

---

# Objektorientierte Modellierung mit der Unified Modeling Language

Michael Neuhold MSc.

---

# Anwendungsfalldiagramm

# Inhalt

---

- Einführung
- Akteure
- Anwendungsfall
- Beziehung zwischen Anwendungsfällen
- Beziehung zwischen Akteuren
- Pakete
- Zusammenfassung

# Einführung

---

- Anwendungsfälle (= Use Cases) sind Ausgangspunkt bei der Anforderungserhebung
- Summe der Anforderungsfälle (= **Anwendungsfalldiagramm**) beschreibt
  - wie sich der Kunde sein geplantes System vorstellt
  - wie er als Anwender damit interagieren möchte
- Anwendungsfalldiagramm beantwortet folgende Fragen
  - Was wird beschrieben? -> **System**
  - Wer interagiert mit dem System? -> **Akteur**
  - Was machen die Akteure? -> **Anwendungsfälle**

# Einführung

---

- Funktionale Zerlegung des zu entwickelnden Systems in Anwendungsfälle und in Akteure
  - Anwendungsfälle (Use Cases) repräsentieren die Anforderungen (Ziele) der Kunden
  - Akteure interagieren mit dem System im Kontext der Anwendungsfälle
- Ergebnisse der Anwendungsfall-Modellierung
  - Globales Anwendungsfalldiagramm (Use Case Diagram)
  - Weitere Anwendungsfalldiagramme nach Bedarf
  - Für jeden Anwendungsfall eine detaillierte textuelle Anwendungsfallbeschreibung

# Exkurs (Nicht Funktionale Anforderungen)

---

- Eine nicht funktionale Anforderung legt fest, welche Eigenschaften (Leistung, Qualität) das Produkt haben soll.
  - “Das System soll 100 Anfragen pro Sekunde verarbeiten können.”
  - “Das System kann 15000 Transaktionen pro Sekunde bearbeiten, ohne dass die CPU zu mehr als 50% ausgelastet ist.”
  - “Das System kann 4 Wochen im Dauerbetrieb ohne ein Neustarten betrieben werden.”
- Arten nicht funktionaler Anforderungen:
  - Zuverlässigkeit (Systemreife, Wiederherstellbarkeit, Fehlertoleranz)
  - Aussehen und Handhabung (Look and Feel, UX)
  - Benutzbarkeit (Verständlichkeit, Erlernbarkeit, Bedienbarkeit)
  - Leistung und Effizienz (Antwortzeit, Ressourcenbedarf)
  - Wartbarkeit (Analysierbarkeit, Stabilität, Prüfbarkeit)
  - Portierbarkeit (Anpassbarkeit, Austauschbarkeit)
  - Sicherheitsanforderungen (Vertraulichkeit, Informationssicherheit, Datenintegrität)
  - Skalierbarkeit (Änderung des Problem-Umfangs bewältigen)

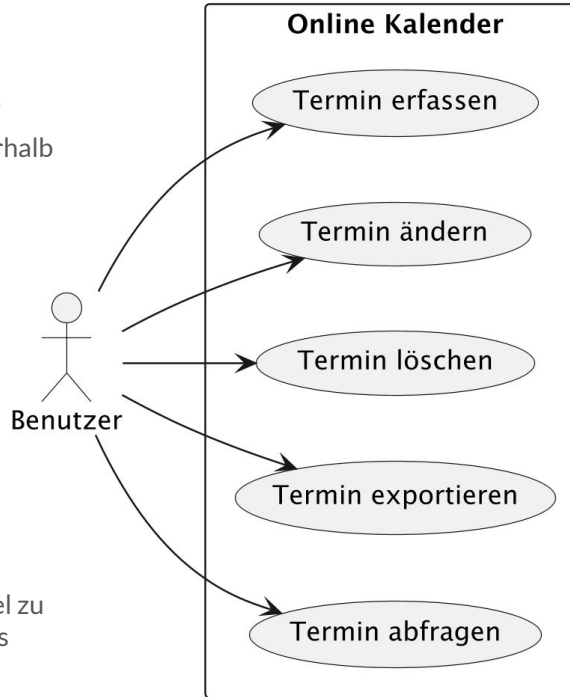
# Einführung

## Akteur: Wer benutzt das System?

Benutzer des Kalenders (befinden sich außerhalb des Systems)

## Beziehung:

Der Akteur benutzt den Use Case, um ein Ziel zu erreichen (z.B.: er führt Use Case durch, muss Eingaben machen o.Ä.)



