

Modellierung Grundlagen

Interaktionsmodellierung und Use Cases

Vers. 1.0

[Martin Zimmermann]



Inhalt

- Use Cases:
 - Motivation und Zielsetzung
 - Einführendes Beispiel
 - Akteure
 - Elemente von Use-Case-Diagrammen
 - include und extend Beziehungen
 - Typische Modellierungsfehler
 - Use Case Beschreibungen
- Aktivitätsdiagramme:
 - Motivation und Zielsetzung
 - Elemente von Aktivitätsdiagrammen
 - Beispiele
- Zusammenfassung

Use Cases: Motivation und Zielsetzung

Ziele:

- Darstellung der **funktionalen Dienstleistungen** eines Systems
- Schaffen einer **Kommunikationsgrundlage zwischen Auftraggeber und Auftragnehmer**
- Verständlich machen von komplexen Systemen und Darstellen auf hohem Abstraktionsniveau

Use Cases (= Anwendungsfälle):

Sind Ausgangspunkt vieler objektorientierter Entwicklungsmethoden.

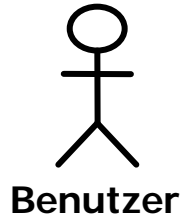
Use Cases: Motivation und Zielsetzung

- Use Cases repräsentieren die Anforderungen der Kunden
- Ein **Use Case** ist eine **Sequenz von Transaktionen** innerhalb eines Systems, deren Aufgabe es ist, einen für den einzelnen **Akteur** (Anwender) **identifizierbaren Nutzen** zu erzeugen. [Ivar Jacobson]
- **Akteure** interagieren mit dem System im Kontext der Use Cases
- Akteur:
 - Rolle, die jemand oder etwas einnimmt und die in Beziehung zum Geschäftsbereich steht, oder
 - `Alles`, das mit dem System interagiert

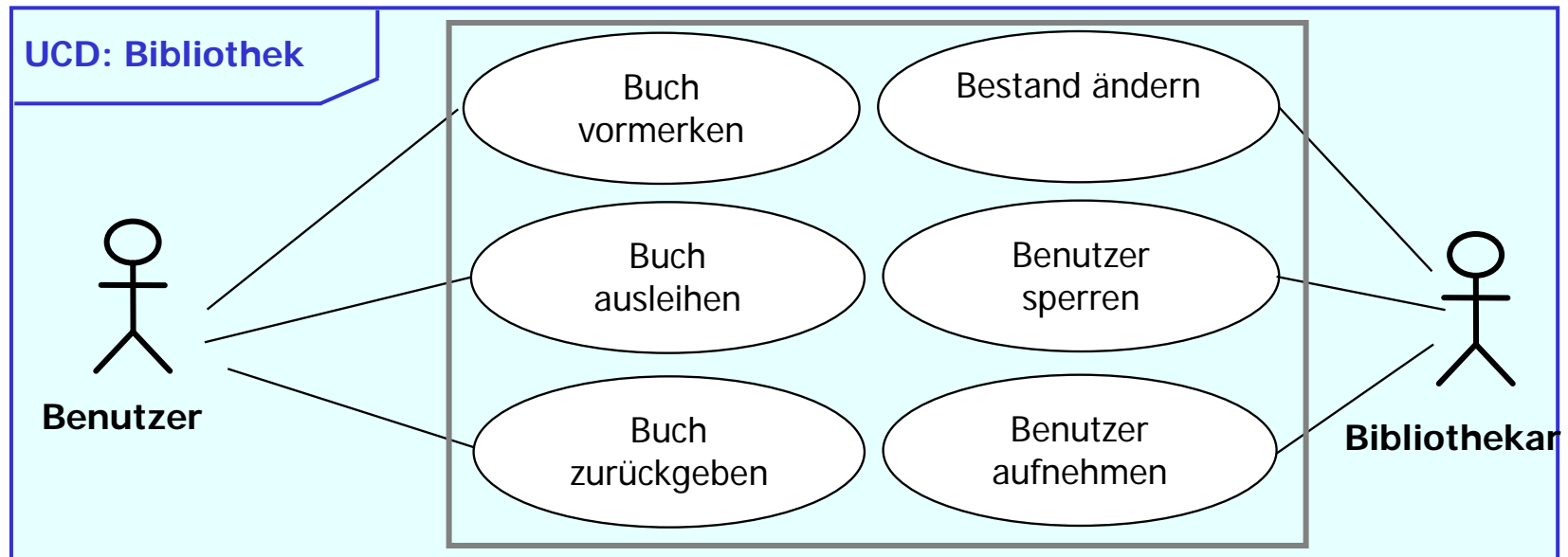
Einführendes Beispiel

■ Beispiel: Bibliothekssystem (mit Ausleihterminals)

Akteure:



Use Case Diagramm:

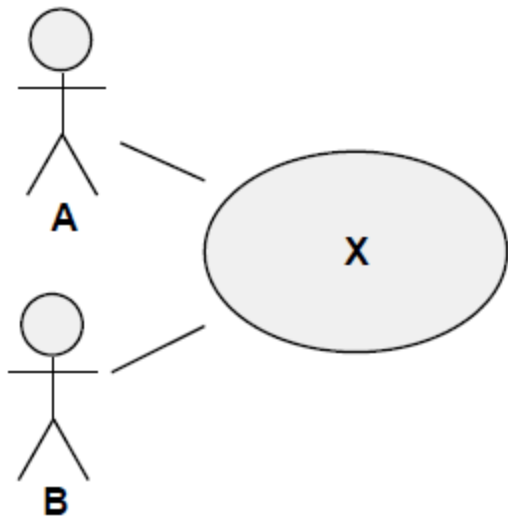


Akteure in Use Cases

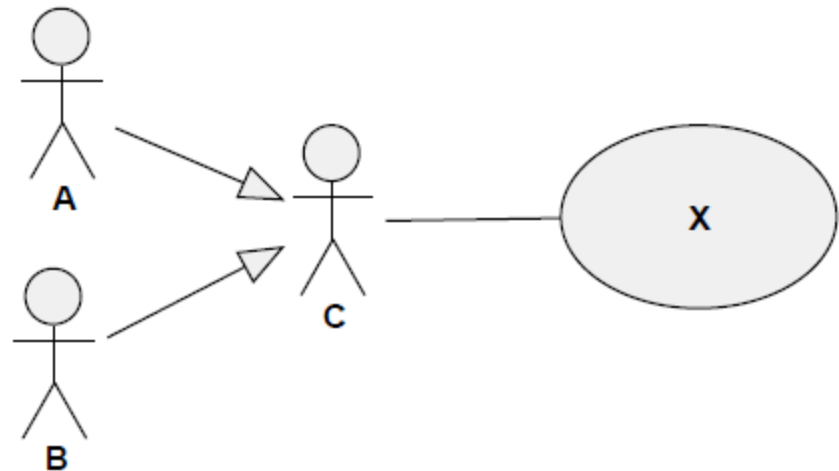
- Akteure **interagieren** mit dem System...
 - indem sie das **System benutzen**,
d.h. die Ausführung von Anwendungsfällen initiieren
 - indem sie **vom System benutzt werden**,
d.h. Funktionalität zur Realisierung von Anwendungsfällen zur Verfügung stellen
- Akteur wird **durch Assoziationen mit Use Cases verbunden**, d.h. er »kommuniziert« mit dem System
- Jeder Akteur muss mit **mindestens einem Anwendungsfall** kommunizieren
- Die Assoziation ist binär und kann **Multiplizitäten** aufweisen
- Notationsvarianten:

Akteure in Use Cases

- Unterscheidung, ob mehrere Akteure gemeinsam mit einem Anwendungsfall kommunizieren können oder müssen.


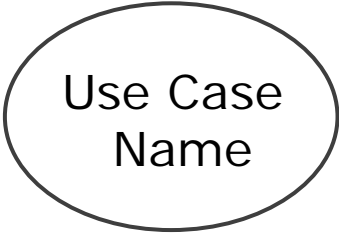



A und B kommunizieren
mit X





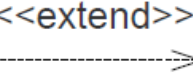
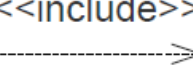
A oder B kommuniziert
mit X

Use Case Diagramm: Basiselemente

Name	Syntax	Beschreibung
Systemgrenze		Grenze zw. dem eigentlichen System und den Benutzern des Systems
Use Case		vom System erwartetes Verhalten
Akteur		Rolle der Systembenutzer

<https://youtu.be/wtBERi7Lf3c>

Use Case Diagramm: Basiselemente

Name	Syntax	Beschreibung
Assoziation		Beziehung zwischen Use Cases und Akteuren
Generalisierung		Vererbungsbeziehung von Use Cases und Akteuren
extend		<i>A extends B</i> : opt. Verwenden von Use Case A durch Use Case B
include		<i>A includes B</i> : notw. Verwenden von Use Case B durch Use Case A

