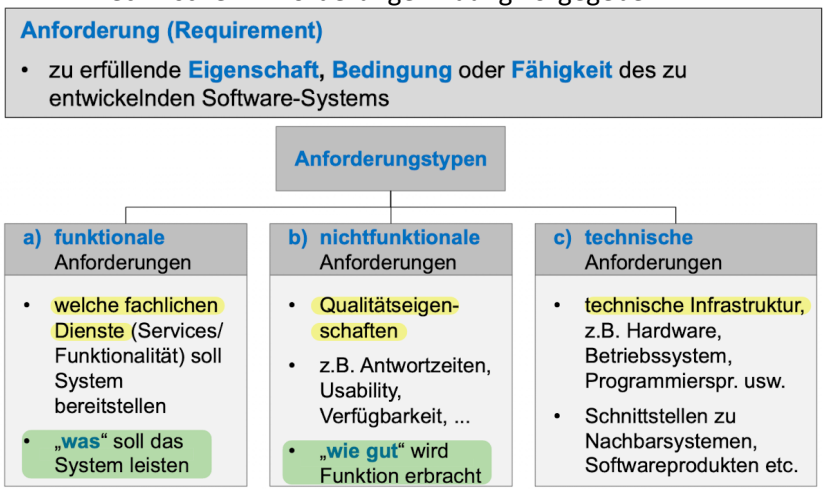
3 Anforderungsanalyse

3.1 Grundlagen

3.1.1 Anforderungen

- Nichtfunktionale Anforderungen sind die Herausforderungen, die auch zum Großteil ca. 80% des Budgets ausmachen
- Technischen Anforderungen häufig vorgegeben



3.1.2 Anforderungsanalyse

Frage: Was soll das System **aus Sicht des Benutzers** leisten?

a) Wer benutzt das Software-System?
→ **Akteure (Actors)**

b) Wie benutzt ein Akteur das Software-System?
→ **Anwendungsfälle (Use Cases)**

• **Ziel** der Anforderungsanalyse
  – Anforderungen des Kunden bzw. der zukünftigen Benutzer (User) des Systems ermitteln und dokumentieren
  – gemeinsames Verständnis zwischen Kunde und SW-Projekt

→ **Anforderungsanalyse ist entscheidend für Projekterfolg!!!**

3.1.3 Schwierigkeiten:

a) hoher Kommunikationsbedarf zwischen Kunde und Entwicklungsteam

- keine gemeinsame Sprache bzw. Fachterminologie
- unterschiedliche Denkmuster
  - Gesprächsdisziplin und konsequente Klärung von Missverständnissen erforderlich
- **Kundenseite**: Auftraggeber (Management), Benutzer*innen,
- Betreiber haben teilweise sich widersprechende Anforderungen

b) Anforderungen sind zu Projektbeginn häufig nicht bekannt

- vage Idee → Lastenheft → Geschäftsprozessmodell
- Benutzer*in kann sich das Ziel-System nicht vorstellen
- verbale Beschreibung von Anforderungen ist oft unpräzise

c) Anforderungsvolatilität (moving targets)

- Anforderungen ändern sich während Projektlaufzeit

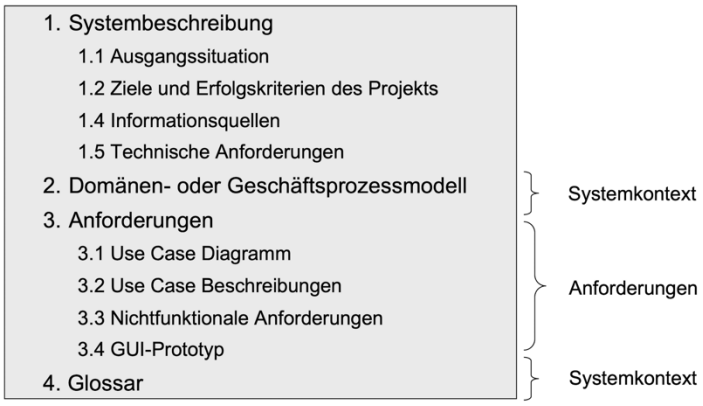
3.2 Ergebnisse

Rollen und Ergebnisse im Rahmen der Anforderungsanalyse

- Ergebnisse dieses Schrittes werden in **Anforderungsmodell** (auch **Pflichtenheft** genannt) festgehalten!

