



# Einführung

Software Architektur

Tim Lüecke (tim.luecke@haw-hamburg.de)





# Agenda

## ■ **Software-intensive Systeme**

- (Software-)Architektur
- Methoden für den Entwurf
- Architektur im Software-Entwicklungsprozess
- Rolle des Architekten
- Historisches



# Arten von Software-intensiven Systemen



Spiele



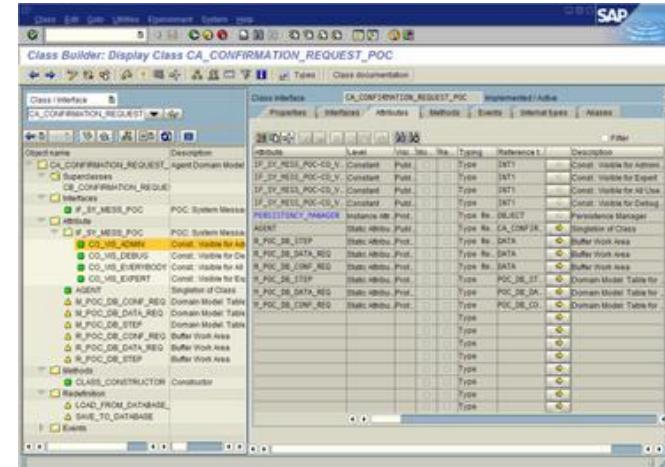
Wettsysteme



Eingebettete  
Systeme



Mobile  
Systeme



Informationssysteme



# Arten von Software-intensiven Systemen

## Eingebette Systeme

- Hardware-nahe System, welche physische Komponenten enthält
- Als Software durch die Hardware "verdeckt" und nicht als Software-System zu erkennen
- Läuft auf spezieller, eingeschränkter Hardware und muss daraufhin angepasst werden
- Oft sicherheitskritisch
- Lange Lebenszyklen
- *Beispiele:*
  - Fahrzeugsteuerung
  - Steuerung für Haustechnik
  - Werkzeugmaschinen-Steuerung
  - ...

## Echtzeitsystem

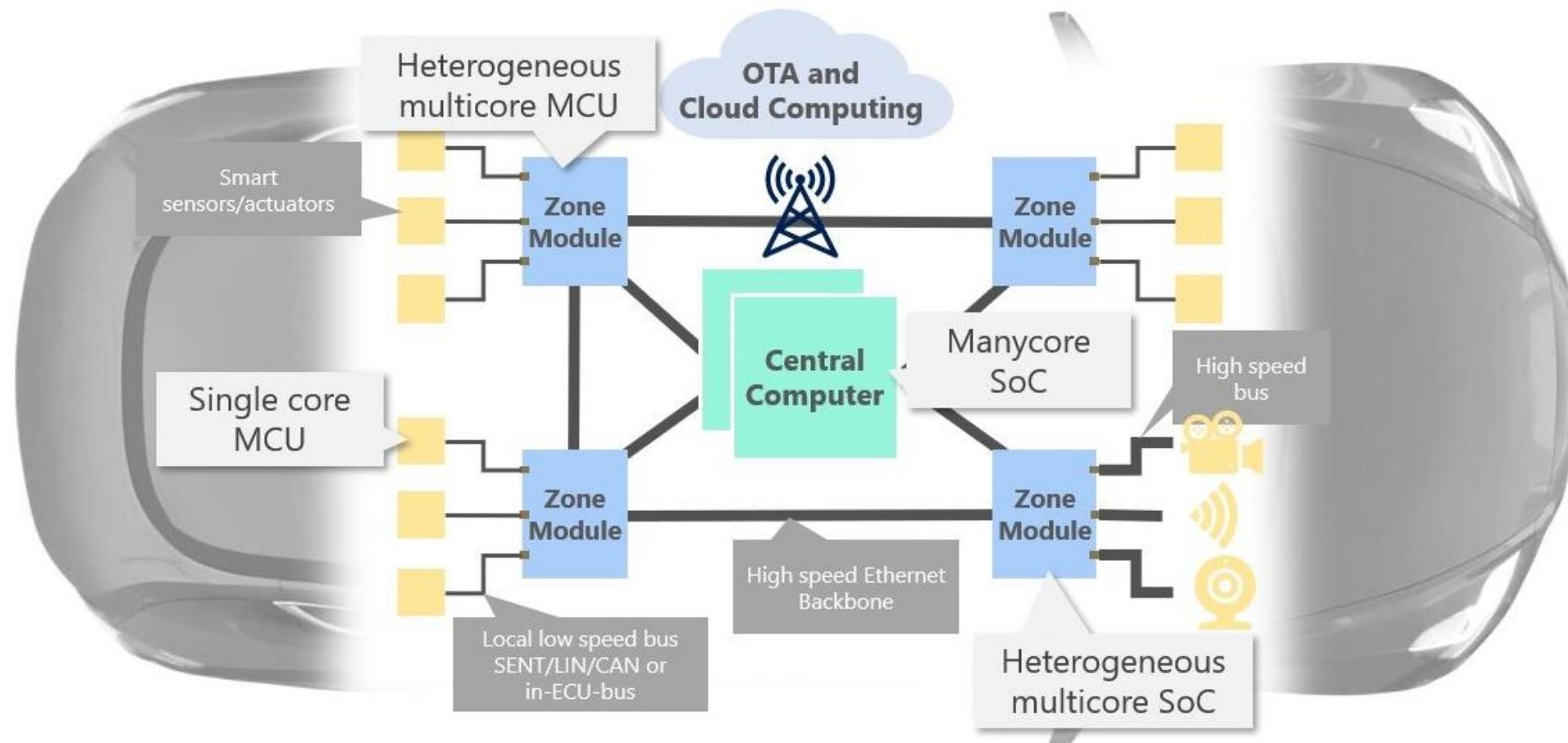
- Systeme, welche in Echtzeit mit dem Benutzer interagieren
- Manchmal auch beinahe Echtzeit
- Häufig verbunden mit einer großen Nutzeranzahl, die das System zeitgleich benutzt
- Muss unterschiedliche Lasten vertragen
- Oft sind Eingebettete Systeme auch Echtzeitsysteme!
- *Beispiele:*
  - Wettsystem
  - MORPG
  - E-Commerce Systeme
  - Streaming Dienste
  - ...

