Unified Modeling Language

Christian Silberbauer

Einführung

- UML ist eine Modellierungssprache.
- UML ist sehr umfangreich und komplex.
- Sie spezifiziert eine Notation und kein Vorgehen.
- UML-Diagramme haben nicht den Zweck, den Code vollständig abzubilden. Sie dienen der Architekturkommunikation.
- Eine nahtlose Toolintegration ist wünschenswert.

Historie

- UML 1 wurde durch die Zusammenführung dreier Modellierungssprachen begründet, nämlich:
  - Booch: Grady Booch
  - OMT: Object-Modeling Technique: James Rumbaugh
  - OOSE (Object-Oriented Software Engineering): Ivar Jacobson
- Die drei sind alch als „Die drei Amigos“ bekannt.
- UML 1.1 wurde 1997 durch die OMG als Standard veröffentlicht.
- Die weitere Entwicklung wurde an die OMG übergeben.
- Version 2 bedeutete umfangreiche Erweiterung. Sie wurde 2005 veröffentlicht.
- U.A. wurde das Metamodell vereinheitlicht, womit MDA unterstützt wird.
- Aktuelle Version ist 2.5.1 vom Dezember 2017.

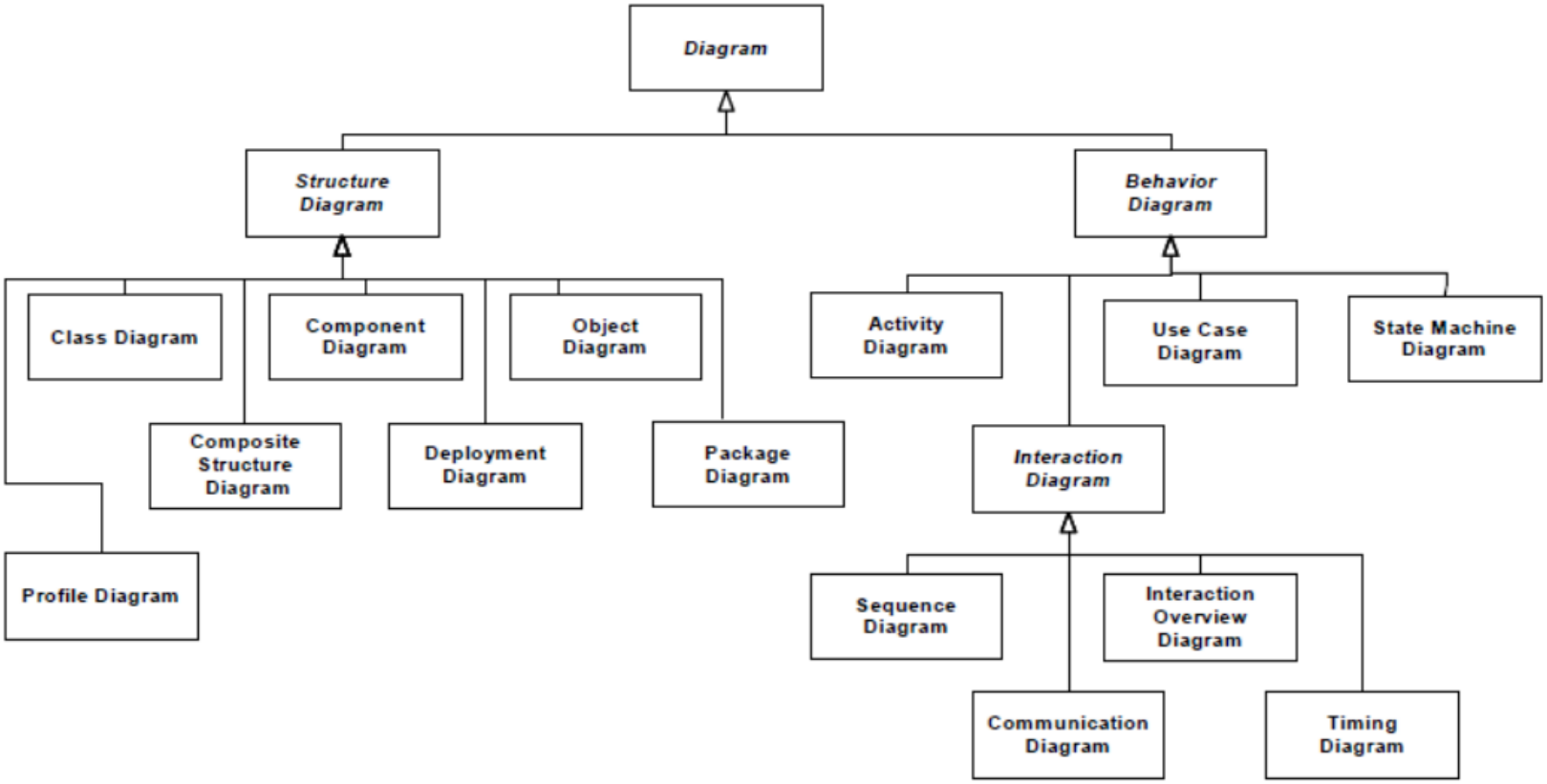
Warum UML?

- Es ist der eine Standard, auf dem sich viele geeinigt haben.

- Wiederholung:

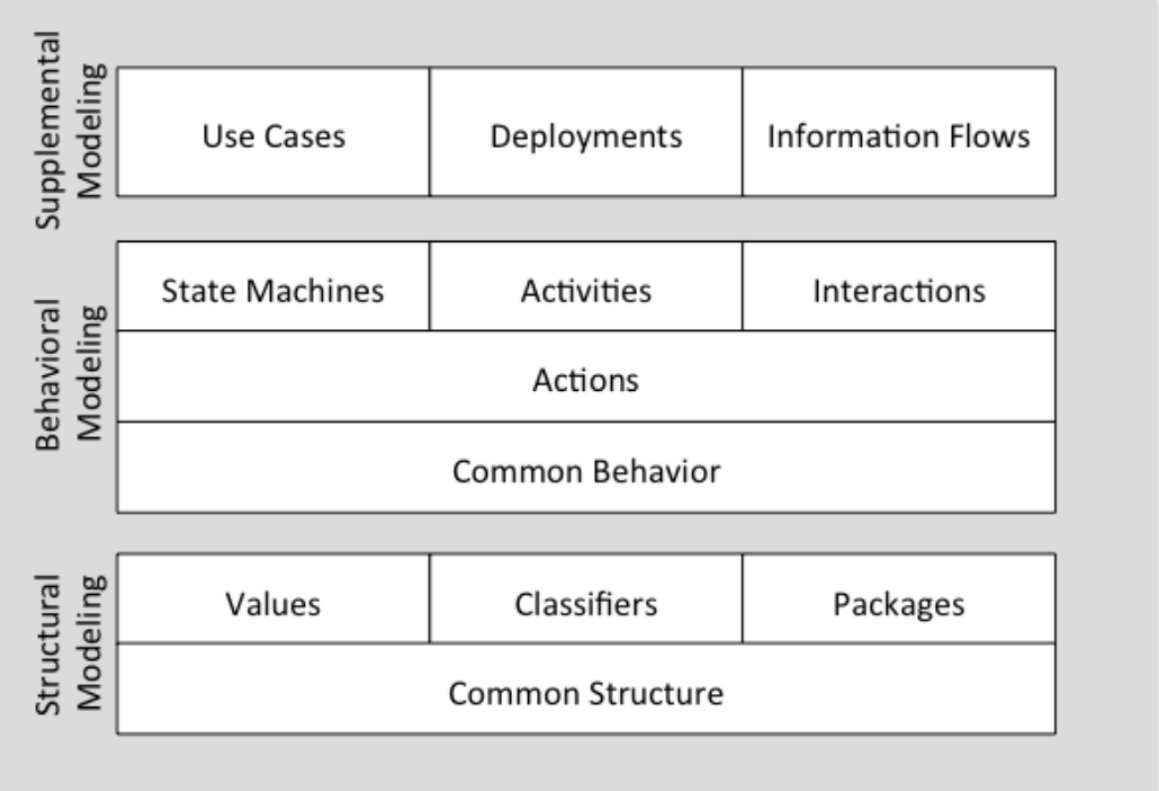
Merksatz: **Achte auf Standards!**

Diagrammtypen



- Quelle: OMG-Spec, S. 685

Semantische Bereiche



- Quelle: OMG-Spec, S. 14

Aspekte

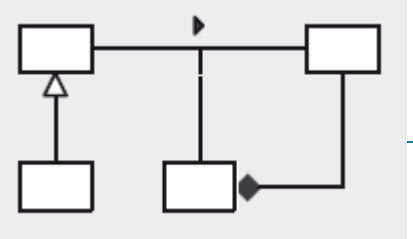
- Gruppierung: Knoten wie Klassen, Objekte, Use-Cases, Diagramme
- Linearisierung: Abläufe, insb. im Behavioral- und Supplemental Modeling
- Metaisierung: Feste Typmodell
- Beständigkeit?

Vorgestellte Diagrammtypen

- Strukturdiagramme:
  - Klassendiagramm
  - Paketdiagramm
  - Objektdiagramm
  - Kompositionsstrukturdiagramm
  - Komponentendiagramm
  - Verteilungsdiagramm
- Verhaltensdiagramme
  - Use-Case-Diagramm
  - Aktivitätsdiagramm
  - Zustandsautomat
  - Sequenzdiagramm
  - Kommunikationsdiagramm
  - Timing-Diagramm
  - Interaktionsübersichtsdiagramm

Alle folgenden Beispiele sind aus Rupp 2012 oder Balzert 2005 entnommen.

Klassendiagramm

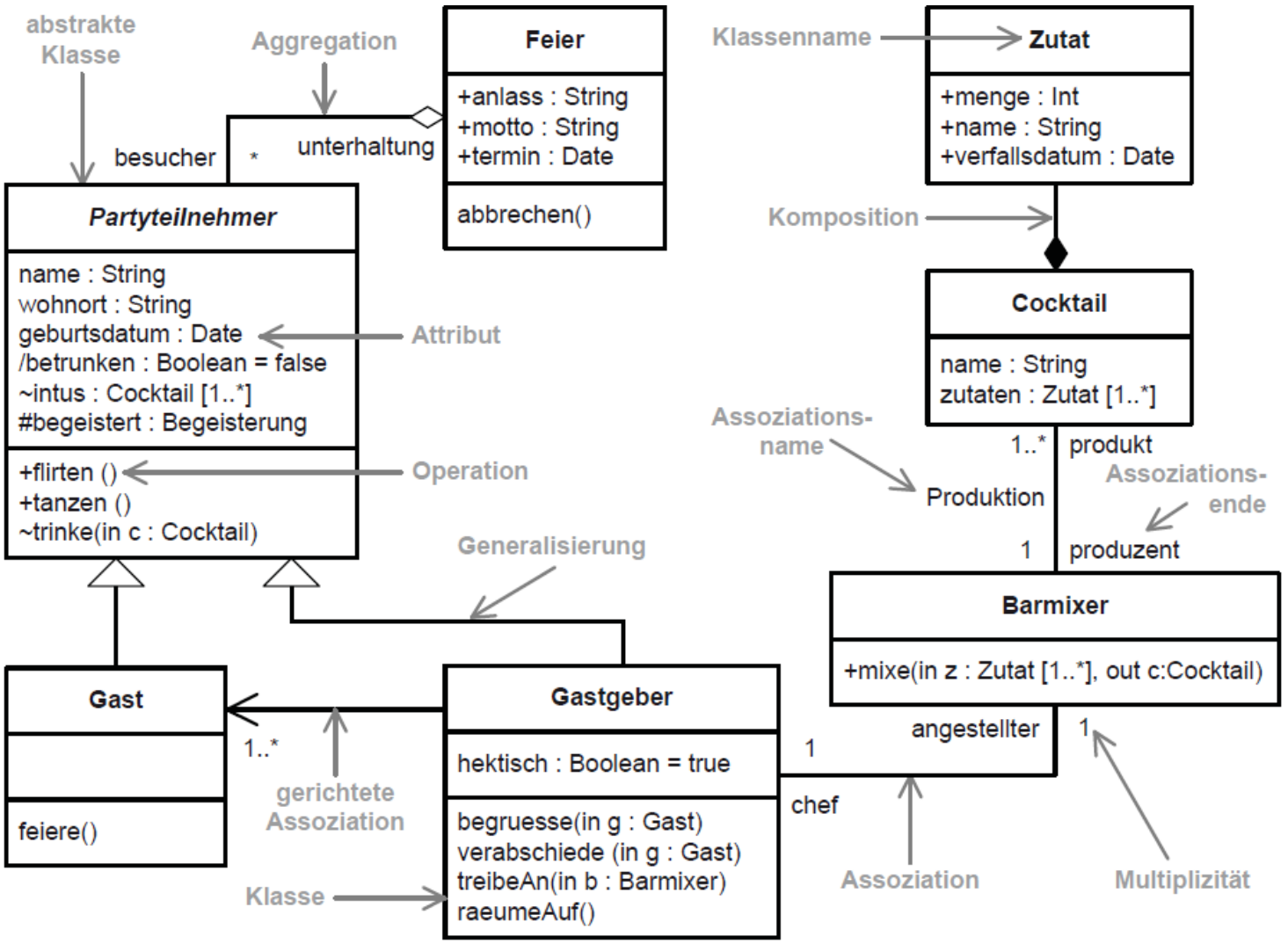


- Zentrale Frage:

  Aus welchen Klassen besteht mein System und wie stehen diese untereinander in Beziehung?

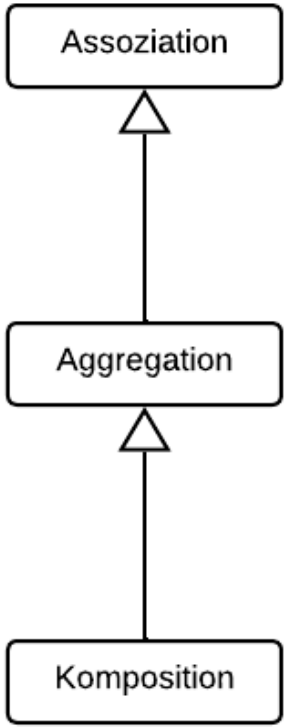
- Stärken:
  - Beschreibt die statische Struktur des zu entwerfenden oder abzubildenden Systems.
  - Enthält alle relevanten Strukturzusammenhänge und Datentypen.
  - Bildet die Brücke zu den dynamischen Diagrammen.

Beispiel


abstrakte Klasse
Aggregation
Feier
+anlass : String
+motto : String
+termin : Date
abbrechen()
besucher
*
unterhaltung
Partyteilnehmer
name : String
wohnort : String
geburtsdatum : Date
/betrunken : Boolean = false
~intus : Cocktail [1..*]
#begeistert : Begeisterung
+flirten ()
+tanzen ()
~trinke(in c : Cocktail)
Attribut
Operation
Generalisierung
Gast
feiere()
1..*
gerichtete Assoziation
Gastgeber
hektisch : Boolean = true
begruesse(in g : Gast)
verabschiede (in g : Gast)
treibeAn(in b : Barmixer)
raeumeAuf()
Klasse
Klassenname
Zutat
+menge : Int
+name : String
+verfallsdatum : Date
Komposition
Cocktail
name : String
zutaten : Zutat [1..*]
Assoziations-name
1..*
produkt
Produktion
Assoziations-ende
1
produzent
Barmixer
+mixe(in z : Zutat [1..*], out c:Cocktail)
angestellter
1
1
chef
Assoziation
Multiplizität

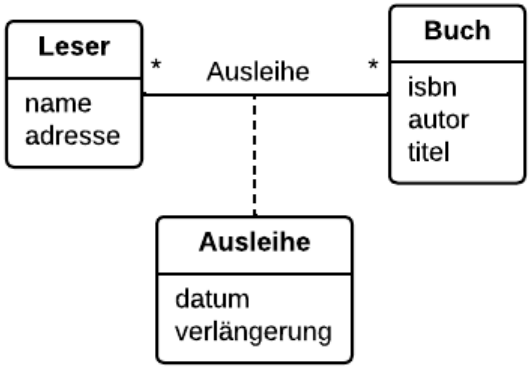
Assoziationen

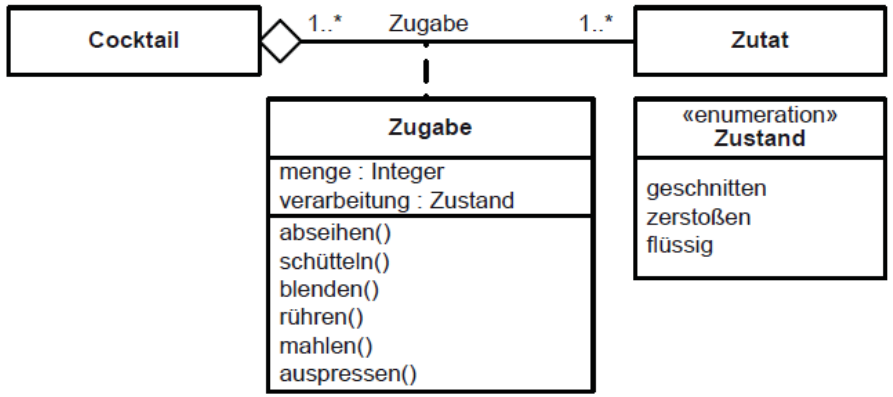
- **Assoziationen** definieren Beziehungen zwischen Elementen.
- Sie können mit Multiplizitäten angereichert werden.
- Pfeile geben ggf. die Zugriffsrichtung an.
- Werden Assoziation angegeben, wird auf die Aufführung entsprechender Attribute verzichtet.
- **Aggregationen** sind Assoziationen, die eine Whole-Part-Beziehung symbolisieren.
- Gehören die Teile zu mehrere Aggregaten, spricht man von *Shared Aggregations*.
- **Kompositionen** sind Aggregationen mit Verantwortung über den Lebenszyklus der Elemente.
- Sind sind stets Unshared Aggregations, das Aggregat-Objekt kann sich zur Laufzeit aber ändern.



Assoziationsklassen

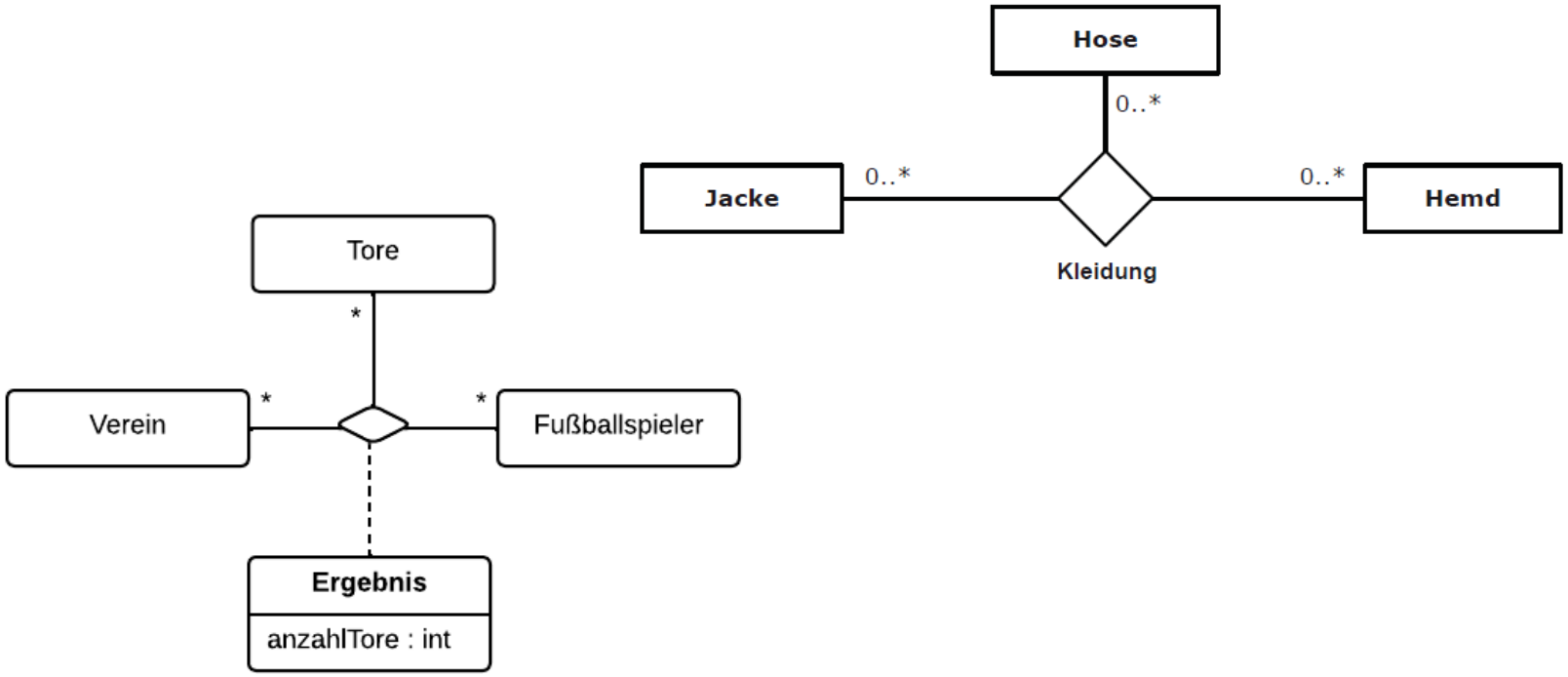
- Assoziationsklassen können wie folgt dargestellt werden:





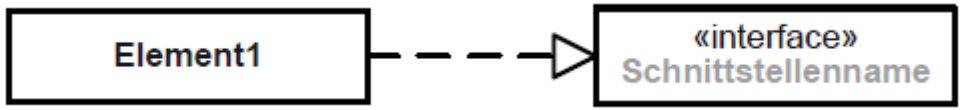
N-äre Assoziation

- Neben binären Assoziationen sind auch n-äre Assoziationen möglich:



Generalisierung

- Generalisierung beschreibt die Beziehung zwischen einer allgemeinen Klasse und einer spezialisierten Klasse.
- So wird eine Klassenhierarchie gebildet.
- Spezialisierungen stellen Erweiterungen bzw. Verfeinerungen dar (Liskovsches Substutionsprinzip).
- Während hierbei „is-a"-Beziehungen gebildet werden, bilden Assoziationen „has-a"-Beziehungen.
- Interfaces werden mit dem Stereotyp „`<<interface>>`" versehen. Der Generalisierungspfeil ist dabei gestrichelt.



