# Software Engineering I

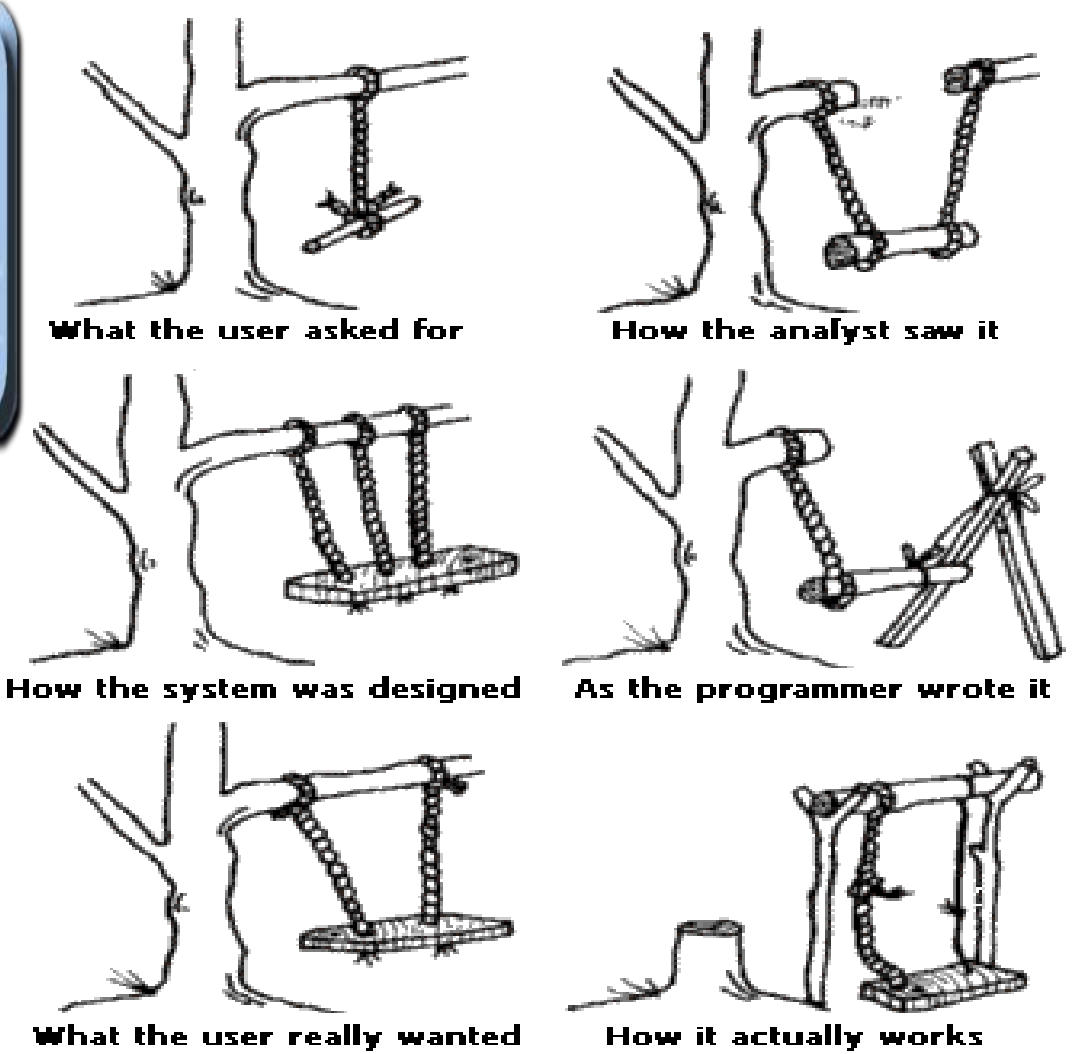
# 2. Anforderungsanalyse und Spezifikation

Prof. Dr. Eckhard Kruse

DHBW Mannheim

# Der Wert eines Softwaresystems...

The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended.
[Nuseibeh&Easterbrook '00]



# Kundenprofil

## Software-Engineering-Projekt

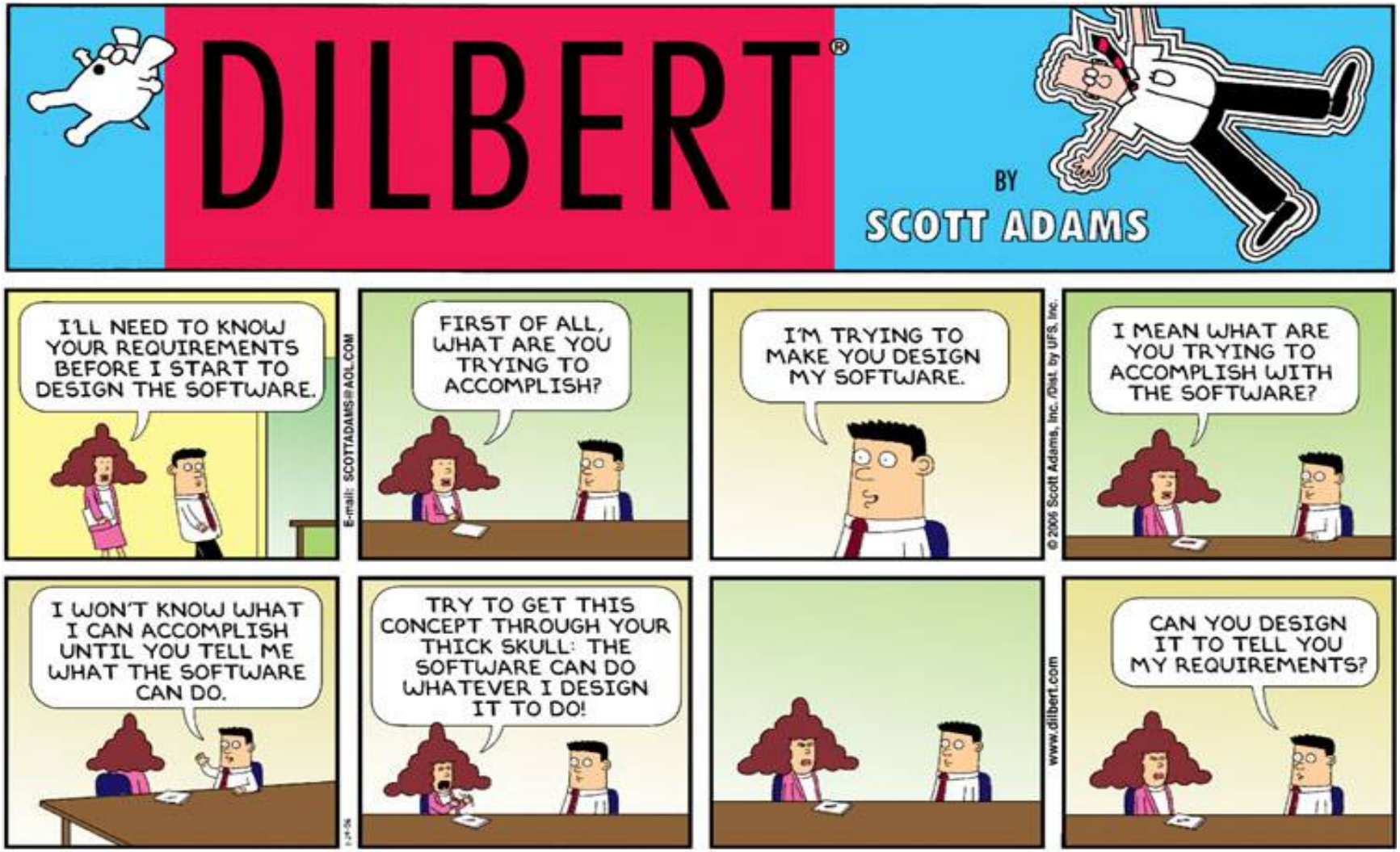
**P.2 Kundensicht: Definition des eigenen Profils**

a) Als Kunde sind Sie an einer in der Themenstellung vorgegebenen Softwarelösung interessiert. Definieren Sie Ihr Kundenprofil und Hintergründe für den Softwareauftag.
   a) In welcher Branche agieren Sie?
   b) Warum/wofür benötigen Sie die Software?
   c) Was sind typische Anwender/Anwendungsszenarien?
   d) Wann wäre der Softwareeinsatz ein Erfolg?

b) Erstellen Sie zwei, drei Folien, die Ihr Profil prägnant wiedergeben und mit denen Sie sich der "Öffentlichkeit" bzw. dem Lieferanten präsentieren können.

