Software Engineering

Rechtsvermerk:

Die Angabe/Verwendung der Quellen je Folie sind im Original PowerPoint-Dokument dieses Skripts im Notizenbereich vermerkt. In der PDF-Ausfertigung dieses Skripts sind die Quellen je Folie nicht ablesbar. Auf Anfrage werden die Quellen bereitgestellt.

Stiven Raso

Ausbildung:

1998-2005	**HS Niederrhein** – Studium Dipl. Wirt.-Ing.
2008-2010	**FOM** – Studium Master IT-Management
2010	**SAP Deutschland AG** – Master Thesis

Beruflicher Werdegang:

2005-heute	**Dr. Glinz COV*i*S GmbH** (135 Mitarbeiter) – Geschäftsführer

Lehraufträge:

seit 2011	**Lehrbeauftragter FOM** (Köln, Düsseldorf, Duisburg, Essen, Neuss) Software Engineering, Projekt- und IT-Management, E-Business, Digitalisierung
gelegentlich	**Lehrbeauftragter FHDW**(Mettmann, Bergisch Gladbach) Methoden im IT-Consulting, E-Commerce **Lehrbeauftragter EUFH** (Neuss) IT-Management

Veranstaltungskomponenten

- Seminaristischer Unterricht/Vortrag
- Beispiele aus der Unternehmenspraxis zur Verdeutlichung
- Übungen/Fallstudien zur Bearbeitung durch die Studenten
- Interaktivität: Diskussionen und aktive Beiträge sind erwünscht

Prüfung und Benotung

- Gemäß Vorgaben der Bildungseinrichtung

Kontakt

- bei Fragen: fom@stivenraso.de oder 0174-2358327

Skript (im Online Campus verlinkt)

- **User: Studfom // PW: IsbSRuw1A!**

Software Engineering

- Definitionen und Begriffsbestimmung
- Software Lebenszyklus
- Requirements Engineering
- Systemkontextanalyse
- Vorgehensmodelle
- Aufwandschätzverfahren
- Projektmanagement

Definition Software und Software Engineering

Software

Unter Software (SW) versteht man ein oder mehrere (Computer) Programm(e), die dazugehörigen Dokumentation (Systemdokumentation, Benutzerhandbuch…) und die Konfigurationsdateien, die zur Einrichtung verwendet werden, damit die Software richtig funktioniert.

Software Engineering

Das Software Engineering ist eine technische Disziplin, die sich mit allen Aspekten der Softwareherstellung beschäftigt, von den frühen Phasen der Systemspezifikation bis hin zu Wartung des Systems, nachdem sein Betrieb aufgenommen wurde.

Software Engineering nach Ian Sommerville

Definition Software und Software Engineering

Software Reengineering

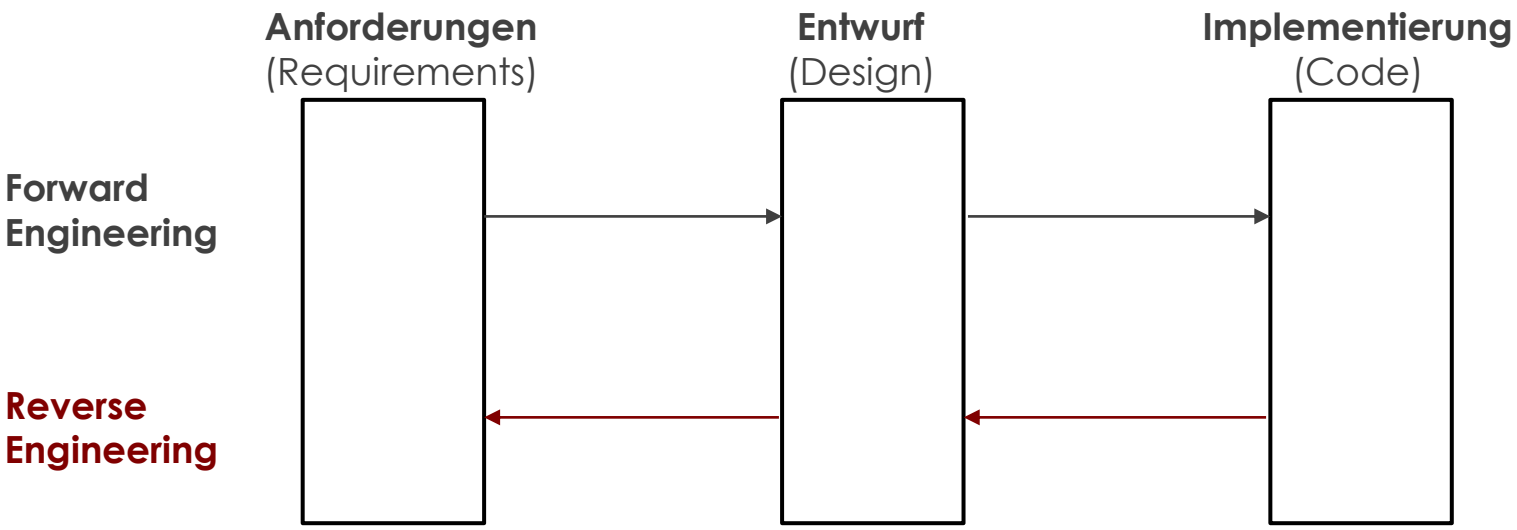
Analyse und Überarbeitung eines vorhandenen Softwaresystems mit dem Ziel, dessen Softwarequalität zu verbessern. Dabei kann es z.B. um die Restrukturierung von Programmcodes, die Gestaltung von Benutzeroberflächen unter softwareergonomischen Gesichtspunkten oder die Integration bisher isoliert geführter Anwendungen unter Beibehaltung der Funktionalität gehen.

Software Reverse Engineering

Kann als Teil bzw. Vorstufe des Software Engineerings verstanden werden. Hierbei geht es darum die Struktur und das Verhalten der Software nachzuvollziehen und ggf. fehlende Systemspezifikationen/-dokumentationen auf einem höheren Abstraktionsniveau zu erstellen

Software Re-Engineering nach Gabler Wirtschaftslexikon

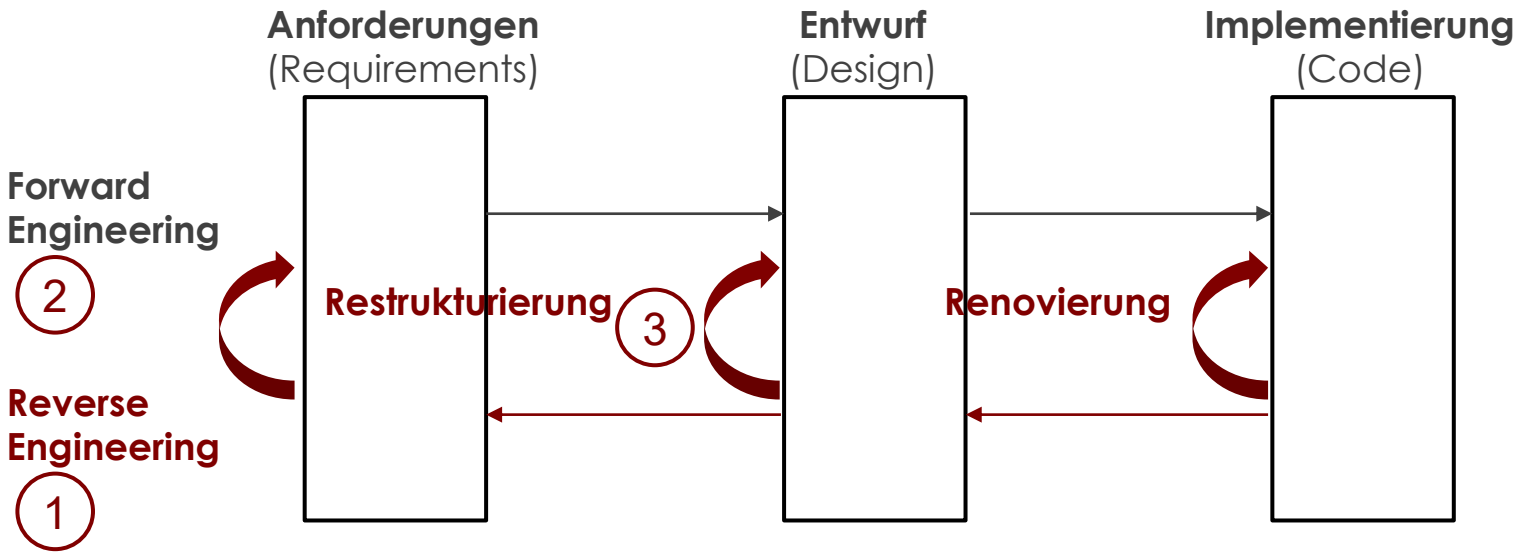
Prozess Software Engineering/Re-Engineering/Reverse Engineering



Identifikation der Systemkomponenten und deren Beziehungen mit dem Ziel eine Beschreibung des Systems (in einer anderen Form/auf einem höheren Abstraktionsniveau) zu erreichen

