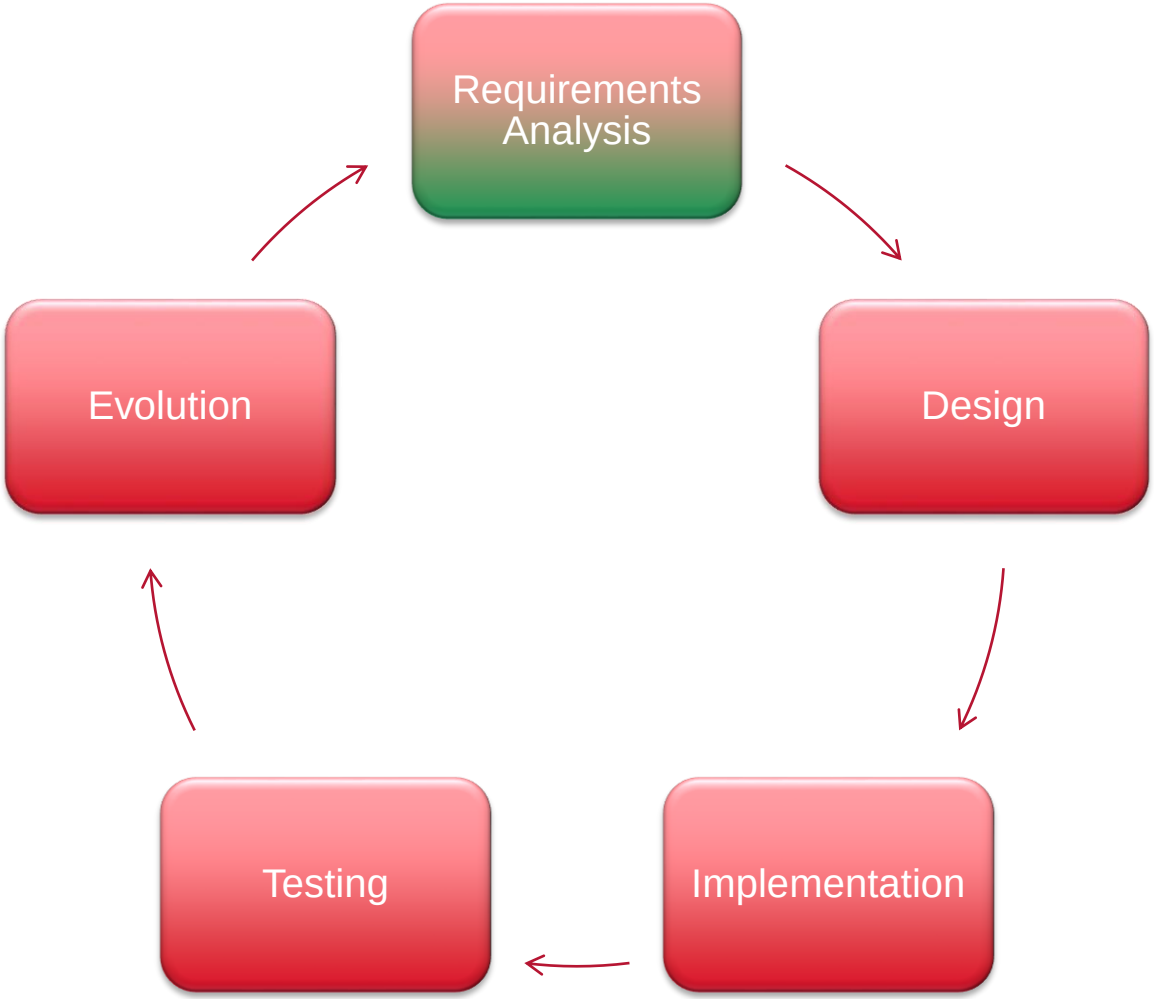
Media Engineering
Requirements Engineering

