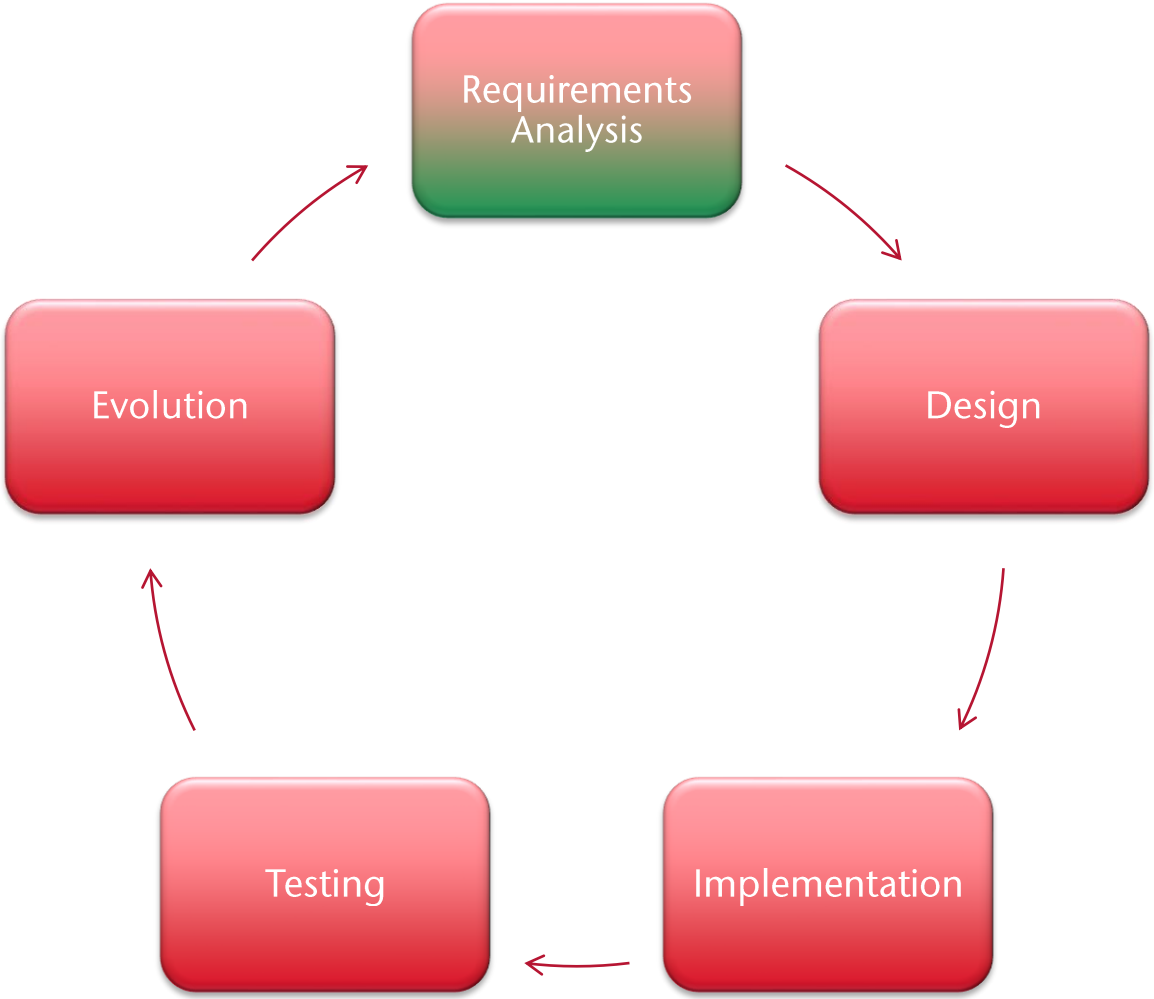
Media Engineering
Requirements Engineering

R. Weller

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Der Software Development-Lifecycle



Übersicht

- Warum braucht man Anforderungen?
- Was sind überhaupt Anforderungen?
- Wie findet man Anforderungen?
- Wie hält man Anforderungen fest?

Zur Erinnerung: Beispiel KSC

- Der Fußballverein KSC wollte ein Verfahren zur Gesichtserkennung im Stadion Testen
- Entwickelt in einem Forschungsprojekt am Karlsruher Instituts für Technologie (KIT)
- Geplanter Test bei drei Heimspielen wurde nach Fanprotesten abgesagt

Stellenwert der Anforderungen

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements … No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.

Fred Brooks,
1987

- Die Anforderungen des Kunden an die Software sind die wichtigsten Informationen in einem Software-Projekt.

Kostenfortpflanzung eines Fehlers

		Zeitpunkt der Erkennung des Fehlers				
		Anforderungen	Architektur	Entwurf	Test	Release
Zeitpunkt der Einführung des Fehlers	Anforderungen	1x	3x	5-10x	10x	10-100x
	Architektur	-	1x	10x	15x	25-100x
	Implement.	-	-	1x	10x	10-25x

McConnell, Steve (2004). *Code Complete* (2nd ed.)

- Kosten für die Behebung von Fehlern abhängig von ihrer Verweildauer in der Software
  - Fehler möglichst frühzeitig erkennen
  - In allen Phasen aktiv gegen Fehler vorgehen
- 71% aller IT-Projekte scheitern wegen einer schlechten Anforderungsanalyse

Bremen

CG VR

Was sind überhaupt Anforderungen? Requirements

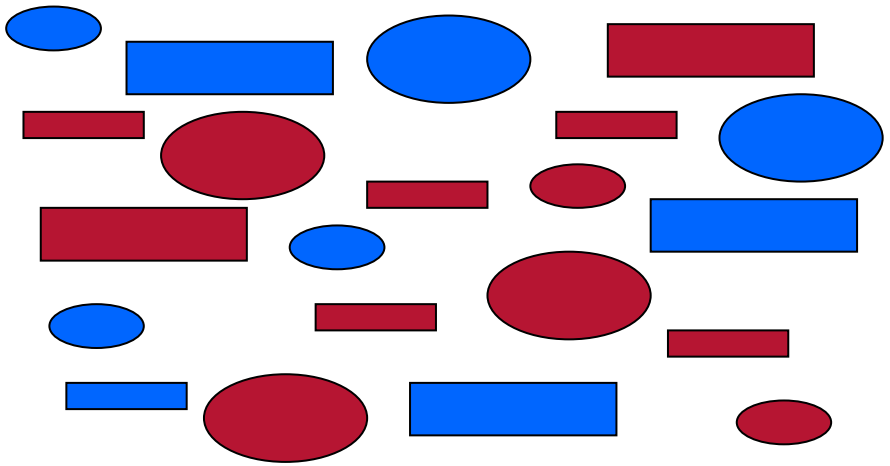
Definition (Nach IEEE Std 610.12-1990)

Requirement

- (1) A condition or capability needed by a user to solve a problem or achieve an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- (3) A documented representation of a condition or capability as in (1) or (2).

Welche Arten von Anforderungen gibt es?

