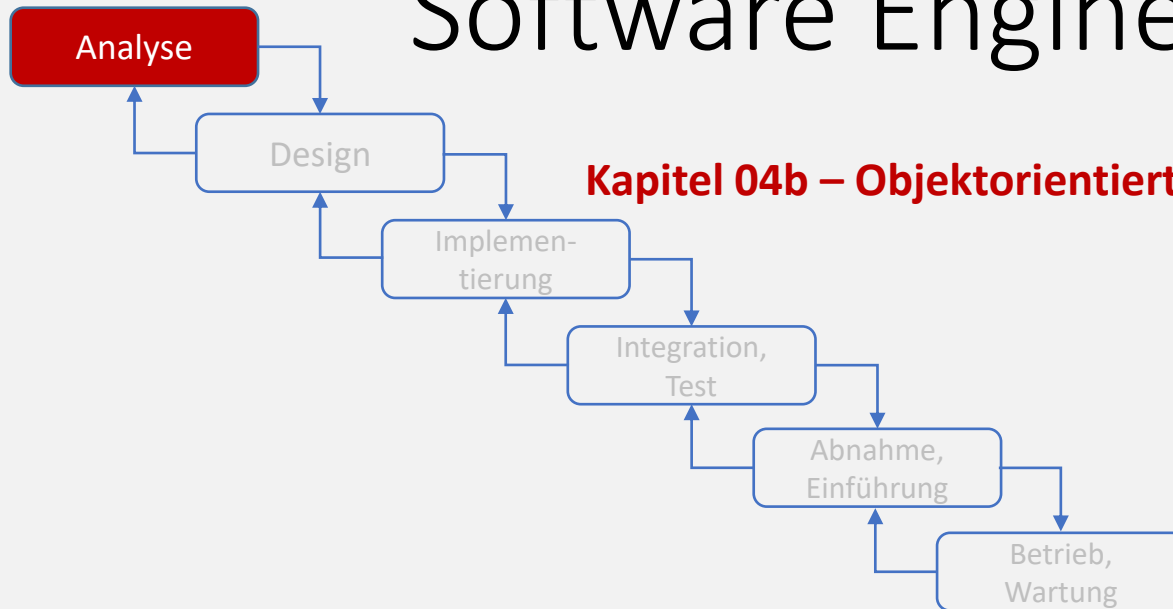




Software Engineering

Kapitel 04b – Objektorientierte Analyse (OOA)



Use-Case-Diagramme

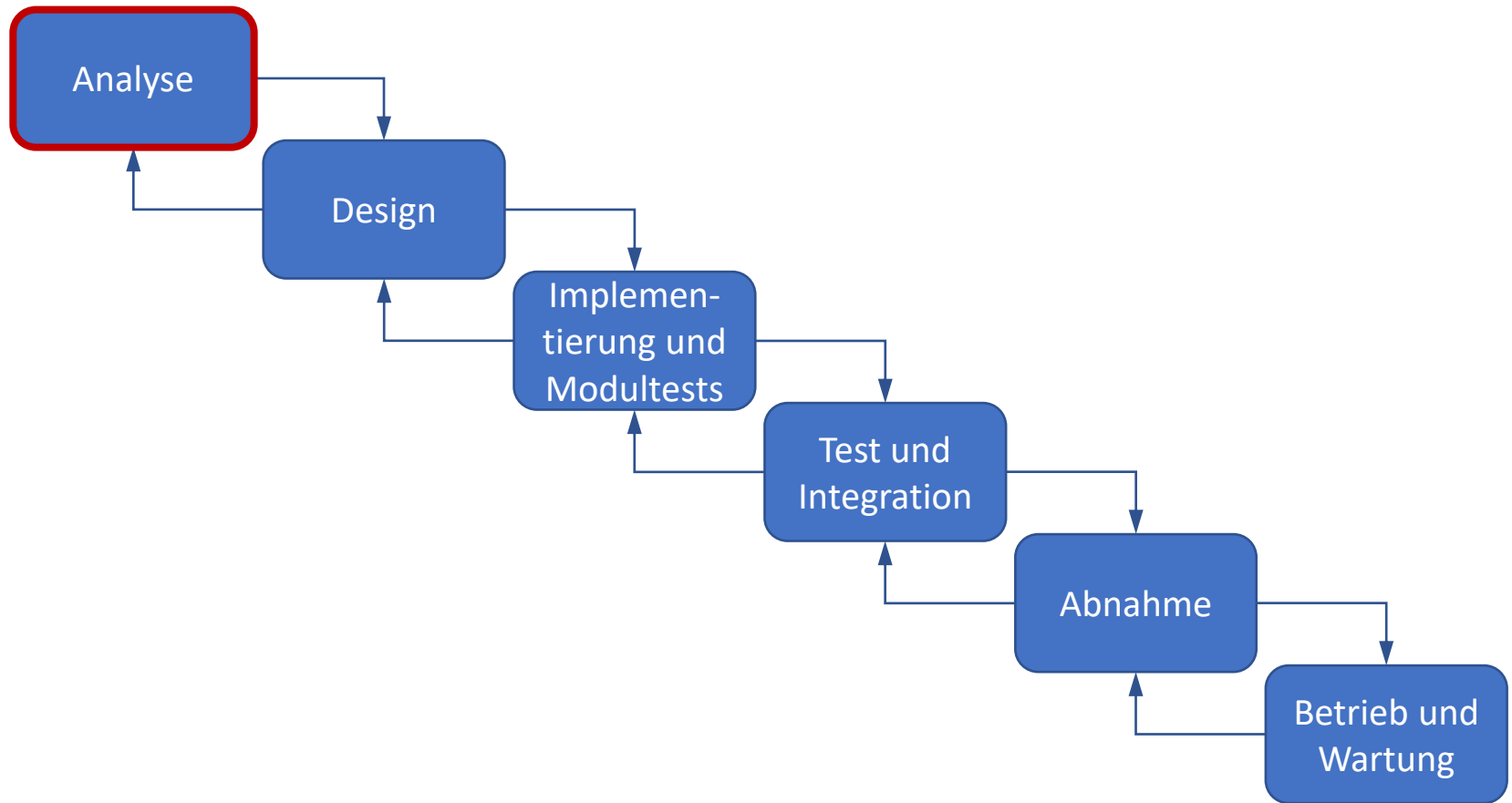
Gliederung

1. Motivation, Definition
2. Start der Objektorientierten Analyse (OOA)
 - Analyse im Großen
 - Statisches und dynamisches Modell
3. Modellieren mit UML

Lernziele

- Was ist Objektorientierte Analyse?
- Warum betreiben wir Objektorientierte Analyse?
- Welche Diagrammarten unterstützen uns bei der OOA?
- Wie modellieren wir damit?

Phasen eines Software-Projektes



Inhalt

Analyse-Phase ist gegliedert in:

- Requirements Engineering
- Anforderungsmodellierung
(Objektorientierte Analyse)
 - Makroprozess
 - UML-Diagramme
 - Analysemuster



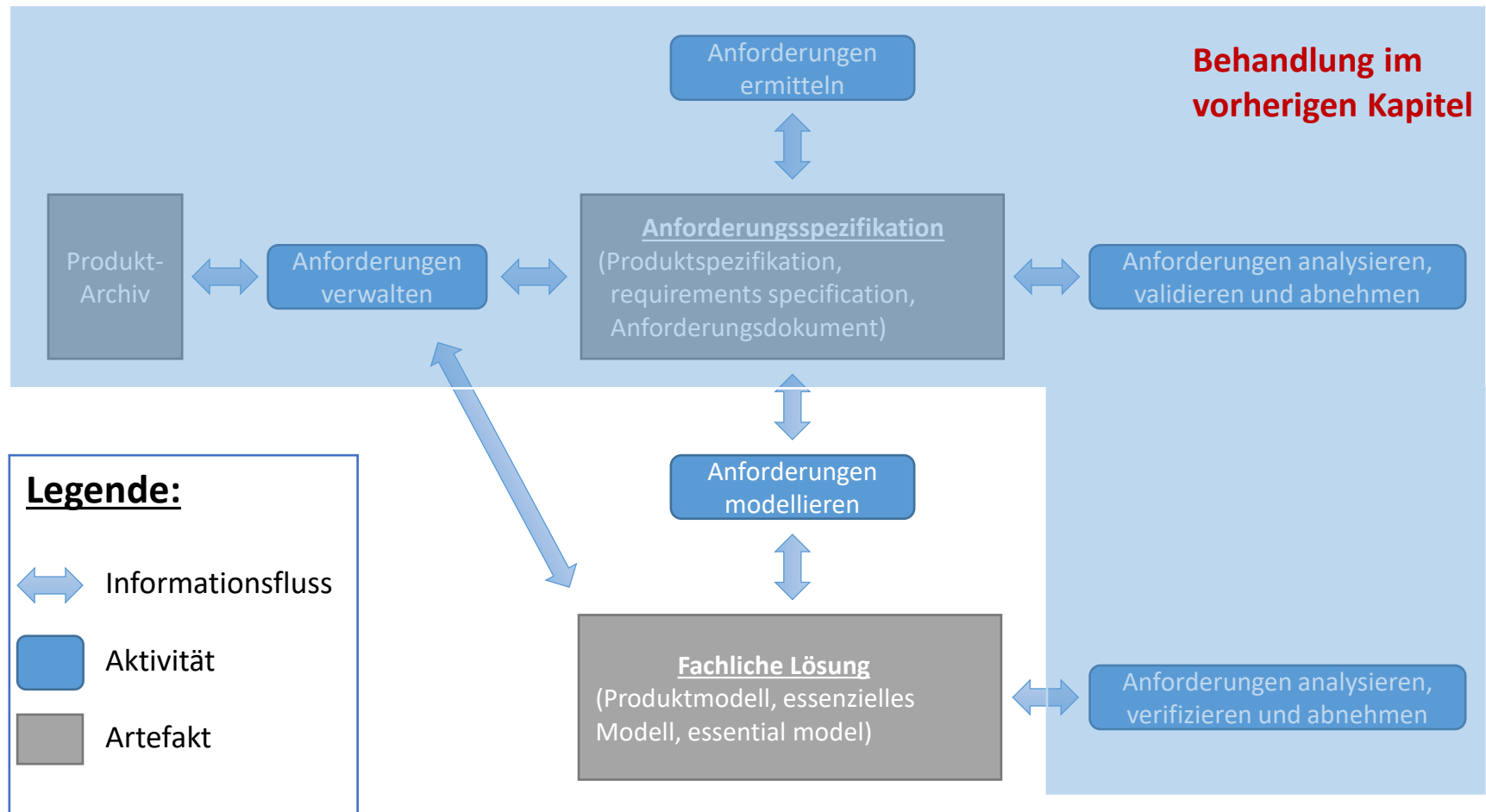
Thema des vergangenen Kapitels



Thema dieses Kapitels

Hinweis: Balzert sieht Modellierung der Anforderungen auch als Teil des Requirement Managements. Hier wird das explizit getrennt.