R. Weller

University of Bremen, Germany

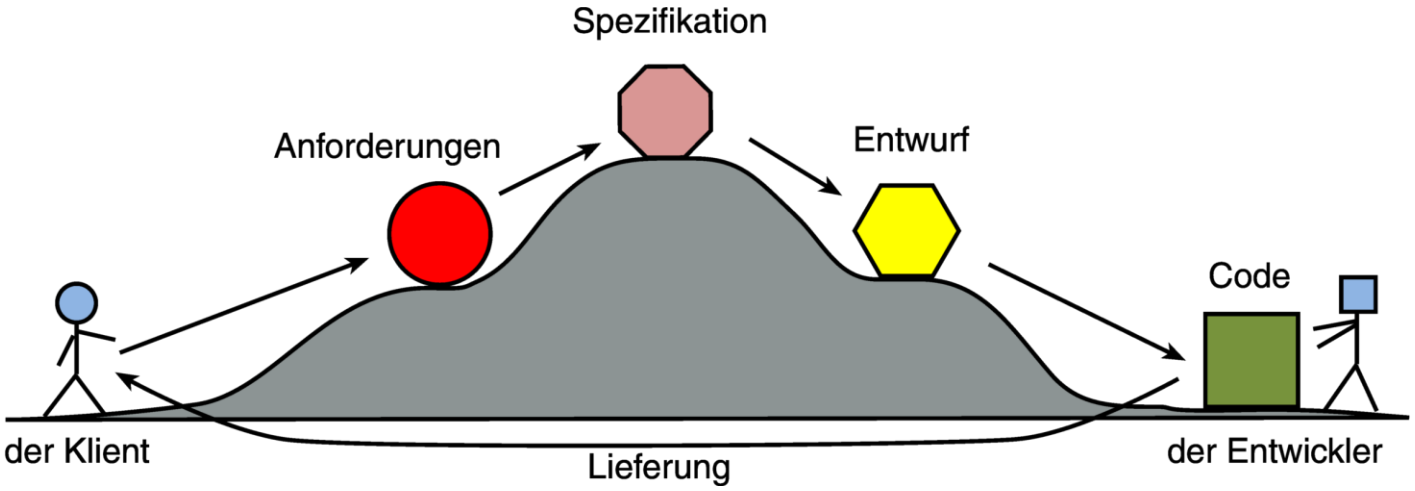
cgvr.cs.uni-bremen.de

Der Software Development-Lifecycle



Requirements Engineering im Überblick

- Die Anforderungen werden
  - Erhoben
  - in der Spezifikation formuliert, geprüft
  - und anschließend in den Entwurf umgesetzt.
  - Schließlich wird implementiert
  - auf verschiedenen Ebenen geprüft und korrigiert
  - Das Resultat geht zurück an den Kunden.



Übersicht

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

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
- Richtig betriebenes Requirements Engineering ist wirtschaftlich
- Analysten berichten, dass 71% aller IT-Projekte wegen einer schlechten Anforderungsanalyse scheitern

Kunde und Anforderungen

- Werden die Erwartungen, die Anforderungen des Kunden, nicht vollständig und präzise erfasst, ist damit zu rechnen, dass das entwickelte Produkt die Anforderungen nicht (vollständig) erfüllt.
- Die vollständige und präzise Erfassung der Anforderungen ist die allerwichtigste technische Voraussetzung für eine erfolgreiche Software-Entwicklung.

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 Produkts Betroffene
    - z.B. Verkäufer, Support

Was sind überhaupt Anforderungen?
Requirements

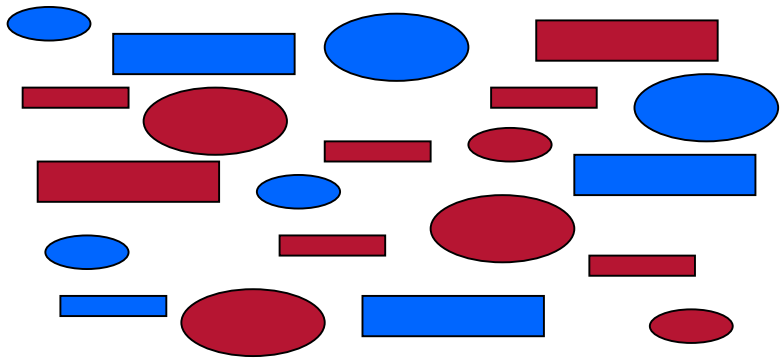
Definitionen (Nach IEEE Std 610.12-1990)

Requirement

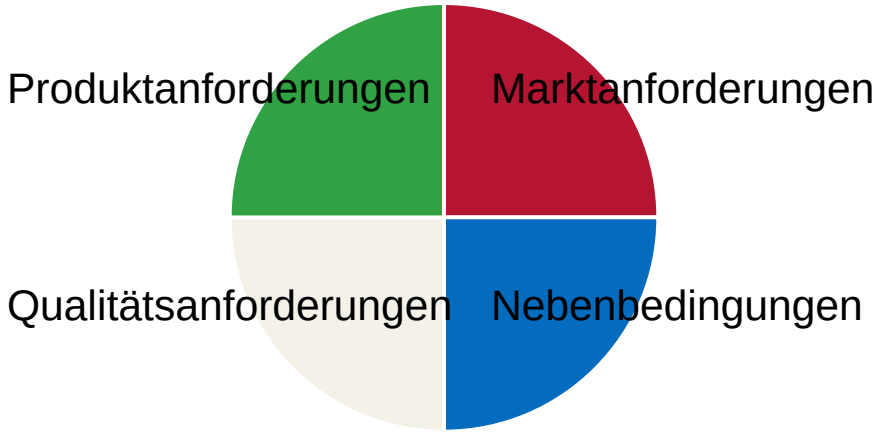
- (1) A condition or capability needed by a user to solve a problem or achieve an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- (3) A documented representation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

Welche Arten von Anforderungen gibt es?