- Kategorisierung abhängig vom Blickwinkel
  - Unterteilung nach Anforderungssteller
    - Offene Anforderungen vom Kunden, latente Anforderungen, Entwickleroptionen
  - Verschiedene Unterteilungen nach Art der Anforderung
    - Hart vs. weich, funktional vs. nichtfunktional



Einteilung nach von Anforderungsteller

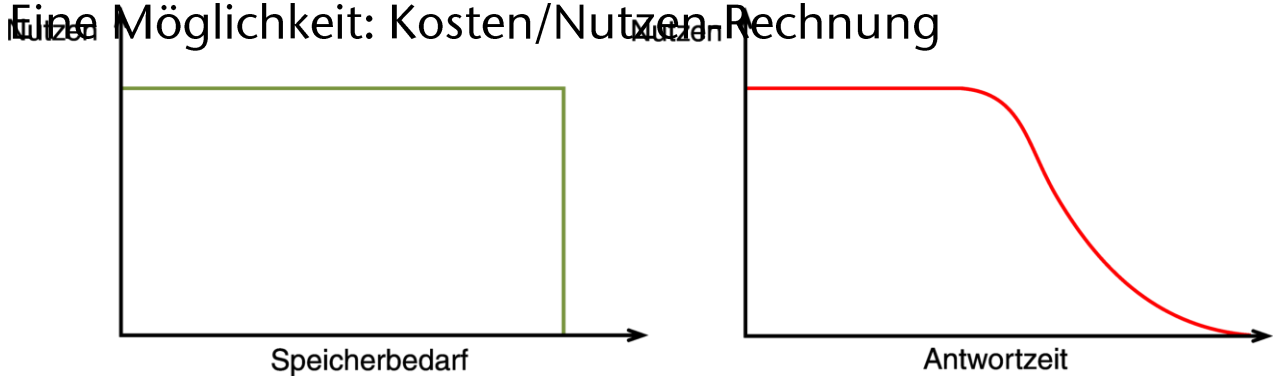
- Offene Anforderungen
  - vom Kunden selbst brauchbar formuliert
- Latente Anforderungen
  - dem Kunden nicht bewusst
- Entwickler-Optionen
  - Der Kunde hat den Punkt offengelassen, es ist ihm gleich
  - Auch diese Anforderungen müssen dokumentiert werden!

Wer ist überhaupt der Kunde?

- Oft nicht so eindeutig wie man denkt:
  - Der, der es bezahlt (Auftraggeber, Chef, Endkunde)
  - Aber auch jeder, der nachher das Produkt verwendet (Benutzer)
  - Jeder, der unter dem Output zu leiden hat
  - Diejenigen, die es auf den Markt bringen
    - Z.B. Verkäufer, Support-Dienstleister
  - Eventuell auch diejenigen, die bislang Konkurrenzprodukte verwenden
- Drei Kategorien von Benutzern (Eason, 1987):
  - Primäre: Verwenden das Produkt regelmäßig selbst
  - Sekundäre: Gelegenheits- oder indirekte Benutzer
  - Tertiäre: Von der Einführung des Produktes Betroffene
    - z.B. Verkäufer, Support

Einteilung nach Art der Anforderung: Hart vs. weich

- Anforderungen dienen nicht zuletzt als Referenz bei der Prüfung der Software.
  - Dazu müssen die Anforderungen objektivierbar sein. Solche Anforderungen nennen wir harte Anforderungen
- Lehrbuchbeispiele enthalten typischerweise harte Anforderungen.
  - Z.B.: Mindestens 3 C++-Klassen, Obergrenze für Umfang der Software
- In der Praxis sind fast alle Anforderungen weich
  - Schwer zu erheben und zu formulieren
  - Eine Möglichkeit: Kosten/Nutzen-Rechnung



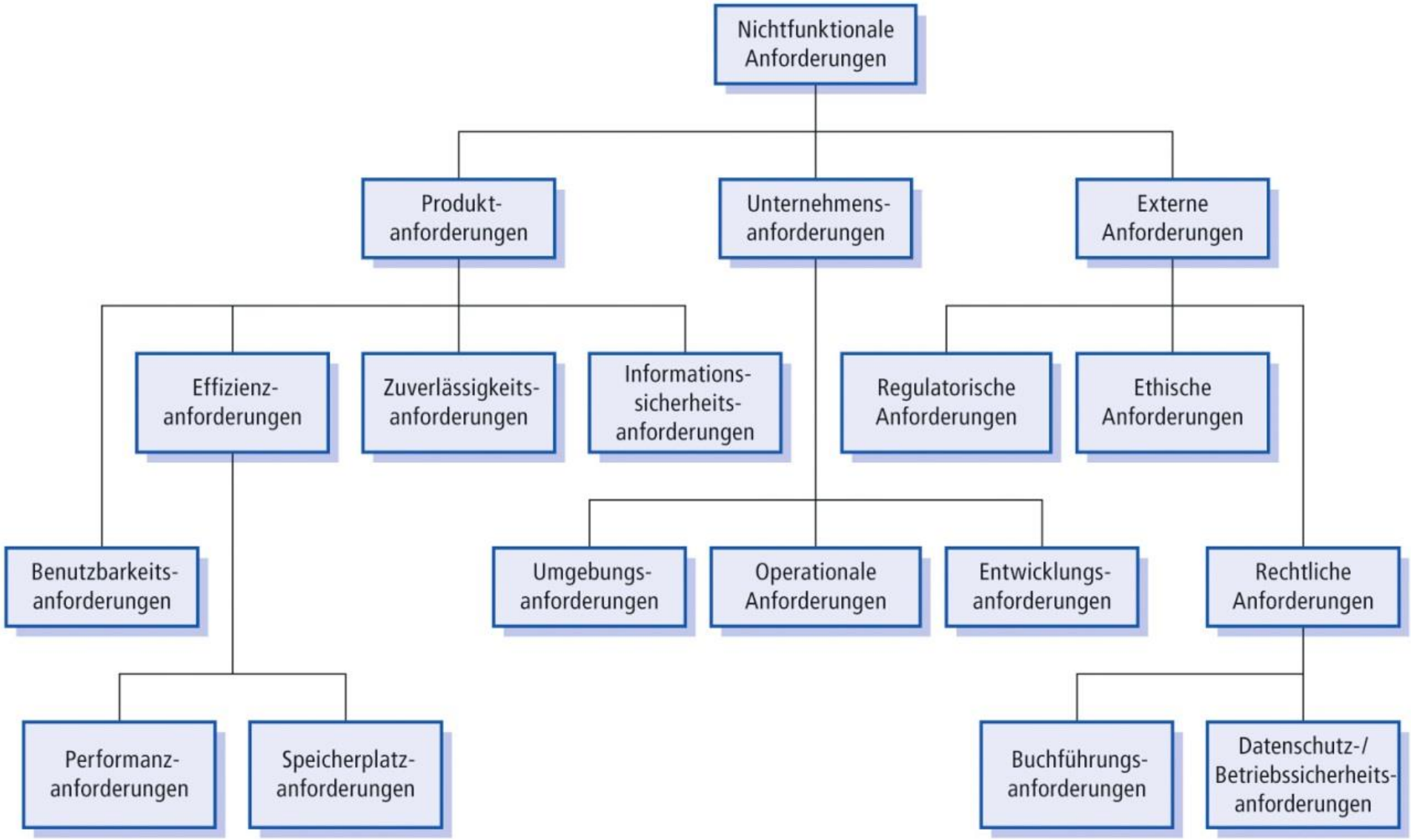
Weitere Kategorien von Anforderungen: Funktional vs. Nichtfunktional

