Martin Glinz

Requirements Engineering I

Kapitel 1

Grundlagen

Universität Zürich
Institut für Informatik

1.1 Was ist Requirements Engineering?

Erste Näherung: eine technische Definition

Requirements Engineering (Anforderungstechnik) ist das systematische, disziplinierte und quantitativ erfassbare Vorgehen beim Spezifizieren (d.h. Erfassen, Beschreiben und Prüfen) sowie beim Verwalten von Anforderungen an ein System.

- Typisch Softwaresysteme oder softwareintensive Systeme
- Ziel ist eine vollständige, eindeutige, widerspruchsfreie Spezifikation aller Anforderungen
- Typisch als erste Phase eines Projekts
- Riecht nach Papier und Bürokratie
- Wo sind die Menschen in diesem Prozess?
- Ist das Ziel überhaupt realistisch?

Einschub: Der Systembegriff

System (system). 1. Ein Ordnungs- und Gliederungsprinzip. 2. Eine zusammengehörige, von ihrer Umgebung abgrenzbare Menge von Komponenten und Wechselwirkungen, die beobachtete oder postulierte Phänomene der Realität beschreiben. 3. Eine zusammengehörige, von ihrer Umgebung abgrenzbare Menge von Komponenten, die durch koordiniertes Zusammenwirken Leistungen erbringen.

Nicht näher bezeichnete Systeme in der Informatik sind vom Typ 3., wobei die Komponenten zum überwiegenden Teil aus Software bestehen.

- Requirements Engineering in der Informatik: Anforderungen an Software bzw. softwareintensive (technische oder soziotechnische) Systeme
- Auch möglich: Anforderungen an Geräte, Anlagen, Apparate, etc.

Zweite Näherung: kundenorientiert

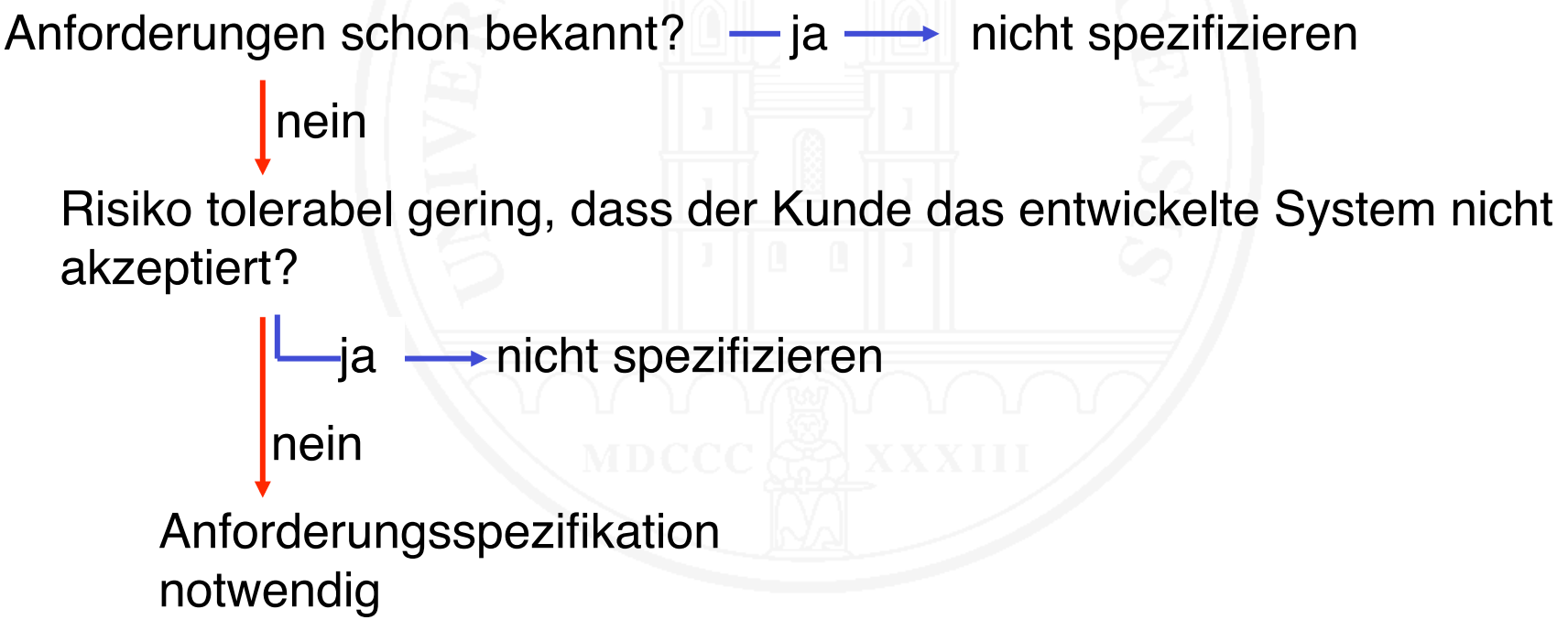
Requirements Engineering (Anforderungstechnik) – Verstehen und Beschreiben, was die Kunden wünschen oder brauchen.

- Eine menschenzentrierte Sicht
- Ziel: zufriedene Kunden

- Was sind Kunden?
- Warum wünschen oder brauchen?
- Warum nicht gleich programmieren?

Dritte Näherung: risikoorientiert

Requirements Engineering (Anforderungstechnik) – Spezifikation und Verwaltung von Anforderungen mit dem Ziel, das Risiko zu minimieren, dass ein System entwickelt wird, welches den Kunden nicht nützt oder gefällt.



Anforderungsspezifikation als Risikominimierung

„Wir haben keine Zeit für eine vollständige Spezifikation.“

„Ist uns zu teuer!“

„Bei agilem Vorgehen genügen grobe Stories vollständig.“

➪ falscher Ansatz

Richtige Frage: „Wie viel müssen wir tun, damit das Risiko eine Größe annimmt, die wir bereit sind zu akzeptieren?“

Merkregel:

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

1.2 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 ein System 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 ein System. Synonyme: Anforderungsdokument, bei Software: Software Requirements Specification.

„Die Spezifikation“: Im Alltag nicht immer eindeutig:

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

Definitionen und grundlegende Begriffe – 2

Requirements Engineering (Anforderungstechnik). 1. Das systematische, *disziplinierte* und *quantitativ erfassbare* Vorgehen beim Spezifizieren, d.h. Erfassen, Beschreiben und Prüfen von Anforderungen an ein System. 2. *Verstehen* und *Beschreiben*, was die Kunden *wünschen* oder *brauchen*. 3. Spezifikation und Verwaltung von Anforderungen mit dem Ziel, das *Risiko* zu minimieren, dass ein System entwickelt wird, welche den Kunden nicht nützt oder gefällt.

