Grundlagen der objektorientierten Modellierung mit UML 2.1

Sequenzdiagramme

Stand 13.04.2016

Vorbemerkung: Was ist ein Szenario?

- Sequenz von Verarbeitungsschritten, die unter bestimmten Bedingungen auszuführen ist.
- Diese Schritte sollen das Hauptziel eines Akteurs realisieren und ein entsprechendes Ergebnis liefern.
- Sie beginnen mit einem auslösenden Ereignis und werden fortgesetzt, bis das Ziel erreicht ist oder aufgegeben wird.
- Eine Kollektion von Szenarios kann einen *Use Case* (Geschäftsprozess) dokumentieren.
- Jedes Szenario wird durch eine oder mehrere Bedingungen definiert, die zu einem speziellen Ablauf des jeweiligen *Use Cases* führen.

Szenario - Beispiel *Telefonverbindung*

Anrufer hebt Hörer ab
Wählton (Freizeichen) beginnt
Anrufer wählt Ziffer (5)
Wählton (Freizeichen) endet
Anrufer wählt Ziffer (4)
Anrufer wählt Ziffer (3)
Anrufer wählt Ziffer (2)
Angerufenes Telefon beginnt zu klingeln
Anrufendes Telefon signalisiert Rufton
Angerufener Teilnehmer meldet sich
Angerufenes Telefon hört auf zu klingeln
Anrufendes Telefon signalisiert Rufton nicht mehr
Telefone werden verbunden
Angerufener Teilnehmer hängt ein
Telefone werden getrennt
Anrufer hängt ein

Was sind Interaktionsdiagramme?

- Szenarios werden durch Interaktionsdiagramme modelliert.
- Interaktionsdiagramme beschreiben die Zusammenarbeit (Interaktion) mehrerer Objekte in einem Verhalten
- Die UML bietet zwei Arten von Diagrammen an:
  - **Sequenzdiagramm (*sequence diagram*)**
  - **Kommunikationsdiagramm (*communication diagram*)**

Was sind Sequenzdiagramme?

Sequenzdiagramme beschreiben:

- die Kommunikation zwischen Objekten in einem bestimmten Szenario
- Welche Objekte am Szenario beteiligt sind
- Welche Informationen (Nachrichten) sie austauschen
- In welcher **zeitlichen** Reihenfolge der Informationsaustausch stattfindet

Ein Sequenzdiagramm besitzt zwei Achsen:

1. die Vertikale repräsentiert die Zeit (von oben nach unten fortschreitend)
2. auf der Horizontalen werden Objekte eingetragen.

Die Reihenfolge der Pfeile gibt die zeitliche Reihenfolge der Nachrichten an

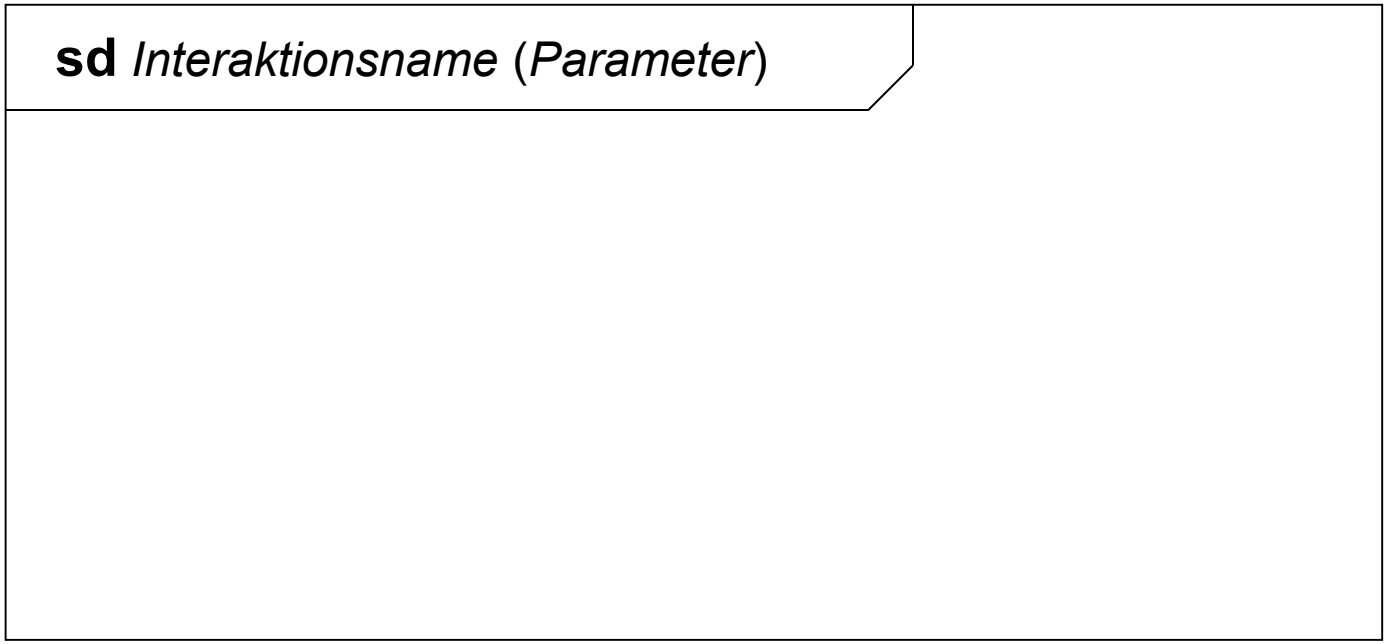
Komponenten eines Sequenzdiagramms

Nachrichten (Botschaften)

- Grundelement einer Interaktion
- Werden von einem Sender zu einem oder mehreren Empfängern gesendet.
- Beispiele:
  - Der Aufruf einer Operation (bei einer Klasse).
  - Die Rückantwort als Ergebnis einer Operationsabarbeitung
  - Ein Signal (z.B. zur Übertragung eines Zeitereignisses)
  - Ein logisches, analytisches Ereignis („Käufer unterschreibt Vertrag“)
- Treten immer als Ereignispaar auf
  1. Sendeereignis beim Sender als Auslöser des Ereignisses
  2. Empfangsereignis beim Empfänger zeitgleich mit dem Eintreffen der Nachricht
- Zeitlicher Abstand der Ereignisse ist beliebig bestimmbar

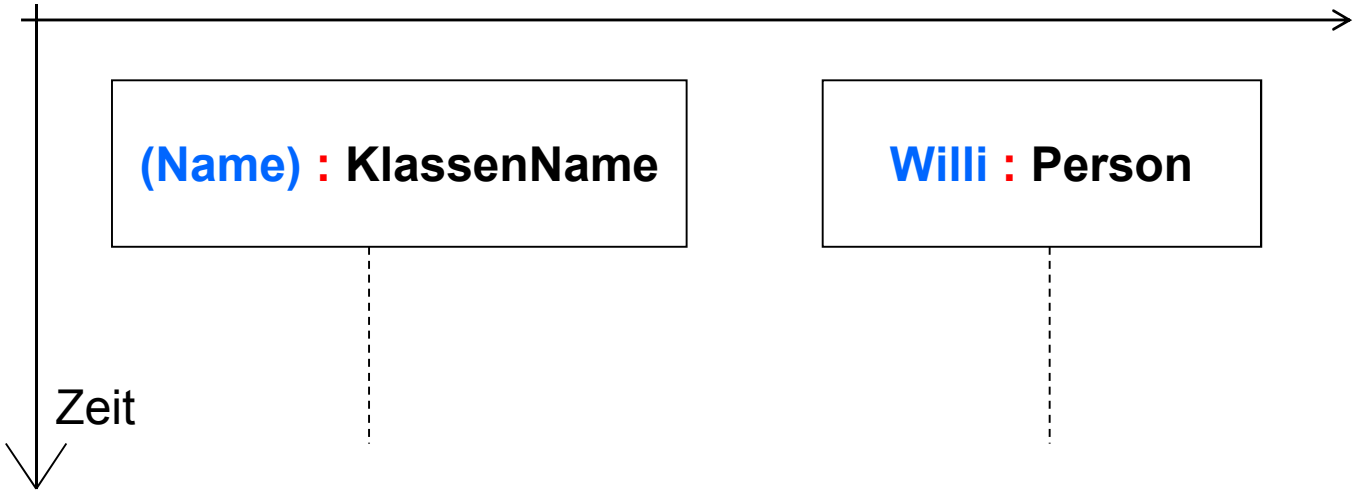
Notation einer Interaktion

- Eine Interaktion wird als Rahmen repräsentiert (umschlossen).
- Interaktionen dürfen Ein- und Ausgabeparameter haben
- Interaktionen dürfen Vor- und Nachbedingungen haben (➔Notiz)



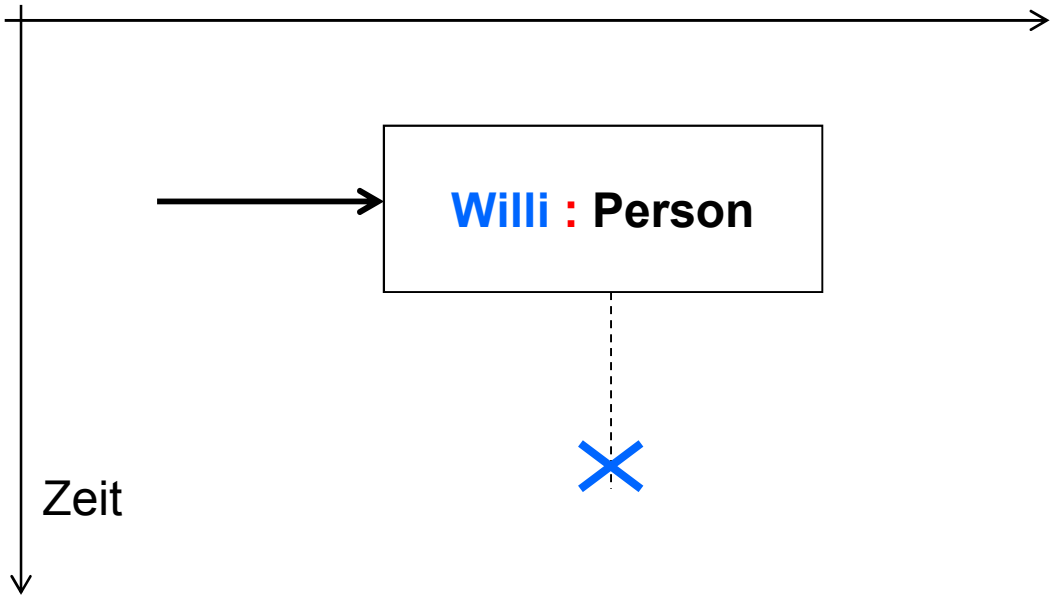
Objektsymbol, Objektlinie

- Jedes Objekt wird durch eine gestrichelte Linie (**Objektlinie**) sowie einem **Objektsymbol** im Diagramm dargestellt.
- Diese Linie repräsentiert die Existenz eines Objekts während einer bestimmten Zeit.
- Die horizontale Reihenfolge der Objekte kann beliebig sein.



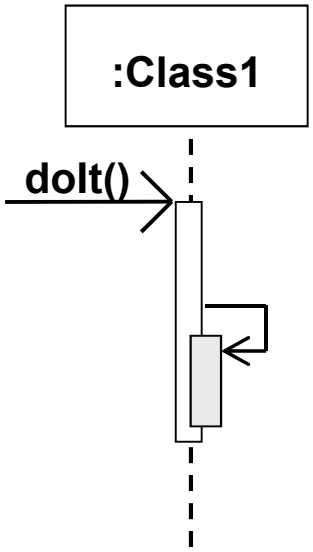
Objekte erzeugen und löschen

- Wird ein Objekt im Laufe der Ausführung erst erzeugt, dann zeigt eine Botschaft (Pfeil) auf dessen Objektsymbol.
- Das Löschen eines Objekts wird durch ein großes **X** dargestellt.