Sonstiges

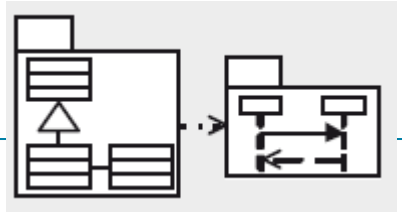
- Abgeleitete Attribute und Assoziation werden mit dem Präfix „/“ eingeleitet.
- Elemente sind abgeleitet, wenn sie bereits durch andere Elemente beschrieben sind.

- Die Sichtbarkeit von Elementen kann durch folgende Präfixe angegeben werden:
  - „+“: public
  - „-“: private
  - „#“: protected
  - „~“: package

- Klassenelemente (statische Elemente) werden unterstrichen.

- In einer Klasse ist der Name Pflicht, Attribute und Methoden sind optional.

Paketdiagramm



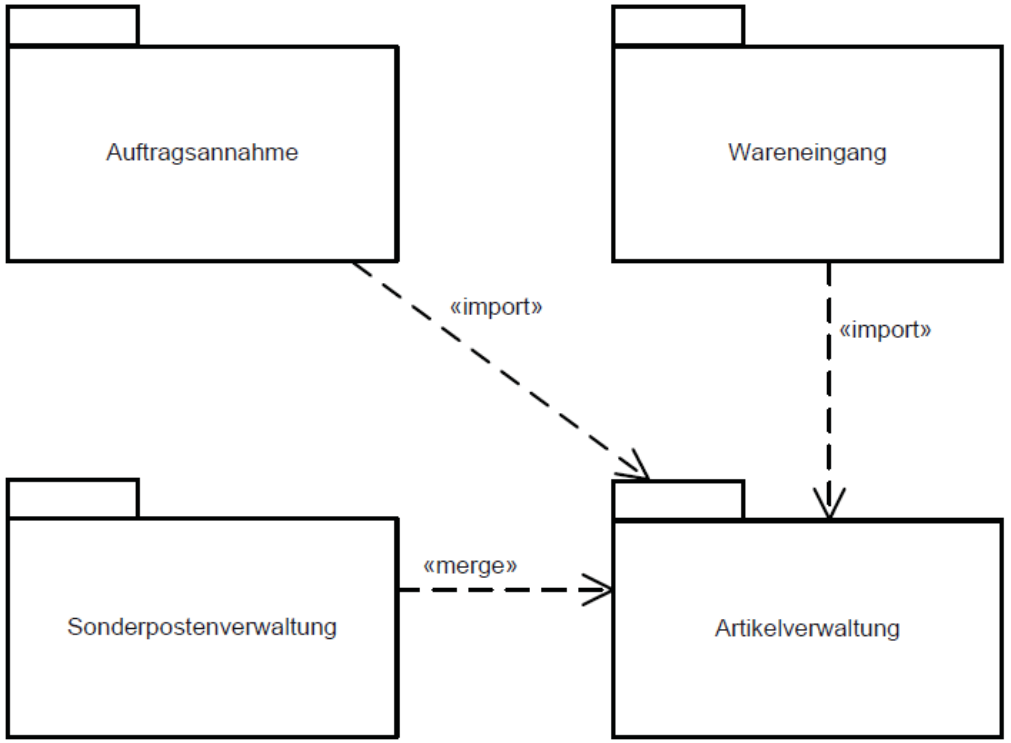
- Zentrale Frage:

  Wie kann ich mein Modell so schneiden, dass ich den Überblick bewahre?

- Stärken:
  - Organisiert das Systemmodell in größere Einheiten durch logische Zusammenfassung von Modellelementen.
  - Modellierung von Abhängigkeiten.

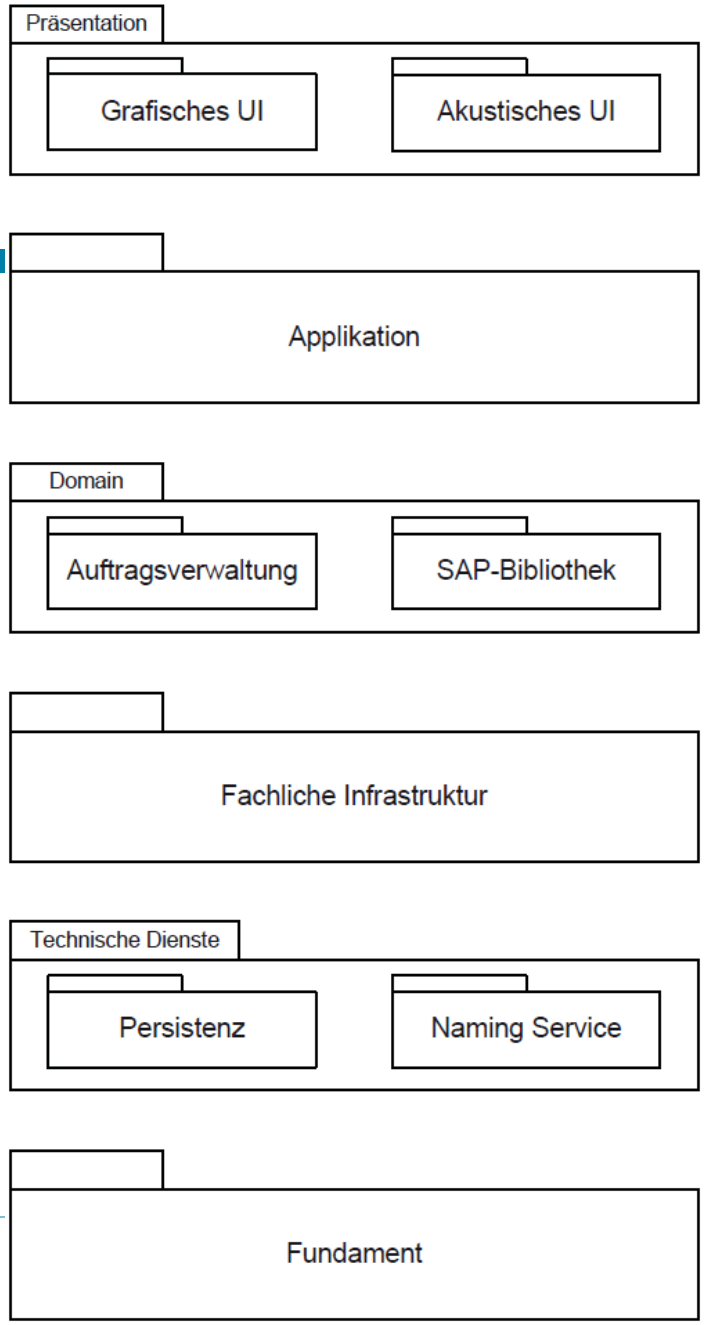
Beispiel

- Funktionale Gliederung:



Beispiel

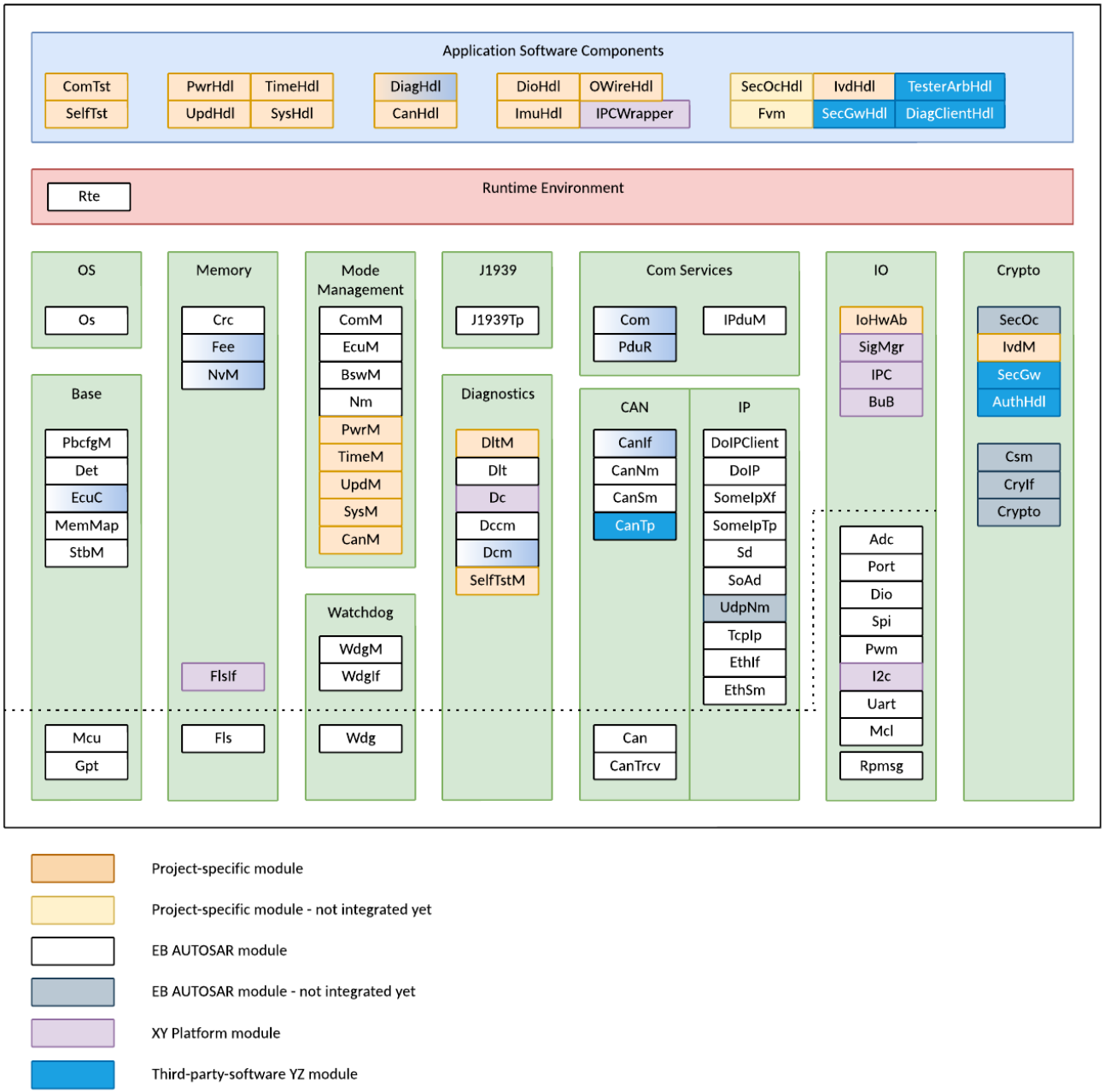
- Beschreibung von Schichten:



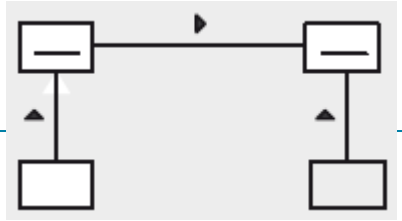
„Big Picture“

- In der Praxis wird für die Architektur oftmals ein „Big Picture“ verlangt, welches alle Subsysteme und Module umfasst.
- Farblich codiert werden dabei vor allem Module von Drittanbietern, Open-Source-Software, Plattform- oder Legacy-Modulen.
- Folgende Folie zeigt ein Beispiel eines Realtime-Controllers der Firma Continental unter Verwendung des Electrobit-AUTOSAR-Stacks.
- Es stellt kein Diagramm nach UML-Standard dar, schafft aber Überblick.

Beispiel


Application Software Components
ComTst
SelfTst
PwrHdl
TimeHdl
UpdHdl
SysHdl
DiagHdl
CanHdl
DioHdl
OWireHdl
ImuHdl
IPCWrapper
SecOcHdl
IvdHdl
TesterArbHdl
Fvm
SecGwHdl
DiagClientHdl
Rte
Runtime Environment
OS
Os
Memory
Crc
Fee
NvM
Mode Management
ComM
EcuM
BswM
Nm
PwrM
TimeM
UpdM
SysM
CanM
J1939
J1939Tp
Com Services
Com
PduR
IPduM
IO
IoHwAb
SigMgr
IPC
BuB
Crypto
SecOc
IvdM
SecGw
AuthHdl
Csm
CryIf
Crypto
Base
PbcfgM
Det
EcuC
MemMap
StbM
Diagnostics
DltM
Dlt
Dc
Dccm
Dcm
SelfTstM
CAN
CanIf
CanNm
CanSm
CanTp
IP
DoIPClient
DoIP
SomeIpXf
SomeIpTp
Sd
SoAd
UdpNm
TcpIp
EthIf
EthSm
Adc
Port
Dio
Spi
Pwm
I2c
Uart
Mcl
Rpmsg
FlsIf
Watchdog
WdgM
WdgIf
Mcu
Gpt
Fls
Wdg
Can
CanTrcv
Project-specific module
Project-specific module - not integrated yet
EB AUTOSAR module
EB AUTOSAR module - not integrated yet
XY Platform module
Third-party-software YZ module

Objektdiagramm

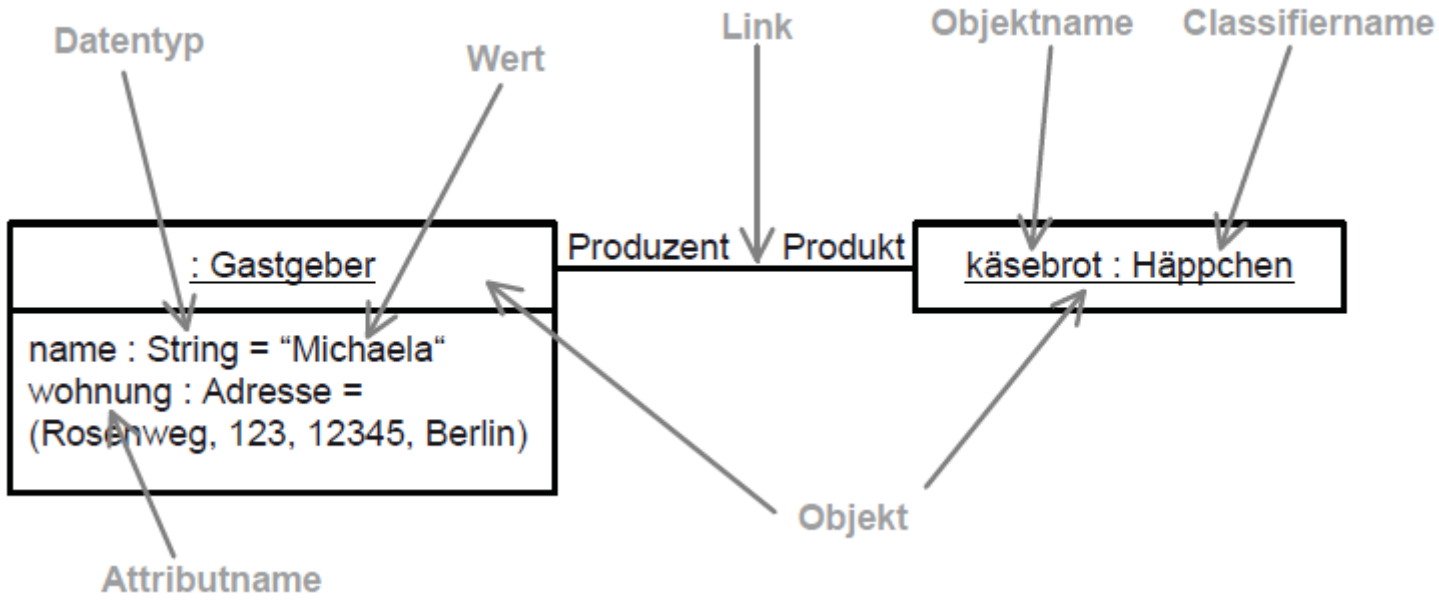


- Zentrale Frage:

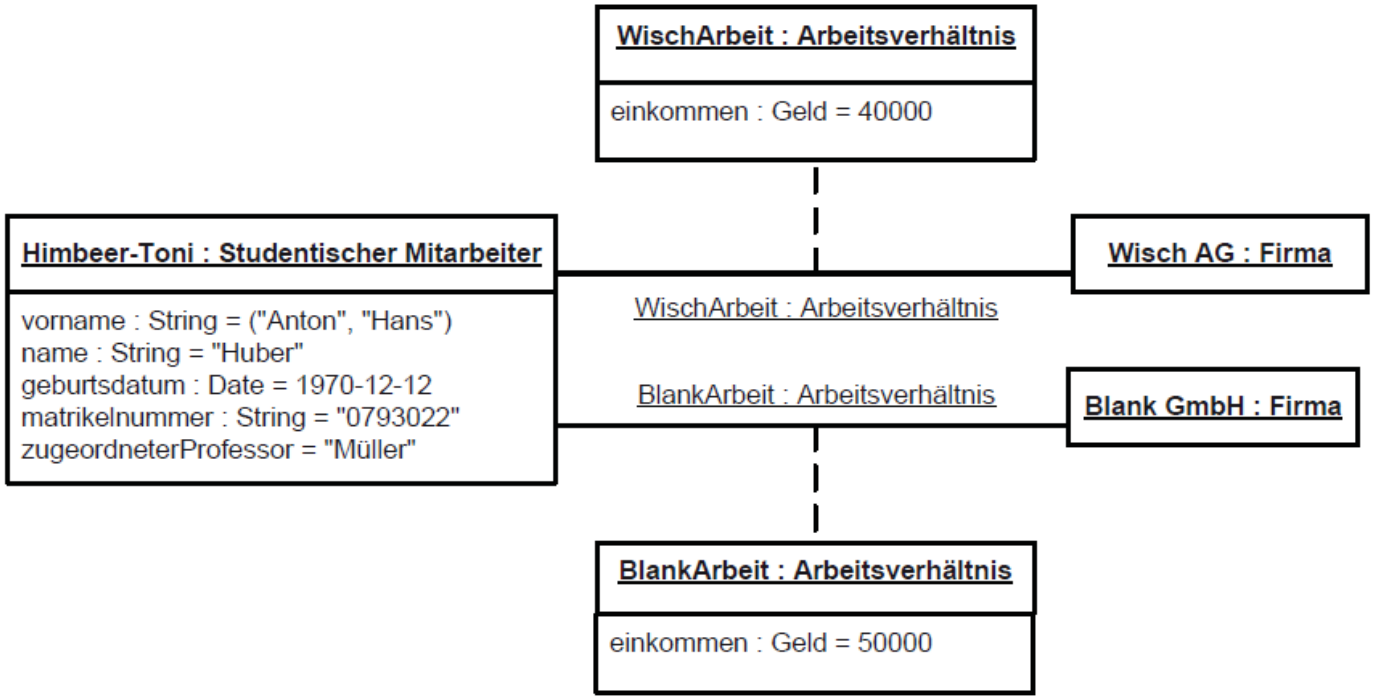
  Welche innere Struktur besitzt mein System zu einem bestimmten Zeitpunkt zur Laufzeit (Klassendiagrammschnappschuss)?

- Stärken:
  - Zeigt Objekte und deren Attributbelegungen zu einem bestimmten Zeitpunkt.
  - Wird nur beispielhaft zur Veranschaulichung verwendet.
  - Detailniveau wie im Klassendiagramm.
  - Sehr gute Darstellung von Mengenverhältnissen.

