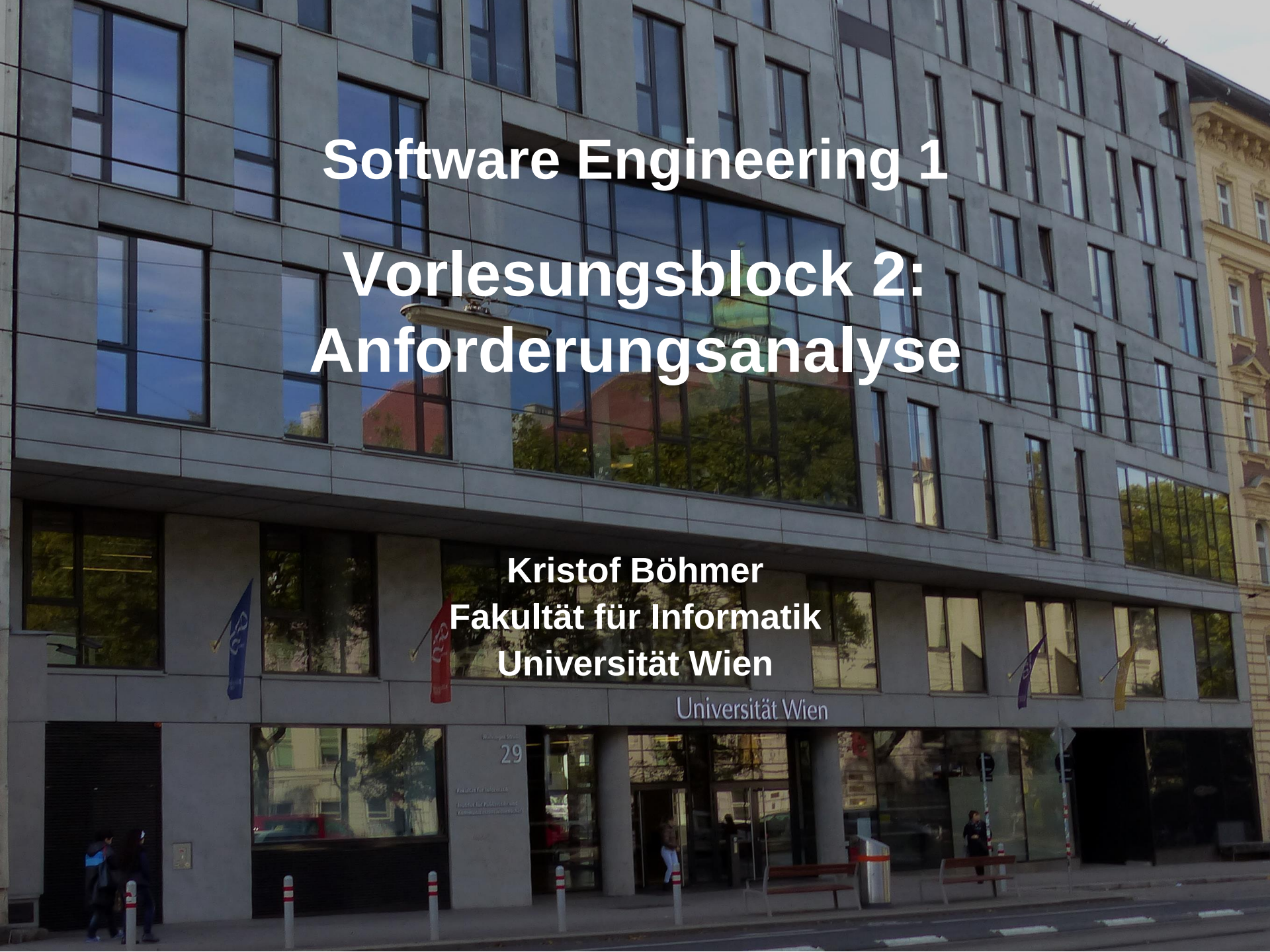


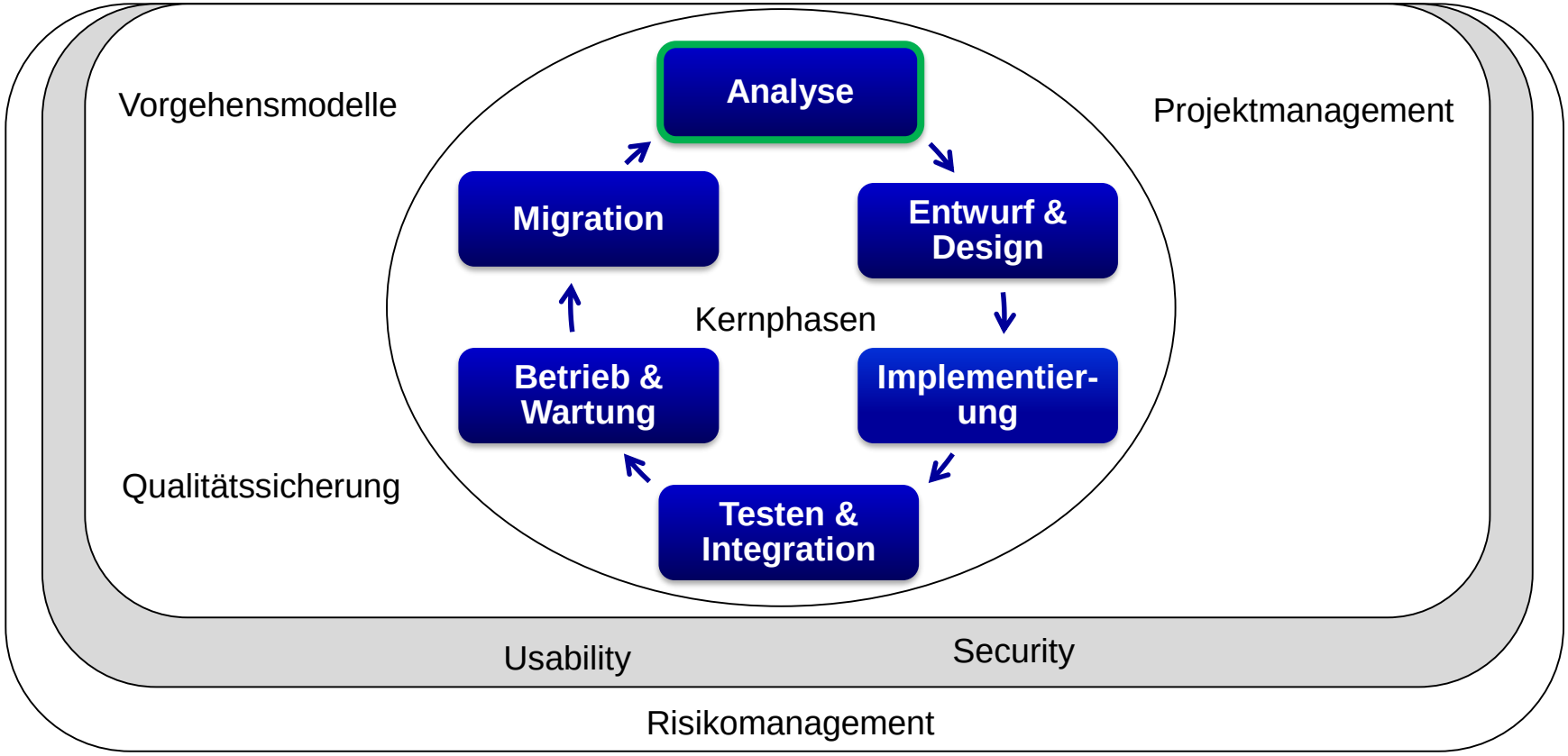
Software Engineering 1

Vorlesungsblock 2: Anforderungsanalyse

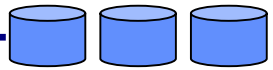
Kristof Böhmer
Fakultät für Informatik
Universität Wien

Zuordnung des Foliensatzes

- **Projektphasen:** Zuordnung des zweiten Vorlesungsblocks – Analyse
  - Diese Phase beschäftigt sich mit der Erhebung, Modellierung, Dokumentation und Validation von Anforderungen.



Inhalt



2.1 Motivation: Lernziele

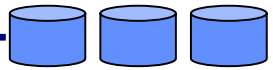
- **Mit diesem Foliensatz möchten wir folgende Lernziele vermitteln:**
  - Was ist die Rolle der Stakeholder in einem Softwareprojekt?
  - Wie setzt man eine Machbarkeitsstudie zur Prüfung der Projektmachbarkeit ein?
  - Soll die Software (oder Teile davon) selbst entwickelt oder zugekauft werden (Make or Buy Entscheidung)?
  - Was sind Anforderungen, welche Anforderungstypen gibt es und welche Eigenheiten weisen „gute“ Anforderungen auf?
  - Welche Schritte müssen durchlaufen werden um zu einer fertig ausgearbeiteten Anforderung zu gelangen?

Block 2: Die nachfolgenden Folien basieren auf
- **T. Grechenig, M. Bernhart, R. Breiteneder, K. Kappel: Softwaretechnik. Pearson Studium 2010**
- **A. Schatten, M. Demolsky, D. Winkler, S. Biffl, E. Gostischa-Franta, T. Östreicher: Best Practice Software-Engineering. Spektrum 2010**
- **I. Sommerville: Software Engineering. Pearson 2016**

2.1 Motivation: Beispiel zur Anforderungsqualität 1

- **Beispiel aus dem Straßenverkehr**
  - Das Verkehrszeichen für die Kurzparkzone wurden mit einem Zusatzschild ergänzt um diese auf Mo-Fr auszuweiten.
  - Dadurch wird die maximale Parkdauer für alle Kurzparktage (**inkl**. Samstag) auf 2 Stunden erhöht.
  - Das alte Schild weist für Samstag aber noch eine Parkdauer von 1,5 Stunden auf.
  - Dies ergibt eine Inkonsistenz!
- **Welche Analogien lassen sich daraus für Anforderungen ableiten?**

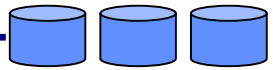
2.1 Motivation: Beispiel zur Anforderungsqualität 2



- **Es lassen sich daraus folgende Analogien für Anforderungen ableiten:**
  - Die beschriebenen Anforderungen können zueinander inkonsistent sein.
  - Es existieren relevante aber trotzdem „unsichtbare“ Vorgaben wie beispielsweise Gesetzestexte oder Eigenheiten der Projektdomäne.
  - Die „richtige“ Interpretation aller Informationsquellen ist teilweise nur mit Wissen über zusätzliche Vorgaben (Hintergrundwissen) möglich.
    - Nur damit lässt sich die im Beispiel angeführte Inkonsistenz auflösen.
  - Es besteht die Gefahr, dass unterschiedliche Projektbeteiligte während unterschiedlicher Projektphasen das Projekt und dessen Anforderungen unterschiedliche interpretieren.
  - Einmal festgelegte Interpretationen werden oftmals nicht gesondert dokumentiert und deshalb oft nicht berücksichtigt.
- **Einfache Lösung: Altes Schild entfernen und durch ein neues ersetzen.**

- **Wofür brauchen wir Anforderungen überhaupt?**
  - Durch Anforderungen wird **ein gemeinsames Verständnis** hergestellt, was das Produkt können soll.
  - Anforderungen repräsentieren die **„reale Welt“** und drücken das gewünschte Verhalten aus Nutzersicht aus.
  - Berücksichtigung der unterschiedlichen Interessen und Erwartungshaltungen von unterschiedlichen Stakeholdern (Anwender, Entwickler etc.) und deren individuellen Blickwinkeln auf das Projekt.

2.1 Motivation: Warum sind Anforderungen wichtig?



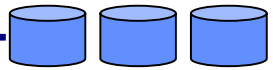
- **Chaos Report:** Dokumentation von Projektfehlschlagursachen.
  - Die häufigsten Gründe betreffen den Problembereich der Anforderungen.

Fehlschlagursachen	Prozent
1. Incomplete Requirements	13.1%
2. Lack of User Involvement	12.4%
3. Lack of Resources	10.6%
4. Unrealistic Expectations	9.9%
5. Lack of Executive Support	9.3%
6. Changing Requirements & Specifications	8.7%
7. Lack of Planning	8.1%
8. Didnt Need It Any Longer	7.5%
9. Lack of IT Management	6.2%
10. Technology Illiteracy	4.3%
Other	9.9%

⇨ Um diese Fehlschlagursachen zu vermeiden, müssen wir herausfinden, was der Kunde will bzw. benötigt!