## System: Was wird beschrieben?

Onlinekalender (=Systemgrenze)

## Use Case: Was machen Akteure?

Termin erfassen, Termin ändern?, Termin löschen, Termin abfragen, etc.

# Einführung

---

- Die **funktionalen Anforderungen** eines Systems auf einen Blick zeigen wollen
- Ihr System aus der Nutzersicht in handhabbare **logische funktionale Teile zerlegen** wollen
- die **Kommunikationspartner** des Systems modellieren möchten
- komplexe Systeme leicht verständlich und auf hohem **Abstraktionsniveau** darstellen möchten
- Planbare Einheiten



# Akteur

---

# Akteur

---

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

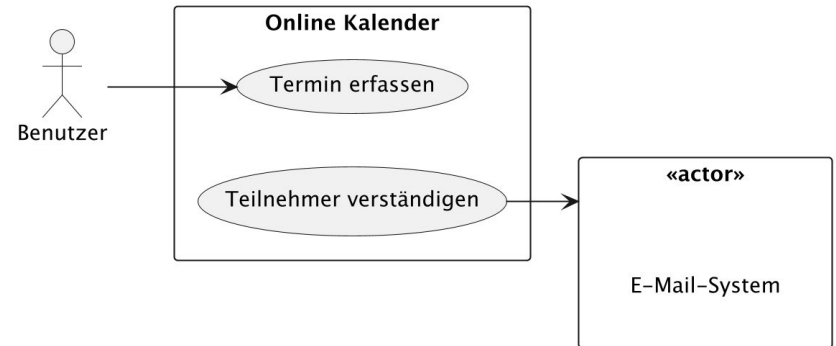
# Akteur

---

- Akteure repräsentieren **Rollen der Benutzer**
  - Konkrete Benutzer können gleichzeitig mehrere Rollen spielen, annehmen und ablegen
- Akteur befindet sich **außerhalb der Systemgrenzen**
- Üblicherweise werden Benutzerdaten auch innerhalb des Systems verwaltet. Diese werden als Objekte bzw. Klassen innerhalb des Systems modelliert.
- Beispiel: Kassier
  - Als Akteur am Kassenterminal
    - Die Rolle, in der die Person mit dem Kassensystem interagiert
  - Die Klasse Kassier umfasst Objekte, welche die Benutzerdaten enthalten (Name, Wohnort, ..)

# Akteur

- **Menschlich:** zb. Benutzer, Administrator, Lehrender
- **Nicht Menschlich:** zb. E-Mail-System, Datenbank
- **Primär:**
  - Hauptnutznießer der Anwendung, hat Anwender-Ziele, die durch Nutzung der Dienste des Systems erfüllt werden
- **Sekundär:**
  - notwendig für das Funktionieren des Systems, stellt einen Dienst für das System zur Verfügung zb. E-Mail-Server
- **Aktiv:** stößt selbst Anwendungsfall an
- **Passiv:** stößt selbst keine Anwendungsfälle an



# Akteur

---

- Wer startet und stoppt das System?
- Wer benutzt die wesentlichen Anwendungsfälle?
- Ist die “Zeit” ein Akteur, weil das System auf zeitliche Ereignisse reagiert? (Akteur Zeitmonitor einführen!)
- Wer ist für die Systemadministration zuständig?
- Mit welchem externen Geräten / Software-Systemen muss das System kommunizieren können?
- Wer interessiert sich für die Ergebnisse des Systems?
- Wer wird bei Fehlern oder Misserfolgen benachrichtigt?
- ...

Frage sollte lauten:

“Was sind Ihre Ziele, deren Ergebnisse einen messbaren Wert haben?”

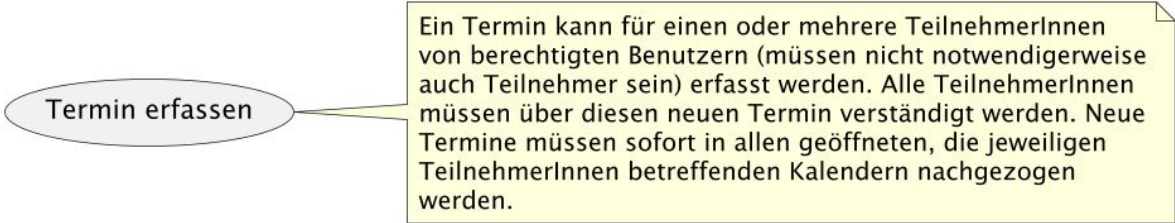
# Anwendungsfall (Use Case)

