cooperate

Einführung in die grafische Beschreibungssprache UML

Vanessa Petrausch

Projektpartner:

Karlsruher Institut für Technologie

FZI

Gefördert durch:

Bundesministerium für Arbeit und Soziales

Förderkennzeichen: 01KM141108

Eigenschaften der UML

- UML: Unified Modeling Language
- 1997 durch die Object Management Group (OMG) standardisiert
- Modellierungssprache, keine Programmiersprache
- Ursprung: Modellierung von objektorientierten (OO) Systemen, jedoch auch für beliebige Bereiche anwendbar
- → effektive Beschreibung von unterschiedlichen Komponenten von Systemen
  - Analyse
  - Entwurf
  - Entwicklung
  - etc.
- Bis heute 14 verschiedene Diagramme, die in Struktur- und Verhaltensdiagramme aufgeteilt werden

Strukturdiagramme

- Modelliert statische, zeitunabhängige Elemente eines Systems [2]
  - Können Strukturen von Systemen darstellen
  - Domänenmodellierung (Datenmodell)
  - Bisher 7 verschiedene Typen

- Vorstellung in diesem Dokument: Klassendiagramm

Verhaltensdiagramme

- Dynamisches Verhalten von Systemen
  - Spezifikation der Anforderungen an ein System
  - Grundsätzliche Abläufe innerhalb des Systems
  - Abläufe und Zusammenhänge im Zeitablauf
  - Bisher 7 verschiedene Typen

- Vorstellung in diesem Dokument:
  - Anwendungsfalldiagramm (Use Case Diagram)
  - Aktivitätsdiagramm (Activity Diagram)
  - Zustandsdiagramm (Statechart)

Allgemeiner Aufbau Diagramme

- Optionale Darstellung in einem Rechteck mit Karteireiter oben links mit Diagrammtyp (als Kürzel) und Name des Diagrammes
- Konzept der „Notiz“ in allen Diagrammen verfügbar
  - dargestellt als Rechteck mit „Eselsohr“
  - Natürliche Sprache, Object Constraint Language (OCL) oder beliebige Syntax
  - Mithilfe von gestrichelter Linie an alle Elemente von Diagrammen zuweisbar

Allgemeiner Aufbau: Verbindungen

- Verbindungen zwischen Elementen der Diagramme als gerichtete (Pfeilende) oder ungerichtete (kein Pfeilende) Linie
- Zusätzliche Verbindungen in verschiedenen Diagrammen vorhanden, Erläuterungen in den jeweiligen Kapiteln
- Stellt Zugriff zwischen Elementen dar
- Multiplizitäten durch Festlegung von Unter- und Obergrenzen an Verbindungen möglich
  - Details beim Klassendiagramm

Allgemeiner Aufbau: Stereotypen

- Geben Auskunft über Zweck und Rolle eines Elementes innerhalb eines Diagramms
- Verändern nicht Semantik des Elements
- Darstellung in doppelten spitzen Klammern, Guillemets <<Stereotypbezeichnung>>

- Beispiel wäre die Deklaration einer Klasse als Interface

Zusammenhang der 4 Diagramme

- Anwendungsfall modelliert Basisfunktionalitäten eines Systems aus Sicht der Nutzer (Menschen, Maschinen)
- Klassendiagramm definiert, welche Klassen in der Realisierung dieser Funktionen benötigt werden
- Zustandsdiagramm zeigt das Verhalten innerhalb eines Objektes (= Realisierung einer Klasse) oder Verhalten zur Erfüllung von Anwendungsfällen
- Aktivitätsdiagramm definiert die Prozesse, welche benötigt werden um die Anwendungsfälle zu „implementieren“

Klassendiagramm

Strukturdiagramm

Klassendiagramm (KD): Einleitung

- Spezifiziert Strukturierung von Daten, Objektstrukturen eines Systems und Beziehungen zwischen Objekten
- Modelliert statische Strukturen, verändern sich nicht über die Zeit
- Hauptkonzepte: Klasse, Generalisierung und Assoziationen
- Klasse = Objekttyp mit Eigenschaften (Attributen) und Operationen

KD: Klassen

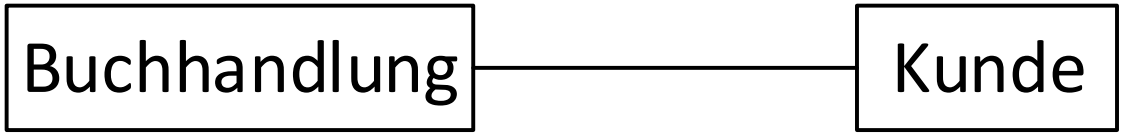
- Beschreibt Art Bauplan für Objekte mit gleicher Struktur und Verhalten
- Horizontal 3-geteiltes Rechteck repräsentiert Klasse mit 3 verschiedenen Kategorien
  - Klassenname
  - Attribute (Struktur)
  - Methoden (Verhalten)
- Es müssen nicht alle Kategorien angegeben werden, eindeutiger Klassenname genügt
- Attribute/Methoden können verschiedene Sichtbarkeiten haben: public(+), protected(#), private(-) oder nur im package (~) sichtbar

Klassenname
Attribute
Methoden

Buch
titel: String isbn: Int

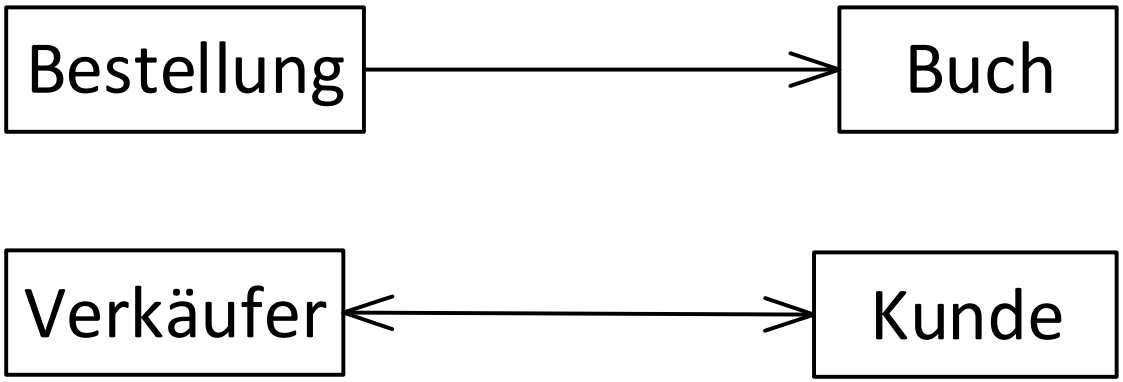
KD: Bedeutung und Darstellung Assoziation

- Assoziationen beschreiben die Zugriffe zwischen Klassen, d.h. welche Klasse kann auf Attribute und Methoden anderer Klassen zugreifen
- Darstellung als Linie zwischen den Klassen
- Beispiel: ungerichtete Assoziation: keine Pfeile an Enden, Interaktion der Klassen nicht ersichtlich



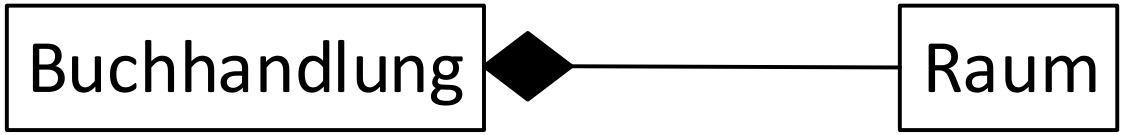
KD: gerichtete Assoziationen

- Gerichtete Assoziationen haben Pfeile an Linienenden
  - Direktional: nur ein Ende hat eine Pfeilspitze
  - Bidirektional: beide Enden haben Pfeilspitze



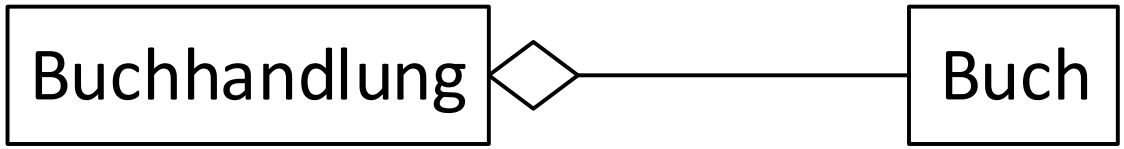
KD: Teil-Ganzes-Beziehung: Komposition

- Komposition = Assoziation, welche starke Abhängigkeit einer Klasse darstellt
- Beschreibt Beziehung zwischen einem Ganzen und seinen Teilen
- Schwarz ausgefüllte Raute am „Ganzen“
- Klasse am anderen Ende ist abhängig von anderer Klasse und kann ohne diese nicht existieren
- Klasse Raum hängt von Buchhandlung ab, wird die Buchhandlung zerstört, dann auch die enthaltenen Räume



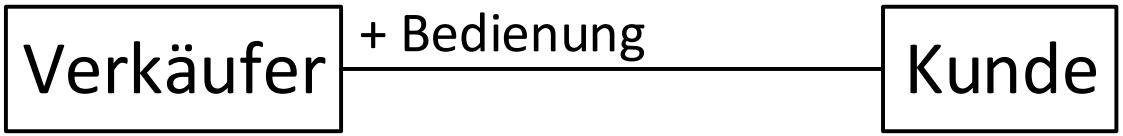
KD: Teil-Ganzes-Beziehung: Aggregation

- Beschreibt Beziehung zwischen einem Ganzen und seinen Teilen
- Weiße Raute am „Ganzen“
- Klasse am anderen Ende ist abhängig von anderer Klasse, kann jedoch auch eigenständig existieren
- Klasse Buch hängt von Buchhandlung ab, wird die Buchhandlung zerstört, können die Bücher jedoch weiter existieren



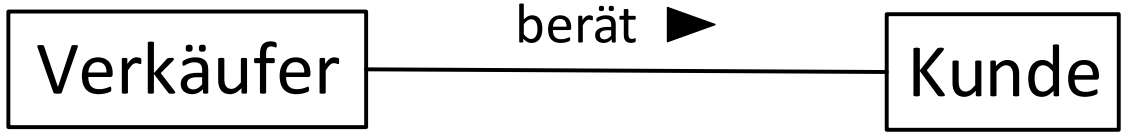
KD: Rollen von Assoziationen

- Beschreibt die Art, in der ein Objekt in der Assoziation beteiligt ist, d.h. welche Rolle das Objekt in dieser einnimmt
- Rollennamen können auch Sichtbarkeiten haben (+,-,#,~)
- Im Beispiel nimmt der Verkäufer die Rolle Bedienung ein, der Kunde dagegen keine



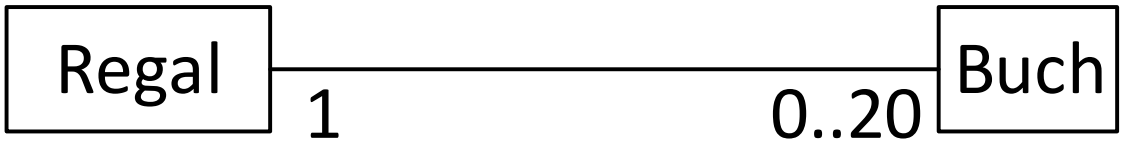
KD: Beschriftung von Assoziationen

- Veranschaulicht Bedeutung der Assoziation
- Wird direkt über oder unter die Assoziation geschrieben
- Optionale Leserichtung, gekennzeichnet als schwarzes Dreieck mit Spitze in Leserichtung
- Beispiel: ein Verkäufer berät den Kunden, Dreieck zeigt von Verkäufer zu Kunde



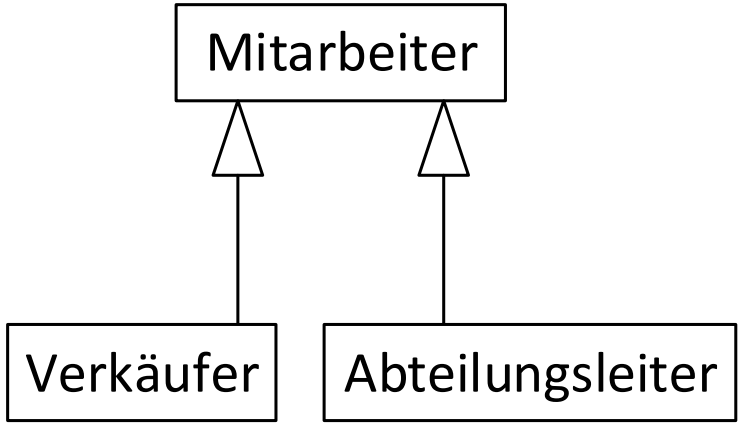
KD: Eigenschaften von Assoziationen: Multiplizitäten

- Angabe von Multiplizitäten an Relationen
  - 1:1
  - 1:n (im Beispiel)
  - n:m
- Angabe von Unter-und Obergrenze möglich: n..m
- Ohne Angabe: 1
- Beispiel: In einem Regal können 0 bis 20 Bücher stehen. Ein Buch steht in genau einem Regal

