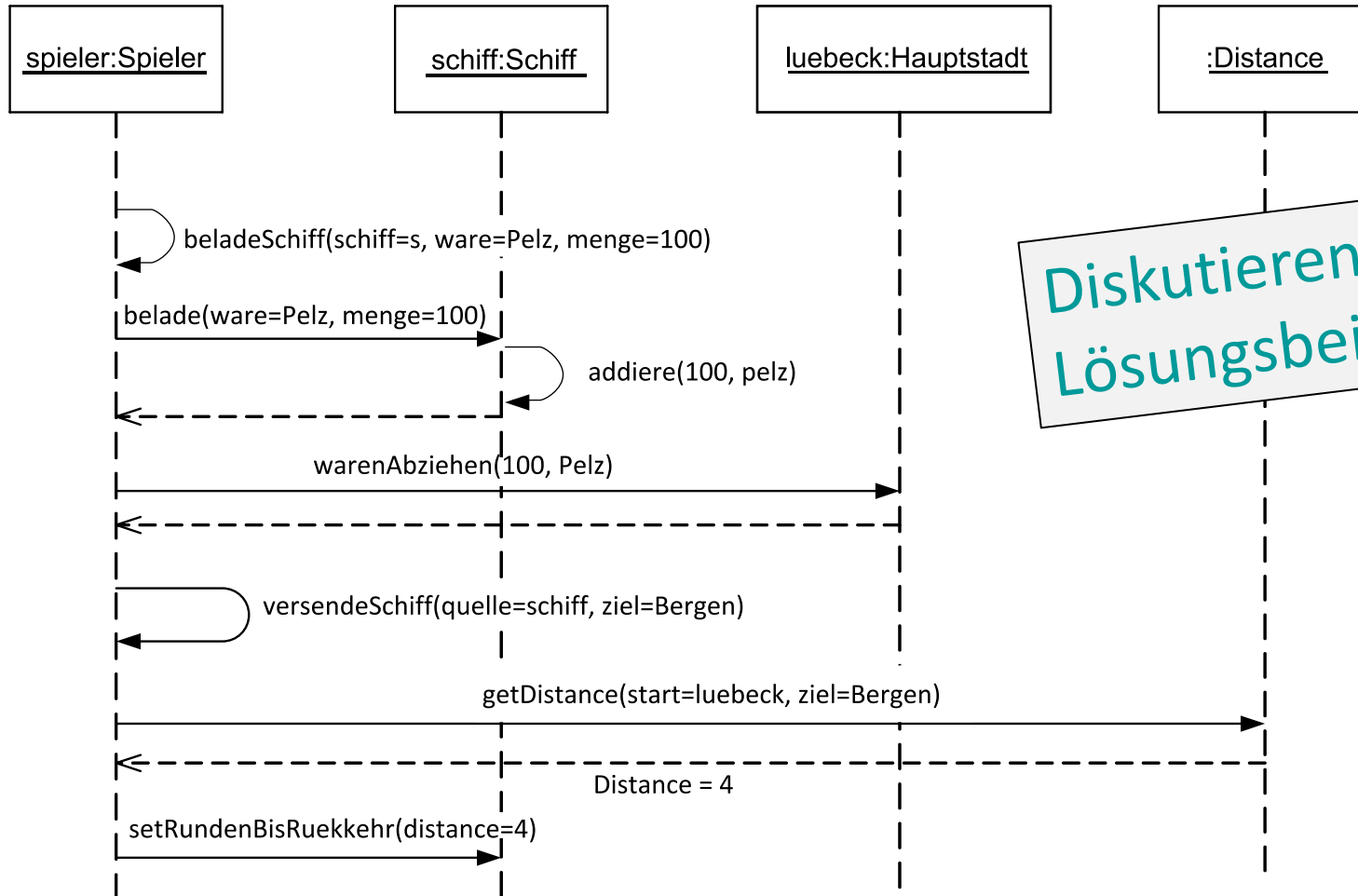
Welche Arten von Nachrichten werden verwendet?

Wo macht ein Selbstaufruf Sinn?

# Handelsszenario bei Hanse

## Lösung zur Aufgabe

sd Schiff beladen  
und verschicken



Diskutieren des  
Lösungsbeispiels

- [RS] C. Rupp, SOPHIST GROUP, Requirements- Engineering und – Management, Hanser Fachbuchverlag, 2004
- [OW] B. Oestereich, C. Weiss, C. Schröder, T. Weilkiens, A. Lenhard, Objektorientierte Geschäftsprozessmodellierung mit der UML, dpunkt.Verlag, 2003



---

# Vielen Dank!