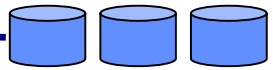
Software Engineering 1

Vorlesungsblock 1: Einleitung

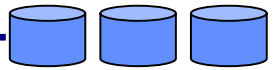
Kristof Böhmer
Fakultät für Informatik
Universität Wien

Inhalt



© Kristof Böhmer

1.1 Motivation: Katastrophale Softwarefehler



- **Mehr als 60% aller Softwareprojekte schlagen (teilweise) fehl**
  - 2014 (Australien): “Ultranet”: Online Lern- und Managementportal für australische Schulen.
    - Ursprüngliche Kostenschätzung: $60 Millionen ⇔ Tatsächliche Kosten: $180 Millionen
    - Schlussendlich in nur 4% der Schulen eingesetzt.
  - 2012 (UK): National Program for IT (NPfIT) sollte die Gesundheitsversorgung des National Health Service (NHS) digitalisieren, wurde aber wegen mangelnder Unterstützung abgebrochen.
    - Ursprüngliche Kostenschätzung: £2,3-Milliarden ⇔ Tatsächliche Kosten: £20-Milliarden
  - 2011 (Süd-Korea): Durch den Ausfall des Systems der Nonghyup Bank waren keine Transaktionen möglich.
    - Dauer: 10 Tage, Betroffen: 30-Millionen Personen.
  - 2008 (Kanada): Ein Netzwerkproblem führt zur ganztägigen Schließung des Toronto Stock Exchange.
  - 2008 (UK): Ausfall des Heathrow T5 Gepäcksystems
    - 636 Flüge wurden innerhalb von 10 Tagen storniert oder waren verspätet.
    - Schaden: US $32 Millionen.

Quelle: https://spectrum.ieee.org/static/the-staggering-impact-of-it-systems-gone-wrong

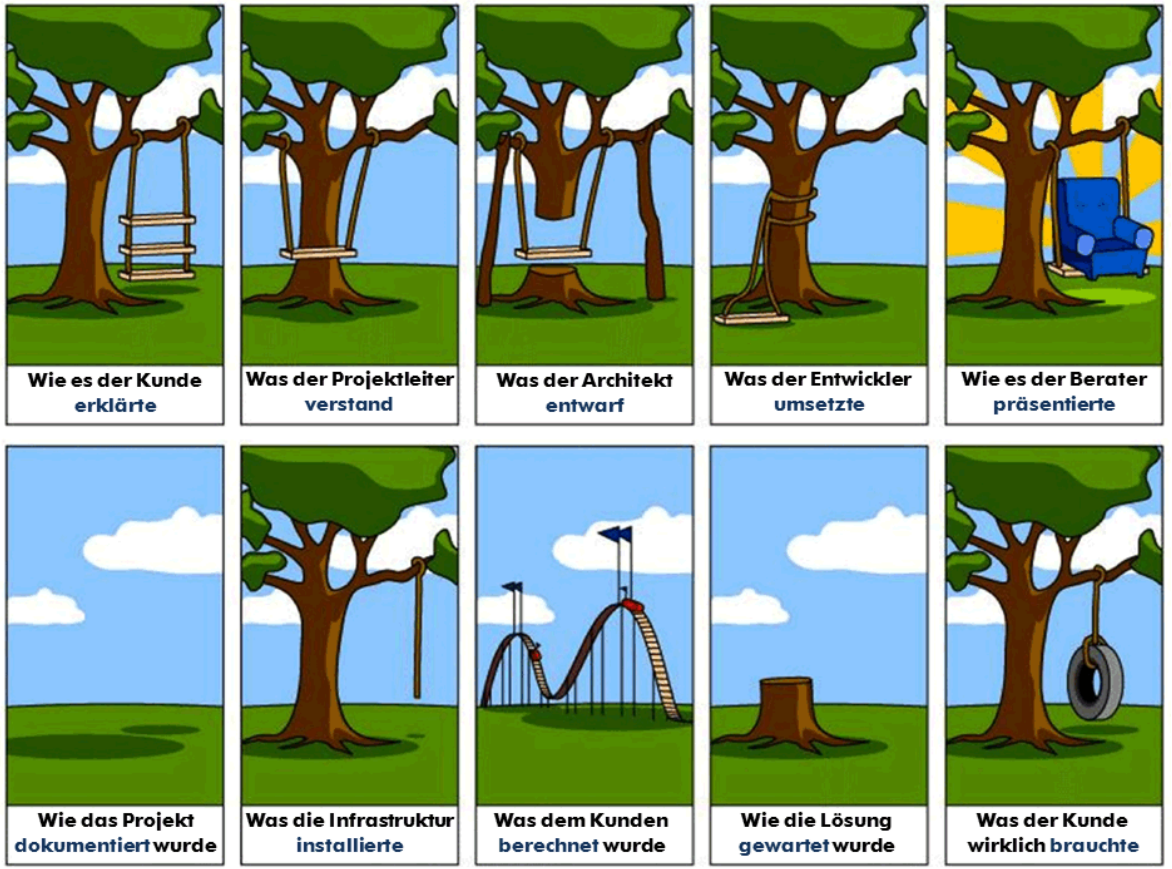
1.1 Motivation: Ursachen für katastrophale Softwarefehler

- **Mögliche Ursachen für katastrophale Softwarefehler bzw. fehlgeschlagene Projekte.**
  - Unrealistische oder undeutliche Projektziele
  - Fehleinschätzung der benötigten Ressourcen
  - Schlecht definierte Systemanforderungen
  - Mangelhafte Projektstatusberichte
  - Unkontrollierte Risiken
  - Schlechte Kommunikation zwischen Kunden, Entwicklern und Anwendern
  - Verwendung unreifer Technologien
  - Projektkomplexität kann nicht bewältigt werden
  - Schlampige Entwicklungsmethoden
  - Mangelnde Projektführung
  - Stakeholder-Politik
  - Konkurrenzdruck

Quelle: https://spectrum.ieee.org/computing/software/why-software-fails

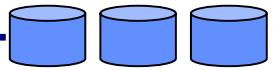
1.1 Motivation: Schlechte Kommunikation zwischen Kunden und Entwicklern

- **Die Ersteller einer Software werden oftmals durch Kommunikationsprobleme und Interessenskonflikte behindert.**



Quelle: http://www.programmwechsel.de/lustig/management/schaukel-baum.html

1.1 Motivation: Software Engineering, eine Lösung?



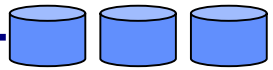
Wie kann Software Engineering helfen?

- Vor dem Softwareprojekt
  - Vorgehensmodelle, Templates, Dokumente, Best Practices usw. um das Softwareprojekt und die damit verbundenen Anforderungen, Ziele, Einschränkungen etc. für alle Partner verständlich und nachvollziehbar zu dokumentieren und zu kommunizieren.
- Bei Softwareprojektbeginn
  - Bekannte Software Design Pattern und Software-Architekturkonzepte erlauben es auch komplexe Software klar und eindeutig zu planen beziehungsweise zu entwerfen.
  - Die Erweiterung, Wartung und Anpassung von Software an neue Herausforderungen wird von Beginn an eingeplant und durch erprobte Techniken unterstützt.
- Während dem Softwareprojekt
  - Best Practices ermöglichen nicht nur eine klare und eindeutige Definition, sondern auch Dokumentation und Verifikation der erstellten Codebasis (erleichterte Weiterentwicklung).
  - Moderne Ansätze zum manuellen und automatischen testen von Software reduzieren die vom Kunden bemerkten Fehler nachweisbar (dies verringert die Wartungskosten).
- Nach dem Softwareprojekt
  - Vereinfachte Außerdienststellung oder Migration auf neue Softwareversionen
  - Neue Versionen werden mit minimalen Aufwand vollautomatisch und mit hohem Durchsatz an alle Kunden verteilt (z.B. für mobile Apps und webbasierte Lösungen)

Inhalt

© Kristof Böhmer

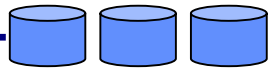
1.2 Reifegrad: Reifegrad der Softwaretechnik als industrielle Disziplin



- **Softwaretechnik als relative junge Disziplin ⇨ Existiert seit etwa einem halben Jahrhundert.**
  - Vgl. Bauwesen (z.B. Brückenbau)
    - Alter: Jahrtausende
  - Vgl. Industrieller Maschinenbau (z.B. Buchdruck)
    - Alter: Ca. 500 Jahre
  - Vgl. Automobilbau
    - Alter: Ca. 200 Jahre
- **Was ist eine "reife" Disziplin?**
  - Disziplin ist erfahren und industrialisiert
  - Es gibt wohldefinierte Standardfälle
    - Beispielsweise Einfamilienhaus im industrialisierten Bauwesen
  - Disziplin ist gut genormt (es existieren z.B. tausende ÖNORMEN)
  - Kosten und Aufwand sind verlässlich abschätzbar

Block 1: Die nachfolgenden Folien basieren auf
- **T. Grechenig, M. Bernhart, R. Breiteneder, K. Kappel: Softwaretechnik. Pearson Studium 2010**

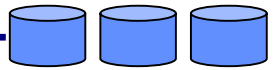
1.2 Reifegrad: Entwicklungsstufen - 1. Reifestufe: (Kunst-) Handwerk



- **Entwicklungsstufen die eine Disziplin durchläuft**
  - **1. Reifestufe: (Kunst-) Handwerk**
    - Am Anfang steht immer das Handwerk mit der Produktion von Waren, mit der Zeit entwickelt sich ein Markt aus Herstellern und Abnehmern

Die 1. Reifestufe lässt sich industriell charakterisieren durch	
Fähigkeit	Virtuosen, Pioniere, talentierte Amateure, Autodidakten, „regionale Genies“
Methodik	Intuition, Versuch und Irrtum, „nackte Gewalt“, Kunstfehler
Innovation	Willkürliche Richtungswechsel, zufällige Fortschritte und neuartige Erfindungen
Wissenstransfer	Gelegentliche oder zufällige Weitergabe von Erfahrungen und Erfindungen
Wirtschaftlichkeit	Verschwenderischer und sorgloser Umgang mit Materialien und Ressourcen
Absatzmarkt	Fabrikation für den konkreten Einsatz im Gegensatz zur Fabrikation für den Verkauf