Pflichtenheft ➜ verschiedene Begriffe:

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

➜ „Pflichtenheft“ mit Vorsicht verwenden

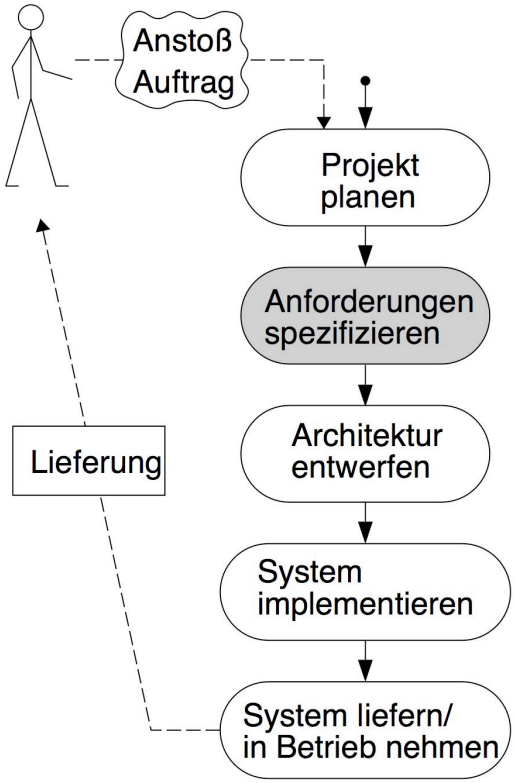
1.3 Aufgaben des Requirements Engineerings

Hauptaufgaben:

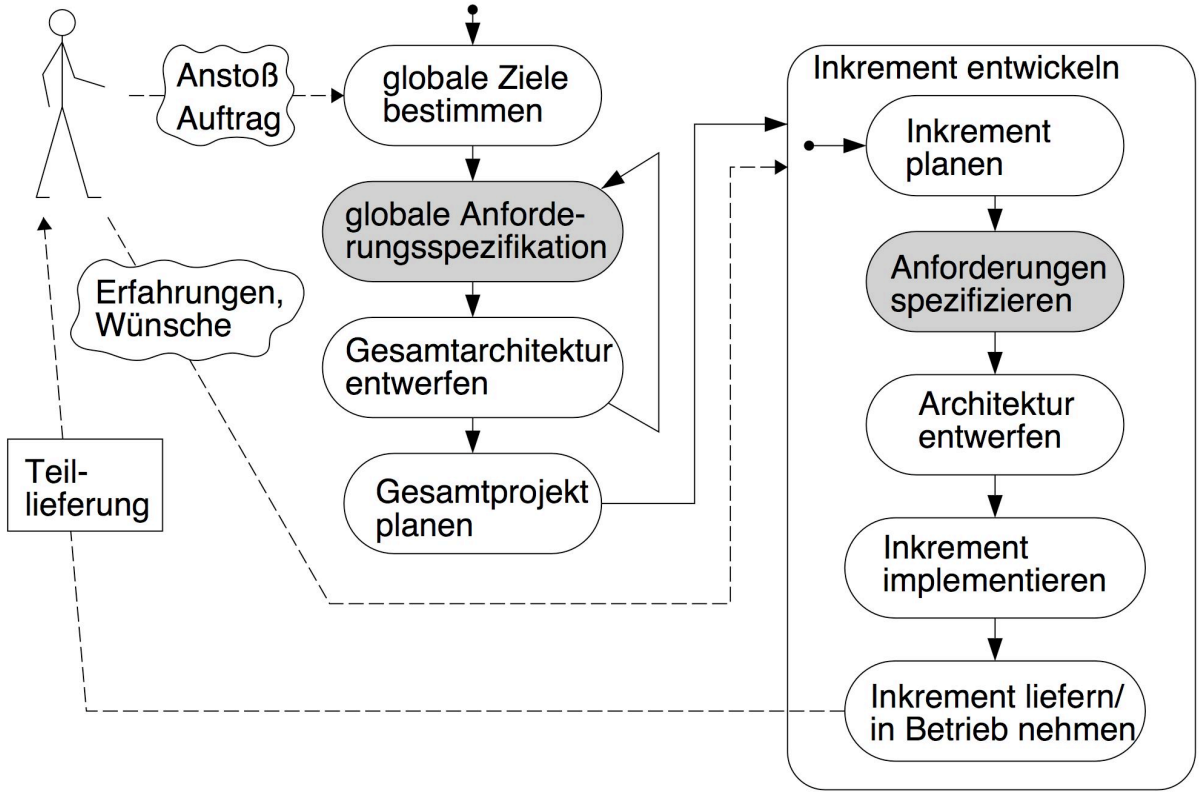
- Anforderungen spezifizieren
  - gewinnen
  - analysieren
  - dokumentieren
  - validieren
- Anforderungen verwalten
  - freigeben
  - ändern
  - rückverfolgen

Einbettung im Software-Lebenslauf

Lineares Modell



Inkrementelles Modell



1.4 Warum Anforderungen spezifizieren?

- Kosten senken
  - Geringere Herstellungskosten (Senken der Fehlerkosten!)
  - Weniger Reklamationen und Nachbesserungen
  - Geringere Pflegekosten
- Risiken verkleinern
  - Kundenerwartungen besser erfüllen
  - Zuverlässigere Prognosen für Termine und Kosten

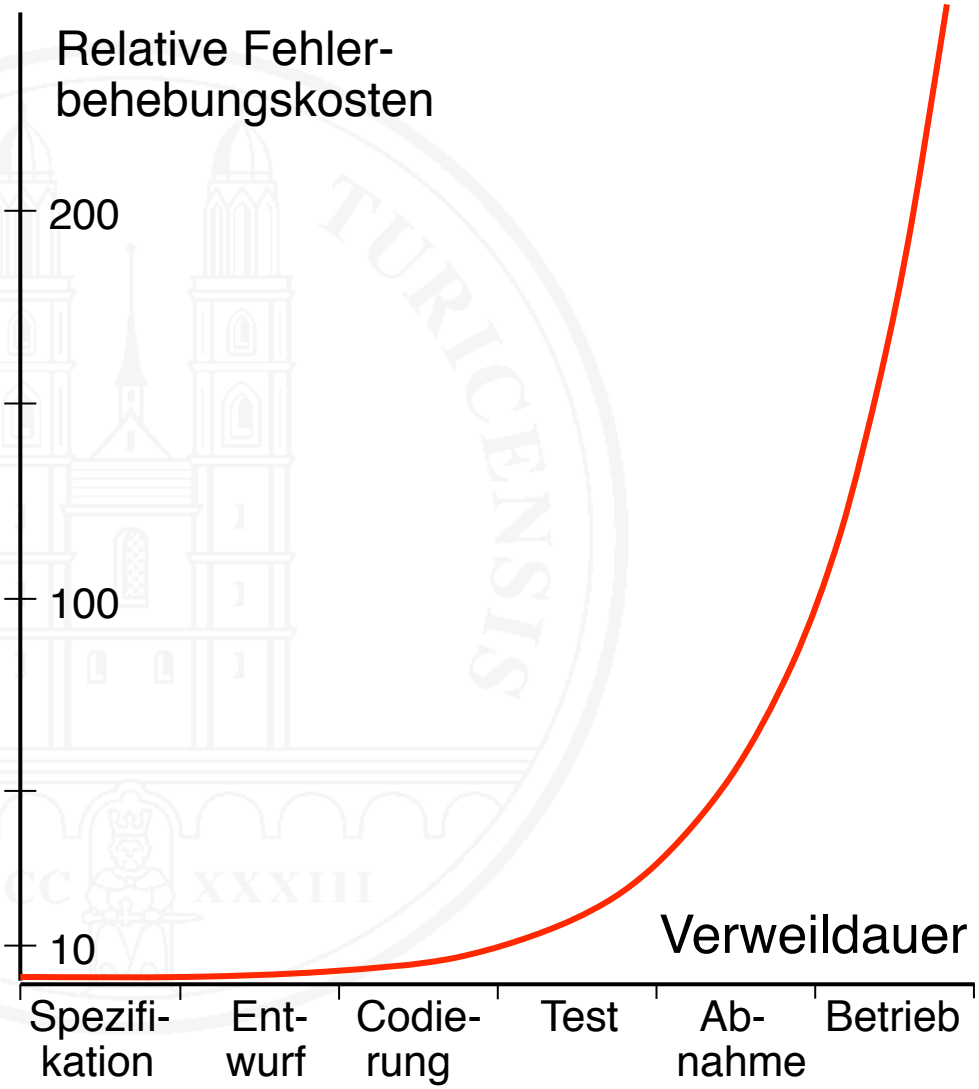
Mehr verdienen!

