



Beschleunigte Softwaretechnik

Dr. Volker Jung

Accso GmbH, Januar 2015

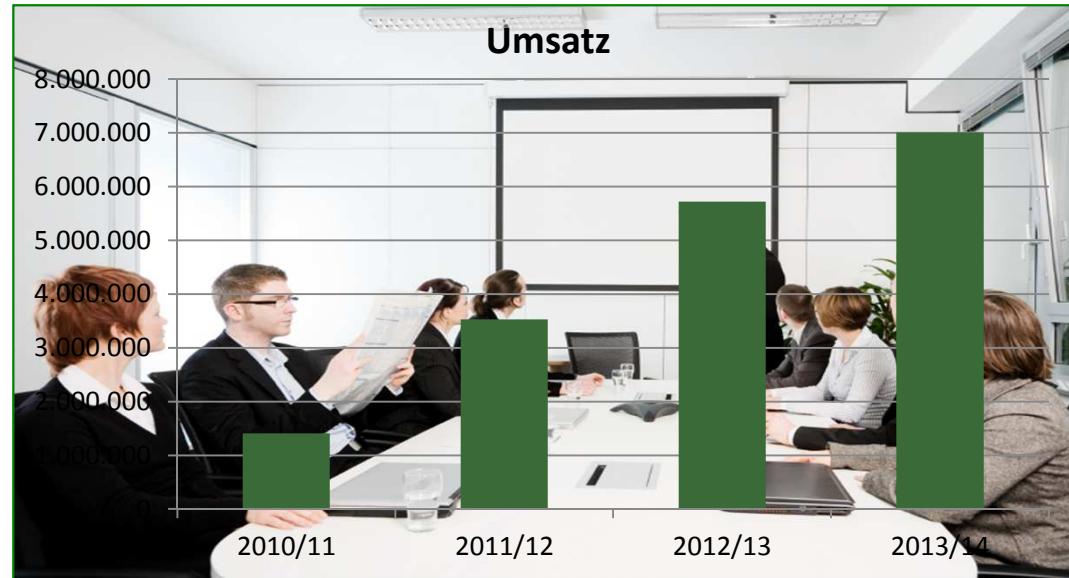
Agenda

- **Accso**

- Beschleunigte Softwaretechnik (BeST)
- Ausgewählte Aspekte
 - BeST-Practices im Projektmanagement
 - BeST-Practices der Analyse
 - BeST-Practices der Architektur
- Abschluss

Fakten

- ≡ Gegründet 2010 in Darmstadt
- ≡ GmbH mit Kapitalbeteiligung des Landes Hessen
- ≡ Geschäftsführung: Jürgen Artmann,
Dr. Markus Voß
- ≡ Niederlassungen: Darmstadt, Köln und
München



Unsere Mitarbeiter

- ≡ 78 Mitarbeiter (Stand 1.1.2015)
- ≡ Software-Ingenieure und IT-Berater
- ≡ Exzellent ausgebildet (fast alle Mitarbeiter
haben einen Hochschulabschluss, fast jeder
vierte mit Promotion)
- ≡ sehr erfahren (> 80% mit mehr- bzw.
langjähriger Berufserfahrung)
- ≡ hoch motiviert



Unsere Kunden





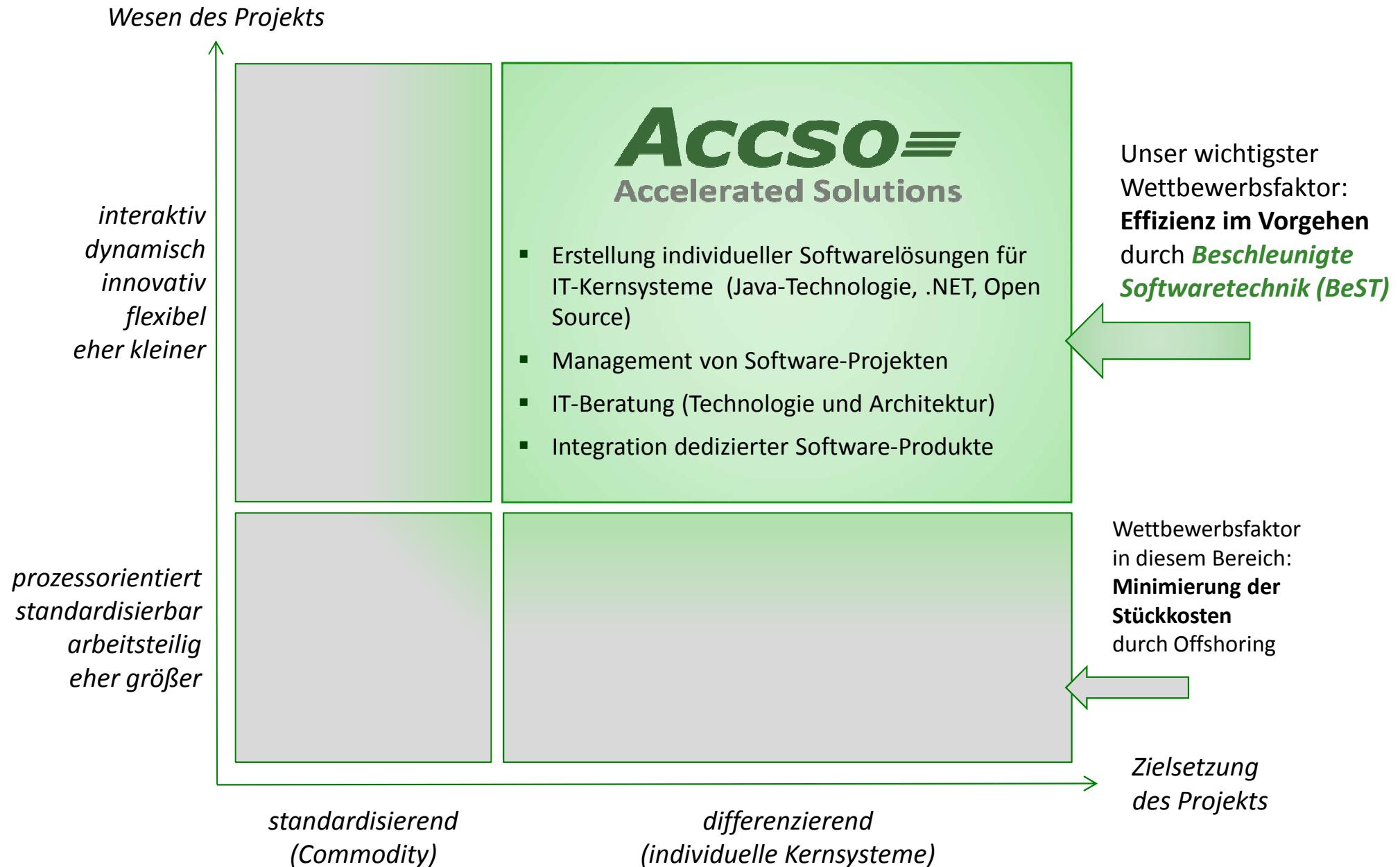
Unser Angebot

- ≡ Individuelle Softwarelösungen für IT-Kernsysteme (Schwerpunkte: Java/JEE, .NET, Open Source, RIA, u.a.)
- ≡ Technologie- und Architektur-Beratung (Schwerpunkte: IT-Architektur, IT-Modernisierung, EAM, SOA, BPM, Security, u.a.)
- ≡ Management von Software-Projekten
- ≡ Integration dedizierter Software-Produkte (Schwerpunkte: CMS, Portal)

Unsere Kunden

- ≡ Wir entwickeln individuelle Softwarelösungen für die Kernkompetenzen unserer Kunden und beraten in aktuellen Fragestellungen von Technologie und Architektur.
- ≡ Accso arbeitet branchenübergreifend für namhafte Unternehmen, dazu zählen Deutsche Telekom, Deutsche Bahn, DFS Deutsche Flugsicherung, Deutsche Wetterdienst, Bundesamt für Wirtschaft und Ausfuhrkontrolle, DekaBank, AOK, ZDF, SWR, HR, WDR, DuMont, HRS, Maxdome, Vodafone und weitere.

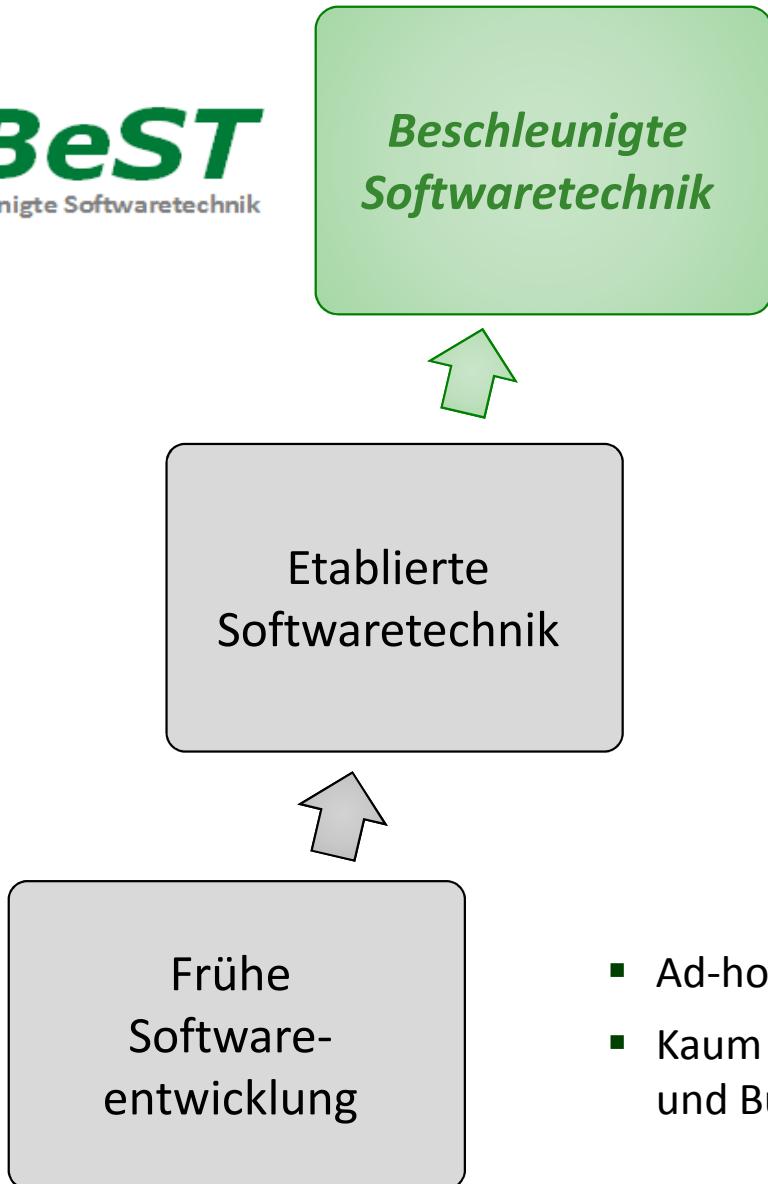
Individuelle Kernsysteme in dynamischen Umfeldern entwickeln wir sicher, mit hoher Qualität und besonders effizient



Agenda

- AccSO
- **Beschleunigte Softwaretechnik (BeST)**
- Ausgewählte Aspekte
 - BeST-Practices im Projektmanagement
 - BeST-Practices der Analyse
 - BeST-Practices der Architektur
- Abschluss

Beschleunigte Softwaretechnik (BeST) entwickelt etablierte Softwaretechnik weiter - hin zu mehr Effizienz und Effektivität.

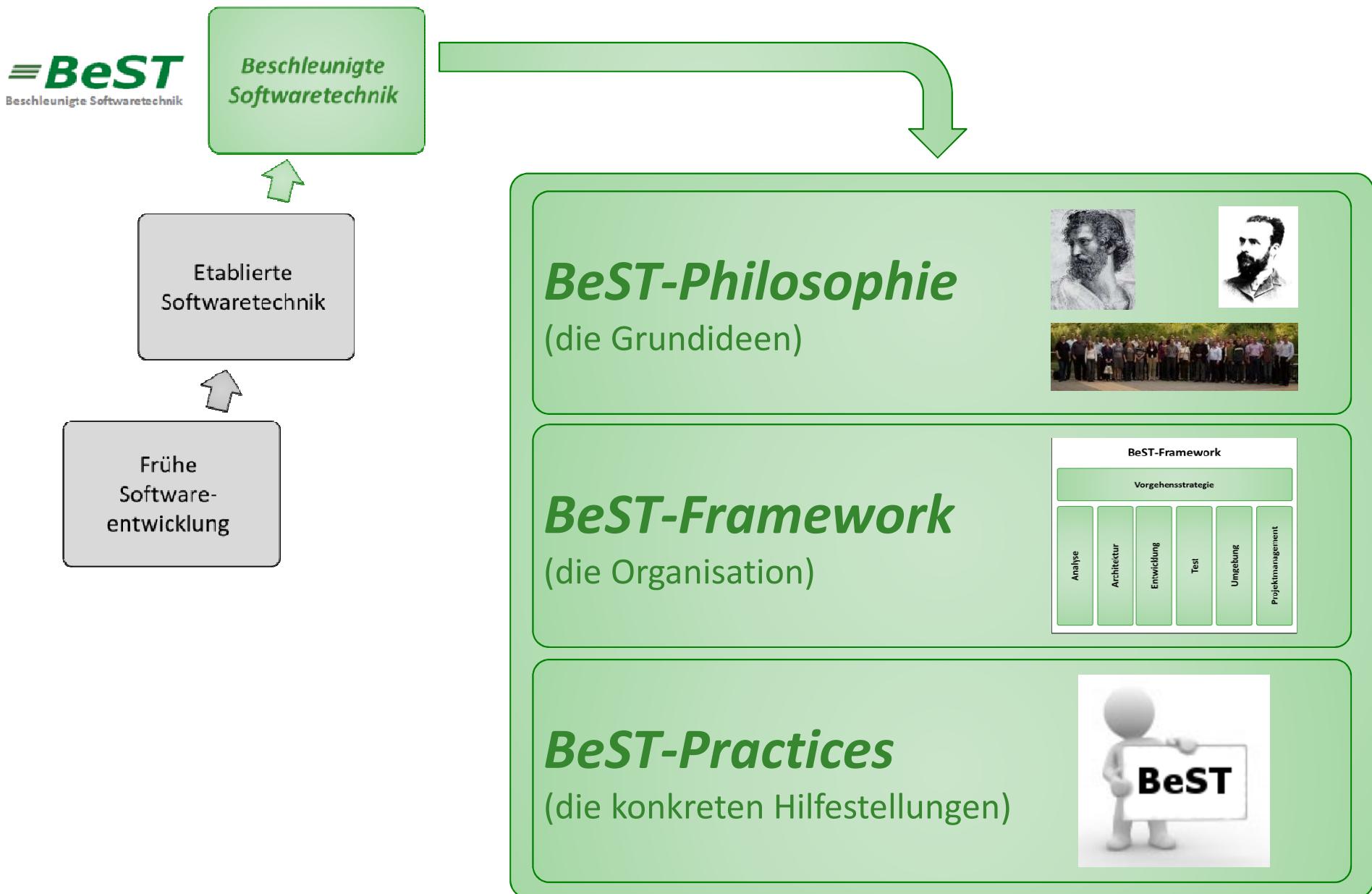


- Basiert auf etablierter Softwaretechnik
- Betont besonders die Aspekte Effizienz und Effektivität
- Ganzheitlich (optimiert alle Dimensionen), Angemessen („quick“, aber nicht „dirty“), Team-Orientiert (fordert exzellente Skills)

- Definierte Vorgehensweisen und Prozesse
- Standardisierte Methoden, Architekturen, Programmiersprachen, Komponenten, Frameworks und Werkzeuge
- Wiederholbar, messbar, steuerbar

- Ad-hoc Vorgehensweisen, keine Standardisierung
- Kaum Aussagen über Ergebnisqualität und Einhaltung von Zeit- und Budgetvorgaben möglich

BeST beinhaltet eine eigene Philosophie, ein Framework als Ordnungsrahmen und eine Sammlung von Best Practices.



BeST-Philosophie: Unser Verständnis von **Effizienz** und **Effektivität**

Allgemein:

$$\text{Effizienz} = \frac{\text{Ertrag}}{\text{Aufwand}}$$

In Software-Projekten:

$$\text{Ertrag} = \text{Effekt} = \text{Kundennutzen bzw. Kundenzufriedenheit}$$

$$\text{Effizienz} = \frac{\text{Effekt}}{\text{Projektaufwand}}$$

1

Es geht nicht um eine möglichst große Menge an Code oder Doku bei gleichem Aufwand oder um irgendeine Lösung bei möglichst geringem Aufwand.

Es geht um eine den eigentlichen Kundenanforderungen **angemessene** Lösung bei möglichst geringem Aufwand.

2

Zudem ist überhaupt nur dann mit einer signifikanten Steigerung der Effizienz und Effektivität zu rechnen, wenn das Projekt **ganzheitlich** in allen Aspekten auf Effizienz getrimmt wird.

3

Und da die Umsetzung eines Projekts auf diese Art anspruchsvoll ist, funktioniert das nur in einem **Team** mit den richtigen **Skills**.

BeST-Philosophie: Die Leitplanken



Ganzheitlichkeit

Angemessenheit

Team-Orientierung

BeST-Philosophie: **Ganzheitlichkeit**



Aristoteles

„Das Ganze ist mehr, als die Summe seiner Teile.“

Anspruch:
Beschleunigte Softwaretechnik ist ganzheitlich.

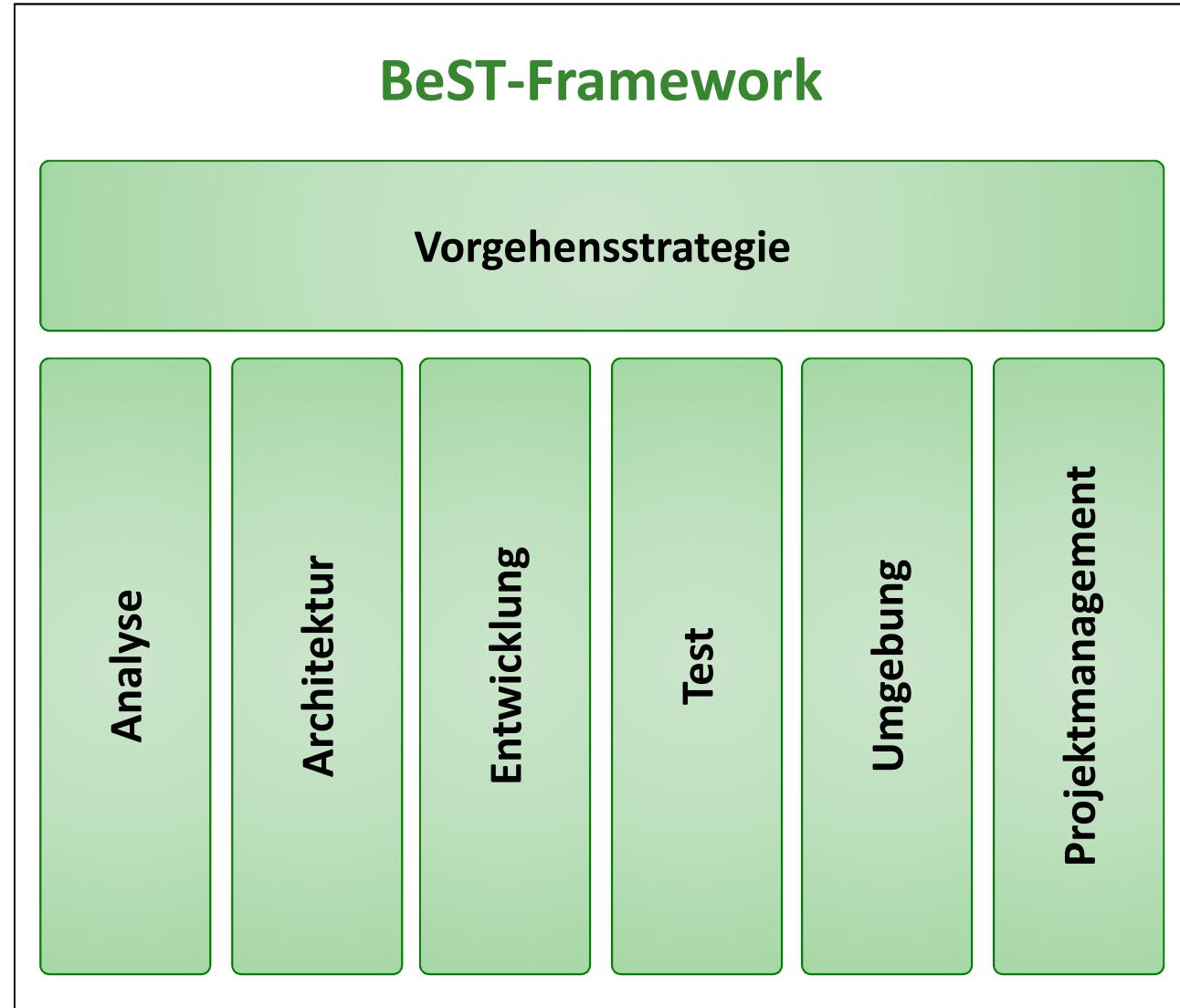
Sie berücksichtigt
alle Domänen der Softwaretechnik,
über den gesamten Lebenszyklus der Lösung,

inkl. der zwischen den einzelnen Domänen und den
einzelnen Phasen existierenden Beziehungen
hinsichtlich ihrer Potenziale zur Effizienzsteigerung.

Genau daraus ergibt sich ein wesentlicher Teil
ihres Mehrwerts.

