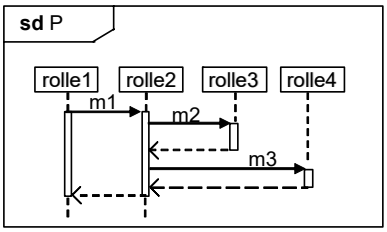
9 Sequenzdiagramm

Inhalt

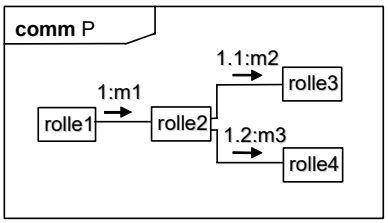
9.1 Exkurs Interaktionsdiagramme

- 4 Arten von Interaktionsdiagrammen
  - Für einfache Interaktionen **semantisch äquivalent**
  - Betonung **unterschiedlicher Aspekte**

- **Sequenzdiagramm** zeigt den zeitlichen und logischen Nachrichtenfluss
  - Reihenfolge von Nachrichten grafisch ersichtlich
  - Zeit ist **eigene** Dimension

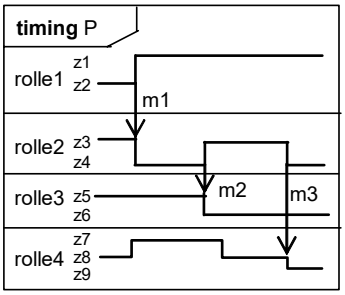


- **Kommunikationsdiagramm** ist »strukturell« orientiert
  - Zeigt die Beziehungen zwischen Interaktionspartnern – Kontextaspekt
  - Reihenfolge von Nachrichten nur über Dezimalklassifikation ausgedrückt
  - Zeit ist **keine eigene** Dimension

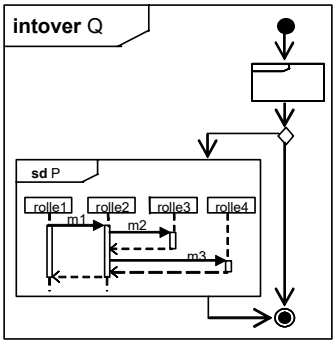


Exkurs Interaktionsdiagramme

- **Zeitdiagramm** zeigt Zustandsänderungender Interaktionspartner aufgrund von Zeitereignissen
  - Vertikale Dimension repräsentiert Interaktionspartner und ihre möglichen Zustände
  - Horizontale Dimension repräsentiert die Zeitachse



- **Interaktionsübersichtsdiagramm** zeigt das Zusammenspiel von verschiedenen Interaktionen
  - Visualisiert in welcher Reihenfolgeund unter welchen Bedingungen Interaktionsabläufe stattfinden



Exkurs Interaktionsdiagramme

- Zeigen wie **Nachrichten** zwischen verschiedenen **Interaktionspartnern** in einem bestimmten Kontext ausgetauscht werden

- Einsatzbereiche
  - Modellierung der **Interaktionen eines Systems mit seiner Umwelt** (Systemgrenzen festlegen, System als Black-Box)
  - Modellierung der **Realisierung eines Anwendungsfalls**
  - Modellierung des **Zusammenspiels der internen Struktur** einer Klasse, Komponente oder Kollaboration
  - Modellierung der Spezifikation von **Schnittstellen zwischen Systemteilen** (Zusammenspiel angebotene/benutzte Schnittstelle)
  - Modellierung der **Operationen einer Klasse**

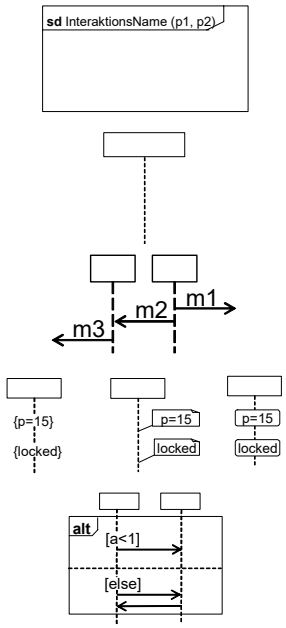
Exkurs Interaktionsdiagramme: Typ- vs. Instanzebene

- **Modellierung** des Nachrichtenaustauschs zwischen **Rollen** und damit prinzipiell auf **Rollenebene**
  - Kontext der Interaktion durch **strukturierte Classifier** festgelegt = Kontext-Classifier
  - Deren **Rollen** stellen die **Interaktionspartner** dar
  - Tatsächliche Interaktion findet selbstverständlich auf **Instanzebene** zwischen **Objekten** statt

- **Modellierung** auf **Instanzebene möglich**, um eine Abfolge von Nachrichten zwischen konkreten Objekten darzustellen = **Trace**

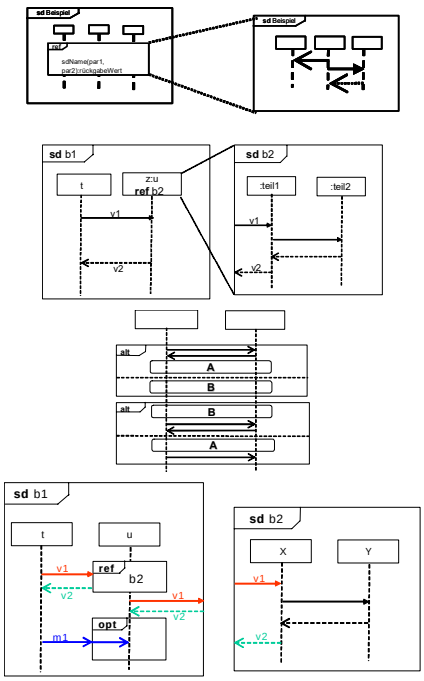
9.2 Übersicht Sequenzdiagramm

- Diagrammrahmen
- Lebenslinie
- Nachricht
- Zustandsinvariante
- Kombiniertes Fragment