- Funktionale Anforderungen
  - Funktion der Software = Beziehung zwischen Ein- und Ausgaben
  - Alle Anforderungen, die sich auf diese Funktion beziehen, sind funktionale Anforderungen, alle anderen sind nichtfunktionale Anforderungen (non-functional requirements, NFRs).
- Die Kurzbeschreibung eines Systems, also eine auf wenige Worte reduzierte Anforderung, ist stets eine grobe Charakterisierung der Funktion.
  - Beispiel: „Das System sichert nachts die Inhalte aller Festplatten.“
- Praktisch alle Aussagen zur Wartbarkeit sind nichtfunktional.
  - Beispiel: „Das System muss leicht portierbar sein.“

Funktionale vs nichtfunktionale Anforderungen

- Die Abgrenzung funktional – nichtfunktional ist unscharf
  - Anforderungen, die zwar die Funktion betreffen, aber nicht präzise formuliert werden können, werden vielfach als nichtfunktional eingeordnet
    - Beispiele: Robustheit, Bedienbarkeit
  - Auch das Zeitverhalten wird meist als NFR behandelt.
- Funktionale und nichtfunktionale Anforderungen werden gebraucht.
- Nichtfunktionale Anforderungen sind oft weiche Anforderungen

Nichtfunktionale Anforderungen



Nichtfunktionale Anforderungen

- Die Formulierung nichtfunktionaler Anforderungen bereitet meist große Probleme.
  - NFRs werden entsprechend stiefmütterlich behandelt, meist ganz weggelassen oder nur durch Schlagwörter ausgedrückt.
  - Das ist aber ihrer großen Bedeutung nicht angemessen.
- NFRs lassen sich oft gut mit Hilfe von Normen ausdrücken.
  - Beispiel: DIN EN ISO 9241 (Ergonomics of Human System Interaction)
  - Problem: Kaum einer der Standards bietet harte Vorschriften, deren Einhaltung einfach und objektiv zu prüfen wäre.
- In keinem Falle sollte man auf einen Versuch zur Präzision verzichten!

ISO 9241: Ergonomics of Human System Interaction*

- 3: Visual display requirements
- 4: Keyboard requirements
- 5: Workstation layout and postural requirements
- 6: Guidance on the work environment
- 7: Requirements for display with reflections
- 8: Requirements for displayed colours
- 9: Requirements for non-keyboard input devices
- 110: Dialogue principles (formerly 10)
- 11: Guidance on usability
- 12: Presentation of information
- 13: User guidance
- 14: Menu dialogues
- 15: Command dialogues
- 16: Direct manipulation dialogues
- 17: Form filling dialogues
- And many more

Nicht gesetzlich vorgeschrieben, aber sehr hilfreich (und manchmal vom Kunden gewünscht)

* Bis 2006: Ergonomic requirements for office work with visual display terminals

Beispiel: ISO 9241-10

3.1 Aufgabenangemessenheit

Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.

Empfehlungen:	**mögliche Beispiele:**
Der Dialog sollte dem Benutzer nur solche Informationen anzeigen, die im Zusammenhang mit der Erledigung der Arbeitsaufgabe stehen.	Formatierungen wie z.B. Farbe und Informationen wie z.B. Wochentag, Datum usw. werden nur angezeigt, wenn sie die Erledigung der Arbeitsaufgabe erleichtern.
Die angezeigte Hilfe-Information sollte von der Aufgabe abhängen.	Wenn der Benutzer Hilfe aufruft, zeigt das Dialogsystem Informationen zur gegenwärtigen Aufgabe an (z.B. während des Editierens eine Liste der Editierbefehle). Wenn eine Dialog-Box angezeigt wird und der Benutzer Hilfe aufruft, zeigt das Dialogsystem Informationen zu dieser Dialog-Box an.
Alle Aufgaben, die sinnvollerweise dem Dialogsystem zur automatischen Ausführung übertragen werden können, sollten durch das Dialogsystem ausgeführt werden, ohne den Benutzer damit zu belasten.	Die Positionsmarke wird automatisch auf das erste Eingabefeld positioniert, das für die Arbeitsaufgabe relevant ist. Startprozeduren des Systems laufen automatisch ab.
Bei der Gestaltung des Dialogs sollte der Komplexität der Arbeitsaufgabe unter Berücksichtigung der Fertigkeiten und Fähigkeiten des Benutzers Rechnung getragen werden.	In einem öffentlich zugänglichen Dialogsystem wird dort, wo es eine Reihe alternativer Eingabemöglichkeiten gibt, ein Menü verwendet, um die unterschiedlichen Auswahlmöglichkeiten anzuzeigen.

Richtlinien für Barrierefreiheit

- Web Content Accessibility Guidelines (WCAG 2.0)
  - Perceivable
  - Operable
  - Understandable
  - Robust

Gesetzlich vorgeschrieben für öffentliche Einrichtungen (BIT-V)

Wie findet man Anforderungen?
Requirements Analysis

Definitionen (Nach IEEE Std 610.12-1990)

Requirements Analysis

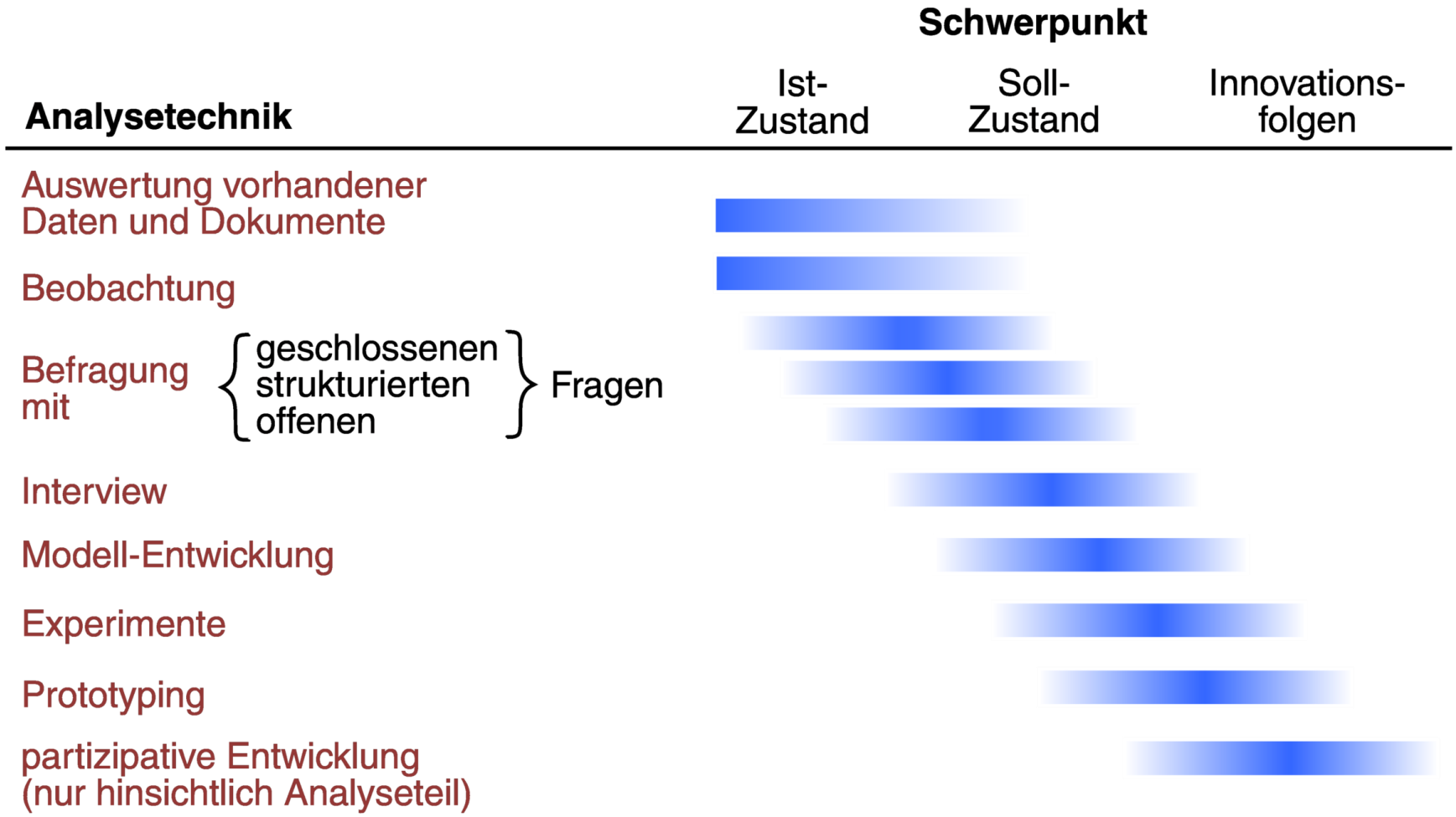
- (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
- (2) The process of studying and refining system, hardware, or software requirements.