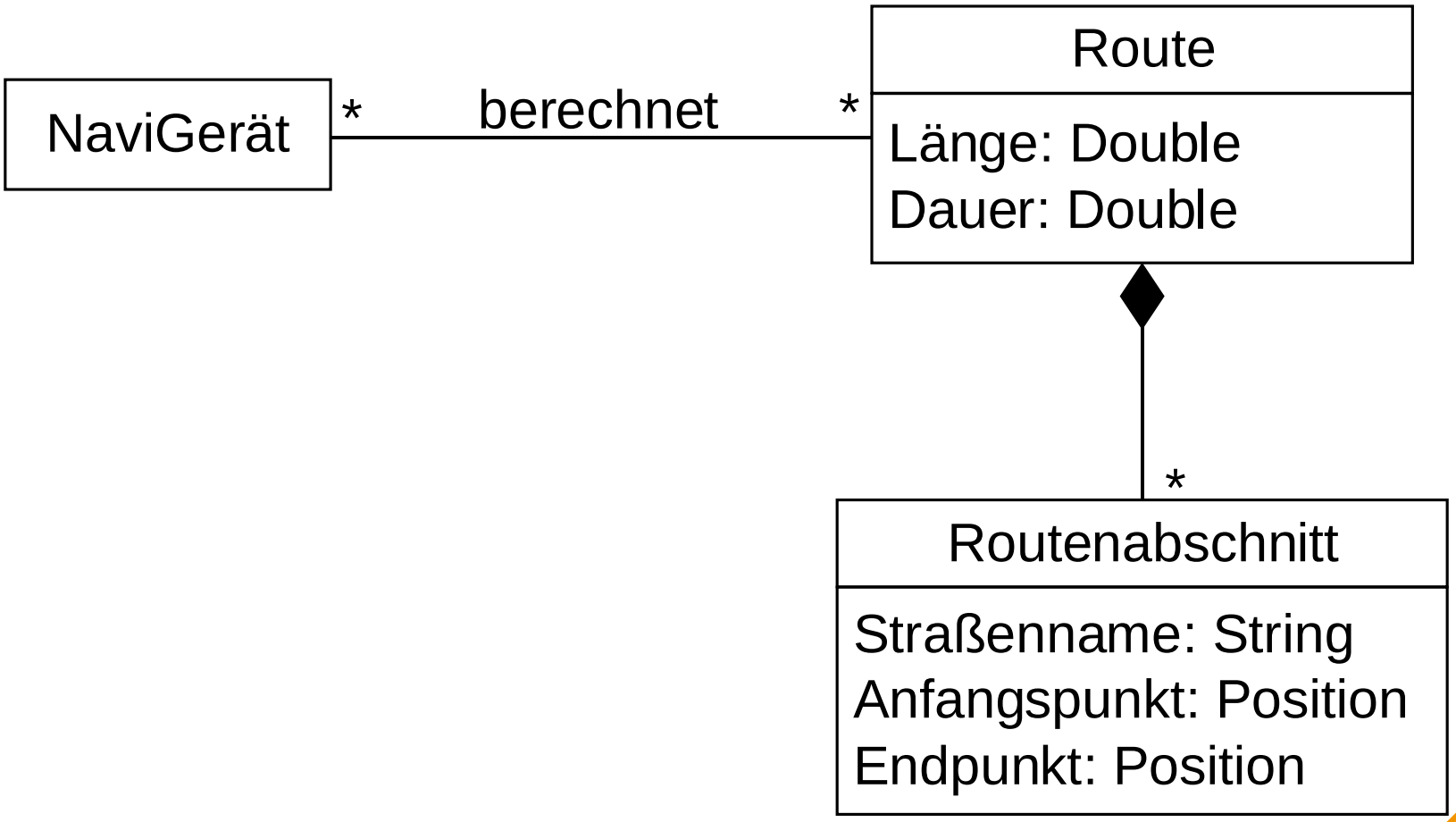


KD: Vererbung

- Attribute, Methoden und Assoziationen können zwischen Klassen vererbt werden
- Verbindungslinie mit geschlossenem, ungefülltem Pfeil an einem Ende (Dreieck)
- Klasse, welche ausgehende Assoziation(Unterklasse) hat, erbt Eigenschaften von Elternklasse (Oberklasse)



KD: Beispiel


NaviGerät
*
berechnet
*
Route
Länge: Double
Dauer: Double
*
Routenabschnitt
Straßenname: String
Anfangspunkt: Position
Endpunkt: Position

Anwendungsfalldiagramm

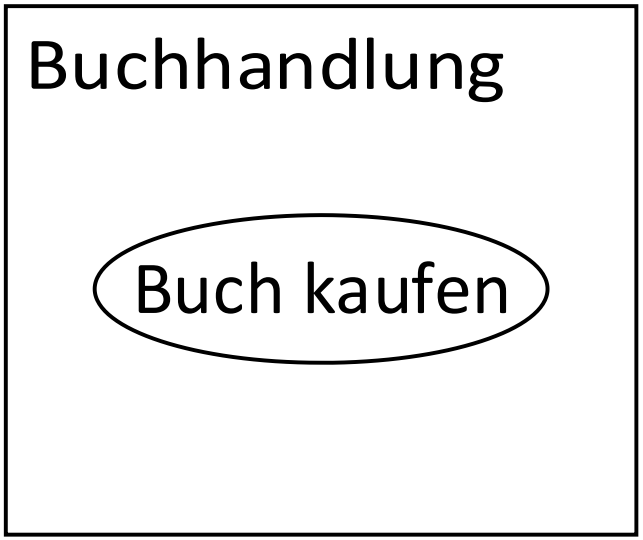
Verhaltendsdiagramm

Anwendungsfalldiagramm (AF): Einleitung

- Grobe Ordnung der Anforderungen an ein System
- Überblick über bereitgestellte Funktionalitäten (genannt Anwendungsfall)
- Stellen nur Funktionalität dar, nicht WIE diese realisiert wird
- Interaktion von Akteuren (Personen, Maschinen) und System
- Welche Akteure nutzen welche Funktionen
- Beantworte die Fragen:
  1. Was wird beschrieben? (Das System)
  2. Wer interagiert mit dem System? (Akteur)
  3. Was kann der Akteur machen? (Anwendungsfall) [1]

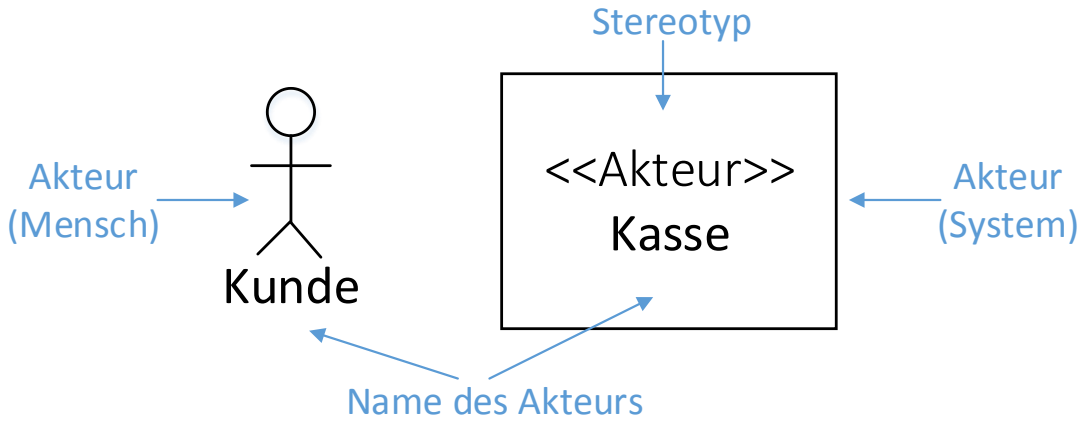
AF: System und Anwendungsfälle

- Systemgrenze: umfasst ein System und enthält Anwendungsfälle, mit denen Akteur interagiert
- Darstellung als Rechteck mit optionalem Namen oben
- Anwendungsfall: spezifiziert vom System bereitgestellte abgeschlossene Menge von Funktionen
- Darstellung als Ellipse (pro Funktion je eine Ellipse) mit Namen darin



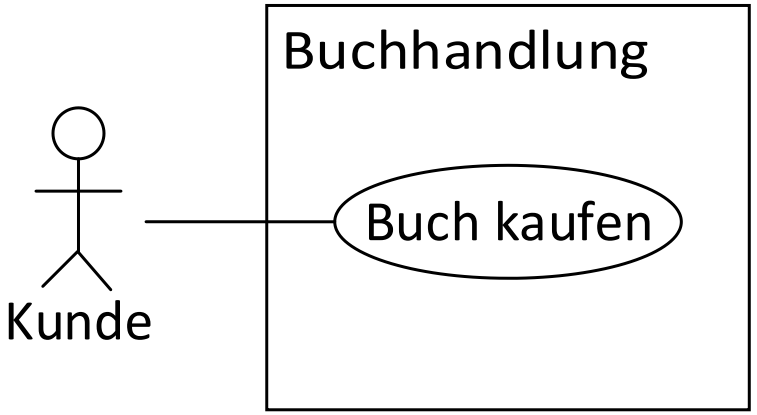
AF: Akteure

- Akteur modelliert Typ oder Rolle, welche während der Interaktion mit System eingenommen wird
- Menschlicher Akteur: Darstellung als Strichmännchen mit Bezeichnung darunter
- Maschineller Akteur: Rechteck mit Stereotyp Akteur und Name



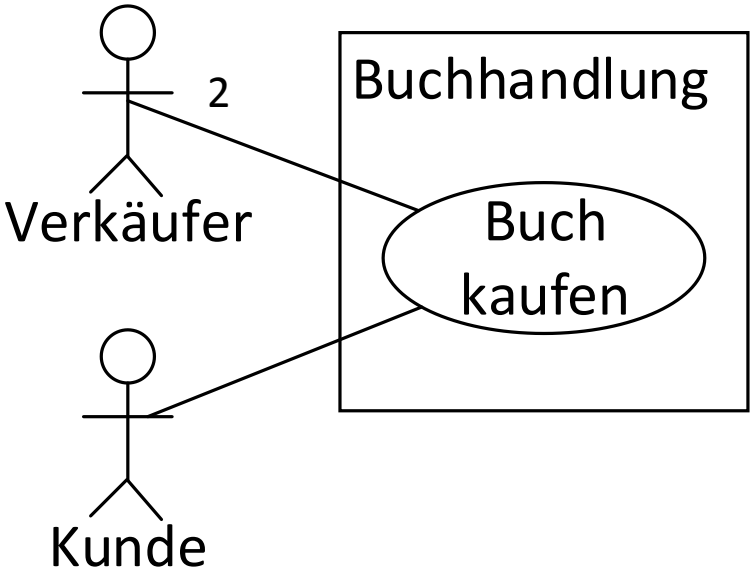
AF: Assoziation Akteur zu System

- Assoziation: durchgezogene Linie
- Akteur interagiert mit Anwendungsfall, wendet Funktion an
- Die Person „Kunde“ kann in einer „Buchhandlung“ ein „Buch kaufen“



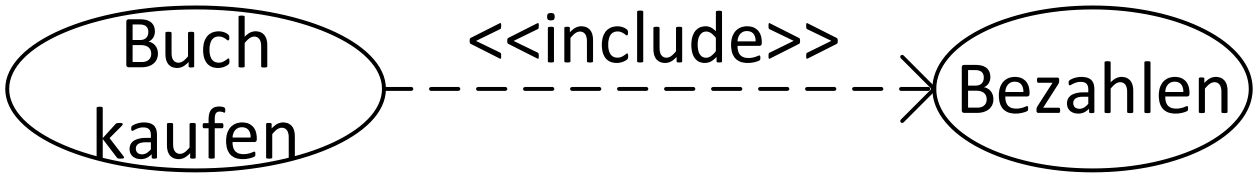
AF: Mehrfachassoziationen und Multiplizitäten

- Mehrfachassoziation am Anwendungsfall: alle Akteure müssen zur Ausführung involviert sein
- Multiplizität am Akteur-Ende: mehrere Akteure sind notwendig zur Ausführung dieses Anwendungsfalls, falls >1
- Multiplizität am Anwendungsfall: Akteur kann in mehren Anwendungsfällen involviert sein



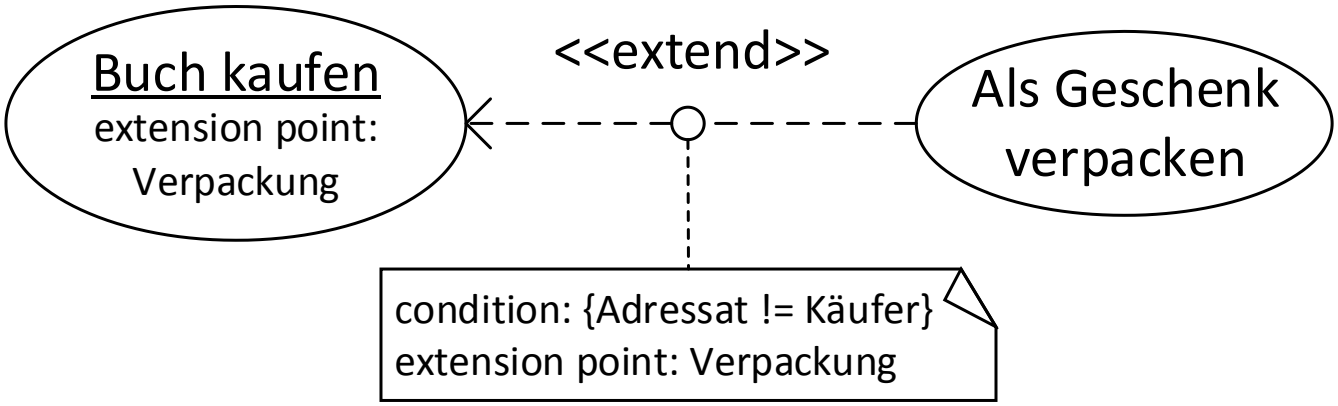
AF: Assoziationen zwischen Anwendungsfällen: include

- Assoziation wird dargestellt durch gerichtete (offenes Pfeilende), gestrichelte Linie + Schlüsselwort include in Guillemets << >>
- Anwendungsfall, von dem Relation (Assoziation) ausgeht, (Basisanwendungsfall) schließt anderen ein (eingebundenen Anwendungsfall)
- Basis nutzt eingebundenen und ist auf diesen angewiesen um seine Funktionalität bereitzustellen
- Eingebundener Anwendungsfall kann unabhängig laufen



