Requirements Engineering

Grundlagen und Überblick

Martin Glinz

Institut für Informatik der Universität Zürich

Juni 2002

ifi

Requirements Engineering – Grundlagen und Überblick

Inhalt

1 Einleitung

1.1 Definitionen und grundlegende Begriffe

Anforderung (requirement). 1. Eine Bedingung oder Fähigkeit, die von einer Person zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird. 2. Eine Bedingung oder Fähigkeit, die eine Software erfüllen oder besitzen muss, um einen Vertrag, eine Norm oder ein anderes, formell bestimmtes Dokument zu erfüllen. (IEEE 610.12-1990)

Anforderungsspezifikation. Die Zusammenstellung aller Anforderungen an eine Software. Synonyme: Anforderungsdokument, Software Requirements Specification.

„Die Spezifikation“: Im Alltag nicht immer eindeutig:

- *Dokument* oder *Prozess*
- *Anforderungs-*, *Lösungs-* oder *Produkt*spezifikation

Requirements Engineering (Anforderungstechnik). 1. Das systematische, *disziplinierte* und *quantitativ erfassbare* Vorgehen beim Spezifizieren, d.h. Erfassen, Beschreiben und Prüfen von Anforderungen an Software. 2. *Verstehen* und *Beschreiben*, was die Kunden *wünschen* oder *brauchen*.

Pflichtenheft ➜ verschiedene Begriffe:

- Synonym zu Anforderungsspezifikation
- Spezifikation + Überblick über Lösung
- Spezifikation + Elemente der Projektabwicklung

➜ „Pflichtenheft“ mit Vorsicht verwenden

1.2 Aufgaben des Requirements Engineerings

- ☆ Gewinnen und Analysieren
- ☆ Darstellen
- ☆ Prüfen
- ☆ Verwalten

} von Anforderungen

- ❑ Nicht linear! ➜ Inkrementeller, iterativer Prozess
- ❑ Einsatz verschiedener Gewinnungs- und Darstellungstechniken
- ❑ Aus Anwendersicht und zusammen mit den Anwendern spezifizieren
- ❑ Alle technischen Probleme, die nicht Bestandteil der Aufgabenstellung sind, werden nicht modelliert
- ❑ Anforderungen werden nicht einfach „entdeckt“, sondern müssen erarbeitet werden
- ❑ Spezifikation ist auch Konsensbildung

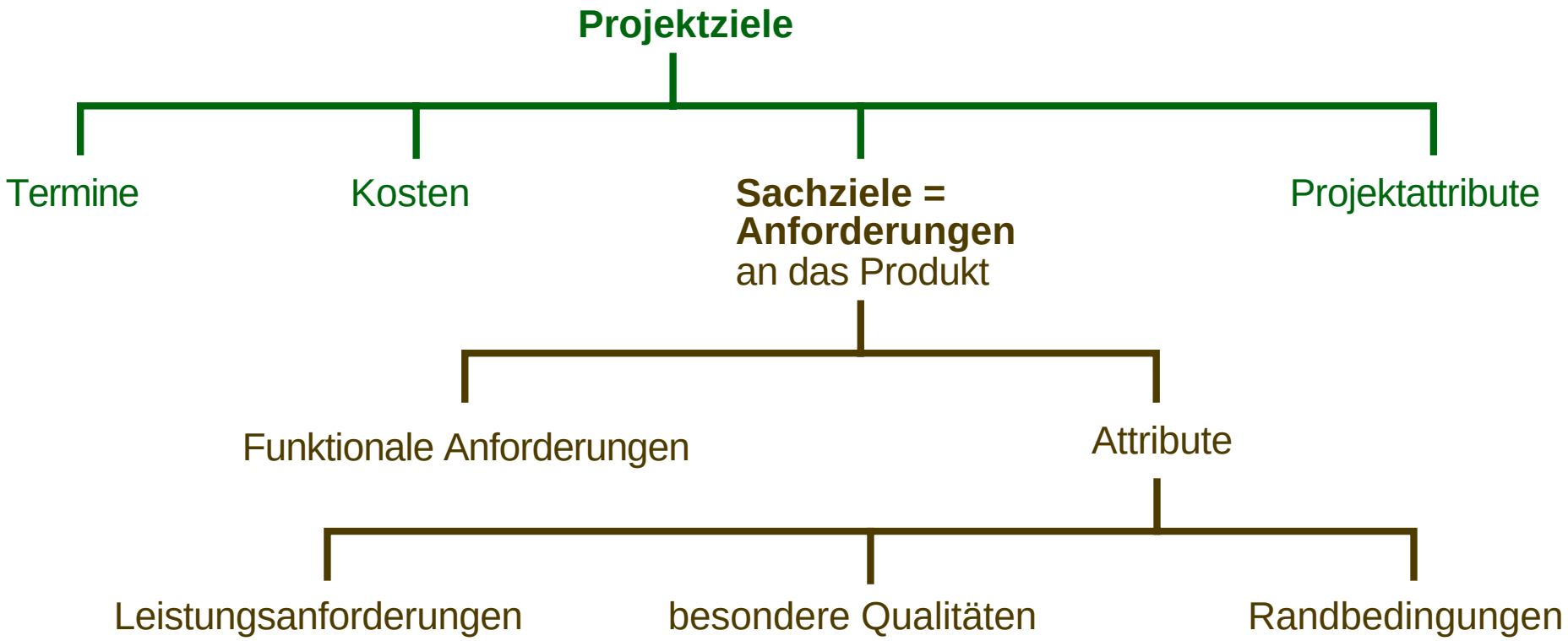
1.3 Warum Anforderungen spezifizieren?

- ☆ Mehr verdienen
  - ✧ Kosten senken
    - geringere Herstellungskosten (Senken der Fehlerkosten!)
    - geringere Pflegekosten
  - ✧ Mehr verkaufen
    - zufriedenere Kunden
    - früher am Markt
- ☆ Projektrisiken beherrschen
  - ✧ Risiken verkleinern
  - ✧ Zuverlässige Prognosen für Termine und Kosten

☞ Die wirtschaftliche Wirkung von Requirements Engineering ist immer indirekt; das RE selbst kostet nur!

1.4 Klassifikation von Anforderungen

Verschiedene Arten von Anforderungen:



1.5 Priorisierung von Anforderungen

- ☆ Muss-Anforderungen – unverzichtbar
- ☆ Soll-Anforderungen – wichtig, aber bei zu hohen Kosten verzichtbar
- ☆ Wunsch-Anforderungen – schön zu haben, aber nicht essenziell

Nötig bei

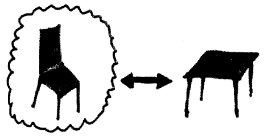
- ❑ harten Preisobergrenzen
- ❑ Beschaffung

Nützlich bei

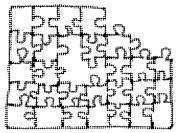
- ❑ Festlegung von Inhalt und Umfang der Inkremente bei inkrementeller Entwicklung
- ❑ Releaseplanung bei der Weiterentwicklung bestehender Systeme

1.6 Qualitätsmerkmale

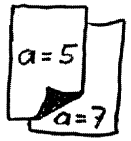
Merkmale einer guten Anforderungsspezifikation



Adäquatheit – das beschreiben, was der Kunde will bzw. braucht



Vollständigkeit – alles beschreiben, was der Kunde will bzw. braucht



Widerspruchsfreiheit – sonst ist die Spezifikation nicht realisierbar

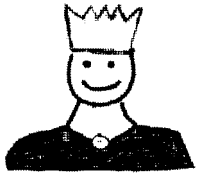
$$\bigwedge_{n\in\mathbb{N}} \bigwedge_{m_i \text{ ist } s} \sum_{i=1}^{n} |m_i| \geq P \wedge$$
$$\sum_{i=1}^{n-1} |m_i| < P \leftrightarrow W(m_1 \circ \ldots \circ m_n) \wedge \neg W(m_1 \circ \ldots \circ m_{n-1})$$

Verständlichkeit – für den Kunden und für die Informatiker

Eindeutigkeit – damit Fehler durch Fehlinterpretationen vermieden werden

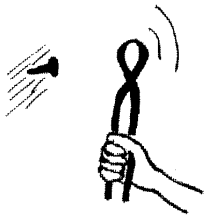
Prüfbarkeit – feststellen können, ob das realisierte System die Anforderungen erfüllt