Methoden um Anforderungen zu finden

- Man analysiert den Ist-Zustand und definiert einen Soll-Zustand

  - Man fragt den Kunden
  - Man nutzt sein Know-How
  - Man recherchiert
    - Lesen
    - Marktbeobachtung
    - Ethnographische Analyse
  - Man phantasiert Beispiel-Szenarien
  - ...

Techniken der Analyse



Probleme beim Fragen: Beispiel

Sie öffnen also morgens das Schloss am Haupteingang?

Ja, habe ich Ihnen doch gesagt.

Jeden Morgen?

Natürlich.

Auch am Wochenende?

Nein, am Wochenende bleibt der Eingang zu.

Und während der Betriebsferien?

Da bleibt er natürlich auch zu.

Und wenn Sie krank sind oder Urlaub haben?

Dann macht das Herr X.

Und wenn auch Herr X ausfällt?

Dann klopft irgendwann ein Kunde ans Fenster, weil er nicht reinkommt.

Was bedeutet „morgens“? ...

Systematisches Fragen

- „offene“ Fragen:
  - keine Antwortvorgabe
  - vollkommene Freiheit des Befragten
  - Geeignet um Wissen zu prüfen
  - Offenheit beschränkt Vergleichbarkeit
- „geschlossene“ Fragen
  - Fragen und Antworten vorgegeben
  - Bessere Quantifizierung der Antworten
  - Spektrum der Antwortmöglichkeiten kann unvollständig sein
  - Oft verwendet: Likert-Skala
  - Quantitative Auswertung mit Software wie SPSS

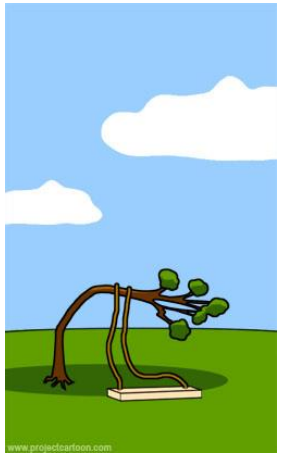
Zustimmung			Ablehnung		
stark	mittel	schwach	stark	mittel	schwach
+3	+2	+1	-1	-2	-3

Exkurs: Wie fragt man richtig?

- Frageformulierung
  - einfache und eindeutige Formulierung
  - kurze prägnante Fragen (möglichst keine Unterfragen) verwenden
  - nicht zu viel bzw. zu wenig Fragen verwenden
  - einfache Sachverhalte ansprechen
  - Vermeidung einer Überbeanspruchung des Befragten
  - Vermeidung von komplizierten Sätzen und unbekannten Begriffen
  - Vermeidung von doppelten Negationen
  - Vermeidung von suggestiven Fragen
  - Verwendung neutraler Fragen
  - Persönlich fragen ist besser als Fragebogen verschicken
- vollstandardisiert
  - explizite Formulierung der Fragen
  - festgelegte Reihenfolge der Fragen
  - kein Spielraum des Interviewers vorhanden
- teilstandardisiert
  - Fragebogengerüst
  - hauptsächlich “offene” Fragen
  - Möglichkeit der Mitstrukturierung des Interviewers
- nicht standardisiert
  - völliger Verzicht auf einen Fragebogen
  - nur Stichwort- oder Themenvorgabe

Weitere Probleme beim Fragen

- Ziel: Soll-Zustand feststellen: Es sollte also genügen, die Kunden nach ihren Wünschen zu fragen.
- Aber: Die Kunden konzentrieren sich auf das, was ihnen derzeit nicht gefällt.
  - Wir sind also bei jeder Umstellung auf die Schwachpunkte des bestehenden Systems fixiert
  - Seine Stärken nehmen wir erst wahr, wenn sie uns fehlen.
  - Implizit erwarten die Kunden, dass alles, was bisher akzeptabel oder gut war, unverändert bleibt oder besser wird.
- Die Anforderungen an das neue System bestehen also nur zum kleinsten Teil aus Änderungswünschen, die allermeisten Anforderungen gehen in Richtung Kontinuität.



Szenarien und Use Cases

Definition (Pohl):

Ein *Szenario* beschreibt ein konkretes Beispiel für die Erfüllung bzw. Nichterfüllung eines oder mehrerer Ziele. Es konkretisiert dadurch eines oder mehrere Ziele. Ein Szenario enthält typischerweise eine Folge von Interaktionsschritten und setzt diese in Bezug zum Systemkontext.

- Hintergrund: Menschen mögen konkrete Beispiele eher als abstrakte Beschreibungen
  - => Wähle Beispiele um spätere Interaktion mit Software zu beschreiben
- Anwendungsfälle (Use Cases) gruppieren Szenarien

Beispiel „automatisches Bremsmanöver“

- Der Fahrer fährt mit konstanter Geschwindigkeit auf der Autobahn. Das vorausfahrende Fahrzeug bremst scharf. Der Fahrer tritt auf das Bremspedal. Das System erkennt eine Unterschreitung des Sicherheitsabstands und gibt eine Warnung aus. Der Abstand zwischen den Fahrzeugen verringert sich weiter. Das System leitet eine automatische Vollbremsung ein und informiert den Fahrer über den Bremsvorgang. Als der Abstand sich nicht mehr verringert, beendet das System die automatische Vollbremsung. Danach stellt es den Mindestabstand zum vorausfahrenden Fahrzeug wieder her und informiert den Fahrer über die Rückgabe der Kontrolle.
  (nach Pohl)

- Was wird klar, was bleibt unklar?

