

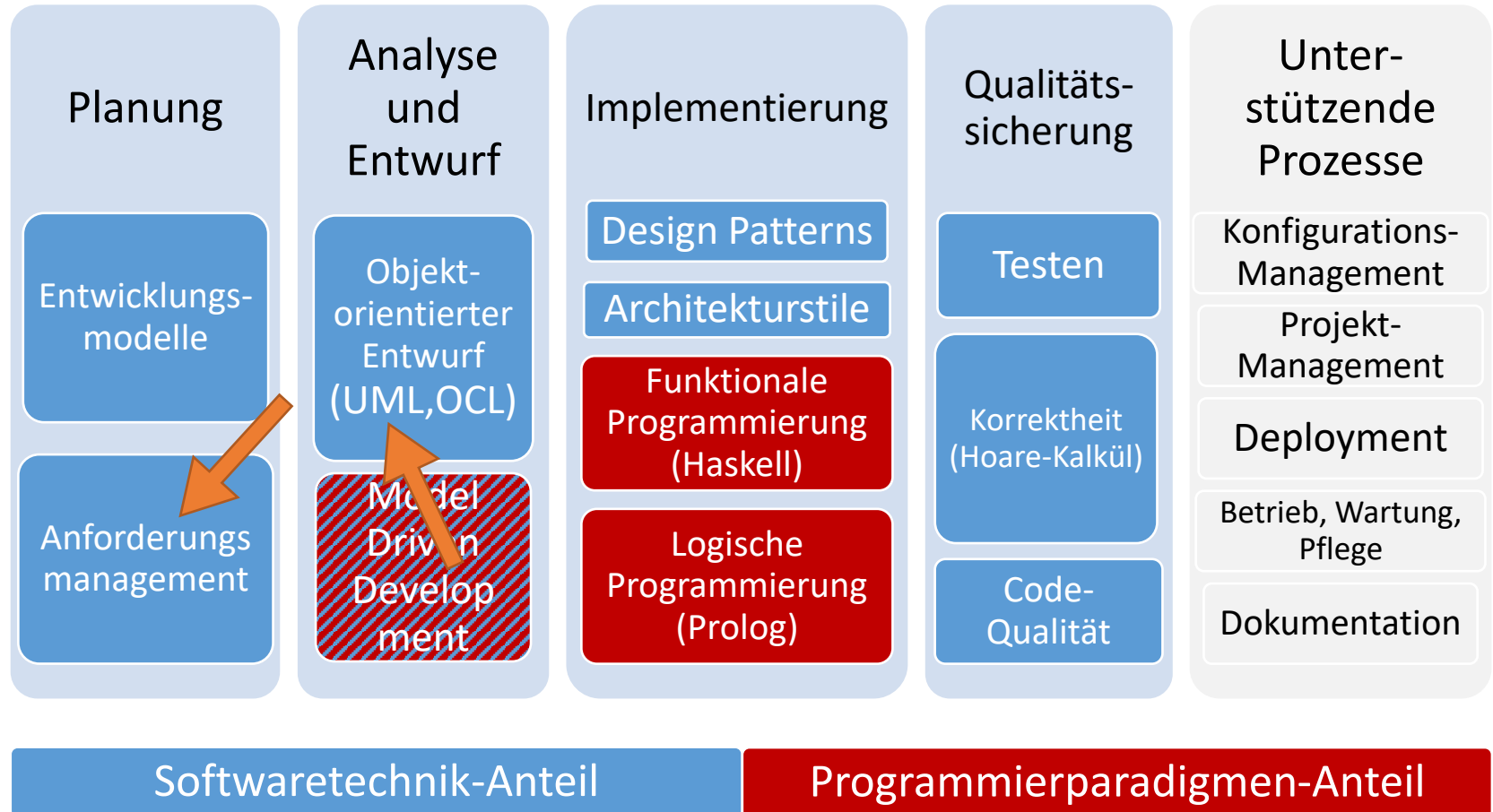
# Softwaretechnik und Programmierparadigmen

## 07 Requirements Engineering

---

Prof. Dr. Sabine Glesner  
Software and Embedded Systems Engineering  
Technische Universität Berlin

# Diese VL



# Inhalt

## Requirements Engineering

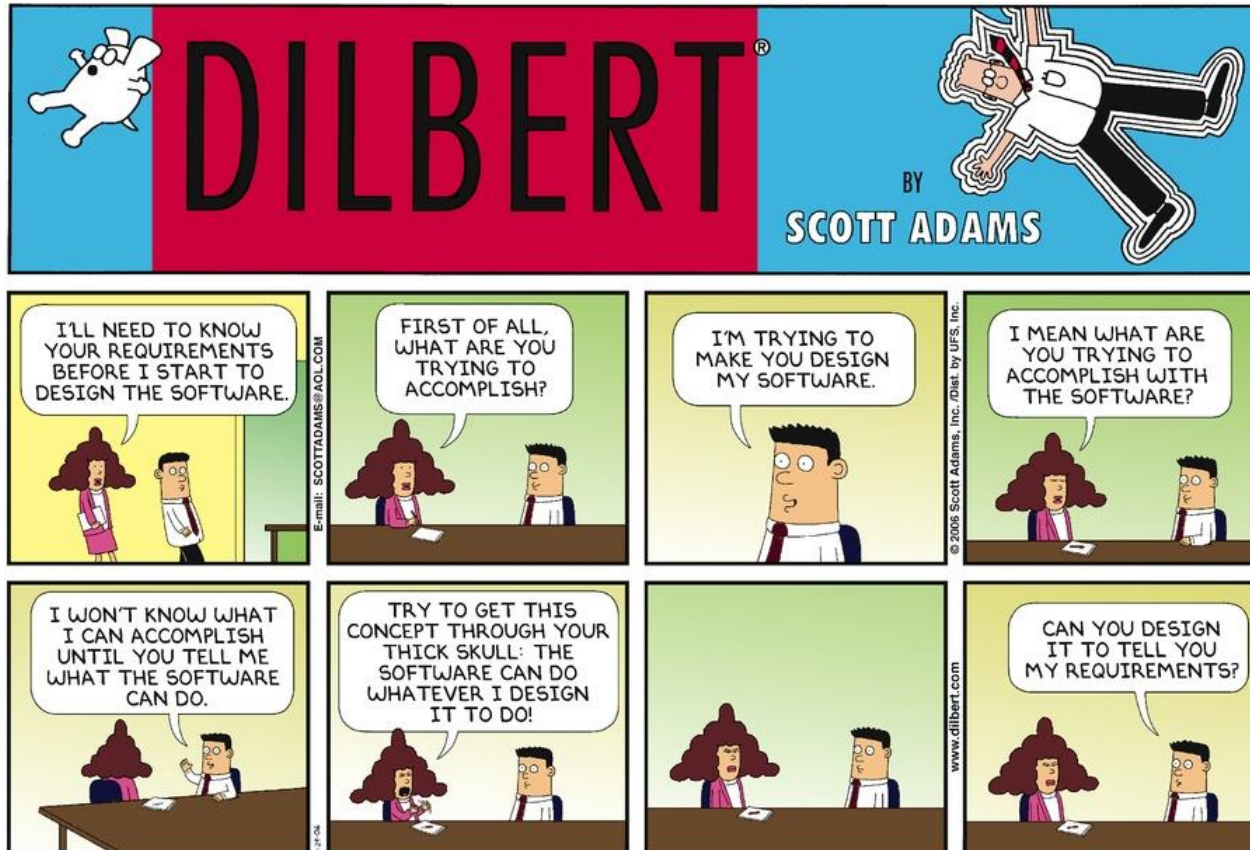
- Grundlagen
- Textuelle Anforderungsspezifikation
- Grafische Anforderungsspezifikation
- Nicht-Funktionale Anforderungen

# Inhalt

## Requirements Engineering

- Grundlagen
- Textuelle Anforderungsspezifikation
- Grafische Anforderungsspezifikation
- Nicht-Funktionale Anforderungen

# Motivation



[License covered by the Classroom Usage Statement](#)

# Requirements Engineering

Der passende Requirements Engineering Prozess hängt von den spezifischen Gegebenheiten ab, besteht aber meist aus vier Elementen