Merkmale eines guten Spezifikationsprozesses

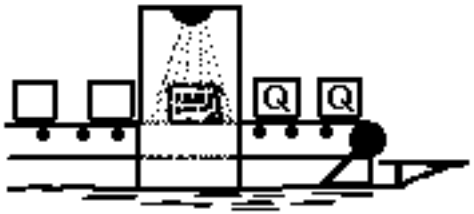


Kundenorientierung

Methodisches und zielgerichtetes Vorgehen



Verwendung geeigneter Mittel



Integration von Erstellung und Prüfung von Anforderungen

1.7 Anforderungen im Kontext, Betrachtungsebenen

Mindestens drei Betrachtungsebenen:

- ☆ Geschäft — «Auf dem bestehenden Schienennetz sollen mehr Leute transportiert werden»
- ☆ System — «Die Minimaldistanz zwischen zwei Zügen ist immer größer als der maximale Bremsweg des nachfolgenden Zuges.»
- ☆ Software — «Der maximale Bremsweg muss alle 100 ms neu berechnet werden.»
- ☆ Faktisch oft n Ebenen

- ☆ Spezifikation und Entwurf – WAS vs. WIE
- ☆ Verzahnung von Anforderungen und Lösungen ist unausweichlich
- ☆ Kontextbestimmung und -abgrenzung ist wichtig

WAS versus WIE im Requirements Engineering

Volksweisheit: WAS = Spezifikation, WIE = Entwurf

Aber: ist folgender Satz eine Anforderung oder eine Entwurfsentscheidung?

„Das System druckt eine wahlweise nach Namen oder Land alphabetisch sortierte Liste von Teilnehmern mit Nummer, Name, Vorname, Affiliation und Land. Auf jeder Seite sind unten links das Erstellungsdatum und unten rechts die Seitenzahl aufgedruckt.“

- ➜ WAS vs. WIE ist kontextabhängig und liefert keine brauchbare Abgrenzung zwischen Anforderungen und Entwurfsentscheidungen. Die gleiche Sache kann je nach Kontext beides sein.

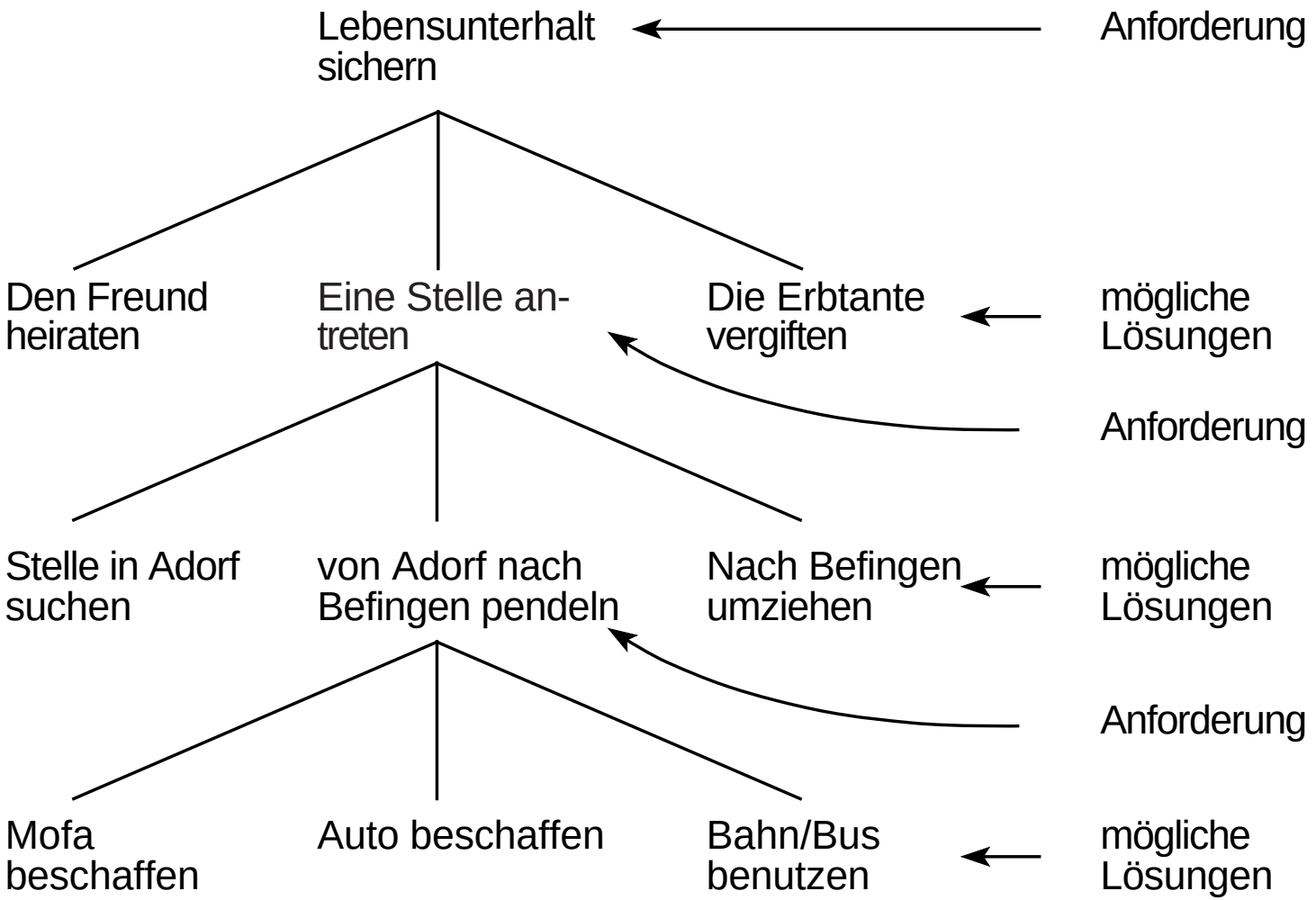
Dennoch ist in Projekten eine Unterscheidung häufig notwendig.

Möglichkeit: *operationale* Abgrenzung:

- ❍ Änderungen der Anforderungen brauchen die Zustimmung des Auftraggebers/Kunden
- ❍ Änderungen im Entwurf kann der Auftragnehmer/Lieferant autonom vornehmen

Betrachtungsebenen – Verzahnung von Anforderungen und Lösungen

Problem: Sonja Müller hat ihr Studium abgeschlossen und erhält keine Unterstützung von ihren Eltern mehr. Sie ist daher mit der Anforderung konfrontiert, ihren Lebensunterhalt zu sichern. Sie wohnt in Adorf und hat ein Stellenangebot bei einer Firma in Befingen. Ferner hat sie einen reichen Freund und eine ebenso reiche Erbtante.



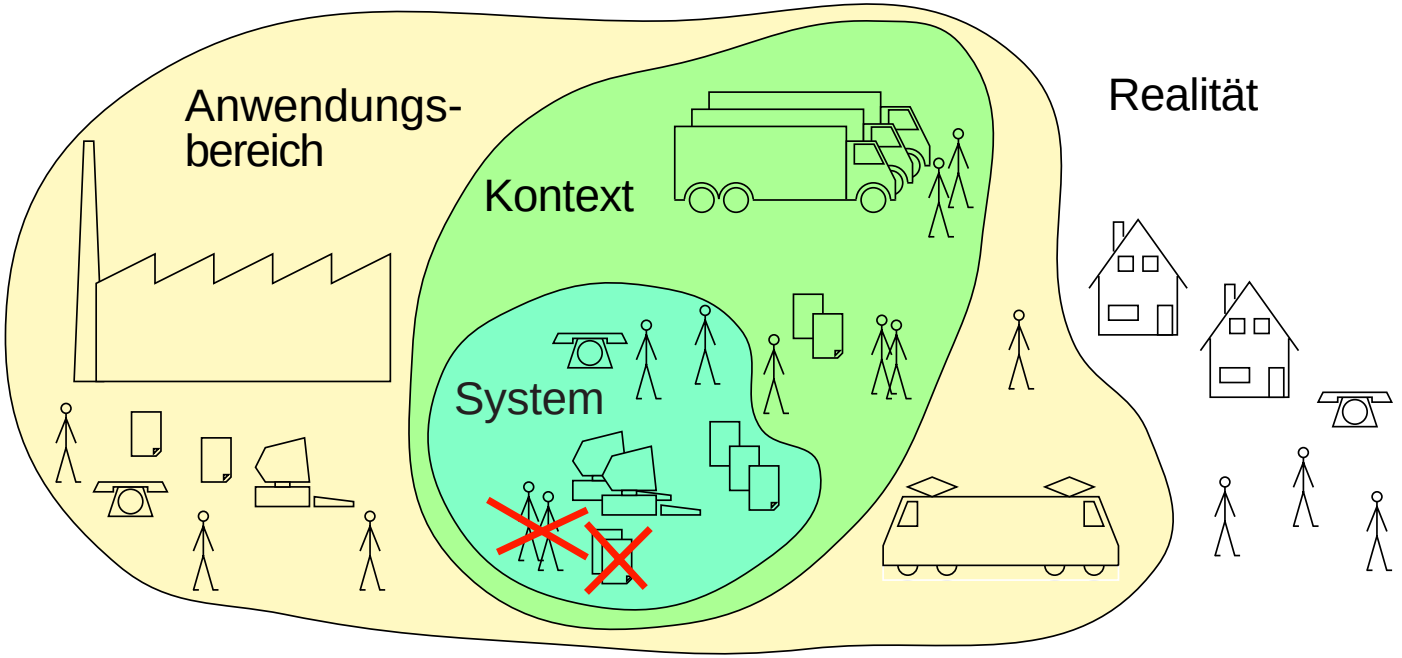
Sind Spezifikation von Anforderungen und Entwerfen von Lösungen voneinander trennbar?

