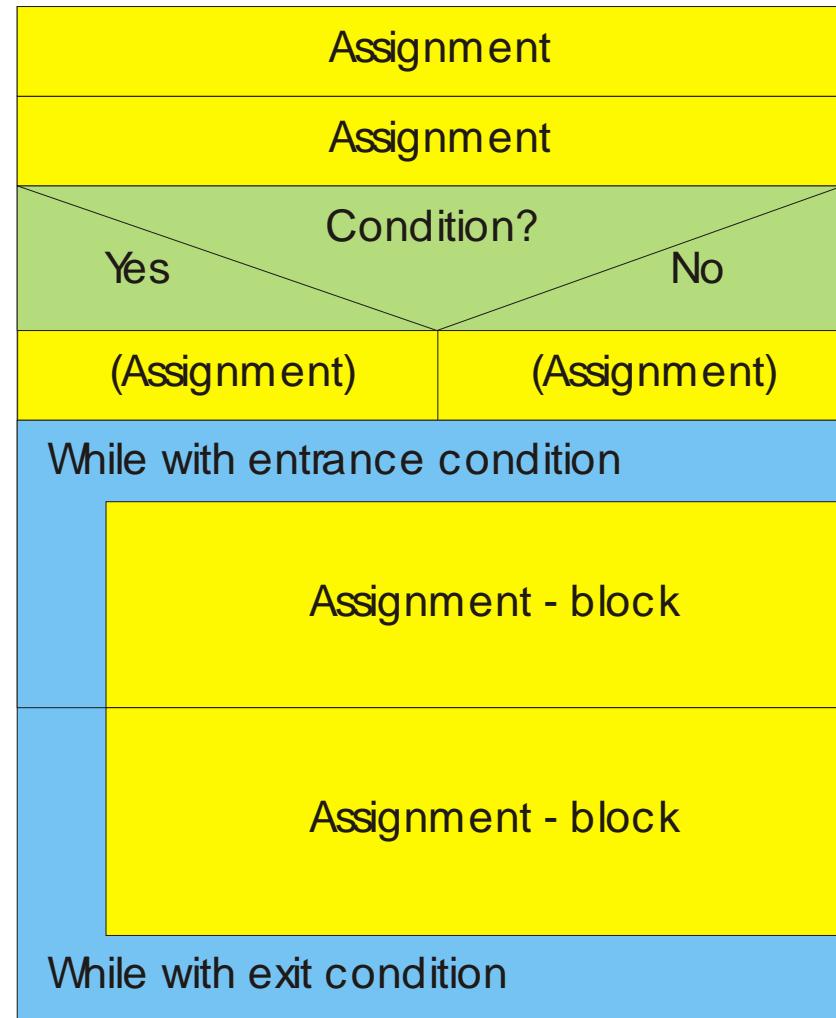


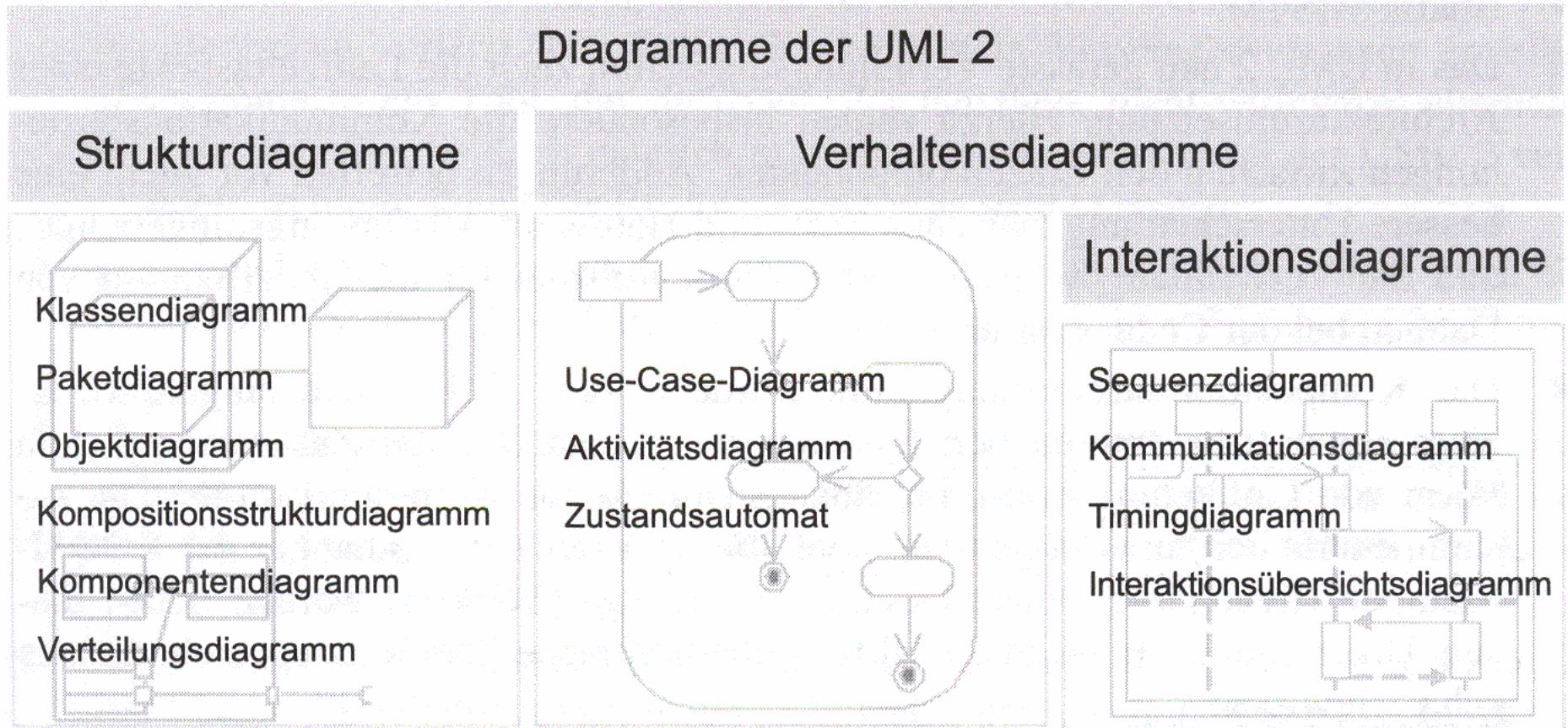
UML (Unified Modelling Language)

Nassi-Shneiderman Strukturdiagramm

- Das Nassi-Shneiderman Diagramm ist ein betagter Vertreter für eine ganze Reihe von Strukturdiagrammen zur Darstellung prozeduraler Algorithmen und Programmabläufe
- Eine Interaktion zwischen Klassen oder die Darstellung von Eigenschaften kann mit dieser Art Diagramm nicht erfolgen.



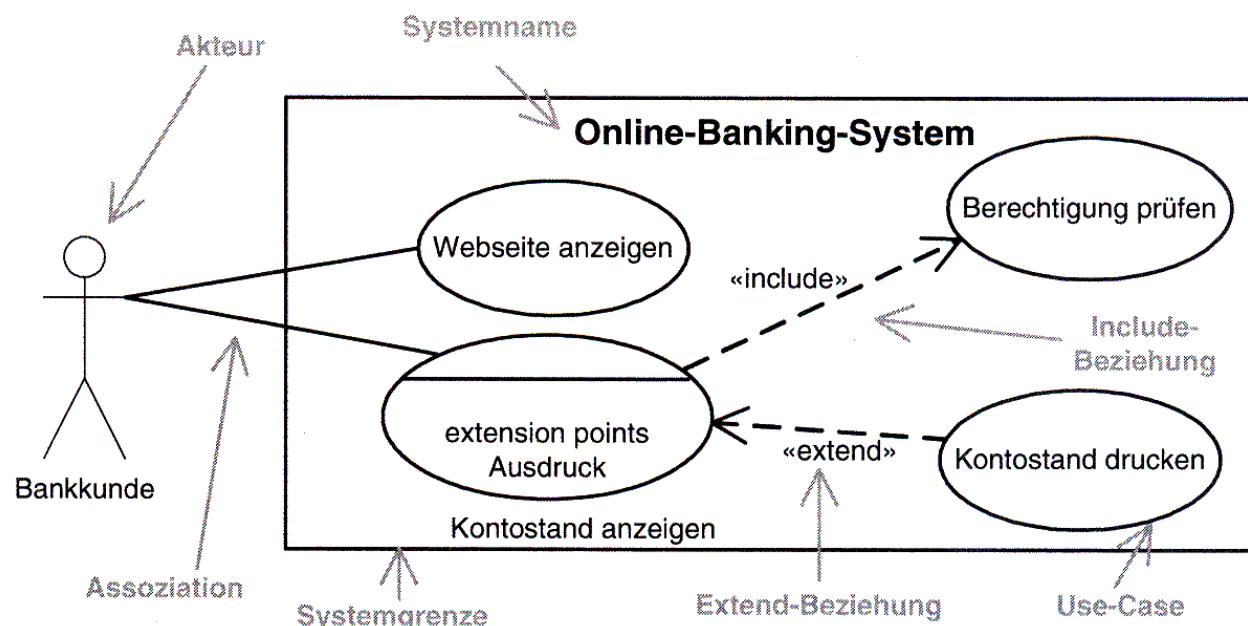
Diagrammtypen der UML



Quelle: Rupp et.al.

Das Use-Case-Diagramm

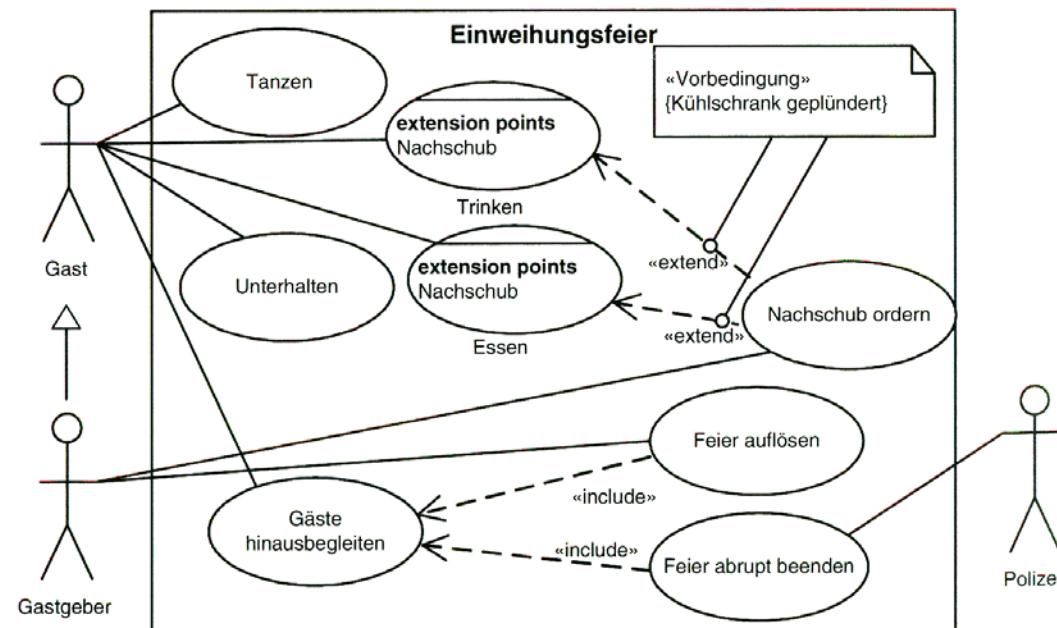
- Use-Cases zeigen das nach außen sichtbare Verhalten eines Elements
- Die wesentlichen Darstellungs-Elemente sind
 - der Akteur
 - das System
 - und die angebotenen Anwendungsfälle (Use-Cases)



Quelle: Rupp et.al.

