

# Requirements Engineering

Dozent: Florian Glufke

# Requirements Engineering

- Was sind Anforderungen
  - Spezifikation was Software an Eigenschaften und Funktionen haben soll
  - Ergeben sich aus Wünschen und Bedürfnissen der Stakeholder
- Was ist Requirements Engineering
  - Systematisches Erfassen und Verwalten von Anforderungen
  - Ergebnis ist eine Anforderungsspezifikation

# Requirements Engineering

- Wieso Requirements Engineering
  - Minimierung des Risikos eine Software zu liefern die nicht den Kundenwünschen entspricht
  - Minimierung des Risikos kostspieliger Änderungen in späteren Entwicklungsphasen
  - Grundlage für die Einschätzung vom Entwicklungsaufwand
  - Voraussetzung, um das System richtig zu testen
  - Besseres Verständnis des Problems

# Requirements Engineering

- Kosten steigen je nachdem wann ein Fehler entdeckt wird
- Kosten für Fehlerbehebung Faktor 100 - 1.000 höher wenn im Einsatz entdeckt, statt bei Anforderungsanalyse

# Requirements Engineering

- Mangelhaftes Requirements Engineering durch
  - Direkt mit der Implementierung beginnen
  - Kommunikationsprobleme zwischen den beteiligten Parteien
  - Annahme, dass die Anforderungen selbstverständlich sind
  - Fehlende Ausbildung und Fähigkeiten

# Requirements Engineering

- Arten von Anforderungen:
  - Funktionale Anforderungen
    - Was das System tun soll
  - Nicht-Funktionale Anforderungen
    - Qualitätsanforderungen

# Requirements Engineering

- Aufgaben beim Requirements Engineering
  - Ermitteln
  - Dokumentieren
  - Validieren
  - Verwalten

# Requirements Engineering

- Faktoren die Requirements Engineering beeinflussen
  - Systementwicklungsprozess (linear vs. Iterativ)
  - Beziehung zwischen dem Lieferanten und dem Kunden
  - Verfügbarkeit und Fähigkeit des Kunden

# Requirements Engineering

- Stakeholder
  - Haben ein bestimmtes Interesse an dem System
  - Können sein:
    - Benutzer
    - Kunde
    - Auftraggeber
    - Betreiber
    - Regulierungsbehörde
    - Entwickler
  - Einbeziehung der richtigen Personen in die entsprechenden Stakeholder-Rollen wichtig für erfolgreiches Requirements Engineering

# Requirements Engineering

- Stakeholder haben unterschiedliche Interessen
  - Kosten vs. Funktionalität
- Einschätzen wie wichtig einzelne Stakeholder sind
- Inkonsistenzen in den Anforderungen finden
- Konflikte zwischen Stakeholdern lösen

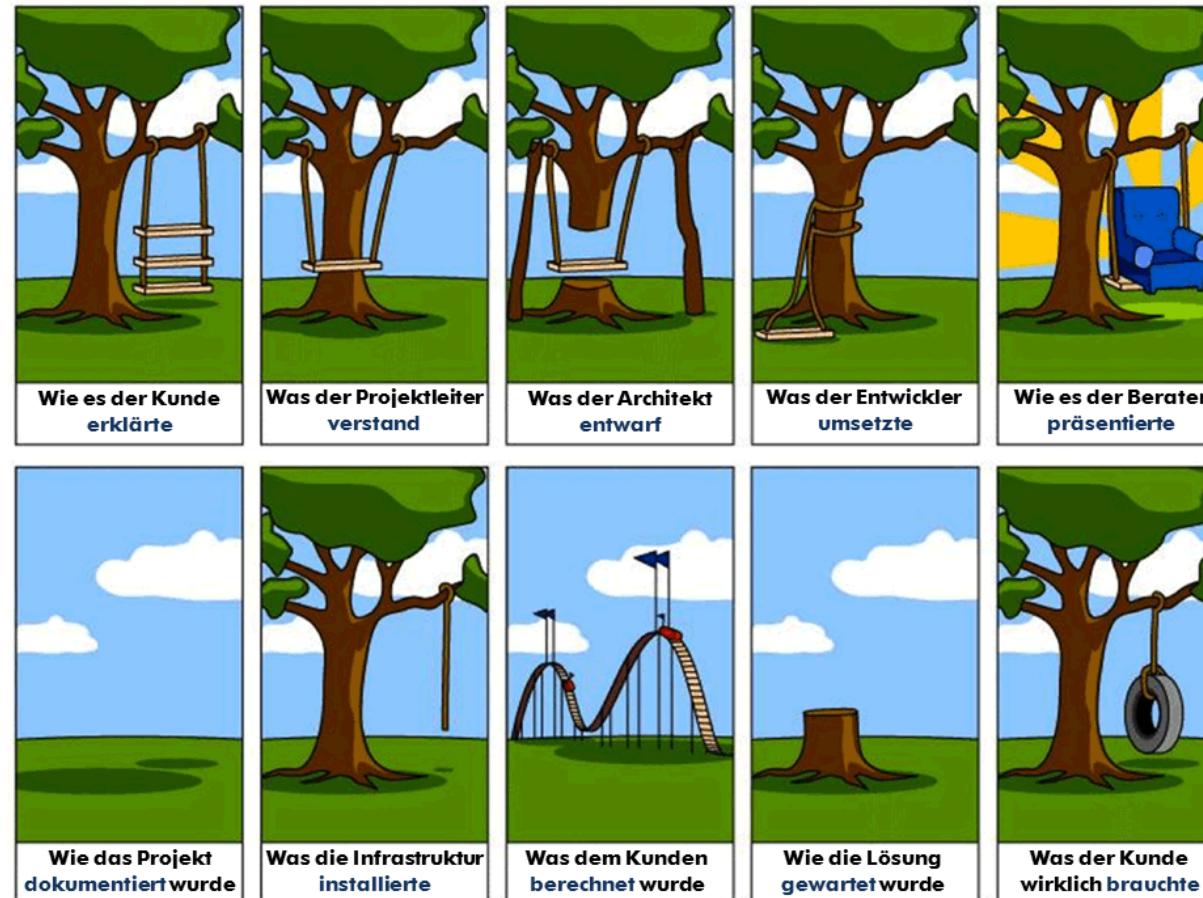
# Requirements Engineering

- Requirements Engineering sorgt für gemeinsames Verständnis
- Explizites gemeinsames Verständnis
  - Schriftlich dokumentierte Anforderungen
- Implizites gemeinsames Verständnis
  - gemeinsames Wissen über Bedürfnisse, Visionen, Kontext

# Requirements Engineering

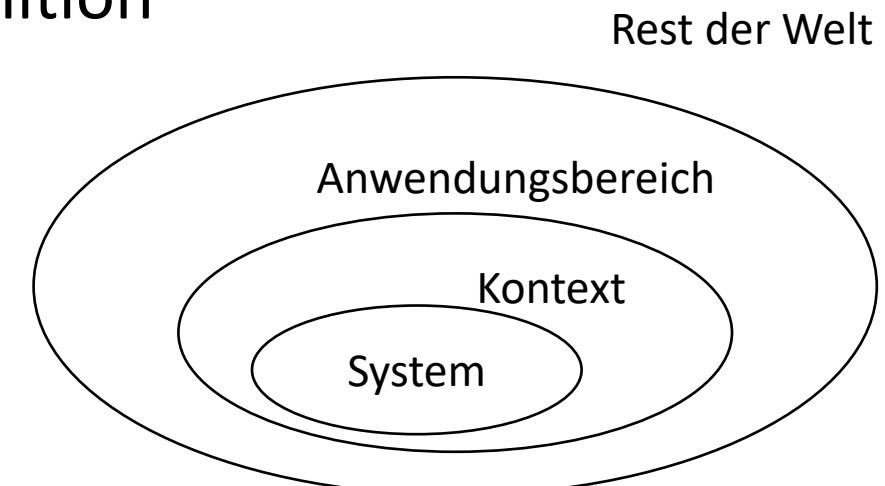
- Gemeinsames Verständnis fördern durch
  - Erstellung von Glossaren
  - Erstellen von Prototypen
  - Verwendung eines bestehenden Systems als Bezugspunkt