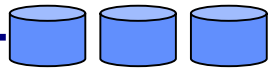
1.2 Reifegrad: Entwicklungsstufen - 2. Reifestufe: Rationalisierung



- **Entwicklungsstufen die eine Disziplin durchläuft**
  - **2. Reifestufe: Rationalisierung**
    - Produktion wird systematisiert und rationalisiert
    - Das entstehende Umfeld fördert Konkurrenz, Differenzierung und Kommerzialisierung

Die 2. Reifestufe lässt sich industriell charakterisieren durch	
Fähigkeit	Qualifizierte Handwerker und (Kunst-)Handwerksgilden
Methodik	Bewährte Arbeitsweisen und etablierte Verfahren
Innovation	Spezialisierung, Veredelung und Verfeinerung
Wissenstransfer	Schulungen, Routinetätigkeiten und Handwerksausbildungen
Wirtschaftlichkeit	Wirtschaftlicher Umgang mit Materialien. Kostenwahrheit wird erreicht
Absatzmarkt	Verkaufsorientierte Produktsicht und Produktentwicklung

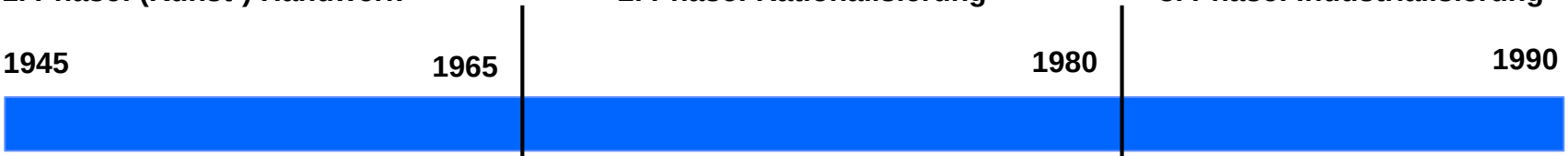
1.2 Reifegrad: Entwicklungsstufen - 3. Reifestufe: Industrialisierung



- **Entwicklungsstufen die eine Disziplin durchläuft**
  - **3. Reifestufe: Industrialisierung**
    - Elaboration, Weiterentwicklung, Marktstrukturen, Berufsstand, Industrialisierung, Wissenschaft und langjährige Erfahrungen ergeben eine etablierte professionelle Ingenieursdisziplin

Die 3. Reifestufe lässt sich industriell charakterisieren durch	
Fähigkeit	Verfügbarkeit von ausgebildeten Fachleuten und Experten
Methodik	Tätigkeit des Ingenieurs basiert auf Wissenschaft und Erfahrung
Innovation	Systemtheorie, Verfahren zur Systemanalyse und Systemklassifizierung
Wissenstransfer	Disziplinfortschritt unterliegt der wissenschaftlichen Prüfung und Bewertung
Wirtschaftlichkeit	Etablierte Meisterklassen und zertifizierte Zulassungen
Absatzmarkt	Der Markt ist nach Produktklassen segmentiert und verdichtet

Reifegradevolution der Softwaretechnik



Pioniere, Gründer und Erfinder

- EDSAC, erster komplett funktionierender Computer basierend auf der von Neumann Architektur
- FORTRAN, erste höhere Programmiersprache (John Backus IBM)
  - Weitere: ALGOL, BASIC
- Erste Konferenz über künstliche Intelligenz (Dartmouth College)
- NATO Science Committee fördert die ersten zwei Konferenzen über Softwaretechnik

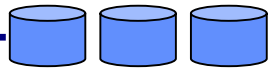
Rationalisierung + Verbreitung

- „Software Crisis“: Zahlreiche Projekte scheitern dramatisch
- Die zunehmende Komplexität überfordert die Beteiligten
- Verschiedenste Probleme und Schwachstellen in den bestehenden Herangehensweisen werden identifiziert

Anfänge der Industrialisierung

- Der Markt ist in keiner Weise durch klare Produktklassen segmentiert und verdichtet.
- Die Disziplin hat noch wenig Gedächtnis und kann Erfolg und Misserfolg nur situativ und nach längerer Rückschau identifizieren

1.2 Reifegrad: Welchen Reifegrad besitzt die Softwaretechnik als Disziplin?



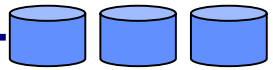
- **Softwaretechnik: Anzeichen für den 3. Reifegrad**
  - Relative klare und stabile Basis in bestimmten Trägerbereichen, siehe beispielsweise Prozessor-Hardware, Betriebssysteme, Protokolle oder Netzwerke
  - Protokolle: HTTP, SMTP, TCP etc. als zuverlässige und verfügbare Technologie
  - Erfahrene EntwicklerInnen sind verfügbar
  - Gute Werkzeuge wie beispielsweise Programmiersprachen, Frameworks
  - Standardisierte Methoden und Vorgehensweisen sind verfügbar: beispielsweise V-Modell, evolutionäre Entwicklung, Spiralmodell, Agile Methoden etc.

Projektcharakter	Reife Disziplin	Softwaretechnik
Standardlösung	80%	10%
Etablierte Verfahren und geeignete Spezialisten	18%	35%
Enthalten Elemente aus Innovation, Erfindung und Forschung	2%	55%

1.2 Reifegrad: Beispielprojekt Krankenhaus 1

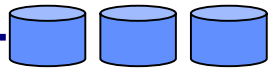
- **Perspektive: Bauwesen**
  - Zielpreis schwankt zwischen 30% und 50% je nach Komplexität, gegebenem Input (z.B. Anzahl der Räume etc.) und ausführendem Team
- **Perspektive: IT-Infrastruktur (Wirtschaftliches Risiko)**
  - Zielpreisbrandbreite schwankt um bis zu 300%
  - Verschiedene Varianten und Umsetzungsentscheidungen werden als gleichberechtigt angesehen, führen jedoch zu den unterschiedlichsten Kostenentwicklungen und Zeitplänen
- **Verschiedene Lösungen verursachen unterschiedliche Migrations- und Integrationsaufwände**
  - A priori schwer abschätzbar
  - Unterschiedliche Aufwände für Lieferanten durch verschiedene Produkte und Gewinnmodelle
  - Auch der konkrete Betrieb einer Software wird beeinflusst
    - Beispiel: beim Outsourcing entsteht kurzfristig eine Kostenersparnis aber langfristig ein hoher Migrations-/Integrationsaufwand sowie ein Wissensverlust

1.2 Reifegrad: Beispielprojekt Krankenhaus 2



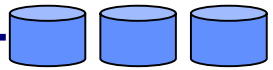
- **Bauwesen ⇔ Softwaretechnik**
  - Im Bauwesen konnten sich die Kernkonzepte über tausende Jahre etablieren (z.B. Türen, Fenster, etc.)
  - Außerdem wird im Bauwesen die Standardisierung der Komponenten durch rechtliche Normen unterstützt dadurch ist ein “Mix-and-Match” dieser Komponenten von verschiedenen Herstellern möglich
  - Dementgegen ist die Standardisierung in der Softwaretechnik domänenabhängig, das heißt das Austauschen von Komponenten resultiert (teilweise) in beträchtlichen Integrationsaufwand

1.2 Reifegrad: Dynamik in der Informationstechnik



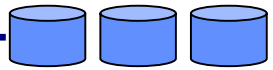
- **Einzelne Teile der Disziplin sind reif, jedoch ist das Zielbild selbst noch nicht bekannt.**
  - Stabilisierungsprozesse werden durch Technologiesprünge und disruptive Veränderungen unterbrochen und anschließend neu durchlaufen
    - Mainframe ⇨ Personal Computer
    - Festnetztelefon ⇨ Mobile Devices
    - Inhouse-Applikationen ⇨ vernetzte Websysteme
    - Lokale Datenspeicher ⇨ Cloud
- **Kostenwahrheit/Gesamtkosten des Betriebs (Total Cost of Ownership - TCO) eines Systems über Dekaden hierdurch schwer zu bestimmen.**

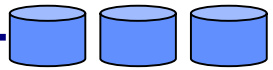
1.2 Reifegrad: Zusammenspiel mehrerer Unternehmensebenen



- **Zusammenspiel mehrerer Unternehmensebenen**
  - **Ebene: Unternehmensorganisation**
    - Standardlösungen von IBM, SAP, Microsoft, etc. mindern zwar das Risiko auf Managementebene, führen aber langfristig zu einer Inflexibilität auf technischer Ebene und vermehrten Abhängigkeiten. Beispiel: „LiMux“
    - Ausspruch: „No one ever got fired for buying IBM“
  - **Ebene: Mitarbeiterorganisation**
    - Ein Konflikt auf dieser Ebene lässt sich nicht (schwer) technologisch lösen
  - **Ebene: Technologie**
    - Die beste Technologie ist von geringen Wert, falls am Arbeitsmarkt nicht genügend qualifizierte Spezialisten verfügbar sind oder das Budget nicht ausreichend ist um die Technologie effektiv einzusetzen
- **Entscheidungen auf einer Ebene haben teilweise auch unvorhersehbare Auswirkungen auf andere Ebenen.**

Inhalt





- **Projektdefinition nach dem Projekt Management Institute (PMI)**
  - "A project is a temporary endeavor undertaken to create a unique product or service. Temporary means that every project has a definite beginning and a definite end. Unique means that the product or service is different in some distinguishing way from all other products or services."
    - Quelle: https://www.pmi.org/about/learn-about-pmi/what-is-project-management
- **Projektdefinition nach PRINCE2**
  - "A project is a management environment that is created for the purpose of delivering one or more business products according to an agreed business case."
    - Quelle: https://www.cupe.co.uk/prince2-definition-of-a-project.html