Zufriedenere Kunden!

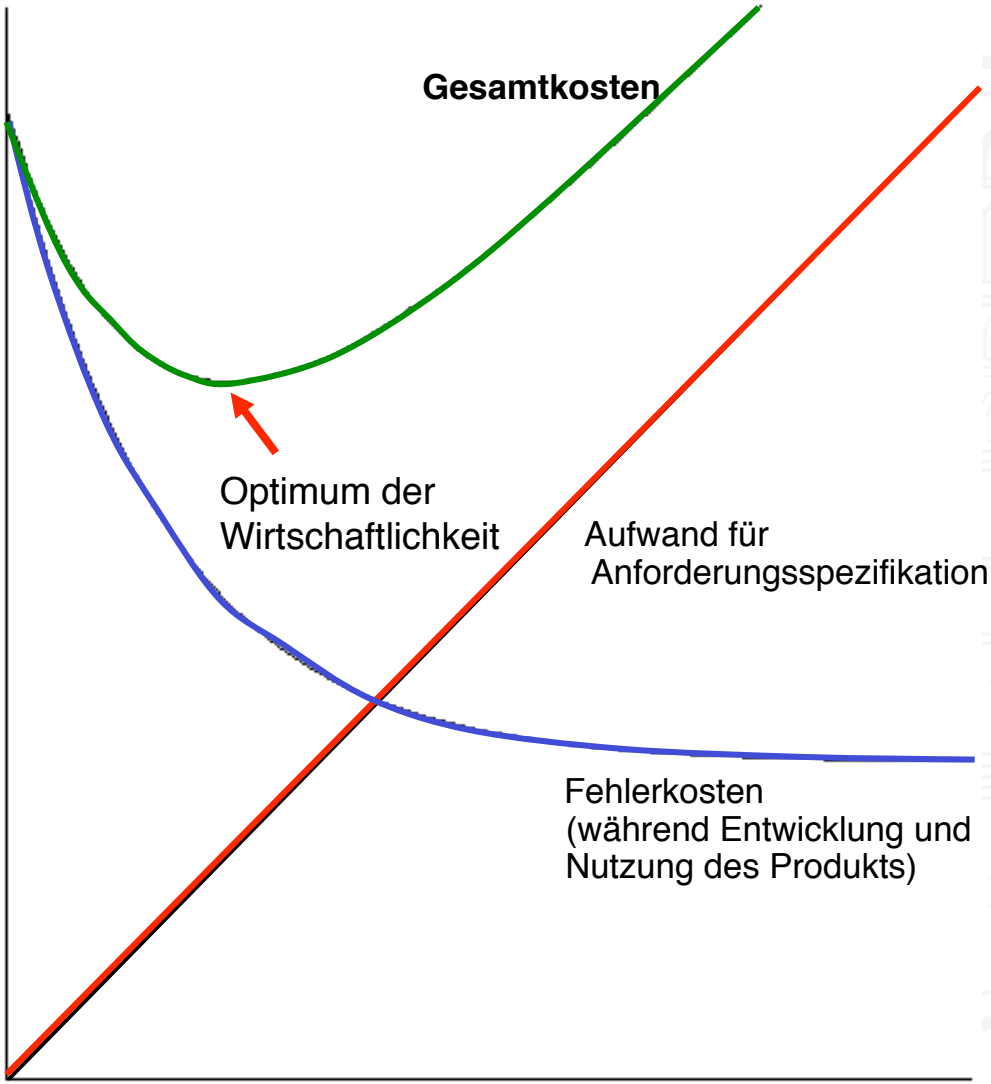
☞ Die wirtschaftliche Wirkung von Requirements Engineering ist immer indirekt; das RE selbst kostet nur!

Anforderungsspezifikation – Warum (2)

Kosten für die Behebung von Fehlern abhängig von ihrer Verweildauer in der Software



Wirtschaftlichkeit der Anforderungsspezifikation



- Wenig Anforderungs-fehler machen
- Möglichst viele der den-noch gemachten Fehler früh finden

1.5 Anforderungen vs. Ziele

Ziel (goal). Ein erwünschter Zustand, den jemand erreichen möchte.

Gemeinsamkeiten: Beide beschreiben etwas zu Erreichendes

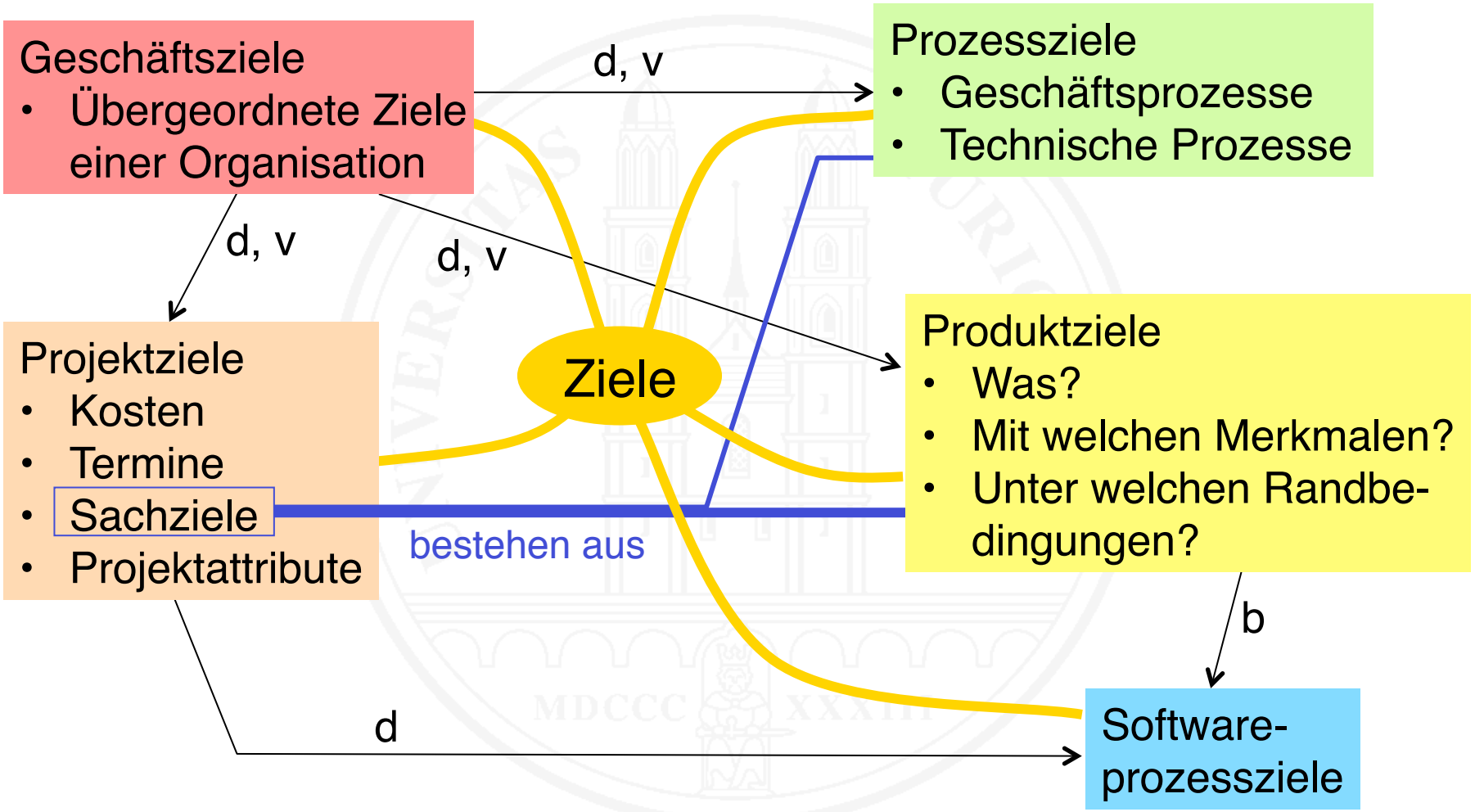
Unterschiede:

- Ziel:
  - ein zu erreichender Zustand
  - meist in umfassendem Sinn gemeint
- Anforderung:
  - eine zu erreichende Eigenschaft
  - meist konkret
  - beschreibt in der Regel eine für die Erreichung eines Ziels notwendige Bedingung oder Eigenschaft

Anforderungen vs. Ziele – 2

- Abgrenzung im Einzelfall oft schwierig
- Häufig geben Ziele auf einer Ebene (z.B. Geschäftsebene) den Rahmen für Anforderungen auf einer tiefer liegenden Ebene (z.B. System- oder Softwareebene) vor
- Im Software Engineering sind die Sachziele eines Projekts typisch beschrieben durch die Anforderungen an das zu entwickelnde (Software-)Produkt und die davon betroffenen Prozesse

Arten von Zielen und ihre Zusammenhänge



b: beeinflusst d: determiniert v: wird verfeinert in

Aufgabe 1.1: Ziele und Anforderungen

Lesen Sie folgende Auszüge aus der Zielbeschreibung* für die deutsche elektronische Gesundheitskarte und identifizieren Sie Geschäftsziele, Produktziele (Anforderungen), Projektziele und Prozessziele.

«Ab 2006 wird die elektronische Gesundheitskarte die bisherige Krankenversichertenkarte schrittweise ersetzen. Das bedeutet, dass nicht von Beginn an alle Funktionen zur Verfügung stehen, sondern mit zunehmender Leistungsfähigkeit des Systems realisiert und nachgeladen werden.

...

Nach der Einführung werden zunächst die **administrativen Funktionen** (für alle Versicherten verpflichtend) realisiert. Den Anfang machen die Daten zur Beschreibung des Versicherungsverhältnisses. Sie ... können künftig in einem Online-Verfahren beim Arztbesuch aktualisiert werden. Ebenfalls zum administrativen Teil der elektronischen Gesundheitskarte zählt das **elektronische Rezept**, das das Papierrezept ablösen wird.

...

*http://www.dimdi.de/static/de/ehealth/karte/basisinformation/ziele besucht am 24.11.05; nicht mehr online

Aufgabe 1.1: Ziele und Anforderungen – 2

Unter Wahrung der **Datenhoheit der Patientinnen und Patienten** und **Stärkung der Patientenselbstbestimmung** soll die Karte auf diese Weise dazu beitragen, die **Qualität der medizinischen Versorgung** von Patientinnen und Patienten zu verbessern.

...

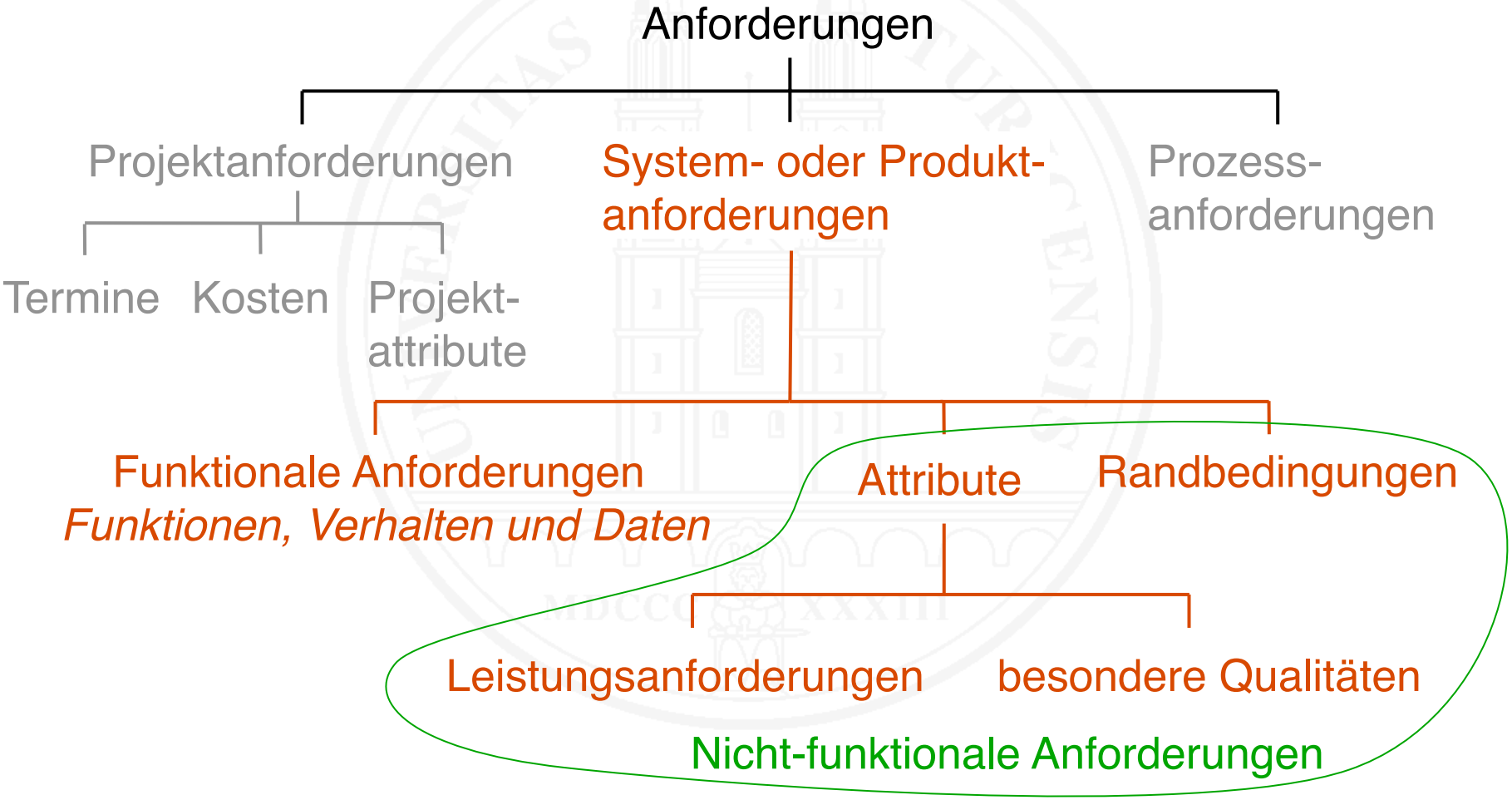
Schließlich sollen Gesundheitskarte und elektronische Datennetze auch dabei helfen, Kosten einzusparen, die im Gesundheitswesen entstehen, weil Verwaltungsvorgänge durch die gängigen Mischlösungen aus elektronischer Dokumentation und Papierdokumentation unnötig komplex werden.

...

Die durch die Einführung der elektronischen Gesundheitskarte erwarteten **Einsparungen** ergeben sich insbesondere durch Erleichterungen bei der verwaltungstechnischen Abwicklung der Rezepte, durch Verminderung behandlungsbedürftiger Wechsel- und Nebenwirkungen von Arzneimitteln, durch die Verringerung von Doppelbehandlungen und die schnellere Verfügbarkeit von Notfall- und sonstigen Behandlungsdaten.