In der Dokumentation: ja (siehe oben)

Im Prozess: nein

- Hierarchische Verzahnung: Übergeordnete Entwurfsentscheidungen beeinflussen untergeordnete Anforderungen
- Nicht realisierbare Anforderungen sind sinnlos ➜ Technische Machbarkeit (d.h. Lösungen) beeinflusst die Anforderungen
- Validierung: Anforderungen sind oft nur mit Hilfe von Lösungen (z.B. Prototypen) beurteilbar und validierbar
- Erkenntnisse und Schwierigkeiten bei der Lösung können Änderungen in den Anforderungen bewirken.

Einbettung und Abgrenzung



- Requirements Engineering bildet Modelle eines Ausschnitts der Realität
- Sobald ein System ein Problem der Realität bearbeitet, ist es eingebettet in die Realität und besitzt einen Kontext
- Requirements Engineering legt die Grenzen eines geplanten Systems fest
- Systeme verändern durch ihren Einsatz die Realität

2 Darstellung von Anforderungen

Darstellungsaspekte

- Funktionalität
- Attribute: Leistungen, Qualitäten, Randbedingungen

Freiheitsgrade

- Wahl der Mittel
- Art der Gliederung / Strukturierung
- Präzision
- Tiefe

2.1 Darzustellende Aspekte

Unabhängig von den verwendeten Gliederungs- und Darstellungsmethoden müssen folgende Aspekte beschrieben sein:

- ☆ Funktionaler Aspekt
  - ✧ Daten: Struktur, Verwendung, Erzeugung, Speicherung, Übertragung, Veränderung
  - ✧ Funktionen: Ausgabe, Verarbeitung, Eingabe von Daten
  - ✧ Verhalten: Sichtbares dynamisches Systemverhalten, Zusammenspiel der Funktionen (untereinander und mit den Daten)
  - ✧ Fehler: Normalfall und Fehlerfälle

- ☆ Leistungsaspekt
  - ✧ Datenmengen (durchschnittlich/im Extremfall)
  - ✧ Verarbeitungs- /Reaktionsgeschwindigkeit (durchschnittlich/im Extremfall)
  - ✧ Verarbeitungszeiten und -intervalle
  - ✧ Wo immer möglich: *messbare* Angaben!

- ☆ Qualitätsaspekt
  - ✧ Geforderte (nicht-funktionale) Qualitäten (z.B. Benutzerfreundlichkeit, Zuverlässigkeit)

- ☆ Randbedingungsaspekt
  - ✧ Einzuhaltende/zu verwendende Schnittstellen
  - ✧ Normen und Gesetze
  - ✧ Datenschutz, Datensicherung
  - ✧ Explizite Vorgaben des Auftraggebers

2.2 Wahl der Mittel

- Texte in natürlicher Sprache
- Strukturmodelle, in der Regel grafisch, angereichert mit natürlicher Sprache
- Interaktionsmodelle
- Formale Modelle auf der Grundlage mathematisch-logischer Formalismen

2.3 Struktur und Aufbau einer Anforderungsspezifikation

- ☆ Keine fixen Vorgaben
- ☆ Teilweise unternehmensinterne Standards
- ☆ IEEE 830-1993 als öffentlicher Standard
- ☆ Hier: einfache Struktur, angelehnt an IEEE 830-1993

- ❍ Drei Hauptkapitel
  1. Einleitung
  2. Überblick
  3. Einzelanforderungen
- ❍ Anhänge

  Glossar

  Verzeichnis referenzierter Dokumente

1. Einleitung

1.1 Anlass

Wozu das Dokument benötigt wird: Projekt, Kunde, Umfeld, ...

1.2 Ziele

Die Zielsetzung für das spezifizierte System im Überblick; bei größeren Systemen mehrstufig: Ziele, Teilziele

1.3 Einsatzbereich

Wo das spezifizierte System verwendet wird: Umgebung, Zielplattform,...

2. Überblick

1.2 Kontext

Betrachtungsebene, Kontext des spezifizierten Systems

2.2 Struktur der Problemstellung

Sachliche Struktur der dem spezifizierten System zugrundeliegenden Problemstellung
Gegebenenfalls Gliederung in Teilprobleme

2.3 Globale Attribute

Systemweit gültige Attribute (Leistungsanforderungen, besondere Qualitäten, Randbedingungen)

2.4 Annahmen

Annahmen, auf denen das spezifizierte System basiert

2.5 Perspektiven

Entwicklungsperspektiven für das spezifizierte System

3. Einzelanforderungen

   Auflistung aller Anforderungen

   Sinnvoll gegliedert, Möglichkeiten:

   - Hierarchisches Objektmodell: Hierarchische Gliederung des Problems in geschlossene Teilprobleme
   - Unterkapitelgliederung; die Unterkapitel fassen logisch zusammengehörige Teile zusammen; Unterkapitel in funktionale Anforderungen und Attribute gegliedert

Anhänge

- Glossar

  Verzeichnis aller verwendeten Fachbegriffe mit Definitionen, Abkürzungen, Synonymen, etc.

- Verzeichnis referenzierter Dokumente

  Nachweis aller Dokumente auf die in der Anforderungsspezifikation Bezug genommen wird

2.4 Präzision

«Die Teilnehmer-Eingabemaske enthält Felder für Name, Vorname, Stellung, Geschlecht,...»

vs.

«...Geschlecht: zwei Ankreuzknöpfe, beschriftet mit männlich und weiblich, Voreinstellung männlich, Ankreuzungen schließen sich gegenseitig aus, eine Ankreuzung erforderlich, Hin- und Herwechseln der Ankreuzung mit Leertaste. ...»

- Präzise darstellen:
  - Wer tut?
  - Was passiert?
  - Welche Resultate?
  - Mit welchen Attributen?
- Sich klar und deutlich ausdrücken, z.B.
  - Aussagekräftige Namen wählen
  - Definierte Subjekte: kein „man“, kein „es wird“
  - Definierte Bedeutungen für „kann“, „muss“, „soll“, etc.

2.5 Tiefe

«Teilnehmerverwaltung: Erfasst Teilnehmer, speichert ihre Daten permanent und ermöglicht Auswertungen durch Formulierung von Anfragen.»

vs.

«...

~.3 Teilnehmerverwaltung

~.3.1 Erfassen neuer Teilnehmer

~.3.1.1 Eingabemaske

...

~.3.2 Mutieren erfasster Teilnehmer

...»