Anwendungsbereiche

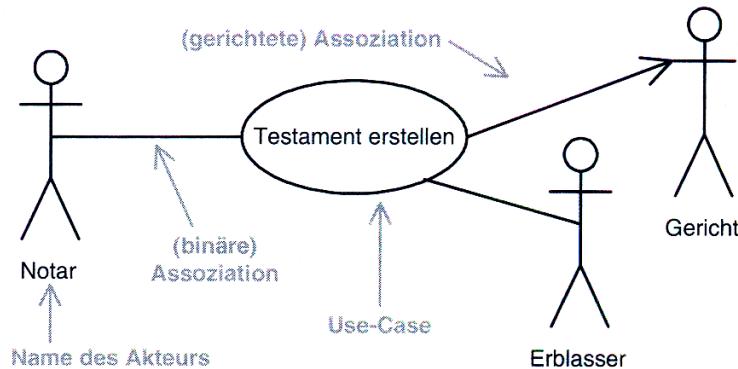
- Aufzeigen der funktionalen Dienstleistungen eines Systems oder einer Komponente
- Hilfsmittel zum Zerlegen eines Systems aus Nutzersicht in logische Teile
- Modellierung der Außenschnittstellen eines Systems
- Einfache und leicht verständliche Darstellung komplexer Systeme mit hohem Abstraktionsgrad
- Festlegen planbarer Einheiten bzw. Elemente für die Entwicklung



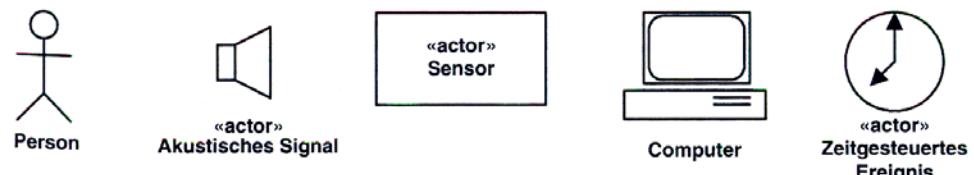
Quelle: Rupp et.al.

Notationselemente - Akteure

- Gerichte Assoziationen

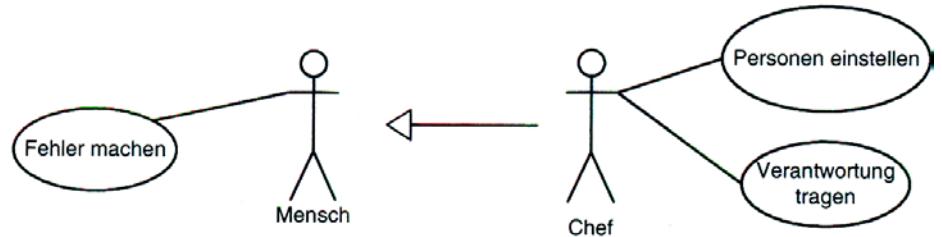
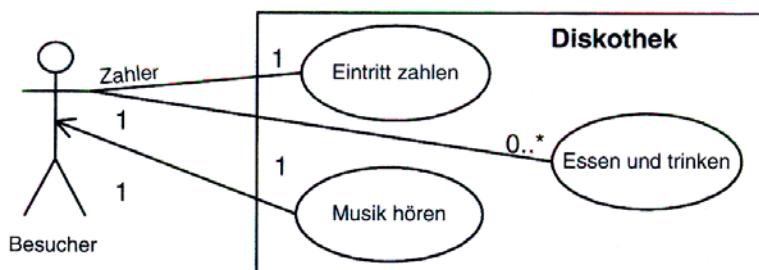


- Akteur-Typen



- Beziehungen zwischen Akteuren – Spezialisierung / Generalisierung

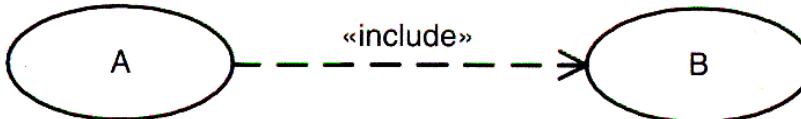
- Assoziationen mit Multiplizitäten



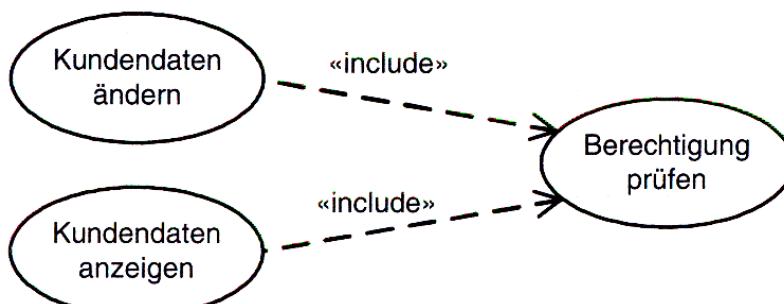
Quelle: Rupp et.al.

Notationselemente Use-Cases

- Verhalten eines anderen Use-Case importieren bzw. verwenden

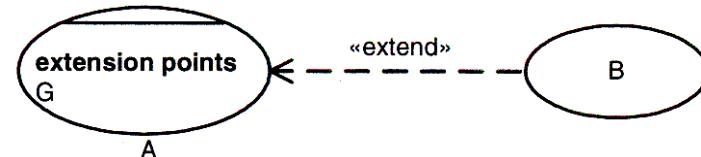


- Dadurch können gemeinsam genutzte Anwendungsfälle bei entsprechend hoher Auflösung gefunden werden – Vermeiden von Redundanz. Ein include kann auch mehrstufig sein

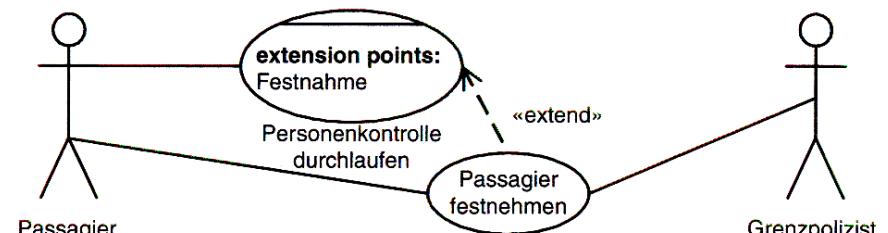


- Ein include-UC wird immer, ein extend-UC nur unter bestimmten Bedingungen ausgeführt

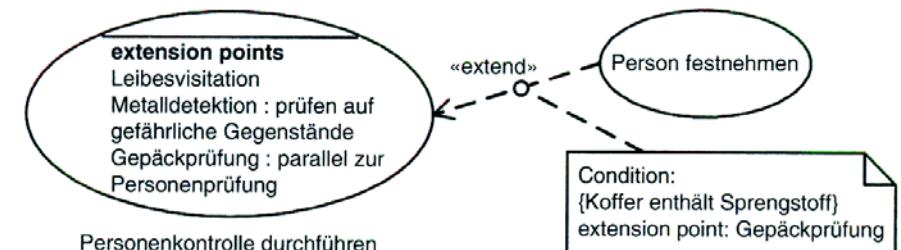
- Ein Use-Case kann einen anderen erweitern



- Der Passagier interagiert mit dem Use-Case "Personenkontrolle durchlaufen". Dieser kann durch eine Festnahme erweitert werden.



- Die Umstände, wann das geschehen soll, können dargestellt werden



Das Aktivitäten-Diagramm

- Mit dem Aktivitäten-Diagramm werden Abläufe beschrieben, dabei werden folgende Aspekte berücksichtigt
 - Start und Ende einer Aktivität
 - Verzweigungen
 - Bedingungen
- Mit Aktivitätsdiagrammen werden damit Regeln für die Abläufe beschrieben, d.h. alle möglichen Abläufe, nicht jedoch ein individueller Ablauf
- Aktivitäten-Diagramme erinnern in ihrer Form an
 - Nassi-Shneidermann-Diagramme (Struktogramme)
 - Datenflussdiagramme
- Sie werden eingesetzt, um
 - Geschäftsprozesse darzustellen
 - Use-Cases (weiterer UML-Diagrammtyp) genauer zu beschreiben
 - Die Funktionsweise einer Operation genau darzustellen

Das Aktivitäten-Diagramm

- In Aktivitäten-Diagrammen werden folgende Elemente verwendet
 - Aktivitäten
 - Aktionen
 - Objektknoten
 - verbindende Kanten
 - Kontrollelemente für die Ablaufsteuerung
- Kontrollelemente
 - parallelisieren und synchronisieren
 - verzweigen und führen zusammen
 - Bedingungen zur Lenkung auswerten
 - mehrfache Prozesse instanziieren
 - asynchrone Ereignisse in Prozesse eingreifen
 - parametrisieren von Abläufen
 - verknüpfen von Abläufen mit Objekten

[Notationselemente für Aktivitäten-Diagramme]

■ Startknoten

- Der Startknoten markiert den Startpunkt einer Aktivität
- Eine Aktivität kann mehrere Startknoten besitzen
- Ein Startknoten kann mehrere wegführende Kanten besitzen
- Startknoten haben keine ankommenden Knoten



■ Endknoten

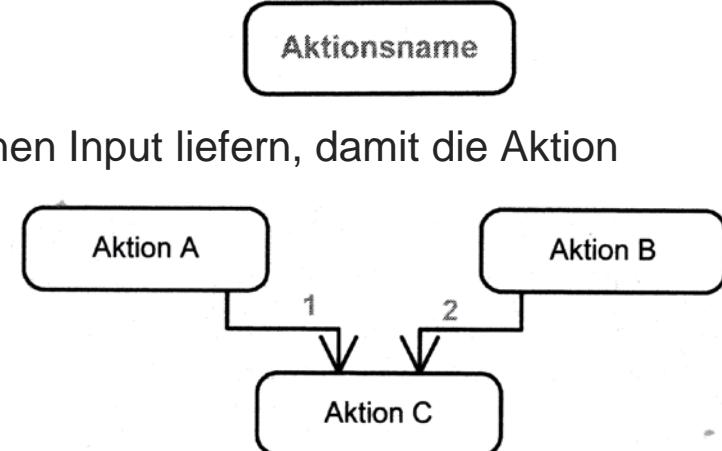
- Definieren den Endpunkt einer Aktivität
 - eine Aktivität kann mehrere Endknoten beinhalten
 - das Erreichen eines Endknotens beendet die gesamte Aktivität
- Als Endpunkt eines Kontrollflusses in veränderter Notation
 - terminiert einen Kontrollfluss, nicht die gesamte Aktivität
 - wenn ein Kontrollfluss nicht parallel abläuft, wird aber auch Aktivität beendet (Mehrdeutigkeit)
- Endknoten können mehrere hinführende Kanten besitzen



[Notationselemente für Aktivitäten-Diagramme]

- Aktion als das wesentliche Element des Aktivitäten-Diagramms, alle anderen Objekte dienen nur dazu, die Aktionen in einen gemeinsamen, logischen Kontext zu stellen

- Aktivitäten stehen über Kanten in Beziehung
 - Die Kanten initiieren den Start einer Aktion
 - Alle eingehenden Kanten einer Aktion müssen einen Input liefern, damit die Aktion ausgeführt werden kann, d.h. Aktion C wird nur dann ausgeführt, wenn sowohl Aktion A als auch Aktion B abgelaufen sind und ihren Input geliefert haben



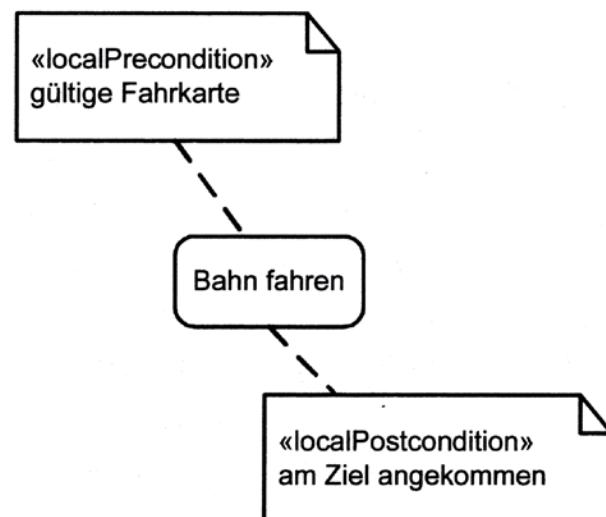
- Die Beschreibung der Aktion darf detailliert sein

- beliebiger Text darstellbar, ausführliche Beschreibung an dieser Stelle möglich
 - Ablaufbeschreibung des Vorgangs
 - Anlehnung an Programmcode

```
FOR jeden Gast  
anrufen  
einladen  
Zu/Absage einholen  
ENDFOR
```