Aktivierungsbalken (*procedure call*)

- geben an, welches Objekt gerade aktiv ist. I.d.R. stellen sie die zeitliche Aktivierung einer Operation einer Klasse (bzw. deren Objekte) dar.
- werden durch die Aktivierung einer Operation erzeugt und durch deren Verlassen beendet.

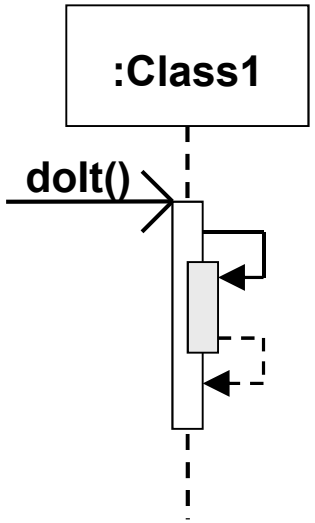


- werden durch schmale Rechtecke dargestellt, die über die Lebenslinie gelegt werden.
- können geschachtelt (überlagert) werden bei Rekursionen oder bedingter Aktivierung (s.u.)

Rekursionen

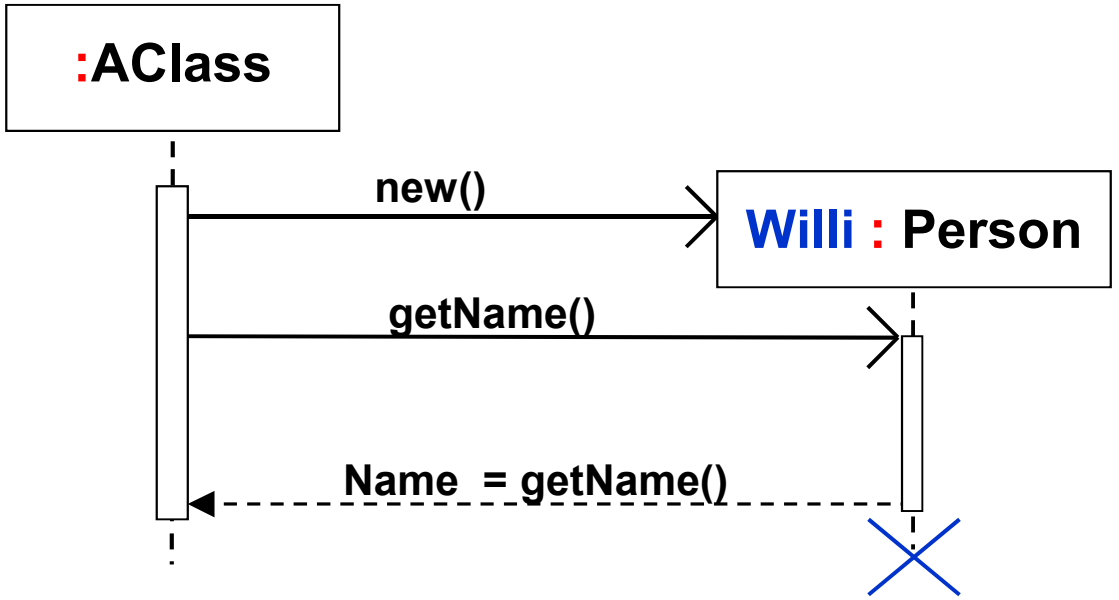
Rekursionen

- werden durch Pfeile auf die eigene Instanz einer Klasse und optional durch Überlagerung zweier Aktivierungsbalken dargestellt.



Botschaften (Messages, Nachrichten)

- in das Sequenzdiagramm werden Botschaften (Pfeile) eingetragen. Sie dienen zum Aktivieren von Operationen.
- Jede Botschaft wird vom Sender zum Empfänger gezeichnet.
- Der Pfeil wird mit dem Namen der aktivierten Operation beschriftet.



Botschaften (Typen)

- Es gibt unterschiedliche Botschaftstypen:

1. **synchron**, d.h. die Botschaft wird als Prozedur-aufruf interpretiert. Der Sender der Botschaft wartet auf die Antwort des Empfängers

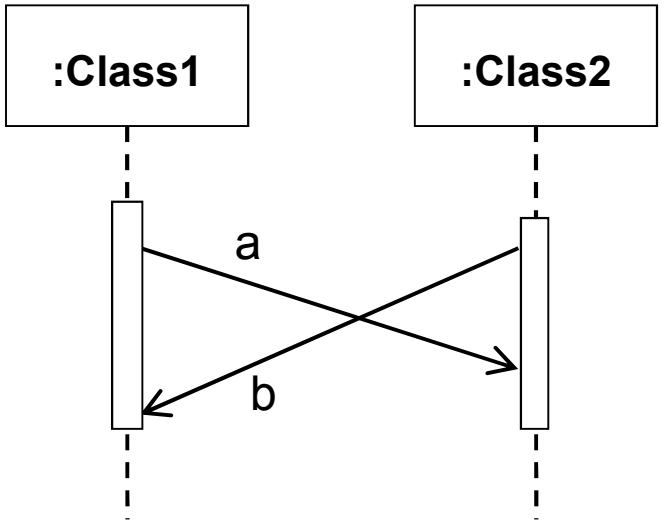
2. **asynchron**, d.h. die Botschaft wird als Signal betrachtet, und der Sender der Botschaft wartet **nicht** auf die Antwort des Empfängers

3. **Rückkehr** zum sendenden Element (optional bei **un**verzweigten Sequenzen). Keine *neue* Nachricht im eigentlichen Sinn!

- Jede Botschaft **muss** benannt werden!

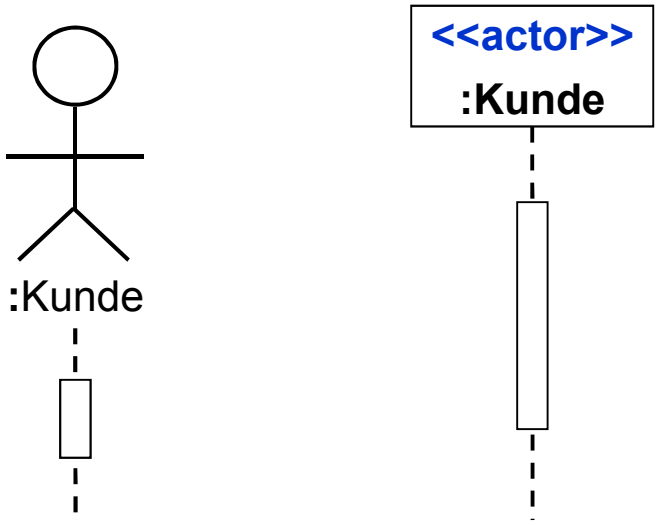
Asynchrone Botschaften

- Botschaftspfeile können sich kreuzen, wenn die Botschaften zu unterschiedlichen Zeiten gesendet bzw. empfangen werden



Akteure

- wie im Use-Case-Diagramm können Akteure verwendet werden (müssen nicht!)
- sie stehen für Außenstehende Personen, Objekte oder Systeme, welche sich nicht im eigenen Systemeinflussbereich befinden
- mit Akteuren lassen sich nur beispielhafte Idealabläufe darstellen
- sie stehen am linken Rand der Diagramme
- Namen der Akteure sollten konsistent zu den Use-Case-Diagrammen sein



Bis hierher …

- Einfache Modellierung eines Sequenzdiagramms (UML-1.4-Level)

Vor der einfachen Übungsaufgabe (übernächste Folie):

Unterscheidung zwischen Analyse und Entwurf bei der Modellierung eines Sequenzdiagramms:

Analyse:

- **Nachrichten**: freie Prosa möglich
- Use Cases können durch Nachrichten abgebildet (verwendet) werden

Entwurf:

- **Nachrichten**: nur die in den Klassen definierten Methoden verwenden
- fehlende Methoden der Klassen können leichter modelliert werden, wenn sie bei der SD-Modellierung noch nicht existieren

Übung

Erzeugen Sie

a) ein Szenario

b) ein Sequenzdiagramm

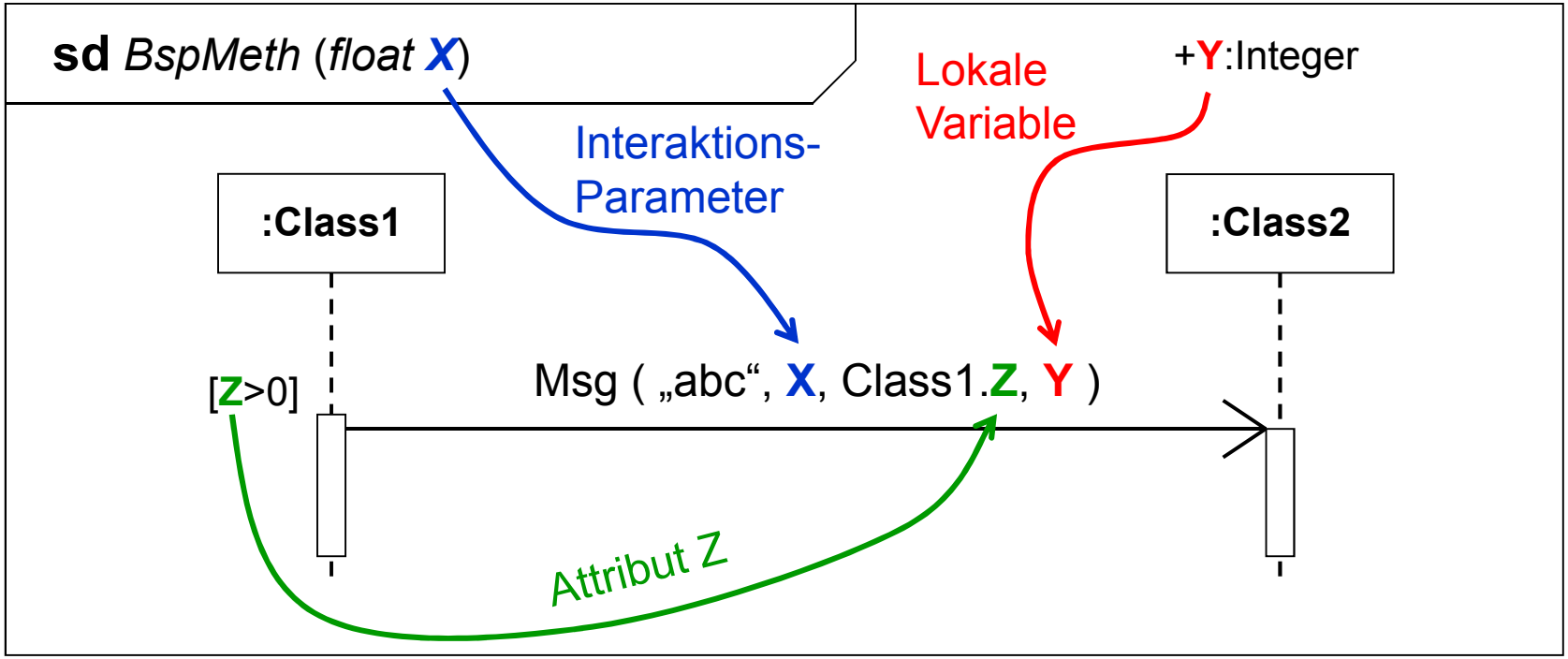
für den Use-Case *Film anlegen* aus dem Semesterbeispiel „Filmverwaltung“

➔ Ohne Verzweigungen (if … then) , d.h. wir gehen davon aus, dass die Referenzen (Darsteller, Regisseure, etc.) bereits vorhanden sind.