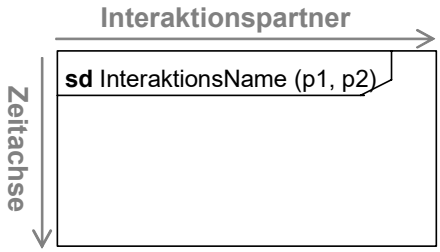
Darstellung von Sequenzdiagrammen

- Interaktionsreferenz
- Zerlegung einer Lebenslinie
- Fortsetzungsmarke
- Verknüpfungspunkt



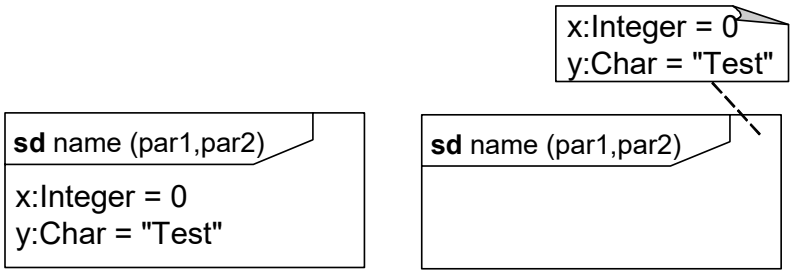
9.3 Digrammrahmen

- Rahmennotation
  - Für alle UML2 Diagrammarten möglich
  - Pentagon
    - Diagrammtypsd für Sequenzdiagramm
    - Interaktionsname
    - Optionale Parameter



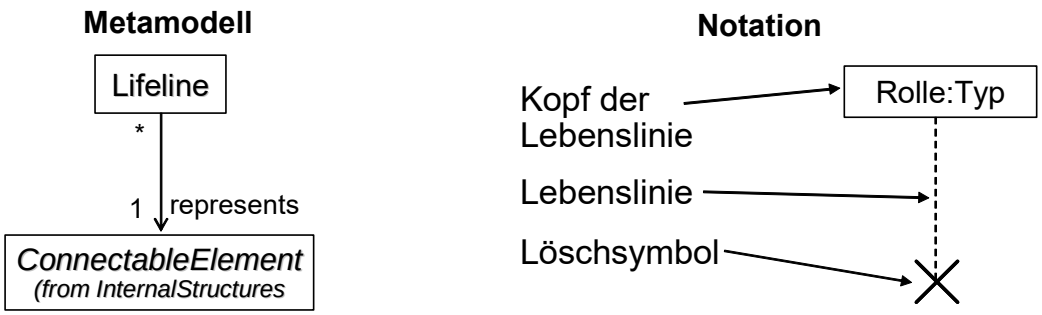
- Zwei Darstellungsdimensionen
  - Vertikale Dimension repräsentiert Zeitachse
  - Horizontale Dimension repräsentiert Interaktionspartner in Form von Rollen

- Lokale Attribute
  - zwei Varianten:



9.4 Lebenslinie

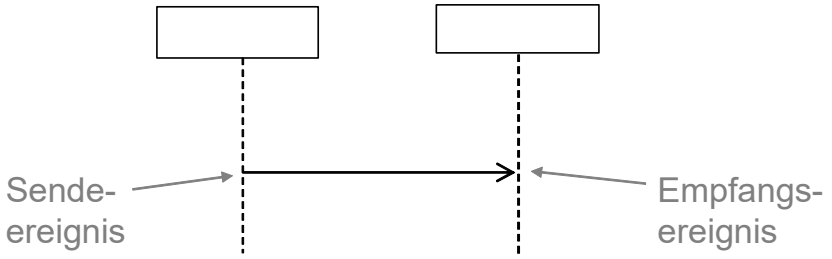
- Eine Lebenslinie beschreibt **genau einen Interaktionspartner**
- Als Interaktionspartner können **alle Rollen** des **Kontext-Classifiers** auftreten
- Rollen sind vom Typ **ConnectableElement** (z.B. Klassen, Attribute oder Ports)



9.4.1 Ereignisspezifikation

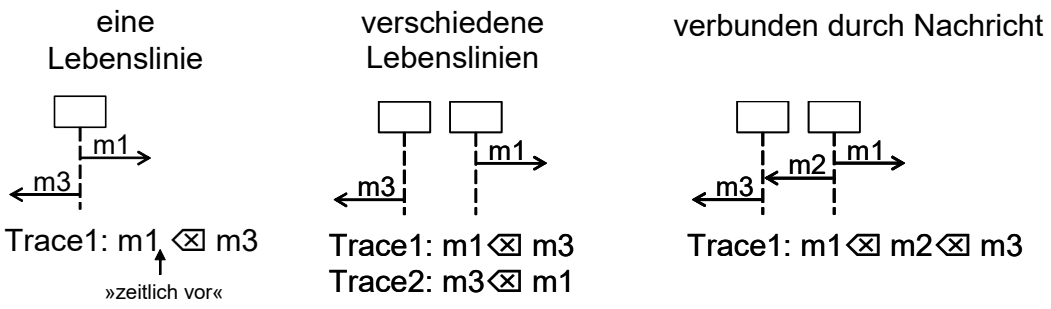
- Interaktionen werden als **Folge von Ereignisspezifikationen** auf Lebenslinien betrachtet

- Beispiel für Ereignisspezifikationen
  - **Senden** und **Empfangen** von Nachrichten auf verschiedenen Lebenslinien oder der gleichen Lebenslinie