2.6 Ausgewählte Darstellungsmethoden und -sprachen

2.6.1 Spezifikation mit natürlicher Sprache

- Weit verbreitet
- Nummerierung und Gliederungsschemata als Strukturierungsmittel
- Linguistische Methoden (Sätze mit Standardstruktur, kein Passiv, beschränkte Menge von Verben mit festen Bedeutungen...) zur Verbesserung der Qualität

\+ leicht zu schreiben und zu lesen

\+ ausdrucksmächtig

– unübersichtlich

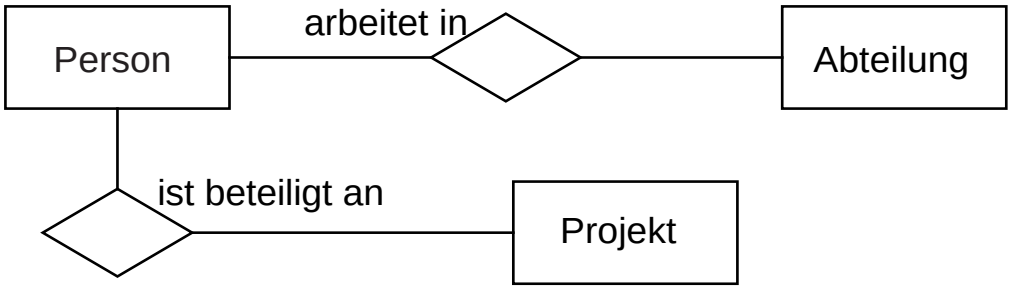
– fehlerträchtig

– schwierig zu prüfen

➜ weniger geeignet als alleiniges Mittel zur Beschreibung von Spezifikationen

2.6.2 Datenmodellierung

- Grundlage: Entity-Relationship Ansatz
- modelliert einen Ausschnitt der Realität mit Hilfe von *Gegenstandstypen* (entity types), *Beziehungstypen* (relationship types) und *Attributen* (attributes)

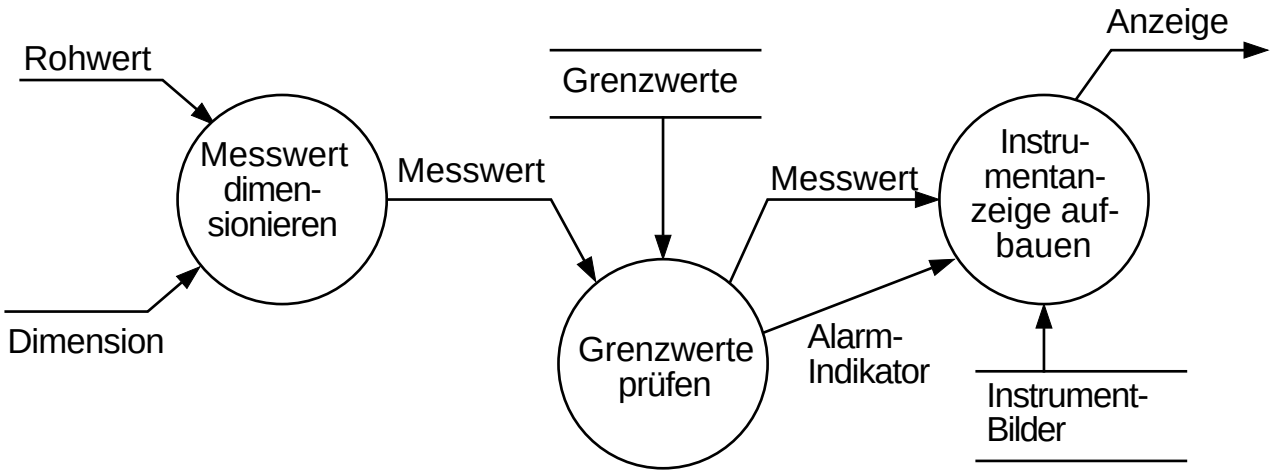


+ Einfach und klar
+ Leicht auf Datenbank-Realisierungen abbildbar

– Ignoriert Funktionalität und Verhalten der Systeme
– Keine Mittel zur Systemdekomposition
– Keine Lokalität oder Einkapselung von Daten

2.6.3 Strukturierte Analyse

☆ Modelliert die Funktionalität eines Systems mit Hilfe von Datenflussdiagrammen



\+ Sehr anschaulich

\+ Unterstützt Systemdekomposition

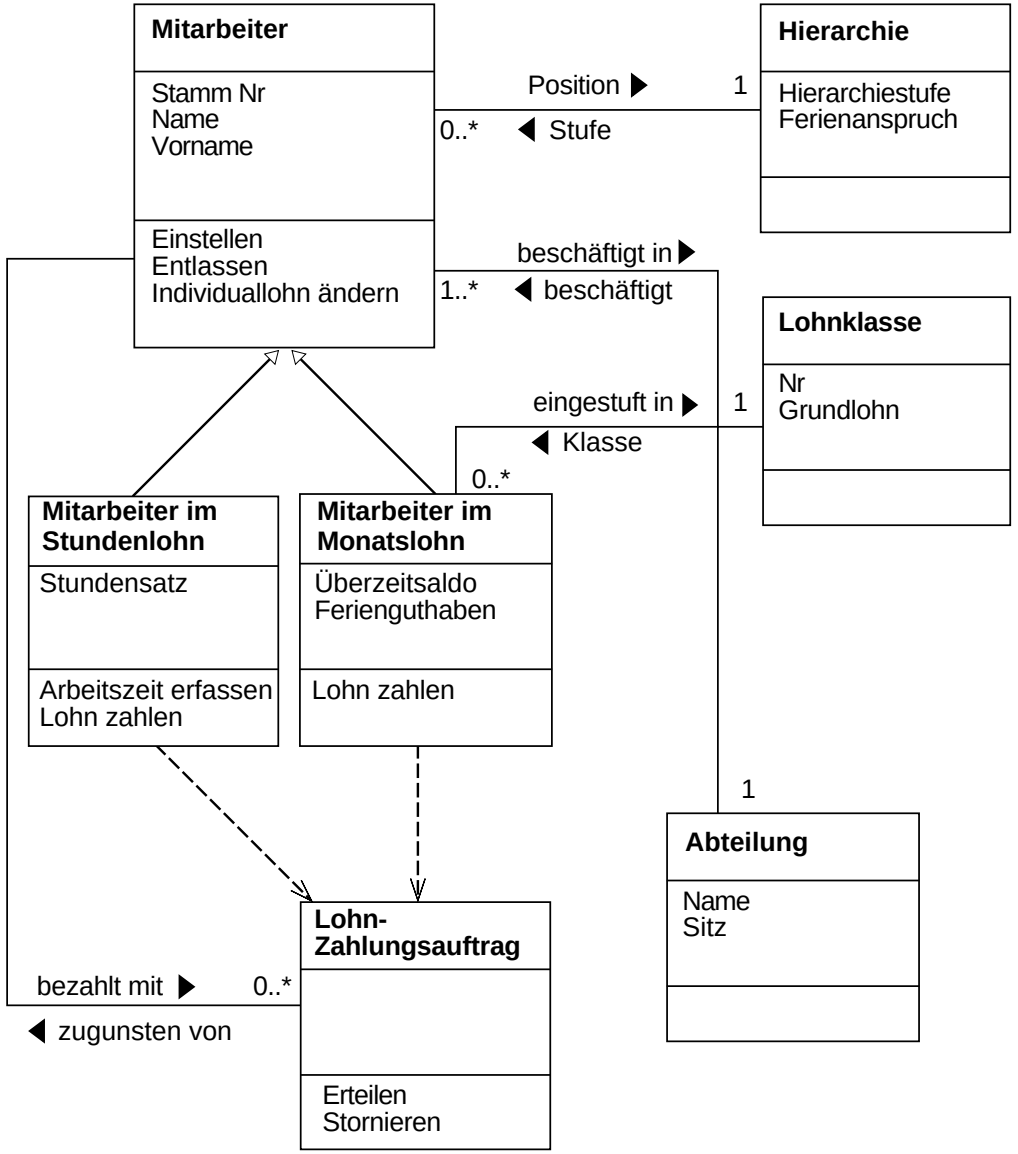
– Strukturbruch zwischen Spezifikation und Implementierung

– Keine Lokalität von Daten, Einkapselung nur begrenzt möglich

– Keine speziellen Verfahren/Darstellungsmittel für nicht-funktionale Anforderungen

2.6.4 Objektorientierte Spezifikation

- ☆ Modelliert die statische Struktur eines Systems mit Hilfe von Objekt- oder Klassendiagrammen
- ❍ Objekt- oder Klassenmodell als Basis
  - ❍ *Objekt* – Abbild eines Elements der Realität:
    „Eva Müller“, „Klaus Meier“
  - ❍ *Klasse* – Zusammenfassung gleichartiger Objekte:
    „Mitarbeiter“
- ❍ Objekte/Klassen beschreiben Daten, Funktionen und zeitliches Verhalten



+ Gut geeignet zur Beschreibung der Systemstruktur
+ Unterstützt Lokalität von Daten und Einkapselung von Eigenschaften
+ Erlaubt strukturähnliche Implementierungen
+ Systemdekomposition möglich

– Funktionalität aus Benutzersicht schlecht modellierbar
– Keine speziellen Verfahren/Darstellungsmittel für nicht-funktionale Anforderungen
– Dekomposition häufig nicht unterstützt

2.6.5 Szenarien und Anwendungsfälle

- ☆ Modellieren die Interaktion zwischen systemexternen Akteuren und dem System
- ☆ Jede Interaktionssequenz wird durch ein Szenario (einen Anwendungsfall) beschrieben

Buch ausleihen

1. Ausweiskarte der Benutzerin lesen und Angaben überprüfen
2. Signatur eines Buchs lesen und zugehörigen Katalogeintrag ermitteln
3. Ausleihe registrieren und Diebstahlsicherungsetikett deaktivieren
4. ...