1.3 Projektdefinition: Typische Merkmale von Projekten

- **Typische Merkmale eines Projekts**

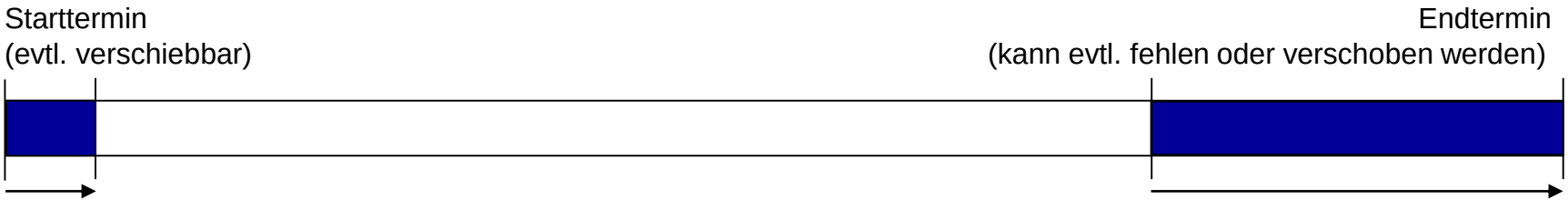
- **Was ist ein Projekt nicht:** Routinetätigkeit, Linienaufgabe, Betriebsablauf

1.3 Projektdefinition: Projektmerkmale – Zielvorgabe

- **Zielvorgabe:** Die Ziele eines Projektes sind
  - Betrieblicher Bedarf
  - Innovative Persönlichkeiten
  - Identifizierte Probleme
- **Was soll durch das Projekt erreicht werden? Welchen Zweck hat das Zielsystem?**
- **Ziele können sich im Laufe des Projektes verändern, erweitern bzw. auflösen.**
  - Über die ganze Projektlaufzeit hinweg muss dieses regelmäßig validiert werden – Bauen wir das richtige System?
- **Hinter Projektzielen stehen Stakeholder**
  - Stakeholder sind Personen, die Interesse am Projekt haben wie Auftraggeber, Sponsoren (Budgetbereitstellung) und Promotoren
    - Promotoren sind projektpolitische Träger sowie Treiber und Unterstützer des Projektes
    - Verlust von Promotoren führt zu Projektversagen trotz technischer Exzellenz

1.3 Projektdefinition: Projektmerkmale – Zeitbegrenzung

- **Zeitbegrenzung:** Ein Projekt ist zeitlich eingrenzbar.
  - Vom ersten Konzept bis zum Ende des Lebenszyklus

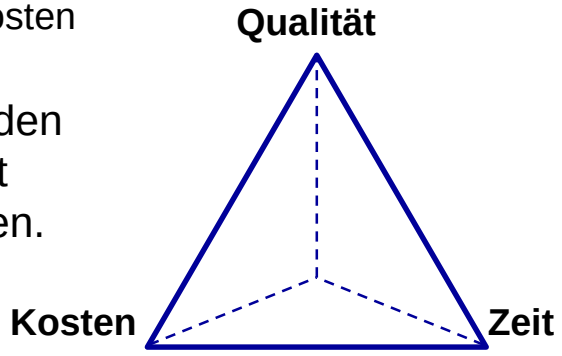


- **Zeitliche Phasen in einem Projekt**

Gründerzeit	Aufbauzeit	Betriebszeit	Entsorgungszeit
• Machbarkeit • Plan • Ziel • Promotion • Rahmen	• Budget • Dauer • Entwicklung • Umsetzung • Roll-Out	• Wartung • Pflege • Laufende Tagesgeschäfte der Anwender	• Außerbetrieb-setzung • Migration auf ein Neu- oder Ablösesystem • „Zerstörung“
S E	S E	S E	S E
S E			

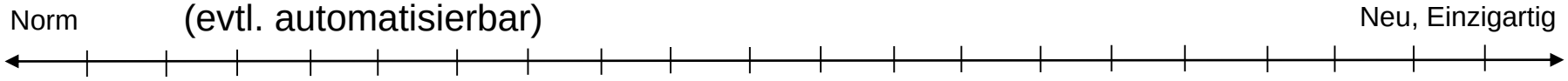
1.3 Projektdefinition: Projektmerkmale – Ressourcen

- **Ressourcen:** Projekte bewegen sich im Rahmen von beschränkten Ressourcen. Diese Ressourcen sind Personal, Zeit und Budget.
- **Der Einsatz dieser Ressourcen steht in einem Zusammenhang:**
  - Mangelndes Personal kann durch ein höheres Budget kompensiert werden.
  - Besseres Personal reduziert den Zeitaufwand bei der Umsetzung.
  - Je länger ein Projekt dauert, desto teurer wird es.
- **Das „Magische Dreieck“ ist eine alternative Darstellungsform. Allerdings mit den Faktoren: Qualität, Kosten und Zeit.**
  - Auch hier gilt: Jede Änderung an einem Faktor beeinflusst die anderen Beiden.
    - Beispiel: Soll die Qualität erhöht werden steigen die Kosten und/oder die aufzuwendende Zeit.
  - **Praxiseinsatz:** Um kritischen Forderungen von Kunden und Vorgesetzten zu begegnen und diesen die damit zusammenhängenden Auswirkungen zu verdeutlichen.
    - Oft gehört*: Das Projekt muss jetzt 3 Wochen früher fertiggestellt werden, weil …. , das schafft Ihr schon!*



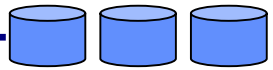
1.3 Projektdefinition: Projektmerkmale – Einmaligkeit und Risiko

- **Einmaligkeit oder Identitätseigenschaft:** Einmaliges Vorhaben
  - Wird auch als Einzigartigkeit bezeichnet.
- **Identitätsskala**
  - Links auf der Skala: Kein Projekt, sondern gewöhnlicher Geschäftsbetrieb (evtl. automatisierbar)



- **Risiko:** Jedes Projekt hat seine besondere Risikostruktur.
  - Einmaligkeitscharakter eines Projektes kann spezifisch und szenarisch zu außergewöhnlichen Risiken führen (z.B. Kostenerhöhung, Zeitverzögerung, Projektstillstand/-scheitern)
  - **Risiken müssen frühzeitig identifiziert und mit geeigneten Maßnahmen beantwortet werden um Projekte erfolgreich durchführen zu können**

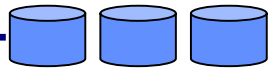
1.3 Projektdefinition: Projektmerkmale – Risiko



- **Risikocharakter:** Spezifisch für eine Fachdomäne.
  - Die selbe technische Ursache für das Versagen eines Systems (z.B. Rechenzentrumsausfall) hat je nach Domäne eine andere Kritikalität, wie beispielsweise:
    - Banksystem: Ausfall der Geldausgabeautomaten
    - Universität: Ausfall des Onlinekursverzeichnisses
- **Risikocharakter:** Szenarisch, nicht themen-stationär.
  - Der gleiche fachliche Bereich hat unterschiedliche Risiken je nach Kontext.
  - Beispiel: Etablierung eines E-Zivilregisters (ist technisch keine außergewöhnliche Aufgabe)
    - Österreich: Projektpolitisches Risiko durch Migration und Ablöse eines Altsystems
    - Algerien: Risiko durch die Leitungsinfrastruktur, weiters ist kein verlässliches Government-Netzwerk vorhanden
- **Risikocharakter:** Immer relativ, lokal und personenabhängig
  - Team-bezogene Risikofaktoren in Softwareprojekten sind beispielsweise das Know-how-Profil von Mitarbeitern, budgetäre Rahmenbedingungen und technologische Fehleinschätzungen, sowie die Selbstüberschätzung der eigenen Skills

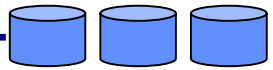
- **Typische Merkmale von Softwareprojekten**

1.3 Projektdefinition: Softwareprojektmerkmale – Zielsetzung



- **Zielsetzung:** Ein Projektverlauf hängt stark davon ab, ob der Sinn und Zweck des Zielsystems durch einen Promoter getragen wird oder ob diese Aspekte den Beteiligten intuitiv klar sind.
  - Mögliches Projektrisiko: Falls Erwartungen an ein Projekt bestehen, die nicht den inneren Zielen und Leistungen entsprechen
  - Große Projekte benötigen ein angemessenes Maß an Öffentlichkeitsarbeit und Projektmarketing ⇨ laufende Validierung der Ziele
- **Validierung:** Überprüfen, ob das richtige System gebaut wird
  - Entspricht das gebaute System den Zielen des Promoters

1.3 Projektdefinition: Softwareprojektmerkmale – Dauer 1

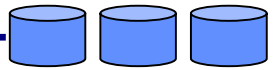


- **Dauer:** Laufzeit des Projektes wird dargestellt durch die Kalenderwochen (bzw. -tage, -jahre) von Projektstart bis zum Projektende

- **Die Projektdauer steht in Abhängigkeit mit:**
  - Der Komplexität der gegebenen Aufgabenstellungen
  - Den Fähigkeiten des Teams
  - Den vorliegenden Projekt- und Umfeldbedingungen

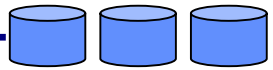
- **Komplexität der Aufgabenstellung**
  - Je nach Aufgabenstellungen benötigen die Beteiligten gewisse Qualifikationen um rasch genug voranzukommen (oder um diese überhaupt lösen zu können) bzw. um diese aufwandsmäßig abschätzen zu können
  - Andere Aufgabenstellungen jedoch (z.B. Testfälle generieren) lassen sich aus Budgetsicht leicht einschätzen (nahezu linear skalierbarer Aufwand)