⇨ Hierfür kann die Anforderungs-erhebung eingesetzt werden.

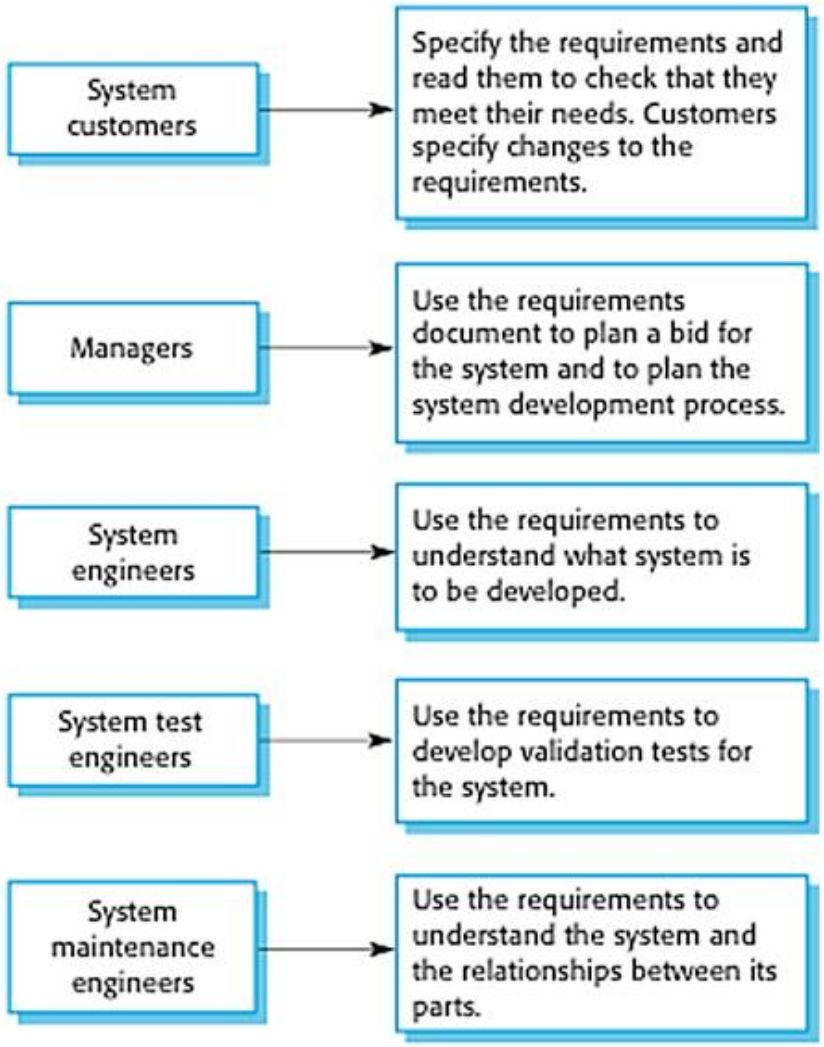
2.1 Motivation: Warum sind Anforderungen wichtig?



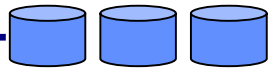
- **Was ist das Ergebnis keiner oder einer ungenügenden Anforderungserhebung bzw. Anforderungsdokumentation?**
  - **Für den Auftraggeber:**
    - Es entsteht eine Diskrepanz zwischen erwarteten und tatsächlichen Projektergebnis.
    - Nachträgliche Änderungen werden gefordert aber nicht bezahlt werden.
    - Der Auftraggeber wird dazu angeregt Vertragsdetails nach wechselnden Vorlieben und Wünschen jeweils unterschiedlich auszulegen. Die dadurch entstehenden Mehrkosten werden an den Auftragnehmer abgeschoben.
    - Die Qualität der Projektergebnisse ist deutlich schlechter als diese sein müssten.
  - **Für den Auftragnehmer:**
    - Die Projektkosten können nur ungenügend abgeschätzt und überwacht werden. Das Projekt ist deshalb nicht mehr kostendeckend.
    - Die Softwarearchitektur wird ungenügend ausgelegt/durchdacht und kann deshalb nur schwierig an die realen Kundenwünsche und zukünftige Entwicklungen angepasst werden.
    - Die Entwicklung dauert länger und ist deutlich aufwändiger als geplant.
    - Entwickler müssen Überstunden leisten um das Projekt noch irgendwie zu retten.
- **Endergebnis:** Unzufriedene Auftraggeber und Auftragnehmer, oft führt dies auch zu Rechtsstreitigkeiten.

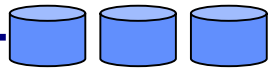
2.1 Motivation: Wer interessiert sich für das Ergebnis der Anforderungserhebung?

- **Unterschiedliche Personengruppen verwenden verschiedene Anforderungen um unterschiedliche Ziele zu erreichen und Interessen zu befriedigen.**
- **Nahezu jede an der Entwicklung beteiligte Person ist von Anforderungen betroffen aber auch davon abhängig.**



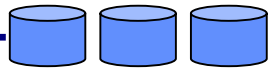
Inhalt





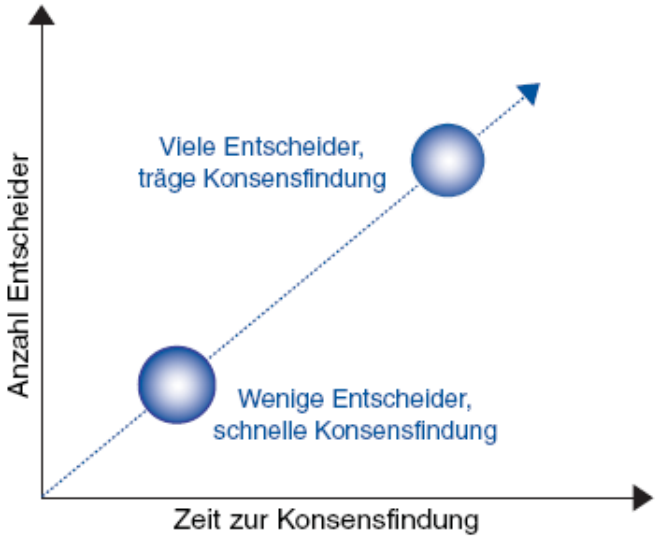
- **Nach dem IEEE-Standard 29148 ist die Anforderungsanalyse (Requirements Engineering) folgendermaßen definiert:**
  - *„Requirements engineering is an interdisciplinary function that mediates between the domains of the acquirer and supplier to establish and maintain the requirements to be met by the system, software or service of interest.*
  - *Requirements engineering is concerned with discovering, eliciting, developing, analyzing, determining verification methods, validating, communicating, documenting, and managing requirements.*
  - *The result of requirements engineering is a hierarchy of requirements that:*
    - *Enables an agreed understanding between stakeholders (e.g., acquirers, users, customers, operators, suppliers)*
    - *Is validated against real-world needs, can be implemented*
    - *Provides a basis of verifying designs and accepting solutions.“*

- **Grundlagen der Anforderungsanalyse**
  - **Vision:** Welchen Zweck und welchen Nutzen soll das Ergebnis des Projektes primär erfüllen.
    - Üblicherweise wird die Vision nur relativ allgemein und grob definiert. Diese ändert sich daher nur wenig.
    - Darüber hinaus ermöglicht diese es einen Business Case abzuleiten um die Wirtschaftlichkeit des Systems zu verdeutlichen.
    - Vision und der Business Case dienen als Entscheidungsgrundlage, ob ein Softwareprojekt überhaupt beauftragt wird**.**
    - **Beispiel für eine Vision:** Our organisation will respect the law.
  - **Scope:** Definiert den Umfang den das System abdecken soll.
    - Üblicherweise veränderbar.
    - Es ist erforderlich die Nicht-Ziele explizit zu dokumentieren.
    - Der Scope hat eine direkte Auswirkung auf die Projektkosten.
    - **Beispiel für einen Scope:** Our organisation will become GDPR compliant.



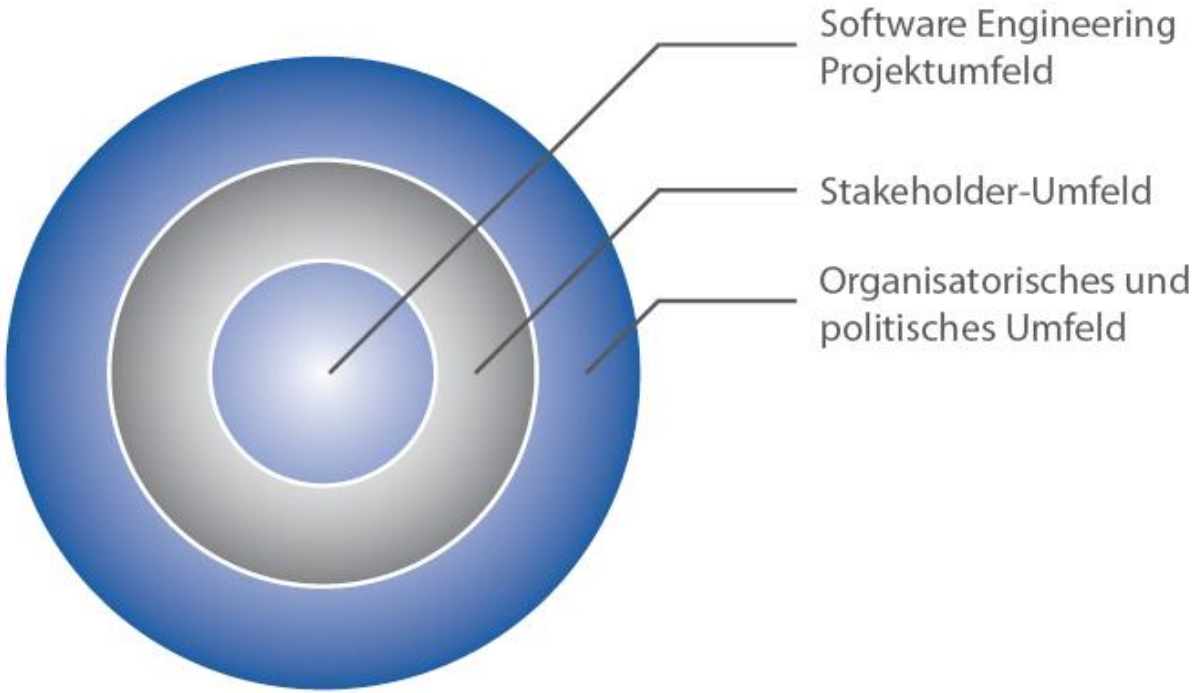
- **Was sind Stakeholder?**
  - Stakeholder sind **alle am Projekt beteiligten Personen**, die ein Interesse an dem jeweiligen Softwareprojekt haben.
  - Stakeholder stellen die wichtigste Anforderungsquelle dar.
  - Stakeholder beeinflussen ein Softwareprojekt.
  - Ein Softwareprojekt hat grundsätzliche nicht nur einen sondern mehrere Stakeholder mit sehr unterschiedlichen Interessen.
  - Viele Stakeholder sind auch keine Techniker und verstehen den typischen Technikerlingo und technische Systembeschreibungen nicht.
  - Jede Stakeholder-Gruppe hat ihre eigene Sicht auf die Anforderungen.
    - Stakeholder-Gruppe: Eine Gruppe von Stakeholdern mit gleichem Interesse.
- **Das heißt: Ein Stakeholder ist jede Person, die ein Projekt beeinflussen kann oder durch die das Projekt beeinflusst wird.**
  - Kann auch eine Institution/Unternehmen sein.