Szenarientypen

- **Positiv:** Wie wird das Ziel erfüllt?
- **Negativ:** Wie wird das Ziel nicht erfüllt?
- **Missbrauchsszenarien:** unerwünschte Verwendung

- **Deskriptiv:** verdeutlicht typische Abläufe
- **Explorativ:** verdeutlicht Lösungsraum
- **Erklärend:** verdeutlicht Gründe für Systemverhalten

- **Systemintern:** Handlungsstränge innerhalb des Systems
- **Interaktiv:** Handlungsstränge zwischen Akteuren
- **Kontextszenarien:** weitere Kontextinformationen

- **Hauptszenario:** wesentlicher Funktionsablauf
- **Alternativszenario:** partielle Abwandlung eines Hauptszenarios
- **Ausnahmeszenario:** Reaktion auf Ereignisse, die Hauptszenario verhindern

Szenario als strukturierte Folge von Schritten

1. Der Fahrer schaltet das Navigationssystem ein.
2. Das System ermittelt den aktuellen Standort des Fahrzeugs.
3. Das System erfragt den gewünschten Zielort.
4. Der Fahrer gibt den Zielort in das System ein.
5. Das System ermittelt den benötigten Fartenausschnitt.
6. Das System zeigt die Karte des Zielgebietes am Bildschirm an.
7. Das System erfragt die Optionen für die Routenberechnung.
8. Der Fahrer wählt die Optionen für die Routenberechnung.
9. Das System bestimmt die Wegführung.
10. Das System zeigt eine Erfolgsmeldung am Bildschirm.
11. Das System stellt eine Liste von Wegpunkten zusammen.
12. Das System zeigt den nächsten Wegpunkt der Navigation.

Darstellung in Tabellenform

Szenario „Navigation zum Zielort"	
Fahrer	**Navigationssystem**
1. Einschalten des Geräts	
	2. Ermittelt den aktuellen Standort
	3. Erfragt den Zielort
4. Eingabe des Zielorts	
	5. Ermittelt den benötigten Kartenausschnitt
	6. Zeigt am Display Karte vom Zielgebiet
	7. Erfragt die Optionen für die Routenberechnung
8. Eingabe der Optionen für die Routenberechnung	
	9. Ermittelt die Wegführung
	10. Anzeige „Weg wurde berechnet"
	11. Ermittelt die Liste der Wegpunkte
	12. Anzeige des jeweils nächsten Wegpunkts

Schwierigkeiten bei der Analyse

- Der Analytiker hat (bewusst oder unbewusst) eigene Interessen, die er durchsetzen möchte (eine „hidden agenda“).
- Der Kunde hat Anforderungen, die er nicht sagen will (also auch eine hidden agenda)
- Der Kunde hat Anforderungen, die ihm so selbstverständlich scheinen, dass er sie nicht erwähnt.
- Wünsche sind technisch nicht realisierbar oder zu teuer: →
  - Anforderungen müssen reduziert oder Mittel aufgestockt werden.
- Konflikte zwischen Wünschen der Klienten: →
  - Probleme vor Klienten ansprechen und auf Einigung drängen.

Weitere Schwierigkeiten der Analyse (cont)

- Warum wird die Analyse (und besonders die Ist-Analyse) in der Praxis oft vernachlässigt oder falsch fokussiert?
  - Wo die Spezifikation keine Rolle spielt, hat auch die Analyse keine Bedeutung
  - Der Kunde will keine Veränderung, sondern eine Verbesserung
    - Ist-Analyse nicht vernachlässigen!
  - Entwickler sind oft der (grundfalschen) Überzeugung, bereits zu wissen, was gewünscht oder benötigt wird.
  - Kunden legen der Analyse Steine in den Weg:
    - Den Analytikern stehen die Kunden (insbesondere die Fachleute auf Kundenseite) nicht zur Verfügung.
    - Die Analyse-Aktivität wird als unproduktiv denunziert.
    - Kunden sabotieren den Prozess durch späte Änderungen.

Regeln für die Analyse

- Die Ausgangslage der Analyse ist extrem unterschiedlich:
  - Anwendungsgebiet der Software
  - Umgebung des Zielsystems
  - Aufgabenstellung
  - Vorkenntnisse des Analytikers
  - Verständnis der Gesprächspartner usw.
- Folge: Es gibt kaum allgemeine Regeln für die Analyse.
- Immer wichtig und richtig:
  - Die Analyse als Tätigkeit wahr- und ernst nehmen!
  - Aufwand und Personal dafür im notwendigen Maße einplanen!
  - Resultate in eine (auch dem Kunden) verständliche Form bringen!

Bremen

CG VR

Wie dokumentiert man Anforderungen?
Requirements Specification

Definitionen: Lasten- / Pflichtenheft

Lastenheft (DIN 69901 – 5):

- Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferung und Leistungen eines Auftragnehmers innerhalb eines Auftrags.

Pflichtenheft (DIN 69901 – 5):

- Das vom Auftragnehmer erarbeitete Realisierungsvorhaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenhefts

Anforderungsspezifikation (Pohl, Rupp, 2011)

- Eine systematisch dargestellte Sammlung von Anforderungen (typischerweise für ein System oder eine Komponente), die vorgegebenen Kriterien genügt.

Lastenheft / Pflichtenheft verständlich

- Lastenheft wird vom Auftraggeber (Kunden) geschrieben
  - welche Funktionalität ist gewünscht
  - welche Randbedingungen (SW/ HW) gibt es
- Pflichtenheft wird vom Auftragnehmer (Software-Entwicklung) geschrieben
  - welche Funktionalität wird realisiert
  - auf welcher Hardware läuft das System
  - welche SW-Schnittstellen (Versionen) berücksichtigt
- Variante: Kunde beauftragt Auftragnehmer direkt in Zusammenarbeit Pflichtenheft zu erstellen
  - ein gemeinsames Heft ist sinnvoll
  - Pflichtenheft ist meist (branchenabhängig) zu bezahlen

Lastenheft

- Beschreibt die fachlichen Anforderungen aus Kundensicht.
- Obwohl es Definitionen für diesen Begriff gibt (z.B. in DIN 69905), bleibt er in der Praxis unscharf, weil der Inhalt des Lastenhefts von Firma zu Firma stark variiert.
- Lücken, Unklarheiten und Widersprüche sind darin normal.
- Erst das Pflichtenheft sollte vollständig, klar und konsistent sein.

Angestrebte Eigenschaften des Pflichtenhefts

- Das Pflichtenheft ist im Idealfall inhaltlich

1. zutreffend: Es gibt die Vorstellungen des Kunden richtig wieder.
2. vollständig: Jede (in einem Kopf oder in einem Dokument) vorhandene Anforderung ist in der Spezifikation enthalten.
3. widerspruchsfrei (oder konsistent): Jede Anforderung ist mit allen anderen Anforderungen vereinbar.
   Ein inkonsistentes Pflichtenheft ist nicht realisierbar, denn kein System kann widersprüchliche Anforderungen erfüllen.
4. neutral (oder abstrakt): Das Pflichtenheft schränkt die Realisierung nicht über die wirklichen Anforderungen hinaus ein.
5. nachvollziehbar: Die Quellen der Anforderungen sind dokumentiert, die Anforderungen sind eindeutig identifizierbar.
6. objektivierbar: Das realisierte System kann gegen die Anforderungen geprüft werden. Auch (missverständlich) „testbar“.