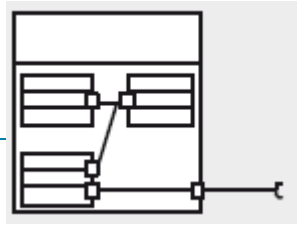
Beispiel


Datentyp
Wert
Link
Objektname
Classifiername
: Gastgeber
name : String = "Michaela"
wohnung : Adresse =
(Rosenweg, 123, 12345, Berlin)
Produzent
Produkt
käsebrot : Häppchen
Objekt
Attributname

Beispiel


WischArbeit : Arbeitsverhältnis
einkommen : Geld = 40000
Himbeer-Toni : Studentischer Mitarbeiter
vorname : String = ("Anton", "Hans")
name : String = "Huber"
geburtsdatum : Date = 1970-12-12
matrikelnummer : String = "0793022"
zugeordneterProfessor = "Müller"
Wisch AG : Firma
WischArbeit : Arbeitsverhältnis
BlankArbeit : Arbeitsverhältnis
Blank GmbH : Firma
BlankArbeit : Arbeitsverhältnis
einkommen : Geld = 50000

Kompositionsstruktur-diagramm

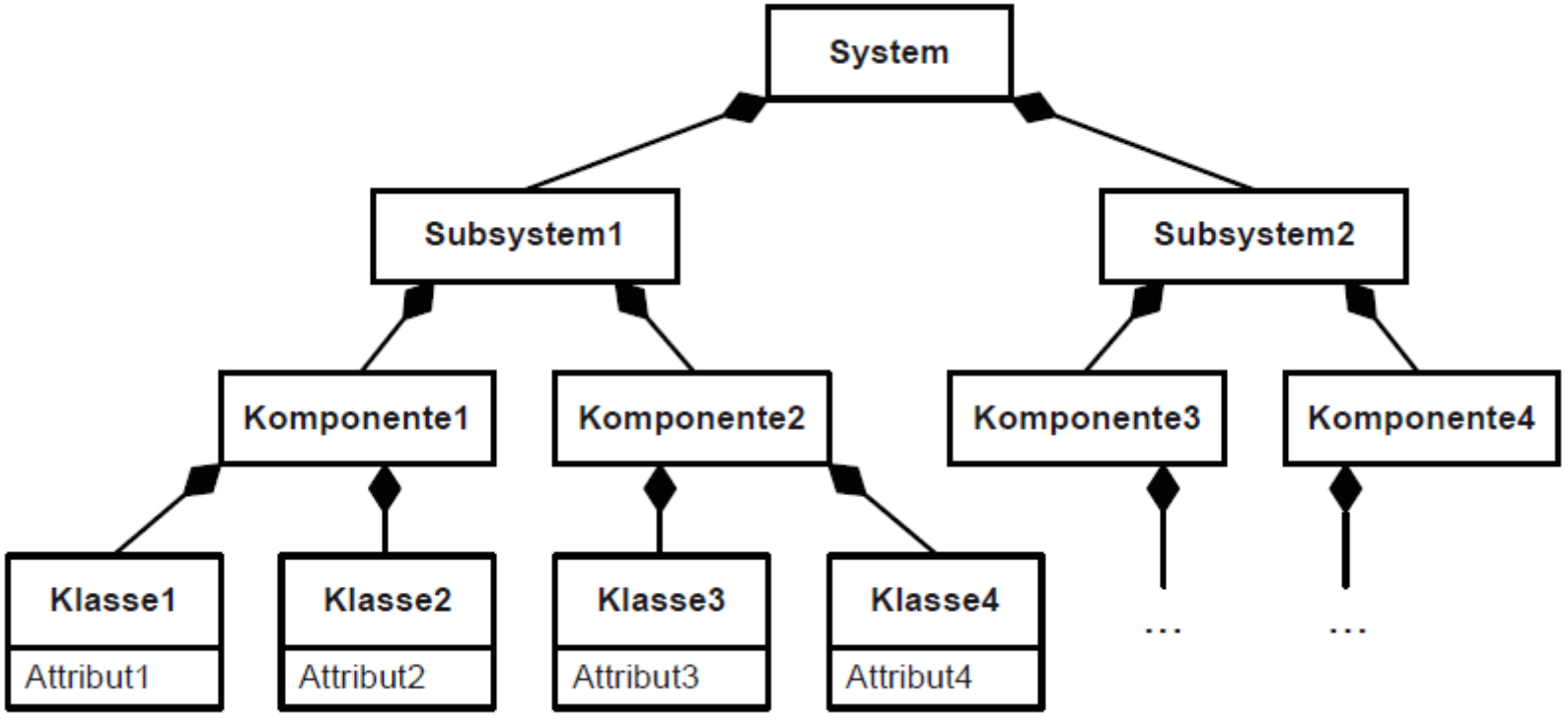


- Zentrale Frage:

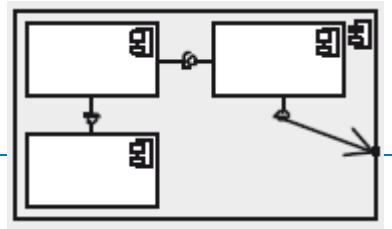
  Wie sieht das Innenleben einer Klasse, einer Komponente, eines Systemteils aus?

- Stärken:
  - Ideal für die Top-Down-Modellierung des Systems.
  - Mittleres Detailniveau, zeigt Teile eines „Gesamtelements“ und deren Mengenverhältnisse.

Beispiel


System
Subsystem1
Subsystem2
Komponente1
Komponente2
Komponente3
Komponente4
Klasse1
Attribut1
Klasse2
Attribut2
Klasse3
Attribut3
Klasse4
Attribut4
...
...

Komponentendiagramm

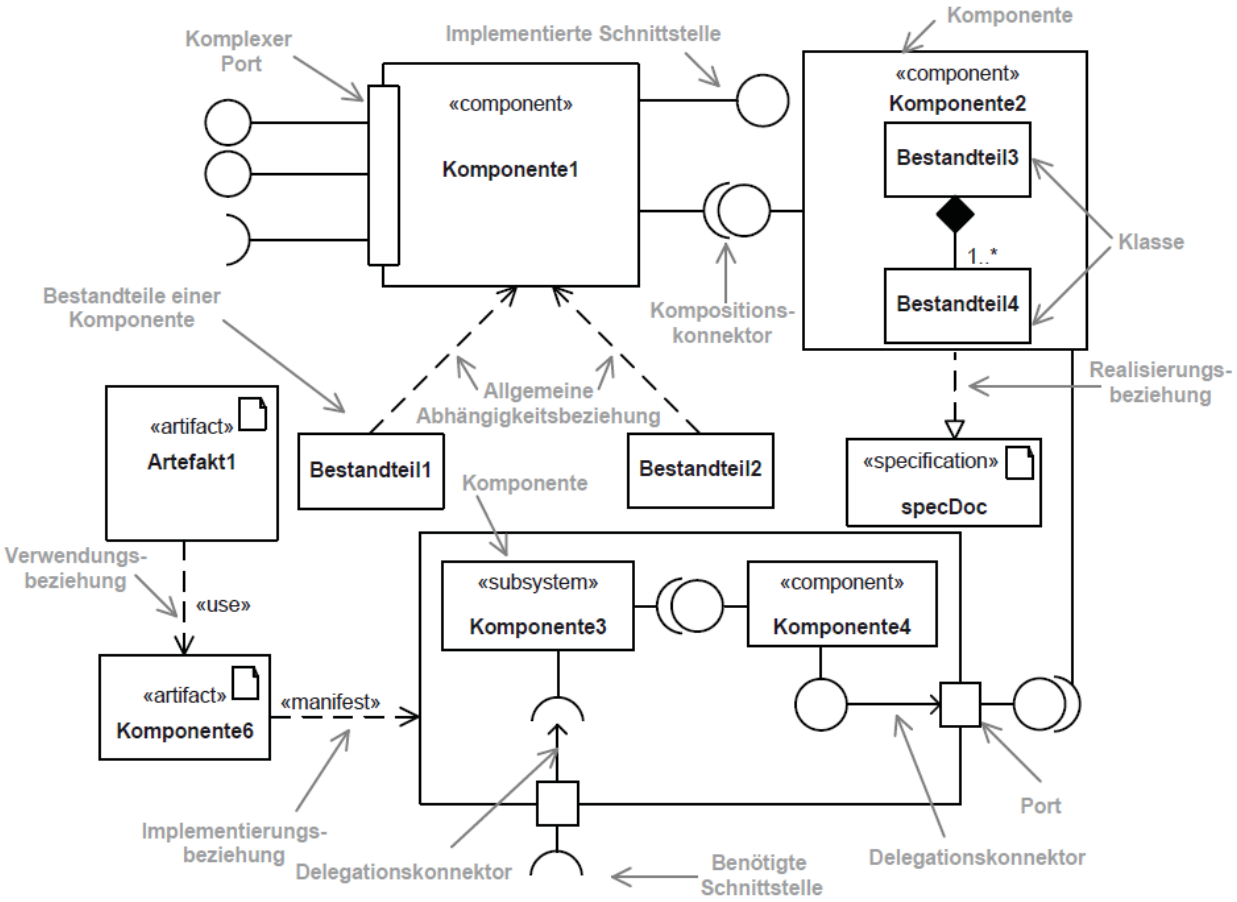


- Zentrale Frage:

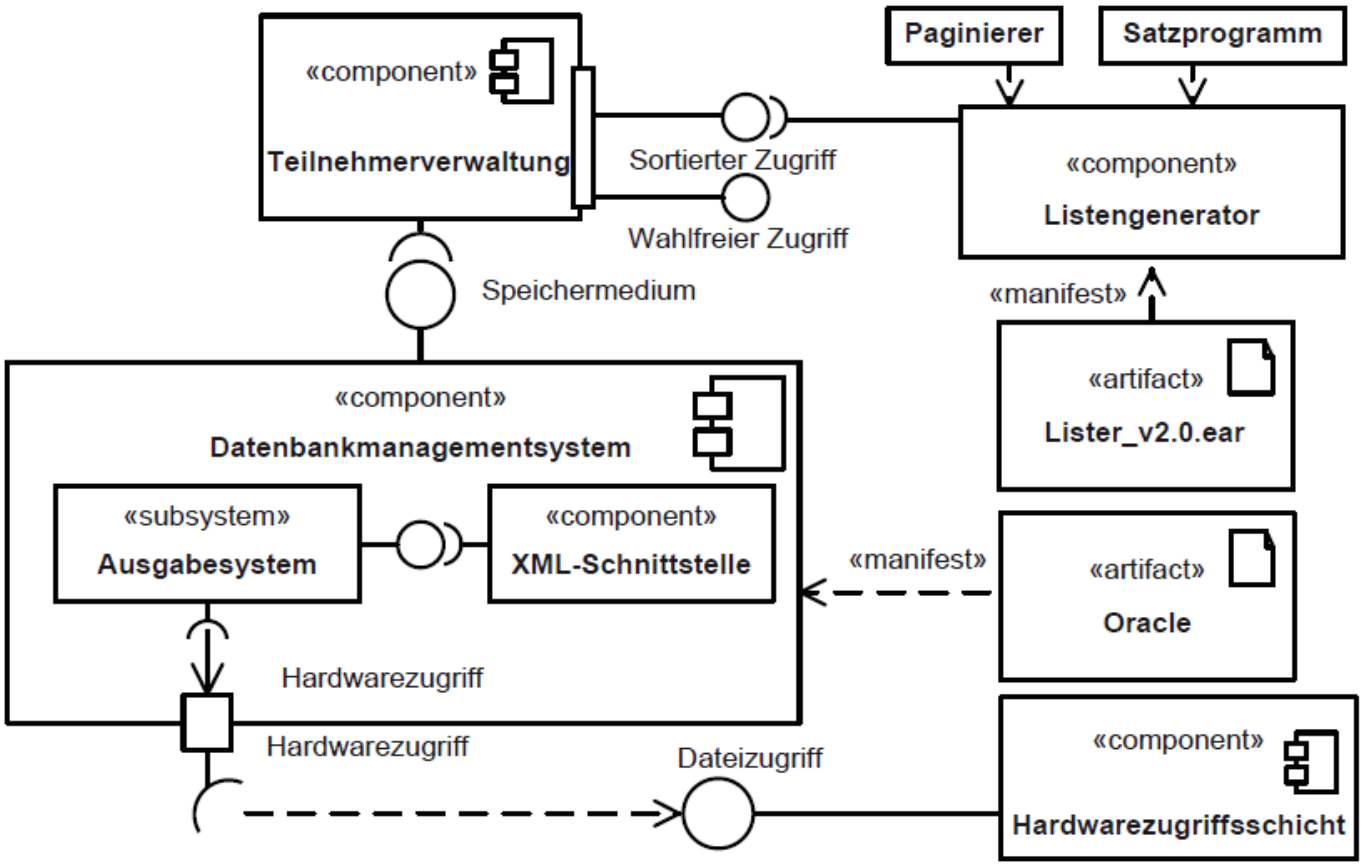
  Wie werden meine Klassen zu wiederverwendbaren, verwaltbaren Komponenten zusammengefasst und wie stehen diese miteinander in Beziehung?

- Stärken:
  - Zeigt Organisation und Abhängigkeiten einzelner technischer Systemkomponenten.
  - Modellierung angebotener und benötigter Schnittstellen möglich.

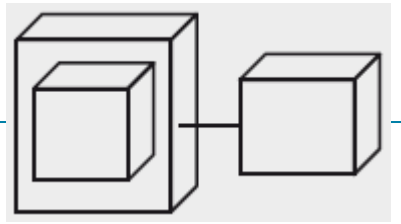
Beispiel


Komplexer Port
Implementierte Schnittstelle
Komponente
«component»
Komponente1
«component»
Komponente2
Bestandteil3
1..*
Bestandteil4
Klasse
Kompositions-konnektor
Bestandteile einer Komponente
Allgemeine Abhängigkeitsbeziehung
Realisierungs-beziehung
«artifact»
Artefakt1
Bestandteil1
Bestandteil2
«specification»
specDoc
Komponente
Verwendungs-beziehung
«use»
«subsystem»
Komponente3
«component»
Komponente4
«artifact»
Komponente6
«manifest»
Port
Implementierungs-beziehung
Delegationskonnektor
Benötigte Schnittstelle
Delegationskonnektor

Beispiel


«component»
Teilnehmerverwaltung
Sortierter Zugriff
Wahlfreier Zugriff
Speichermedium
Paginierer
Satzprogramm
«component»
Listengenerator
«manifest»
«artifact»
Lister_v2.0.ear
«component»
Datenbankmanagementsystem
«subsystem»
Ausgabesystem
«component»
XML-Schnittstelle
«manifest»
«artifact»
Oracle
Hardwarezugriff
Hardwarezugriff
Dateizugriff
«component»
Hardwarezugriffsschicht

Verteilungsdiagramm

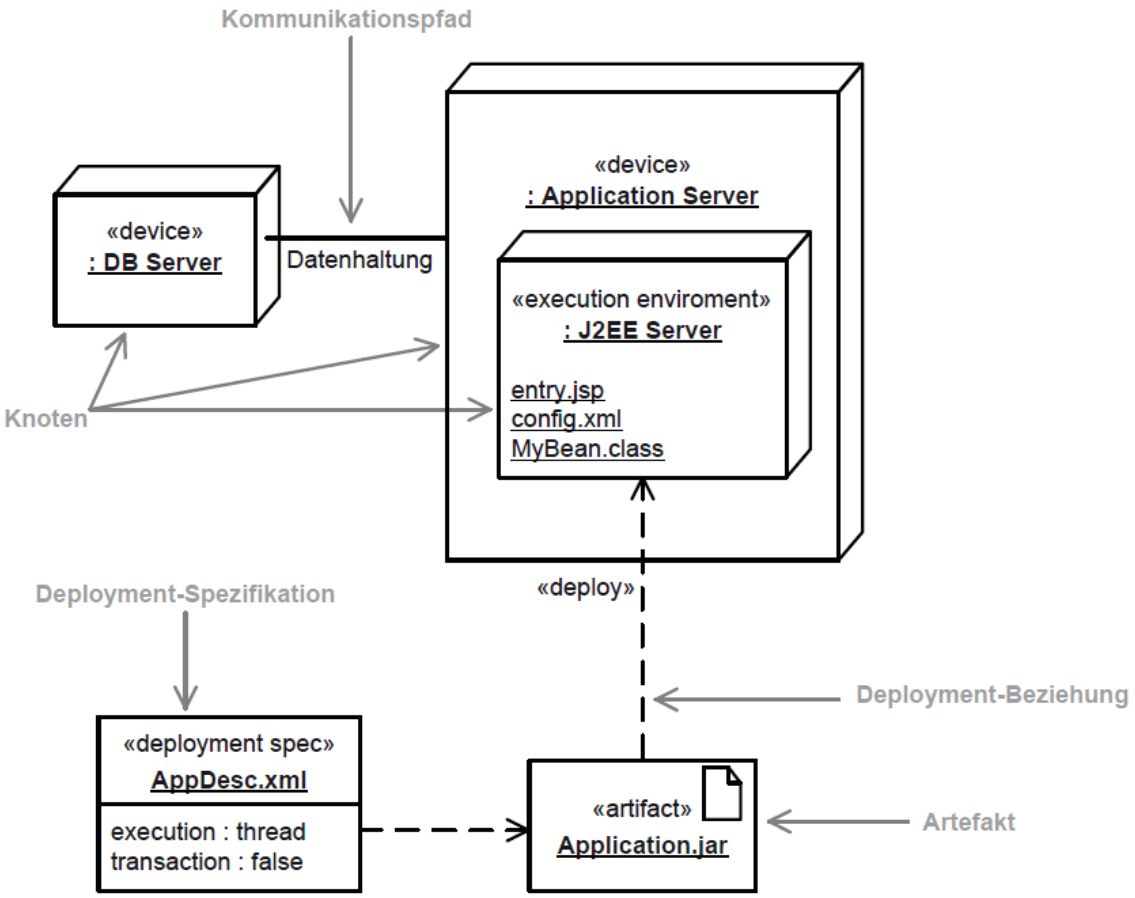


- Zentrale Frage:

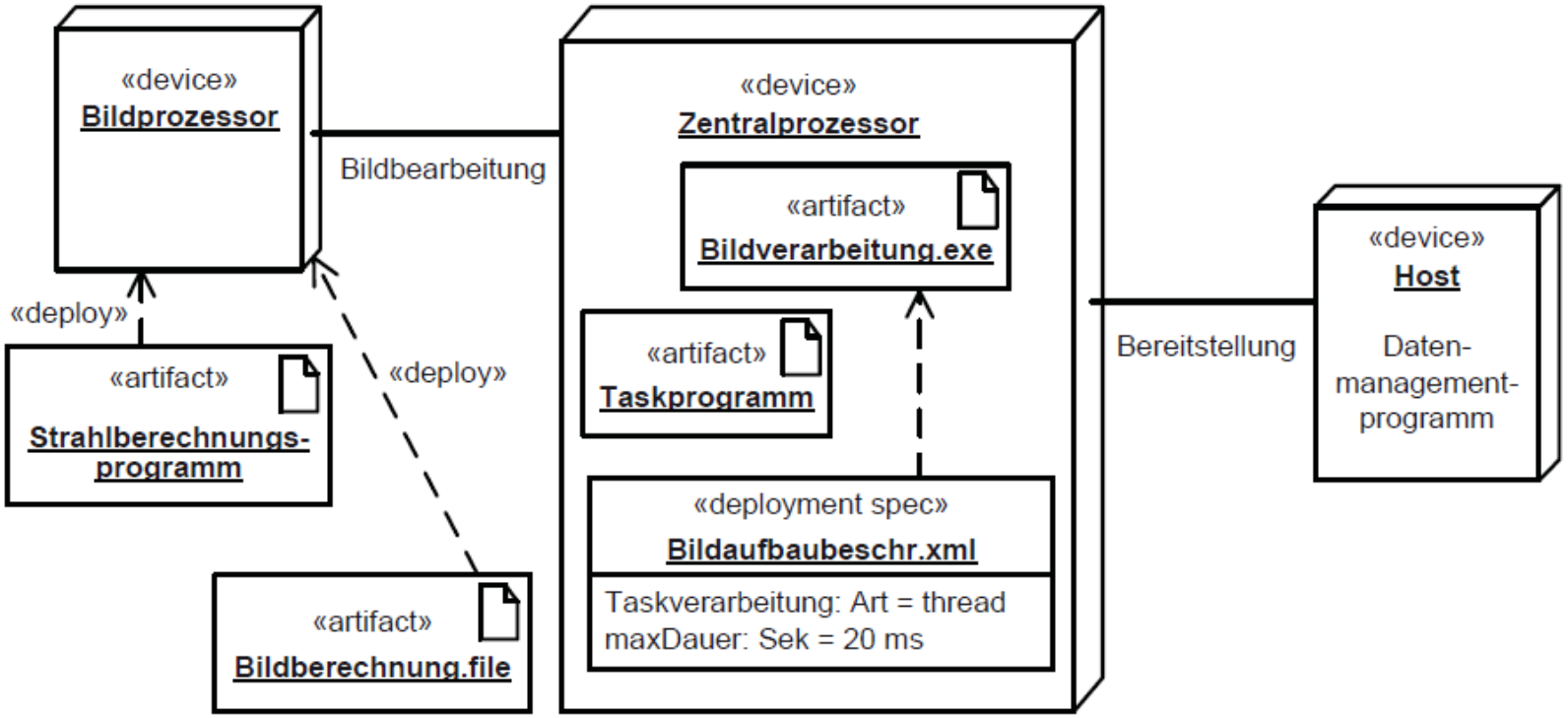
  Wie sieht das Einsatzumfeld (Hardware, Server, Datenbanken, …) des Systems aus? Wie werden die Komponenten zur Laufzeit wohin verteilt?

- Stärken:
  - Zeigt das Laufzeitumfeld des Systems mit den „greifbaren" Systemteilen (meist Hardware).
  - Hohes Abstraktionsniveau, kaum Notationselemente.