Anforderungsmodell (Pflichtenheft)

- **Auftraggeber** beschreibt die Gesamtheit der Forderungen, die bei dem von ihm beschriebenen Auftrag erfüllt werden sollen => **Lastenheft**
- **Auftragnehmer** beschreibt fachliche Zusammenhänge als Basis für Analyse und Design
  => **Anforderungsmodell** (**Pflichtenheft)**
  - Erweiterung des Lastenhefts



1) Systembeschreibung

- beschreibt oberflächlich Kontext und wesentliche Merkmale des Systems

1.1. Ausgangssituation

- Überblick über Kunden, Unternehmen und Situation

1.2. Ziele und Erfolgskriterien des Projekts (Ziele, Ausgrenzungen, Prämissen)

- Nutzen und Zweck des Systems (Kernanforderungen)
- Abgrenzung: welche Teile des Unternehmens sind vom neuen System betroffen bzw. welche **nicht?**
- vereinbarte Voraussetzungen (Prämissen) für Projektdurchführung

1.3. Informationsquelle

- welche Informationen stehen zur Verfügung? (Ausschreibung, Angebot, grobes Lastenheft, ...)
- gibt es Vorarbeiten?

1.4. Technische Anforderungen

- welche vergleichbaren Alt- oder Standardsysteme gibt es?
- welche Schwachstellen haben sie?
- zu welchen Altsystemen bestehen **Schnittstellen**?

Domänenmodell

Ziele: Auffinden der zentralen **Begriffe** und **Konzepte** der **fachlichen** Welt (Schlüsselkonzepte, Domänenvokabular)

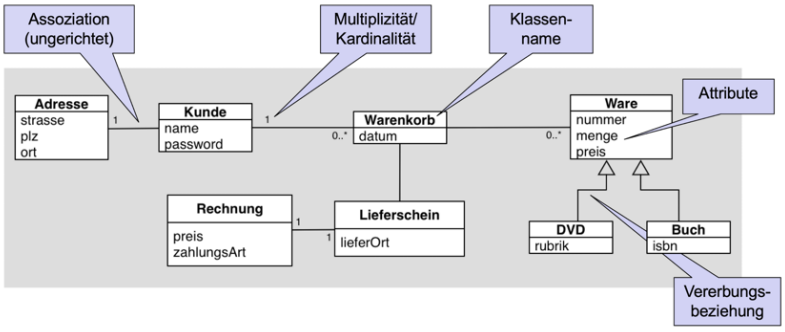
Ergebnis

- **Modell** (UML) mit **fachlichen Kernklassen**
- Geschäftsobjekte (Entitäten)
  - » Bestellungen, Konten, Verträge, Bücher, Personen, Kunden, Produkte, ...
- Ereignisse
  - » Ankunft, Ausleihe, Auftragseingang, ...

Fachliche Begriffe und Konzepte

- entsteht in Workshops mit Analysten und fachlichen Experten*innen
- wird weiterverwendet für Analyse-Modell
- **Vorsicht**: Domänenmodell darf **keine** Implementierungsaspekte enthalten