Nach Prof. Dr. Koschke

Prozess Software Engineering/Re-Engineering/Reverse Engineering

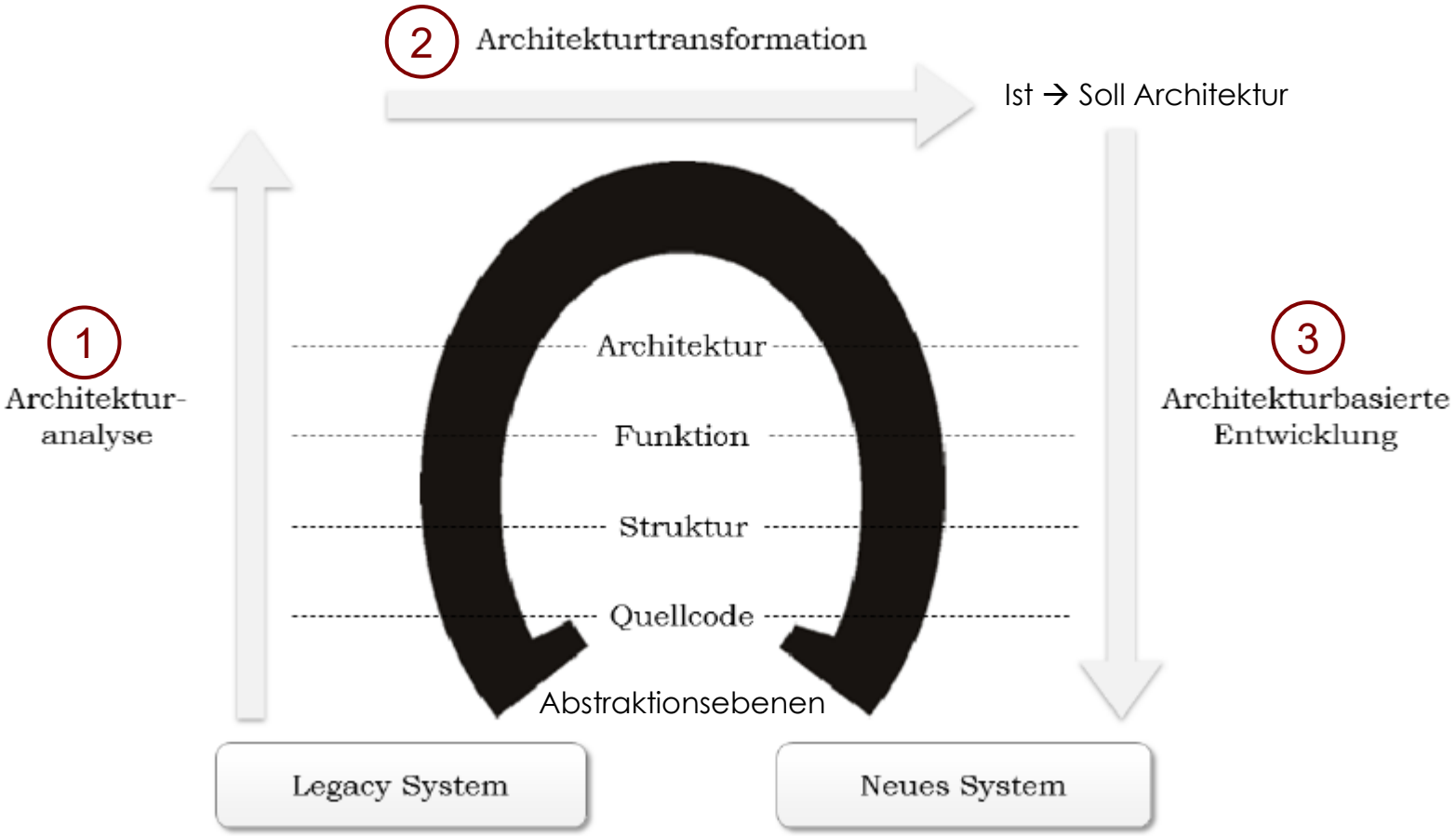


Reengineering Varianten

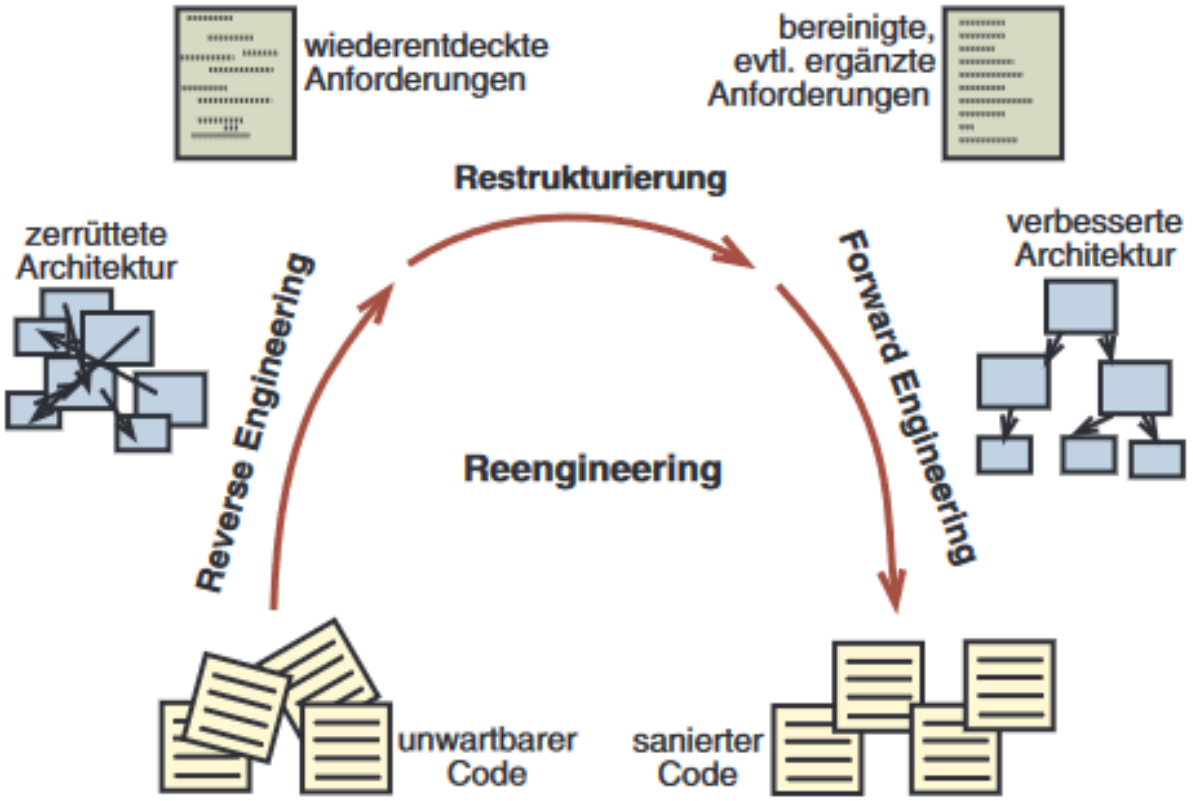
- **Reines Reengineering**
  - lediglich Restrukturierung
  - keine Ergänzung von Funktionalitäten
- **Erweitertes Reengineering**
  - Zunächst Systemanalyse und Strukturierung, anschließend folgt die Ergänzung neuer Funktionalitäten

angelehnt an Prof. Dr. Koschke

Prozess Software Reengineering – Horseshoe Modell



Prozess Software Reengineering – Horseshoe Modell



IT-Projekte sind gekennzeichnet durch…

- technisch anspruchsvolle Inhalte
- abstrakte/ nicht greifbare Anforderungen
- instabilen und volatilen Umfang (Scope)
- komplexe Zusammenhänge
- eine hohe Anzahl von Projektbeteiligten/Stakeholder
- schwer zu begründende Budgetforderungen
- hohen Zeitdruck → kurzer Time to Market
- Projektvorgehensmodell oftmals unklar/undefiniert

Allgemeine und übergreifende Herausforderungen in der Praxis

Immaterialität des Produkts: Software ist unsichtbar

- Ist für den Kunden schwer erlebbar (Kostenbewusstsein)
- Bzgl. des Fertigungsgrades schwer messbar
- Risiko: Umfang, Kosten und Laufzeit

Innovationsgrad des Produkts

- Mit dem Einsatz neuester Technologie verbunden
- Mit der Realisierung umfassender neuer Funktionen verbunden
- Risiko: Stabilität (Qualität) und Laufzeit

Mangelnde Prozessreife: SE ist eine junge Disziplin

- Es fehlen standardisierte und etablierte Entwicklungsprozesse
- Verständnis für die ingenieurmäßige SE (Kunst vs. Handwerk)
- Risiko: Qualität und Laufzeit

Herausforderung Effizienz vs. Flexibilität:

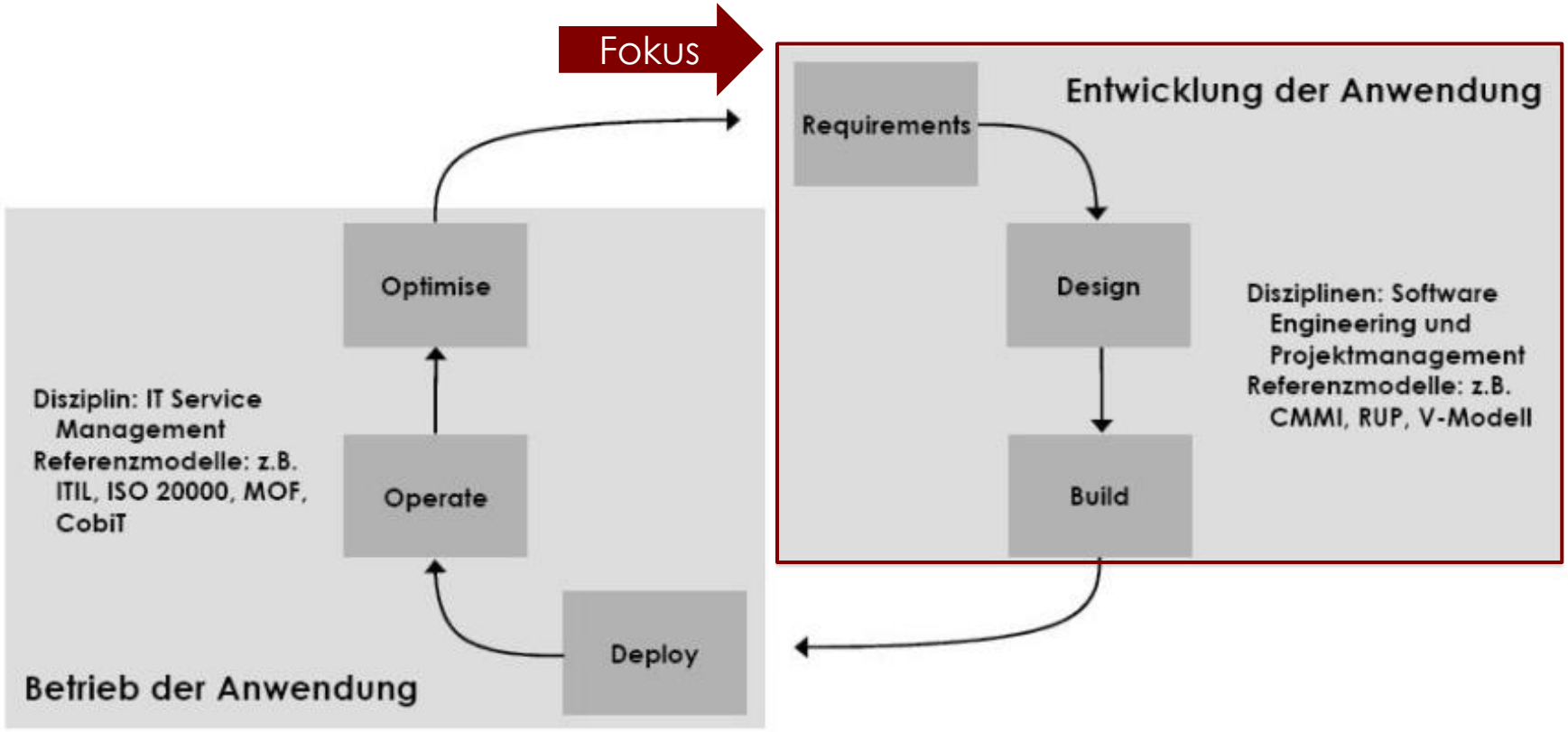
Effizienz: Einsatz bewährter Methoden zur Kostenreduktion

- Standardanwendungsdomäne
- Standardentwicklungsplattform
- Standardentwicklungsprozess
- Standardentwicklungsumgebung

Flexibilität: Das Unvorhersehbare muss planbar sein

- Änderung der Anforderungen durch den Kunden
- Änderung der Entwicklungsplattform durch das Management
- Mitarbeiterfluktuation
- Änderung des Auslieferterminsdurch das Management
- Verzögerungen durch die Zulieferer

Lebenszyklus einer Anwendung – Entwicklungspfad



Herausforderungen im Software Engineering Lifecycle

1. Voranalyse

- Geschäftsarchitektur
- Definition Anwendungsfälle (Use Cases)
- Greenfield / Brownfield
- Individualentwicklung / Standardsoftware

1. Phase Requirements Engineering

- Produktvision
- Anforderungserhebung/-management
- Systemkontextanalyse (Problemraum, Lösungsraum)
- Definition eines MVP

2. Phase Design / Konzept

- Architekturdesign (monolithisch / Microservices / serviceorientiert…)
- Technologieauswahl Best-of-breed / Best-of-Suite

Herausforderungen im Software Engineering Lifecycle

3. Phase Development / Umsetzung

- Vorgehensmodell und Rollenverständnis (klassisch / agil)
- Kapazitätsplanung und -Bereitstellung
- Aufwandsschätzverfahren
- Codier Richtlinien und Qualitätssicherung
- DevOps Lifecycle (CI/CD, IaC/IaaS)
- Projektmanagement

4. Phase Rollout und Betrieb

- Datenmigration / Datentransformation
- Rolloutstrategie Big Bang / Co-Existenz
- Softwarewartung (korrektiv / adaptiv / perfektionierend)

5. Phase

- Weiterentwicklung (Continuous improvement)

Software Engineering – weiterführende Gedanken

Software Engineering ist die Disziplin, die sich damit beschäftigt, „bessere“ Software herstellen zu können