Darstellung und Form

- Eine ideales Pflichtenheft ist

1. leicht verständlich: Alle Interessenten sind in der Lage, das Pflichtenheft zu verstehen.
2. präzise: Das Pflichtenheft schafft keine Unklarheiten und Interpretationsspielräume.
3. leicht erstellbar: Die Anfertigung und Nachführung des Pflichtenhefts sind einfach und erfordern keinen nennenswerten Aufwand.
4. leicht verwaltbar: Die Speicherung des Pflichtenhefts und der Zugriff darauf sind einfach und erfordern keinen nennenswerten Aufwand.

- Diese Merkmale konkurrieren!
- Auch hier suchen wir also einen Kompromiss.

Pflichtenheft vs Entwurf

- Die Forderung nach Neutralität war in der Frühzeit des Software Engineerings radikal:
  - Idee: Das Pflichtenheft soll aussagen, was das System leistet, aber nicht, wie diese Leistung erbracht wird („What, not how!“).
    - Beispiel: Programm zur Ermittlung der Nullstellen einer Funktion kann ohne Aussagen zum Algorithmus spezifiziert werden.
- Inzwischen wurde die Forderung nach Neutralität aufgeweicht.
- Als Ideal bleibt sie aber gültig!
  Denn in der Praxis kann man immer wieder beobachten, dass ein Pflichtenheft gefordert, aber ein Entwurf geliefert wurde. Auf diese Weise wird (evtl.) die optimale Lösung ausgeschlossen.

Pflichtenheft vs Entwurf (cont)

- Die geforderte Neutralität der Spezifikation ist als Ideal sinnvoll, aber praktisch nicht durchzuhalten. Gründe:

1. Vorhandene Komponenten sind einzubeziehen.
2. Vorgaben von außen betreffen die Struktur und Details
3. Die Beschreibung des Lösung ist einfacher.
4. Erst das gegliederte System ist überschaubar.
5. Die Konsistenz der Spezifikation wird erst durch die Realisierung geklärt.
6. Die Fragen an die Spezifikation entstehen erst im Entwurf.

- Darum kann eine abstrakte Spezifikation nur unter Idealbedingungen entstehen: Das Problem ist gut verstanden, sicher lösbar, stabil und relativ „flach“.

Lastenheft / Pflichtenheft: möglicher Aufbau

0. Administrative Daten: von wem, wann genehmigt, ...
1. Zielbestimmung und Zielgruppen
   In welcher Umgebung soll System eingesetzt werden?
   Ziele des Systems, welche Kunden betroffen?
2. Funktionale Anforderungen
   Produktfunktionen (Szenarien, Use Cases, Anforderungen)
   Produktschnittstellen (GUI-Konzept , andere SW)
3. Nichtfunktionale Anforderungen
   Qualitätsanforderungen
   weitere technische Anforderungen
4. Lieferumfang
5. Abnahmekriterien
6. Anhänge (insbesondere Begriffslexikon)

Darstellung: Formal, graphisch, natürlichsprachlich

- Eine Pflichtenheft soll einerseits präzise und einer Bearbeitung mit Werkzeugen zugänglich, andererseits auch für Laien handhabbar, mindestens aber verständlich sein.
- Da die Informatik starke Wurzeln in der Mathematik hat, war es naheliegend, für die Spezifikation formale Notationen zu entwickeln.
- Mit diesen Notationen verwandt sind grafische Darstellungen der Spezifikation.
  - Dabei steht aber weniger die formale Präzision als die Anschaulichkeit im Vordergrund.
- Wer weder formal noch grafisch spezifiziert, bedient sich der natürlichen Sprache.
  - Diese hat den Vorteil, für jeden Menschen verständlich zu sein und ganz ohne spezielle Regeln und Werkzeuge auszukommen..

Die sieben Regeln zur Formulierung von Anforderungen

1. Formulieren Sie die Ziele kurz und prägnant!
2. Verwenden Sie Aktivformulierungen!
3. Formulieren Sie überprüfbare Ziele!
4. Verfeinern Sie nicht überprüfbare Ziele!
5. Stellen Sie den Mehrwert eines Ziels dar!
6. Geben Sie eine Begründung für das Ziel an!
7. Formulieren Sie deklarativ, d.h. vermeiden Sie die Beschreibung von Lösungsansätzen!

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

Prägnante Anforderungen

- Schlecht: Das geplante System soll sowohl von Experten als auch von unerfahrenen Personen (Nutzerinnen und Nutzern) benutzbar sein. Unerfahrene User sollen auch ohne große Vorkenntnisse der Bedienerführung oder des Vorgängersystems die vorgesehene Systemfunktionalität nutzen können. Zur Benutzung des Systems sollen die Anwender also keinerlei Schulungen oder spezielle Hilfestellungen benötigen. Der Umgang mit dem System muss daher leicht verständlich sein und ohne große Erfahrung mit dem Vorgängersystem oder vergleichbaren Systemen erfolgen können.
  - Besser: Ein unerfahrener Benutzer soll das System ohne spezielle Schulung verwenden können

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

Aktive Formulierungen

- Schlecht: Die Dauer für die Erfassung und Verarbeitung der Messdaten soll halbiert werden.
  Dadurch soll die Wartezeit bis zum Vorliegen von Ergebnissen verkürzt werden.

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

- Besser: Das System erfasst und verarbeitet die Messdaten im Vergleich zum System xy doppelt so schnell. Dadurch muss der Nutzer kürzer auf das Vorliegen von Ergebnissen warten

Überprüfbare, quantitative Anforderungen

- Schlecht: Das System soll besser sein als das Vorgängersystem.

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

- Besser: Das System soll folgende Verbesserungen gegenüber dem System xy bieten:
  - ...
  - ...

Verfeinerung von Anforderungen

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

- Schlecht: Die Benutzung des Systems soll selbsterklärend sein

- Besser: Das System soll selbsterklärend sein, d.h. ein durchschnittlicher Nutzer soll durchschnittlich nach 2 Min. folgende Funktionen aufrufen können: ...
- Das System soll selbsterklärend sein, d.h. den Vorgaben nach W3C... in Bezug auf ... folgen

Mehrwertbildung

- Schlecht: Das System soll leicht benutzbar sein.

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

  - Besser: Das System soll … so dass sich die Nutzer auf andere Aufgaben konzentrieren können

Nachvollziehbare Begründungen

- Schlecht: Das System soll auch von ungeschulten Benutzern intuitiv benutzbar sein.

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

- Besser: ... weil es auch in Mietfahrzeugen einsetzbar sein soll

Keine Lösungsvorwegnahme

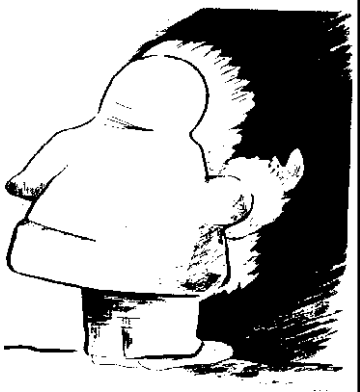
- Schlecht: Durch komprimierte Datenübertragung im Cache soll das geplante System um 10% kürzere Antwortzeiten aufweisen

1. Prägnant
2. Aktiv
3. Überprüfbar
4. Verfeinerbar
5. Wertschöpfend
6. Begründbar
7. Deklarativ

- Besser: Das System soll um 10% kürzere Antwortzeiten aufweisen als System xy.