BeST-Framework: Ein **Ordnungsrahmen** für die Beschleunigte Softwaretechnik

- Sechs fachliche Domänen für die Organisation des disziplinbezogenen fachlichen Wissens
- Eine querschnittliche Domäne für die Organisation des disziplinübergreifenden und kontextabhängigen Wissens



BeST-Framework: Die fachlichen Domänen

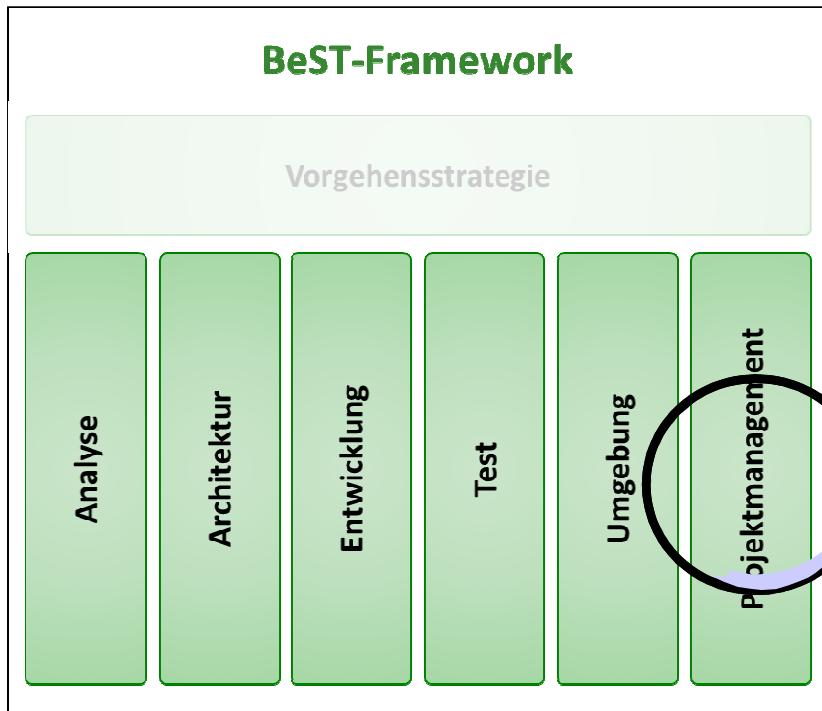
- Wir betrachten alle Domänen der Softwaretechnik im Sinne von Hauptaufgabenbereichen
- Das impliziert keine zeitliche Abfolge der einzelnen Tätigkeiten im Sinne eines Wasserfallmodells sondern ist unabhängig vom konkreten Vorgehensmodell

BeST-Philosophie:
Ganzheitlichkeit



BeST-Framework: Die fachlichen Domänen

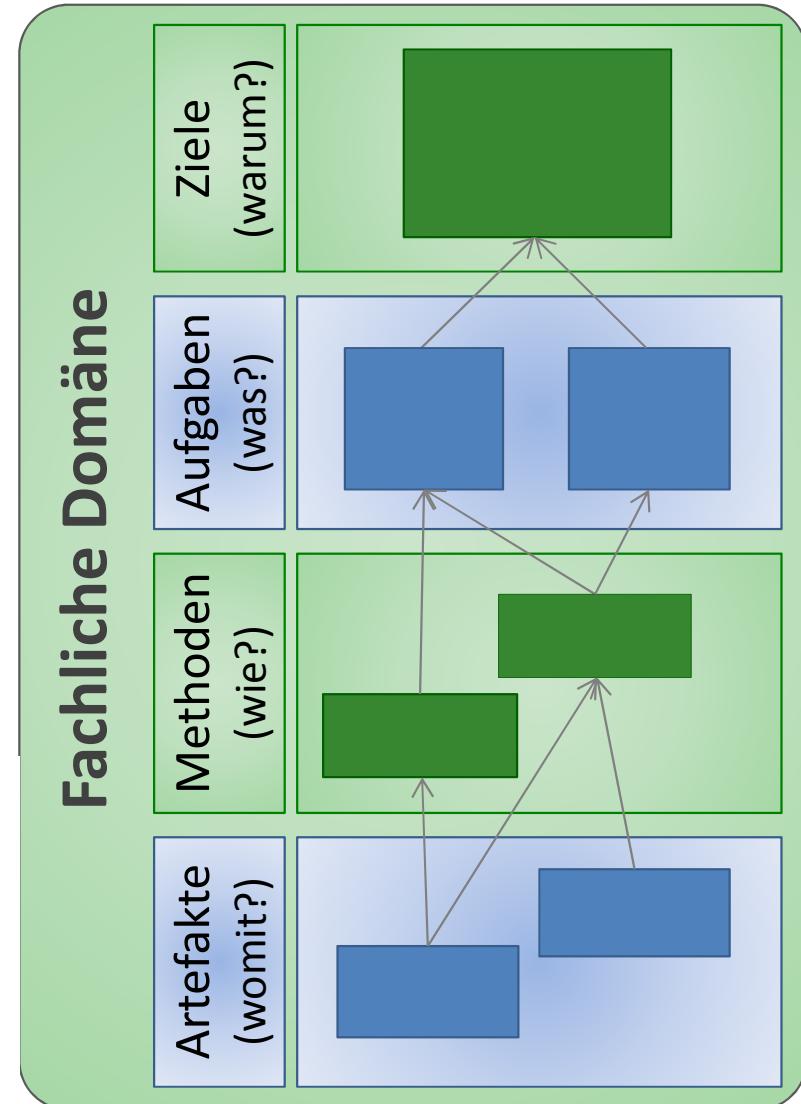
Überblick



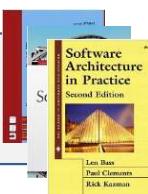
Für alle fachlichen Domänen enthält das BeST-Framework

- die **Ziele** der Domäne,
- die **Aufgaben**, um diese Ziele zu erreichen,
- die **Methoden**, um diese Aufgaben zu erledigen und
- die **Artefakte**, die bei Anwendung dieser Methoden entstehen,

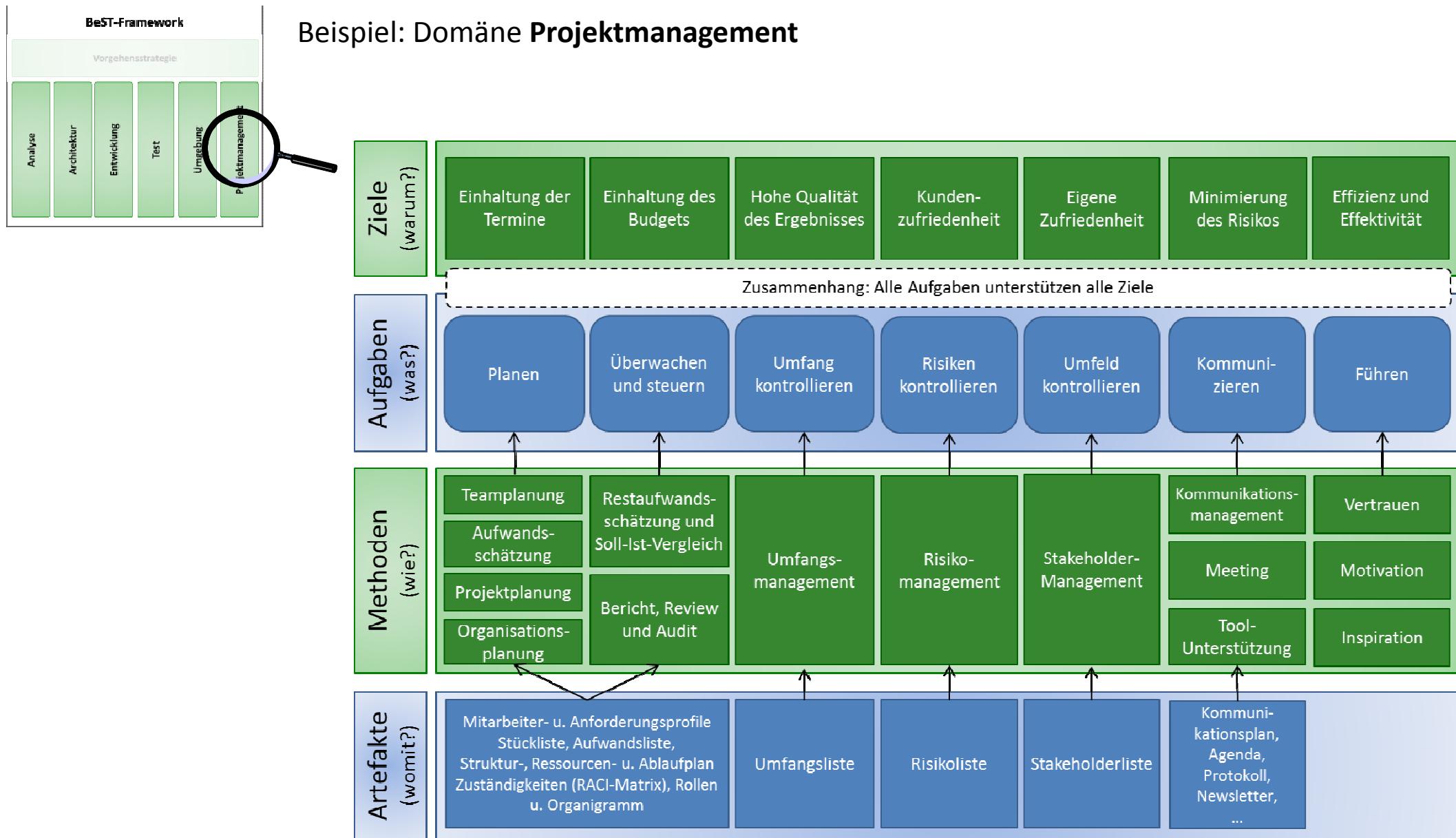
flankiert durch einen Überblick und Referenzen.



Referenzen



BeST-Framework: Die fachlichen Domänen

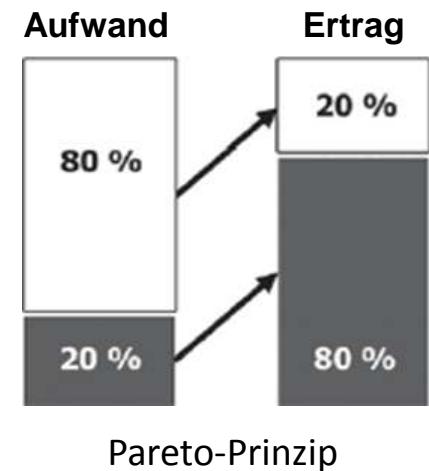
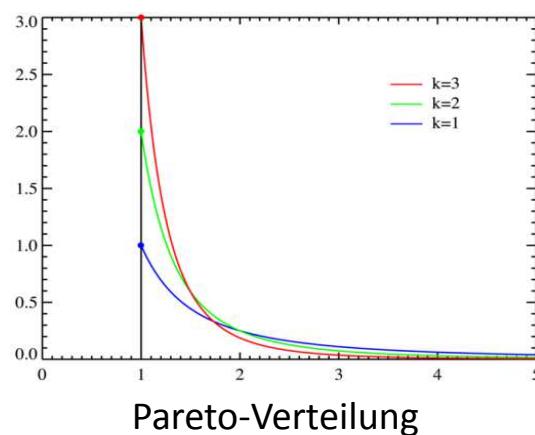


BeST-Philosophie: Angemessenheit



Vilfredo Pareto

Anspruch:
Beschleunigte Softwaretechnik ist angemessen und pragmatisch.



"Es ist immer wieder verblüffend, wie viele Dinge wir tun, die, wenn wir sie nicht mehr tun, keinem abgehen." (Peter Drucker)

"Do first priority things first, do second priority things never" (Jack Welch)

BeST-Philosophie: Unser Verständnis von **Agilität**

Agilität ist keine spezielle Methode und kein spezielles Vorgehensmodell.
Agilität ist eine Vorgehens- und Management-Philosophie.

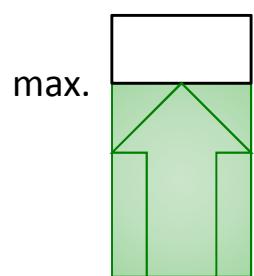


VS.

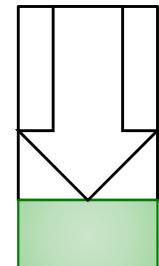


Agil	Generalstabsmäßig
Projekte können gar nicht vollständig vorab durchgeplant und Produkte nicht vollständig vorab spezifiziert werden. Deswegen muss ein Projekt iterativ umgesetzt werden und auf Änderungen, die auf jeden Fall kommen werden, muss flexibel reagiert werden können.	Projekte müssen top-down durchgeplant werden, bevor sie durchgeführt werden. Produkte müssen vollständig spezifiziert werden, bevor sie umgesetzt werden.
Die besten Lösungen entstehen aus sich selbst organisierenden Teams.	Teams brauchen eine Organisation mit klar definierten Rollen und Zuständigkeiten.
Es herrscht eine Kultur des Vertrauens.	Alle relevanten Dinge sind vertraglich zu regeln.

BeST-Philosophie: Effizienz und Effektivität, Angemessenheit und Agilität kombiniert



*Effizienz und Effektivität erzielen wir durch Angemessenheit im Sinne einer **situationsgerechten** und **projektspezifischen Auswahl** aus den verschiedenen Handlungsalternativen:
So agil wie möglich, so generalstabsmäßig wie nötig.*



min.

Das entspricht auch dem eigentlichen Geist des „Agilen Manifests“

“We have come to value:

Individuals and interactions	over	processes and tools
Working software	over	comprehensive documentation
Customer collaboration	over	contract negotiation
Responding to change	over	following a plan

So viel
wie
möglich

So viel
wie
nötig

That is, while there is value in the items on the right,
we value the items on the left more”

BeST-Philosophie: Agilitätsprofil

Jedes Projekt ist anders.

Projektkontext und Einflussfaktoren unterscheiden sich von Projekt zu Projekt.

Entsprechend gibt es für jedes Projekt ein spezifisches optimales Agilitätsprofil im Hinblick auf Effizienz und Effektivität seiner Durchführung.

Agilitätsprofil (Reglermodell)

Keine Vorab-Planung



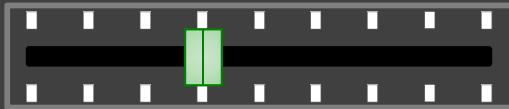
Vollständige Vorab-Planung

Keine Vorab-Spezifikation



Vollständige Vorab-Spezifikation

Vollständige Selbstorganisation



Strikte Organisation und Rollenteilung

Volles Vertrauen



Kein Vertrauen

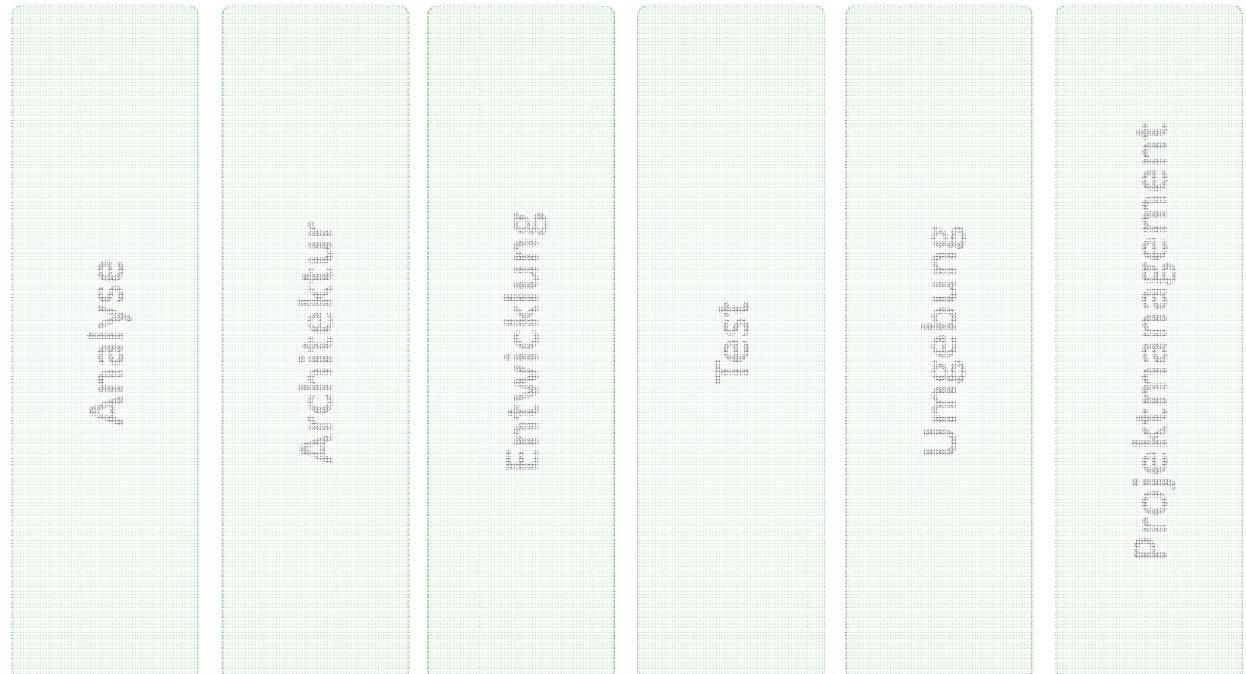
BeST-Framework: Vorgehensstrategie

BeST-Philosophie:
Angemessenheit

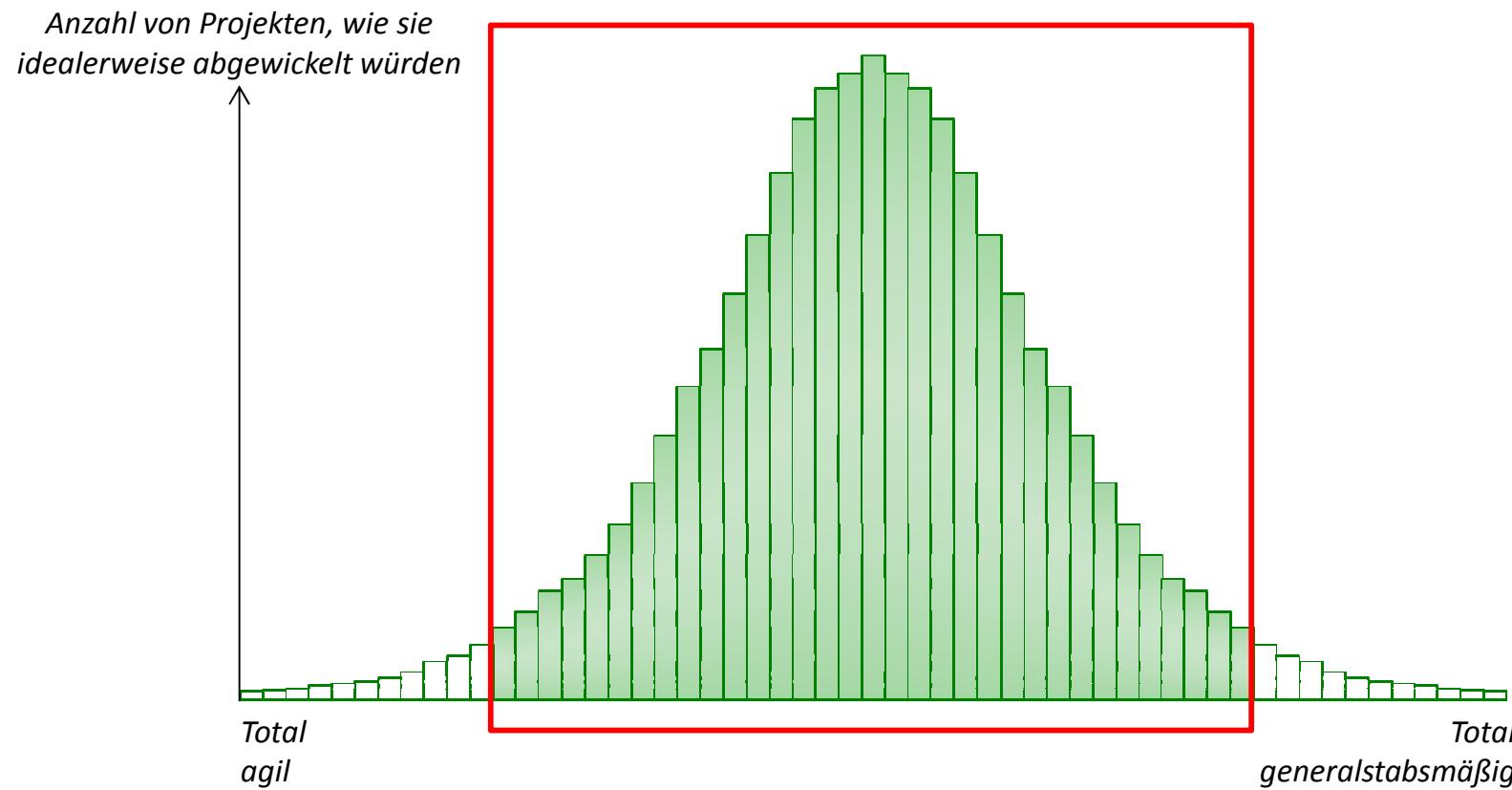
- Mit den Mitteln der Vorgehensstrategie wird das konkrete Projektvorgehen abhängig vom **Projektkontext**, d.h. den speziell für das Projekt geltenden **Einflussfaktoren** optimal im Sinne der Effizienz und Effektivität zugeschnitten
- Dazu wird des **Agilitätsprofil** ermittelt und es werden aus den einzelnen fachlichen Domänen **Methoden ausgewählt und spezifisch zugeschnitten**
- Als Ergebnis entsteht das konkrete **Vorgehensmodell**

BeST-Framework

Vorgehensstrategie



BeST-Framework: Bottom-Line Erkenntnisse



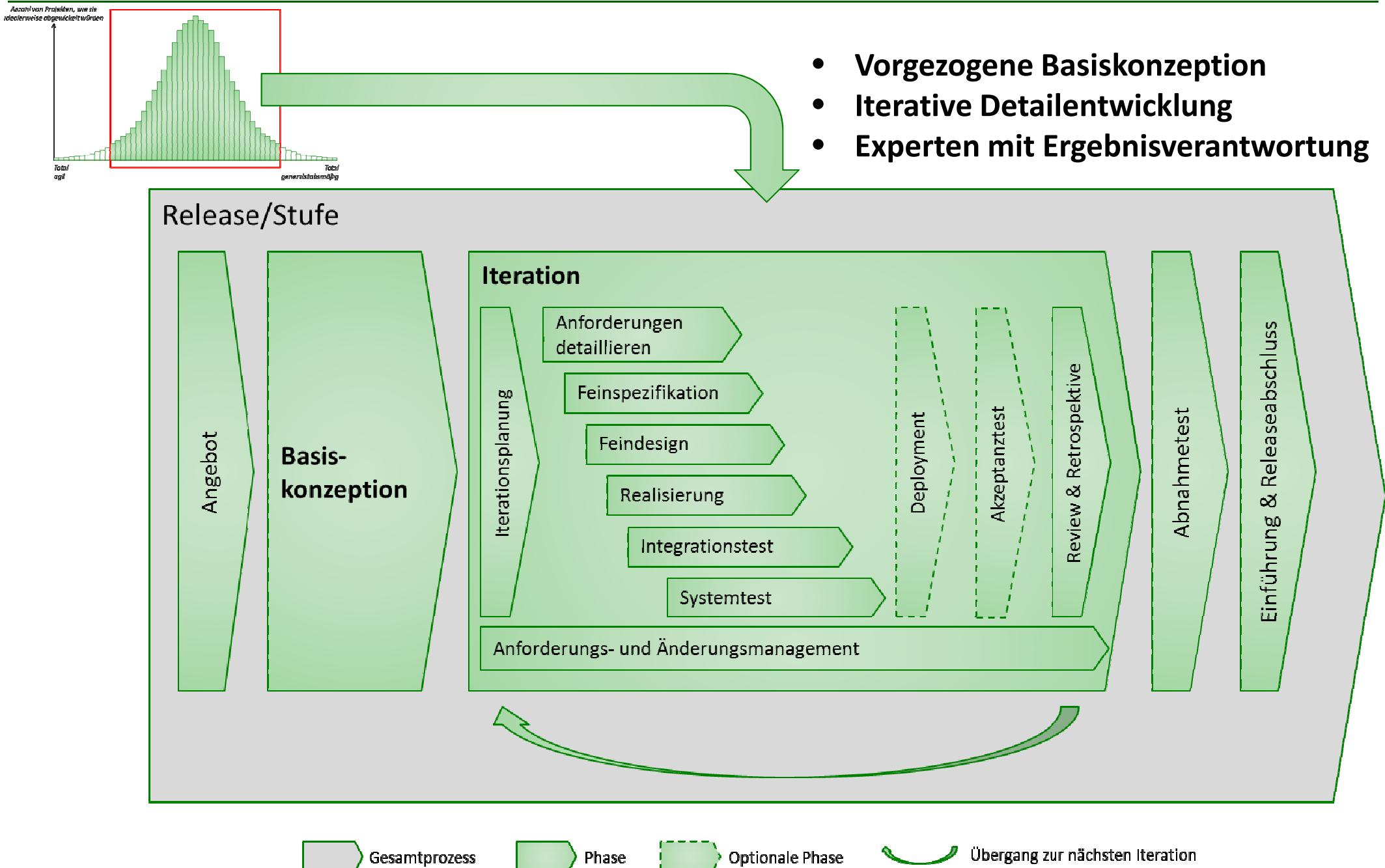
Für 95% aller Software-Projekte gilt:

Grundlegende Aspekte der Planung und Spezifikation sollten in einer frühen Projektphase **vorab** festgelegt werden (Basiskonzeption bzw. Envisioning).