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



Betriebswirtschaftliche Einteilung



- Produktanforderungen
  - Beschreiben, was verschiedene Benutzer mit dem Produkt machen können
  - Betrachten Softwareprodukt oder Dienst innerhalb der Umgebung, in der das System einmal arbeiten muss
- Marktanforderungen
  - Definieren, warum ein Projekt durchgeführt werden soll
  - Enthalten oft Priorisierung aus betriebswirtschaftlicher Sicht

Betriebswirtschaftliche Einteilung (cont)

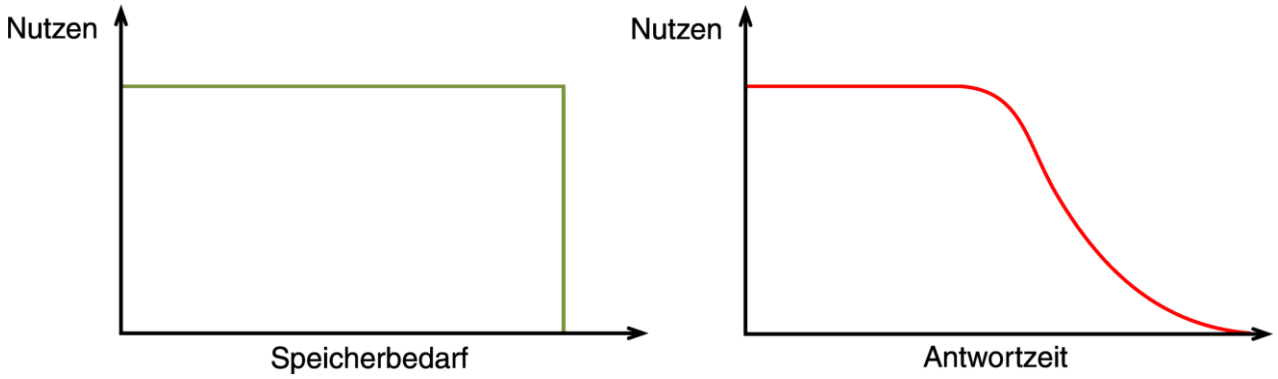
- Qualitätsanforderungen
  - Beschreiben eine qualitative Eigenschaft, die das System oder eine Funktion aufweisen müssen
  - Sie ergänzen die Produktanforderungen
    - Beispiele: Zuverlässigkeit, Verfügbarkeit, Wartbarkeit
  - Sind oft schwer zu spezifizieren und zu testen
- Randbedingungen
  - Eine organisatorische oder technische Anforderung, die die Art und Weise einschränkt, wie ein System realisiert werden kann
  - Sie ergänzen Produktanforderungen und Qualitätsanforderungen
    - Beispiele: Geschäftsprozesse, Gesetze
  - Organisatorische Randbedingungen spielen auch eine Rolle, obwohl sie nicht als Anforderungen zugegeben werden

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!