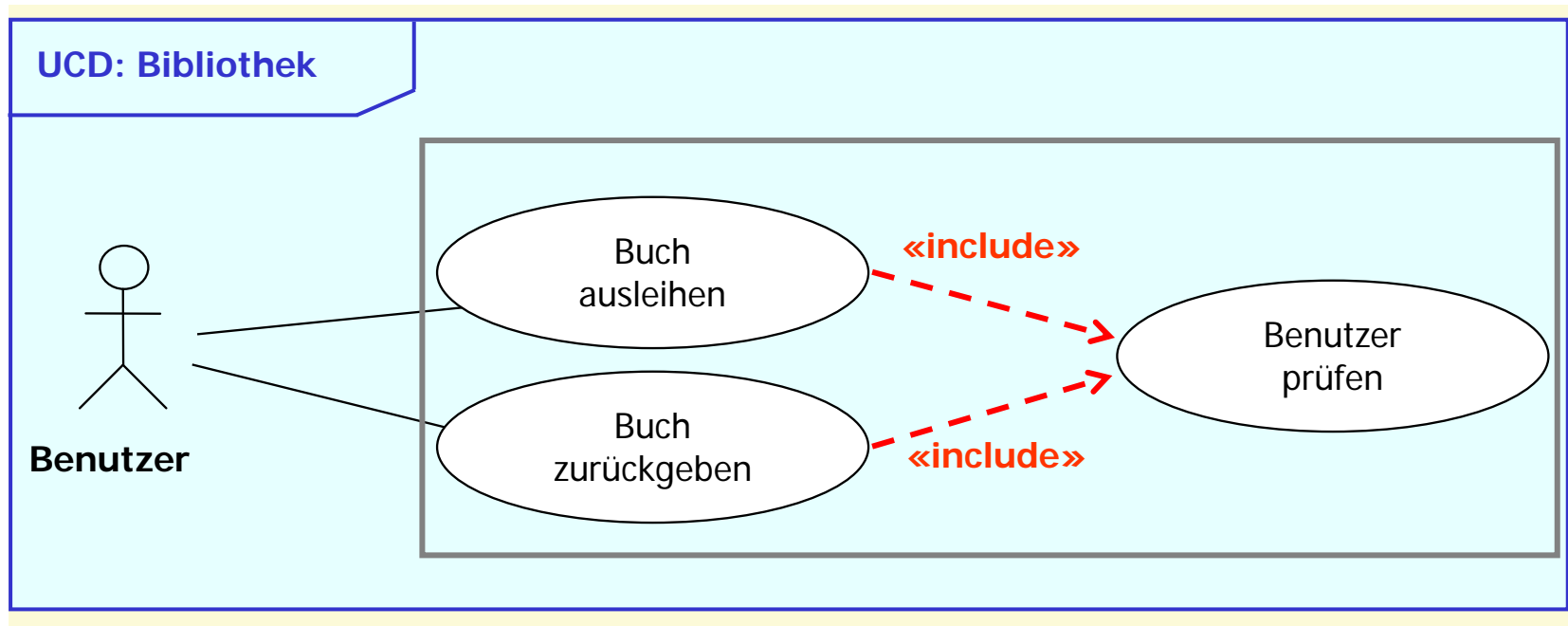
Use-Case Diagramm: include Beziehung

«include»-Beziehung:

- Use-Case kann durch andere Use Cases mehrfach inkludiert werden.

Vorteil:

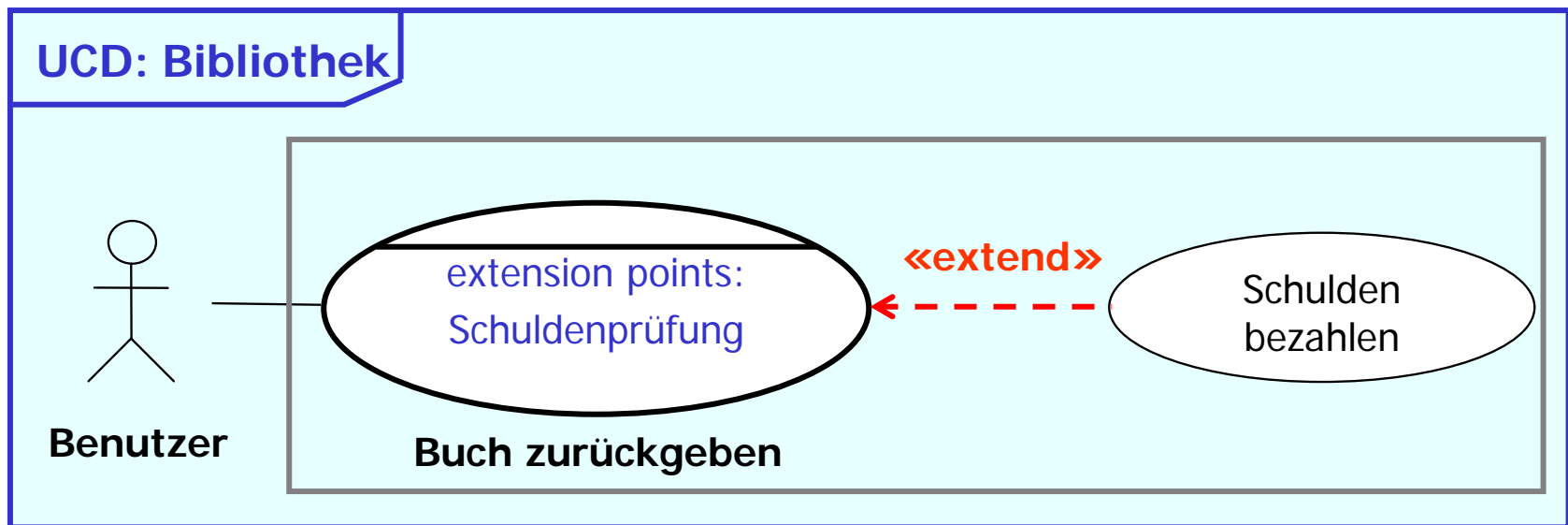
- mehrfach benötigtes Verhalten einmalig an zentraler Stelle beschreiben und beliebig oft nutzen.



Use-Case Diagramm: extend Beziehung

«extend»-Beziehung:

- zeigt an, dass das Verhalten eines Use-Case (A) durch einen anderen Use-Case (B) erweitert werden kann, aber nicht muss.