## Ermittlung der Anforderungen

Entwickler:innen und Kunden/Kundinnen bestimmen gemeinsam die Entwicklungsziele

## Spezifikation der Anforderungen

Formale oder informale Dokumentierung von Anforderungen

## Validierung der Anforderungen

Überprüfung ob die Anforderungen das gewünschte System definieren

## Dokumentation der Anforderungen

Zusammenfassung der Ergebnisse im Systemanforderungsdokument

Ermittlung

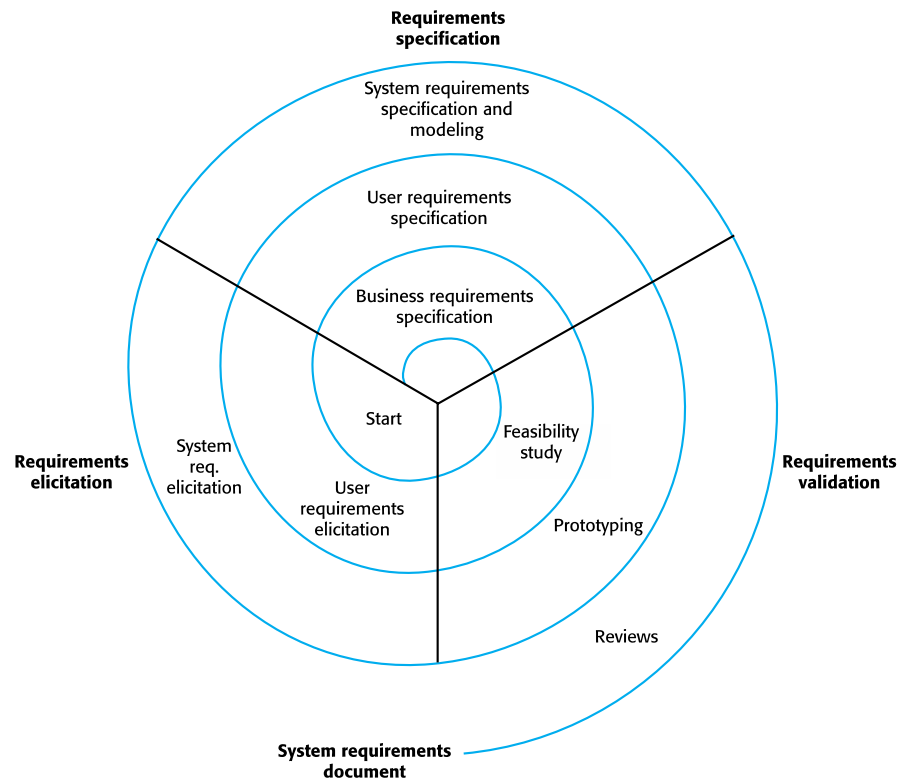
Spezifikation

Validierung

Dokumentation

# Iteratives Requirements Engineering

Häufig wird der Prozess iterativ durchlaufen



[Ian Sommerville, Software-Engineering, Chapter 4](#)

# Anforderungserhebung und -analyse

Umfasst die Ermittlung und Spezifikation der Anforderungen

## *Sammeln von Anforderungen*

Ermittlung der Anforderungen aller Projektbeteiligten

## *Klassifizierung und Organisation der Anforderungen*

Gruppieren von Anforderungen, Identifizierung von Subsystemen

## *Priorisierung der Anforderungen und Auflösung von Konflikten*

Meist Treffen der Projektbeteiligten, um Kompromisse auszuarbeiten

## *Spezifikation der Anforderungen*

Formale oder informale Dokumentierung von Anforderungen (s.o.)

Ermittlung

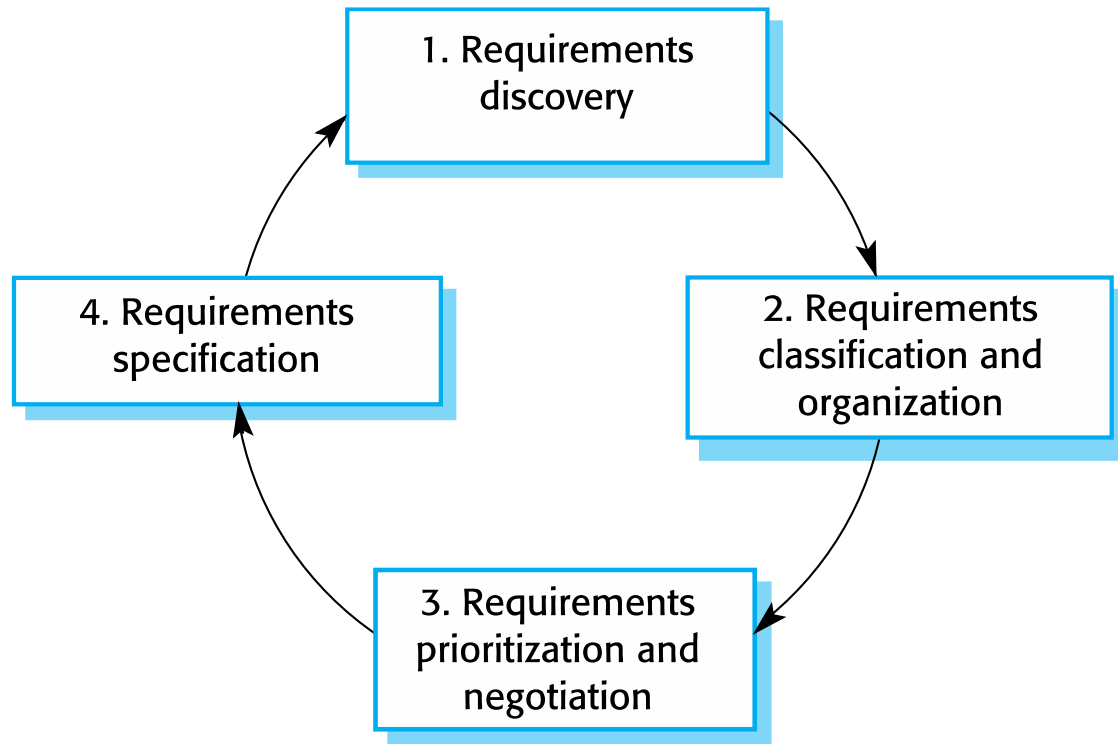
Spezifikation

Validierung

Dokumentation



# Anforderungserhebung und -analyse



[Ian Sommerville, Software-Engineering, Chapter 4](#)

Ermittlung

Spezifikation

Validierung

Dokumentation

# Validierung der Anforderungen

Vermeidung von hohen Kosten durch **späte Änderungen** der Anforderungen

Prüfkriterien

*Gültigkeit, Konsistenz, Vollständigkeit, Realisierbarkeit, Verifizierbarkeit*

Techniken zur Validierung:

*Anforderungsreviews: Systematische Analyse durch Gutachten*

*Prototypen: Experimente an Modell durch*

*Endbenutzer:innen und Kunden/Kundinnen*

*Testfallerzeugung: Offenbart Probleme bei der Erzeugung von Testfällen*

Ermittlung

Spezifikation

Validierung

Dokumentation

# Dokumentation der Anforderungen

Besteht im Wesentlichen aus...

**Lastenheft** (aka. C-Requirements aka. Customer Requirements aka. User Requirements)

- Alle Anforderungen, die Benutzer:innen an das System als Blackbox stellen
- Anforderungen aus Sicht der Kunden/Kundinnen bzw. Endanwender:innen
- „Was soll die Software können?“

**Pflichtenheft** (aka. D-Requirements aka. Development Requirements aka. System Requirements)

- Aus dem Lastenheft abgeleitete Anforderungen an das System
- Anforderungen aus Sicht der/des Auftragnehmenden
- „In welchem Umfang und unter welchen Bedingungen wird die Software eingesetzt?“

Ermittlung

Spezifikation

Validierung

Dokumentation

# Inhalt

## Requirements Engineering

- Grundlagen
- Textuelle Anforderungsspezifikation
- Grafische Anforderungsspezifikation
- Nicht-Funktionale Anforderungen

# Textuelle Anforderungsspezifikation



[License covered by the Classroom Usage Statement](#)

# Textuelle Anforderungsspezifikation

## 1. Spezifikation in natürlicher Sprache

- *Ausführliche* textuelle Beschreibung der Anforderungen an ein System bzw. einer Funktionalität
- **Problem:** Häufig entsteht viel Interpretationsspielraum, da natürliche Sprache nicht eindeutig ist

## 2. Strukturierte Spezifikation

- Übersichtlichere Erfassung von Anforderungen im Vergleich zu natürlicher Sprache
- Tabellarische Erfassung von Anforderungen mit einheitlichen Eckdaten wie z.B.

Funktion	Beschreibung	Inputs	Outputs	Aktion	Pre-Condition	Post-Condition
----------	--------------	--------	---------	--------	---------------	----------------

## 3. Mathematische Spezifikation

- Nutzen eines mathematischen Formalismus zur Beschreibung der Anforderung
- **Formalismen:** Formale Logiken, Automaten, (Object) Z, OCL, Prozesskalküle, ...



Spätere VL

# Spezifikation in natürlicher Sprache

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

**Findet funktionale Anforderungen**

# Spezifikation in natürlicher Sprache

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte ① in einen Warenkorb zu legen und ② diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. ③ Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

④ Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

⑤ Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.



# Funktionale Anforderungen

- ① Produkte in einen Warenkorb legen
- ② Angelegten Warenkorb bezahlen
- ③ Bezahlung überprüfen
- ④ Nutzer verwalten
- ⑤ Produkte suchen

# Alternative mit mehr Details

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte ① in einen Warenkorb zu legen und ② diesen zu bezahlen. Als Bezahlmethoden sind zunächst ②a Bankeinzug und ②b Kredit-kartenzahlung vorgesehen. ③ Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

④ Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl ④a Kunden und Kundinnen als auch ④b Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

⑤ Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

# Alternative mit mehr Details

- ① Produkte in einen Warenkorb legen
- ②a Angelegten Warenkorb **per Bankeinzug** bezahlen
- ②b Angelegten Warenkorb **per Kreditkarte** bezahlen
- ③ Bezahlung überprüfen
- ④a **Mitarbeitende** registrieren
- ④b **Kunde/Kundin** registrieren
- ⑤ Produkte suchen

# Strukturierte Spezifikation

Informelle, **tabellarische** Darstellung nach einheitlichem Schema

- **Übersichtlicher** als reiner Text
- unterstützt **Konsistenz** und **Vollständigkeit**

Typische Felder:

- **Name, Beschreibung** (verbal)
- **Inputs, Outputs**: Datenaustausch mit der Komponente
- **Pre**: Nötige Vorbedingungen für die Ausführung
- **Post**: Nachbedingung – Zustand nach Ausführung; Zielbeschreibung
- **Aktion**: Ausgeführte Aktionen, auch Details/Zwischenschritte mgl.