„Bessere“ Software bedeutet dabei
- sicherer
- zuverlässiger
- benutzbarer
- performanter

und „besser“ herstellen
- billiger
- schneller
- planbarer

Qualitätsmerkmale an Software

Funktionalität

- Beschreibt den Erfüllungsgrad der Spezifikation
- Das vereinbarte bzw. erwartete Ergebnis muss durch die Applikation geliefert werden
- Zur Funktionalität zählen auch Merkmale wie
  - Einhaltung von Authentifizierungsrichtlinien
  - Abbildung gesetzlicher Bestimmungen
  - Interoperabilität mit anderen Umsystemen

Zuverlässigkeit

- Beschreibt die Aufrechterhaltung eines bestimmten Leistungsniveaus unter festgelegten Bedingungen in einem definierten Zeitraum
- Weiteres Merkmal ist die Fehlertoleranz, d.h. das geforderte Leistungsniveau bei fehlerhafter Bedienung durch den Anwender oder Fehlerverhalten von Schnittstellen aufrechtzuerhalten

Qualitätsmerkmale an Software

Benutzbarkeit

- Wird als Software-Ergonomie bezeichnet
- Merkmal ist der Grad der Intuitivität einer Anwendung
- Beschreibt die Bedienbarkeit der Anwendung, also die Frage nach der Unterstützung des Anwenders durch die Applikation (z.B. geführte Prozesse)
- Umfasst ebenfalls das GUI- und Navigationskonzept sowie die Informationsarchitektur

Effizienz

- Meint das Verhältnis zwischen dem vereinbarten Leistungsniveau und den benötigten Betriebsmitteln, um dieses zu erreichen
- Die Software-Performance und die notwendigen – meist hardwaretechnischen – Ressourcen spielen hierbei eine wesentliche Rolle
- Weitere Punkte: Belegung des Arbeitsspeichers, der CPU…

Qualitätsmerkmale an Software

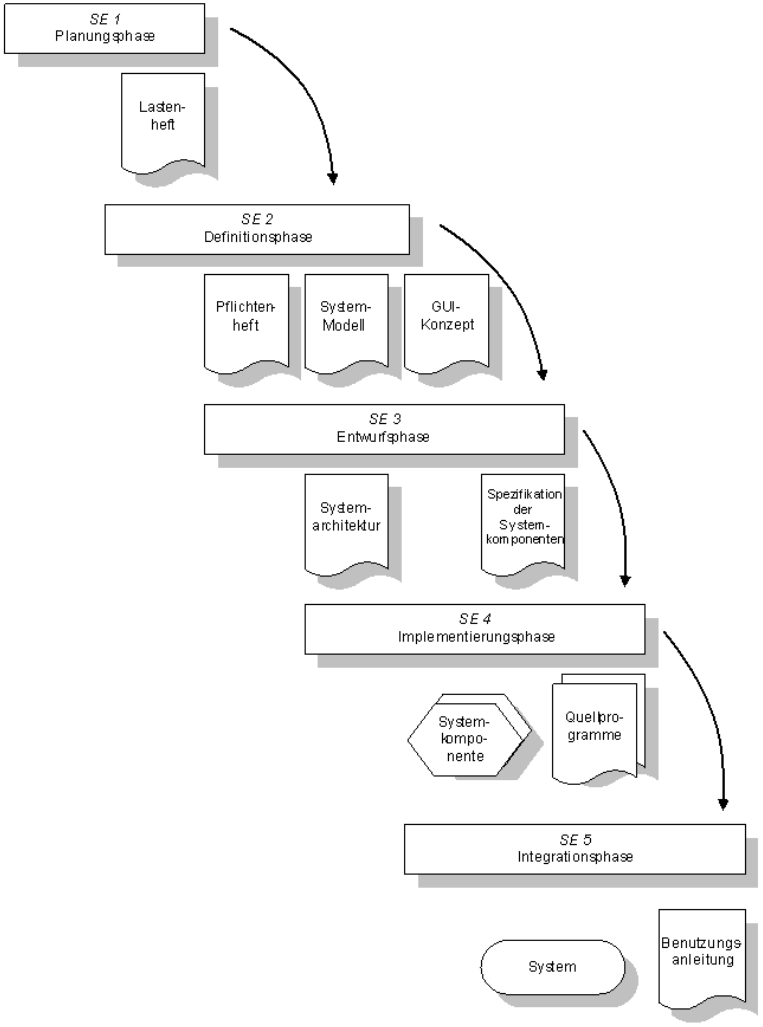
Wartbarkeit

- Wichtige Merkmale hierfür sind die Analysierbarkeit und Modifizierbarkeit
- Die Grundlagen hierfür werden durch „sauberen Code“ in der Entwicklung gelegt
- Meint ebenfalls den Aufwand, der für Fehlerbehebungen und Erweiterungen der Software anfällt

Übertragbarkeit

- Beschreibt die Fähigkeit, die Software in unterschiedlich spezifizierten Umgebungen verwenden zu können, z. B. verschiedene Betriebssysteme, Netzwerke o. ä.

Der Prozess der Systemerstellung wie er gemeint ist…

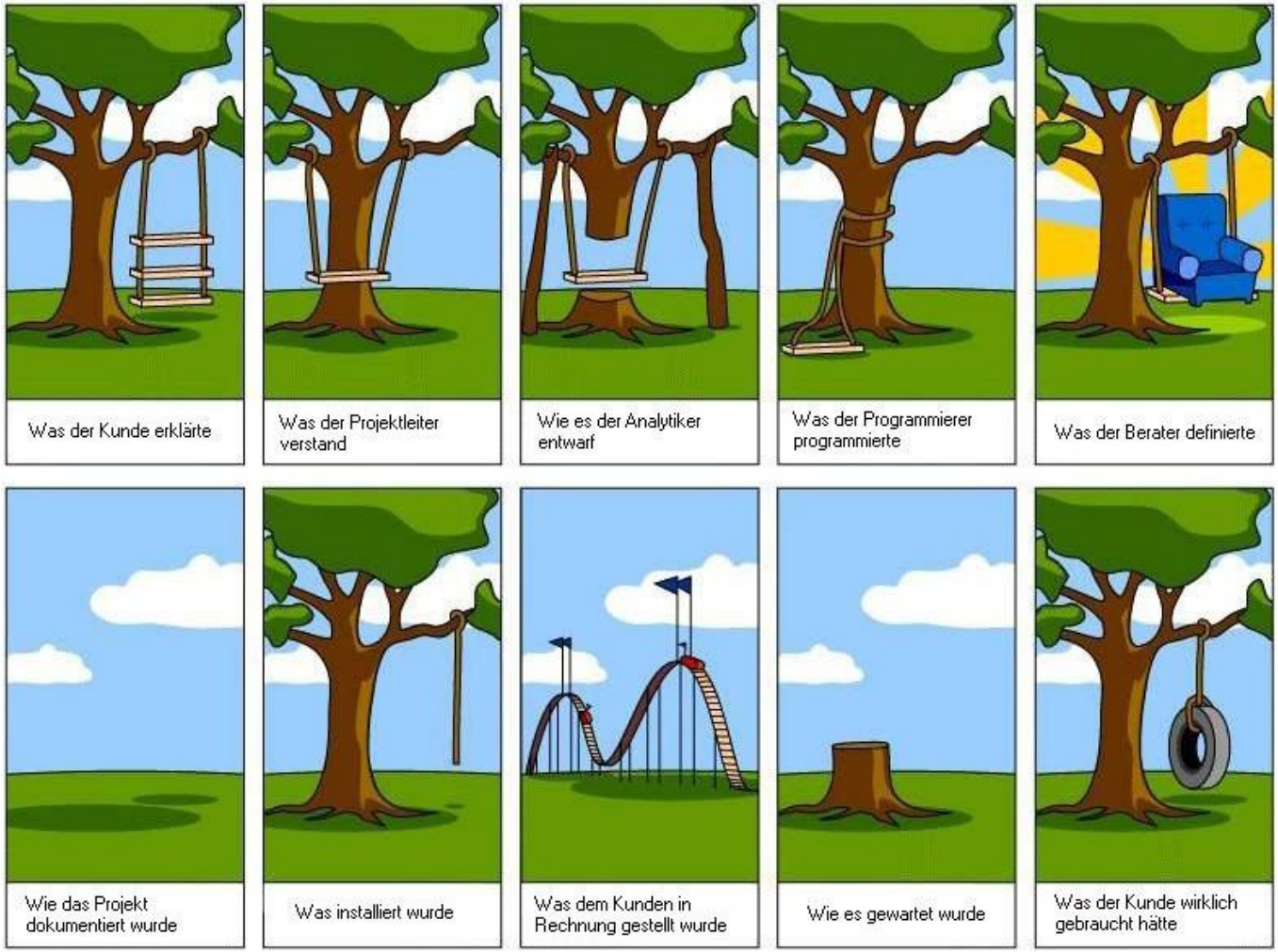


1. Planung des Projekts
2. Anforderungserhebung
3. Technischer Lösungsansatz
4. Erstellen eines Entwurfs
5. Entwicklung der Software
6. Integration und Testing/QS
7. Inbetriebnahme und Wartung
8. Aufnahme neuer Anforderungen

9.1 Change Req. → Fortsetzung bei 1.

9.2. Deinvestition/Rückbau

…und wie es meistens endet…



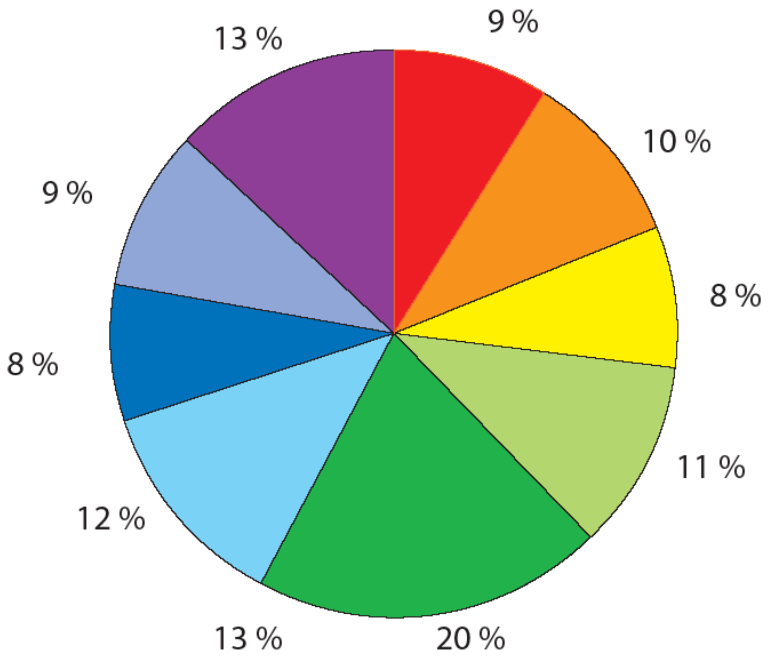
Fazit: Eine Verkettung ungünstiger Ereignisse und Aktivitäten führt zu mangelhaften Projektergebnissen

42 % der gescheiterten Projekte haben direkt mit dem Anforderungsmanagement zu tun

- Unvollständige Anforderungen 13 %
- Änderung der Anforderungen 9 %
- Nicht mehr benötigte Features 8 %
- Unzureichende Partizipation der Anwender 12 %

58 % indirekt

- Fehlende Managementunterstützung 9 %
- Unrealistische Erwartungen 10 %
- Mangelhafte Planung 8 %
- Fehlende Ressourcen 11%
- Sonstige 20 %



Quelle: Standish Chaos Report

Ursache-Wirkungsbeziehung

Ursachen:

- Mangelnde Identifizierung des tatsächlichen Bedarfs
- Unausgeprägtes Verständnis der Realisierungsmöglichkeiten (was ist sinnvoll/nutzenstiftend?)
- Nicht ausreichend formulierte und spezifizierte Anforderungen
- Unvollständiges Lastenheft
- Fehlende Kommunikation/Abstimmung der Projektteilnehmer
- Vernachlässigung wichtiger Projektphasen zugunsten *Zeit*, *Budget* aber auf Kosten der *Qualität*