- **Typische Stakeholder in einem Softwareprojekt sind?**
  - Kunde/Auftraggeber/Sponsor
  - Geschäftsführung/Abteilungsleiter (Auftraggeber- und Auftragnehmerseite)
  - Legacy Owner (Besitzer des Altsystems oder der Altdaten)
  - Benutzer/Benutzervertreter (pro Benutzergruppe)
  - Betreiber/Entwickler von Schnittstellensystemen (Systemen, mit denen integriert wird)
  - Projektleiter
  - Projektsteuergruppe
  - Systemarchitekten/IT-Strategie
  - Softwareentwickler
  - Qualitätssicherung und Test
  - Betrieb und Wartung

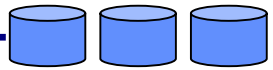


- **Je mehr Stakeholder an einem Projekt beteiligt sind, umso schwieriger wird es einen Konsens zu finden.**

- **Projektumfeld:** Stakeholder beeinflussen ein Softwareprojekt, diese Stakeholder werden aber wiederrum vom organisatorischen und politischen Umfeld beeinflusst.

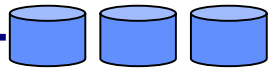


2.2 Grundlagen: Grundlagen der Anforderungsanalyse – Machbarkeitsstudie



- **Machbarkeit:** Bei jedem Softwareprojekt muss analysiert werden, ob das Vorhaben überhaupt umsetzbar ist.
  - Ist ein Vorhaben technisch und/oder organisatorisch sehr neuartig, steigt das Risiko der Nichtmachbarkeit. Wurde allerdings ein Vorhaben in ähnlicher Weise schon mehrfach durchgeführt, wird die Machbarkeit zumeist kaum in Frage gestellt.
- **Machbarkeitsstudie (feasibility study):** Wird durchgeführt um das Risiko eines Totalverlustes (=Projekt nicht machbar) vorzubeugen.
  - Kann komplett analytisch oder durch eine prototypische Umsetzung der kritischen Systemanteile erfolgen. Eine Machbarkeitsstudie besteht aus folgenden Elementen:
    - Identifikation des Inventionsanteils
    - Identifikation von ähnlichen Systemen (auch von der Konkurrenz)
    - Technische Modellierung
    - Technische Prototypen
    - Rechtliche Prüfung
    - Akzeptanzanalyse bei den zukünftigen Benutzern
    - Identifizierung der wesentlichen Kostentreiber
    - Kostenanalyse

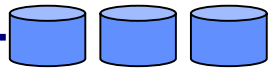
2.2 Grundlagen: Grundlagen der Anforderungsanalyse – Make or Buy

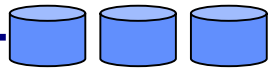


- **Make or Buy:** Soll man die technischen Komponenten selbst entwickeln (make) oder soll beispielweise ein bestehendes COTS (Commercial Of The Shelf) System zum Einsatz kommen (buy)?
  - Die Entscheidung kann auch nur für einzelne Teile (Subsysteme, Teilkomponenten) des Gesamtsystems individuell getroffen werden.
  - Eine Entscheidung mit hoher strategischer Bedeutung. Diese beeinflusst nicht nur die weitere Anforderungsanalyse sondern den gesamten Entwicklungsprozess.
  - Folgende Aspekte sollten bei der Make or Buy Entscheidung berücksichtigt werden:
    - Abdeckung des aktuellen und des antizipierten Projekt-Scope
    - Entwicklungskosten und Lizenzkosten (Anschaffung und laufender Betrieb)
    - Anpassungsmöglichkeiten und Anapassungsaufwand
    - Schulungsaufwand für Entwickler und Betreiber
    - Hardwareanforderungen
    - Support und Wartung
    - Nutzung von offenen oder proprietären Protokollen oder Formaten
    - Herstellerabhängigkeit (Vendor Lock-In)
    - Besitzverhältnisse
    - Weitere strategische Elemente wie Weiterverkaufsmöglichkeiten einer Neuentwicklung

- **Einfluss der Grobarchitektur auf die Anforderungsanalyse:** Grundsätzlich sollten Anforderungen so architekturunabhängig wie möglich erhoben werden, da die Anforderungen die genauen Fähigkeiten und Eigenheiten der Architektur festlegen/steuern sollten.
- Jedoch ist eine strickte Trennung oft nicht möglich da:
  - Oft bereits vor der Entwurfsphase ein grobes ungefähres Architekturkonzept feststehen muss, um die Anforderungserhebung zu fokussieren und um bereits bestehende Lösungen mit einzubeziehen.
  - Die Architektur bestimmt wesentliche nichtfunktionale Eigenschaften des Systems (und umgekehrt).
  - Die Architektur hat einen Einfluss auf die zu wählenden Methoden der Anforderungsanalyse und Anforderungsbeschreibung.
  - Wesentliche Systemkomponenten oft bereits früh in die Anforderungsbeschreibungen integriert werden müssen.
  - Die Architektur die flexiblen und unveränderlichen Systemteile oder Funktionen bestimmt. Vorgegebene unveränderliche Teile müssen dem Kunden schmackhaft gemacht werden, anstatt seine Wünsche (Anforderungen) zu erheben.

Inhalt

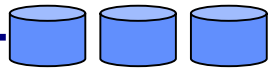




- **Nach IEEE-Standard 610.12 ist eine Anforderung (requirement) folgendermaßen definiert:**
  - *1) „A **condition** or **capability** needed by a **user** to solve a **problem** or achieve an **objective**.“*
  - *2) „A condition or capability that must be **met** or **possessed** by a **system** or **system component** to satisfy a **contract, standard, specification,** or other **formally imposed document.**“*
  - *3) „A **documented representation** of a condition or capability as 1) or 2).“*
- **Quelle:** https://standards.ieee.org/findstds/standard/610.12-1990.html

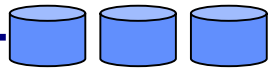
- **Bedeutung von Anforderungen für das Software Engineering**
  - Anforderungen sind **die** inhaltliche Ausgangsbasis für jedes Software Engineering Projekt.
    - Sie sind die Vorgabe für den Entwurf, die Basis für Validation und Verifikation sowie für die Abnahme des Projektes (bei Projektabschluss) durch den Kunden.
  - Der Detaillierungsgrad einer Anforderung ist abhängig vom Fortschritt des Anforderungsanalyseprozesses.
    - Eine Anforderung kann eine abstrakte Beschreibung eines Services oder eine detaillierte funktionale Spezifikation sein.
  - Identifizierte und dokumentierte Anforderungen werden oft auch als **Vertragsgrundlage** für Software Engineering Projekte genutzt.

2.3 Anforderung: Woher kommen Anforderungen?



- **Anforderungsquellen:** Woher kommen eigentlich die Anforderungen für ein Softwareprojekt**?**
  - Hauptquelle für Anforderungen sind die am Softwareprojet beteiligten **Stakeholder.**
  - Neben dieser Hauptquelle gibt es noch weitere wichtige Anforderungsquellen wie:
    - Standards und Normen
    - Richtlinien z.B. zum Datenschutz
    - Gesetze
    - Anforderungen und Funktionsweisen von alternativen Systemen (Konkurrenzanbieter)
    - Verdeckte Regeln: Glaubensgrundsätze/Tabus/Macht/Werte/Einstellungen/Status/Ethik
    - Das Selbstverständnis der beteiligten Organisationen (z.B. Don‘t be evil)

2.3 Anforderung: Wie können wir im Projektverlauf nachvollziehen woher eine Anforderung gekommen ist?

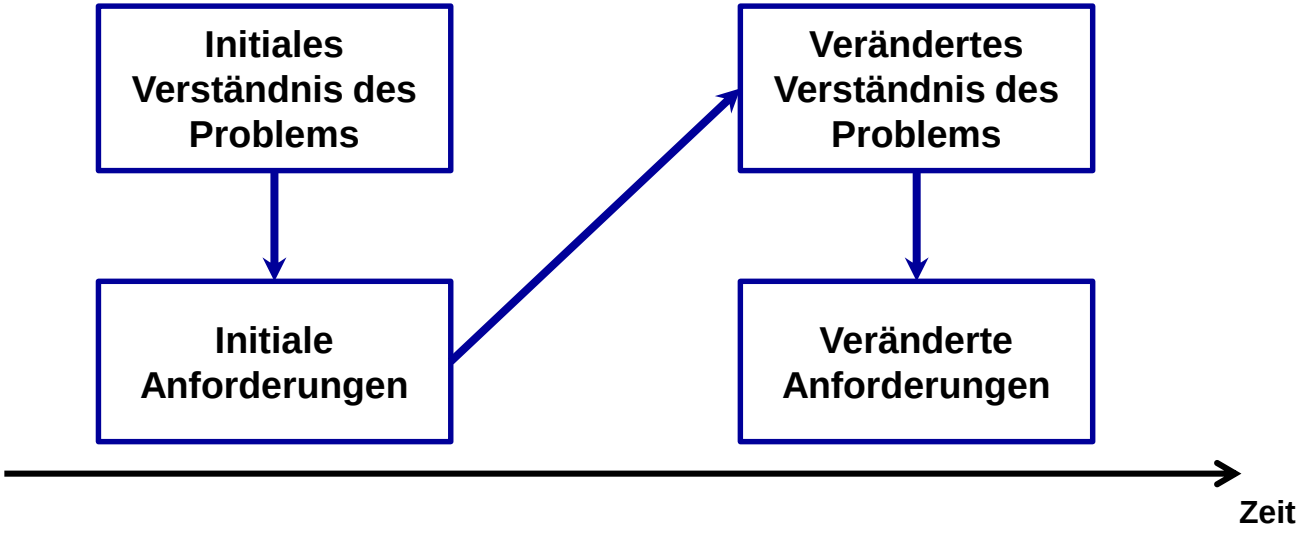


- **Nach dem IEEE- Standard 610.12 ist Traceability wie folgt definiert:**
  - *„The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; e.g., the degree to which the requirements and design of a given system element match.“*
- **Nachverfolgbarkeit der Quelle:** Verweis auf den Stakeholder, welcher diese Anforderung eingebracht hat. „Wann, wie, warum?“
- **Nachverfolgbarkeit zu anderen Anforderungen:** Zusammenhänge zwischen voneinander abhängigen Anforderungen.
- **Nachverfolgbarkeit zu abhängigen Artefakten:** Zusammenhang zwischen den Anforderungen und Elementen des Systemdesigns oder auch zu den Testfällen welche diese Anforderung abdecken.

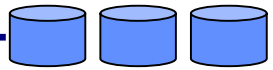
2.3 Anforderung: Wie stabil sind Anforderungen?

- **Anforderungsvolatilität:** Anforderungen ändern sich während der Projektlaufzeit oft, daher ist es notwendig die Anforderungsbeschreibung/Software/Architektur so zu gestallten, dass solche Änderungen einfach und schnell umgesetzt werden können.
- **Die Anforderungen werden dazu in zwei Klassen unterteilt:**
  - **Stabile Anforderungen:** Diese Anforderungen ändern sich voraussichtlich nicht oder nur mehr geringfügig.
  - **Volatile Anforderungen:** Diese Anforderungen ändern sich voraussichtlich während der Anforderungsanalyse oder während des Betriebs des Systems.
- **Auswirkung auf das Systemdesign:**
  - Stabile Anforderungen können mit simpleren Designs bzw. „hard-wired“ implementiert werden.
  - Die Umsetzung volatiler Anforderungen erfordert ein adaptiveres Systemdesign („soft-wired“). *Dies stellt, bei passenden Verträgen, eine Möglichkeit dar dem Kunden mit geringen eigenen Aufwand hohe Zusatzkosten zu verrechnen.*
  - Stabile Anforderungen werden typischerweise früher und detaillierter dokumentiert bzw. modelliert und auch früher im Entwicklungsprozess implementiert.

- **Warum ändern sich Anforderungen?**
  - Stakeholder ändern ihre Meinung darüber was sie vom System erwarten über die Zeit. Beispielsweise aufgrund neuer Erkenntnisse oder Entwicklungen.
  - Falls Anforderungen unvollständig oder ungenügend beschrieben wurden, und hierdurch missverstanden werden können, gelangen Stakeholder oft zu unterschiedlichen oder sich ändernden Interpretation der Anforderung.