1.3 Projektdefinition: Softwareprojektmerkmale – Dauer 2



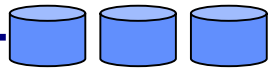
- **Die Anzahl der Projektteilnehmer hat eine variable Auswirkung auf die Projektdauer**
  - Mehr Personal kann zur Beschleunigung aber auch zur Verlangsamung der Projektdauer führen
  - Anzahl der Projetteilnehmer bestimmen Aufwand der Entscheidungsfindung (Abstimmungen, Richtungsdiskussionen, Kommunikationsaufwand)
  - Je mehr Teilnehmer, desto mehr Rollen ⇨ Mehr Kommunikation, Organisation und Projektkontrolle sind notwendig
  - Umplanen, Aussortieren, Nachbauen von Teilaufgaben kostet viel Zeit

1.3 Projektdefinition: Softwareprojektmerkmale – Größe 1



- **Die Größe eines Softwareprojekts wird dargestellt durch:**
  - Quantifizierung des erforderlichen Aufwands
    - Beispielsweise in Personenjahren, -monaten, -tagen)
  - Anzahl der beteiligten Personen
  - Infrastrukturkosten
  - Kosten für Rechte und Materialien (Hardware, Software, Lizenzen)
- **Anforderungen:** Viele Anforderungen machen ein Projekt größer.
  - Eine große Anzahl an Anforderungen erhöht die Komplexität durch eine steigende Anzahl von Zusammenhängen und Abhängigkeiten zwischen den Anforderungen und die Wahrscheinlichkeit von Anforderungsänderungen
- **Innere Synergien**
  - Der Aufwand für die Projektinitialisierung (z.B. für Infrastruktur, Schulungen, Werkzeugwahl und Personalfindung) ist bei größeren Projekten anteilig geringer als bei kleineren Projekten.
  - Oft ist der durchschnittliche Aufwand für die Umsetzung von zusätzlichen Anforderungen aufgrund von inhaltlichen Ähnlichkeiten bei größeren Projekten geringer.

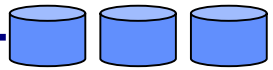
1.3 Projektdefinition: Softwareprojektmerkmale – Größe 2



- **Exemplarische Beispiele für die Größe von Softwareprojekten**

Projektgröße	Dauer	Teamgröße	Personenmonate	Kosten (2010)
Sehr kleines Projekt	0,25 Jahre	1 Person	3 PM	15.000 €
Kleineres Projekt	0,5 Jahre	2 Personen	12 PM	80.000 €
Mittleres Projekt	1,5 Jahre	5 Personen	60 PM	600.000 €
Große Hochschule	3,0 Jahre	15 Personen	500 PM	6.000.000 €
Ticketsystem Bahn	3,5 Jahre	40 Personen	1400 PM	25.000.000 €
ID Paraguay	2,5 Jahre	120 Personen	3500 PM	55.000.000 €
eHealth in Deutschland	5,0 Jahre	500 Personen	30000 PM	600.000.000 €

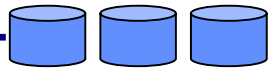
1.3 Projektdefinition: Softwareprojektmerkmale – Größe 3



- **Größe erzeugt Komplexität:** Exemplarische Grobschätzung der relativen Aufwände abhängig von der Projektgröße

Projektgröße	Personenmonate	Aufwand	Dauer	Größensymptom
Sehr kleines Projekt	3 PM	0.5	0.8	User = Entwickler = Auftraggeber
Kleineres Projekt	12 PM	1	1	Benutzerhandbuch
Mittleres Projekt	60 PM	3	1.5	Reviews, automatische Tests
Große Hochschule	500 PM	6	4	User-Workshops
Ticketsystem Bahn	1400 PM	15	4	Call-Center, Uptime
ID Paraguay	3500 PM	10	8	Präventive Medienarbeit
eHealth in Deutschland	30000 PM	15	20	Projekt wird zur Behörde

1.3 Projektdefinition: Softwareprojektmerkmale – Domäne

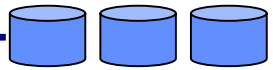


- **Domäne:** Hilft bei der Orientierung und Einordnung des Systems und erlaubt typische Eigenschaften zur Identifizierung und Klassifizierung abzurufen
  - Typische etablierte Domänen für Softwaresysteme sind: Finanzwirtschaft, Gesundheit, Government, Aviation, Automotive, Telekommunikation, Produktionssteuerung.
  - Jede Industriebranche hat ihre eigene Softwareproduktlandschaft.
  - **Ungenügendes Domänenwissen ist ein Projektrisiko und muss entsprechend eingeplant werden**
- **Beispiele für die Charakteristiken einer Domäne**
  - **Transaktionssicherheit:** Eine Transaktion wird immer vollständig durchgeführt oder gar nicht gestartet ⇨ „Verlorene“ Transaktionen gibt es nicht.
    - In finanzwirtschaftlichen Systemen ist eine Transaktion beispielsweise eine Überweisung.
  - **Protokollierung & Dokumentation:** Sehr aufwändig in der medizinischen Domäne, da hier auch die Wahrung des Arztgeheimnisses und die Datenhoheit der Patienten höchste Priorität haben. In der Telekommunikation (EU) wiederum ist die Speicherung der Inhaltsdaten (Telefongespräche, SMS, E-Mails) verboten**.**

1.3 Projektdefinition: Softwareprojektmerkmale – Technologie 1

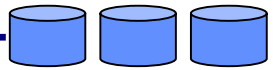
- **Technologie:** Definiert ein Projekt aus software- und informationstechnischer Sicht.
  - Anschaffungskosten, Verfügbarkeit von Ressourcen, Realisierungsaufwand sind abhängig von der gewählten Technologie.
  - Technologiebereiche sind starken Wechseln unterworfen und oft auch produktbezogen zu bewerten (starke Abhängigkeiten sollten vermieden werden).
- **Softwaresysteme sind oft wie Zwiebeln aufgebaut (Schichtenmodell).**
  - Eine innere Schicht liegt unter oder innerhalb der darüberliegenden Schicht.
  - Diese Struktur – oft auch als Technologie Stack oder Technologieebenen bezeichnet – können die Effizienz negativ beeinflussen.
  - Oft jedoch der einfachste Weg um auf bestehenden Technologien aufzubauen.
- **Technologie ist von Softwareklassen abhängig**
  - Anwendungssoftware, Systemsoftware, Echtzeitsoftware

1.3 Projektdefinition: Softwareprojektmerkmale – Technologie 2



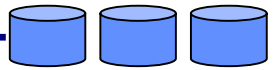
- **Softwareklasse Anwendungssoftware:** Wird direkt und unmittelbar vom End-User (Anwender) benutzt
  - Wird im privaten und betrieblichen/öffentlichen Bereich verwendet.
    - Die Hauptaufgabe von betrieblichen Systemen umfasst die Verwaltung und Aufbereitung von Daten für den User zum Support von operativen Arbeits- und Verwaltungsprozessen.
  - Komplexität entsteht aus der Funktionalität bzw. liegt im User Interface.
  - Typische Herausforderungen: Starke Integrationspflichten, individuelle Service-Levels, Interaktion mit anderen Anwendungen.
    - Der elektronische Akt in einem Amt wäre ein Klassiker dieser Systemart.
  - Beispiele
    - Office-Pakete
    - Browser
    - Spiele
    - Lernprogramme
    - ERP-Systeme
    - CRM-Systeme
    - GIS-Systeme

1.3 Projektdefinition: Softwareprojektmerkmale – Technologie 3

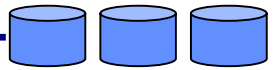


- **Softwareklasse Systemsoftware:** Bezeichnet die Gesamtheit all jener Programme und Systeme, die sämtliche Abläufe des für den User nicht sichtbaren Betriebs eines Systems steuern
  - IT-Infrastruktur
  - Ähnlich der Infrastruktur eines Gebäudes: Wasserleitungen, Elektroleitungen, Klimaanlage, usw.
  - Anwendungssoftware benötigt immer Systemsoftware auf der sie basiert und mit der sie interagiert.
  - Systemsoftware stellt die Verbindung zwischen Anwendungssoftware und Hardware her, sie steuert interne und externe Hardwarekomponenten wie beispielsweise Speicherplatz, Prozessor, Drucker, usw.
  - Beispiele: Betriebssysteme, Firewalls, Sicherheitssoftware, Datenbank, Entwicklungssoftware, Entwicklungswerkzeuge, Entwicklungssprachen, Compiler, Interpreter.

1.3 Projektdefinition: Softwareprojektmerkmale – Technologie 3

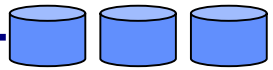


- **Softwareklasse Echtzeitsoftware**: Oft handelt es sich dabei um sogenannte Safety-Critical-Systeme, deren Fehlverhalten einen Unfall, die Verletzung bzw. den Tod von Menschen, großen physischen Schaden oder Umweltzerstörung direkt verursachen kann.
  - Ist als gesonderte Komplexitätskategorie zu betrachten zu welcher beispielsweise autonome Automobile, verschiedene Bereiche der Luftfahrt oder auch Kraftwerkssteuerungen gehören.
  - Kann Anwendungssoftware (Airbuscockpit) und Systemsoftware (Echtzeitbetriebssystem) beinhalten.
  - Formale Methoden spielen hier eine stärkere Rolle. Der Aufwand eines Systembaus ist größer, die Anforderungen können naturgemäß nur eingeschränkter sein.
  - Das wesentliche Qualitätsmerkmal von Echtzeitsoftware ist die Erbringung des richtigen Ergebnisses innerhalb von genau festgelegten Zeitschranken, die innerhalb einer realweltlichen Ereigniskultur in der physikalischen Welt funktioniert.