Wirkung/Konsequenzen:

- Unvollständiger Anforderungskatalog
- Keine Priorisierung der „Must haves“ ggü. „Nice to haves“
- Aufwandsschätzung und Terminplanung schwer möglich
- Erschwerte Angebotserstellung
- Fehlende Konsolidierung der Anforderungen
- Unvollständiges Pflichtenheft (unzureichende Realisierungsgrundlage)
- Kein effizientes Projektmanagement und Steuern des Projekts möglich

Anforderungen sind generell schwierig zu ermitteln

- Interessensbeteiligte können ihre Anforderungen oft nicht ausdrücken
  - Schwierigkeit, Anforderungen mit Worten zu beschreiben
  - Unrealistische Erwartungen
  - Fehlendes Kostenbewusstsein
- Fachsprache der Interessensbeteiligten
- Implizites und unbewusstes Wissen der Interessensbeteiligten
- Unterschiedliche Interessensbeteiligte haben unterschiedliche Anforderungen und drücken diese unterschiedlich aus
- Einfluss politischer Faktoren
  - Manager äußern spezifische Anforderungen, um ihren Einfluss zu vergrößern
  - Rascher Wandel und geringe Stabilität von Anforderungen

Definition eines Requirements / einer Anforderung

1. Eine Bedingung oder Fähigkeit, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird
2. Eine Bedingung oder Fähigkeit, die ein System oder Teilsystem erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen

Übersetzt aus [IEEE]

Anspruch an die Requirements/ Anforderungen

- Vollständig
- Widerspruchsfrei
- Redundanzfrei
- Klar/eindeutig formuliert
- Mit allen Teilnehmern abgestimmt

Man unterscheidet:

- Benutzer- und Systemanforderungen
- Funktionale und nichtfunktionale Anforderungen

Zielsetzung bei der Erfassung von Anforderungen

Die an das zu entwickelnde System gestellten Anforderungen möglichst vollständig und fehlerfrei aufnehmen, damit die zu entwickelnde Lösung zum Kundennutzen entwickelt wird

Hauptaktivitäten

- Anforderungsquellen ermitteln
- Anforderungen aufnehmen
- Anforderungen kategorisieren
- Anforderungen auf Vollständigkeit prüfen
- Anforderungen zur Dokumentation vorbereiten

Erfassung von Anforderungen

- Ergebnisse aus Systemkontextanalyse
- Stakeholder Analyse
- Workshops & Interviews
- Kano-Modell
- Mind Mapping
- Use-Case-Modellierung
- Prototypen
- Reverse Engineering

Quellen für Anforderungen

- Stakeholder
  - Involvierte Benutzergruppen
- Existierende Dokumentation
  - Gesetze, Normen
  - Branchen-/organisationsspezifische Dokumentation
- System im Betrieb
  - Legacy-/ Vorgängersystem
- Konkurrenzprodukte
- Strategie Richtlinien

Anforderungsdokumentation

Qualitätskriterien für Anforderungen (gem. IEEE-Standard 830)

- Abgestimmt
  - Zwischen allen Beteiligten (Stakeholder, IT, Betrieb…)
- Bewertet
  - Wichtigkeit/Priorisiert
  - Auswirkungen/Risiken
- Eindeutig
  - Interpretationsfrei
- Gültig und aktuell
  - Muss in den aktuellen Systemkontext passen
  - z. B. müssen alle aktuellen Schnittstellen beachtet werden

Anforderungsdokumentation

Qualitätskriterien für Anforderungen (gem. IEEE-Standard 830)

- Korrekt
  - Lösungsanforderung muss wiederspiegeln was Anforderungssteller erwartet (nicht mehr oder weniger)
- Konsistent
  - Unabhängig vom Abstraktionsgrad müssen die Anforderungen widerspruchs- und konfliktfrei zu/ggü. anderen Anforderungen sein
- Prüfbar
  - Anforderung muss testbar/validierbar sein
- Realisierbar
  - Technisch/finanziell/organisatorisch umsetzbar

Anforderungsdokumentation

Qualitätskriterien für Anforderungen (gem. IEEE-Standard 830)

- Vollständig
  - Die geforderte und zu liefernde Funktionalität muss inhaltlich vollständig beschrieben werden
  - Unvollständige Anforderungen werden gekennzeichnet, müssen einen Realisierungsbeginn aber nicht zwangsläufig verhindern
- Verständlich
  - Es müssen Notationen und Formulierungen gewählt werden, die auch für alle Stakeholder nachvollziehbar sind, um einen Konsens/gemeinsames Verständnis bei der Anforderungsabstimmung zu erzielen

Requirements Engineering - Definition

Das Requirements-Engineering ist ein kooperativer, iterativer, inkrementeller Prozess, dessen Ziel es ist zu gewährleisten, dass:

- alle relevanten Anforderungen bekannt und in dem erforderlichen Detailierungsgrad verstanden sind
- die involvierten Stakeholder eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen
- alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind

Requirements Engineering – Hauptaufgaben

- **Ermitteln:**
  - Anforderungen ermitteln
- **Dokumentieren:**
  - Anforderungen adäquat beschreiben
- **Prüfen und abstimmen:**
  - Anforderungen müssen geprüft und abgestimmt werden
  - Wichtig für die Qualitätssicherung
- **Verwalten:**
  - Anforderungen müssen verwaltet werden
  - Aufbereiten für unterschiedliche Rollen
  - Änderungen sind nachzuverfolgen

Anforderungsdokumentation

Arten der Anforderungsdokumentation

- Natürlichsprachlich – häufigste Art der Dokumentation
- Vorteil:
  - Keine Voraussetzungen für das Lesen/Schreiben notwendig
- Nachteil:
  - Sprache unterliegt Mehrdeutigkeiten und birgt damit Interpretationsspielräume
  - Unterschiedliche Detailtiefe
  - Unpräzise Ausdrucksmöglichkeiten
  - Keine einheitliche Notation, Qualität ist stark abhängig vom Anforderungssteller

Requirements Engineering – Arten der Dokumentation

Natürlichsprachige Anforderungen

REQ001: Der Kunde gibt eine Bestellung auf

REQ002: Der Kunde zahlt für die Bestellung

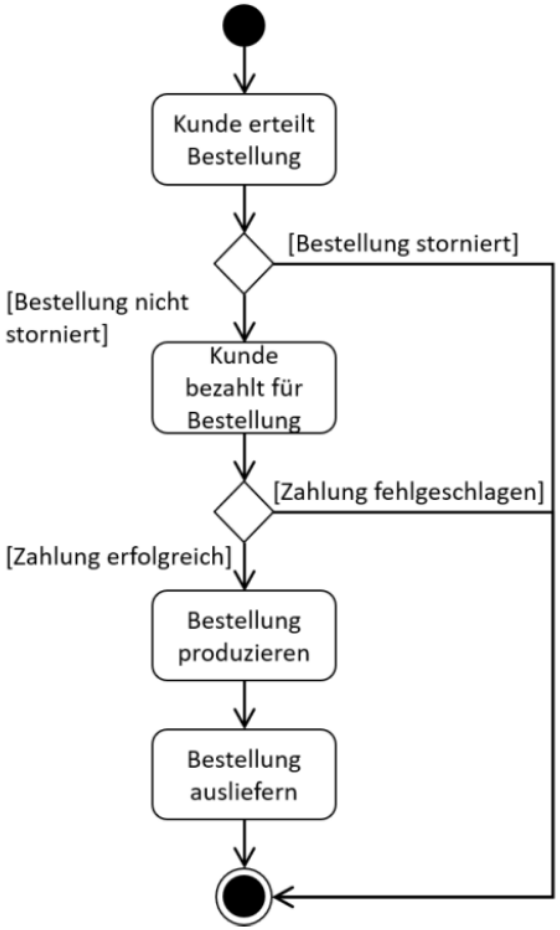
REQ003: Wenn die Zahlung erfolgreich war, dann wird die Bestellung produziert

REQ004: Wenn die Zahlung fehlgeschlagen ist, wird die Bestellung storniert

REQ005: Nachdem die Bestellung produziert wurde, wird sie an den Kunden ausgeliefert

REQ006: Die Bestellung kann storniert werden, wenn der Kunde sie nicht bezahlt

Modellierte Anforderungen



Quelle: Handbuch für das CPRE Foundation Level nach dem IREB-Standard, S. 44

Requirements Engineering – Vorteile modellierter Anforderungen

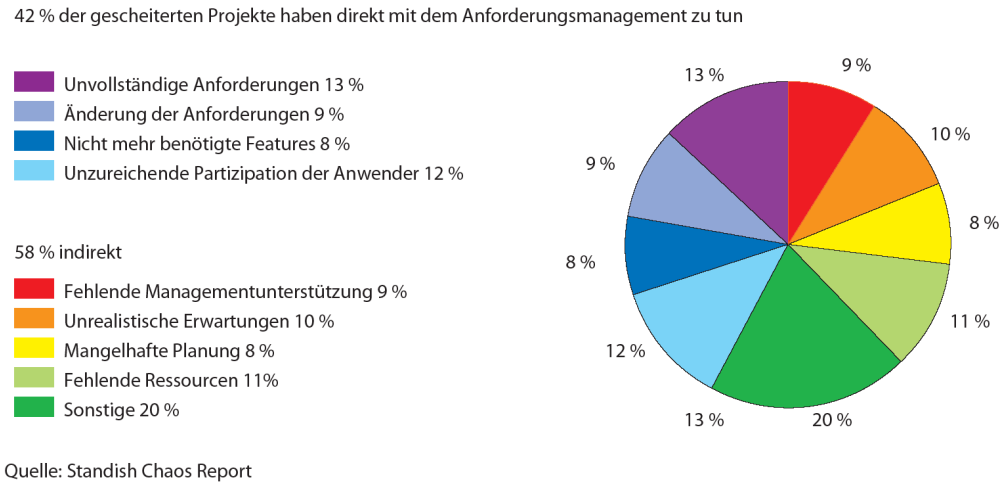
- Die Elemente und ihre Verbindungen sind leichter zu verstehen und zu merken.
- Da ein Modell einen spezifischen Zweck und eine reduzierte Informationsmenge hat, kann das Verständnis der modellierten Realität weniger Aufwand erfordern.
- Modellierungssprachen für Anforderungen haben eine eingeschränkte Syntax, die mögliche Mehrdeutigkeiten und Auslassungen reduziert.
- Modellierungssprache (Syntax und Semantik) ist einfacher - d.h. eine begrenzte Anzahl von Notationselementen und strengere Sprachregeln im Vergleich zur natürlichen Sprache