Botschaften (Benennung)

Benennung von Botschaften

- Einfacher Name (ohne Klammern).
- Operationsaufrufe mit Argumenten



Botschaften (Benennung (2))

Benennung von Botschaften (formal)

Gegeben: modellierte Operation „*findeNamen*“ einer Telefonbuchanwendung

```
Boolean findeNamen( in TelNummer : String, out Name : String )
```

➔ Mögliche Benennung im Sequenzdiagramm:

findeNamen(varNummer, -)

findeNamen(„+497218353654“, -)

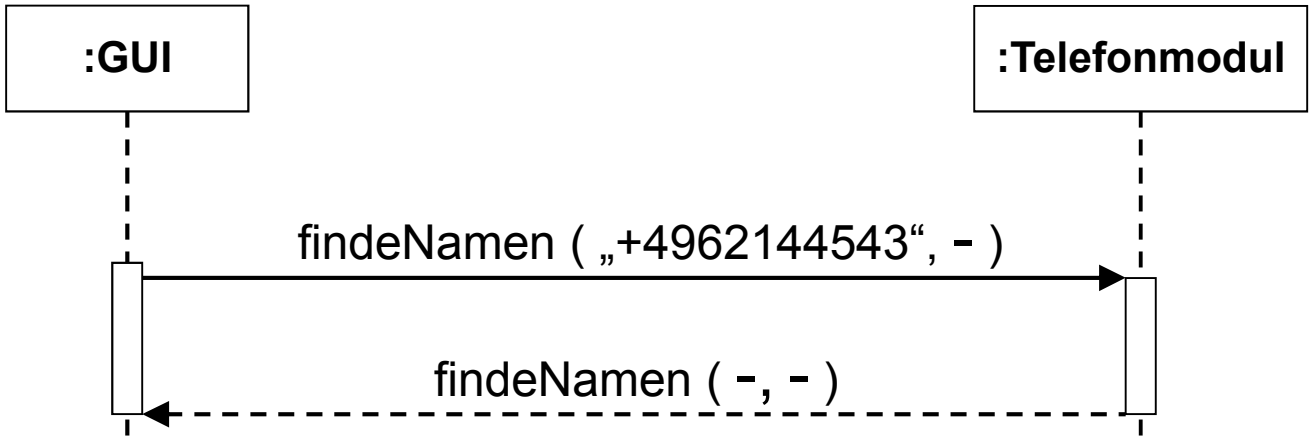
findeNamen(TelNummer = „+497218353654“, -)

findeNamen(TelNummer = varNummer, -)

Botschaften (Benennung (3))

Benennung von einfachen Rückkehr-Botschaften (Antworten)

- „Wiederholung“ des aufrufenden Parametersatzes
- Genügt der Name der Operation, werden Argumente anonymisiert („-“)



Botschaften (Benennung (4))

Benennung von Rückkehr-Botschaften (formal)

Botschaft: **findeNamen(„+497218353654“, -)**

➔ Mögliche Benennung der Rückkehr-Botschaft im Sequenzdiagramm:

- Einfache Antwort, wenn der Name der Botschaft genügt:

  findeNamen (- , -)

- Vollständig für alle OUT-Parameter:

  RET = findeNamen (- , Name : „Schröder“) : True

- Zuweisung der OUT-Parameter zu lokalem Attribut:

  RET = findeNamen (- , *LocalAttribut* = Name : „Schröder“) : True

Zusätzliche Beschriftungen

Sequenzdiagramme können an ihrem (i.a. linken) Rand oder an den Lebens-linien mit zusätzlichen textuellen Angaben ausgestattet sein, die sich auf Botschaften bzw. Aktivierungen beziehen:

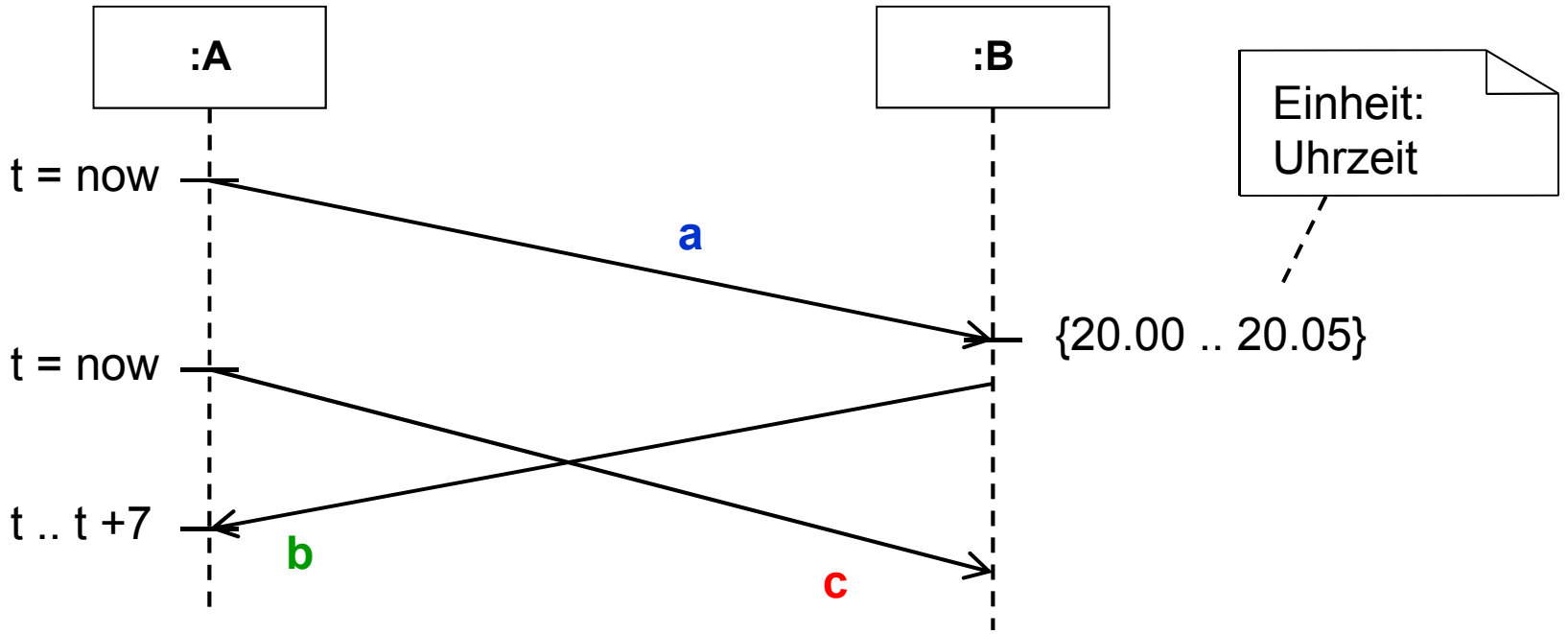
- **Einschränkungen** (logische Ausdrücke in geschweiften Klammern).
  ➔ meist für Sende- bzw. Empfangszeit (max. Übermittlungsdauer)

  Beispiel: {*n.receiveTime – n.sendTime < 1 sec*} (n = Botschafts-Nr.)

- **Ausführende Aktionen**:
  ➔ zur ausführlicheren Beschreibung einer ausgelösten Aktivierung.
  ➔ natürlichsprachlich oder Pseudocode

- **Zeitangaben** (s. nächste Folie)

Zusätzliche Beschriftungen (Zeitangaben)



Ereignis **a** muss genau zwischen 20.00 und 20.05 Uhr eintreffen

Empfangsereignis **b** muss maximal 7 ms nach Absenden von **c** eintreffen.

Zusätzliche Beschriftungen (Zeitangaben)

Beliebige Zeitpunkte:

12.34, 4800, 15, 17.04.2008, „alle 5 Min.“ (Einheiten als Notiz!)

Attribut = **now** (aktuelle Zeit, muss einem Attribut zugewiesen werden)

Zeitintervalle:

{ 20.15 .. 20.55 } Menge aller Zeitpunkte zwischen 20.15 und 20.55

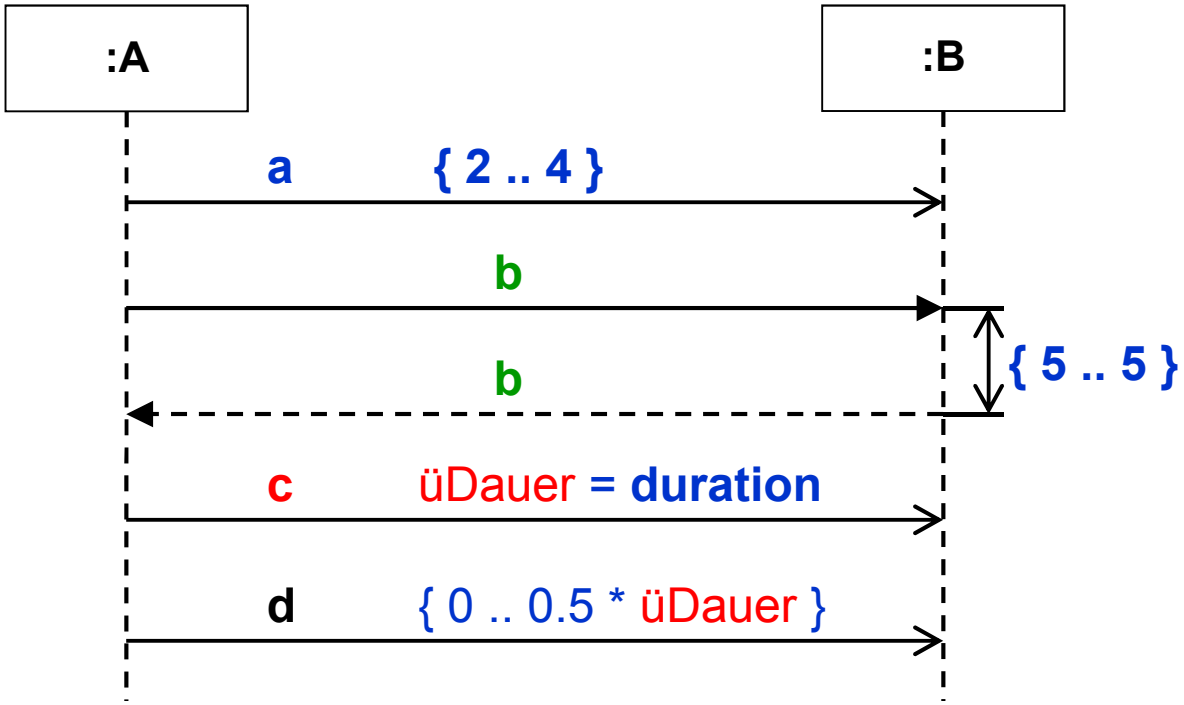
{ t .. t+7 } Menge aller Zeitpunkte zwischen t und t+7 Zeiteinheiten

{ Mo .. So } Menge aller Wochentage

Darstellung im Sequenzdiagramm:

Prinzipiell an beliebiger Stelle an Pfeilspitze oder -fuß mit kleiner Hilfslinie

Zusätzliche Beschriftungen (Zeitangaben)



- Das Senden von **a** darf minimal 2 und maximal 4 Zeiteinheiten dauern
- Operation **b** muss in genau 5 Zeiteinheiten ausgeführt werden
- Übertragung von **d** darf max. halb so lange dauern wie **c**