Requirements Engineering: Aktivitäten und Artefakte



Definition von OOA

Methodische Vorgehensweise zur Erstellung eines objektorientierten Analysemodells

Definition von OOA

Heide Balzert:

“Ermittlung und Beschreibung der Anforderungen an ein Software-System mittels objektorientierter Konzepte und Notationen. Das Ergebnis ist ein OOA-Modell.“

Artefakt dieser Phase: Fachliche Lösung (Produktmodell) mit den Eigenschaften:

1. Korrekte Umsetzung der Anforderungen
2. Vollständige Umsetzung der Anforderungen
3. Referenzierung der Anforderungen
4. Präzise, eindeutige und konsistente Formulierung der fachlichen Lösung

OO Fachkonzept: Beispielaufbau (1)

1. Einleitung
2. Überblick
3. Anforderungen
4. Geschäftsprozesse
5. Use Cases
6. Statisches Modell
7. Dynamisches Modell

OO Fachkonzept: Beispielaufbau (3)

8. Mengengerüst

9. IT Architektur (fortgeschrieben)

10.Exception Handling, Logging

11.Sicherheit

12.Persistente Datenhaltung

13.Schnittstellen

14.Benutzermodell, Benutzerdokumentation

OO Fachkonzept: Beispielaufbau (3)

15. Testspezifikation

16. Abnahmeverfahren und Kriterien

17. Betriebliche Dokumentation

18. Sollkonzept Infrastruktur (fortgeschrieben)

19. Wirtschaftlichkeit (fortgeschrieben)

20. Risikoanalyse (fortgeschrieben)

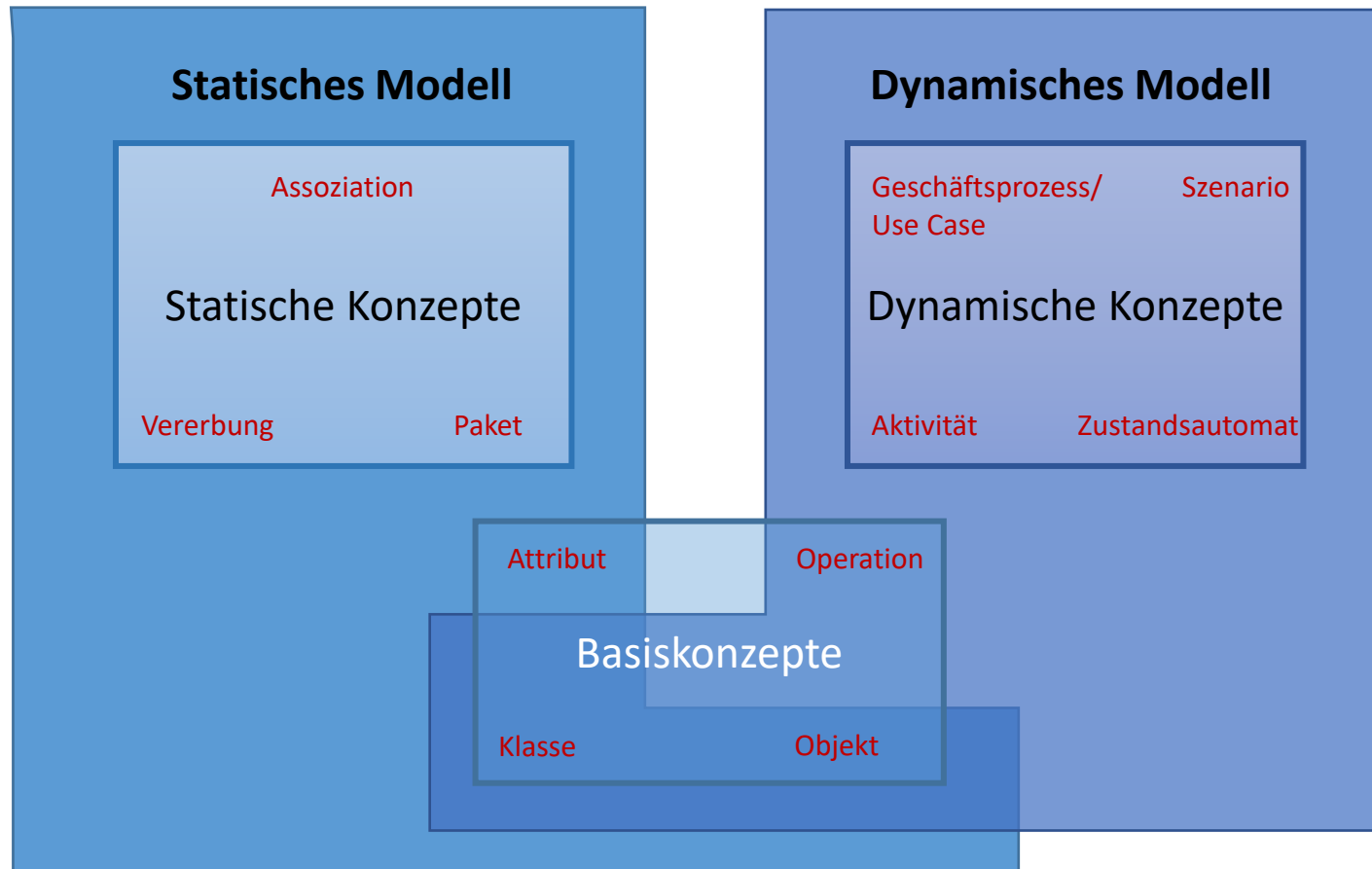
21. Offene Punkte (aktualisiert)

22. Projektplanung (aktualisiert)

23. Glossar

24. Referenzen

OOA Konzepte



[Quelle: Heide Balzert]

UML und Diagrammtypen

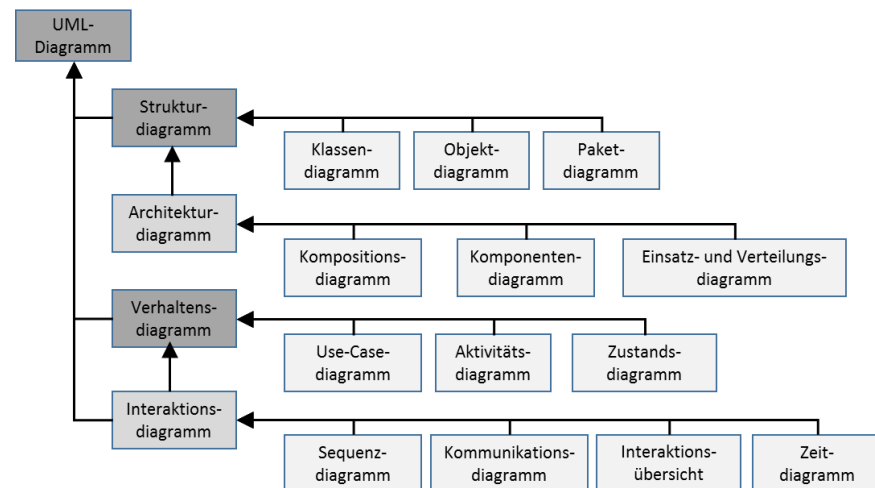
UML:

Die Unified Modeling Language (UML) ist eine grafische Darstellungsform:

- Zur Visualisierung, Spezifikation, Konstruktion, Dokumentation von (Software-)Systemen
- Sie bietet eine Menge an standardisierten Diagrammtypen, mit denen
- Komplexe Sachverhalte, Abläufe und Systeme
- Einfach, übersichtlich und verständlich dargestellt werden können

UML-Diagramme (Auszug):

- Klassendiagramm
- Paketdiagramm
- Use-Case-Diagramm
- Aktivitätsdiagramm
- Zustandsdiagramm
- Sequenzdiagramm
- ...

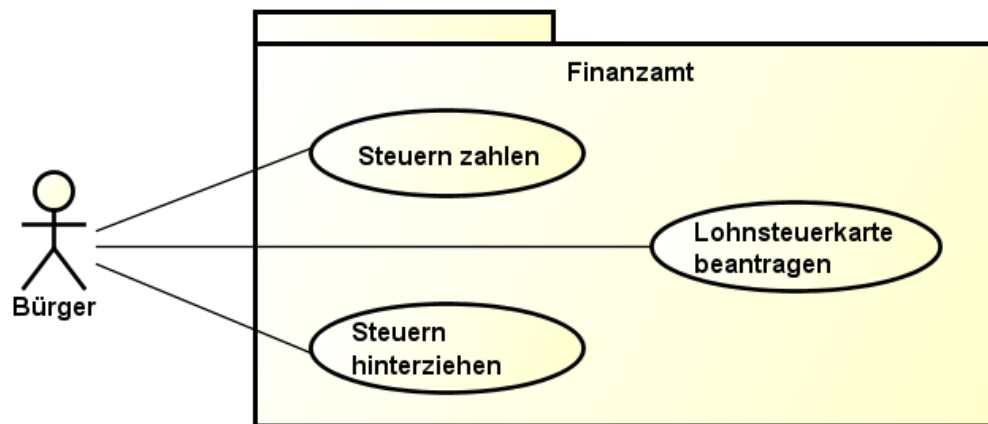


Grobübersicht der Diagrammtypen

Use-Case-Diagramm (Analysephase):

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

- Welche Use Cases sind in der Anwendung enthalten?
- Welche Akteure lösen die Use Cases aus?
- Welche Abhängigkeiten bestehen zwischen den Use Cases?

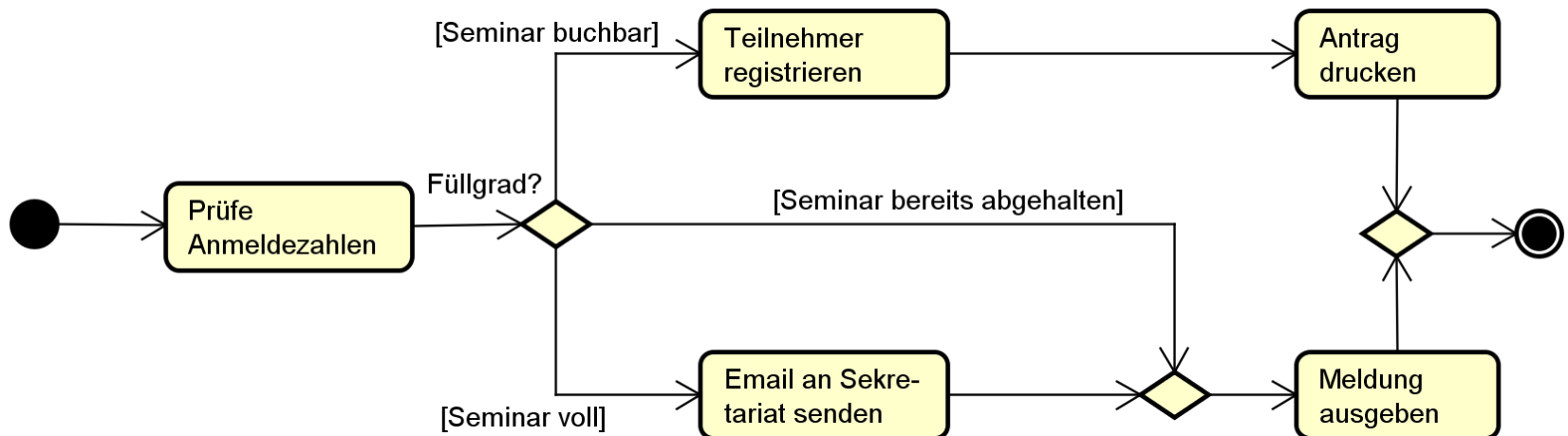


Grobübersicht der Diagrammtypen

Aktivitätsdiagramm (Analyse- und Designphase):

Klärt: Wie läuft ein bestimmter flussorientierter Prozess oder ein Algorithmus ab?

- Welche Schritte werden innerhalb eines Use Cases durchlaufen?
- Welche Zustandsübergänge erfahren die beteiligten Objekte?

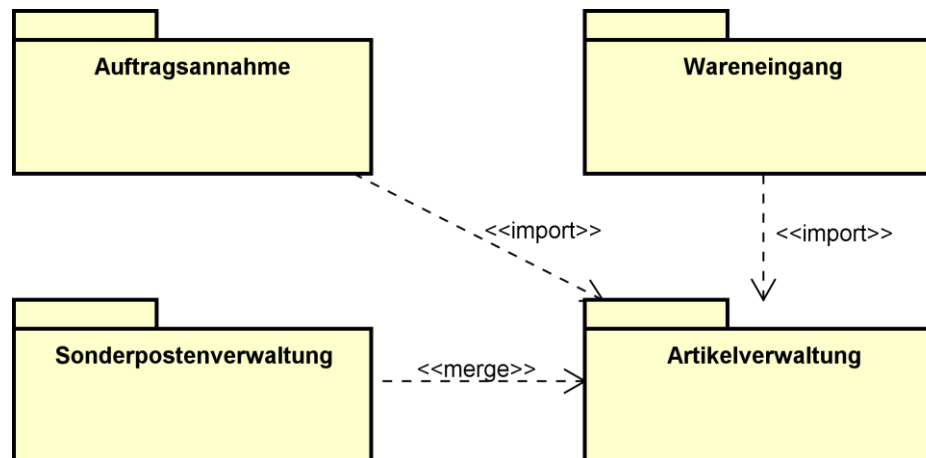


Grobübersicht der Diagrammtypen

Paketdiagramm (Analyse- und Designphase):

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

- In welche Pakete kann die Anwendung zerlegt werden?
- Welche Pakete ermöglichen eine weitere Unterteilung?
- Welche Kommunikation muss zwischen den Paketen realisiert werden?

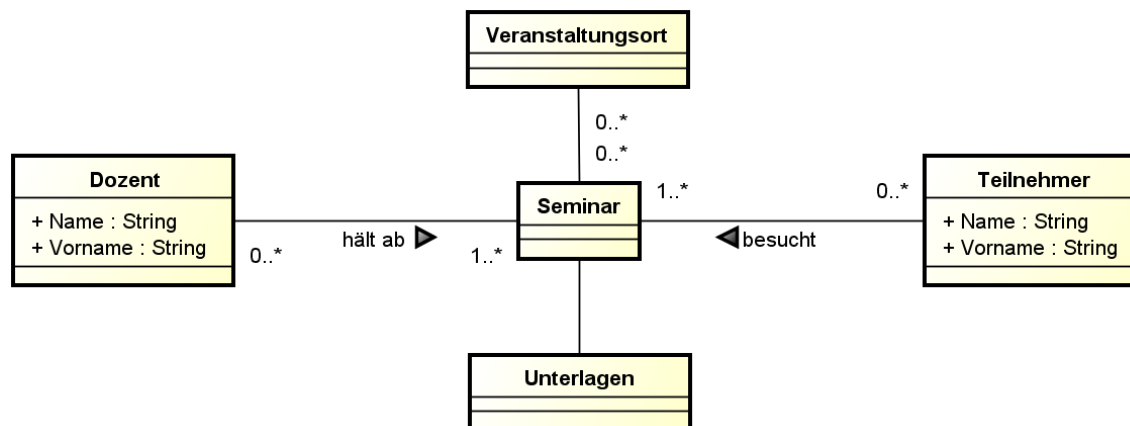


Grobübersicht der Diagrammtypen

Klassendiagramm (Analyse- und Designphase):

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

- Welche Zusammenhänge bestehen in der Aufgabenstellung (Domain Model)?
- Welche Klassen, Komponenten und Pakete sind beteiligt?
- Über welche Kommunikation findet die Zusammenarbeit statt?
- Welche Methoden und Eigenschaften benötigen die Klassen?
- Wie viele Objekte stehen mindestens und höchstens in Verbindung?