Quelle: Handbuch für das CPRE Foundation Level nach dem IREB-Standard, S. 44

Bedeutung der Analysephase im Requirement Engineering

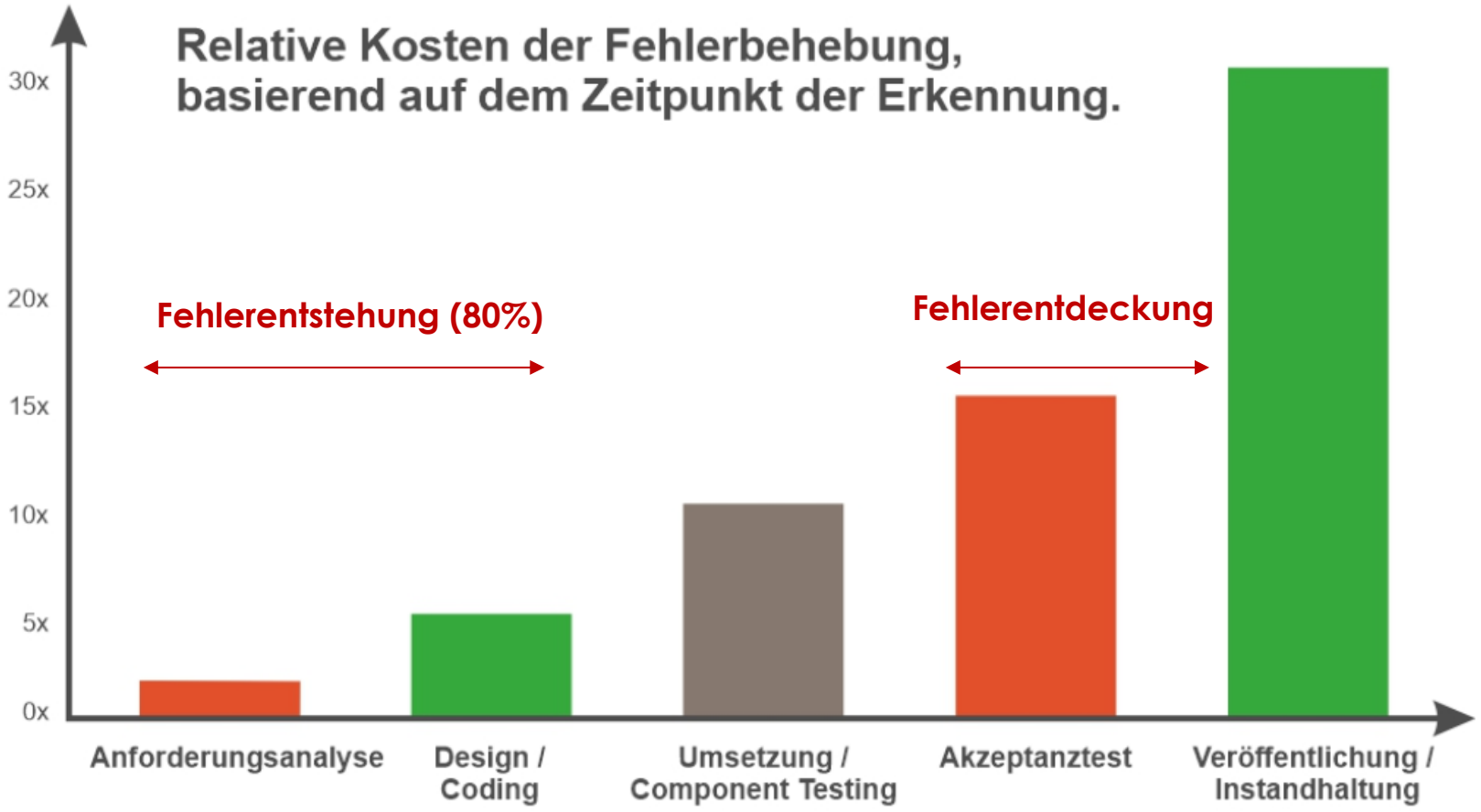
- 60% der Softwareprojekte scheitern durch Analysefehler
- 50% der Ausfälle im industriellen Sektor sind auf Software-Fehler zurückzuführen
- Grundsatz: Frühzeitige Fehlererkennung und –behebung spart Zeit, Kosten und erhöht die Qualität
- Die Analysephase kann in
  - Planungsphase und
  - Definitionsphase

  unterteilt werden



Quelle: Standish Chaos Report

Bedeutung der Analysephase im Requirement Engineering



Quelle

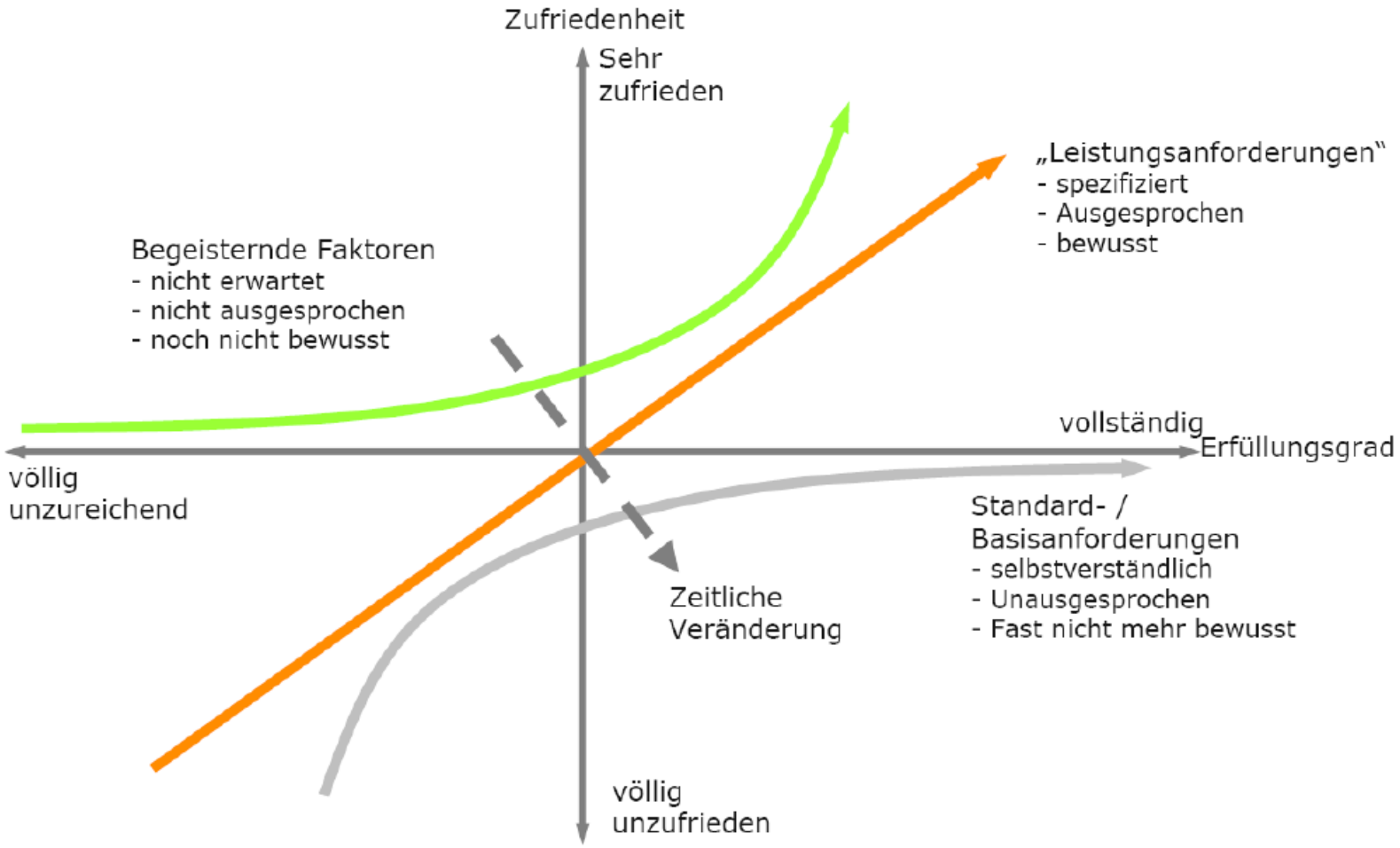
Ein wesentliches Ziel im Requirement Engineering ist die Erhebung und Kategorisierung von Anforderungen (bspw. nach dem Kano-Modell

- **Basisfaktoren:** Selbstverständlich vorausgesetzte Systemmerkmale (unterbewusstes Wissen). Diese Merkmale sind vollständig zu erfüllen, andernfalls droht eine massive Unzufriedenheit. Vollständig erfüllte Basisfaktoren erzeugen allerdings auch keine positive Stimmung

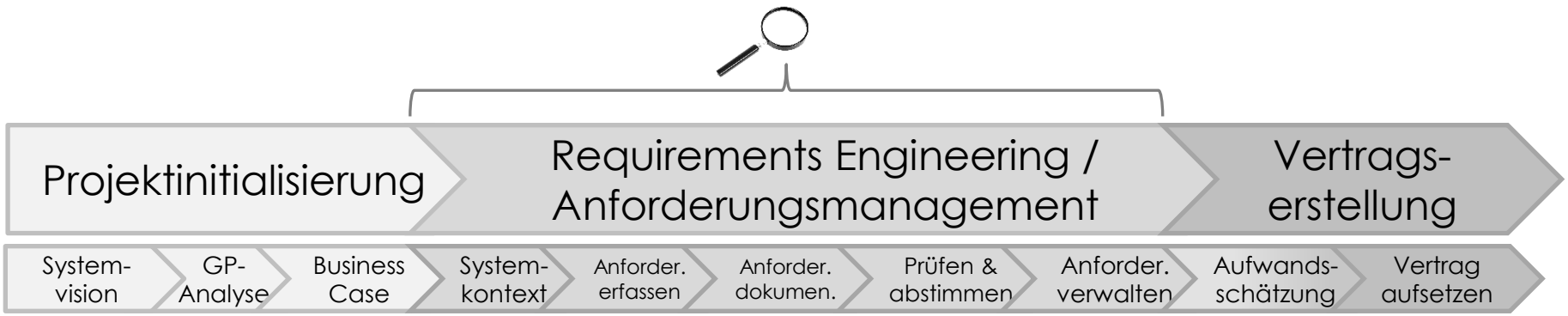
- **Leistungsfaktoren:** Explizit geforderte Systemmerkmale (bewusstes Wissen). Die Erfüllung erzeugt Zufriedenheit. Fehlende Leistungsfaktoren vermindern die Akzeptanz beim Endanwender. Ermittlung erfolgt häufig anhand von Befragungen

- **Begeisterungsfaktoren:** Resultieren aus Vorschlägen der Anforderungsanalysten, sodass der Anwender den Wert erst erkennt, wenn er diese ausprobieren kann (unbewusste Anforderungen)

Anforderungskategorisierung nach dem Kano-Modell



Einordnung des Requirements Engineerings



Die Ergebnisse des Requirements Engineerings dienen als Input für / zur

- die Aufwandsschätzung / Budgetierung des Projekts
- Vertragsgestaltung
- Projekt- und Zeitplanung
- Ressourcen- und Kapazitätsplanung
- Änderungsanforderungen
- Einschätzung / Früherkennung möglicher Risikofaktoren und Abhängigkeiten
- das gemeinsame Verständnis (Stakeholder, Anforderungssteller, Auftraggeber/ -nehmer)
- Test- und Anwendungsfälle
- Qualitätssicherung
- Fortschrittsreporting / Projektcontrolling
- den späteren Betrieb der Anwendung (z.B. nichtfunktionale Anforderungen)

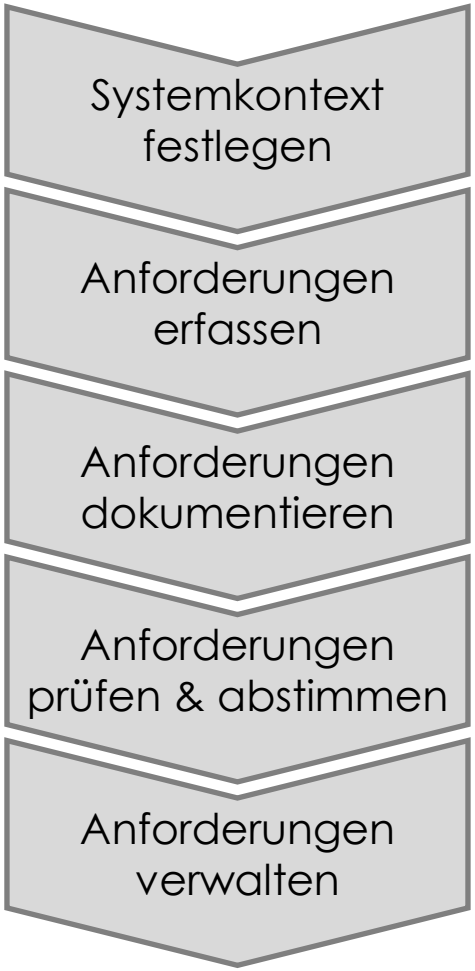
Ziele und Schwerpunkte der Analysephase im Requirements Engineering