## Informationssysteme

- Systeme meist für den Einsatz in Unternehmen als Backoffice-System
- Fokus auf der Verarbeitung und Zubereitung von Daten
- Oft intern verwendet und mit eingeschränkter Nutzeranzahl (=Mitarbeiter)
- Konzipiert für Experten und nicht den 08/15 Benutzer
- Oft auch global verwendet bei großen Konzernen
- *Beispiele:*
  - SAP Systeme
  - Transport
  - Managementsysteme
  - Warenwirtschaftssystem
  - ...



# Beispiel: E/E-Architektur bei modernen Fahrzeugen



Quelle: <https://www.all-electronics.de/e-mobility/leistungsfähigkeit-und-sicherheit-von-e-fahrzeugen-optimieren-382.html>



# Beispiel Echtzeitsystem: Amazon

amazon.de Lieferung an Düsseldorf 40213 Standort aktualisieren Angebote Suche Amazon.de DE Hallo, anmelden Konto und Listen Warenrücksendungen und Bestellungen Einkaufswagen

☰ Alle Bestseller Amazon Basics Neuerscheinungen Angebote Prime Video Musik Prime Shopping-Tipps Bücher Mode Gutscheine Lebensmittel Küche, Haushalt & Wohnen Geschenkideen

Angebote Outlet Amazon Retourenkauf Amazon Renewed Coupons Fashion-Sale Family Student Spar-Abo Mehr kaufen und sparen Geschenke Amazon Apps

Entdecke Artikel für den Schulanfang Jetzt entdecken ▶

Trendangebote Blitzangebote Angebote unter 20€ Amazon Geräte Smartphones & Zubehör Computer & Software TV & Filme Amazon Marken Kosmetik Mode, Schuhe & Taschen Spielwaren Küche, Haushalt & Möbel

Kategorie  
Alle  
Amazon-Geräte & Zubehör  
Auto & Motorrad  
Baby  
Baumarkt  
Mehr

Kundenrezensionen  
Alle  
5★ & mehr  
4★ & mehr  
3★ & mehr  
2★ & mehr

Preis  
Alle  
Unter 20 €  
20 € bis 50 €  
50 € bis 100 €  
100 € bis 200 €

50 % Rabatt Befristetes Angebot Amazon eero Pro 6 Tri-Band-Mesh WiFi-6-System...  
60 % Rabatt Befristetes Angebot Amazon Fire TV-4-Serie Smart-TV, 55 Zoll (140 cm)...  
8 % Rabatt Befristetes Angebot dreame L40 Ultra Roboterstaubsauger mit a...  
33 % Rabatt Befristetes Angebot De'Longhi Magnifica S ECAM 22.110.B Kaffeevollautom...  
27 % Rabatt Befristetes Angebot roborock Qrevo S Saugroboter mit Wischfunk...  
20 % Rabatt Befristetes Angebot Aqara Smart Lock M200 (mit Fingerabdruck), Matter ov...  
Spare 16,00 € mit Rabattgutschein