---

# Anwendungsfall (Use Case)

---

- Anwendungsfall umfasst eine **Menge von Aktionen**, die einen **messbaren Nutzen** für die Akteure bringen
- Anwendungsfälle **beschreiben das Verhalten**, das von dem zu entwickelnden System erwartet wird
- Identifizierung durch Sammeln von **Kundenwünschen** und Analyse der textuellen Problemstellung
- Kurzbeschreibung als Notiz



Termin erfassen

Ein Termin kann für einen oder mehrere TeilnehmerInnen von berechtigten Benutzern (müssen nicht notwendigerweise auch Teilnehmer sein) erfasst werden. Alle TeilnehmerInnen müssen über diesen neuen Termin verständigt werden. Neue Termine müssen sofort in allen geöffneten, die jeweiligen TeilnehmerInnen betreffenden Kalendern nachgezogen werden.

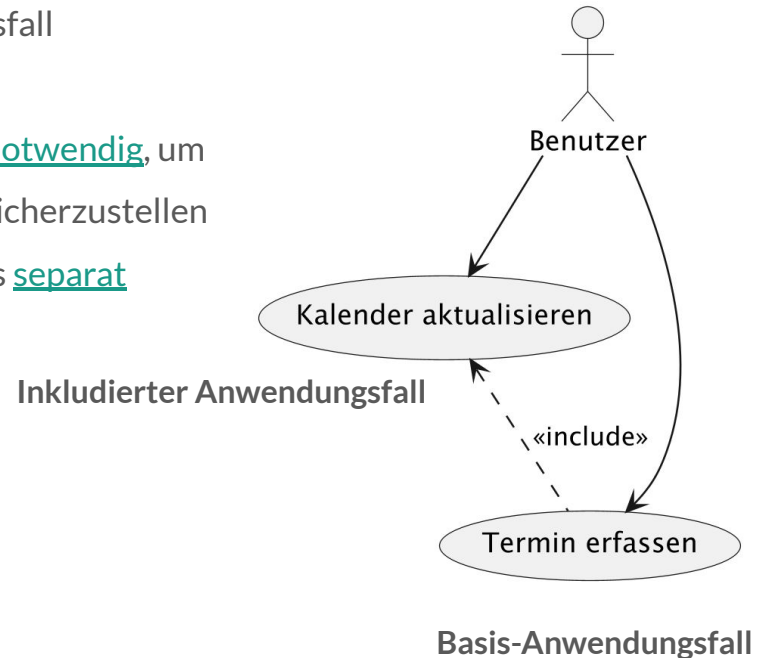
# Beziehungen zwischen Anwendungsfällen und Akteuren

---



# Beziehung zwischen Anwendungsfällen <<include>>

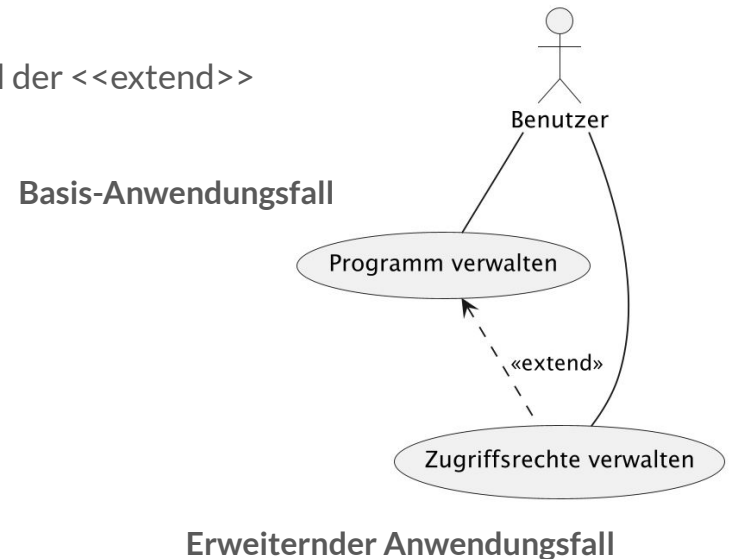
- Verhalten des **benutzten** Anwendungsfalls (inkludierter Anwendungsfall) wird in den **benutzenden** Anwendungsfall (Basis-Anwendungsfall) eingebunden
  - Der inkludierte Anwendungsfall B ist unbedingt notwendig, um die Funktionalität des Basis-Anwendungsfalls A sicherzustellen
  - Der inkludierte Anwendungsfall B kann allerdings separat ausgeführt werden



