V04 - Anforderungsanalyse

Softwaretechnik 1

Wintersemester 2021/2022

Prof. Dr.-Ing. Jasminka Matevska

(jasminka.matevska@hs-bremen.de)

Hochschule Bremen, Fakultät Elektrotechnik und Informatik, TI, ISTI, DSI & ATMEI

Rechtliche Hinweise

- Die Rechte an geschützten Marken liegen bei den jeweiligen Markeninhabern
- Die Rechte an referenzierte Literatur, Folien, Notizen und sonstigen Materialien liegen bei den jeweiligen Autoren
- Diese Veranstaltung benutzt Materialien von Prof. Dr. Spillner (HSB in Ruhestand) und Prof. Dr. Hasselbring (Universität Kiel)
- Alle Rechte an den Materialien zu dieser Veranstaltung liegen bei ihrem Autor, Prof. Dr.-Ing. Jasminka Matevska
- Jede Form der teilweisen oder vollständigen Weitergabe, Speicherung auf Servern oder Nutzung in Lehrveranstaltungen, die nicht von dem Autor selbst durchgeführt werden, erfordert seine schriftliche Zustimmung. Eine schriftliche Zustimmung ist darüber hinaus für jede kommerzielle Nutzung erforderlich
- Für inhaltliche Fehler kann keine Haftung übernommen werden

Der **rote** Faden

- Einführung
- Entwicklungsprozess
- Vorgehensmodelle
- **Anforderungsanalyse**
- Entwurf
- Objektorientierter Entwurf mit UML
- Test
- Qualitätssicherung
- Konfigurationsmanagement
- Implementierung
- Projektmanagement

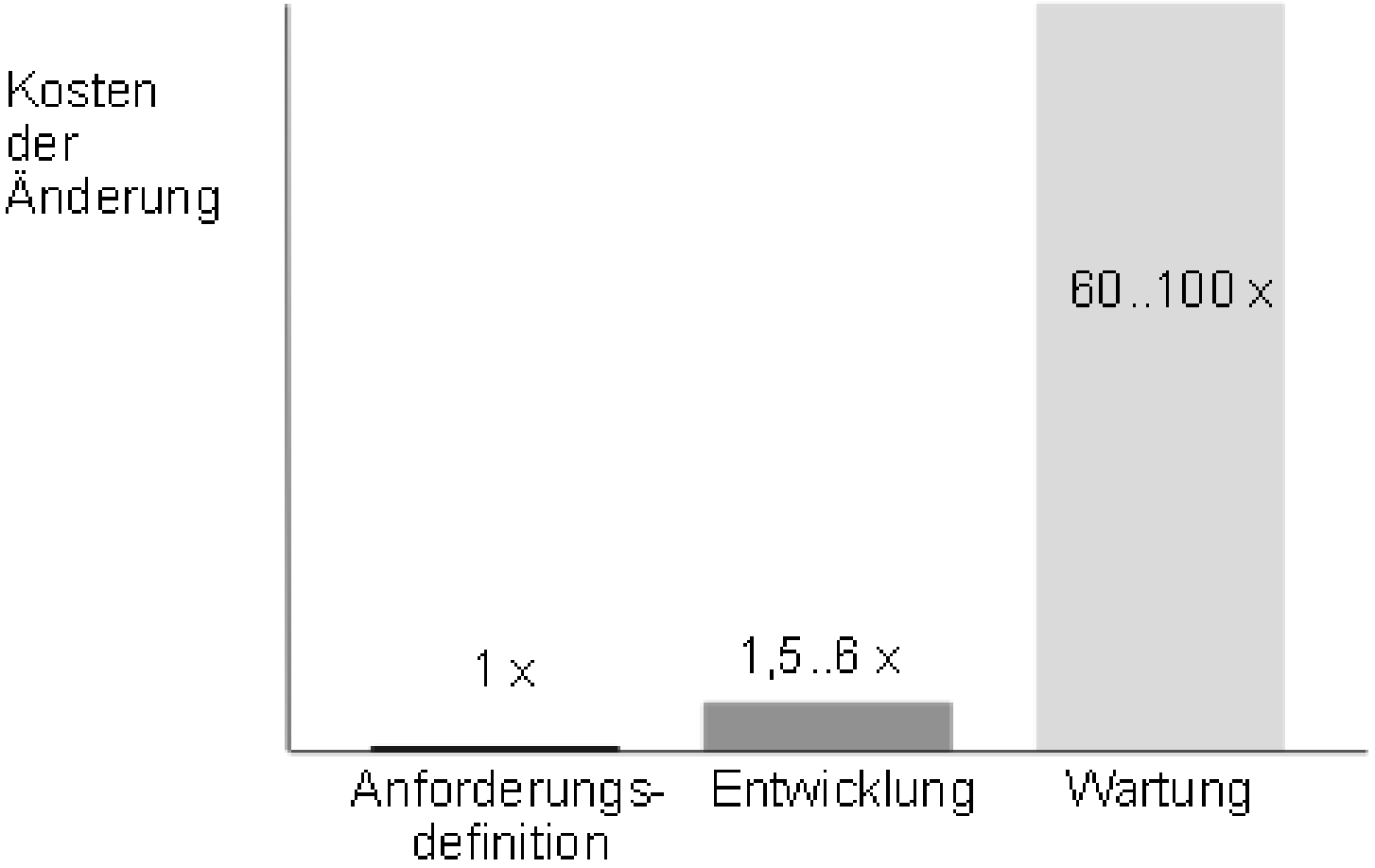
Was ist eine Anforderung?

- Ausdruck des Kundenbedarfs
- Ausgangspunkt der Entwicklung → Vom Kunden zum System
- Ursache von Folgefehlern und Streitigkeiten

Was ist eine Anforderung (Requirement)?

- Nach IEEE ist eine Anforderung die dokumentierte Darstellung einer Beschaffenheit oder Fähigkeit…
  - … die von einem Benutzer zur Lösung eines Problems oder Erreichung eines Ziels benötigt wird oder
  - … die ein System oder Systemteil erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen

Warum ist Anforderungs-Engineering notwendig?



Aufgaben des Anforderungs-Engineering

- **Ermittlung**
  - Interviews
  - Geschäftsprozesse/Szenarien/Anwendungsfälle
  - Prototyping
- **Spezifikation**
  - Szenarien/Anwendungsfälle (Use Cases/User Stories)
  - Text/Beschreibung
  - → Pflichtenheft / Product Backlog

Anforderungsanalyse

- Prüfung
  - Review, Inspektion
  - Verifikation & Validierung
- Verwaltung
  - Releases, Branches, Versionen (verwandt mit Änderungs- und Konfigurationsmanagement)
  - Werkzeuge (z.B. DOORS)