# Requirements Engineering



# Requirements Engineering

- Verständnis über Systemkontext ist wichtig
- Systemkontext
  - Umgebung des Systems welche relevant für Verständnis und Anforderungen ist
- Klärung der Systemgrenze und die Definition der externen Schnittstellen



# Requirements Engineering

- Ziel im Requirements Engineering
  - Anforderungen festzuhalten und nicht deren Lösung
    - Schwierig zu Trennen
  - Was soll das System tun (Anforderung)
  - Wie soll das System etwas tun (Lösung)
  - Alternativ: Welche Entscheidungen können ohne Zustimmung der Stakeholder getroffen werden (Lösung)

# Requirements Engineering

- Anforderungen validieren
  - Entsprechen die dokumentierten Anforderungen den Wünschen des Stakeholders
  - Sind sich alle Stakeholder über die Anforderungen einig
  - Sind die Kontextannahmen korrekt

# Requirements Engineering

- Anforderungen ändern sich ständig
- Änderungen durch z.B.:
  - Stakeholder ändert seine Meinung
  - Geschäftsprozesse ändern sich
  - Fehler in den Anforderungen
- Im Requirements Engineering müssen diese Änderungen beachtet werden
- Anforderungen möglichst stabil halten

# Requirements Engineering

- „Wenn ich die Menschen gefragt hätte, was sie wollen, hätten sie gesagt schnellere Pferde.“ – Henry Ford
- Requirements Engineering ist nicht nur bloßes Aufschreiben der Kundenwünsche
- Stakeholder wissen oft nicht was alles möglich ist

# Requirements Engineering

- Anforderungen müssen systematisch und diszipliniert erfasst werden
- Geeignete Prozesse und Praktiken zur systematischen Ermittlung
- Keine allgemeingültigen Prozesse
- Prozess muss zum Problem passen

# Requirements Engineering

- Traditionelles Requirements Engineering:
  - Erstellung einer umfassenden, vollständigen und eindeutigen Anforderungsspezifikation
- Kosten können Nutzen übersteigen
- Je nach Umfeld ist es fraglich ob das erforderlich ist
  - Bei z.B. iterativen Prozessen eher nicht
  - Bei z.B. Ausschreibungen erforderlich

# Requirements Engineering

- Arbeitsprodukte im Requirements Engineering unterscheiden sich durch:
  - Zweck
    - Welchen Nutzen erfüllt das Arbeitsprodukt
  - Umfang
    - Einzelter Satz oder ganzes Anforderungsdokument
  - Darstellung
    - Natürliche Sprache, Diagramm, Zeichnung, GUI-Prototyp
  - Lebensdauer
    - z.B. händische Zeichnung oder Anforderungsspezifikation
  - Speicherung
    - z.B. Papier, Excel-Tabelle, Requirements Engineering Softwaresystem

# Requirements Engineering

- Detailierungsgrad ist abhängig von:
  - Zusammenarbeit mit dem Kunden
  - Verständnis über Problem und der Projektkontext
  - Grad des gemeinsamen Verständnisses des Problems
  - Freiheitsgrad von Designern und Entwicklern
  - Verfügbarkeit von schnellem Stakeholder-Feedback
- Abwägung von Risiko und Kosten
  - Detaillierte Anforderungen: niedriges Risiko und hohe Kosten

# Requirements Engineering

- Verschiedene Aspekte die bei Anforderungen beachtet werden:
  - Daten und Struktur
  - Funktionen und Ablauf
  - Zustand und Verhalten
  - Qualitätsanforderungen
    - Benutzerfreundlichkeit
    - Zuverlässigkeit
    - Verfügbarkeit
    - Leistungsanforderungen

# Requirements Engineering

- Leistungsanforderungen
  - Zeit (z.B. für die Ausführung einer Aufgabe oder die Reaktion auf externe Ereignisse)
  - Volumen (z.B. erforderliche Datenbankgröße)
  - Frequenz (z.B. der Berechnung einer Funktion oder des Empfangs von Signalen von Sensoren)
  - Durchsatz (z.B. Datenübertragungs- oder Transaktionsraten)
  - Ressourcenverbrauch (z.B. CPU, Speicher, Bandbreite, Batterie)
  - Genauigkeit einer Berechnung

# Requirements Engineering

- Leistungsanforderungen können einfach definiert werden
  - Z.B. Reaktionszeit bei Benutzereingabe maximal 3 Sekunden
  - Oder Verfügbarkeit 99%
  - Quantitative Darstellungen
- Qualitätsanforderungen sind schwierig zu definieren
  - Z.B. „das System soll einfach zu bedienen sein“
  - Können nur schwer quantifiziert werden
  - Operationalisierte Darstellungen
    - Qualitätsanforderung in Form von funktionalen Anforderungen
    - Z.B. Anforderung zur Datensicherheit in Form einer Anmeldefunktion und Einschränkungen des Datenzugriffs ausgedrückt

# Requirements Engineering

- „Nur eine quantifizierte Qualitätsanforderung ist eine gute Qualitätsanforderung“
  - Ist veraltet und hat hohen Aufwand
  - Eher risikobasierter Ansatz verwenden
- Qualitative Darstellungen von Qualitätsanforderungen sind ausreichend wenn:
  - Implizites gemeinsames Verständnis vorhanden
  - Bekannte Lösung als Referenz verwendet wird
  - Vertrauen, dass die Entwickler das korrekt machen
  - Kurze Rückkopplungsschleifen sind vorhanden

# Requirements Engineering

- Bei sicherheitskritischen Qualitätsanforderungen eine vollständig quantifizierte Darstellung verwenden
  - Application Security Verification Standard (ASVS) von Open Web Application Security Project (OWASP) kann hier als Grundlage verwendet werden
  - Definiert z.B. Anforderungen an Passwortsicherheit
  - Definiert drei Sicherheitsstufen (Stufe 1, 2 und 3)
  - Anforderung kann dann einfach sein, dass ASVS Stufe 2 erfüllt sein muss

# Requirements Engineering

- Bei Spezifizierung von Randbedingungen folgendes beachten
  - Technisch: Vorgegebene Schnittstellen oder Protokolle, Komponenten oder Frameworks
  - Rechtlich: Einschränkungen durch Gesetze, Verträge, Normen oder Vorschriften
  - Organisatorisch: Randbedingungen in Bezug auf Organisationsstrukturen, Prozesse oder Richtlinien
  - Kulturell: Benutzergewohnheiten und -erwartungen werden bis zu einem gewissen Grad von der Kultur geprägt

# Requirements Engineering

- Bei Spezifizierung von Randbedingungen folgendes beachten
  - Umweltbezogen: Umgebungsbedingungen wie Temperatur, Feuchtigkeit, Strahlung oder Vibrationen
  - Physikalisch: Wenn ein System physikalische Komponenten umfasst oder mit ihnen interagiert, wird das System durch die Gesetze der Physik eingeschränkt

# Requirements Engineering

- Allgemeine Richtlinien beim Erstellen von Arbeitsprodukten
  - Typ von Arbeitsprodukten auswählen, der den beabsichtigten Zweck erfüllt
  - Vermeidung von Redundanz, nicht Wiederholen sondern verweisen
  - Vermeiden von Inkonsistenzen zwischen Arbeitsprodukten
  - Begriffe konsistent verwenden (wie im Glossar definiert)
  - Strukturierte Arbeitsprodukte

# Requirements Engineering

- Planen der Arbeitsprodukte
  - In welchen Arbeitsprodukten sollen die Anforderungen erfasst werden
  - Welche Abstraktionsebenen sind zu berücksichtigen
  - Welcher Detaillierungsgrad soll verwendet werden
  - Darstellung der Arbeitsprodukte (z.B. natürliche Sprache)