Produktübersicht: 1. Amazon eero Pro 6 Tri-Band-Mesh WiFi-6-System (50 % Rabatt) 2. Amazon Fire TV-4-Serie Smart-TV, 55 Zoll (140 cm) (60 % Rabatt) 3. dreame L40 Ultra Roboterstaubsauger mit a... (8 % Rabatt) 4. De'Longhi Magnifica S ECAM 22.110.B Kaffeevollautom... (33 % Rabatt) 5. roborock Qrevo S Saugroboter mit Wischfunk... (27 % Rabatt) 6. Aqara Smart Lock M200 (mit Fingerabdruck), Matter ov... (20 % Rabatt)



# Beispiel Informationssystem: SAP Ariba

The screenshot shows the SAP Ariba interface for the 'Sourcing' module. The top navigation bar includes links for 'HOME', 'VISIBILITY', 'SOURCING' (which is highlighted in blue), and 'MEER...'. The main content area features several dashboard cards:

- Event Status:** A donut chart showing the distribution of events across Draft (87), Preview (1), and Over (1).
- Supplier Approvals:** A bar chart showing the count of Pending (1), Approved (1), and In Progress (1) supplier approvals.
- My Tasks:** Displays 0 Completed Tasks.
- Expiring Contracts:** A bar chart showing the count of contracts expiring in 1 day (1), 7 days (2), and 30 days (1).

Below these cards is a section for **Notifications**, which currently shows "No items".

The left sidebar contains a vertical navigation menu with sections like 'Common Actions', 'Recently Viewed', and 'Sourcing Project Analysis'. Under 'Sourcing Project Analysis', there are several sub-options: Active Projects by Owner, Active Projects by Status, Actual Savings Report, Baseline Spend by Project, Duration of Projects by Start Date, and Project Templates.

At the bottom right, there is a table titled 'Event Status (Last 12 months)' with columns for Draft, RFP, Auction, Survey, and Forward Auction. The data is summarized in the following table:

Status	Draft	RFP	Auction	Survey	Forward Auction
Draft	43	152	18	80	2
Preview	0	7	0	0	0
Over	1	8	4	4	0
Pending Selection	16	105	38	0	2
Completed	9	91	24	118	2

**Fokus in dieser Vorlesung!**

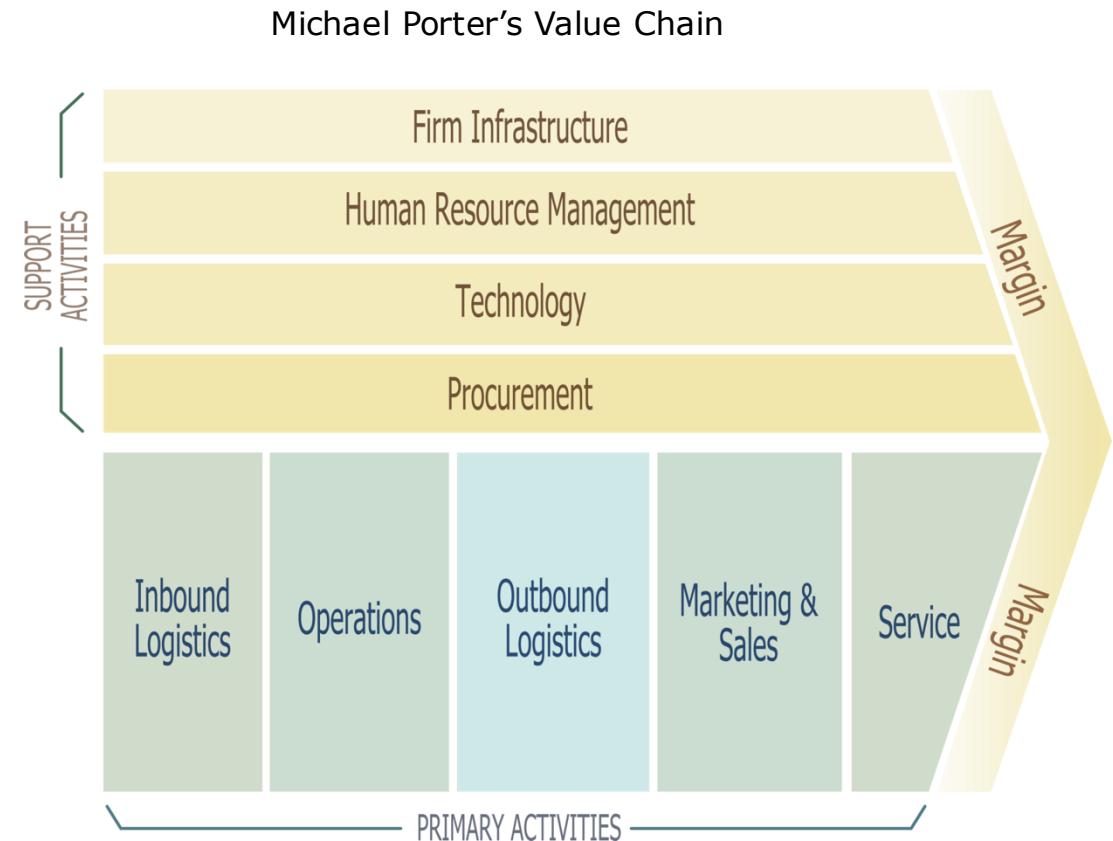
Langweilig?