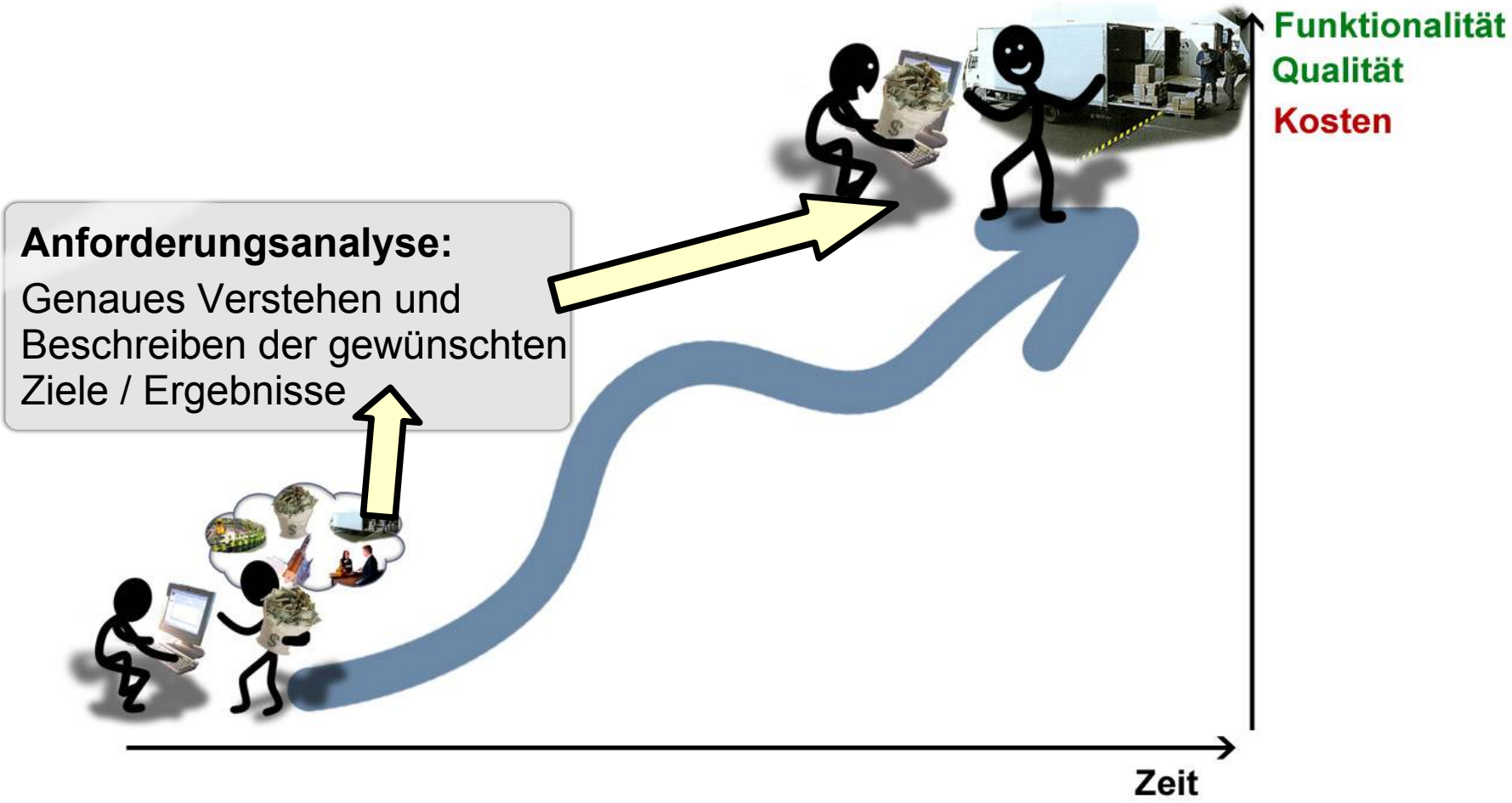
c) Ein Mitglied des Teams soll mit diesen Folien eine 5-minütige Kurzvorstellung von Ihrer Firma geben.

# Anforderungen – woher nehmen?



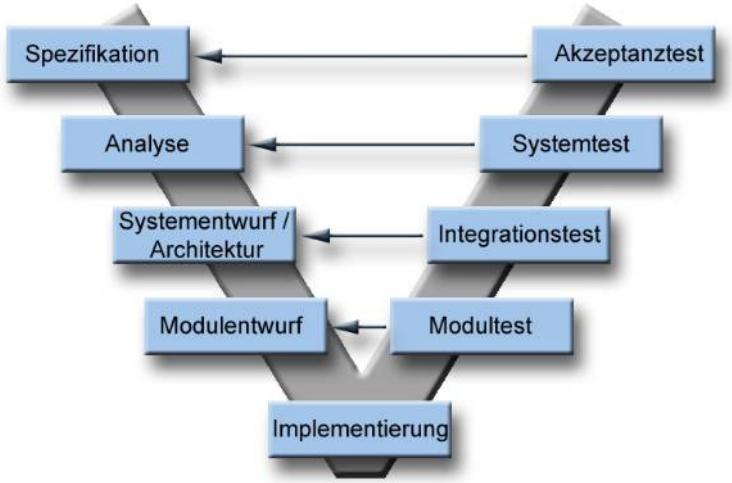

© Scott Adams, Inc./Dist. by UFS, Inc.