1.4 Klassifikation von Anforderungen

Klassifikation nach der Art der Anforderungen

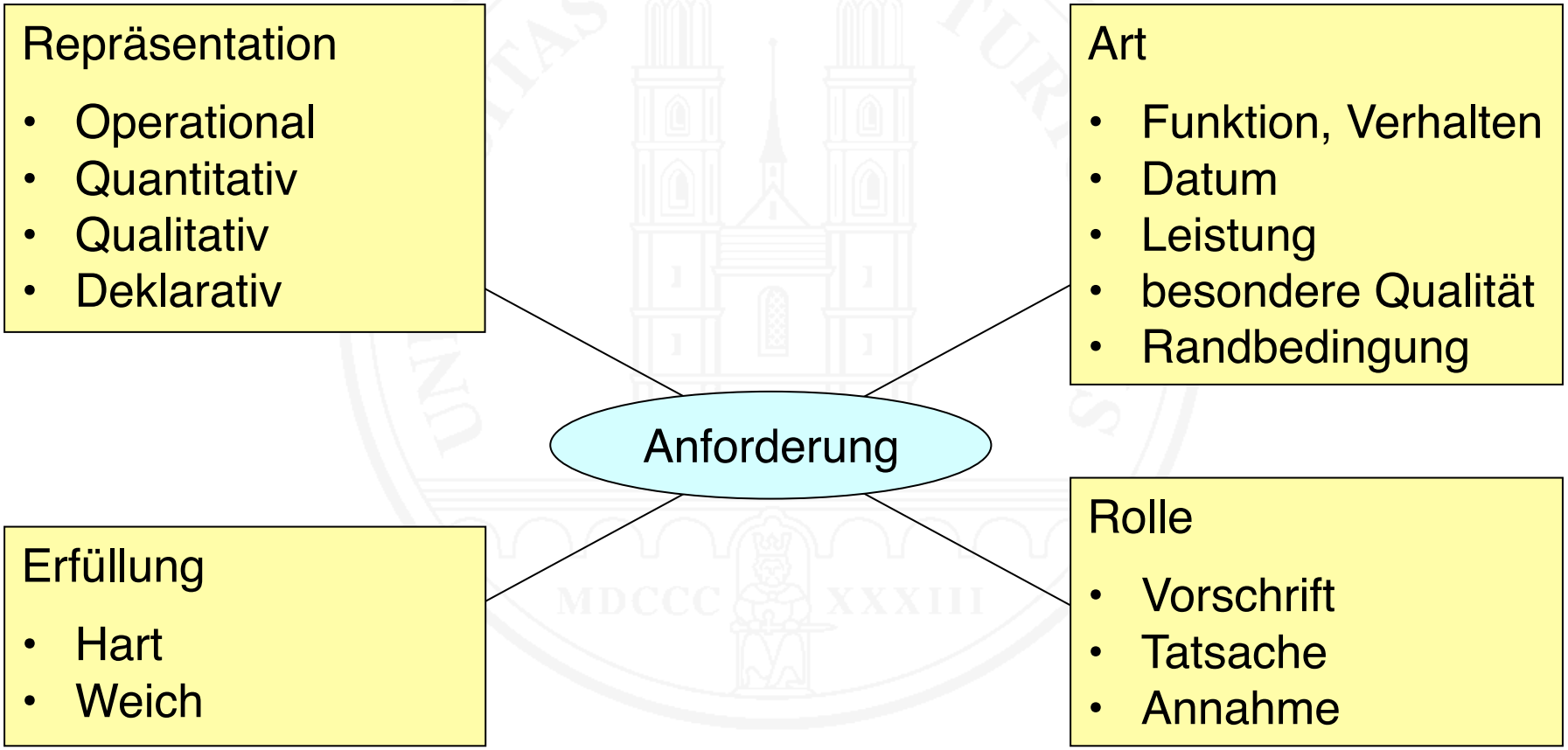


Probleme

- Nicht-funktionale Anforderungen: ein vielfach unscharf gefasster und uneinheitlich verwendeter Begriff
- Verbreitete *falsche / unzutreffende* Auffassungen:
  - Nicht funktional = global
  - Nicht funktional = weich
  - Was das System tun soll → funktionale Anforderungen / wie das System dies tun soll → nicht-funktionale Anforderungen
  - Hauptanforderungen → funktionale Anforderungen / ergänzende Anforderungen → nicht-funktionale Anforderungen
  - Operational dargestellt → funktionale Anforderungen / qualitativ oder quantitativ dargestellt → nicht-funktionale Anforderungen
- Vermeidbar mit Orientierung an der zu Grunde liegenden Intention

Facettierte Klassifikation von Anforderungen

Anforderungen unterscheiden sich nicht nur nach ihrer Art:



Erläuterungen zu den Facetten

Repräsentation

- Operational: „Der Kontostand wird um den eingezahlten Betrag erhöht“
- Quantitativ: „Antwortzeit < 0,5 s“
- Qualitativ: „Das System muss eine hohe Verfügbarkeit aufweisen“
- Deklarativ: „Das System soll auf einer Linux-Plattform laufen“

Erfüllung

- Hart – Anforderung ist ganz oder gar nicht erfüllt (binäres Verhalten): „Das System soll abgelaufene Wertkarten sperren“
- Weich – Anforderung kann graduell erfüllt sein: „Das System soll für Gelegenheitsbenutzer einfach zu bedienen sein“

Erläuterungen zu den Facetten – 2

Art

- siehe Darstellungsaspekte in Kapitel 2

Rolle

- Vorschrift – Forderung an das zu erstellende System: „Der Füllstandsensor soll alle 100 ms einmal abgetastet werden“
- Tatsache – Fakten in der Systemumgebung : „Alleinstehende werden nach Tarif A besteuert“
- Annahme – Annahmen über das Verhalten von Akteuren in der Systemumgebung: „Jeder Alarm wird vom Operateur quittiert“

1.7 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

Wertbasierte Priorisierung

- Nutzen der Realisierung einer Anforderung

  vs. Kosten für die Entwicklung (Denne und Cleland-Huang, 2004)

- Kosten und Nutzen realistisch rechnen:
  - kumulierter Nutzen über die gesamte erwartete Lebensdauer
  - Kapitalkosten (einschl. Verzinsung) über die Entwicklungsdauer und die erwartete Lebensdauer

- Insbesondere zur Priorisierung in Situationen, wo eine Vielzahl wichtiger, aber nicht unverzichtbarer Funktionen und Merkmale eines Systems gefordert wird

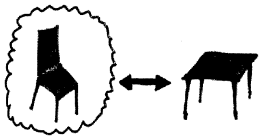
Aufgabe 1.2: Wertbasierte Priorisierung

- Die Realisierung von Anforderung A kostet Fr. 100'000, dauert 6 Monate und bringt einen Nutzen von Fr. 25'000 pro Jahr.
- Die Realisierung von Anforderung B kostet Fr. 200'000, dauert 12 Monate und bringt einen Nutzen von Fr. 54'000 pro Jahr.

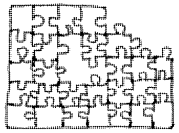
Welche Anforderung realisieren Sie zuerst, wenn Sie einen Zeithorizont von 5 Jahren betrachten?