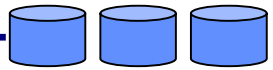


2.3 Anforderung: Wie mit Anforderungsänderungen umgehen?



- **Wie werden Anforderungsänderungen durchgeführt?**
  - Vor dem Setzen eine Baseline werden Anforderungsänderungen direkt und ohne komplexe Entscheidungsprozesse durchgeführt.
  - Nach dem Setzen eine Baseline durchlaufen Anforderungsänderungen einen formellen Entscheidungsprozess.
    - Re-Costing: Anforderungsänderungen können mit einer Änderung der Kosten verbunden sein. *Möchte der Kunde diese Änderung dann überhaupt noch?*
    - Auswirkungsanalyse (Change-Impact-Analysis): Wie wirkt sich die Anforderungsänderung auf die Baseline aus.

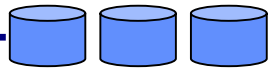
- **Baseline:** Anforderungen ändern sich ständig. Trotzdem benötigt man zu einem definierten Zeitpunkt eine „stabile“ Menge von Anforderungen (die Baseline) zum Entwurf oder zur Implementierung einer Software.
  - Nach IEEE Standard ist eine Baseline wie folgt definiert:
    - *„A specification or system that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development and can be changed only through formal change control procedures.“*



Anforderungen

KLASSIFIKATION

2.3 Anforderung: Klassifikation



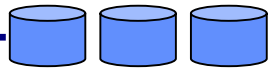
- **Klassifikation von Anforderungen:** Diese ist notwendig um die Anforderungen zu organisieren und so den Beteiligten (Stakeholder) zu ermöglichen genau die Informationen zu erhalten bzw. einzusehen, welche für Sie besonders interessant/relevant sind.
  - Anforderungen lassen sich dem Detailgrad nach klassifizieren in:
    - Benutzeranforderungen
    - Systemanforderungen
  - Anforderungen lassen sich nach der Art der Anforderung klassifizieren in:
    - Funktionale Anforderungen
    - Nichtfunktionale Anforderungen
    - Domänenanforderungen
    - Designbedingungen
    - Prozessbedingungen

2.3 Anforderung: Klassifikation – Detailgrad 1

- **Benutzeranforderungen (user requirements):** Generische, abstrakt gehaltene Beschreibung der Anforderung.
  - Keine Details darüber was das System genau macht.
  - Aussagen in natürlicher Sprache eventuell unterstützt durch Diagramme.
  - Systembeschreibung aus Kundensicht (Lastenheft).
  - Beispiel einer Benutzeranforderung:

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

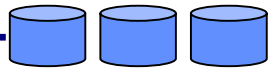
2.3 Anforderung: Klassifikation – Detailgrad 2



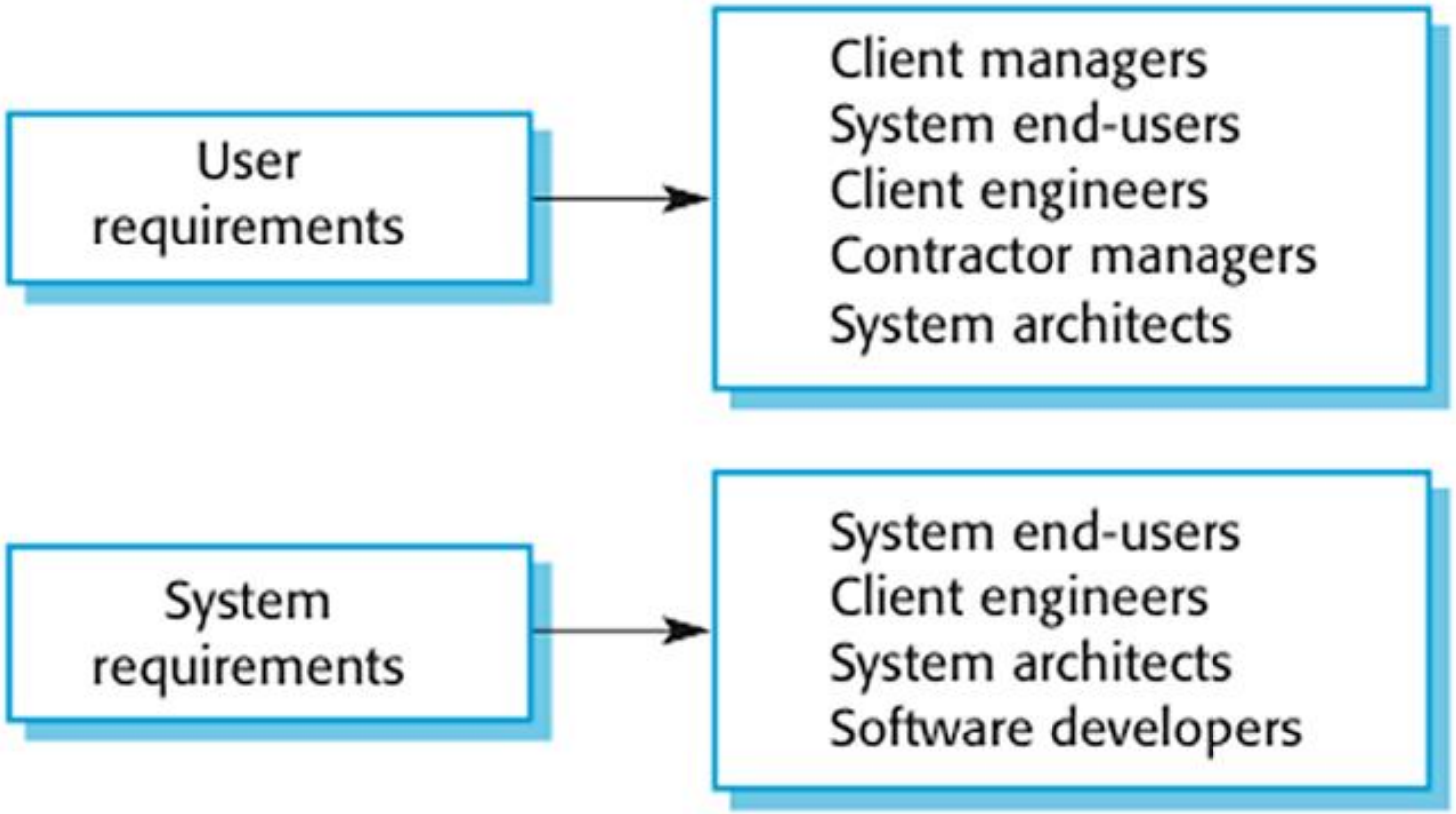
- **Systemanforderungen (system requirements):** Bieten eine detailliertere Beschreibung der Services und Beschränkungen.
  - Beschreibung was implementiert werden soll.
  - Systembeschreibung aus technischer Sicht (Pflichtenheft).
  - Beispiel einer Systemanforderung:

1.1 On the last working day of each month, a summary of drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17:30 on the last working da of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

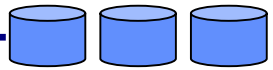
2.3 Anforderung: Klassifikation – Detailgrad 3



- **Lesertypen:** Verschiedene Stakeholder interessieren sich für unterschiedliche Anforderungen und Anforderungsklassen:



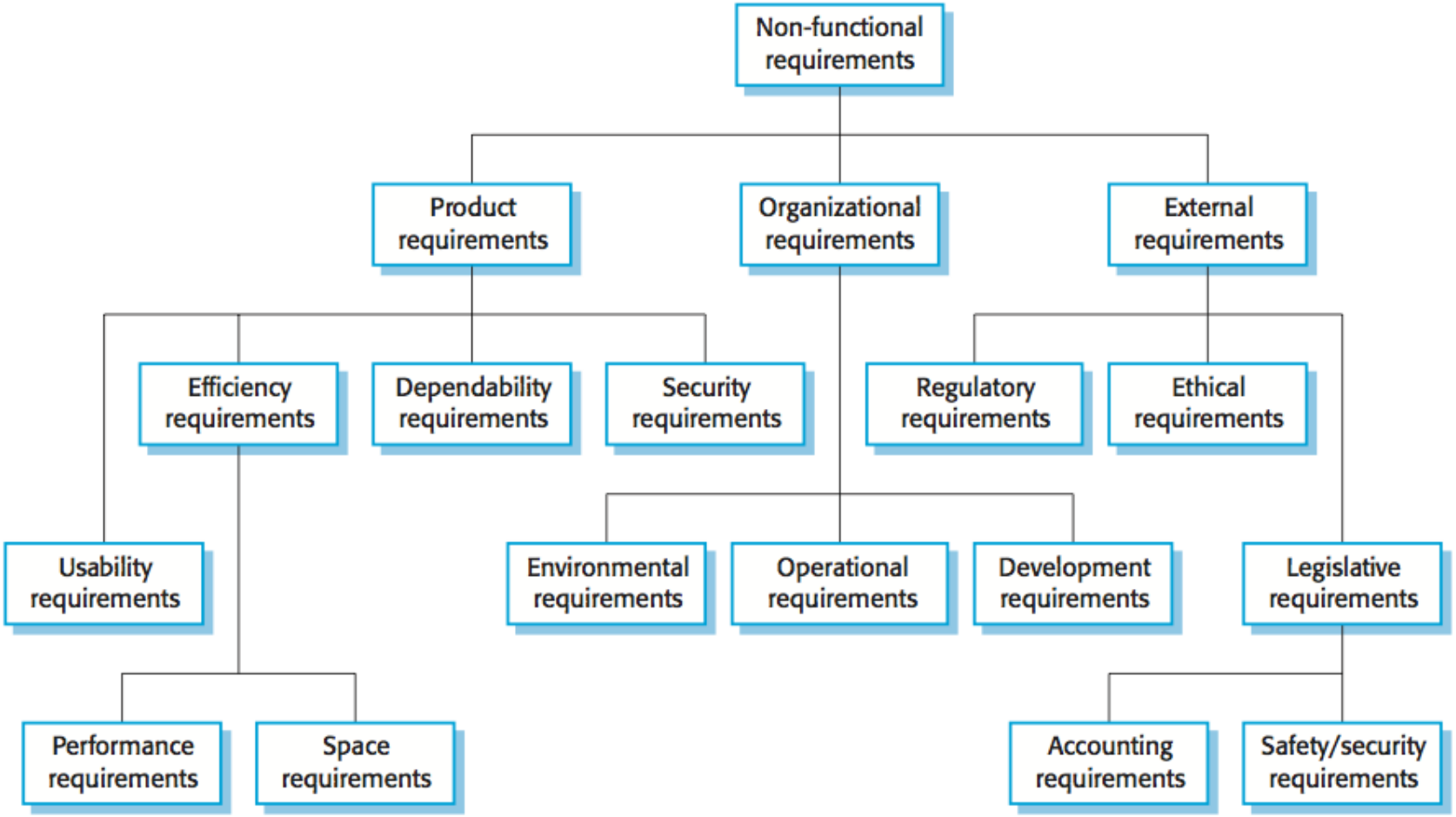
2.3 Anforderung: Klassifikation – Arten 1



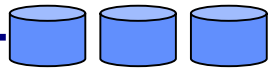
- **Funktionale Anforderungen:** Bilden das eigentliche Systemverhalten und die jeweiligen Funktionen des zu erstellenden Produkts ab.
- **Nichtfunktionale Anforderungen:** Zielen auf zusätzliche Systemattribute und Qualitätsmerkmale ab, die nicht unmittelbar die Funktionalität des Systems betreffen.
  - Diese umfassen beispielsweise: Anwenderfreundlichkeit (Usability), Effizienz oder Performance.
  - *Nichtfunktionale Anforderungen sind oft kritischer als funktionale Anforderungen*: Benutzer können normalerweise Wege finden, um mit einer Systemfunktion umzugehen, die ihren Anforderungen nicht vollständig entspricht. Wenn jedoch eine nichtfunktionale Anforderung nicht erfüllt wird, kann dies bedeuten, dass das gesamte System unbrauchbar ist (z.B. überlange Ladezeiten führen zur Unbenutzbarkeit).
  - Nichtfunktionale Anforderungen müssen ebenfalls überprüfbar sein.
    - Überprüfung gestaltet sich schwieriger als bei funktionalen Anforderungen.