# Strukturierte Spezifikation

Funktion	Beschreibung	Inputs	Outputs	Aktion	Pre	Post
Produkte in einen Warenkorb legen	Eine beliebige Anzahl eines Produkts wird von Kunden/Kundinnen ausgewählt und dem Warenkorb hinzugefügt	Produkt, Anzahl	Erfolgsnachricht	Warenkorb aktualisieren	Anzahl größer 0	Anzahl des Produkts im Warenkorb um Anzahl erhöht
Angelegten Warenkorb bezahlen	Inhalt des Warenkorbs wird bestellt. Zahlungsmethode „EC“ oder „Kredit“ wird zur Auswahl gestellt	Warenkorb	Erfolgsnachricht	Bank-Transaktion initiieren, Versand vorbereiten	Kunde/Kundin registriert, Bankdaten verifiziert	Warenkorb geleert, Bestelldaten hinterlegt

# User Stories

Agile Methoden gehen von **häufigen Anforderungsänderungen** aus

- Detaillierte/vollständige Dokumentation vorab nicht möglich (oder Zeitverschwendung)
- Anforderungen werden inkrementell entsprechend dem Entwicklungsprozess ermittelt

Passendes Format: **User Stories**

- **Kurze** Beschreibung einer einzelnen Anforderung
- 1-2 Sätze (**natürlichsprachlich**)
- Aus Sicht von Benutzer:innen (**Rolle**) geschrieben
- Konzentriert sich auf das **Ergebnis** (“was brauchen Benutzer:innen”, statt “was sollte das System (wie) liefern”)

# User Stories

Je nach Vereinbarung gibt es ein einheitliches Format, z.B.:

As a (role) I want (something) so that (benefit). [Quelle:2]

Beispielsweise:

- User Stories werden auf Karten notiert (bzw. möglichst einfaches Tool)
- Auf der Rückseite werden Kriterien für die Validierung notiert
- Weitere Infos: Priorität (kann, sollte, muss), Aufwandsabschätzung

#123
Als Kundin/Kunde möchte ich die Produkte im Warenkorb bezahlen, damit sie mir zugesendet werden.

#1
Als Student:in möchte ich eine Prüfung anmelden.
Prio: muss
Aufwand (in TU-SAP): 5000

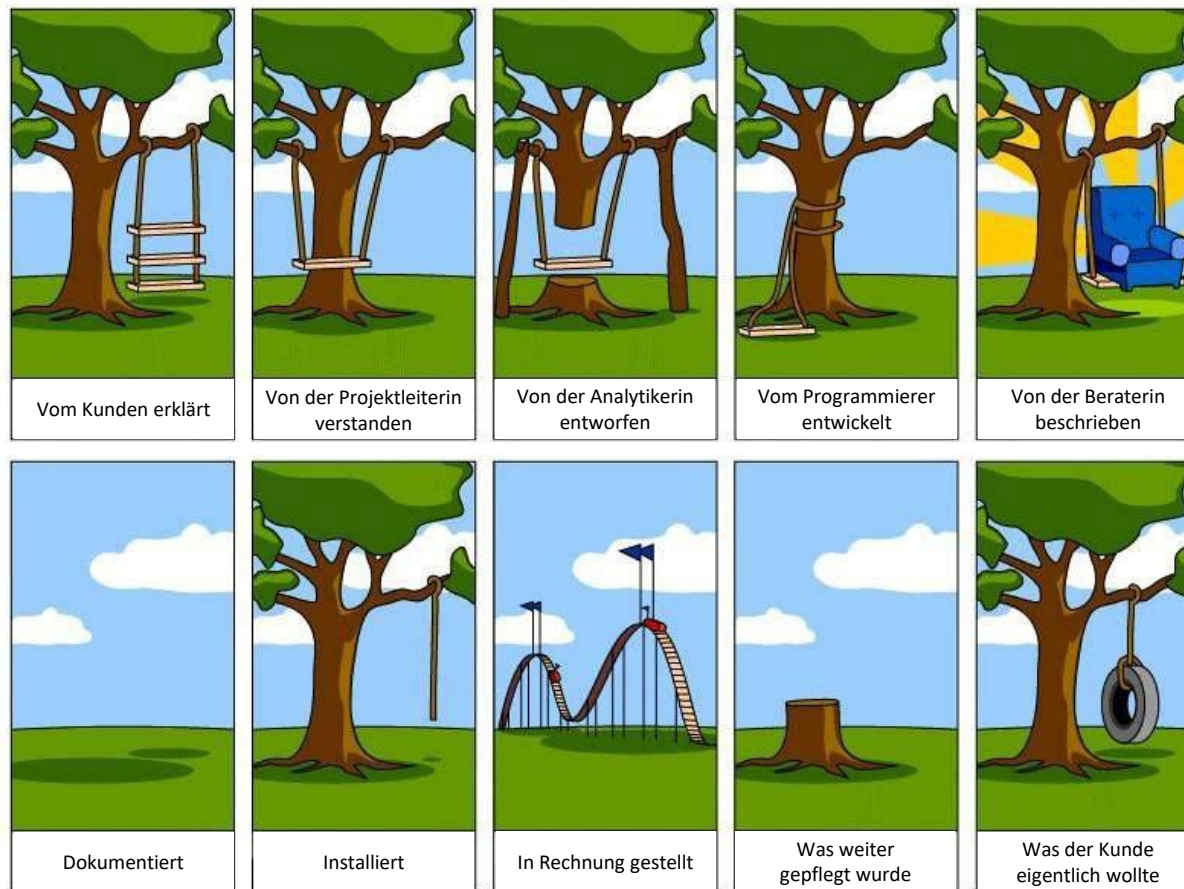
# Inhalt

## Requirements Engineering

- Grundlagen
- Textuelle Anforderungsspezifikation
- **Grafische Anforderungsspezifikation**
- Nicht-Funktionale Anforderungen



# Grafische Anforderungsspezifikation



# UML

Die **Unified Modeling Language (UML)** ist eine **visuelle Sprache** zur Spezifikation, Konstruktion und Dokumentation technischer Systeme

- Erste Ansätze Mitte der 90er von Grady Booch, James Rumbaugh und Ivar Jacobson (Rational Software Corporation)
- Ziel: „Unified Method“ verschiedener objektorientierter Modellierungsmethoden
- als einheitliche Modellierungssprache seit 1997 von der OMG (Object Management Group) standardisiert
- Aktuelle Version (seit Juni 2015): UML 2.5
- Spezifikation: <http://www.omg.org/spec/UML/>

# UML Spezifikation

## UML Infrastructure

- Definiert den Sprachkern der UML (z.B. Konzepte wie Klasse, Assoziation, Attribut und Methode)
- Erweiterbar durch Erweiterungsmechanismen auf Nutzer:innen-Ebene und Profile

## UML Superstructure

- Erweitert den Sprachkern auf den vollständigen UML-Sprachumfang
- Definiert Modellelemente, Notationen, Diagrammtypen
- Definiert welche Eigenschaften Sprachelemente haben dürfen und welche Beziehungen zulässig sind

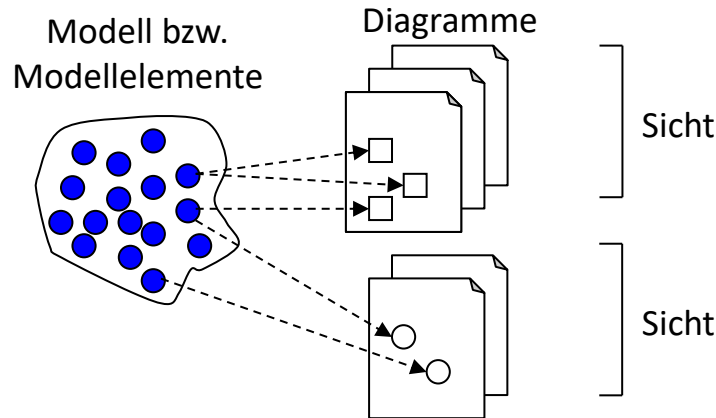
## UML Object Constraint Language (OCL)

- Zur Spezifikation von Invarianten und Bedingungen
- Metamodell-basierte Definition
- Konsistent zum UML-Metamodell