- Ziel: Was soll realisiert werden? → Systemvision
  - Entscheidung, ob das Produkt entwickelt/ Projekt durchgeführt wird
  - Anforderungserstellung/ bewertung (ggf. Pflichtenheft)
- Aktionen:
  - Analyse des Ist-Zustands
  - Analyse der Machbarkeit
    - Risiken
    - Kosten
  - Bestimmung der Requirements/ Anforderungen
  - Definition des Systemkontexts
    - Einsatzfelder der Applikation (Geschäftszwecks)

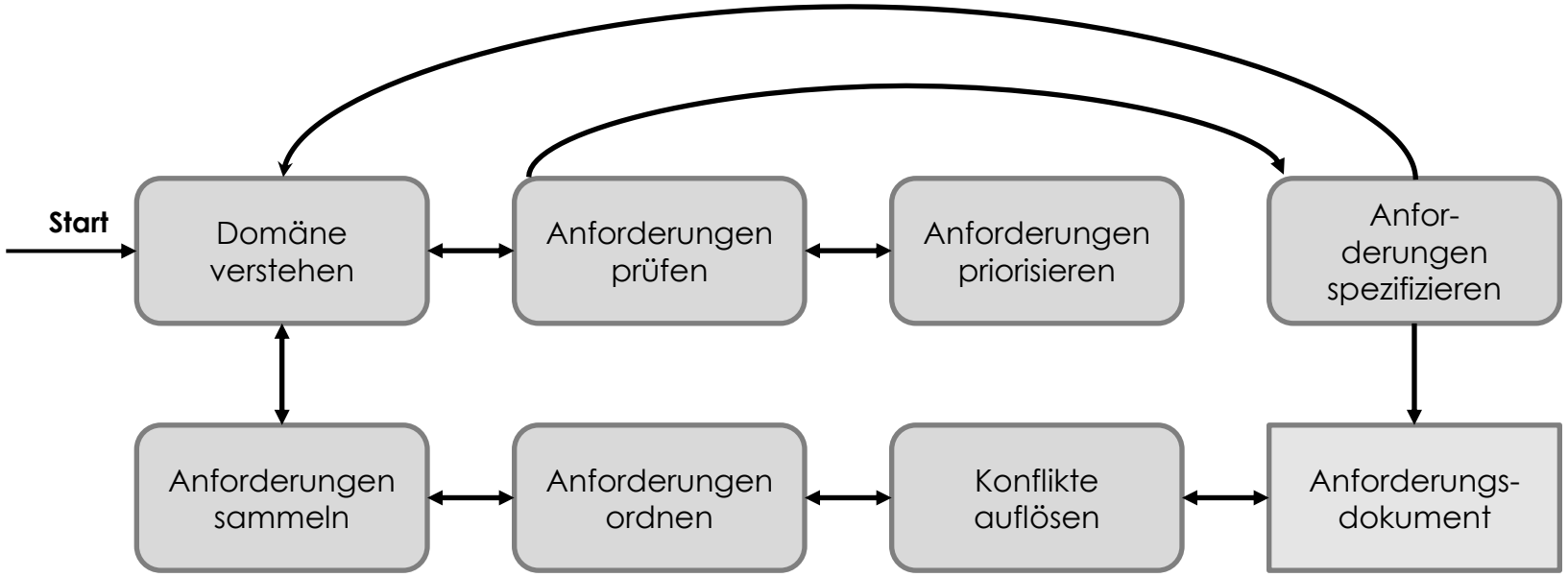
Systemvision

- Sicherstellen, dass das geplante System einen Beitrag zur Erreichung der Unternehmensziele leistet
- Gemeinsames Verständnis der Anwendung für alle Beteiligten schaffen
- Definition des Geschäftswertes/ Mehrwertes, der für die Benutzer erzielt werden soll
- Voranalyse und sondieren der Anforderungen
  - Nutzstiftende Anforderungen aufnehmen
  - „Überflüssige“ Anforderungen (nice-to-have) eliminieren/ nach hinten priorisieren

Requirements Engineering – Hauptaktivitäten im Rahmen der Anforderungsanalyse



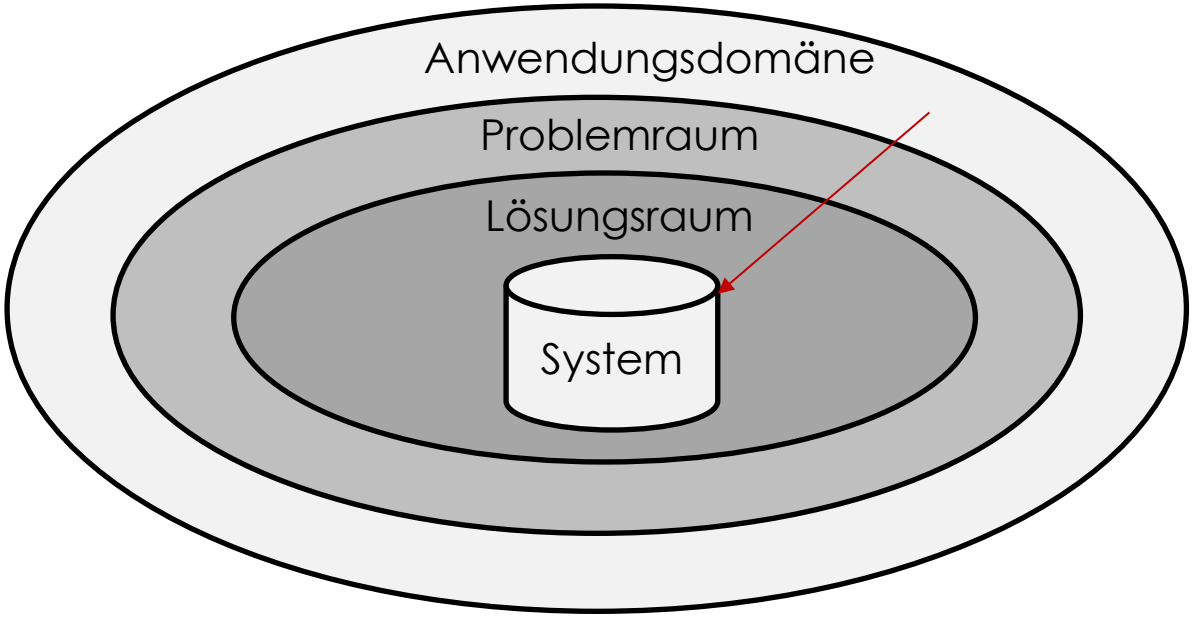
Requirements Engineering – Prozess der Anforderungsanalyse



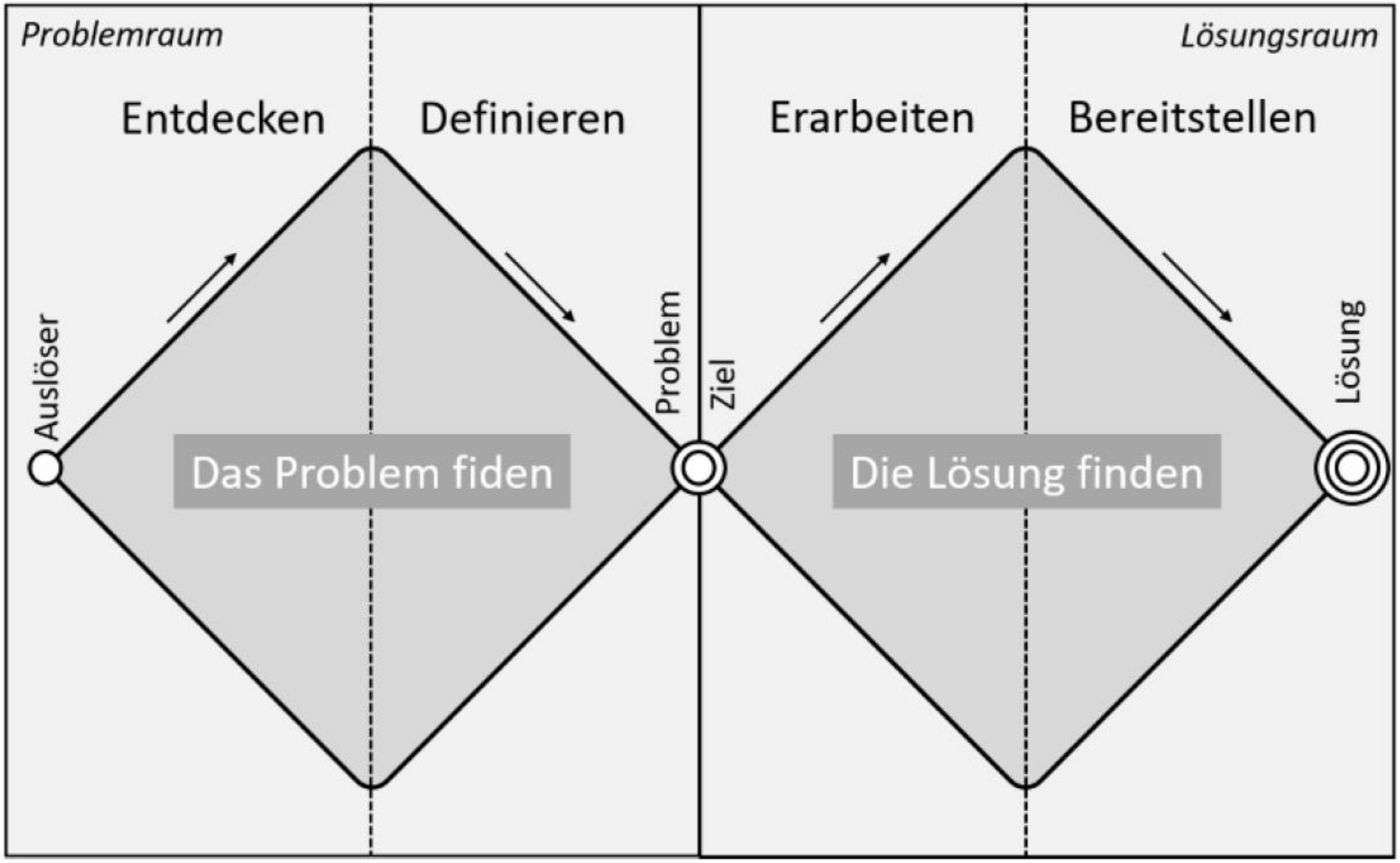
Requirements Engineering – Hauptaufgaben

- Ziel:
  - Abgrenzen des Systems von der Umgebung
  - Identifikation der für die Anforderungen relevanten Umgebungsteile
- Motivation:
  - „Der Ursprung aller Anforderungen liegt in dessen Umgebung“