2.3 Anforderung: Klassifikation – Arten 2

- **Nichtfunktionale Anforderungen – Arten:**



2.3 Anforderung: Klassifikation – Arten 3



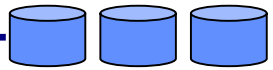
- **Nichtfunktionale Anforderungen – Messbarkeit:**

Property	Measure
Speed	Processed transactions/second User/event response time
Size	Megabytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability Correctness	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

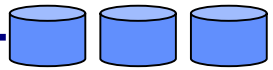
2.3 Anforderung: Klassifikation – Nichtfunktionale Anforderungen erkennen

- **Nichtfunktionale Anforderungen beantworten Fragen wie:**
  - Wie gut nützt <X> die Ressourcen aus um <Y> zu erreichen?
  - Welche Leistungsfähigkeit wird von <X> bei <Y> erreicht? (Antwortzeit, Durchsatz, etc.)
  - Wie sicher ist <X> während <Y> durchgeführt wird?
  - Wie korrekt arbeitet <X> beim Bearbeiten von <Y>?
  - Wie gut arbeitet <X> in Hochlastszenarien?
  - Wie leicht lässt sich <X> einsetzen um <Y> zu erreichen? (Usability)
  - Wie vollständig genügt <X> den Anforderungen?
  - Wie leicht ist es die Leistungsfähigkeit von <X> während <Y> verifizieren?
  - Wie leicht kann <X> mit Lösung <Y> integriert werden?
  - Wie leicht kann <X> an neue Anforderungen <Y> angepasst werden?
  - Wie leicht kann <X> mit <Y> erweitert werden?
  - **Tipp:** Achten Sie darauf, dass die Formulierung der Anforderung diese und vergleichbare Fragen mit einem Fokus auf Qualität beantwortet.

2.3 Anforderung: Klassifikation – Arten 4



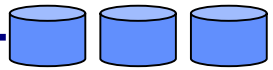
- **Domänenanforderungen:** Funktionale oder nichtfunktionale Anforderungen, die aus der Domäne des Systems stammen und die besonderen Eigenschaften dieses Gebietes abbilden.
  - Oft inhärent und nicht explizit pro Projekt dokumentiert.
  - Beispielsweise Compliance-Anforderungen des Finanzwesens.
- **Designbedingungen:** Neben den produktrelevanten Eigenschaften müssen auch Eigenschaften die auf den Herstellungsprozess abzielen definiert werden.
  - Legen die technischen Rahmenbedingungen, wie beispielsweise Entwicklungsumgebung und Zielplattform fest.
- **Prozessbedingungen:** Definieren die grundlegende Vorgehensweise bei der Entwicklung eines Software-Produkts.
  - Legen den konkreten Software-Prozess (eventuell unter Einsatz von Vorgehensmodellen) und die einzusetzenden Methoden fest.



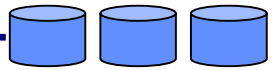
Anforderungen

QUALITÄT

2.3 Anforderung: Qualität 1

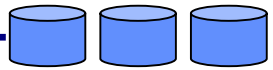


- **Ziel:** Gute Anforderungen sollten komplett und vollständig sein.
  - **Vollständigkeit:** Alle für das System benötigten Anforderungen sollten abgedeckt sein.
  - **Konsistenz:** Die definierten Anforderungen enthalten keine Konflikte oder Widersprüche und werden von allen Stakeholdern gleich aufgefasst.
- **Aber:** In der Realität ist es kaum möglich dieses Ziel (gute Anforderungen) zu erreichen, da diverse Probleme während der Definition der Anforderungen auftreten. Beispielsweise:
  - Unverständlichkeit
  - Selbstverständlichkeit
  - Wenig Präzision
  - Vermischung
  - Zusammenführung
  - Mehrdeutigkeit
  - Überflexibilität



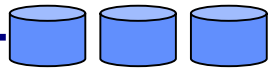
- **Probleme bei der Definition von Anforderungen**
  - **Unverständlichkeit:** Anforderungen werden typischerweise in der Sprache der Applikations-Domain repräsentiert.
    - Diese Sprache wird von Softwareingenieuren oft nicht verstanden.
  - **Selbstverständlichkeit:** Domain-Spezialisten sind mit ihrer Domäne so vertraut, dass sie bestimmte Anforderungen als selbstverständlich erachten und deshalb nicht explizit dokumentieren.
  - **Wenig Präzision:** Präzise Definition von Anforderungen ⇔ Lesbarkeit
  - **Vermischung:** Funktionale und nichtfunktionale Anforderungen werden häufig vermischt.
  - **Zusammenführung:** Gemeinsame (vermischte) Beschreibung von verschiedenen Anforderungen.
  - **Mehrdeutigkeit:** Spezifikationen sind oft (absichtlich) mehrdeutig gehalten.
  - **Überflexibilität:** Die natürliche Sprache ist oft mehrdeutig interpretierbar und somit zu flexibel um exakte Anforderungen zu erzeugen.

2.3 Anforderung: Qualität 3



- **Qualitätsattribute:** Sind der Versuch den Problemen bei der Definition von Anforderungen entgegenzuwirken.
  - **Vollständig:** Alle Anforderungen müssen explizit beschrieben werden.
    - Es darf keine impliziten Annahmen eines Stakeholders geben.
  - **Eindeutig definiert/abgegrenzt:** Präzise Definitionen helfen, Missverständnisse zwischen Stakeholdern zu vermeiden.
  - **Verständlich beschrieben:** Stakeholder können unter vertretbarem Aufwand die gesamte Anforderung lesen und verstehen.
  - **Atomar:** Beschreibung von nur einer Anforderung pro Anforderungs-ID.
  - **Identifizierbar:** Jede Anforderung ist eindeutig identifizierbar.
  - **Einheitlich dokumentiert:** Keine unterschiedlichen Dokumente oder Strukturen.
  - **Notwendig:** Klären ob Anforderung überhaut benötigt wird.
  - **Nachprüfbar:** Verknüpfung mit Abnahmetestfällen damit nachweisbar und reproduzierbar überprüft werden kann, ob eine Anforderung erfüllt wurde.
  - **Rück- und vorwärts verfolgbar:** Wurde jede Anforderung vollständig erfüllt und für jede implementierte Funktionalität ist erkennbar zu welcher Anforderung sie gehört.
  - **Priorisiert:** Festgelegen der Priorität der Anforderung.

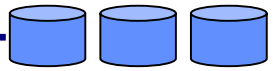
2.3 Anforderung: Qualität - Beispiele für gute und schlechte Anforderungen



Gute Anforderungen	Schlechte Anforderung
Als Kunde kann ich mich mit E-Mail und Passwort anmelden um meinen Account einzusehen.	Es muss eine Möglichkeit geben sich anzumelden.
Es ist klar, **wer**, **was** und **warum** etwas gemacht wird. Nähere Details z.B. zum Account mit zusätzlichen Anforderungen abbilden.	Wenig genaue Informationen, die Umsetzung erfolgt voraussichtlich nicht wie gewünscht. Sicherstellen, dass klar ist wer, was und warum möchte.

Gute Anforderungen	Schlechte Anforderung
Als Mitarbeiter kann ich den Helpdesk mit einem online Formular erreichen um Unterstützung bei IT-Problemen zu erhalten.	Ich benötige eine Support-System mit Chat, Formularen, Hilfesystem und Tickets.
Unterstützung wird zumeist auf vielen Wegen gegeben werden. Dokumentieren Sie in einer Anforderung aber immer nur einen **kleinen Bereich** nach dem Anderen. Dies erleichtert die Verhandlung, Priorisierung, Diskussion.	Überlegen Sie sich wie viel diese Anforderung umfasst, Telefonsupport, 24/7 Mail Support, Websites, etc. - hunderte Use-Cases. Halten Sie Anforderungen einfach, teilen Sie diese hierzu auf.

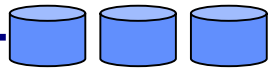
2.3 Anforderung: Qualität - Beispiele für gute und schlechte Anforderungen



Gute Anforderungen	Schlechte Anforderung
Als Kunde möchte ich mich mittels Facebook anmelden um Bestellungen aufzugeben.	Ich möchte die Facebook ID in der Datenbank speichern.
Anforderungen sollen Aspekte beschreiben die für einen Use-Case und einen Anwendungsfall **relevant** sind – "High Level", nicht zu technisch.	Die Anforderung für sich gesehen macht wenig Sinn. Warum wird die ID benötigt? Kombinieren Sie es mit einem Use-Case oder einer Szenariobeschreibung

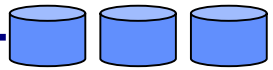
Gute Anforderungen	Schlechte Anforderung
Als Marketingleiter möchte ich durch ein Diagramm sehen wie viel Zeit auf den einzelnen Webseiten verbracht wird um deren Usability überblicksartig zu beurteilen.	Als Anwender möchte ich ein Tortendiagramm sehen welches mit der Bibliothek highlights oder D3.js gebaut wurde.
Zielgerichtete Anforderung die auf einen Use-Case verweist. Nützen Sie weitere Anforderungen und UI-Designer sowie Mockups um abzubilden wie genau die Visualisierung erfolgen soll z.B. die Diagrammart.	Technische Anforderungen und Bibliotheken zu nennen schränken zumeist den Blickwinkel zu stark ein. Neue Technologien werden dadurch verhindert. Der Fokus auf die eigentlichen Anforderungen und Use-Cases geht verloren, vermeiden Sie dies.

2.3 Anforderung: Qualität - Anforderungen wirklich vollständig abbilden



Gute Anforderungen	Schlechte Anforderung
Als Kunde kann ich mich mit E-Mail und Passwort anmelden um meinen Account einzusehen. Erfolgreich umgesetzt wenn: Szenario: Erfolgreicher Login Angenommen der Kunde ist auf Loginseite Wenn Kunde E-Mail & Passwort eingibt Und diese Daten absendet Dann erhält der Kunde ein Bestätigungsmail mit Bestätigungslink. 1. Passwörter länger wie 6 Zeichen 2. Passwort und E-Mail sind nicht leer 3. @gmail.com ist nicht erlaubt	Ich möchte Anmeldungen nur von respektablen Domains. Können wir auch Bestätigungsmails einbauen?
Anforderungen sollten mit Use-Cases und Szenarien kombiniert werden. Wie reagiert ein System? Welche Edge-Cases gibt es? Wie kann eine erfolgreiche Umsetzung festgestellt werden?	Unklarheit im Falle fehlender Daten (z.B. leere E-Mail-Felder). Die Beschreibung lässt viel Freiraum ("respektable Domains"?) und vermischt mehrere Anforderungen (Anmelden und Bestätigungsmails).

Inhalt

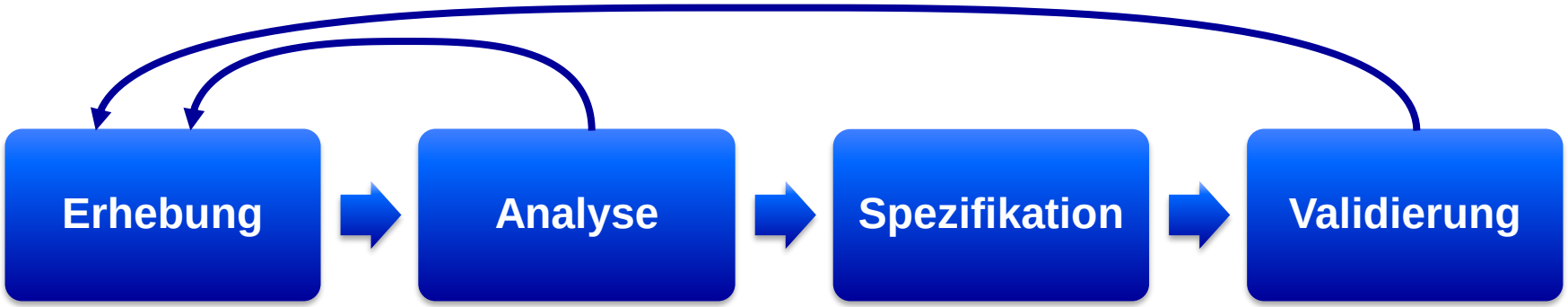


2.4 Anforderungsanalyseprozess: Bestandteile

- **Bestandteile des Anforderungsanalyseprozesses nach IEEE**
  - **Anforderungserhebung (requirements elicitation):** Anforderungen der Stakeholder sammeln.
  - **Anforderungsanalyse (requirements analiysis):** Erwartetes Systemverhalten verstehen und die gesammelten Anforderungen so darstellen, dass sie eindeutig, testbar und verständlich sind.
    - Wird teilweise (je nach Autor) als Teil der Anforderungserhebung oder der Anforderungsspezifikation betrachtet.
  - **Anforderungsspezifikation (requirements specification):** Die modellierten und bewerteten Anforderungen fließen in Spezifikationen, welche das Systemverhalten in Bezug auf die Anforderungen beschreiben, ein. Die so entstandenen Dokumente sind die Grundlage für die spätere Planung und technische Umsetzung.
  - **Anforderungsvalidierung (requirements validation):** Sicherstellen, dass die definierten Anforderungen die Bedürfnisse des Kunden erfüllen. Nach Abschluss dieser Phase liegt die fertige Anforderungsspezifikation für das zu entwickelnde Produkt vor.