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



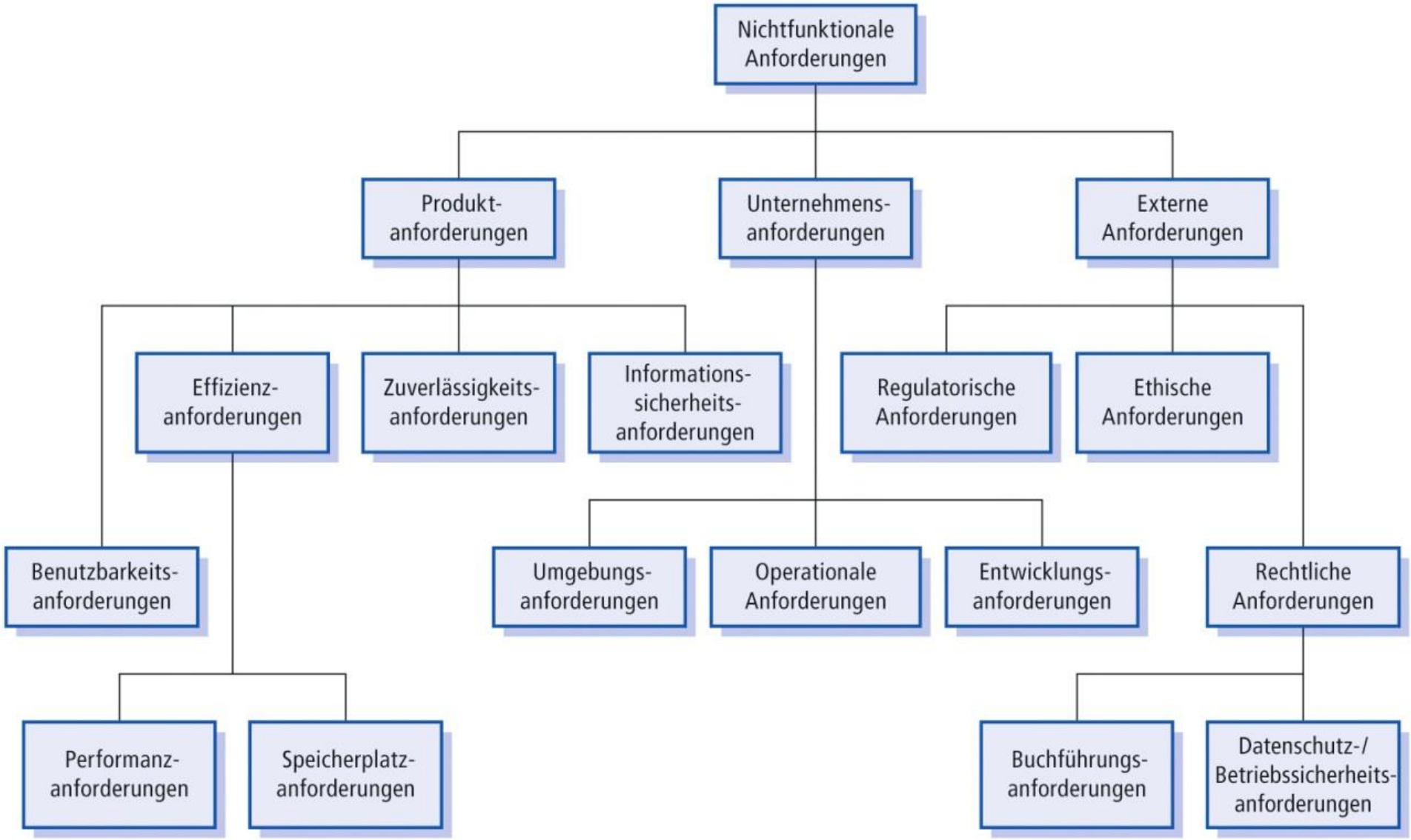
Weitere Kategorien von Anforderungen Funktional vs. Nichtfunktional

- 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
  - Aber auch Marktanforderungen, und Nebenbedingungen sind NFRs

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 spezifizieren.
  - 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!