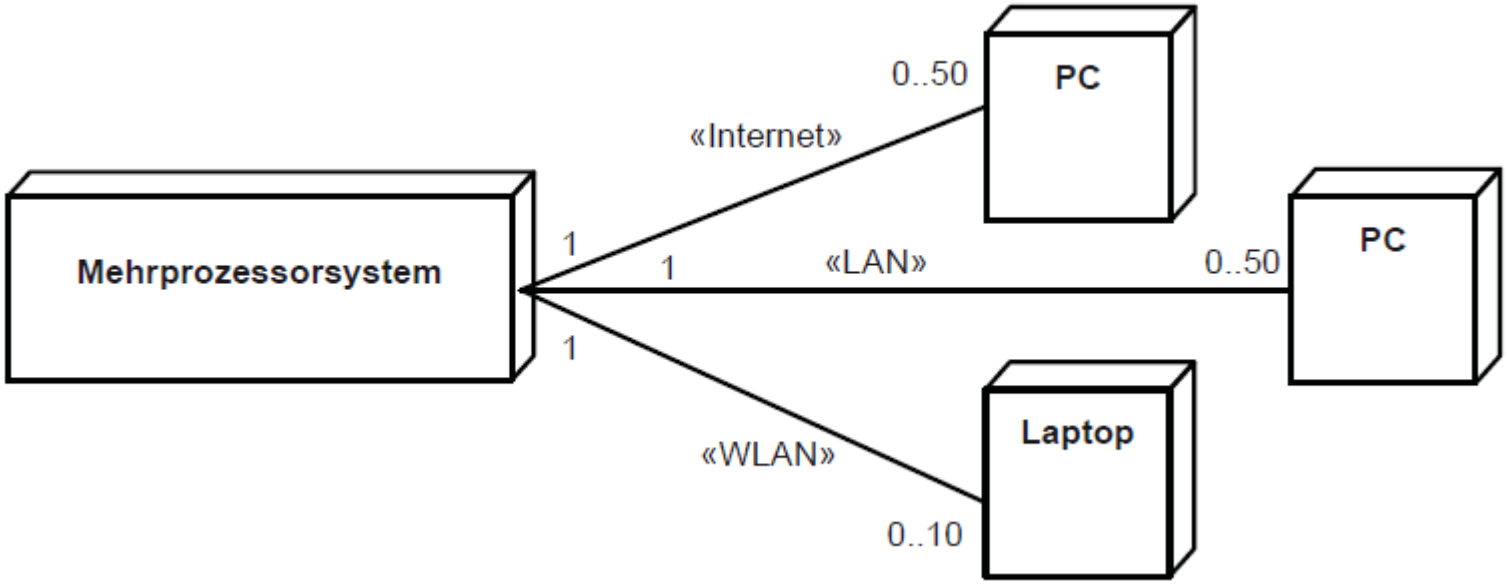
Beispiel


Kommunikationspfad
«device»
: DB Server
Datenhaltung
«device»
: Application Server
«execution enviroment»
: J2EE Server
entry.jsp
config.xml
MyBean.class
Knoten
Deployment-Spezifikation
«deploy»
Deployment-Beziehung
«deployment spec»
AppDesc.xml
execution : thread
transaction : false
«artifact»
Application.jar
Artefakt

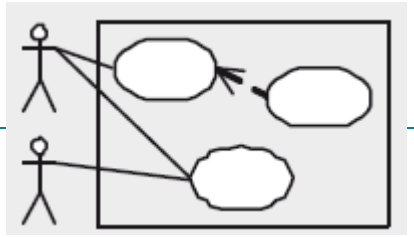
Beispiel


«device»
Bildprozessor
Bildbearbeitung
«device»
Zentralprozessor
«artifact»
Bildverarbeitung.exe
«deploy»
«artifact»
Strahlberechnungs-
programm
«deploy»
«artifact»
Taskprogramm
Bereitstellung
«device»
Host
Daten-
management-
programm
«deployment spec»
Bildaufbaubeschr.xml
Taskverarbeitung: Art = thread
maxDauer: Sek = 20 ms
«artifact»
Bildberechnung.file

Beispiel


Mehrprozessorsystem
1
«Internet»
0..50
PC
1
«LAN»
0..50
PC
1
«WLAN»
0..10
Laptop

Use-Case-Diagramm

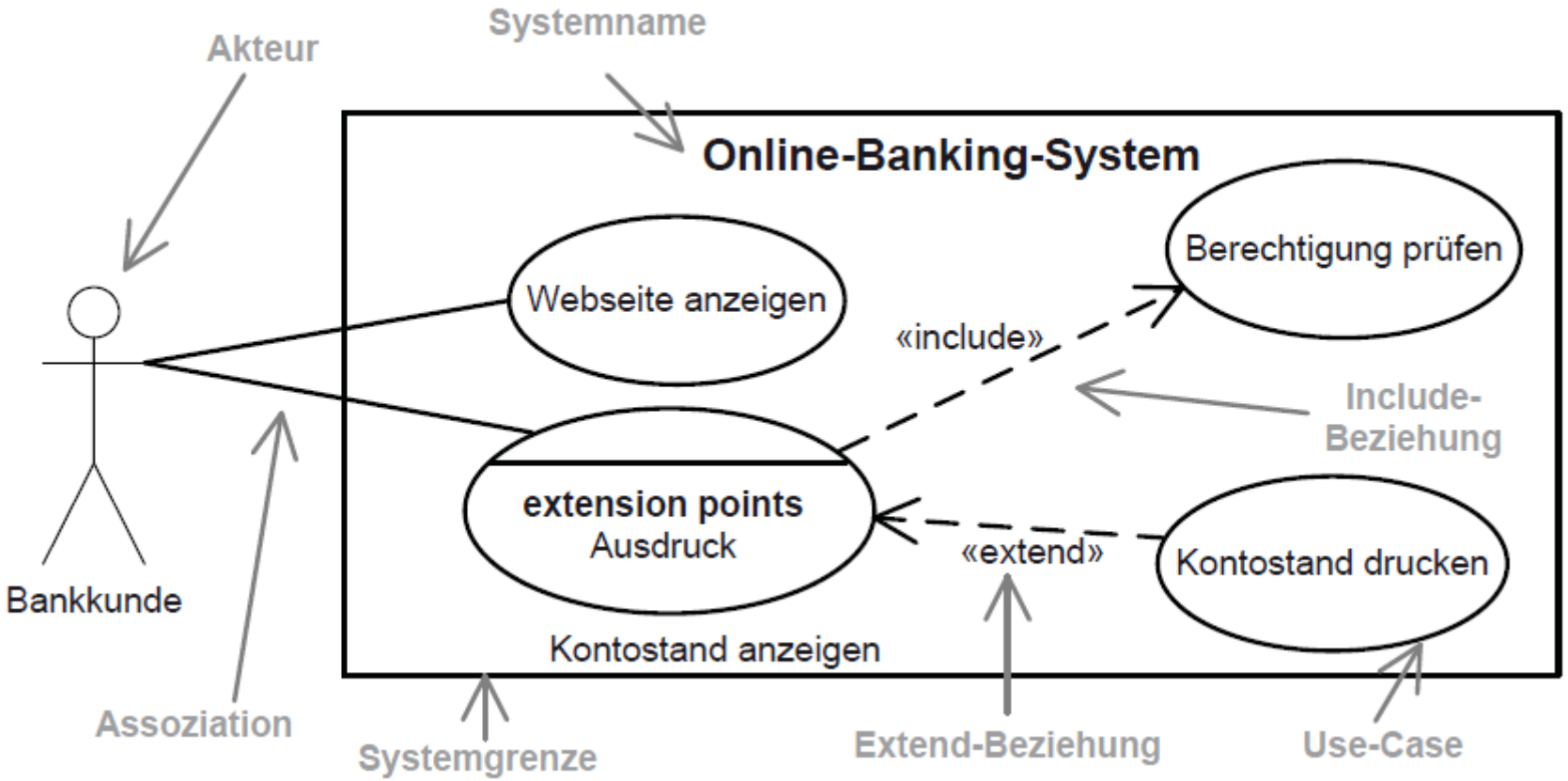


- Zentrale Frage:

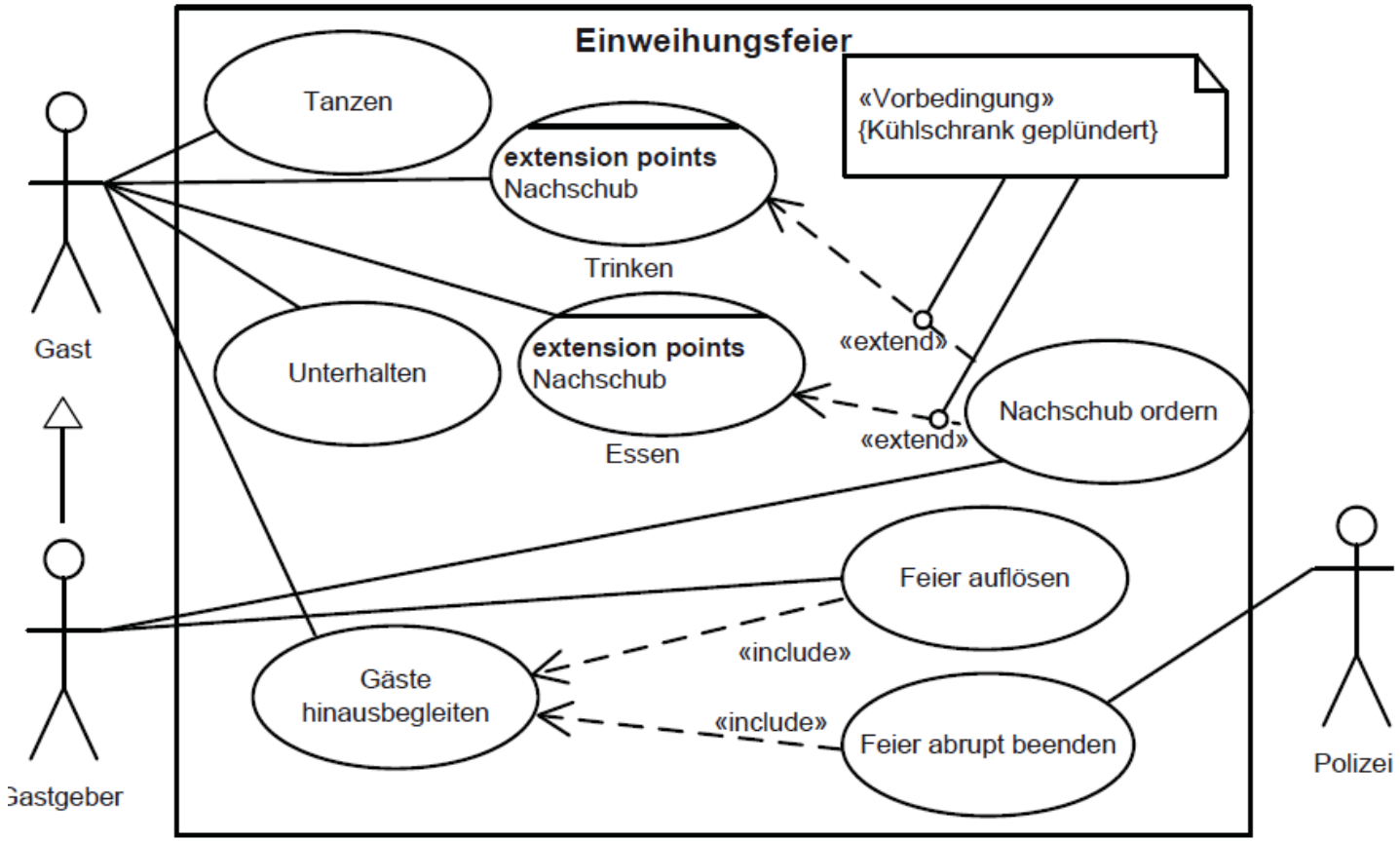
  Was leistet mein System für seine Umwelt (Nachbarsysteme, Stakeholder)?

- Stärken:
  - Präsentiert die Außensicht auf das System.
  - Geeignet zur Kontextabgrenzung.
  - Hohes Abstraktionsniveau, einfache Notationsmittel.

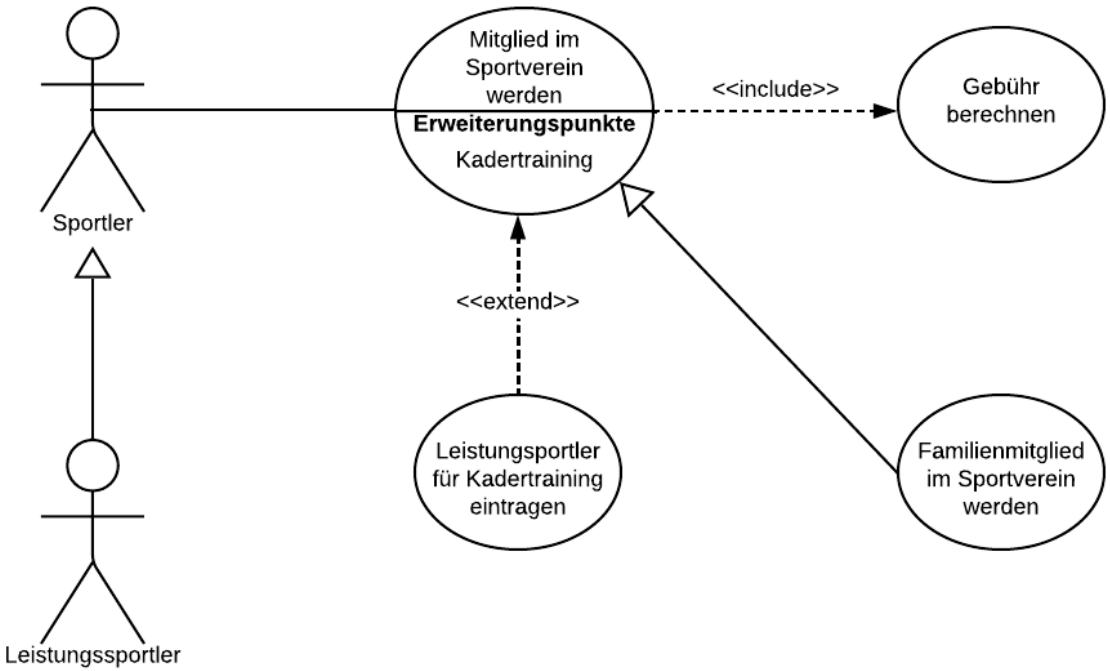
Beispiel


Akteur
Systemname
Online-Banking-System
Webseite anzeigen
Berechtigung prüfen
«include»
Include-Beziehung
extension points
Ausdruck
«extend»
Kontostand drucken
Kontostand anzeigen
Bankkunde
Assoziation
Systemgrenze
Extend-Beziehung
Use-Case

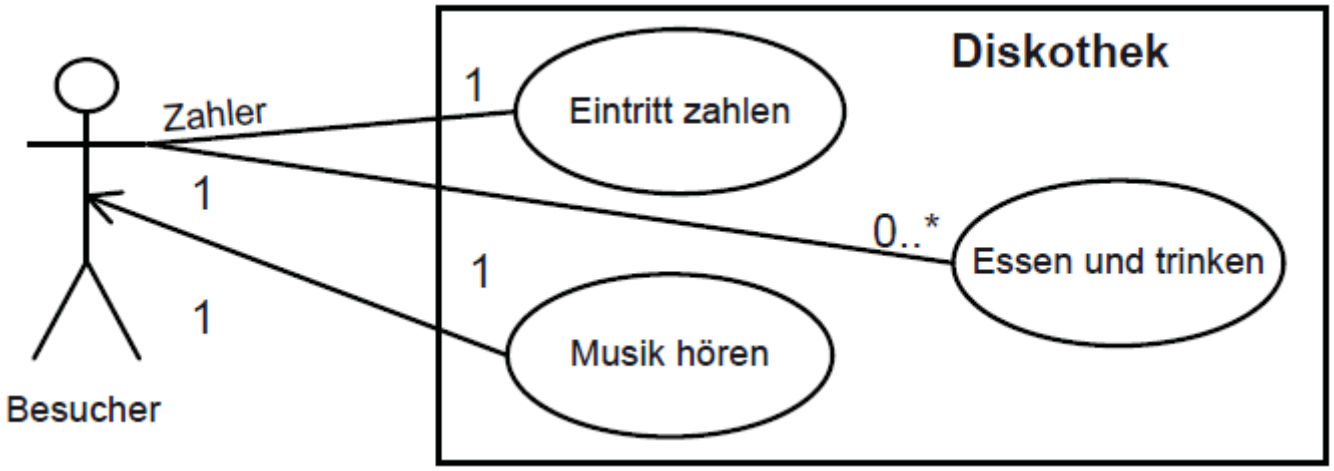
Beispiel


Einweihungsfeier
Tanzen
extension points
Nachschub
Trinken
«Vorbedingung»
{Kühlschrank geplündert}
Gast
Unterhalten
extension points
Nachschub
Essen
«extend»
«extend»
Nachschub ordern
Feier auflösen
«include»
Gäste
hinausbegleiten
«include»
Feier abrupt beenden
Gastgeber
Polizei

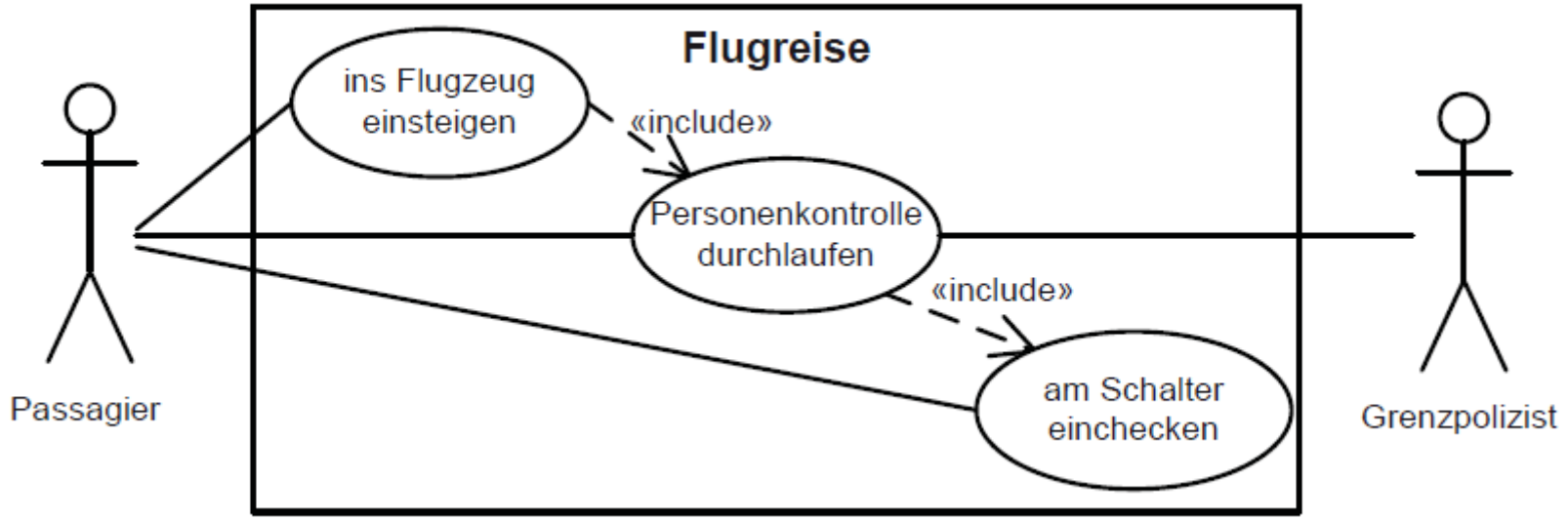
Beispiel


Mitglied im Sportverein werden
Erweiterungspunkte
Kadertraining
<<include>>
Gebühr berechnen
Sportler
<<extend>>
Leistungsportler für Kadertraining eintragen
Familienmitglied im Sportverein werden
Leistungssportler

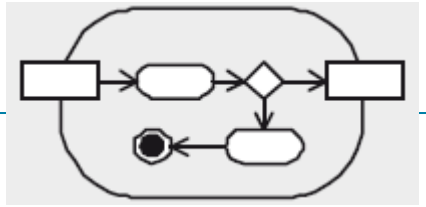
Beispiel


Diskothek
Zahler
1
Eintritt zahlen
1
0..*
Essen und trinken
1
1
Musik hören
Besucher

Beispiel


Flugreise
ins Flugzeug einsteigen
«include»
Personenkontrolle durchlaufen
«include»
am Schalter einchecken
Passagier
Grenzpolizist

Aktivitätsdiagramm

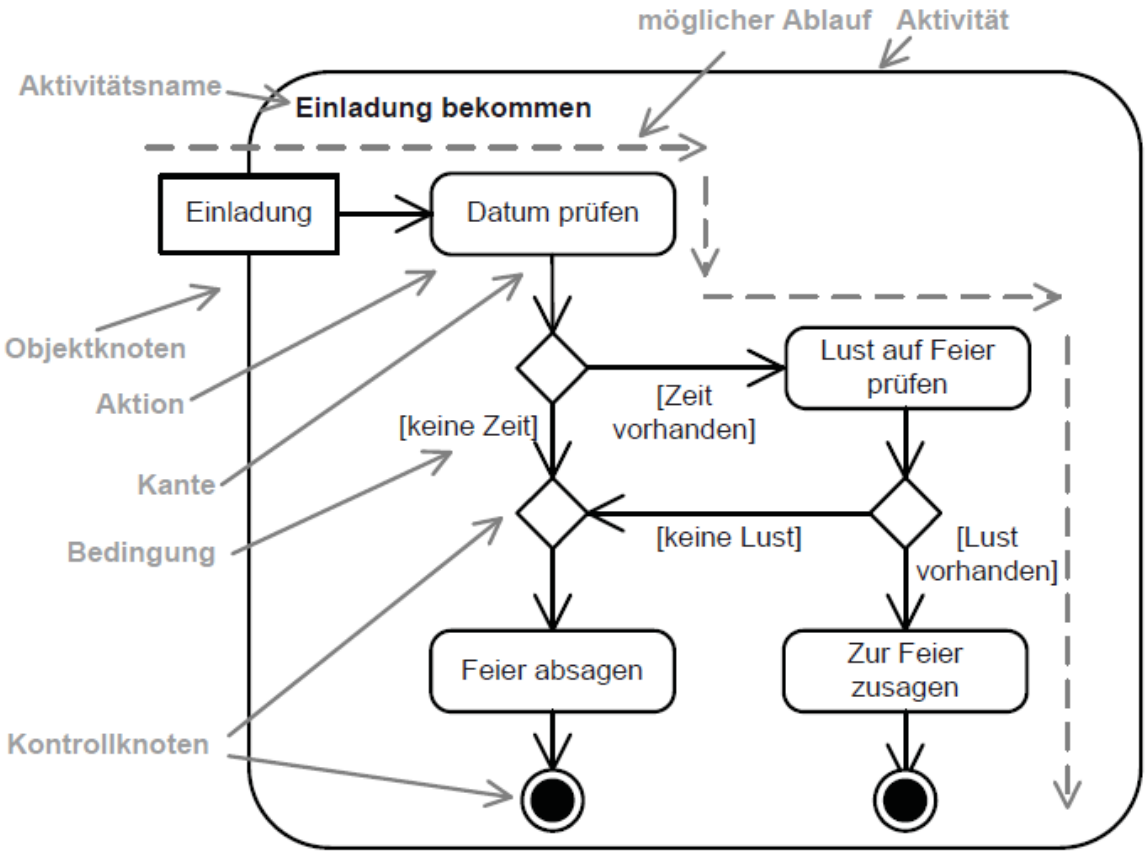


- Zentrale Frage:

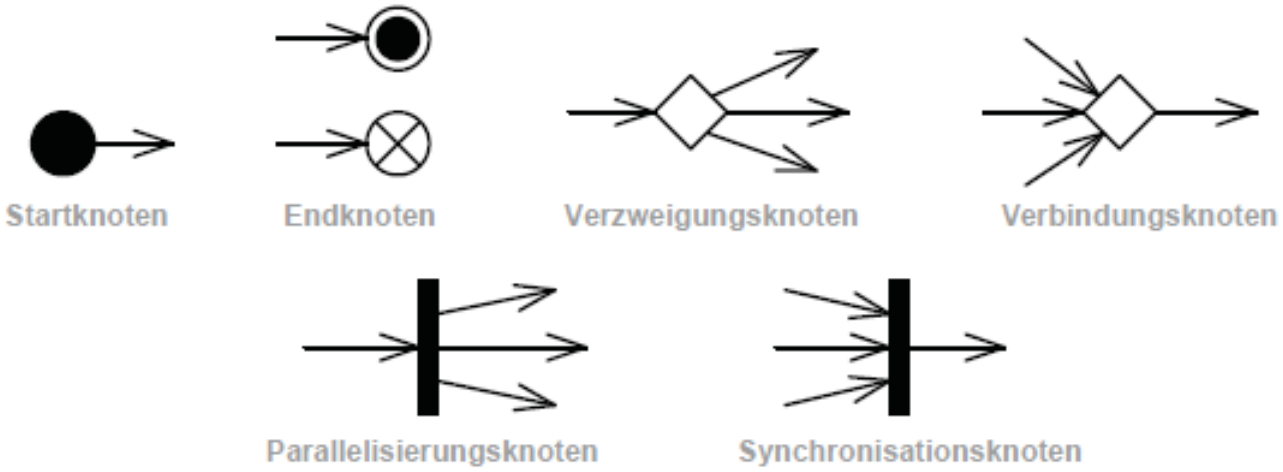
  Wie läuft ein bestimmter flussorientierter Prozess oder ein Algorithmus ab?

- Stärken:
  - Sehr detaillierte Visualisierung von Abläufen mit Bedingungen, Schleifen, Verwzeigungen.
  - Parallelisierung und Synchronisation möglich.

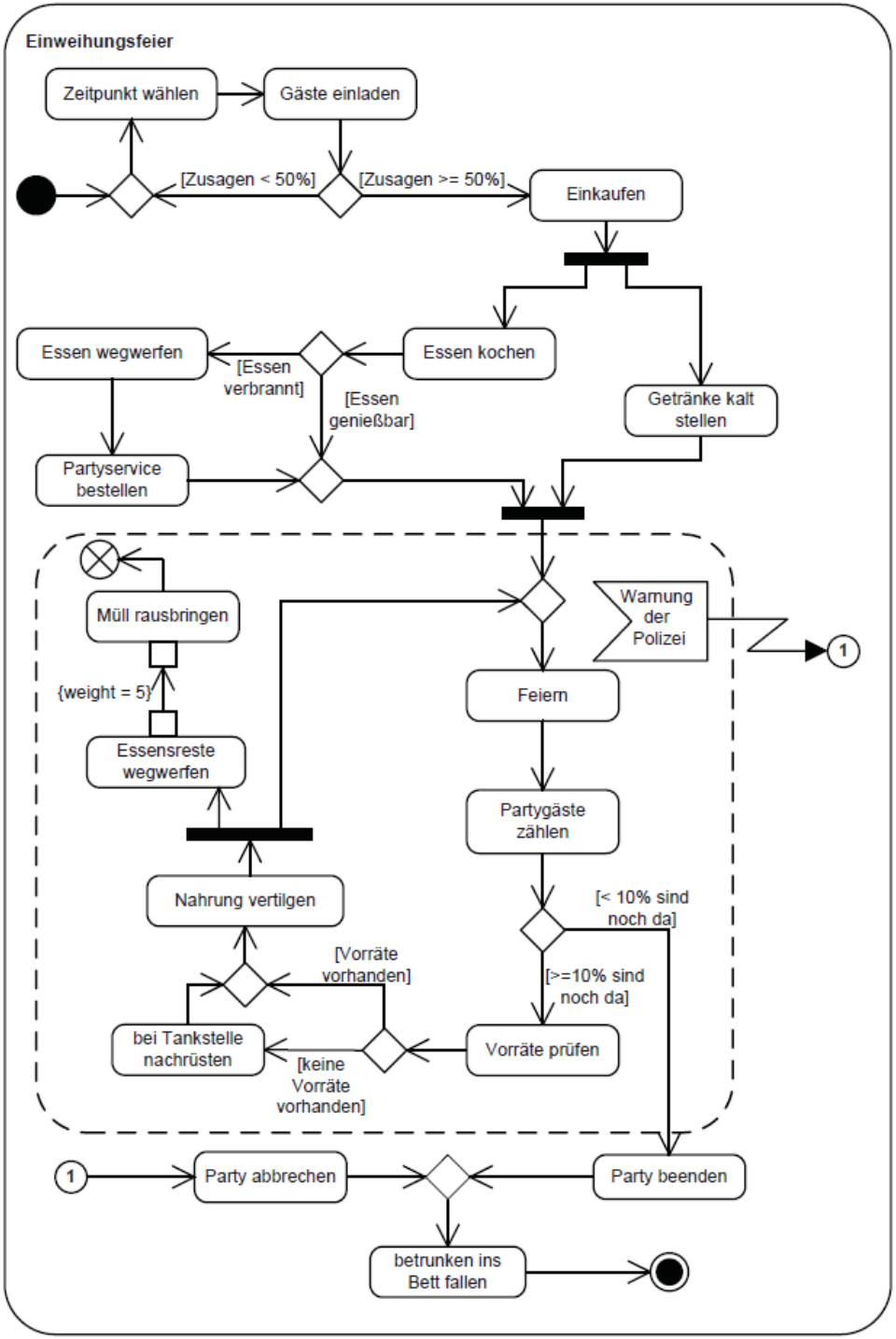
Beispiel


möglicher Ablauf
Aktivität
Aktivitätsname
Einladung bekommen
Einladung
Datum prüfen
Objektknoten
Aktion
Kante
Bedingung
Kontrollknoten
[keine Zeit]
[Zeit vorhanden]
Lust auf Feier prüfen
[keine Lust]
[Lust vorhanden]
Feier absagen
Zur Feier zusagen

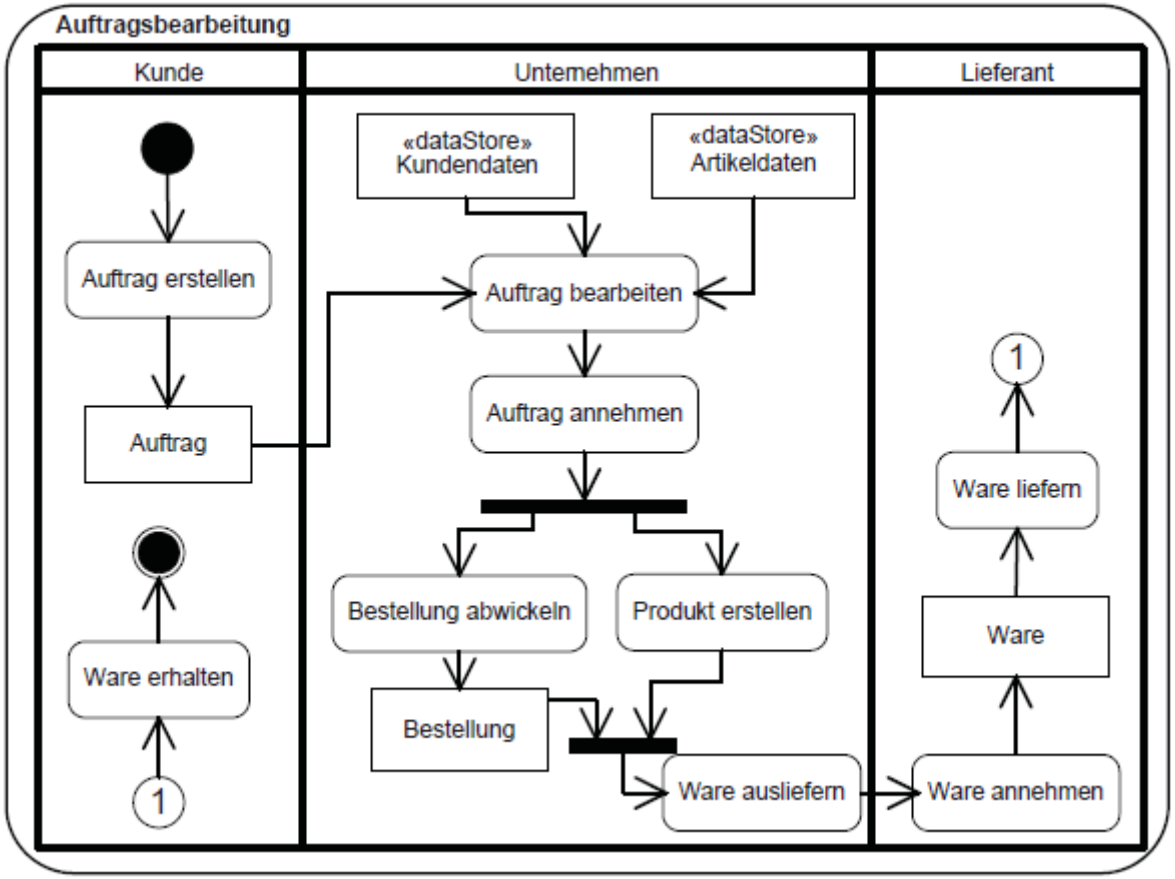
Kontrollknoten



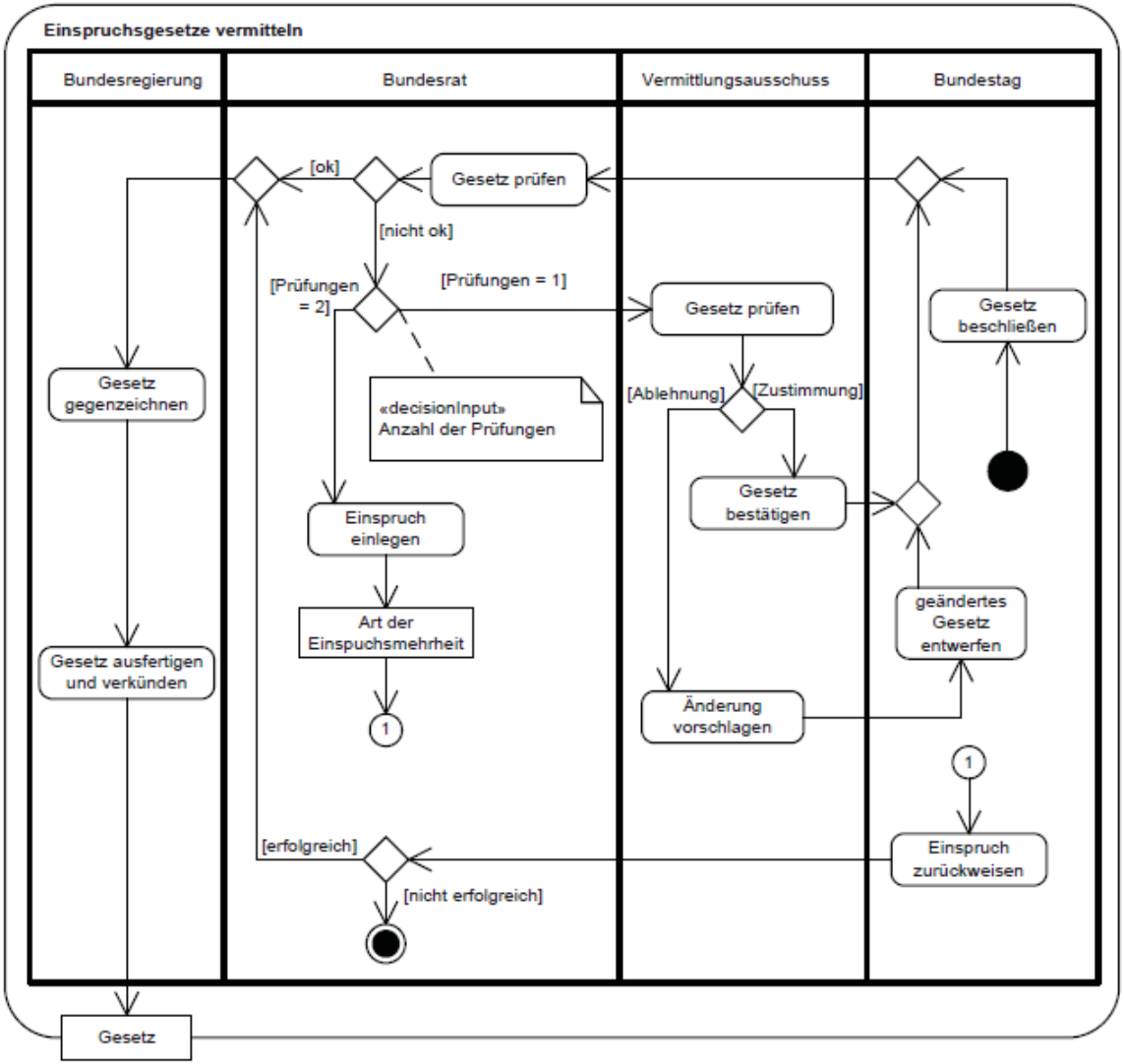
Beispiel


Einweihungsfeier
Zeitpunkt wählen
Gäste einladen
[Zusagen < 50%]
[Zusagen >= 50%]
Einkaufen
Essen wegwerfen
[Essen verbrannt]
Essen kochen
[Essen genießbar]
Getränke kalt stellen
Partyservice bestellen
Müll rausbringen
{weight = 5}
Essensreste wegwerfen
Warnung der Polizei
1
Feiern
Partygäste zählen
[< 10% sind noch da]
[>=10% sind noch da]
Nahrung vertilgen
[Vorräte vorhanden]
bei Tankstelle nachrüsten
[keine Vorräte vorhanden]
Vorräte prüfen
1
Party abbrechen
Party beenden
betrunken ins Bett fallen

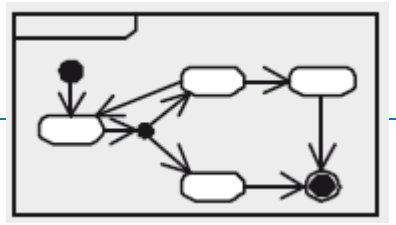
Beispiel


Auftragsbearbeitung
Kunde
Unternehmen
Lieferant
«dataStore»
Kundendaten
«dataStore»
Artikeldaten
Auftrag erstellen
Auftrag bearbeiten
Auftrag
Auftrag annehmen
Bestellung abwickeln
Produkt erstellen
Bestellung
Ware erhalten
1
Ware ausliefern
Ware annehmen
1
Ware liefern
Ware

Beispiel


Einspruchsgesetze vermitteln
Bundesregierung
Bundesrat
Vermittlungsausschuss
Bundestag
[ok]
Gesetz prüfen
[nicht ok]
[Prüfungen = 2]
[Prüfungen = 1]
Gesetz prüfen
Gesetz beschließen
Gesetz gegenzeichnen
«decisionInput» Anzahl der Prüfungen
[Ablehnung]
[Zustimmung]
Gesetz bestätigen
Einspruch einlegen
Art der Einspuchsmehrheit
geändertes Gesetz entwerfen
Gesetz ausfertigen und verkünden
1
Änderung vorschlagen
1
[erfolgreich]
Einspruch zurückweisen
[nicht erfolgreich]
Gesetz

Zustandsautomat

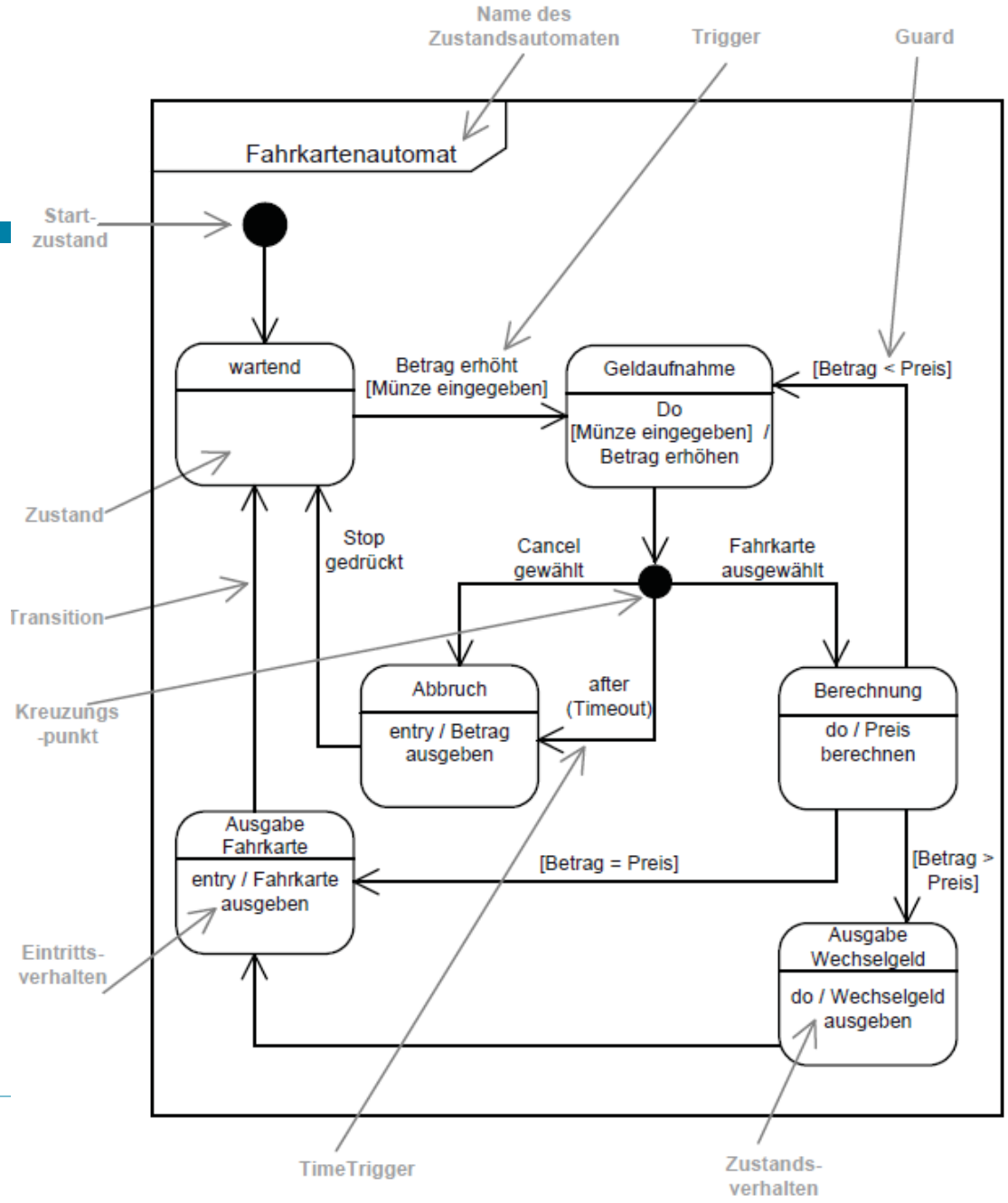


- Zentrale Frage:

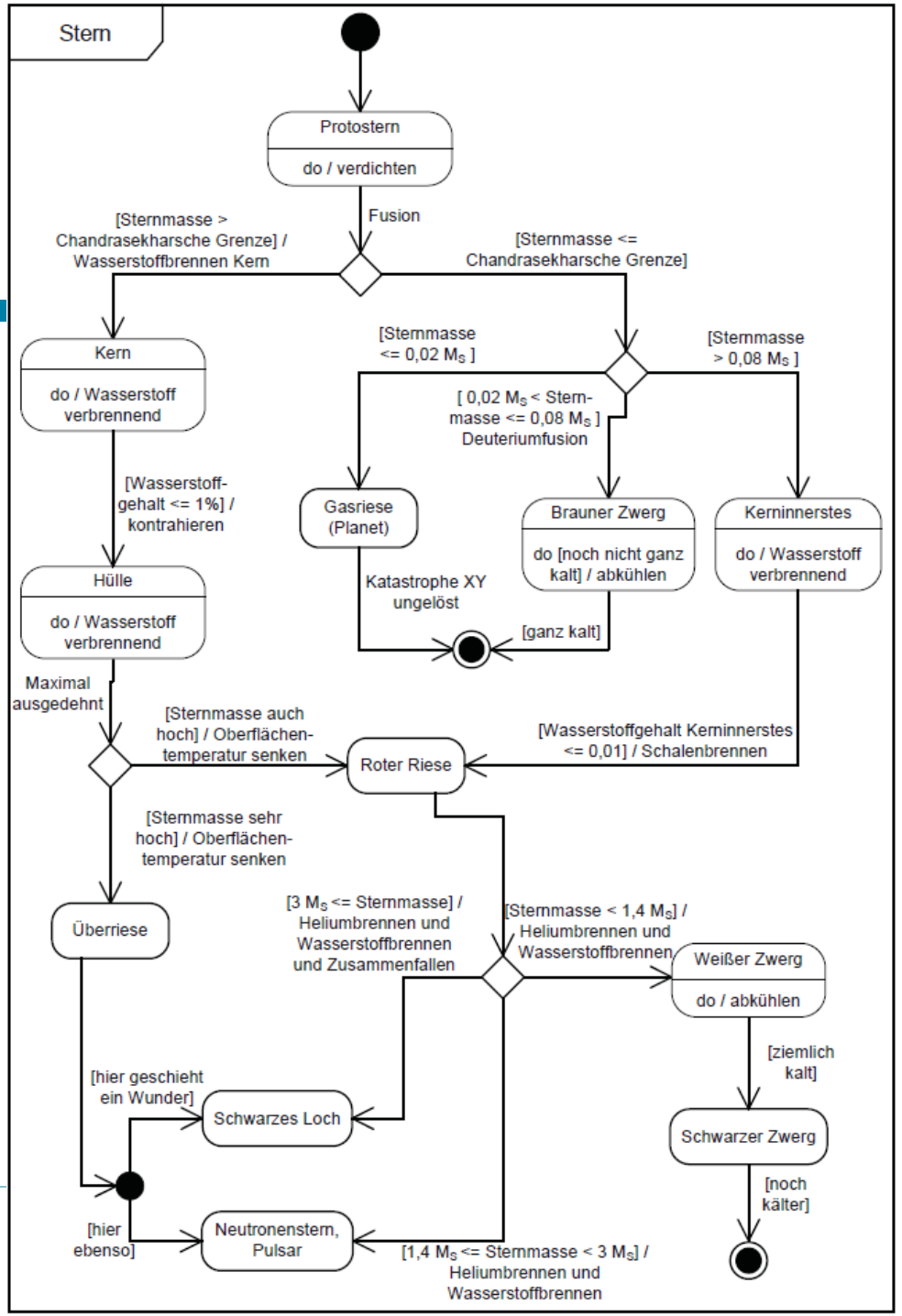
  Welche Zustände kann ein Objekt, eine Schnittstelle, ein Use Case, … bei welchen Ereignissen annehmen?