# Requirements Engineering

- Frühe Definition der Arbeitsprodukte:
  - Hilft, Aufwand und Ressourcen zu planen
  - Stellt sicher, dass geeignete Notationen verwendet werden
  - Stellt sicher, dass alle Ergebnisse im richtigen Arbeitsprodukten erfasst werden
  - Stellt sicher, dass keine größere Umstrukturierung von Informationen erforderlich ist
  - Hilft, Redundanz zu vermeiden

# Requirements Engineering

- Anforderungen mit natürlicher Sprache erfassen
  - Zentrales Mittel zur Kommunikation von Anforderungen
  - Ist ausdrucksstark und flexibel
  - Kann Anforderung in all ihren Aspekten ausdrücken
  - Wird von allen verstanden

# Requirements Engineering

- Herausforderungen bei natürlicher Sprache
  - Ist oft mehrdeutig und unpräzise
  - Schwierig Mehrdeutigkeiten, Auslassungen und Inkonsistenzen zu erkennen
  - Problematik lässt sich durch Befolgen von Regeln verringern

# Requirements Engineering

- Regeln bei Anforderungen in natürlicher Sprache
  - Kurze und gut strukturierte Sätze
  - Eine einzige Anforderung in einem Satz
  - Satzschablonen verwenden
  - Gut strukturierte Arbeitsprodukte
  - Verwendung von hierarchischer Struktur aus Teilen, Kapiteln, Abschnitten und Unterabschnitten
  - Definieren und Verwenden von konsistent einheitlicher Terminologie
  - Erstellung und Verwendung eines Glossars

# Requirements Engineering

- Regeln bei Anforderungen in natürlicher Sprache:
  - Vermeiden ungenauer oder mehrdeutiger Begriffe und Phrasen
  - Bekannte Fallstricke vermeiden
    - Unvollständige Beschreibungen. Verben haben Platzhalter, die ausgefüllt werden sollten.
      - Z.B. das Verb „geben“ hat drei Platzhalter: wer wem was gibt.
    - Unspezifische Substantive
      - Z.B. die Daten oder der Benutzer
    - Unvollständige Bedingungen. Nicht nur Normalfall beschreiben sondern auch Ausnahmefälle
    - Unvollständige Vergleiche. Bei Vergleichen sollte immer ein Bezugsobjekt angegeben werden. Z.B. „schneller als 0,1 ms“

# Requirements Engineering

- Regeln bei Anforderungen in natürlicher Sprache
  - Bekannte Fallstricke vermeiden (Fortsetzung)
    - Passive Formulierung. Sätze im Passiv haben kein Subjekt. Nicht deutlich beschrieben wer für die Handlung verantwortlich ist
      - Z.B. „Es soll eine Rechnung erstellt werden können“
      - Besser: „Der angemeldete Benutzer soll eine Rechnung erstellen können“
    - Universalquantoren. Wörter wie alle, immer oder nie.
      - Sollten immer hinterfragt werden und ggf. die Bedingungen genauer definieren

# Requirements Engineering

- Natürliche Sprache ist mächtiges Mittel, um Anforderungen zu formulieren, wenn
  - Bekannte Fallstricke vermieden werden
  - Schreibregeln befolgt werden

# Requirements Engineering

- Satzschablonen für Anforderungen
  - [<Bedingung>] <Subjekt> <Aktion> <Objekte> [<Einschränkung>] (nach ISO/IEC/IEEE 29148)
  - Beispiel: „Wenn eine gültige Karte erkannt wird, soll das System die Meldung „Geben Sie Ihre PIN ein“ auf dem Dialogbildschirm anzeigen, und zwar innerhalb von 200 ms“
  - Verwendung von Hilfsverben
    - „Muss“ bezeichnet obligatorische Anforderung
    - „Sollte“ bezeichnet eine Anforderung, die nicht obligatorisch, aber stark erwünscht ist
    - „Kann“ kennzeichnet einen Vorschlag
    - „Wird“ (oder die Verwendung eines Verbs im Präsens ohne eines der oben erwähnten Hilfsverben) bezeichnet eine Sachaussage, die nicht als Anforderung gilt

# Requirements Engineering

- Weitere Satzschablonen:
  - Für allgegenwärtige Anforderungen:
    - Das <Systemname> soll <Systemantwort>
      - Z.B. „Das System soll die Größe der Logdateien auf 10 MB begrenzen.“
  - Für ereignisgesteuerte Anforderungen (ausgelöst durch externes Ereignis):
    - SOBALD | NACHDEM <optionale Vorbedingungen> <Auslöser> soll das <Systemname> <Systemantwort>.
  - Für unerwünschtes Verhalten (Beschreibung zu vermeidenden Situationen):
    - WENN <optionale Vorbedingungen> <Auslöser>, DANN soll das <Systemname><Systemreaktion>

# Requirements Engineering

- Noch mehr Satzschablonen:
  - Zustandsorientierte Anforderungen (gelten nur in bestimmten Zuständen):
    - SOLANGE <in einem bestimmten Zustand> soll das <Systemname> <Systemantwort>
- Satzschablone für User Stories:
  - Als <Rolle> möchte ich <Anforderung>, damit <Nutzen>
  - Beispiel: „Als Buchhalter möchte ich die Umsätze des aktuellen Monats angezeigt bekommen, damit ich die Finanzen meines Unternehmens im Blick habe“

# Requirements Engineering

- Formulare für Anforderungen
  - Einsatz von Formularen, die ausgefüllt werden um Anforderungen zu definieren
  - Formularvorlage für Use Cases

Name	< Eine kurze aktive Verbphrase>
Vorbedingung	<Bedingung(en), die erfüllt müssen, wenn die Ausführung des Use Case ausgelöst wird>
Erfolgs-Endbedingung	<Zustand nach erfolgreichem Abschluss des Use Cases>
Fehler Endbedingung	<Zustand bei fehlgeschlagener Ausführung des Use Cases>
Haupt Akteur	<Name Akteur>
Andere Akteure	<Liste der anderen beteiligten Akteure, falls vorhanden>

# Requirements Engineering

Auslöser	<Ereignis, das die Ausführung des Use Cases einleitet>
Normaler Ablauf	<Beschreibung des Haupt-Erfolgsszenarios in einer Abfolge von Schritten: <Schritt 1> <Aktion 1> <Schritt 2> <Aktion 2> ... <Schritt n> <Aktion n> ... >
Alternative Abläufe	<Beschreibung von alternativen oder Ausnahmeschritten, mit Verweisen auf die entsprechenden Schritte im normalen Ablauf>
Erweiterungen	<Erweiterungen zum normalen Ablauf (falls vorhanden), mit Verweisen auf die erweiterten Schritte im normalen Ablauf>
Verwandte Informationen	<Optionales Feld für weitere Informationen, wie z.B. Leistung, Häufigkeit, Beziehung zu anderen Use Cases, etc.>

# Requirements Engineerin

- Formularvorlage für Qualitätsanforderungen

Vorlage	Beispiel
<b>ID</b> <Nummer der Anforderung>	A137.2
<b>Ziel</b> <Qualitativ erklärtes Ziel>	Zimmerreservierungen sofort bestätigen
<b>Maßstab</b> <Maßstab zur Messung der Anforderung>	Verstrichene Zeit in Sekunden
<b>Messverfahren</b> <Verfahren zur Messung der Anforderung>	Zeitstempelung der Momente, in denen der Benutzer auf die Schaltfläche „Reservieren“ drückt und wenn die Anwendung die Bestätigung angezeigt hat. Messung der Zeitdifferenz.