# UML Diagramme

## UML definiert eine Menge von Diagrammtypen

- Mehrere Diagramme können gemeinsam eine Sicht auf ein UML-Modell definieren

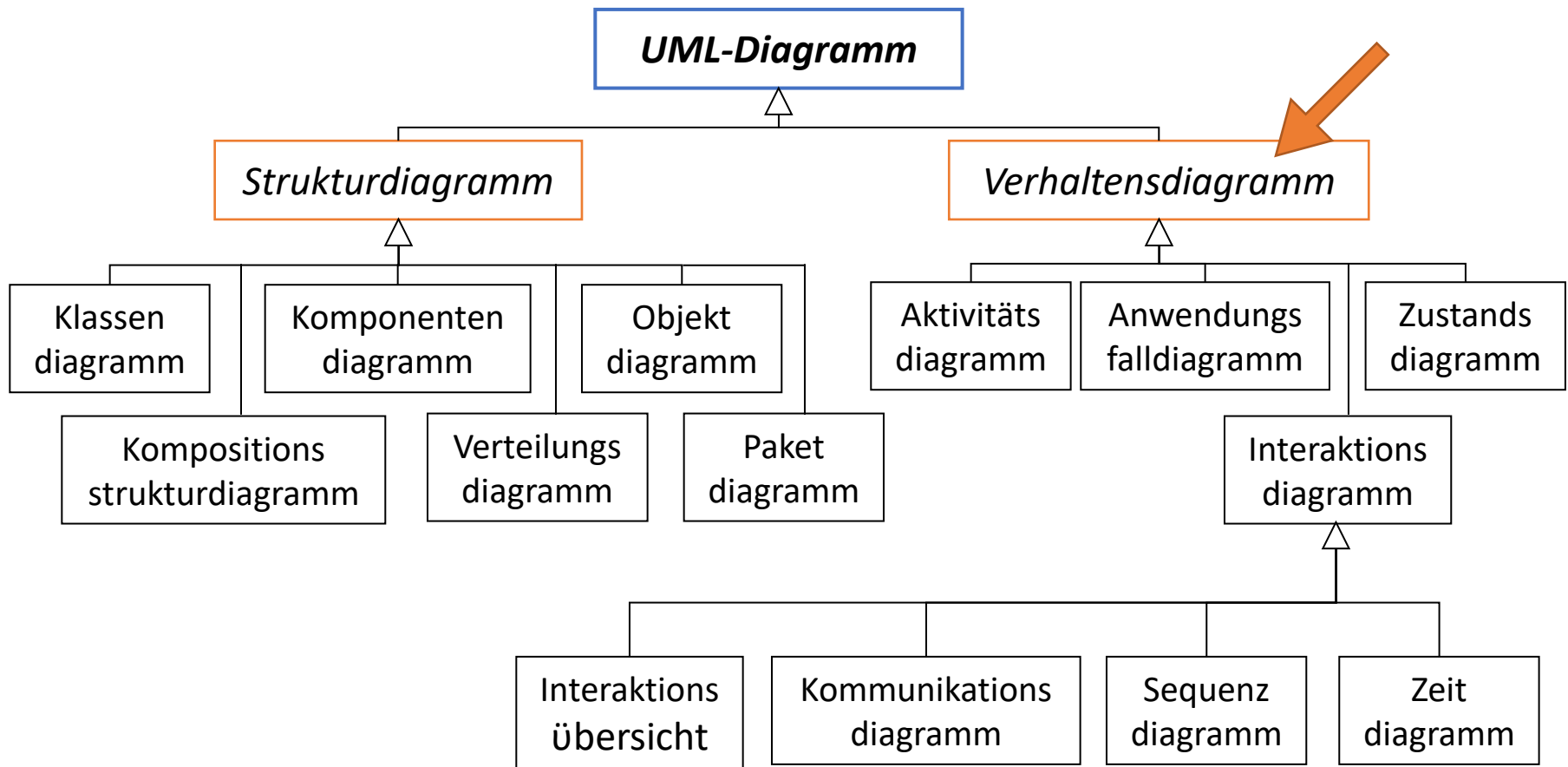


## UML-Modelle haben keinen Vollständigkeitsanspruch

- Dass bestimmte Modellteile nicht aufgeführt werden heißt nicht, dass sie nicht da sind

Modelle können schrittweise erweitert und gemischt werden

# UML Diagrammübersicht



# Verhaltensmodellierung mit UML

Verhaltensmodellierung betrifft **dynamische** Aspekte des Systems

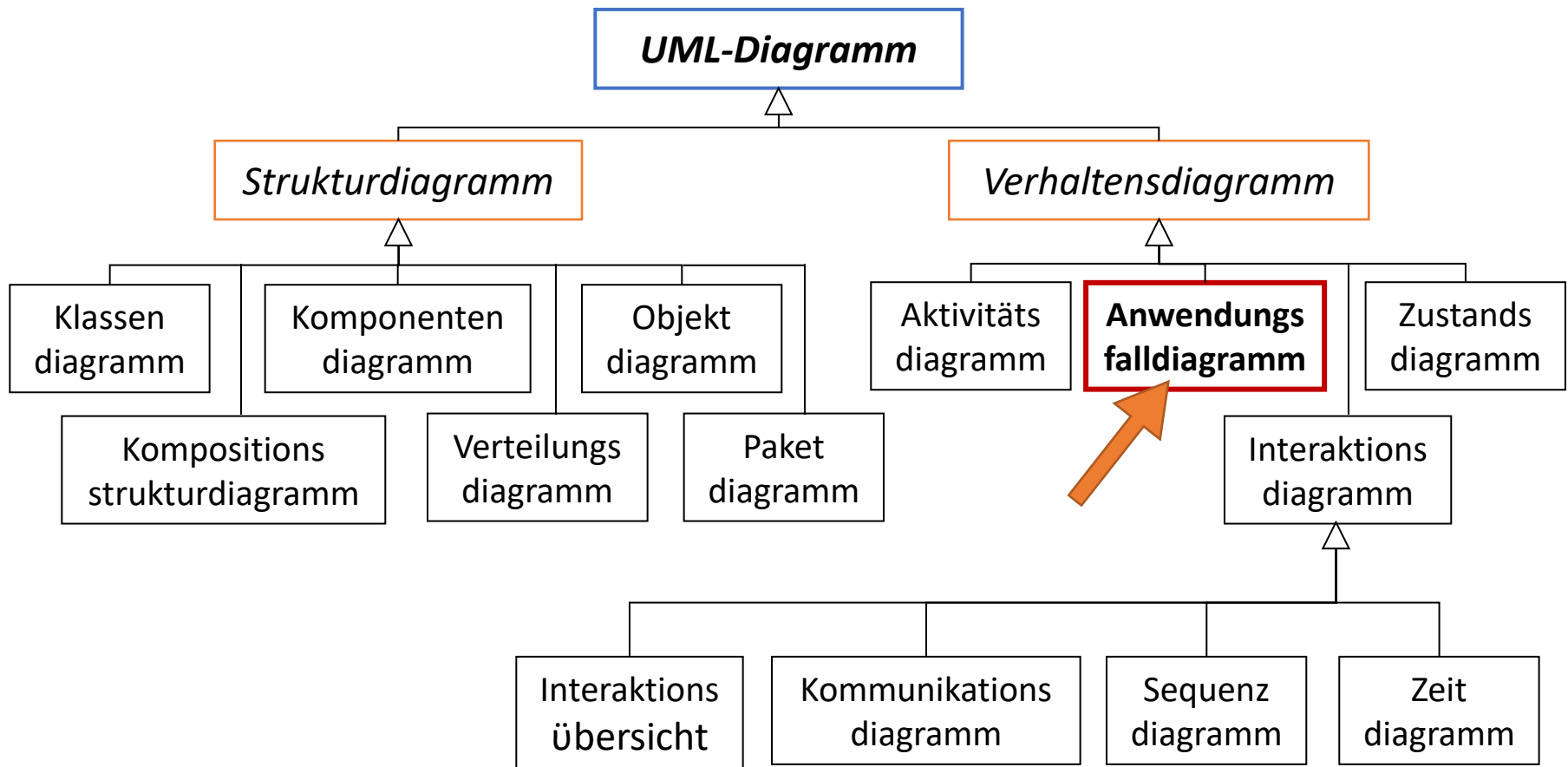
Verhalten ist beobachtbar als Veränderungen...