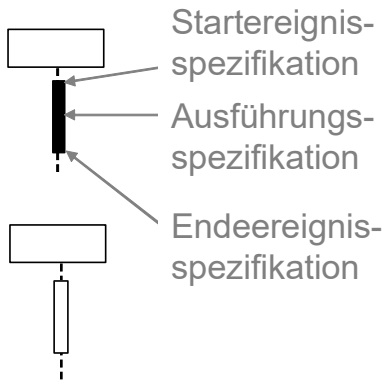
9.4.2 Reihenfolge

- Reihenfolge von Ereignisspezifikationen
  - **Vertikale Zeitachse** bestimmt nur die **Ordnung** von Ereigniseintritten **pro Lebenslinie**
    - Jedoch **nicht** die Reihenfolge von Ereigniseintritten **auf verschiedenen Lebenslinien**
  - Erst durch **Nachrichten zwischen Lebenslinien** wird eine Ordnung über Lebenslinien hinweg erzwungen



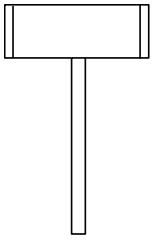
9.4.3 Ausführungsspezifikation

- Die **Ausführung** einer Aktivität/Operation wird durch zwei Ereignisspezifikationen (Start und Ende) auf der gleichen Lebenslinie definiert
- Diese sogenannte **Ausführungsspezifikation** kann durch einen Balken dargestellt werden
- Ausführungsarten
  - **Direkt**
    - Interaktionspartner führt Verhalten selbst aus
  - **Indirekt**
    - Ausführung wird an andere Interaktionspartner delegiert



9.4.4 Aktives Objekt

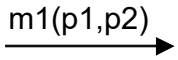
- **Aktive Objekte** verfügen über **eigenen Kontrollfluss** (Prozess oder Thread)
- Können **unabhängig** von anderen Objekten operieren
- **Notation**
  - Kopf der Lebenslinie wird links und rechts mit **doppeltem Rand** versehen
  - **durchgehender** Balken über gesamte Lebenslinie



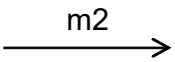
9.5 Nachricht

- Arten der Nachrichtenübermittlung

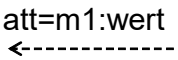
- **Synchroner Kontrollfluss**
  - Der Sender wartet bis zur Beendigung der Interaktion, die durch die Nachricht ausgelöst wurde



- **Asynchroner Kontrollfluss**
  - Die Nachricht wird als Signal betrachtet
  - Der Sender **wartet nicht** auf das Ende der Interaktion

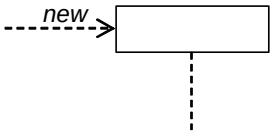


- **Antwortnachricht** (optional)

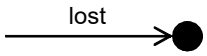


9.5.1 Spezielle Nachrichtenarten

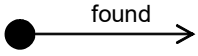
- **Objekterzeugung**
  - Ermöglicht, einen Interaktions-partner erst im Laufe der Interaktion zu erzeugen



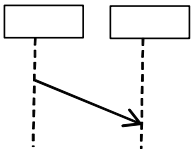
- **Verlorene** Nachricht
  - Senden einer Nachricht an unbekannten oder nichtrelevanten Interaktionspartner



- **Gefundene** Nachricht
  - Empfang einer Nachricht von einem unbekannten oder nicht relevantenInteraktionspartner