Zusätzliche Beschriftungen (Zeitangaben)

Zeitdauern:

{15 .. 55 } Menge aller Zeitdauern zwischen 15 und 55

{ t .. t*2 } Menge aller Zeitdauern zwischen t und t*2

{ Mo .. Mo } Dauer eines Tages (Mo)

Darstellung im Sequenzdiagramm:

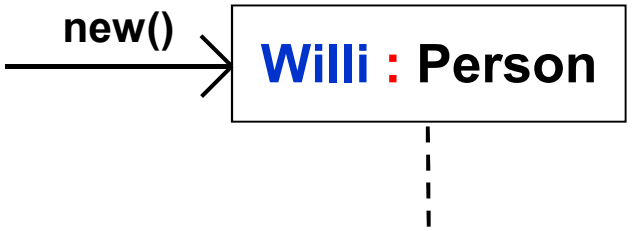
- Prinzipiell an beliebiger Stelle an Pfeilspitze oder –fuß mit kleiner Hilfslinie (mit senkrechtem Doppelpfeil)
- Hinter die Benennung einer Botschaft

Zeitdauerbeobachtungsaktion:

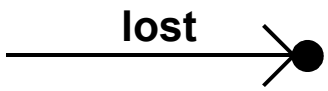
Zuweisung an Attribut: *Attribut* = **duration**

Spezielle Botschaften

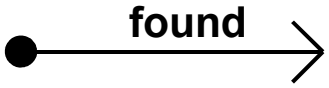
- Erzeugungsbotschaften



- Verlorene (*lost*) Botschaften
  (Empfänger wird nicht modelliert,
  da momentan nicht bekannt)

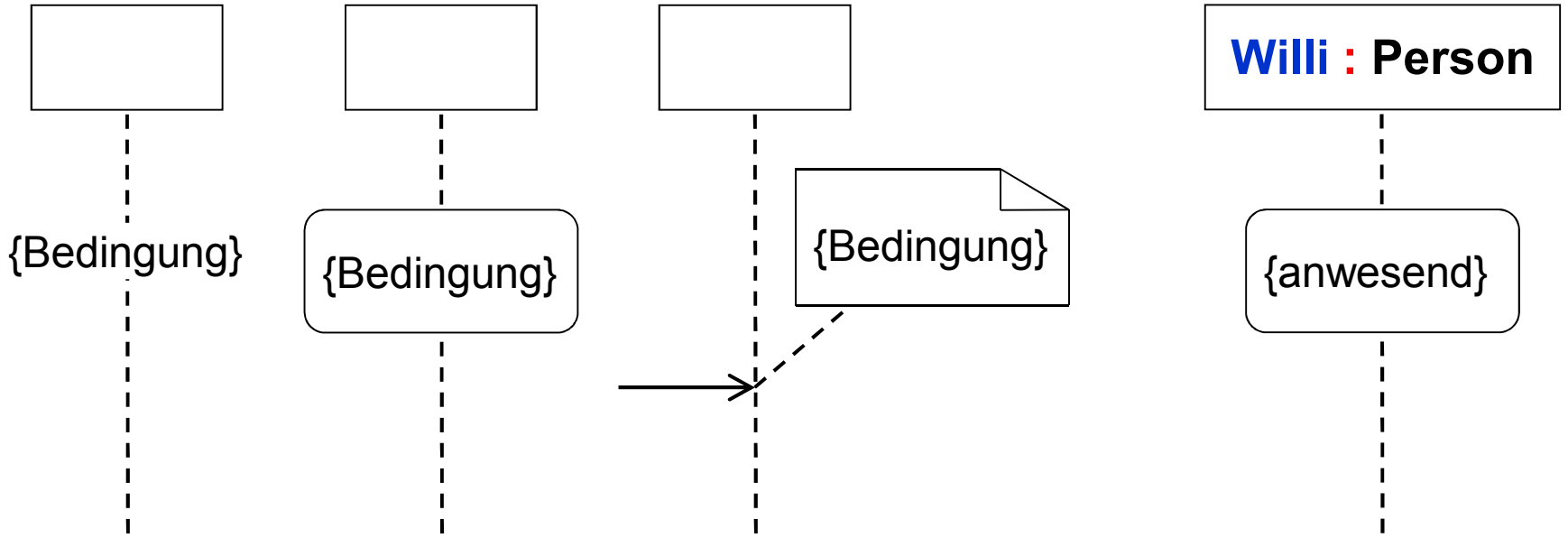


- Gefundene Botschaften
  (Sender ist unbekannt)



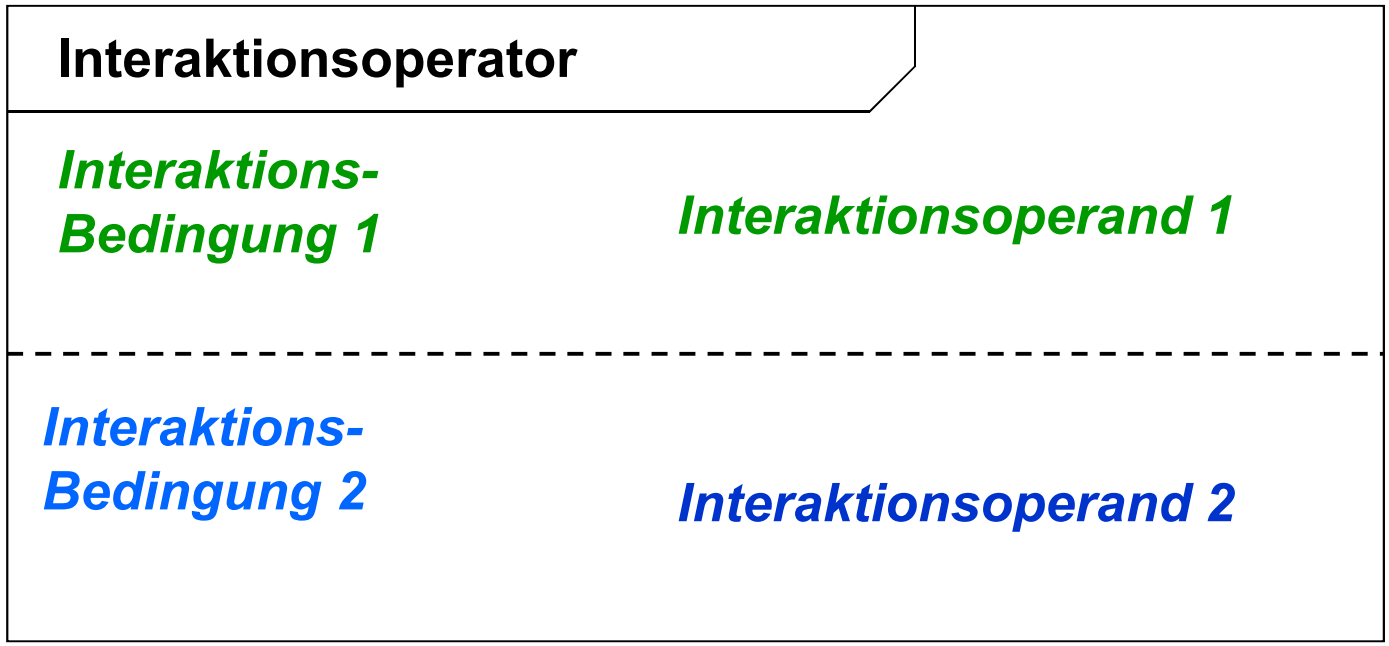
Zustandsinvarianten

- Lebenslinien können mit Zustandsinvarianten versehen werden
- Zustandsinvarianten sind Gültigkeitsbedingungen für eine Interaktion
- Darstellung im Sequenzdiagramm:



Kombinierte Fragmente

- Kennzeichnung eines Teils einer Interaktion, für den best. Regeln gelten



Interaktionsoperatoren

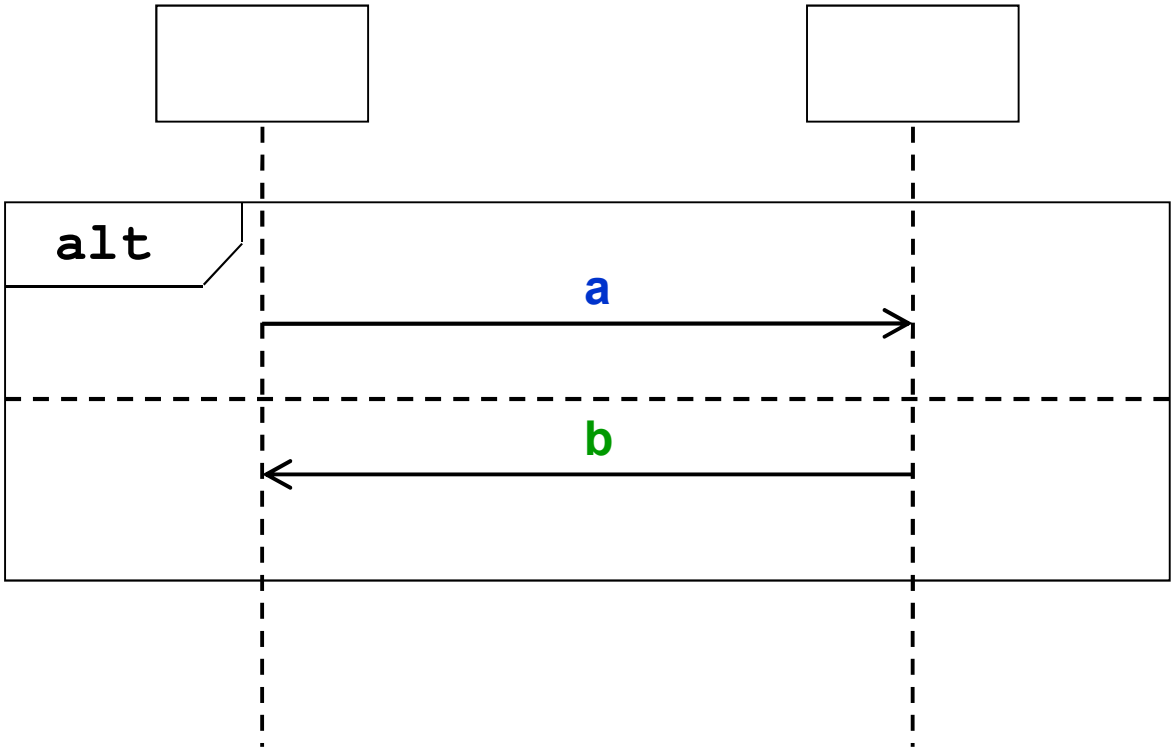
Typ	Bezeichnung	Beschreibung
`alt`	Alternative	Alternative Ablaufmöglichkeiten
`opt`	Option	Optionale Interaktionsteile
`break`	Break	Interaktionen in Ausnahmefällen
`neg`	Negation	Ungültige Interaktionen
`loop`	Loop	Iterative Interaktionen
`par`	Parallel	Nebenläufige Interaktionen

Interaktionsoperatoren

Typ	Bezeichnung	Beschreibung
`seq`	Weak Sequencing	Von Lebenslinien und Operanden abhängige chronologische Abläufe
`strict`	Strict Sequencing	Von Lebenslinien und Operanden unabhängige chronologische Abläufe
`critical`	Critical Region	Atomare Interaktionen
`ignore`	Ignore, Irrelevanz	Filter für unwichtige Nachrichten
`consider`	Consider, Relevanz	Filter für wichtige Nachrichten
`assert`	Assertion, Sicherstellung	Nebenläufige Interaktionen