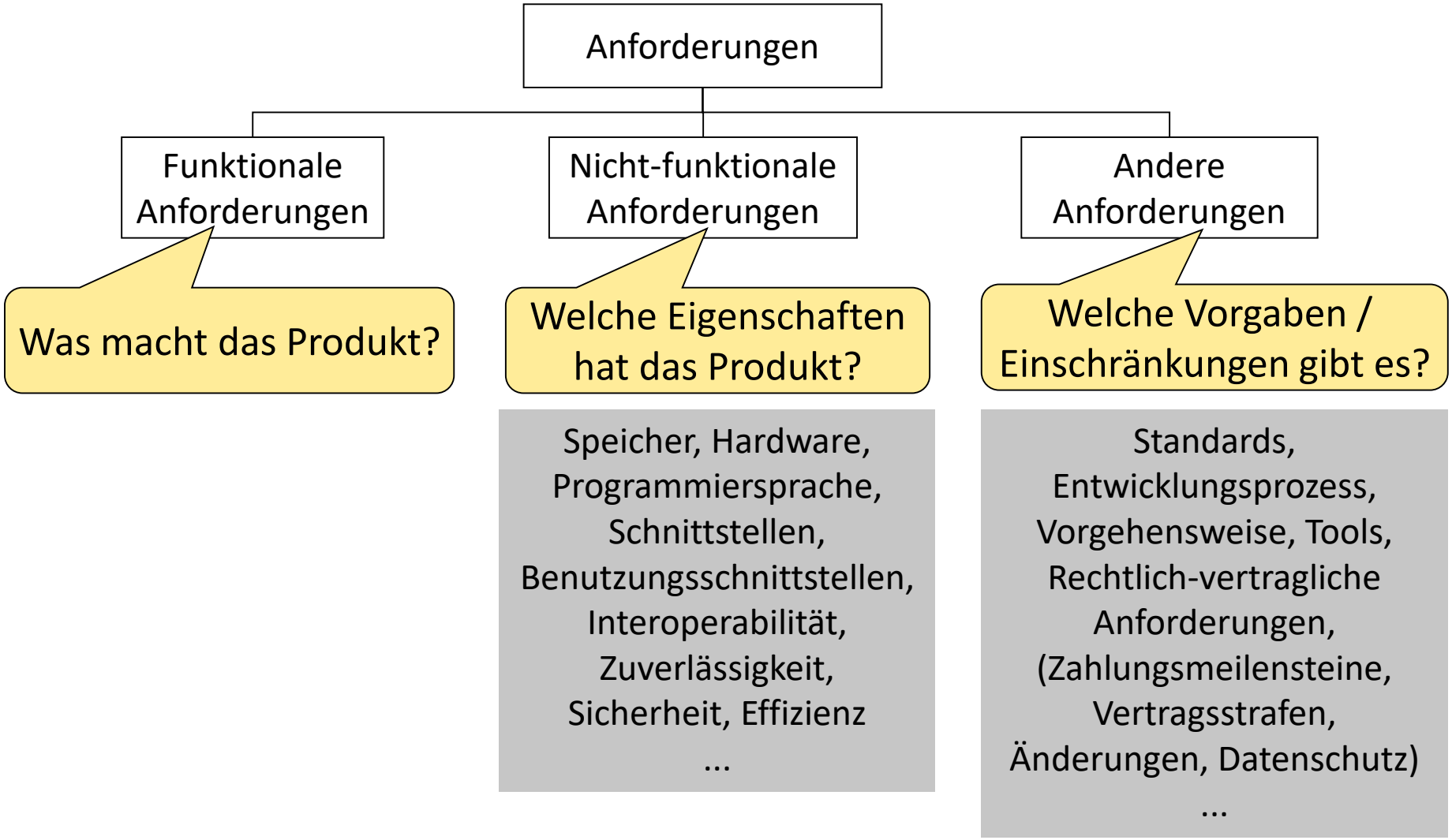
Anforderungsanalyse - Grundlagen

- Anforderungsanalyse ist ein Teilgebiet des Anforderungs-Engineering
- In der Anforderungsanalyse werden Anforderungen des Kunden an das System identifiziert
- Sie ist ein kommunikativer und iterativer Prozess mit dem Ziel, Anforderungen vollständig und konsistent zu erfassen
- Als Ergebnis soll ein Pflichtenheft (Anforderungsspezifikation) entstehen, welches die Anforderungen an das Softwareprodukt schriftlich fixiert
- Iterative evolutionäre Softwareentwicklungs-Modelle und agile Vorgehensweisen ermitteln die Anforderungen schrittweise

Lasten- und Pflichtenheft

- Das Lastenheft beschreibt die Anforderungen aus Kundensicht
- Das Pflichtentenheft beschreibt die Anforderungen aus Lieferantensicht
  - Vertragsgrundlage
  - von allen am Projekt Beteiligten akzeptiert
- Lasten- und Pflichtenhefte enthalten üblicherweise natürlich-sprachliche Beschreibungen dessen, was die zu realisierenden Software-Systeme leisten sollen
- Die VDI-Richtlinie 2519 Blatt 1 unterscheidet explizit zwischen Lasten- und Pflichtenheft und beschreibt eine Vorgehensweise, um Lasten- und Pflichtenhefte zu erstellen [VDI 2001]

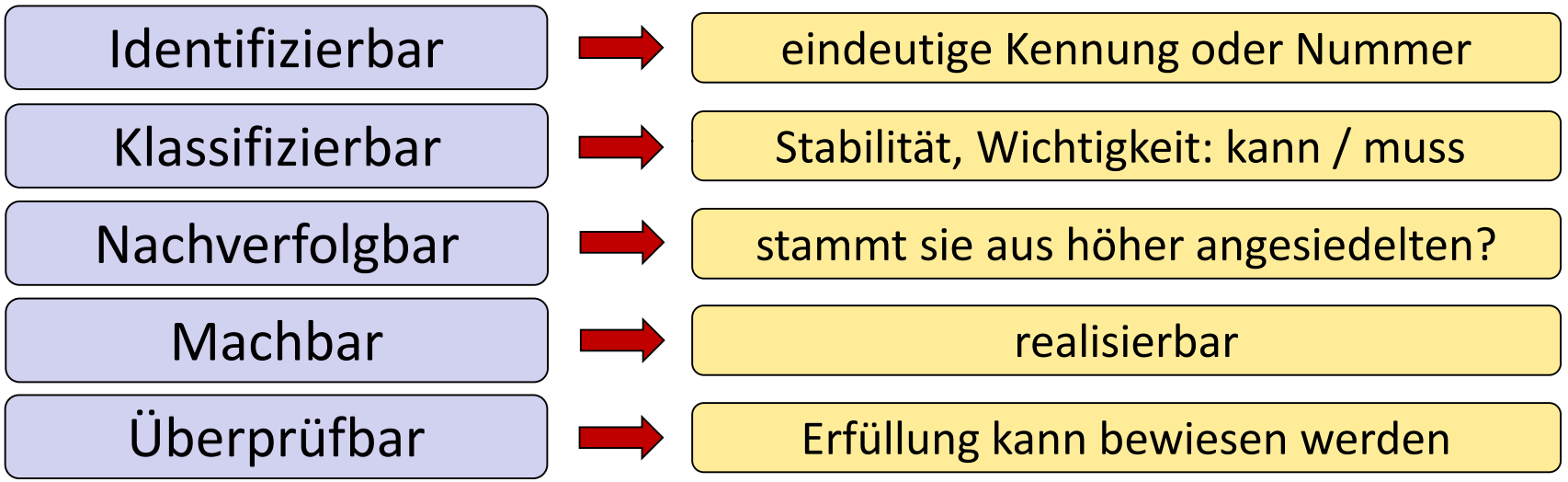
Arten von Anforderungen



Eigenschaften einer Anforderung: Inhalt

Notwendig	→	vom Auftraggeber gefordert
Eindeutig	→	von allen Beteiligten eindeutig verstanden
Atomar	→	eine Eigenschaft pro Anforderung
Vollständig	→	beschrieben ohne Interpretationsräume
Korrekt / Spezifisch	→	entspricht dem zu realisierendem System
Konsistent	→	Ist in sich widerspruchsfrei
Implementierungsfrei	→	beschreibt was und nicht wie

Eigenschaften einer Anforderung: Behandlung



- Weitere mögliche Angaben zu jeder Anforderung:
  - Risiko und Maßnahmen dagegen

A Requirement is MUST

- **M**easurable
  - We must be able to verify and validate the product with respect to the written requirements
- **U**seful
  - Explicitly identified and justified
- **S**imple
  - Atomic, no unnecessary and redundant, but clear & precise formulation
- **T**raceable
  - Through the links with the sources, the justification, the implementation in the low-level specification, and verification/validation activities

Attribute einer Anforderungsbeschreibung

- ID
- Typ
- Name
- Beschreibung
- Priorität
- Klassifizierung / Zuordnung
- Verifikationsmethode
  - Test (T)
  - Analyse (A)
  - Review of Design (RoD)
  - Inspektion (I)
- Status
- Optional: Risiken, Aufwand, Quellen …

ID	**FE-F-1**
Typ	funktional
Name	Host-Erreichbarkeit
Beschreibung	Das Web-Frontend soll die Erreichbarkeit der angeschlossenen Hosts anzeigen
Priorität	hoch
Klassifizierung	Frontend
Verifikations-methode	T
Status	muss

Übung: Beispiel 1

- *“The ACRV System shall provide special medical life-support accommodations for one ill or injured crew member consisting of medical life-support **and** monitoring equipment **and** the capability of limiting impact accelerations on that crew member to be not greater than ... **and** for a total impulse not to exceed ...”*

- **Frage:** Ist diese Anforderung gut formuliert?

Übung: Beispiel 1 (cont.)

- *"The ACRV System shall provide special medical life-support accommodations for one ill or injured crew member consisting of medical life-support **and** monitoring equipment **and** the capability of limiting impact accelerations on that crew member to be not greater than ... **and** for a total impulse not to exceed ..."*
- **Problem**: es handelt sich hier um mehrere Anforderungen → nicht atomar
- Besser drei Anforderungen formulieren:
- *"... shall provide monitoring equipment for one crew member."*
- *"... shall limit impact accelerations to the ill or injured crew member to …"*
- *"... shall limit total impulse to the ill or injured crew member to ..."*

Übung: Beispiel 2

- *“A suitable means shall be provided for conversion of maintenance data to a format that will permit analysis by ground facilities.”*

- **Frage:** Was ist hier das Problem?