Beispiel: Systemkontext – Domänenmodell für Internet-Buchladen

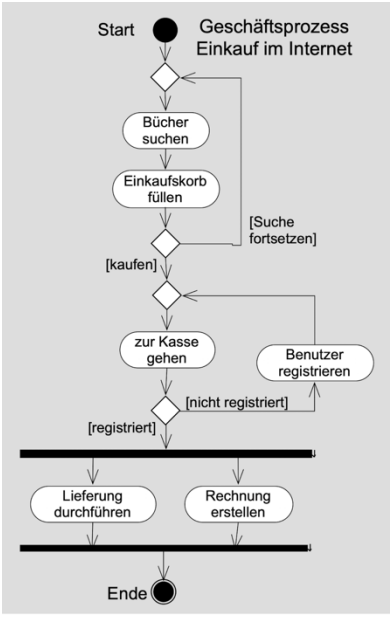


- Fokus: **statische Systemsicht**

Geschäftsprozessmodell

Geschäftsprozess (business process)

- Zusammenfassung/ Abfolge von fachlich zusammenhängenden (organisatorisch jedoch evtl. verteilten) Aktivitäten entlang einer Wertschöpfungskette
- initiiert durch ein Ereignis, erzeugt sichtbares Ergebnis (Mehrwert)
  - besteht i.d.R. aus einer Reihe von Anwendungsfällen



- **Ziele**
  - Auffinden der zentralen **Prozesse** der **fachlichen** Welt
- **Ergebnis**
  - Geschäftsprozessmodell beschreibt Abläufe der zentralen **Geschäftsprozesse**
- entsteht in Workshops mit Analysten und fachlichen Experten*innen

Notation

- es existiert keine etablierte UML-Notation für Geschäftsprozessmodellierung
- verwendet werden u.a.
  - **a) UML-Aktivitätsdiagramme**
  - b) eigene Prozessmodellierungs- sprachen

- **Fokus**: dynamische Sicht
- Beispiel: Geschäftsprozessmodell für Internet-Buchladen (als UML-Aktivitätsdiagramm)

Use Case (Anwendungsfall):

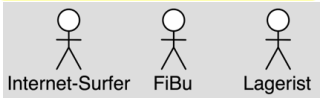
- Anwendungsfälle beschreiben, wie das System genutzt wird und welche (fachlichen) Funktionen genutzt werden
- Anwendungsfälle betrachten das System von **außen!**

3.1) Use Case Diagramm (Anwendungsfalldiagramm)

Akteure (Actors): Nutzer*in eines Softwaresystems

- repräsentieren die **Systemumgebung** (nutzen das **System** von **außen)**
- haben **Nutzen** von Durchführung eines Use Cases

UML-Notation für Akteure



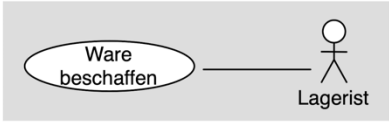
es gibt **menschliche** und **technische** Akteure

- z.B. Sachbearbeiter*innen, Kunden **oder** andere Systeme, Anwendungen

Use Case (Anwendungsfall)

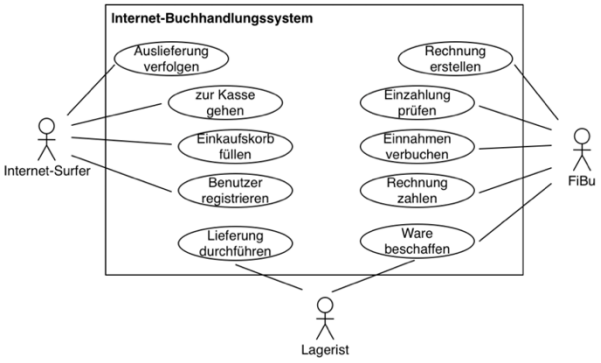
- **typische Interaktion** zwischen **Akteur** und **Softwaresystem**
  - Anwendungsfall besteht aus **mehreren zusammenhängenden** Aufgaben, die von Akteur durchgeführt werden, um Ziel zu erreichen
- durch informellen Text beschrieben
- beschreibt den Leistungsumfang des Systems

UML-Notation für Use Cases



Use Case Diagramm (Anwendungsfalldiagramm)