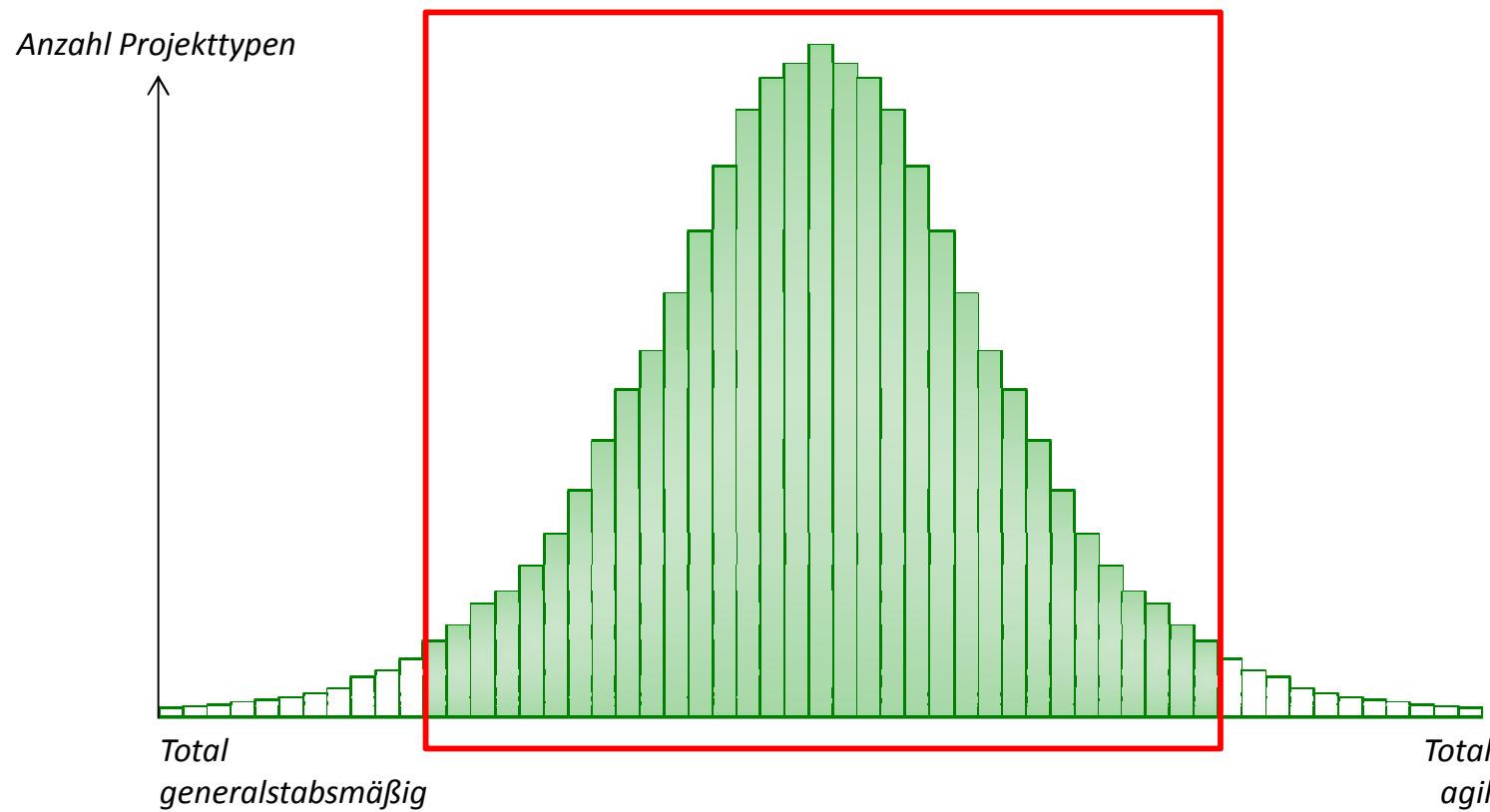
Die Entwicklung der **Details** sollte danach **iterativ** erfolgen (in Iterationen bzw. Sprints).

Grundlegende Aufgaben der Planung und Spezifikation sollten von **Experten** ihres Fachs durchgeführt oder unterstützt werden, die damit auch **Ergebnisverantwortung** übernehmen.

BeST-Framework: Das Basis-Vorgehensmodell



BeST-Framework: Bottom-Line Erkenntnisse



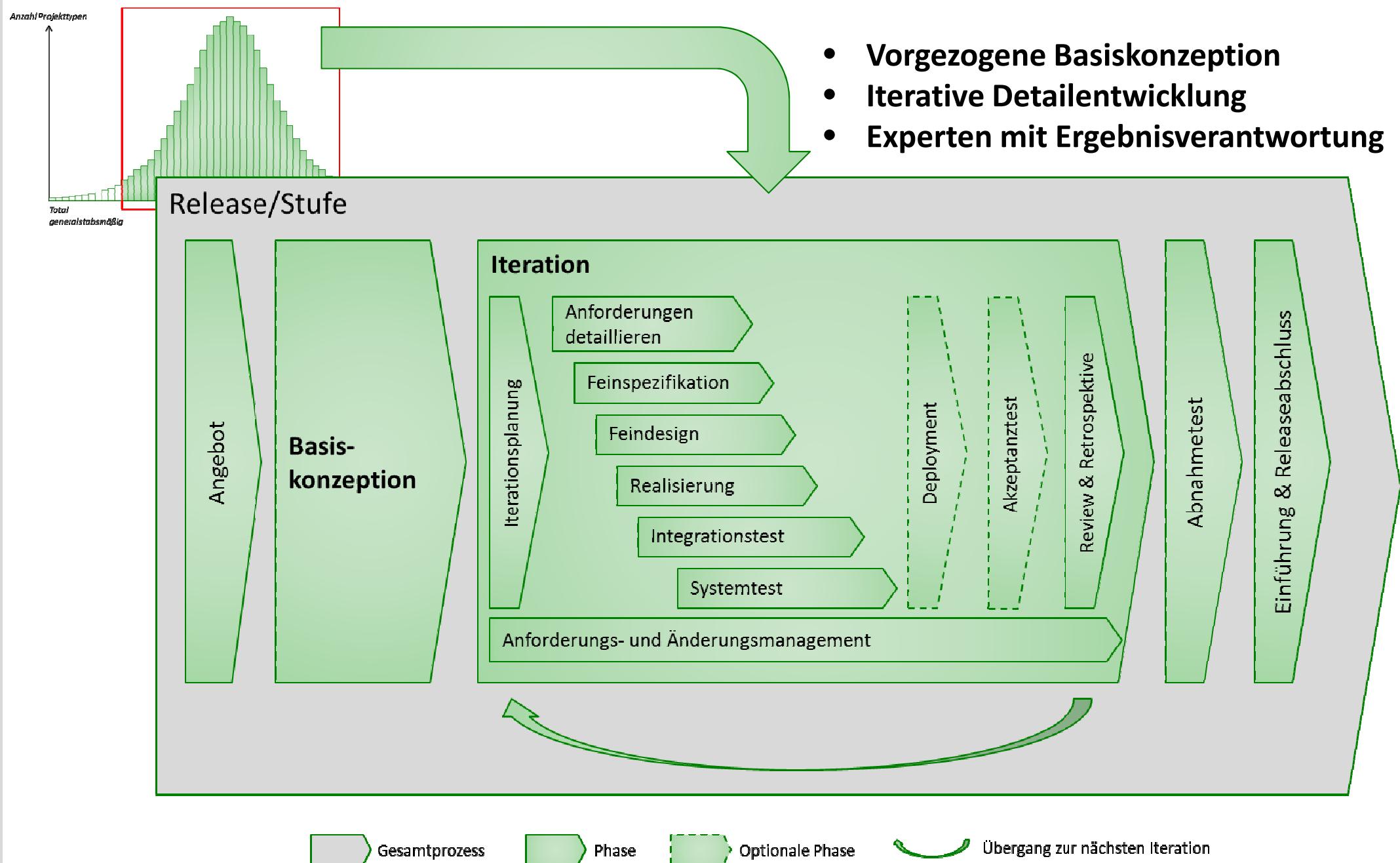
Für 95% aller Software-Projekte gilt:

Grundlegende Aspekte der Planung und Spezifikation sollten in einer frühen Projektphase **vorab festgelegt werden (Basiskonzeption bzw. Envisioning).**

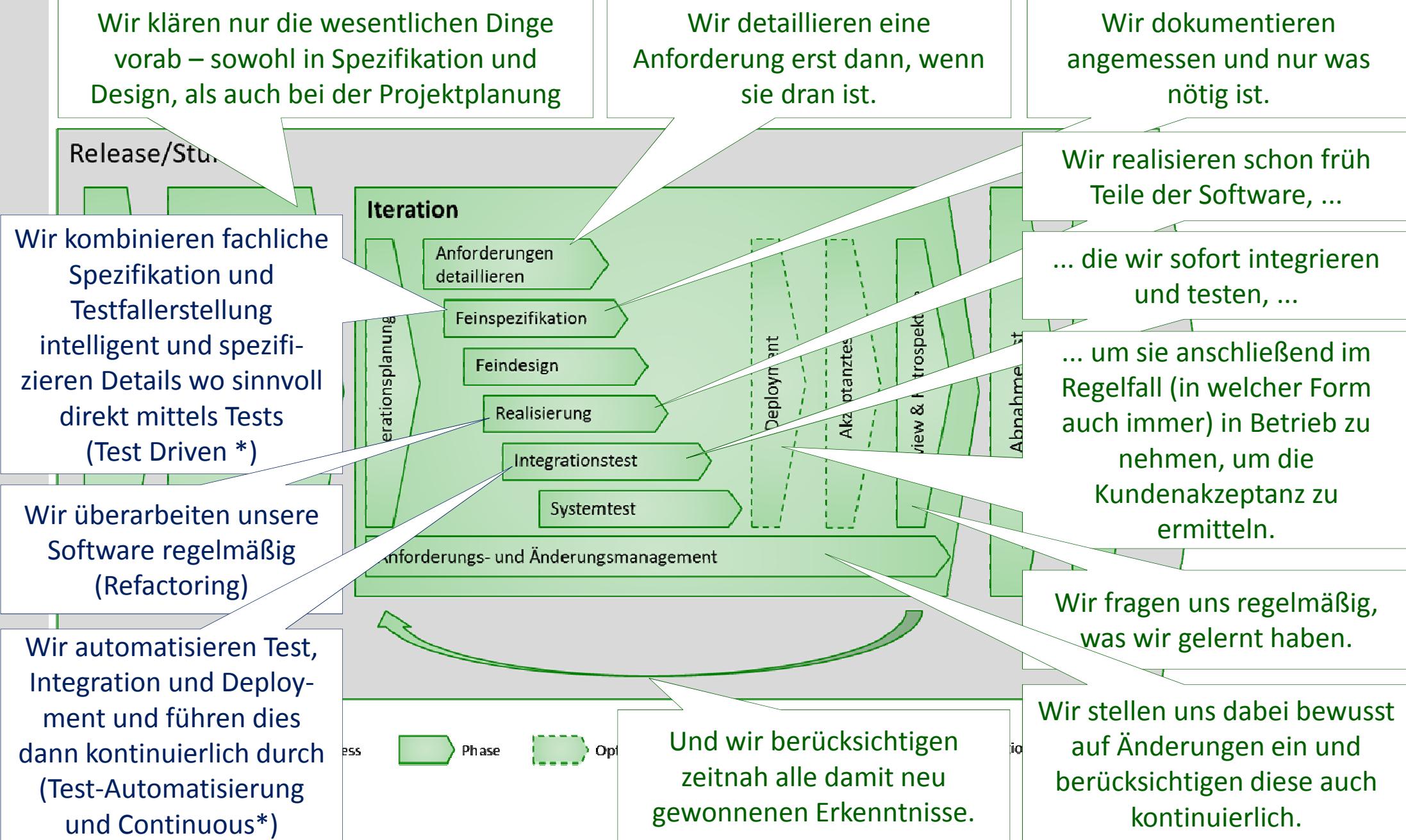
Die Entwicklung der **Details** sollte danach **iterativ** erfolgen (in Iterationen bzw. Sprints).

Grundlegende Aufgaben der Planung und Spezifikation sollten von **Experten** ihres Fachs durchgeführt oder unterstützt werden, die damit auch **Ergebnisverantwortung** übernehmen.

BeST-Framework: Das Basis-Vorgehensmodell



BeST-Framework: Agiles Vorgehen und andere effiziente Techniken



BeST-Practices: Die eigentlichen Hilfsmittel

Überblick

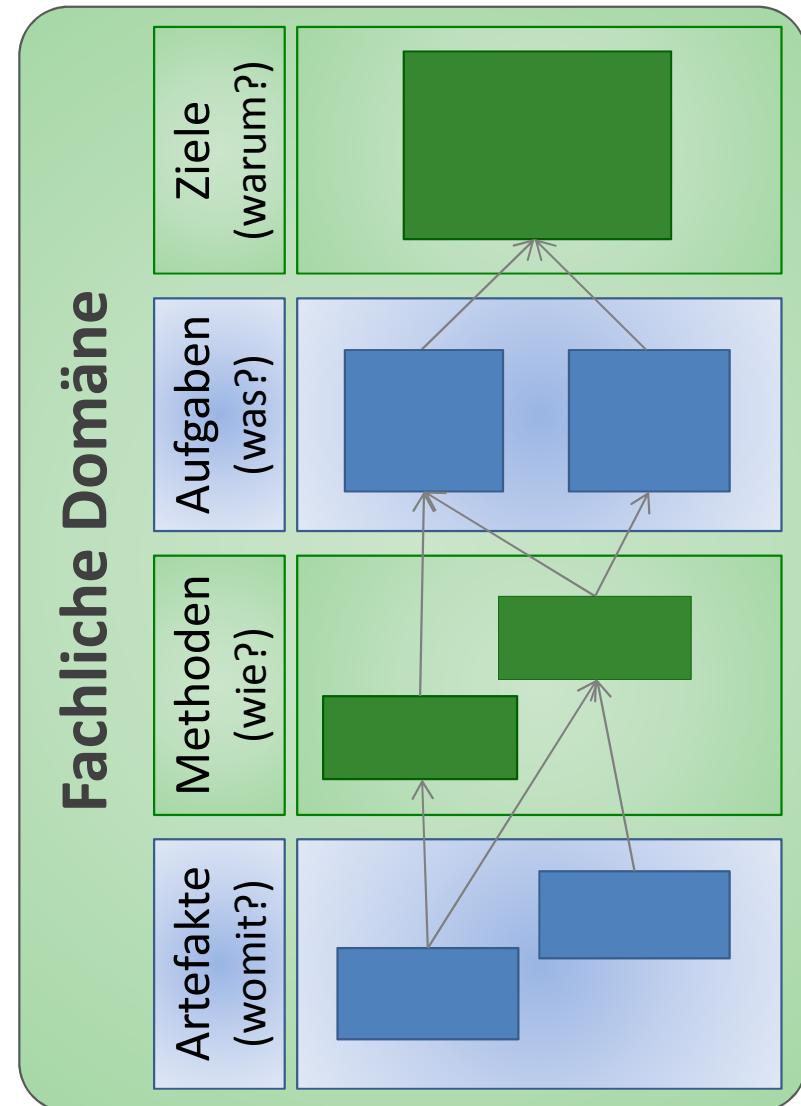


- BeST-Practices ...

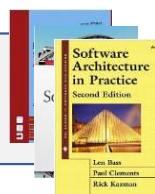
... unterstützen **konkret** bei der Auswahl geeigneter Methoden zur Lösung einer Aufgabe

... geben **konkrete** Hinweise, wie die gewählte Methode genau anzuwenden ist

... und beziehen sind dabei oft auf einen speziellen Projektkontext bzw. sind nur in diesem wirklich „best“.



Referenzen

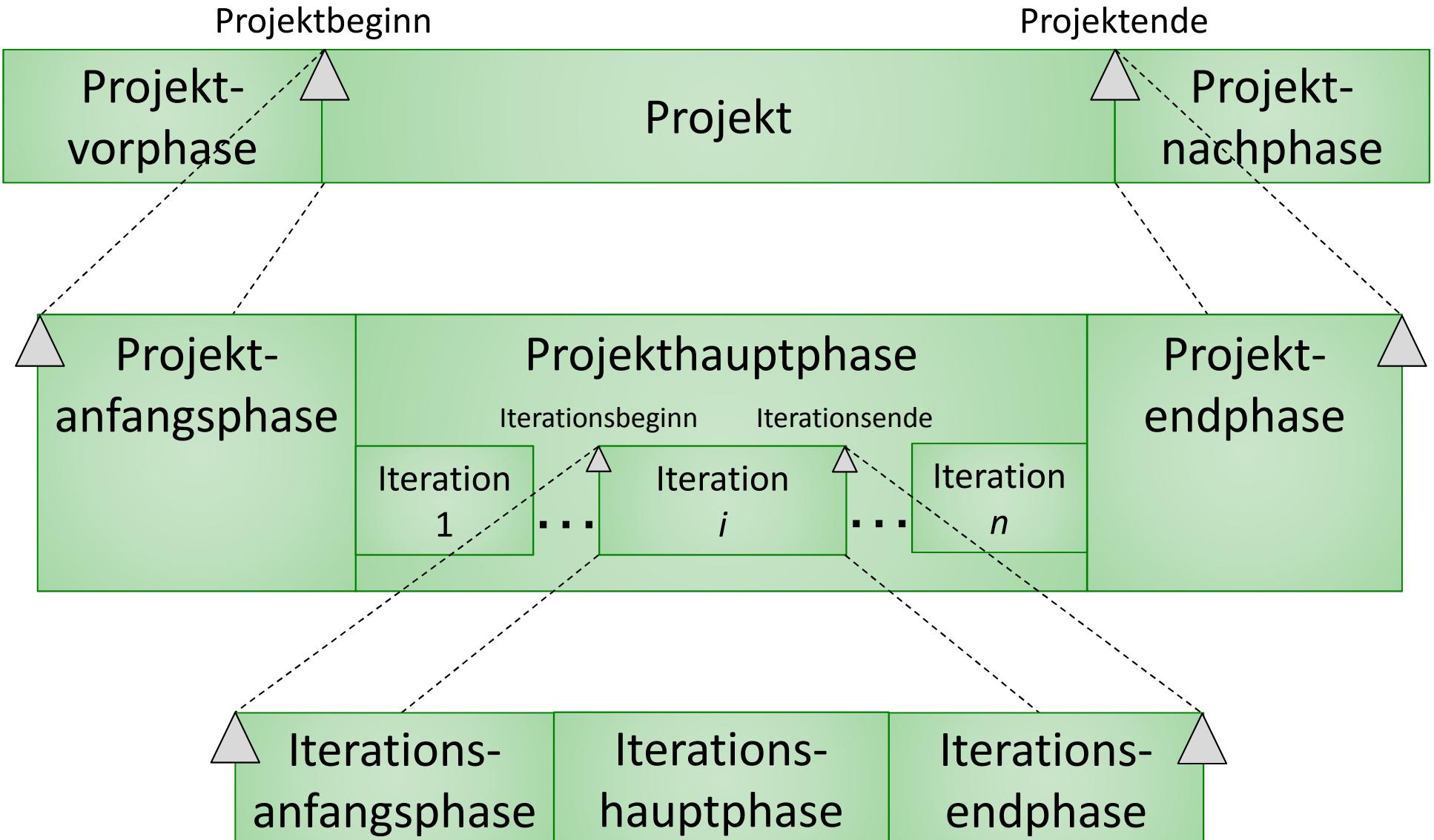


BeST-Practices: Strukturierbar als **Musterkataloge**



Aspekt	Bedeutung	Beispiel
Tätigkeit	Referenz auf eine Tätigkeit aus einer bekannten Methode	„Spezifikation von Use-Cases“
Nutzen	Konkretisierung der Tätigkeit im Hinblick auf die Verwendung der Ergebnisse	„als Vorlage für die Entwicklung der Software“ oder „als Grundlage zur Abstimmung der Fachlichkeit mit dem Kunden“
Akteur	Qualifikation oder über spezifische Zuständigkeiten definierte Projektrolle	„der Chef-Architekt“
Durchführungszeitpunkt	Konkretisierung, in welchen typischen Projektphasen die Tätigkeit durchgeführt wird (s. Phasenraster)	„in einer Basiskonzeptphase vor Beginn der iterativen Entwicklung“
Ausprägung	Weitere Details zur näheren Beschreibung der Tätigkeit	„grob auf Ebene der Hauptakteure“ oder „notiert mittels UML-Use Case-Diagrammen“
Kontext	Menge von Einflussfaktoren, unter denen die Practice effizient ist	„bei einem Product-Owner mit IT Know-how“

BeST-Practices: Durchführungszeitpunkte normieren: Das Projektphasenraster



BeST-Practices: Details finden sich in den Katalogen der BeST-Foundation

BeST-Foundation

Ziele	Beschleunigung (Effizienz und Effektivität)	Minimierung des fachlichen Risikos	Hohe Qualität der Analyseergebnisse	Hohe Zufriedenheit der Kunden	Gemeinsames Verständnis der Fachlichkeit	Hohe Zufriedenheit der Anwender-Mitarbeiter	Gegenseitiges Vertrauen
Aufgaben	Fachliche Risiken identifizieren und behandeln						
Methoden	Anforderungen analysieren	Fachliche Begriffe definieren	Fachliche Abgrenzungen vornehmen	Fachliches Umfeld managen	Mit fachlichen Interessensvertretern und Projektteam kommunizieren	Fachliche QM-Meetings durchführen	Funktionen und Testurteile ermitteln
Artefakte	Glossar erstellen	Anforderungen priorisieren und bewerten	Ein-/Ausgabeverhalten beschreiben	Workshops durchführen	Rückmeldungen einholen	Alzeptanzkriterien beschreiben	
	Anforderungen analysieren	Fachkonzept	Leistungsbeschreibung	Meetings halten	Ergebnisse präsentieren	Testscenarien beschreiben	
	Anforderungen spezifizieren	Ist-Analyse durchführen	Dialoge visualisieren	Interviews durchführen	Arbeitsablauf begleiten	Kreativitätstechniken	Artefakte reviewen
	Geschäftsprozessmodell	Oberflächenkonzept	Anwendungshandbuch	Workshop-Artefakte	Scenarien durchspielen	Testfälle spezifizieren	Testfälle spezifizieren
	Ergebnisse dokumentieren		Checklisten		Review-anmerkungen		Testartefakte
Ziele	Stabilität und Flexibilität	Angemessenheit von Weg und Lösung	Nachvollziehbarkeit von Entscheidungen	Minimierung von Risiken	Hohe Qualität der Lösung	Beschleunigung (Effizienz und Effektivität)	
Aufgaben	Orientierung geben und Leitplanken etablieren						
Methoden	Anforderungen berücksichtigen	Lösungsarchitektur entwerfen	Dokumentieren und Kommunizieren	Prüfen und bewerten	Beraten und begleiten		
Artefakte	Architekturprinzipien festlegen	Architekturstrategie festlegen	Qualitätsziele für Architektur festlegen	Messkriterien für Architektur definieren	Richtlinien für Entwicklung festlegen		
	Anforderungen bewerten, priorisieren und verfolgen	Lösungsideen entwickeln	Architektururteile erstellen	Architekturreviews durchführen (lassen)			
	Rahmenbedingungen und Einflussfaktoren beachten	Architektur strukturiert entwerfen	Architektur angemessen dokumentieren	Architektur szenarienbasiert bewerten (lassen)			
	Risiken identifizieren und einschätzen	Technische Konzepte, Technologien, Produkte auswählen	Architektur im Team etablieren und entwickeln	Erfüllung von Architekturrichtlinien prüfen			
	Machbarkeit, Kosten und Nutzen betrachten	Aufwand schätzen und Planung unterstützen	Projektsteuerung und -organisation unterstützen	Implementierung (begleiten)	Testbarkeit und Kritikabilität bewerten		
Ziele	Evolvierbarkeit	Korrektheit - Erfüllung der Anforderungen	Lauffähige Software	Robustheit und Sicherheit	Produktionseffizienz	Hohe Qualität der Lösung	
Aufgaben	Feindesign erstellen	Code schreiben	Entwicklertests und Qualitäts sicherung	Entwicklungsprozess etablieren	Analyse von bestehendem Code		
Methoden	Generatoren und DSLs	Systemverständnis	Coding Standards etablieren	Parallelentwicklung	Analysewerkzeuge einsetzen	Entwicklungsprozess dokumentieren	
Artefakte	Convention over Configuration	RAD	Refactoring	Issue-Tracking			
	Feindesign dokumentieren	Collective Ownership	Taktischer Durchstich	Testgetriebene Entwicklung			
		Pair Programming	Code-Reviews	Entwicklungsprozess dokumentieren			
		Effiziente Werkzeuge	Codemetriken	Leichtgewichtige Unit-Tests			