+ Modelliert die Funktionalität aus Benutzersicht: leicht verstehbar und überprüfbar
+ Hilft bei der Abgrenzung zwischen System und Kontext
+ Dekomposition möglich

– Zusammenhänge und Abhängigkeiten zwischen den Szenarien werden nicht modelliert
– Statische Struktur, insbesondere notwendige Daten, werden nicht modelliert

2.6.6 Formale Spezifikation

- Modelle auf der Grundlage mathematisch-logischer Formalismen
- Formal definierte Syntax und Semantik
- Spielt trotz theoretischer Vorteile nur marginale Rolle in der Praxis
- Einsatz punktuell sinnvoll und möglich, vor allem für sicherheitskritische Komponenten

Berechtigung erteilen

Δ Autorisierung
einzusehendesDokument?: Dokument
Autorisierter?: Person

---

einzusehendesDokument? $\in$ Bestand \ **dom** gesperrt
Autorisierter? $\in$ Mitarbeiter
autorisiert' = autorisiert $\cup$ {(einzusehendesDokument?, Autorisierter?)}
Bestand' = Bestand
Mitarbeiter' = Mitarbeiter
gesperrt' = gesperrt

- Formale Spezifikationsmodelle erlauben
  - die Gültigkeit von Eigenschaften zu *beweisen* (zum Beispiel für eine Liftsteuerung, dass sie nie einen Lift mit offenen Türen fahren lässt)
  - die Gültigkeit von Eigenschaften *automatisiert zu testen* und ggf. Gegenbeispiele zu finden (Model Checking)

+ Immer eindeutig (da Semantik formal definiert)
+ Widerspruchsfreiheit formal prüfbar
+ Erfüllung wichtiger Eigenschaften beweisbar
+ Lösungsneutral
+ Formale Verifikation von Programmen möglich

– Erstellung sehr aufwendig
– Prüfung/Nachweis der Vollständigkeit wird nicht einfacher
– Nicht ohne profunde Ausbildung lesbar → Prüfung auf Adäquatheit schwierig
– Große Spezifikationen auch für Fachleute schwer zu verstehen

2.7 UML (Unified Modeling Language)

- UML ist eine Sammlung vorwiegend grafischer Sprachen zur Erstellung von Anforderungs- und Entwurfsmodellen aus verschiedenen Perspektiven
- Eine UML-Spezifikation besteht aus einer Sammlung sich ergänzender und teilweise überlappender Modelle
- Im Zentrum steht ein *Klassenmodell*, das den strukturellen Aufbau eines Systems spezifiziert
- Das Klassenmodell beschreibt
  - bei Anforderungsmodellen die Gegenstände der Realität, mit denen das System umgehen muss
  - bei Entwurfs- und Implementierungsmodellen die Klassen der Lösung
- Bei der Modellierung von Anforderungen kommt als zentrales Element das *Anwendungsfallmodell* hinzu, das die Benutzer-System-Interaktion aus Benutzersicht modelliert
- Nach Bedarf beschreiben weitere Modelle zusätzliche Systemsichten, insbesondere dynamisches Verhalten, Funktionalität und Zusammenarbeit

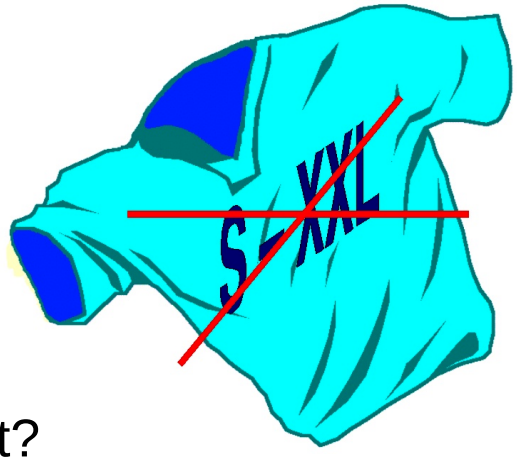
3 Der Spezifikationsprozess

3.1 Grundsätze

„Der“ Spezifikationsprozess besteht aus einer Reihe von Teilprozessen:

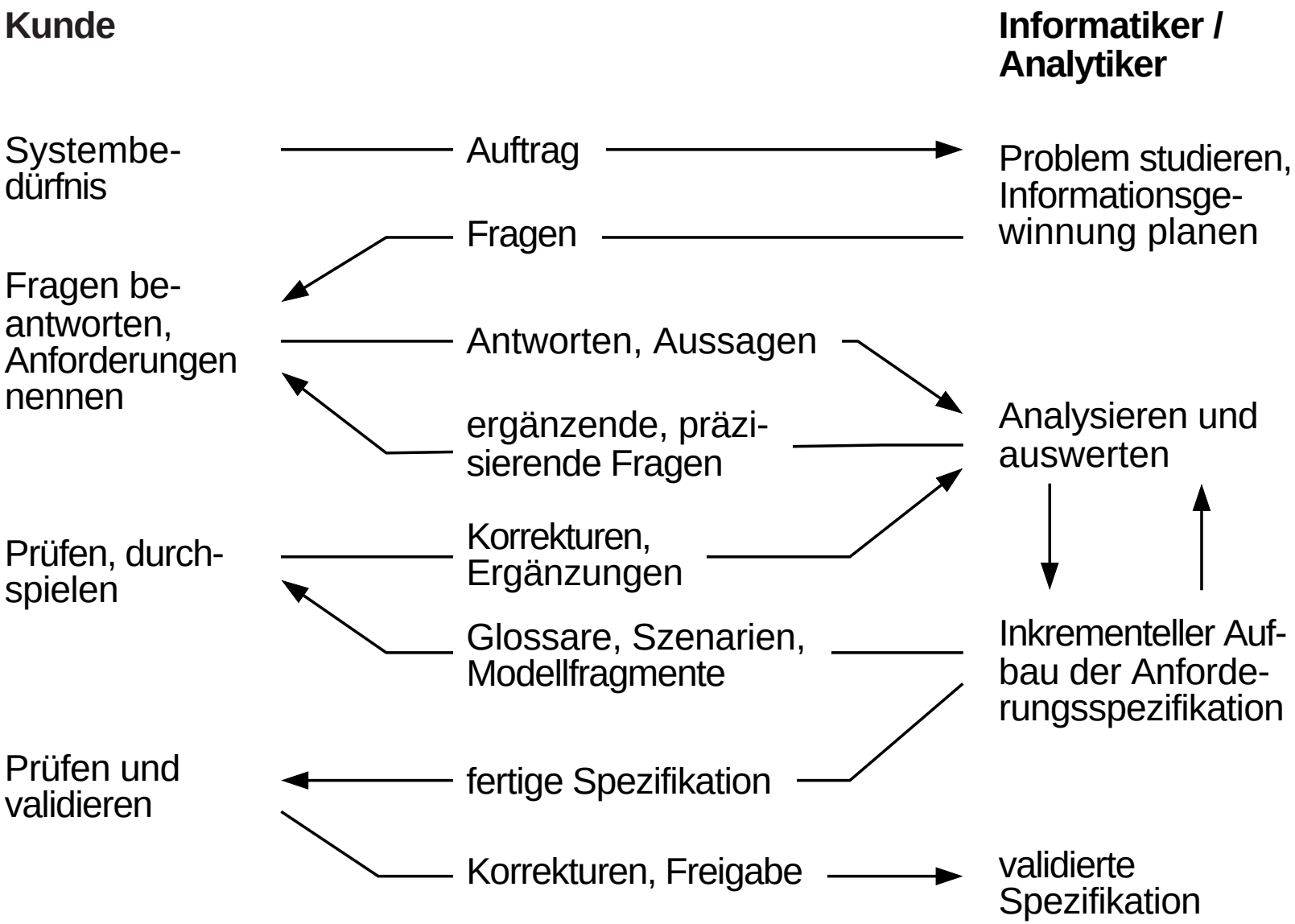
- Anforderungen spezifizieren
  - gewinnen
  - dokumentieren
  - prüfen
- Anforderungen verwalten
  - freigeben
  - ändern
  - rückverfolgen (wo kommt eine Anforderung her)
  - vorwärtsverfolgen (wo wird eine Anforderung verwendet)

Es gibt keinen idealen RE-Prozess (keine Einheitsgröße für alles)



- zuschneiden auf konkrete Projektsituation
- zu berücksichtigende Faktoren:
  - lineares oder inkrementelles Vorgehen im Projekt?
  - muss die Spezifikation wasserdicht sein (Vertrag; Realisierung durch Dritte)?
  - sind die Kunden/Benutzer bekannt und können sie in die Erstellung der Spezifikation einbezogen werden?
  - wird das zu spezifizierende System im Kundenauftrag oder für den Markt entwickelt?
  - soll Standardsoftware zum Einsatz kommen?