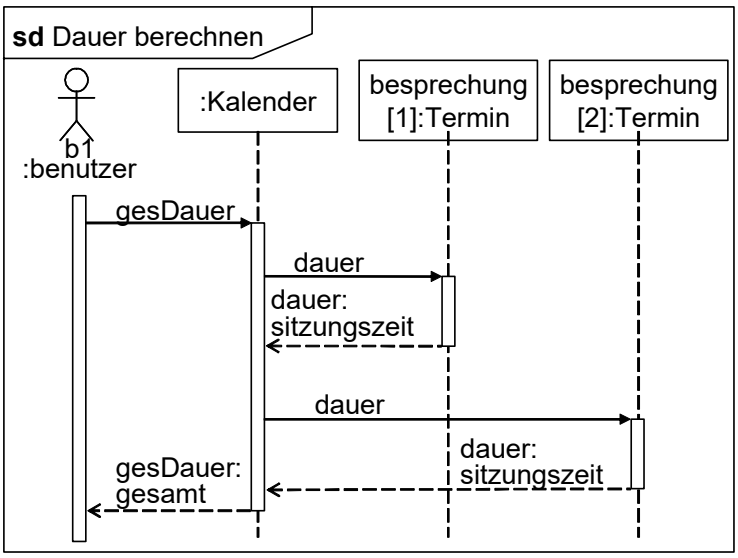


- **Zeitkonsumierende** Übertragung



9.5.2 Beispiel

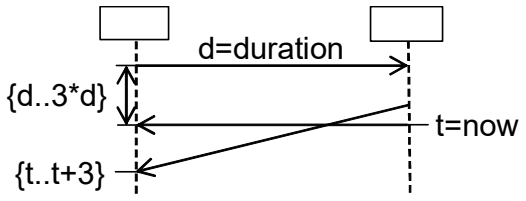
- Berechnung der Dauer eines Termins



9.5.3 Zeiteinschränkungen

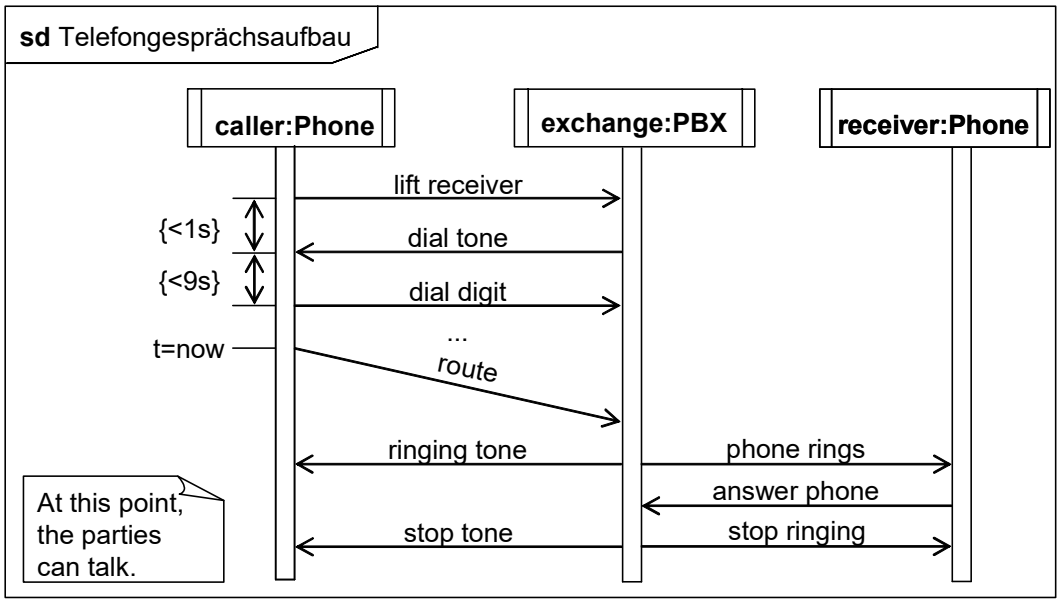
- Arten
  - **Zeitpunkt** (time constraint)
    - Bezieht sich auf einzelne Ereignisspezifikation
  - **Zeitdauer** (duration constraint)
    - Bezieht sich auf Zeitintervall zwischen zwei Ereignisauftritten

- Vordefinierte Aktionen zur Zeitberechnung
  - **now**: Berechnung der **aktuellen Zeit**
  - **duration**: Berechnung einer **Zeitdauer**
  - Erhaltene Werte können Variablen zugewiesen werden
  - Variablen können in Zeitausdrücken verwendet werden



9.5.4 Beispiel

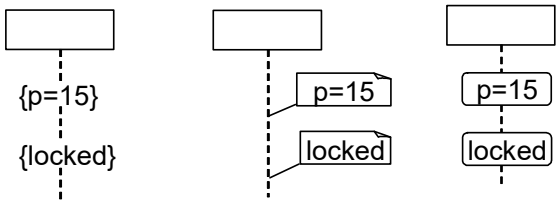
- Aufbau eines Telefongesprächs



(basierend auf [Rumb05])

9.6 Zustandsinvariante

- **Zusicherung**, dass eine **bestimmte Bedingung** zu einem **bestimmten Zeitpunkt** erfüllt ist
- Bezieht sich immer auf eine **bestimmte Lebenslinie**
- Wird **vor Eintritt** des darauf folgenden **Ereignisses ausgewertet**
- Falls Zustandsinvariante **nicht erfüllt** ist - **Fehler**
- Notationsvarianten

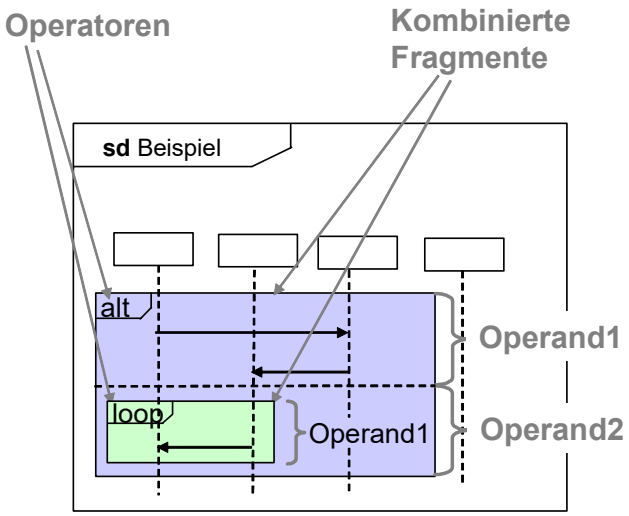


9.7 Kombinierte Fragmente

- Modellierung von **Kontrollstrukturen**
- Bestandteile: **Operator** und **Operanden**
- **Operator**
  - Definiert Art des kombinierten Fragments
  - 12 vordefinierte Operatoren
- **Operand**
  - Ein Operator enthält **1 oder mehrere Operanden**, je nach Operatorart
  - Kann **Interaktionen, kombinierte Fragmente** (Schachtelung!) und **Referenzen auf Sequenzdiagramme** umfassen

9.7.1 Notation

- Kombiniertes Fragment wird wie Sequenzdiagramm mit Rahmen dargestellt
- Art des Fragments wird durch Operator im Pentagon festgelegt (default: seq)
- Operanden werden durch gestrichelte Linien voneinander getrennt

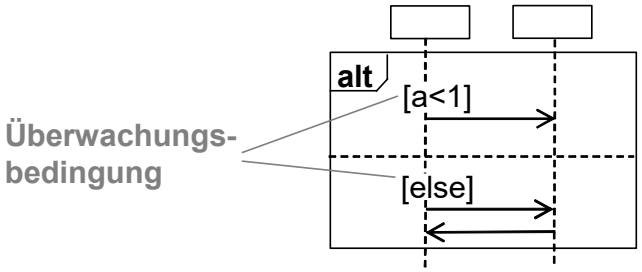


9.7.2 Operatorarten

	Operator	Zweck
Verzweigungen und Schleifen	**alt**	Alternative Interaktionen
	opt	Optionale Interaktionen
	break	Ausnahme Interaktionen
	loop	Iterative Interaktionen
Parallelität und Ordnung	**seq**	Sequentielle Interaktionen mit schwacher Ordnung (Default-Operator)
	strict	Sequentielle Interaktionen mit strenger Ordnung
	par	Parallele Interaktionen
	critical	Atomare Interaktionen
Filterungen und Zusicherungen	**ignore**	Irrelevante Interaktionen
	consider	Relevante Interaktionen
	assert	Zugesicherte Interaktionen
	neg	Ungültige Interaktionen