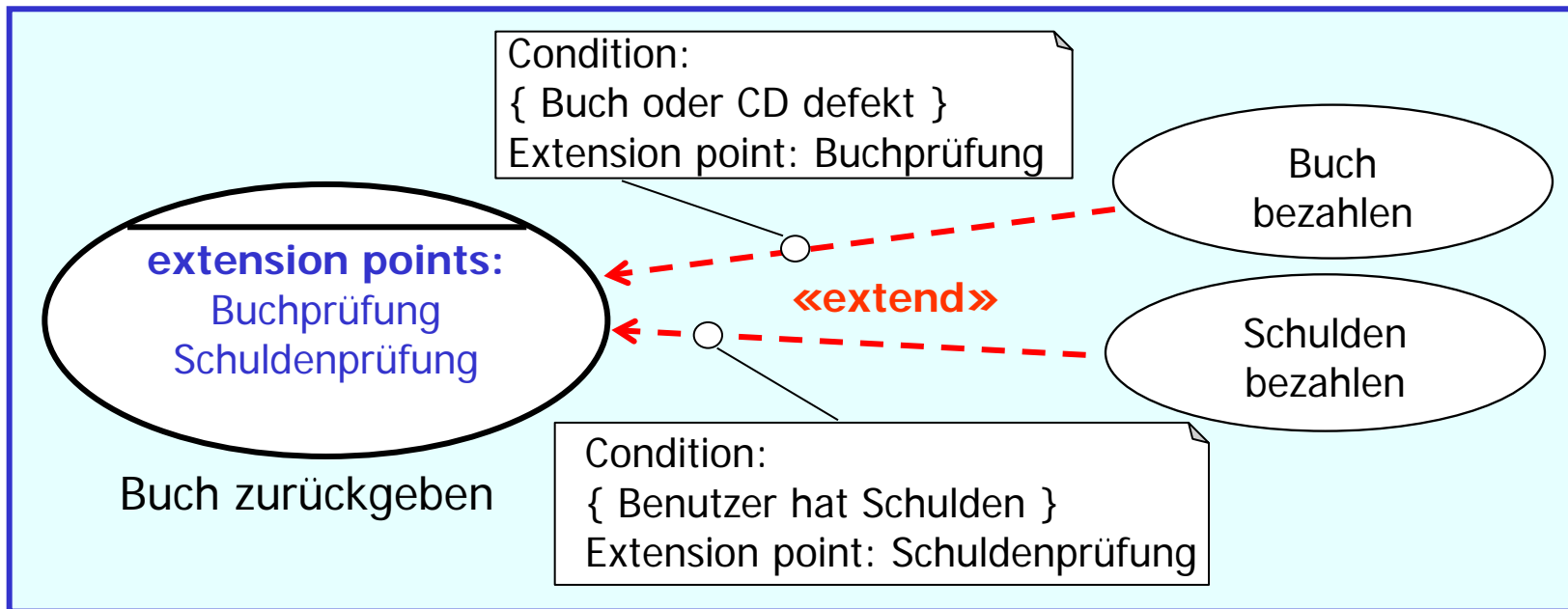
- Den Zeitpunkt, an dem ein Verhalten eines Use-Case erweitert werden kann, bezeichnet man als **Erweiterungspunkt** (engl. **extension point**).
- Ein Use-Case darf mehrere Erweiterungspunkte besitzen.

Use-Case Diagramm: extend Beziehung

Zusätzlich möglich: Bedingung für die Erweiterung

Sie wird bei Erreichen des Erweiterungspunktes geprüft.

- falls erfüllt: Referenzierter Use-Case wird durchlaufen
- sonst: Use-Case-Ablauf läuft „normal“ weiter



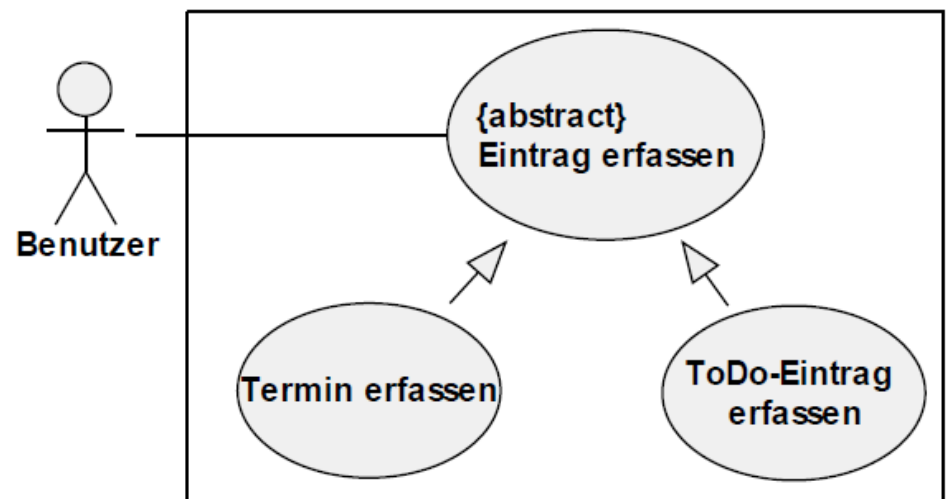
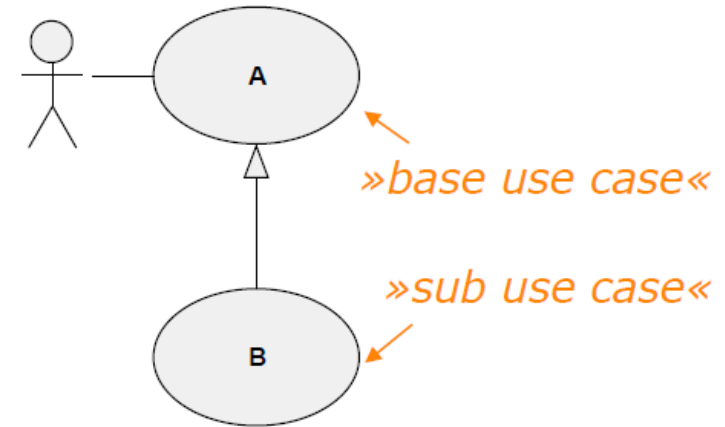
Fallbeispiel: Use-Cases einer Fluggesellschaft

Gegeben sei eine Fluggesellschaft.