Typischer, interaktiver Prozess für die Spezifikation von Anforderungen:



3.2 Voraussetzungen

Beteiligtenanalyse (stakeholder analysis)

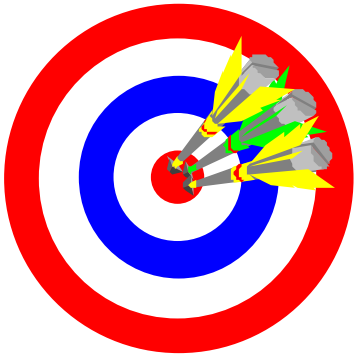
- Wer hat in welcher Rolle hat mit dem zu erstellenden System zu tun?
- Wer kann/soll/darf/muss Anforderungen an das System stellen?
- Wer stellt Anforderungen an den Projektablauf?
- Wer kann/soll/darf/muss Randbedingungen vorgeben?

Typische Beteiligtenrollen

- Endbenutzer
- Auftraggeber
- Betreiber
- Entwickler
- Projektleitung
- und viele weitere ...

Zielanalyse (goal analysis)

- ☆ Welches sind die übergeordneten Ziele («Geschäftsziele», «Vision») für das anstehende Vorhaben?
- ☆ Welchen Nutzen hat welcher Beteiligte von der Erreichung eines Ziels?
- ☆ Wie wird die Zielerreichung festgestellt?
  - Abnahmebedingungen
  - quantifizierte, messbare Ziele
- ☆ Gibt es Zielkonflikte?
  - Wenn ja, Ziele priorisieren
- ☆ Etwa drei bis sieben übergeordnete Ziele formulieren



3.3 Gewinnung von Anforderungen

Aufgaben

- Wünsche und Bedürfnisse der Beteiligten erkennen, analysieren und darstellen
- Den Beteiligten Möglichkeiten aufzeigen, wenn diese sie selbst nicht erkennen
- Wenn nötig, zunächst den IST-Zustand erheben
- Bei Produktentwicklungen Marktpotential klären
- Randbedingungen erkennen, analysieren und dokumentieren

Schwierigkeiten und Hindernisse

- Erwartungs- und Begriffsdiskrepanzen bei den Beteiligten
- Beteiligte wissen zwar, was sie wollen, können ihre Vorstellungen aber nicht formulieren
- Beteiligte wissen nicht, was sie wollen
- Beteiligte haben verdeckte Ziele, die sie absichtlich nicht offen legen

⇨ Requirements Engineering ist immer auch

- Aufgabenklärung
- Konsensbildung
- Konflikterkennung und -auflösung

3.3.1 Organisationsformen für die Anforderungsgewinnung

Form	**Eignung für...**			
	Wünsche ausdrücken	Möglichkeiten aufzeigen	IST-Zustand erheben	Marktpotenzial klären
Interviews	+	–	+	o
Beobachtung der Benutzer	o	–	+	o
Rollenspiele	+	o	o	–
Beispiele analysieren	o	–	+	–
Staffagen und Prototypen	o	+	–	o
Umfragen/Fragebogen	o	–	+	+
Gemeinsame Arbeitstagungen	+	o	o	–
Marktstudien	–	–	o	+
Problemmeldungsauswertung	+	–	–	o
Benchmarking	o	+	–	+

3.3.2 Methodik der Problemanalyse

Fokus der Analyse:

A. Anwendungsbereich – Ziel: Anwendungsbereich verstehen, grundsätzliche Probleme des Anwendungsbereichs kennen, konkretes Problem in seinem Anwendungskontext verstehen

B. Konkretes Problem – Ziel: Konkretes Problem verstehen und Anforderungen an eine Lösung formulieren

- Strukturorientierte Methoden:
  Begriffe, Objekte, strukturelle Zusammenhänge analysieren/erkennen
- Prozessorientierte Methoden:
  Prozesse, Interaktionen, Abläufe, dynamisches Verhalten analysieren/erkennen

Ausgewählte Methoden der Problemanalyse

Anwendungsbereich	konkretes Problem	
Begriffe bestimmen (Glossar) Objekte des Anwendungs-bereichs modellieren (Geschäftsobjektmodell)	Objekt- bzw. Klassenmodell des Problems erstellen Problem hierarchisch in Teilprobleme zerlegen	**struktur-orientiert**
Prozesse des Anwendungs-bereichs beschreiben (Geschäftsprozessmodell)	Benutzungsszenarien beschreiben Dynamisches System-verhalten modellieren	**prozess-orientiert**

3.3.3 Die Rolle der IST-Analyse

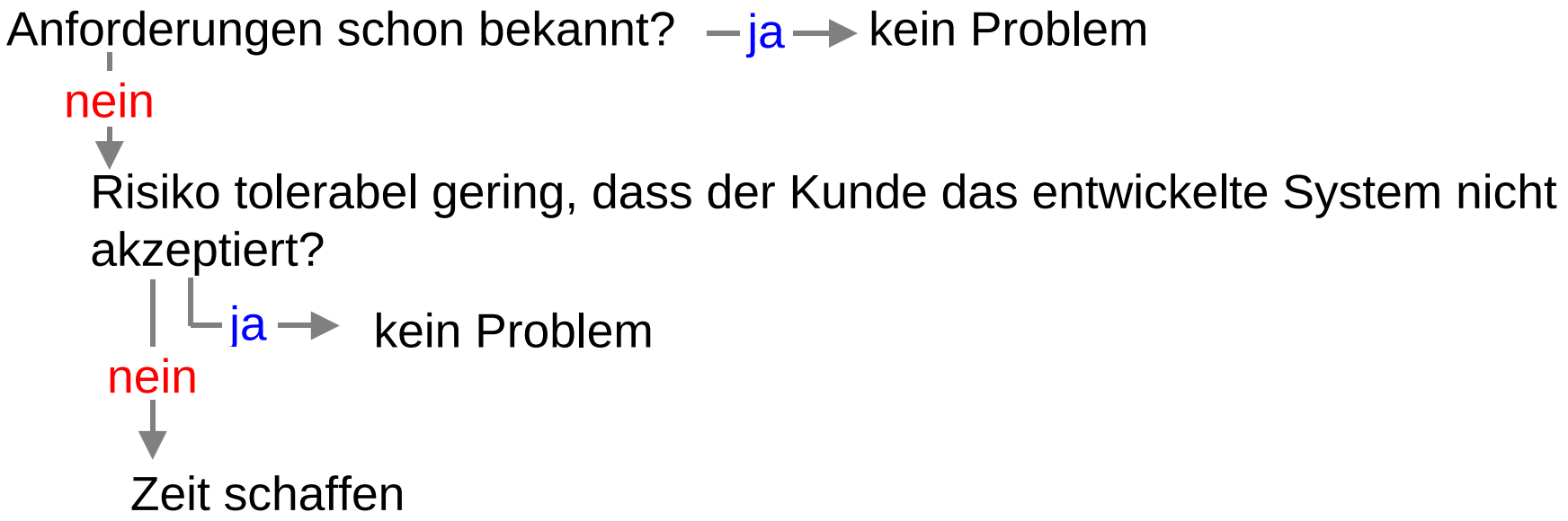
- ☆ Traditionelles Vorgehen
  (vor allem bei Informationssystemen; McMenamin und Palmer 1984):

  (1) Physischen IST-Zustand erheben
  (2) Essentiellen IST-Zustand extrahieren
  (3) Essentiellen SOLL-Zustand (= Anforderungen) ermitteln

- ☆ Heute nicht mehr zeitgemäß

- ➪ Beschäftigung mit IST-Zustand *nur wenn notwendig*
  - ❍ zum Verstehen der Arbeit und der Bedürfnisse der Benutzer
  - ❍ zur Ermittlung von Stärken und Schwächen des IST-Systems (falls diese nicht bekannt sind)
  - ❍ zur Übernahme von Teilen eines IST-Systems

3.4 Keine Zeit für die Spezifikation von Anforderungen?



- Risikogerecht spezifizieren:

  Der *Aufwand* für das Requirements Engineering soll *umgekehrt proportional* zum *Risiko* sein, das man bereit ist, einzugehen.