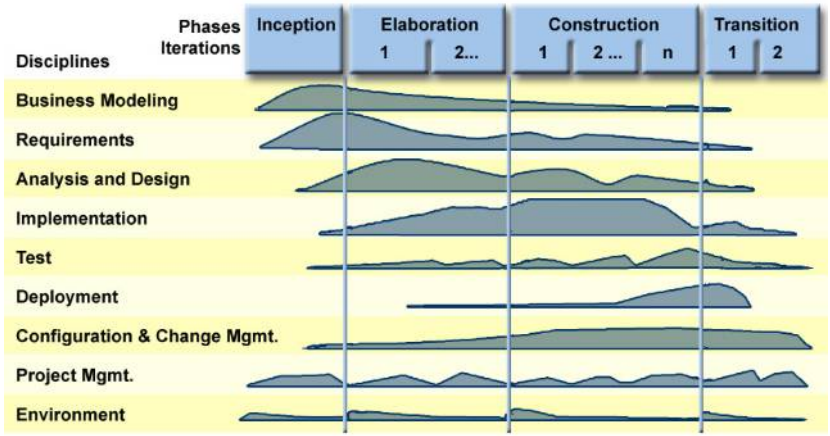
# Am Anfang die Anforderungen

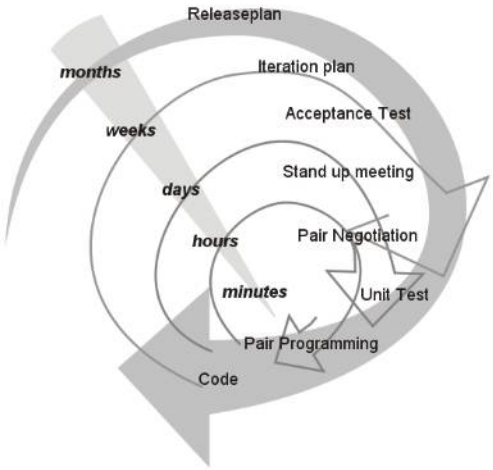



**Anforderungsanalyse:**
Genaues Verstehen und Beschreiben der gewünschten Ziele / Ergebnisse

# 1. Phase in allen Vorgehensmodellen

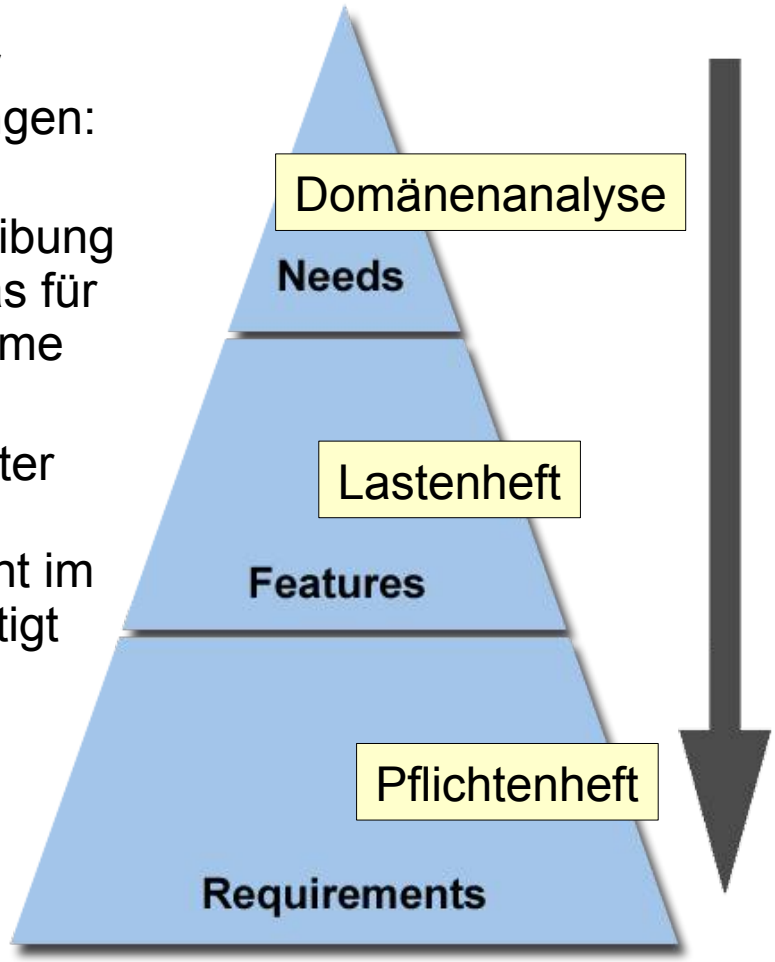



# Anforderungen: Detaillierung

**Needs:** Bestandsaufnahme existierender Geschäftsprozesse, Problembeschreibungen: Warum soll Software entwickelt werden.

**Features:** Lastenheft mit grober Beschreibung von Hauptfunktionen neuer Software: Was für Funktionen sollen die gefundenen Probleme lösen.

**Requirements:** Pflichtenheft mit detailierter Beschreibung der Aufgaben der neu zu entwickelnden Software: Immer noch steht im Vordergrund „was für Funktionalität benötigt wird“ und nicht „wie die gewünschte Funktionalität zu realisieren ist“.



# Domänenanalyse

Definition

In der **Domänenanalyse (domain analysis)** wird das Anwendungsgebiet hinsichtlich grundlegender Fragen untersucht: Was sind wesentliche Vorgänge, Datenmodelle, Konzepte, Anwender/Stakeholder usw., die bei der Entwicklung von Lösungen in dem Gebiet eine Rolle spielen?

- Insbesondere wichtig bei Anstrebung von Wiederverwendung, mehreren Projekten in der Domäne oder Aufbau einer Produktlinienarchitektur.
- Oft in Zusammenhang damit: Stakeholderanalyse
- Ziel: Wissen (z.B. von Experten in dem Bereich) sichtbar machen, dokumentieren und strukturieren.
- Ggf. auch Betrachtung bereits existierender Systeme (und deren Architekturen)
- Gemeinsamkeiten und Unterschiede verschiedener Systeme in dem Bereich aufzeigen
- Ergebnis: Domänenmodell, strukturierte (ggf. semiformale) Beschreibung von dem Anwendungsgebiet, Informationsmodellen, Abläufen usw.

# Lastenheft und Pflichtenheft

**DIN 69905, "Projektabwicklung, Begriffe"**

- Das **Lastenheft** beschreibt ergebnisorientiert die Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers.
- Das **Pflichtenheft** detailliert die Auftraggebervorgaben und beschreibt das Realisierungsvorhaben unter Berücksichtigung konkreter Lösungsansätze.

# Lastenheft

Definition

Das **Lastenheft** führt grob alle fachlichen Anforderungen auf, die das fertige (Software-)Produkt aus Sicht des Auftraggebers erfüllen soll.

Es ist das grundlegende Dokument im Entwicklungsprozess.

**Beispielstruktur eines Lastenhefts:**

1. **Ziel**: Welche Ziele sollen mit dem Produkt erreicht werden?
2. **Einsatzbereich**: Welche Anwendungsbereiche und Zielgruppen werden addressiert?
3. **Funktionen**: Was sind die Hauptfunktionen aus Sicht des Auftraggebers? (Verwende systematische Nummerierung/Kennung für spätere Referenzierbarkeit)
4. **Daten**: Welche Arten von Daten sind für die Software von zentraler Bedeutung. (z.B. Benutzerdaten, Produktdaten, )
5. **Leistungsmerkmale**: Welche Anforderungen gibt es an Daten und Funktionen bezüglich Datenvolumen, Antwortzeiten, Durchsatz, Genauigkeit, usw.?
6. **Qualitätsanforderungen**: Welche Anforderungen gibt es z.B. in Hinblick auf Zuverlässigkeit, einfache Bedienung, Effizienz, Wartbarkeit?
7. **Ergänzungen**: Weitere wichtige Punkte, z.B. technische Randbedingungen?

# Lastenheft

## Software-Engineering-Projekt

### P.3 Kunde: Erstellen eines Lastenhefts

Entsprechend Ihres Kundenthemas möchten Sie eine Softwareentwicklung in Auftrag geben. Als Grundlage für eine Ausschreibung bzw. für die Diskussion mit potenziellen Auftragnehmern ist ein Lastenheft zu erstellen.

a) Überlegen Sie, welche groben Vorstellungen Sie an das Ergebnis haben. Machen Sie ein Brainstorming, recherchieren Sie, welche Lösungen es (z.B. bei anderen Firmen in der Branche) gibt usw.

b) Was muss unbedingt geliefert werden? Welche Punkte sind optional und können evtl. verhandelt werden?

c) Optional: Erstellen Sie ein kurze Präsentation (2-3 Folien), mit der Sie einen groben Überblick über Ihr Vorhaben geben können (z.B. Ihrer Firmenleitung).

d) Welche Arbeitsschritte sind für die Erstellung des Lastenheftes erforderlich? Wie organisieren Sie sich im Team?

e) Senden Sie Ihr Lastenheft rechtzeitig dem potenziellen Lieferanten, damit er sich auf ein erstes Treffen mit Ihnen vorbereiten kann.

# Lieferant: Analyse des Lastenhefts

## Software-Engineering-Projekt

### P.4 Lieferant: Analyse des Lastenhefts

Von einem potenziellen Kunden haben Sie im Rahmen einer Ausschreibung ein Lastenheft für eine zu entwickelnde Softwarelösung erhalten. Da das Thema genau in Ihren Kompetenzbereich fällt und sechs Mitarbeiter bis Anfang Februar noch nicht ausgelastet sind, möchten Sie diesen Auftrag unbedingt erhalten.

a) Untersuchen Sie das Lastenheft. Lässt sich die geforderte Lösung in der zur Verfügung stehenden Zeit realisieren? Wo sollten Abstriche gemacht werden?

b) Haben Sie eigene Ideen zu dem Thema, die Sie dem Kunden schmackhaft machen wollen?

c) Bereiten Sie sich auf das erste Treffen mit dem Kunden vor. Sie wollen zeigen, dass Sie die richtige Entwicklungsfirma für das Projekt sind, gleichzeitig aber auch nur Ergebnisse versprechen, die Sie hinterher tatsächlich liefern können.

# Erstes Treffen Kunde-Lieferant

## Software-Engineering-Projekt

### P.5 Kunde-Lieferant: Erstes Treffen

Zwischen dem Kunden und dem Lieferanten kommt es zu einem ersten persönlichen Treffen. Da der Kunde eine gewisse Offenheit bezüglich Details der zu entwickelnden Lösung besitzt, hat er vom Lieferanten noch kein Pflichtenheft oder konkretes Angebot eingefordert. Statt dessen möchte er mit dem Lieferanten das Lastenheft besprechen bzw. dessen Ideen vorgestellt bekommen, um dann ggf. weitere Schritte zu planen.

a) Was sind Ihre gegenseitigen Erwartungen? Besteht Aussicht auf einen Vertragsabschluss? Was sind die nächsten Schritte bis zu einem konkreten Angebot / Auftrag?