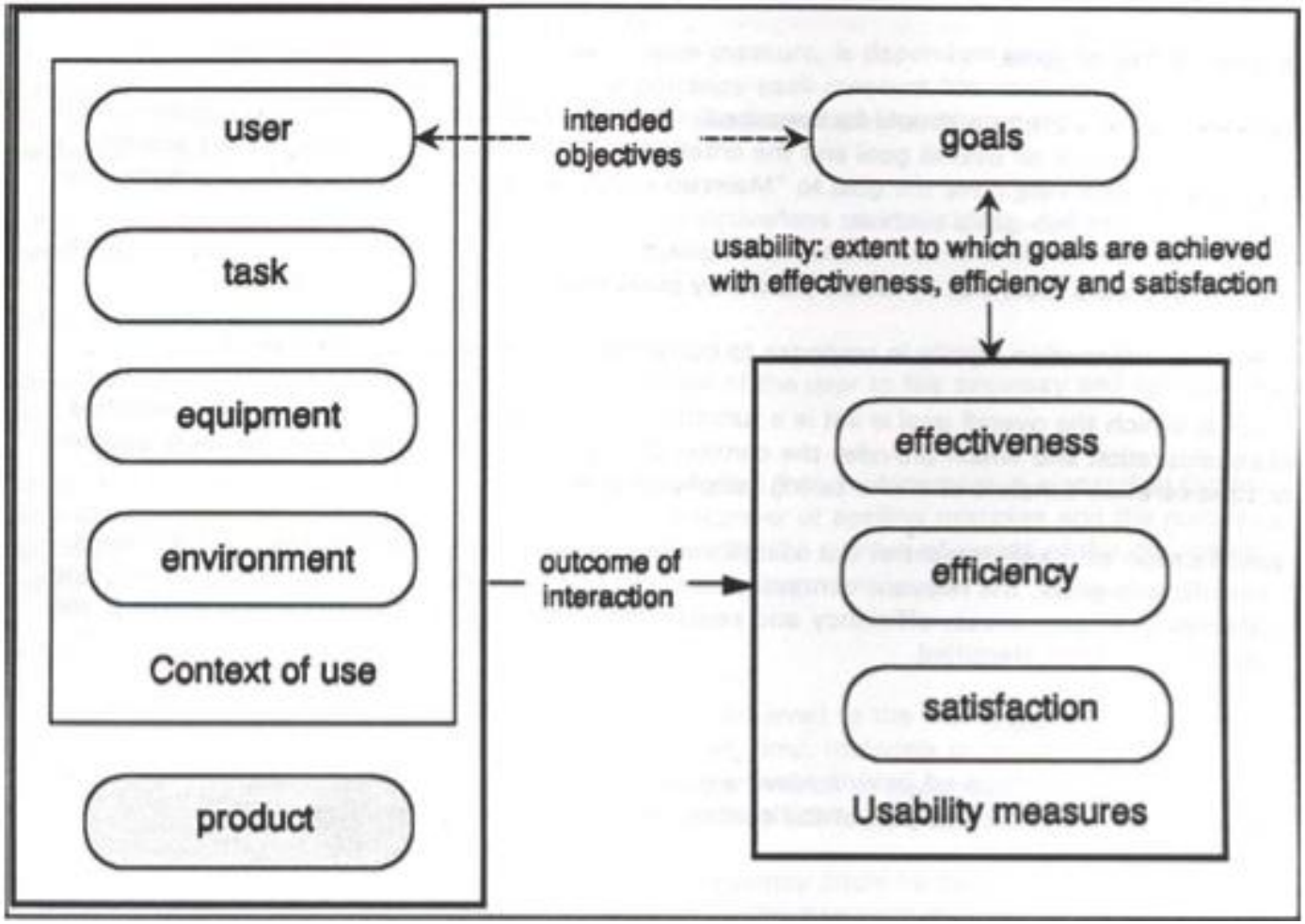
ISO 9241: Ergonomics of Human System Interaction*

- 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: Vorschriften und Regeln



Usability Framework (ISO 9241)

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.

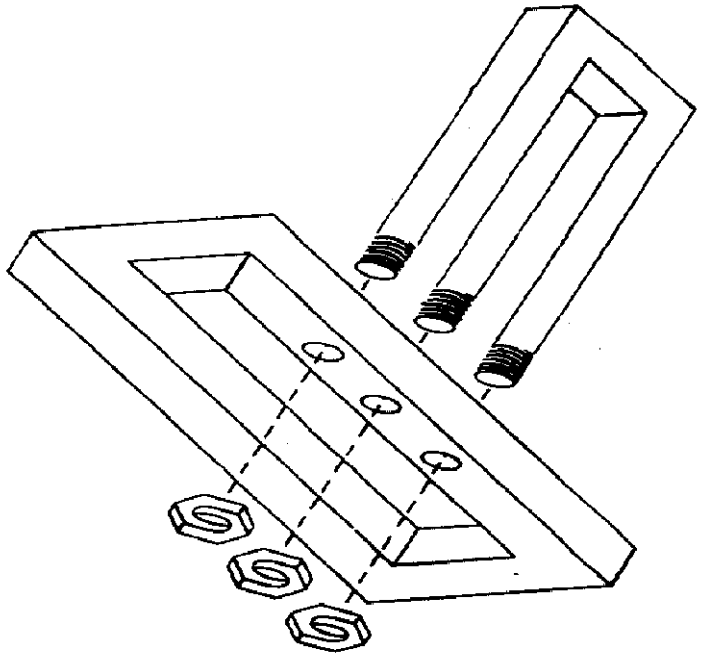
Richtlinien für Barrierefreiheit

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

Gesetzlich vorgeschrieben für öffentliche Einrichtungen (BIT-V)

Qualitätskriterien für eine Anforderung

- Anforderungen sollen folgende Kriterien erfüllen:
  - Vollständig
  - Verständlich
  - Korrekt
  - Widerspruchsfrei
  - Eindeutig
  - Prüfbar



Probleme mit Anforderungen

- Anforderungen können mehrdeutig, überspezifisch, unvollständig, kontextspezifisch, unmöglich oder falsch sein, auf jeden Fall gibt es immer zu viele.
- Bei Anforderungen stellen sich viele Fragen:
  - Ist die Anforderung wichtig?
  - Sind alle Inhalte gleichermaßen wichtig und wie korrelieren sie?
  - Wer ist für die Anforderung verantwortlich?
  - Ist die Anforderung machbar? Welche Einflüsse hat sie?
  - Wurde diese Anforderung geändert und durch wen?
  - Wo sind die Analyseergebnisse dokumentiert?
  - Wie wird die Anforderung validiert?
  - Wie konkretisiert sich die Anforderung?