# Requirements Engineering

Vorlage	Beispiel
<b>Minimum</b> <i>&lt;Mindestens zu erreichende akzeptable Qualität&gt;</i>	Weniger als 5 s in mindestens 95% aller Fälle
<b>OK-Bereich</b> <i>&lt;Wertebereich, der OK ist und angestrebt wird&gt;</i>	Zwischen 0,5 und 3 s in mehr als 98% aller Fälle
<b>Gewünscht</b> <i>&lt;Qualität erreicht im bestmöglichen Fall&gt;</i>	Weniger als 0,5 s in 99,9% aller Fälle

# Requirements Engineering

- Dokumentationsstruktur
  - Anforderungen müssen strukturiert abgelegt werden
  - Dokumentvorlagen können hierfür verwendet werden
  - Vorlagen gibt es in Literatur und Normen:
    - ISO/IEC/IEEE29148: Systems and software engineering - Life cycle processes - Requirements Engineering
    - Mastering the Requirements Process: Getting Requirements Right von Suzanne Robertson und James Robertson
  - Produkt-Backlog und Sprint-Backlog können hierfür auch verwendet werden

# Requirements Engineering

- Einsatz von Dokumentenvorlagen
  - Helfen bei der systematischen Strukturierung von Anforderungsdokumenten
  - Bieten klare, wiederverwendbare Struktur
  - Einheitliches Aussehen und Verbessern so die Lesbarkeit
  - Relevantesten Informationen zu erfassen und weniger Auslassungsfehler
  - Achtung: Ggf. Vernachlässigung von Themen die nicht in die Vorlage passen

# Requirements Engineering

- Beispiel Dokumentvorlage:

## **Teil I: Einführung**

1. Zweck des Systems
2. Umfang der Systementwicklung
3. Stakeholder

## **Teil II: Systemübersicht**

4. System Vision und Ziele
5. Systemkontext und -grenze
6. Gesamtstruktur des Systems
7. Merkmale der Benutzer

## **Teil III: Systemanforderungen**

Hierarchisch nach der Systemstruktur organisiert, unter Verwendung eines hierarchischen Nummerierungsschemas für Anforderungen

Pro Subsystem/Komponente:

- Funktionale Anforderungen (Struktur und Daten, Funktion und Ablauf, Zustand und Verhalten)
- Qualitätsanforderungen
- Randbedingungen
- Schnittstellen

## **Referenzen**

## **Anhänge**

Glossar (falls nicht als eigenständiges Arbeitsprodukt verwaltet)  
Annahmen und Abhängigkeiten

# Requirements Engineering

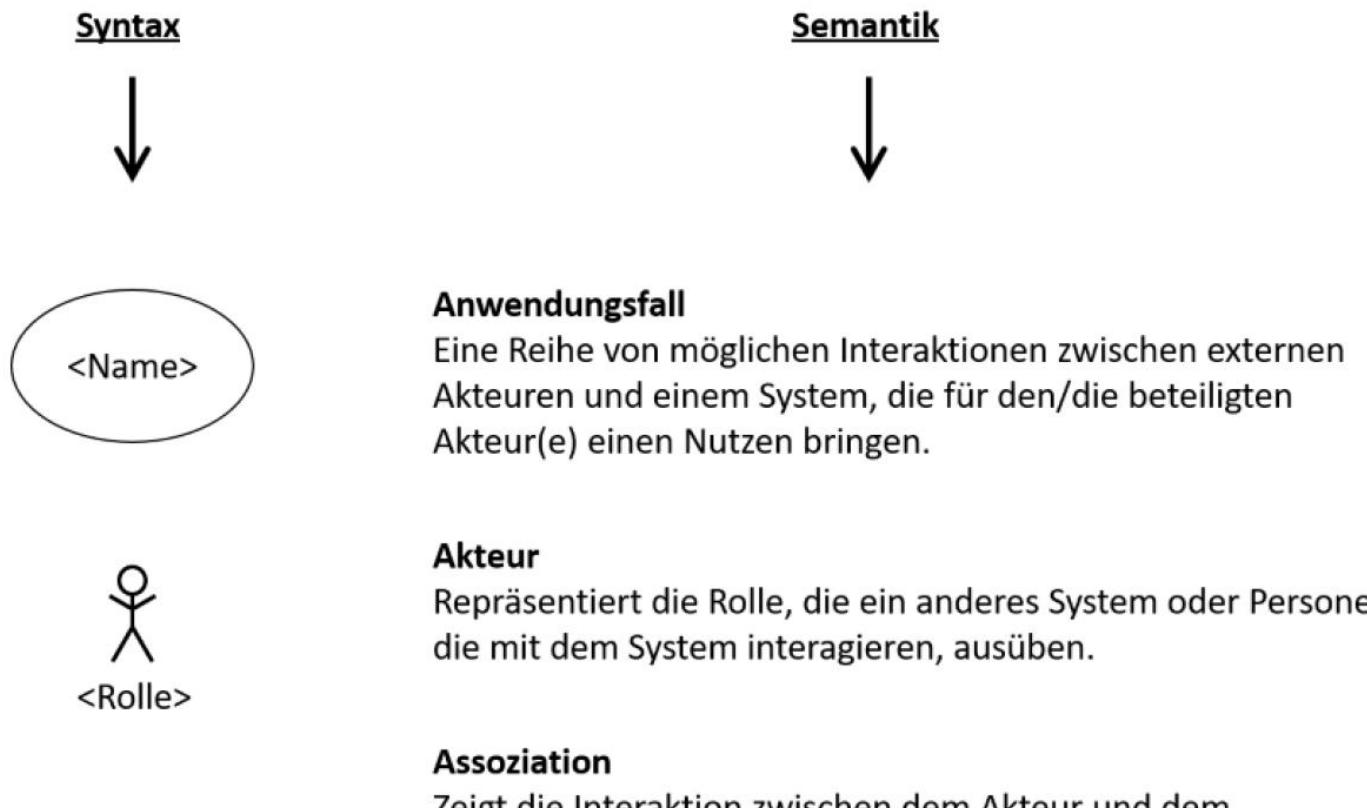
- Einsatz von Modellen
  - Natürliche Sprache hat ihre Grenzen und Nachteile
  - Bei umfangreichen und komplexen Anforderungen besser als natürliche Sprache
  - Modell ist eine abstrakte Darstellung der Wirklichkeit
  - Schematische Darstellung eines Modells wird als Diagramm bezeichnet
  - Erfordert Verständnis über das Diagramm
  - Passendes Modell muss gefunden werden

# Requirements Engineering

- Einsatz von Modellen
  - Modelle haben eine Syntax und eine Semantik
  - Syntax beschreibt, welche Notationselemente (Symbole) verwendet werden
  - Und wie diese Notationselemente in Kombination verwendet werden
  - Semantik definiert die Bedeutung der Notationselemente
  - Und legt die Bedeutung der Kombination von Elementen fest

# Requirements Engineering

- Beispiel UML  
Use Case Diagramm



# Requirements Engineering

- Eigenschaften eines Modells
  - Wird für einen bestimmten Zweck erstellt
  - Reduziert die Informationen, um Fokus auf einen Aspekt der Realität zu lenken
  - Kann bestehende Realitäten abbilden, um diese zu analysieren (beschreibendes Modell)
  - Kann zukünftige Realitäten abbilden, um diese zu definieren (vorschreibendes Modell)

# Requirements Engineering

- Eigenschaften von Modellen
  - Enthält nur Informationen, die für den Zweck relevant sind
    - Ermöglicht besseres Verständnis der Realität
  - Möglichkeiten Informationen zu reduzieren
    - Aggregation. Informationen von irrelevanten Details bereinigen
    - Selektion. Nur die relevanten Informationen verwenden
  - Vergleich Hausbau
    - Zeichnung von Gebäudegrundriss
    - Elektrischer Schaltplan
    - Beide stellen einen bestimmten Aspekt des Gebäudes dar

# Requirements Engineering

- Vorteile von Modellen im Vergleich zur natürlichen Sprache
  - Leichter zu verstehen und zu merken
  - Durch Reduzierung auf einen Aspekt sind sie leichter zu verstehen
  - Durch definierte Syntax sind mögliche Mehrdeutigkeiten und Auslassungen reduziert
  - Modellierungssprache ist einfacher als natürliche Sprache

# Requirements Engineering

- Nachteile von Modellen im Vergleich zur natürlichen Sprache
  - Modelle konsistent zu halten ist schwierige Herausforderung
  - Informationen aus mehreren Modellen müssen vereint werden, um die Anforderung zu verstehen
  - Hauptsächlich geeignet für funktionale Anforderungen. Für Qualitätsanforderungen und Randbedingungen natürliche Sprache verwenden
  - Eingeschränkter Syntax einer grafischen Modellierungssprache, begrenzt die Möglichkeiten
- Anforderungsmodelle immer mit natürlicher Sprache beschreiben