- ... der Eigenschaften der beteiligten Elemente (Zustandsänderungen)
- ... der Struktur des Gesamtsystems

Grundformen der Verhaltensbeschreibung zur Unterstützung verschiedener Sichten auf das Verhalten

- **Anwendungsfälle (Use Cases)**
- Zustandsautomaten
- Aktivitäten
- Interaktionen

# UML Diagrammübersicht



# Use cases

Spezifikation eines fachlichen Ziels von Akteur:in (Anwendungsfall)

- **Akteur:in** (Rolle) wird identifiziert
- Use Case wird benannt und ggfs. genauer spezifiziert
- **Wesentliche Spezial- und Fehlerfälle** werden mit aufgeführt

Darstellung:

- **UML Use-Case-Diagramm** (enthält mehrere Use-Cases, Details später)
- Detaillierte Dokumentation von Use-Cases z.B. **durch strukturierte Spezifikation**
- Abläufe durch weitere Diagramme dokumentiert (z.B. **Sequenzdiagramm**)

Nach dem Requirements Engineering:

- alle möglichen Interaktionen mit dem System als Use-Cases dokumentiert



# Use-Case (Anwendungsfall-) Diagramm

## **Grafische** Erfassung von Akteuren/Akteurinnen und Anwendungsfällen

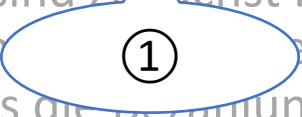
### *Modellierungselemente*

- Akteur:in
- Anwendungsfälle
  - Akteur:in versucht mit dem System ein fachliches Ziel zu erreichen
  - Kann mehrere Ablauf-Szenarien zusammenfassen (Bsp. Erfolg/Misserfolg)
- Beziehungen

### Use-Case Diagramm möglichst einfach halten

- Konzentration auf sichtbares Verhalten
- Von Akteur:in angestoßen
- Darstellung von Details/Abläufen nicht im Use-Case-Diagramm!
- Grundlage für detailliertere Verhaltensdiagramme

# Fallbeispiel (Auszug)

Sie werden gebeten für einen kleinen Unternehmen Ruhe und Kleidung verkauft, die Verwaltungssoftware eine Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen.  gestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Akteur:in

②

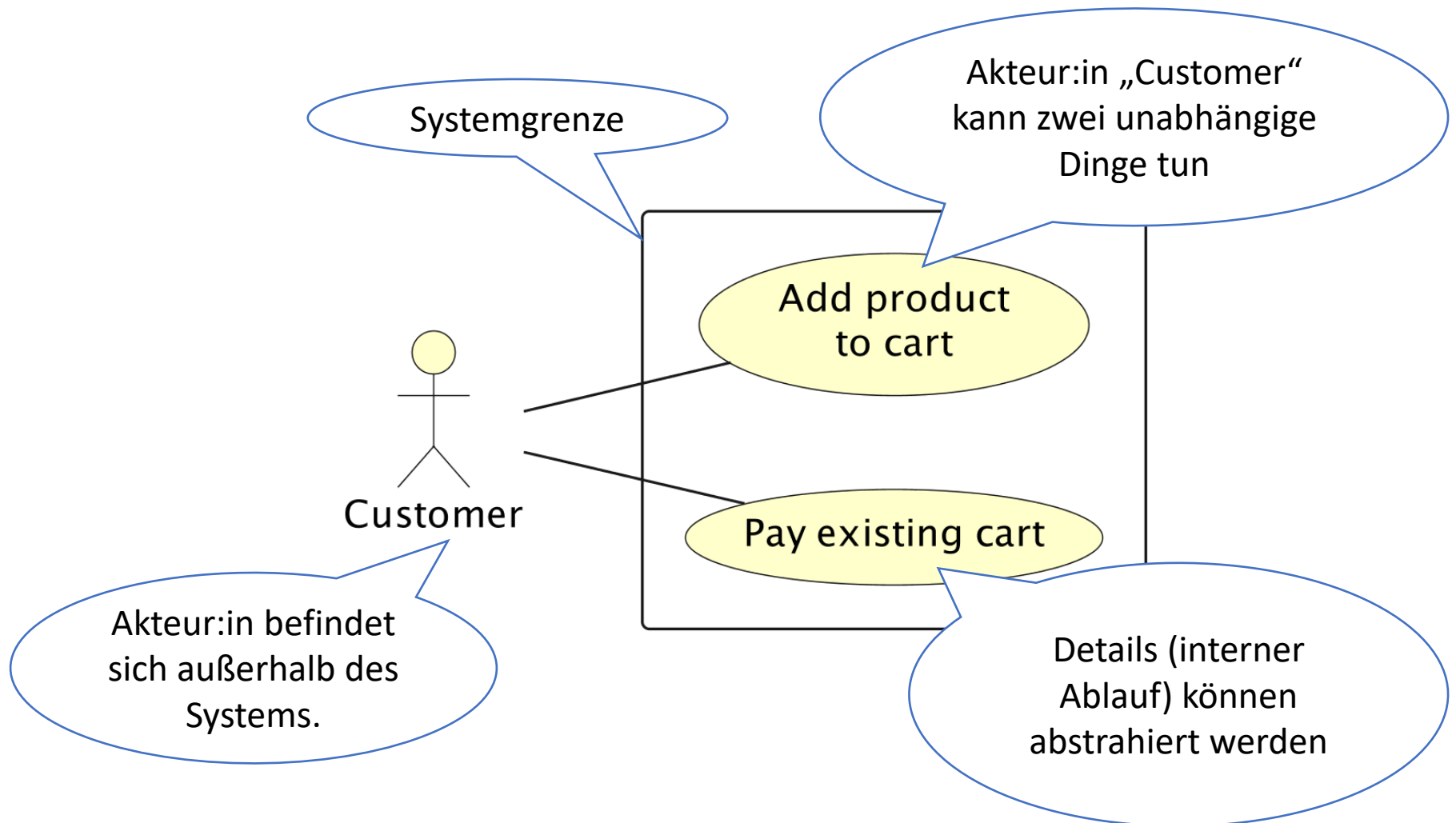
①

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

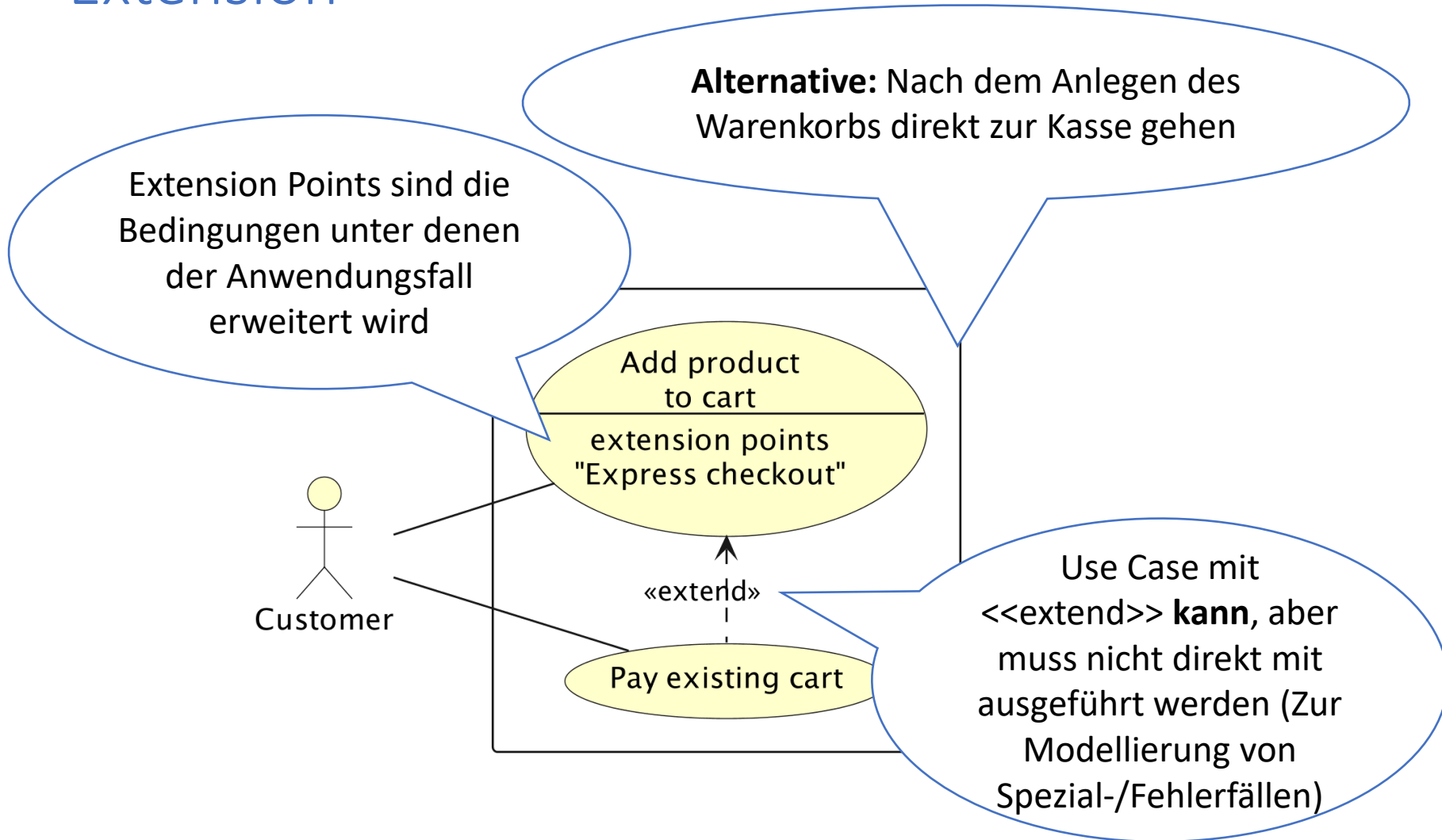
Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