Wie findet man Anforderungen?
Requirements Analysis

Definitionen (Nach IEEE Std 610.12-1990)

Requirements Analysis

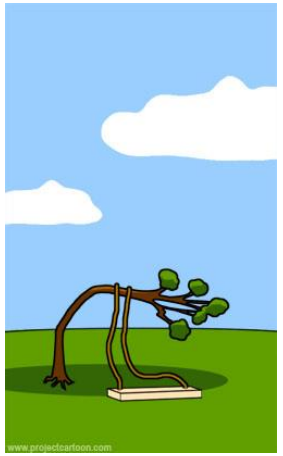
- (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
- (2) The process of studying and refining system, hardware, or software requirements.

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
  - Man phantasiert Beispiel-Szenarien
  - ...

Die Ist-Analyse

- Ziel der Analyse: 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.



Die Ist-Analyse

- Darum hat die Ist-Analyse, also die Feststellung des bestehenden Zustands, große Bedeutung für die erfolgreiche Projektdurchführung. Bei Widersprüchen ist der Ist-Zustand zudem ein sicherer Halt!
- Der Analytiker muss sich also gut einfühlen und genau beobachten, um vom Kunden zu erfahren, welche Aspekte des Ist-Zustands für ihn wichtig sind und beibehalten werden sollten.
- Die Ist-Analyse ist eine undankbare Tätigkeit!

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“? ...

Die Soll-Analyse

- Gefahr: Der Analytiker wird zum Prellbock zwischen
  - den Kunden (mit großen Wünschen) und
  - den Entwicklern (mit der Sorge, die Wünsche nicht erfüllen zu können, und dem Wunsch, eigene Vorstellungen zu realisieren).
- Aber der Analytiker sollte sich nicht als deus ex machina fühlen, der die unlösbaren Probleme löst.
- Sinnvoll:
  - Alle Wünsche sammeln, nicht kritisieren („Wunschzettel“)
  - Widersprüche offenlegen
- 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.