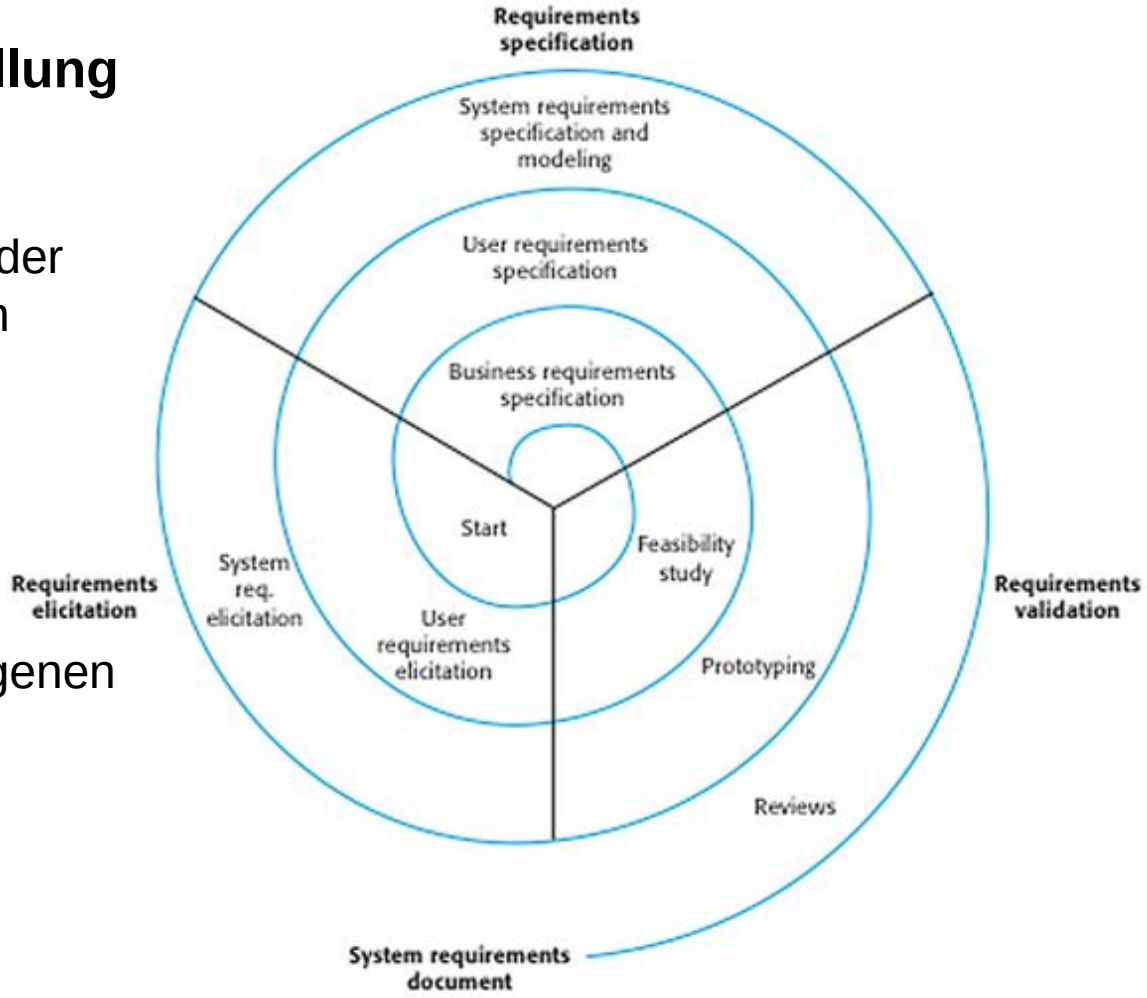
2.4 Anforderungsanalyseprozess: Bestandteile – Grafische Darstellung 1

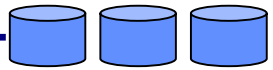
- **Grafische Darstellung der Bestandteile des Anforderungsanalyseprozesses**
  - Zurückführende Pfeile von der Analyse und Validierung dienen zur Korrektur und Nachbesserung der im ersten Schritt erhobenen Anforderungen.
  - Der Einfachheit halber wurde das Wort „Anforderung“, welches bei jedem Schritt dazugehören würde, weggelassen.



2.4 Anforderungsanalyseprozess: Bestandteile – Grafische Darstellung 2

- **Detailliertere Darstellung**
  - Soll nur zur Verdeutlichung des iterativen Ablaufes der vorher dargestellten Phasen dienen.
  - Details zur Machbarkeitsstudie (Feasibility study) wurden bereits in den vorangegangenen Folien beschrieben.

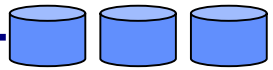




Anforderungsanalyseprozess

ANFORDERUNGSERHEBUNG

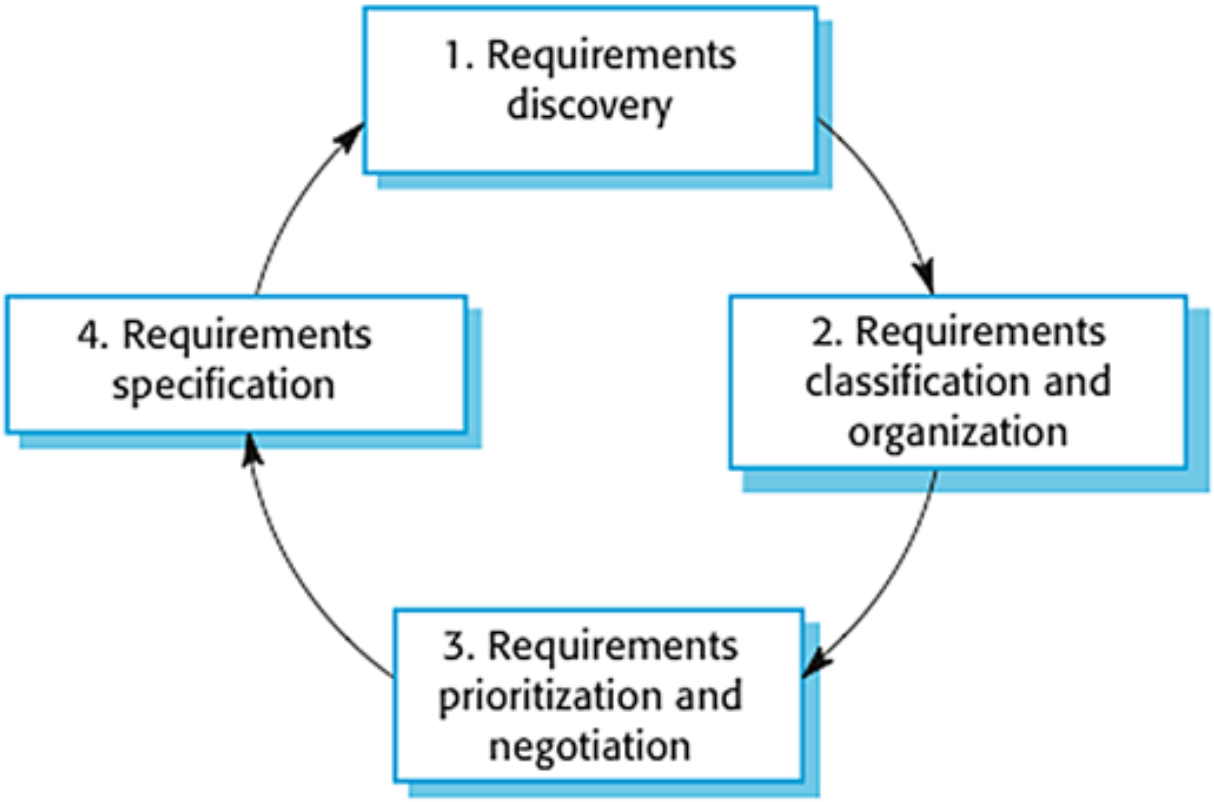
2.4 Anforderungsanalyseprozess: Anforderungserhebung 1



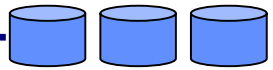
- **Anforderungserhebung (requirements elicitation):** Ziel ist es, genau abgestimmte Anforderungen in einem genau definierten Projektumfeld zu sammeln, klassifizieren und zu priorisieren.
  - Dazu ist die Zusammenarbeit mit allen Stakeholdern notwendig, welche sich zumeist als schwierig erweist, da:
    - Stakeholder oft nicht wissen was sie vom zu entwickelnden System erwarten.
    - Stakeholder of unrealistische Anforderungen haben, weil sie meist nicht abschätzen können was machbar ist und was nicht.
    - Stakeholder die Anforderungen in ihrer „Sprache“ ausdrücken und dabei unbewusst bestimmte Annahmen und Wissen voraussetzen. Das heißt die Anforderungen können nur mit entsprechenden Wissen über die Domäne verstanden werden.
    - Verschiedene Stakeholder formulieren ihre Anforderungen auf unterschiedliche Weise.
    - Politische Faktoren können die Anforderungen eines Systems beeinflussen.
    - Das Umfeld in welchem die Analyse stattfindet ändert sich beständig, das heißt es können sich auch die Anforderungen ändern.
  - Lösungsansatz: Die Sprache der Stakeholder erlernen (z.B. Domänenwissen aneignen) und die Anforderungen nach Stakeholdergruppen (dies sind Stakeholder welche eine gemeinsame Sicht auf die Software teilen) gruppieren.

2.4 Anforderungsanalyseprozess: Anforderungserhebung 2

- **Wie läuft die Anforderungserhebung ab?**

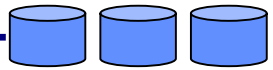


2.4 Anforderungsanalyseprozess: Anforderungserhebung – Schritt 1



- **Anforderungsermittlung (requirements discovery):** Interaktion mit den Stakeholdern des Systems, um die Anforderungen zu ermitteln.
  - Es sind diverse Techniken zur Anforderungsermittlung verfügbar, beispielsweise:
    - **Workshops**
    - **Interviews**
    - **Dokumentenanalyse:** Analyse bestehender Dokumente und Prozesse.
    - **Beobachtungen:** Bei existierenden Systemen/Prozessen wird das Verhalten bzw. der Arbeitsalltag der Anwender beobachtet und ein Analyst versucht durch gezielte Fragen das Vorgehen der Anwender zu verstehen.
    - **Mitarbeit:** Wie Beobachtungen, nur arbeitet der Analyst nun auch selbst mit.
    - **Fokusgruppen**
    - **Fragebögen**
    - **Prototyping**
  - Für Stakeholder ist es oftmals schwierig Systemanforderungen direkt zu kommunizieren, deshalb hilft es wenn diese beschreiben, wie das System in einer bestimmten Situation benutzt werden soll bzw. wird.
    - Techniken hierzu sind beispielsweise Benutzergeschichten (User Stories) und Szenarios.
- Wichtig: Bei allen Interaktionen mit Stakeholdern Protokolle erstellen!