# Beziehung zwischen Anwendungsfällen <<extend>>

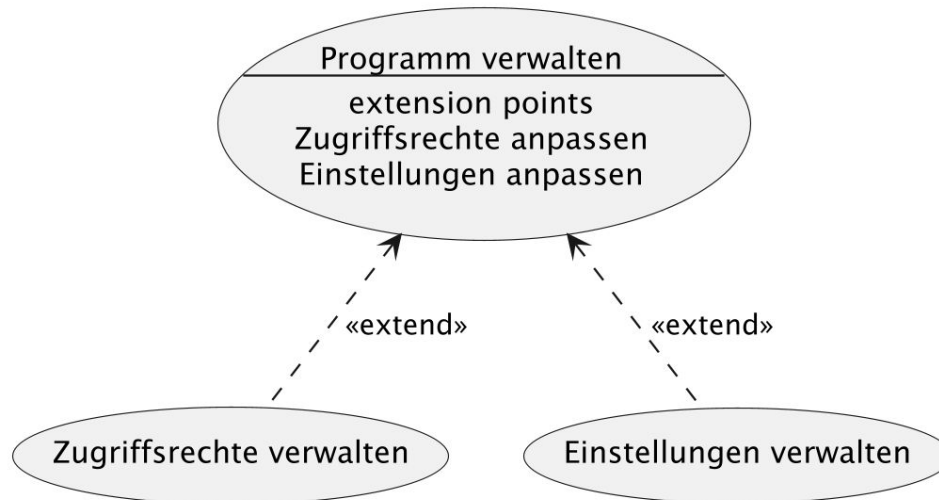
- Das Verhalten von B **kann** in A inkludiert werden
  - A entscheidet, ob B ausgeführt wird
- Der erweiternde Anwendungsfall B kann von A aktiviert werden, muss aber nicht
- Angabe des >>Wo<< durch **Erweiterungsstellung** in A
- Angabe des >>Wann<< durch **Bedingung** in A bzw. als Teil der <<extend>>

Beziehung



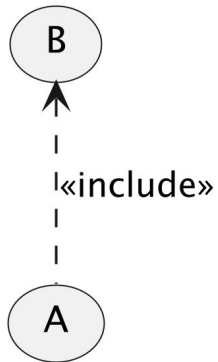
# Beziehung zwischen Anwendungsfällen

- Mehrere **Erweiterungsstellen** je nach Anwendungsfall möglich
- **Namen von Erweiterungsstellen**
  - müssen eindeutig sein
  - müssen nicht mit den Namen der erweiternden Anwendungsfällen übereinstimmen



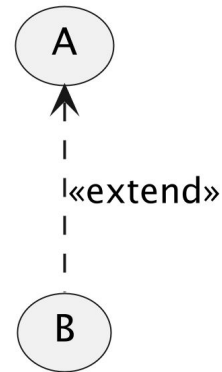
# Beziehung zwischen Anwendungsfällen

Include - entspricht Unterprogrammaufruf



```
void B() { /* TODO */ }  
void A() { /* TODO */ }  
  
// ...  
  
B();  
  
// ...
```

Extend - entspricht bedingtem Unterprogrammaufruf

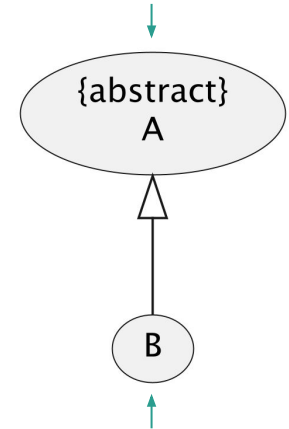


```
void B() { /* TODO */ }  
void A() { /* TODO */ }  
  
// ...  
  
if (condition) { B(); }  
  
// ...
```