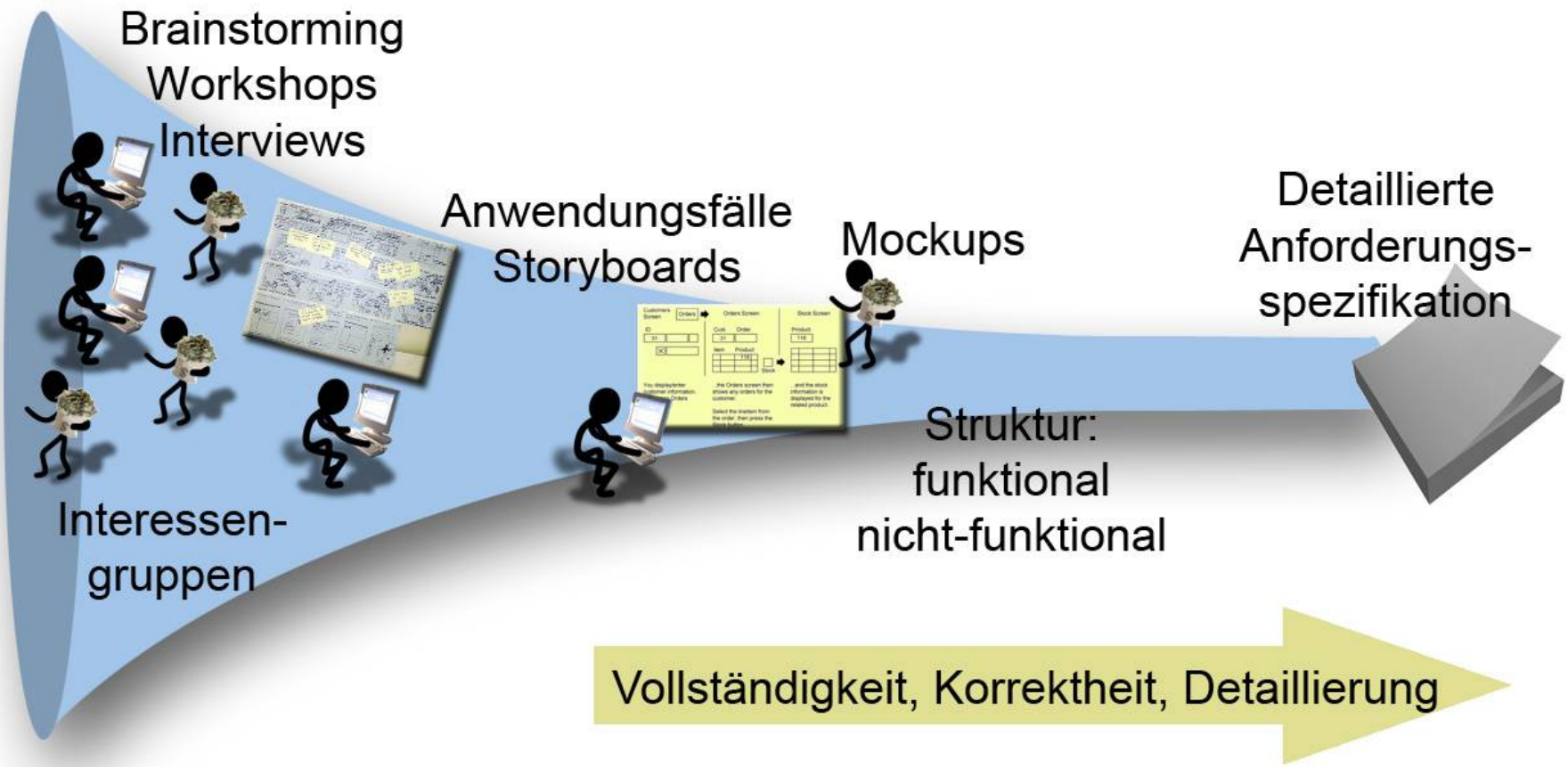
b) Idealerweise können Sie sich darauf einigen, dass der Lieferant auf Basis des Lastenheftes und ggf. besprochener Änderungen ein Pflichtenheft erstellt, welches Grundlage des Vertrages wird.

# Anforderungsanalyse

**Systematische Anforderungsanalyse**

Woher kommt der 'Input'?
Was zeichnet ein gutes Lastenheft / Pflichtenheft aus?
Wie kann eine hohe Qualität sichergestellt werden?

# Anforderungsanalyse Vorgehen



# Pflichtenheft

Definition

Das **Pflichtenheft** definiert die Anforderungen an das zu liefernde (Software-)Produkt. Basierend auf dem Lastenheft wird es vom Auftragnehmer in Abstimmung mit dem Auftraggeber durch Ausarbeitung weiterer Anforderungsdetails erstellt. Das Pflichtenheft ist verbindliche Grundlage des abzuschließenden Vertrages.

**Beispielstruktur eines Pflichtenhefts** (nach Balzert, Softwaretechnik)**:**

1. **Ziele:** Was muss das Produkt unbedingt leisten? Was ist wünschenswert? Was ist nicht Produktbestandteil?
2. **Einsatzbereich:** Anwendungsbereiche? Zielgruppen?
3. **Umgebung:** Welche Hardware- / Softwareumgebung muss unterstützt werden?
4. **Funktionen**: Detaillierte, systematische Auflistung aller wesentlichen Produktfunktionen
5. **Daten**: Detaillierte, systematische Auflistung aller wesentlichen zu handhabenden Daten
6. **Leistungsmerkmale**: Auflistung aller nicht-funktionalen Anforderungen (Performanz, Datenvolumen, Skalierbarkeit, Bedienbarkeit, Zuverlässigkeit usw.)
7. **Benutzeroberfläche:** Grundlegende Anforderungen an die Benutzeroberfläche?
8. **Qualitätsziele:** Wie wird die Produktqualität sichergestellt? Welche Standards/Normen werden befolgt?
9. **Testszenarien:** Anhand welcher Testfälle wird die Erfüllung der Anforderungen überprüft?
10. **Entwicklungsumgebung:** In welcher Umgebung / mit welchen Technologien wird die Software entwickelt?
11. **Ergänzungen:** Weitere wichtige Punkte, z.B. Lizenzen, Patente, 3rd Party Komponenten
12. **Glossar:** Definition von verwendeten Abkürzungen und Fachbegriffen