Übung: Beispiel 2 (cont.)

- *"A suitable means shall be provided for conversion of maintenance data to a format that will permit analysis by ground facilities."*

- **Problem:** Anforderung ist ungenau und nicht verifizierbar
  - Was ist "suitable"?
  - Was für eine "conversion"?
  - Was für eine Analyse? ("permit analysis")
- Mögliche Verbesserung:
- *"The maintenance data management function shall output a failure message as specified in … to the Central Maintenance System (CMS) when receiving a Continuous Built-In Test (CBIT) message."*

Übung: Beispiel 3

- *“New and modified air distribution components shall be designed to minimize noise levels .”*

- **Frage:** Ist diese Anforderung besser formuliert?

Übung: Beispiel 3 (cont.)

- *“New and modified air distribution components shall be designed to minimize noise levels .”*
- **Probleme:**
  - Nicht spezifisch:”be designed to” ist nutzlos für das Verständnis der Aussage
  - Nicht messbar: “minimize” ist ungenau, es soll quantifiziert werden, damit es verfizierbar ist
  - Nicht eindeutig: welche ”noise levels” (z.B. zu welchen Standards)?
  - Möglicherweise nicht relevant: “Noise levels” sind kein zu behandelndes Thema wenn diese Komponenten nicht im Arbeitsraum installiert warden können.
- Mögliche Verbesserung:
- *“New and modified air distribution components installed in cabin areas shall emit less than 30 dB(A) of acoustic noise.”*

Übung

- Welche Aufgaben gehören zum Anforderungs-Engineering?
- Womit beschäftigt sich die Anforderungsanalyse?
- Was ist das Ergebnis einer Anforderungsanalyse?
- Was beschreibt ein Lastenheft?
- Was beschreibt ein Pflichtenheft?
- Welche Arten von Anforderungen kennen Sie?
- Welche Eigenschaften soll eine Anforderung haben? Warum?
- Welche Attribute soll eine Anforderungsbeschreibung haben?

Anforderungsermittlung

- Fragen:
  - **Wer** möchte **was** mit dem System machen **warum**?
  - **Was** soll das System **wo/wann** können/leisten, damit das gelingt?

- Analyse der **Ist**-Situation, (**verstehen/beobachten**):
  - Wie wurden die Aufgaben bisher erledigt?
  - Gibt es ein altes System, das abgelöst werden soll?
  - ...
- Analyse der **Soll**-Situation (**Ideen finden/Szenarien „durchspielen“**):
  - Geschäftsprozesse
  - Anwendungsfälle
  - Benutzungsschnittstelle
  - Objektmodell
  - Nachbarsysteme
  - ...

Techniken zur Anforderungsermittlung

Review der Documentation

Interviews

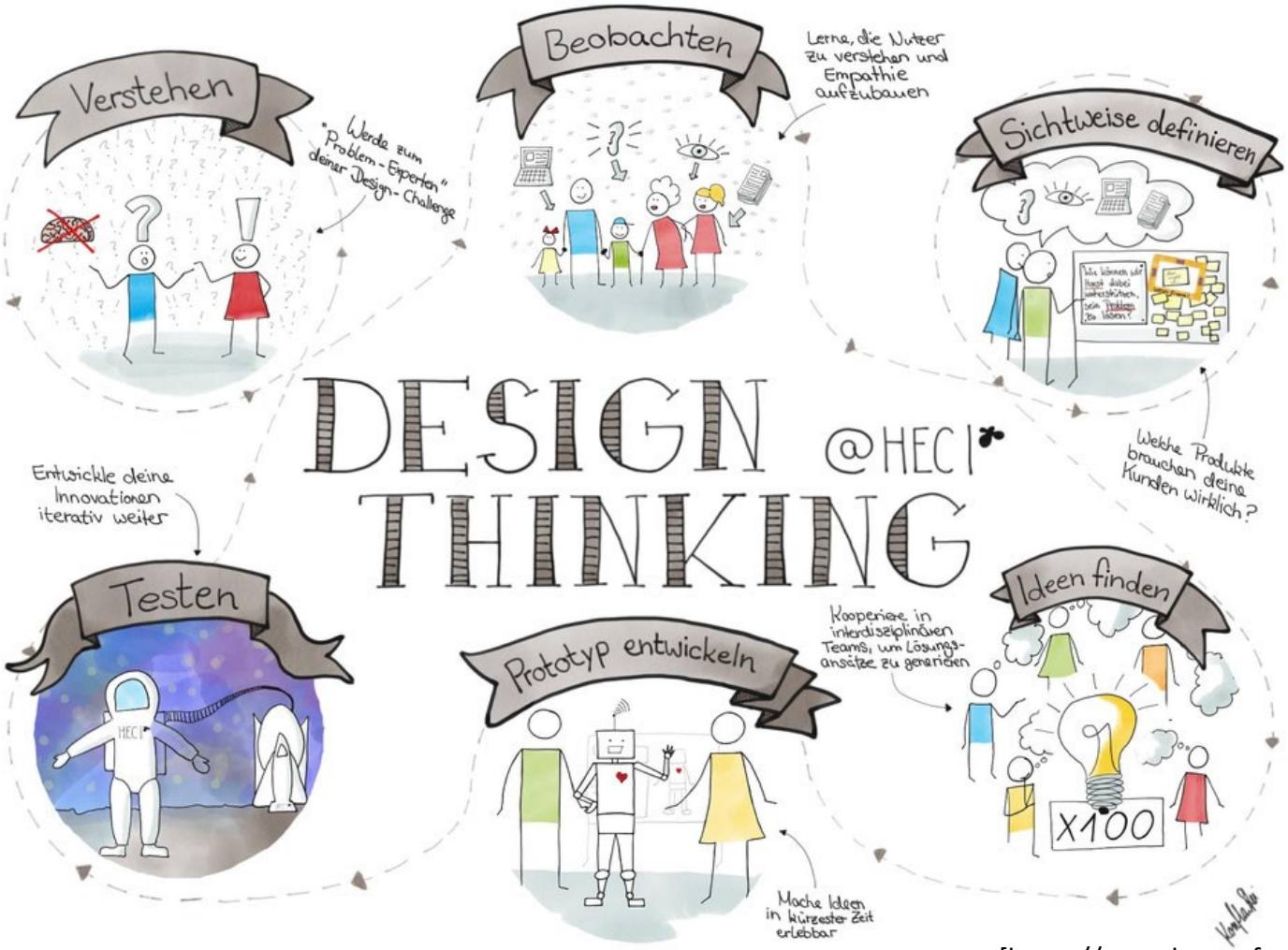
- Vorhandene Dokumentation einsehen
- Informieren als Vorbereitung
- Face-to-face Gespräche mit kritischen Stakeholder
- Eine Check-Liste vorbereiten
- Ergebnisse / Einigungen dokumentieren (während oder direct nach dem Interview)
- 5 “W” – Fragen stellen
- Konkrete Szenarien “durchspielen”

Fokus-Gruppen/Experten Befragung

- Ausgewählte Stakeholder (6-10) in einer offenen Runde befragen
- 5 “W” – Fragen stellen
- Konkrete Szenarien “durchspielen”
- Probleme und Erwartungen hoch bringen

[nach Forsberg, Visualizing PM, 2013]