Bedingungen für Aktionen

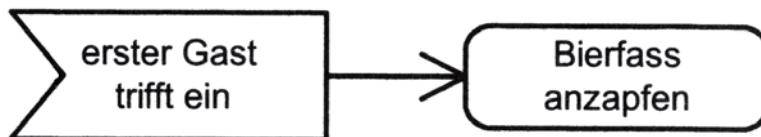
- Bedingungen werden in Form eines Kommentars an eine Aktion angehängt
 - Vorbedingungen erhalten das Schlüsselwort "localPrecondition"
 - Nachbedingungen "localPostcondition"
- Die Bedingungen drücken aus, dass vor Beginn der Aktion die Vorbedingung erfüllt sein muss, die Aktion kann erst dann verlassen werden, sofern die Nachbedingung erfüllt ist
 - Vor- und Nachbedingungen werden in einer konkreten Situation für ein konkret vorliegendes Objekt überprüft
 - Welche Aktionen erfolgen sollen, wenn die Bedingungen nicht eintreten, bleibt unklar -> Modellierungsschwäche, die in der Regel einer Verfeinerung bedarf



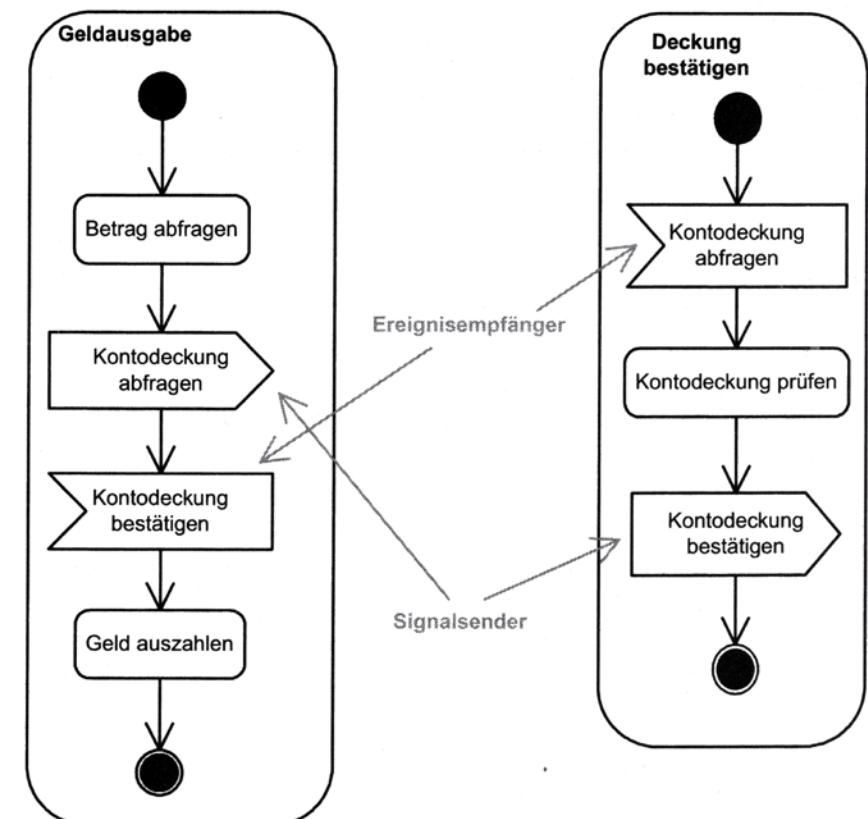
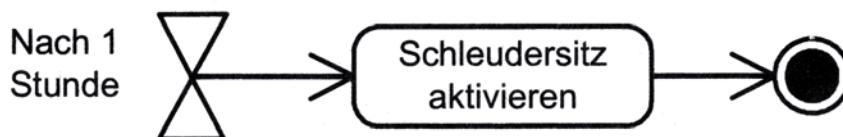
Quelle: Rupp et.al.

- Ähnlich wie Vor- und Nachbedingung

- Ein Signalsender erzeugt ein Signal, das an einen Empfänger versendet wird
- Der Signalempfänger wartet so lange, bis das erwartete Ereignis eintrifft, erst danach wird die Bearbeitung fortgesetzt
- Ein Signalsender kann als Startknoten eines Ablaufs verwendet werden



- Zeitliche Ereignisse können durch entsprechende Trigger modelliert werden, das Symbol ist eine stilisierte Eieruhr, der Zeittrigger kann durch Beschriftung entsprechend beschrieben werden

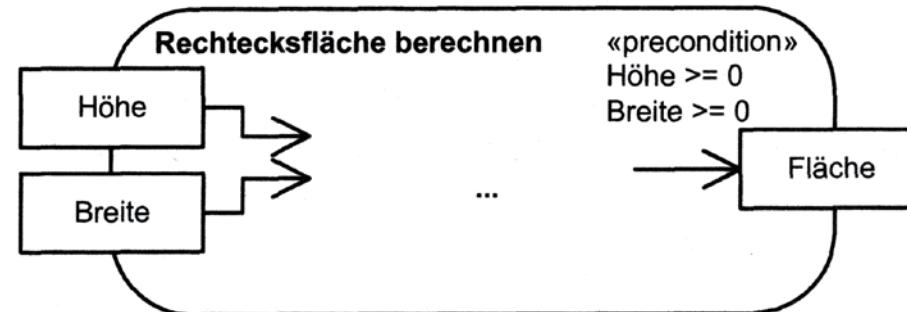


Quelle: Rupp et.al.

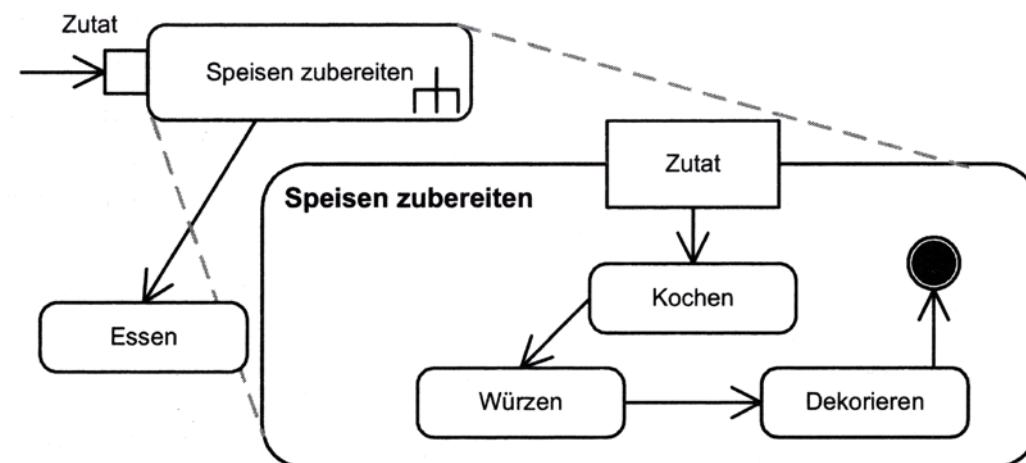
[Erweiterte Notation einer Aktivität]

- Die Darstellung einer Aktivität als abgerundetes Rechteck kann durch weitere Elemente erweitert werden

- Eingangsparameter
- Ausgangsparameter
- Vorbedingung
- Nachbedingung



- Die Aktionen innerhalb einer Aktivität können selbst weitere Aktivitäten aufrufen und damit verschachtelt dargestellt werden
 - komplexe Aktivitäten können auf unterschiedlichen Ebenen unterschiedlich detailliert abgebildet werden, ohne dass der Gesamtzusammenhang verloren geht

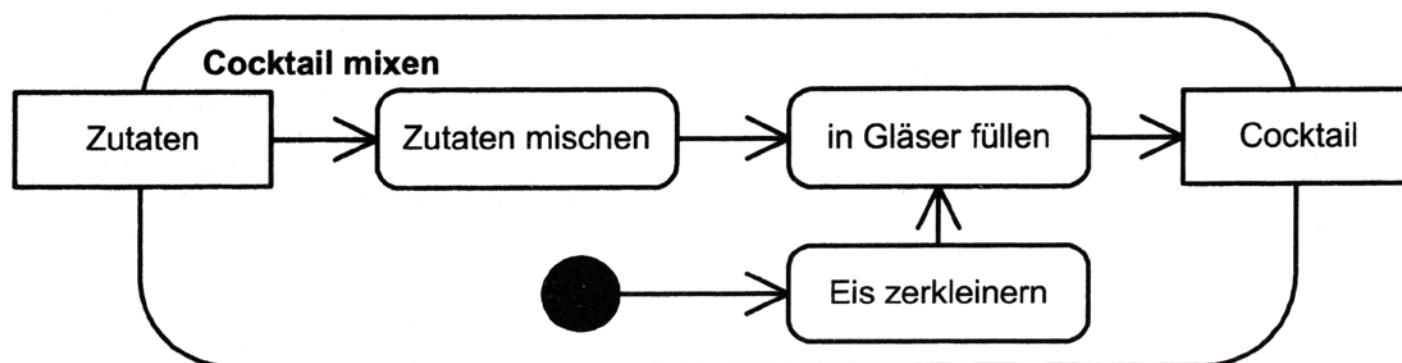


Quelle: Rupp et.al.

[Erweiterte Notation einer Aktivität]

■ Beispiel einer Aktivität

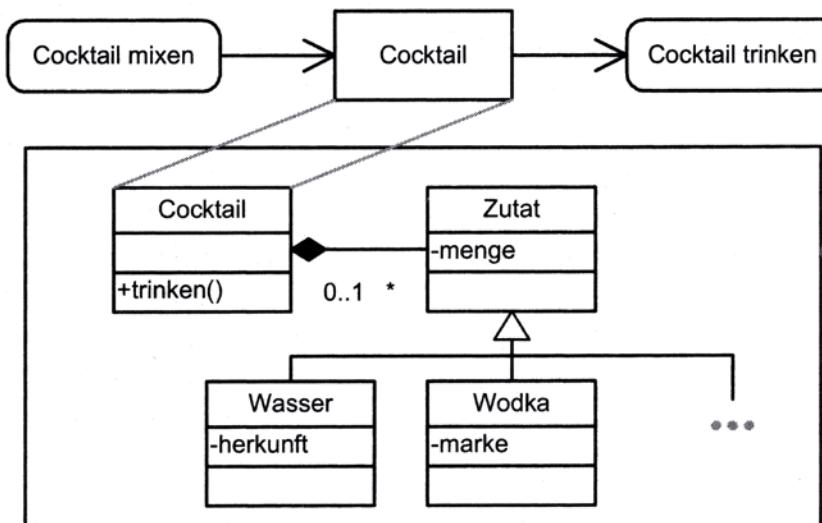
- Aktivität: Cocktail mischen
- Eingangsparameter: Zutaten
- Ausgangsparameter: Cocktail
- Unteraktionen:
 - Zutaten mischen
 - In Gläser füllen
 - Eis zerkleinern mit Startpunkt
- Die Aktivität Cocktail mischen beginnt mit dem Eingangsparameter "Zutaten" und dem Startpunkt für die Aktion Eis zerkleinern



Quelle: Rupp et.al.

Objektknoten

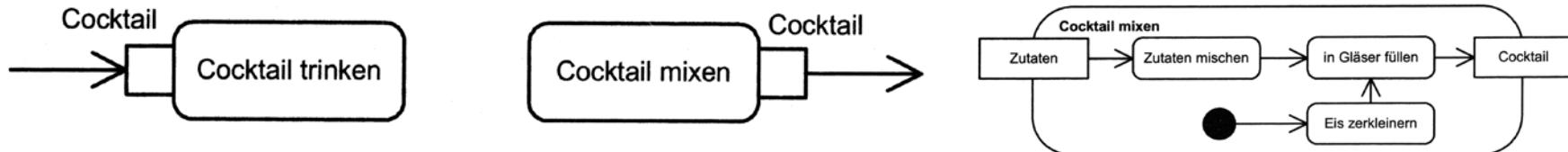
- Objektknoten wie Zutaten, Cocktail in den vorigen Abbildungen stellen Objektknoten dar
 - Objektknoten beschreiben Ausprägungen eines bestimmten Typs (häufig einer bestimmten Klasse), aber nicht im Sinne einer Klasseninstanz oder expliziten Wertes, sondern nur als Platzhalter für einen solchen
 - Modellierungsprinzipien wie Spezialisierung und Generalisierung werden von Objektknoten vollständig übernommen
 - Es ist möglich, auch Variablen, Konstanten oder Speicher jeglicher Art mit Objektknoten zu modellieren
 - Dadurch, dass Objektknoten als Platzhalter für Ausprägungen eines Typs stehen, kann der Objektknoten mit einem Klassendiagramm verknüpft werden



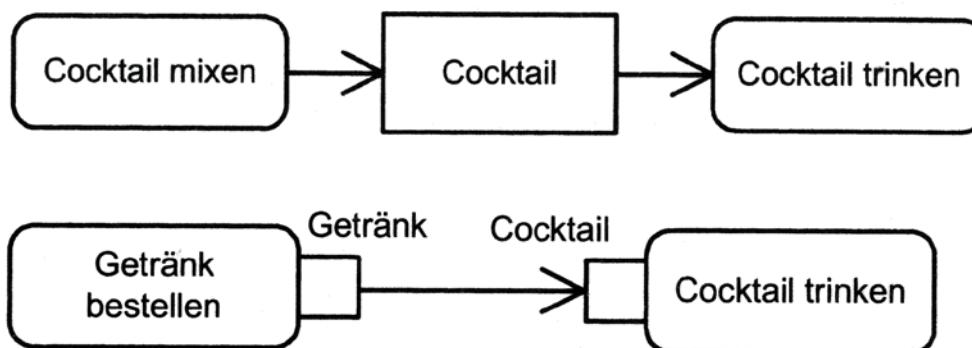
Quelle: Rupp et.al.