1.8 Qualitätsmerkmale

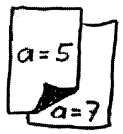
Merkmale einer guten Spezifikation



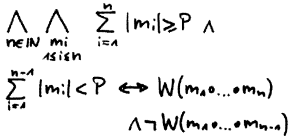
Adäquat – beschreibt das, was der Kunde will bzw. braucht



Vollständig – beschreibt alles, was der Kunde will bzw. braucht



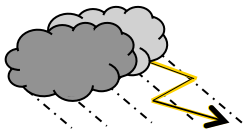
Widerspruchsfrei – sonst ist die Spezifikation nicht realisierbar



Verständlich – für alle Beteiligten, Kunden wie Informatiker

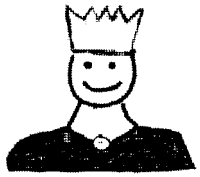
Eindeutig – vermeidet Fehler durch Fehlinterpretationen

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



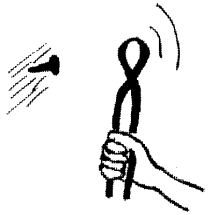
Risikogerecht – Umfang umgekehrt proportional zum Risiko, das man eingehen will

Merkmale eines guten Spezifikationsprozesses

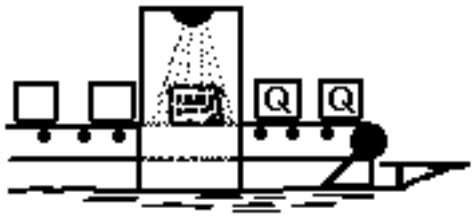


Kundenorientierung

Methodisches und zielgerichtetes Vorgehen



Verwendung geeigneter Mittel



Integration von Erstellung und Prüfung von Anforderungen

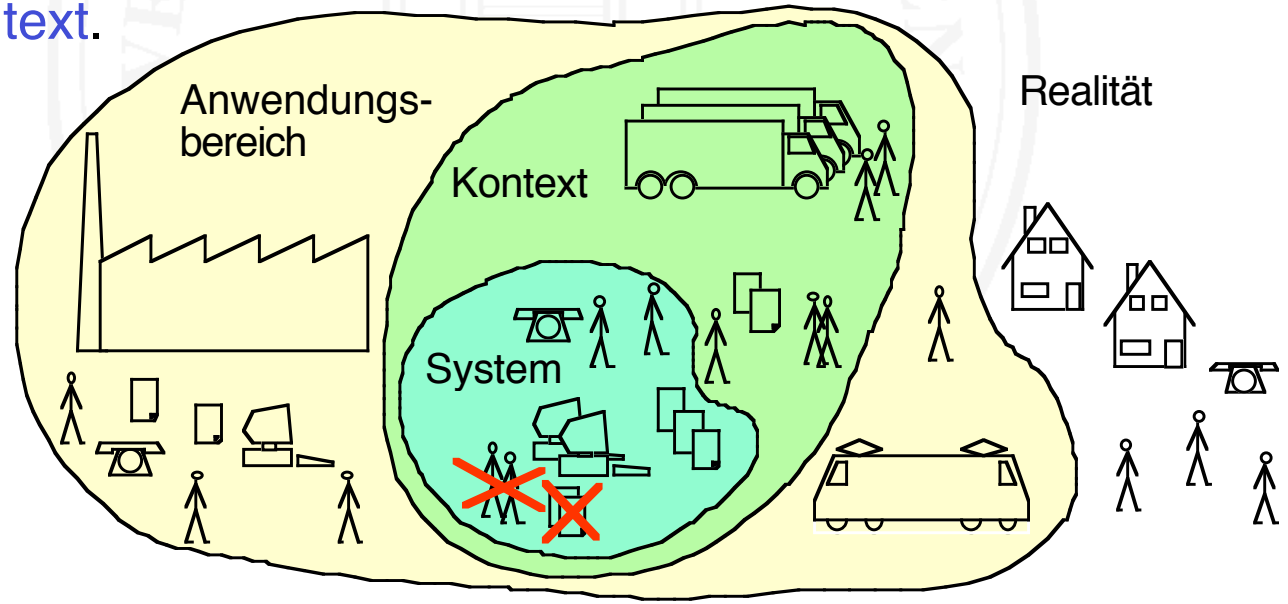
Aufgabe 1.3: Qualität von Anforderungen

Beurteilen Sie die Eindeutigkeit folgender Anforderung:

„Das System soll die Zahl der Tage zwischen zwei gegebenen Kalendertagen berechnen.“

1.9 Anforderungen und ihr Kontext

- Systeme sind in der Regel nicht isoliert, sondern eingebettet in eine Umgebung
- In dieser Umgebung gibt es Akteure, die für das zu spezifizierende System relevant sind
- Die Akteure im unmittelbaren Umfeld eines Systems bilden den Systemkontext.



Terminologie

Kontext (context). 1. Der umgebende Text einer sprachlichen Einheit. 2. Der Gedanken- und Sinnzusammenhang, in dem ein Phänomen oder eine Äußerung verstanden werden muss. 3. (in der Informatik) Diejenigen Komponenten des Anwendungsbereichs, welche mit einem System interagieren, aber selbst nicht Bestandteil des Systems sind.

Anwendungsbereich (application domain). Derjenige Ausschnitt der Realität, welcher im Zusammenhang mit einem in diesem Bereich eingesetzten System relevant ist.

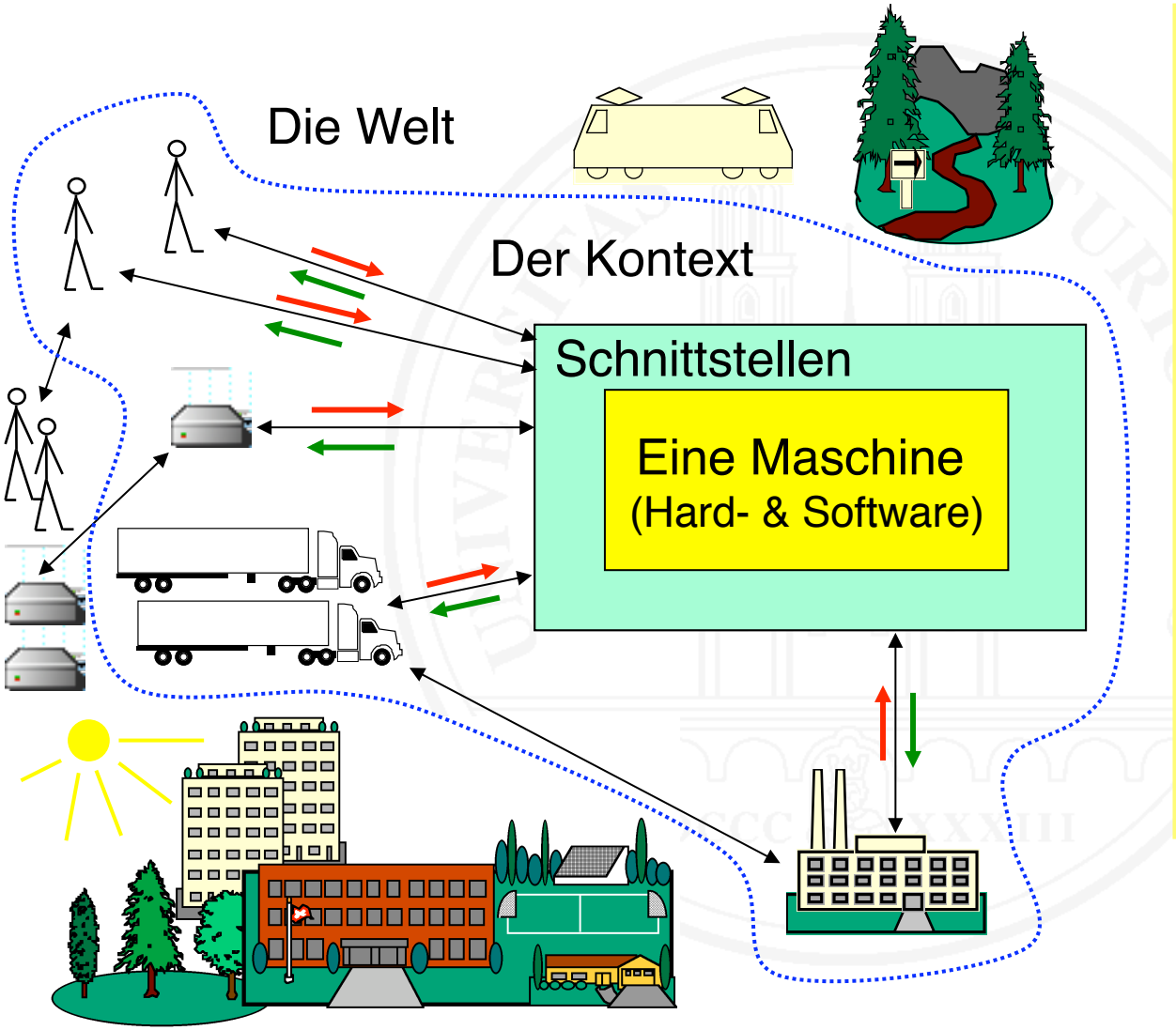
Realität (reality). Die Wirklichkeit: Alle beobachtbaren oder sonstwie gedanklich fassbaren Dinge dieser Welt.