- beschreibt Zusammenspiel mehrerer Use Cases untereinander und mit den Akteuren
- **System** durch Rechteck modelliert, dass alle UCs umschließt



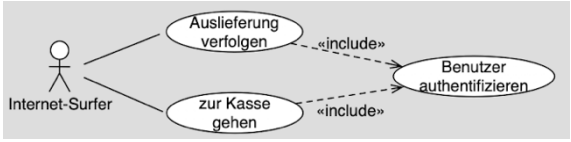
Problem: sinnvolle **Granularität** von Use Cases

- wie viel Funktionalität ist in einem Use Case beschrieben?
- wenige (15-20) große Use Cases versus viele (100) kleine Use Cases
- häufiger Fehler: zu viele kleine Use Cases

Strukturierung von Use Cases mit Stereotypen

(a) **<<include>>** – Beziehung verweist auf einen anderen (Sub-)Use Case

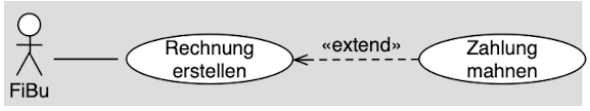
- Vermeidung von redundanten Ereignisflüssen



(b) **<<extend>>** – Beziehung für **optionalen** Ereignisfluss

- spezielle Use Cases für umfangreiche Ausnahmen, Varianten und Sonderfälle
- wenn Use Case den Erweiterungspunkt erreicht, wird entspr. Ereignisfluss Ausgeführt

Vorsicht: Strukturierung macht UC-Diagramm unübersichtlicher !



Sehr große Iterationen

Vorteil:

- Alle Mitarbeiter beschäftigt
- Weniger Redundanz

Nachteil:

- Testen der Ergebnisse zieht sich
- Wahrscheinlichkeit für Misserfolg steigt, da zu viel Use Cases parallel gemacht werden

Sehr kleine Iterationen

Vorteil:

- Schnell Ergebnisse zum Präsentieren

Nachteil:

- Mitarbeiter ohne Beschäftigung
- Sehr viel redundante Arbeit

3.3) Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind Qualitätsanforderungen, wie gut eine Funktion wird die man quantifizieren kann, da sonst keine Aussage getroffen werden kann.

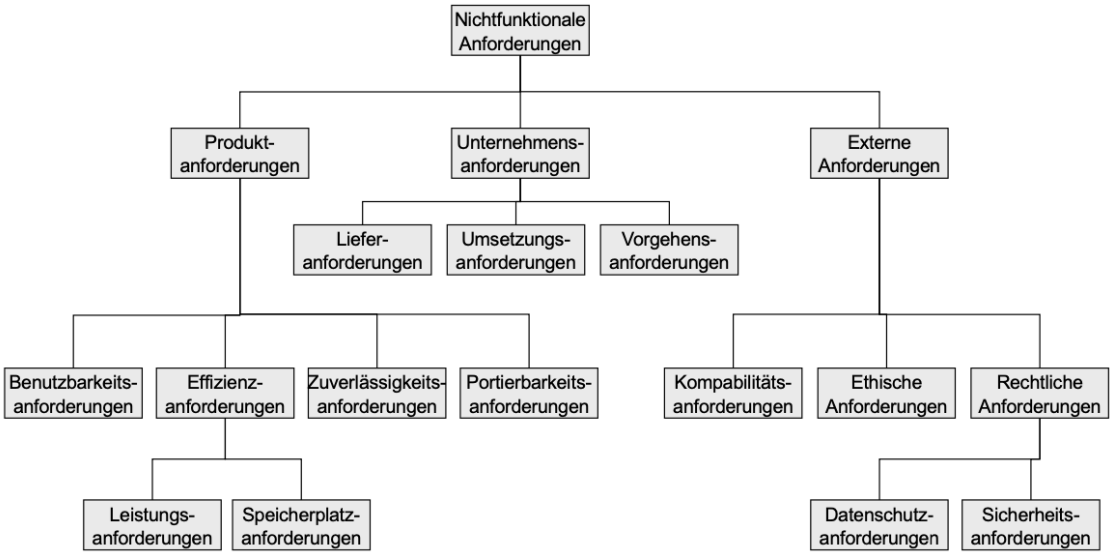
> **nichtfunktionale Anforderungen** (nonfunctional requirements)
> – **Qualitätsanforderungen** an das Produkt, **Prozessanforderungen** und **externe Anforderungen**, die sich auf wichtige qualitative und quantitative Eigenschaften des Systems beziehen