- Stärken:
  - Präzise Abbildung eines Zustandsmodells mit Zuständen, Ereignissen, Nebenläufigkeiten, Bedingungen, Ein- und Austrittsaktionen.
  - Schachtelung möglich.

Beispiel

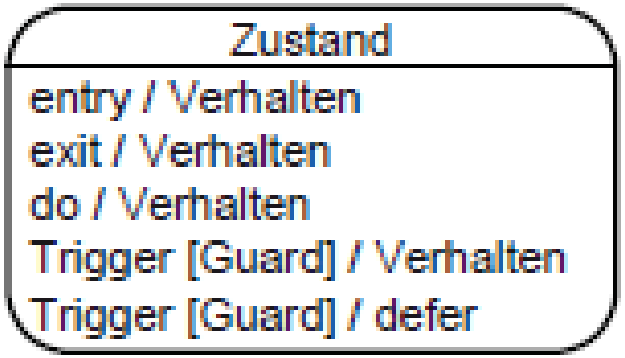

Name des Zustandsautomaten
Trigger
Guard
Fahrkartenautomat
Start-zustand
wartend
Betrag erhöht [Münze eingegeben]
Geldaufnahme
Do [Münze eingegeben] / Betrag erhöhen
[Betrag < Preis]
Zustand
Stop gedrückt
Cancel gewählt
Fahrkarte ausgewählt
Transition
Kreuzungs-punkt
Abbruch
entry / Betrag ausgeben
after (Timeout)
Berechnung
do / Preis berechnen
Ausgabe Fahrkarte
entry / Fahrkarte ausgeben
[Betrag = Preis]
[Betrag > Preis]
Eintritts-verhalten
Ausgabe Wechselgeld
do / Wechselgeld ausgeben
TimeTrigger
Zustands-verhalten

Beispiel

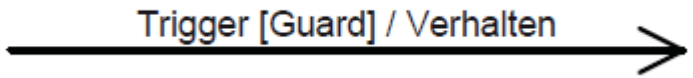

Stern
Protostern
do / verdichten
Fusion
[Sternmasse > Chandrasekharsche Grenze] / Wasserstoffbrennen Kern
[Sternmasse <= Chandrasekharsche Grenze]
Kern
do / Wasserstoff verbrennend
[Sternmasse <= 0,02 MS]
[Sternmasse > 0,08 MS]
[0,02 MS < Sternmasse <= 0,08 MS] Deuteriumfusion
[Wasserstoffgehalt <= 1%] / kontrahieren
Gasriese (Planet)
Brauner Zwerg
do [noch nicht ganz kalt] / abkühlen
Kerninnerstes
do / Wasserstoff verbrennend
Hülle
do / Wasserstoff verbrennend
Katastrophe XY ungelöst
[ganz kalt]
Maximal ausgedehnt
[Sternmasse auch hoch] / Oberflächentemperatur senken
Roter Riese
[Wasserstoffgehalt Kerninnerstes <= 0,01] / Schalenbrennen
[Sternmasse sehr hoch] / Oberflächentemperatur senken
Überriese
[3 MS <= Sternmasse] / Heliumbrennen und Wasserstoffbrennen und Zusammenfallen
[Sternmasse < 1,4 MS] / Heliumbrennen und Wasserstoffbrennen
Weißer Zwerg
do / abkühlen
[ziemlich kalt]
[hier geschieht ein Wunder]
Schwarzes Loch
Schwarzer Zwerg
[noch kälter]
[hier ebenso]
Neutronenstern, Pulsar
[1,4 MS <= Sternmasse < 3 MS] / Heliumbrennen und Wasserstoffbrennen

Notation

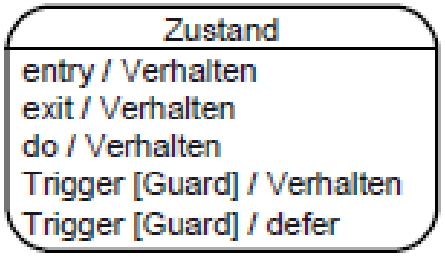
- Zustandsinternes Verhalten:



- Transitionen:



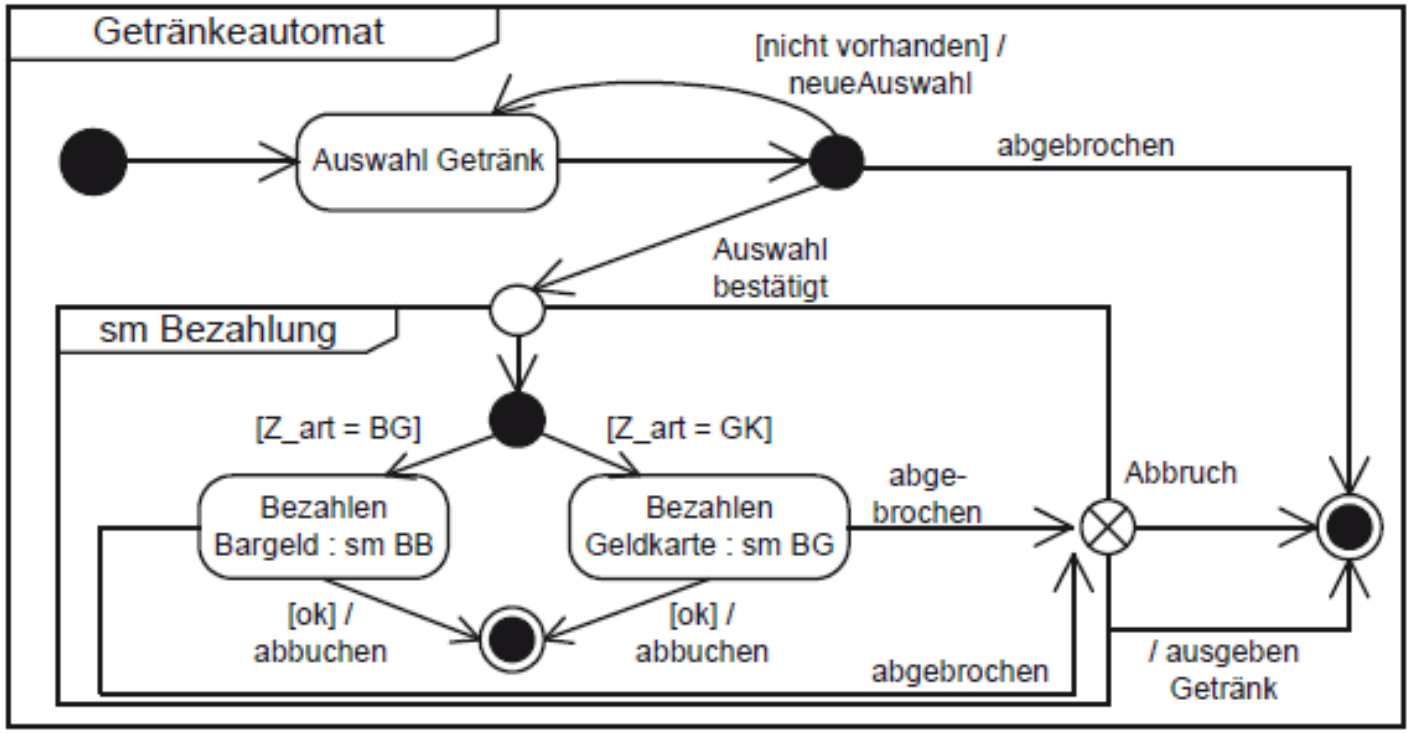
Notation



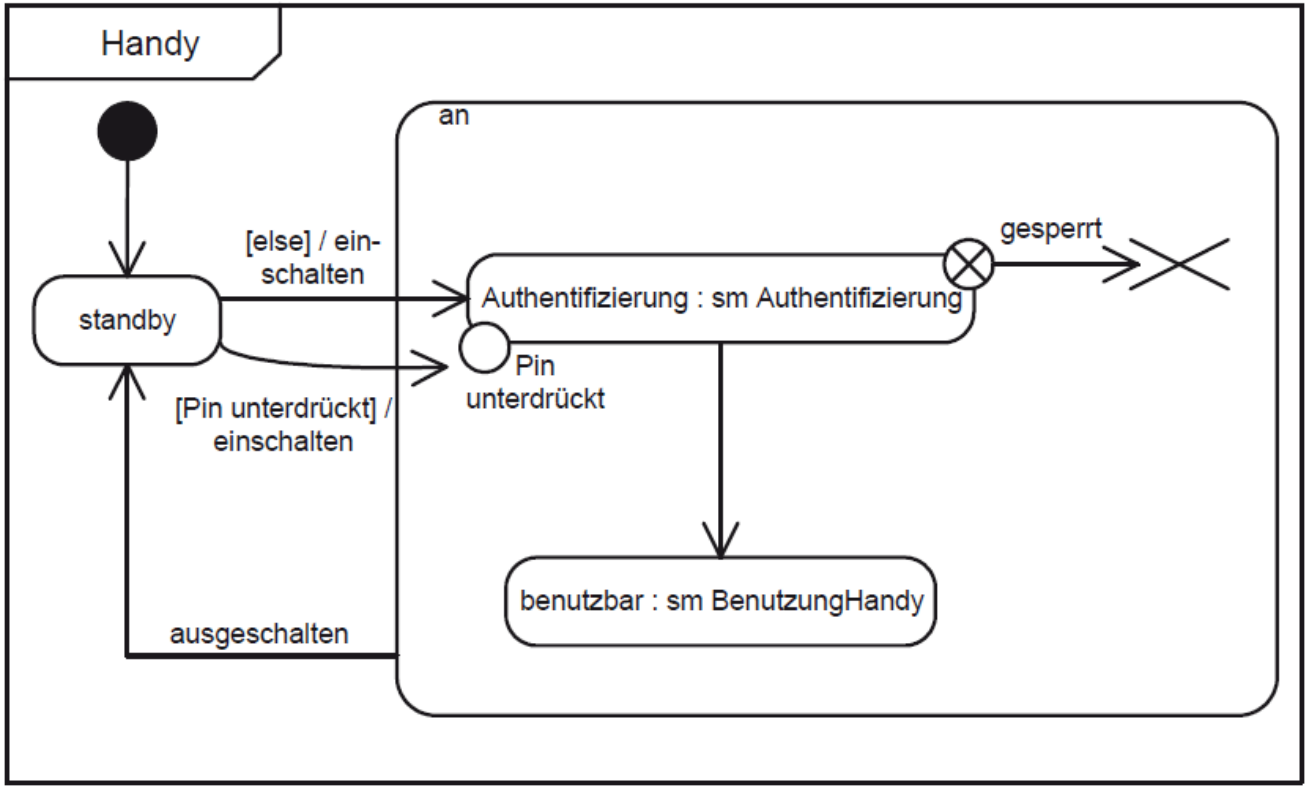
- `defer` führt zur verzögerten Betrachtung des Triggers in Folgeaktivitäten.
- (Rupp, 2012), S. 345:

> „Wenn ein Zustand in einem Zustandsautomaten aktiv ist und ein Trigger für ihn erzeugt wird, der in keiner ihm zugeordneten Transition beschrieben ist, so verfällt dieser Trigger. Dieser Trigger löst keine Transition aus. Dies gilt auch, wenn das Modell zu einem späteren Zeitpunkt einen Zustand erreicht, in dem eine entsprechende Transition definiert ist. Soll diese durchlaufen werden, muss ein entsprechender Trigger neu erzeugt werden. Der Trigger verfällt ebenso ohne Auswirkung, wenn zwar beim aktiven Zustand eine Transition mit ihm definiert ist, jedoch die Guard bei der Transition aktuell nicht auf True ausgewertet ist. Möchten Sie diesen Fall vermeiden, so haben Sie die Möglichkeit, die Trigger, die in einem bestimmten Zustand nicht verfallen sollen, durch ein „defer" zu verzögern. Der Trigger wird dann nach dem Abarbeiten der nachfolgenden Transition erneut betrachtet."

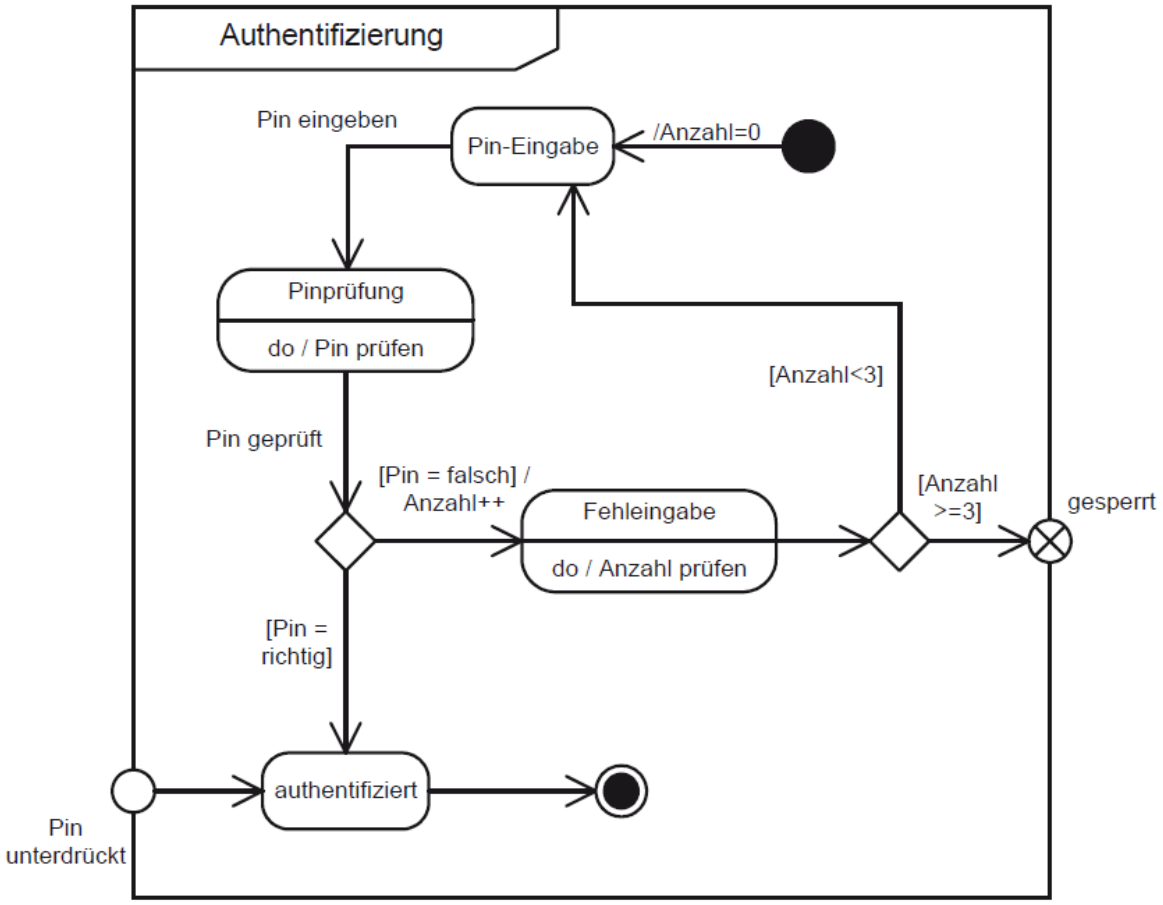
Beispiel Unterzustandsautomat



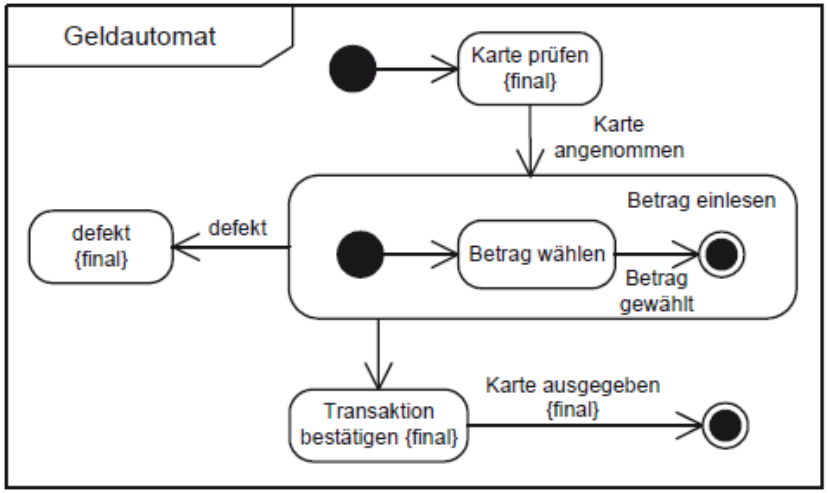
Beispiel Unterzustandsautomat

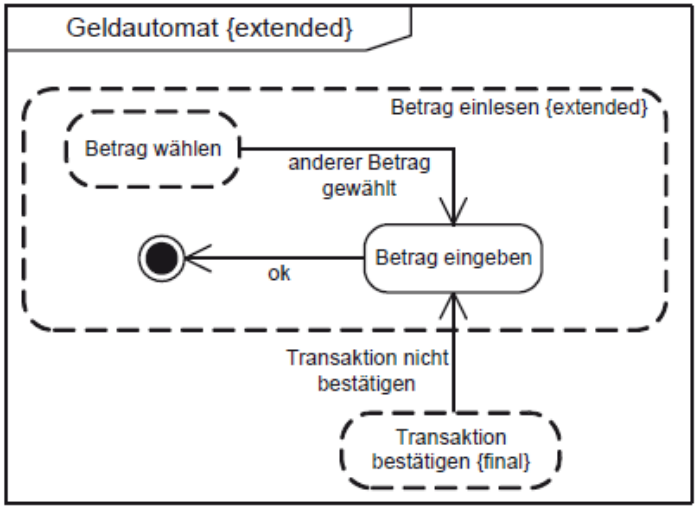


Beispiel Unterzustandsautomat



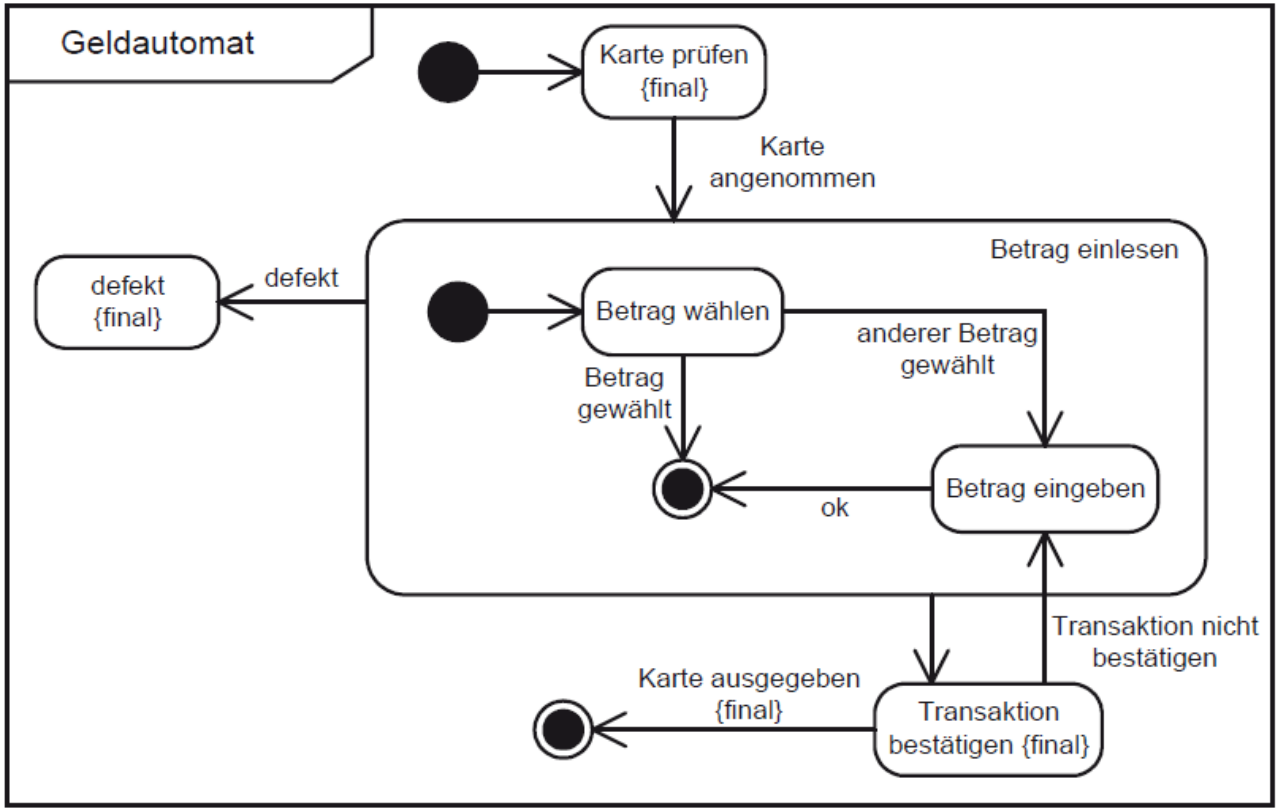
Beispiel Spezialisierung



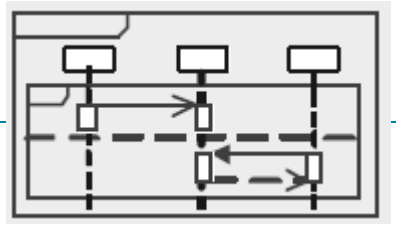


Beispiel Spezialisierung

- Oder der „komplette“ Zustandsautomat „Geldautomat“:



Sequenzdiagramm

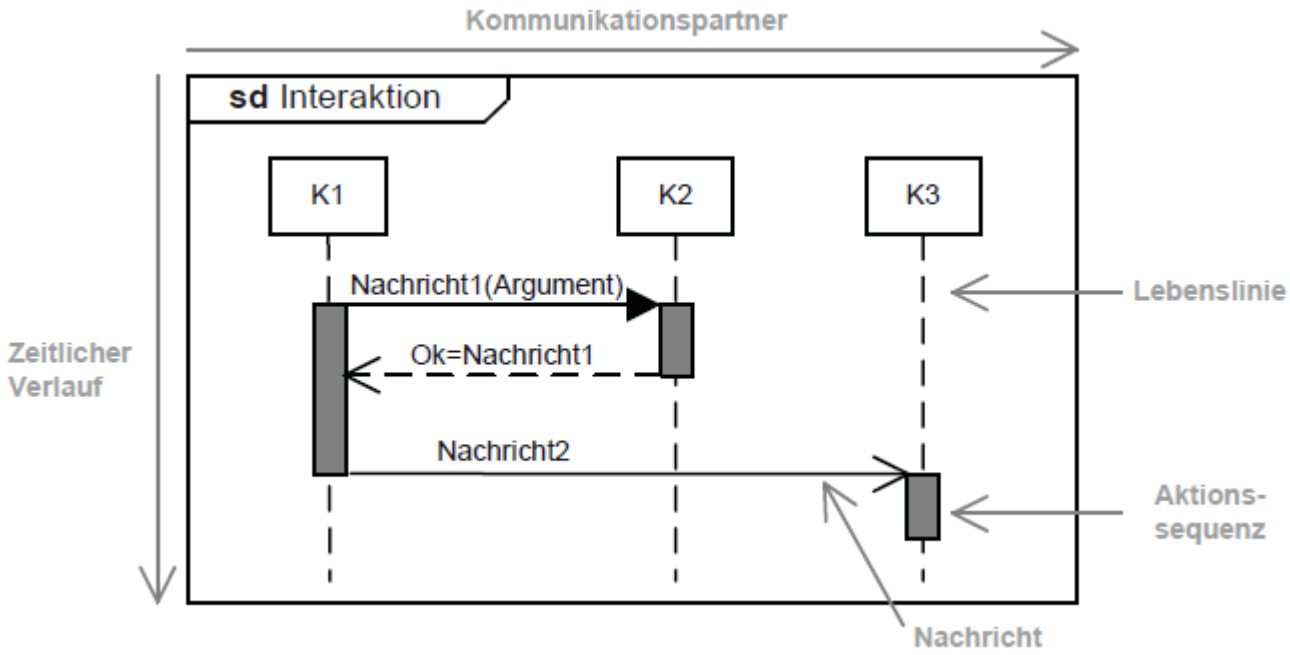


- Zentrale Frage:

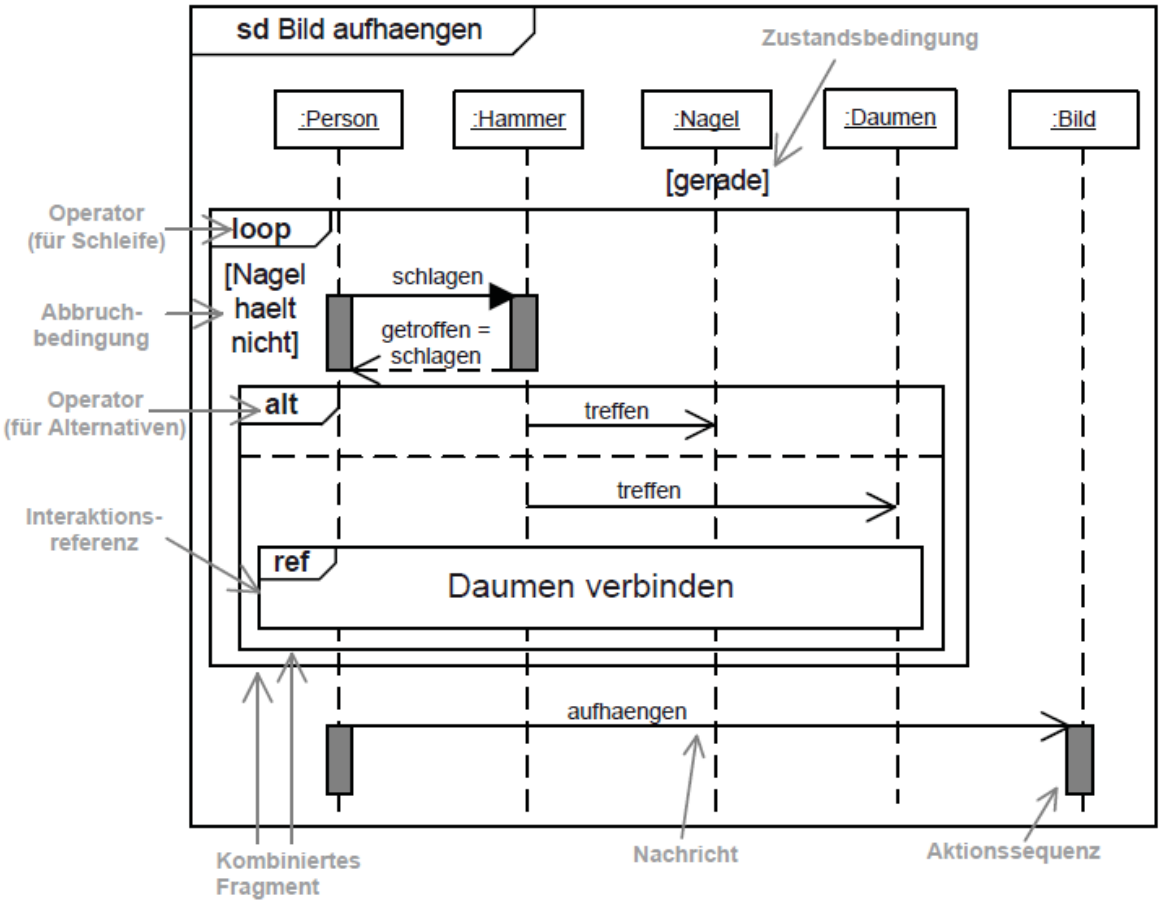
  Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?

- Stärken:
  - Stellt den zeitlichen Ablauf des Informationsaustausches zwischen Kommunikationspartnern dar.
  - Schachtelung und Flusssteuerung (Bedingungen, Schleifen, Verzweigungen) möglich.

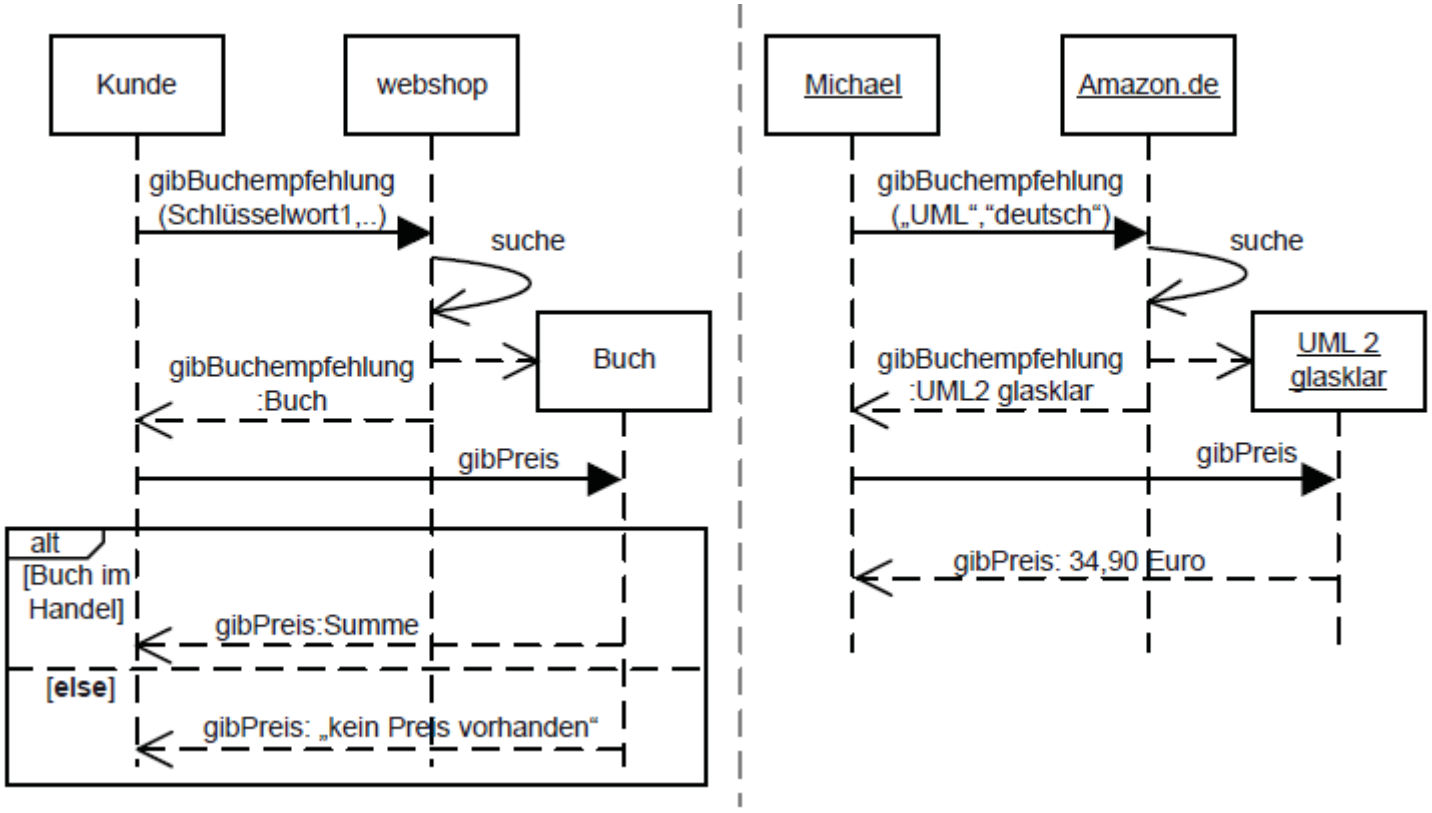
Beispiel


Kommunikationspartner
sd Interaktion
K1
K2
K3
Nachricht1(Argument)
Ok=Nachricht1
Nachricht2
Lebenslinie
Zeitlicher Verlauf
Aktions-sequenz
Nachricht

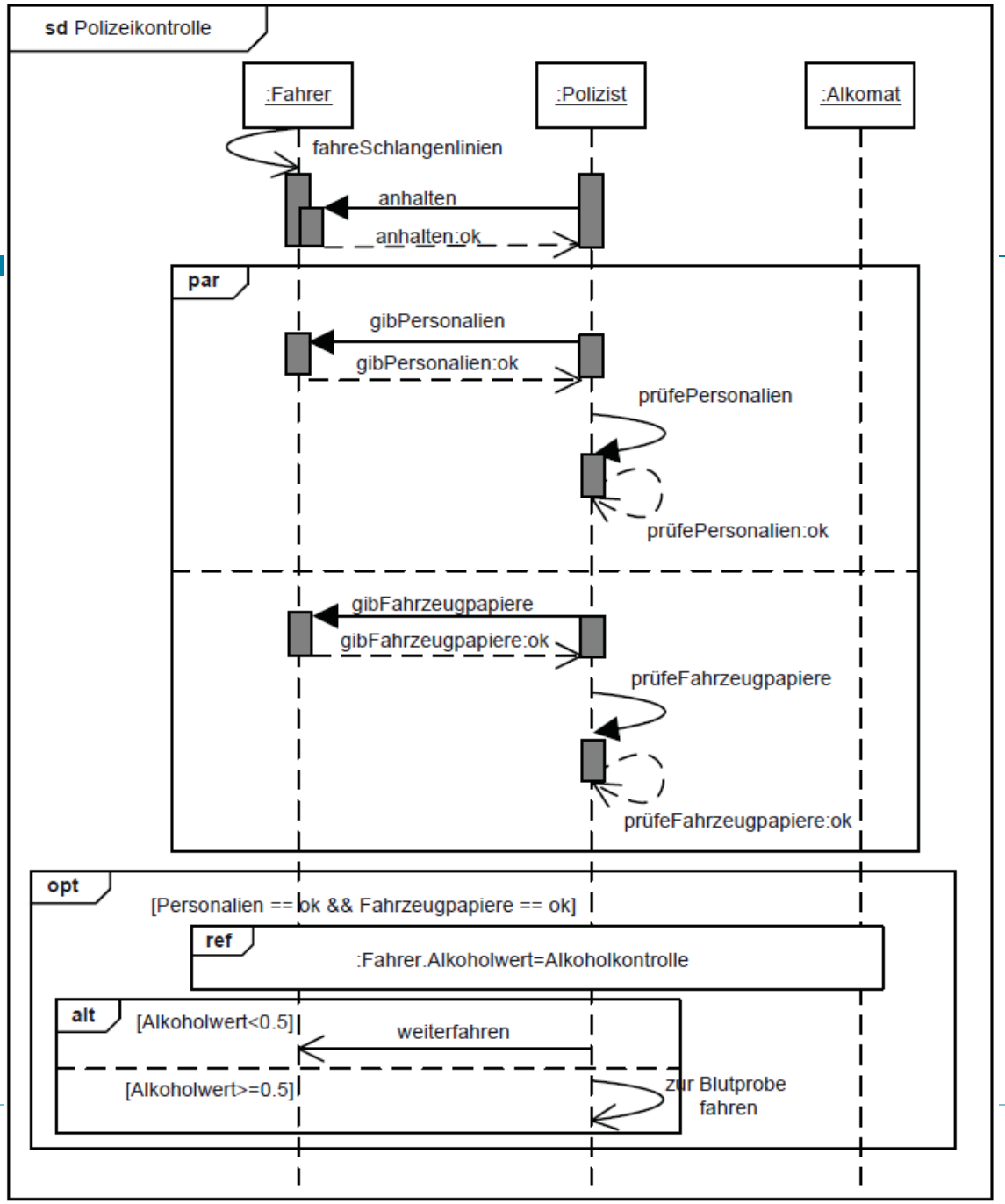
Beispiel


sd Bild aufhaengen
Zustandsbedingung
:Person
:Hammer
:Nagel
:Daumen
:Bild
[gerade]
Operator (für Schleife)
loop
[Nagel haelt nicht]
Abbruch-bedingung
schlagen
getroffen = schlagen
Operator (für Alternativen)
alt
treffen
treffen
Interaktions-referenz
ref
Daumen verbinden
aufhaengen
Kombiniertes Fragment
Nachricht
Aktionssequenz

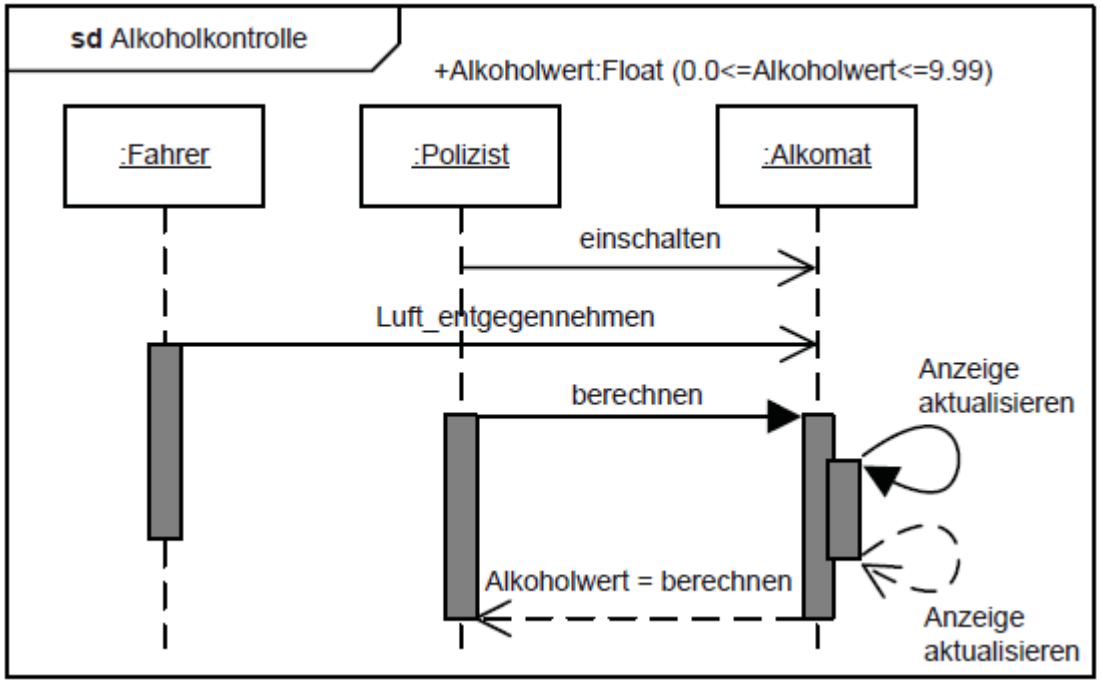
Beispiel


Kunde
webshop
gibBuchempfehlung
(Schlüsselwort1,..)
suche
Buch
gibBuchempfehlung
:Buch
gibPreis
alt
[Buch im
Handel]
gibPreis:Summe
[else]
gibPreis: „kein Preis vorhanden“
Michael
Amazon.de
gibBuchempfehlung
(„UML“,“deutsch“)
suche
UML 2
glasklar
gibBuchempfehlung
:UML2 glasklar
gibPreis
gibPreis: 34,90 Euro