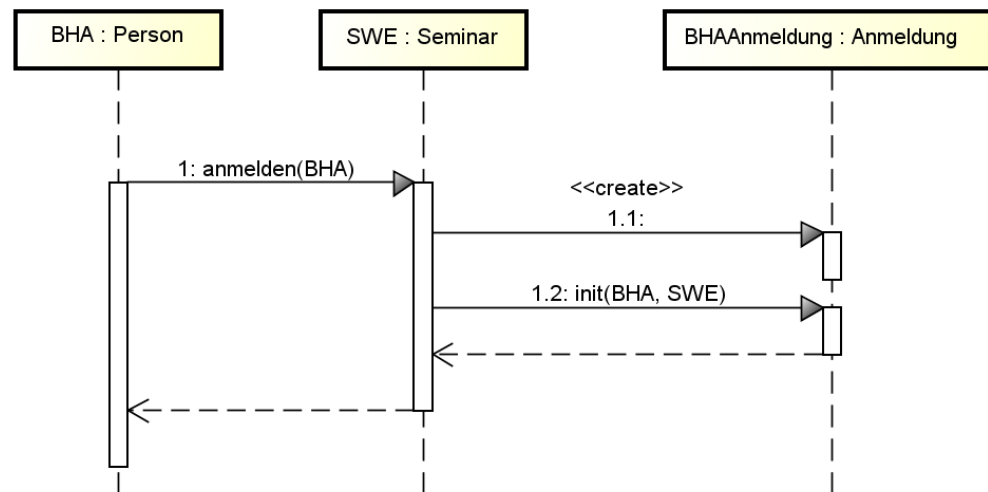


Grobübersicht der Diagrammtypen

Sequenzdiagramm (Designphase):

Klärt: Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?

- Welche Methoden sind für die Kommunikation zwischen ausgewählten Objekten zuständig?
- Wie ist der zeitliche Ablauf von Methodenaufrufen?
- Welche Objekte werden erstellt/zerstört?

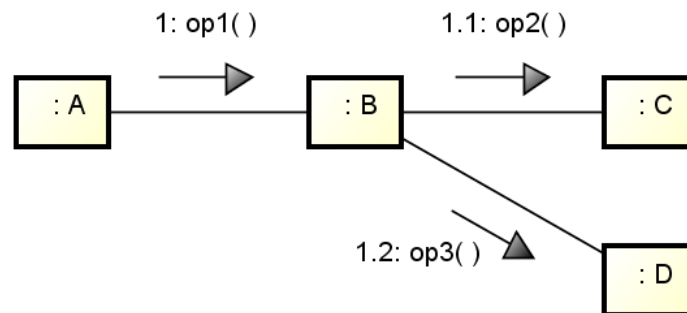


Grobübersicht der Diagrammtypen

Kommunikationsdiagramm (Designphase):

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

- Wie kommunizieren ausgewählte Objekte miteinander?
- Wie lautet die grobe Reihenfolge von Methodenaufrufen?
- Welche alternativen Methodenaufrufe gibt es?

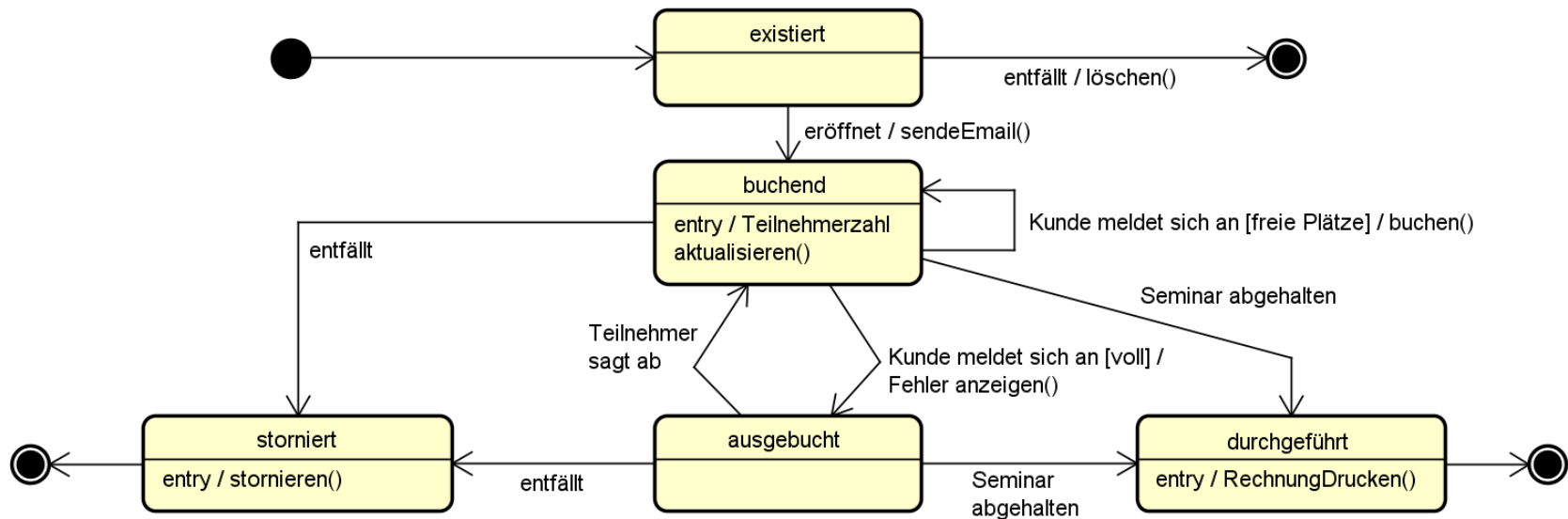


Grobübersicht der Diagrammtypen

Zustandsdiagramm (Designphase):

Klärt: Welche Zustände kann ein Objekt, eine Schnittstelle, ein Use Case, ... bei welchen Ereignissen annehmen?

- Welche Zustandsübergänge werden durch welche Methoden ausgelöst?
- Welcher Zustand wird nach Erzeugen eines Objekts eingenommen?
- Welche Methoden zerstören das Objekt?

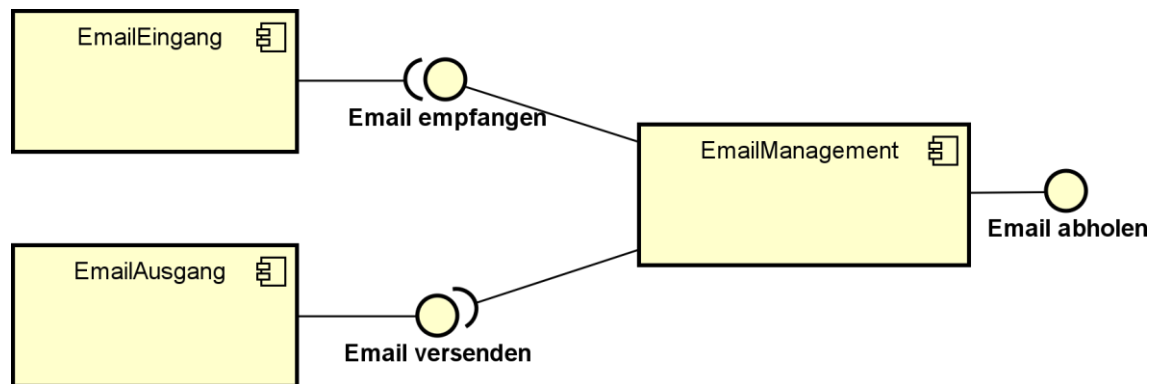


Grobübersicht der Diagrammtypen

Komponentendiagramm (Designphase):

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

- Wie werden Soft- und Hardwareteile mit definierter Funktion und definierten Interfaces gekapselt?
- Welche Komponenten haben Interfaces zueinander?
- Welche Softwareteile erzeugen die Funktionalität in Komponenten?

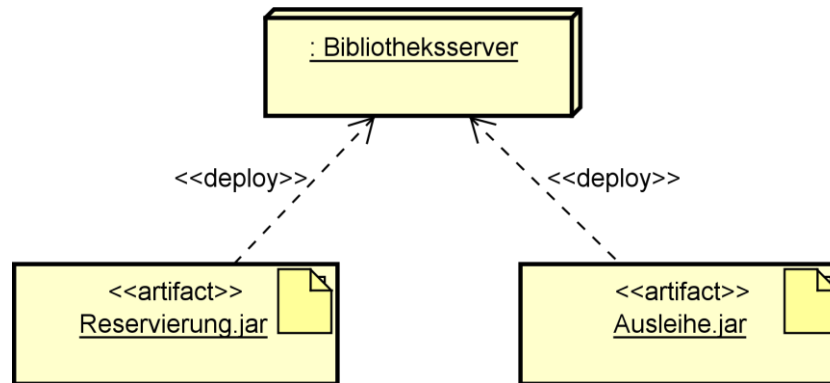


Grobübersicht der Diagrammtypen

Einsatz- und Verteilungsdiagramm (Designphase):

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

- Welche Computer arbeiten zusammen?
- Welche Module werden auf welcher HW ausgeführt?
- Auf welchen Kommunikationsmöglichkeiten basiert die Zusammenarbeit?



Start der OOA

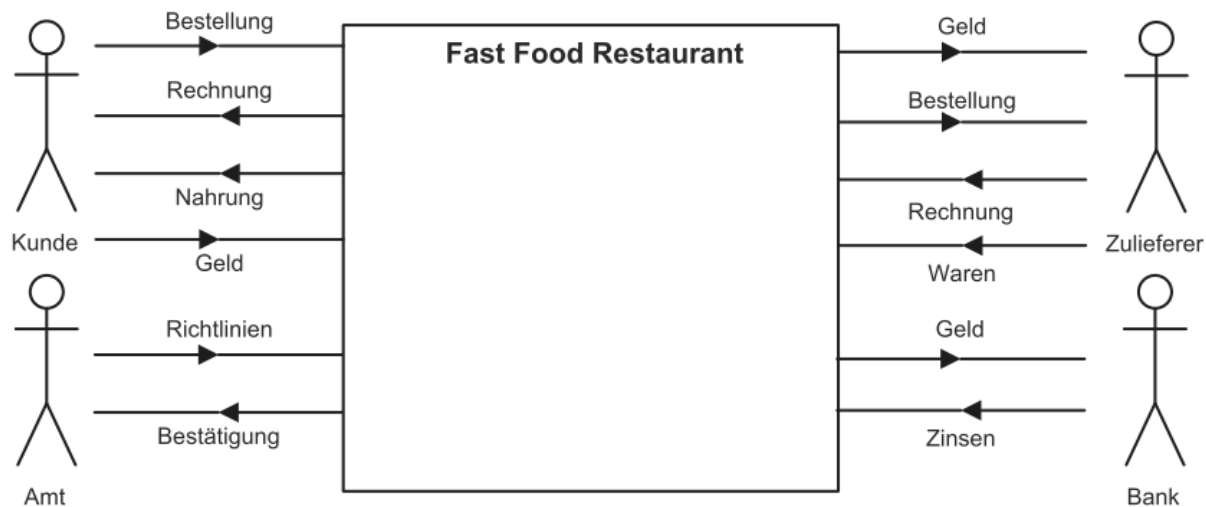
Konkretes Vorgehen bei der OOA

Das Kontextdiagramm

Früh in Analysephase: Systemkontext erfassen und dokumentieren

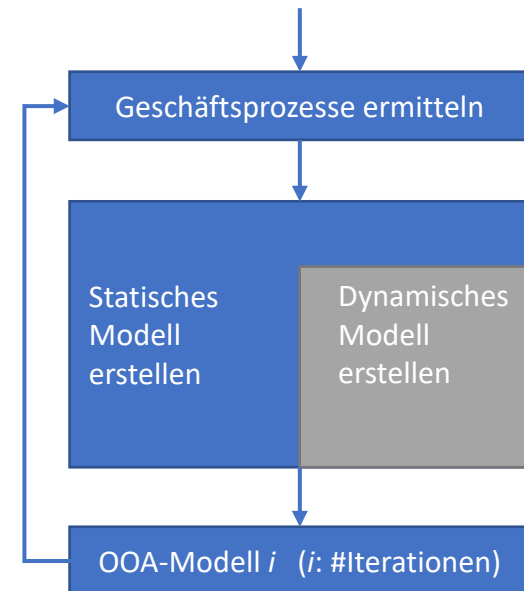
- Kontextdiagramm beschreibt Abgrenzung von Systemen zu Nachbarsystemen und zeigt Ein- und Ausgangsfluss
- Kann in unterschiedlicher Notation (z.B. als Paket-, Klassen, Use-Case-Diagramm) dargestellt werden