# Use-Case Modellierung

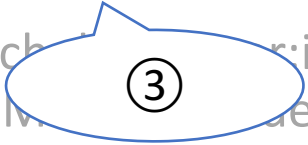


# Extension



# Fallbeispiel (Auszug)

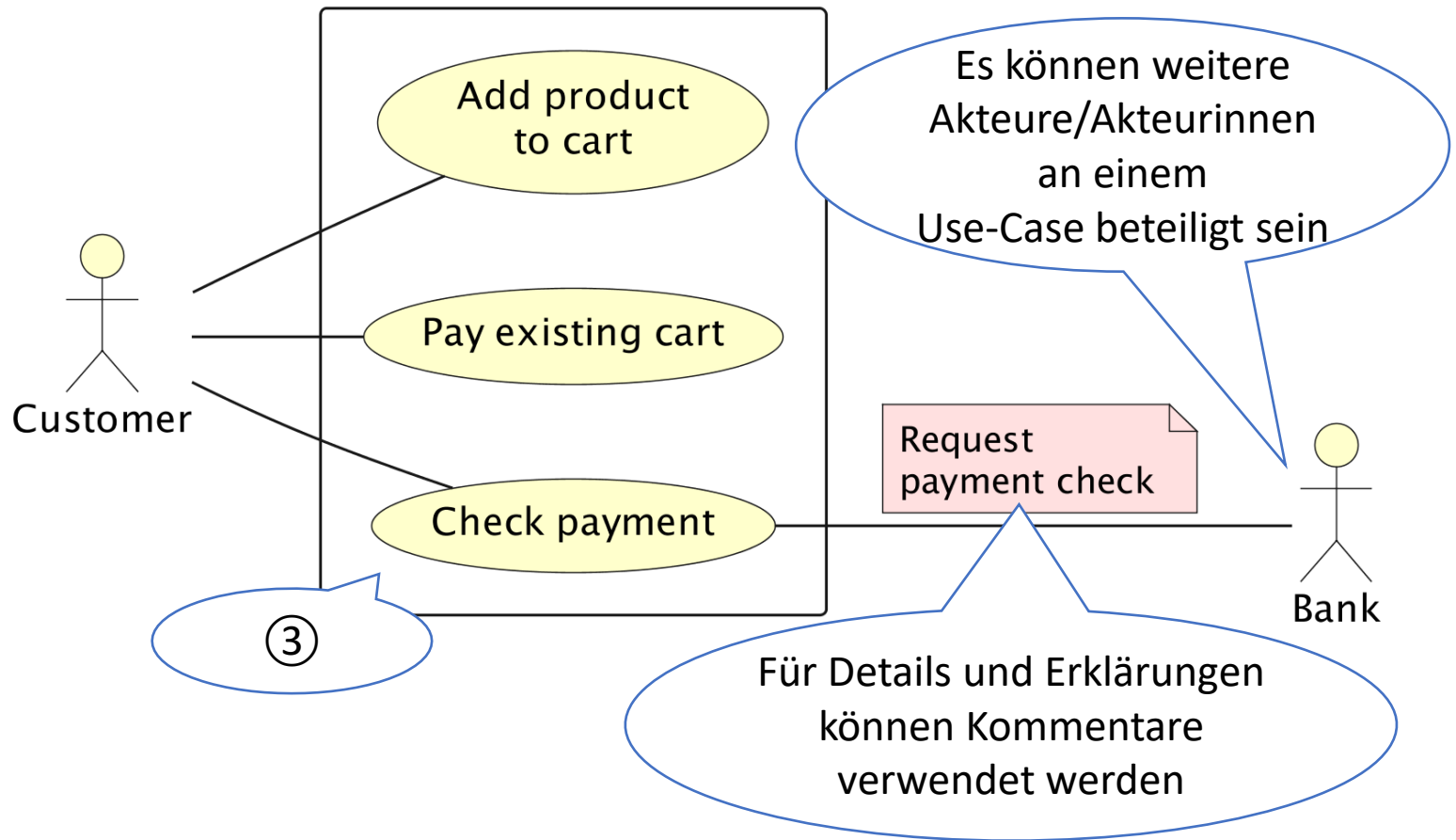
Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Weiterhin soll das System gleichzeitig auch :innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeiter:innen sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

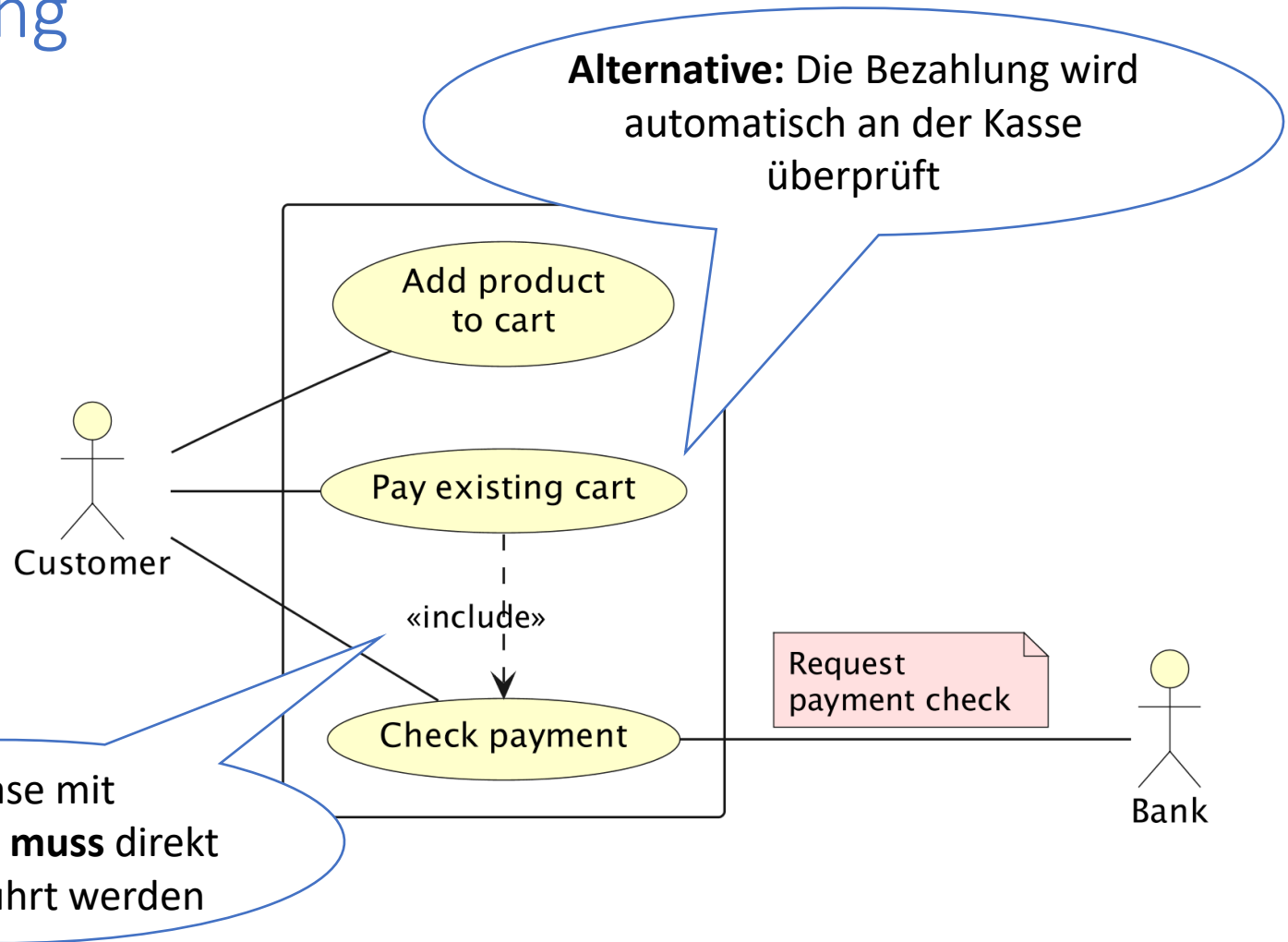
Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

# Mehrere Akteure/Akteurinnen



# Including



# Fallbeispiel (Auszug)

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung erfolgreich erfolgen kann.