Was ist (fast immer) unverzichtbar?

- Die wichtigen Beteiligten kennen und einbeziehen
- Das Problem, das zum Bedarf für das zu spezifizierende System geführt hat, kennen
- Die drei bis sieben wichtigsten Systemziele identifizieren und aufschreiben
- Für jedes Ziel
  - Analysieren und festhalten, zu welchem Geschäftsziel das Ziel wie beiträgt
  - Analysieren und festhalten, welchen Nutzen das Ziel für welchen der Beteiligten hat
- Die Schlüsselbegriffe des Systems und des Anwendungsbereichs in einem Glossar definieren
- Die Hauptfunktionen identifizieren und dokumentieren
- Kritische Randbedingungen und Risiken identifizieren und dokumentieren

Erschwerende Faktoren ($\rightarrow$ Mehr Aufwand für das RE notwendig)

- Hohe Komplexität des Anwendungsbereichs
- Entwicklungsteam mit dem Anwendungsbereich nicht vertraut
- Viele Beteiligte
- Verteilte Entwicklung und/oder Beteiligte
- Lange Durchlaufzeit
- Sicherheitskritische Anforderungen
- Hohe Projektrisiken

4 Prüfen von Anforderungen

4.1 Prinzipien

- ☆ Abweichungen von der geforderten Qualität der Spezifikation feststellen
- ☆ möglichst viele Fehler, Lücken, Unklarheiten, Mehrdeutigkeiten, etc. finden und beheben

- ☆ Konflikte zwischen den Wünschen / Forderungen verschiedener beteiligter Personen erkennen und lösen
- ☆ Verdeckte Wünsche / Erwartungen / Befürchtungen erkennen und thematisieren

4.2 Organisation

Beteiligte

- ☆ Informatiker
- ☆ Kunde(n)
- ☆ beide gemeinsam

} je nach Verfahren

Zeitpunkt(e)

(1) Fortlaufend, d.h. begleitend zur Erstellung der Spezifikation, z. B. Autor-Kritiker-Zyklus

(2) Wenn die Spezifikation fertig ist (aber noch genug Zeit bleibt, die gefundenen Mängel zu beheben)

4.3 Prüfverfahren

- Reviews
- Prüf- und Analysemittel in Werkzeugen
- Simulation/Animation
- Prototypen

Review

- Das Mittel der Wahl zur Prüfung von Spezifikationen
- Walkthrough: Autor führt durch das Review
- Inspektion: Gutachter prüfen eigenständig, tragen in Sitzung Befunde zusammen
- Autor-Kritiker-Zyklus: Kunde liest und kritisiert, bespricht Befunde mit Autor

Prüf- und Analysemittel in Werkzeugen

- Einsatz bei werkzeuggestützter Erstellung der Spezifikation
- Auffinden von *Lücken* und *Widersprüchen*

Simulation/Animation

- Untersuchung der *Adäquatheit* des Systemverhaltens
- Dynamisches Verhalten des spezifizierten Systems wird durch Simulator ausgeführt und/oder durch Animator visualisiert

Prototyp

- Beurteilung der Adäquatheit / praktischen Brauchbarkeit des spezifizierten Systems anhand der im Prototyp realisierten Systemteile
- *Erprobung* eines Systems in der geplanten *Einsatzumgebung*
- Modell für die weitere Entwicklung oder für das zu schaffende Produkt
- Mächtigstes (aber auch aufwendigstes) Mittel zur Beurteilung der *Adäquatheit*

5 Verwalten von Anforderungen (Requirements Management)

- ☆ Beherrschen der Evolution von Anforderungen
- ☆ Rückverfolgbarkeit (Traceability)

5.1 Evolution von Anforderungen

Aufgrund vielfältiger Ursachen unterliegen Anforderungen einer *Evolution*

- ❍ Fortschritte der Technologie
- ❍ Änderung der unternehmensinternen Organisation
- ❍ Veränderte Bedürfnisse von Kunden
- ❍ Veränderung von Märkten / neue Märkte
- ❍ Neue / geänderte politische oder rechtliche Randbedingungen

Problem:

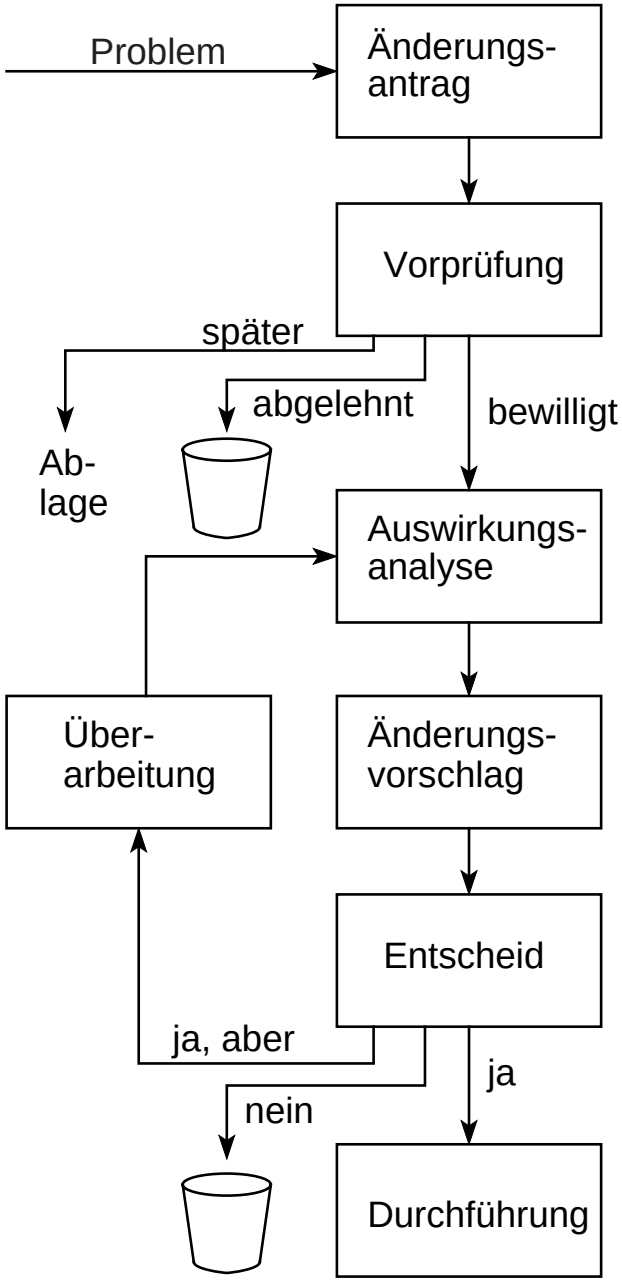
- Anforderungen stabil halten und
- Veränderung kontrolliert zulassen

⇨ Konfigurationsmanagement für Anforderungen

- Anforderungen einzeln identifizierbar
- geordneter Änderungsprozess
- Geregelter Dokumentenfluss
- Klare Zuständigkeiten und Verantwortlichkeiten
- Rückverfolgbarkeit aller Entscheide und Änderungen

Der Änderungsprozess für Anforderungen

Entscheidungen durch *Steuerkomitee* (typisch zusammengesetzt aus Vertretern von Auftraggeber und Auftragnehmer sowie dem Projektleiter)



Neue oder zu ändernde Anforderung

- Bedeutung?
- Notwendigkeit?
- Priorität?
- Jetzt nicht, aber in späterer Version?