[Pin-Darstellung von Objektknoten]

- Zur Vereinfachung der Darstellung von Objektknoten und Aktionen können diese in Pin-Notation an die Aktion geheftet werden



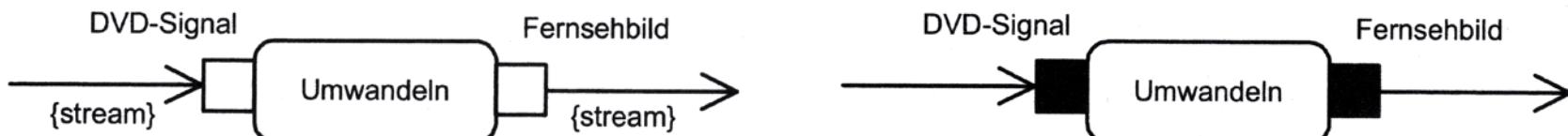
- Bei Objektknoten, die als unterschiedliche Parameter zwischen Aktionen verwendet werden, ist die Pin-Darstellung erforderlich, beim gemeinsam genutzten Objektknoten muss dieser als Ausgangs- und Eingangsparameter vom gleichen Typ sein.



Quelle: Rupp et.al.

Zusatzangaben bei Objektknoten

- Aktionen beginnen immer dann, wenn die als Eingangsparameter modellierten Objektknoten vollständig vorliegen. Soll die Aktion vorher beginnen und die Daten während der laufenden Aktion nachgelesen werden, kann das durch den Parameter {stream} deutlich gemacht werden. Alternativ ist der Pin zu schwärzen.



- Bei Zutreffen der Exception an einem Objektknoten wird eine bestimmte Aktion ausgeführt



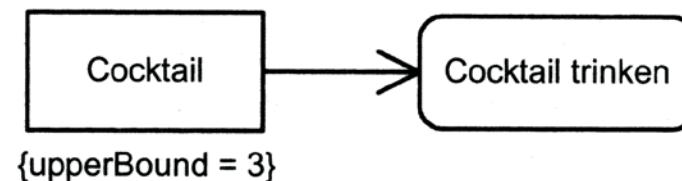
- Die Zusatzangabe <<datastore>> in einem Objektnoden besagt, dass alle Daten, die diesen Knoten passieren, aufbewahrt und beliebig oft ausgelesen werden können
 - bestehende Datenobjekte werden aktualisiert

Quelle: Rupp et.al.

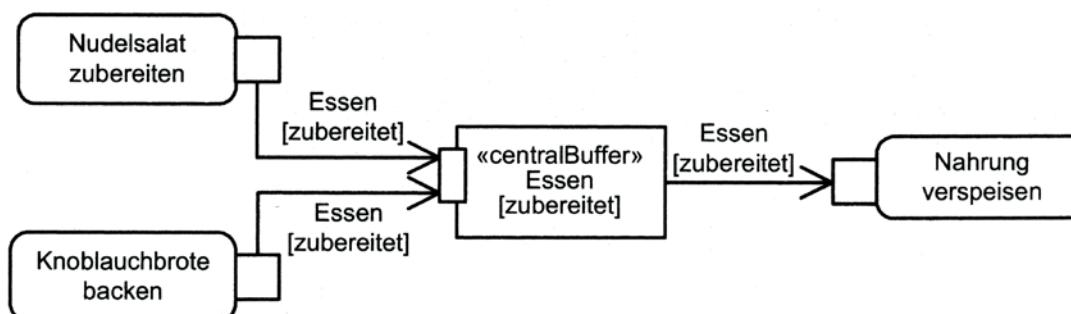
Objektknoten als Puffer

- Objektknoten können auch mit Multiplizitäten versehen werden, die aussagen wie Mengen von Objekten behandelt werden sollen

- Der Objektknoten Cocktail darf maximal 3 Cocktails beinhalten
 - Durch Angabe des Schlüsselworts {ordering=LIFO/FIFO} kann die Entnahmestrategie dargestellt werden



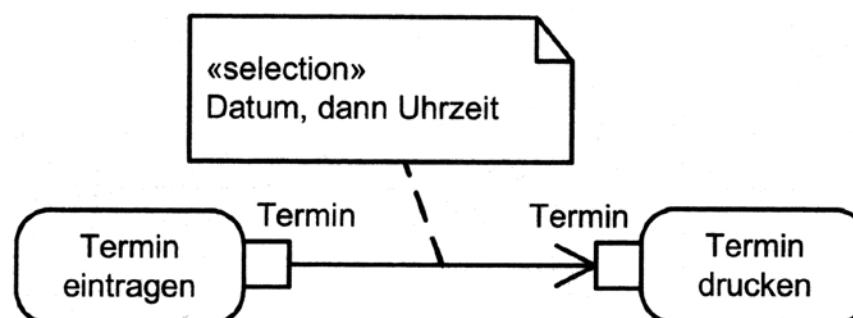
- Objektknoten können als Puffer zwischengespeichert werden und bei freier Bahn weitergegeben werden, das Schlüsselwort centralBuffer kennzeichnet einen Objektknoten, der Objekte des Typs aufnehmen und dann entsprechend weitergeben kann, Anordnung nur zwischen Objektknoten möglich (hier Pins)



Quelle: Rupp et.al.

Kanten im Aktivitäten-Diagramm

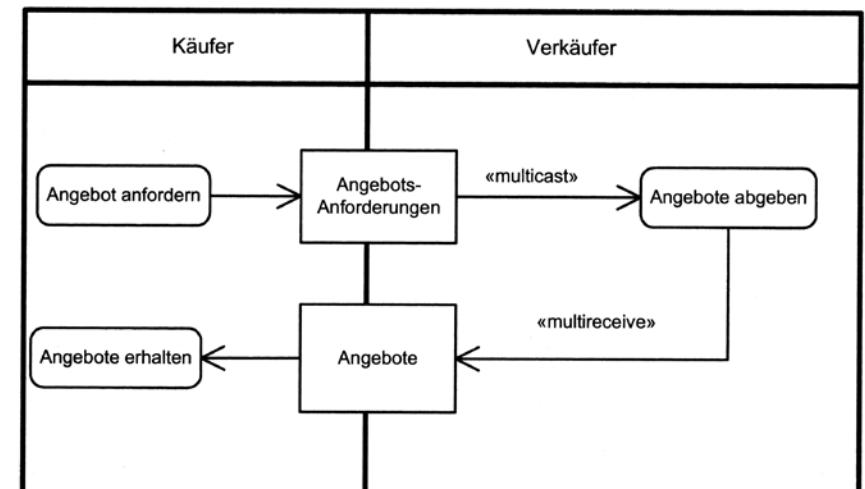
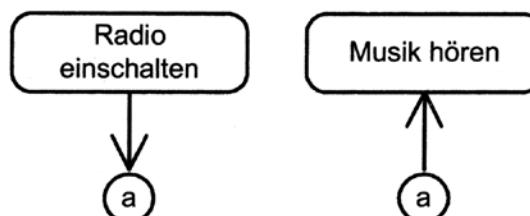
- Kanten sind die Übergänge zwischen zwei Knoten (Aktion, Objektknoten,...), zwei Typen von Kanten werden unterschieden
 - Objektfluss
 - mindestens ein Objektknoten beteiligt
 - Übertragung von Daten zum oder vom Objekt
 - Kontrollfluss
 - zwischen zwei Aktionen oder zwischen Aktion und Kontrollelement
 - es werden keine Daten transportiert
- Kanten können mit Bedingungen belegt werden, dann kann diese Kante nur bei Erfüllung der Bedingung beschritten werden
 - Wenn nur eine Kante aus einer Aktion entspringt und diese mit einer Bedingung versehen ist, könnte eine Sackgasse entstehen, Bedingung auf Kante in []
 - An Objektfluss-Kanten kann definiert werden, dass die Daten in sortierter Reihenfolge passieren müssen



Quelle: Rupp et.al.

Kanten im Aktivitäten-Diagramm

- Weitere Möglichkeiten von Objektflusskanten sind
 - Gewichtung hinsichtlich der Anzahl der Objekte, die am Anfang der Kante vorliegen müssen
 - Ursprung kann dabei nur ein Objektknoten sein
 - Die Gewichtung wird in Form von {weight = *} angegeben
 - Bei keiner Gewichtungsangabe wird von 1 ausgegangen
 - Gewichtete Kanten, die centralBuffer (Datenspeicher) bestücken, muss das Gewicht kleiner als der Pufferspeicher sein
 - Durch Schlüsselwörter wie <<multicast>> und <<multireceive>> kann eine Multiplizität zwischen Objektknoten und Aktionen abgebildet werden
 - Kanten können an einer Sprungmarke enden, die durch einen Kreis mit enthaltenem Buchstaben/Zahl für den Namen der Sprungmarke angegeben ist
 - Übersichtlichkeit
 - Synchronisation/Platz

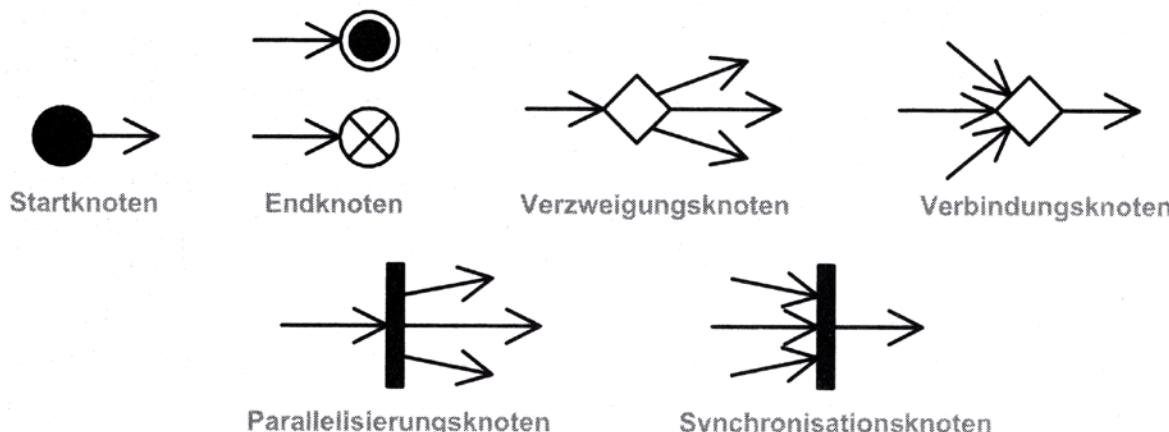


Quelle: Rupp et.al.