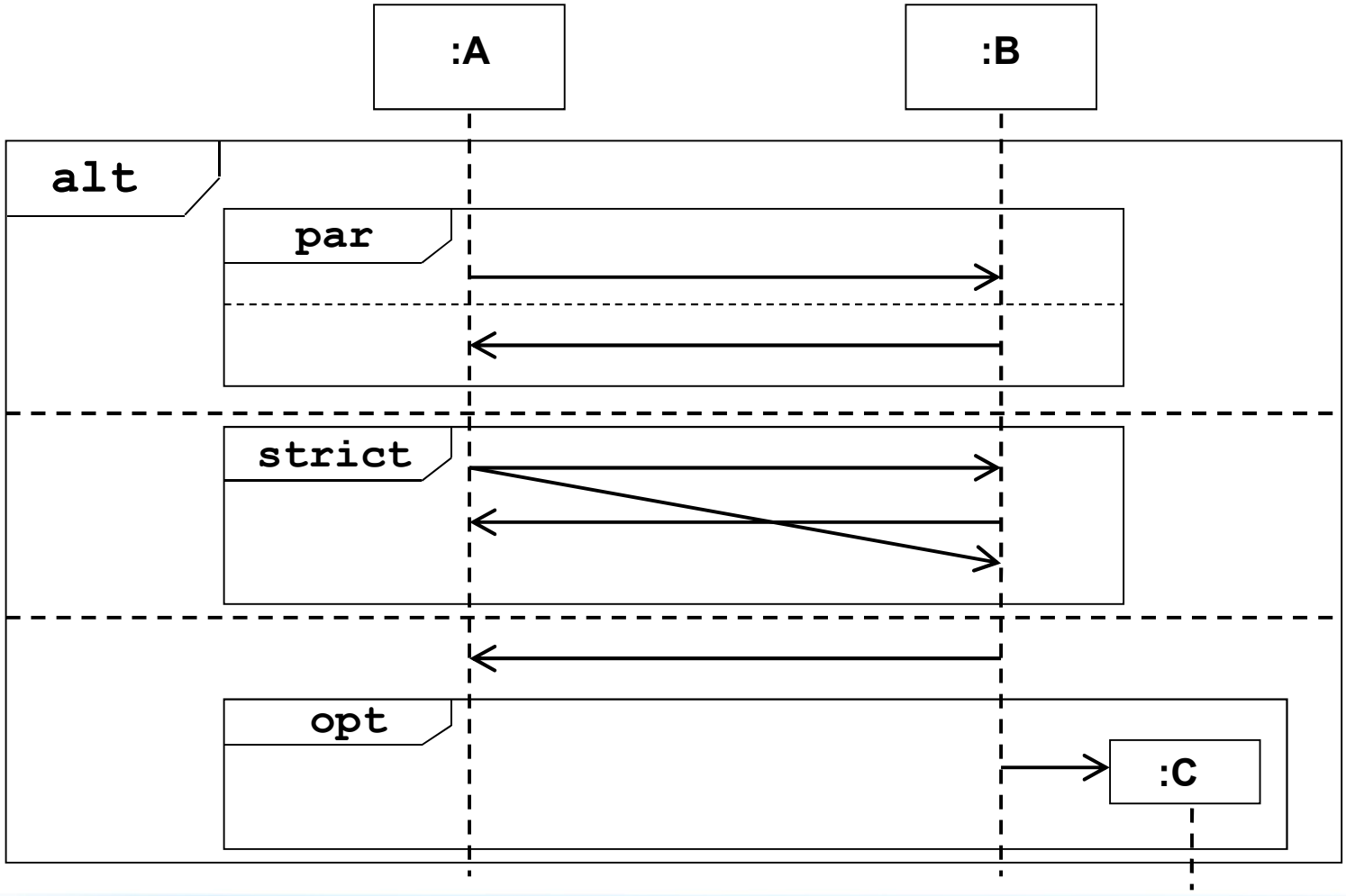
Positionierung von Fragmenten

- Über die Lebenslinien aller betroffenen Klassen



Schachtelung von Fragmenten

- Fragmente können beliebig tief geschachtelt werden!

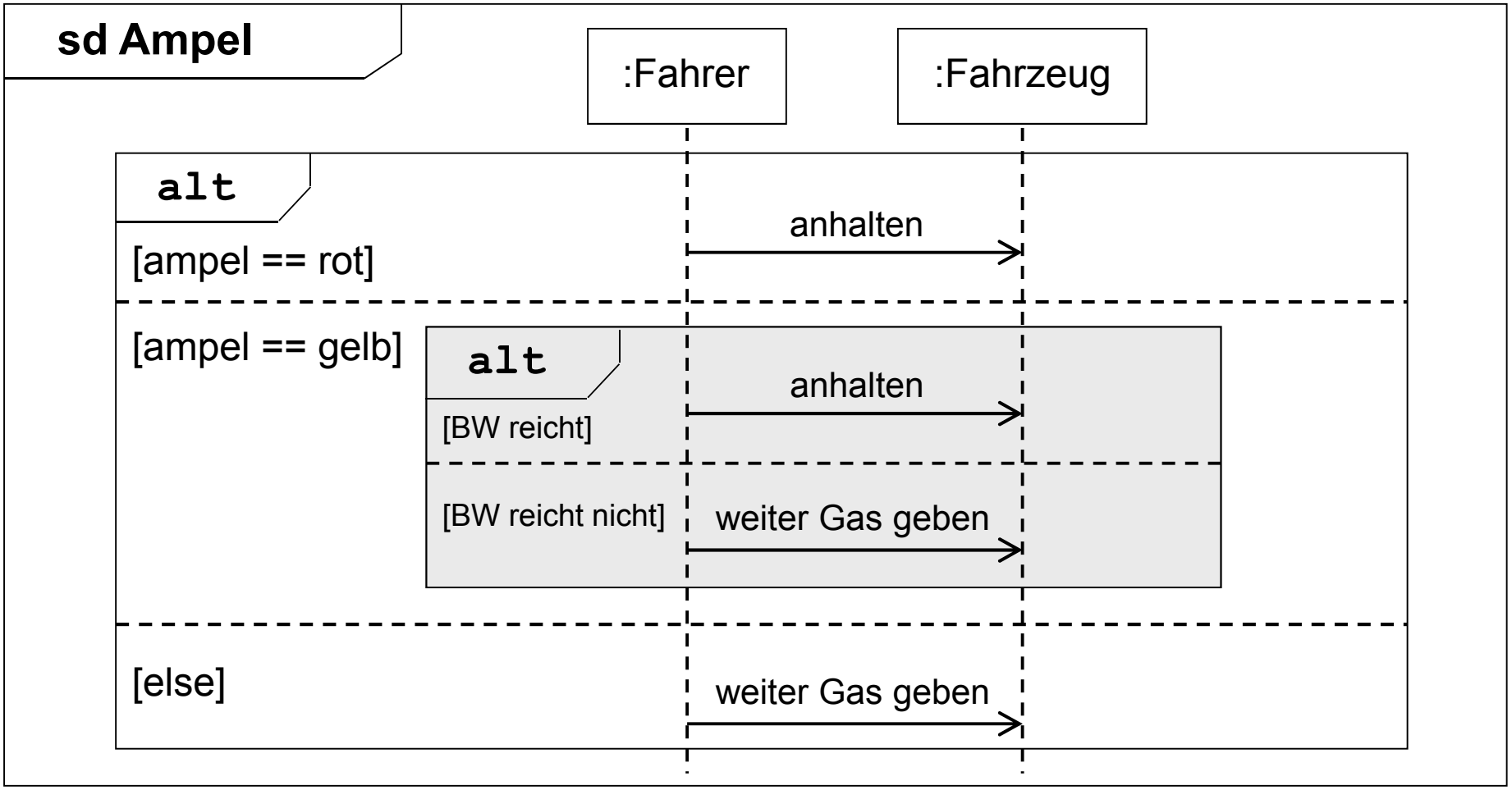


Ausgewählte Fragmente:

- **Alternativ-Fragment**
- **Schleife**
- **Parallel-Fragment**

Alternativ-Fragment (`alt`)