# Was ist ein (betriebliches) Informationssystem?

- Ziel eines Betriebes?
  - Gewinn erwirtschaften
  - Soziale Verantwortung
- Wie?
  - Organisation
  - Prozesse
  - Informationen!
- Ziel eines Informationssystems?
  - Versorgung der Organisation und Kunden mit Daten
  - Möglichst effizient:
    - Zum richtigen Zeitpunkt im Prozess
    - An die richtige Person
    - Wirtschaftlich

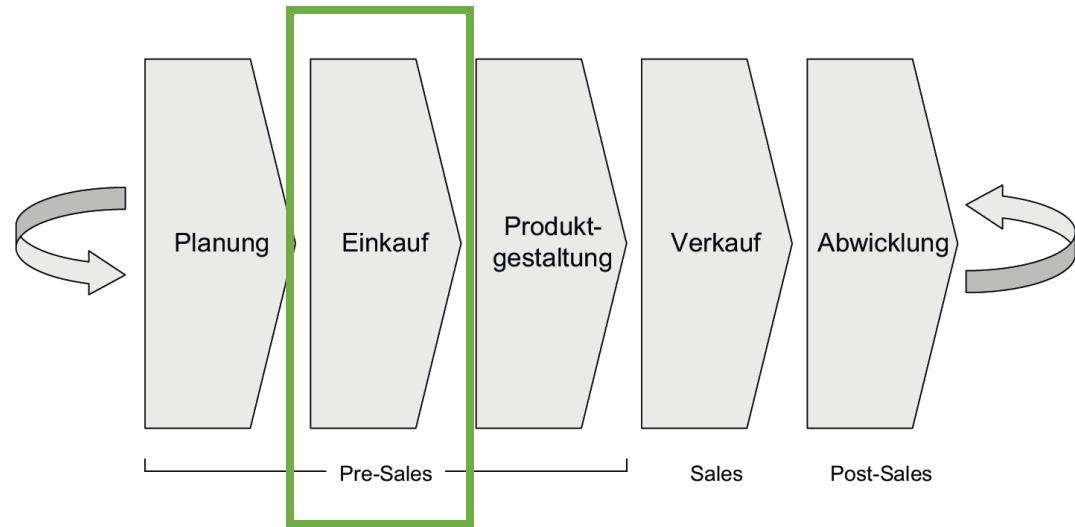


Source: [https://en.wikipedia.org/wiki/Value\\_chain](https://en.wikipedia.org/wiki/Value_chain)



# Unser durchgehendes, fiktives Praxisbeispiel: ein Reiseveranstalter

## Der touristische Kreislauf:



**Warnung:** im folgenden handelt es sich um  
fiktive Anforderungen und Annahmen  
(kein Anspruch auf fachliche Korrektheit!)