Design Thinking zur Anforderungsermittlung

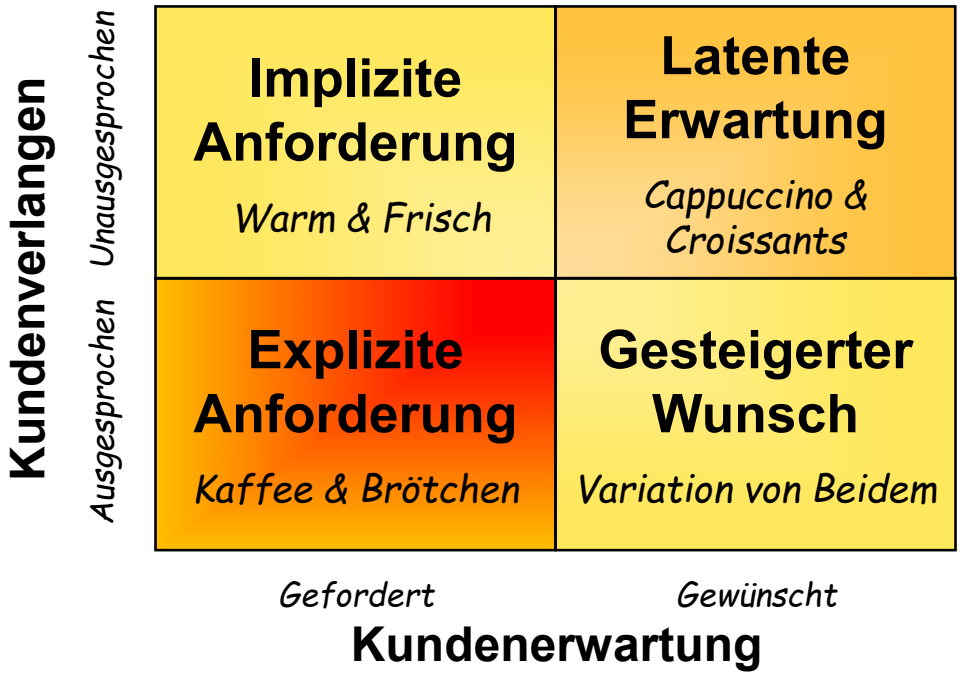


[https://www.hec-software-akademie.de]

Kundenanforderungen: über das Ausgesprochene hinaus

Klarheit über **Kundenanforderungen**

Wünsche und **Erwartungen** abgrenzen



- Anforderungen ergründen
- (versteckte) Erwartungen berücksichtigen
- Lösungen definieren, die den Zweck erfüllen
- Sicherstellen, dass die Lösung die Erwartungen trifft

[FORS/VPM, Center for System Management]

MoSCoW-Methode: Klassifizierung der Anforderungen

- **M** – MUST (20 %)
  - **Erforderlich**
  - Essentiell wichtig für das Projekt
  - Nicht verhandelbar
- **S** – SHOULD (30 %)
  - Hohe Relevanz für das Projekt, aber nicht erfolgskritisch
  - Umsetzung **sollte** erfolgen nur wenn MUST-Anforderungen nicht beeinträchtigt
- **C** – COULD (50 %)
  - *Nice to have* (geringe Relevanz, aber können Mehrwert erzeugen)
  - Umsetzung **kann** erfolgen nur wenn MUST-Anforderungen nicht beeinträchtigt
  - **Problem: sind das dann überhaupt noch Anforderungen?**
- **W** - WON'T
  - fachlich und/oder technisch wichtig, aber nicht zeitlich kritisch
  - wird im aktuellen Projekt **nicht umgesetzt**, aber für die Zukunft vorgemerkt

Übung

- Wie können Anforderungen ermittelt werden?
- Was ist zu berücksichtigen?
- Was ist der Unterschied zwischen einer expliziten und einer impliziten Anforderung?
- Was beschreibt die MoSCoW-Methode?

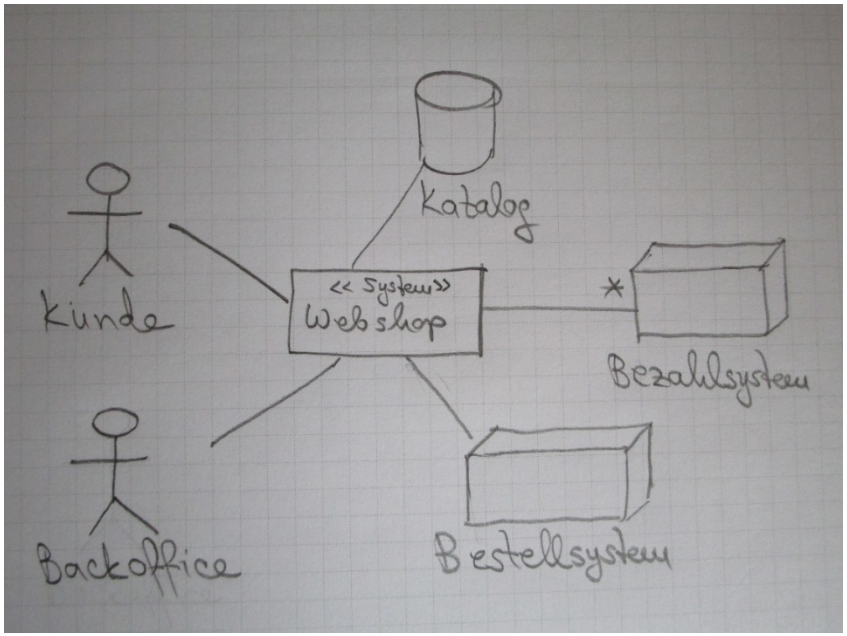
Fragen:

Wer möchte **was** mit dem System machen **warum**?

Was soll das System **wo/wann** können/leisten, damit das gelingt?

1. Schritt: Kontext ermitteln

- Eine Kontextbetrachtung grenzt ab, wofür das zu beschreibende System verantwortlich ist, und wofür nicht. Wer sind die Benutzer des Systems? Mit welchen Fremdsystemen interagiert es?
- **Hauptfrage:** Wer macht was mit dem System warum?



2. Stakeholder identifizieren

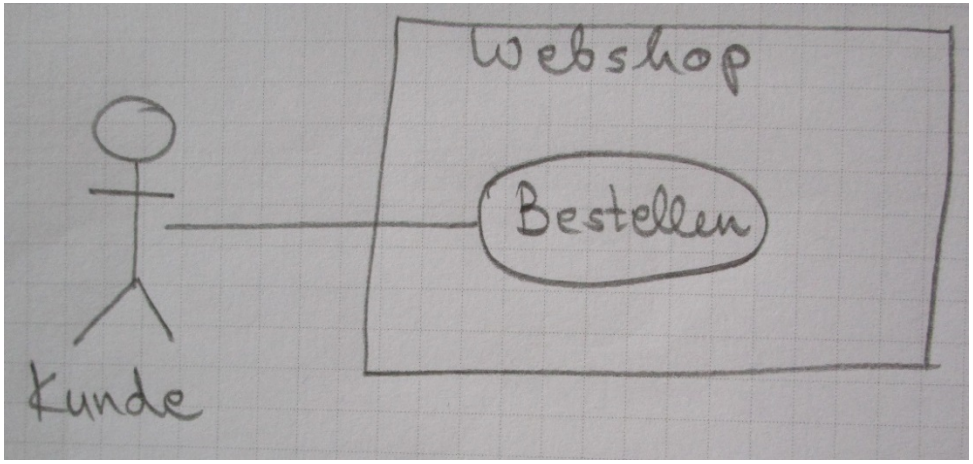
- Stakeholder wird eine Person oder Gruppe bezeichnet, die ein berechtigtes Interesse am Verlauf oder Ergebnis eines Prozesses oder Projektes hat
  [de.wikipedia.org/wiki/Stakeholder]
- Stakeholder können
  - aktiv im Projekt involviert sein
  - das Projekt positiv oder negativ beeinflussen
- Unterschiedliche Stakeholder haben
  - unterschiedlich großes Interesse
  - unterschiedlich große Macht
- Benutzer/Anwender/Akteure sind die Stakeholder, die das System einsetzen

3. Schritt: Akteure identifizieren

- Anforderungsspezifikation: „Was soll das System können?“
- Motivation: Das System soll zunächst aus Benutzersicht beschrieben werden
- Wer sind die Benutzer des Systems?
  - Primary Actor = Hauptbenutzer, für die das System implementiert wird
  - Secondary Actor = Benutzer, der die Benutzung durch die Primary Actors möglich macht
- Welche Rollen nehmen die Benutzer ein?
- Beispiel:
  - „Administrator“ als Secondary Actor für "normale Benutzer“ als Primary Actor
  - Aber der Administrator kann auch die Rolle eines „normalen Benutzers“ einnehmen

Was ist ein Akteur?