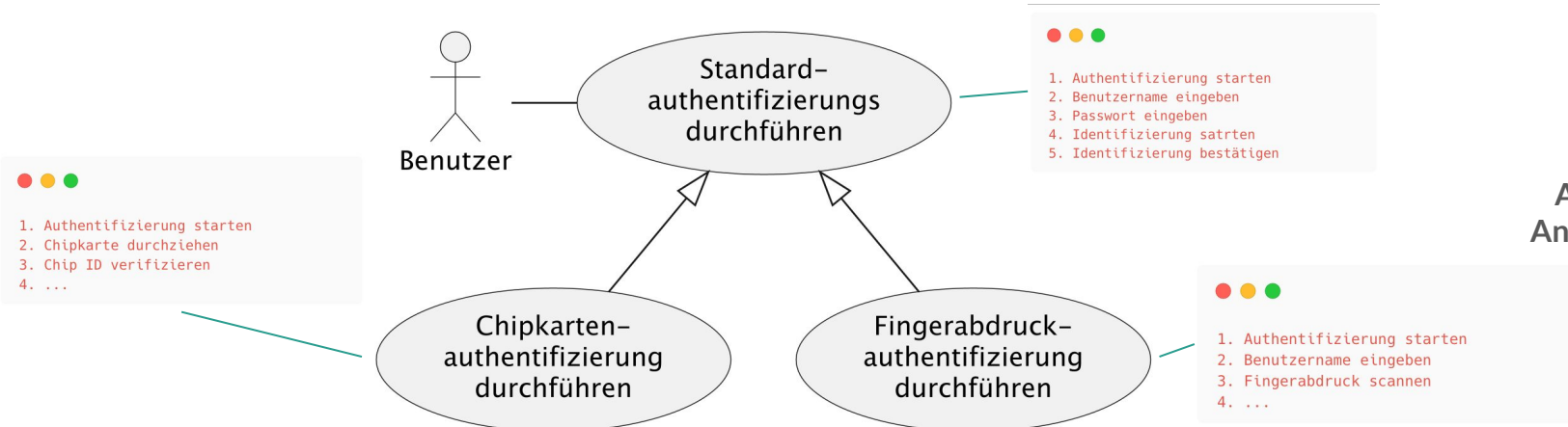
# Beziehung zwischen Anwendungsfällen

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

Basis-Anwendungsfall

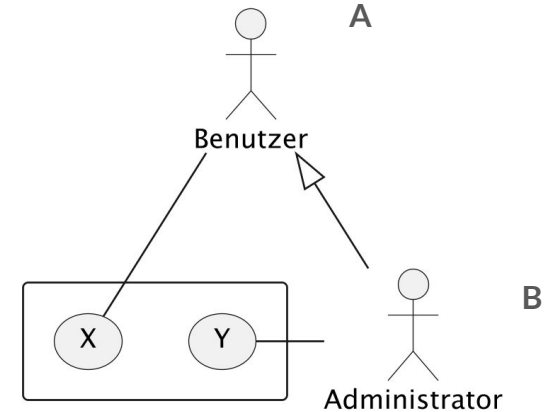
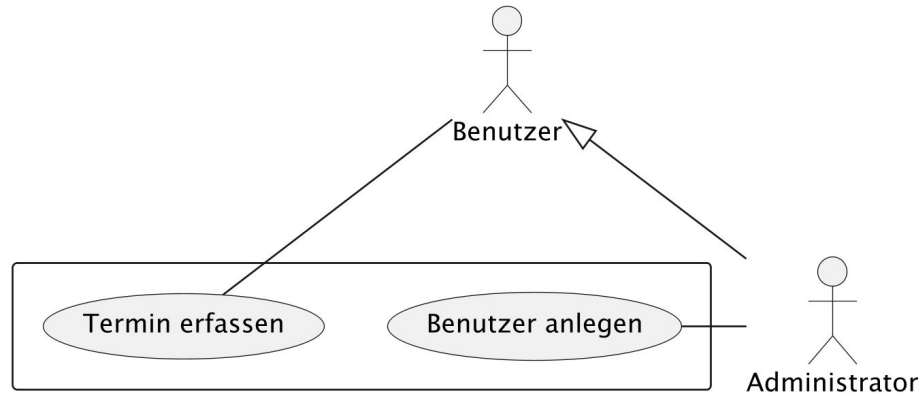


Abgeleiteter Anwendungsfall



# Beziehung zwischen Akteuren

- Akteur B **erbt** von Akteur A
- B kann mit den Anwendungsfällen X und Y kommunizieren
- A kann nur mit X kommunizieren



# Beispiel

---





# Pakete

---

# Pakete - Gliederung des Anwendungsfalldiagramms

---

- Anwendungsfalldiagramme werden schnell unübersichtlich
- Allgemeiner UML-Abstraktionsmechanismus: Paket (Package)
- Gliederung erfolgt nach fachlichen Kriterien