# Requirements Engineering

- Anwenden von Modellen
  - Welches Modell ist für welchen Kontext geeignet
  - Dabei wird unterschieden zwischen
    - Struktur und Daten
      - statischen Struktureigenschaften eines Systems
    - Funktion und Ablauf
      - Abfolge von Aktionen, um gewünschtes Ergebnis zu erhalten
      - Kontroll- und Datenflusses zwischen den Aktionen
      - Wer ist für eine Aktion verantwortlich
    - Zustand und Verhalten
      - zustandsabhängige Reaktionen auf Ereignisse
      - Dynamik der Interaktion von Komponenten

# Requirements Engineering

- Anwenden von Modellen
  - Eigenschaften des zu entwickelnden Systems geben die Modelle vor
  - System aus mehreren Perspektiven modellieren
  - Umschreiben von textuell beschriebenen Anforderungen, um ihre Verständlichkeit zu verbessern
  - Validieren von textuell beschriebenen Anforderungen mit dem Ziel, Auslassungen, Mehrdeutigkeiten und Inkonsistenzen aufzudecken
  - Besseres Verständnis des Systems und seines Kontexts

# Requirements Engineering

- Modellierung des Kontexts
  - Ermitteln in welcher Umgebung das System sich befindet
  - Identifizierung von Benutzerschnittstellen und Schnittstellen zu anderen Systemen
  - Liefert Informationen darüber, was alles in einem späteren Schritt definiert werden muss (z.B. Schnittstelle zu einem bestehenden System)

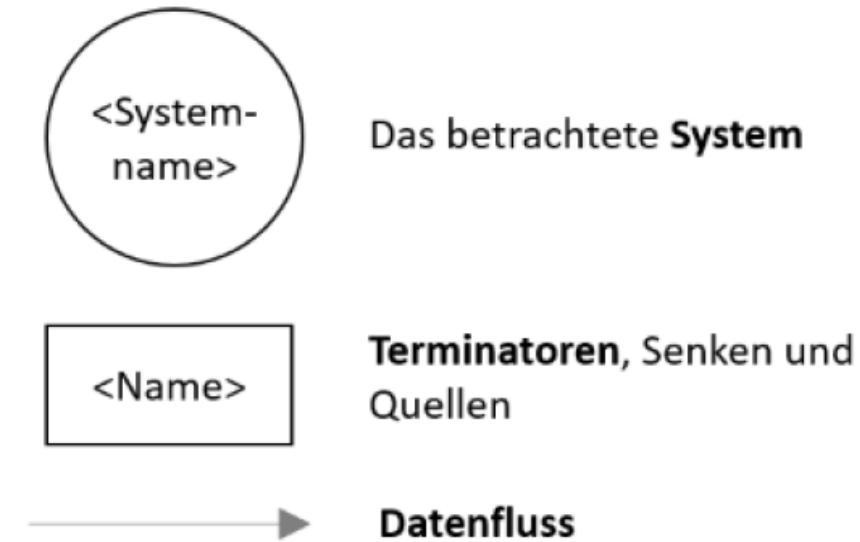
# Requirements Engineering

- Modelle zur Darstellung des Kontexts
  - Datenflussdiagramm aus der strukturierten Analyse
  - UML Use Case Diagramm

# Requirements Engineering

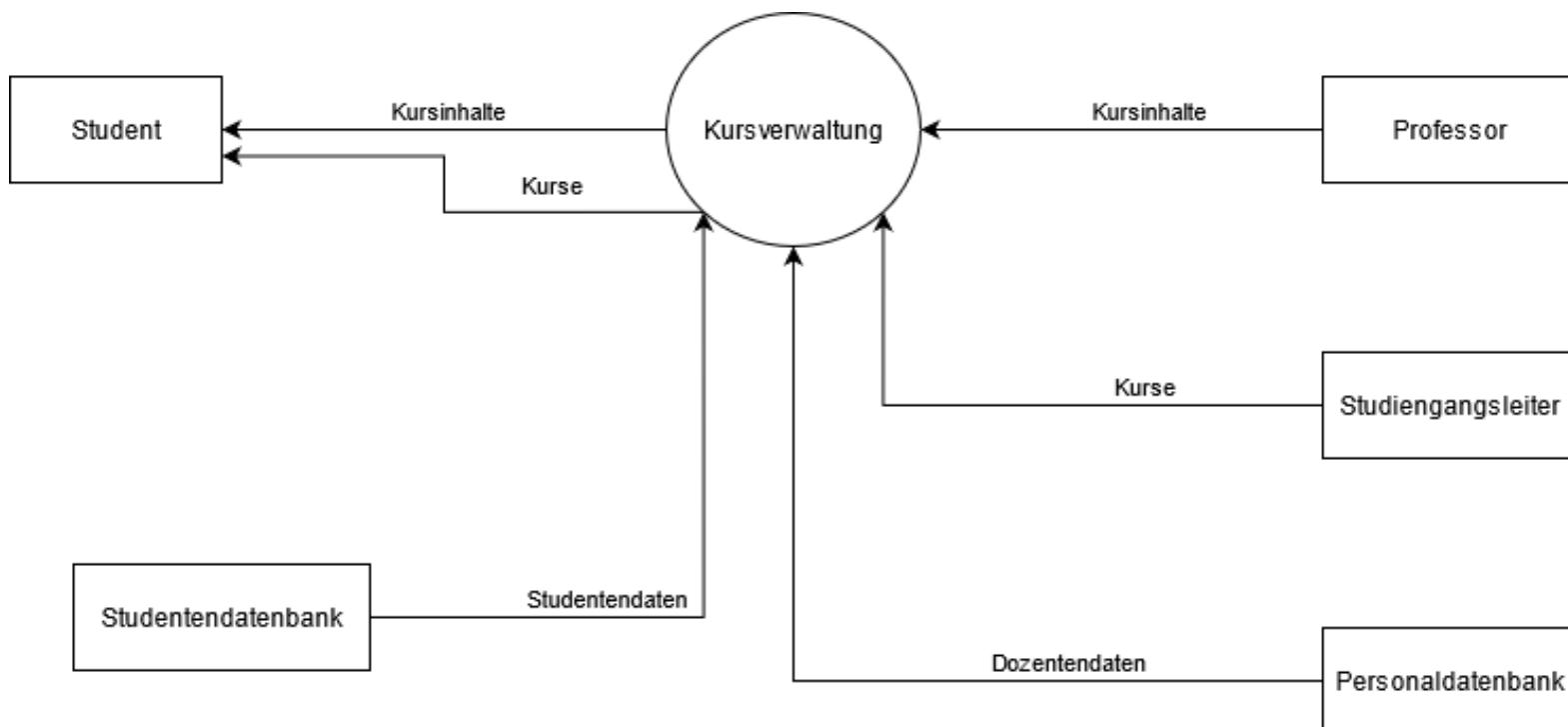
- Modellieren des Kontexts mit Datenflussdiagramm

- Datenflüsse werden dargestellt
- Welche Daten gehen in das System
- Welche Daten gehen aus dem System
- Terminatoren
  - Wer oder was erhält die Daten (Senke)
  - Wer oder was stellt Daten zur Verfügung (Quelle)
- Sehr abstrakte Ebene



# Requirements Engineering

- Modellieren des Kontexts mit Datenflussdiagramm
  - Beispiel Kontextdiagramm



# Requirements Engineering

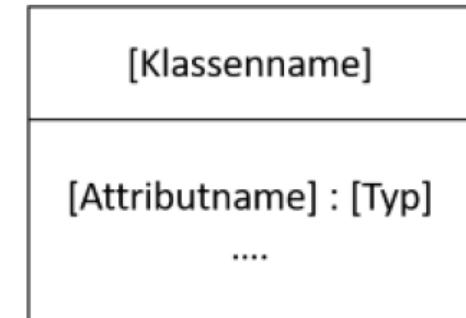
- Modellierung von Struktur und Daten
  - Erfassen von Anforderungen im Hinblick auf Geschäftsobjekte
  - System verwendet diese Objekte als:
    - Eingabe zur Verarbeitung
    - zur Speicherung
    - zur Bereitstellung von Ausgaben
  - Datenmodelle werden verwendet, um die Objekte zu beschreiben
    - die Attribute des Objekts
    - die Beziehungen zwischen Objekten