- Entwickeln Sie ein Use Case Diagramm für den Akteur „Kunde“.
- Verwenden Sie folgende Use Cases:
- „Flug buchen“, „Essen bestellen“ und „Platz reservieren“.

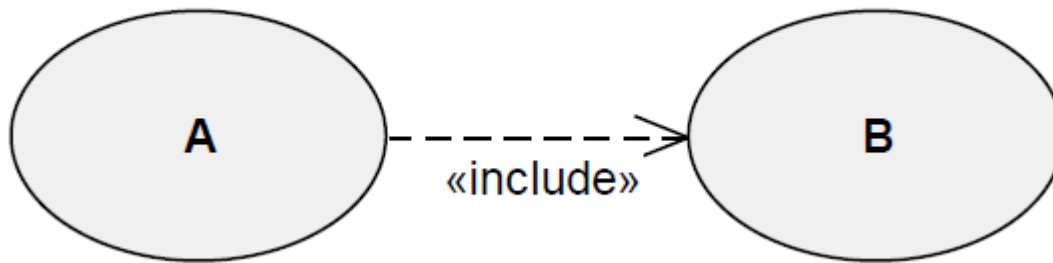
Generalisierung bei Use Cases

- B erbt das Verhalten von A und kann dieses überschreiben oder ergänzen
- B erbt alle Beziehungen von A
- B benötigt A (übernimmt Grundfunktionalität von A)
- B entscheidet, was von A ausgeführt bzw. geändert wird
- Modellierung abstrakter Anwendungsfälle möglich: {abstract} abstrakte Use Cases sind nicht ausführbar!



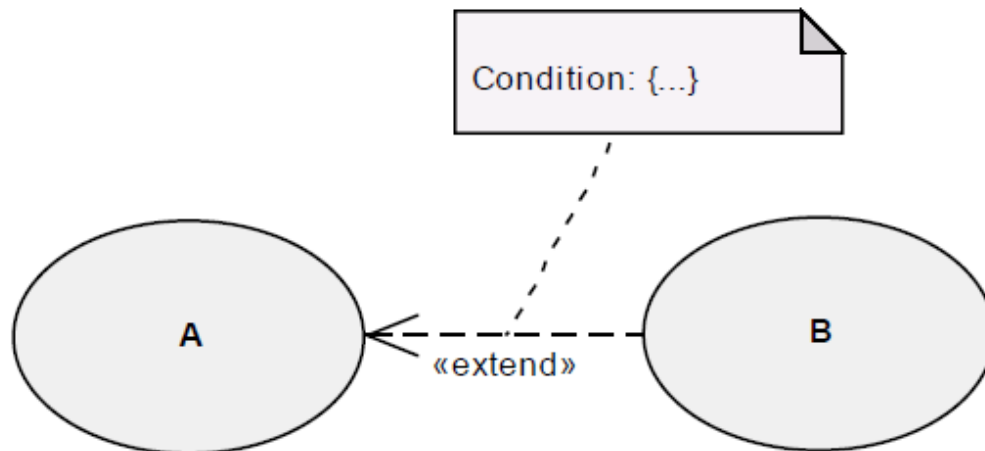
Analogien zu Programmiersprachen

- **<<include>>** entspricht Unterprogrammaufruf



```
void B () { ... }  
void A () {  
    ...  
    B();  
    ...  
}
```

- **<<extend>>** entspricht bedingtem Unterprogrammaufruf



```
void B () { ... }  
void A () {  
    ...  
    /* EP */  
    if Condition then B();  
    ...  
}
```

Typische Modellierungsfehler

Typische Modellierungsfehler

Use-Case Beschreibung

Use Case Name: Buch ausleihen

Auslösender Aktor: Benutzer

Zweck / Ziel: beschreibt den typischen Ablauf bei einer Buchausleihe

Eingehende Information: Barcode des Buchs, Benutzerausweis

Ergebnis: Buch wird vom System als ausgeliehen geführt

Grundlegender Ablauf:

1. Benutzerin legt Buch und Benutzerausweis vor.
2. System prüft, ob Ausleihsperrung vorliegt.
3. System speichert Exemplar-Nummer zusammen mit dem Tagesdatum und der Nummer des Benutzerausweises.
4. System berechnet aus der Ausleihfrist unter Berücksichtigung von Feiertagen und den Öffnungszeiten das Rückgabedatum.
5. System druckt Beleg mit Buchdaten und Rückgabedatum.

Erweiterungen:

- 1a [Ausweis abgelaufen] Benutzer muss Ausweis verlängern.
- 2a [Ausleihsperrung liegt vor] Benutzer muss Mahngebühren bezahlen.

Alternativen: -

Checkliste Use Case Diagramme

- Ist bei jedem Use Case **mindestens ein Akteur** beteiligt?
- Repräsentiert der **Akteur** eine **klare Rolle** oder ein klar definiertes technisches System?
- Enthält der Name des Use Case ein **Substantiv** und ein **Verb**
- Ist der Name des (primären) Use Case aus **Sicht des Akteurs** formuliert?
- Sind die **Abhängigkeiten** (Include, Extend) korrekt?
- Ist die Anwendungsfall Beschreibung vollständig?
- **Ist ein Trigger** für jeden Use Case vorhanden? - interne, externe und zeitliche Trigger.