Analyse

Ziele	Kundenzufriedenheit	Eigene Zufriedenheit	Effizienz und Effektivität	Hohe Qualität des Ergebnisses	Vertrauen/Risikominimierung
Aufgaben	Teststrategie definieren	Test planen, steuern und Ergebnisse berichten	Test vorbereiten	Test durchführen	Test abschließen und archivieren
Methoden	Priorisierung	An Standards orientieren (IEEE 829)	Testarten	Testcaten-bestimmung	Testreview
Artefakte	Test in Teststufen	Testfallauswahl	Testentwurf-verfahren	Solidataten-bestimmung	Qualitätsethos
	Test First / TDD	Mocking and Stubbing	Testdurchführungsverfahren	Testdurchführungsverfahren	
	Fehlermanagement	Testmetriken			
	Qualitätsziele	Teststrategie	Fehlerstatus-modell	testbare Eigenschaften	Automatisierter Testfall
	Vereinbarung zum Abnahmeverfahren	Testbasistabelle	Testplan	Testfälle	Testumgebung
	Testabnahmeverfahren	Priorisierungsmatrix	Testfortschrittsbericht	Testdaten	Testumgebung
	Testkarte	Testkonzept	Testergebnisse	Sollergebnisse	Bugreport

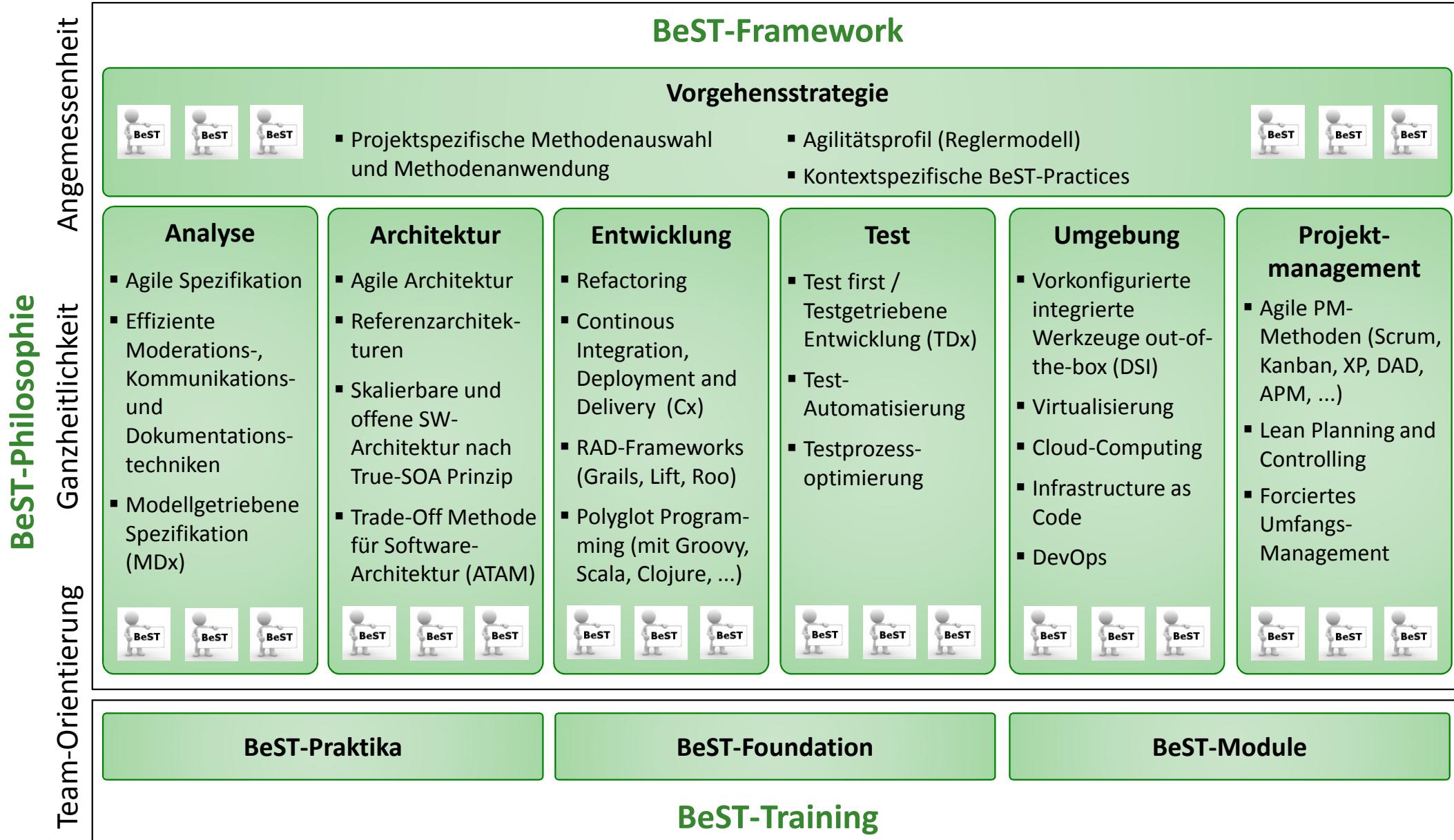
Test

Ziele	Unterstützung aller Domänen	Integrierte Toolsets	Effizienz und Beschleunigung	Nachvollziehbarkeit und Reproduzierbarkeit	Risikominimierung	Verständlichkeit/Klarheit/Einfachheit (KISS)
Aufgaben	Infrastruktur berestellen	Dienste berestellen	Standardisieren	Automatisieren	Toolauswahl unterstützen	Angemessen dokumentieren
Methoden	Virtualisierung	DevOps	Automatisierter Buildprozess	Pflege von Werkzeuglisten		
Artefakte	Cloud Computing	Infrastructure as Code	Continuous Integration	Continuous Delivery	Continuous Deployment	Verwaltung von Tutorials
	Systeme	Dienste	Software-lizenzen	Buildskripte	Dokumentationen	
	Buildumgebung	Issue Tracker	Version-Control-System	Teknologie-bausteine	Tutorials	
	Testumgebung	Wiki	Continuous Integration Tool	Projekt-grundgerüste	Werkzeuglisten (mit Bewertungen)	
	Bugreport					

Umgebung

Ziele	Einhaltung der Termine	Einhaltung des Budgets	Hohe Qualität des Ergebnisses	Kundenzufriedenheit	Eigene Zufriedenheit	Minimierung des Risikos	Effizienz und Effektivität
Aufgaben	Planen	Überwachen und steuern	Umfang kontrollieren	Risiken kontrollieren	Umfeld kontrollieren	Kommunizieren	Führen
Methoden	Teamplanung	Aufwands-schätzung und Sell-List-Vergleich	Projektplanung	Bericht, Review und Audit	Umfangs-management	Risiko-management	Kommunikations-management
Artefakte	Mitarbeiter- u. Anforderungsprofile	Stückliste, Aufwandsliste, Struktur, Ressourcen u. Ablaufplan	Zuständigkeiten (RACI-Matrix), Rollen u. Organigramm	Meeting	Vertrauen	Stakeholder-Management	Motivation
						Tool-Unterstützung	Inspiration

Projektmanagement



Agenda

- Accso
- Beschleunigte Softwaretechnik (BeST)
- Ausgewählte Aspekte
 - **BeST-Practices im Projektmanagement**
 - BeST-Practices der Analyse
 - BeST-Practices der Architektur
- Abschluss

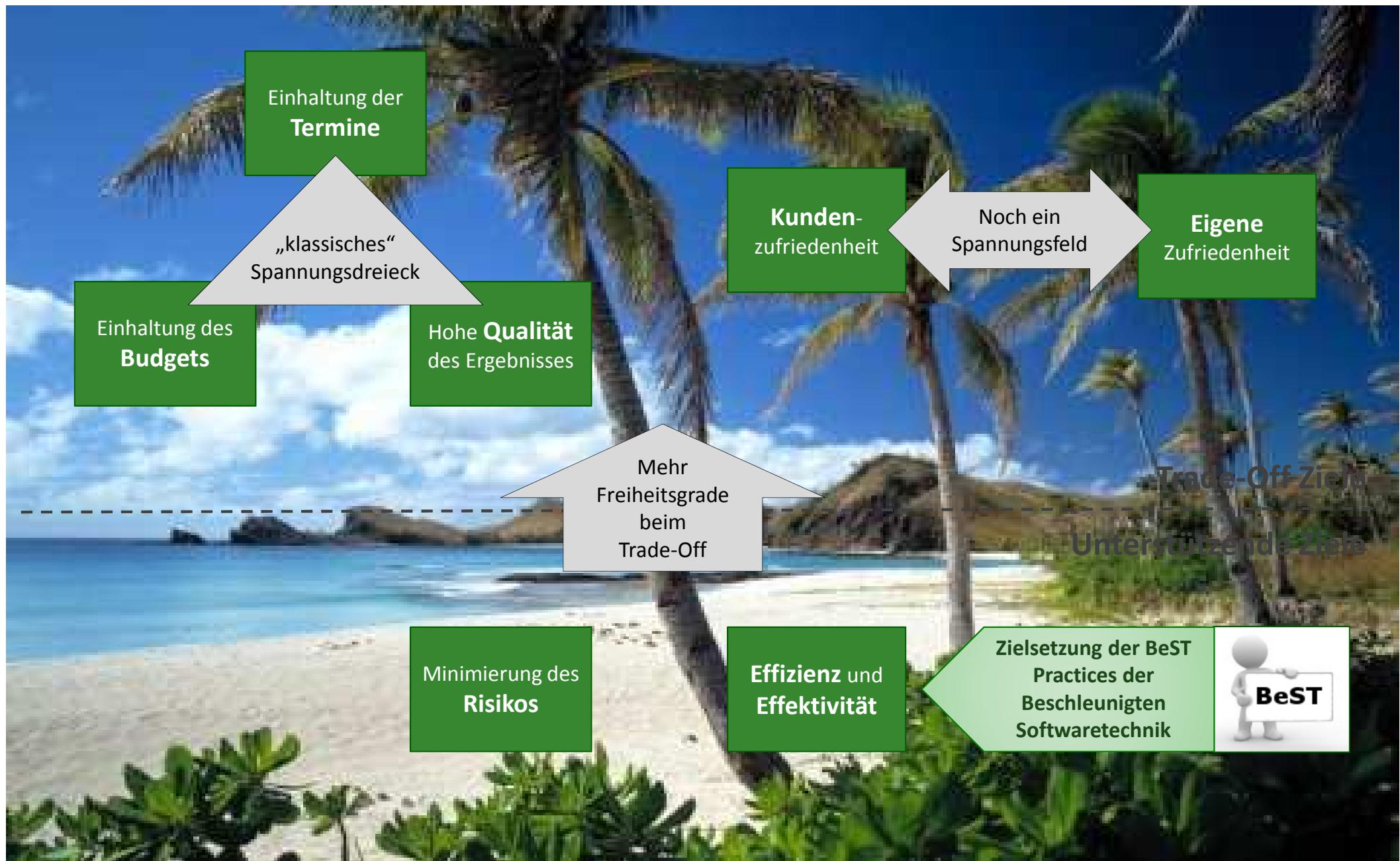
Im Projekt sind mehrere Leute unterwegs zu einem mehr oder weniger klaren Ziel und immer kann Unvorhergesehenes passieren.



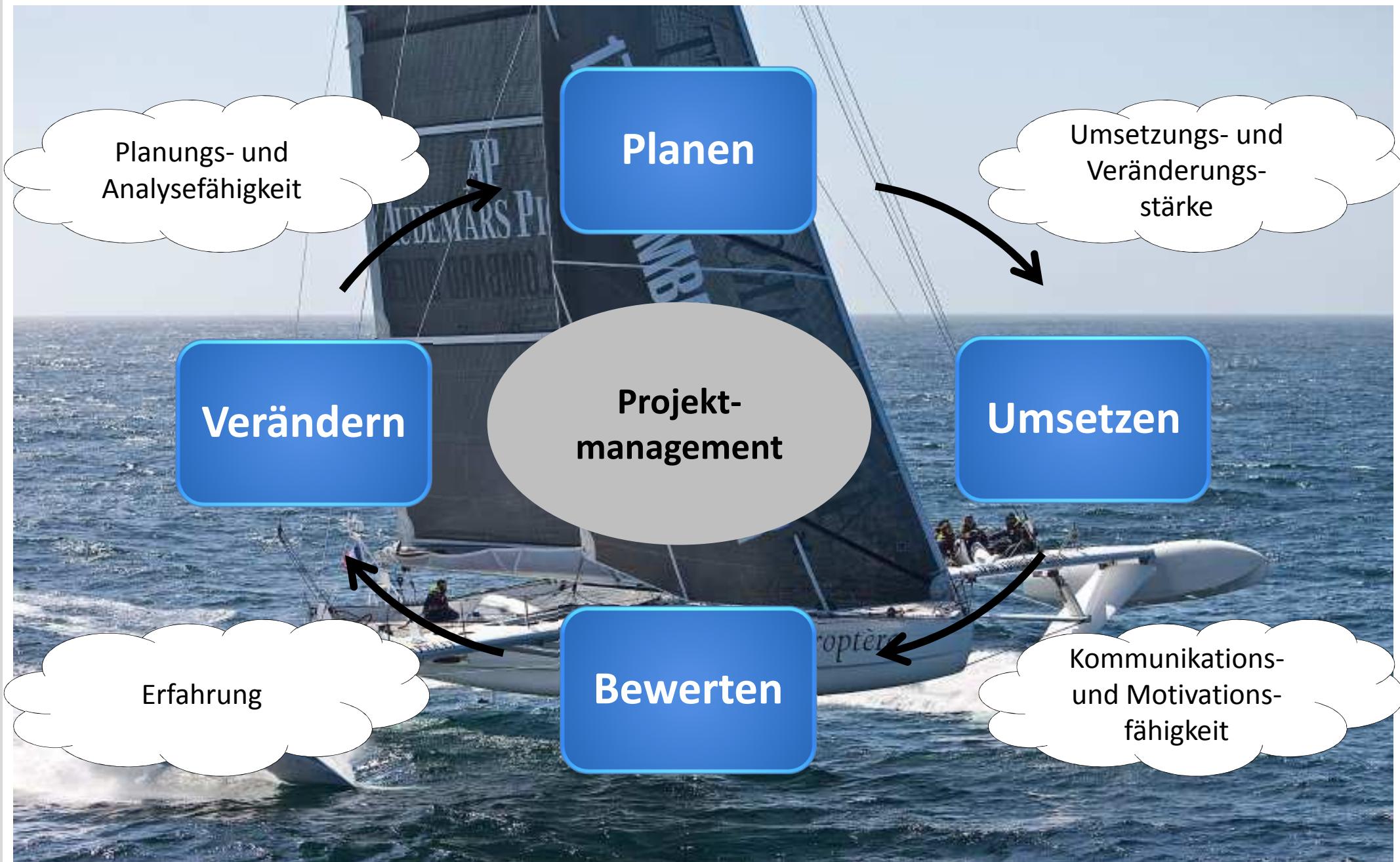
Die Aufgabe des Projektmanagements ist es, das Schiff bestmöglich ans Ziel zu bringen.



Die Ziele im Projektmanagement sind teilweise gegenläufig. Die Kunst ist, einen möglichst guten Trade-Off zu finden.

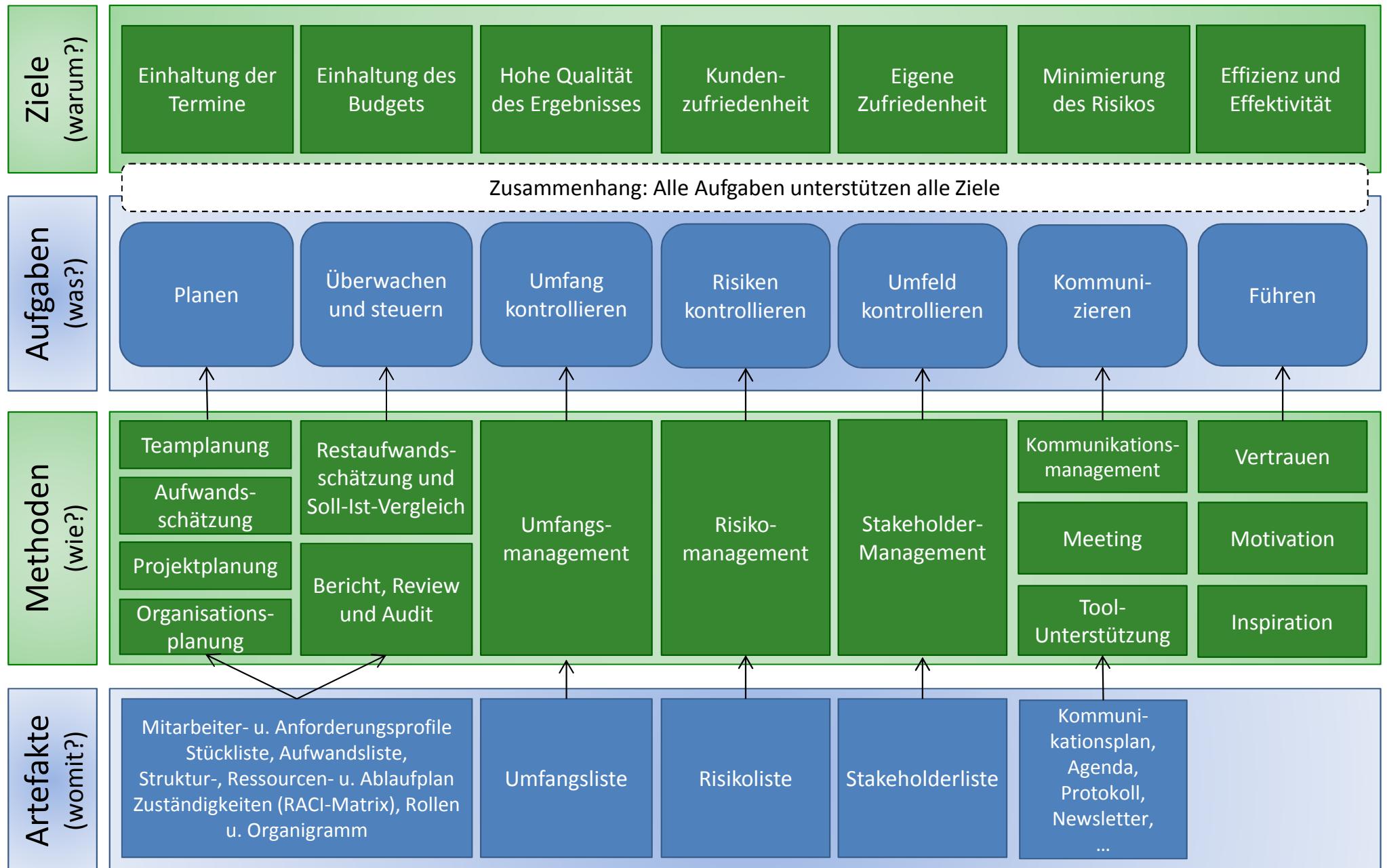


Projektmanagement funktioniert dabei wie ein Regelkreis. Entsprechende methodische und persönliche Fähigkeiten werden benötigt.



BeST-Domäne Projektmanagement:

Ziele, Prinzipien, Methoden, Artefakte





BeST-Practices sorgen für ein effektives und effizientes Projektmanagement (1/2)

Teamplanung:

- Auf benötigte Skills achten
- Keine faulen Kompromisse
- Trau keinem unter 70
- Verschiedene „Brillen“
- Verschiedenartigkeit zulassen
- Gute Teams zusammenhalten

Aufwandsschätzung:

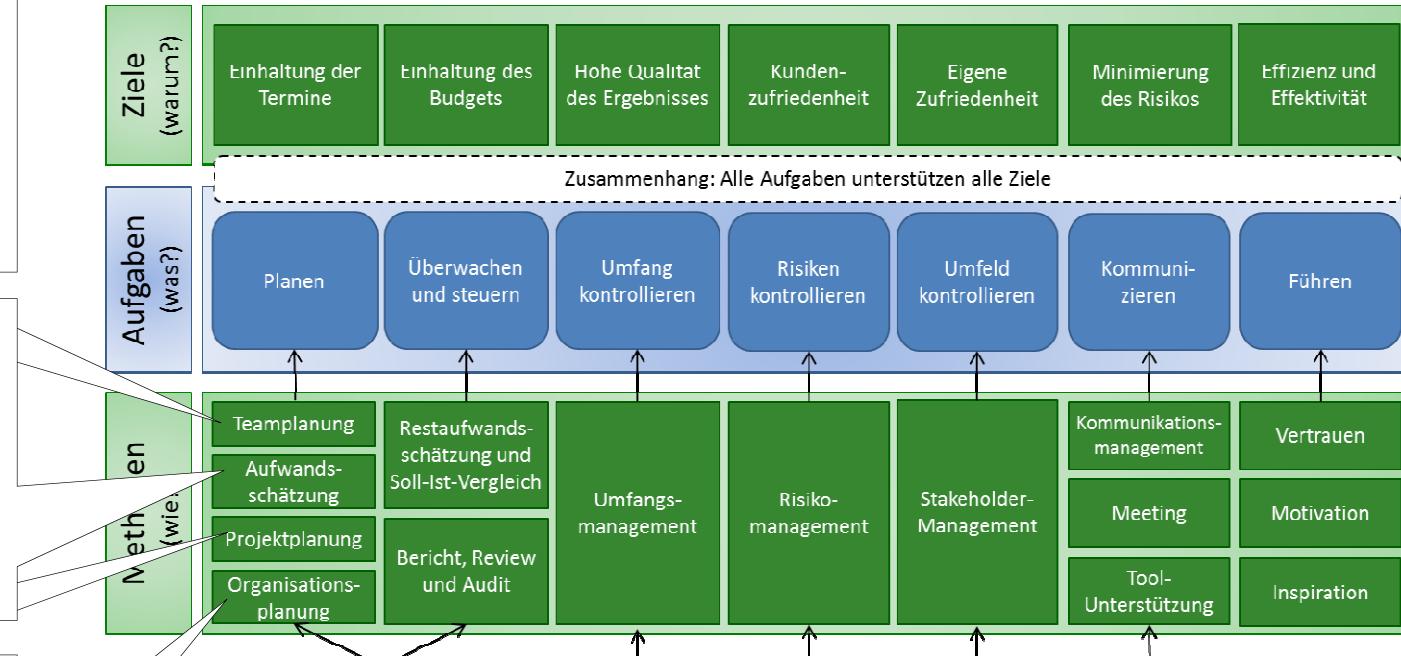
- Echte Experten schätzen lassen
- Aufwände nicht kleinreden
- Preisfestlegung und Schätzung trennen
- Kontext und Team berücksichtigen
- Aufwände kategorisieren
- Design to Cost evaluieren

Projektplanung:

- Blick aufs Ganze
- Ablaufplan grob, Strukturplan fein
- Wichtiges früh klären
- Von Anfang an Gas geben
- Spielräume lassen
- Einfache Werkzeuge

Organisationsplanung:

- Rollen thematisieren
- Nur so viele Rollen wie nötig
- Rollenmodell wiederverwenden
- Vertreterregelung besprechen





BeST-Practices sorgen für ein effektives und effizientes Projektmanagement (2/2)

Restaufwandsschätzung und Soll-Ist Vergleich:

- Restaufwände regelmäßig bestimmen
- „Fast fertig“ hinterfragen
- Veränderungspotenziale ableiten
- Adding manpower to a late project ...