- Rolle, die ein Benutzer des Systems spielt
- Hat einen Einfluss auf das System
- Ist häufig eine Person
- Kann auch eine Organisationseinheit oder ein anderes System sein
- Befindet sich immer außerhalb des zu entwickelnden Systems



4. Schritt: Geschäftsprozesse identifizieren

- Zeitliche und logische Folge von Arbeitsschritten (Aktivitäten), die materielle oder immaterielle Dinge erstellen, manipulieren oder weitergeben.
- Daran können Menschen und Maschinen mitwirken.
- Jeder Geschäftsprozess beschreibt einen Ablauf der realen Geschäftswelt unabhängig von jeder digitalen Unterstützung
- Beispiele
  - Reisebuchung bei einem Reiseveranstalter
  - Schadensfallabwicklung bei einer Kfz-Versicherung
  - Frequenzzuweisung von neuen Satelliten
  - Check-in-Prozess am Flughafen
  - Berufungsverfahren an einer Hochschule

5. Schritt: Anwendungsfälle identifizieren

- Anwendungsfälle bilden die Geschäftsprozesse als Szenarien in einem Softwaresystem ab
- Alle identifizierten Anwendungsfälle beschreiben den Funktionsumfang des Systems
- Damit sollen die gesamten fachlichen Anforderungen an das System iterativ ermittelt werden
- Die Anwendungsfälle werden aus der Sicht der Akteure abgeleitet:
  - Die Hauptaufgaben jedes Akteurs ermitteln!
  - Liest/schreibt/ändert ein Akteur Informationen?
  - Informiert ein Akteur das System über Umweltänderungen?
  - Möchte ein Akteur über die Systemzustände informiert werden?
- Agil: User Stories als Szenarien

Was ist ein Anwendungsfall (Use Case)?

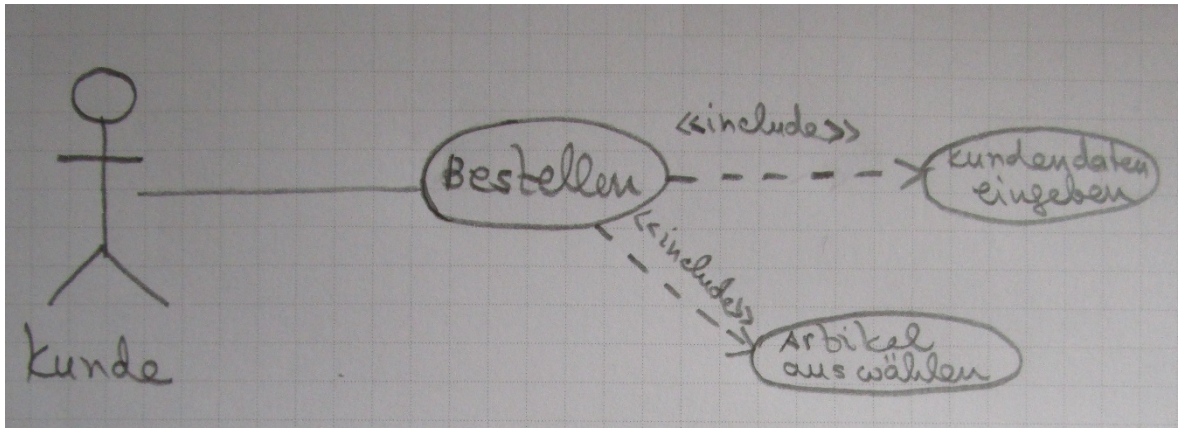
- Der Begriff des Anwendungsfall wurde von Jacobson in die Objektorientierung eingeführt [Jacobson u. a. 1992]
- Ein Anwendungsfall in einem Softwaresystem ist die Sequenz von zusammengehörenden Transaktionen, die von einem Akteur im Dialog mit einem System durchgeführt wird, um ein Ziel zu erreichen
- Ein Use Case zeigt ein nach außen sichtbares Verhalten des Systems
- Alle Anwendungsfälle zusammen dokumentieren alle Möglichkeiten der Benutzung des Systems und somit die benötigte Funktionalität
- Gängige Notation: UML 2

6. Schritt: Beziehungen unter Anwendungsfällen

- Anwendungsfälle zerlegen die Geschäftsprozesse in Portionen, die aus Sicht des Anwenders sinnvoll (und aus Sicht des Entwicklers programmierbar) sind
- Jeder Anwendungsfall unterstützt einen, in seltenen Fällen auch mehrere Geschäftsprozesse
- Deshalb ist es notwendig, die Strukturen und Beziehungen zwischen den Anwendungsfällen zu erkennen
- Beziehungen unter Anwendungsfällen werden u.a. genutzt, um Wiederverwendung von Anwendungsfällen zu ermöglichen

Beziehungen zwischen Anwendungsfällen: **Include**

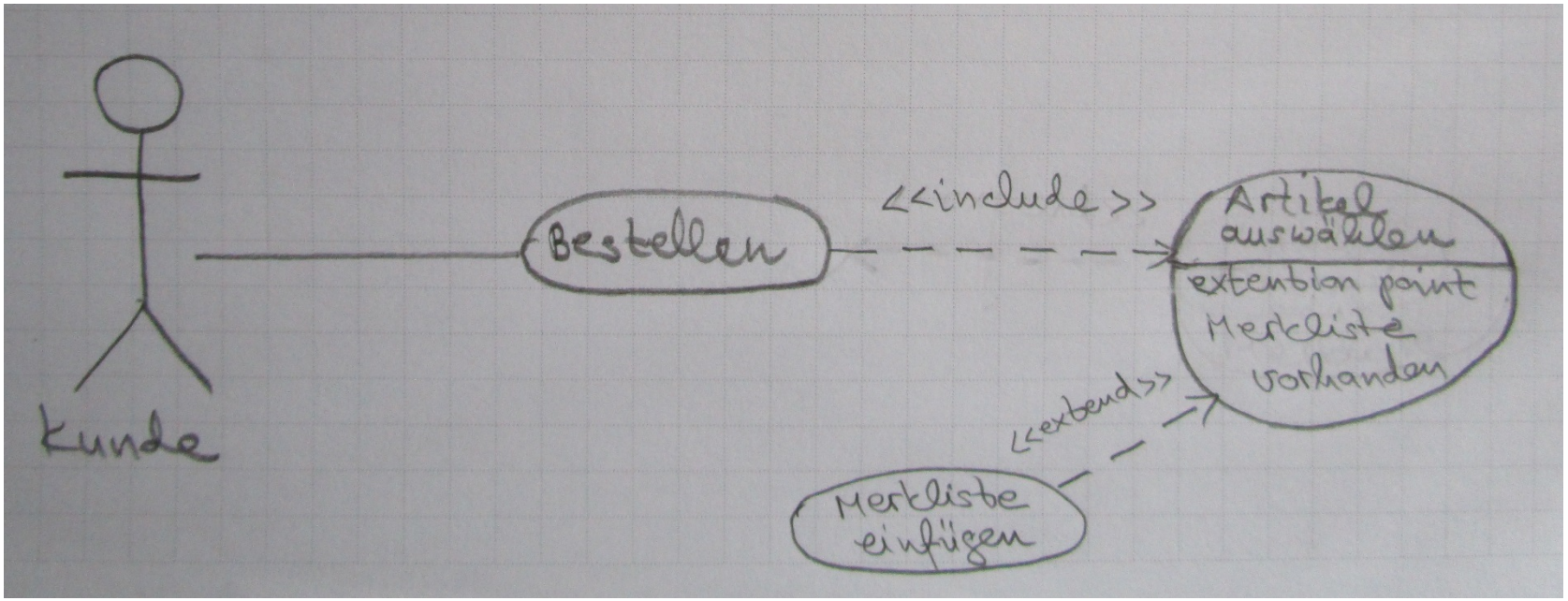
- Ein Anwendungsfall enthält alle Aktivitäten eines anderen Anwendungsfalls
- Eine **include**-Beziehung von Anwendungsfall A zu Anwendungsfall B besagt, dass das Verhalten von B in A eingefügt wird
- Die **include**-Beziehung besagt, dass der benutzte Anwendungsfall **unbedingt notwendig** ist, um die Funktionalität des benutzenden Anwendungsfalls sicherzustellen