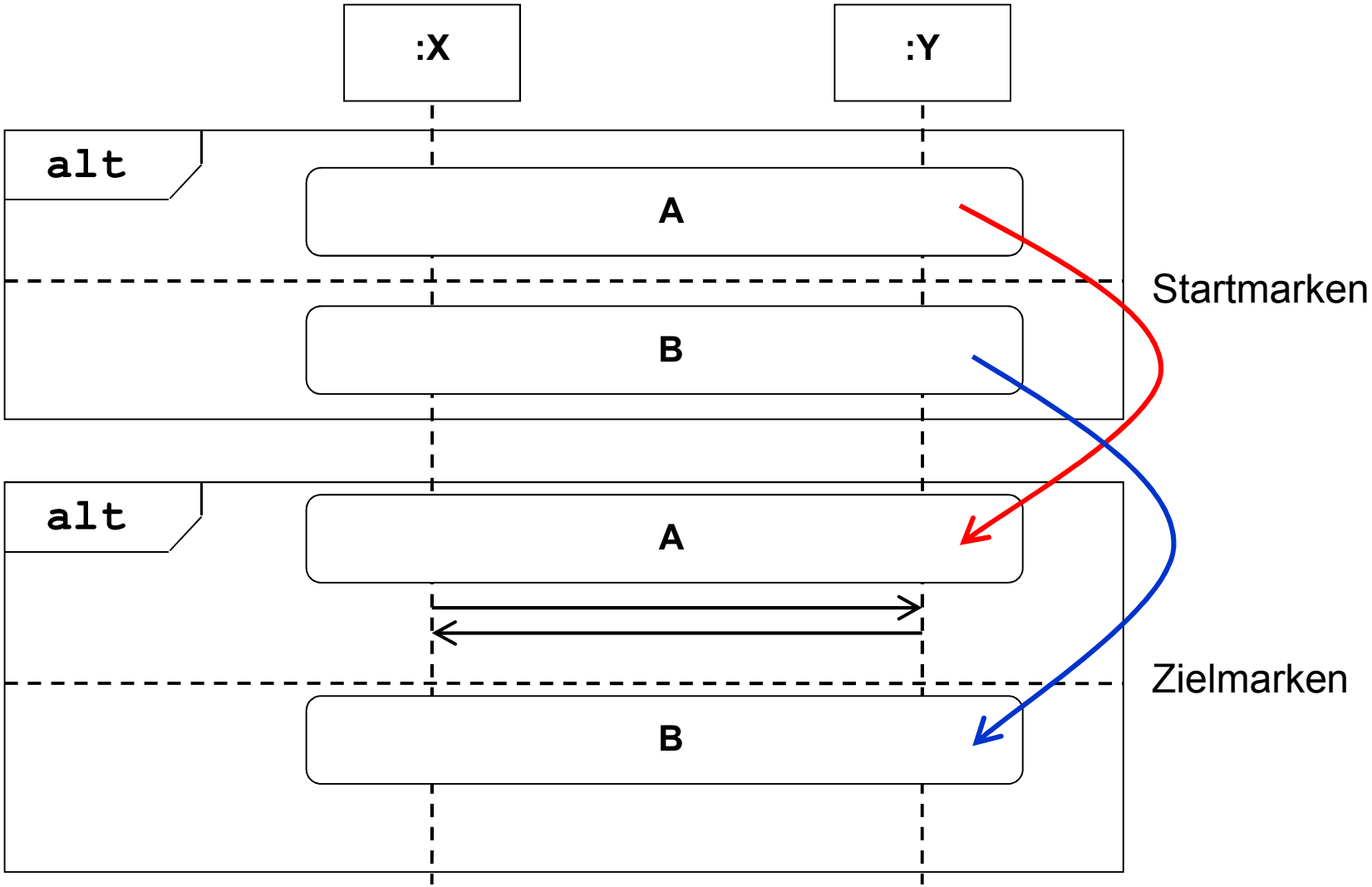
Entspricht `switch/case`-Anweisung



Alternativ-Fragment: Sprungmarken

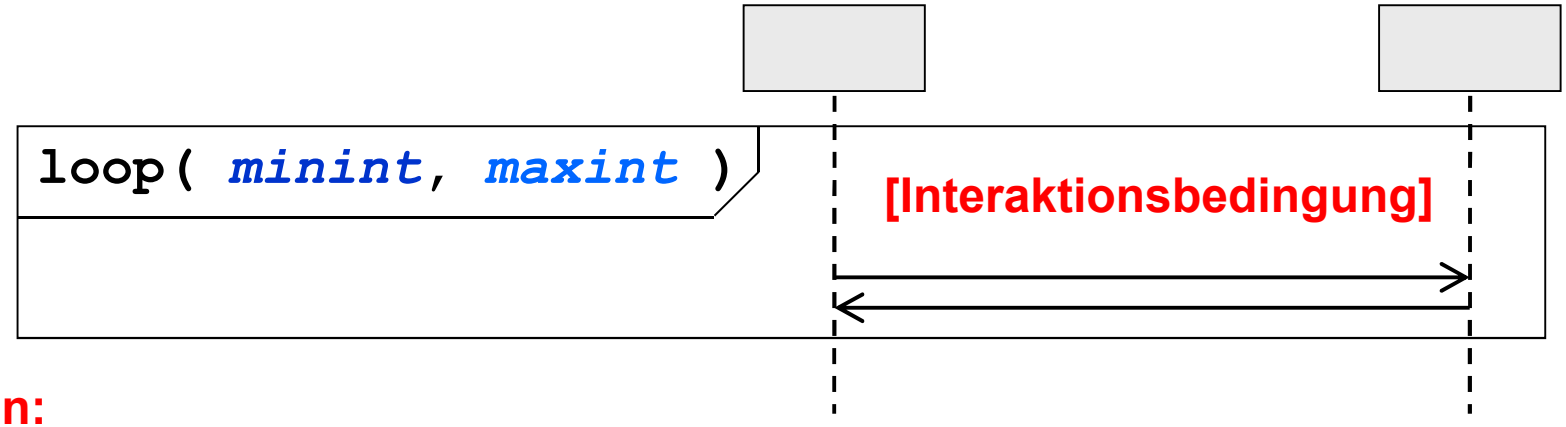
- Zur Fortsetzung eines **`alt`**-Operanden an einer beliebigen Stelle innerhalb der Interaktion ➔ auch Fortsetzungsmarken genannt
- Symbol: wie Zustände, jedoch über mehrere Lebenslinien möglich
- Sprungmarken treten immer paarweise auf
- Start- und Zielmarken tragen denselben Namen
- Start- und Zielmarken müssen dieselbe Lebenslinie abdecken

Alternativ-Fragment: Sprungmarken



Schleife (`loop`)

Kennzeichnet wiederholt ablaufende Bereiche innerhalb einer Interaktion

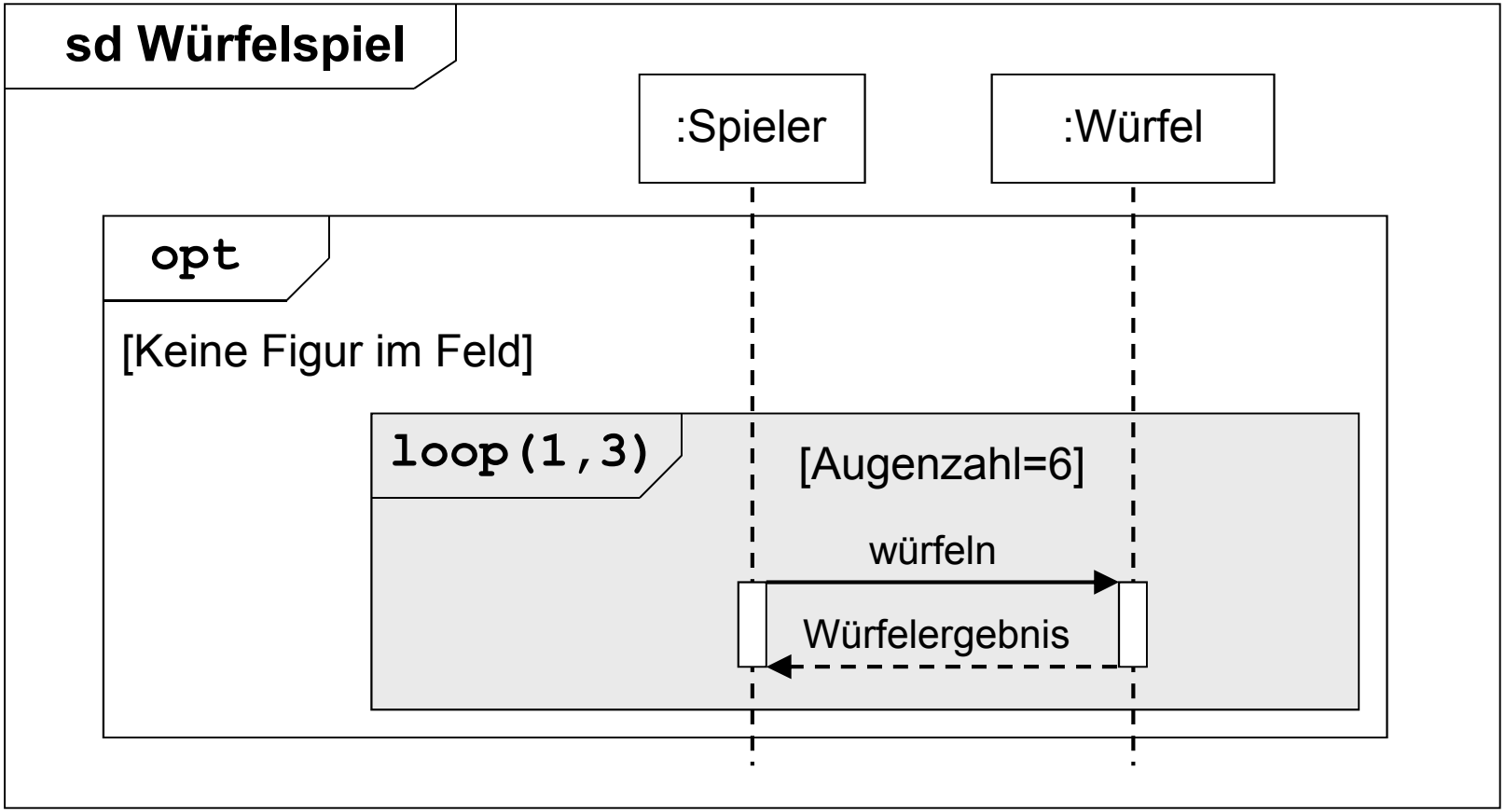


Regeln:

- ***minint***: Mindestzahl der Wiederholungen (ganzzahliger Wert)
- ***maxint***: Höchstzahl der Wiederholungen (ganzzahliger Wert)
- „*“ definiert ***maxint*** als unendlich
- Wird nur ***minint*** angegeben, wird die Schleife exakt ***minint*** mal durchlaufen
- Endlosschleife: `loop` ohne Angabe von ***minint*** und ***maxint***
- ***minint*** und ***maxint*** sowie die Interaktionsbedingung können Attribute und lokale Variablen enthalten (loop(i), loop(x=1,x<10), [x!=0])

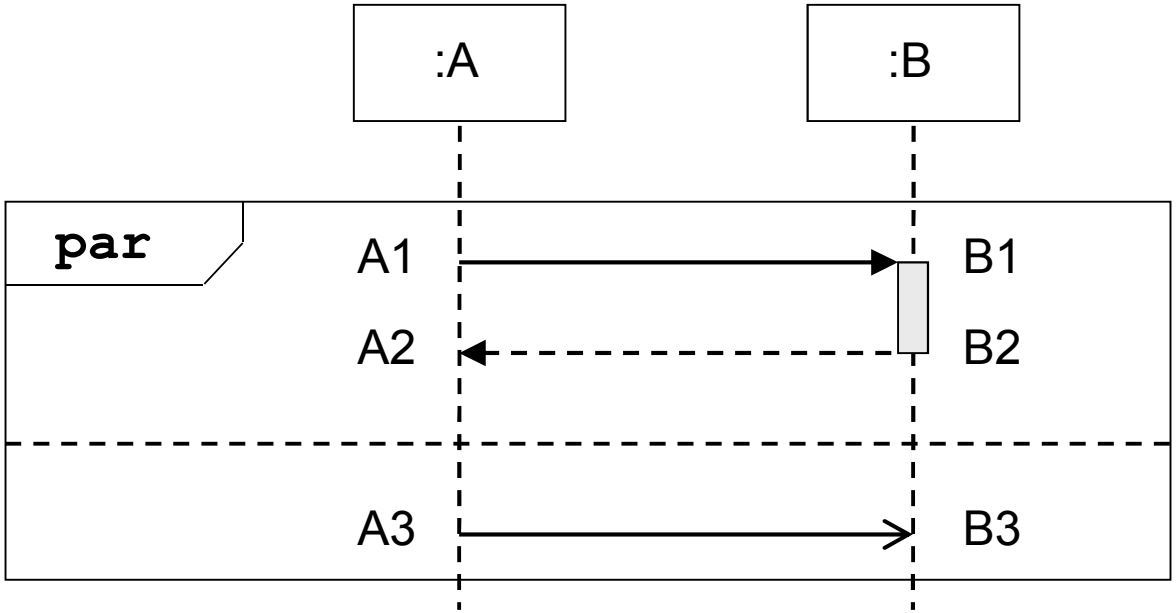
Beispiel für eine Schleife (loop)

Würfeln bei einem Würfelspiel



Parallel-Fragment (`par`)

Definition von Teilen einer Interaktion zur Ausführung in beliebiger Reihenfolge



Feste Vorgabe (ohne *par*):