Beispiele
- Einhaltung von Normen, Richtlinien und Gesetzen
- Zuverlässigkeit, Wartbarkeit, ...
- Datenschutz
- Einsatz eines vorgegebenen Vorgehensmodells
- ggf. Festlegung der Infrastruktur (Betriebssysteme, Datenbanken, ...)

- Problem: nichtfunktionale Anforderungen lassen sich i.d.R. schwer verifizieren

→ ***Quantifizierung** der Erfüllung notwendig*
(z.B. Antwortzeit kleiner 1 Sekunde; Suche eines Buches nach Autor darf nicht länger als 3 Sekunden dauern)

Arten nichtfunktionaler Anforderungen



Beispiel

Beispiel: Nichtfunktionale Anforderungen für Internet-Buchhandlung

1. Infrastruktur (= technische Anforderungen)
- Clients
  - plattformunabhängig mit JavaScript-fähigem Browser
  - mobile App für Android (> 12.x.x) und iOS (> 14.2)
- Server
  - Betriebssystem Ubuntu Server 18.04
  - Apache Webserver

2. sonstige Anforderungen
- Sicherheit: Datenübertragung mit SSL-Protokoll
- Lastverhalten: Lastspitzen größer als 100.000 Zugriffe pro Sekunde
- Verfügbarkeit: weniger als 1 Stunde Ausfallzeit pro Monat
- Usability/ Ergonomie
  - Bestellung nach zwei Mausklicks möglich
  - Einhaltung der ISO-Norm 9241 zur Dialoggestaltung
  - Erfüllung der Standards der Barrierefreiheit (W3C Web Content Accessibility Guidelines)

3.4) GUI-Prototyp

GUI-Prototyp (I)

Prototyp (allgemein in Software-Entwicklung)
- funktionsfähiges, i.d.R. vereinfachtes Versuchsmodell eines geplanten IT-Systems, das Systemfunktionalitäten/ -verhalten aufzeigt

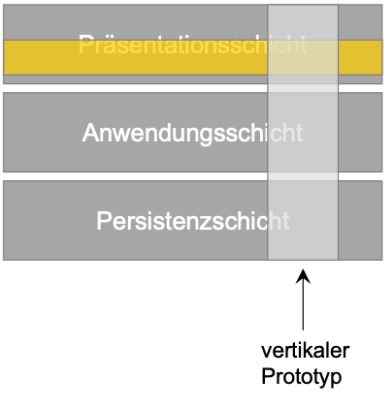
- Einsatzzwecke
  - *Diskussionsgrundlage*
  - *Machbarkeitsstudie*
  - *Evaluierung von Alternativen*
  - *Experimentierobjekt*
- **Vorteil**: minimiert Entwicklungsrisiken
- **Nachteil**: erfordert Entwicklungsressourcen

GUI-Prototyp in Anforderungsanalyse (nur bei Bedarf erforderlich)
- auf Basis ausgewählter Use Cases erstellt
  - *verdeutlicht **Interaktion** zwischen **Benutzer** und **Softwaresystem***
  - *dient einem besseren Verständnis der Use Cases*
  - *erster Schritt für GUI-Styleguide und Design (Þ Usability Engineering)*