Beziehungen zwischen Anwendungsfällen: **Extend**

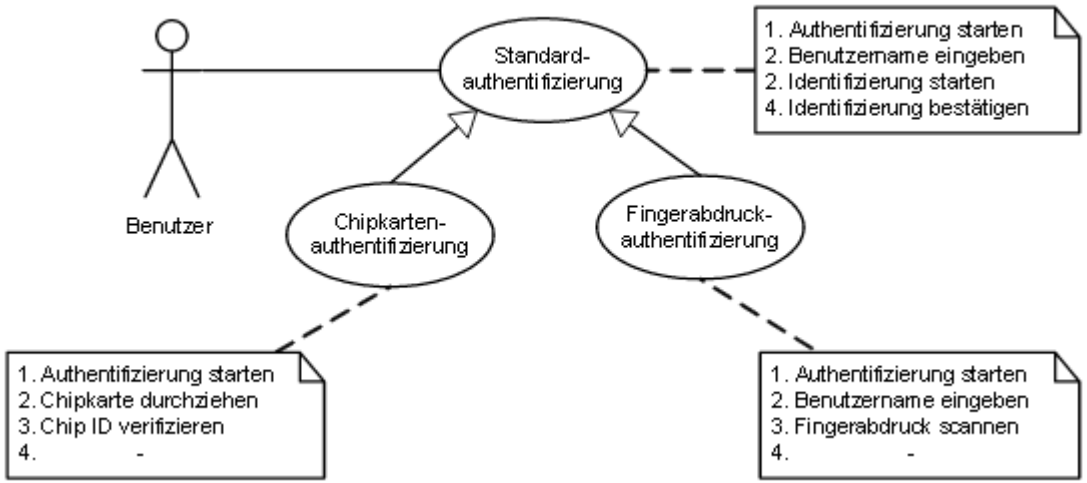
- Ein Anwendungsfall **ergänzt** einen anderen mit zusätzlichen Aktivitäten
- Eine extend-Beziehung von Anwendungsfall A zu Anwendungsfall B besagt, dass das Verhalten von A in B an einer Erweiterungsstelle eingefügt werden kann
- Für jede extend-Beziehung können **Erweiterungsstellen** (extension points) im zu erweiternden Anwendungsfall definiert werden, die angeben, wo der erweiternde Anwendungsfall einzufügen ist
- Die extend-Beziehung bedeutet, dass der zu erweiternde Anwendungsfall vom erweiterten Anwendungsfall **übernommen werden kann**, aber nicht notwendigerweise übernommen werden muss

Beziehungen zwischen Anwendungsfällen: **Extend**



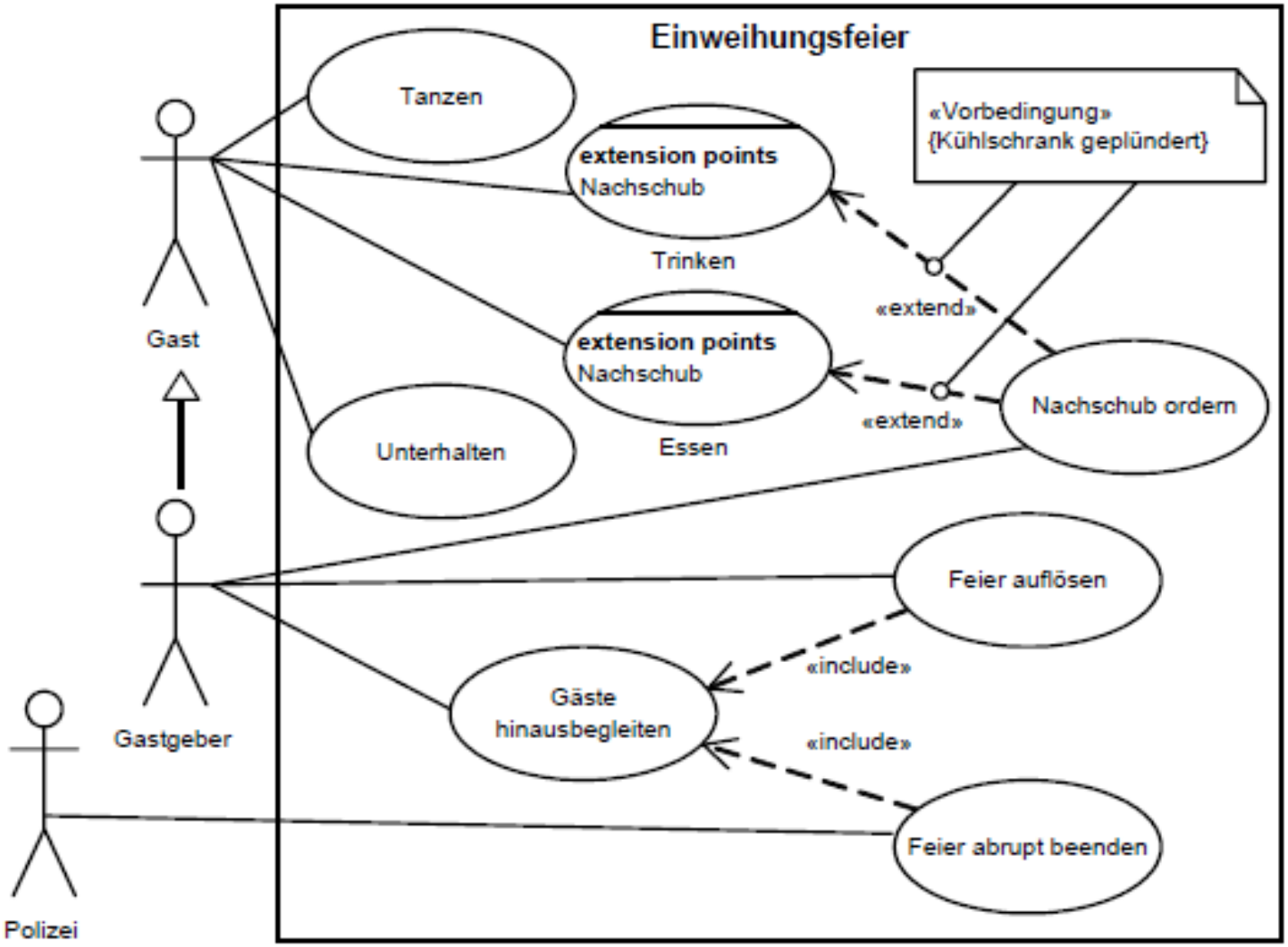
Beziehungen zwischen Anwendungsfällen: **Vererbung**

- Beschreibt die Generalisierung/Spezialisierung zwischen Anwendungsfällen, auch zwischen Akteuren
- Der erbende Anwendungsfall erbt das gesamte Verhalten des vererbenden Anwendungsfalls und kann dieses überschreiben und ergänzen
- Die Generalisierungsbeziehung beschreibt die Beziehung zwischen konkreten und abstrakten Anwendungsfällen



[UML Glasklar, Jeckle]

Anwendungsfalldiagramm: Beispiel



[UML Glasklar, Jeckle]

Übung

- Was beschreibt ein Kontextdiagramm?
- Was ist ein Stakeholder?
- Welche Rolle spielt ein Akteur?
- Was beschreibt ein Geschäftsprozess?
- Was ist ein Anforderungsfall (Use Case)?
- Welche Beziehungen kann es zwischen Anwendungsfällen geben?
- Was ist der Unterschied zwischen „include“ und „extend“?

7. Schritt: Anwendungsfälle ausarbeiten

- Anwendungsfälle textuell beschreiben
  - Typischerweise mit einer Spezifikationsschablone
  - Es sind unterschiedliche Schablonen im Einsatz
  - Eine Anpassung im Projektkontext ist empfehlenswert
- Benutzungsschnittstellen gestalten und textuell oder graphisch beschreiben
- Evtl. Prototypen zur Validierung implementieren
  - Üblicherweise werden Prototypen für Benutzungsschnittstellen implementiert
  - Prototypen zur Überprüfung funktionaler Aspekte können auch nützlich sein, sind aber oft zu teuer

Anwendungsfallbeschreibung 1/3: Pflichtfelder

- Das ist eine Beispielschablone, die in mehreren meiner Projekte eingesetzt wurde
- Sie beinhaltet Pflichtfelder und optionale Felder

Anwendungsfall	ID / Bezeichner des Anwendungsfalls
Name	Name des Anwendungsfalls
Initiierender Akteur	Der Systemteilnehmer, der den Anwendungsfall auslöst
Weitere Akteure	Weitere am Anwendungsfall beteiligte Systemteilnehmer, falls gegeben
Kurzbeschreibung	Kurzbeschreibung Eine kurze Zusammenfassung des Ablaufs

Anwendungsfallbeschreibung 2/3: Pflichtfelder

Vorbedingung	Vorbedingung Sachverhalt, der vor Beginn des Anwendungsfalls zugesichert werden muss. Vorbedingungen werden in einem Anwendungsfall nicht überprüft, sondern als gegeben hingenommen
Nachbedingung	Nachbedingung Sachverhalt, der mit dem Ende des Anwendungsfalls zugesichert wird
Ablauf	Schematische Beschreibung der einzelnen Vorgänge und Beteiligung der Akteure an diesen Vorgängen. Im Ablauf wird lediglich der zu erwartenden Normalablauf beschrieben
Alternativen	Abweichende Abläufe für einen korrekten Ablauf, falls gegeben
Ausnahmen	Abweichende Abläufe, die sich aus einem Fehlerfall ergeben, falls gegeben
Benutzte Anwendungsfälle	Eingebettete Anwendungsfälle, die von diesem Anwendungsfall verwendet, erweitert oder ererbt werden, falls vorhanden