Bericht, Review, Audit:

- Etabliertes hinterfragen
- „Smells“ nachgehen
- Keine toten Pferde reiten

Umfangsmanagement:

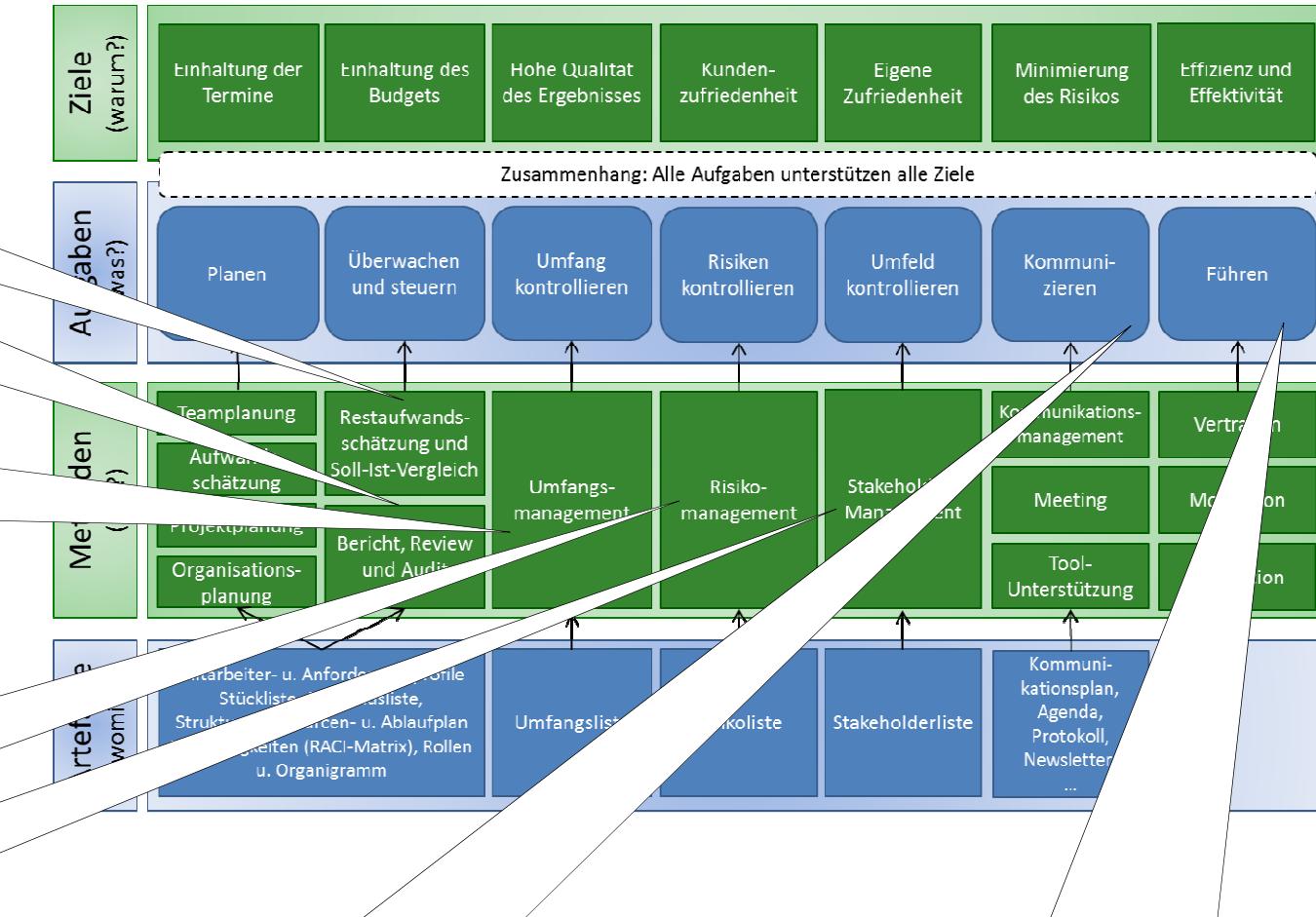
- Erster Ansatz: Projekt klein halten
- Nicht alles reinlassen
- Bei Anforderungen beginnen
- Umfang regelmäßig hinterfragen

Risikomanagement:

- Risikomanagement leben
- „Smarte“ Maßnahmen
- Alte Probleme sammeln
- Risikoentwicklung visualisieren

Stakeholdermanagement:

- Beziehung herstellen
- Vertrauenspersonen nicht austauschen
- Transparent informieren
- Flexibel sein, nicht formal
- Bei Wichtigem hart bleiben
- Kompromisse aktiv anbieten



Kommunizieren:

- Kommunikationsmanagement
- Meetings ja,
- ... aber sparsam
- Projekt und Stakeholder
- Angemessene Mittel
- Keine SPOFs in der Kommunikation

Führen:

- Vertrauen geben
- Verantwortung geben
- Makromanagement
- Feedback geben
- Motivation
- Inspiration und Pragmatismus

Die Soft Skills des Projektmanagers sind und bleiben dabei die wichtigsten Erfolgsfaktoren für effizientes Projektmanagement

Die wichtigsten Körperteile des Projektmanagers

- Das Herz um zu führen,
- Der Bauch (oder das Gefühl darin) um zu vertrauen,
- Die Seele um die Organisation zu beseelen und,
- Die Nase um zu riechen, wenn etwas stinkt.

Grundsätze des Managements („Alles andere sind Administrivialitäten“)

- Wählen Sie die richtigen Leute aus.
- Betrauen Sie die richtigen Mitarbeiter mit den richtigen Aufgaben.
- Motivieren Sie die Mitarbeiter.
- Helfen Sie Teams durchzustarten und abzuheben.



Tom DeMarco

Sicherheit und Veränderung

- Menschen können Veränderungen nur in Angriff nehmen, wenn sie sich sicher fühlen.
- Veränderung ist eine entscheidende Voraussetzung für den Erfolg jeder Projektarbeit (und wahrscheinlich auch der meisten anderen lohnenden Unternehmungen).
- Fehlende Sicherheit bewirkt fehlende Risikobereitschaft.
- Risikovermeidung ist fatal. Sie führt dazu, dass die mit einem Risiko verbundenen Chancen ungenutzt bleiben.

Negative Verstärkung

- Drohungen motivieren nur bedingt zu höheren Leistungen.
- Eine Drohung kann so ernst sein, wie sie will: Eine Aufgabe wird nicht termingerecht erledigt werden, wenn die veranschlagte Zeit zu knapp bemessen ist.
- Schlimmer noch: Wenn das Ziel nicht erreicht wird, muss man seine Drohungen womöglich wahr machen.

Der Mensch im Projekt

- Menschen unter Druck denken nicht schneller.
- Man kann Menschen ohne Fürsorge und Zuneigung nicht dazu bringen anders als bisher zu handeln. Um sie zu Veränderungen zu bewegen, muss man verstehen, woher sie kommen und warum sie sind, wie sie sind.

Quelle: Tom DeMarco, *Der Termin (The Deadline: A Novel About Project Management)*. Dorset House, 1997)

Agenda

- Accso
- Beschleunigte Softwaretechnik (BeST)
- Ausgewählte Aspekte
 - BeST-Practices im Projektmanagement
 - **BeST-Practices der Analyse**
 - BeST-Practices der Architektur
- Abschluss

Die Analyse beinhaltet drei Aufgabengebiete: Den Kunden, dessen Anforderungen und die benötigte fachliche Anwendung.

Mit dem Kunden arbeiten



Anforderungen analysieren



Fachlichkeit der Anwendung analysieren



- Wie denken und handeln die fachlichen Interessensvertreter?
- Welche Interessen haben die verschiedenen Nutzer der Anwendung?
- Welche Aufgaben lösen die verschiedenen Nutzer in ihrer täglichen Arbeit und wie tun sie das? Was behindert sie?
- Welche Begriffe verwenden die Interessensvertreter und was bedeuten die Begriffe?
- Was möchte der Nutzer umgesetzt bekommen? Warum?
- Bei welchen Aufgaben soll die Anwendung unterstützen?
- Wie soll die Anwendung bei einer Aufgabe unterstützen?
- Wie wichtig ist die Anforderung? Wie teuer ist sie?
- Was braucht der Kunde wirklich?
- Was ist zusammen umzusetzen? Was sind die Abhängigkeiten?
- Welche fachlichen Funktionen besitzt die Anwendung?
- Wie interagieren die Nutzer mit der Anwendung?
- Was sind die fachlichen Abnahmekriterien?

BeST-Domäne Analyse: Ziele, Aufgaben, Methoden, Artefakte

Ziele

Beschleunigung
(Effizienz,
Effektivität)

Minimierung des
fachlichen Risikos

Hohe Qualität der
Analyse-
ergebnisse

Hohe
Zufriedenheit des
Kunden

Gemeinsames
Verständnis der
Fachlichkeit

Hohe Zufrieden-
heit der Accso-
Mitarbeiter

Gegenseitiges
Vertrauen

Aufgaben

Fachliche Risiken identifizieren und behandeln

Fachliche Begriffe
definieren

Fachliche Abgrenzungen
vornehmen

Fachliches Umfeld managen

Anforderungen
verwalten

Fachlichkeit einer
Anwendung analysieren

Mit fachlichen
Interessensvertretern und
Projektteam kommunizieren

Fachliche QS-
Maßnahmen
durchführen

Funktionalen
Testumfang
ermitteln

Methoden

Glossar erstellen

Anforderungen
analysieren

Ein-/Ausgabeverhalten
beschreiben

Anforderungen
priorisieren und
bewerten

Dialoge visualisieren

Ist-Analyse durchführen

Workshops durchführen

Interviews durchführen

Meetings halten

Szenarien durchspielen

Arbeitsablauf begleiten

Kreativitätstechniken

Rückmeldungen
einholen

Ergebnisse
präsentieren

Akzeptanzkriterien
beschreiben

Testszenarien
beschreiben

Testfälle
spezifizieren

Ergebnisse dokumentieren

Artefakte

Glossar

Anforderungs-
spezifikation

Fachkonzept

Leistungsbe-
schreibung

Workshop-Artefakte

Checklisten

Testartefakte

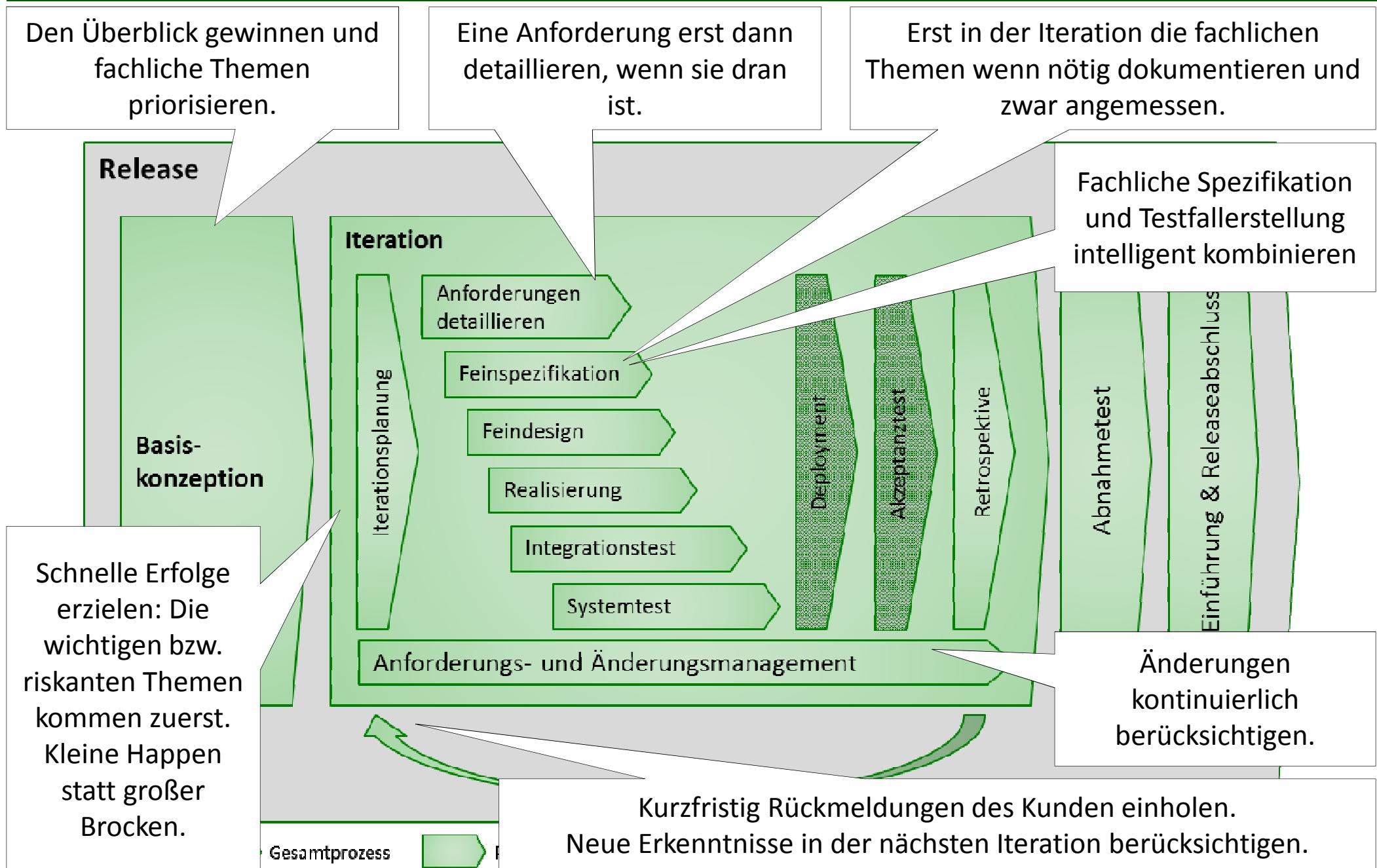
Geschäftsprozess-
modell

Oberflächen-
konzept

Anwender-
handbuch

Review-
anmerkungen

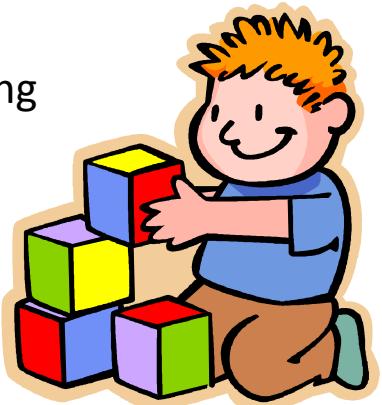
Eine effektive und effiziente Analyse verteilt sich charakteristisch über die verschiedenen Phasen und die Laufzeit des Projekts





Auszuarbeitende „Bausteine“ der Spezifikation explizit angemessen auswählen und zeitlich einplanen

Baustein	üblicherweise wann/wie
▪ Einleitung/Ziel	Pflicht Basiskonzeption
▪ Fachliche Grundlagen	Pflicht Basiskonzeption
▪ Statische Systembeschreibung (z.B. Systemkontext/Fachliche Komponenten)	Pflicht Basiskonzeption
▪ Dynamische Systembeschreibung (z.B. Geschäftsprozesse, Anwendungsfälle, User Stories, Testfälle)	Pflicht grob in der BK ¹⁾ , fein in der Iteration Details tendenziell als Testfälle
▪ Dialogbeschreibungen	optional zumeist besser implizit direkt über Code
▪ Druckausgaben	optional wenn nötig
▪ Schnittstellenbeschreibung	Pflicht grob in der BK ¹⁾ , fein in der Iteration
▪ Fachliches Datenmodell	Pflicht grob in der BK ¹⁾ , fein in der Iteration
▪ Batches	optional wenn nötig
▪ Migrationskonzept	optional wenn nötig
▪ Glossar	Pflicht Basiskonzeption



1) Basiskonzeption

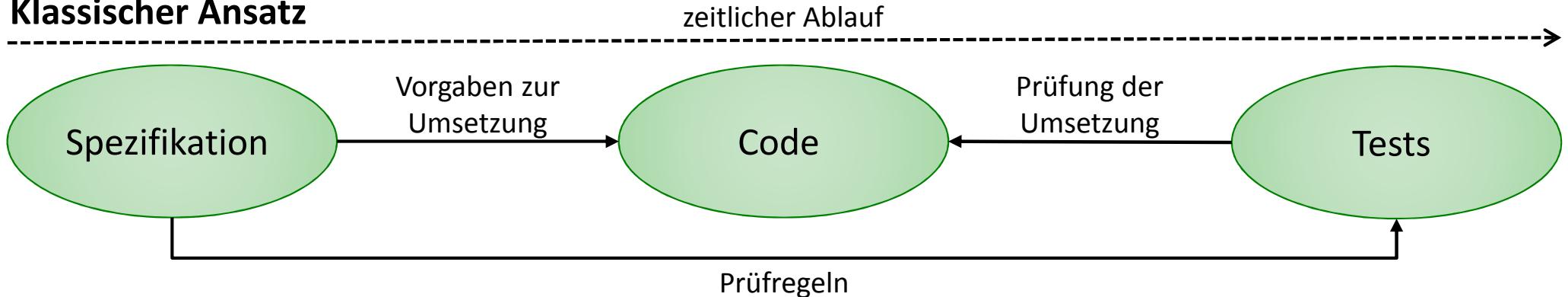


Vorgegebene Spezifikationsartefakte und –methoden können dazu verleiten, im Rahmen der Analyse unangemessen zu spezifizieren und zu dokumentieren. Effizient arbeitet, wer das dokumentiert, was an der entsprechenden Stelle im Projekt auch wirklich benötigt wird und nicht mehr.

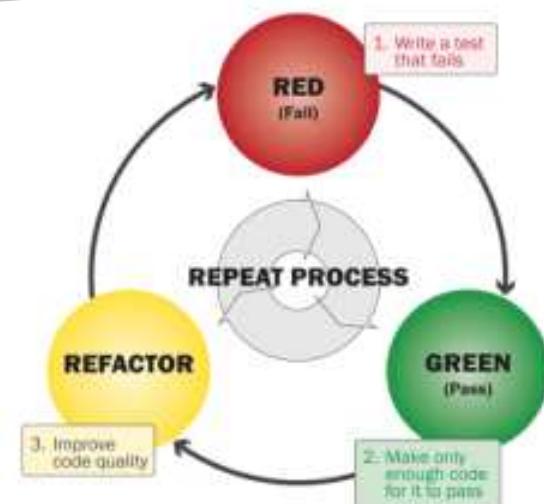
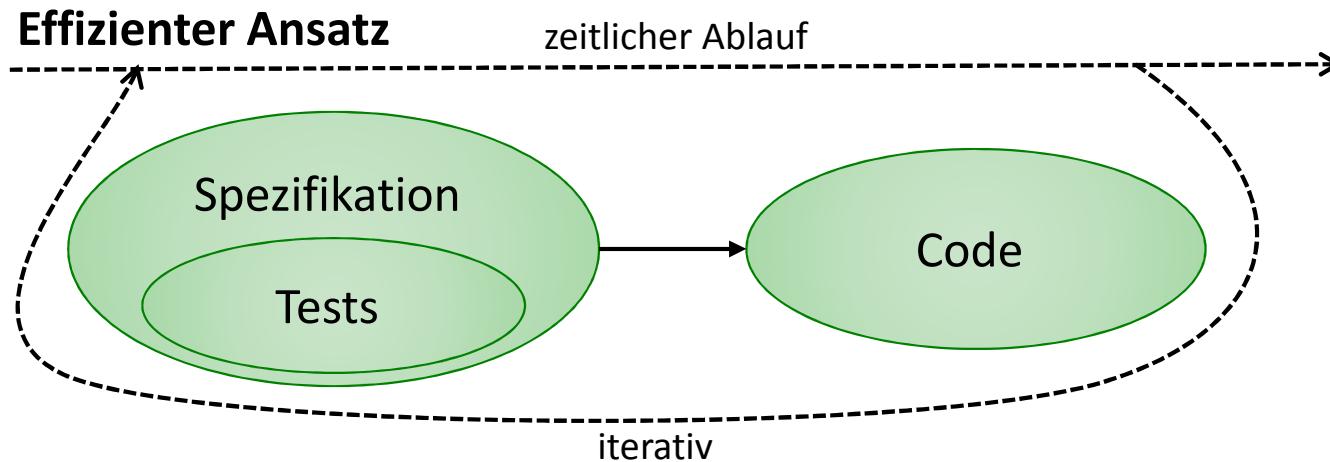


Verhältnis von Analyse und Test projektspezifisch optimieren und fachliche Details in Testszenarien spezifizieren

Klassischer Ansatz



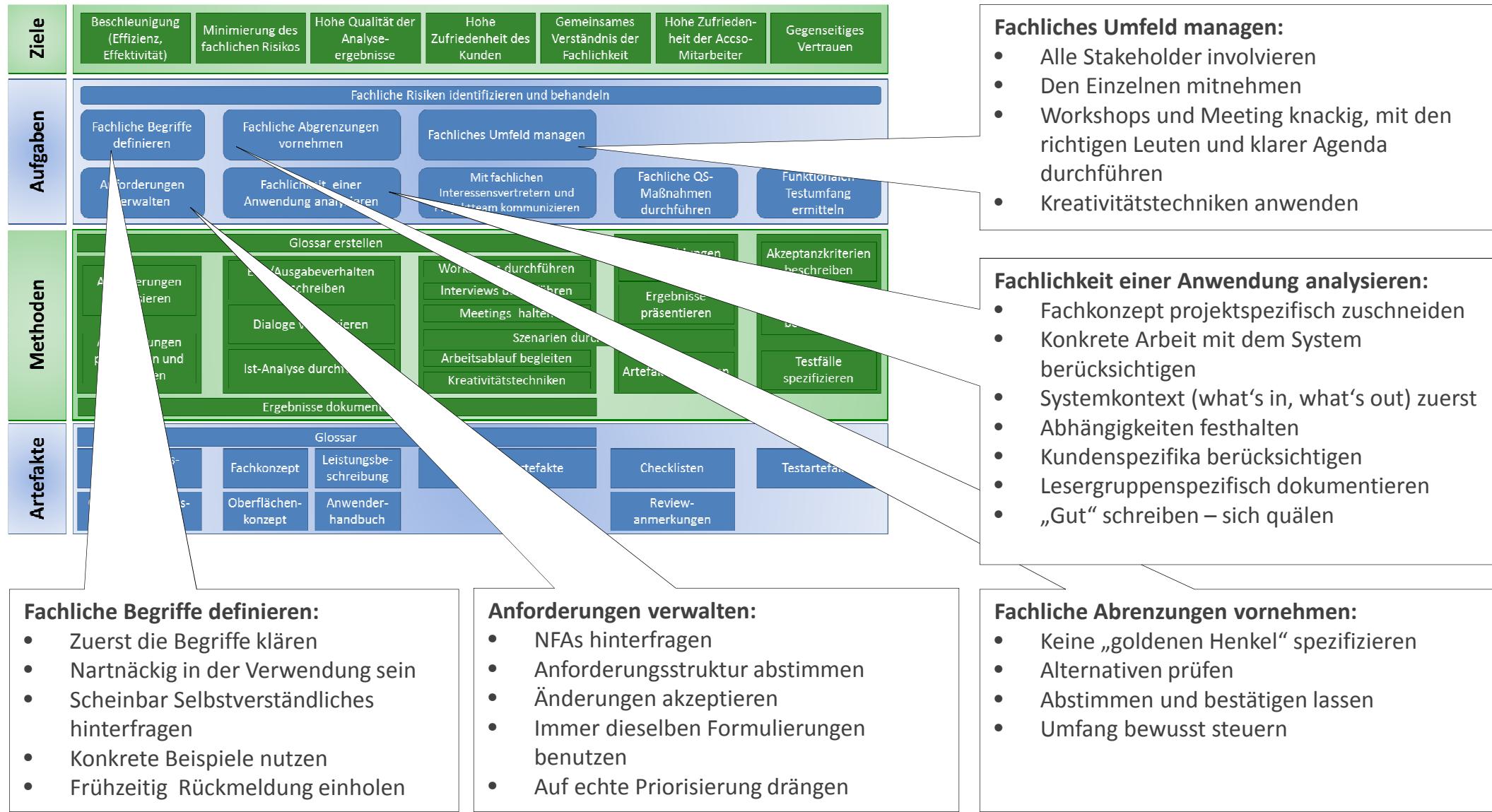
Effizienter Ansatz



Ein Grundprinzip effizienter Projektvorgehensweise ist, die Aspekte Spezifikation, Umsetzung der Spezifikation und Prüfung der Umsetzung der Spezifikation möglichst eng zu verzähnen. In diesem Sinn sind die Spezifikation und die Prüfregeln weitgehend identisch und es kann nicht effizient sein, diese redundant und zeitlich stark versetzt zu erfassen.