Arten von GUI-Prototypen

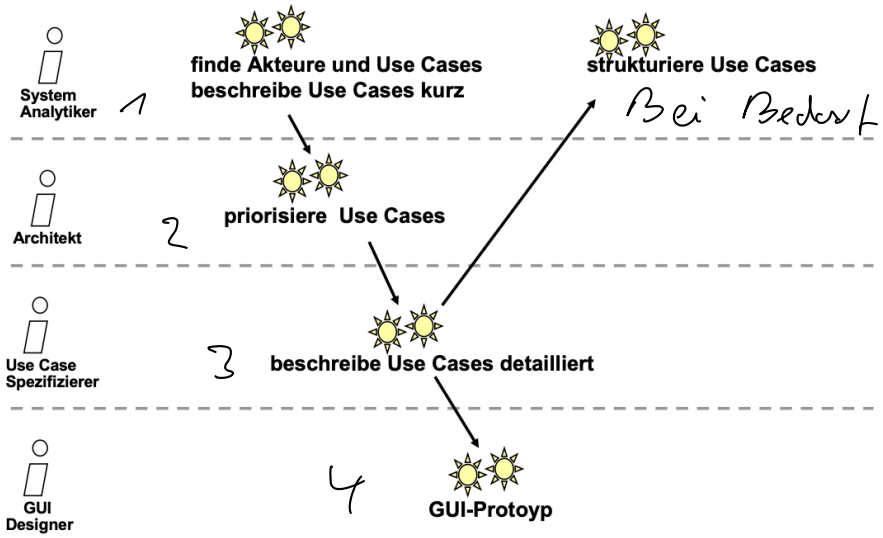
Funktionalität
- a) horizontaler Prototyp
  - ohne Funktionalität (Fachlogik, Datenbankzugriffe,....)
  - ggf. relativ viele Fenster (komplette GUI ?)
- b) vertikaler Prototyp
  - mit voller Funktionalität
  - nur wenige Fenster

Verwendung
- a) experimenteller Prototyp
  - Wegwerfprodukt
  - wird nicht weiter verwendet
- b) evolutionärer Prototyp
  - wird weiter verwendet



3.3 Vorgehen

Vorgehensweise zur Anforderungsanalyse



3.4 Zusammenfassung

Unified Process ist Use-Case Driven

- Use Cases bestimmen die Iterationsstufen
- jede Iteration betrachtet nur ihren Teil der GUI, Klassen und DB-Tabellen

Kriterien für erfolgreiche Anforderungsanalyse

- **richtige** Ansprechpartner*innen auf Kundenseite einbinden (zukünftige Anwender*innen für Anforderungen, Verantwortliche für techn. Infrastruktur, Management für Geschäftsziele)
- Anforderungsdokumentation berücksichtigt **Sprache** und **Begriffe des Kunden** und ist für diesen auch **verständlich**
- hohe Qualität der Anforderungsanalyse durch
  - intensive Zusammenarbeit mit Kunden/ Anwendern
  - **vollständige** und **konsistente Dokumentation** aller Ergebnisse
  - mehrere Iterationen mit abschließenden Reviews
  - **Reviews** (= formaler Bewertungsprozess der Qualität von Ergebnissen) als
    - Interne Reviews - Kundenreviews – Abnahmereview

Anforderungsanalyse

- **Anforderungsanalyse** ist entscheidend für Projekterfolg
- Use Cases: **die** Methode zur Anforderungsanalyse
  a.) **AkteureundUseCases**
  b.) **Use Case Diagramm**
- Vorteile von Use Cases (Anwendungsfällen)
  - legen **Systemfunktionalität** fest
  - sind **anschaulich** und **exemplarisch** (werden sehr gut von Fachabteilungen verstanden)
  - helfen Software aus **Sicht des Benutzers** zu entwickeln
- **Anforderungsmodell** (entspricht **Pflichtenheft**)
  - besteht aus allen Produkten des Arbeitsschrittes
  - beschreibt fachliche Zusammenhänge in einem Dokument