Zwiebelprinzip – von außen nach innen



Probleme- und Lösungsraum in der Anwendungsdomäne



Quelle: Handbuch für das CPRE Foundation Level nach dem IREB-Standard, S. 95

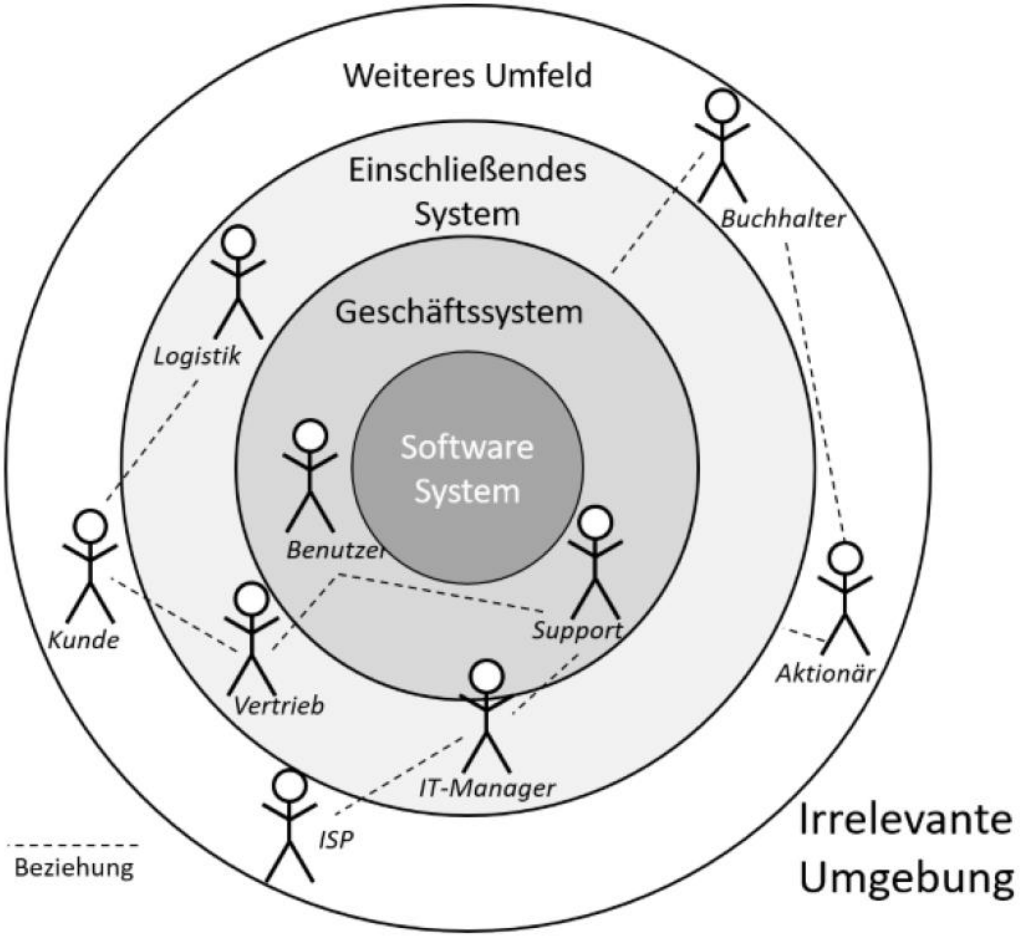
Probleme- und Lösungsraum in der Anwendungsdomäne

- **Anwendungsdomäne**
  - Bezeichnet den Anwendungsbereich der Software; Z.B. Applikation für Bedienung/ Verwaltung der Geldautomaten
- **Problemraum**
  - Innerhalb der Anwendungsdomäne treten häufig Probleme auf – technisch oder organisatorisch, bspw.
    - laufen Prozesse ineffizient ab
    - Entsprechend die Prozessergebnisse nicht den Erwartungen
    - Ist die IT-Unterstützung für den Prozess ungeeignet
  - Erkenntnisse im Problemraum können die Entscheidung für die Weiterentwicklung einer bestehenden Applikation oder Neuentwicklung/ Zukauf beeinflussen

Kontextanalyse

- Analyse der Geschäftsprozesse, die von der Einführung des Systems betroffen sind
- Analyse der bestehenden IT-Landschaft, in die das System integriert wird (siehe auch Systemkontext)
  - Welche bestehenden Schnittstellen sind betroffen/ erforderlich?
  - Gibt es laufende Prozeduren/ Aufbereitungsjobs, Daten Im-/ Exporte
  - Zu welchen Applikationen bestehen Abhängigkeiten
  - Handelt es sich um eine Neuentwicklung, Weiterentwicklung, Migration?
- Analyse der Organisationsstrukturen mit Rollen und Verantwortlichkeiten
  - Wer nutzt das System künftig
  - Wer betreibt und wartet die Applikation

Kontextanalyse: IT-Landschaft – Stakeholderanalyse



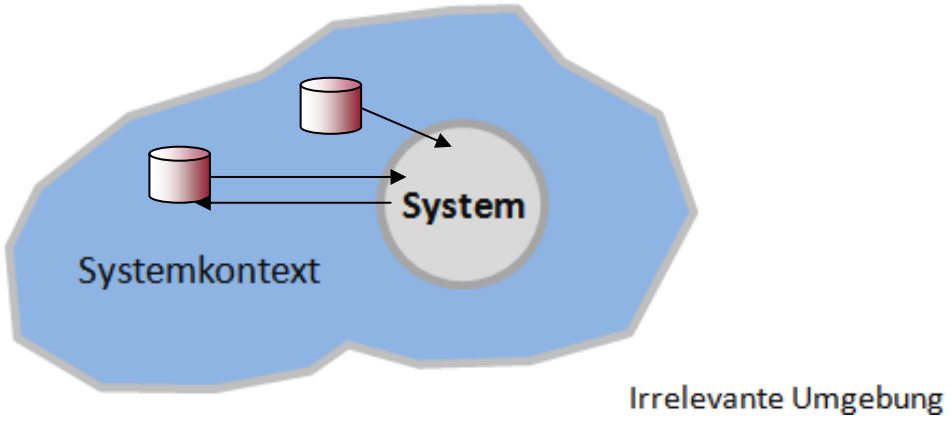
Quelle: Handbuch für das CPRE Foundation Level nach dem IREB-Standard, S. 75

Kontextanalyse – Geschäftsprozesse

- **Geschäftsprozessanalyse**
  - Schnittstelle der Betriebswirtschaft zur Informatik
  - Betrachtung der Prozesse, die einen Mehrwert liefern (Produktion eines Produkts, Bereitstellung/ Durchführung eines Dienstes)
  - Komplexe Geschäftsprozesse erstrecken sich oftmals über mehrere Organisationseinheiten mit unterschiedlichem Grad an IT-Unterstützung
  - Ändern sich Geschäftsziele und Strategien, wirken sich diese unmittelbar auf die Prozessabläufe und damit auf die IT-Systeme aus
  - Geschäftsprozesse werden auf Funktionalitäten in den unterstützenden IT-Systemen abgebildet
  - Analyse und Modellierung von Geschäftsprozessen erfolgt typischerweise auf Basis von eEPKs, BPMN oder UML

Kontextanalyse: IT-Landschaft – Systemkontext & Systemgrenze

- **Analyse der IT-Landschaft**
  - Der Systemkontext ist der Teil der Umgebung eines Systems, der für die Definition und das Verständnis der Anforderungen des betrachteten Systems relevant ist
  - Die Systemgrenze separiert das geplante System von seiner Umgebung. Sie grenzt den im Rahmen des Entwicklungsprozesses gestaltbaren und veränderbaren Teil der Realität von Aspekten in der Umgebung ab, die durch den Entwicklungsprozess nicht verändert werden können

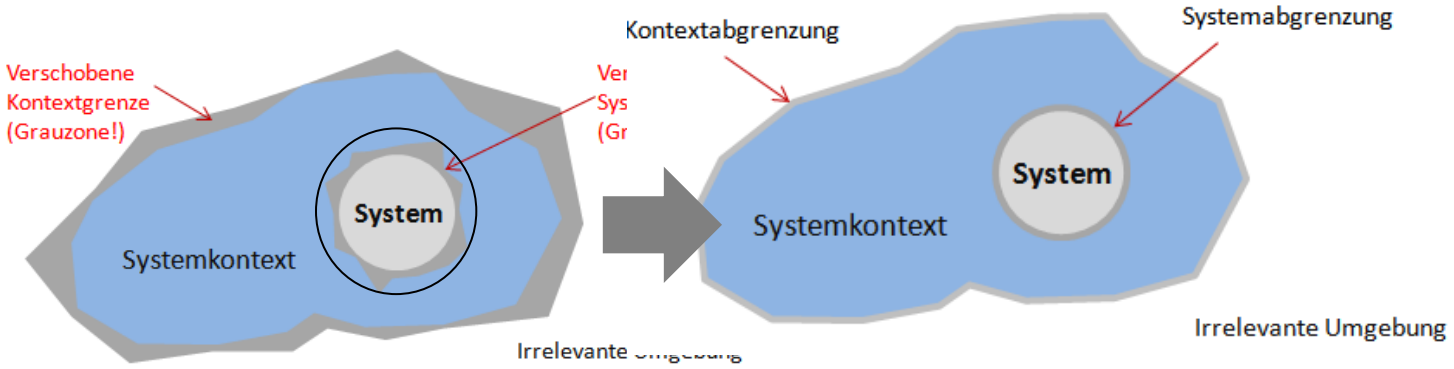


Kontextanalyse – Geschäftsprozesse

- Systemgrenzen sind zu Beginn eines Projekts oft „verwaschen“. Die Folge ist eine unklare Systemabgrenzung
- Es bilden sich undefinierte Grauzonen, diese stellen Projektrisiken dar
- Grauzonen ergeben sich oft dadurch, dass nicht alle Stakeholder berücksichtigt wurden
- Ziel eines Projektleiters ist die schnelle Eliminierung der Grauzonen

Kontextanalyse: Systemabgrenzung und Kontextabgrenzung

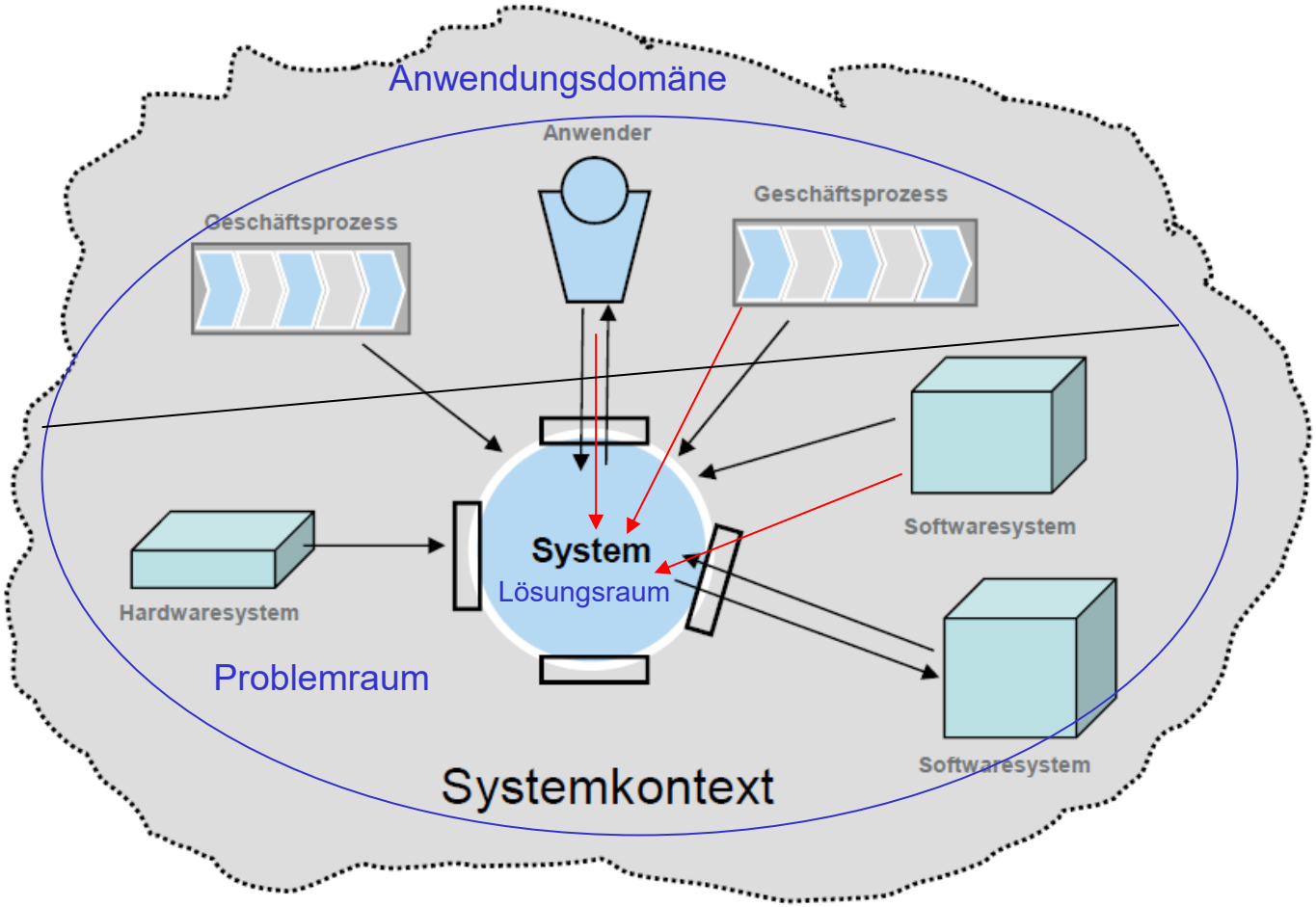
- **Systemabgrenzung:** Es wird die Systemgrenze bestimmt, die festlegt, welche Aspekte durch das geplante System abgedeckt werden sollen/ Teil dieses Systems sind bzw. welche durch die Umgebung erbracht werden
- **Kontextabgrenzung:** Bestimmt die Grenze des Kontexts zur irrelevanten Umgebung, indem analysiert wird, welche Aspekte in der Umgebung eine Beziehung zum geplanten System haben