# Lieferant: Pflichtenheft

Übung

## Software-Engineering-Projekt

### P.6 Lieferant: Pflichtenheft

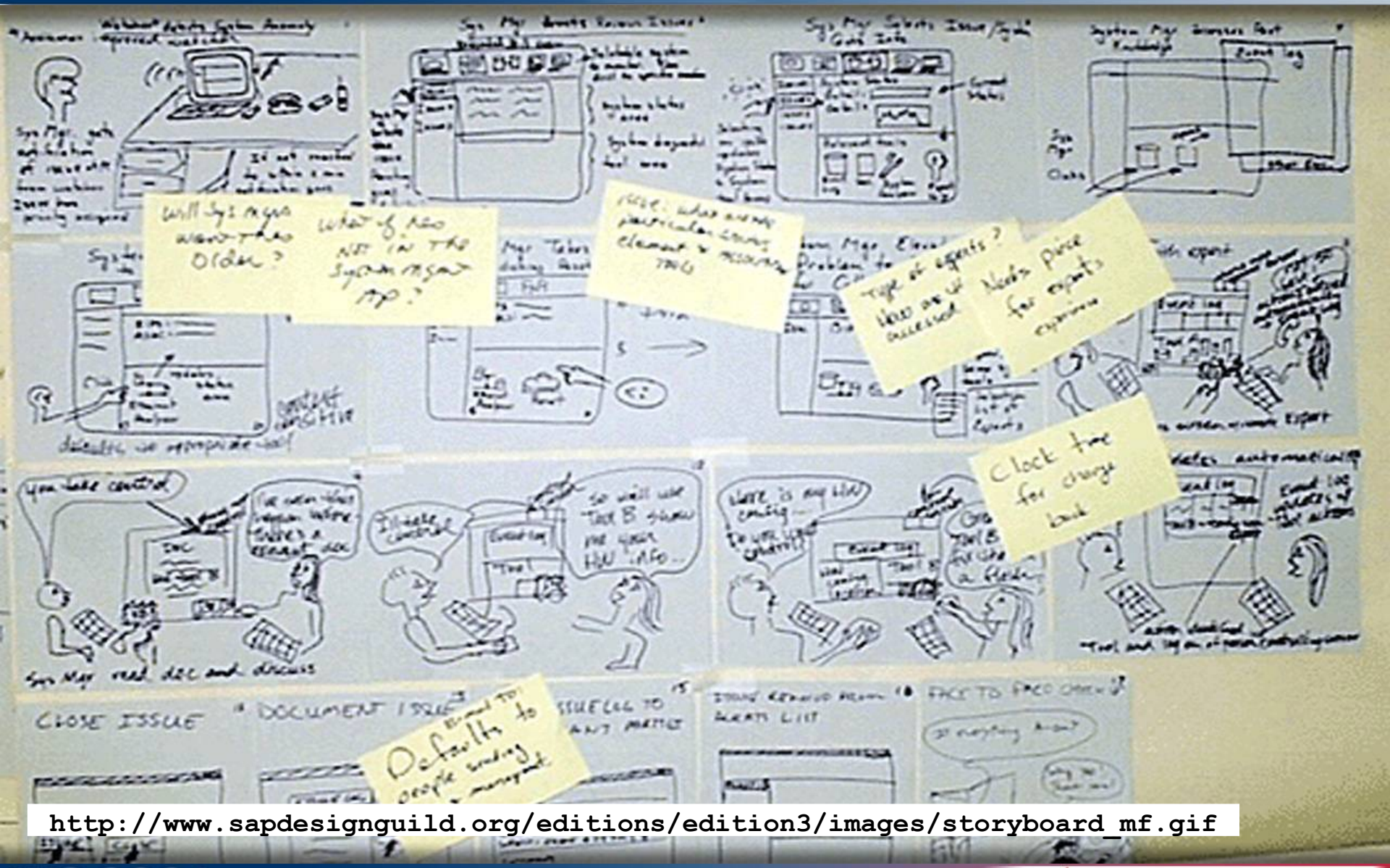
Erstellen Sie auf Basis des Lastenheftes und des Gespräches mit Ihrem Kunden ein Pflichtenheft. Senden Sie das Pflichtenheft dem Kunden rechtzeitig, damit dieser vor dem nächsten Treffen noch genügend Zeit für die Durchsicht hat.

# Ermittlung von Anforderungen

**Ermittlung der Anforderungen (Requirements elicitation):**

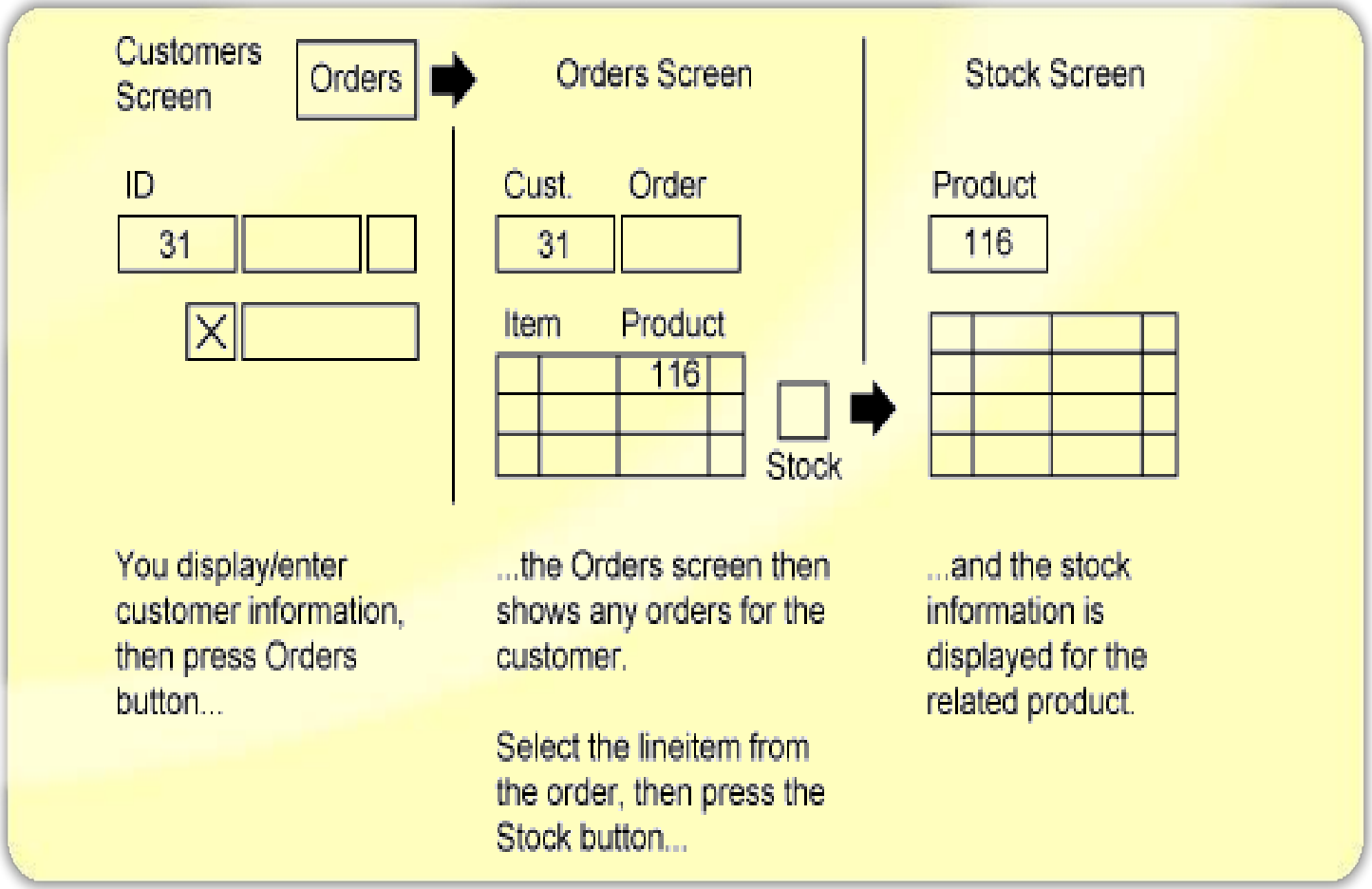
- Verschiedene Interessengruppen (stakeholders) einbeziehen:
  - z.B. Anwender, Administrator, Manager ...
  - Interviews, Diskussionen
- Verschiedene Sichten (Viewpoints) berücksichtigen:
  - Anwendersicht: z.B. einfache Bedienbarkeit
  - Administrator: z.B. Sicherheit, einfache Installation und Verwaltung
  - Manager: z.B. Total cost of ownership
  - ...
- Verschiedene Techniken verwenden:
  - Workshops
  - Brainstorming
  - Anwendungsfälle (typische Funktionsabläufe), Storyboards
  - (UI) Mockups
- Umfeldanalyse:
  - Andere Produkte auf diesem Markt, Wettbewerber?
  - Trends (Technologien, Benutzererwartungen)
  - Richtlinien und Standards (im Unternehmen, im Marktumfeld)

# Storyboard (Brainstorming)




http://www.sapdesignguild.org/editions/edition3/images/storyboard_mf.gif

# Storyboard




http://sqltech.cl/doc/dev2000/guide21/guia3.gif

# Anwendungsfälle (Use Cases)

Definition

Ein **Anwendungsfall (Use case)** beschreibt einen typischen Vorgang aus mehreren zusammenhängenden Arbeitsschritten, die von einem **Akteur (Actor)** durchgeführt werden, um ein bestimmtes Ziel zu erreichen / eine Aufgabe zu erledigen.