Beispiel:



Start der eigentlichen OOA

- Wie fange ich an?
- Ausgangspunkt: Requirement Specification und Kontextdiagramm liegen vor
- **Makroprozess der OOA** (Prozess der OOA kann auf verschiedenen Wegen erfolgen)
 - Ermitteln Sie die relevanten Geschäftsprozesse
 - Leiten Sie daraus Klassen ab
 - Erstellen Sie das statische Modell
 - Erstellen Sie parallel dazu das dynamische Modell
 - Berücksichtigen Sie die Wechselwirkung beider Modelle



[Quelle: Heide Balzert, Lehrbuch der Objektmodellierung]

Aufgabenbereiche des Makroprozesses

- **Makroprozess:**

- Analyse im Großen
- Erstellung eines statischen Modells
- Erstellung eines dynamischen Modells

- **Alternativen:**

- Szenariobasierter Makroprozess:

- Bei umfangreichen funktionalen Anforderungen
- Es existieren keine alten Datenbestände

- Datenbasierter Makroprozess

- Bei umfangreichen existierenden Datenbeständen
- Der Umfang der funktionalen Anforderungen ist zunächst unbekannt

Vergleich von Makroprozessen

Szenariobasiert:

(keine alten Datenbestände)

- Geschäftsprozesse formulieren
- Daraus Szenarios ableiten
- Daraus Interaktionsdiagramme ableiten
- Klassendiagramme erstellen
- Zustandsdiagramme erstellen

Datenbasiert:

(alte Datenbestände liegen vor)

- Klassendiagramme erstellen
- Geschäftsprozesse formulieren
- Daraus Szenarios ableiten
- Interaktionsdiagramme ableiten aus Klassendiagrammen und Szenarios
- Zustandsdiagramme erstellen

Aufgabenbereiche des Makroprozesses

- Analyse im Großen
- 6 Schritte zum statischen Modell
- 4 Schritte zum dynamischen Modell

Analyse im Großen

Use-Case-Modell aufstellen, liefert als Ergebnisse:

- Use-Case-Diagramme
- Use-Case-Definitionen (s. Schablone)
- Aktivitätsdiagramme (falls Fokus auf Verarbeitungsschritten liegt)
- Zustandsdiagramme (falls ausgedrückt werden soll, welche Zustände im Verlauf der Verarbeitung angenommen werden)

Pakete bilden, beinhaltet:

- Teilsysteme festlegen (Modellelemente zu Paketen zusammenfassen)
- Bei großen Systemen erfolgt Paketbildung meist zu Anfang
- **Ergebnis: Paketdiagramm**

Entwicklung eines statischen Modells (1)

(6 Schritte)

1. Klassen identifizieren:

- Für jede Klasse nur so viele Attribute, Operationen identifizieren, wie für Problemverständnis und einwandfreies Erkennen der Klasse notwendig
- **Ergebnis: Klassendiagramm, Kurzbeschreibung der Klassen**

2. Assoziationen identifizieren:

- Zunächst nur reine Verbindungen (z.B. ohne Multiplizitäten/ Art der Assoziationen)
- **Ergebnis: Klassendiagramm**

3. Attribute identifizieren:

- Alle Attribute des Fachkonzeptes identifizieren
- **Ergebnis: Klassendiagramm**

4. Vererbungsstruktur identifizieren

- Auf Basis der identifizierten Attribute Vererbungsstrukturen erstellen
- **Ergebnis: Klassendiagramm**

Entwicklung eines statischen Modells (2)

(6 Schritte)

5. Assoziationen vervollständigen:

- Unterscheiden in “normale” Assoziation, Aggregation oder Komposition
- Festlegen der Multiplizitäten, Rollen, Namen und Restriktionen
- **Ergebnis: Klassendiagramm, Objektdiagramm**

6. Attribute spezifizieren:

- Vollständige Spezifikation für alle identifizierten Attribute erstellen (dazu gehören Typ, Multiplizität und Eigenschaftswerte)
- **Ergebnis: Attributspezifikation im Klassendiagramm**

Entwicklung eines dynamischen Modells (4 Schritte)

1. Szenarios erstellen:

- Jeden Geschäftsprozess/Use Case durch Menge von Szenarios beschreiben
- **Ergebnis: Sequenzdiagramm, Kollaborationsdiagramm (Alternativ/ergänzend auch Aktivitätsdiagramme)**

2. Zustandsautomat erstellen:

- Für jede Klasse prüfen: muss/kann ihr dynamisches Verhalten durch Zustandsdiagramm präziser spezifiziert werden?
- **Ergebnis: Zustandsdiagramm**

3. Operationen eintragen:

- **Ergebnis: Klassendiagramm**

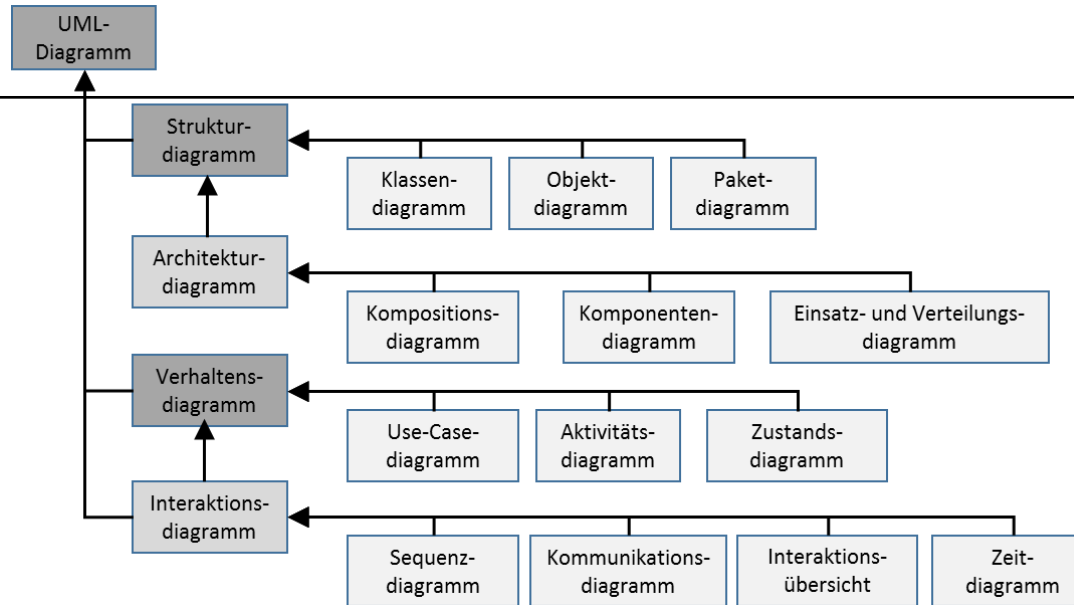
4. Operationen beschreiben:

- Überlegen, ob eine Beschreibung notwendig ist. Falls „ja“ auch über Komplexitätsgrad Gedanken machen
- **Ergebnis: Klassendiagramm, fachliche Beschreibung der Operationen, Zustandsautomaten, Aktivitätsdiagramme**

Erstellung von Modellen

Allgemeine Hinweise:

- OOA-Modellerstellung ist **kreativer** Prozess
- Es gibt **keine richtigen oder falschen** Modelle
- Es gibt nur Modelle, die **mehr oder weniger gut** ihren Zweck erfüllen
- Ein gutes Modell ist immer **verständlich** (d.h., es sieht einfach aus)
- Erstellung verständlicher Modelle erfordert **viel Arbeit und Erfahrung**
- Modellieren Sie **kein System**, das zu flexibel ist/**zu viele Sonderfälle** enthält
Warum?
 - Solche Modelle sind wegen ihrer Komplexität immer schwer verständlich
- Prüfen Sie für jeden **Sonderfall**, ob es wert ist, die **Komplexität** des Modells dadurch zu erhöhen



Modellieren mit UML

Analyse im Großen

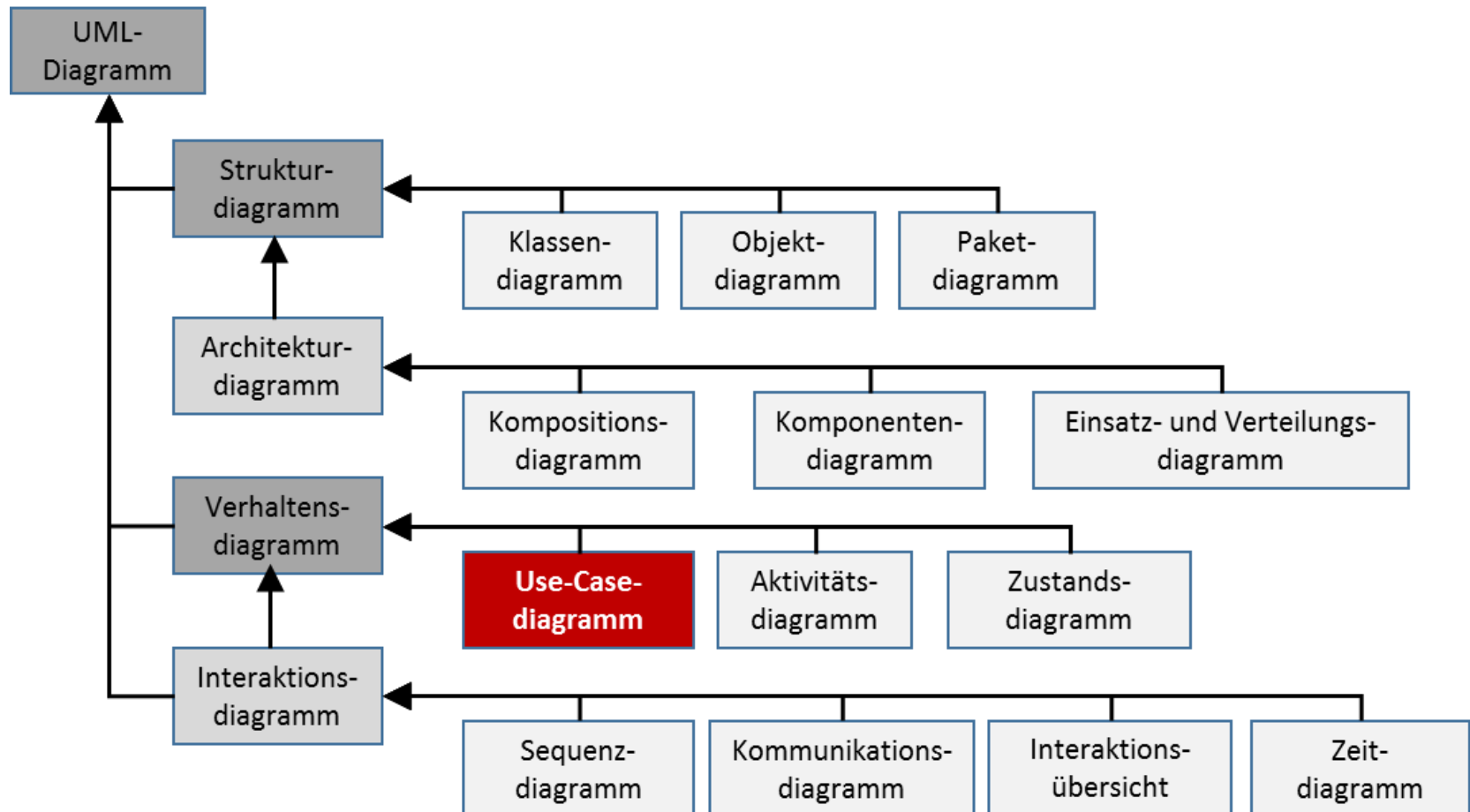
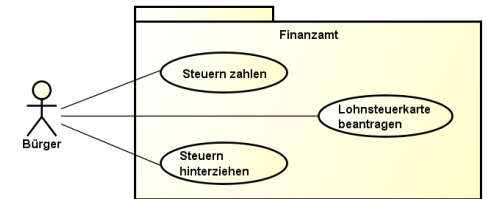
Use Case Modell aufstellen, liefert als Ergebnisse:

- Use-Case-Diagramme
- Use-Case-Definitionen
- Aktivitätsdiagramme
- Zustandsdiagramme

Pakete bilden, beinhaltet:

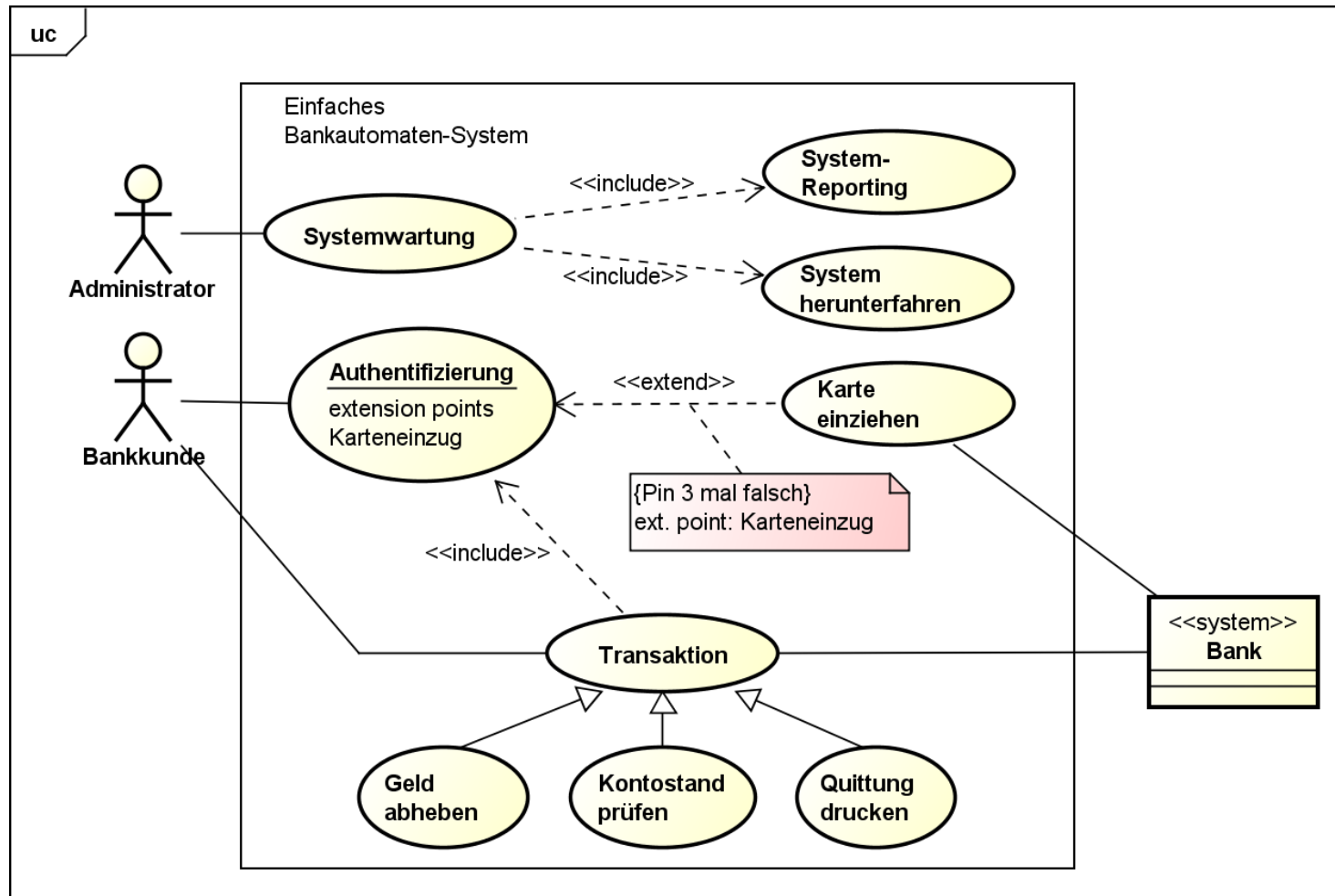
- Teilsysteme festlegen (Modellelemente zu Paketen zusammenfassen)
- Bei großen Systemen erfolgt Paketbildung meist zu Anfang
- Ergebnis: Paketdiagramm

UML: Use Cases



UML: Use-Case-Diagramm

Beispiel



UML: Use Case

Unterschiedliche Notationen für Use Cases (Anwendungsfälle):

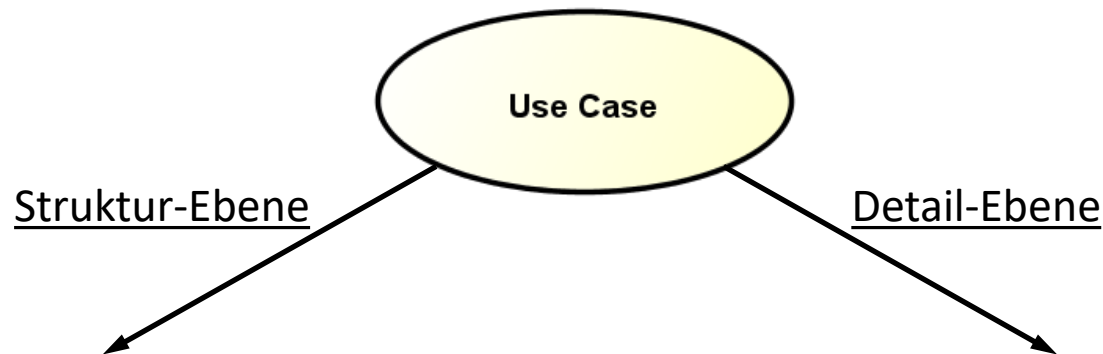
- Use-Case-Definition
- Tabellarische Use-Case-Definition
- Use-Case-Diagramm

Definition “Use Case”:

- Spezifiziert Sequenz von Aktionen einschließlich möglicher Varianten, die das System in Interaktion mit Akteuren auslöst [Quelle: Heide Balzert]
- Abstrahiert von konkreter technischer Lösung
- Wird durch bestimmtes Ereignis ausgelöst
- Wird ausgeführt, um ein konkretes Ziel zu erreichen/gewünschtes Ergebnis zu erstellen
- Ist als Black-Box zu verstehen
 - Beschreibt also extern wahrnehmbares Verhalten
 - Geht nicht auf interne Struktur oder Realisierungsdetails ein

UML: Use Case

Zwei Beschreibungsvarianten



Use-Case-Diagramm

- Zeigt externes Verhalten eines Systems mit seinen Akteuren
- Zeigt strukturelle Abhängigkeiten zwischen Anwendungsfällen und Akteuren

Use-Case-Definition

- Natürlichsprachige Beschreibung der Arbeitsabläufe
- D.h.: detailliertere Beschreibung der Interaktionen eines Systems mit seinen Akteuren
- Häufig ergänzt um Zustands- und Aktivitätsdiagramme (s. nächste Kapitel)

UML: Use Case

Beschreibungsvariante: Use-Case-Definition

- **Natürlichsprachige Beschreibung** des Use Cases (s. ELO-Template):
- Auch Use-Case-Beschreibung genannt

Beschreibung Anwendungsfall				
Name				
Kurzbeschreibung				
Akteure				
Auslöser				
Eingehende Daten				
Vorbedingungen				
Nachbedingungen, Ergebnis				
Essentielle Schritte				
Alternativszenarien				
Offene Punkte				
Änderungshistorie	Wann	Wer	Neuer Status	Was
Sonstiges, Anmerkungen				

UML: Use Case

Beschreibungsvariante: Use-Case-Definition

• Beispiel einer Use-Case-Definition (Use-Case-Beschreibung)

Beschreibung Anwendungsfall				
Name	Seminar registrieren			
Kurzbeschreibung	Erfassen eines neuen Seminars und Registrieren in der Seminarverwaltung			
Akteure	Dozent, Sachbearbeiter			
Auslöser	Ein neues Seminar soll abgehalten werden			
Eingehende Daten	Seminarname, Beschreibung, Inhalt, Umfang, Kosten			
Vorbedingungen	Account existiert			
Nachbedingungen, Ergebnis	Das Seminar wurde neu erfasst und abgespeichert			
Essentielle Schritte	<ol style="list-style-type: none"> 1. <<include>>: Berechtigung prüfen 2. Seminardaten erfassen 3. Seminardaten prüfen 4. Seminardaten in Datenbank eintragen 			
Alternativszenarien	Seminar bereits vorhanden <ul style="list-style-type: none"> • System zeigt eine Fehlermeldung an • Benutzer wird zur erneuten Einfabe aufgefordert 			
Offene Punkte	- Keine -			
Änderungshistorie	Wann 02.11.2016	Wer CKE	Neuer Status In Arbeit	Was Alternativszenario
Sonstiges, Anmerkungen	---			

UML: Use Case

Beschreibungsvariante: Use-Case-Definition

Regeln zum Aufbau einer Use-Case-Definition:

- Use-Case zeigt in erster Linie die **Absicht der Akteure** an (nicht das Verhalten des Systems)

– Falsch:

1. Das System fragt nach dem Namen
2. Der Nutzer gibt den Namen ein
3. Das System fragt nach der Adresse
4. Der Nutzer gibt die Adresse ein
5. Der Nutzer klickt auf OK
6. Das System zeigt das Profil des Nutzers

Warum falsch?

– Richtig:

1. Der Nutzer gibt Namen und Adresse ein
2. Das System zeigt das Profil des Nutzers an

UML: Use Case

Beschreibungsvariante: Use-Case-Definition

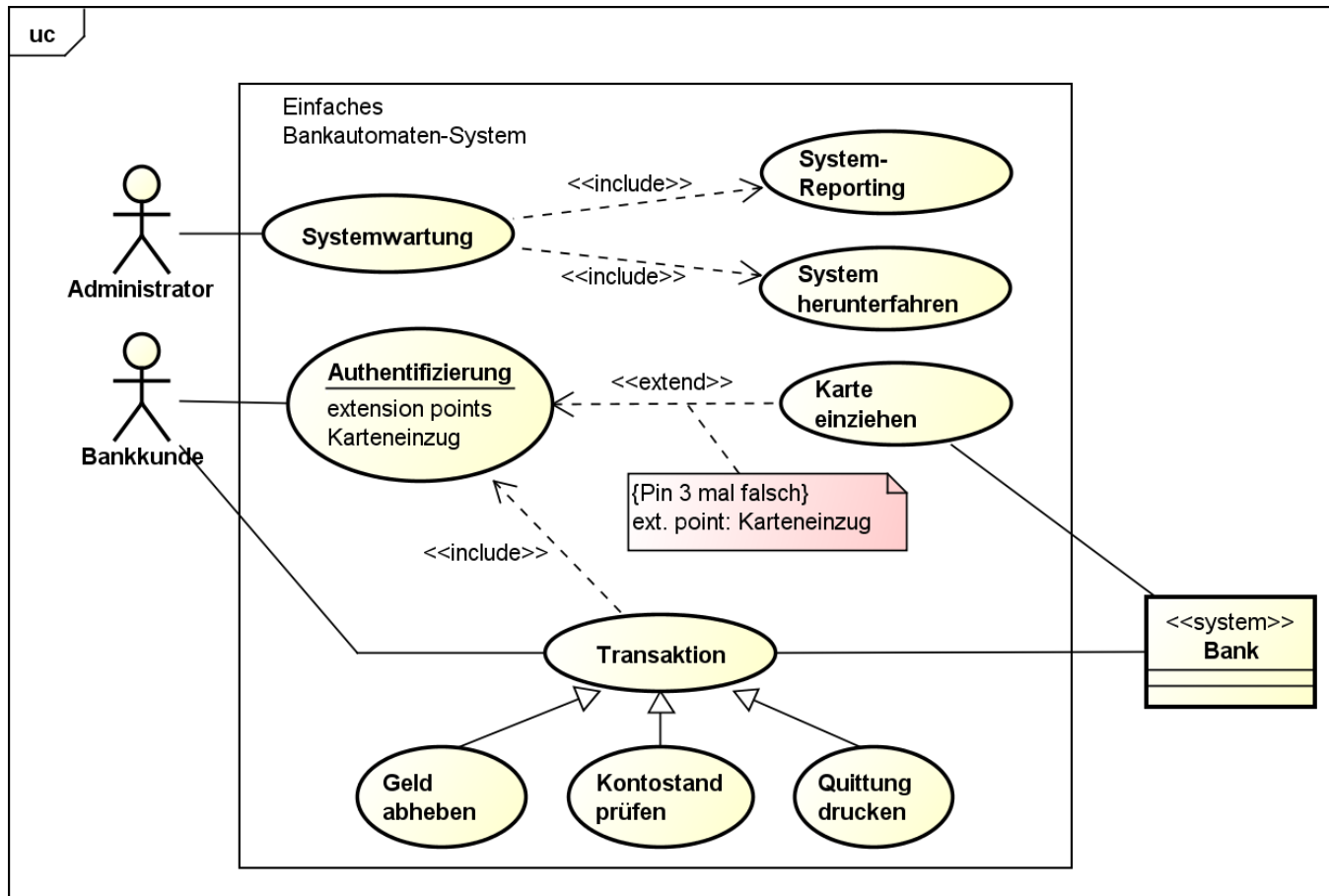
Regeln zum Aufbau einer Use-Case-Definition (continued):