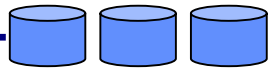


- **Moderne Systeme sind zumeist deutlich offener und weniger isoliert als Altsysteme.**
  - Entsprechend ist ein standardisiertes Vorgehen und eine genaue Planung notwendig um beispielsweise steigenden Sicherheitsrisiken gerecht zu werden.
- **Bauklassen vs. Vorgehensmodelle**
  - **Vorgehensmodelle: Befassen sich mit dem Softwarelebenszyklus.** Ein Vorgehensmodell bestimmt, wie in einem Projekt die Phasen Analyse, Entwurf, Implementierung, Test und Wartung realisiert werden. Siehe Block 2 für zusätzliche Details.
  - **Bauklassen:** Ermöglichen bestehende Strukturen und Topologien zu berücksichtigen
    - Relevante Faktoren: Vorgänger, verwandte Systeme, Bauweisen, Lieferantenstrukturen, zukünftige Betriebsumgebungen, etc.
- **Beispiele für möglich Bauklassen**
  - Ablöse eines Legacy-Systems, Greenfield-Projekt, Customizing-Projekt, Early-Life-Cycle-Projekt, Late-Life-Cycle-Projekt, Wartungsprojekt

1.3 Projektdefinition: Softwareprojektmerkmale – Baumethoden 2

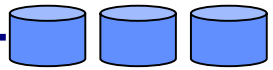


- **Bauklasse Ablöse eines Legacy-Systems:** Bestehendes System wird durch ein neues System ersetzt
  - System hat jahrelang gehalten oder ist über die Jahre gewachsen
  - Wenig konsistente Dokumentation vorhanden
  - Altes System bestimmt mehr als 50% der Funktionalität des neuen Systems
- **Bauklasse Greenfield-Projekt:** Standardprojekt, im Gegensatz zur Ablöse eines Legacy-Systems
  - Das System wird von Grund auf neu geplant und hat dadurch keine anderen IT-Systeme als Vorgänger oder Nachbarn
  - Bei neuen Unternehmen oft der Fall, bei etablierten Unternehmen wird seltener solch eine Konstellation vorgefunden werden
- **Bauklasse Customizing-Projekt:** Verwendet in den Phasen Entwurf und Implementierung ein vorgefertigtes Basissystem oder eine Standardsoftware für die softwaretechnische Realisierung



- **Bauklasse Early-Life-Cycle-Projekt (auch Analyseprojekt):** Liegt dann vor, wenn die Aufgabe darin besteht, eine gute Analyse und einen guten Entwurf zu einem gewünschten System zu erstellen.
  - Dient als Vorstufe für ein Realisierungsprojekt. Dessen Vorgaben werden im Rahmen des Analyseprojekts mittels Planungen und Aufwandsschätzungen bestimmt.
- **Bauklasse Late-Life-Cycle-Projekt (auch Realisierungsprojekt):** Liegt dann vor, wenn Sie als Entwicklungsunternehmen eine fertige Spezifikation und Analyse unverändert übernehmen und darauf aufbauend eine Implementierung vornehmen.
  - Ziel: Funktionierende Applikation mitsamt relevanter Dokumentation.
- **Bauklasse Wartungsprojekt:** Übernahme der Wartungsphase eines Software-Systems, wobei die Entwicklung und Inbetriebnahme von einem anderen Dienstleister durchgeführt wurde.

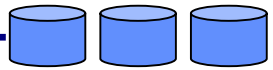
1.3 Projektdefinition: Softwareprojektmerkmale – Komplexität



- **Komplexität:** Die bereits angeführten Merkmale eines Softwareprojektes wie beispielsweise Größe, Dauer etc. sind die natürlichen und unmittelbaren Quellen der Komplexität und wirken direkt auf Risiken und Kosten.

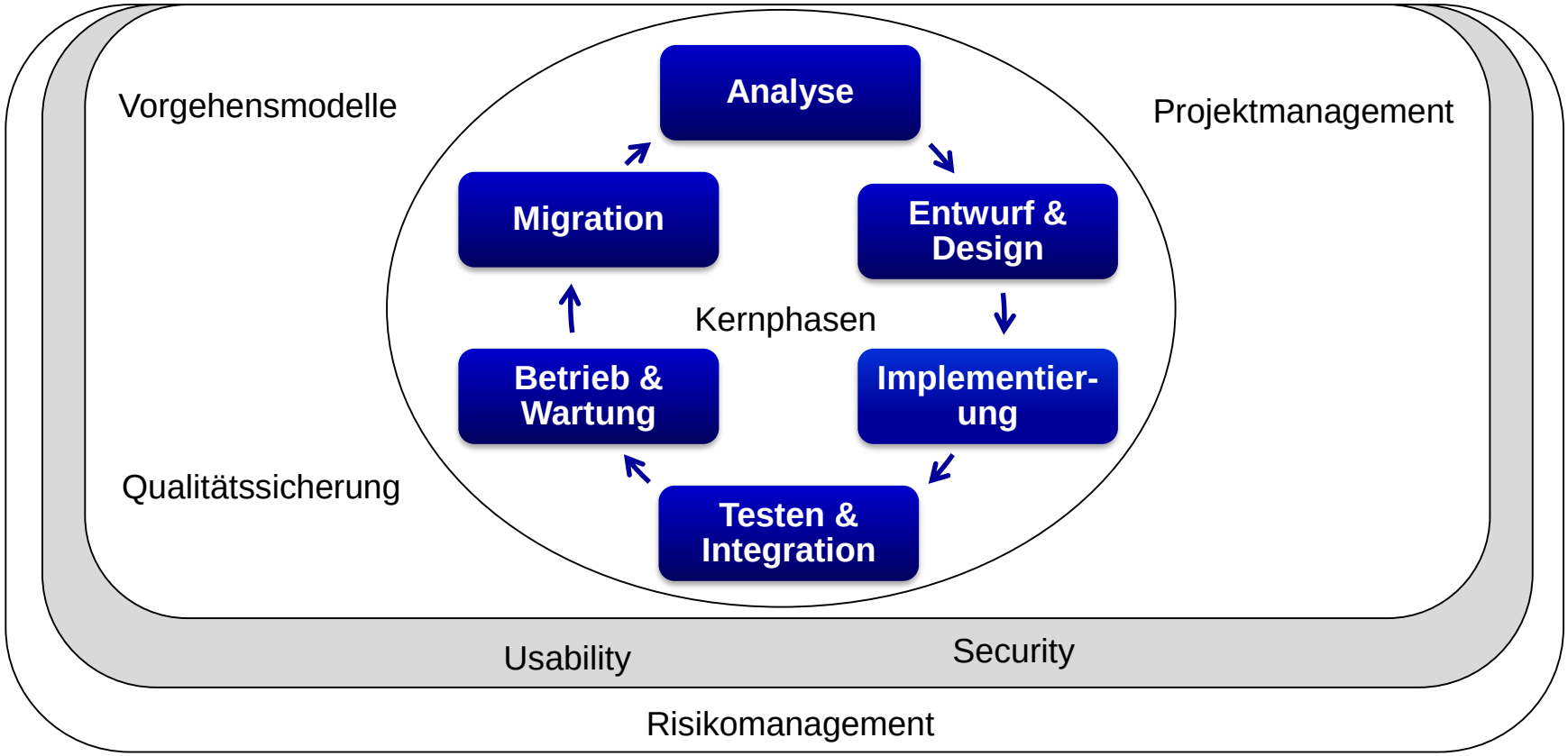
- **Beispiele**
  - Projektgröße wird zum Risiko, wenn sie für ein Unternehmen relativ ungewöhnlich ist.
    - Zu groß = Risiko aus Mangel an Erfahrung.
    - Zu klein = verhältnismäßig teuer im Bau ⇨ Kostenrisiko.
  - Zielsetzungen werden zum Problem, wenn sie nicht klar sind, schlecht kommuniziert werden oder im Laufe des Projektes unkoordiniert verändert werden.
  - Eine ganze Domäne kann zur Schwierigkeit werden, wenn die grundlegenden Geschäftsabläufe nicht verstanden wurden.

Inhalt



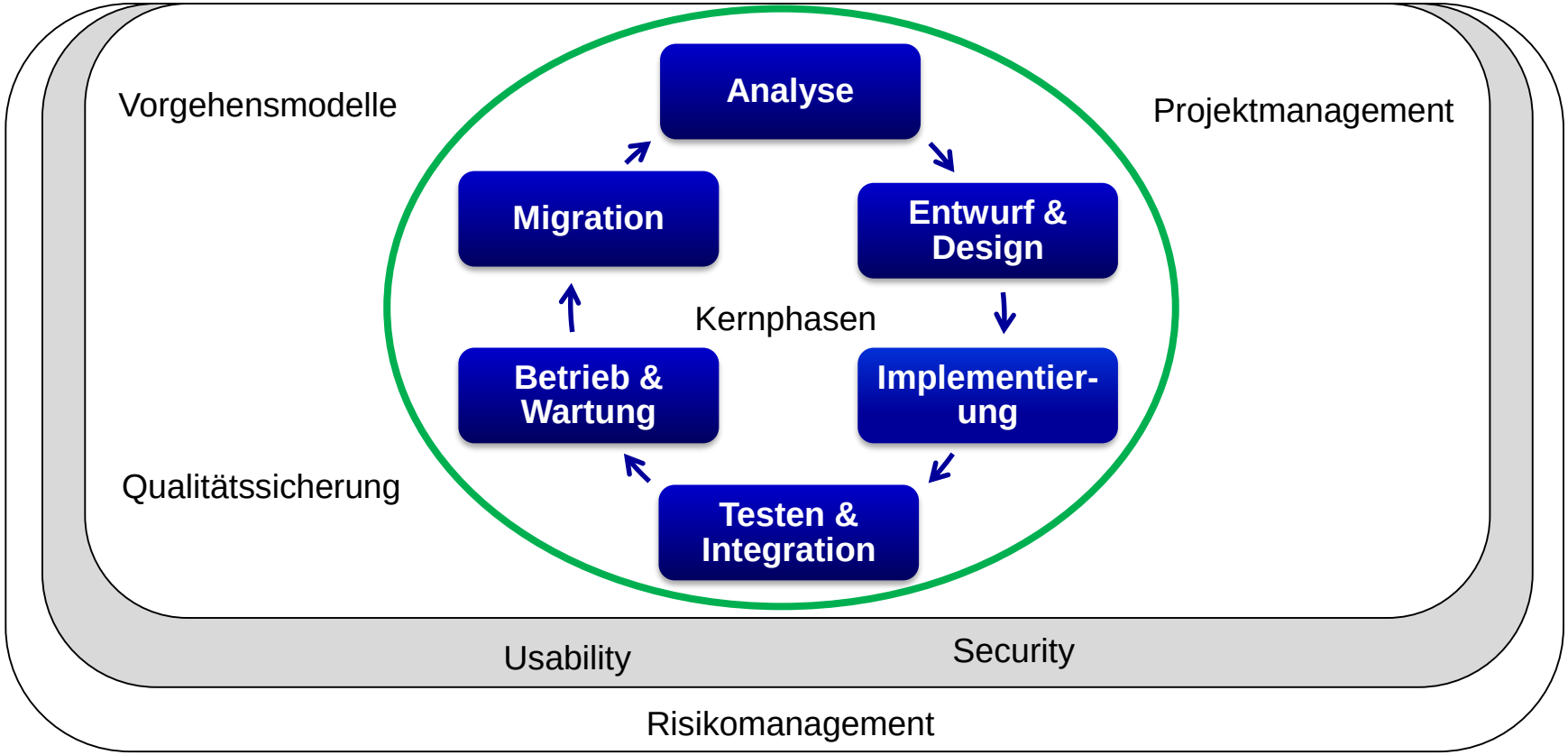
1.4 Struktur der Softwaretechnik: Projektphasen