Checkliste Use Case Beschreibungen

- Eine Use Case Beschreibung
 - skizziert **typischen Fall, ein System zu verwenden**.
 - ist wie ein **Theaterstück**. Die Use Case Beschreibung enthält die Choreographie.
 - hat eine **Einleitung**, einen **Hauptteil** und einen **Schluss**.
 - soll so einfach wie möglich – aber dennoch präzise definiert werden.
 - ist dann fertig beschrieben, wenn der Kunde, die Anwender und die Softwareentwickler ihn akzeptieren.
 - stellt die **Grundlage für einen Systemtest** dar.
 - sollte mit maximal zwei Seiten beschrieben werden.

Aktivitätsdiagramme

Aktivitätsdiagramme

Aufbauend auf Use Cases betrachten wir jetzt sog. **Aktivitätsdiagramme** (activity diagrams)

- **Aktivitätsdiagramme** sind **UML-Diagramme** mit denen man

- Reihenfolgen von Aktivitäten
- parallele Aktivitäten, etc.

modellieren kann.

- **Einsatzgebiete:**

- Modellieren von Geschäftsprozessen
- Modellieren von Abläufen innerhalb eines Use Cases (Alternative zu textuellen Use Case Beschreibungen)

Elemente von Aktivitätsdiagrammen

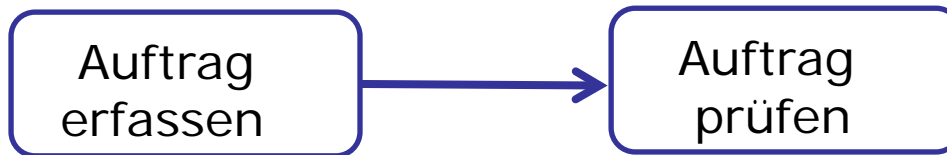
▪ Aktionen

Eine Aktion wird durch ein Rechteck mit abgerundeten Ecken dargestellt.



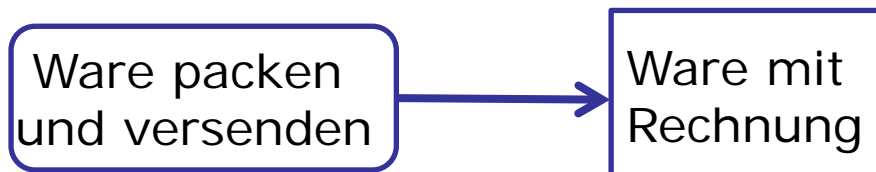
▪ Kontrollfluss

Der Kontrollfluss, d.h. die Kanten, zwischen den Aktionen wird durch gerichtete Hilfsstellen dargestellt



▪ Objektknoten

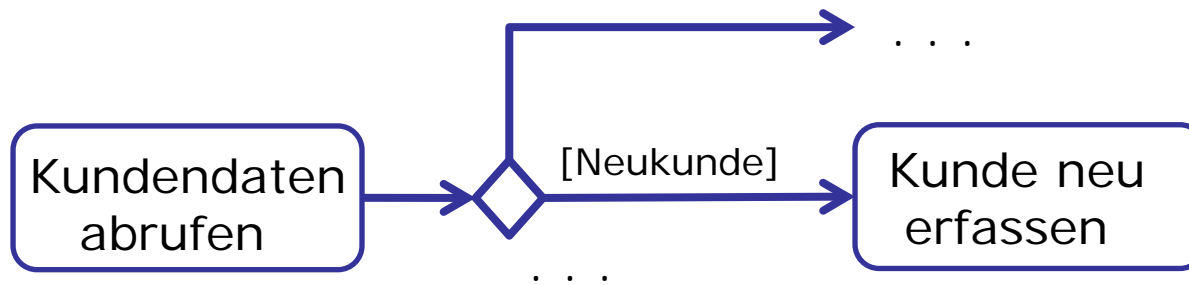
beschreiben Speicher für die Übergabe von Objekten bzw. Ressourcen



Elemente von Aktivitätsdiagrammen

▪ Entscheidungsknoten

beschreiben eine Verzweigung des Kontrollusses, wobei aus den möglichen Kontrollfüßen genau einer ausgewählt wird.

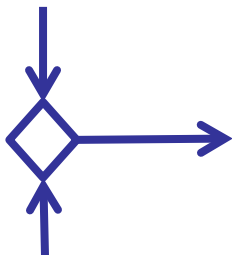


▪ Überwachungsbedingungen (Guards)

Steuern den Kontrolluss in eckigen Klammern an den ausgehenden Kontrollflüssen

▪ Verbindungsknoten

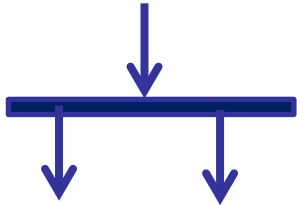
Es gibt auch sogenannte Verbindungsknoten (merge nodes), die mehrere alternative Kontrollflüsse zusammenfassen.



Elemente von Aktivitätsdiagrammen

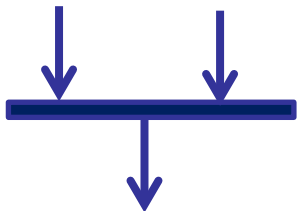
- **Gabelung**

Eine Gabelung (fork node) teilt einen Kontrollfluss in mehrere parallele Kontrollflüsse auf.