2.4 Anforderungsanalyseprozess: Anforderungsermittlung – Techniken 1



- **Anforderungsermittlung mittels Workshops:** Ein Analyst trifft mehrere Stakeholder um gemeinsam Ergebnisse zu erarbeiten.
  - Dies unterstützt den Aufbau eines effektiven Teams, welches gemeinsam auf ein Ziel hinarbeitet
  - Alle Stakeholder werden einbezogen ⇨ Schafft Stakeholder Buy-In
  - Erarbeitung eines Konsens an die Eigenschaften des zu entwickelnden Systems
  - Identifizierung und Lösung von politischen/verborgenen Aspekten welche den Projekterfolg gefährden könnten (z.B. Rivalitäten zwischen Abteilungen)
  - Ergebnis ist Vorabversion der funktionalen oder nicht-funktionalen Anforderungen.
- **Anforderungsermittlung mittels Interviews:** Analyst trifft sich mit einem Vertreter der Stakeholder um dessen Wünsche und Anforderungen aufzunehmen.
  - Weniger offen aber strukturierter als Workshops.
  - Wichtig: Kontextfreie, also allgemeine Fragen stellen wie beispielsweise
    - Warum existiert dieses Problem?
    - Wie lösen Sie das Problem heute?
    - Wie würden Sie das Problem gerne lösen?

Interviews oder Was der Kunde *wirklich* will?

Planung: Interview Skript

Gibt Ablauf vor
logischer Aufbau

Strukturierung

1. Einleitung
   - ↳Ziel verdeutlichen
   - ↳Ergebnisverwertung
   - ↳Aufzeichnen (Erlaubnis!)
   - ↳Keine falschen Antworten
   - ↳Einfache Eingangsfragen
2. Hauptteil
   - ↳Überblicksfragen
   - ↳Typische Abläufe
   - ↳Eigene Fragen:
     Erkannte Herausforderungen
     Annahmen überprüfen, ...
3. Schluss
   - ↳Fragen der Interviewpartner
   - ↳Kontaktdaten austauschen

Fragen: Do's & Don'ts

Suggestivfragen

: Was möchten Sie?
Würden Sie XY kaufen?

Wie viel möchten
Sie ausgeben?

Erfahrungen,
prägende Momente

Wie werden Aufgaben
erledigt (vorzeigen)?

Offene Fragen, z.B. Wie...

Motivation ermitteln.
Was soll erreicht werden?
Warum?

Herangehens-weise

Nachfragen möglich

Warum? Wieso?
Wiederholt fragen,
Hintergründe ermitteln

Fragen umformulieren

Nicht unterbrechen,
auch bei Missverständ-
nissen ausreden lassen

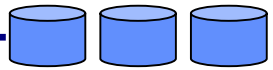
Flexibel auf überraschende
Erkenntnisse reagieren

Bis zum Ende nachfragen

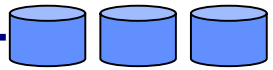
Konversation, kein Verhör,
keine Wertung

Was würden Sie sich wünschen
wenn alles möglich wäre?
Wie würde es dann laufen?

2.4 Anforderungsanalyseprozess: Anforderungsermittlung – Techniken 2

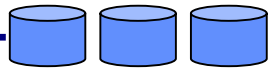


- **Anforderungsermittlung mittels User Stories und Szenarios:** Diese stellen eine Beschreibung darüber dar wie das System für eine spezifische Aufgabe verwendet werden kann/soll.
  - **User Stories**: Sind eine textuelle Beschreibung (eine „Geschichte“).
  - **Scenarios:** Entstehen aus Stories, strukturieren diese und fokussieren sich auf die Interaktionen der Anwender mit dem System.
    - Während des Prozesses der Anforderungsermittlung werden Details hinzugefügt um eine komplette Beschreibung zu erhalten.
- **Bestandteile eines Scenarios**
  - Eine Beschreibung über:
    - Was System und Benutzer beim Start des Scenarios erwarten.
    - Den „normalen“ Ablauf der Benutzung.
    - Was schieflaufen kann und wie damit umgegangen wird.
    - Information über andere Aktivitäten die parallel ablaufen können.
    - Den Systemzustand nach Abschluss des Scenarios.

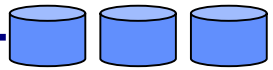


- **User Stories und Szenarios anhand des Beispiels AlwaysSafe**
  - **Projekt:** Kunden geben ihren Reklamationswunsch über ein Onlineformular ein. Dieses enthält obligatorisch die Versicherungsnummer, die Schadensumme und eine Beschreibung des Schadens. Optional können Fotos zur weiteren Dokumentation hochgeladen werden. Liegt die Schadensumme unter einem Bagatellwert von 100 Euro wird eine stichprobenartige Prüfung durchgeführt und der Schadensfall bewilligt. Der Kunde wird per Email benachrichtigt und die Überweisung in die Wege geleitet. Im Falle einer höheren Schadensumme erfolgt eine Prüfung durch einen Sachbearbeiter. Die Entscheidung wird wieder per Email an den Kunden übermittelt. Kunden soll im Falle einer Ablehnung ein optionales Kundengespräch angeboten werden.

2.4 Anforderungsanalyseprozess: Anforderungsermittlung – Techniken 4

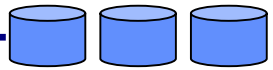


- **Stories und Szenarios anhand des Beispiels AlwaysSafe**
  - **User Story aus Sicht eines Kunden:** Mein Auto hat einen Hagelschaden. Der Schaden wurde auf 500 Euro geschätzt. Ich gebe den Schadensfall zusammen mit Fotos meines Autos und dem geschätzten Schadenswert in das System ein. Ich erwarte dann eine kurze Bestätigung, dass mein Fall bearbeitet wird und wie lange dies ungefähr dauern wird. Innerhalb der Frist erwarte ich eine Entscheidung. Während der Frist möchte ich den aktuellen Bearbeitungsstand online einsehen können. Ich möchte über die Entscheidung mit meinem Versicherungsberater sprechen können.
  - **User Story aus Sicht des Sachbearbeiters:** Das System präsentiert mir einen Überblick über alle zu bearbeitenden Schadensfälle. Diese sind bereits in Bagatell- und Schadensfälle vorsortiert. Die Bagatellfälle kann ich stichprobenartig einsehen, wobei das System mir nach verschiedenen Kriterien die Stichproben vorschlägt. Das System bietet zu jedem Schadensfall die notwendige Information. Ich kann zu jeden Kunden die bisherige Historie einsehen. Ich kann den Kunden bei Rückfragen kontaktieren und alles im System dokumentieren. Die Entscheidung wird in das System eingegeben und der Kunde automatisch kontaktiert.

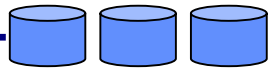


- **Ein aus den beiden User Stories entstandenes Szenario**
  - **Ausgangssituation**: Ein oder mehrere Kunden möchten einen Schadensfall eintragen. Die Kunden sind im System registriert. Infos zum Schadensfall liegen den Kunden (zum Teil) digital vor (z.B. Fotos).
  - **Normaler Ablauf:** Die Kunden füllen das Formular zum Schadensfall aus und laden ggf. Fotos hoch. Eine Schadensumme mit Begründung muss angegeben werden. Der Kunde enthält eine Bestätigungsemail mit Kontaktdaten für Rückfragen.
  - **Mögliche Fehlerfälle (nicht vollständig):**
    - Kunde trägt Schadensumme von 0 Euro ein. ⇨ Rückfrage an den Kunden und Nichtweiterverarbeitung des Formular.
    - Kunde lädt ein falsches Foto hoch. ⇨ Möglichkeit zum erneuten Upload / Überschreiben.
  - **Parallele Aktivitäten:** Sachbearbeiter prüft Bagatellschadensfälle stichprobenartig.
  - **Systemzustand am Ende:** Kunde ist eingeloggt. Die Daten für den Schadensfall sind im System für den Sachbearbeiter verfügbar. Eine Bestätigungsemail wurde an den Kunden versandt.

2.4 Anforderungsanalyseprozess: Anforderungserhebung – Schritte 3 bis 5



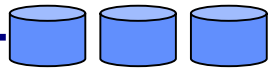
- **Klassifikation und Organisation von Anforderungen (requirements classification and organization):** Übernimmt die unstrukturierte Sammlung der Anforderungen aus dem vorhergehenden Schritt und gruppiert verwandte Anforderungen damit kohärente Cluster entstehen.
- **Priorisierung von Anforderungen (requirements prioritization and negotiation):** Ermittelte Anforderungen werden priorisiert und es wird durch Verhandlungen mit den Stakeholdern versucht eine Lösung für Anforderungskonflikte (z.B. widersprüchliche Anforderungen) zu finden.
- **Anforderungsdokumentation (requirements documentation):** Zum Abschluss werden die Anforderungen dokumentiert. Diese Dokumentation stellt den Input für die nächsten Schritte dar.



Anforderungsanalyseprozess

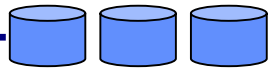
ANFORDERUNGSANALYSE

2.4 Anforderungsanalyseprozess: Anforderungsanalyse 1



- **Anforderungsanalyse (requirements analysis):** Das Ziel dieses Schrittes ist es, dass aus den im vorangegangenen Schritt gesammelten Anforderungen, wohlgeformte Anforderungen entstehen.

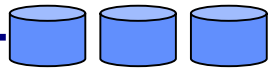
- Nach IEEE Standard 29148 ist eine wohlgeformte Anforderung wie folgt definiert:
  - *"A well-formed requirement is a statement that **can be verified**, has to be **met or possessed by a system** to solve a stakeholder problem or to achieve a stakeholder objective, is **qualified by measurable conditions** and **bounded by constraints**, and **defines the performance of the system** when used by a specific stakeholder or the corresponding capability of the system, but **not a capability of the user, operator, or other stakeholder**."*



- **Wie können wohlgeformte Anforderungen dargestellt werden?**
  - Benutzeranforderungen werden typischerweise nur in natürlicher Sprache, unterstützt durch Diagramme und Tabellen, dargestellt.
  - Für Systemanforderungen gibt es mehrere Darstellungsmöglichkeiten:

Notation	Description
Natural language sentences (unstructured method)	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language (semi-structured method)	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations (structured method)	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML use case and sequence diagrams are commonly used.
Mathematical specifications (structured method)	These notations are based on mathematical concepts such as finite-state machines or sets.

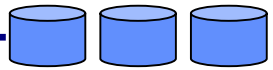
2.4 Anforderungsanalyseprozess: Anforderungsanalyse – Darstellung 1



- **Darstellung der Anforderung mittels Sätzen in natürlicher Sprache (natural language sentences):** Ausdrucksstarke, intuitive und universelle Möglichkeit um Systemanforderungen zu darzustellen.
  - Beispielanforderung für Insulinpumpen:

> 3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. (Changes in blood sugar are relatively slow, so more frequent measurement is unnecessary; les frequent measurement could lead to unnecessarily high sugar levels.)
>
> 3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. (A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)

2.4 Anforderungsanalyseprozess: Anforderungsanalyse – Darstellung 2

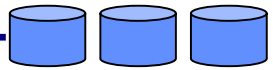


- **Darstellung der Anforderung mittels strukturierter natürlicher Sprache (structured natural language):** Verwendung von Vorlagen um Systemanforderungen strukturiert/vergleichbar zu spezifizieren.
  - Beispielanforderung für Insulinpumpen (strukturiert):

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose – the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result is rounded to zero then CompDose is set to the minimum dose that can be delivered. (see 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