Das Begriffslexikon

- Hilfreich ist es, schon während der Analyse ein Begriffslexikon anzulegen; dieses wird während der gesamten Software-Entwicklung verwendet und ergänzt.
- In den frühen Phasen wird dieses Dokument aufgebaut und weiterentwickelt.



- Das Begriffslexikon enthält solche Begriffe, die
  - wichtig sind und
  - von verschiedenen Leuten, v.a. von Klienten, Analytikern und Entwicklern, unterschiedlich ausgelegt werden könnten.
- Dazu gehören häufig Begriffe, die auf den ersten Blick völlig klar zu sein scheinen.
- Beispiel: Informationssystem für die Prüfungsdaten von Studenten
  - „Student“, „Prüfung“, „Note“, „Erstprüfung“

Informationen im Begriffslexikon

a) Begriff und Synonyma (im Sinne der Spezifikation)
b) Bedeutung (Definition, Erklärung)
c) Abgrenzung (wo ist dieser Begriff nicht anzuwenden?)
d) Gültigkeit (zeitlich, räumlich, sonst)
e) Fragen der Bezeichnung, Eindeutigkeit u. A.
f) Unklarheiten, die noch nicht beseitigt werden konnten
g) verwandte Begriffe (Querverweise)

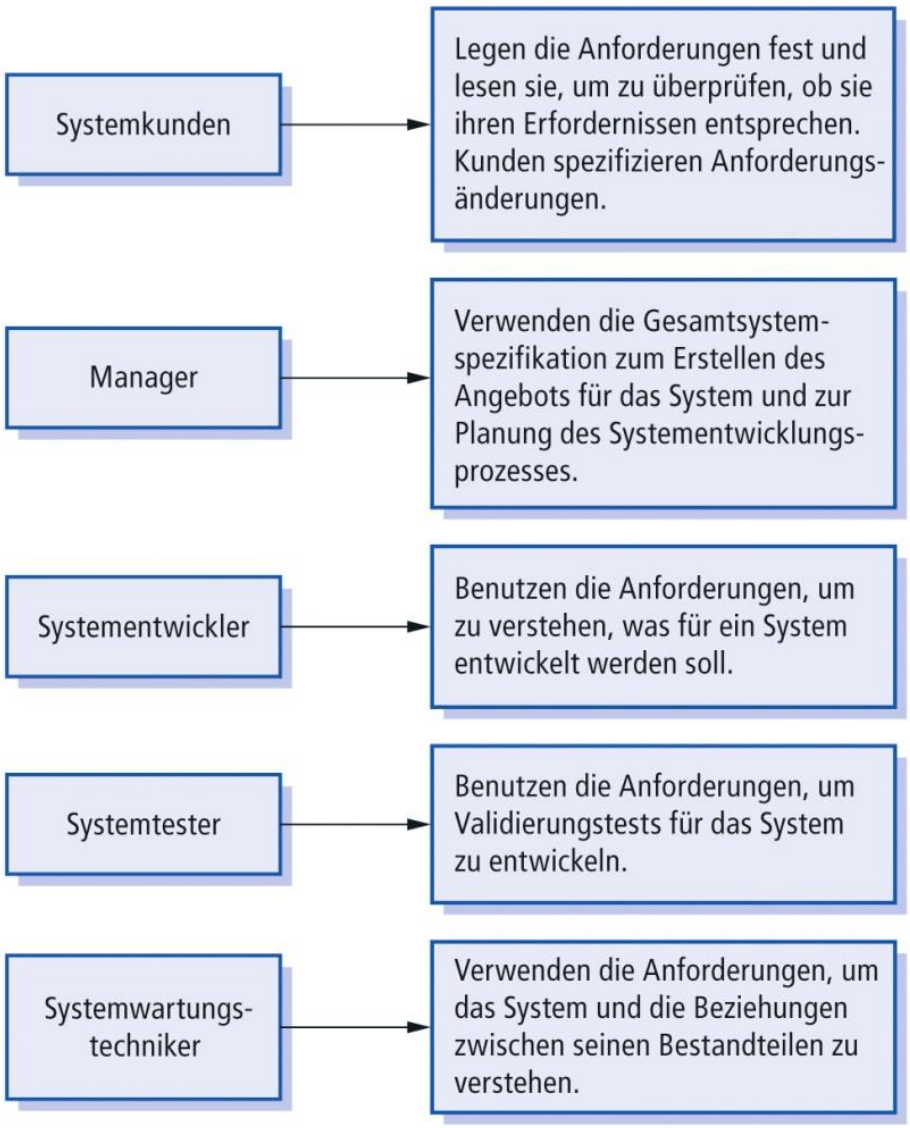
- Die Angaben werden aus den Gesprächen und Interviews abgeleitet oder, wenn sie von den Analytikern kommen, sorgfältig mit den Klienten überprüft.
- Typische Quellen sind im genannten Beispiel die Angestellten in der Verwaltung, die Juristen der Uni, die Gesetze und Ordnungen der Universität.

Beispiel

Begriff	**Student**, synonym Studentin, Studierender, Studierende
Bedeutung	Eine Person, die an der Universität Stuttgart immatrikuliert ist und noch nicht exmatrikuliert wurde, die folglich legal einen Studentenausweis der Universität Stuttgart hat oder haben könnte.
Abgrenzung	Gasthörer und Studierende anderer Hochschulen sind im Sinne dieses Systems keine Studenten.
Gültigkeit	Mit der Immatrikulation an der Universität Stuttgart entsteht ein neuer Student; er existiert bis zur Exmatrikulation, gleichgültig, wie sie zustande kommt. Ein Fachwechsel oder eine Namensänderung implizieren keine Exmatrikulation. Hat sich eine Person im Laufe ihres Lebens mehrfach an der Universität Stutt-gart immatrikuliert, so handelt es sich um mehrere, nicht identische Studenten.
Bezeichnung	Ein Student ist durch die Matrikelnummer und einen Zeitpunkt (zu dem die Matrikelnummer gültig war oder ist) eindeutig bestimmt, alle anderen Attribute, insbesondere der Name, können mehrfach vorkommen.
Unklarheiten	Es ist noch ungeklärt, wie Namen aus anderen Schriftsystemen (z. B. Russisch, Arabisch, Chinesisch) dargestellt werden.
Querverweise	Gasthörer, Matrikelnummer, Studentenausweis

Warum machen wir das Ganze?

Wer liest überhaupt das Pflichtenheft?



Der Nutzen des Requirements Engineering

- In der Praxis gibt es viele schlechte Anforderungsanalysen; oft gibt es gar keine
- Die Anforderungsanalyse ist aber notwendig für

1. die Abstimmung mit dem Kunden bzw. mit dem Marketing,
2. den Entwurf und die Implementierung,
3. das Benutzungshandbuch,
4. die Testvorbereitung,
5. die Abnahme,
6. die Wiederverwendung,
7. die Klärung späterer Einwände, Regressansprüche usw.,
8. eine spätere Re-Implementierung.