- **Projektphasen:** Müssen bei der erfolgreichen Entwicklung eines Softwareprojektes durchlaufen werden.
  - Kernprojektphasen umgeben von nebenher ablaufenden Phasen

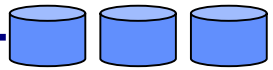


1.4 Struktur der Softwaretechnik: Projektphasen – Kernphasen

- **Projektphasen:** Zuordnung der Kernphasen
  - Die Inhalte der Kernphasen werden auf den folgenden Folien kurz skizziert.

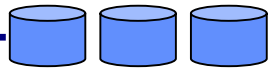


1.4 Struktur der Softwaretechnik: Projektphasen Analyse, Entwurf & Design



- **Analyse:** Grundstein für den Erfolg eines Softwareprojektes. Je größer das Projekt, desto wichtiger ist diese Phase
- **Anforderungsanalyse (Requirements Engineering)**
  - "Was soll das System können?"
  - Festhalten und Strukturierung von Anforderungen in Form eines Anforderungsdokuments (Use Cases, Business-Rules, etc.), durch Gespräche mit:
    - Anwendern
    - Entscheidungsträgern
    - Experten
- **Entwurf & Design:** Die erstellte Sammlung der Anforderungen wird als funktionierendes Softwaresystem umgesetzt
  - Legt das Ziel des Softwareprojekts fest
  - Der technische Gesamtaufbau muss genau geplant werden
  - Software-Entwurf = Technisches Modell des Gesamtsystems, an das sich alle Teilnehmer bei der Projektausführung halten

1.4 Struktur der Softwaretechnik: Projektphase Implementierung



- **Implementierung:** Anforderungen werden auf der technischen Basis des Designs bzw. der Architektur in einer Programmiersprache ausformuliert, kompiliert oder interpretiert, in Betrieb gesetzt, getestet und angepasst etc. bis das Verhalten des Softwaresystems den Anforderungen entspricht.

- **Eine gezielte Auswahl der Technologie (Programmiersprache, Frameworks, Implementierungsplattform, Technologie-Stack) ermöglicht:**
  - Produktivitätssteigerungen
  - Fehlerratensenkungen
  - Erleichterungen während dem Suchen und Beheben von Fehlern
  - Ersetzt jedoch keine angemessene Vorgehensweise oder Methode

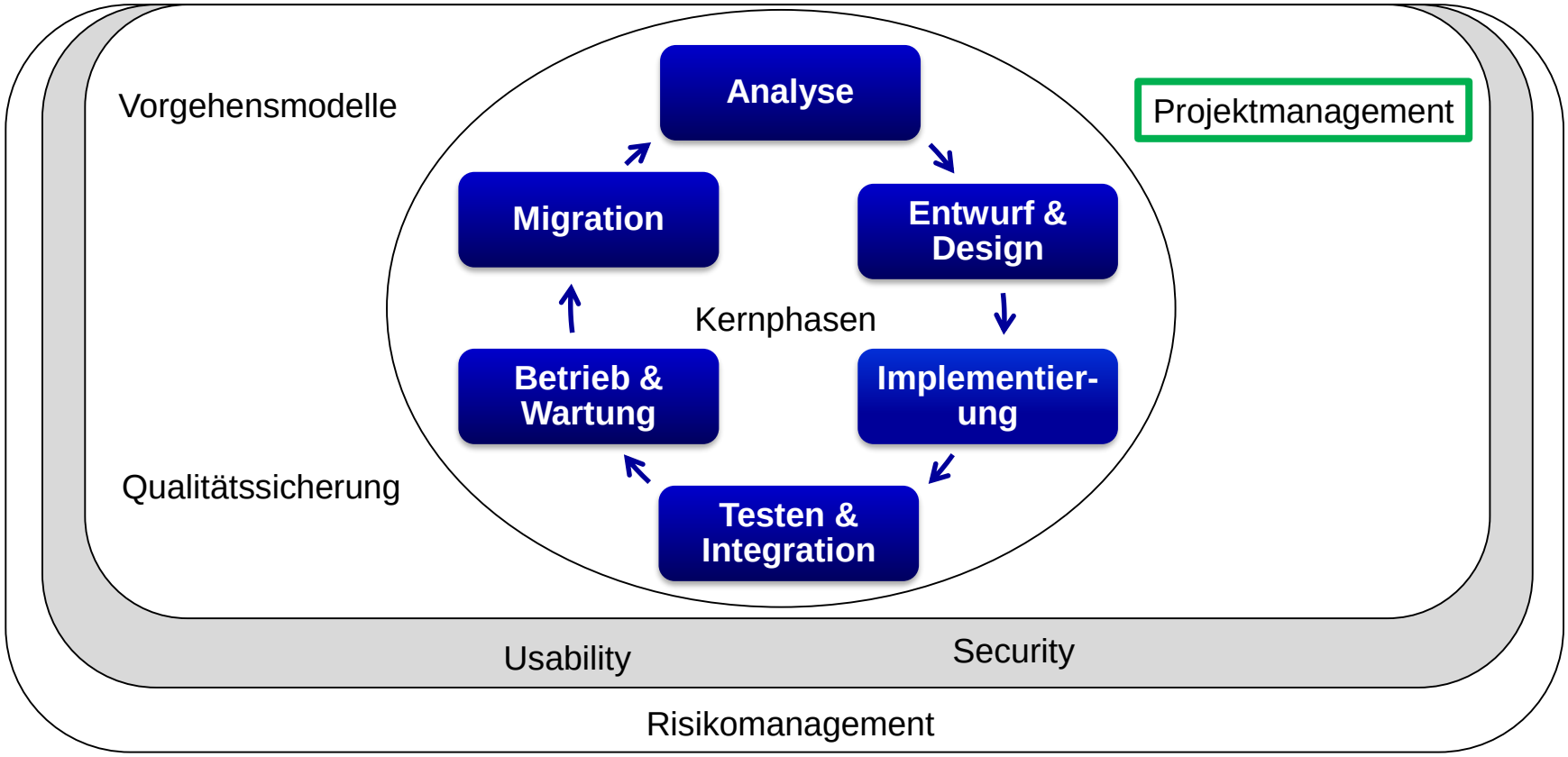
  - **“A fool with a tool, is still a fool”**

1.4 Struktur der Softwaretechnik: Projektphasen Testen, Betrieb & Wartung

- **Testen & Integration:** Das ausprogrammierte System muss ausreichend getestet werden, um seine „Fehlerfreiheit“ zu garantieren.
  - Testen der inneren Korrektheit einzelner Komponenten.
  - Testen der Kommunikation zwischen Komponenten (= Integration).
    - Je größer das System, desto bedeutender sind solche Tests.
  - **Ein Testplan kann bereits in der Designphase erstellt und während der Implementierungsphase umgesetzt werden.**
- **Betrieb & Wartung:** Die Inbetriebsetzung von Softwaresystemen muss ebenfalls geplant sein.
  - Anwender müssen eingeschult und laufend informiert werden.
  - Wartungen ermöglichen:
    - Fehler auszubessern
    - Anpassungen vorzunehmen
    - Die mittels einer Software realisierten Funktionen zu perfektionieren
    - Die Stabilität einer Software zu erhöhen
  - Einzelne Wartungsarbeiten können nach der ursprünglichen Projektphase erfolgen, wenige Freiheitsgrade (limitiert durch das bestehende, laufende System).