**Beispiel:**

**#15 Adding a User**
A new user needs an account to access the system. A new user is added to the system by creating an account consisting of a user name, password, and home directory.
*Priority: 5* *Estimation: Easy*

**#16 Logging onto the System**
A user logs onto the system by entering their username and password. If they enter an incorrect value for either field, they are provided an error message indicating. if they enter three incorrect values, their account is locked and the user must inform the IT department to restore their account.
*Priority: Medium* *Estimation: 3*

**#135 Paying for a Book**
A customer may pay for a book using a credit card (VISA,AMEX,MC), PayPal, or by check.
*Priority: 12* *Estimation: 1*

# UML Use Case Diagram

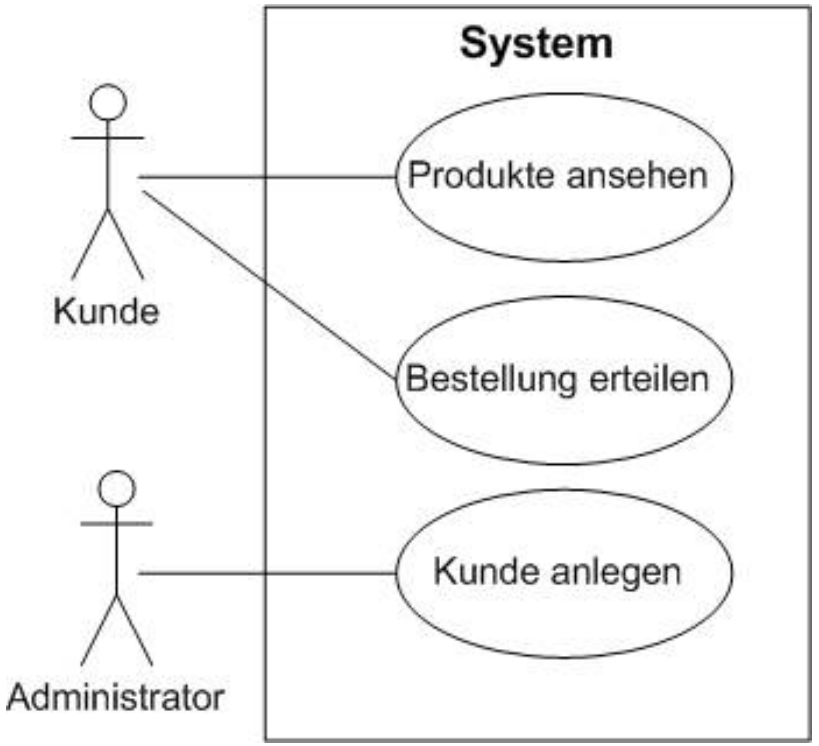
## UML Use Case Diagram (Anwendungsfalldiagramm)

**System Boundary (Systemgrenze):**
System, das die benötigten Anwendungsfälle bereitstellt und mit dem die Anwender interagieren.

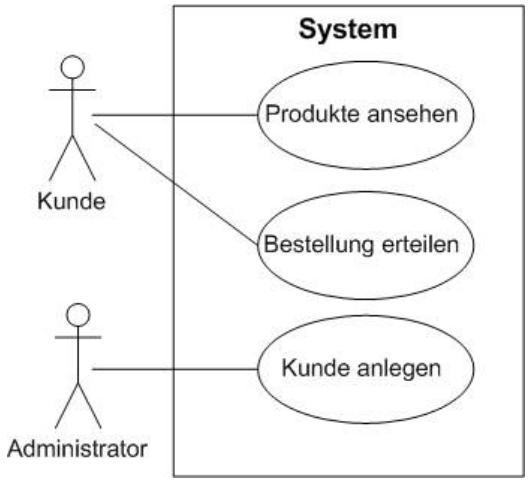
**Actor (Akteur):**
Typ oder Rolle, die ein externer Benutzer oder ein externes System während der Interaktion mit einem System einnimmt.

**Use Case (Anwendungsfall):**
Eine abgeschlossene Menge von Aktionen, die von einem System bereitgestellt werden und definierten Nutzen für einen oder mehrere Actors erbringen.

**Association (Assoziation):**
Beziehung zwischen Akteuren und Anwendungsfällen.



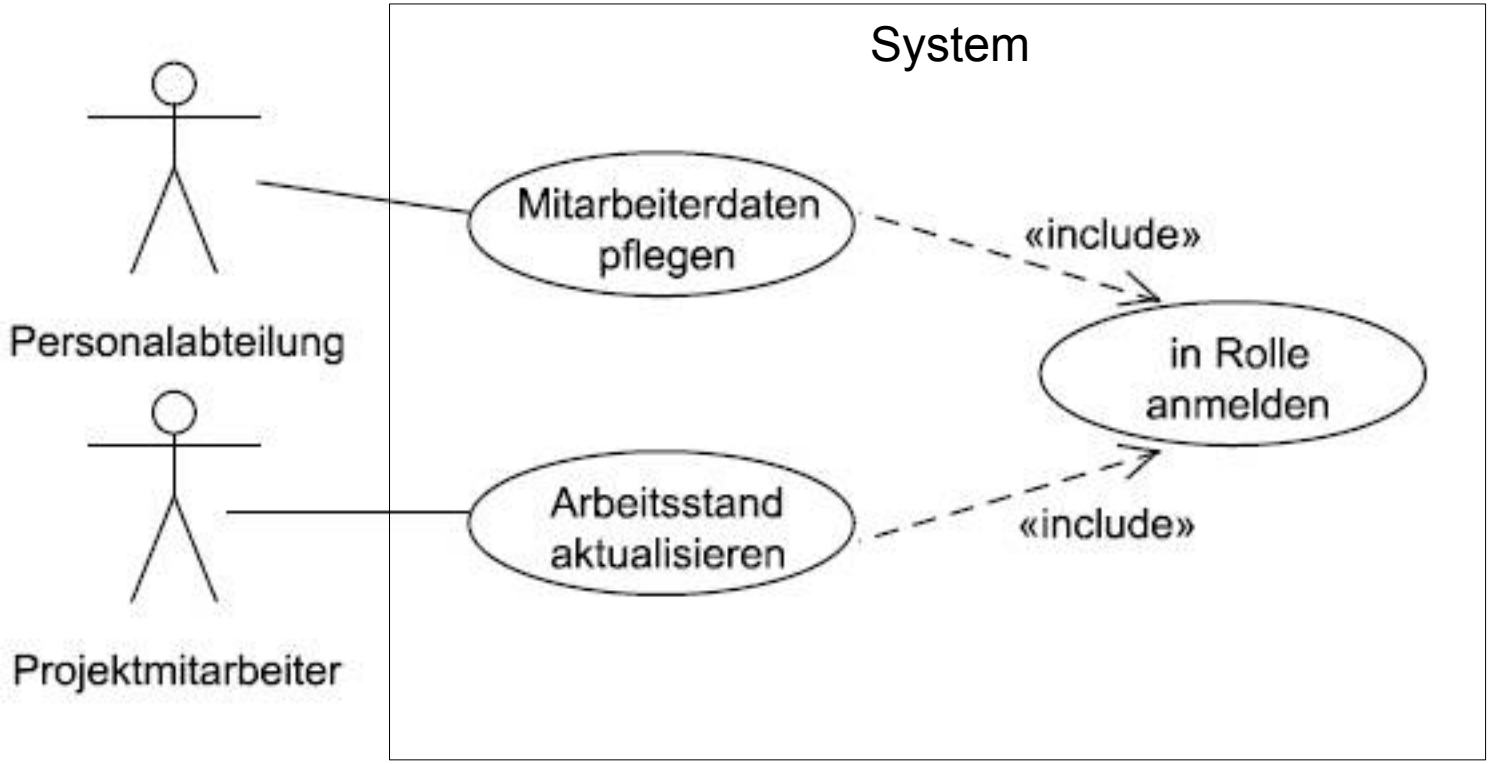
# Use Case Details - Checkliste




Jeder Use Case sollte detailliert dokumentiert werden:

- Name (typisch: Substantiv+Verb)
- Eindeutige Kennung/Nummer
- Priorisierung (wie wichtig ist der Use Case?)
- Kurze Beschreibung
- Beteiligte Actors
- Vorbedingungen (z.B. System ist gestartet, in Modus xy...)
- Typischer Ablauf (ggf. mit Alternativen, z.B. Fehlerfall)
- Nachbedingungen (gewünschtes Ergebnis)
- Referenzen: Verweise auf weitere Informationen/Quellen und zu anderen Use Cases
- Autor, Version, Datum
- Fachverantwortlicher (Kundenseite)