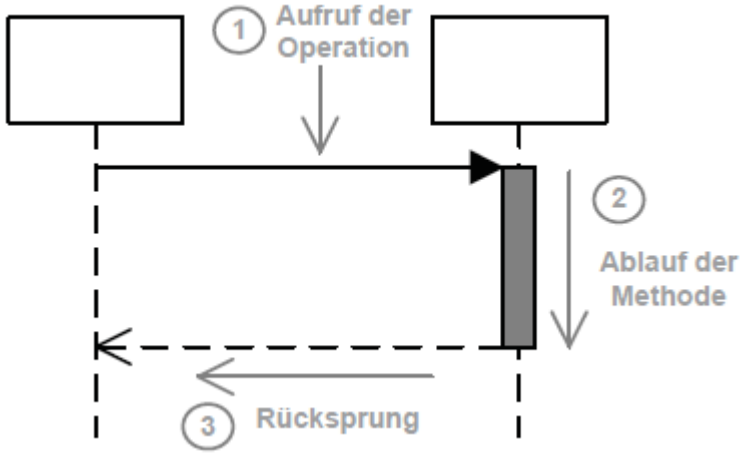
Beispiel


sd Polizeikontrolle
:Fahrer
:Polizist
:Alkomat
fahreSchlangenlinien
anhalten
anhalten:ok
par
gibPersonalien
gibPersonalien:ok
prüfePersonalien
prüfePersonalien:ok
gibFahrzeugpapiere
gibFahrzeugpapiere:ok
prüfeFahrzeugpapiere
prüfeFahrzeugpapiere:ok
opt
[Personalien == ok && Fahrzeugpapiere == ok]
ref
:Fahrer.Alkoholwert=Alkoholkontrolle
alt
[Alkoholwert<0.5]
weiterfahren
[Alkoholwert>=0.5]
zur Blutprobe fahren

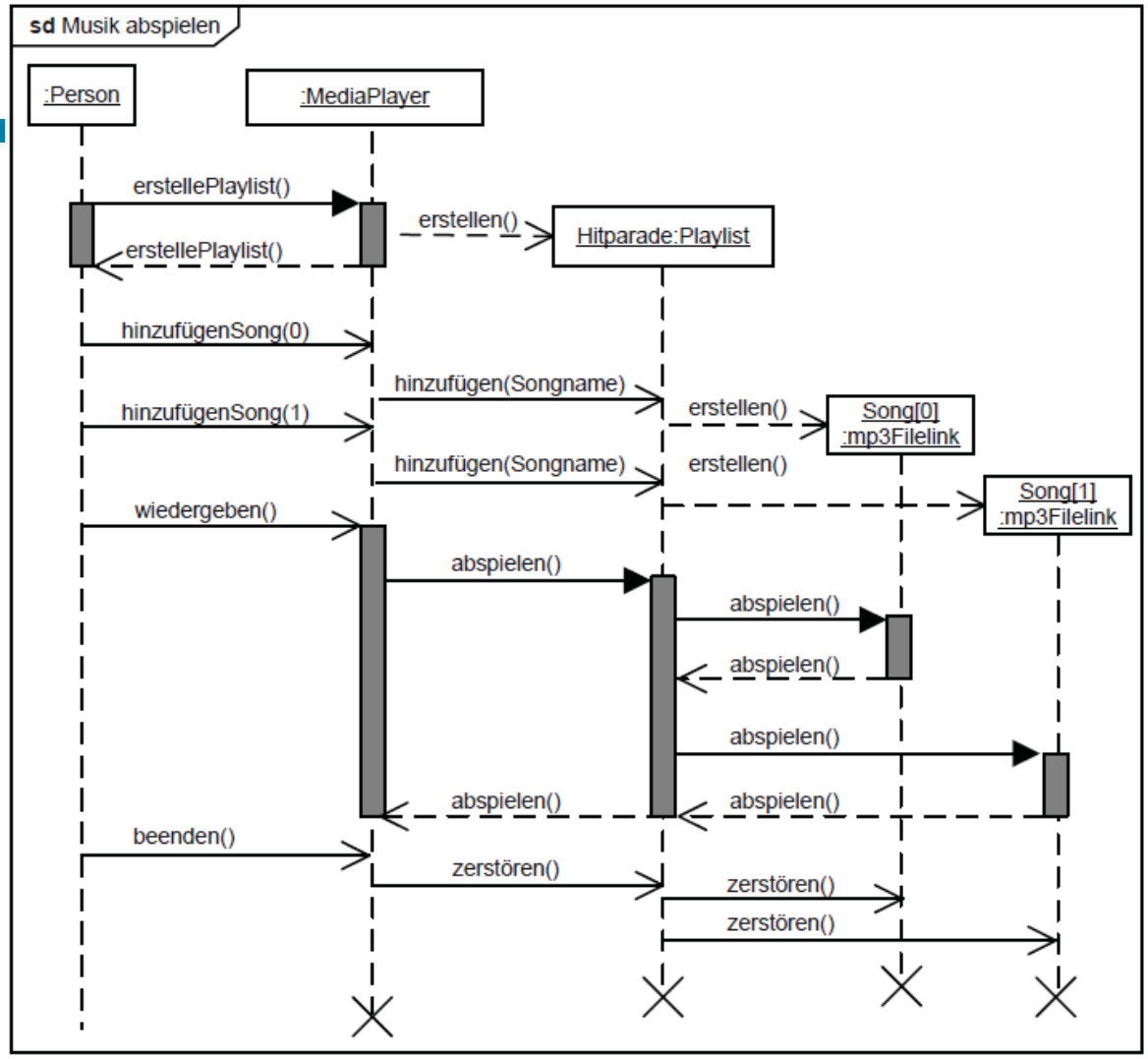
Beispiel


sd Alkoholkontrolle
+Alkoholwert:Float (0.0<=Alkoholwert<=9.99)
:Fahrer
:Polizist
:Alkomat
einschalten
Luft_entgegennehmen
berechnen
Anzeige aktualisieren
Alkoholwert = berechnen
Anzeige aktualisieren

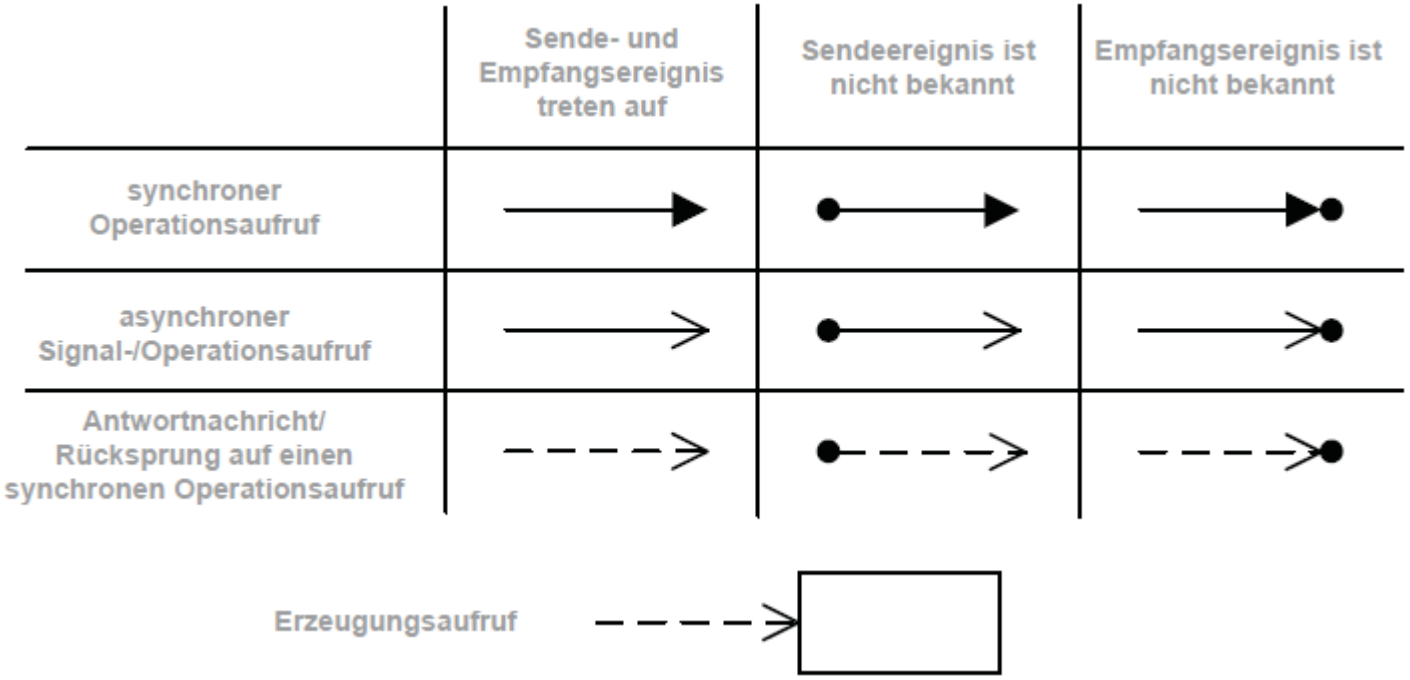
Notation



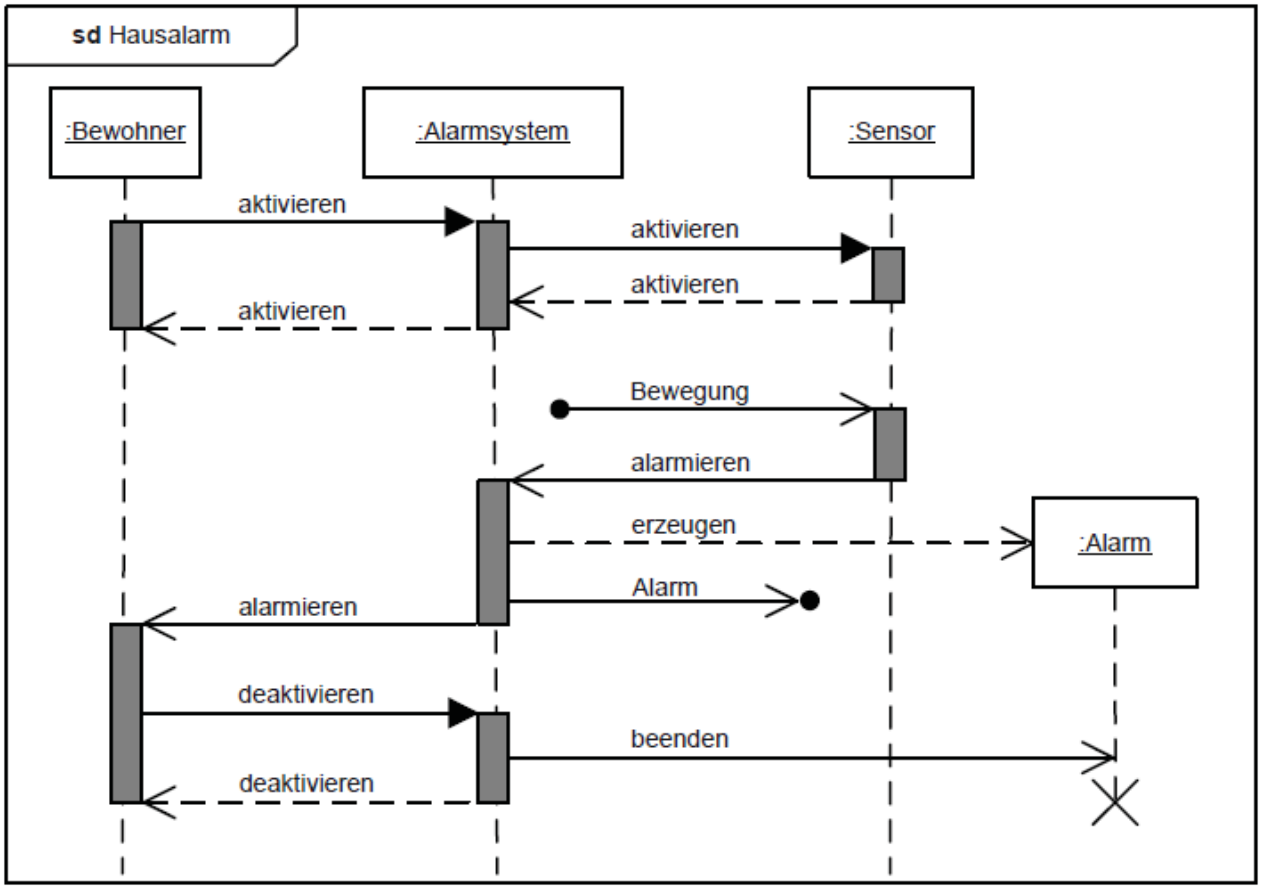
Beispiel


sd Musik abspielen
:Person
:MediaPlayer
erstellePlaylist()
erstellen()
Hitparade:Playlist
erstellePlaylist()
hinzufügenSong(0)
hinzufügen(Songname)
erstellen()
Song[0]
:mp3Filelink
hinzufügenSong(1)
hinzufügen(Songname)
erstellen()
Song[1]
:mp3Filelink
wiedergeben()
abspielen()
abspielen()
abspielen()
abspielen()
abspielen()
abspielen()
beenden()
zerstören()
zerstören()
zerstören()

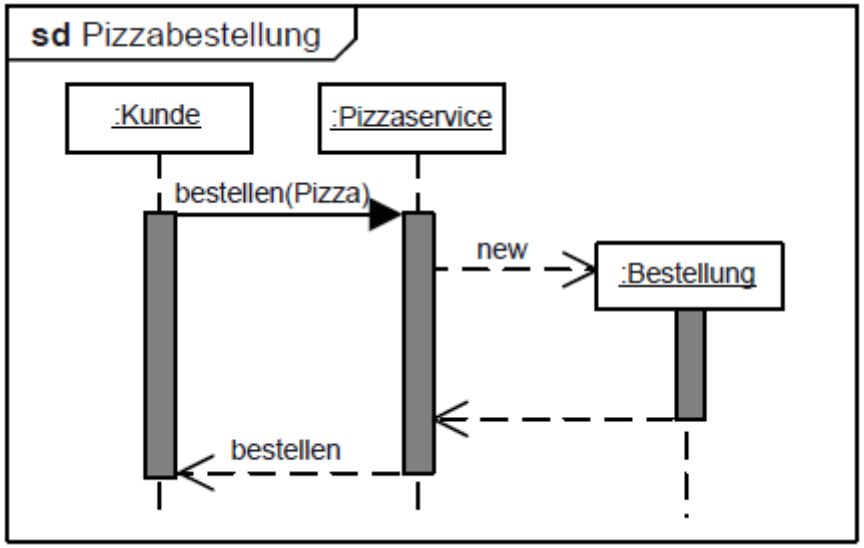
Notation


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

Beispiel


sd Hausalarm
:Bewohner
:Alarmsystem
:Sensor
:Alarm
aktivieren
aktivieren
aktivieren
aktivieren
Bewegung
alarmieren
erzeugen
Alarm
alarmieren
deaktivieren
beenden
deaktivieren

Beispiel

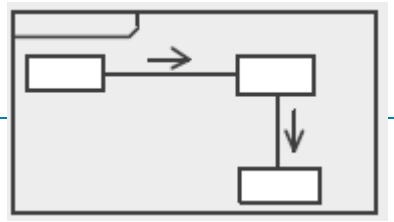


- Die Angabe von „new“ ist für Erzeugungsnachrichtigen in UML nicht spezifiziert, wird aber von (Rupp, 2012) empfohlen.

In der Praxis…

- In der Praxis sind Sequenzdiagramme beliebt.
- Die Sequenzen sind häufig konkret, d.h.:
  - kein `alt`, `opt`, `par` etc.
  - Normaldurchlauf und Fehlerfall wird in verschiedene Sequenzdiagramme aufgeteilt.
  - Nur synchrone/synchronisierte Aufrufe
  - Keine unbekannten Sender oder Empfänger
- Wird zu viel in Sequenzdiagramme gepackt, erschwert das die Architekturkommunikation.
- Zudem stellt sich dann auch immer die Frage, ob man nicht gleich ein Code-Review besser ist, als die Durchsicht eines (zusätzlichen) UML-Diagramms.

Kommunikations-diagramm

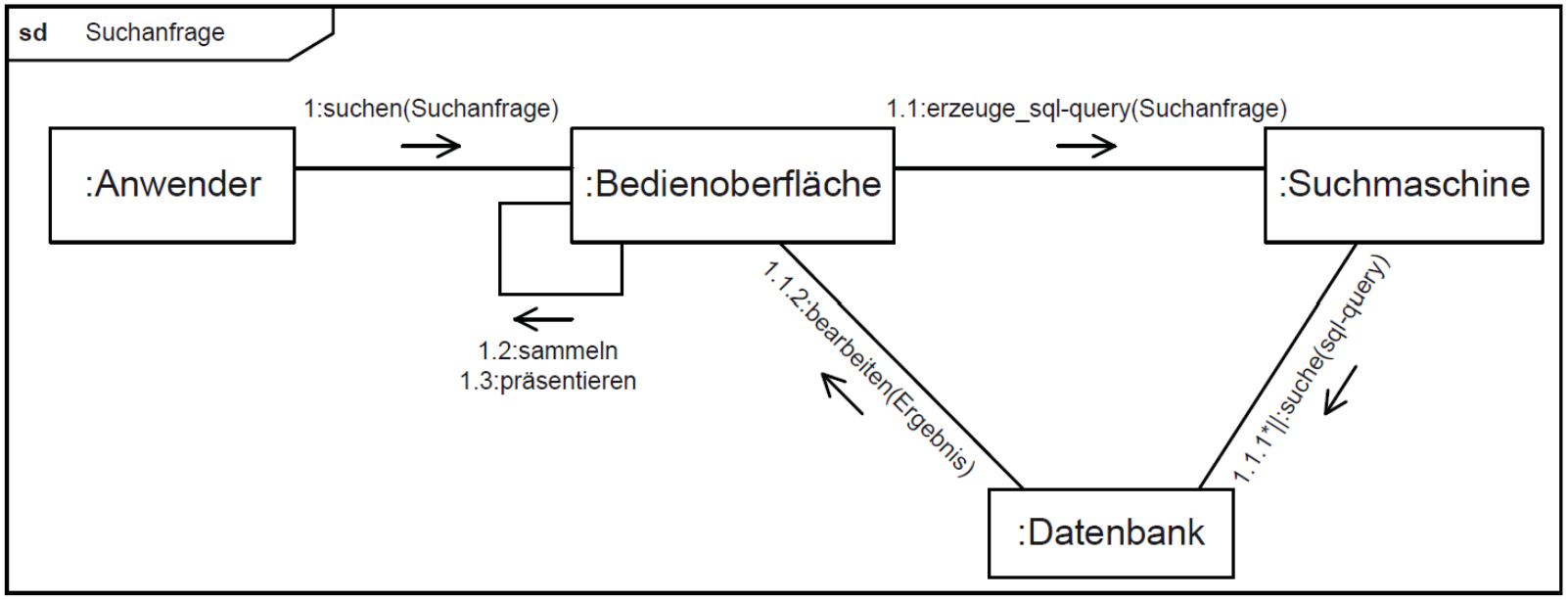


- Zentrale Frage:

  Wer kommuniziert mit wem? Wer „arbeitet“ im System zusammen?

- Stärken:
  - Stellt den Informationsaustausch zwischen Kommunikationspartnern dar.
  - Überblick steht im Vordergrund (Details und zeitliche Abfolge weniger wichtig).

Beispiel


sd Suchanfrage
1:suchen(Suchanfrage)
1.1:erzeuge_sql-query(Suchanfrage)
:Anwender
:Bedienoberfläche
:Suchmaschine
1.2:sammeln
1.3:präsentieren
1.1.2:bearbeiten(Ergebnis)
1.1.1*[]:suche(sql-query)
:Datenbank

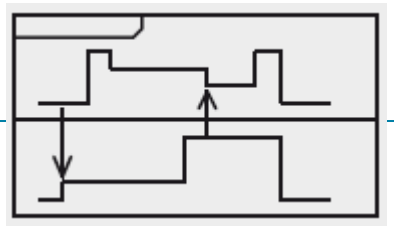
Vergleich zu Sequenzdiagrammen

- Kommunikationsdiagramme bestimmen primär die Kommunikationspartner und ihre Abhängigkeiten.
- Sequenzdiagramme haben wesentlich präzisere Spezifikationsmöglichkeiten. Der Fokus liegt stärker auf Abläufe.
- Sie sind damit mächtiger.
- Kommunikationsdiagramme sind hingegen schlichter, können aber ggf. einen besseren Überblick schaffen.

Beispiel

- Todo, Gegenüberstellung von Sequenz- und Kommunikationsdiagramm

Timingdiagramm

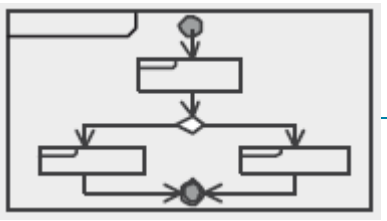


- Zentrale Frage:

  Wann befinden sich verschiedene Interaktionspartner in welchem Zustand?

- Stärken:
  - Visualisiert das exakte zeitliche Verhalten von Klassen, Schnittstellen, …
  - Geeignet für Detailbetrachtungen, bei denen es überaus wichtig ist, dass ein Ereignis zum richtigen Zeitpunkt eintritt.

Interaktionsübersichts-diagramm



- Zentrale Frage:

  Wann läuft welche Interaktion ab?

- Stärken:
  - Verbindet Interaktionsdiagramme (Sequenz-, Kommunikations- und Timingdiagramme) auf Top-Level-Ebene.
  - Hohes Abstraktionsniveau.
  - Gut geeignet als Strukturierung der Interaktionsdiagramme.

Bewertung

- UML ist ein Standard und führt damit zu einer einheitlicheren Beschreibung von Software.
- UML lässt aber auch viele Freiheiten, dies zu tun.
- Templates für Projekte sind hilfreich, um weiter zu vereinheitlichen.

Literatur

- Balzert, H., 2005. *Lehrbuch der Objektmodellierung, 2. Auflage*.
- OMG, 2017. *OMG Unified Modeling Language*. [Online] Available at: https://www.omg.org/spec/UML/2.5.1/PDF
- Rupp, C. , Queins, S. & die SOPHISTen, 2012. *UML 2 glasklar, 4. Auflage.*