Akteur (actor). Ein Mensch, der Ziele hat und zu deren Erreichung handelt oder ein Element des Anwendungsbereichs, das zur Erreichung bestimmter Ziele dient und hierzu handeln und/oder Informationen verarbeiten kann.

Von Akteuren und Zielen zu Anforderungen

- Zu Beginn interessiert man sich im Requirements Engineering für die Akteure im Umfeld eines geplanten Systems, deren Ziele, sowie ihre Handlungen und Interaktionen zur Erreichung dieser Ziele.
- Daraus wird der Systemkontext gewonnen, d.h. diejenigen Akteure, welche
  - Aufgaben an das zu erstellende System delegieren
  - dem zu erstellenden System Informationen liefern, die es zur Erfüllung seiner Aufgaben benötigt
  - benötigte Informationen vom zu erstellenden System erhalten
- Die eigentlichen Anforderungen spezifizieren, welche Dienstleistungen das zu erstellende System für die Akteure in seinem Kontext erbringen soll, welche Informationen es von diesen Akteuren zu diesem Zweck benötigt und unter welchen Bedingungen dies geschieht

Anforderungen spezifizieren



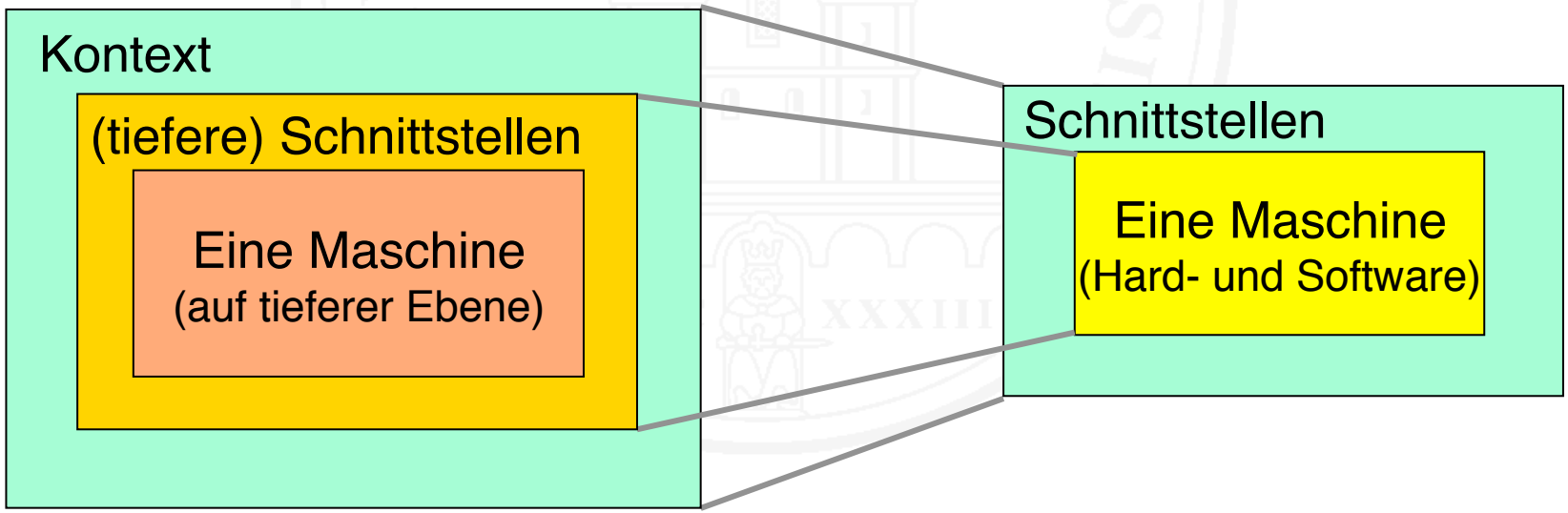
Anforderungsspezifikation bedeutet:

- Kontext identifizieren
- Anstöße spezifizieren...
- und Antworten darauf
- sowie Restriktionen (Leistung, Qualität, Randbedingungen)

Problem: mehr als eine Betrachtungsebene

- Mehrere Betrachtungsebenen mit unterschiedlichem Kontext, typisch
  - Geschäftsebene
  - Systemebene
  - Softwareebene
  - Oft weitere Ebenen, z. B. Komponentenebene

➪ Verzahnung von Anforderungen und Entwurf



Betrachtungsebenen – 2

Beispiel:

- ☆ 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.»

- ❍ Verzahnung von Anforderungen und Lösungen ist unausweichlich
- ❍ Kontextbestimmung und -abgrenzung ist wichtig

1.10 Spezifikation vs. Entwurf

Volksweisheit: WAS = Spezifikation, WIE = Entwurf

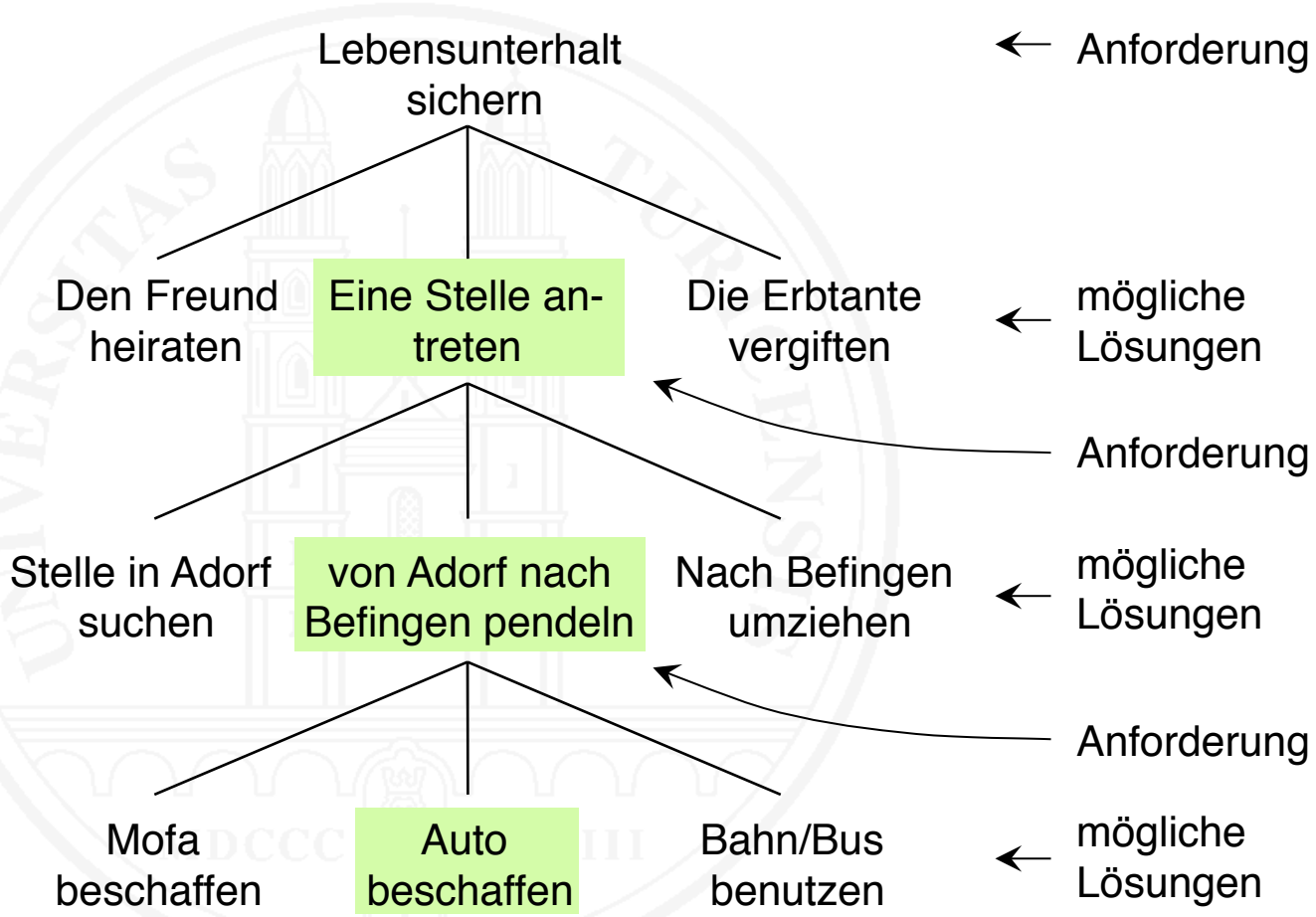
Aber: ist folgender Satz eine Anforderung oder eine Entwurfsentscheidung?