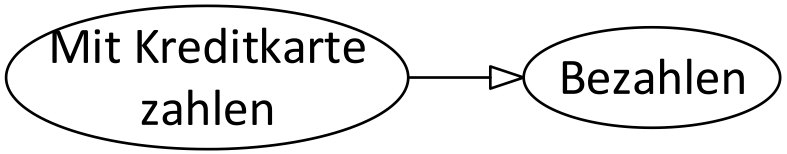
AF: Assoziationen zwischen Anwendungsfällen: extend

- Darstellung wie bei include-Beziehung mit Schlüsselwort extend
- Erweiternder Anwendungsfall (ausgehende Relation) kann optional den Basisanwendungsfall erweitern
- Extension Point gibt Bedingung an, wann der Anwendungsfalls eingebunden werden kann.
- Extension Point wird als Kreis auf der Assoziation dargestellt und mittels einer Linie die Bedingung in Textform innerhalb eines Rechtecks mit „Eselsohr“ angehängt.
- Beide können auch unabhängig voneinander ausgeführt werden



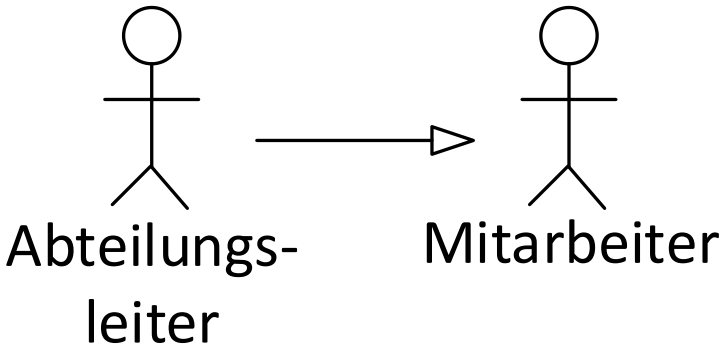
AF: Vererbung zwischen Anwendungsfällen

- Anwendungsfälle vererben Verhalten und Assoziationen zu Akteuren
- Symbol wie bei Klassendiagrammen: weißes Dreieck als Pfeilspitze



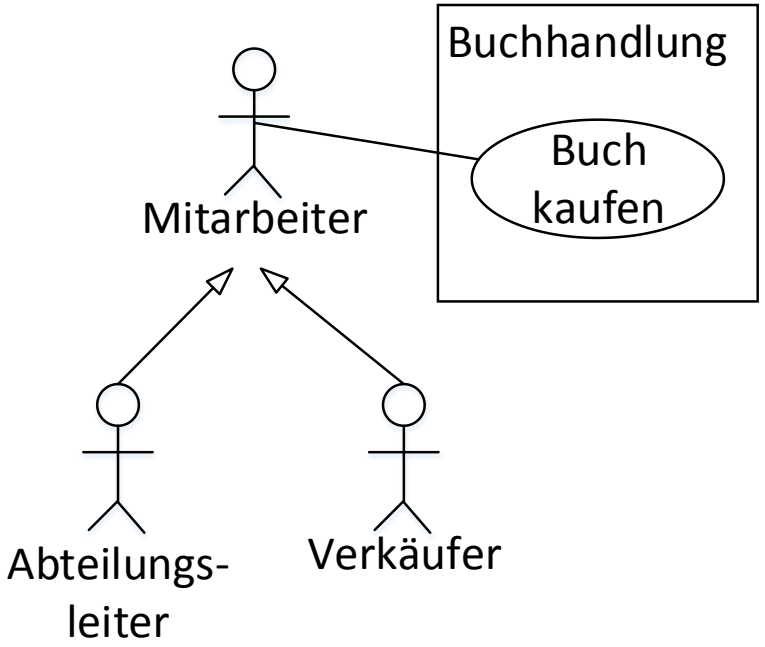
AF: Vererbung zwischen Akteuren

- Akteure können eine Hierarchie bilden (Vererbungskonzept in OO)
- Vererbung von Kommunikations-beziehungen zwischen Akteur und Anwendungsfällen
- Erbender Akteur hat gleichen Zugriff auf Anwendungsfälle wie Akteur, von welchem er erbt

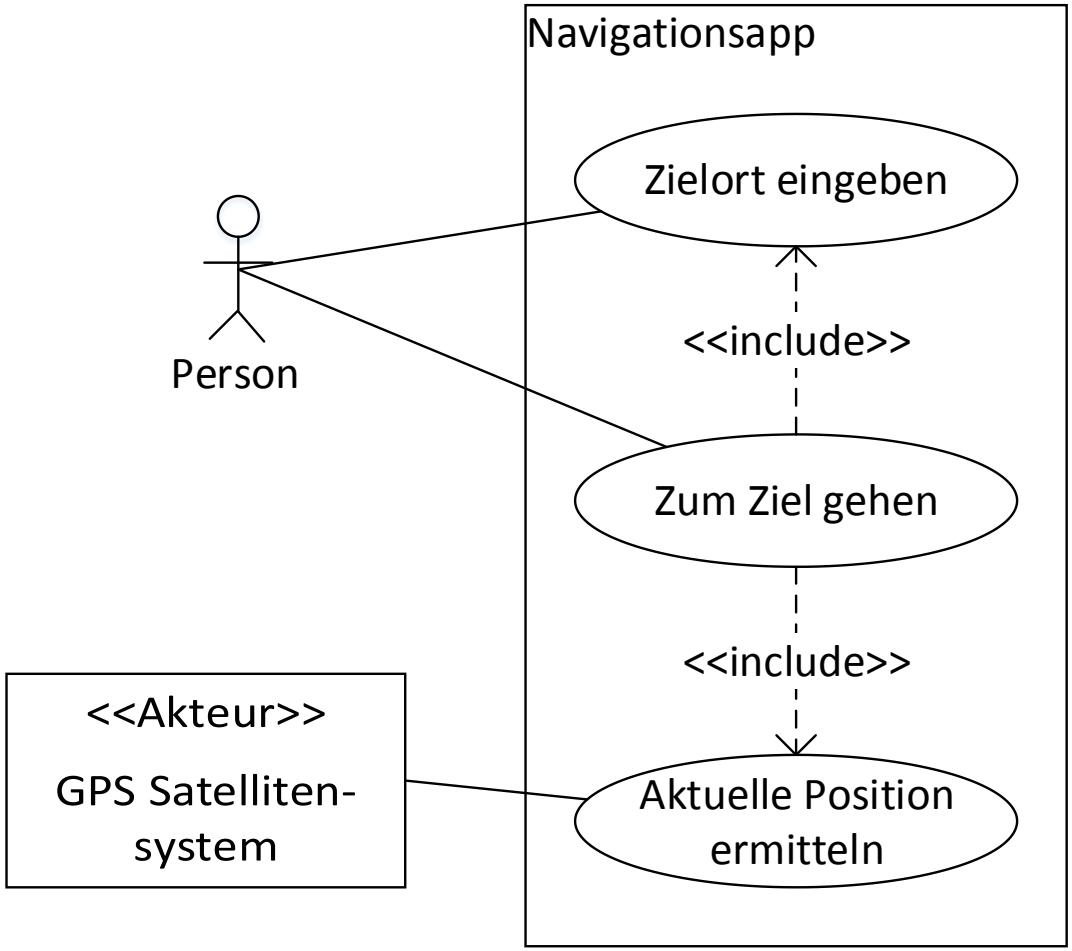


AF: Vererbung von Akteuren bei Assoziationen

- Erben mehrere Akteure von einem Akteur, welcher mit einem Anwendungsfall verbunden ist, muss nur einer dieser Akteure in AF involviert sein
- Gegensatz zu Szenario auf Folie 26, in welchem beide involviert sein müssen



AF: Beispiel


Navigationsapp
Zielort eingeben
<<include>>
Zum Ziel gehen
<<include>>
Aktuelle Position ermitteln
Person
<<Akteur>>
GPS Satelliten-system

Aktivitätsdiagramm

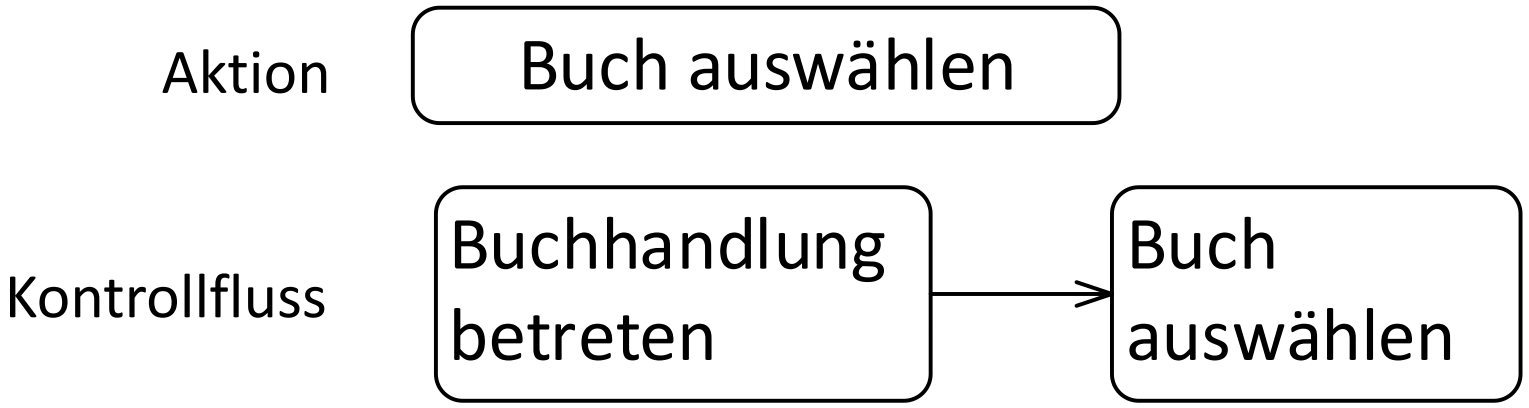
Verhaltensdiagramme

Aktivitätsdiagramm (AD): Einleitung

- Beschreibt Prozesse, z.B. Geschäfts- oder Software-Prozesse
- Stellt Daten- und Kontrollflüsse durch Ablauf von Aktivitäten dar
- Beispiel: Ein Diagramm zeigt Aktivitäten, welche benötigt werden, sodass ein Student an einer Vorlesung teilnehmen kann
- Kann sowohl OO- als auch nicht-OO-Systeme modellieren
- Besteht aus Knoten und Kanten

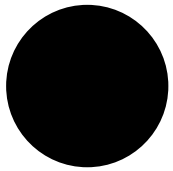
AD: Aktionen

- Idee: Tokens wandern von Knoten zu Knoten entlang Kanten
- Aktionen (Knoten) sind nicht weiter zerlegbare Funktion, in Aktivitäten enthalten
- Aktionen werden über Kontrollflüsse (Verbindungen) in eine Ablaufreihenfolge gesetzt

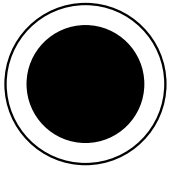


AD: Start- und Endknoten

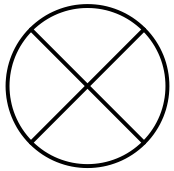
- Mehrere Start- und Endknoten möglich
  - Startknoten: Kontrollfluss beginnt, 1 Token pro Startknoten wird initialisiert
  - Endknoten: alle Kontrollflüsse enden, Token werden zerstört
  - Flussende: nur aktueller Fluss endet: Kreis mit X darin



Startknoten



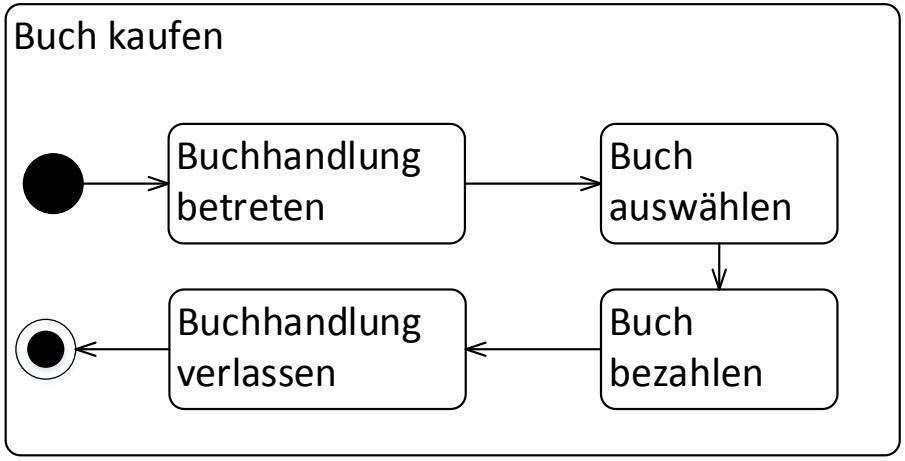
Endknoten



Flussende

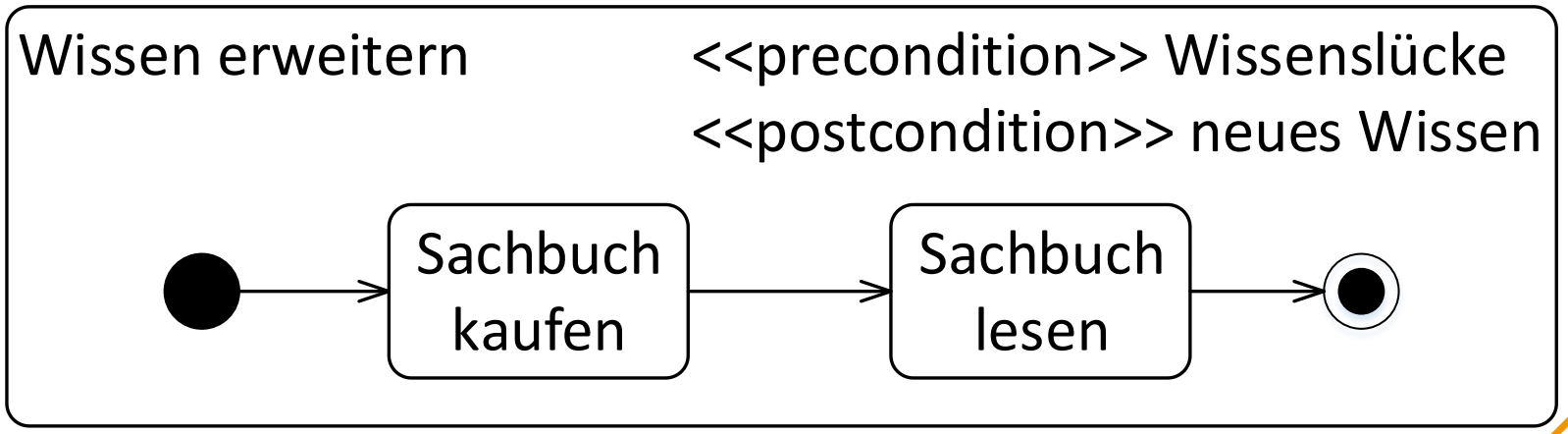
AD: Aktivitäten

- spezifizieren Verhalten eines Systembestandteils über Aktionen, z.B. häufig nutzerabhängiges Verhalten
- verschiedene Abstraktionsgrade
  - Beschreibung von Anwendungsfällen
  - Modellierung von vielen Details: Verhalten einer Operation
  - Wenige Details: Funktionen eines Geschäftsprozesses
- Darstellung als Rechteck mit abgerundeten Ecken
- Name der Aktivität am oberen Rand