2.4 Anforderungsanalyseprozess: Anforderungsanalyse – Darstellung 3



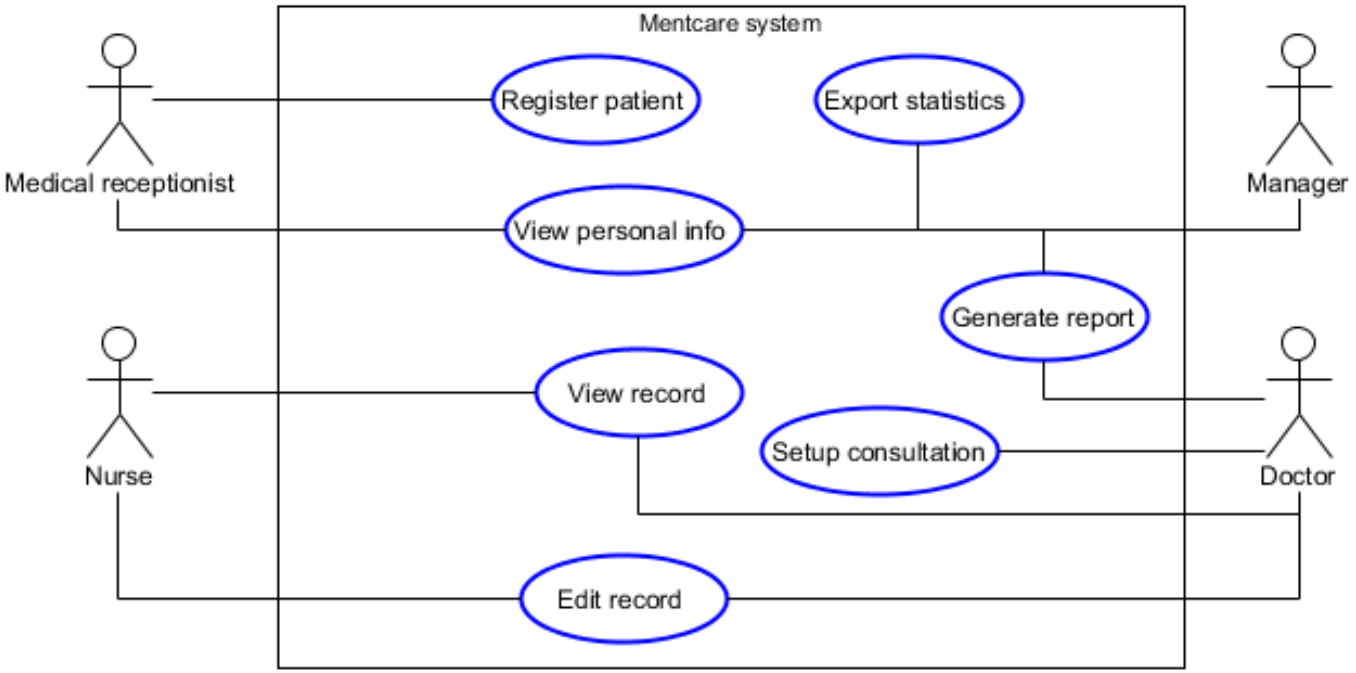
- **Darstellung der Anforderung mittels formaler Methoden (mathematical specifications):** Verwendung von Regeln um Systemanforderungen formal zu spezifizieren (Formeln, Pseudocode).
  - Beispielanforderung für Insulinpumpen (formal): Formale, tabellarische Spezifikation der Berechnung in einer Insulinpumpe (Wenn/Dann Regeln)

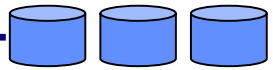
Condition	Action
Sugar level falling $(r2 < r1)$	CompDose = 0
Sugar level stable $(r2 = r1)$	CompDose = 0
Sugar level increasing and rate of increase decreasing $\big((r2 - r1) < (r1 - r0)\big)$	CompDose = 0
Sugar level increasing and rate of increase stable or increasing $r2 > r1$ & $\big((r2 - r1) \geq (r1 - r0)\big)$	CompDose = $round(r2 - r1)/4$ If rounded result = 0 then CompDose = MinimumDose

2.4 Anforderungsanalyseprozess: Anforderungsanalyse – Darstellung 4

- **Darstellung der Anforderung mittels graphischer Notationen (graphical notations):** Folgt einer definierten Syntax, Notation oder formalen Spezifikation um Systemanforderungen zu beschreiben.
  - Die relevanteste Modellierungssprache zur grafischen Darstellung von Anforderungen ist die Unified Modeling Language (UML).
    - Die Sprache besteht aus verschiedenen Modelltypen wie beispielsweise Anwendungsfalldiagramm, Aktivitätsdiagramm, Zustandsdiagramm, Klassendiagramm und Komponentendiagramm. Mit diesen Modelltypen können die Anforderungen der Stakeholder und deren Sicht auf ein System grafisch modelliert werden.
  - Weitere relevante Modellierungssprachen/Modelle
    - Domain Specific Language (DSL): Spezialisierte Modellierungssprache für einen spezifischen Anwendungsbereich.
    - Entity-Releationship-Modelle (ER-Diagramm): Wird für die Modellierung eines Datenmodells (z.B. für Datenbanken) eingesetzt.

- **Darstellung der Anforderung mittels graphischer Notationen (graphical notations):** Folgt einer definierten Syntax, Notation oder formalen Spezifikation um Systemanforderungen zu beschreiben.
  - Beispiel für eine grafische Darstellung anhand eines Anwendungsfalldiagramms (use case diagrams):





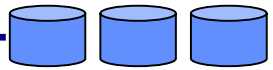
Anforderungsanalyseprozess

ANFORDERUNGSSPEZIFIKATION

2.4 Anforderungsanalyseprozess: Anforderungsspezifikation 1

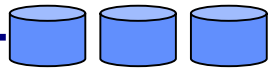
- **Anforderungsspezifikation (requirements specification):** Das Ziel ist es, die im vorhergehenden Schritt beschriebenen Anforderungen in ein gemeinsames Dokument zusammenzufassen.
  - Je nach Autor, Quelle, Zielsetzung und Standard werden Anforderungsspezifikationen unterschiedlich bezeichnet:
    - Spezifikation
    - Anforderungsanalyse (-dokument)
    - Lastenheft/Pflichtenheft
    - Softwareanforderungsdokument/Softwareanforderungsspezifikation (Software-Requirements-Specification kurz SRS)
- **Nur im deutschsprachigen Raum wird zwischen dem Lastenheft und Pflichtenheft unterschieden:**
  - Lastenheft: Ist im Besitz des Auftraggebers und definiert die Anforderungswünsche.
  - Pflichtenheft: Wird vom Auftraggeber geführt und definiert die Anforderungsbasis für das Entwicklungsvorhaben.

2.4 Anforderungsanalyseprozess: Anforderungsspezifikation 2

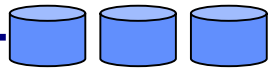


- **Details zum Anforderungsdokument:**
  - Ist oft ein offizielles Dokument und Teil der Vertragsgrundlage für ein Softwareprojekt.
  - Unabhängig von den Analysemethoden und der Modellierung wird das Ergebnis immer in einer Dokumentation zusammengefasst.
  - Der Detailgrad des Dokuments ist abhängig von der Art des zu entwickelnden Systems und welcher Entwicklungsprozess dazu verwendet wird.
    - Kritische Systeme benötigen eine detailliertere Anforderungen.
  - Anforderungsbeschreibungen folgen typischerweise einer definierten Vorlage.
    - Ein möglicher allerdings sehr generischer Aufbau eines solchen Dokuments wird durch den IEEE Standard für Anforderungsdokumente vorgegeben und enthält beispielsweise eine Definition der Benutzeranforderungen und Systemanforderungen, die Systemarchitektur, Anforderungslevel und Modelle des Systems.
    - Eine komplette Vorlage für ein solches Dokument haben wir auf Moodle zur Verfügung gestellt.

2.4 Anforderungsanalyseprozess: Anforderungsspezifikation 3



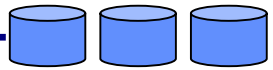
- **Anforderungslevel:** Aufgrund des offiziellen Charakters der Anforderungsbeschreibung werden häufig die nach dem RFC2119 definierten und immer großgeschriebenen Schlüsselwörter für das Anforderungslevel verwendet:
  - MUSS (MUST): Absolute Festlegung einer Anforderung
  - DARF NICHT (MUST NOT): Absoluter Ausschluss einer Anforderung
  - SOLL (SHOULD): Empfehlung für eine Anforderung
    - Abweichung zu diesen Festlegungen sind in begründeten Fällen möglich.
  - SOLL NICHT (SHOULD NOT): Empfehlung, eine Anforderung auszuschließen.
    - Abweichungen sind in begründeten Fällen möglich.
  - KANN (MAY): Die Anforderung ist optional



Anforderungsanalyseprozess

ANFORDERUNGSVALIDIERUNG

2.4 Anforderungsanalyseprozess: Anforderungsvalidierung 1

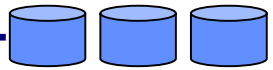


- **Anforderungsvalidierung (requirements validation):** Überprüfen, ob die Anforderungen wirklich das System definieren, welches die Stakeholder möchten und benötigen.
  - Dieser Schritt überlappt sich mit der Erhebung und Analyse da dabei ebenfalls mögliche Fehler in den Anforderungen gefunden werden sollen.
  - Die Anforderungsvalidierung ist von entscheidender Bedeutung, denn Fehler in der Anforderungsdokumentation können teuer werden, wenn diese erst während der Implementierung oder dem Betrieb gefunden werden und nicht vorher durch die Validierung abgefangen werden.
    - **Merksatz**: Eine schlechte/falsche/widersprüchliche Anforderungen während der Implementierung auszubügeln kostet im Durchschnitt das Hundertfache einer Korrektur während der Anforderungsvalidierung.
  - Während der Anforderungsvalidierung sollten unterschiedliche Arten von Überprüfungen durchgeführt werden wie:
    - Validität, Konsistenz, Vollständigkeit, Realisierbarkeit und Überprüfbarkeit.
  - Außerdem kommen eine Reihe von Validierungstechniken zum Einsatz:
    - Anforderungsreviews, Prototyping und die Generierung von Testfällen.

2.4 Anforderungsanalyseprozess: Anforderungsvalidierung 2

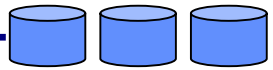
- **Anforderungsvalidierung: Was wird geprüft?**
  - **Validität (validity checks):** Erfüllen die Anforderungen die realen Bedürfnisse der Benutzer des Systems.
    - Notwendig, da sich Benutzeranforderungen seit der Erhebung geändert haben können. Diese Checks sollten in regelmäßigen Intervallen erfolgen.
  - **Konsistenz (consistency checks):** Anforderungen sollen keine Konflikte oder Widersprüche enthalten.
  - **Vollständigkeit (completeness checks):** Das Anforderungsdokument sollte alle Anforderungen bezüglich der Funktionen und deren Bedingung enthalten.
  - **Realisierbarkeit (realism checks):** Können die Anforderungen in der gewünschten Zeit, Budget, Ressourcen umgesetzt werden?
  - **Überprüfbarkeit (verifiability):** Systemanforderungen sollten immer so geschrieben werden, dass sie überprüfbar sind. Das heißt es muss möglich sein (automatische) Tests zu schreiben, die demonstrieren, dass das System die spezifizierten Systemanforderungen erfüllt.

2.4 Anforderungsanalyseprozess: Anforderungsvalidierung 3

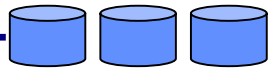


- **Anforderungsvalidierung: Wie wird geprüft?**
  - **Anforderungsreviews (requirement reviews):** Anforderungen werden systematisch analysiert durch ein Reviewer-Team, welches die Anforderungen auf Fehler und Inkonsistenzen überprüft.
  - **Prototyping:** Beinhaltet die Entwicklung eines ausführbaren Modells des Systems und das Testen dieses Modells mit Endanwendern und Kunden um zu prüfen, ob das System deren Bedürfnisse und Erwartungen erfüllt.
    - Stakeholder experimentieren mit dem System und geben Änderungswünsche und Vorschläge an das Entwicklungsteam weiter.
  - **Generierung von Testfällen (test-case generation):** Anforderungen müssen testbar und messbar sein. Falls es schwierig bis unmöglich ist einen Test für eine Anforderung zu erstellen bedeutet dies meistens auch, dass es schwierig (unmöglich) wird die Anforderung zu implementieren. Diese Anforderung sollte daher überdacht/überarbeitet werden.
    - Die Entwicklung von Testfällen, basierend auf den Anforderungen, ist der Grundgedanke von test-driven-development. Weitere Informationen hierzu erhalten Sie in den kommenden Blöcken dieser LV.

Inhalt



2.5 Zusammenfassung



- **Vision und Scope müssen klar definiert und abgegrenzt sein.**
- **Alle relevanten Stakeholder müssen identifiziert und eingebunden werden.**
- **Die Machbarkeit muss vor dem formellen Projektbeginn sichergestellt sein und alle Komponenten müssen hinsichtlich „Make or Buy“ evaluiert werden.**
- **Es ist von Vorteil wenn die Architektur während der Anforderungsanalyse schon grob feststeht.**
- **Alle funktionalen, nichtfunktionalen und Domänenanforderungen müssen identifiziert und qualitativ beschrieben sein.**
- **Anforderungsbeschreibungen müssen für alle Stakeholder verständlich sein.**