Nachteile bei fehlender Anforderungsanalyse

1. Die Anforderungen bleiben ungeklärt, sie werden darum auch nicht erfüllt.
2. Den Entwicklern fehlt die Vorgabe, darum fragen sie „auf dem kurzen Dienstweg“ Bekannte, die beim Kunden arbeiten, oder sie legen mangels Kontakten die eigenen Erfahrungen und Erwartungen zu Grunde.
3. Die Basis für das Handbuch fehlt, es wird darum phänomenologisch, d.h. experimentell, verfasst.
4. Ein gutes Handbuch ist ein umformulierter Auszug aus der Anforderungsanalyse!
5. Ein systematischer Test ist ohne Anforderungsanalyse unmöglich, denn es ist nicht definiert, welche Daten das System akzeptieren muss und welche Resultate es liefern soll.

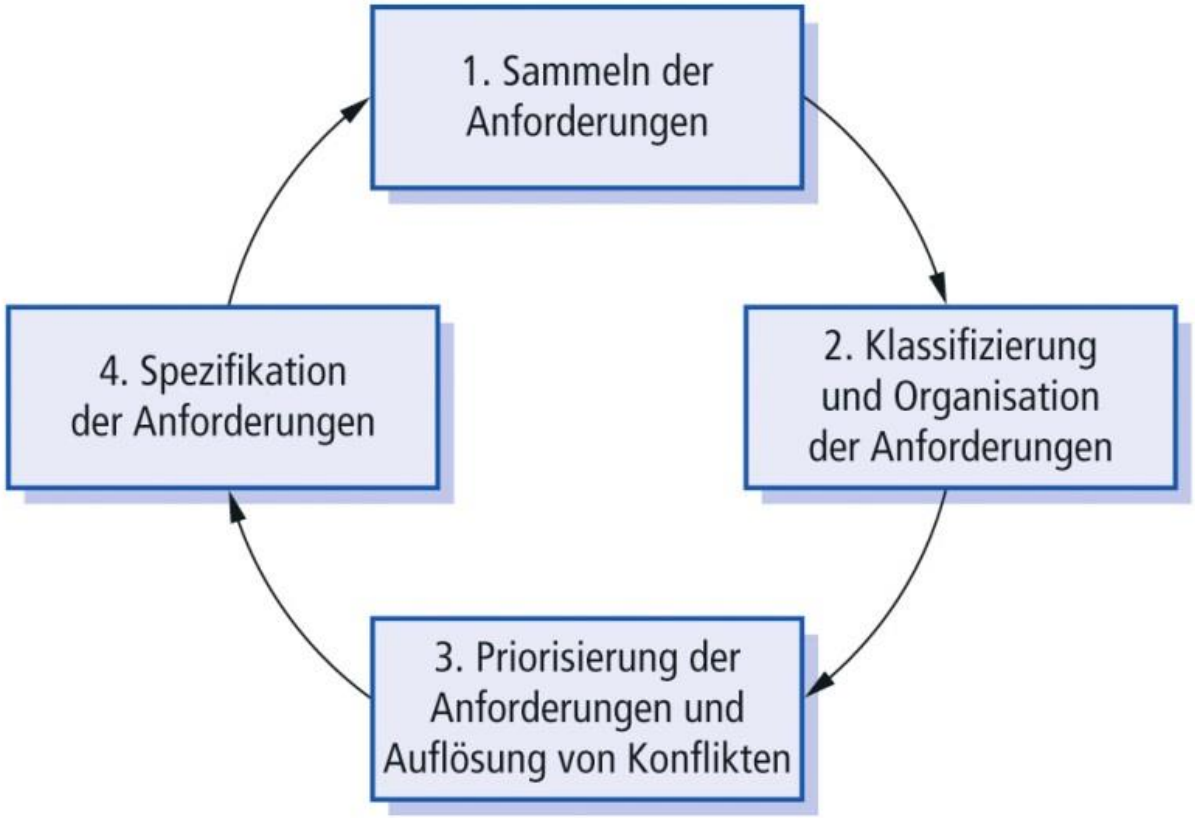
Nachteile bei fehlender Anforderungsanalyse (cont)

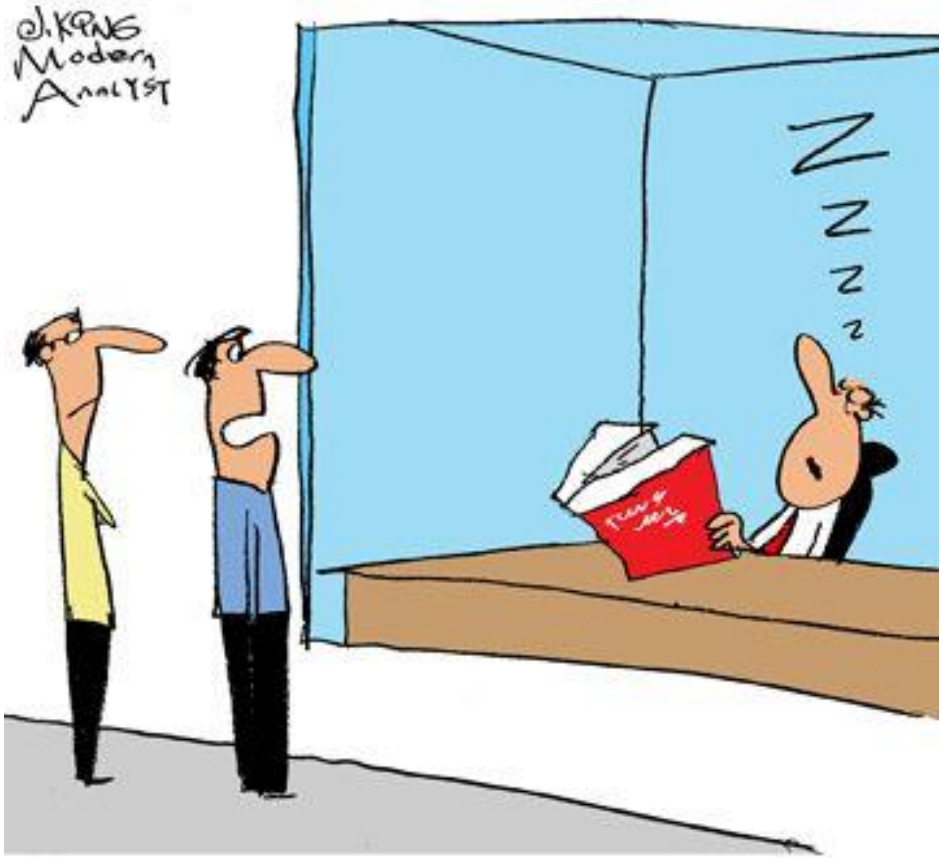
6. Wenn bei der Abnahme nicht entschieden werden kann, ob das System richtig arbeitet, wird die Korrektheit zur Glaubensfrage.
7. Oft zeigen sich echte oder vermeintliche Mängel der Software erst nach längerem Gebrauch. Ohne Anforderungen kann diese Unterscheidung aber nicht getroffen werden.
8. Wer eine Software(-Komponente) wiederverwenden will, muss wissen, was sie leistet. Das ist in der Spezifikation dokumentiert.
9. Wenn ein System ausgemustert und ersetzt wird, ist Aufwärtskompatibilität gefordert (vgl. Heninger et al., 1978).

- „Spezifikation im Kopf“ gibt es nicht!

Requirements-Engineering

- Requirements-Engineering ist kein linearer Prozess, sondern ein iterativer





"Based on our tests, the business stakeholders fall asleep around page 37 of the Functional Requirements Specification. Put the Issues Section on page 40."