Kontrollelemente in Aktivitäts-Diagrammen

■ Kontrollelemente steuern den Ablauf der Aktivität

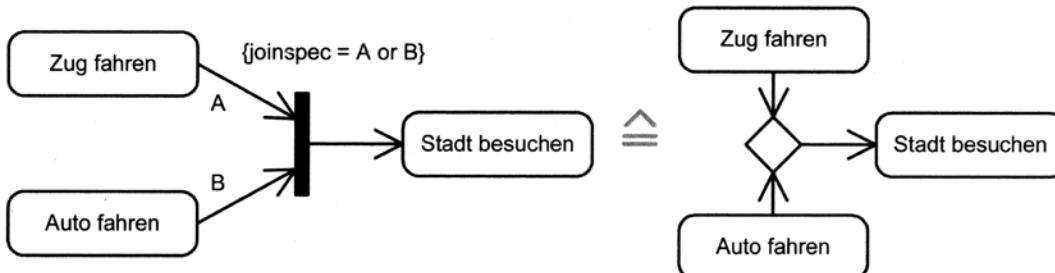
- Verzweigungsknoten verzweigen anhand einer Bedingung in unterschiedliche Aktionen
- Verbindungsknoten führen verschiedene Aktionen auf eine gemeinsame Aktion weiter, ohne dass eine Synchronisation stattfindet
- An Parallelisierungsknoten wird eine Aktion in mehreren Aktionen parallel weitergeführt
- An Synchronisationsknoten werden mehrere parallele Aktionen zusammengeführt, wobei alle am Synchronisationsknoten einlaufenden Aktionen abgeschlossen sein müssen, bevor die nächste Aktion beginnen kann



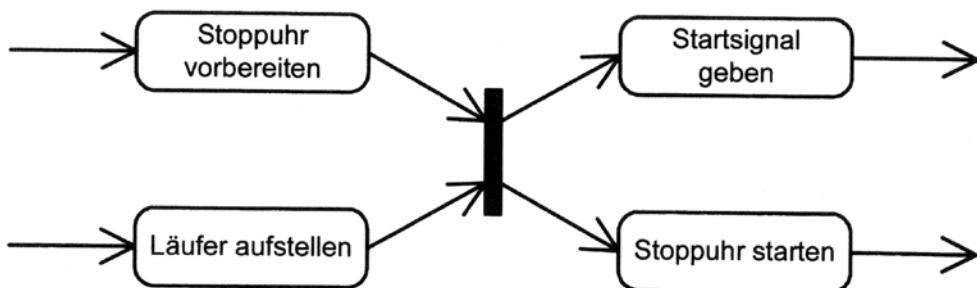
Quelle: Rupp et.al.

spezielle Knoten-Eigenschaften

- Durch Angabe der Verknüpfungsbedingung kann ein Verknüpfungsknoten eine Verbindung darstellen (hier Verknüpfungsbedingung joinspec= A or B)

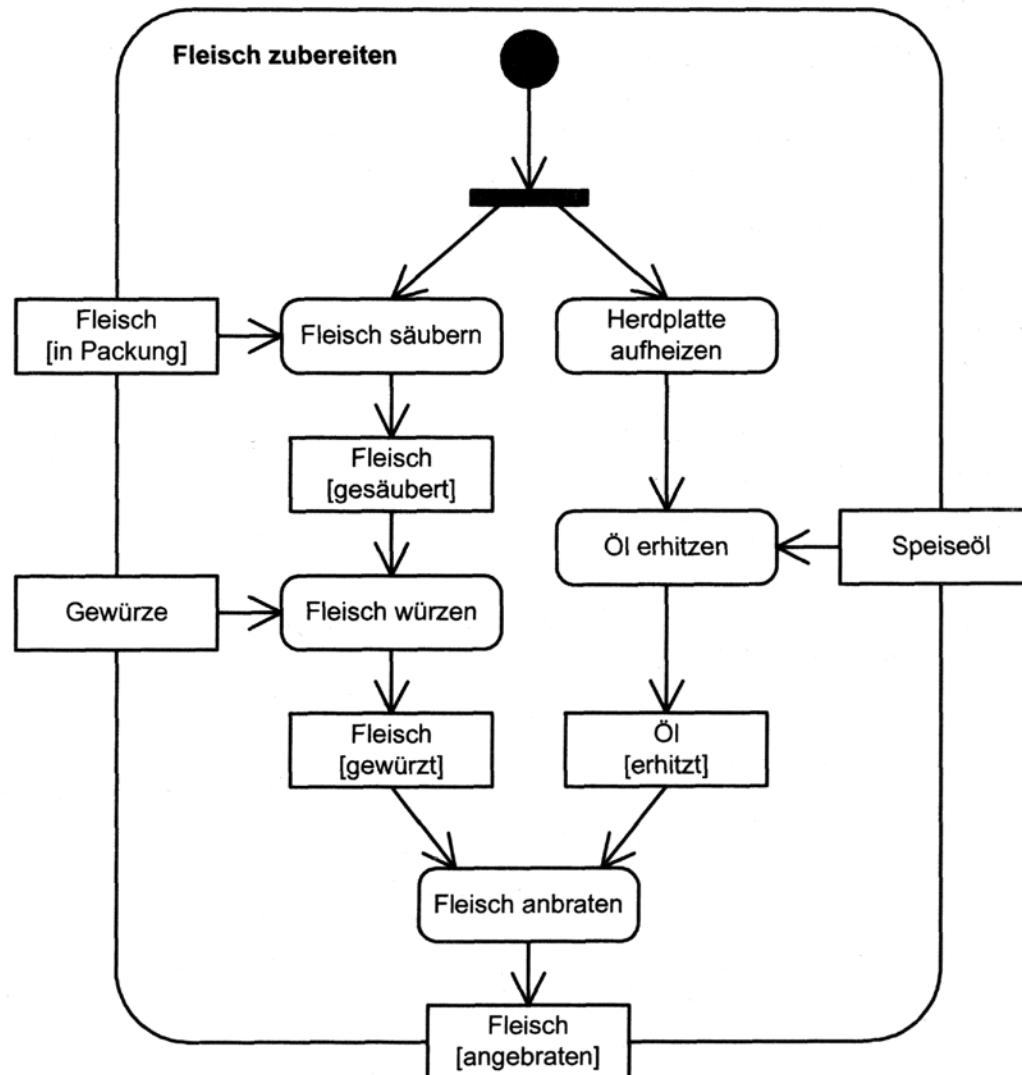


- Ein Parallelisierungsknoten kann auf der Eingangsseite gleichzeitig als Verknüpfungsknoten (mit Synchronisation) wirken



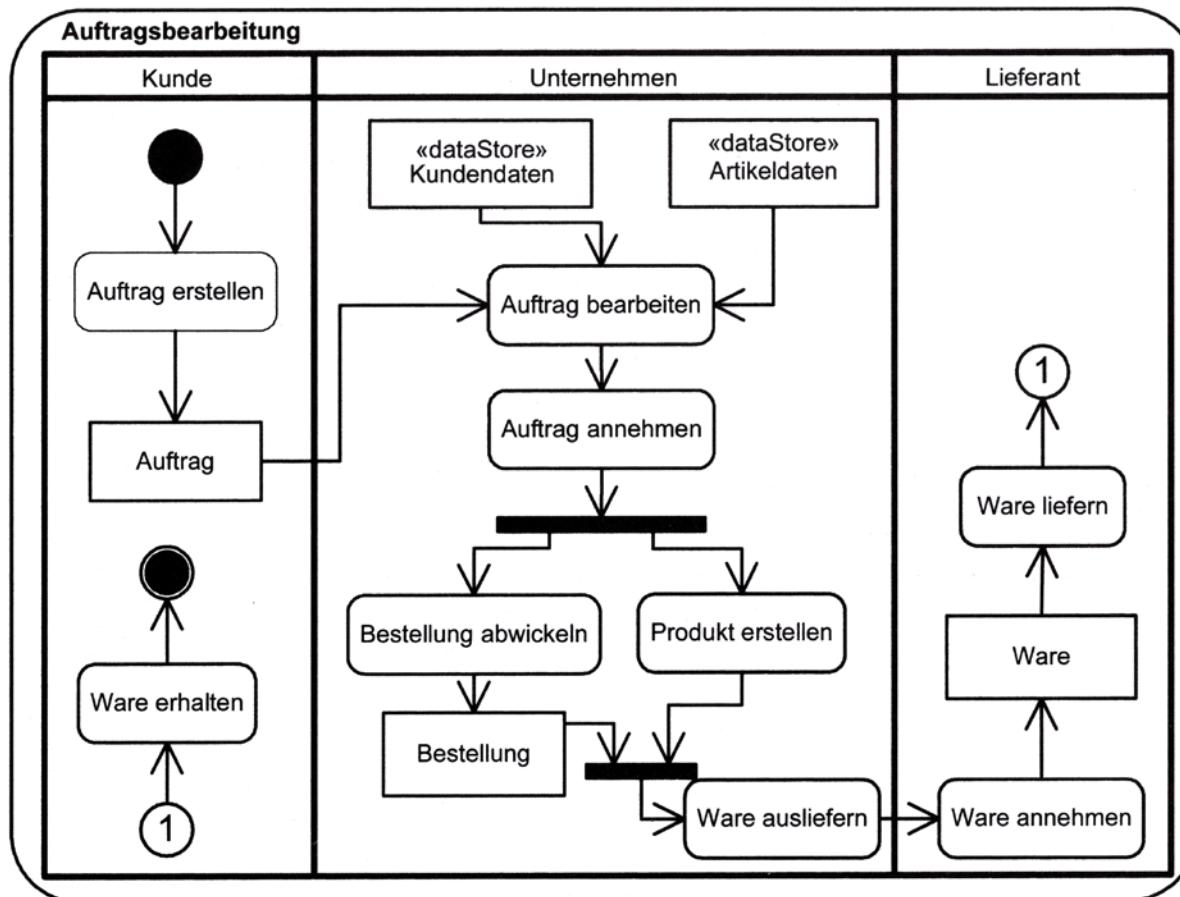
Quelle: Rupp et.al.

Beispiel eines Aktivitäten-Diagramms



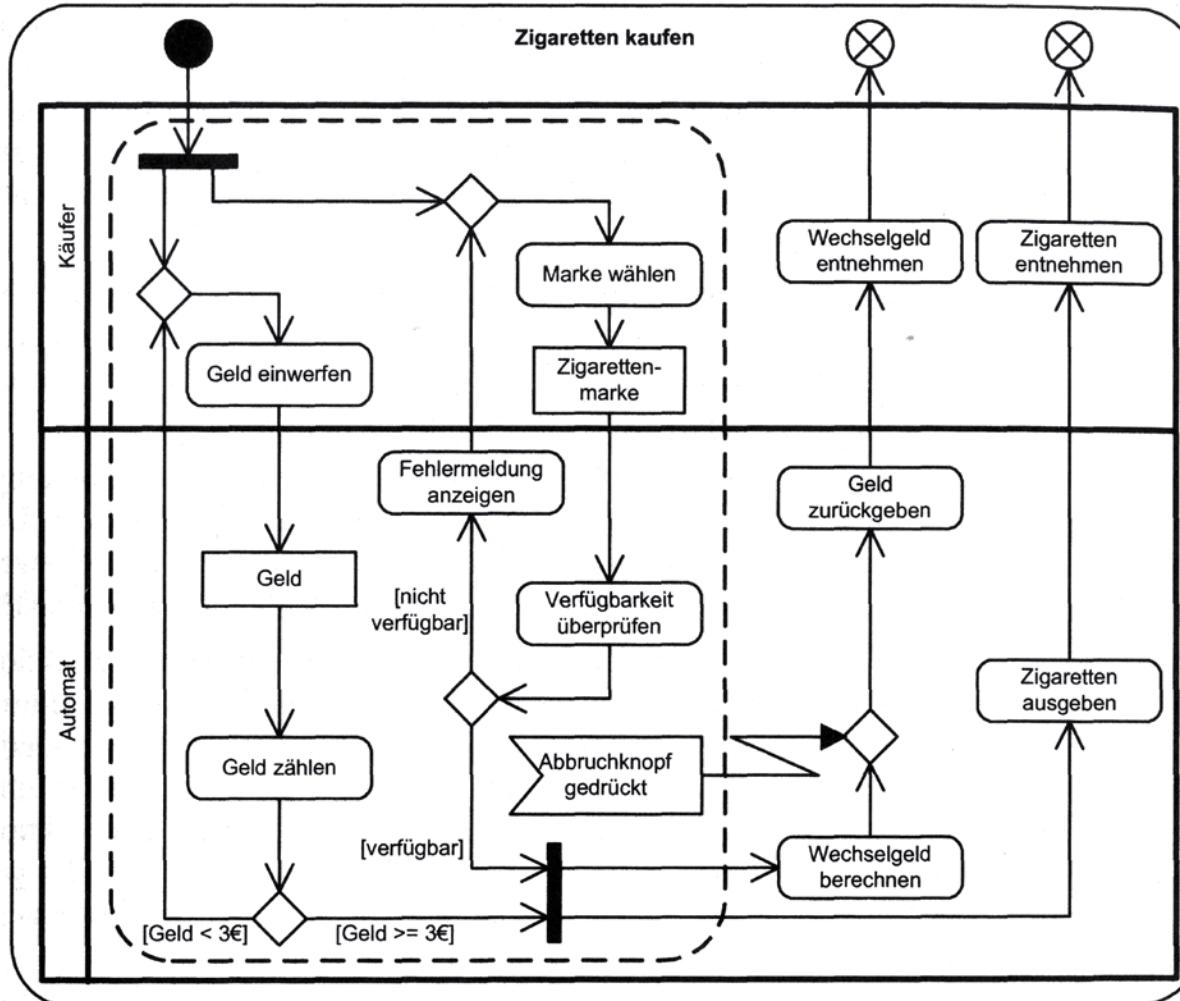
Quelle: Rupp et.al.