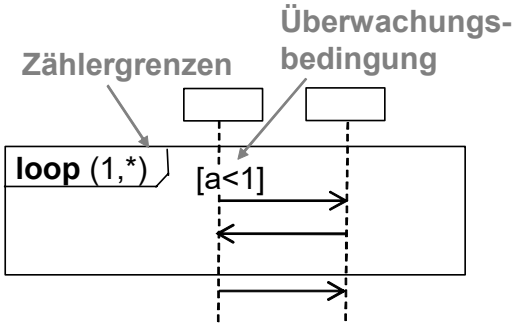
9.7.3 Verzweigungen und Schleifen: alt-Operator

- Darstellung von zwei oder mehreren **alternativen Interaktionsabläufen**
- Zur Laufzeit wird **maximal ein Operand** ausgeführt
- Auswahl eines Operanden anhand von **Überwachungsbedingungen**
  - **Boolscher Ausdruck** in **eckigen Klammern**
  - Vordefinierte **else-Bedingung**: Operand wird ausgeführt, falls die Bedingungen aller anderen Operanden nicht erfüllt sind



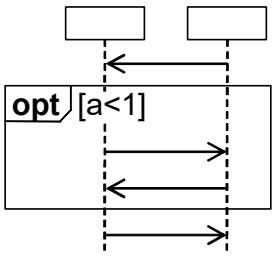
9.7.4 Verzweigungen und Schleifen: loop-Operator

- Darstellung einer **Schleife** über einen bestimmten Interaktionsablauf
- Fragment enthält nur **einen Operanden**
- Ausführungshäufigkeit wird durch **Zähler** mit Unter- und Obergrenze dargestellt
- Optional: **Überwachungsbedingung**, wird bei jedem Durchlauf überprüft

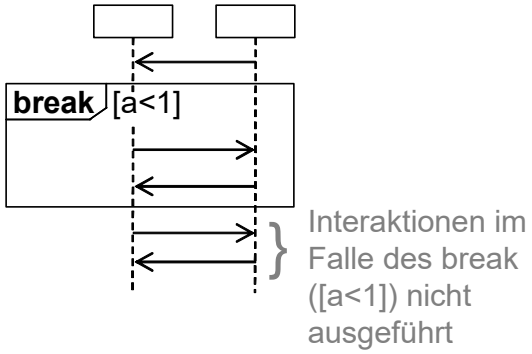


9.7.5 Verzweigungen und Schleifen: opt- und break-Operator

- Überwachungsbedingung steuert Durchlauf der Interaktionen
- Optionale Interaktionen



- Ausnahme-Interaktionen



9.7.6 Parallelität und Ordnung: seq-Operator / strict-Operator

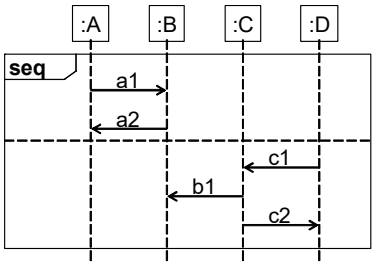
- Sequentielle Interaktion mit schwacher Ordnung

  Beispiele für mögliche Abfolgen:

  Trace1: a1 ⌫ a2 ⌫ c1 ⌫ b1 ⌫ c2

  Trace2: a1 ⌫ c1 ⌫ a2 ⌫ b1 ⌫ c2

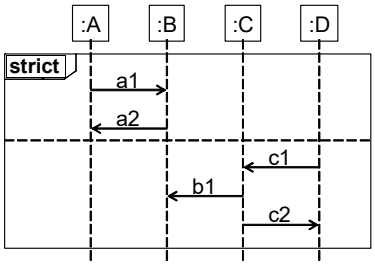
  Trace3: c1 ⌫ a1 ⌫ a2 ⌫ b1 ⌫ c2



- Sequentielle Interaktion mit strenger Ordnung

  mögliche Abfolge:

  Trace1: a1 ⌫ a2 ⌫ c1 ⌫ b1 ⌫ c2



9.7.7 Parallelität und Ordnung: par-Operator

- Nebenläufige Interaktionen
  - Lokale Reihenfolge pro Operand muss erhalten bleiben - z.B. **a1 vor b1**