> „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.

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.



Formen der Verzahnung

- 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
- Rückwirkung: Erkenntnisse und Schwierigkeiten bei der Lösung können Änderungen in den Anforderungen bewirken.

Abgrenzung Anforderungen – Entwurf

- WAS vs. WIE funktioniert nicht, da abhängig von Betrachtungsebene
- Im Prozess sind Anforderungen und Entwurf nicht vollständig trennbar
- In der Dokumentation ist eine Trennung angezeigt

- Möglichkeit: operationale Abgrenzung:
  - Änderungen der Anforderungen brauchen die Zustimmung des Auftraggebers/Kunden
  - Änderungen im Entwurf kann der Auftragnehmer/Lieferant autonom vornehmen

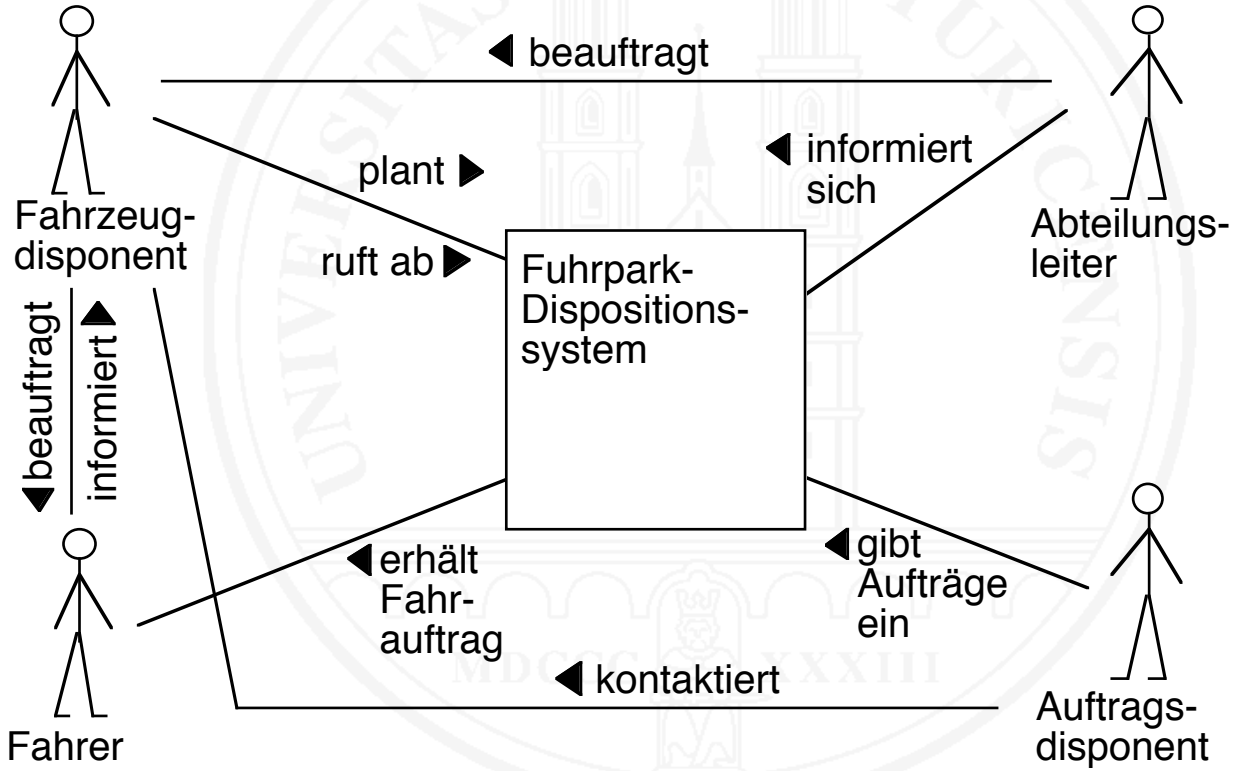
1.11 Kontextmodelle

Modell eines Systems in seinem Kontext

- Betrachtungsebene festlegen
- In der Regel keine Systeminterna (➜ System als schwarzer Kasten)
- Akteure modellieren, welche direkt mit dem System interagieren
- Interaktion zwischen System und diesen Akteuren modellieren
- Interaktion von Akteuren untereinander modellieren
- Ergebnis grafisch darstellen

Beispiel

Kontextdiagramm eines Fuhrpark-Dispositionssystems



Aufgabe 1.4: Kontextmodell

Gegeben sei die Fallstudie über die Erneuerung eines Bibliotheks-systems (siehe separate Datei).

Bestimmen Sie die Akteure im Kontext dieses Systems und erstellen Sie ein Kontextdiagramm.

Geben Sie an, auf welcher Betrachtungsebene dieses Kontextdiagramm liegt.

Literatur

Allgemeine Literatur

Davis, A. (2005). *Just Enough Requirements Management*. New York: Dorset House.

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

IEEE (1998). *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Standard 830-1998.

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

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

Pohl, K. (2007). *Requirements Engineering: Grundlagen, Prinzipien,Techniken*. Heidelberg: dpunkt.

Pohl, K., Rupp, C. (2009). *Basiswissen Requirements Engineering.* Heidelberg: dpunkt.

Robertson, S., Robertson, J. (2006). *Mastering the Requirements Process*. 2nd edition, Addison-Wesley.

Literatur – 2

Literatur zu Kapitel 1

Denne, M., J. Cleland-Huang (2004). *Software by Numbers: Low-Risk, High-Return Development*. Sun Microsystems Press, Upper Saddle River, N.J.: Prentice Hall.

Glinz, M. (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich.

Glinz, M. (2005). Rethinking the Notion of Non-Functional Requirements. *Proceedings of the Third World Congress for Software Quality (3WCSQ 2005)*, München, Vol. II, 55-64.

Glinz, M. (2007). On Non-Functional Requirements. *Proceedings of the 15th IEEE International Requirements Engineering Conference*, Delhi, India. 21-26.

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

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

Swartout, W., R. Balzer (1982). On the Inevitable Intertwining of Specification and Implementation. *Communications of the ACM* **25**, 7 (Jul 1982). 438-440.