An die alten Weisheiten denken

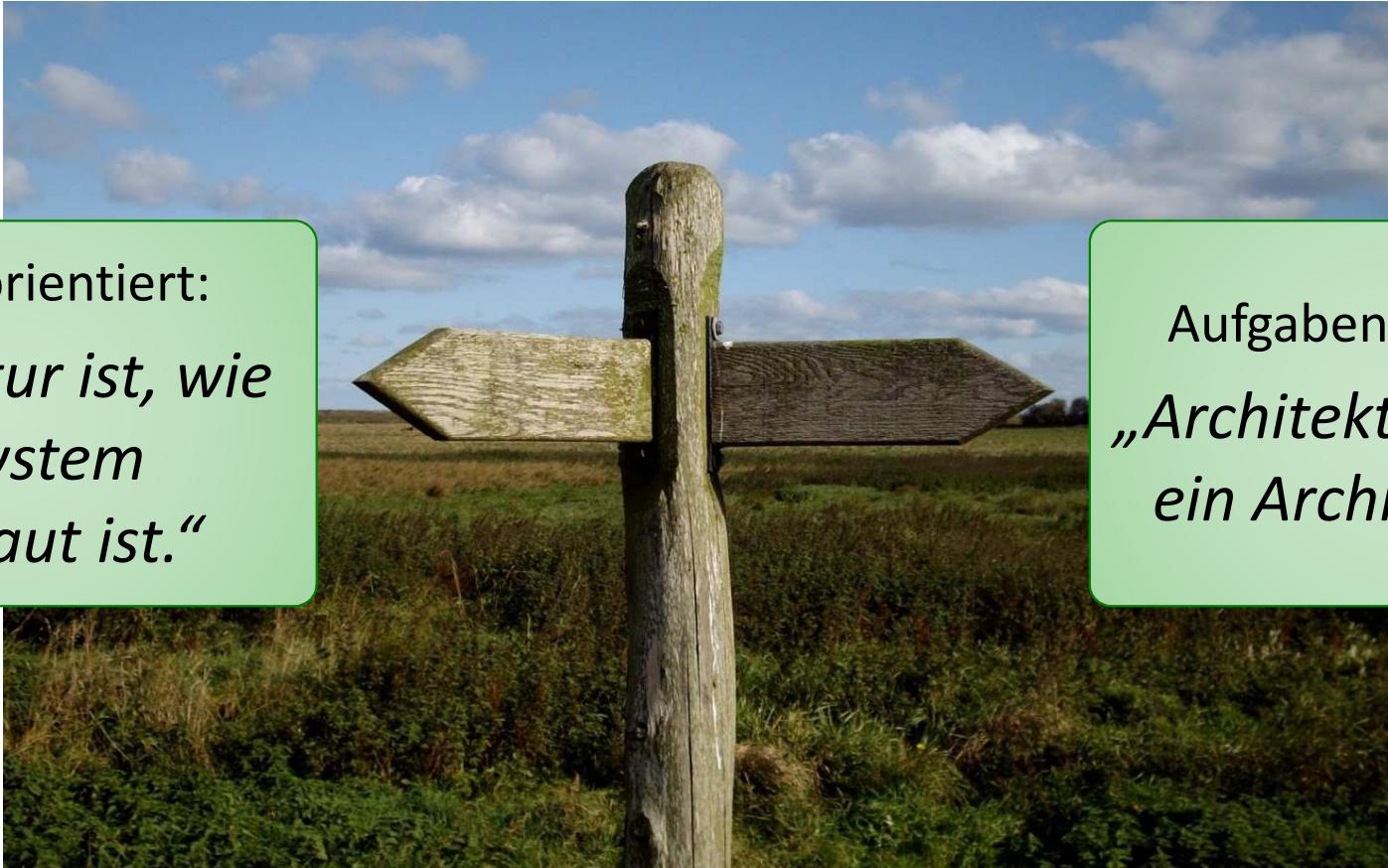


Die Grundvoraussetzungen effizienter Arbeit sind lange bekannt.
Man muss sich „nur“ diszipliniert daran halten.

Agenda

- Accso
- Beschleunigte Softwaretechnik (BeST)
- Ausgewählte Aspekte
 - BeST-Practices im Projektmanagement
 - BeST-Practices der Analyse
 - **BeST-Practices der Architektur**
- Abschluss

Es gibt zwei Wege, sich dem Begriff Architektur zu nähern



Strukturorientiert:
*„Architektur ist, wie
ein System
aufgebaut ist.“*

Aufgabenorientiert:
*„Architektur ist, was
ein Architekt tut.“*

Strukturorientierte Herangehensweise

Strukturorientiert:
„Architektur ist, wie
ein System
aufgebaut ist.“



Software-Architektur nach IEEE 1471



*„Architecture is defined as the **fundamental** organization of a system, embodied in its components, their relationships to each other and to the environment, and the principles governing its **design** and evolution.“*

Eine kleine Verschärfung, die hilft, sich auf das **Grundsätzliche** zu konzentrieren



Strukturorientiert:
„Architektur ist, wie
ein System
aufgebaut ist.“



Aufgabenorientiert:
„Architektur ist, was
ein Architekt tut.“

*„Architecture is defined as the **fundamental** organization of a system, embodied in its **types of** components, their **types of** relationships to each other and to the environment, and the principles governing its **design** and evolution.“*

In der Unterscheidung zwischen dem Grundsätzlichen und den Details liegt der Hebel für die **Effizienz**



*“All architecture is design but not all design is architecture.
Architecture represents the **significant** design decisions
that shape a system, where significant is measured by cost
of change.”*

Grady Booch*

* In <https://www.ibm.com/developerworks/mydeveloperworks/blogs/gradybooch>

Vorschlag zur Benennung:

„Architektur“

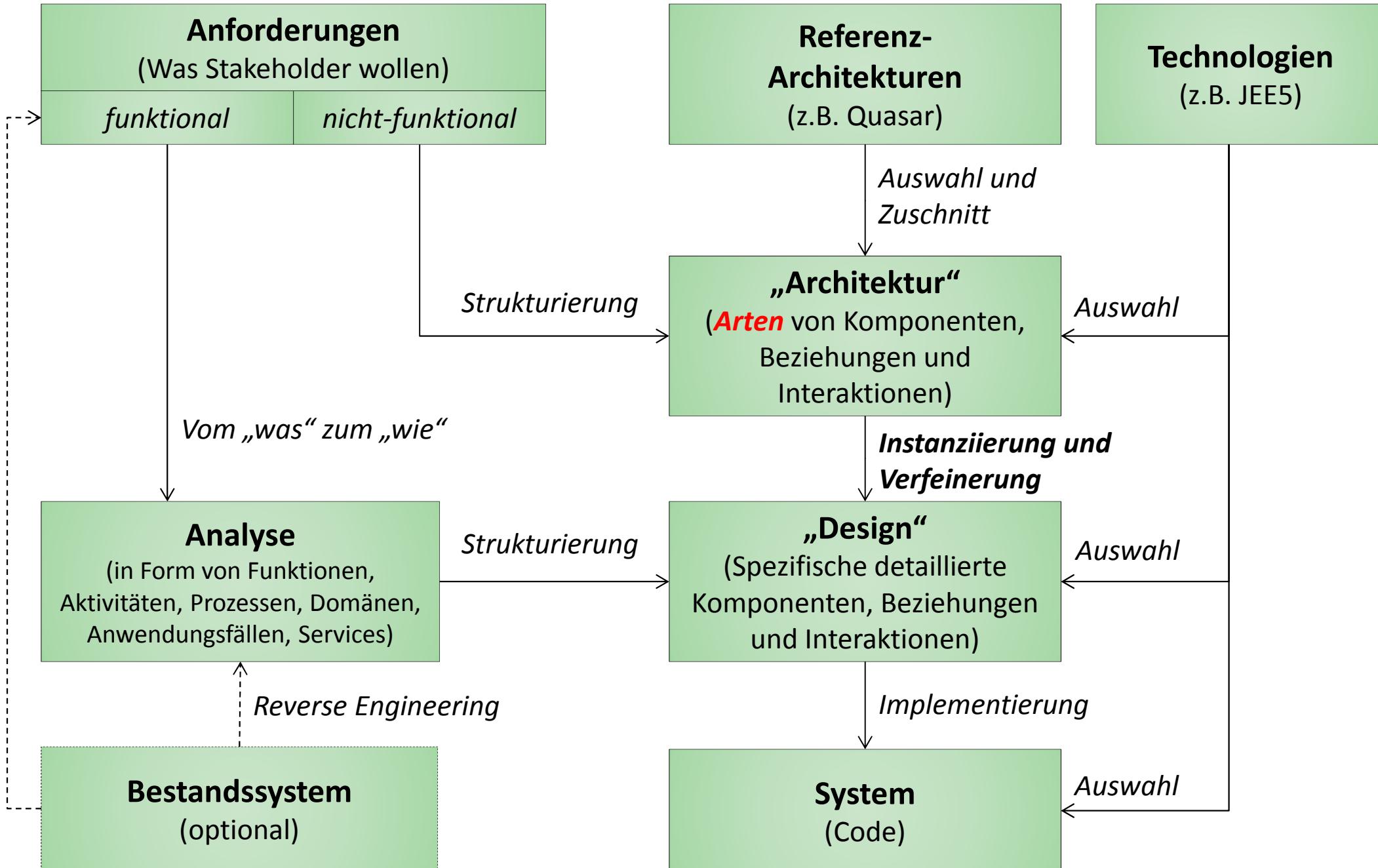


„Design“

grundsätzlich
wesentlich
signifikant
teuer zu ändern

die Details

Mit dieser Semantik folgt die „Architektur“ insbesondere den nicht-funktionalen Anforderungen und das „Design“ entsteht als deren konkrete Instanziierung und Verfeinerung



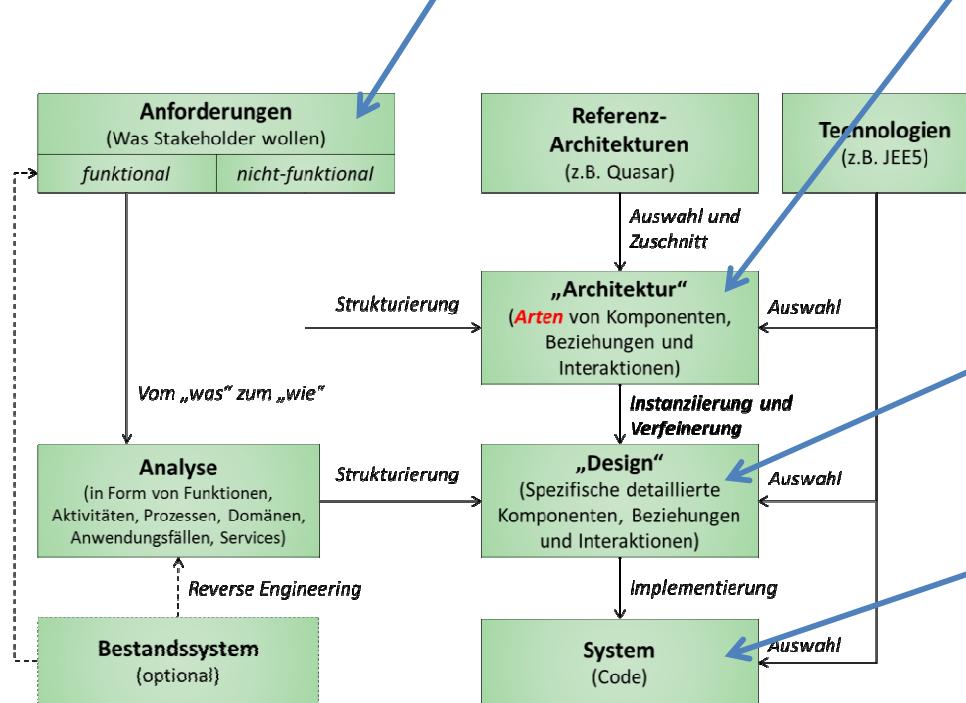
Analogie zum Bauwesen

(Ursprung des Begriffs Architektur)

Ich möchte ein neues Gotteshaus bauen.

Die Gemeinde von Köln soll dort den Gottesdienst feiern können (funktionale Anforderung).

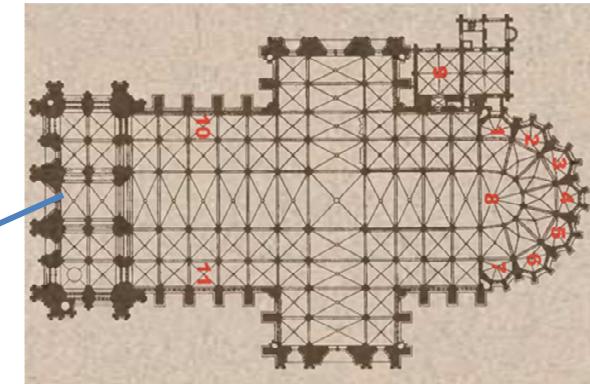
Im Inneren soll es möglichst hell sein (nicht-funktionale Anforderung).



Gotische Architektur (rote Elemente in der Abbildung):

- Wir verwenden Spitzbögen, Kreuzrippengewölbe, Strebewerke und Strebepfeiler
- Kreuzrippen und Strebewerk verwenden wir als tragende Elemente. Das erlaubt eine Minimierung der Wandflächen und eine Maximierung der Fensterflächen. Dadurch wird es hell.

Design des Kölner Doms



Der Kölner Dom



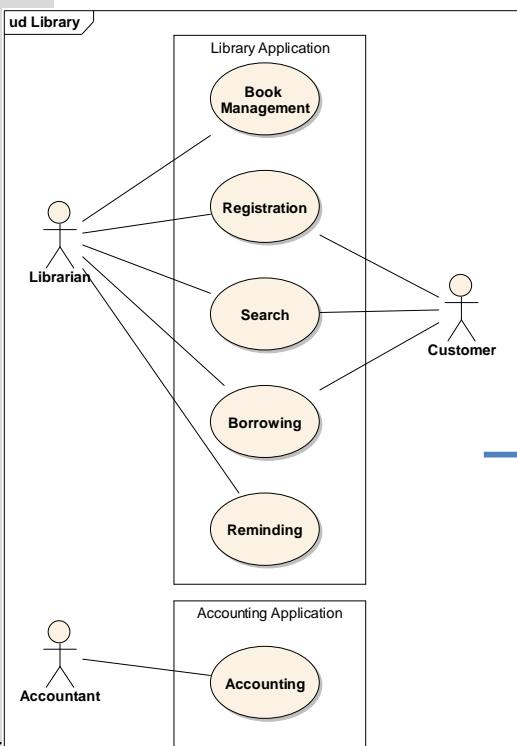
Am Beispiel der Software-Entwicklung



I want a library management system to be built (FA):

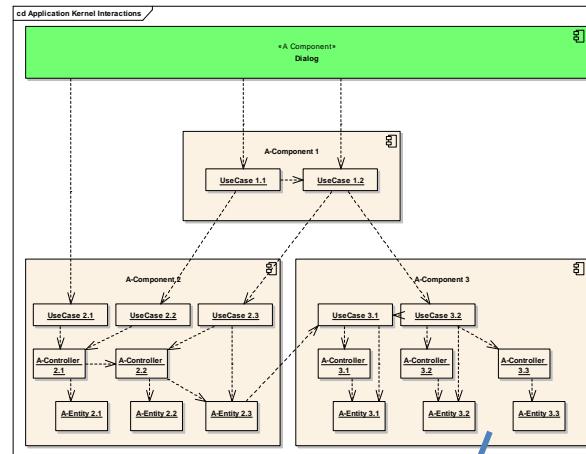
- Customers of a library can apply for a library membership. Then, they can search for books and borrow them. They have to return the borrowed books according to specific loan periods.
- Librarians manage the book stock in the library. They register new customers in the library application. They enter the loan of a book by a customer in the system. Books that are overdue result in a reminding /dunning process. The librarian manages this process.
- Registration and reminding fees are to be paid by customers. Accountants deal with those payments with a separate accounting application.

And I want sustainability / maintainability (NFA)



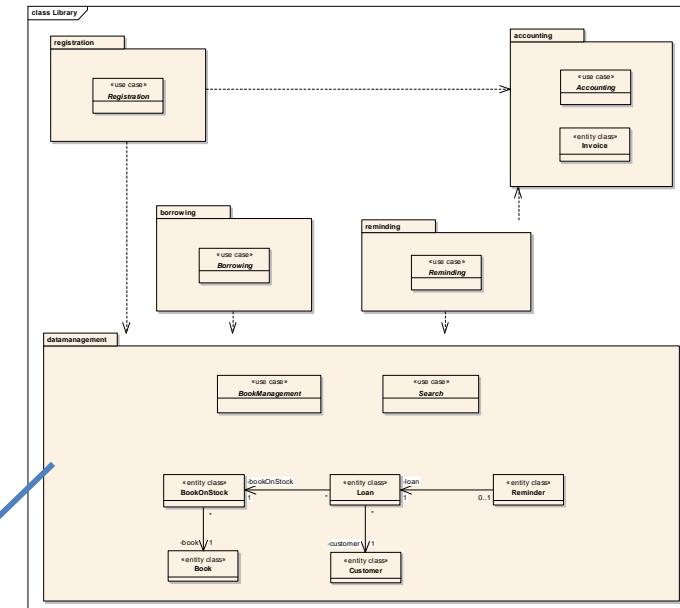
Software-Architektur (nach Quasar)

- Wir verwenden A-Komponenten bestehend aus
- A-Fällen (Use Cases)
- A-Verwaltern (Entity Controller)
- A-Entitäten (Entities)



Software-Design

- Konkret gibt es die A-Komponenten Accounting, BookManagement, Registration, ...
- mit den A-Fällen Registration, Search, ..., den A-Verwaltern BookManager, LoanManager, ... und den A-Entitäten Book, Loan, Reminder, Person, ...