  Trace1: a1 ⌫ a2 ⌫ c1 ⌫ b1 ⌫ c2

  Trace2: a1 ⌫ c1 ⌫ a2 ⌫ b1 ⌫ c2

  Trace3: a1 ⌫ c1 ⌫ b1 ⌫ a2 ⌫ c2

  Trace4: a1 ⌫ c1 ⌫ b1 ⌫ c2 ⌫ a2

  Trace5: c1 ⌫ a1 ⌫ a2 ⌫ b1 ⌫ c2

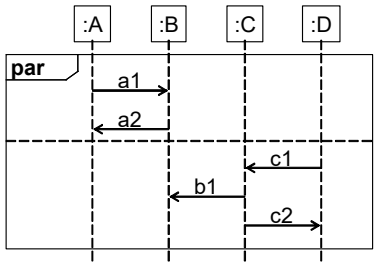
  Trace6: c1 ⌫ a1 ⌫ b1 ⌫ a2 ⌫ c2

  Trace7: c1 ⌫ a1 ⌫ b1 ⌫ c2 ⌫ a2

  Trace8: c1 ⌫ b1 ⌫ a1 ⌫ a2 ⌫ c2

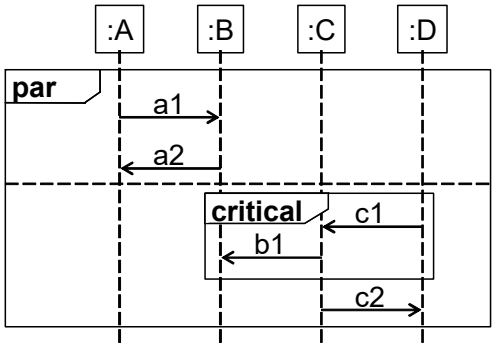
  Trace9: c1 ⌫ b1 ⌫ a1 ⌫ c2 ⌫ a2

  Trace10: c1 ⌫ b1 ⌫ c2 ⌫ a1 ⌫ a2



9.7.8 Parallelität und Ordnung: critical-Operator

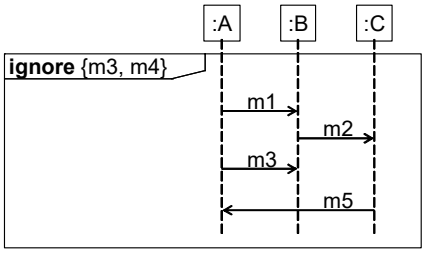
- Kritischer Bereich
  c1 unmittelbar vor b1



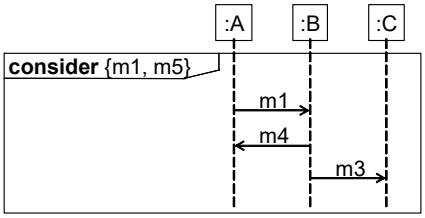
Trace1: a1 ⌫ a2 ⌫ c1 ⌫ b1 ⌫ c2
Trace2: a1 ⌫ c1 ⌫ b1 ⌫ a2 ⌫ c2
Trace3: a1 ⌫ c1 ⌫ b1 ⌫ c2 ⌫ a2
Trace4: c1 ⌫ b1 ⌫ a1 ⌫ a2 ⌫ c2
Trace5: c1 ⌫ b1 ⌫ a1 ⌫ c2 ⌫ a2
Trace6: c1 ⌫ b1 ⌫ c2 ⌫ a1 ⌫ a2

9.7.9 Filterungen und Zusicherungen: ignore-Operator / consider-Operator

- Irrelevante Interaktionen



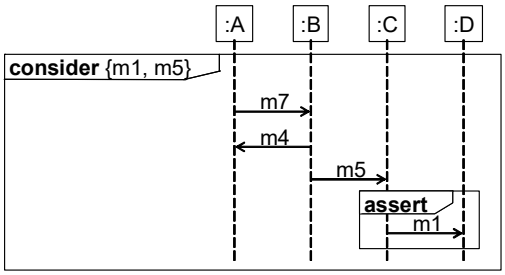
- Relevante Interaktionen



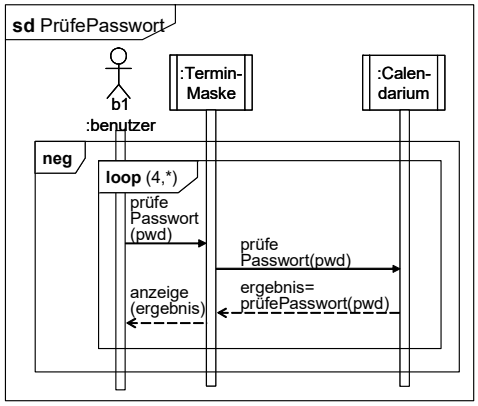
- Diese Operatoren werden meist in Kombination mit Operator *assert* verwendet.

9.7.10 Filterungen und Zusicherungen: assert-Operator / neg-Operator

- Zugesicherte Interaktionen



- Ungültige Interaktionen



9.8 Modularisierung

- **Zweck**: Wiederverwendung und Reduktion der Komplexität

- **Interaktionsreferenz**
  - Zur Referenzierung anderer Sequenzdiagramme
  - Dadurch können Interaktionsabläufe und Lebenslinien zerlegt werden

- **Fortsetzungsmarke**
  - Zur Zerlegung der Operanden eines **alt**-Operators

- **Verknüpfungspunkt**
  - Zur Verbindung von Nachrichten zwischen Sequenzdiagrammen, Interaktionsreferenzen oder kombinierten Fragmenten

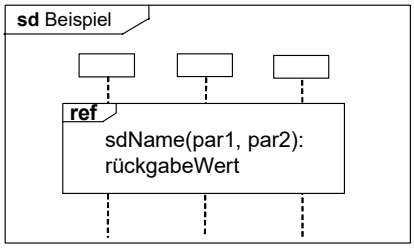
9.8.1 Interaktionsreferenz

- **Interaktionen** des referenzierten Sequenzdiagramms werden **ausgeführt**, so als wären sie direkt in das referenzierende Diagramm **eingebettet**

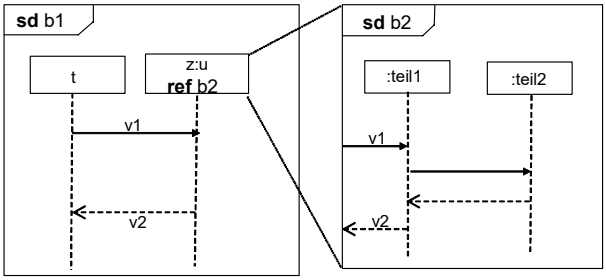
- Eventuell vorhandene **Parameter** werden gebunden