②b

②a

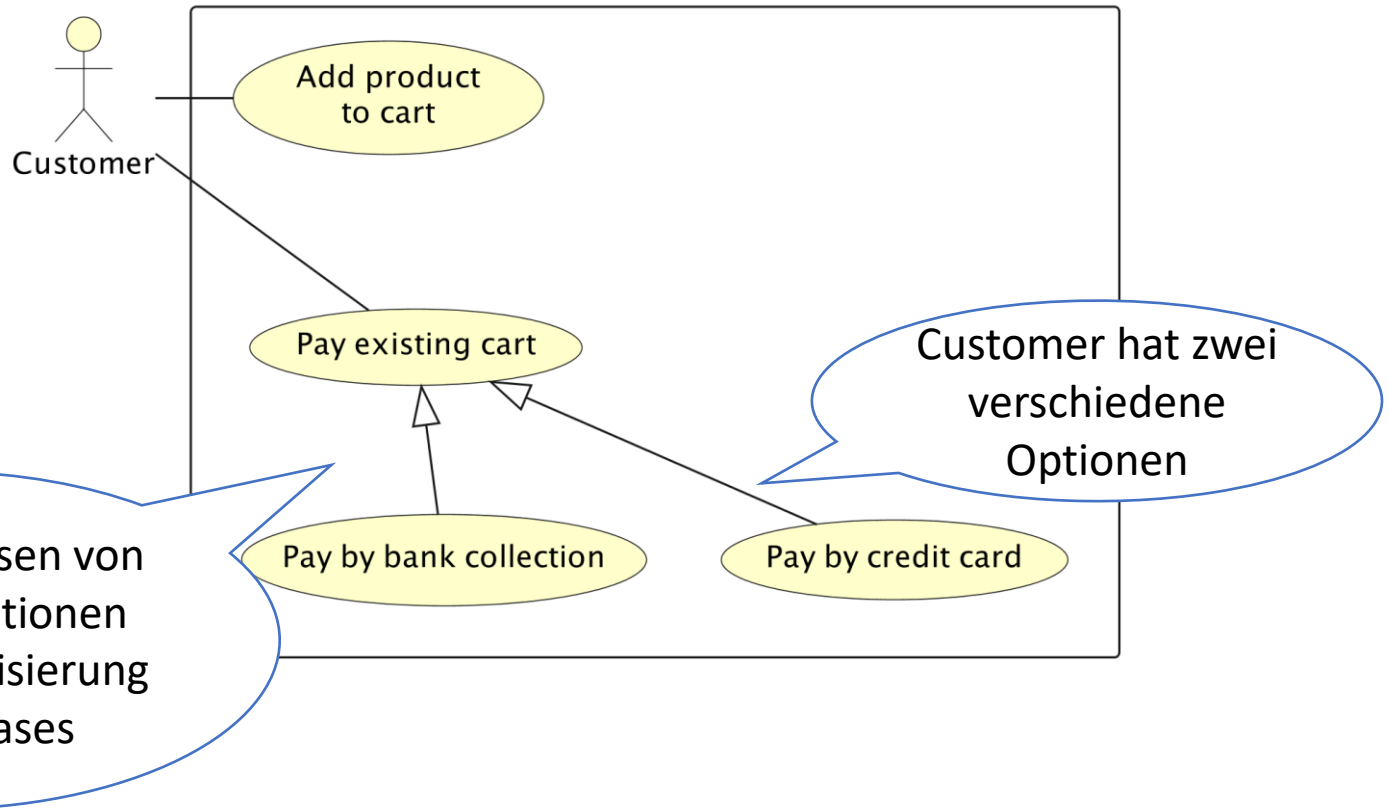
Weiter soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.



# Generalisierung der Use-Cases



# Fallbeispiel (Auszug)

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung abgegeben wird, muss sichergestellt werden, dass die Bezahlung erfolgreich erfolgen kann.

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

Produkte sollen über das Webinterface auch gesucht werden können. Das System soll Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

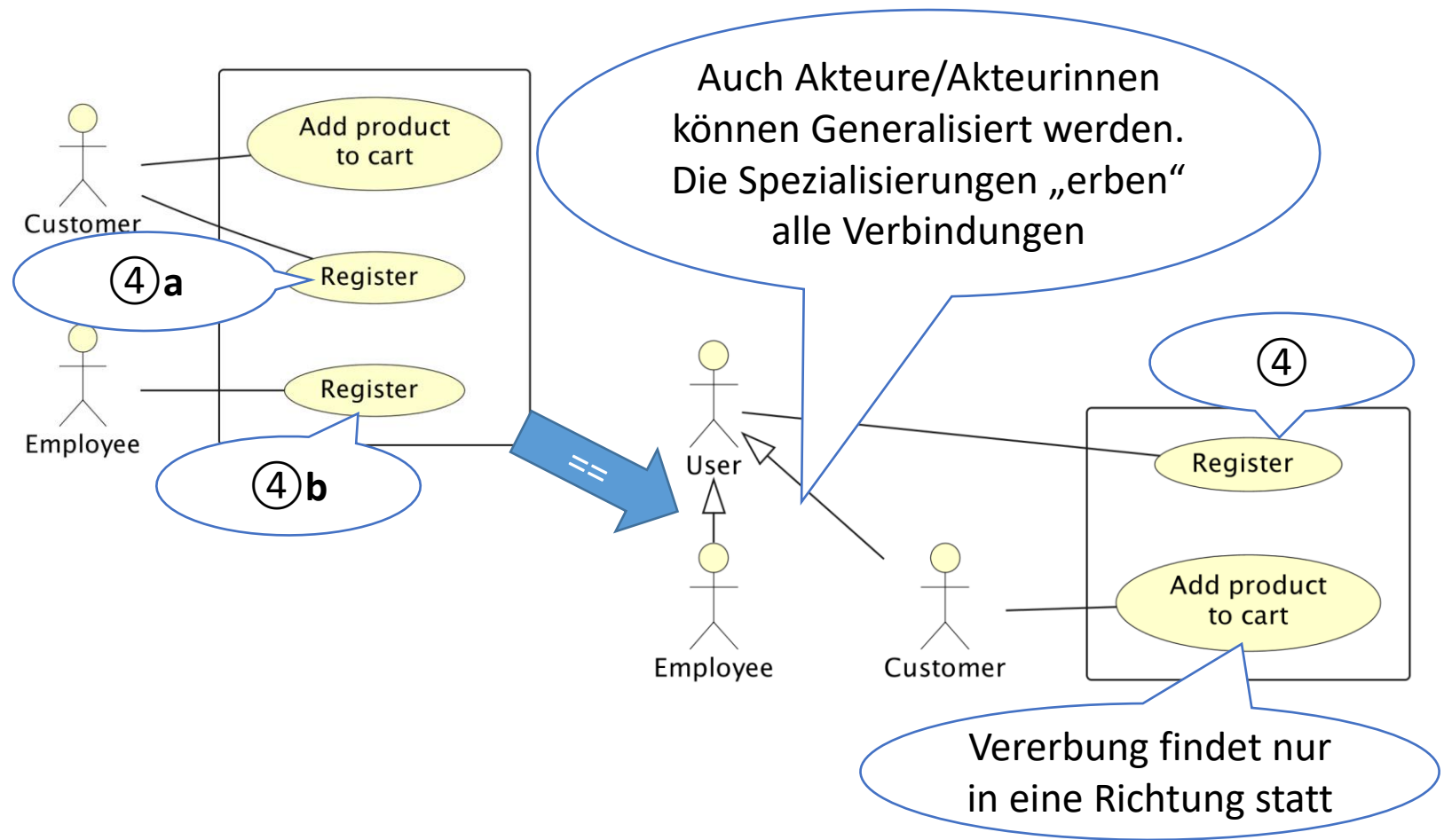
Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

④ a

④ b

④

# Generalisierung der Akteure/Akteurinnen



# Zusammenfassung

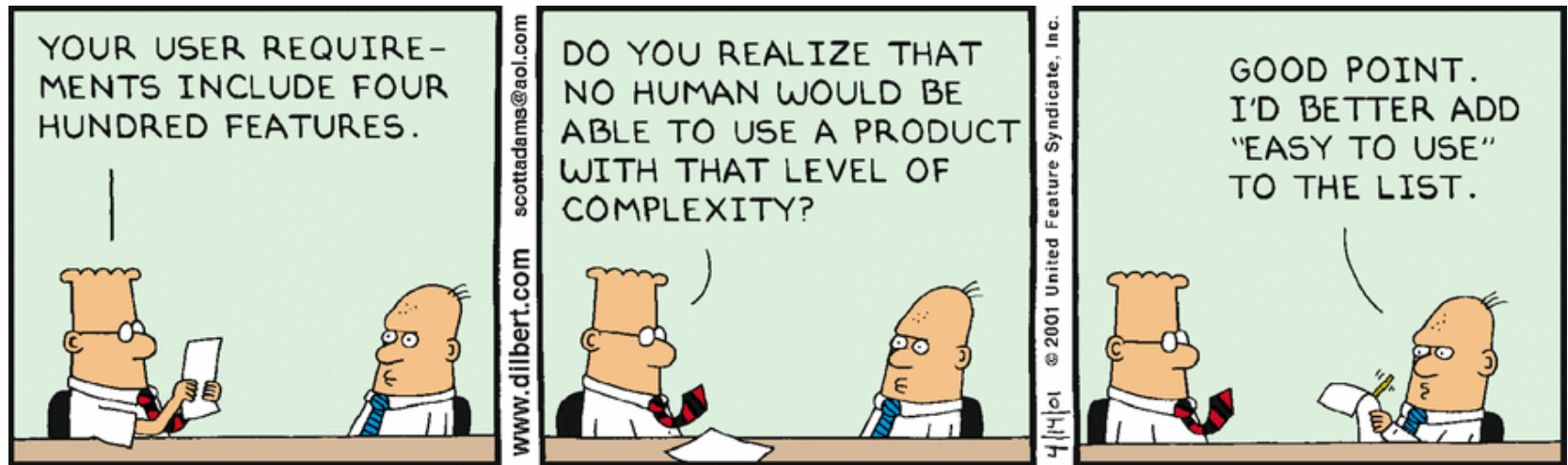
- Anforderungen müssen sorgfältig erfasst werden, um „richtiges“ Produkt entwickeln zu können
- Qualitätskriterien für Anforderungsbeschreibung einhalten
- Beschreibungsmöglichkeiten
  - Natürliche Sprache
  - Strukturierte natürliche Sprache
  - Mathematische Beschreibung
  - Grafische Beschreibung
- Anforderungsanalyse mit dem Use-Case Diagramm