Anwendungsfallbeschreibung 3/3: optionale Felder

Spezielle Anforderungen	Über Vorbedingungen hinausgehende oder spezielle Anforderungen (z.B. Sicherheitsvorgaben).
Annahmen	Im Rahmen einer weitergeführten Architekturfestlegung noch zu treffende Entscheidungen, die hier vorweggenommen werden müssen. Sind diese Annahmen nicht erfüllt, muss der Anwendungsfall im Allgemeinen komplett neu entworfen werden.
Offene Themen	Aspekte, die bei der Realisierung des Anwendungsfalls noch geklärt werden müssen, aber nicht als Entscheidung der Rahmenarchitektur gesehen werden
Referenzen	Verweise auf Gesetze und andere Quellen, auf die sich der Anwendungsfall stützt oder die er umsetzt
Datenanforderungen	Im Anwendungsfall beteiligte Daten, sowie durch diese bedingte Anforderungen an die Ausgestaltung des Anwendungsfalls
Nichtfunktionale Anforderungen	Anforderungen an den Anwendungsfall, die sich nicht auf den (logischen) Ablauf auswirken, aber starken Einfluss auf die Realisierung nehmen

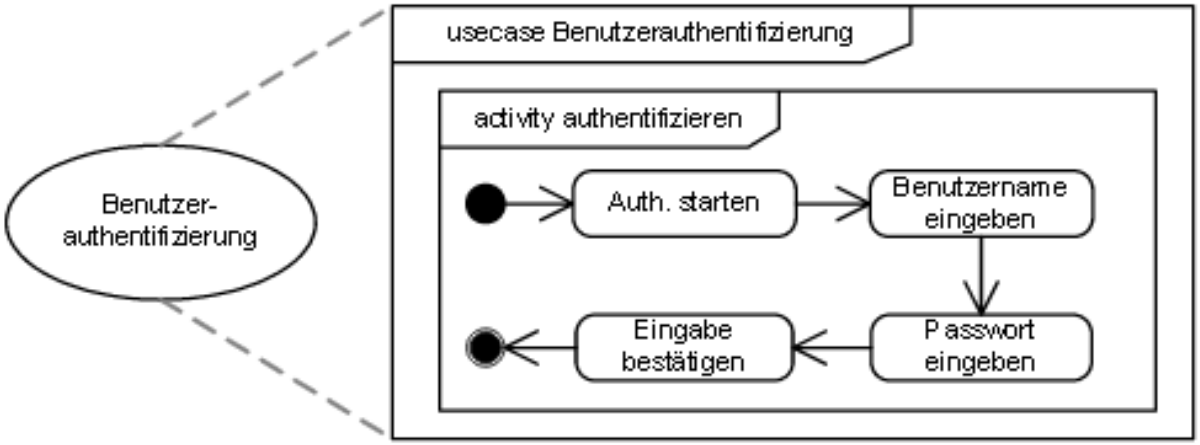
Beispiel Use Case: Benutzerauthentifizierung

Anwendungsfall	UC-1
Name	Benutzerauthentifizierung
Initiierender Akteur	Benutzer
Weitere Akteure	keine
Kurzbeschreibung	Ein Benutzer wird vom System authentifiziert
Vorbedingung	Benutzer hat ein Konto und ist nicht angemeldet
Nachbedingung	Der Benutzer ist im System angemeldet
Ablauf	1. Webseite aufrufen 2. Benutzername eingeben 3. Passwort eingeben 4. Eingabe bestätigen
...	

8. Schritt: Anwendungsfälle und Szenarien

- Ein Anwendungsfall wird durch eine Menge von Szenarien dokumentiert / verfeinert
- Ein Szenario ist ein exemplarischer Ablauf eines Use Cases
  - Jedes Szenario wird durch eine oder mehrere Bedingungen definiert, die zu einem speziellen Ablauf des Anwendungsfalles führen
  - Szenarien werden im Rahmen von Diskussionen durchgespielt um wichtige Szenarien herauszukristallisieren
- Folgende Kategorien von Szenarien werden in der Regel eingesetzt:
  - Szenarien, die eine erfolgreiche Bearbeitung des Anwendungsfalles beschreiben
    - Als Normalablauf
    - Als Alternativablauf
  - Szenarien, die zu einem Fehlschlag führen
- Die Szenarien werden danach im Entwurf durch Sequenzdiagramme oder Aktivitätsdiagramme spezifiziert

Beispiel Use Case: Benutzerauthentifizierung



[UML Glasklar, Jeckle]

- Normaler Ablauf (siehe Schablone):
  1. Webseite aufrufen
  2. Benutzername eingeben
  3. Passwort eingeben
  4. Eingabe bestätigen

9. Anforderungen aus den Use Cases erkennen

- Eine detaillierte Beschreibung eines Use Cases beinhaltet den Ablauf bzw. die notwendige Funktionalität, ggf. Eigenschaften um den Use Case durchführen zu können
- → **Ergebnis: Funktionale und nicht-funktionale Anforderungen**

Beispiel Use Case: Benutzerauthentifizierung

- Normaler Ablauf (siehe Schablone):
  1. Webseite aufrufen
  2. Benutzername eingeben
  3. Passwort eingeben
  4. Eingabe bestätigen

- Anforderung aus Schritt 1: Der Benutzer soll die Webseite aufrufen können
- Anforderung aus Schritt 2: Der Benutzer muss Benutzername und Passwort eingeben können
- …

Anforderungstabelle

Anforderungs-Nr.	Beschreibung	Use Case	Verifikations-methode
FA-1	Der Benutzer soll die Webseite aufrufen können	UC-1	T
FA-2	Der Benutzer soll sein Benutzername und Passwort eingeben können	UC-1	T
FA-3	Das System soll den Benutzernamen und das Passwort überprüfen	UC-1	T
FA-4	Der Benutzername und das Passwort sollen persistent gespeichert werden	UC-1	I, RoD
…			

10. Benutzungsschnittstelle zusammenstellen

- Dialoge
  - Bearbeitungseinheit des Anwenders
  - Jeder Dialog unterstützt einen oder mehrere Anwendungsfälle
  - Jeder Dialog besitzt ein oder mehrere Formulare (Maske, Fenster)
- Benutzungsschnittstelle: Gesamtheit aller Dialoge
- Batch-Verarbeitung
- Druckausgaben
- **Ergebnis:** Benutzungsschnittstellen-Anforderungen

11. Daten- und Funktionsmodell definieren

- Datenmodellierung
  - Zentrale Begriffe der Anwendung (z.B. Kunde, Konto, Bestellung) als Entitätstypen
  - Attribute, die die Entitäten beschreiben (z.B. Kunde hat eine Rechnungsadresse und eine Lieferadresse)
  - Beziehungstypen/Abhängigkeiten zwischen den Entitäten
- Funktionsmodell
  - Beschreibt dynamische Aspekte, wie Anwendungsfälle/Szenarien und dazu notwendige Algorithmen
  - Stellt die Abhängigkeiten zwischen den Funktionen bzw. Funktionseinheiten dar
- **Ergebnis:** Objektmodell-Anforderungen

12. Nachbarsysteme identifizieren

- Welche Nachbarsysteme gibt es?
  - Schnittstellen zu den Nachbarsysteme identifizieren
  - Datenfluss (Datenquelle und Datensenke) ermitteln
- Zu jedem Nachbarsystem folgende Angaben:
  - Organisatorische Zuständigkeiten
  - Anpassungen des Nachbarsystems
  - Art der Kopplung
  - Datentransformation
  - Frequenz und Art der Kommunikation
- **Ergebnis:** Schnittstellenanforderungen, Randbedingungen

Übung

- Was soll eine Anwendungsfallbeschreibung beinhalten?
- Was kann man aus den Anwendungsfallbeschreibungen herleiten?
- Was beschreiben alle Anwendungsfälle zusammen?
- Was beinhaltet eine Benutzungsschnittstelle?
- Was beinhaltet das Objektmodell?
- Welche Anforderungen kann man durch die Analyse der Nachbarsysteme identifizieren?