- Nach Ausführung der referenzierten Interaktionen wird **unterhalb der Interaktionsreferenz fortgesetzt**

- Zerlegung von Interaktionsabläufen
  - Rahmen mit Pentagon in der linken oberen Ecke
  - Pentagon enthält Schlüsselwort **ref**
  - Rahmen enthält Namen der referenzierten Interaktion, optional Parameter und Rückgabewerte



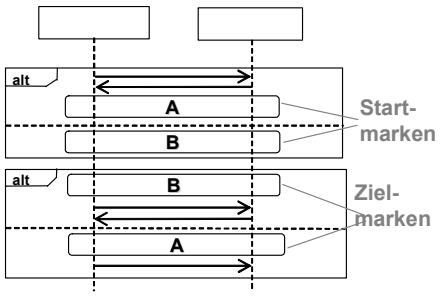
- **Zerlegung von Lebenslinien**
  - **Lebenslinien** können **interne Strukturen** aufweisen, für die eigene Sequenzdiagramme spezifiziert werden können
  - Schlüsselwort **ref** im Kopf der Lebenslinie



9.8.2 Fortsetzungsmarke

- **Zweck**: Zerlegung der Operanden eines alt-Fragments

- **Startmarke** am Ende eines Interaktionsteils verweist auf **Zielmarke** am Beginn eines anderen Interaktionsteils
  - Gleiche Markennamen
  - Überdeckung gleicher Menge an Lebenslinien
  - Rechtecke mitabgerundeten Ecken



- Erreicht die Ausführung die Startmarke, wird mit den Interaktionen unterhalb der Zielmarke fortgefahren
  - **Achtung**: kein Rücksprung zur Startmarke möglich!

9.8.3 Verknüpfungspunkt

- **Zweck**: ermöglicht den **Nachrichtenfluss zwischen**
  - Sequenzdiagrammen und/oder
  - kombinierten Fragmenten und/oder
  - Interaktionsreferenzen

- Dadurch können **hereinkommende** oder **hinausgehende Nachrichten** modelliert werden

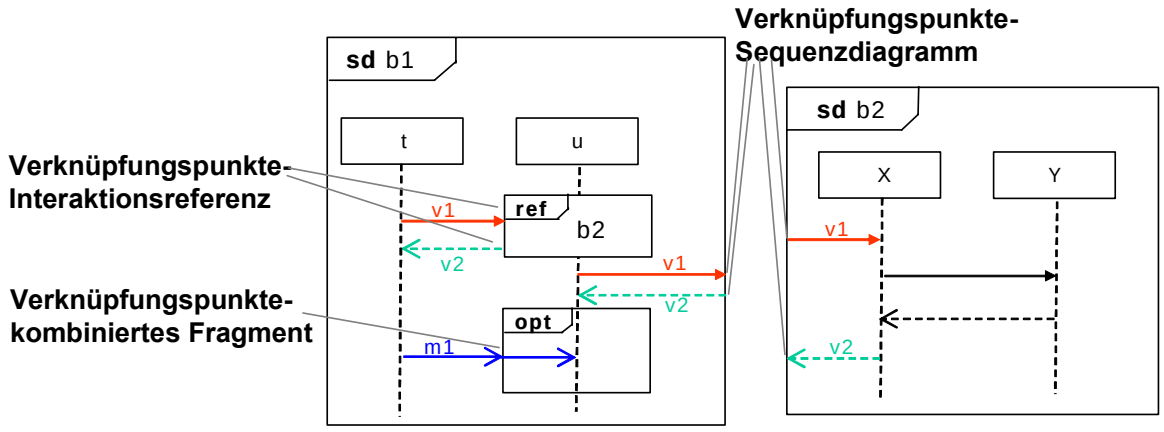
- Verknüpfungspunkte zwischen Sequenzdiagrammen können
  - entweder implizit über Namensgleichheit
  - oder explizit über eine Interaktionsreferenz

  modelliert werden

Verknüpfungspunkt

- Spitze oder Ende eines Nachrichtenpfeils berührt Rand des Rahmes

- Optional: Name für Verknüpfungspunkt wird neben Schnittpunkt von Nachrichtenpfeil und Rahmen angegeben

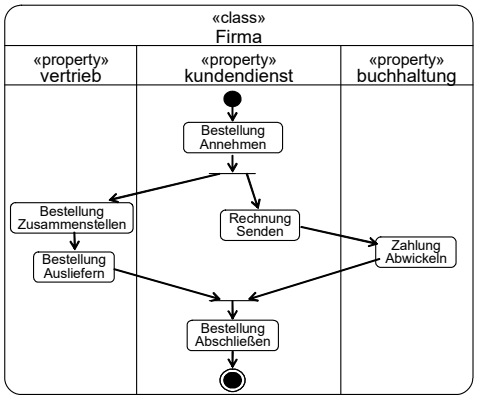


9.9 Exkurs: Sequenzdiagramm vs. Aktivitätsdiagramm

- Pfeil im Aktivitätsdiagramm
  - Zeigt Abhängigkeiten zwischen Aktionen
  - Zeigt NICHT: Nachrichten zwischen Objekten
  - Entspricht einer, mehreren oder keiner Nachricht