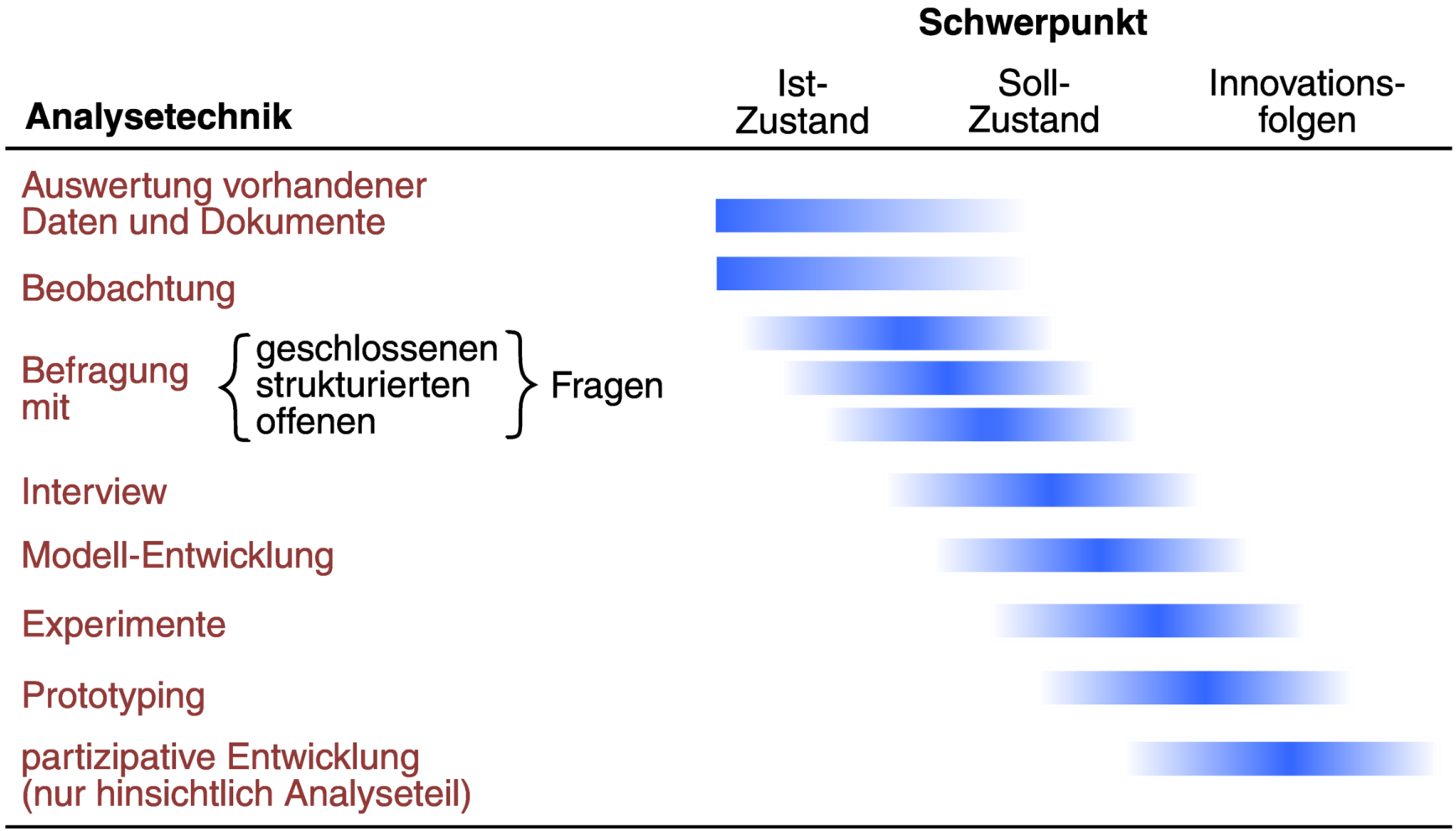
Die Spielräume der Soll-Analyse

- In aller Regel haben die Kunden kein absolut festes Bild des Produkts. Das verschafft dem Analytiker Spielräume.
  - Die zentrale Komponente des Systems hat Vorrang
    - andere Teile werden nicht sofort benötigt. → Entwicklung in Ausbaustufen.
    - Vorteile: Zentrale Funktion steht schnell zur Verfügung, und die weitere Entwicklung kann auch erste Erfahrungen und neue Fakten berücksichtigen.
  - Wenn käufliche (oder vorhandene) Komponenten integriert werden, sinken die Kosten und die Entwicklungszeit. Der Kunde muss den Preis „handgeschmiedeter“ Lösungen kennen.

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!

Techniken der Analyse



Techniken der Analyse

- Natürlich sollte der Analytiker sollte die einschlägigen Vorschriften und Regeln kennen und beachten.
  - Auswertung der Dokumente
- Wenn möglich, direkt am Alltag derer teilnehmen, die später mit der Software umgehen werden oder deren Funktion später von der Software übernommen wird (Ethnographische Analyse).
  - Über die Schulter schauen oder, besser noch, selbst mitarbeiten.
- Dann kann der Analytiker die Klienten systematisch befragen
  - geschlossene und strukturierte, auch offene Fragen
  - Gespräch bringt mehr als Fragebogen-Versand!

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
- 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

Techniken der Analyse (cont)

- Wenn die abstrakte Vorstellung vom Zielsystem nicht ausreicht, muss etwas ausprobiert oder gezeigt werden.
  - Modell-Entwicklung
  - Experiment
  - ausführbarer Prototyp
- Bei der partizipativen Entwicklung wird die Trennung zwischen Analyse und Entwicklung bewusst aufgehoben. Die neue Software wächst in den sozialen Kontext hinein.
  - Vergleiche auch die Ansätze der „agilen Software-Entwicklung“! (dazu später mehr)

Schwierigkeiten der Analyse

- Der Kunde kann sehr viele Anforderungen nicht nennen, weil er sie nicht hat. Er hat nur Wünsche und Ziele, die sich im Gespräch mit dem Analytiker auf Anforderungen abbilden lassen.
- 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.

- Am Ende der Analyse und Spezifikation sollte eine Vision entstanden sein, in der
  - der Kunde die Erfüllung seiner (reduzierten) Wünsche,
  - der Entwickler ein realisierbares Produkt sieht.
- Beschrieben durch ein Dokument evtl. auch durch einen Prototypen.

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, wird auch die Analyse keine Bedeutung haben.
  - Der Kunde will keine Veränderung, sondern eine Verbesserung. Entwickler, die das nicht begreifen, vernachlässigen die Ist-Analyse.
  - 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.