Übung

- Erstellen Sie ein Use Case Diagramm für den Geschäftsprozess:

  "Online-Überweisung"

- Beschreiben Sie einen der darin enthaltenen Anwendungsfälle mit der Schablone aus der Vorlesung

- Definieren Sie die wichtigsten fünf daraus resultierenden Anforderungen

Ergebnis der Anforderungsanalyse: Pflichtenheft

- Erstellt durch Auftragnehmer auf Basis des Lastenhefts
  - von allen am Projekt Beteiligten akzeptiert
- Beinhaltet alle identifizierten Anforderungen und Rahmenbedinungen
- Essentiell
  - Positive Abgrenzungen (Ziele): was das Produkt können wird
  - Abgrenzungen (Nicht-Ziele): was das Projekt nicht können wird
- Vorteile
  - → Klare Aussage über die Erfüllung der Leistungen
  - → Diskussionsfreie Produktabnahme
- Genug Zeit für ein gutes Pflichtenheft investieren
  - → Klare Vorgaben führen sicher zum Zeitersparnis im Projektverlauf

Pflichtenheft Inhalt

- Einordnung
  - Umfeld, Ausgangsituation, Ziele etc.
  - Projektstruktur
  - Abweichung: Pflichtenheft vs. Lastenheft
- Voraussetzungen
  - Hardware-/Software-Umgebung, Entwicklungsmittel, Fremdprodukte
- Randbedingungen
  - Technische, terminliche, vertragliche etc. Einschränkungen
- Anforderungen
  - Funktionale, Qualitäts-, Schnittstellen-, Bedienbarkeitsanforderungen
  - Konfigurationen, Ausbaustufen, Varianten, Grenzen
- Lieferumfang - Liste der zu liefernden HW-/SW- Einheiten
- Funktionsprüfung/Abnahmekriterien

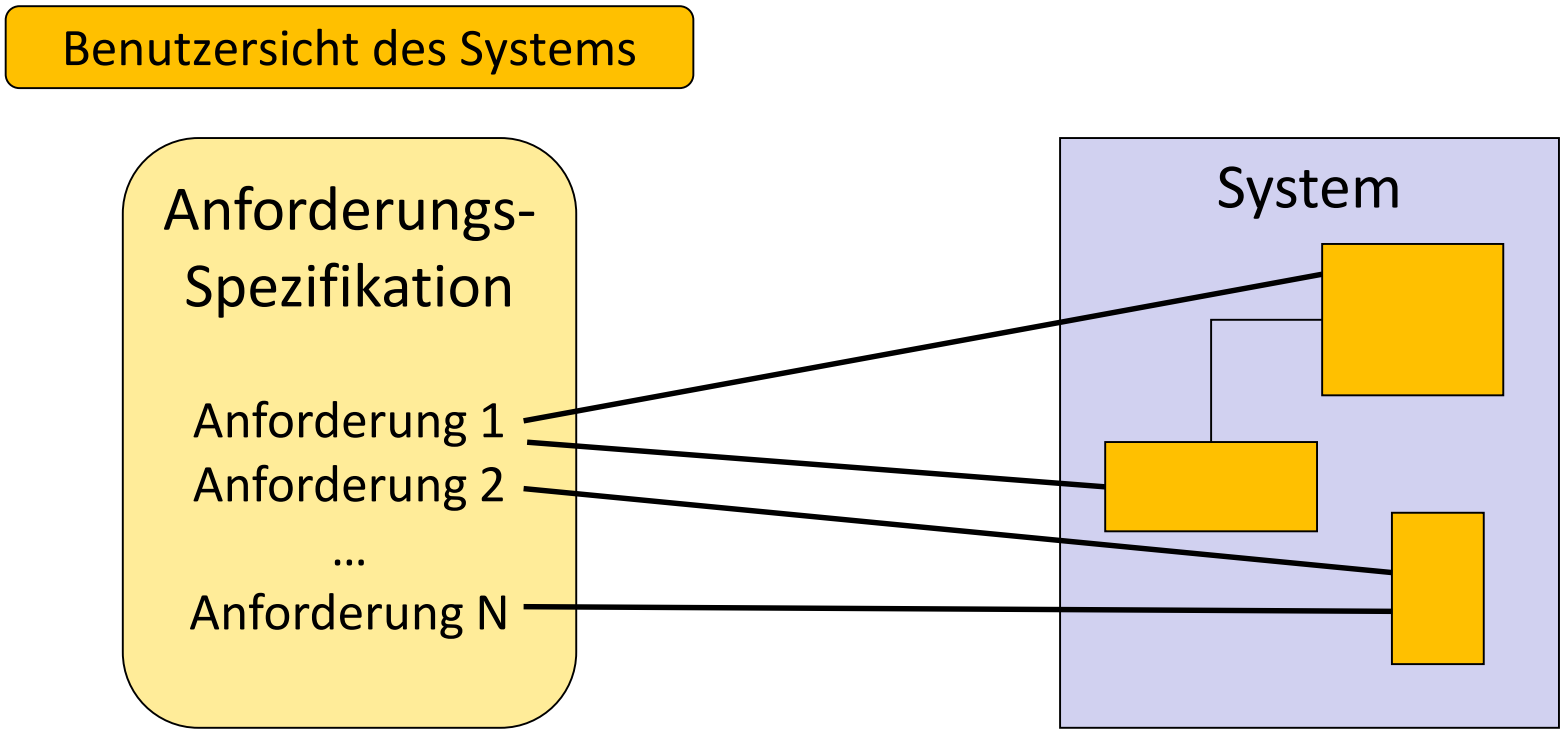
Agile Praktiken: Product Backlog Inhalt

- Geordnete Liste von Anforderungen
  - Produktanforderungen
  - User Stories, Features, Funktionen
  - Verbesserungen, Fehlerkorrekturen
- Reihenfolge richtet sich nach
  - Wert
  - Risiko
  - Priorität
  - Notwendigkeit
- Einträge werden in jeder Iteration (z.B. Sprint) verfeinert
- Daraus wird ein Sprint Backlog für die nächste Iteration extrahiert
- Der Produkt Owner ist für das Product Backlog, seine Inhalte, Bereitstellung und Reihenfolge verantwortlich

Pflichtenheft/Backlog und dann?

- Das Pflichtenheft (oft zusammen mit einem Angebot) ist die **vertragliche Grundlage** für das Projekt
  - Laufzeit
  - Preis
  - Wartung / Gewährleistung
  - Sanktionen
- Das Pflichtenheft ist aber auch die **technische Grundlage** für das Projekt
  - Anforderungen werden ggf. auf Subsystemebene verfeinert und in konkrete technische Spezifikationen umgesetzt
  - Interne und Externe Subsystem- und System-Schnittstellen werden in entsprechende Schnittstellendokumente (Interface Control Document) definiert
  - Diese Spezifikationen werden als Grundlage für den Systemarchitekturentwurf und Subsystementwurf eingesetzt

Von Anforderungen zum System



Mehr zum dazu in den nächsten Vorlesungen und in Softwaretechnik 2

Zusammenfassung - Was Sie beantworten können sollten

- Warum ist die Ermittlung der Anforderungen schwierig?
- Das Ergebnis einer Anforderungsanalyse ist …
- Ein Lastenheft ist …
- Ein Pflichtenheft ist …
- Die Anforderungen sollen folgenden Kriterien genügen: ...
- Ermittlung der Anforderungen kann aufgeteilt werden in ...
- Anwendungsfälle/Use Cases sind ...
- Wofür stellt ein Pflichtenheft die Grundlage?
- **+ alle Fragen in den Übungsfolien!**

Literaturhinweise

- **Primär: Materialien im AULIS!**
- I. Sommerville - Software Engineering, 2012
- C. Rupp, S. Queins, die SOPHISTen - UML 2 glasklar: Praxiswissen für die UML-Modellierung, 2012
- E. Gamma, R. Helm, R. E. Johnson - Design Patterns. Elements of Reusable Object-Oriented Software, 1996
- P. Clements - Documenting Software Architectures: Views and Beyond, 2003
- L. Bass, P. Clements, R. Kazman - Software Architecture in Practice, 2012
- ISO/IEC/IEEE 15288:2008 - Systems and software engineering -- System life cycle processes
- ISO/IEC/IEEE 16326:2009 - Systems and software engineering -- Life cycle processes -- Project management
- ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models
- A. Spillner, T. Linz - Basiswissen Softwaretest, 2019
- M. Broy, M. Kuhrmann - Projektorganisation und Management im Software Engineering, 2013
- S. Röpstorff, R. Wiechmann - Scrum in der Praxis, 2016