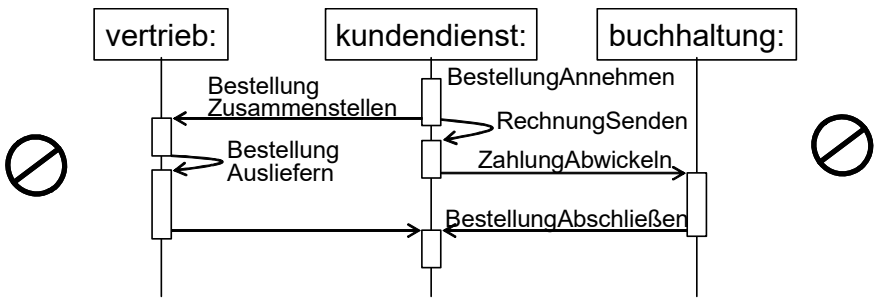
- Pfeil im Sequenzdiagramm
  - Zeigt Nachrichten zwischen Objekten
  - Entspricht Operation einer Klasse

- Beispiel: Akitivitätsdiagramm



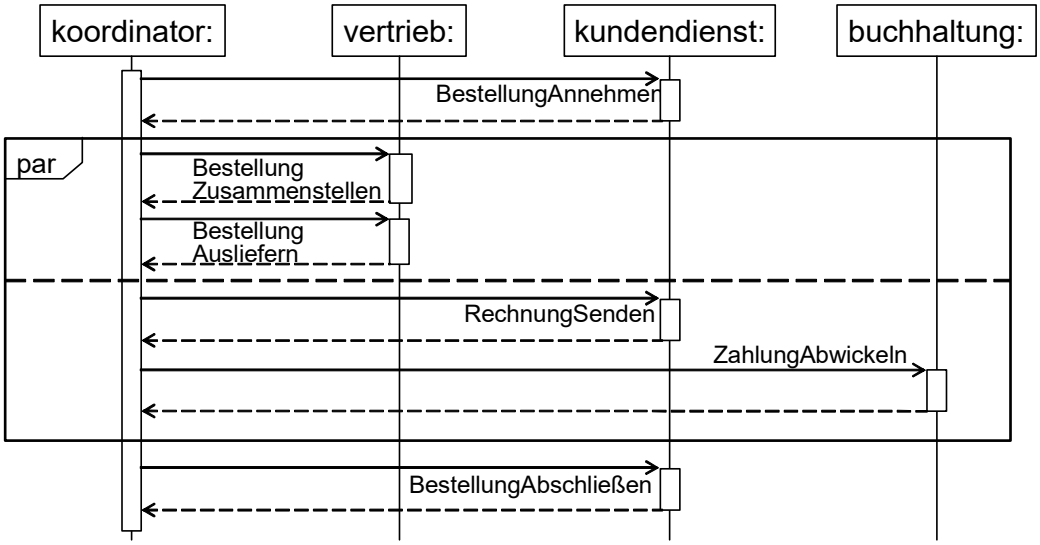
Exkurs: Sequenzdiagramm vs. Aktivitätsdiagramm

- Beispiel: Sequenzdiagramm – **falsche Transformation**
  - Pfeile können nicht zu Nachrichten transformiert werden
  - Beispielsweise kann RechnungSenden nicht das Abschicken der Nachricht ZahlungAbwickeln beinhalten
  - RechnungSenden ist zu Ende, bevor ZahlungAbwickeln gestartet wird
  - Damit wird sichergestellt, dass RechnungSenden wiederverwendbar ist, ohne festzulegen, was vorher und nachher passieren muss



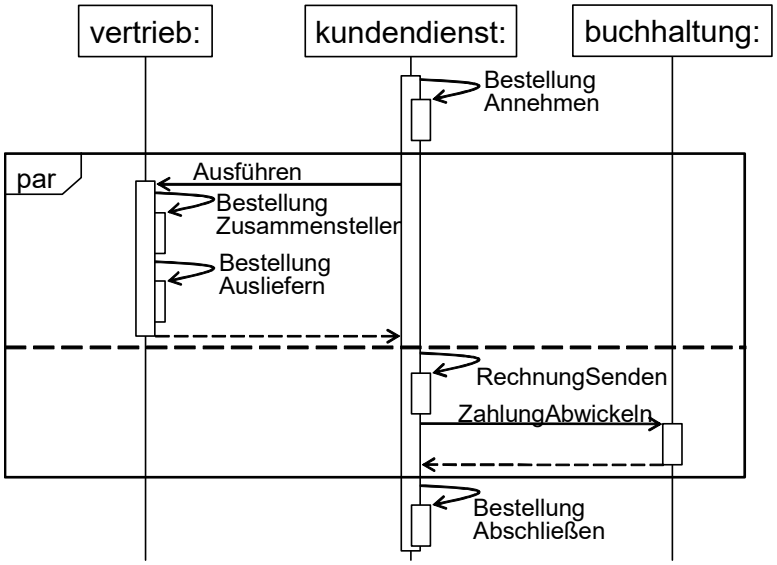
Exkurs: Sequenzdiagramm vs. Aktivitätsdiagramm

- Beispiel: Sequenzdiagramm – Alternative 1



Exkurs: Sequenzdiagramm vs. Aktivitätsdiagramm

- Beispiel: Sequenzdiagramm – Alternative 2



Literatur und Quellen:

- [OMG 2015] OMG Unified Modeling Language™ (OMG UML), Version 2.5
  Normative Reference: http://www.omg.org/spec/UML/2.5
  OMG Document Number formal/2015-03-01

- Kapitel 4.4 in:
  Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, Werner Retschitzegger:
  UML @ Work - Objektorientierte Modellierung mit UML2.
  dpunkt Verlag 2005 / 3. aktualis. u. überarb. Aufl. 2005.
  ISBN-13: 9783898642613 ISBN-10: 3898642615

- Tool zur Vertiefung: BEE-UP Modelling Tool:
  http://austria.omilab.org/psm/content/bee-up/info
  http://www.omilab.org/web/guest/omilab-in-education/cmmc