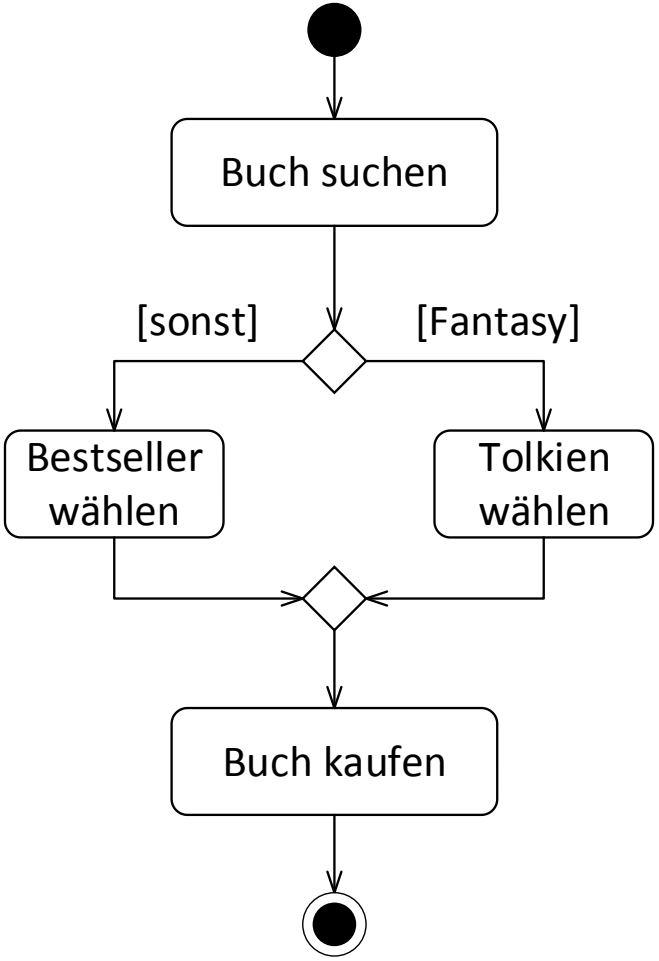
AD: Aktivität mit Vor-Nachbedingungen

- Aktivitäten können Vor- und Nachbedingungen haben, welche vor, bzw. nach der Ausführung einer Aktivität erfüllt sein müssen
- Darstellung mit Schlüsselwort in Guillemets <<Schlüsselwort>> Bedingung
  - <<precondition>> für Vorbedingungen
  - <<postcondition>> für Nachbedingungen
- Am oberen rechten Rand der Aktivität



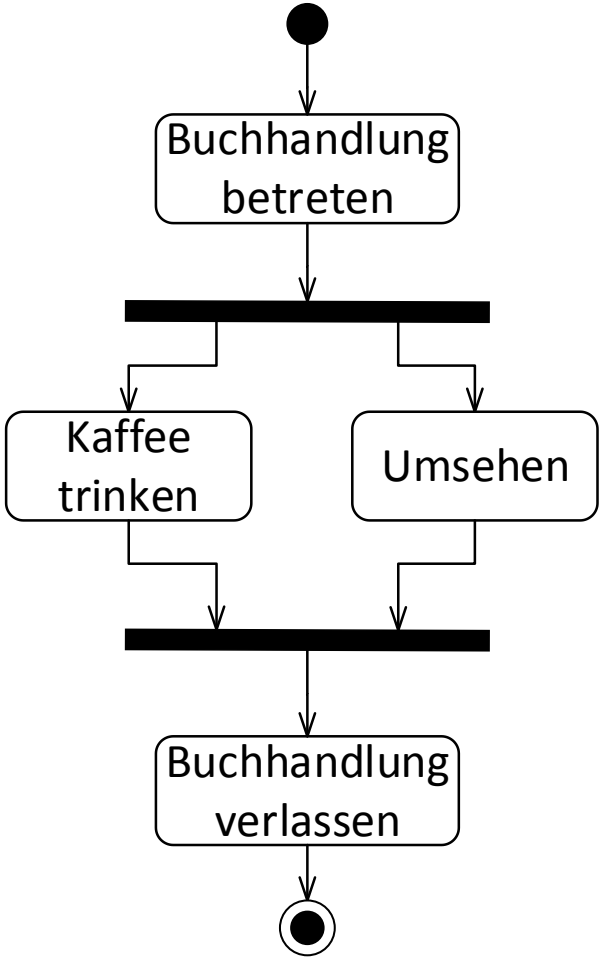
AD: Bedingte Verzweigung

- Bedingte Verzweigungen modellieren Alternativen
- Genau 1 Alternative wird ausgeführt
- Darstellung als Diamant/Raute mit ein- und ausgehenden Kanten
- Ausgehende Kanten stellen Alternativen dar
- Sollten mehrere Alternativen möglich sein, wird zufällig eine davon gewählt
- Alle Verzweigungen müssen wieder zusammengeführt werden, Darstellung auch als Diamant



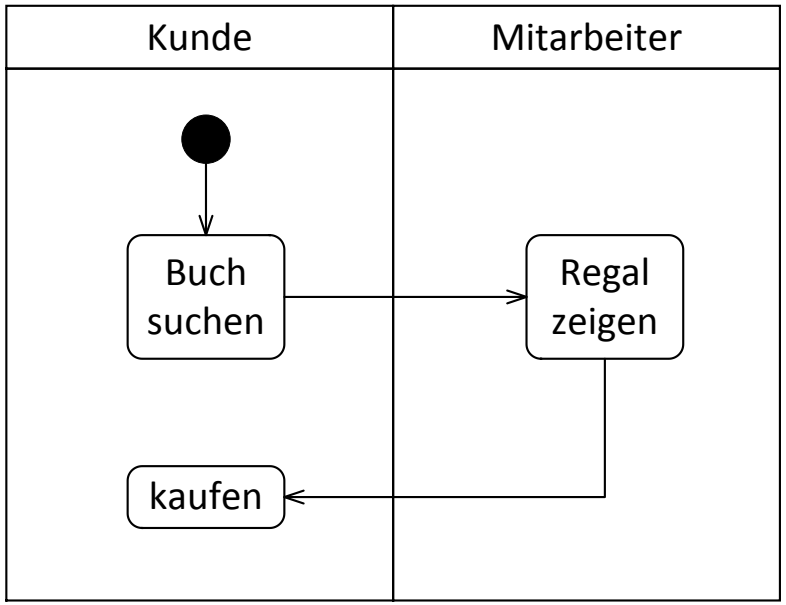
AD: Parallelisierung/Synchronisierung

- Mehrere Flüsse werden parallel ausgeführt im Vergleich zur bedingten Verzweigung, bei welcher genau 1 Alternative ausgeführt wird
- Darstellung als farbiger/schwarzer Balken
- Parallelisierung hat genau 1 Eingangskante und mindestens 2 Ausgangskanten, parallelisiert Flüsse
- Synchronisierung hat mindestens 2 Eingangskanten und genau 1 Ausgangskante, führt parallele Flüsse zusammen
- Erst wenn alle parallele Flüsse fertig sind (hier Umsehen und Kaffee trinken) wird Fluss weitergeleitet

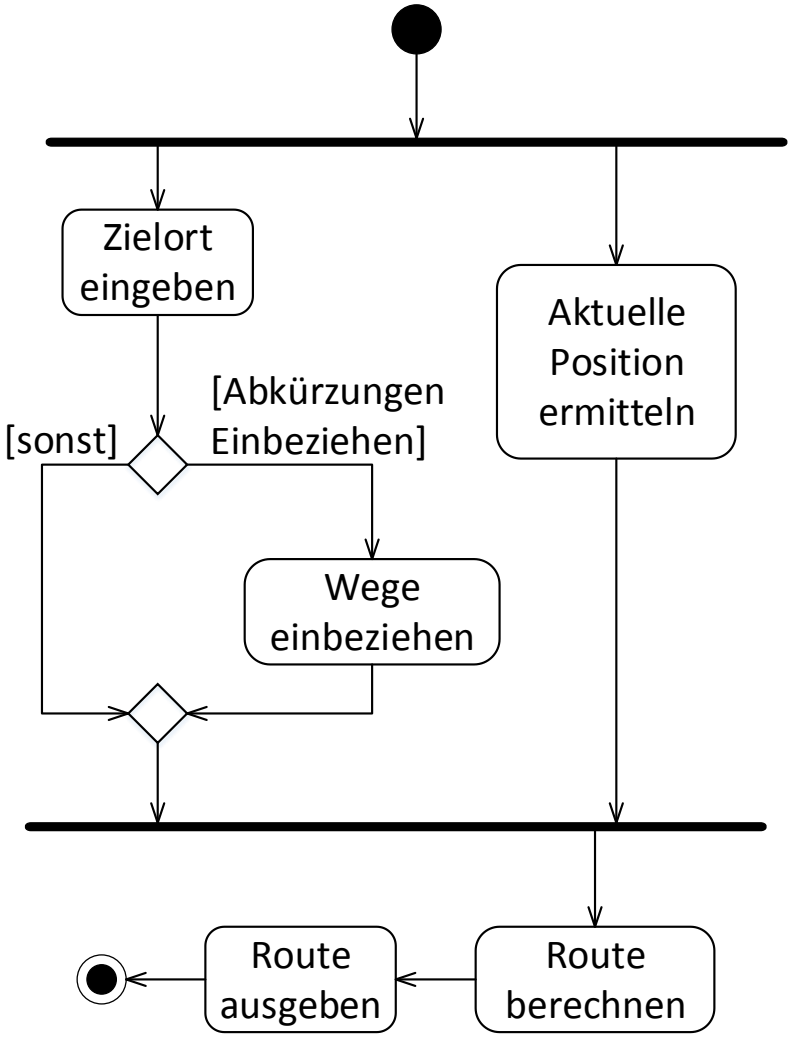


Aktivitätsbereiche/Swimlanes

- Gruppierung von Aktionen zu Organisationseinheiten
- Anordnung vertikal und/oder horizontal
- Darstellung als Rechteck mit dem Namen oben (oder links)
- Aktivitäten werden in Bereich der Zuständigkeit platziert



AD: Beispiel


Zielort eingeben
Aktuelle Position ermitteln
[Abkürzungen Einbeziehen]
[sonst]
Wege einbeziehen
Route ausgeben
Route berechnen

Zustandsdiagramm

Verhaltensdiagramm

Zustandsdiagramm (ZD) (Statechart): Einführung

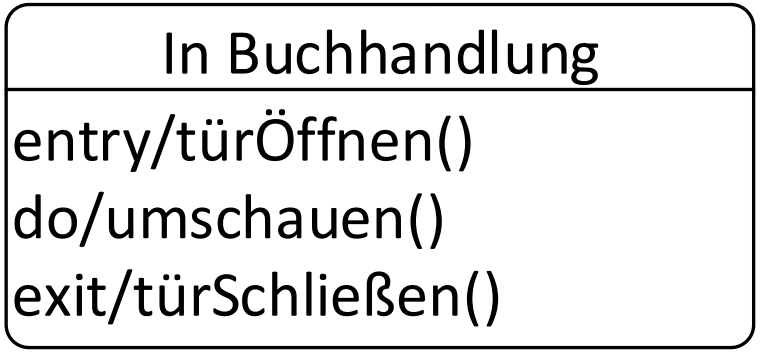
- Beschreibt von außen sichtbaren Zustände eines Systems, d.h. die Zustände, welche das System während der Ausführung annehmen kann
- Basiert auf deterministischen, endlichen Automaten
- Hauptsächlich verwendete Elemente:
  - Start- und Endzustände
  - Terminalknoten
  - Zustände mit eindeutigen Namen
  - Zustandsübergänge = Transitionen

ZD: Zustände

- Zustand modelliert Situation, in der genau definierte Bedingungen gelten
- Zustand kann verschiedene interne Aktivitäten haben, welche zu verschiedenen Zeitpunkten ausgeführt werden
- Anwendung: Ein Zustand hat viele eingehende Transitionen, welche einen Effekt bei der Ausführung auslösen, dann sinnvoller den Effekt beim Betreten des Zustandes als Aktion auszuführen
- Syntax: Schlüsselwort/Aktivität()
  - Entry: Aktivität wird ausgeführt sobald das Objekt/System den Zustand betritt
  - Do: Aktivität wird ausgeführt während das Objekt/System in dem Zustand ist
  - Exit: Aktivität wird ausgeführt sobald das Objekt/System den Zustand verlässt
- Die Aktivitäten an sich werden in anderen Diagrammen beschrieben