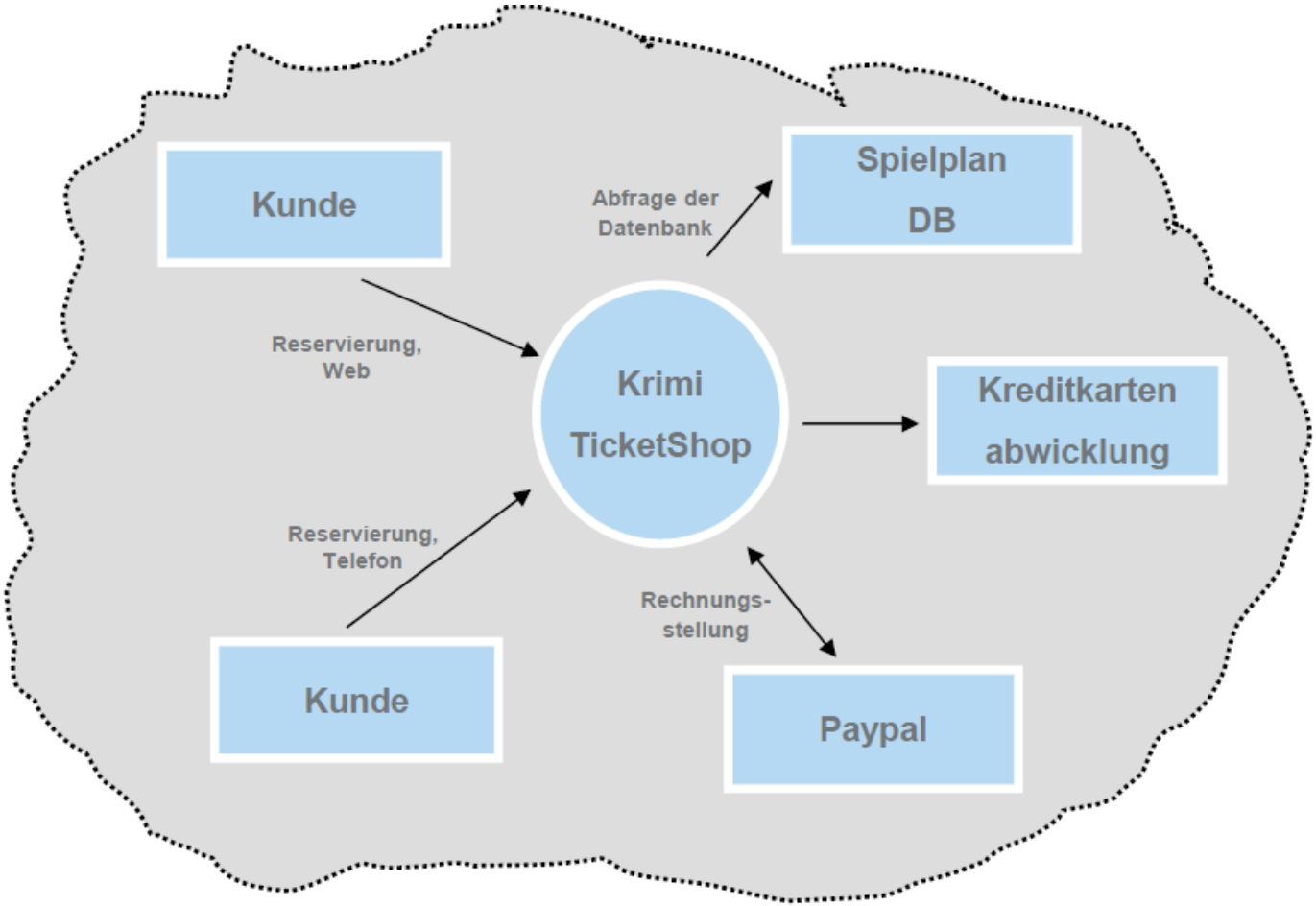
System - Kontextanalyse

- Kontextaspekte sind:
  - Personen
    (Stakeholder/-gruppen)
  - Systeme im Betrieb
    (technische Systeme, Hardware)
  - Prozesse
    (technisch oder physikalisch, Geschäftsprozesse)
  - Ereignisse
    (technisch oder physikalisch)
  - Dokumente
    (Gesetze, Systemdokumentationen…)

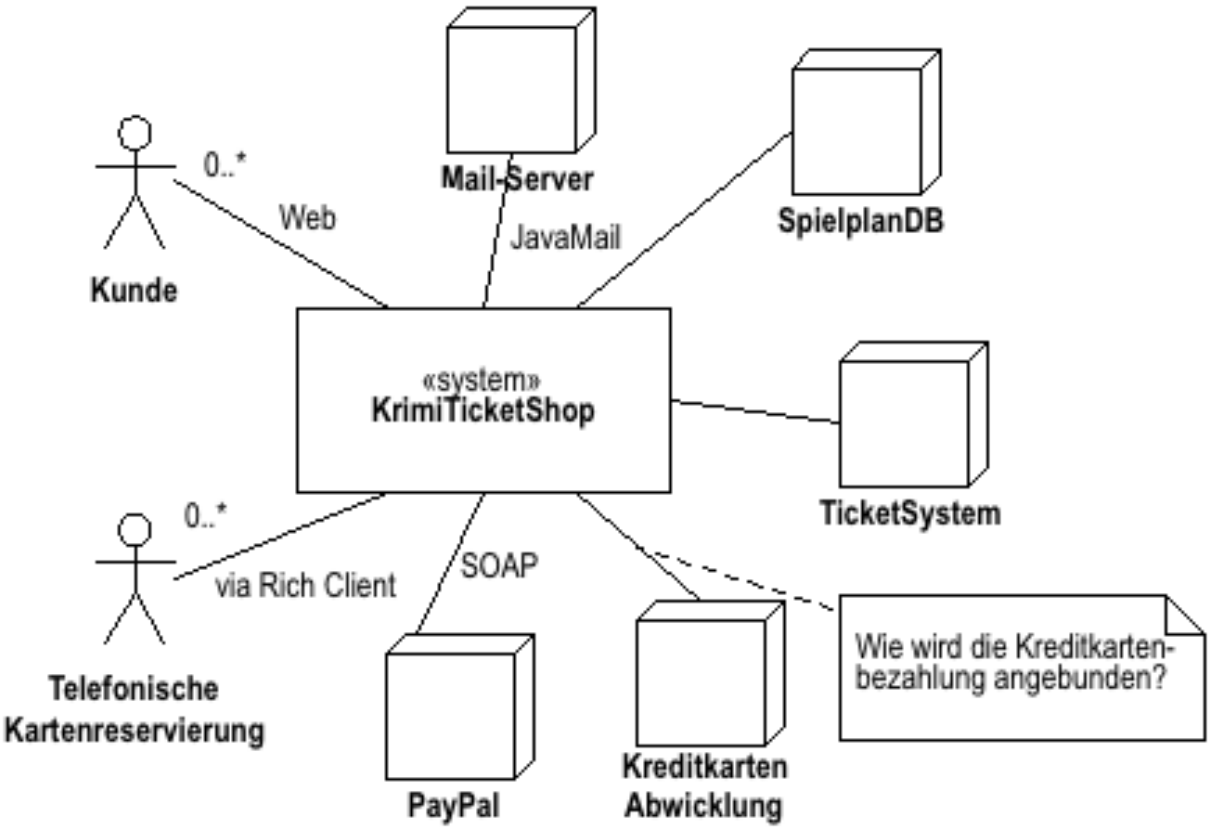
Systemkontextdiagramm – Beispielhafte Darstellung



Systemkontextdiagramm – als Datenflussdiagramm



Systemkontextdiagramm – als Use Case Diagramm (UML)



Problem- und Lösungsraum in der Anwendungsdomäne

- **Lösungsraum**
  - Entspricht dem Gegenstück zum Problemraum
  - Der Lösungsraum definiert die fachliche Schnittstelle, die ein entsprechendes System erfüllen muss (Scope)
  - Anforderungen bilden die Basis für die Entwicklung des Lösungsraums

Arten von Anforderungen

- **Benutzeranforderungen**
  - beschreibt die Anforderungen an die Software aus Sicht der späteren Anwender
  - sind Aussagen in natürlicher Sprache/in Form von Diagrammen zur Beschreibung der Dienste
  - beschreiben die abstrakte Systemidee, die das ermittelte Problem aus fachlicher Sicht lösen könnte
  - Benutzeranforderungen definieren den fachlichen Rahmen eines Systems, insbesondere bei Vertragsverhandlungen und den damit verbundenen Aufwänden
  - Grundlage für die Benutzeranforderungen ist der Problemraum

Arten von Anforderungen

- **Systemanforderungen**
  - legen die Funktionen, Dienste und Beschränkungen detailliert fest. Die Spezifizierungen können Teil des Vertrages sein
  - Verfeinern die Benutzeranforderungen und übersetzen die allgemein formulierte Benutzeranforderung in eine Systemfunktion
  - Die Notwendigkeit einer Systemanforderung muss somit aus einer konkreten Benutzeranforderung ableitbar sein

Arten von Anforderungen

- **Funktionale Anforderungen**
  - Sind die „offensichtlichen“ Anforderungen an ein System
  - sind Aussagen darüber wie das System auf Eingaben und in gewissen Situationen reagieren soll (Daten, Operationen, Verhalten)
  - Definiert somit eine vom System oder Systemkomponente bereitzustellende Funktion
  - Bspw: In datenzentrierten Systemen betreffen funktionale Anforderungen häufig die Manipulation der Daten (neu anlegen, ändern, löschen…)
  - Funktionale Anforderungen haben einen hohen Einfluss auf Komplexität und damit auf die Realisierungskosten einer Applikation (Workflowintegration, Schnittstellen-anbindung…)

Arten von Anforderungen

- **Nicht funktionale Anforderungen**
  - sind Beschränkungen der durch das System angebotenen Dienste oder Funktionen (Verfügbarkeit, Einhaltung von Standards wie z.B. Cross-Browserkompatibilität, Einsetzbarkeit auf mobilen Endgeräten, Sicherheitsaspekte, Zuverlässigkeit, Benutzbarkeit…)
  - Hierzu zählen auch Qualitätsanforderungen, z.B. infrastrukturelle Anforderungen (Skalierbarkeit-, Clusterfähigkeit der Software…)
  - Nichtfunktionale Anforderungen werden stark vom Einsatzkontext beeinflusst (Verfügbarkeit als 24/7 Betrieb vs. innerhalb der Kernarbeitszeiten)
  - Abgrenzung zu funktionalen Anforderungen sind nicht immer eindeutig (z.B. Anforderung an die Mehrsprachigkeit einer Anwendung)