# Requirements Engineering

- Modelle zur Darstellung von Struktur und Daten
  - Entity-Relationship-Diagramme (ERD)
  - UML Klassendiagramme
  - SysML-Blockdefinitionsdiagramme

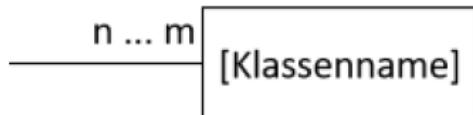
# Requirements Engineering

- Modellierung von Struktur und Daten mit UML Klassendiagrammen
  - Stellt Menge von Klassen und ihre Assoziationen dar
  - Klasse ist eine Beschreibung von Objekten mit gleichen Eigenschaften
  - Klasse hat Namen und Attribute
  - Attribute beschreiben die Eigenschaften
  - Attribute haben Namen und Typ
  - Typ definiert den Wertebereich der Eigenschaft



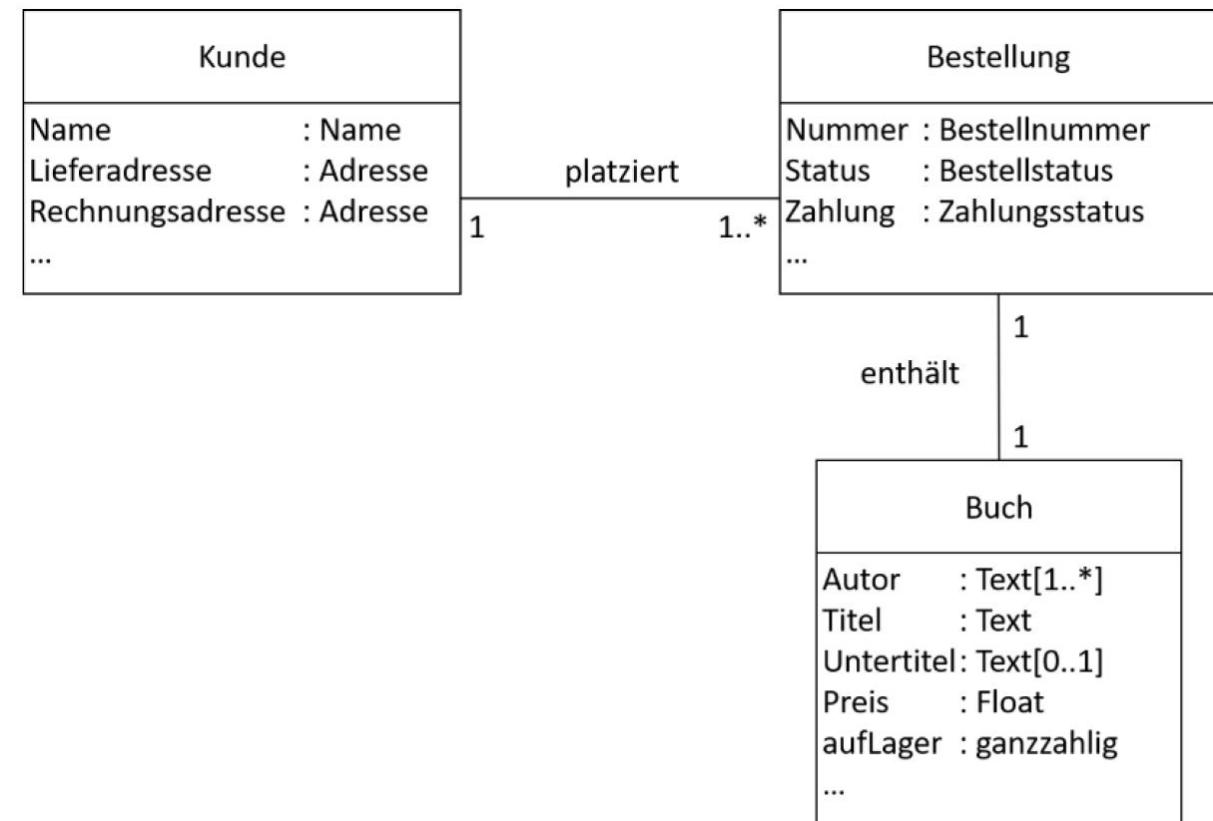
# Requirements Engineering

- Modellierung von Struktur und Daten mit UML Klassendiagrammen
  - Assoziation definiert die Beziehung zwischen zwei Klassen
  - Multiplizität definiert wie viele Instanzen einer Klasse an der Assoziation teilnehmen. Beispiele:
    - 0..1 (null oder einmal)
    - 0..\* (null oder mehrere Male)
    - 1..\* (einmal oder mehrere Male)
    - 6 (genau sechs Mal)
    - 1 (genau ein Mal)
  - Rolle definiert, welche Rolle die Klasse in der Assoziation spielt



# Requirements Engineering

- Modellierung von Struktur und Daten mit UML Klassendiagrammen
  - Klassen können im Glossar genauer beschrieben werden
  - Beispiel Klassendiagramm



# Requirements Engineering

- Modellieren von Funktion und Ablauf
  - Beschreiben, wie das System Eingaben in Ausgaben umwandeln soll
  - Hierfür stehen verschiedene Diagrammtypen zur Verfügung
  - Ggf. mehrere Modelle erforderlich, um die Anforderungen zu dokumentieren

# Requirements Engineering

- Modelle zur Darstellung von Funktionen und Ablauf
  - UML Use Case Diagramm
  - UML Aktivitätsdiagramm
  - Datenflussdiagramm
  - Business Process Modeling Notation (BPMN, deutsch Geschäftsprozessmodell und -notation)
    - Beschreibung von Geschäftsprozessen oder technischen Prozessen

# Requirements Engineering

- UML Aktivitätsdiagramm zum Modellieren von Funktionen
  - Stellt Elemente für Modellierung von Aktionen und Kontrollfluss zwischen Aktionen zur Verfügung
  - Verarbeitungslogik von Use Case-Szenarien im Detail zu spezifizieren
  - Startknoten definiert den Beginn
  - Endkonten definiert das Ende des Aktivitätsflusses
  - Beendigung des Kontrollflusses
  - Aktion/Aktivität definiert eine nicht unterbrechbare Berechnung, die zu einer Zustandsänderung des Systems führt oder einer Ausgabe des Ergebnisses
  - Aktions- oder Kontrollfluss definiert den Übergang zwischen Aktionen

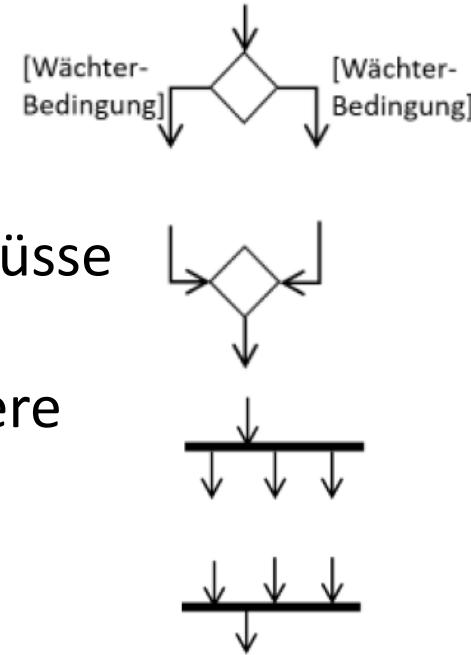


[Name]