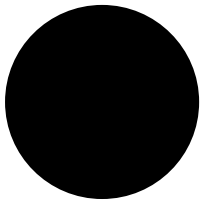
ZD: Beispiel interner Aktivitäten

- Beispiel des Zustands, dass man sich in einer Buchhandlung befindet
- Bei, Eintreten öffnet sich die Tür
- Während Besuchs schaut man sich um
- Beim Verlassen wird die Tür geschlossen

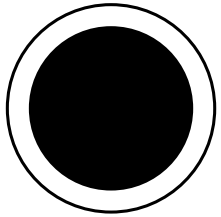


ZD: Start- und Endzustände

- Startzustand: Gefüllter Kreis, nur ausgehende Kanten, Startpunkt der Ausführung
- Endzustand: Gefüllter Kreis mit weißem Kreis darum, nur eingehende Kanten, Ausführung eines Pfades endet hier
- Terminalknoten: Ein x, Fluss endet dort und Zustandsmaschine endet vollständig



Startzustand



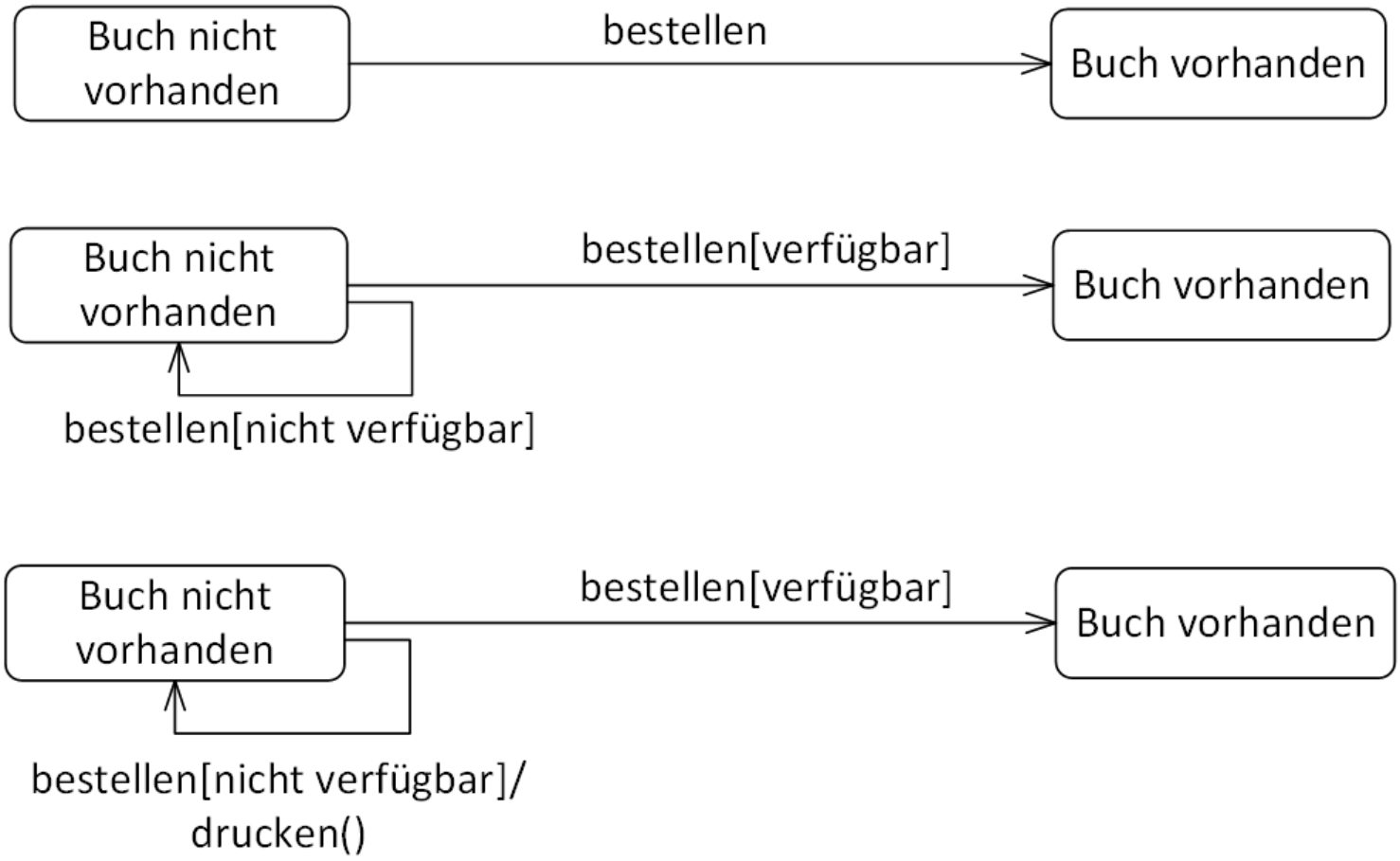
Endzustand

Terminator

ZD: Transitionen

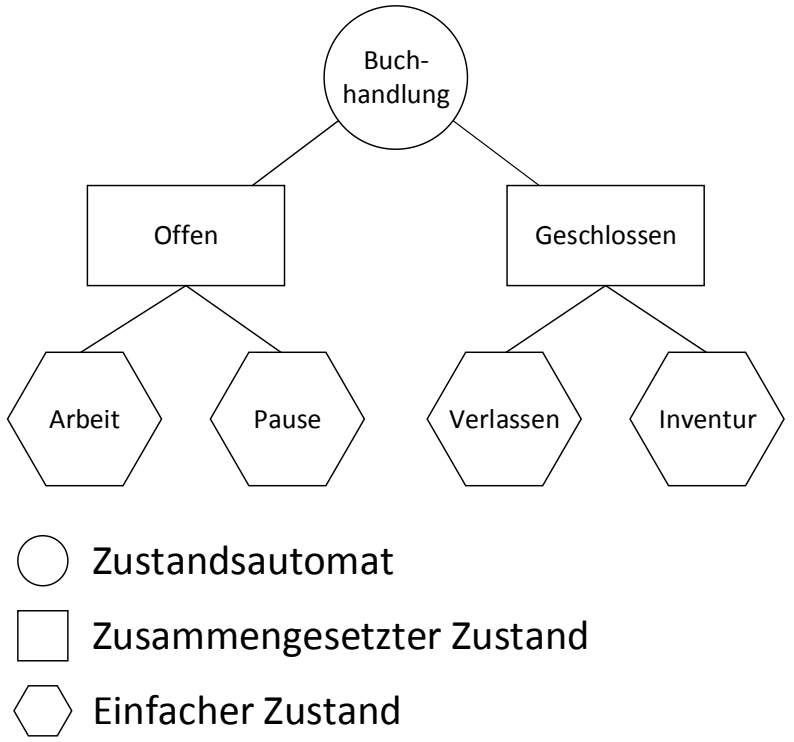
- Zustandsübergänge werden Transitionen genannt
- Darstellung als Linie mit offenem Pfeil
- Keine Beschriftung ε-Transition): die Transition kann ausgeführt werden, sobald ein Zustand es zulässt (z.B. interne Aktivitäten, welche beendet sein müssen)
- Können verschiedene Eigenschaften haben
  - Event: Stimuli, welche den Übergang von außerhalb des Systems auslösen
  - Guard: Bedingung, welche als „true“ ausgewertet werden muss, um das Event auszuführen
  - Aktivitäten: Aktionen, welche bei Schaltung der Transition ausgeführt werden

ZD: Transitionen Beispiel



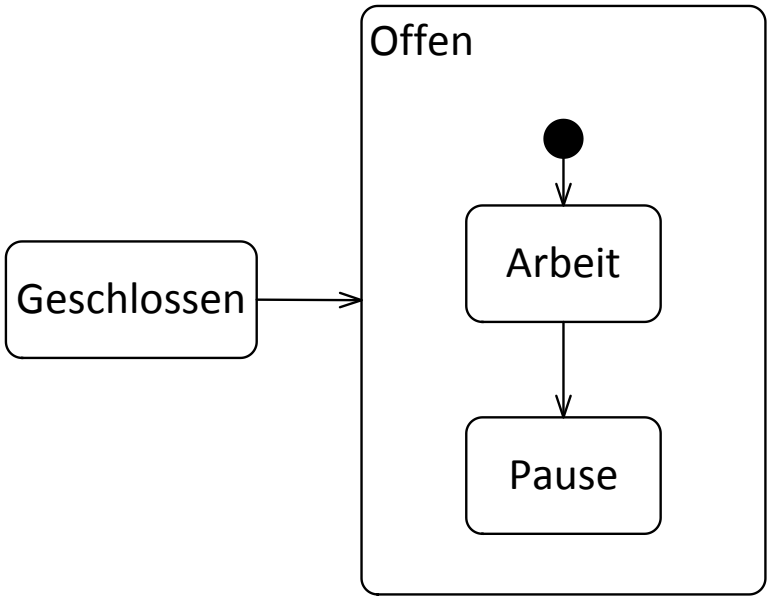
ZD: Zusammengesetzter Zustand (Composite State)

- Modelliert Hierarchien von Zuständen
- Ermöglicht z.B. gleichzeitiges Modellieren verschiedener Abstraktionsebenen eines Systems
- Zustand kann mehrere Unterzustände enthalten
- Kann zusätzlich Start-/Endzustände enthalten, jedoch nicht zwingend erforderlich



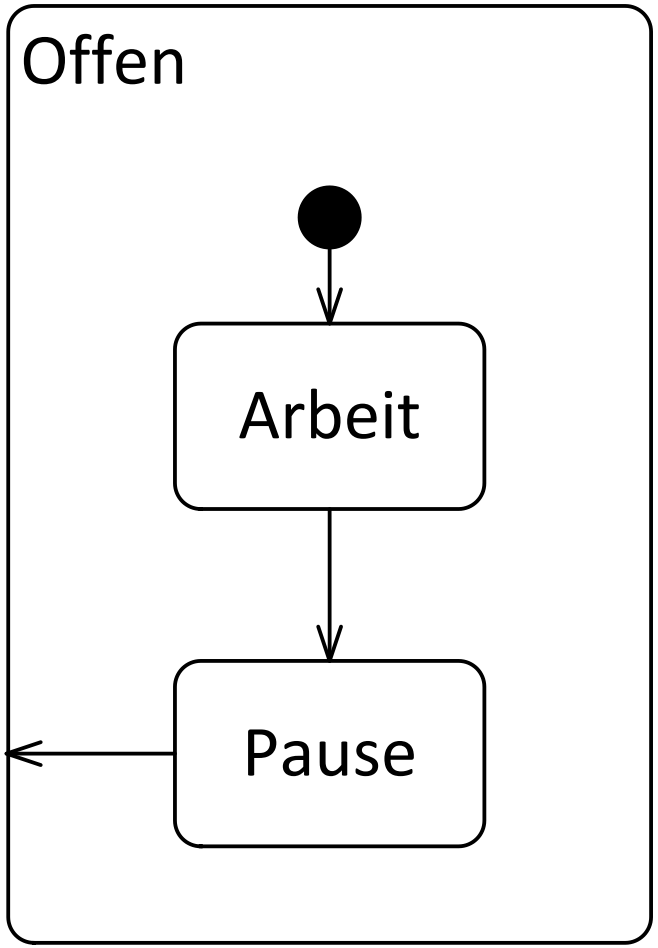
ZD: Betreten zusammengesetzter Zustände: Standard-Eintritt von „außen“

- Zusammengesetzter oder komplexer Zustand(ZZ): Zustand, welcher wieder Subzustände enthält
- Transition von außerhalb auf Rand des ZZ
  - Startzustand des ZZ wird erreicht



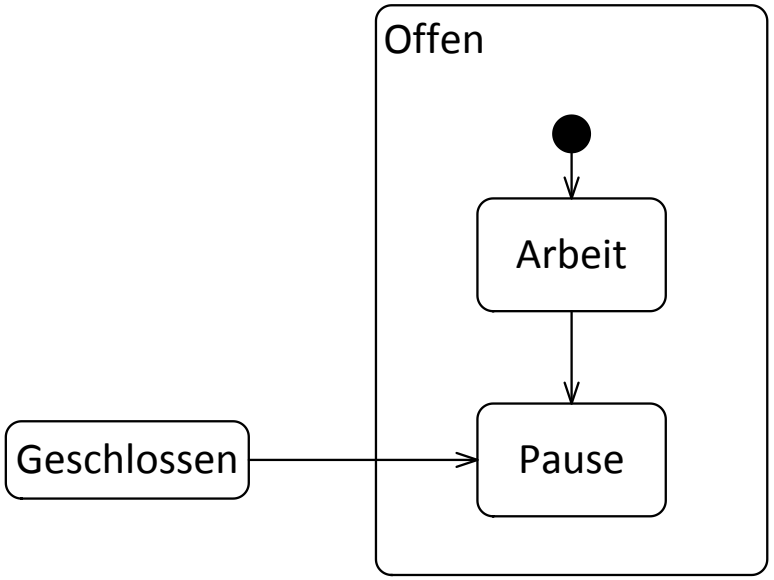
ZD: Betreten zusammengesetzter Zustände: Standard-Eintritt von „innen“

- Transition von einem Unterzustand innerhalb des ZZ auf Rand des ZZ
- Startzustand des ZZ wird erreicht
- ZZ wird dabei nicht verlassen