Software-System *

```

@Stateless
@Transactional(TransactionAttributeType.REQUIRED)
public class BookManagerImpl implements BookManager {

    @PersistenceContext
    private EntityManager em; void edit(Book book);

    void destroy(Book book);
    Book findById(Long id);
    List findAll();
    Book create(Book book);
    List findByTemplate(Book book);
}
  
```

*) Implementierung des A-Verwalters BookManager in JEE5

Strukturorientiert:
„Architektur ist, wie ein System aufgebaut ist.“

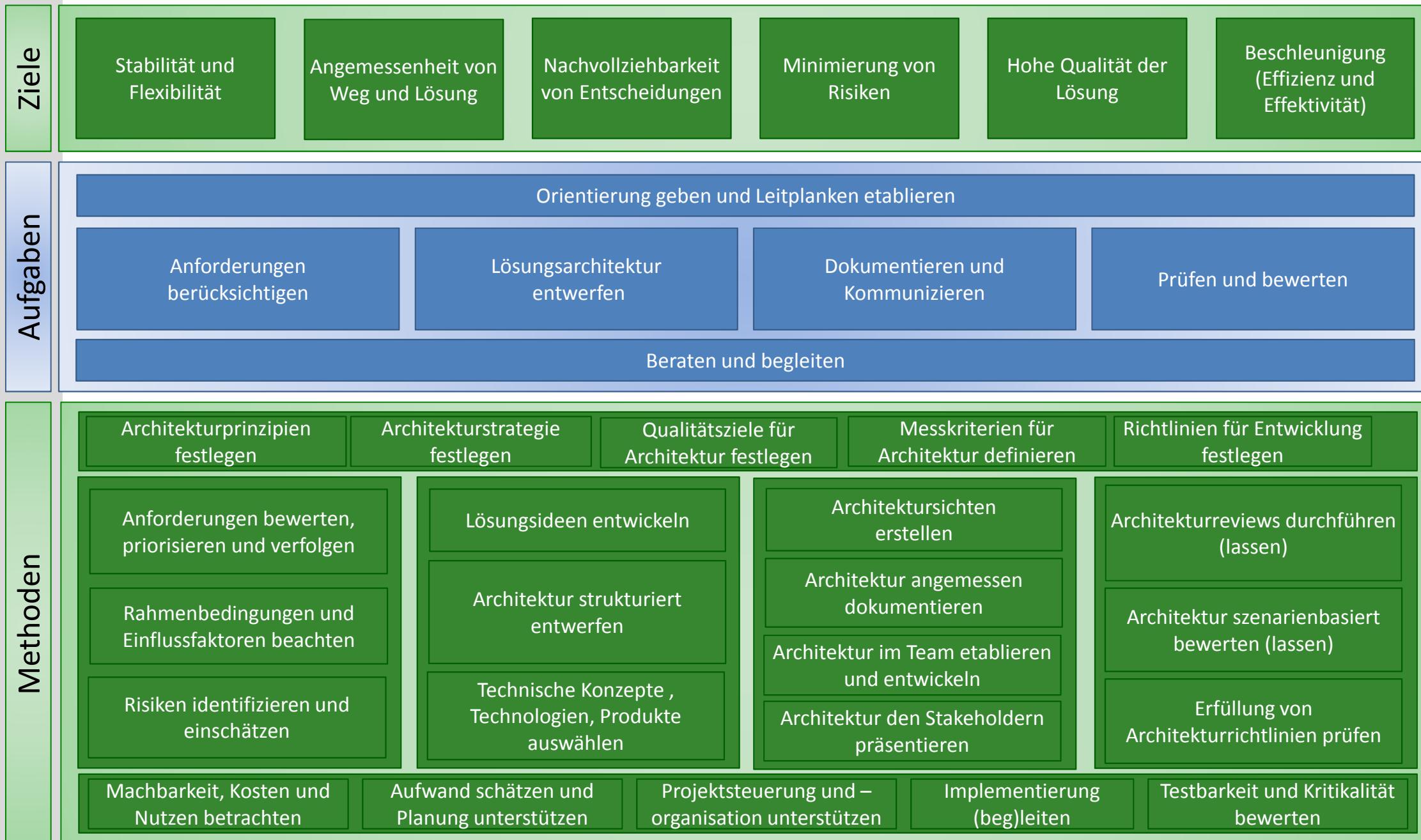
Aufgabenorientiert:
„Architektur ist, was ein Architekt tut.“

Aufgabenorientierte Herangehensweise

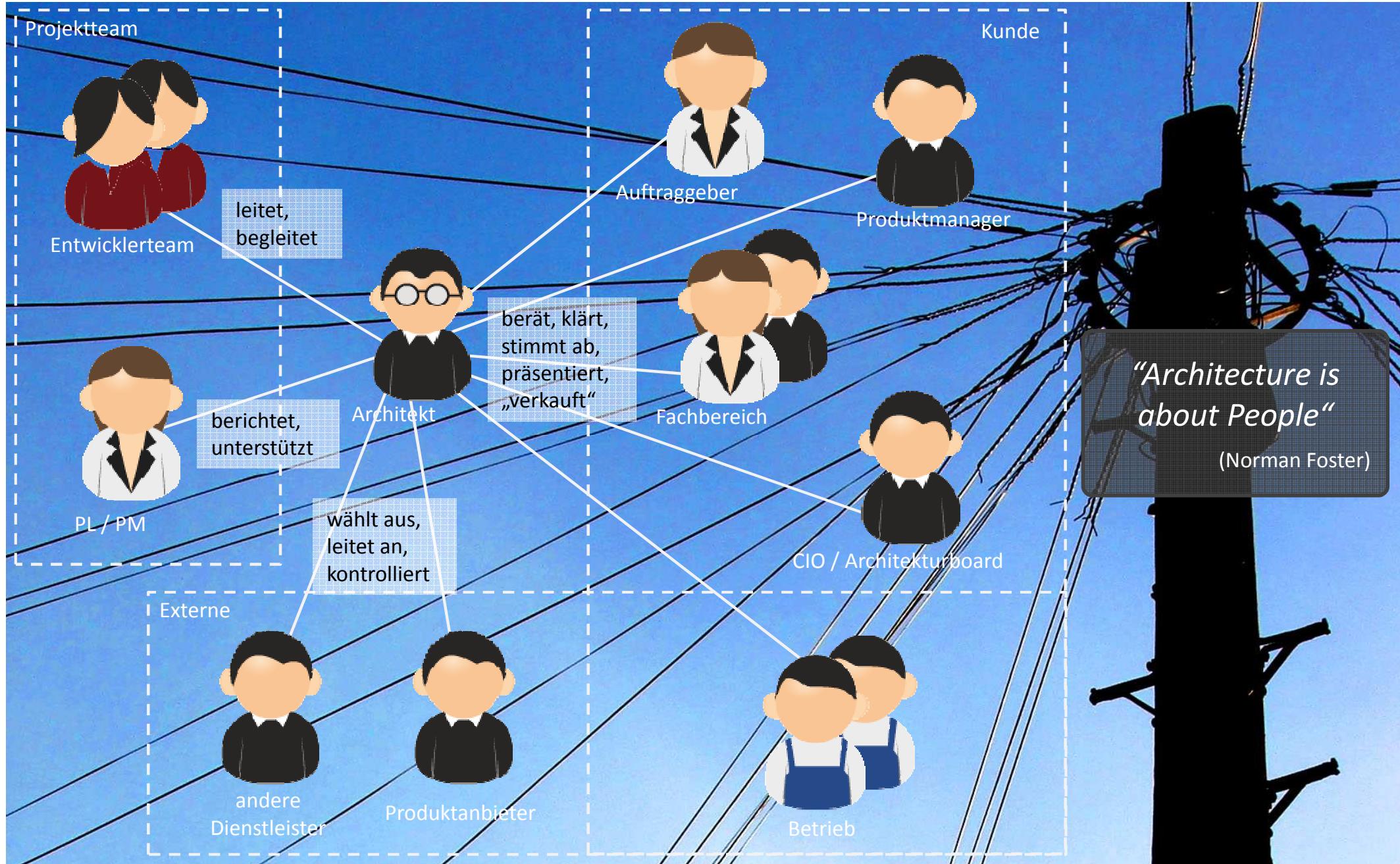


Aufgabenorientiert:
„Architektur ist, was
ein Architekt tut.“

BeST beschäftigt sich innerhalb der Domäne Architektur ganzheitlich mit allen Aufgaben des Architekten



Und diese Aufgaben sind äußerst vielfältig.



Für beide Interpretationen von Architektur gibt es wertvolle BeST-Practices



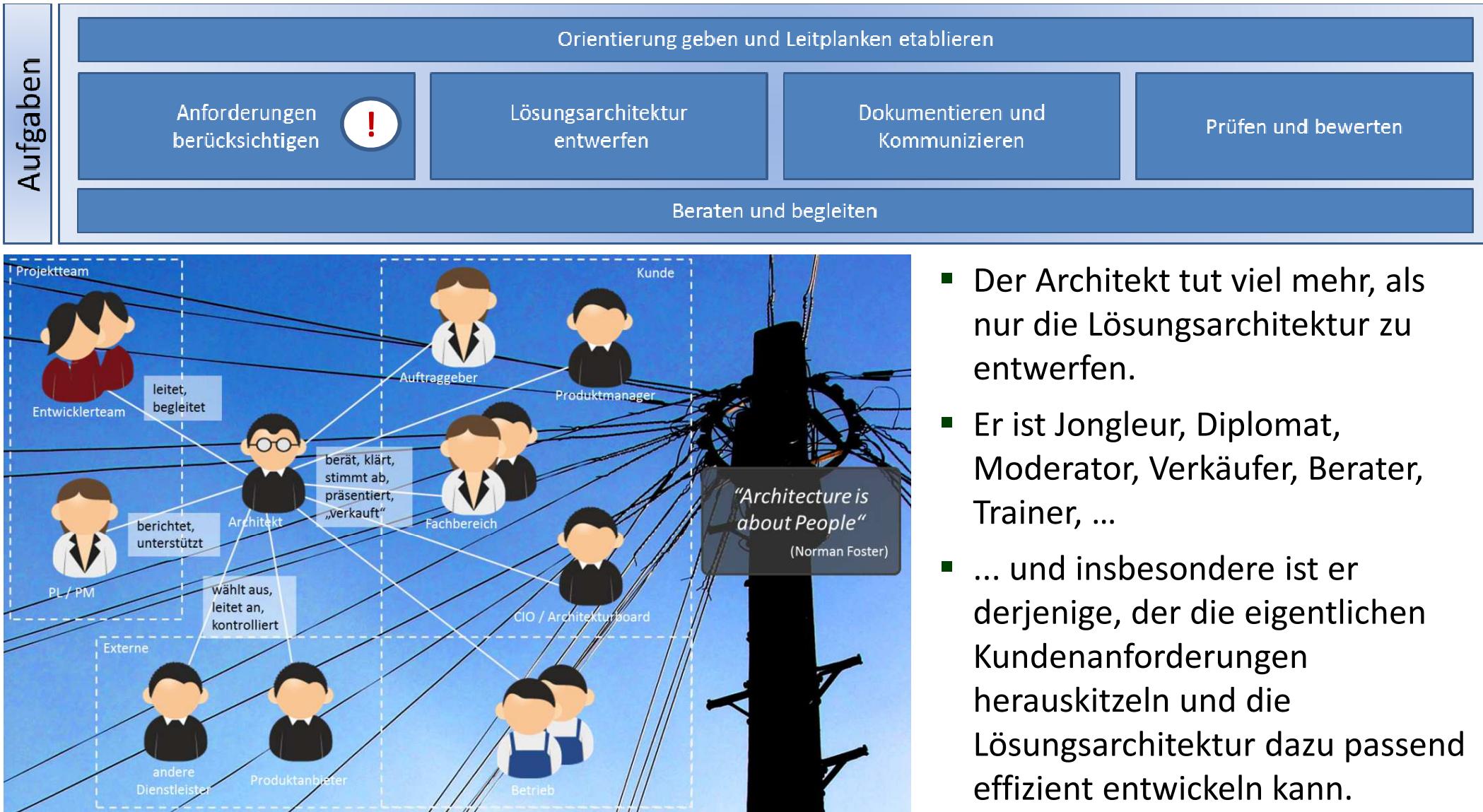
Strukturorientiert:
„Architektur ist, wie
ein System
aufgebaut ist.“

Aufgabenorientiert:
„Architektur ist, was
ein Architekt tut.“





Alle Aufgabenbereiche des Architekten berücksichtigen



- Der Architekt tut viel mehr, als nur die Lösungsarchitektur zu entwerfen.
- Er ist Jongleur, Diplomat, Moderator, Verkäufer, Berater, Trainer, ...
- ... und insbesondere ist er derjenige, der die eigentlichen Kundenanforderungen herauskitzeln und die Lösungsarchitektur dazu passend effizient entwickeln kann.



Ganz viele Hebel der Effizienzsteigerung liegen in den begleitenden Aktivitäten des Architekten neben dem reinen Entwurf der Lösungsarchitektur – insbesondere beim Management der Anforderungen.



Sich das Grundsätzliche explizit anhand der NFAs klar machen



Es soll
hell sein!

! Es ist entscheidend, dass der Architekt Software-Architektur grundsätzlich versteht und über die passenden Grundstrukturen zielgerichtet entlang der konkreten nicht-funktionalen Anforderungen entscheiden kann.

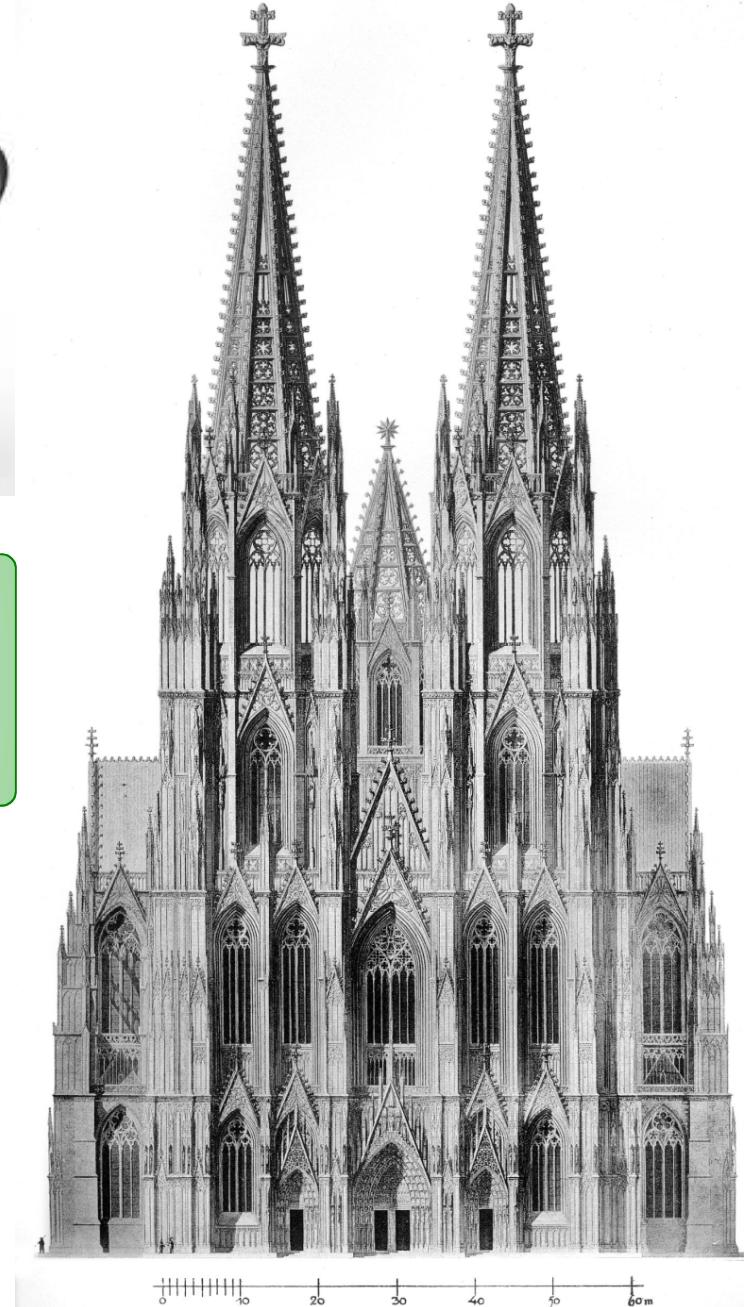
Execution qualities

Wie viel
Performance, Security,
Usability

brauche ich wirklich und mit welchen Architekturmustern erreiche ich das?

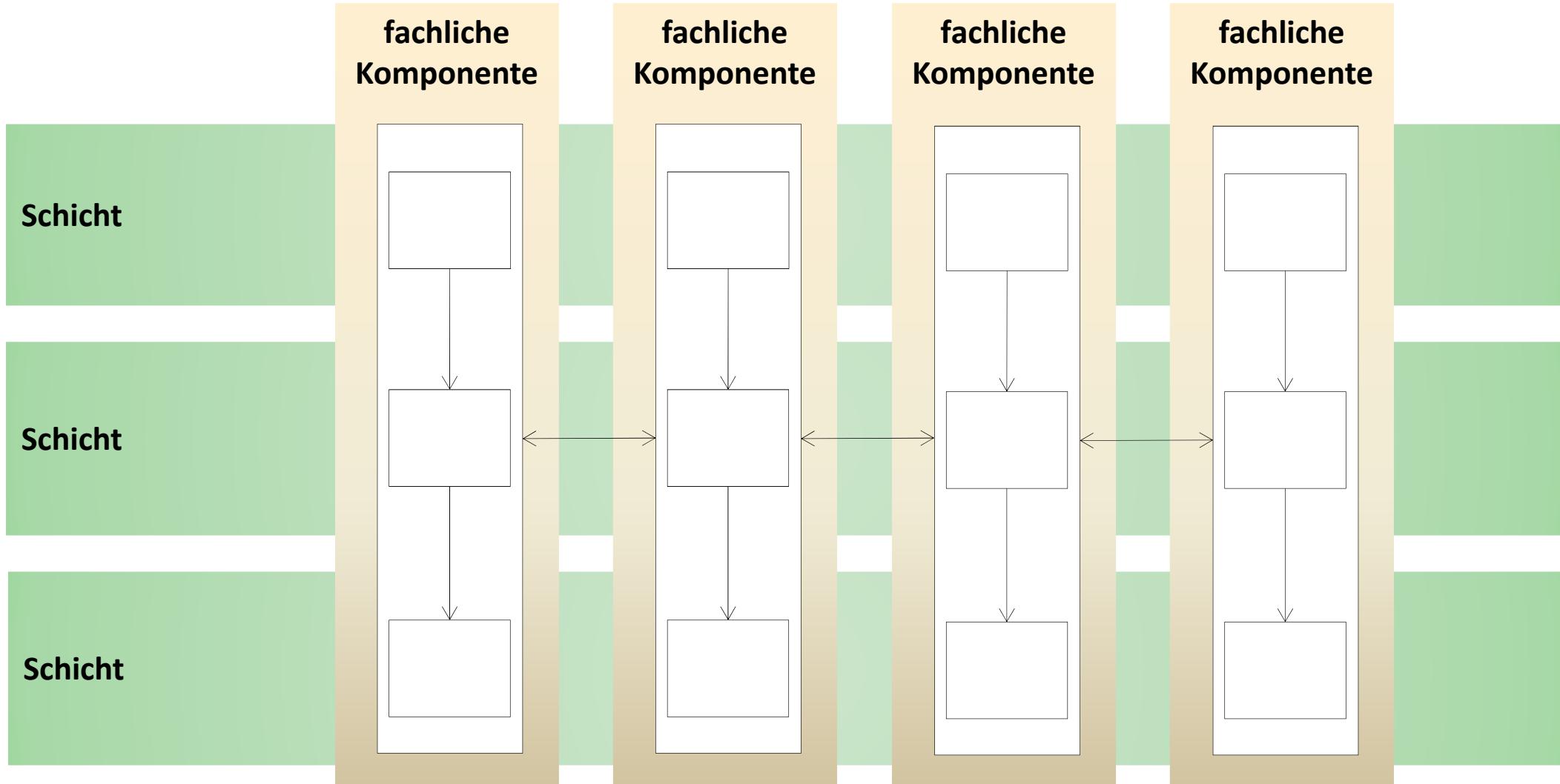
Evolution qualities

Testability, Maintainability,
Extensibility, Scalability





Software-Architekturen in mindestens zwei Dimensionen entwerfen



Für jedes Software-Element (Paket, Klasse, Interface, ...) ist damit klar, wo es hingehört.



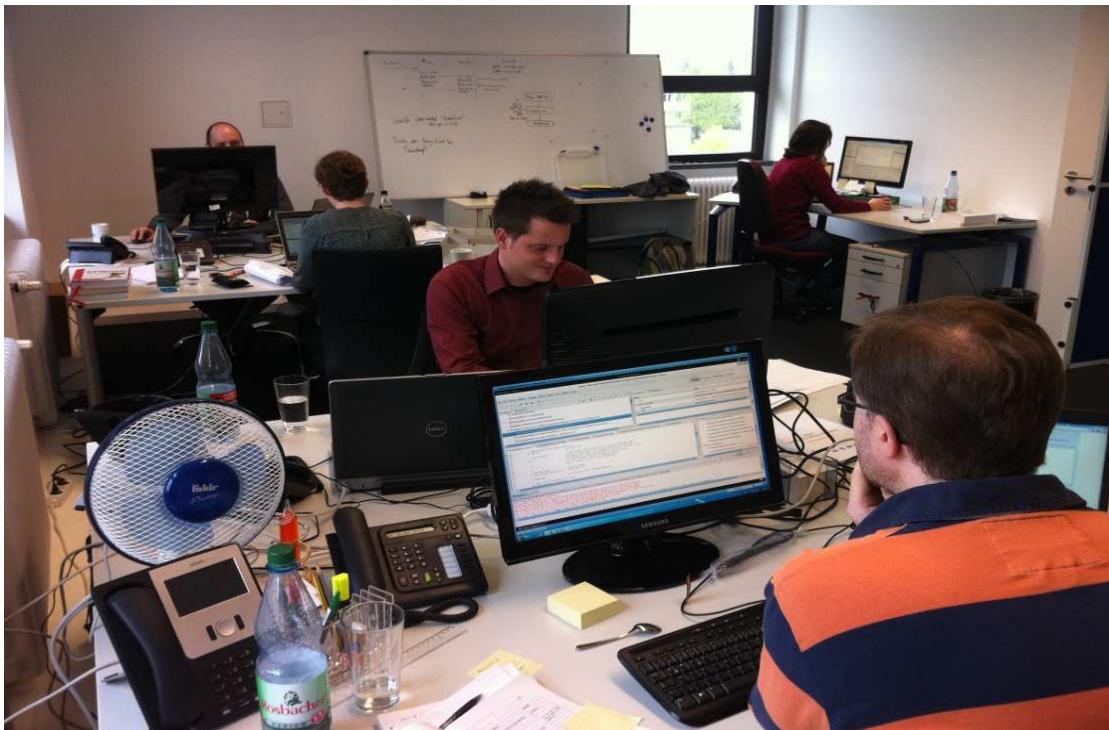
Technische Schichtenarchitektur haben alle – fachliche Komponenten nur ganz wenige.
Dabei ist das mindestens genau so wichtig.



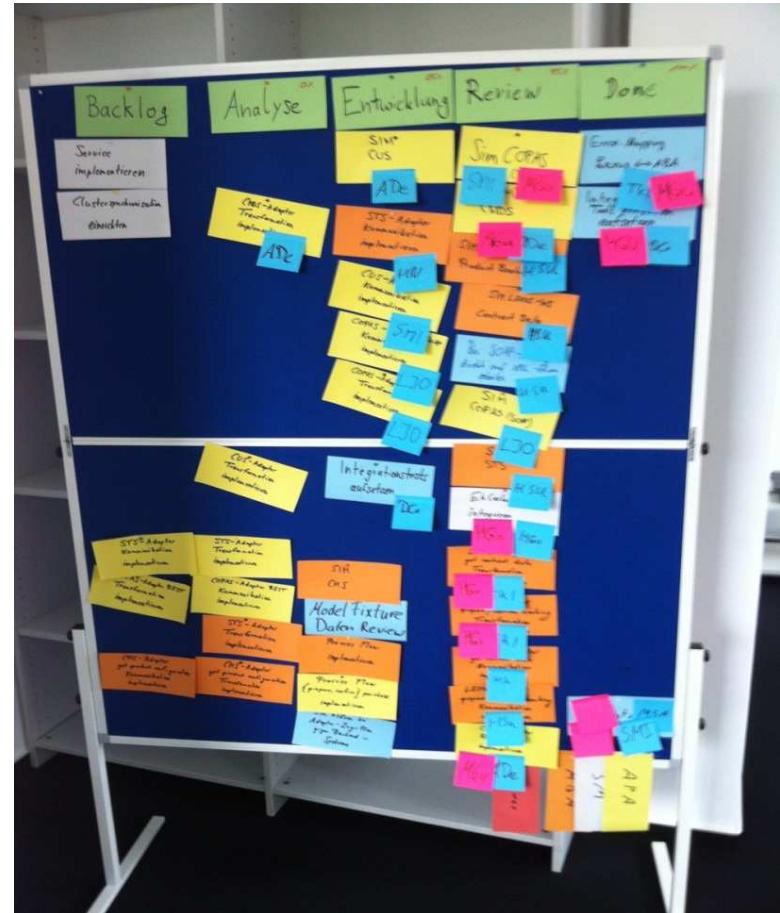
Auch agile Software-Projekte brauchen eine klare Architektur

Beispiel: Aktuelles Entwicklungsprojekt bei Accso

- Entwicklung von drei neuen Backend-Services
- 200 PT Aufwand und nur 7 Wochen Zeit
- 9 Mitarbeiter (6 FTE), von 0 aufzubauen
(3 Erfahrene, 4 Neueinsteiger, 2 Werkstudenten)
- Agiles Vorgehen (Mischung aus Scrum und Kanban)

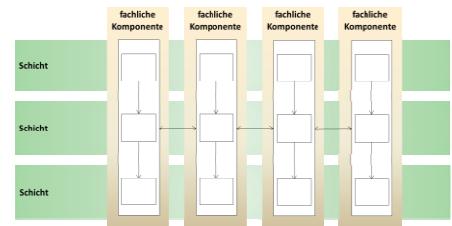


! Nur aufgrund einer klaren Architektur ist das selbstorganisierte Team in der Lage, die Aufgaben auch effizient zu verteilen.