1.4 Struktur der Softwaretechnik: Zuordnung des Projektmanagements

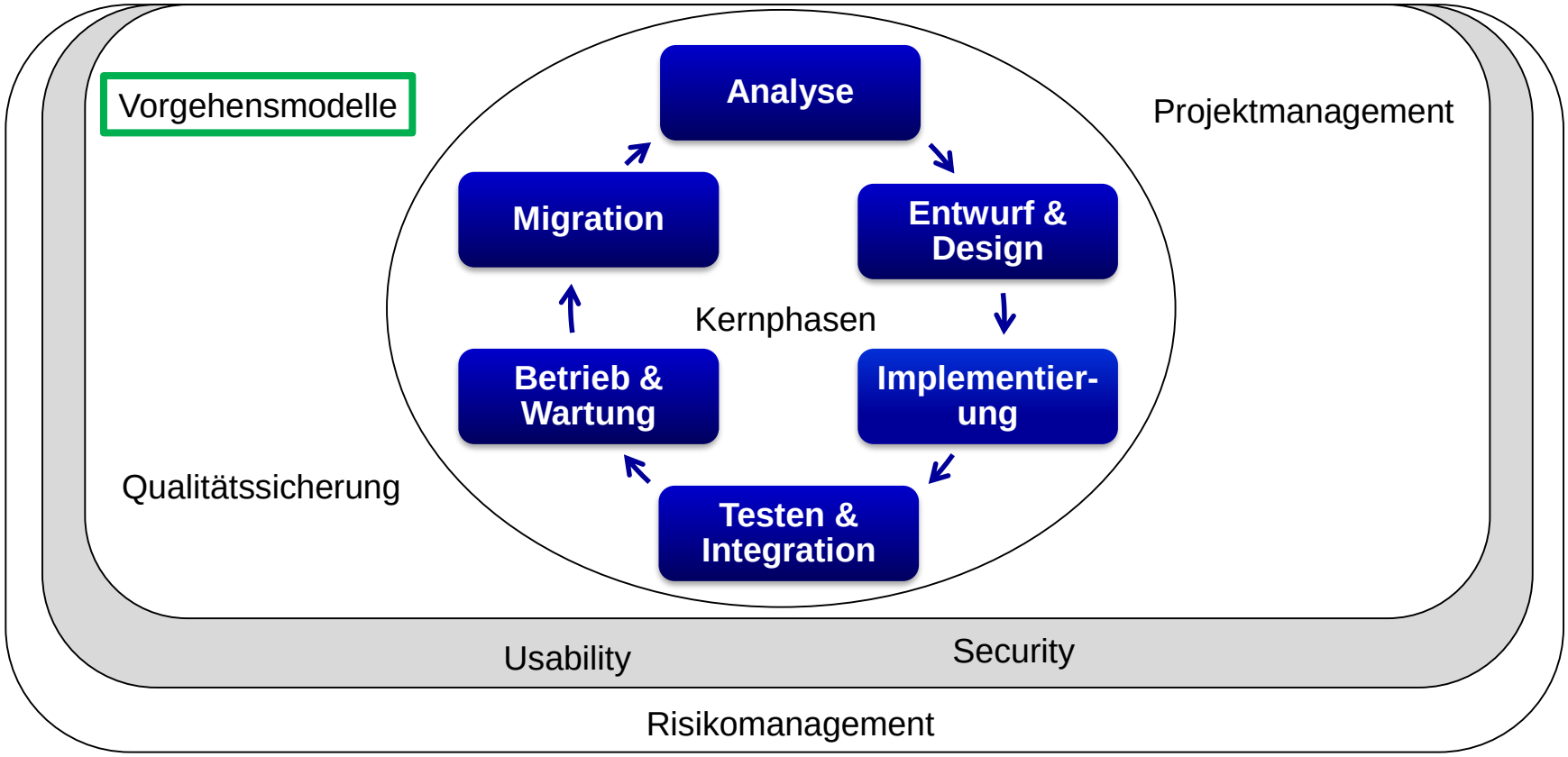
- **Projektphasen:** Zuordnung des Projektmanagements



  - Für detaillierte Informationen zu dieser Phase besuchen Sie bitte die Vorlesung 051039 VU Projektmanagement.

1.4 Struktur der Softwaretechnik: Zuordnung der Vorgehensmodelle

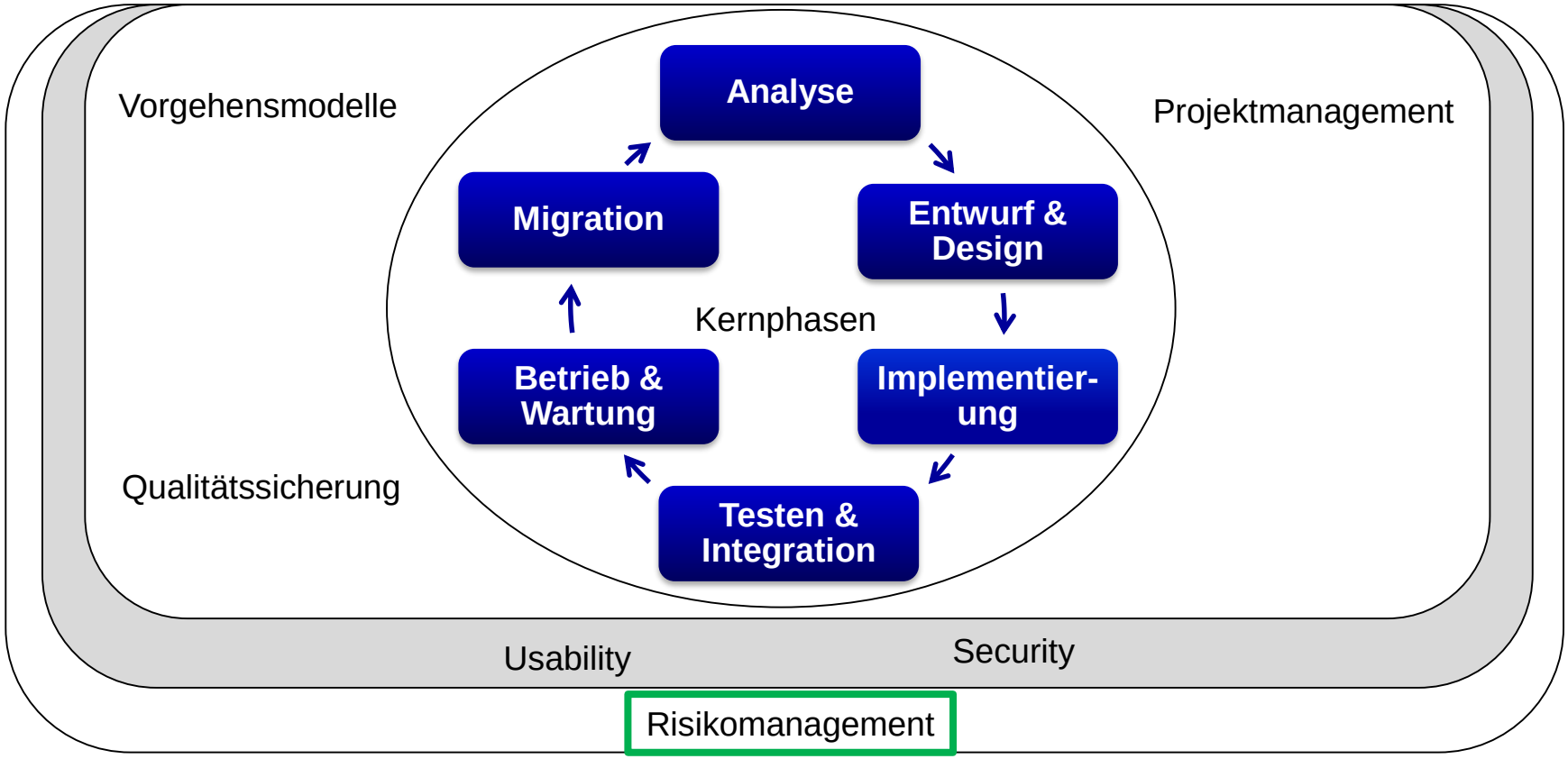
- **Projektphasen:** Zuordnung der Vorgehensmodelle



- **Unterscheidung zwischen Methoden, Verfahren und Vorgehensmodellen**
  - **Methoden:** Eine einfache, praktische und wirkungsvolle Methode ist die Etablierung, Pflege und Einhaltung eines Projektplanes
  - **Verfahren:** Gliederung der Projekte in Projektphasen
  - **Vorgehensmodelle:** Dienen zur Darstellung des eigenen Verhaltens und zur genormten Kommunikation mit den eigenen Mitarbeiten und dem Umfeld
    - Ziel: Einschätzbarkeit, (innere) Transparenz, Messbarkeit, Kontrollierbarkeit und Wiederholbarkeit
    - Unterscheidung zwischen traditionellen und agilen Vorgehensmodellen
    - Beispiele: Wasserfallmodell, V-Modell, Spiralmodell, Scrum, eXtreme Programming
    - Nähere Details dazu folgen in den kommenden Einheiten.

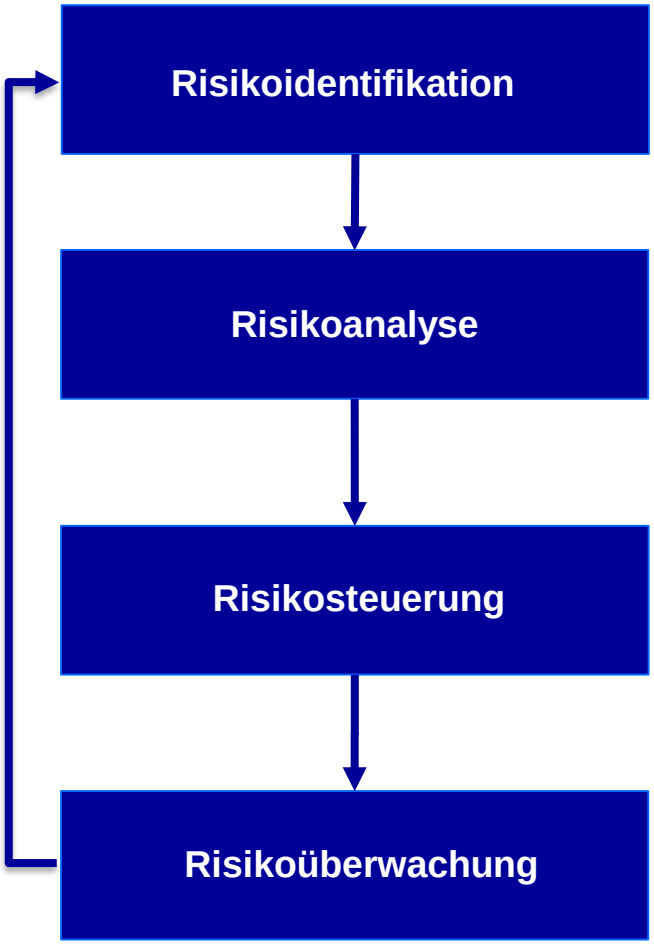
1.4 Struktur der Softwaretechnik: Zuordnung des Risikomanagements