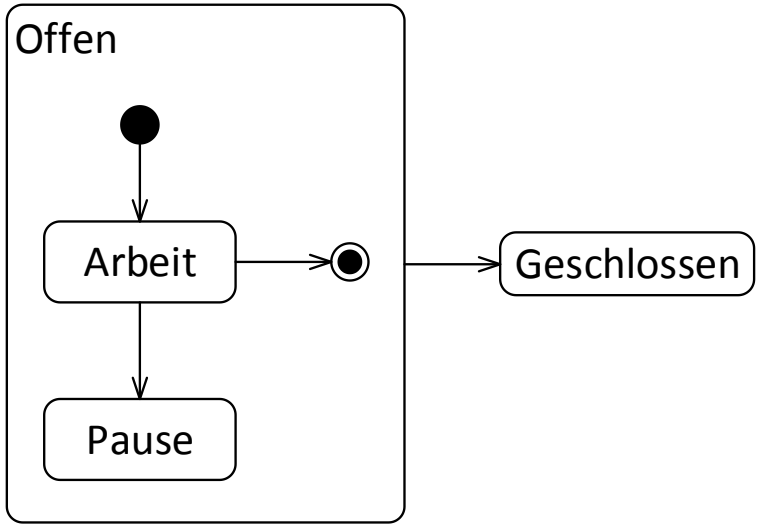
ZD: Betreten zusammengesetzter Zustände: Expliziter Eintritt

- Transition von außerhalb direkt auf einen Unterzustand innerhalb des ZZ
- Dieser Zustand innerhalb des ZZ wird aktiv
- Im Beispiel: Zustand Pause



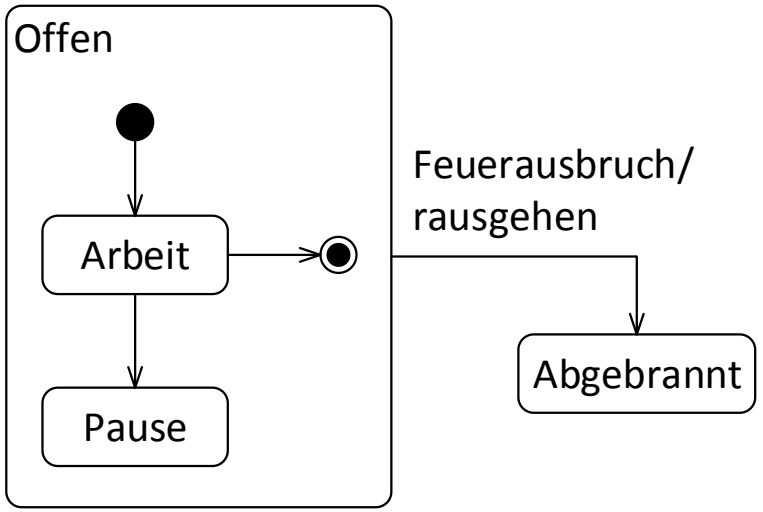
ZD: Verlassen komplexer Zustände beim Erreichen des Endzustands

- ε-Transition vom Rand des Zustands „Offen“ zum Zustand „Geschlossen“
- Keine Bedingungen an Transition
- Muss modelliert sein



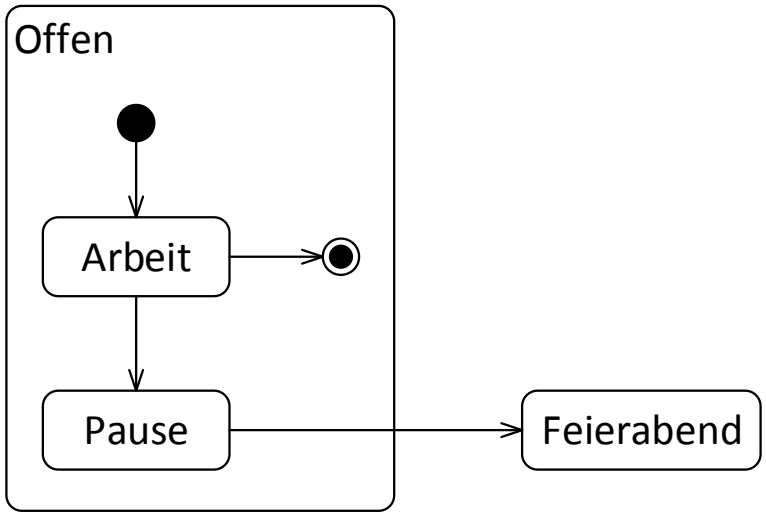
ZD: Verlassen komplexer Zustände mittels Event

- Transition vom Rand des zusammengesetzten Zustands „Offen“ zum Zustand „Abgebrannt“ mittels Event
- Sobald Event eintritt kann Transition von jedem Unterzustand ausgeführt werden



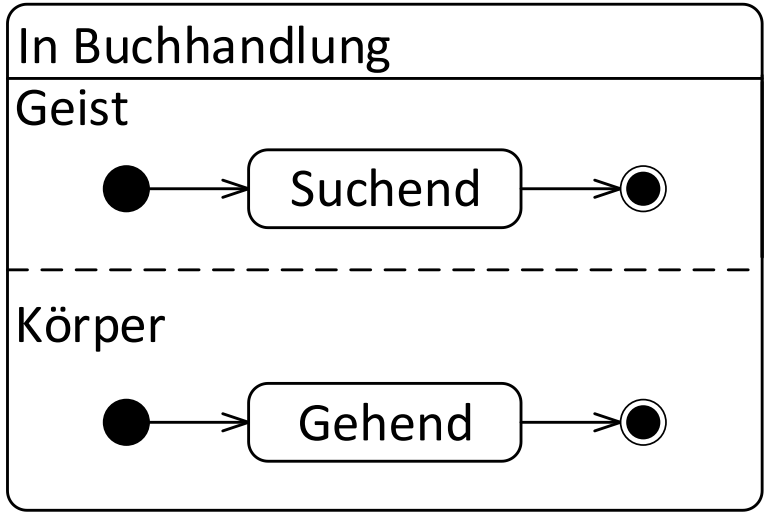
ZD: Verlassen komplexer Zustände durch Transition eines Unterzustands

- Transition von Unterzustand „Pause“ zu Zustand „Feierabend“ außerhalb von „Offen“
- Zusammengesetzter Zustand wird verlassen und Zustand außerhalb wird aktiv

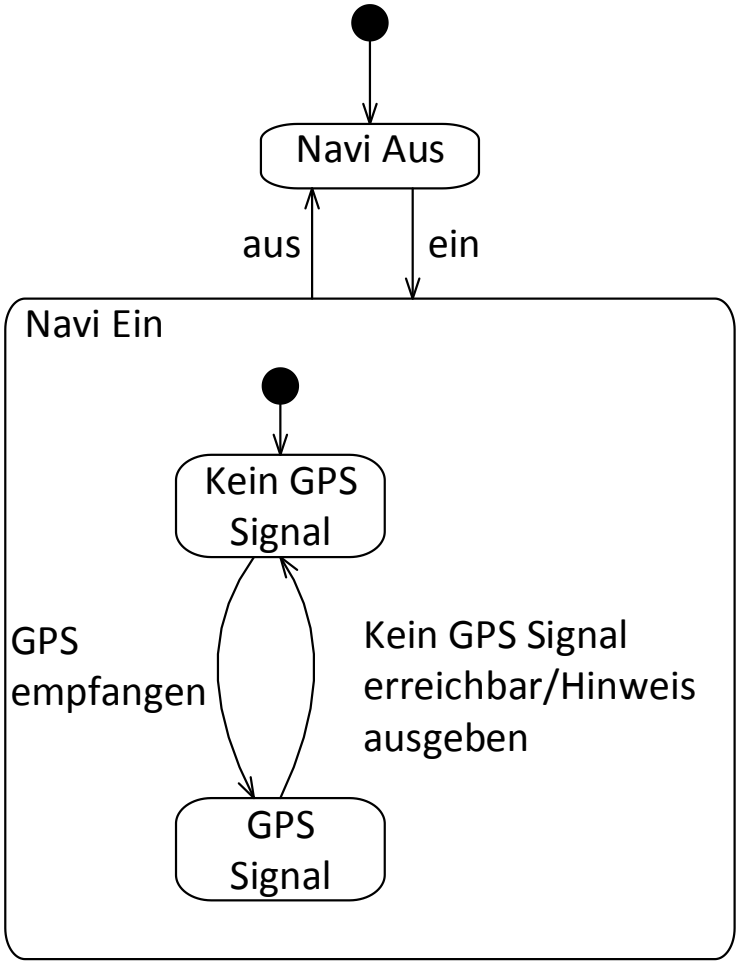


ZD: Orthogonaler Zustand

- In zusammengesetzten Zuständen immer nur 1 Zustand aktiv
- In orthogonalen Zuständen werden Zustände parallel verarbeitet
- Können Start-/Endpunkte haben
- Transition auf Subzustand oder auf Begrenzung des orthogonalen Zustands möglich



ZD: Beispiel


Navi Aus
aus
ein
Navi Ein
Kein GPS Signal
GPS empfangen
Kein GPS Signal erreichbar/Hinweis ausgeben
GPS Signal

Sequenzdiagramm

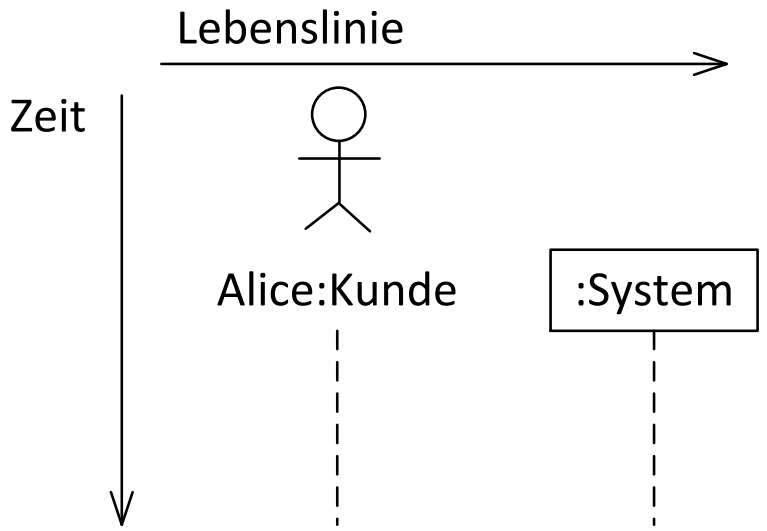
Verhaltensdiagramm

Sequenzdiagramm (SQ): Einleitung

- Unterkategorie: Interaktionsdiagramme
- Beschreibt Nachrichtenfluss zwischen Teilnehmern
- Zeigen zeitlichen Ablauf
- Verdeutlichen konkret möglichen Ablauf, z.B.
  - Klassen und ihre Methodenaufrufe
  - Präzisierung des Aktivitätsdiagramms
  - Zusammenspiel von Akteuren und Anwendungsfällen

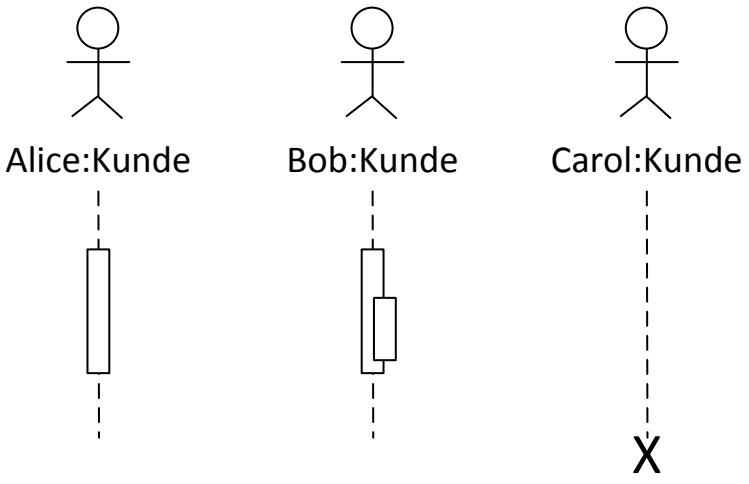
SQ: Interaktions- und Zeitachse

- Teilnehmer werden horizontal angeordnet
  - Menschen als Strichmännchen
  - Systeme als Rechteck
  - Bezeichnung: Name:Type
  - Typangabe ist notwendig
  - Name ist optional
- Lebenslinie (gestrichelt) verläuft vertikal unter Teilnehmern



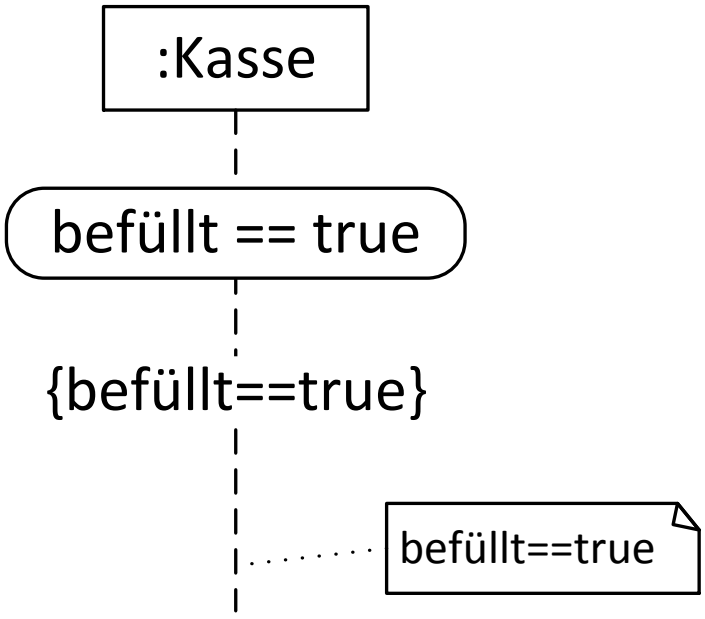
SQ: Aktivitätsbereiche

- Aktive Bereiche der Teilnehmer werden durch vertikale Balken dargestellt
- Doppelte Balken stellen parallele Tätigkeiten dar
- Ein „X“ kennzeichnet die Zerstörung der Lebenslinie