- **Vereinigung (auch: Synchronisationsknoten)**

Analog dazu gibt es die Vereinigung (join node), die mehrere parallele Kontrollflüsse zusammenfasst.

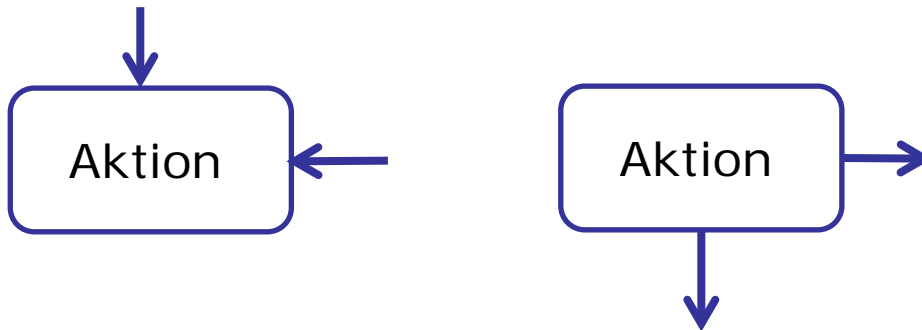


Elemente von Aktivitätsdiagrammen

▪ Bitte beachten:

- Kontrollflüsse dürfen nur an Objektknoten, Entscheidungsknoten und Gabelungen aufgespalten und
- an Objektknoten, Verbindungsknoten und Vereinigungen wieder zusammengeführt werden.

▪ Nicht erlaubt:



Elemente von Aktivitätsdiagrammen

▪ Startknoten

- entspricht einer initial markierten Stelle.
- ein Aktivitätsdiagramm darf nur einen Startknoten haben, in den kein Kontrollfluss hineinführt.



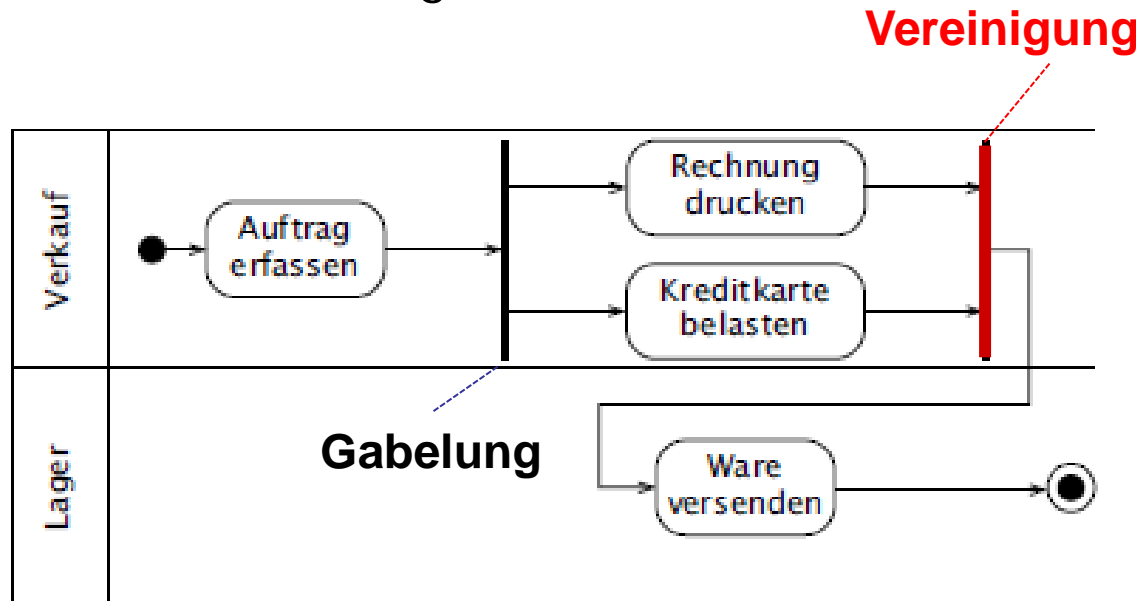
▪ Aktivitätsende

- signalisiert, dass alle Kontrollflüsse beendet werden.



Elemente von Aktivitätsdiagrammen

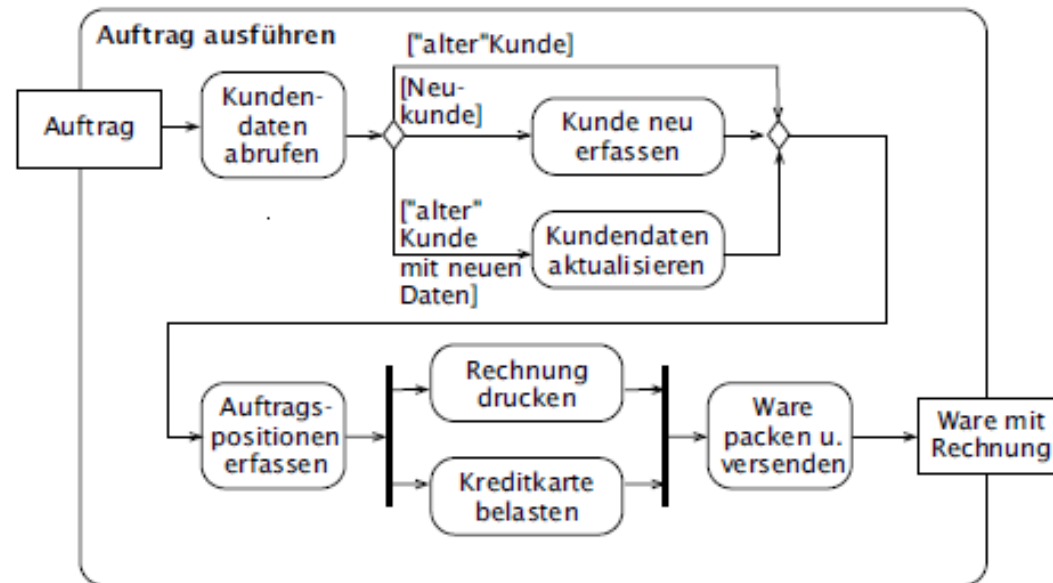
- **Aktivitätsbereiche (activity partitions/swimlanes):**
 - Zusammenfassung mehrerer Knoten (Aktionen, Objektknoten, etc.) zu einer Einheit
 - dient im allgemeinen dazu, um die Verantwortung für bestimmte Aktionen festzulegen.