- **Einfache Grammatik** benutzen:
 - Beispiel: Namen und Adresse eingeben (oder: Der Nutzer gibt Namen und Adresse ein)
 - Beispiel: Kontostand aktualisieren (oder: Das System aktualisiert den Kontozustand)
 - Ziel: Klarheit
- **Lesbarkeit beachten:**
 - Zu viele Schritte in einer Use Case-Definition vermeiden (Empfohlen: max. 9 Schritte)
- **Präzise formulieren:**
 - Schlecht: Das System prüft, ob das Passwort richtig ist.
 - Gut: Das System verifiziert, dass das Passwort richtig ist.
- **Falls Wiederholungen notwendig sind:** “Mache Schritte i - j bis Bedingung k erfüllt ist”
 - Beispiel: 1. Der Verbraucher wählt ein Produkt aus
 - 2. Das System fügt das Produkt in den Einkaufswagen
 - 3. Der Verbraucher wiederholt die Schritte 1-2, bis alle gewünschten Produkte gewählt
 - 4. Der Verbraucher zahlt die Produkte im Einkaufswagen


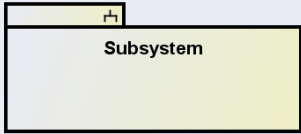


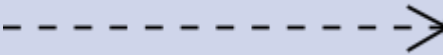
UML: Use Case

Beschreibungsvariante: Use-Case-Diagramm



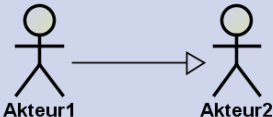
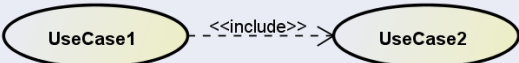
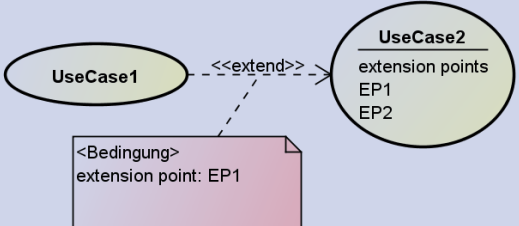
Beschreibt strukturelle Abhängigkeiten zwischen Anwendungsfällen und Akteuren:



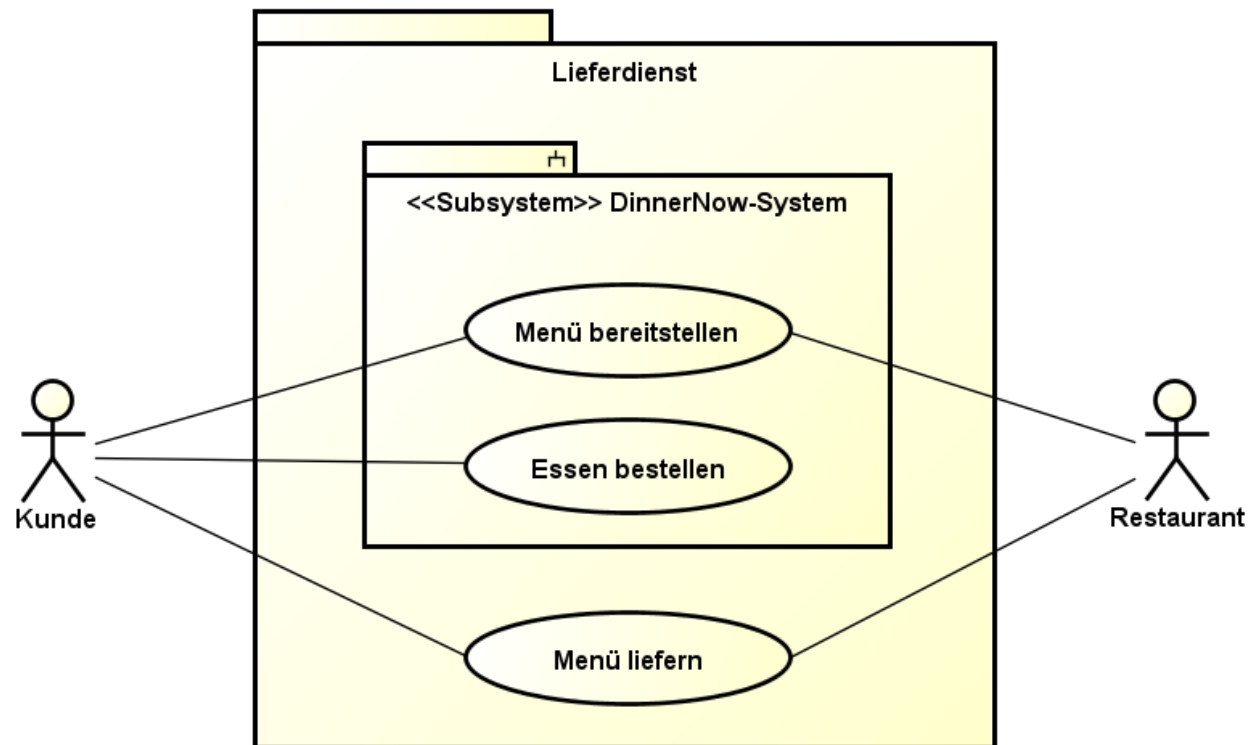
UML: Use-Case-Diagramm

Notation	Name	Bedeutung
	Use-Case	<ul style="list-style-type: none"> - Definiert fachlichen Anwendungsfall - Verhalten wird beschrieben (keine internen Realisierungsdetails) - Detailliertere Darstellung der Aktionen durch andere UML-Diagramme oder im Klartext (Use-Case-Definition) - Wird von einem Akteur gestartet - Führt zu einem fachlichem Ergebnis
	Subsystem	<ul style="list-style-type: none"> - Teil des Systems - Beinhaltet Use Cases, die vom Subsystem unterstützt werden
	Akteur	<ul style="list-style-type: none"> - Externes Objekt oder Person, die mit System interagiert - Use Case wird stets von einem Akteur gestartet
	Akteur	<ul style="list-style-type: none"> - Links: Fremdsystem als Akteur - Rechts: Zeitgesteuertes Ereignis als Akteur
	Abhängigkeit	<ul style="list-style-type: none"> - Aufbau des Quell-Use-Cases hängt vom Aufbau des Ziel-Use-Cases ab

UML: Use-Case-Diagramm

Notation	Name	Bedeutung
	Assoziation (ungerichtet)	<ul style="list-style-type: none"> - Akteur ist an Use Case beteiligt bzw. - Akteur kommuniziert mit dem System - Evtl. mit Multiplizität: wie oft kann Element an Aktion gleichzeitig teilnehmen
	Assoziation (gerichtet)	<ul style="list-style-type: none"> - Beschreibt welcher Teil der Aktive ist - (Beschreibt nicht Richtung des Informationsflusses)
	Generalisierung/ Vererbung	<ul style="list-style-type: none"> - Ziele, Use Cases, etc. der Generalisierung werden von der Spezialisierung geerbt (zwischen Akteuren oder zwischen Use Cases)
	<<include>>- Beziehung	<ul style="list-style-type: none"> - Use Case1 importiert Verhalten eines anderen Use Cases - Beziehung ist nicht optional, Verhalten wird also immer importiert
	<<extend>>- Beziehung	<ul style="list-style-type: none"> - Verhalten von UseCase2 kann durch UseCase1 erweitert werden - Zeitpunkt, an dem Verhalten erweitert werden kann: Extension Point (UseCase darf mehrere EP haben) - Optionale <Bedingung>: wenn true, EP ausführen

Beispiel: Restaurant

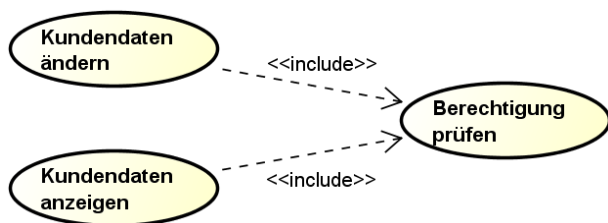




Use-Case-Diagramm: <<include>>

<<include>>

- Visualisiert, dass Use Case A das Verhalten von Use Case B importiert
- Das Verhalten wird immer importiert (Verhalten ist nicht optional)
- Erst die Beschreibung der Aktionen definiert, an welcher Stelle B inkludiert wird



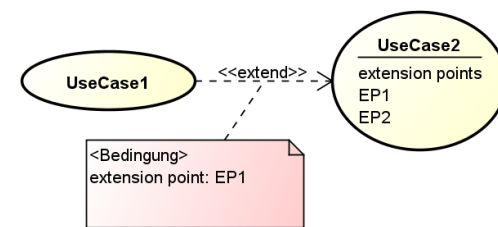
1. Modellieren Sie einen Kalender, in dem Termine erfasst und der Kalender aktualisiert werden kann
2. Modellieren Sie das Kaufen eines Kaugummis an einem Kaugummiautomaten

Beispiel: <<include>>-Beziehung in Use-Case-Definition

Beschreibung Anwendungsfall				
Name	Seminar registrieren			
Kurzbeschreibung	Erfassen eines neuen Seminars und Registrieren in der Seminarverwaltung			
Akteure	Dozent, Sachbearbeiter			
Auslöser	Ein neues Seminar soll abgehalten werden			
Eingehende Daten	Seminarname, Beschreibung, Inhalt, Umfang, Kosten			
Vorbedingungen	Account existiert			
Nachbedingungen, Ergebnis	Das Seminar wurde neu erfasst und abgespeichert			
Essentielle Schritte	<ol style="list-style-type: none"> 1. <<include>>: Berechtigung prüfen 2. Seminardaten erfassen 3. Seminardaten prüfen 4. Seminardaten in Datenbank eintragen 			
Alternativszenarien	Seminar bereits vorhanden <ul style="list-style-type: none"> • System zeigt eine Fehlermeldung an • Benutzer wird zur erneuten Eingabe aufgefordert 			
Offene Punkte	- Keine -			
Änderungshistorie	Wann 02.11.2016	Wer CKE	Neuer Status In Arbeit	Was Alternativszenario
Sonstiges, Anmerkungen	---			

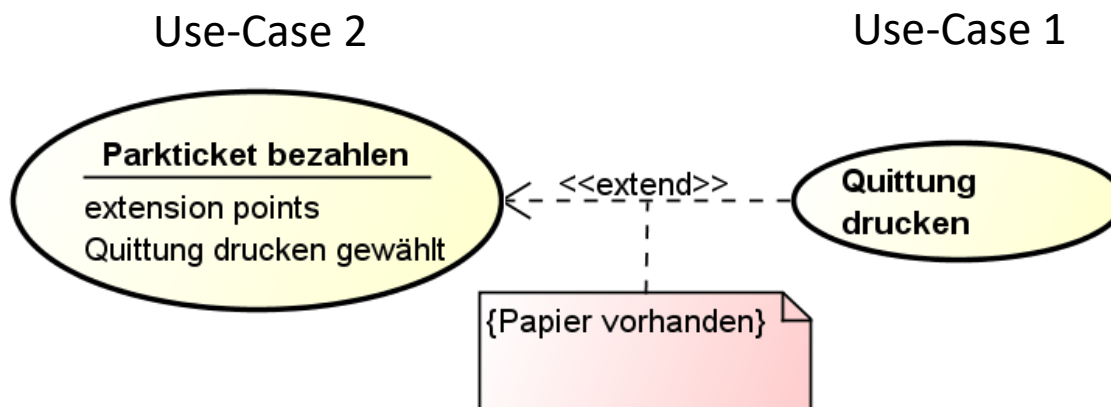
Hier wird ein
neuer Use Case
eingebunden

Use-Case-Diagramm: <<extend>>



<<extend>>

- Verhalten von Use-Case 2 kann durch Use-Case 1 erweitert werden (... muss aber nicht)
- Zeitpunkt, an dem ein Verhalten eines Use-Case erweitert werden kann, bezeichnet man als Erweiterungspunkt (extension point)
- Ein Use Case darf mehrere Erweiterungspunkte besitzen
- Um optionale Bedingung ergänzbar (nur wenn Bedingung wahr ist, wird Erweiterung ausgeführt)