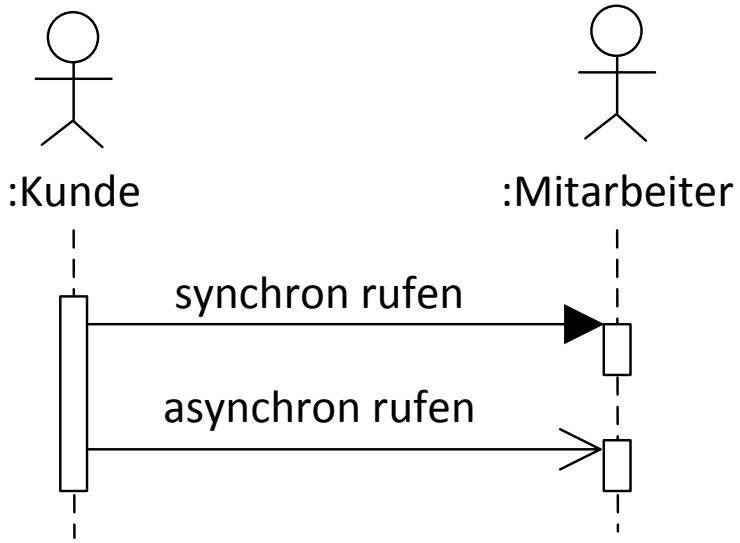
SQ: Zustandsinvariante

- Bedingung eines Teilnehmers, um an Interaktion teilnehmen zu können
- Bedingung wird vor nächster Interaktion geprüft
- Verschiedene Darstellungsalternativen
  - Abgerundetes Rechteck auf Lebenslinie
  - In geschweiften Klammern auf Lebenslinie
  - Als Notiz an Lebenslinie



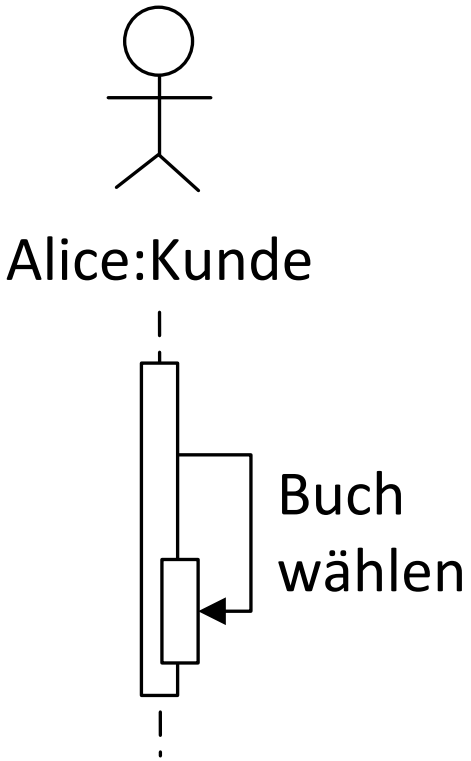
SQ: Nachrichten

- Kommunikation zwischen Teilnehmern
- Gerichtete Pfeile zwischen Aktivitäts-bereichen von Teilnehmern mit Namen
- Synchron
  - Gefüllter Pfeil
  - Sender wartet auf Antwort, blockiert Tätigkeit
- Asynchron
  - Offener Pfeil
  - Fortführen der Tätigkeit



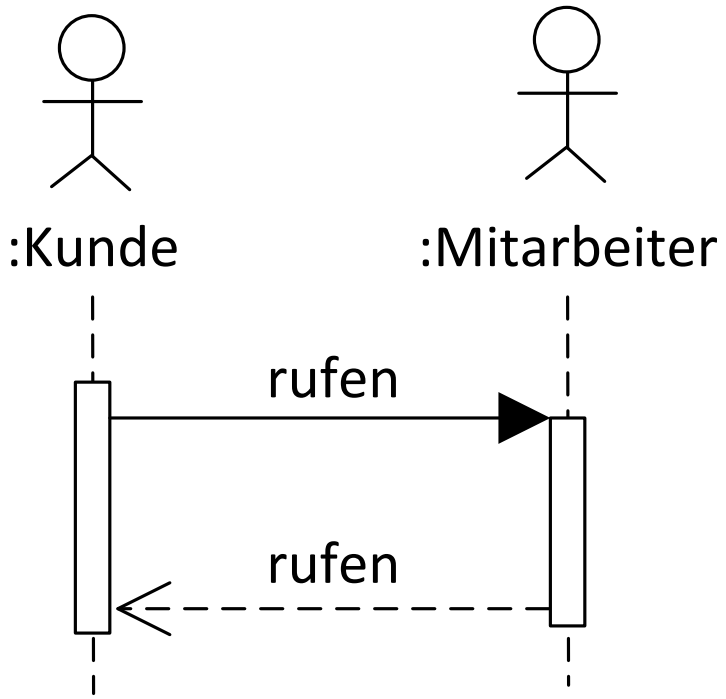
SQ: Reflexive Nachrichten

- Nachrichten müssen nicht auf andere Teilnehmer gerichtet sein
- Nachrichten an sich selbst sind möglich
- Darstellung als gefüllter Pfeil auf parallelen Aktivitätsbereich



SQ: Antwortnachrichten

- Nachrichten können Antworten nach sich ziehen, vor allem synchrone
- Antwort ist offener, gestrichelter Pfeil
- Antwort enthält mindestens Namen der Originalnachricht zur Zuordnung



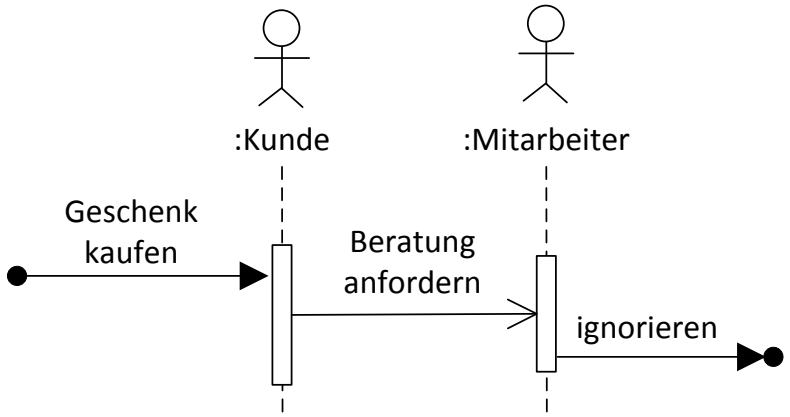
SQ: Gefundene und Verlorene Nachrichten

Gefundene Nachricht:

- Kein Sender, oder irrelevant
- Ausgehend aus Kreis

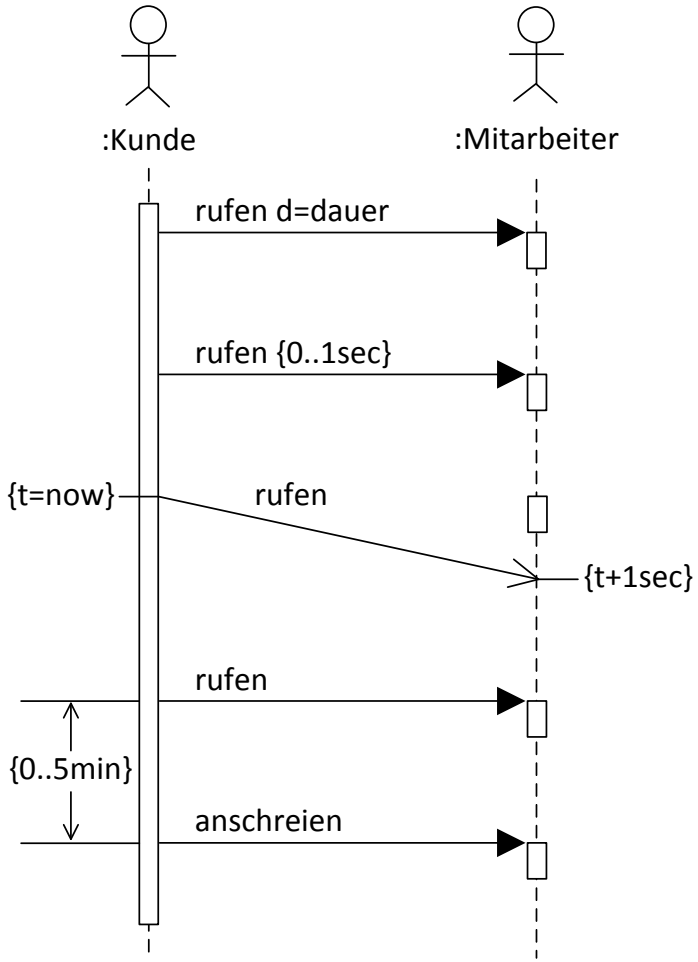
Verlorene Nachricht

- Kein Empfänger oder irrelevant
- Pfeil endet in Kreis



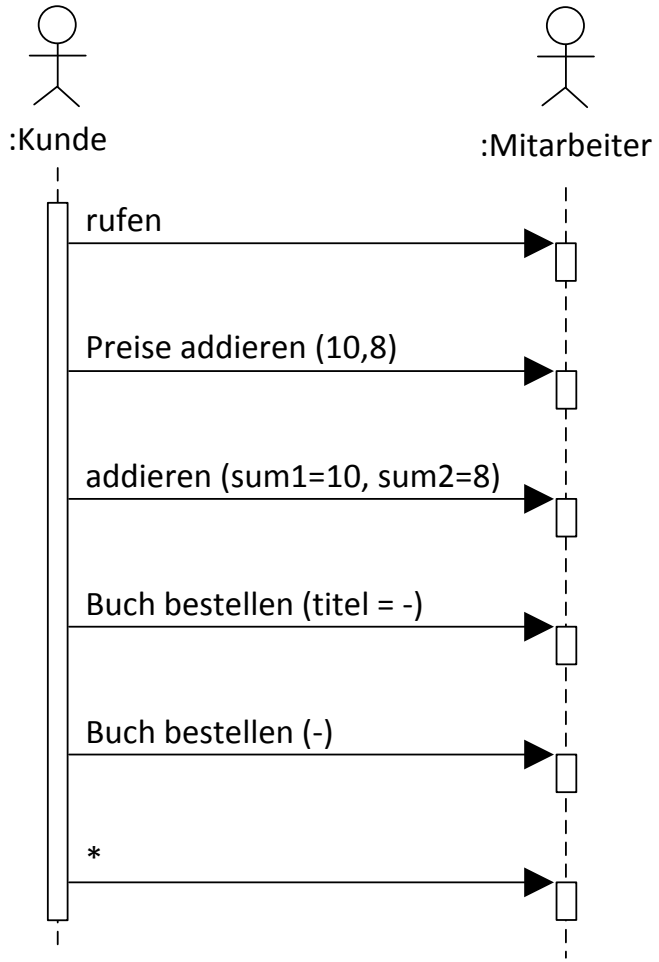
SQ: Zeitbedingungen bei Nachrichten

- Per Default ist die Dauer einer Nachricht Null
- Zeitbedingungen einzelner Nachrichten oder zwischen Nachrichten ist jedoch möglich
- 4 Varianten sind rechts abgebildet



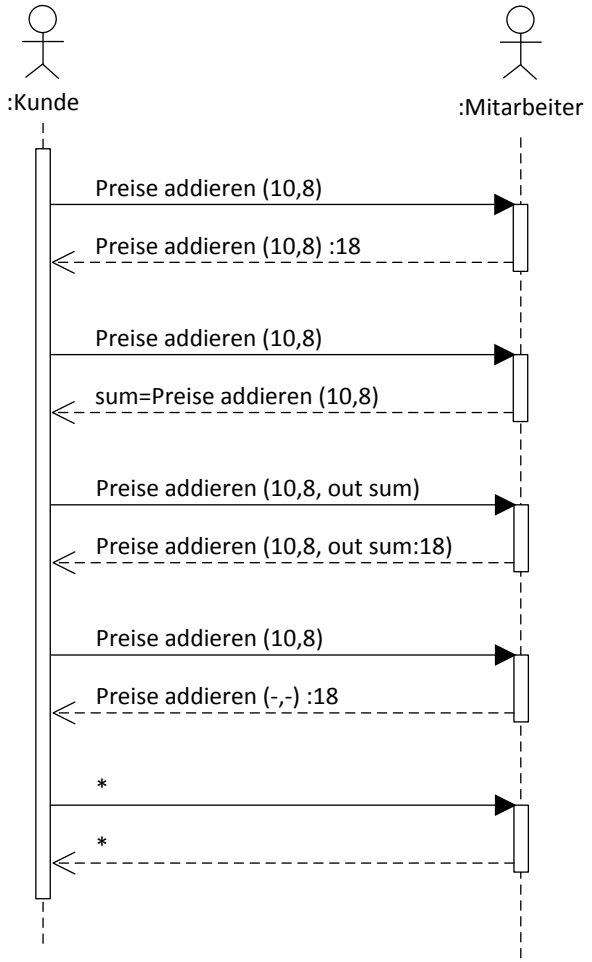
SQ: Argumentübergabe von Nachrichten

- Nachrichten können optionale Argumente und Parameter enthalten
- Name[(Argument, Argument,…)]
- Argument = [Parametername=] Argumentwert | - | *
- [] bedeutet optional