Beispiel eines Aktivitäten-Diagramms



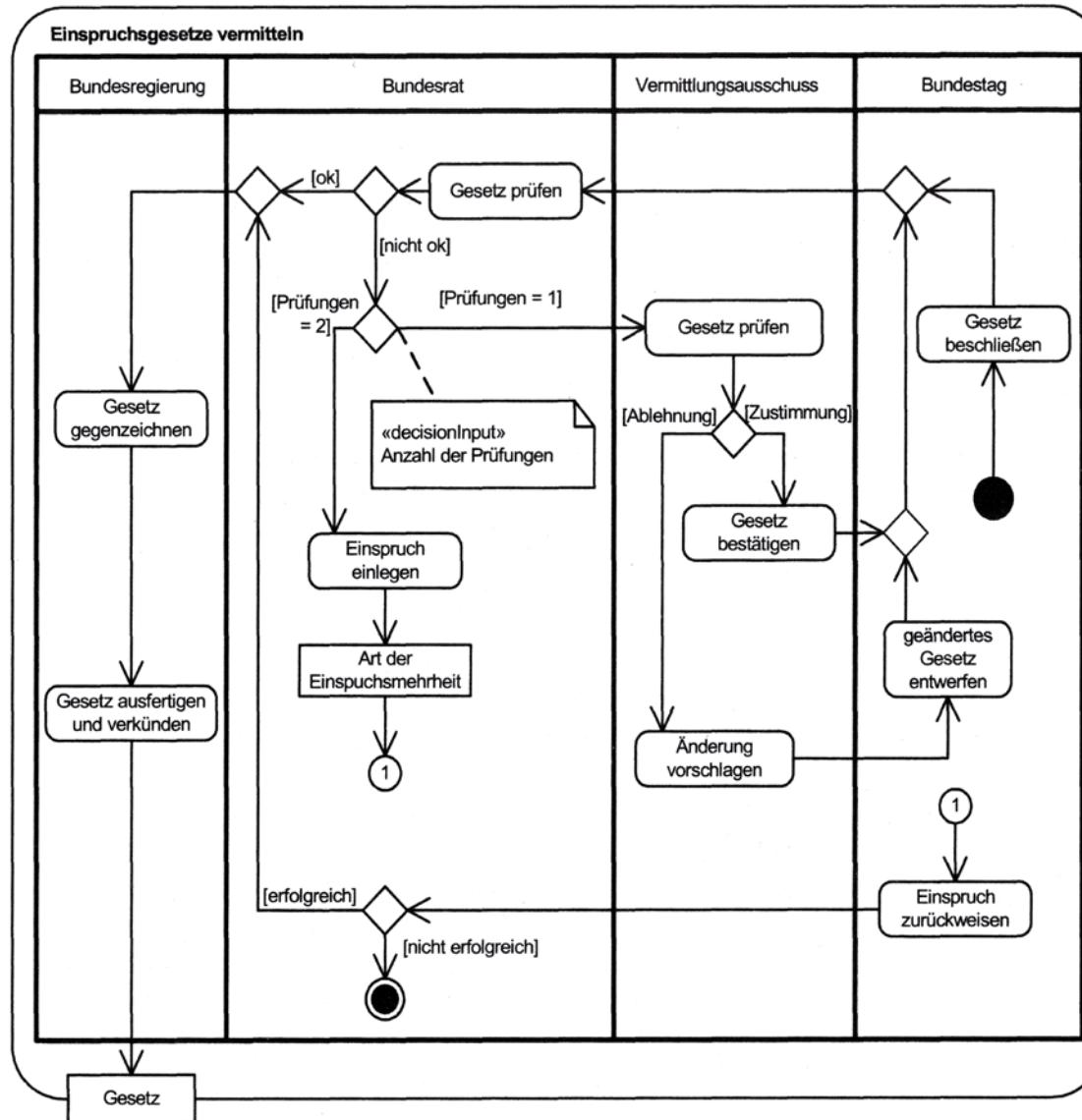
Quelle: Rupp et.al.

Definition von Unterbrechungs- und Aktivitätsbereichen



Quelle: Rupp et.al.

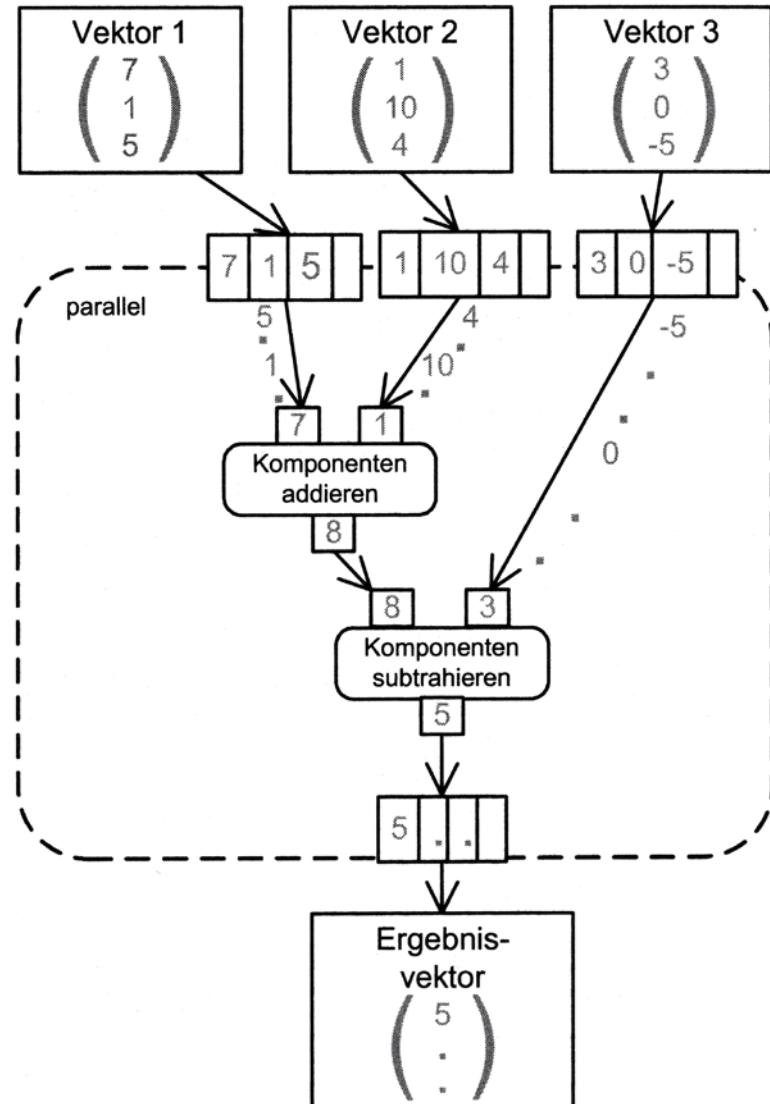
Aktivitätsbereich



Quelle: Rupp et.al.

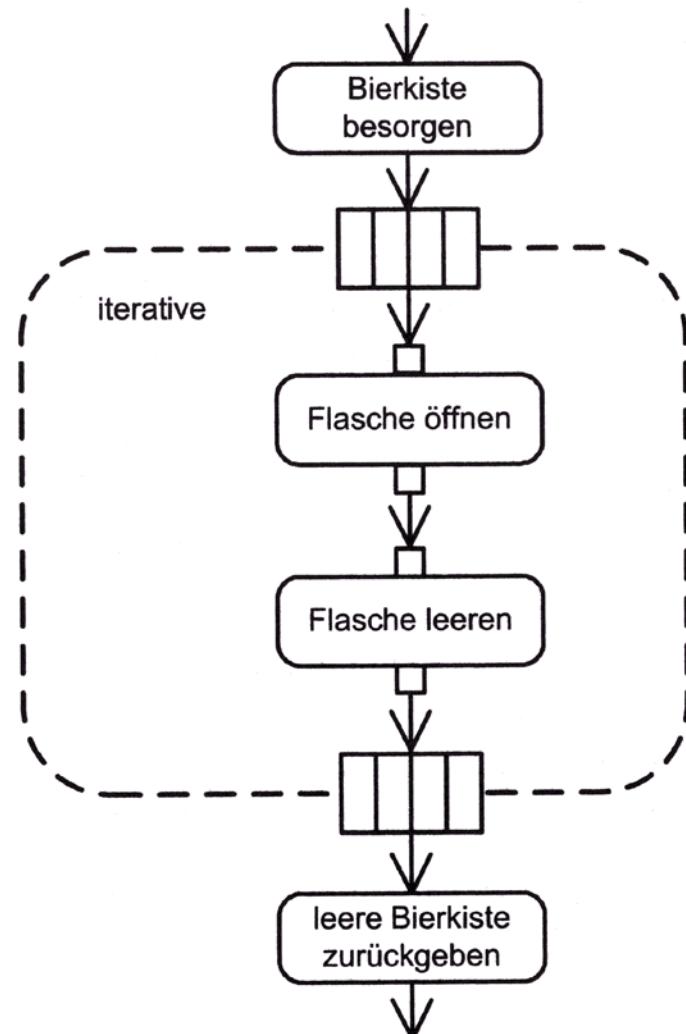
Mengenverarbeitungsbereich

- Mögliche Schlüsselwörter
 - parallel
parallele Bearbeitung der Eingabewerte
 - iterative
Abarbeitung der Eingabewerte nacheinander
 - streaming
Entnahme aller Werte auf einmal
- Typen innerhalb eines Knotens müssen gleich sein
- Ausgangstypen müssen zumindest einem Ausgangsknoten entsprechen



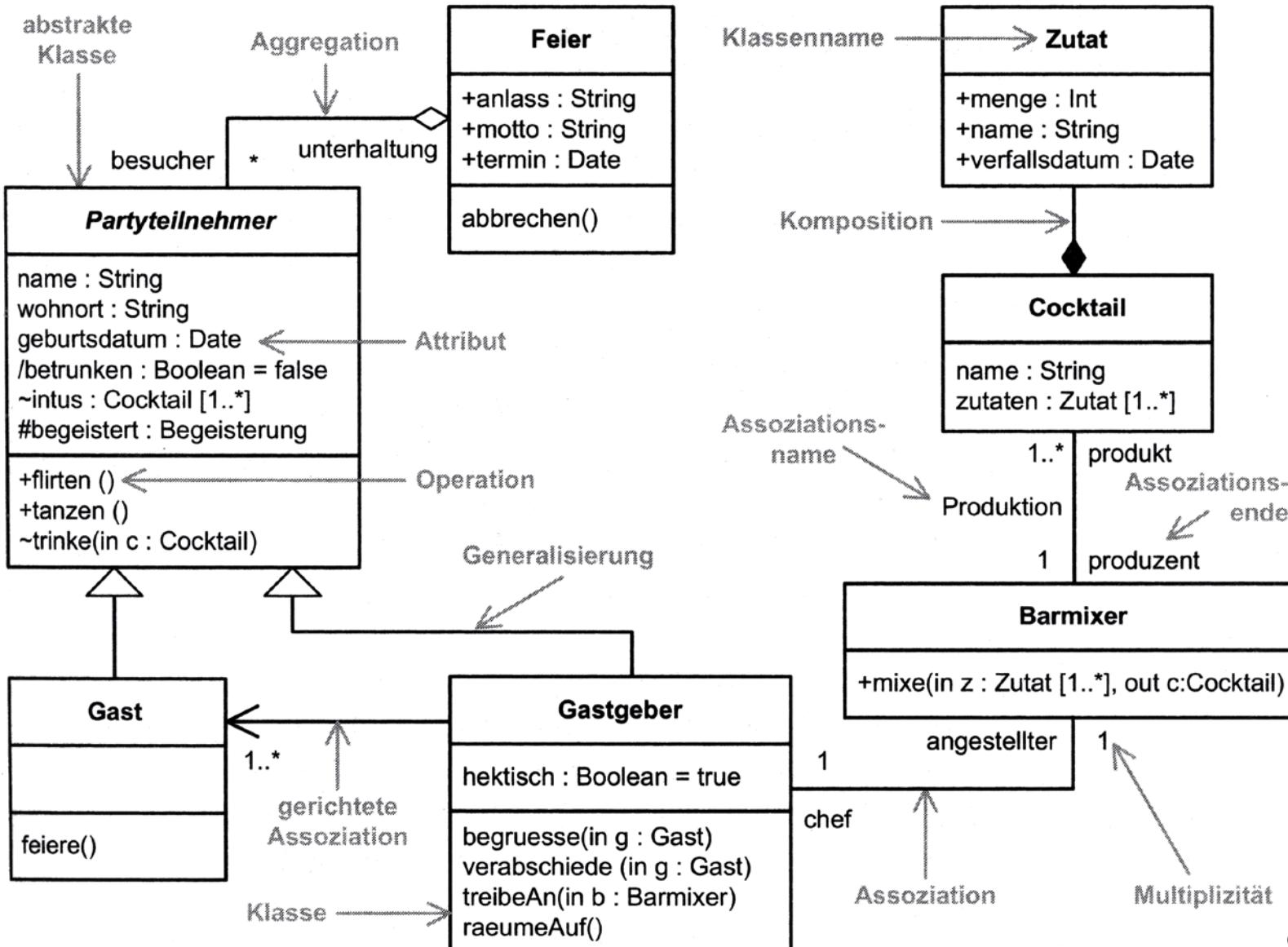
Quelle: Rupp et.al.