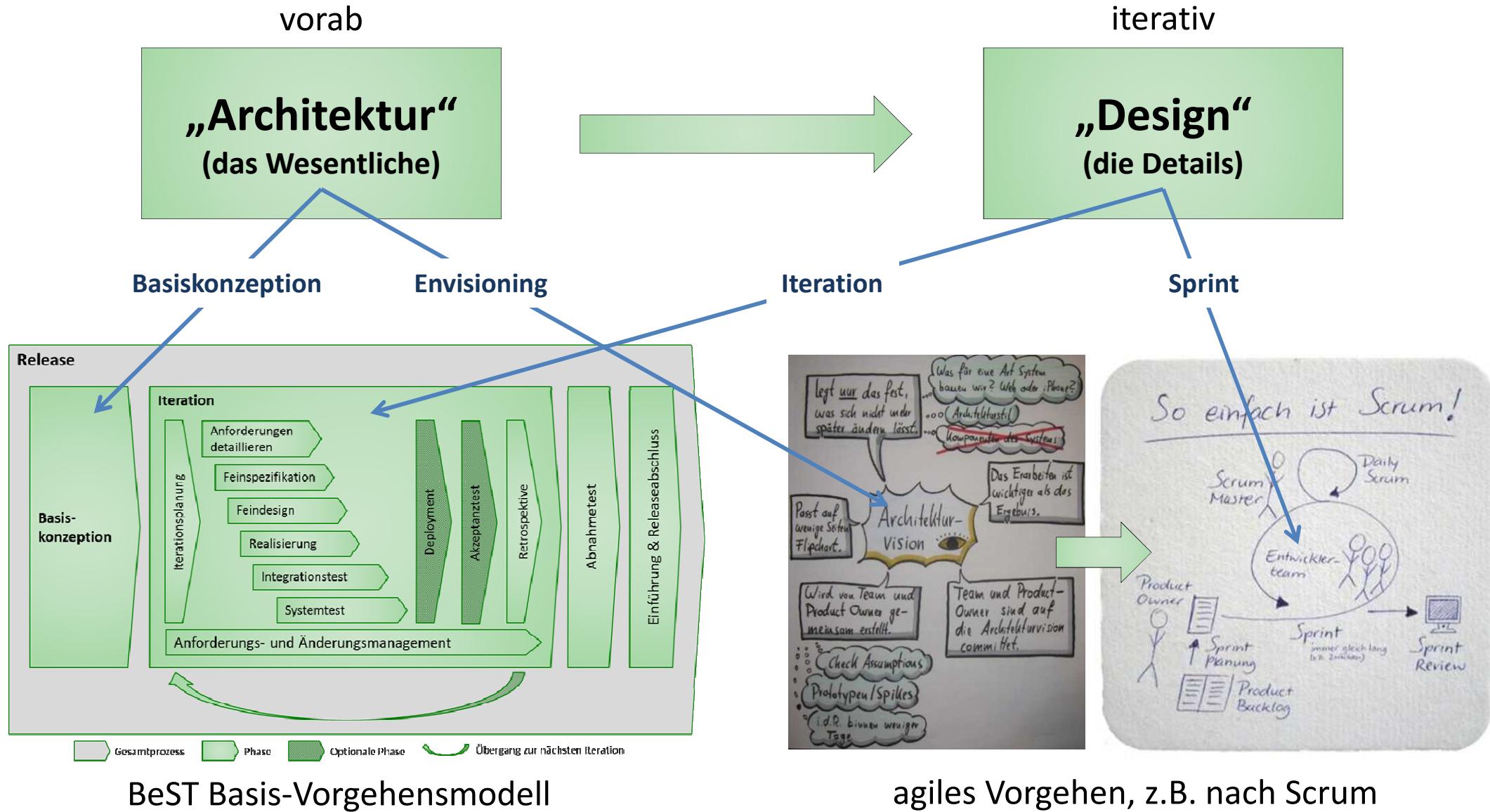
Im Beispiel:

- Kartenfarbe = Fachliche Komponente
- Karten = Tasks bezogen auf Schichten





„Architektur“ (das Wesentliche) vorab und „Design“ (die Details) iterativ – unabhängig vom speziellen Vorgehensmodell



! Unabhängig vom konkreten Vorgehensmodell ist des Wesentliche frühzeitig zu klären.



Ausgewiesener Architekt als „Architecture Owner“ im Team

Architektur ist Teamsache!

„Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.“

(Das agile Manifest)

“Architecture should be a team effort.”

(Scott Ambler)



Ein
Widerspruch?
Nein!

Architektur benötigt Erfahrung!

„Daher muss der Architekt begabt sein und fähig und bereit zu wissenschaftlich-theoretischer Schulung. Und er muss im schriftlichen Ausdruck gewandt sein, des Zeichenstiftes kundig, in der Geometrie ausgebildet sein, mancherlei geschichtliche Ereignisse kennen, fleißig Philosophen gehört haben, etwas von Musik verstehen, nicht unbewandert in der Heilkunde sein, juristische Entscheidungen kennen, Kenntnisse in der Sternkunde und vom gesetzmäßigen Ablauf der Himmelserscheinungen besitzen.“ ☺

(Vitruv)



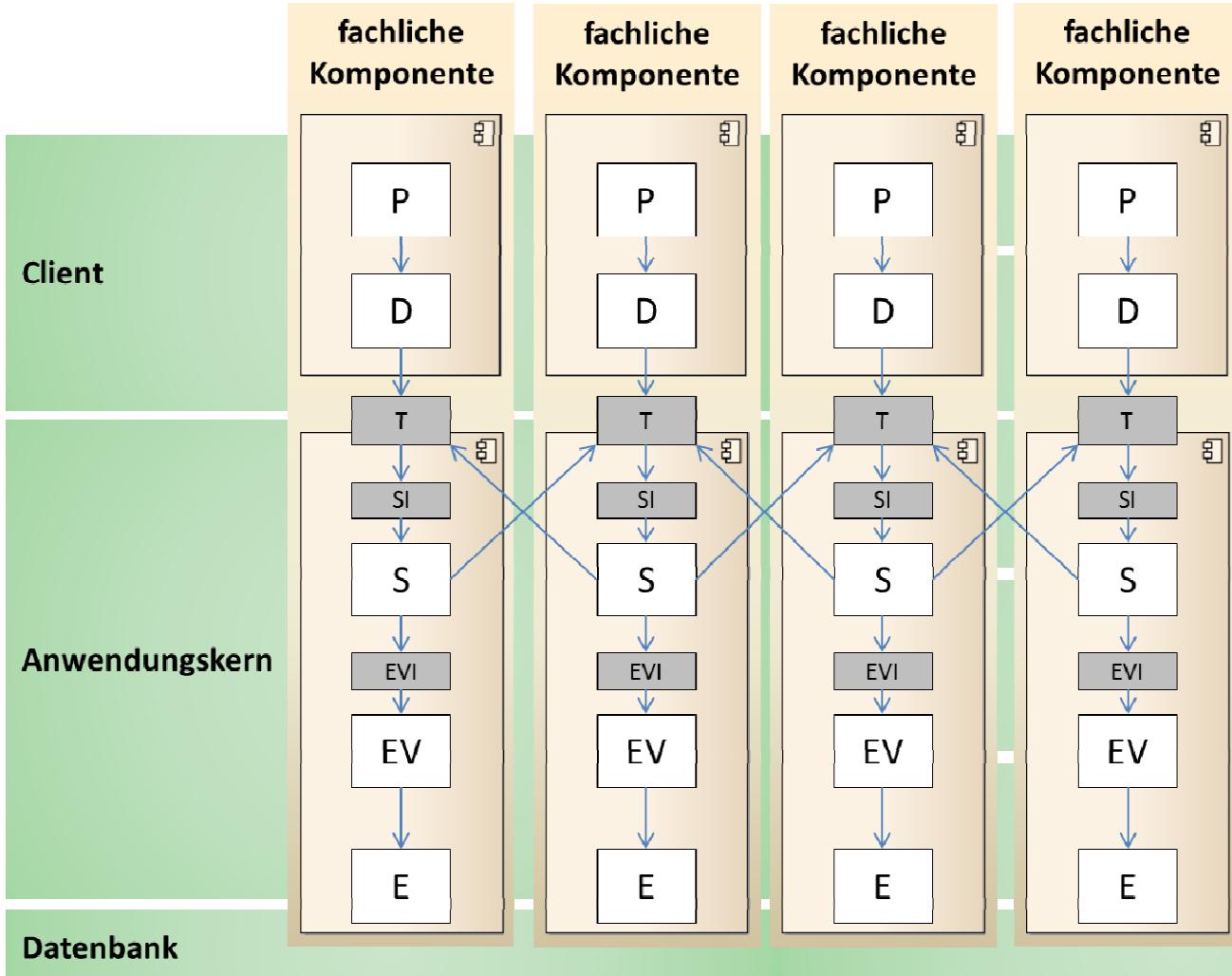
„Architecture Owner“



! Auch höchste Motivation im selbstorganisierten Team ersetzt nicht die Erfahrung eines Experten.



Referenzarchitekturen nutzen



Präsentationsschicht
(Presentation Layer)

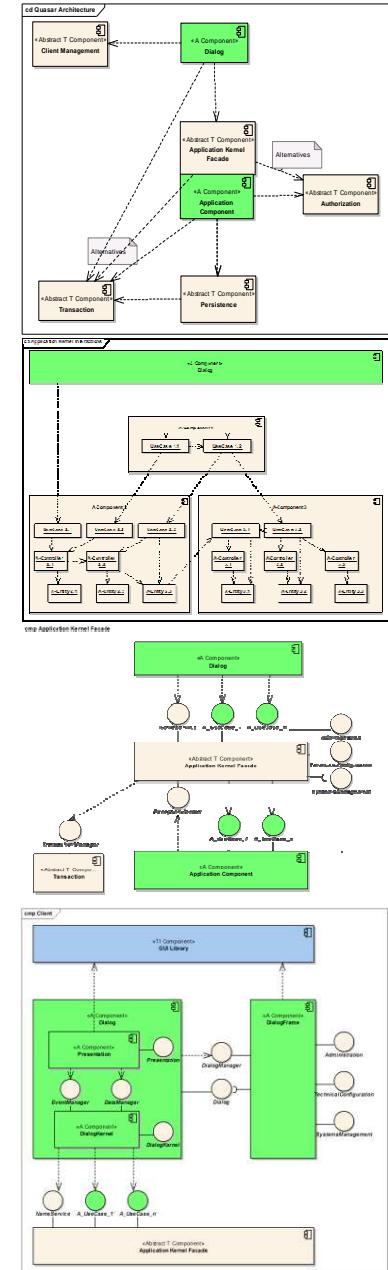
Dialogkernschicht
(Dialog Core Layer)

Servicezugriffsschicht
(Service Access Layer)

Geschäftslogikschicht
(Business Logic Layer)

Datenzugriffsschicht
(Data Access Layer)

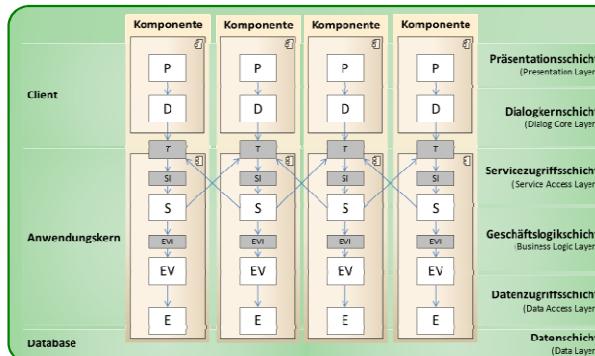
Datenschicht
(Data Layer)



Es ist wichtig, die Architektur eines Software-Systems gedanklich klar von der Implementierungstechnologie zu trennen. Referenzarchitekturen enthalten grundlegende rein architektonische Strukturierungen, durch deren Wiederverwendung vermieden werden kann, gleiche Fehler wieder und wieder zu machen.

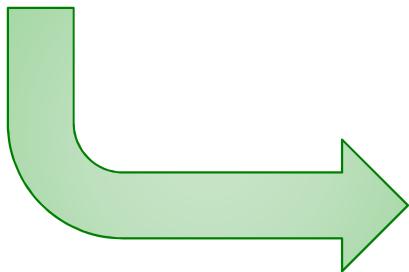


Referenzarchitekturen auf die speziellen nicht-funktionalen Anforderungen zuschneiden

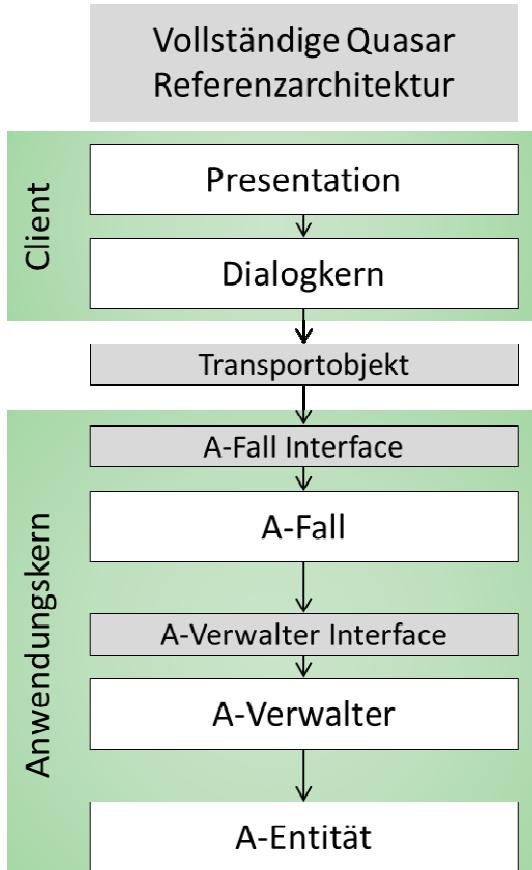


Beispiel: Quasar

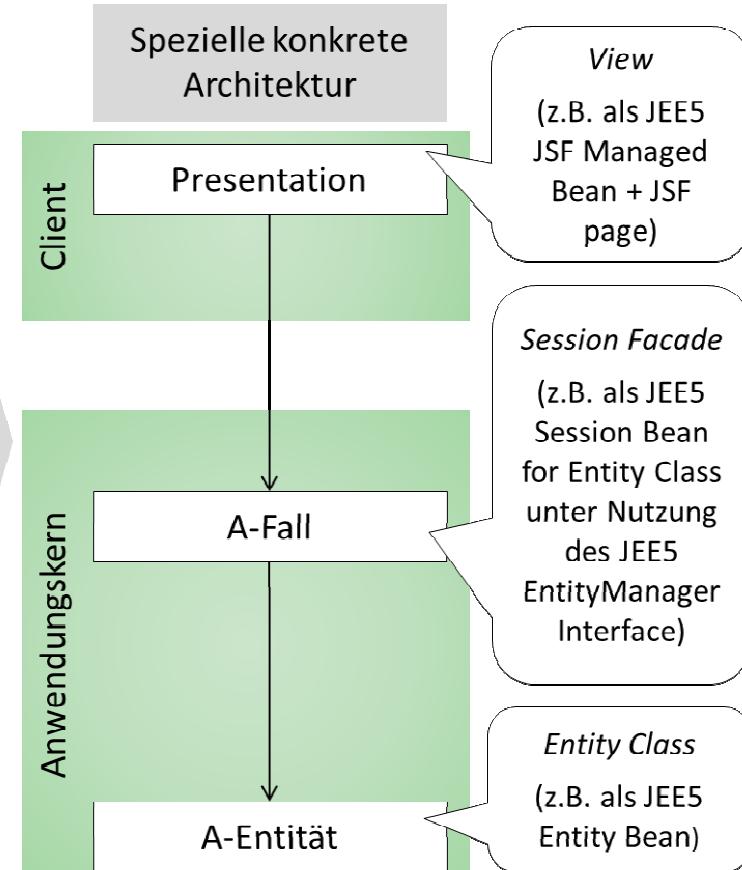
- z.B. bei reduzierten Anforderungen an Oberflächenkomplexität
- z.B. bei reduzierten Anforderungen an Plattformunabhängigkeit
- z.B. bei reduzierten Anforderungen an die Komplexität der Geschäftslogik
- z.B. bei reduzierten Anforderungen an Technologieunabhängigkeit
(Einlassen auf moderne Enterprise Technologien wie JEE5/6 oder .NET)



Der Trade-Off zwischen Anforderungen und architektonischer Komplexität muss explizit erfolgen. Das benötigt viel Erfahrung.



Spezieller Zuschnitt der Referenzarchitektur unter reduzierten nicht-funktionalen Anforderungen und mit Bindung an konkrete Technologie JEE5



Agenda

- Accso
- Beschleunigte Softwaretechnik (BeST)
- Ausgewählte Aspekte
 - BeST-Practices im Projektmanagement
 - BeST-Practices der Analyse
 - BeST-Practices der Architektur
- Abschluss

Lassen Sie uns in Kontakt bleiben ...

- www.accso.de
- twitter.com/accso
- [Xing](#)
- [Facebook](#)

The screenshot shows the homepage of the ACCSO website. At the top right, there is a search bar with the placeholder "Suche...", a "Kontakt | Impressum" link, and social media icons for LinkedIn, Facebook, and Twitter. The main header features the ACCSO logo with the tagline "Accelerated Solutions". Below the header, a large banner image shows five professionals (three men and two women) standing together. To the left of the banner, a white callout box contains the text "UNSER TEAM BEI DB SCHENKER RAIL" and a description of ACCSO's support for DB Schenker Rail. A navigation menu at the bottom of the page includes links for Accso, Angebot, BeST, Beratung, Referenzen, Team, Jobs, News, and Blog.

WILLKOMMEN BEI ACCSO!

Accso - Accelerated Solutions GmbH ist ein Software- und IT-Beratungshaus. Wir entwickeln individuelle Softwarelösungen für die Kernkompetenzen unserer Kunden und beraten in aktuellen Fragestellungen von Technologie und Architektur. Mit unserem erstklassig ausgebildeten Onsite-Team und Methoden der **Beschleunigten Softwaretechnik** erstellen wir Qualitäts-Lösungen besonders konkurrenzfähig. Die jahrelange Erfahrung unserer Experten ermöglicht unseren Kunden eine sichere Abwicklung Ihrer Projekte.

NEWS

Accso freut sich über den Zuschlag in der Ausschreibung "Meteorologisches Arbeitsplatzsystem NinJo" beim Deutschen Wetterdienst.

[Weiterlesen... ▾](#)

Wir begrüßen im Februar herzlich unsere zwei neuen Kollegen Yannik Tessa und Sebastian Metz an unserem Standort in Darmstadt.

Unser Blog zu aktuellen Themen der Softwaretechnik

- blog.accso.de

The screenshot shows the homepage of the blog accso.de. At the top, there is a navigation bar with links to 'Home', 'Accso', and 'Feeds'. Below the navigation, there is a large image of four people in professional attire, identified as 'UNSER TEAM BEIM BAFA'. To the right of the image, a text box states: 'Gemeinsam mit dem Bundesamt für Wirtschaft und Ausfuhrkontrolle arbeiten wir an der Weiterentwicklung der IT-Systeme zur Abwicklung von Förderprogrammen und zur Ausfuhrkontrolle.' Below the image, there is a section titled 'Datenbankanbindung mit MyBatis' with a subtext 'Veröffentlicht am 10. Januar 2015 von Iryna Schmidt'. Underneath this, there is a heading 'Was ist MyBatis?' followed by a detailed explanation of what MyBatis is and how it works. Further down, there is another section with text about MyBatis abstraction. On the right side of the page, there is a sidebar titled 'Archive' containing a list of months from January 2013 to January 2015.

Accso
Accelerated Solutions

UNSER TEAM BEIM BAFA

Gemeinsam mit dem Bundesamt für Wirtschaft und Ausfuhrkontrolle arbeiten wir an der Weiterentwicklung der IT-Systeme zur Abwicklung von Förderprogrammen und zur Ausfuhrkontrolle.

Datenbankanbindung mit MyBatis

Veröffentlicht am 10. Januar 2015 von Iryna Schmidt

Was ist MyBatis?

MyBatis ist ein Framework für die Datenbankanbindung, das durch seine Einfachheit besticht. Es unterstützt benutzerdefinierten SQL-Code, Stored Procedures sowie ein fortgeschrittenes Mapping von komplexen Joins und Objektgraphen.

Die SQL-Anweisungen werden vom Entwickler geschrieben und in XML-Dateien gespeichert. MyBatis erstellt daraus automatisch PreparedStatement und verbirgt dazugehörigen JDBC-Code. Die Ergebnisse der SQL-Abfragen werden automatisch in den Objekten abgebildet.

Das Framework kann ein eigenes Transaktionsmanagement für die Operationen an der Datenbank einsetzen oder ein externes von Spring, EJB CMT, etc. nutzen.

MyBatis abstrahiert also von solchen Details der Datenbankkommunikation wie Laden der Treiber, Instanziieren und Managen der Connection, Verwalten der Transaktionen, etc.

Archive

- Januar 2015
- Dezember 2014
- November 2014
- August 2014
- Juli 2014
- Juni 2014
- April 2014
- März 2014
- Februar 2014
- Januar 2014
- Dezember 2013
- November 2013
- Oktober 2013
- September 2013
- August 2013
- Juli 2013
- Juni 2013
- Mai 2013

Wir freuen uns über Ihre Bewerbung ...

- Werkstudenten
- Abschlussarbeiten
- Berufseinstieg



Suche...

Kontakt | Impressum



WACCSEN

So nennen wir unser Programm, mit dem wir weiter auf Wachstumskurs bleiben wollen.



Accso | Angebot | BeST | Beratung | Referenzen | Team | Jobs | News | Blog

Manche Erkenntnisse veralten nicht: Spitzenleistungen - in der IT wie in anderen Branchen - sind das Ergebnis einiger weniger zentraler unternehmerischer Grundtugenden. Persönlicher Erfolg und individuelle persönliche Entwicklung, Partizipation und Eigenverantwortung, Pragmatismus, gelebte Werte und echte Kundenorientierung sind entscheidend für den Erfolg.

Accso - das ist ein hochkarätiges, eigenverantwortliches, pragmatisches und kundenorientiertes Team, das weiter wächst. Wir suchen die Besten auf ihrem jeweiligen Gebiet - Macher und Köner mit verschiedenen Qualifikationen. Berufserfahrene IT-Fachleute sind bei uns genauso willkommen, wie Hochschulabsolventen mit überdurchschnittlichem Abschluss in Informatik oder einer verwandten Disziplin. Werkstudenten können bei Accso erste Industriearbeit sammeln.

Erfahren Sie, was arbeiten bei Accso bedeutet, wie Sie bei Accso Ihre Ziele erreichen können und wie Sie gemeinsam mit Accso beschleunigen. Einige Informationen finden

WILLKOMMEN

WARUM ACCSO?

STUDENTEN

BERUFSEINSTEIGER

BERUFSERFAHRENE

FÜHRUNGSKRÄFTE

BEWERBUNG

DOWNLOADS



*Individuelle
Kernsysteme*

*Beschleunigte
Softwaretechnik*

Team=

Begeisterung für die
anspruchsvollen Aufgaben unserer Kunden

ACCSO=
Accelerated Solutions