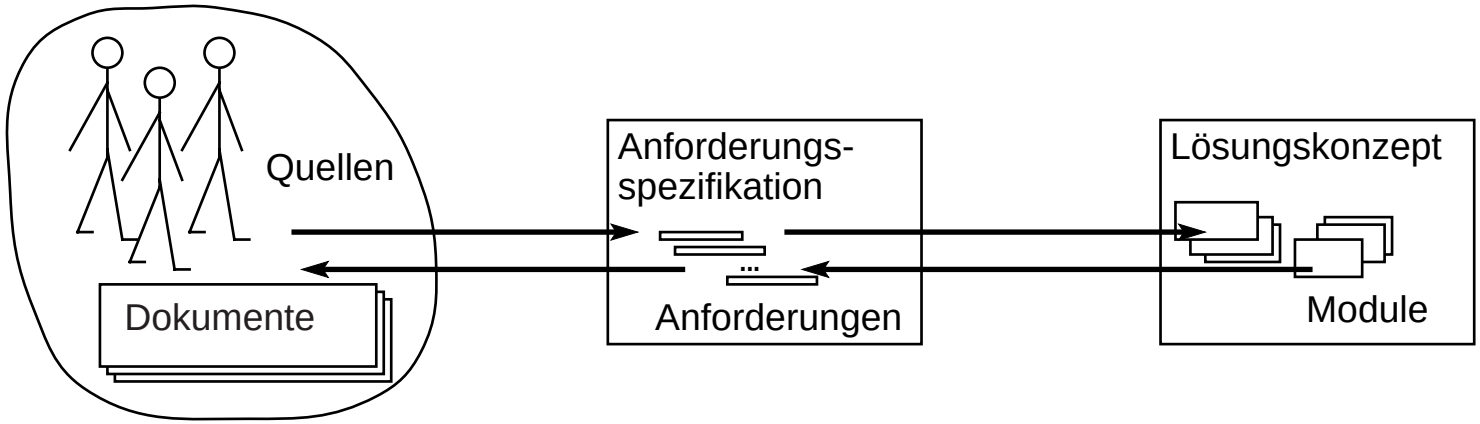
- Welche Anforderungen sind betroffen?
- Welche schon erstellten Teile der Lösung sind betroffen?
- Kosten/Zeitaufwand für Durchführung der Änderung?

Durch Steuerkomitee

- Anforderungsspezifikation ändern
- Alle betroffenen Teile der Lösung ändern

5.2 Rückverfolgbarkeit (traceability) von Anforderungen

- Wo kommt welche Anforderung her?
- Wo ist welche Anforderung entworfen bzw. implementiert?
- ☆ Jeweils vorwärts und rückwärts:



- Aufwand und Ertrag müssen gegeneinander abgewogen werden
- Rückverfolgungsbeziehungen müssen gepflegt werden, sonst sind sie nutzlos
- Benötigt Werkzeugunterstützung

Zusammenfassung

Requirements Engineering

- ☆ heißt Anforderungen
  - ☆ gewinnen
  - ☆ aufschreiben
  - ☆ prüfen
  - ☆ verwalten
- ☆ legt den Grundstein für die Qualität des Produkts

Requirements Engineering –

Es von Anfang an richtig machen.

Literatur

Einführende Literatur zum Requirements Engineering

Davis, A.M. (1993). *Software Requirements : Objects, Functions and States*. Englewood Cliffs, N.J.: Prentice Hall.

Gause, D.C., G.M. Weinberg (1989). *Exploring Requirements: Quality before Design*. New York: Dorset House. 299 p. [In deutscher Übersetzung erschienen 1993 als: *Software Requirements : Anforderungen erkennen, verstehen und erfüllen*. München: Hanser.]

IEEE (1993). *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Standard 830-1993.

Jackson, M.A. (1995). *Software Requirements and Specifications : A Lexicon of Practice, Principles and Prejudices*. Addison-Wesley (ACM Press books): Wokingham, etc. 228 p.

Kotonya, G., I. Sommerville (1998). Requirements Engineering: Processes and Techniques. Chichester: John Wiley & Sons. 282 p.

Robertson, S., Robertson, J. (1999). *Mastering the Requirements Process*. Addison-Wesley.

Sommerville, I., P. Sawyer (1997). *Requirements Engineering*. Chichester, etc.: John Wiley & Sons. 391 p.

Quellen und weiterführende Literatur

Björner, D., C. Jones (1978). The Vienna Development Method. Berlin, etc.: Springer.

Booch, G. (1986). *Object-Oriented Development*. IEEE Transactions on Software Engineering **12**, 2 (Feb. 1986). 211-221.

Booch, G., Jacobson, I., Rumbaugh, J. (1999). *The Unified Modeling Language User Guide.* Reading, Mass., etc.: Addison-Wesley.

Coad, P., E. Yourdon (1991). *Object-Oriented-Analysis*. Prentice-Hall, Englewood Cliffs, N.J. [auf Deutsch: Objektorientierte Analyse, 1994 im gleichen Verlag]

DeMarco, T. (1978). *Structured Analysis and System Specification*. Yourdon Press, New York.

Firesmith, D.C. (1994). Modeling the Dynamic Behavior of Systems, Mechanisms and Classes with Scenarios. *Report on Object Analysis and Design (ROAD)* **1**, 2 (Jul/Aug 1994). 32-36, 47.

Glinz, M. (1987). *Objektorientierte, halbformale Spezifikation mit SPADES*. GI Fachtagung Requirements Engineering '87, St. Augustin (GMD-Studien Nr. 121). 163-174.

Glinz, M. (1991). Probleme und Schwachstellen der Strukturierten Analyse. In Timm, M. (Hrsg.): *Requirements Engineering '91*. Informatik-Fachberichte Band 273, Berlin, etc.: Springer. 14-39.

Glinz, M. (1995). An Integrated Formal Model of Scenarios Based on Statecharts. In Schäfer, W. and Botella, P. (eds.): *Software Engineering – ESEC '95*. Proceedings of the 5th European Software Engineering Conference, Sitges, Spain. Lecture Notes in Computer Science 989, Berlin, etc.: Springer. 254-271.

Glinz, M. (1998). *Software Engineering I*. Vorlesungsskript, Universität Zürich.

Glinz, M. (2000). A Lightweight Approach to Consistency of Scenarios and Class Models. *Proceedings 4th IEEE International Conference on Requirements Engineering*. Schaumburg, Ill. 49-58.

Glinz, M. (2000). Improving the Quality of Requirements with Scenarios. *Proceedings of the Second World Congress on Software Quality*. Yokohama. 55-60.

Harel, D. (1988). On Visual Formalisms. *Communications of the ACM* **31**, 5 (May 1988). 514-530.

IEEE (1990). *Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. IEEE Computer Society Press.

IEEE Software (1994). Special Issue on Requirements Engineering. *IEEE Software* **11**, 2 (March 1994).

Jackson, D. (2002). Alloy: A Lightweight Object Modelling Notation. *ACM Transactions on Software Engineering and Methodology* **11**, 2. 256-290.

Jacobson, I., M. Christerson, P. Jonsson, and G. Övergaard (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Amsterdam, etc.: Addison-Wesley.

Jacobson, I (1994). Basic Use Case Modeling. *Report on Object Analysis and Design (ROAD)* **1**, 2 (Jul/Aug 1994). 15-19.

Krabbel, A., I. Wetzel, S. Ratuski (1996). Objektorientierte Analysetechniken für übergreifende Aufgaben. GI-Fachtagung Softwaretechnik '96, Koblenz. *Softwaretechnik-Trends* **16**, 3 (Sept. 1996). 65-72.

McMenamin, S.M., J.F. Palmer (1984). *Essential Systems Analysis*. Yourdon Press, New York.

Meyer, B. (1998). *Object Oriented Software Construction*. 2nd ed. Prentice-Hall, Englewood Cliffs.

Oestereich, B. (1998). *Objektorientierte Softwareentwicklung*. R. Oldenbourg Verlag München.

Pohl, K. (1994). The Three Dimensions of Requirements Engineering: A Framework and Its Application. *Information Systems* **19**, 3 (June 1994). 243-258.

Pohl, K. (1996). *Requirements Engineering: An Overview.* Aachener Informatik-Berichte 96-05, Fachgruppe Informatik, RWTH Aachen.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, N.J.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Reading, Mass., etc.: Addison-Wesley.

Schach, S.R. (1996). Classical and Object-Oriented Software Engineering (3rd Edition). Chicago etc.: Irwin. 603 p.

Spivey, J.M. (1992). *The Z Notation: A Reference Manual*. Second Edition. Hemel Hempstead: Prentice Hall International.

Stein, W. (1994). *Objektorientierte Analysemethoden: Vergleich, Bewertung, Auswahl.* BI-Wissenschaftsverlag (Reihe Angewandte Informatik Band 12), Mannheim. 310 p.

Wirfs-Brock, R., B. Wilkerson, L. Wiener (1990). *Designing Object-Oriented Software*. Prentice-Hall, Englewood Cliffs, N.J.

Züllighoven, H. (1998). *Das objektorientierte Konstruktionshandbuch.* dpunkt Verlag Heidelberg.