Achtung:
Pfeil weist in die Richtung des
Use Cases, der erweitert werden soll



Use-Case-Diagramm: <<extend>>

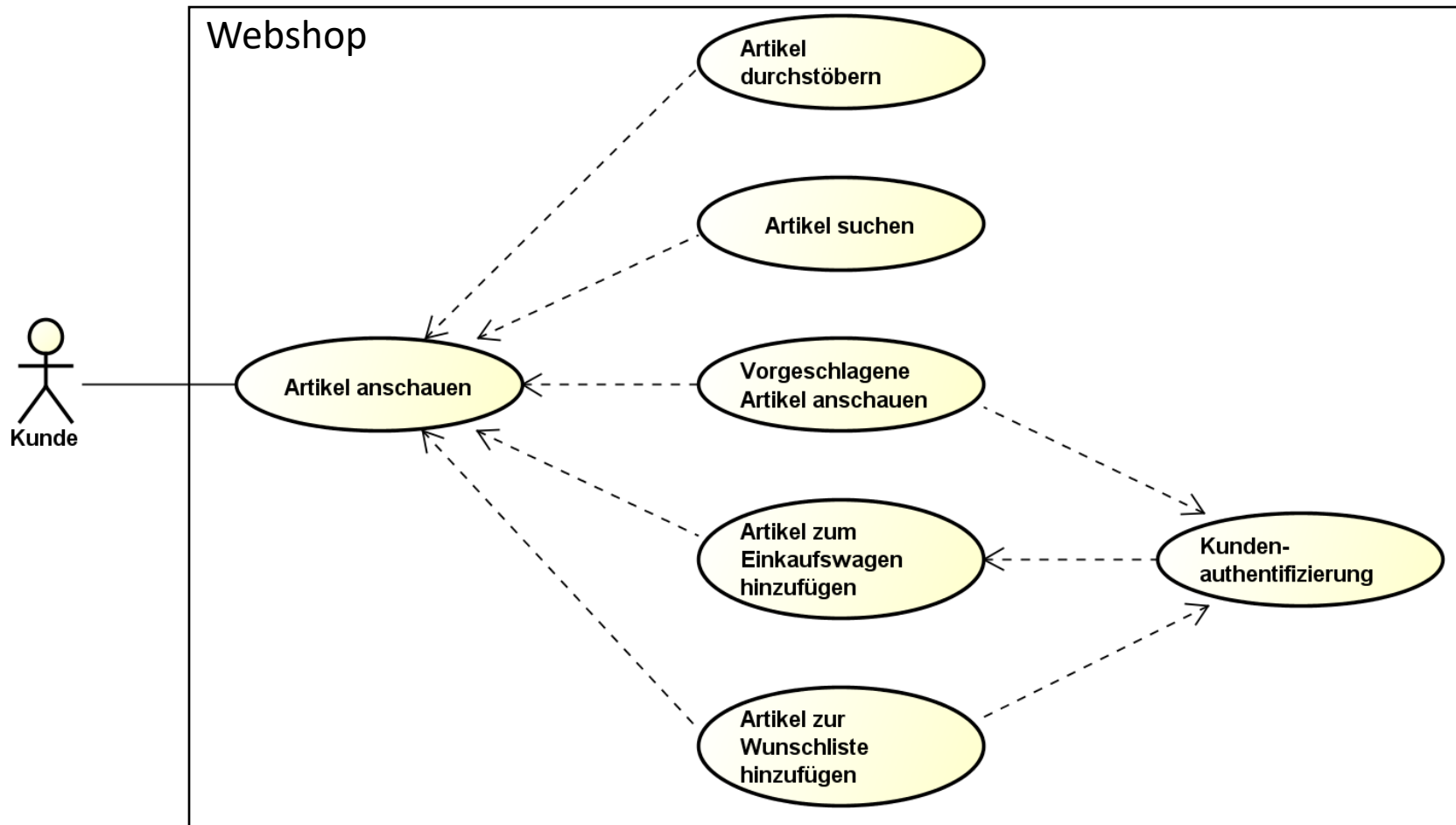
Beispiele unter Verwendung von <<extend>>-Beziehungen:

1. Auf einem Computer kann ein Benutzer DVDs erstellen. Anschließend können die DVDs beschriftet werden. Natürlich können DVDs auch unabhängig vom Brennen beschriftet werden.
2. Sobald ein Kunde sich am Geldautomaten 3 mal falsch angemeldet hat, wird die EC-Karte eingezogen
3. In einem Restaurant isst ein Kunde ein Gericht. Wird Wein zu dem Gericht gereicht, trinkt er diesen natürlich auch. Am Ende zahlt der Kunde beim Kellner sein Gericht und natürlich auch den Wein, falls er welchen getrunken hat.

Use-Case-Diagramm: <<include>>, <<extend>>



4. Vervollständigen Sie folgendes Modell mithilfe von <<include>> und <<extend>>



Beispiel: Mensa-Essensausgabe



Modellieren Sie eine Mensa-Essensausgabe

- Kunden können in der Mensa Essen bestellen.
- Dazu wählen sie ein Menü, sowie dessen Bestandteile und müssen das Essen anschließend bezahlen.
- Zusätzlich haben registrierte Kunden die Möglichkeit, sich vor der Wahl Menüs filtern zu lassen.
- Mögliche Filter sind momentan: ein Filter nach Unverträglichkeiten und ein Filter nach Art der Küche.

→ Erstellen Sie ein Use-Case-Diagramm, das die oben beschriebene Essensausgabe visualisiert

Beispiel: Einweihungsfeier



Modellieren Sie eine Einweihungsfeier als Use-Case-Diagramm und Definition:

- Gäste können tanzen, sich unterhalten, essen und trinken.
- Der Gastgeber, der natürlich auch Gast auf seiner eigenen Party ist, wird am Ende die Party auflösen und die Gäste zur Tür begleiten.
- Falls während der Party der Kühlschrank geplündert ist und die Gäste Essen oder Trinken nachfordern, sorgt der Gastgeber für Nachschub.
- Falls die Party zu ausgelassen wird, kann es passieren, dass die Polizei die Feier abrupt beendet. Der Gastgeber begleitet in diesem Fall natürlich trotzdem noch seine Gäste zur Tür.

Use Case: Anwendungsbereiche

Use Cases:

- Erlauben **Black-Box-Sicht** auf zu entwickelndes System ohne Realisierungsdetails
- Erlauben Abgrenzung des Systems von der Umwelt
- Einsatzgebiete:
 - Funktionale Dienstleistungen des Systems auf einen Blick zeigen
 - System aus Nutzersicht in handhabbare, logische Teile zerlegen
 - Außenschnittstellen und Kommunikationspartner des Systems modellieren
 - Komplexe Systeme auf hohem Abstraktionsniveau darstellen
 - Planbare Einheiten (Inkremente) für Entwicklung bestimmen

Use-Case-Erstellung: Checkliste

1. Akteure ermitteln

2. Use Cases für Standardverarbeitung ermitteln:

- Auf Basis von: Akteuren, Ereignissen (externe Systeme als Akteure), Aufgabenbeschreibungen (Gesamtziele, Top-10-Aufgaben, etc.)

3. Use Cases für Sonderfälle ermitteln:

- <<extends>>-Beziehung verwenden, um Standardfälle zu erweitern

4. Aufteilen komplexer Anwendungsfälle:

- Komplexe Schritte als eigene Use Cases (mit <<include>> einbinden)
- Use Cases mit vielen Sonderfällen aufspalten (für Gemeinsamkeiten <<include>>)
- Umfangreiche Erweiterungen mit <<extend>> spezifizieren

5. Gemeinsamkeiten mit <<include>> wiederverwenden

[Auszug aus: Heide Balzert, Lehrbuch der Objektmodellierung]

Use-Case-Qualitätskriterien

- **Was einen guten Use Case ausmacht:**

- Für Auftraggeber verständlich
- Beschreibt nur extern wahrnehmbares Verhalten
- Beschreibt Arbeitsablauf aus fachlicher Sicht
- Beschreibt Standardfall vollständig und Sonderfälle separat

- **Was ein gutes Use-Case-Diagramm ausmacht:**

- Besteht aus 3 bis 15 Use Cases (ansonsten weiter unterteilen)

- **Welche Fehlerquellen gibt es:**

- Use Cases zu feingranular bzw. Beschreibung zu detailliert
- Verwechslung von <<include>>, <<extend>> und Generalisierung
- Zu frühe Betrachtung von Sonderfällen
- Use Cases beschreiben (Dialog-)Abläufe

Use Case vs. Szenario

Use Case (Anwendungsfall):

- beschreibt abstrakte Interaktion zwischen Akteuren und Systemen

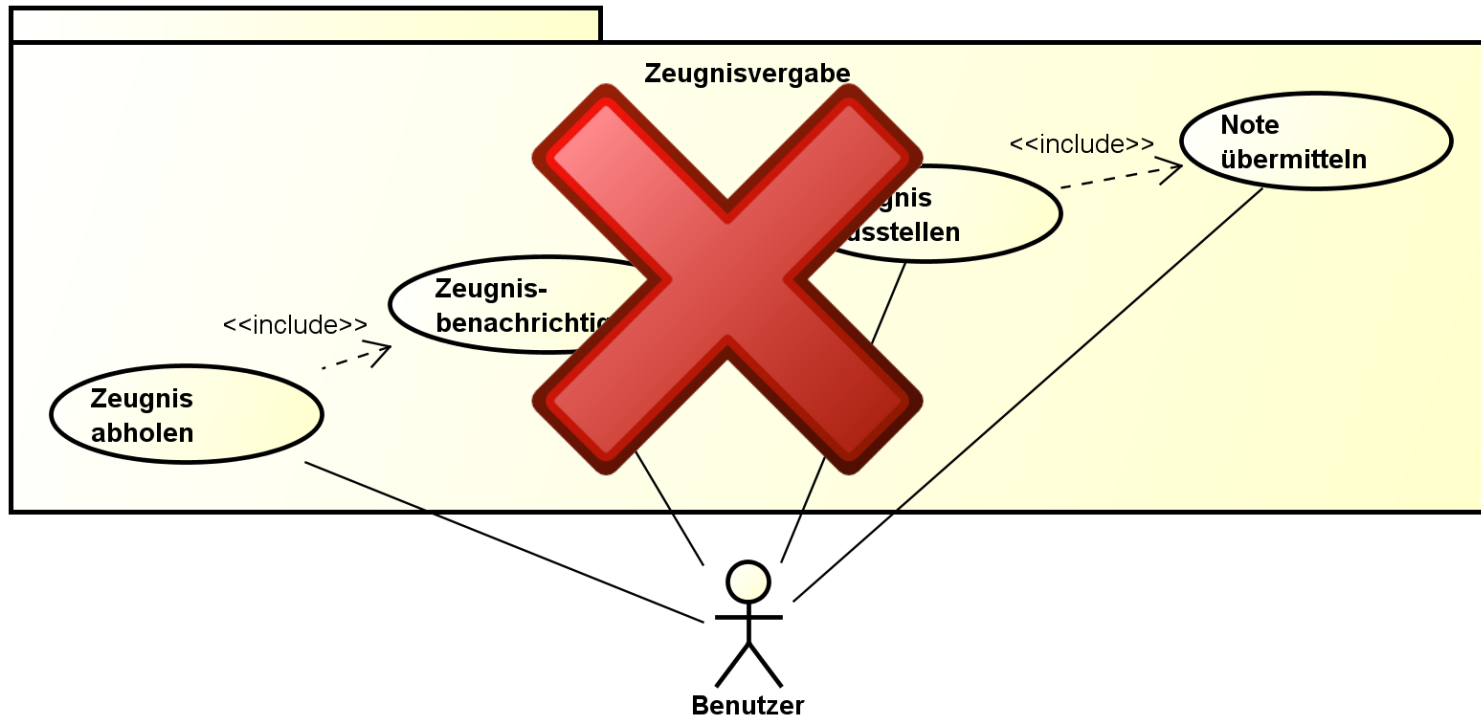
Szenario:

- ist eine konkrete Ausprägung eines Use Cases (also ein möglicher Ablauf evtl. mit konkreten Werten)
- ist eine Instanz eines Use Cases

Zusammengefasst:

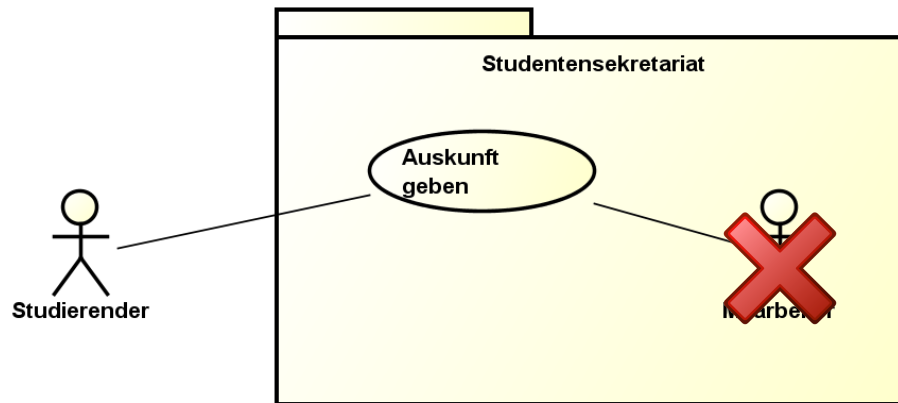
- Szenarios werden durch Sequenzdiagramme oder textuell beschrieben
- Zu einem Use Case existieren i.d.R. mehrere Szenarios