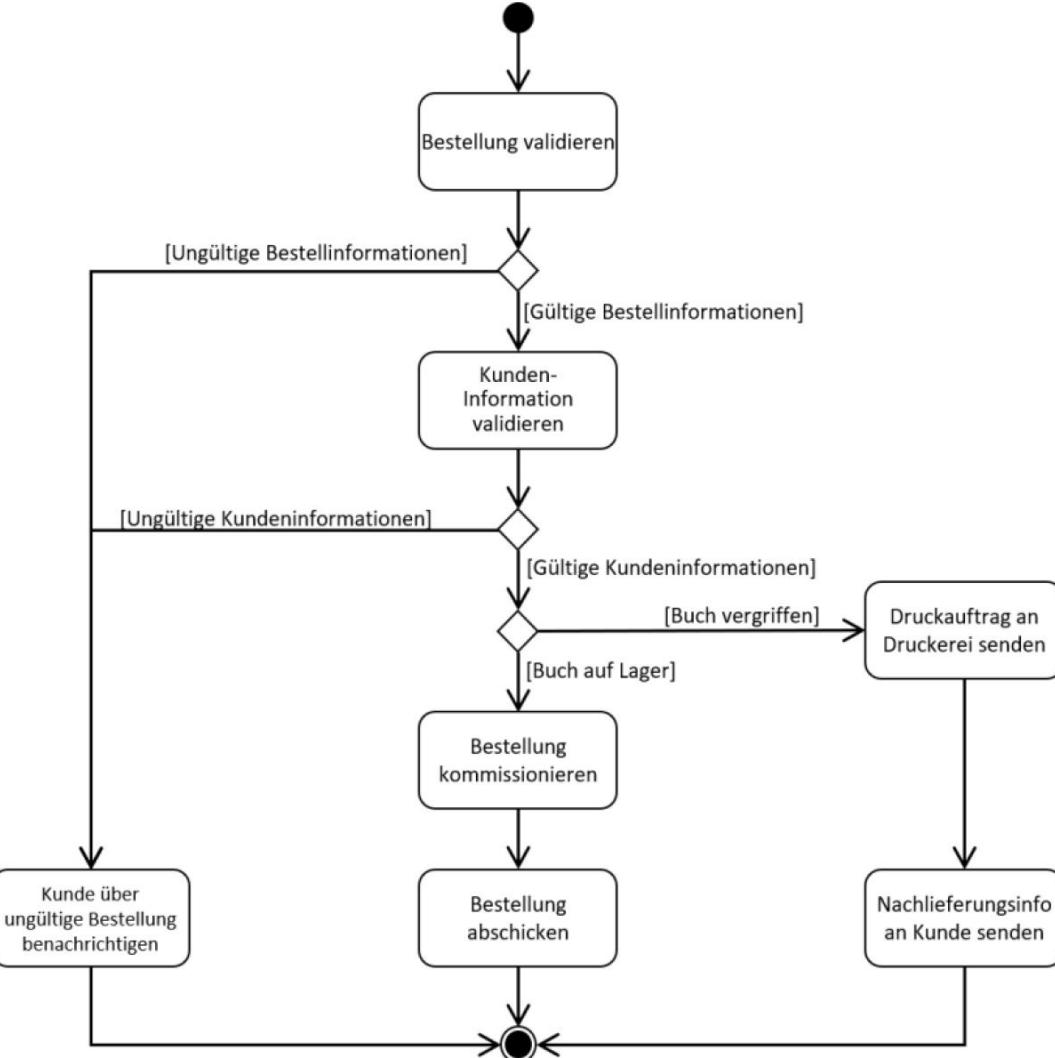
# Requirements Engineering

- UML Aktivitätsdiagramm zum Modellieren von Funktionen
  - Entscheidungsknoten verzweigt den Kontrollfluss auf Grundlage einer Bedingung
  - Ausgehende Kontrollflüsse mit Wächterbedingung kennzeichnen
  - Zusammenführung bringt mehrere Kontrollflüsse zusammen
  - Gabelpunkt teilt einen Kontrollfluss in mehrere parallel laufende auf
  - Verbindungspunkt bringt mehrere parallel laufende Kontrollflüsse zusammen



# Requirements Engineering

- UML Aktivitätsdiagramm zum Modellieren von Funktionen
  - Beispiel Aktivitätsdiagramm



# Requirements Engineering

- Modellieren von Zustand und Verhalten
  - Funktionale Anforderungen, die das Verhalten, die Zustände und die Übergänge eines Systems oder eines Geschäftsobjekts beschreiben
  - Geschäftsobjekts kann einen Lebenszyklus haben, der eine Reihe von vorgeschriebenen Zuständen durchläuft
  - Beispiel Zustände von Geschäftsobjekt Auftrag:
    - Erteilt
    - Validiert
    - Bezahlt
    - Versandt
    - Abgeschlossen

# Requirements Engineering

- Modelle zur Darstellung von Verhalten und Zuständen
  - Zustandsdiagramm nach Harel
  - UML Zustandsdiagramm

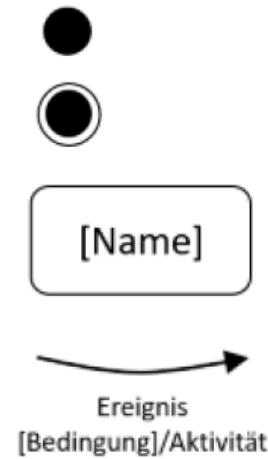
# Requirements Engineering

- Modellieren von Zuständen mit Zustandsdiagramm
  - Beschreiben Zustandsmaschinen, die endlich sind
  - Zeigt die Zustände, die das System oder ein Objekt annehmen kann
  - Gibt auch an, wie der Zustand wechselt, d.h. den Zustandsübergang
  - Wechseln des Zustands erfordert Trigger aus dem System oder aus der Systemumgebung

# Requirements Engineering

- UML Zustandsdiagramm zur Modellierung von Zuständen

- Startzustand, ist der Startpunkt einer Zustandsmaschine
- Endzustand, ist der Endpunkt einer Zustandsmaschine
- Zustand, definiert die Reaktion der Zustandsmaschine auf Ereignisse
- Zustandsübergang, definiert den Übergang von einem Zustand in den nächsten
  - Zustandsübergang wird durch Ereignis ausgelöst
  - Optional kann eine Bedingung angegeben werden
  - Optional kann eine Aktivität angegeben werden