# Inhalt

## Requirements Engineering

- Grundlagen
- Textuelle Anforderungsspezifikation
- Grafische Anforderungsspezifikation
- **Nicht-Funktionale Anforderungen**

# Nicht-funktionale Anforderungen



[License covered by the Classroom Usage Statement](#)

# Funktionale vs. Nicht-Funktionale Anforderungen

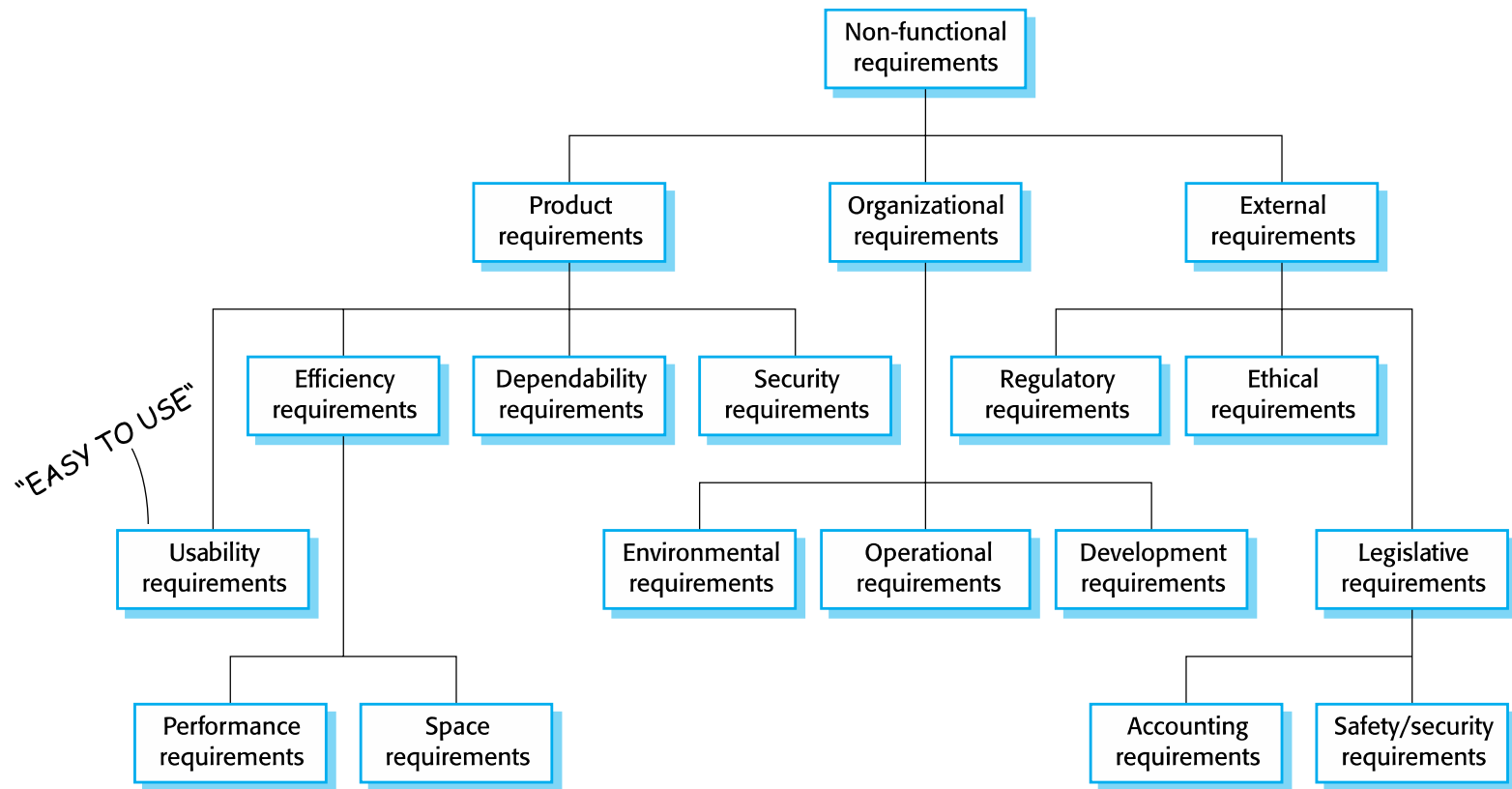
## Funktional

- Beschreibt **was** das System tun bzw. **was** es nicht tun soll
- Betrifft meist **einzelne Aufgaben** des Systems
- Kann meistens **unmittelbar geprüft** werden
  - Funktionalität wie vorhanden ja/nein?

## Nicht-Funktional

- Beschreibt **auf welche Weise** das System bestimmte Dinge tun soll, bzw. **wie** das System sein soll
- Betrifft oft das **gesamte System**
- Kann **nicht immer direkt** überprüft werden
  - Wieviel Speicher wird maximal gebraucht? Wie lange braucht die Berechnung schlimmstenfalls? ...

# Nicht-funktionale Anforderungen



[Ian Sommerville, Software-Engineering, Chapter 4](#)



# Fallbeispiel

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Auf Sicherheit soll entsprechend geachtet werden.

Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer akzeptablen Zeit sollen passende Produkte angezeigt werden.

Alle Funktionen werden von Nicht-Entwickler:innen ausgiebig getestet.

**Findet nicht-funktionale Anforderungen**

# Fallbeispiel

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es den Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. **Auf Sicherheit soll entsprechend geachtet werden.**

Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und **innerhalb von einer akzeptablen Zeit** sollen passende Produkte angezeigt werden.

Alle Funktionen werden **von Nicht-Entwickler:innen ausgiebig getestet.**

- **nicht-funktionale Anforderungen**

# Auszug

- „ Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. **Auf Sicherheit soll entsprechend geachtet werden.**“
- „Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von einer **akzeptablen Zeit** sollen passende Produkte angezeigt werden.“

**Wie könnte man das besser formulieren?**

# Auszug verbessert

- „ Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Es soll über Authentifizierungsmethoden sichergestellt werden, dass nur berechtigte Personen neue Mitarbeitende anlegen können.“
- „Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von höchstens 5 Sekunden sollen passende Produkte angezeigt werden. Gegebenenfalls soll nur ein unvollständiger Teil der möglichen Ergebnisse angezeigt werden.“

# Fallbeispiel

Sie werden gebeten für ein kleines Unternehmen, das Schuhe und Kleidung verkauft, die Verwaltungssoftware eines Online-Shops zu entwickeln. Der Onlineshop soll es Kunden und Kundinnen ermöglichen, Produkte in einen Warenkorb zu legen und diesen zu bezahlen. Als Bezahlmethoden sind zunächst Bankeinzug und Kreditkartenzahlung vorgesehen. Bevor die Bestellung aufgegeben wird, muss sichergestellt werden, dass die Bezahlung tatsächlich erfolgen kann.

Weiterhin soll das System gleichzeitig auch die Nutzer:innen verwalten. Sowohl Kunden und Kundinnen als auch Mitarbeitende sollen registriert werden können. Es soll über Authentifizierungsmethoden sichergestellt werden, dass nur berechtigte Personen neue Mitarbeitende anlegen können.

Produkte sollen über das Webinterface auch gesucht werden können. Dabei sollen Rechtschreibfehler toleriert werden und innerhalb von höchstens 5 Sekunden sollen passende Produkte angezeigt werden. Gegebenenfalls soll nur ein unvollständiger Teil der möglichen Ergebnisse angezeigt werden.

Alle Funktionen sollen von Nicht-Entwickler:innen ausgiebig getestet werden.

# Lernziele

- ☐ Welche Information enthält das Pflichtenheft/welche das Lastenheft?
- ☐ Wie lassen sich Anforderungen textuell notieren?
- ☐ Wie lassen sich natürlichsprachliche Anforderungen strukturieren?
- ☐ Was sind sinnvolle Felder für strukturierte Anforderungen?
- ☐ Was sind User Stories und wann ist ihr Einsatz sinnvoll?
- ☐ Was ist UML?
- ☐ Welche Diagramme eignen sich zur Modellierung von Anforderungen?
- ☐ Woraus bestehen Anwendungsfalldiagramme?
- ☐ Wie detailliert sollten Anwendungsfalldiagramme sein?
- ☐ Wie werden Beziehungen zwischen Use-Cases im Anwendungsfalldiagramm modelliert?
- ☐ Wie lassen sich mehrere Use-Cases bzw. Akteure/Akteurinnen zusammenfassen?
- ☐ Wann verwendet man include, wann extend?
- ☐ Was ist der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?
- ☐ Welche Arten von nicht-funktionalen Anforderungen gibt es?
- ☐ Können nicht-funktionale Anforderungen genauso wie funktionale validiert werden?

# Quellen

1. IAN SOMMERVILLE, *Software-Engineering*, Pearson, 2012
2. MIKE COHN, *User Stories Applied: For Agile Software Development*, 2004