[iterative Mengenverarbeitung]



Quelle: Rupp et.al.

Klassendiagramm



Quelle: Rupp et.al.

[Notation einer Klasse im UML-Klassendiagramm]

- Attribute (Merkmale) einer Klasse, auch Membervariablen
 - Name des Attributs
 - Datentyp des Attributs (nicht zwingend erforderlich)
 - Sichtbarkeit des Attributs
 - + public (allgemein verwendbar)
 - - private (nur innerhalb der Klasse zu verwenden)
 - # protected (nur von spezialisierten Klassen verwendbar)
 - ~ Verweis auf andere Elemente notwendig
 - Direktes oder indirektes Attribut
 - / zeigt an, daß das Attribut aus vorliegenden Größen berechnet werden muß
 - Multiplizität
 - 0..1 optionales Attribut, entweder kein oder genau 1 Wert vorhanden
 - 1..1 zwingendes Attribut, genau 1 Wert vorhanden
 - 0..* optionales Attribut in beliebiger Anzahl
 - 1..* zwingendes Attribut mit mindestens 1 Wert
 - n..m mindestens n, aber höchstens m Werte, m=n ist erlaubter Sonderfall
 - Vorgabewert
 - Eigenschaftswert
 - readOnly keine Änderung am Attributwert erlaubt
 - ordered Werte liegen in geordneter Reihenfolge vor
 - unique Wertemengen dürfen keine Dubletten enthalten

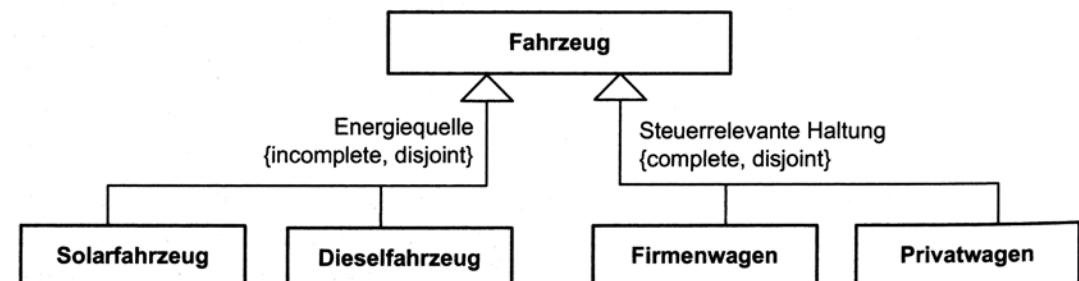
[Operationen einer Klasse in UML-Notation]

■ Operationen (Methoden) einer Klasse

- Name
- Sichtbarkeit in analoger Verwendung wie bei Attributen
- Parameterliste
 - Übergaberichtung in, out, inout als mögliche Werte
 - Name Name des Parameters
 - Typ Datentyp des Parameters
 - Multiplizität mengenmäßige Festlegung des Parameters
 - Vorgabewert Standardwert eines Parameters
 - Eigenschaftswert wie bei Attributen (readOnly, unique, ordered)
- Rückgabe
 - Datentyp Datentyp des Rückgabewerts
 - Eigenschaftswert
 - query für Rückgabewert geltend, analog Attributen, zusätzlich:
 - redefines Operation ändert keine Daten (für readOnly)
 - Operation ist anders als eine geerbte Operation gleichen Namens#
- Operationen, die nicht für ein Objekt gelten, sondern allgemein der Klasse zugeordnet werden müssen (z.B. Objekterzeugung) werden unterstrichen

Generalisierung in der UML

- Klassen können über die UML generalisiert werden
- Eine Generalisierung stellt eine Verallgemeinerung der darunter liegenden Klasse dar (Haus – Bungalow)
- Eine Generalisierung kann von mehreren Klassen gemeinsam genutzt werden (Haus mit Bungalow und Einfamilienhaus,...)
- Die Übertragung von Eigenschaften eines generalisierten Typs auf die darunter liegende Klasse wird als Vererbung bezeichnet (betrifft Attribute und Operationen)
- Die Generalisierungs-Beziehung wird für die eingeordneten Subtypen häufig als "is-A" - Beziehung bezeichnet
- Die Generalisierung kann unterschiedlichen Aspekten erfolgen
 - complete alle momentan denkbaren Spezialisierungen wurden berücksichtigt
 - incomplete nicht alle Spezialisierungen erfaßt
 - disjoint Eine Instanz gehört nur zu einem Subtyp
 - overlapping eine Instanz kann mehreren Subtypen angehören



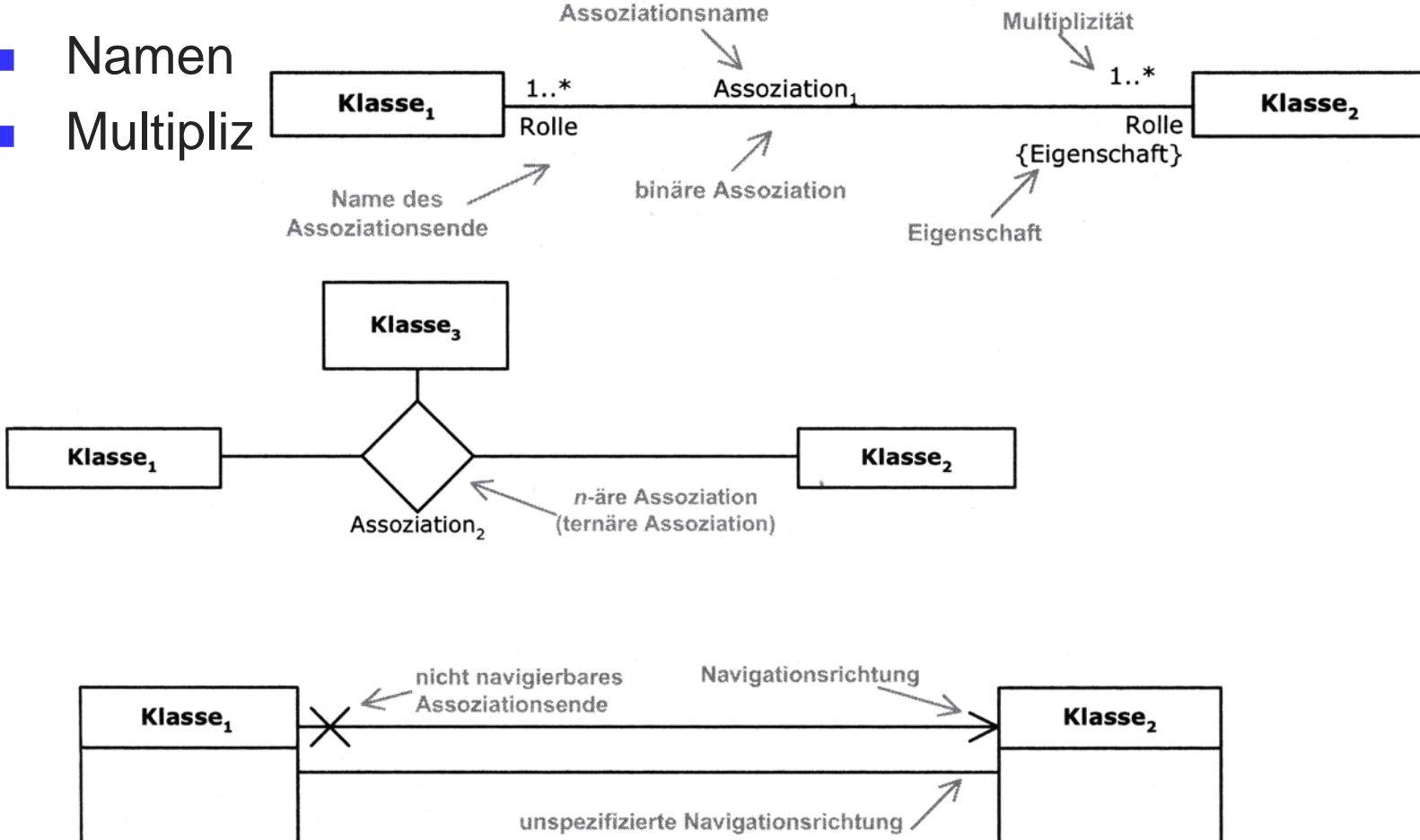
Quelle: Rupp et.al.

Assoziationen in der UML

- Stellen Beziehungen zwischen Klassen dar

- Eine Beziehung wird in Form einer Rolle definiert

- Namen
 - Multipliz

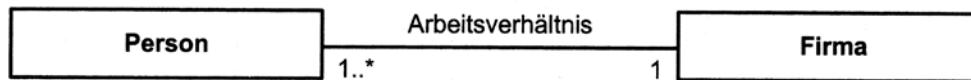


- Besitzt eine Assoziation eine Richtung, so wird dies durch Pfeile gekennzeichnet

Quelle: Rupp et.al.

Beispiele von Assoziationen

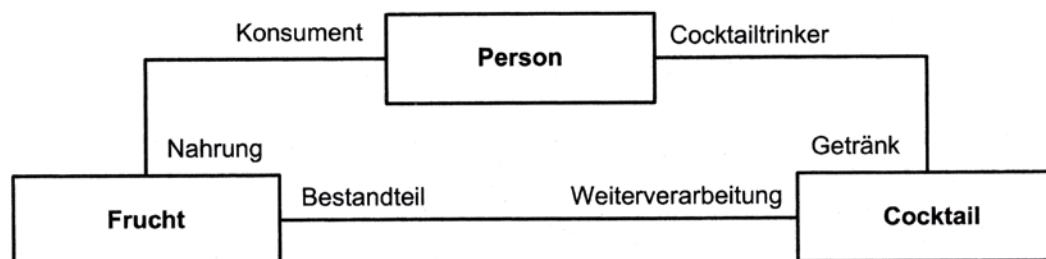
- Person steht in Arbeitsverhältnis



- gerichtete Assoziation



- Beziehungen und Rollen



- Welche Multiplizität könnte für jede Rolle angegeben werden?

Quelle: Rupp et.al.