- **Projektphasen:** Zuordnung des Risikomanagements

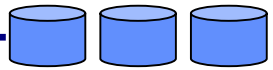


1.4 Struktur der Softwaretechnik: Phasen des Risikomanagements 1

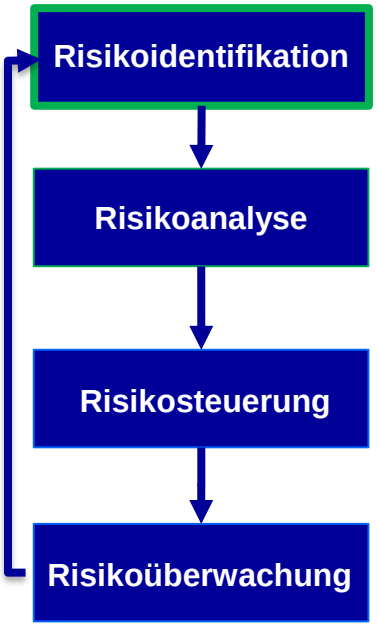
- **Phasen des Risikomanagements**



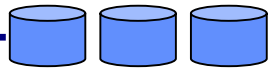
1.4 Struktur der Softwaretechnik: Phasen des Risikomanagements 2



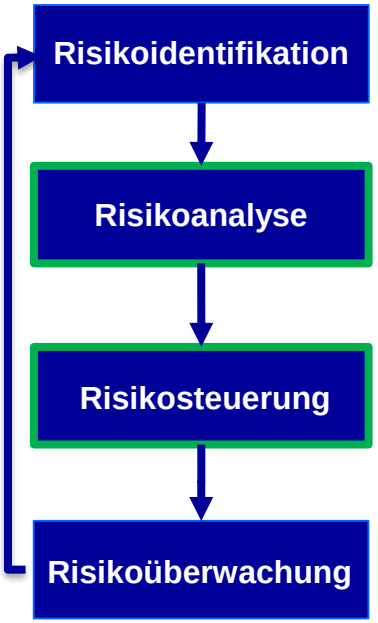
- **Risikoidentifikation:** Dient zur Ermittlung aller wesentlichen Risiken und Risikoquellen von Beginn bis zum Abschluss eines Projekts.
  - z.B. Risikoträchtige Entwicklungen, unzuverlässige Lieferanten, enge Termine, Abhängigkeiten einer Phase von vorhergehenden



1.4 Struktur der Softwaretechnik: Phasen des Risikomanagements 3



- **Risikoanalyse:** Setzt sich zusammen aus der
  - **Risikoqualifizierung:** Reale Einschätzung der Auswirkungen von Risiken auf den Geschäftsverlauf.
  - **Risikoquantifizierung:** Wie wahrscheinlich tritt das Risiko ein?
    - Falls sich ein Risiko realisiert, welche quantitative Auswirkungen (Zeit, Kosten) kommen ins Spiel
- **Risikosteuerung:** Festlegen der richtigen Maßnahmen
  - Aktive Beeinflussung der ermittelten Risiken
  - Für den Umgang mit identifizierten Risiken gibt es folgende Alternativen:



Präventive Maßnahmen

Vermeiden	Vermindern

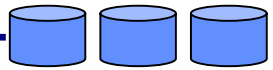
- Was tun um die Eintrittswahrscheinlichkeiten von Risiken zu verringern?
- Es muss entschieden werden wann/welche Maßnahmen einzusetzen sind.

Korrektive Maßnahmen

Verlagern	Akzeptieren

- Risiko ist bereits zum Problem geworden
- z.B. Mehraufwand, Strafzahlungen

1.4 Struktur der Softwaretechnik: Phasen des Risikomanagements 3



- **Risikoüberwachung:** Kontinuierliche Überwachung des Erfolges der vorgeschlagenen Steuerungsmaßnahmen.
  - Die Effektivität und Angemessenheit des Risikomanagements muss regelmäßig geprüft werden.
  - Dies ermöglicht es das Kosten-Nutzen-Verhältnis der Risikoüberwachung zu optimieren.
  - Überwachung von:

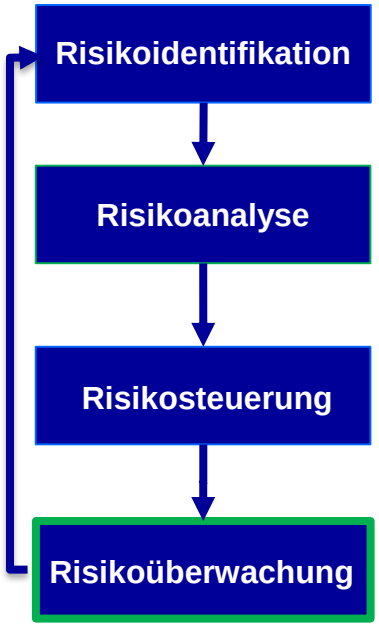
Bereits identifizierte Risiken

- Wie entwickeln sich diese im Laufe des Projekts? (Verringerung, Auflösung, Vergrößerung)

Neuen Risiken

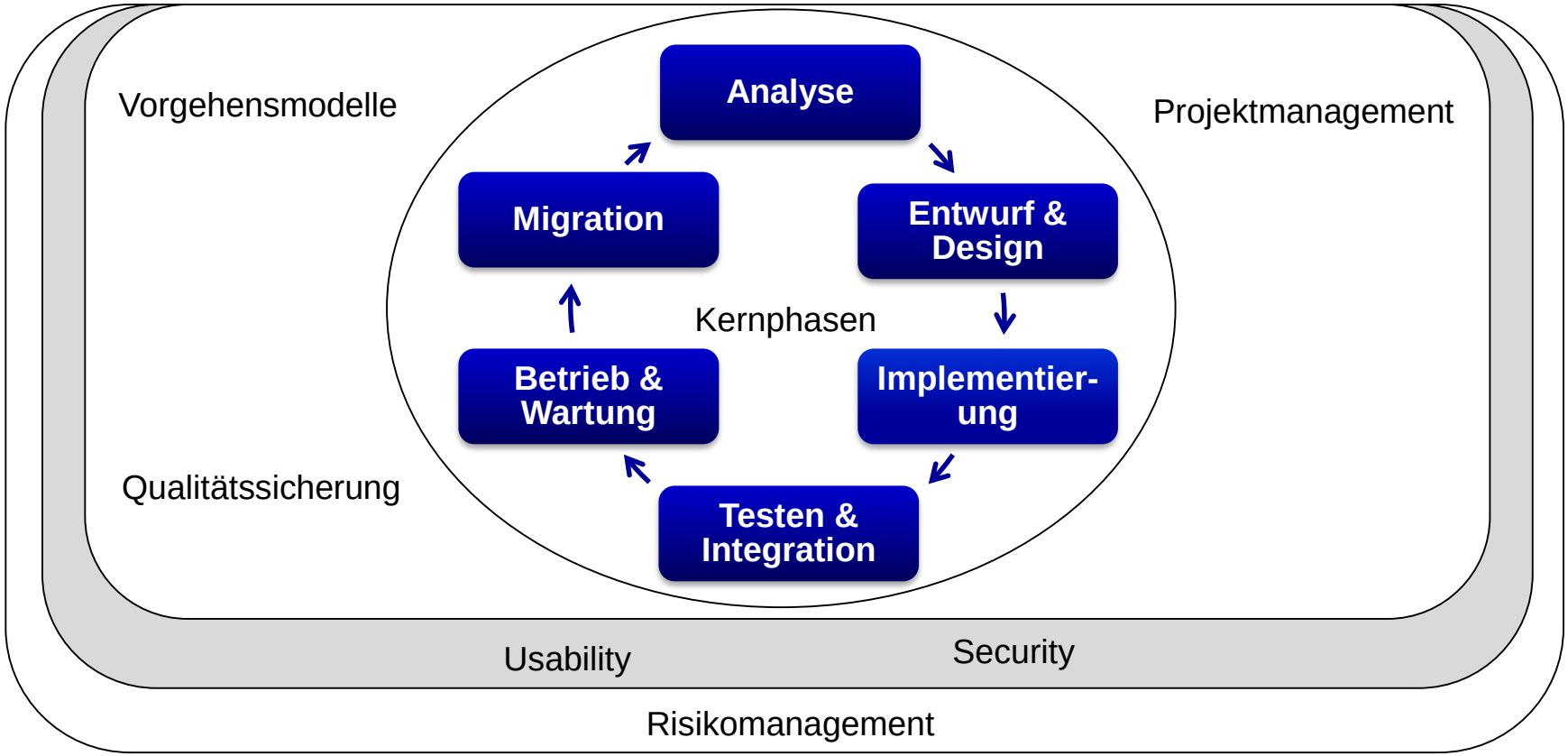
- völlig neue, bislang nicht absehbare Risiken tauchen auf

Erneute Risikoanalyse und Risikosteuerung

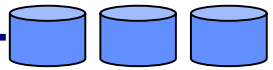


1.4 Struktur der Softwaretechnik: Projektphasen – Ausblick

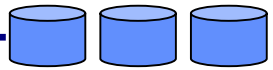
- **Ausblick:** Die kommenden Vorlesungseinheiten geben einen Einblick in die Inhalte ausgewählter Projektphasen.



Inhalt



1.5 Zusammenfassung



- **Software-Engineering:** Übergang von „jugendlicher“ Disziplin zur reifen Technik?
- **Software-Technik birgt insbesondere bei größeren Projekten einen hohen Komplexitätsgrad und stellt auch ein mögliches Risiko dar.**
- **Von überschaubaren Projekten, die man alleine umsetzt, bis hin zu großen Projekten mit vielen Beteiligten.**
- **Unerlässlich:**
  - Saubere Umsetzung aller Projektphasen
  - Erfassung und Definition von Anforderungen
    - Siehe die kommenden Kapitel
  - Kommunikation mit Stakeholdern
  - Eine gründliche Dokumentation ist zur Planung aber auch zur eigenen Absicherung notwendig