A1 ➔ B1 ➔ B2 ➔ A2
A3 ➔ B3

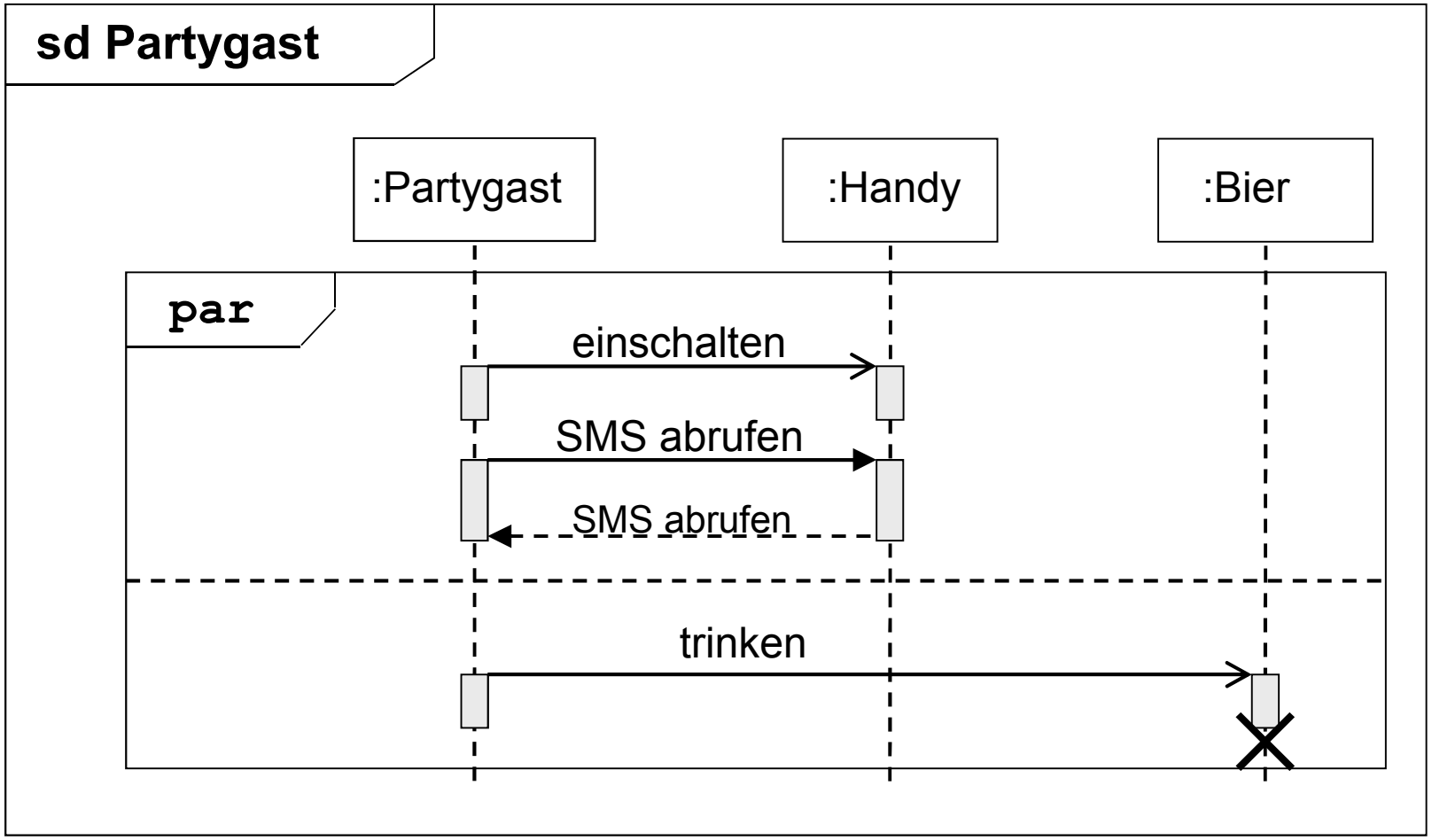
Mit *par*:

A3 ➔ A1 ➔ B3 ➔ B1 ➔ B2 ➔ A2 oder
A1 ➔ B1 ➔ B2 ➔ A3 ➔ B3 ➔ A2 oder
A1 ➔ B1 ➔ B2 ➔ A2 ➔ A3 ➔ A1 usw.

(A ➔ B : Ereignis B folgt zeitlich auf Ereignis A)

Beispiel für Parallel-Fragmente (`par`)

Definition von Teilen einer Interaktion zur Ausführung in beliebiger Reihenfolge



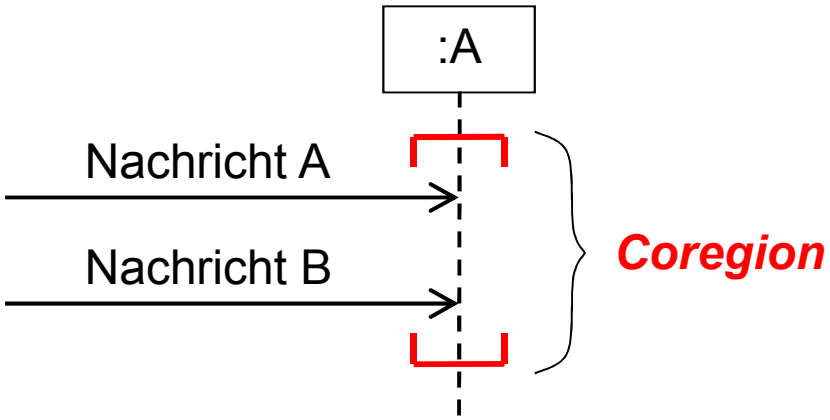
Parallel-Fragment (`par`)

Achtung: keine echte Parallelität!

d.h. kein Aussenden von mehreren Ereignissen *zum exakt gleichen* Zeitpunkt.

Stattdessen Erzeugung mehrerer möglicher Ablaufvarianten

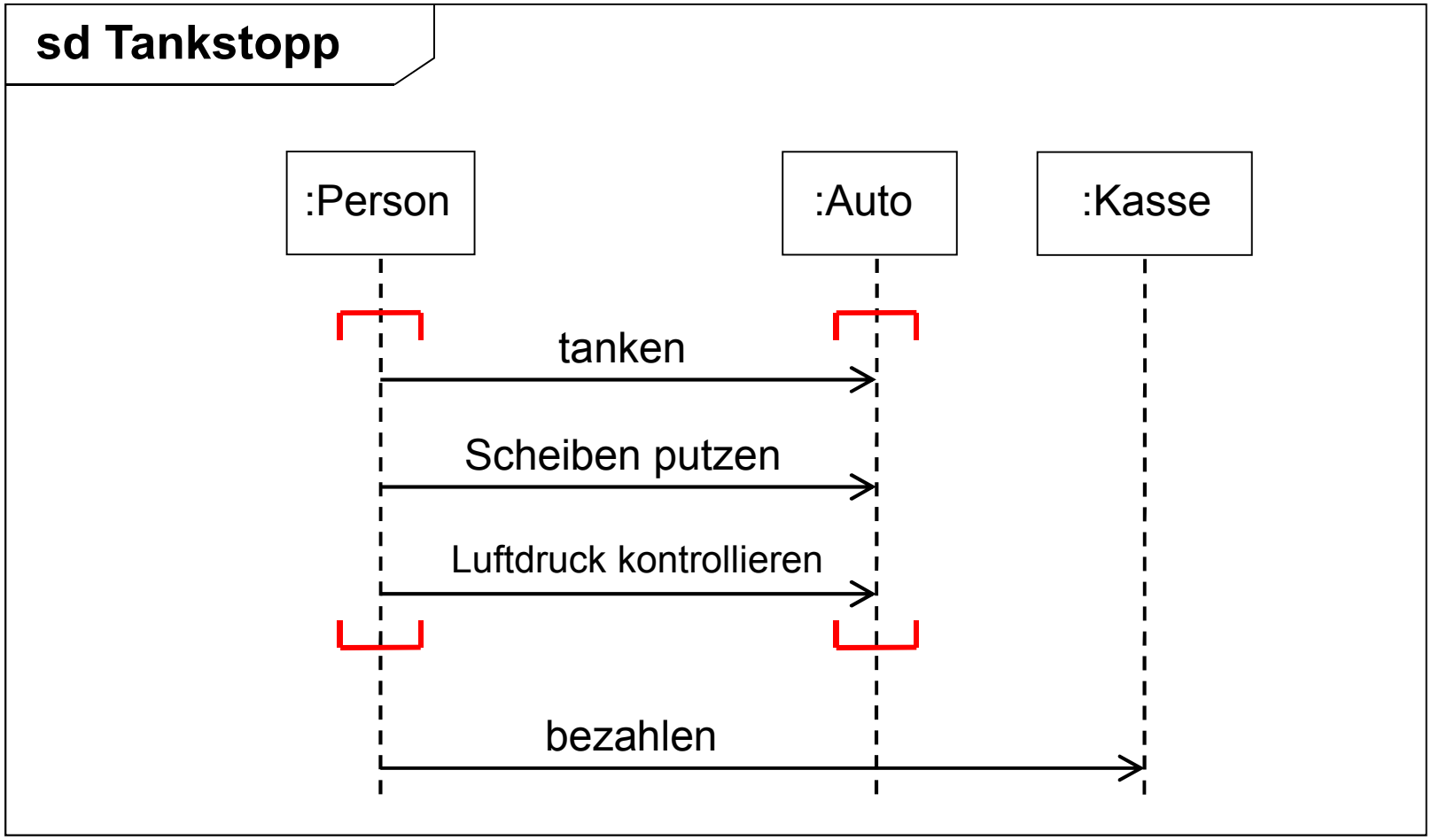
Alternative Darstellungsform für parallele Abläufe: ***Coregion***



Reihenfolge beliebig,
d.h. tauschbar

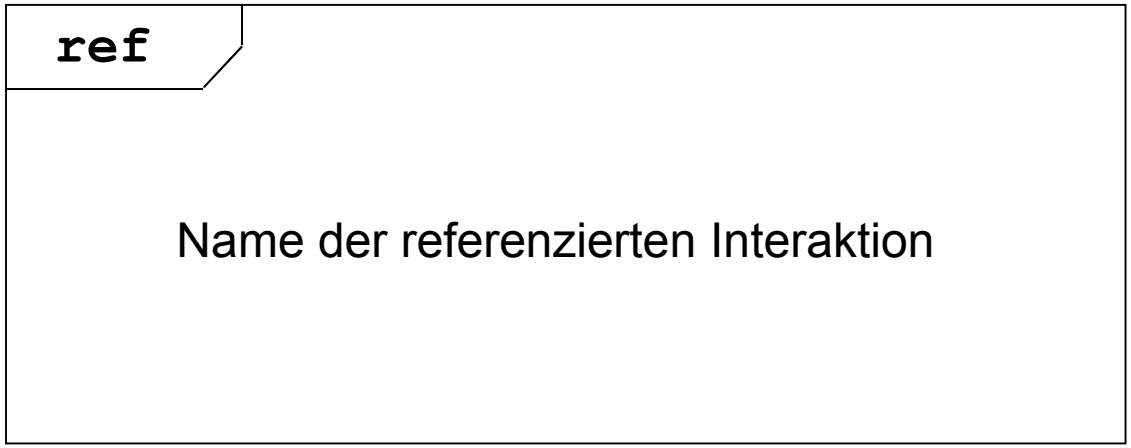
Beispiel für Parallele Abläufe (Coregion)