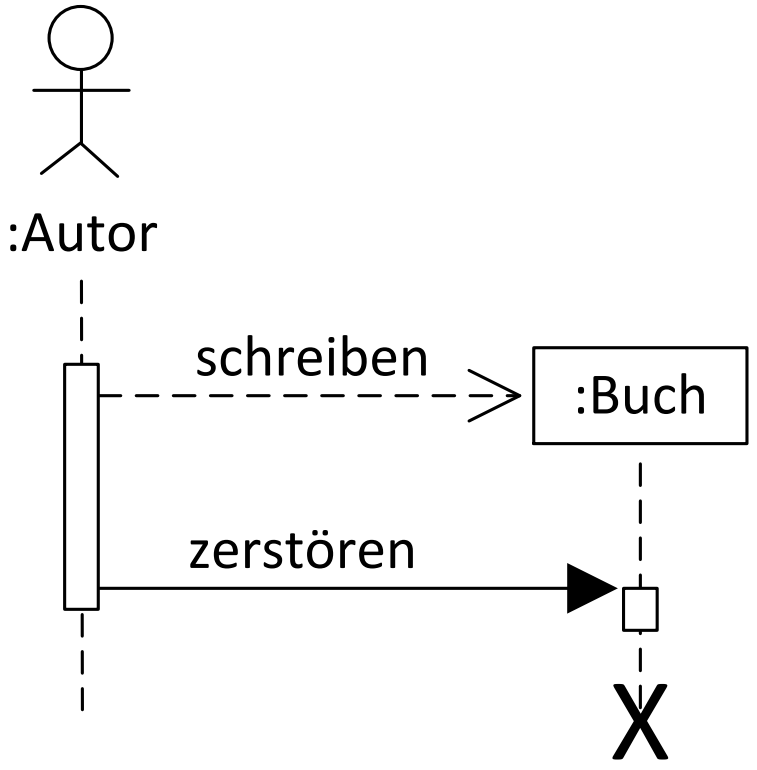
SQ: Argumente von Antwortnachrichten

- Antwortnachrichten wiederholen den Namen und können optional zusätzlich Rückgabewerte besitzen
  - Attributzuweisung
  - Zuweisung eines out-Parameters
  - Unspezifizierter Parameter



SQ: Erzeugen und Zerstören von Teilnehmern

- Erzeugen und Zerstören von Teilnehmern während der Interaktion
- Kopf des neuen Teilnehmers ist auf Höhe der Erzeugungsnachricht
- Lebenslinie endet direkt nach der Zerstörungsnachricht, markiert durch ein „X“

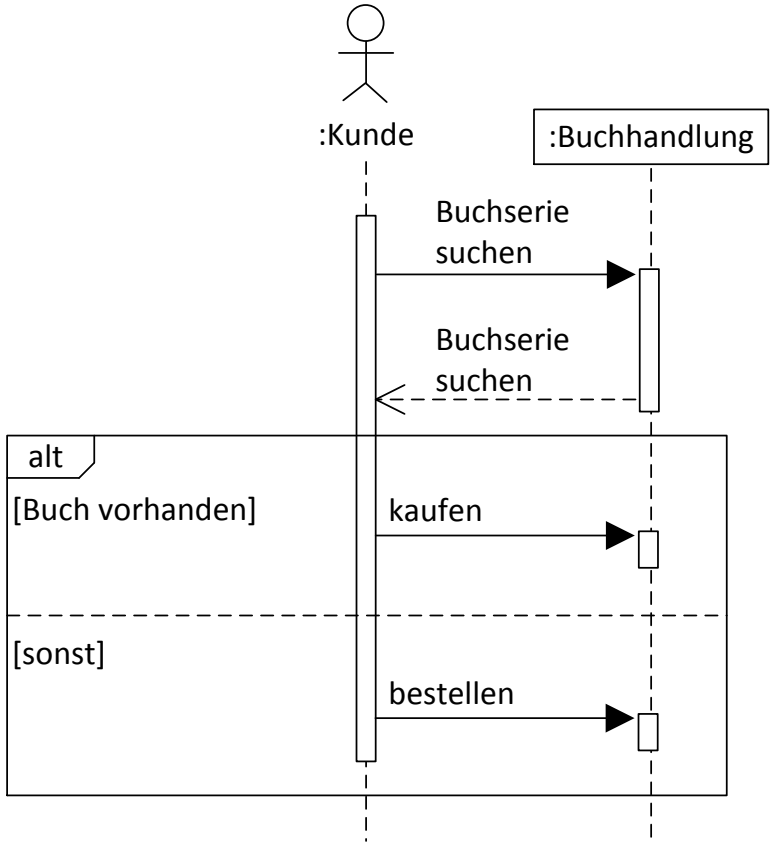


SQ: Kombinierte Fragmente

- Modellieren Kontrollstrukturen in Sequenzdiagrammen, z.B. Alternativen, Parallelität, Optionale Nachrichten, et.
- Darstellung des kombinierten Fragments als Rechteck mit dem Typ (=Interaktions-Operator) in einem Pentagon oben links
- Fragmente bestehen aus 1..n Operanden (Bereiche), welche durch waagerecht gestrichelte Linien abgegrenzt werden
- Fragmente können ineinander geschachtelt werden

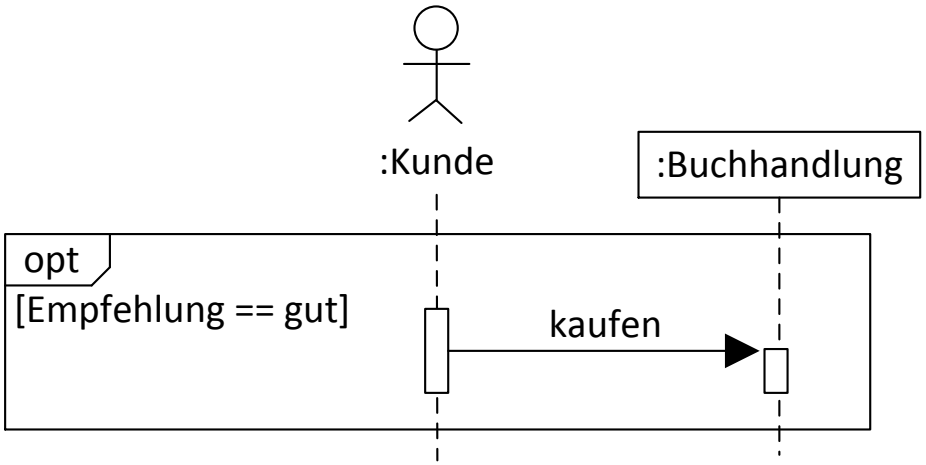
SQ: Alternative Ausführung

- Operator: alt
- Entspricht „if-then-else“ Konstrukt
- Bedingungen werden oben links in Operand notiert
- Operanden müssen disjunkt sein und alle möglichen Fälle abdecken



SQ: Optionale Ausführung

- Operator: opt
- Optionale Ausführung, falls spezifizierte Bedingung wahr ist
- Bedingung wird oben links unter das Pentagon notiert



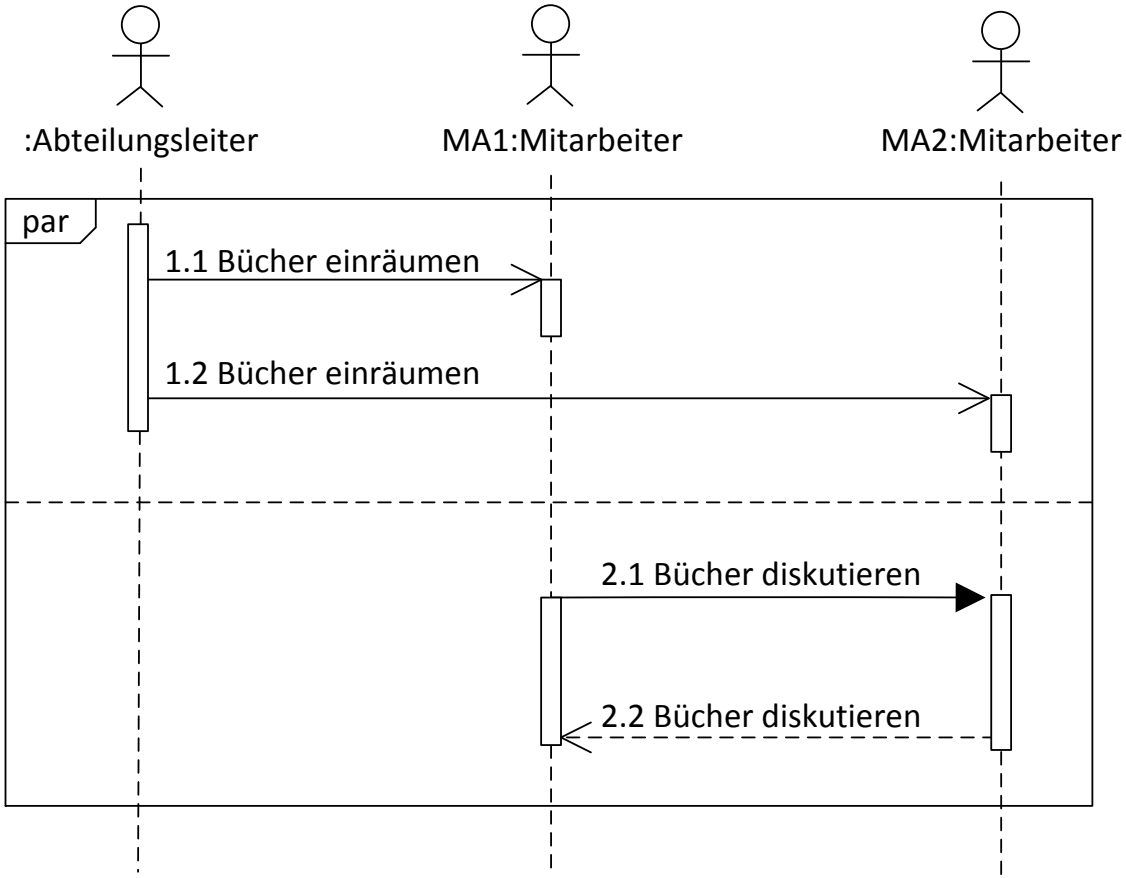
SQ: Parallele Ausführung

- Operator: par
- Parallele Ausführung von Operanden
- Abgrenzung der einzelnen Operanden durch gestrichelte Linien.
- Beliebige Anzahl an Operanden möglich
- Ausführung der Nachrichten der einzelnen Operanden in beliebiger Reihenfolge möglich, innerhalb dieser jedoch vorgegeben

SQ: Beispiel paralleler Ausführung

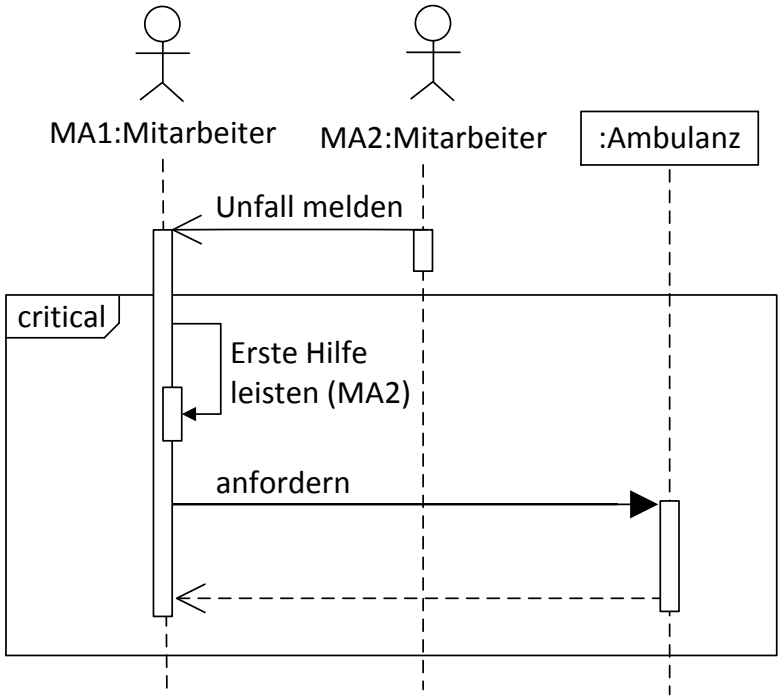
Mögliche Pfade:

1.1 → 2.1 → 1.2 → 2.2
1.1 → 2.1 → 2.2 → 1.2
1.1 → 1.2 → 2.1 → 2.2
2.1 → 2.2 → 1.1 → 1.2
2.1 → 1.1 → 2.2 → 1.2
2.1 → 1.1 → 1.2 → 2.2



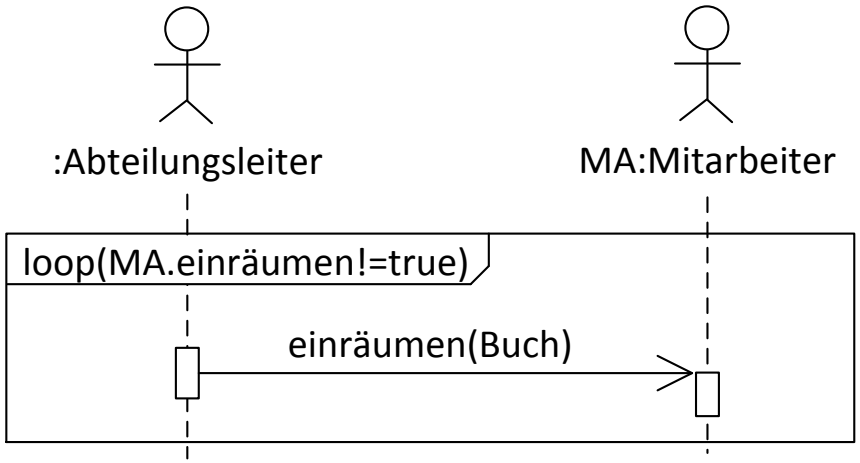
SQ: Kritischer Bereich

- Operator: critical
- Nur ein Operand
- Ausführung der enthaltenen Nachrichten darf niemals unterbrochen werden
- Strikte Reihenfolge kann nicht durch andere Fragmente aufgehoben werden



SQ: Wiederholte Ausführung: Schleifen

- Operator: loop
- Optionale Angabe von (min,max) oder (Bedingung)
  - loop(8): genau 8 Wiederholungen
  - Loop(2,5): mindestens 2, maximal 5 Wiederholungen
  - Loop oder loop(*): beliebige Anzahl Wiederholungen



SQ: Navigationsbeispiel