# Use Case Hierarchie



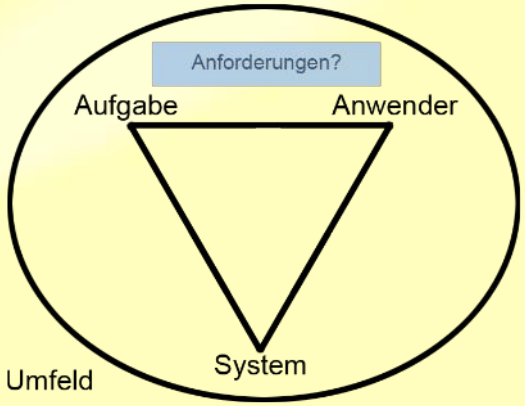
# Benutzerorientierte Analyse

- Benutzeranalyse
  - Modell von künftigen Anwendern entwickeln
- Aufgabenanalyse
  - Welche Aufgaben (mit welchem Ziel) sollen vom Anwender mit dem System erledigt werden?
  - Anwendungsszenarien, typische Abläufe
- Verhalten und Wünsche des Anwenders
  - Anwenderverhalten bei bisherigen Systemen, Prototypen, beobachten
  - Interviews, Fragebögen
  - Anwenderverhalten bei Konkurrenzprodukten (kompetitive Analyse)
- Usability Ziele
  - Frühe Festlegung von Usability Zielen
  - Frühe/regelmäßige Überprüfung mit Mockups und Prototypen

# User Story

Eine **User Story** ist ein knappe (ein, zwei Sätze) aus Anwendersicht in Alltagssprache formulierte Anforderung, die eine gewünschte Funktion beschreibt und begründet. User Stories sind insbesondere in der agilen Softwareentwicklung üblich.

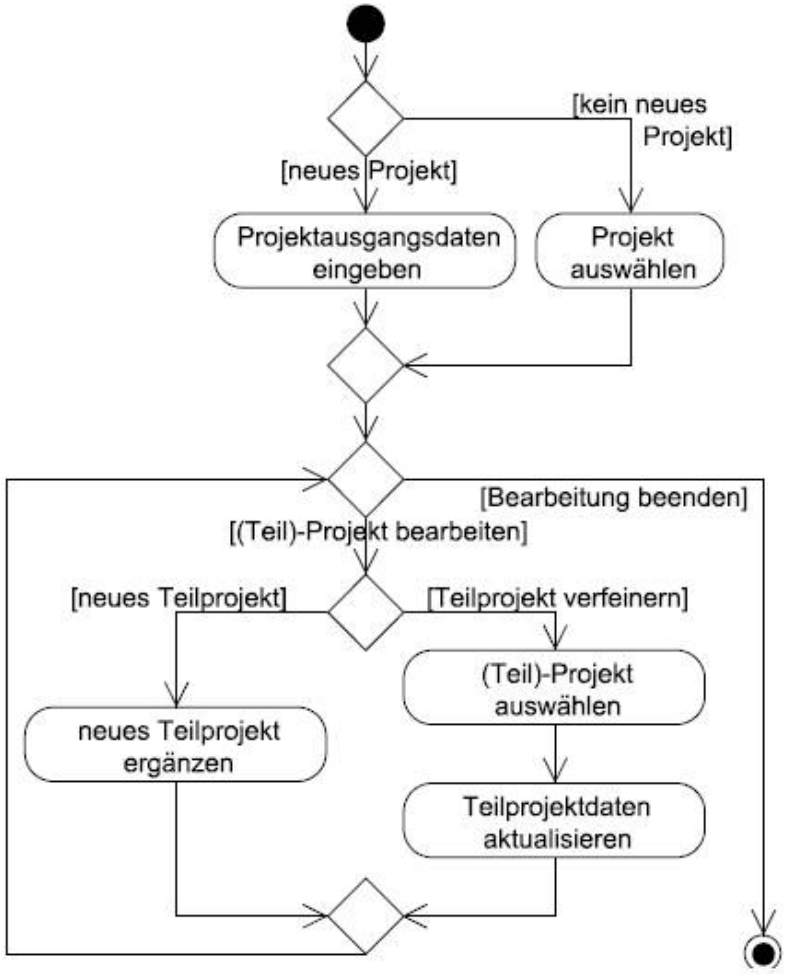
- *Der Support-Mitarbeiter möchte jederzeit eine Übersicht aller Bestellungen des Kunden anzeigen können, um zu wissen, welche Produkte der Kunde seit wann verwendet.*
- *Wenn der Anwender das Fenster schließt und die Datei noch nicht gespeichert ist, soll eine Sicherheits-/Speichern-Abfrage kommen, damit Dateiänderungen nicht versehentlich verloren gehen.*




User-Story-Template
(Mike Cohn: User Stories Applied. For Agile Software Development)
*"As a (role) I want (something) so that (benefit)."*

# UML Activity Diagram

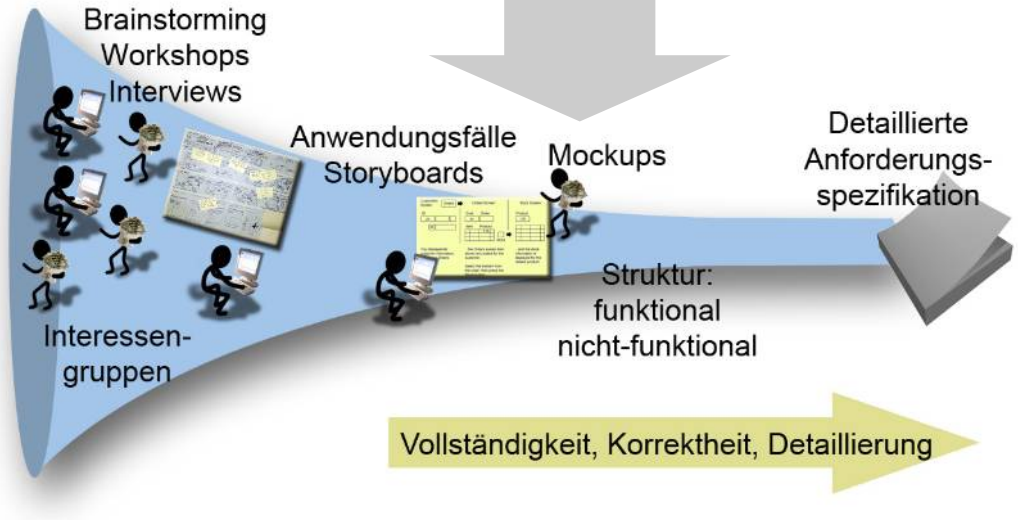
Um Abläufe zu beschreiben lassen sich **UML Aktivitätsdiagramme (Activity Diagrams)** einsetzen.




Außerdem möglich:
Hierarchische Gliederung → zusätzliche Diagramme für die detaillierte Beschreibung einzelner Aktivitäten

# Lastenheft und Pflichtenheft

**Wie können Anforderungen angemessen und vollständig dargestellt werden?**




Anwendersicht

Entwicklersicht

# Funktionale Anforderungen

**Funktionale Anforderungen**

- Wozu soll die Software dienen?
  - Aus Sicht des Kunden/der Benutzer (Anwender, Administrator etc.)
- Entscheidend: "Was" macht die Software – nicht "wie" macht sie es.
- Strukturierte Liste aller wesentlichen Funktionen / Features
- ggf. zusätzlich: typische, anschauliche Anwendungsfälle
- Angemessene Detaillierungsebene
  - die wesentlichen Abläufe, Sichten und Interaktionsschritte
  - aber nicht jeden einzelnen Mausklick

# Nicht-funktionale Anforderungen

**Nicht-funktionale Anforderungen (Qualitätsmerkmale)**

- Leistung: Performanz, Durchsatz, Skalierbarkeit, Ressourcenbedarf
- Benutzersicht: Einfache Bedienung (Usability), Look&Feel, einfache Installation
- Zuverlässigkeit, Ausfallsicherheit, Verfügbarkeit
- IT-Sicherheit (Security), Datenschutz
- Technologieumfeld, Kompatibilität, Standards: z.B: unterstützte Browser, Betriebssysteme, PC-Anforderungen
- Total cost of ownership: Wartbarkeit, Erweiterbarkeit, Portierbarkeit

**Sonstige Anforderungen im Lasten-/Pflichtenheft**

- Dokumentation, Training, Einarbeitung
- Abnahmekriterien, geforderte Testprotokolle
- Vorgehen bei Übergabe/Deployment
- Nachweis der Qualitätssicherung
- Wartung und Support (Garantien)

# Anforderungsfehler

Übung

**Übung**

**2.1 Anforderungsfehler**

Anforderungsfehler, d.h. falsch erfasste Anforderungen, unterscheiden sich in verschiedener Hinsicht von Softwarefehlern, die während des Entwurfs und der Implementierung auftreten.

a) Diskutieren Sie die Unterschiede.

b) Wie kann sichergestellt werden, dass es möglichst keine Anforderungsfehler gibt bzw. die Konsequenzen begrenzt werden?