Assoziationen und Einschränkungen

- Assoziationen können neben der Multplizität weiter eingeschränkt werden

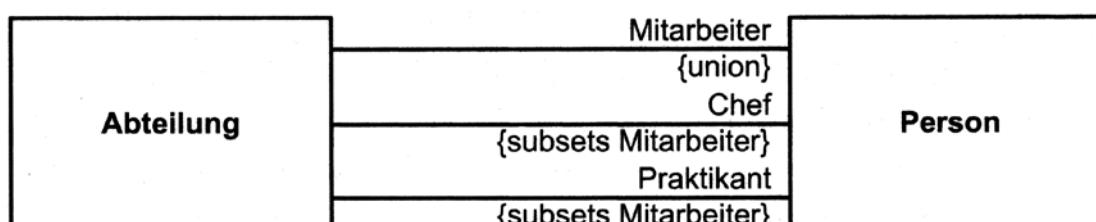
- ordered Reihenfolge der Beziehung Partygast relevant (zeitlich immer nur eine möglich)



- bag Ein Objekt kann mehrmals an derselben Assoziation teilnehmen
- sequence ordered und bag zusammengefasst
- redefines Eine Rolle auf generalisierter Ebene wird neu definiert (ersetzt)



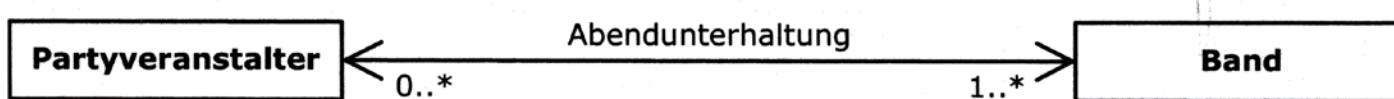
- subsets die an der Assoziation teilnehmenden Objekte entstammen einer anderen Objektmenge, die auch durch Assoziation gebildet wird
- union vereint die subsets



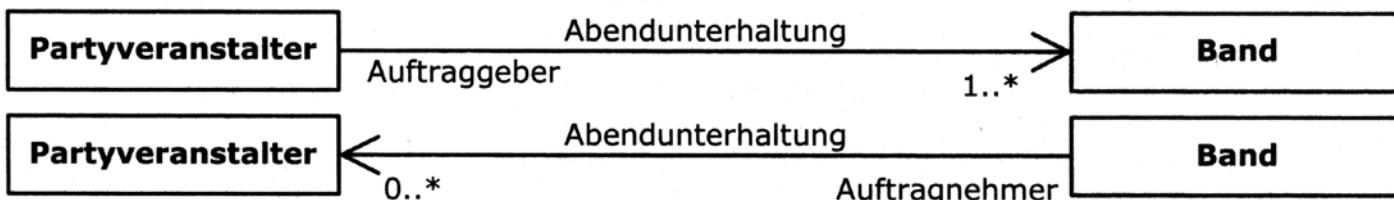
Quelle: Rupp et.al.

Beispiele navigierbarer Beziehungen

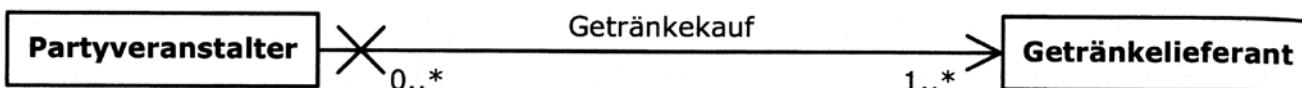
- doppelseitig navigierbare Assoziation, die Instanzen der assoziierten Klassen stehen in einer bidirektionalen Beziehung, jede Instanz kennt die andere



- Die bidirektionale Assoziation kann in zwei getrennte Assoziationen aufgetrennt werden, um die Beziehung eindeutiger zu beschreiben



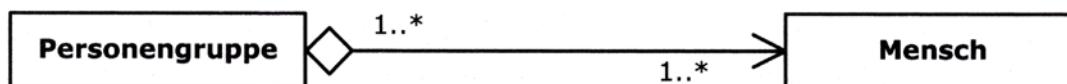
- Ist eine Instanz nicht über die Beziehung erreichbar (oder darf sie nicht erreichbar sein), so ist an dieser Stelle die Assoziation zu negieren (z.B. auch Kunde im Supermarkt)



Quelle: Rupp et.al.

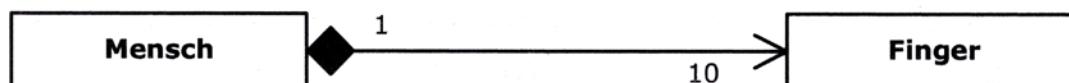
■ Aggregation

- Die Aggregation beschreibt einen Zusammenhang zwischen einem Ganzen und seinen Bestandteilen
- Die in Beziehung stehenden Instanzen der verbundenen Klasse sind Teil eines Ganzen
- Als Beziehungsnamen kann man "besteht aus" bzw. aus der anderen Richtung "ist Teil von" oder "part of" verwenden



■ Komposition

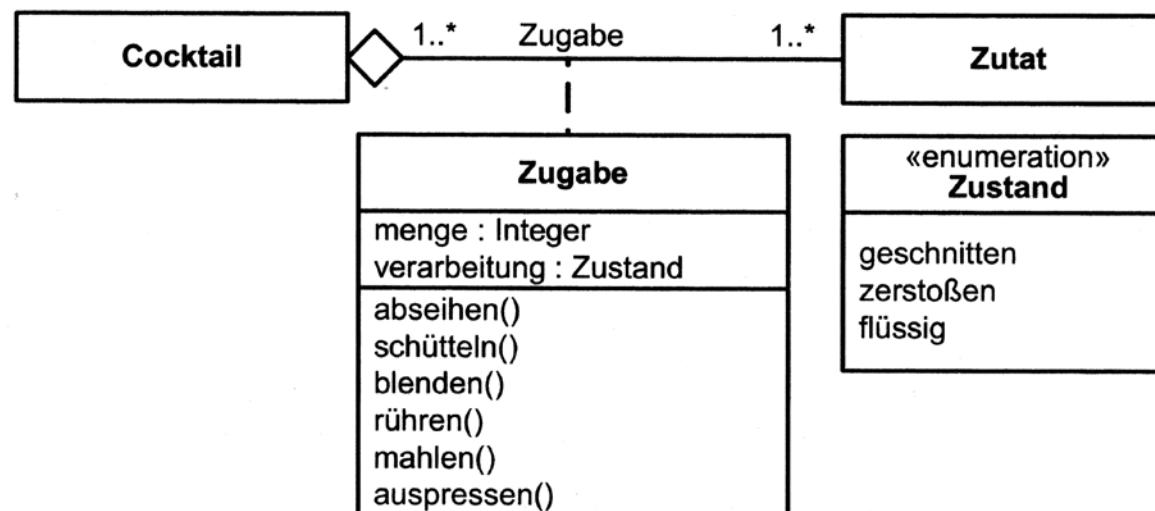
- verschärkte Form der Aggregation, bei der das Ganze ohne das zugeordnete Teil nicht bestehen kann
- wird die Instanz der Komposition entfernt, so sterben damit auch alle Teile, die in der Komposition verbunden sind, dies ist bei der Aggregation nicht der Fall
- Ein Teil einer Komposition kann aus Gründen der Eindeutigkeit nur einer Komposition vorkommen, mehrfache Verwendung ist aufgrund der dann unklaren Lebenslinie nicht möglich



Quelle: Rupp et.al.

Assoziationsklasse

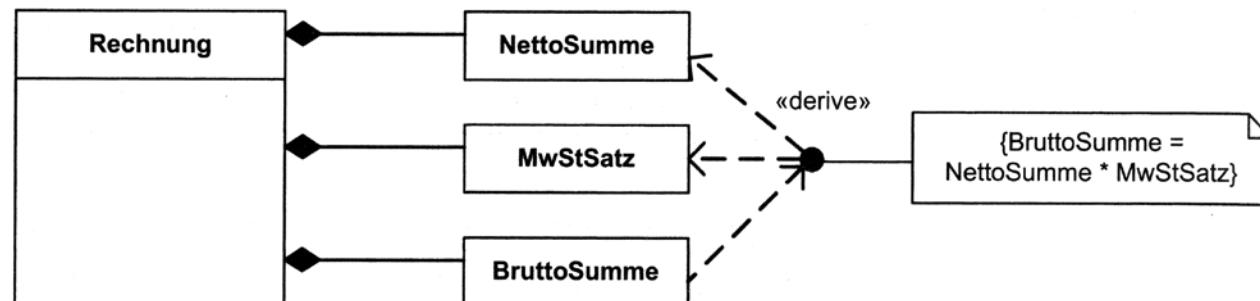
- Eine Assoziationsklasse ist eine eigene Klasse, die zweckgebunden an eine Assoziation gleichen Namens ist.
- Stehen zwei Klassen in einer Assoziation, so kann diese Assoziation mit einer entsprechenden Klasse verbunden und darüber exakter als über den Beziehungskontext allein beschrieben werden
 - Die Assoziationsklasse stellt Eigenschaften/Operationen der Beziehung dar
 - Operationen in der Assoziationsklasse lassen die Beschreibung von Operationen zu, die zur Beziehung gehören und nicht eindeutig einer Klasse zugeordnet werden können
 - Bei Aggregation wäre hier denkbar, dass die Beziehung auch über das Zerstören beteiligter Instanzen entscheidet



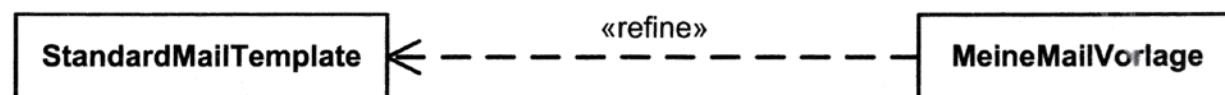
Quelle: Rupp et.al.

Weitere Beziehungen

- Abstraktionsbeziehungen zur Veranschaulichung der semantischen Zusammenhänge (informeller Charakter), aber auch als formale Beziehung, also über Ausdruck definierbar
 - **derive** eine Klasse leitet sich aus einer anderen ab, dabei werden Daten evtl. redundant gehalten, da Zustände gemerkt werden müssen, die nur über eine Beziehung verloren gingen, (Bruttosumme leitet sich ab)



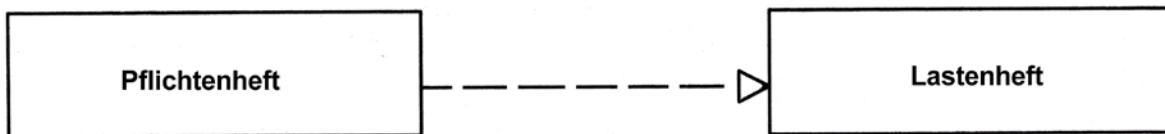
- **refine** Eine Klasse stellt eine Verfeinerung einer anderen Klasse dar



Quelle: Rupp et.al.

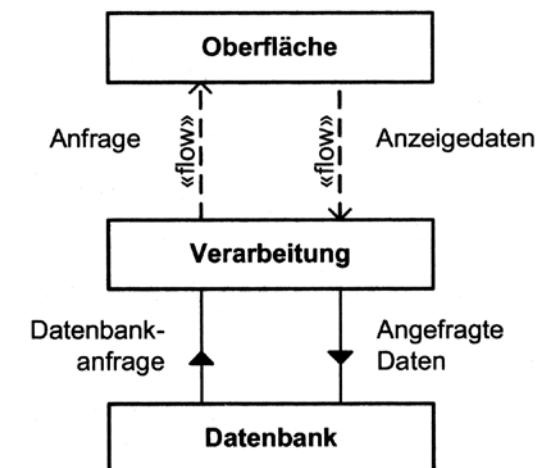
Weitere Beziehungen

- Realisierungsbeziehung
 - als Form der Abstraktionsbeziehung, die ausdrückt, dass die Klasse Pflichtenheft die Klasse Lastenheft realisiert, also die Umsetzung der Klasse darstellt (Pfeilrichtung beachten – lies "implementiert")



- Substitutionsbeziehung
 - als weitere Form der Abstraktion mit dem Schlüsselwort "substitute" dargestellt. Beziehung zwischen zwei Klassen, bei denen eine Instanz der einen Klasse durch eine Instanz der anderen Klasse ersetzt werden kann, dabei müssen alle Verpflichtungen der Ursprungsklasse übernommen werden können (Schnittstellengleichheit)
Beispiel: Eine reelle Zahl kann eine ganze Zahl ersetzen

- Informationsfluss-Beziehung
 - Darstellung als gestrichelter Pfeil von Quelle zum Ziel des Informationsflusses und dem Schlüsselwort "flow"
 - Darstellung der Informationseinheit an "flow"-Schlüsselwort oder über geschwärzten Pfeil



Quelle: Rupp et.al.

Aufgabe

- Benennen Sie Vor- und Nachteile der UML-Notation mit Hilfe der Klassendiagramme
 - Modellbildung, Darstellbarkeit und ihre Grenzen
 - Eindeutigkeit, Einfachheit

Aufgabe

- Es ist eine Konsolenanwendung zu programmieren, bei der eine "Suchen und Ersetzen" Aktion in zu spezifizierenden Dateien durchgeführt werden soll - die Optionen des Programms sollen in der Befehlszeile angegeben werden.
 - Erstellen Sie eine Liste möglicher Optionen und ihrer Funktion
 - Für eine objektorientierte Umsetzung des Programms ist ein Klassendiagramm zu erstellen