(Quelle: Heide Balzert:
Analyse und Entwurf mit der UML 2)

Beispiele für Aktivitätsdiagramme

- Einfaches Beispiel mit verschiedenen **Knoten**



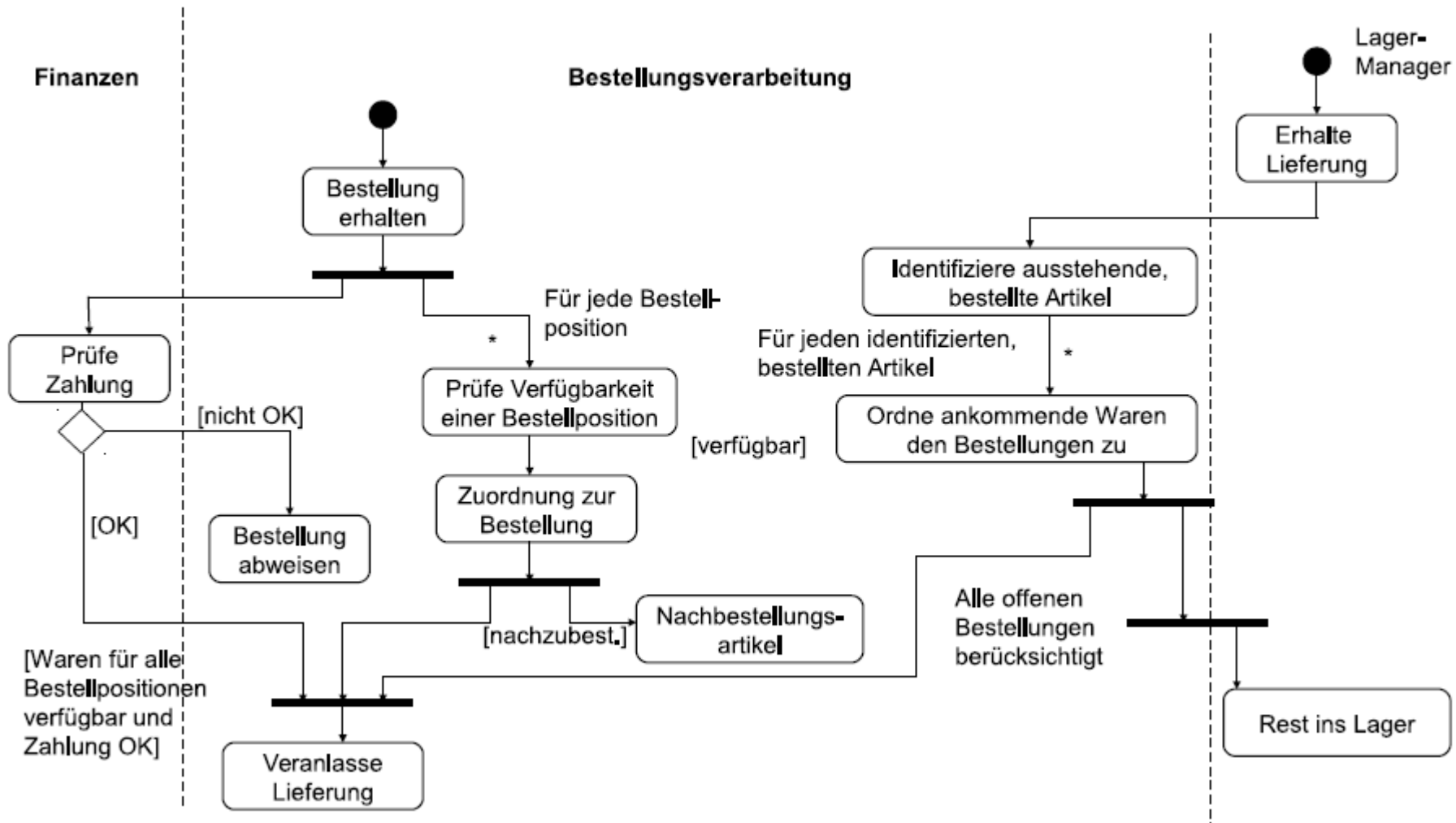
Beispiel: Ausführung eines Auftrags

(Quelle: Heide Balzert:

Analyse und Entwurf mit der UML 2)

Beispiele für Aktivitätsdiagramme

■ Komplexeres Beispiel



Zusammenfassung