Aus einem Praxisbericht

- Analyse wird vom Kernteam durchgeführt (ca. 3-4 Leute). Die Mitglieder kommen teilweise aus der Anwendung, teilweise handelt es sich um Entwickler.
- An der Analyse müssen Entscheidungsträger, Fachleute und Anwender beteiligt sein. Die Anwender werden typisch durch Interviews beteiligt. (Zwei Leute befragen eine Person ca. 90 min lang).
- Abstimmung der Analyse in „Workshops“ mit 6 bis 10 Teilnehmern, ein halber bis ganzer Tag.
- Es ist aussichtslos, den Kunden mit formalen Darstellungen zu kommen.

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. Dafür gibt folgende 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 (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.

Einige Sprachregeln

	Regel	Erläuterung, Beispiel
R 1	Formulieren Sie jede Anforderung im Aktiv.	Der Akteur wird angegeben, und es wird sichtbar, ob das System oder der Benutzer etwas tut. Fordert man, dass etwas »*gelöscht wird*«, so bleibt das unklar.
R 2	Drücken Sie Prozesse durch Vollverben aus.	Vollverben (wie »liest«, »erzeugt«, nicht »ist«, »hat«) verlangen weitere Informationen (Objekte, Ergänzungen), die den Prozess genauer beschreiben. Nicht: »*Wenn die Daten konsistent sind*«, sondern »*Wenn Programm ABC die Konsistenz der Daten geprüft hat*«. Und natürlich muss auch spezifiziert sein, was geschehen soll, wenn sie *nicht* konsistent sind (R4).
R 3	Entdecken Sie unvollständig spezifizierte Prozesswörter (Verben).	Fehlen Angaben (Objekte), dann wird nach diesen Angaben gesucht, um vollständige Aussagen zu erhalten. Wenn eine Komponente *einen Fehler meldet*, fragt sich, wem.
R 4	Ermitteln Sie unvollständig spezifizierte Bedingungen.	Für Bedingungen der Form »Wenn-dann-sonst« müssen sowohl der Dann- als auch der Sonst-Fall beschrieben sein (vgl. das Beispiel für R2).
R 5	Bestimmen Sie die Universalquantoren.	Sind Sätze mit »nie«, »immer«, »jedes«, »kein«, »alle« wirklich universell gültig, oder gibt es Einschränkungen? Sind »*alle Personen*« wirklich alle Personen, oder nur die Anwesenden, die Mitarbeiter, die Besitzer einer Eintrittskarte?

Einige Sprachregeln (cont)

	Regel	Erläuterung, Beispiel
R6	Überprüfen Sie Nominalisierungen.	Nomen (z. B. »Generierung«, »Datenverlust«) weisen oft auf einen komplexen Prozess hin, der beschrieben werden muss. Verwendet man das Substantiv »*Anmeldung*«, dann fehlen meist die Ergänzungen, die beim Verb »*anmelden*« erwartet werden: Wer meldet sich wo und wofür an?
R7	Erkennen und präzisieren Sie unbestimmte Substantive.	Oft bleibt unklar, ob es sich um einen generischen Begriff oder um ein bestimmtes Objekt handelt. Ist vom »*Bediener*« die Rede, so fragt es sich, ob es nur einen gibt oder welcher gemeint ist. Ähnliches gilt für viele Begriffe (Gerät, Meldung).
R8	Klären Sie die Zuständigkeiten bei Möglichkeiten und Notwendigkeiten.	Ein Zwang muss realisiert werden. Steht in einer Anforderung, dass etwas *möglich* oder *unmöglich* ist, *passieren kann*, *darf* oder *muss*, so ist zu klären, wer dies erzwingt oder verhindert.
R9	Erkennen Sie implizite Annahmen.	Begriffe in den Anforderungen (»*die Firewall*«), die nicht erläutert sind, deuten oft auf implizite Annahmen (hier vermutlich auf die, dass es *eine Firewall gibt*).

Beispiele

Schlagwort	bessere Anforderung
einfach bedienbar	Das Programm soll auch von Laien ohne weitere Einweisung benutzt werden können.
robust	Eine Bedienung über die Tastatur darf unter keinen Umständen zu einem irregulären Abbruch des Programms führen.
portabel	Das Programm muss von einem Entwickler in höchstens 6h von Windows XP auf Linux portiert werden können.
übersichtlich kommentiert	Die Module des Programms müssen einen Kopfkommentar nach Std. xyz enthalten.
plattformunabhängig	Alle prozessorspezifischen Teile des Programms müssen in einem speziellen Modul liegen.
nicht zu große Module	Module dürfen max. 300 Zeilen ausführbaren Code enthalten.
angemessenes Handbuch	Das Benutzungshandbuch wird gemäß Richtlinie ABC aufgebaut. Es wird vom Gutachter N.N. geprüft.