# Anforderungsfehler

Übung

**Anforderungsfehler**

- Entstehen früh im Prozess → potenziell hohe Kosten, da in der Folge viel Arbeit davon abhängt.
- Werden nicht beim Testen im Rahmen der normalen Softwareentwicklung erkannt, sondern potenziell erst bei der Endabnahme / im späteren Betrieb.
- Kunde/Auftraggeber trägt i.d.R. die Hauptverantwortung für derartige Fehler

**Wie vermeiden?**

- Saubere Anforderungsanalyse
- Kunden eng einbeziehen
- ggf. Iterationen / Zwischenchecks
- Werkzeuge zum Anforderungsmanagement

# Lastenheft und Pflichtenheft

**Was zeichnet ein gutes Lastenheft / Pflichtenheft aus?**

- Verständlichkeit: Klare Struktur, zusammenhängender, "lesbarer" Text, Motivation/Hintergründe/Erläuterungen; Zusammenfassen,was zusammengehört.
- Vollständigkeit
- Genauigkeit und Korrektheit: Möglichst präzise Angaben, Keine Widersprüche, Prioritäten, sorgfältig überprüft, Glossar/Begriffsdefinitionen
- Effizienz: Möglichst wenig Redundanz, keine überflüssigen Angaben (die evtl. in andere Dokumente gehören, z.B. Implementierungsdetails)
- Nachverfolgbarkeit: Kennungen für Anforderungen, Dokumentenhistorie

Typisch: Verwendung von vorgegebenen, unternehmenseigenen Dokumentenschablonen (Templates), um sicherzustellen, dass alle Punkte erfasst werden.

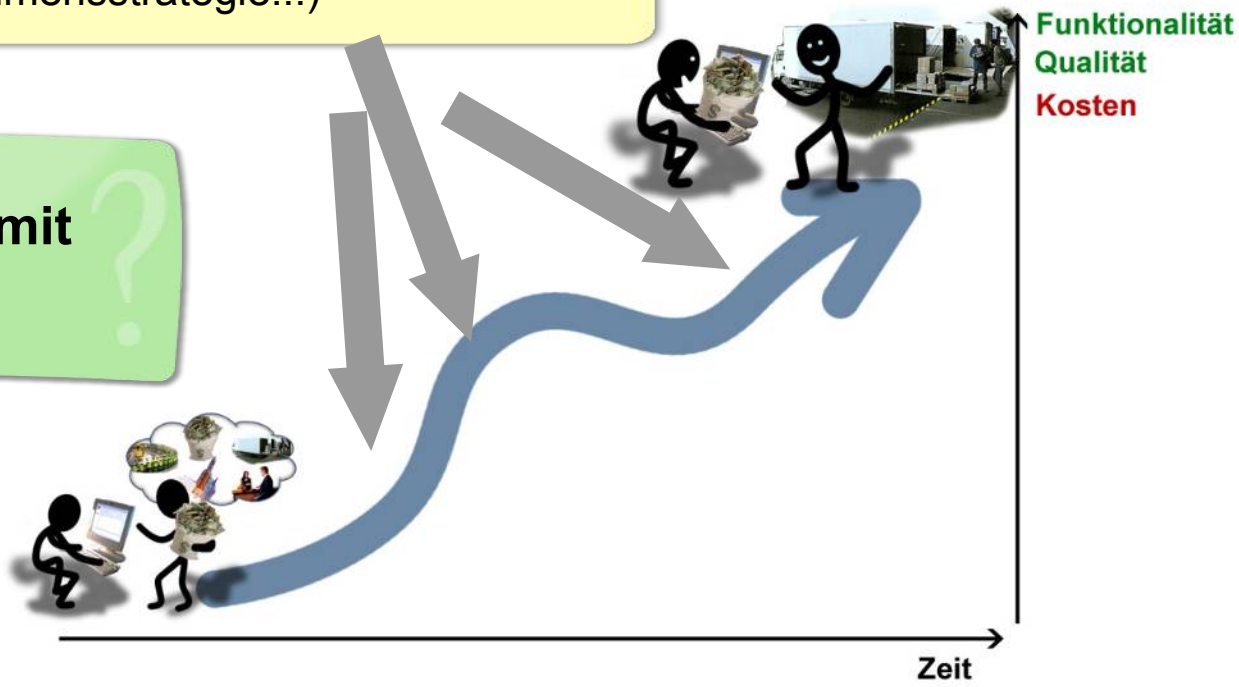
Das Pflichtenheft ist Bestandteil des Vertrages (und somit ggf. maßgeblich bei rechtlichen Auseinandersetzungen). Sorgfalt ist sehr wichtig!

# Anforderungsmanagement

Anforderungen können sich über die Projektlaufzeit und auch nach Release des Produktes verändern:

- Fehler bei der Anforderungsanalyse werden erkannt
- Neue, unvorhergesehene Kundenwünsche
- Anforderungen von außen (neue Standards, Änderung der Produkt-/Unternehmensstrategie...)

**Wie soll man damit umgehen?**



# Anforderungsmanagement

Anforderungen können sich über die Projektlaufzeit und auch nach Release des Produktes verändern:

- Fehler bei der Anforderungsanalyse werden erkannt
- Neue, unvorhergesehene Kundenwünsche
- Anforderungen von außen (neue Standards, Änderung der Produkt-/Unternehmensstrategie...)

gewünscht:

- Systematische Verfolgung von Anforderungen und Änderungen
- definierter Entscheidungsprozess (mit Kosten-Nutzen-Abwägung)
- Möglichkeit zur Umplanung des Projektes

**Flexibilität hängt auch vom Vorgehensmodell ab!**

# Anforderungsmanagement: Werkzeuge

Übung

**Übung**

**2.2 Werkzeuge für Anforderungsmanagement**

a) Recherchieren Sie im Internet nach Werkzeugen für Anforderungsanalyse und Anforderungsmanagement.
b) Was für besondere Möglichkeiten bieten diese Werkzeuge, z.B. im Vergleich mit normalen Büroanwendungen wie z.B. Textverarbeitung.
c) Diskutieren Sie Für und Wider des Einsatzes dieser Werkzeuge in Ihrem Projekt.

# Kunde: Review Pflichtenheft

**Software-Engineering-Projekt**

## P.7 Kunde: Review Pflichtenheft

Begutachten Sie das von Ihrem Softwarelieferanten erstellte Pflichtenheft. Sind aus Ihrer Sicht alle wichtigen Punkte hinreichend detailliert erfasst? Gibt es Punkte, die vor Vertragsabschluss noch geklärt oder überarbeitet werden müssen?

# Kunde-Lieferant: Vertragsabschluss?

## Software-Engineering-Projekt

### P.8 Kunde-Lieferant: Vertragsabschluss?

Der Kunde und Lieferant möchten das Pflichtenheft bzw. evtl. erforderliche Änderungen besprechen. Wenn die Besprechungen erfolgreich verlaufen, kann (nach ggf. noch kleineren Änderungen am Pflichtenheft) der Kunde den Auftrag an den Lieferanten erteilen. Falls es noch größere Differenzen gibt, müssen weitere Treffen und Diskussionen geplant werden.