Definition von Teilen einer Interaktion zur Ausführung in beliebiger Reihenfolge

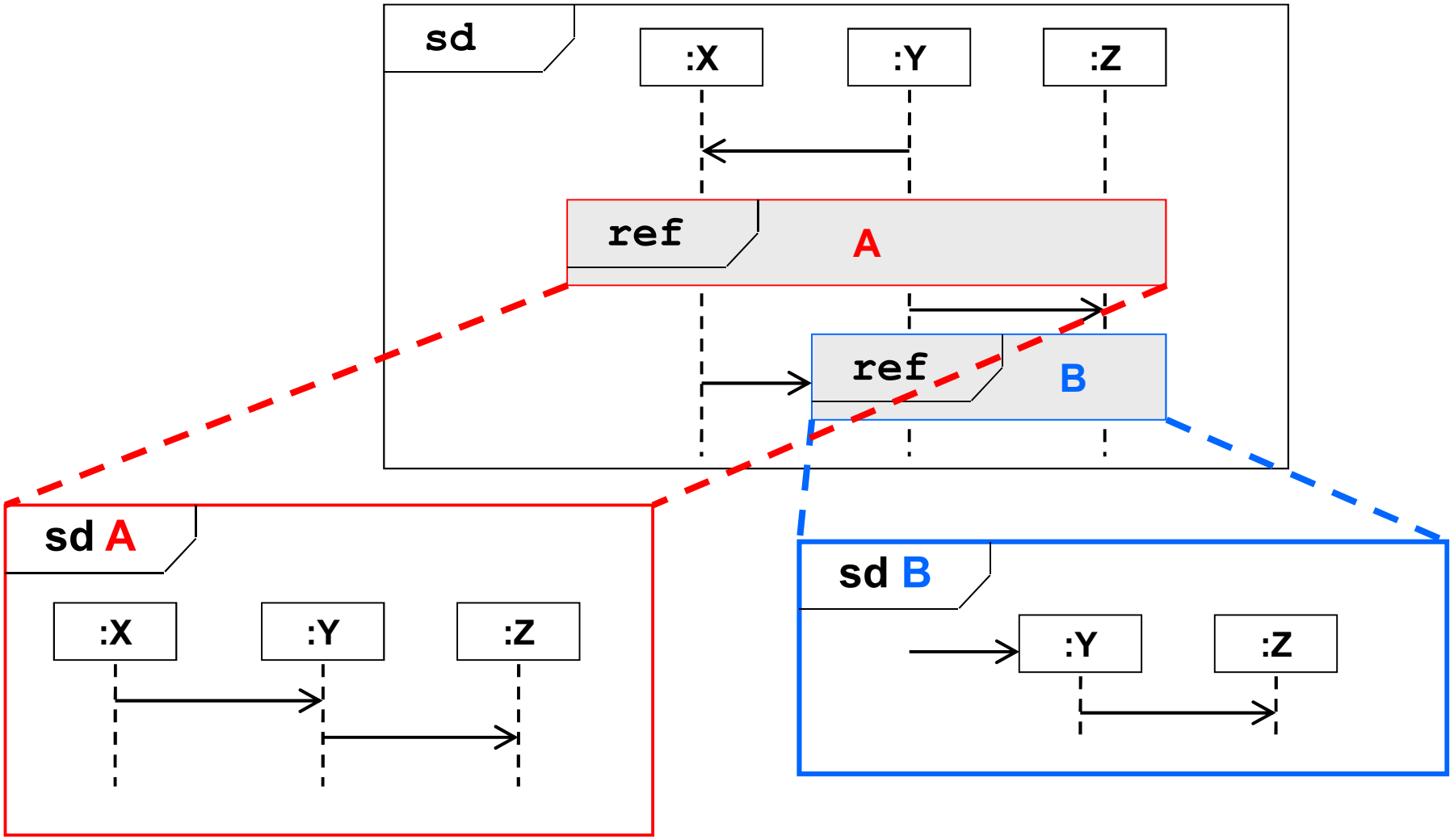


Verfeinerung durch Interaktionsreferenzen (`ref`)

Eine Interaktionsreferenz ist ein Bereich innerhalb einer Interaktion, der auf eine andere, ausgelagerte Interaktion referenziert.



Beispiel für Interaktionsreferenzen (ref)



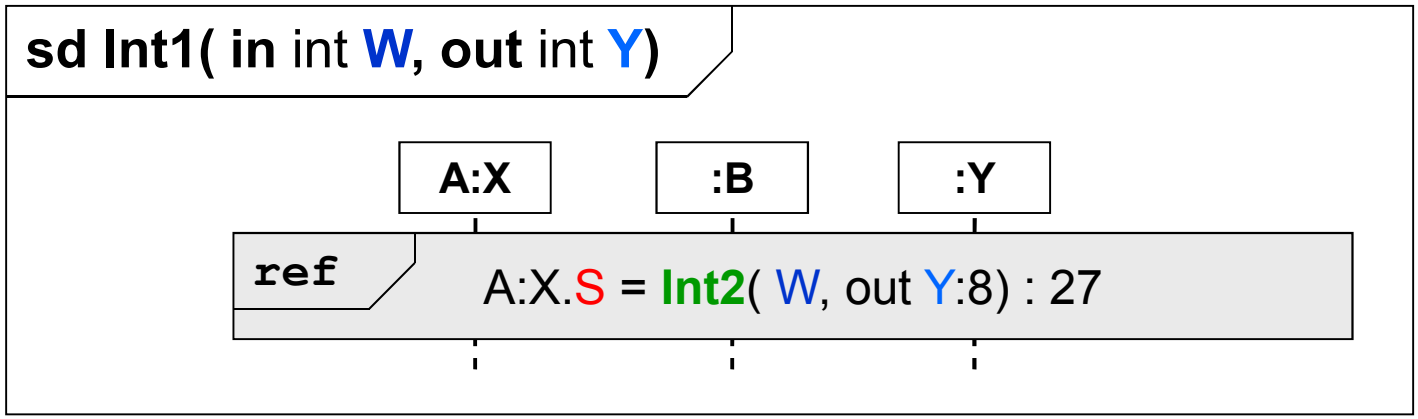
Regeln für Interaktionsreferenzen (`ref`)

- Interaktionsreferenzen übersichtlich gestalten
- Auslagern von weniger wichtigen Aspekten in separate Interaktionen
- Einmalige Definition von wiederkehrenden Interaktionen und Mehrfach-(Wieder-)verwendung an den entsprechenden Stellen
- Vergabe von verständlichen Referenznamen. Die Interaktion mit den Referenzen soll ohne Detailkenntnis der **`ref`**-Interaktionen lesbar und verstehbar sein.
- Nahtlose Substitution der Referenz durch das Referenzierte muss möglich sein.
- Die Verbindung wird über den Interaktionsnamen hergestellt:

Analog wie bei Botschaften:

Angabe von:
- Ein- und Ausgabeparametern (IN/OUT) und -argumenten(INOUT)
- Rückgabewerten und Attributzuweisungen

Semantik einer Interaktionsreferenz (ref)



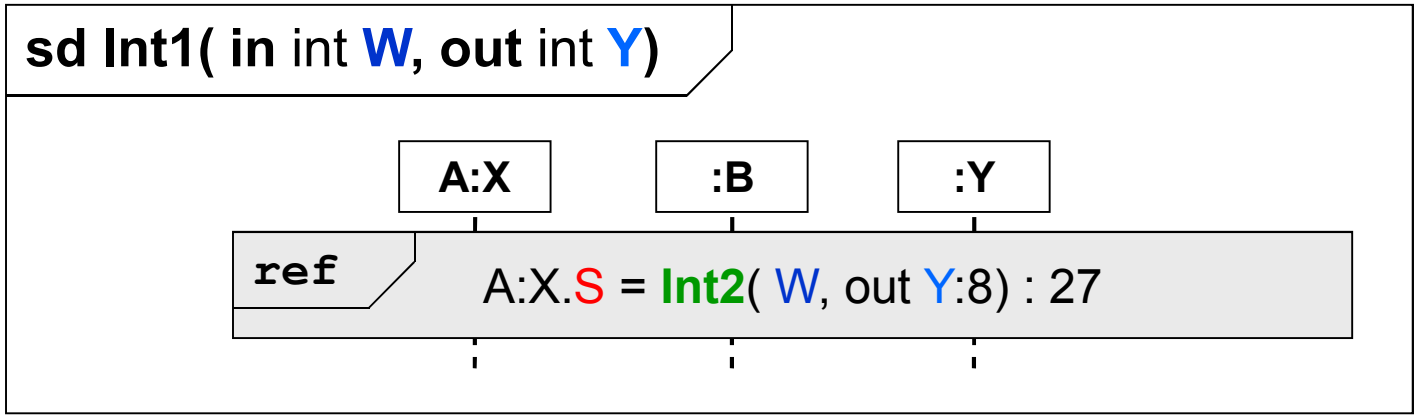
S = Attribut der Lebenslinie X, bekommt den Rückgabewert 27 zugewiesen

Int2 = Name der referenzierten Interaktion

W = Eingabeparameter

out Y:8 = Output-Parameter Y wird der Wert 8 zugewiesen

Semantik einer Interaktionsreferenz (ref)

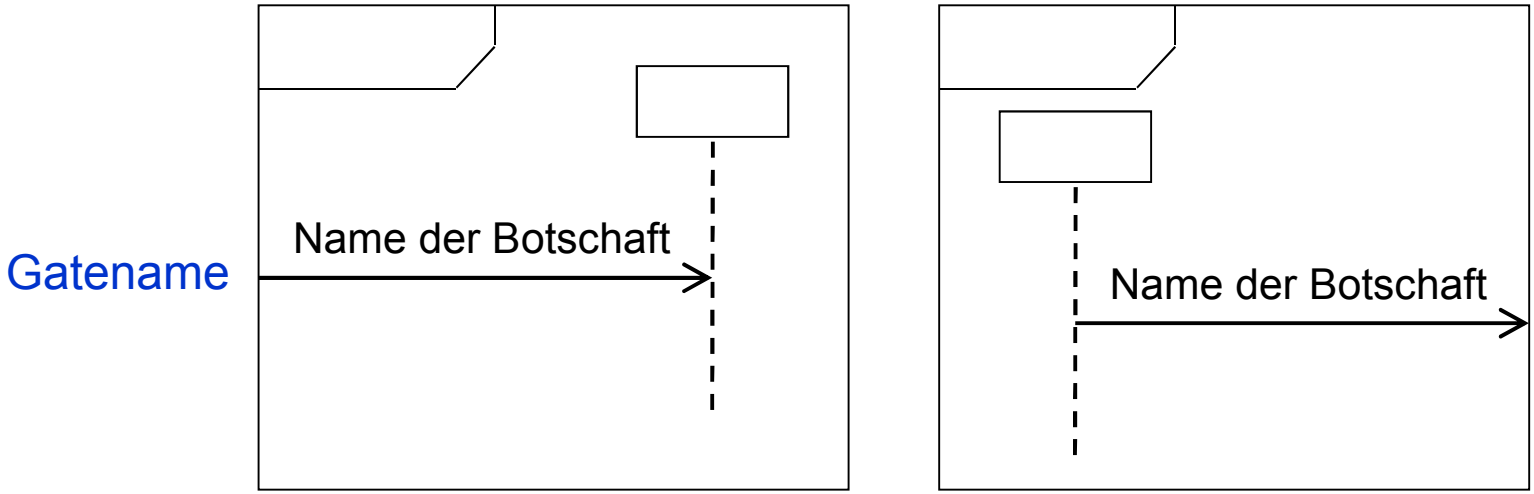


Regeln:

- Argumente der Interaktionsreferenz müssen den Parametern der referenzierten Interaktion entsprechen
- Attributname ist der Name eines in der Interaktion bekannten Lebenslinien-attributs oder einer lokalen Variablen (s. Botschaften)
- Eine Interaktionsreferenz muss mindestens diejenigen Lebenslinien überdecken, die auch in der referenzierten Interaktion überdeckt werden.
- Als Argumente dürfen die gleichen Elemente (Konstanten, Attribute, ...) wie bei Botschaften übergeben werden

Verknüpfungspunkte (Gates)

- *Gates* sind Punkte auf einem Interaktions- oder Fragmentrahmen, zu denen Botschaften hin- oder wegführen
- *Gates* können einen Namen besitzen (zur besseren Übersicht)
- *Gates* ermöglichen den Nachrichtenfluss über „Rahmengrenzen“ hinweg zwischen (referenzierten) Interaktionen, Fragmenten und deren Operanden



Literatur

- **UML 2 glasklar**
  Mario Jeckle, Chris Rupp, Jürgen Hahn, Barbara Zengler, Stefan Queins
  Hanser Verlag München Wien, 2004

- **UML 2.0 in a Nutshell**
  Dan Pilone, Neil Pitman
  O'Reilly Verlag, 2006