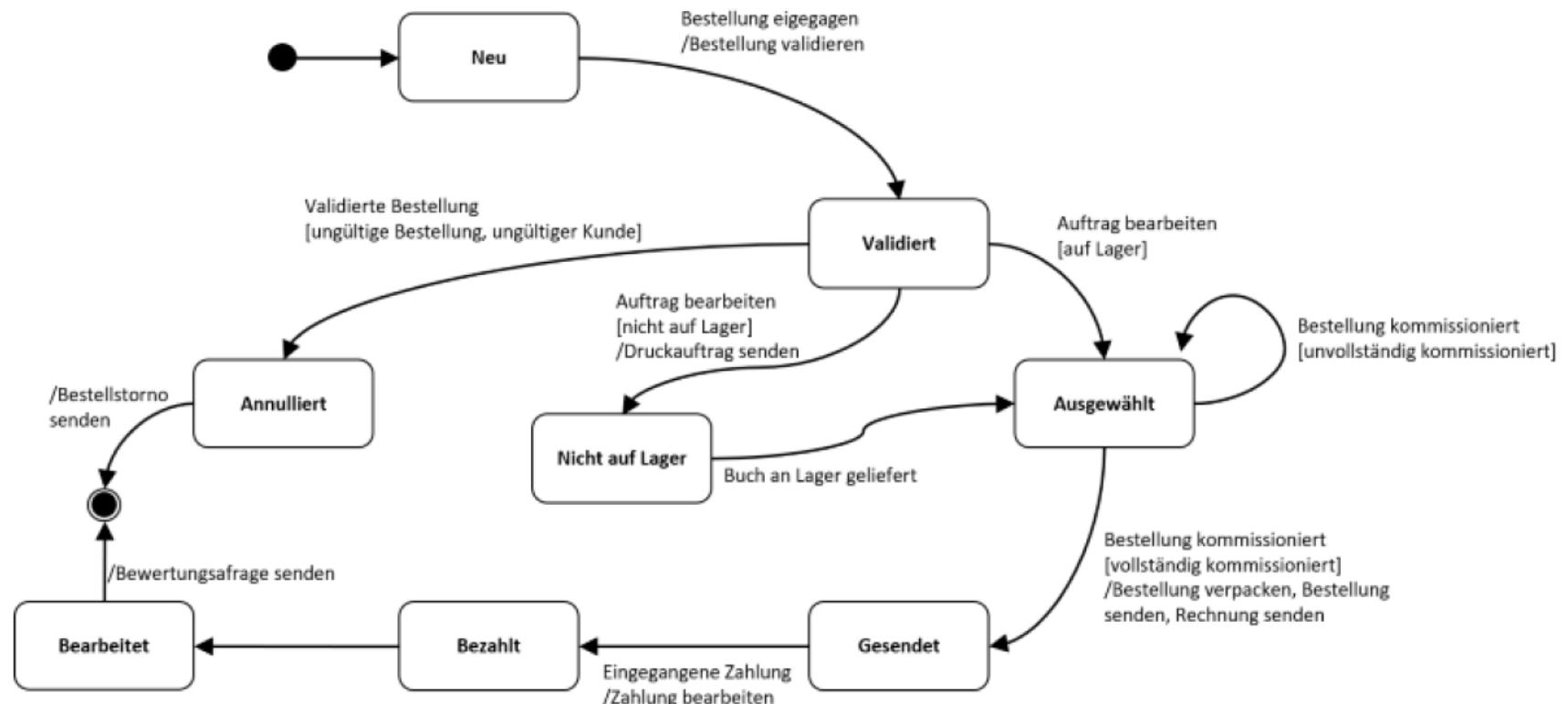


# Requirements Engineering

- UML Zustandsdiagramm zur Modellierung von Zuständen
  - Klassen, die im Klassendiagramm definiert wurden können Attribute haben die Zustände darstellen
  - Mithilfe des Zustandsdiagramms können diese Zustände modelliert werden

# Requirements Engineering

- UML Zustandsdiagramm zur Modellierung von Zuständen
  - Beispiel Zustandsdiagramm



# Requirements Engineering

- Glossare
  - Schaffen gemeinsames Verständnis der verwendeten Terminologie
  - Enthält Definitionen für alle Begriffe, die für das System relevant sind
  - Und verwendete Abkürzungen und Akronyme
  - Sollte während der Projektlaufzeit immer aktuell und konsistent gehalten werden
  - Definierte Begriffe müssen in anderen Arbeitsprodukten verwendet werden
  - Vermeidet Missverständnisse bezüglich der verwendeten Terminologie

# Requirements Engineering

- Prototypen
  - Vorläufige Entwicklung einer bestimmten Charakteristik eines Systems
  - Unterscheidung zwischen drei verschiedenen Typen
  - Explorative Prototypen werden verwendet um
    - Gemeinsames Verständnis zu schaffen
    - Anforderungen zu klären
    - Anforderungen zu validieren
    - Sind kurzlebige, die nach Gebrauch weggeworfen werden

# Requirements Engineering

- Prototypen
  - Experimentelle Prototypen werden verwendet um
    - Lösungskonzepte für technische Entwürfe zu untersuchen
    - Machbarkeit zu analysieren
    - Sind kurzlebige, die nach Gebrauch weggeworfen werden
    - Werden nicht im Requirements Engineering eingesetzt
  - Evolutionäre Prototypen werden verwendet um
    - Pilotsysteme, die den Kern eines zu entwickelnden Systems bilden
    - Werden iterativ weiterentwickelt
    - Werden in der agilen Systementwicklung eingesetzt

# Requirements Engineering

- Einsatz explorativer Prototypen im Requirements Engineering
  - Hilft dabei Anforderungen zu ermitteln. Insbesondere dann wenn Stakeholder ihre Wünsche nicht klar ausdrücken können
  - Hilft dabei bestehende Anforderungen zu validieren
  - Explorative Prototypen unterscheiden sich nach Detailgrad:
    - Wireframe
    - Mock-up
    - Native Prototypen

# Requirements Engineering

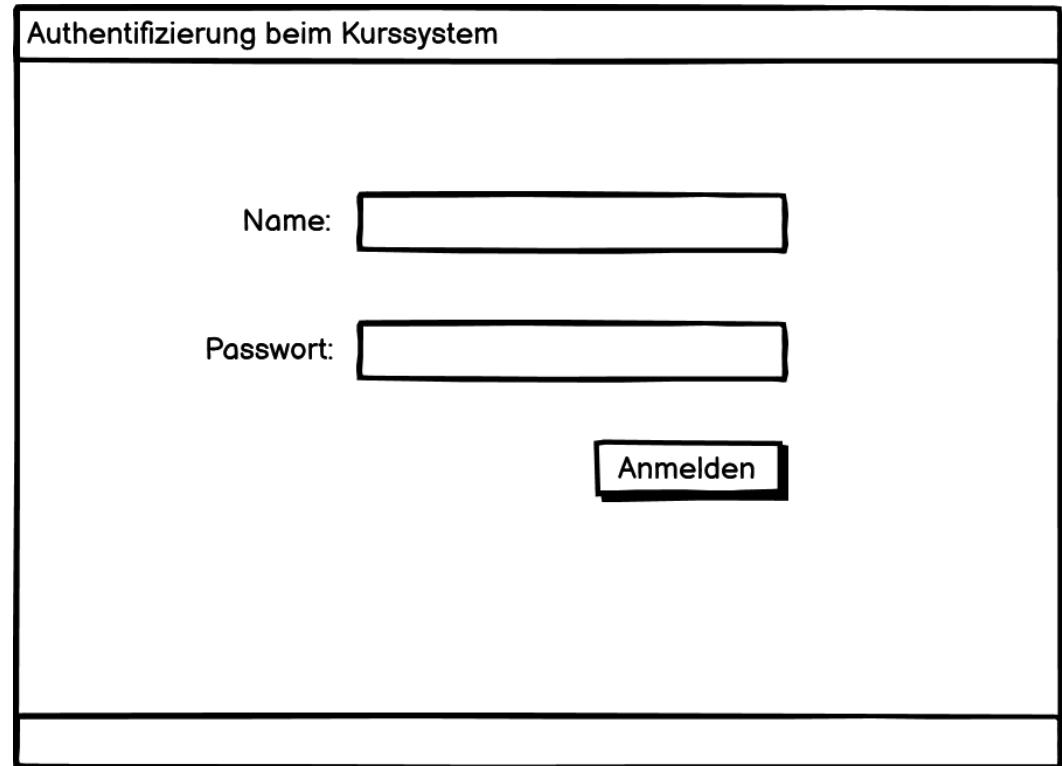
- Wireframe Prototypen
  - Auch Drahtmodell oder Papierprototyp genannt
  - Haben geringe Genauigkeit
  - Dienen zur Validierung und Klärung von Anforderungen
  - Tools:
    - Balsamiq Wirefames
    - Mockplus

Authentifizierung beim Kurssystem

Name:

Passwort:

**Anmelden**



# Requirements Engineering

- Mock-up Prototypen
  - Haben mittlere Genauigkeit
  - Verwenden reale Bildschirmschirmdarstellung und Klickfolgen
  - Haben keine Funktionalität
  - Zur Spezifikation und Validierung von Benutzerschnittstellen

# Requirements Engineering

- Native Prototypen
  - Haben eine hohe Genauigkeit
  - Kritische Teile des Systems werden implementiert
  - Enthalten Funktionalität, die damit validiert werden kann
  - Dienen als Entscheidungsgrundlage bei Anforderungsvarianten
- Abwägen zwischen Nutzen und Kosten bei Prototypen

# Referenzen

- International Requirements Engineering Board  
(<https://www.ireb.org/en>)
- ASVS Application Security Verification Standard  
(<https://owasp.org/www-project-application-security-verification-standard/>)