Use Cases: Typische Modellierungsfehler (1)



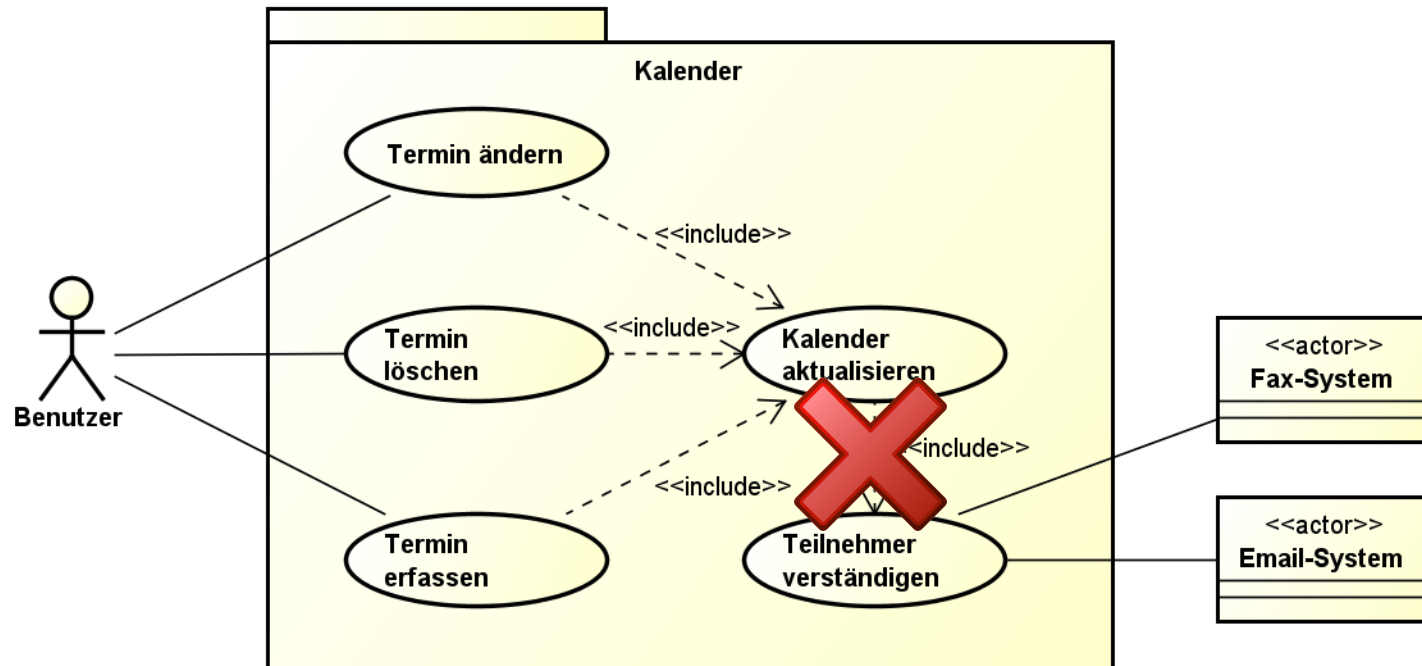
- Use-Case-Diagramme modellieren **keine Abläufe!**

Use Cases: Typische Modellierungsfehler (2)



- Akteure stehen immer außerhalb der Systemgrenzen!

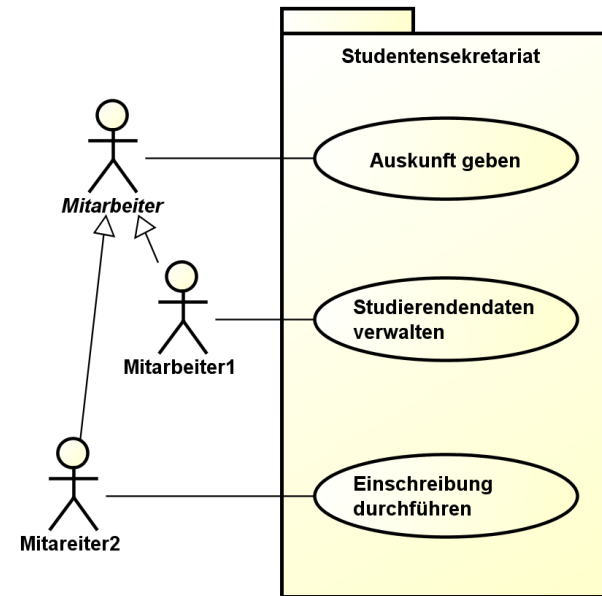
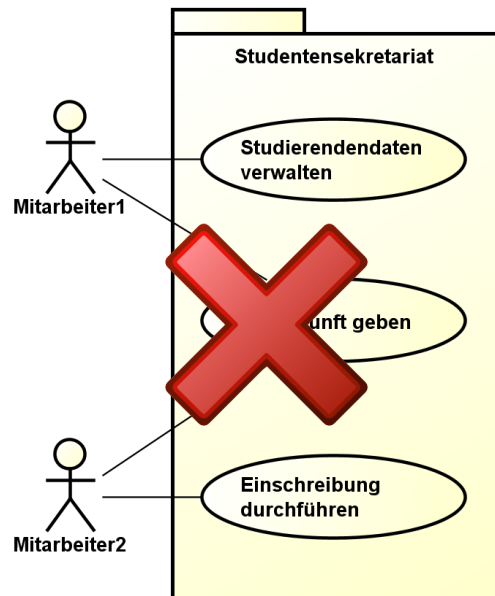
Use Cases: Typische Modellierungsfehler (3)



- “Teilnehmer verständigen” ist nicht Teil von “Kalender aktualisieren”

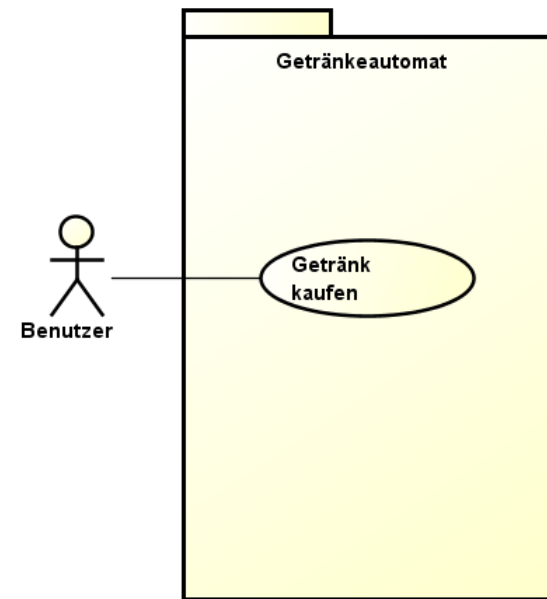
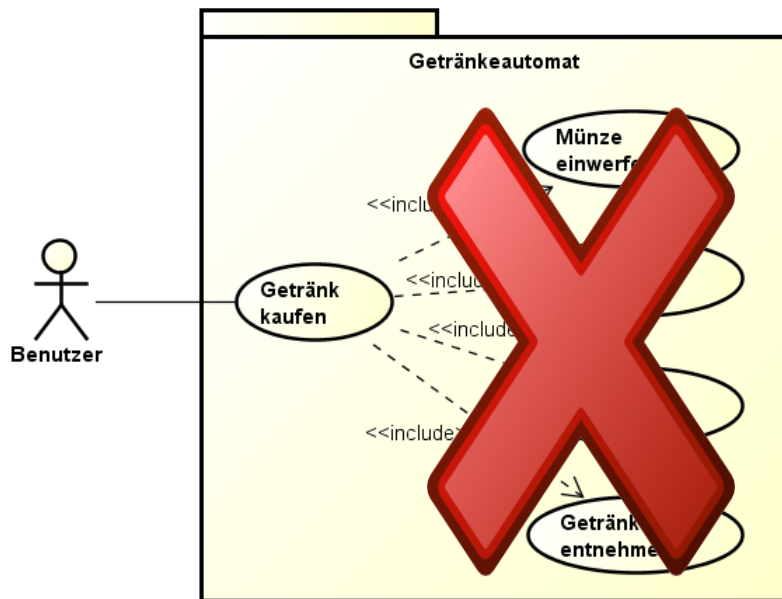
Use Cases: Typische Modellierungsfehler (4)

- Wir wollen ausdrücken, dass entweder Mitarbeiter1 oder Mitarbeiter2 den Anwendungsfall “Auskunft geben” ausführen. Was ist das Problem?



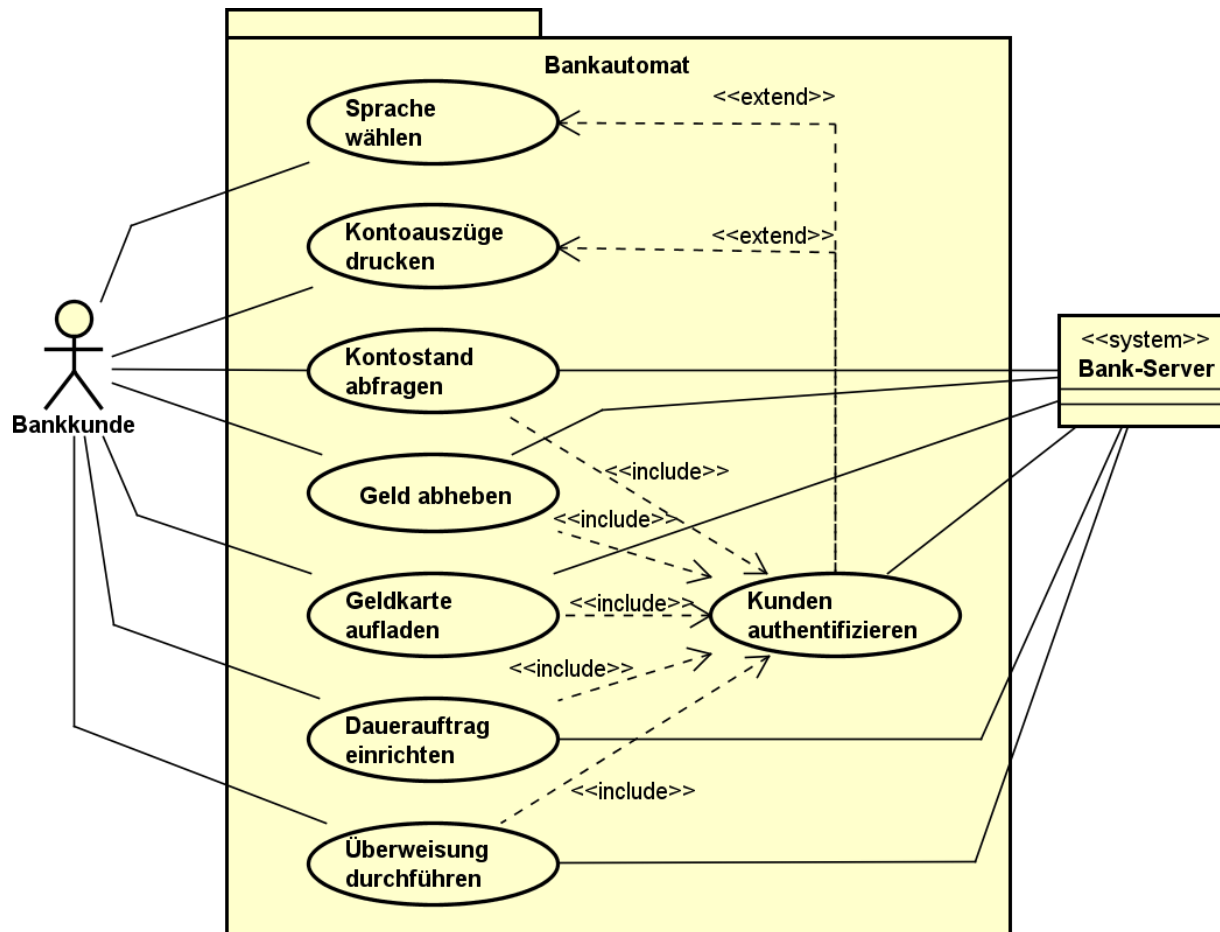
- Links würden beide Mitarbeiter den Use Case “Auskunft geben” gemeinsam ausführen

Use Cases: Typische Modellierungsfehler (5)



- Nicht alle Einzelschritte als Use Cases modellieren (s. Kaugummiautomatbeispiel)
- Wichtig ist: Anwendungsfall soll für den Nutzer messbaren Nutzen erzeugen

Beispiel für ein gutes Use-Case-Diagramm



Literatur

- *UML 2 glasklar*, Chris Rupp et al., Hanser, 2012
- *Lehrbuch der Objektmodellierung*, Heide Balzert, Spektrum Akademischer Verlag, 2011