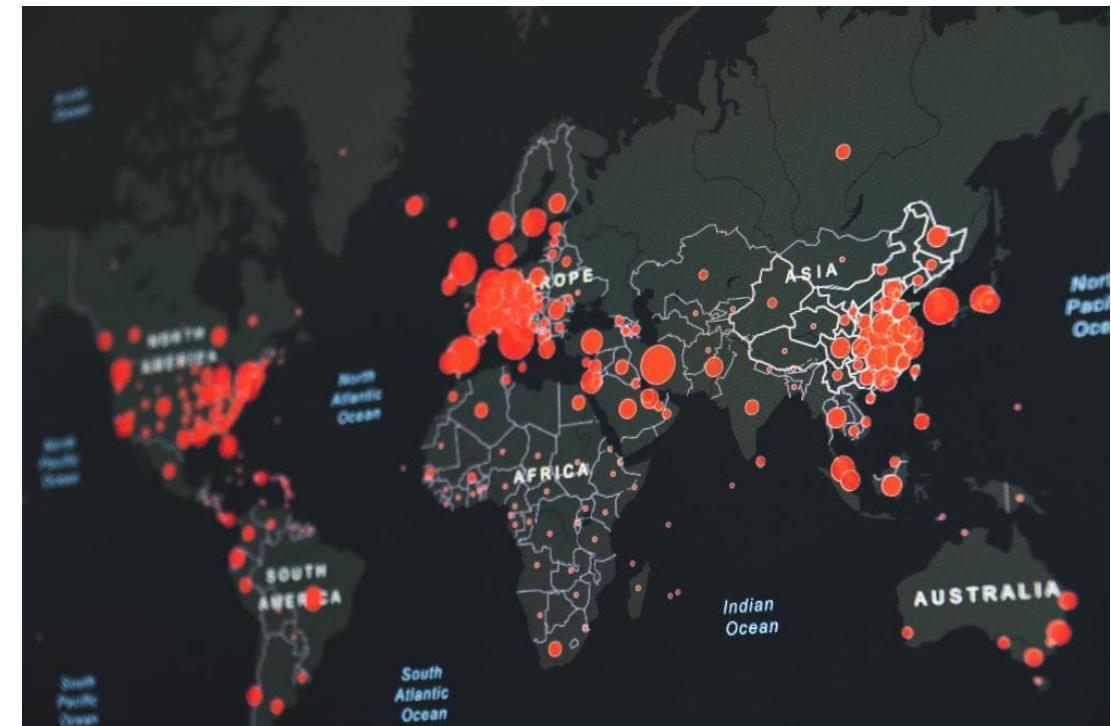
## Fokus für unsere Vorlesung:

- Das System: *ERK* (*Einkauf Reise Kapazitäten*)
- Informationssystem zur Steuerung des Einkaufprozesses
- Ermöglicht Verwaltung der Anbieter
  - Performance Rating
  - Risk Analysis
- Ermöglicht Pflege der Saisonalen Kontrakte mit Anbietern
  - Raten
  - Konditionen
  - Leistungen
- Integriert mit Systemen der Anbieter



# Die Ausgangsbasis für den Einkaufsprozess bildet die Planung

- Wieviele Reisen in welche Urlaubsregion?
  - In welche Stadt?
  - In welches Hotel?
  - Wieviel % inkl. Autovermietung?
  - Wieviel Städtereisen?
- Basierend auf Werten der Vergangenheit
- Forecast auf die Zukunft inkl. Wirtschaftsindikatoren etc.
- **Wichtig:**
  - für ERK ist nur das Ergebnis wichtig
  - Verantwortung für die Erstellung liegt bei anderen Systemen
  - Daten müssen jedoch übernommen werden





# Weitere Basis: Stammdaten

- Geographie der Urlaubsregionen
  - Städte
  - Hotels
  - Ausflugsziele
  - Transportmittel
- Liste der Dienstleistungen für Urlaube
  - Transport
  - Aufenthalt
  - Veranstaltungen
  - Ausflüge
- Liste der Anbieter
  - Fluglinien
  - Schifffahrtslinien
  - Busunternehmen
  - Event Manager
  - Ausflugsunternehmen
  - Hotels
- Stammdaten sind auch für andere Systeme relevant (zentral verwaltet)
- Müssen für ERK erweiterbar sein





# Komplex: Raten Management

- Raten pro Anbieter and Saison
- Bilden die Kosten für eine Reise ab (nicht die Einnahmen!)
- Unterschiedlich pro Typ:
  - Transport: inkl. km. Entfernung
  - Hotel: pro Zimmer (teilw. pro Guest)
- Verschiedene Mengengerüste:
  - Fixkosten für Reservierung von Kontingent
  - Flugpreis / Passagier pro 100km
  - ab 1000km: Fernreisepauschale
- Abgebildet in flexiblen Ratenstrukturen
- Müssen später in der Abrechnung von Anbieterrechnungen abgeglichen werden

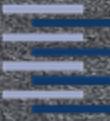




# Das Herzstück: der Einkauf (Procurement)

- Legt konkrete Mengengerüste auf Basis der Raten fest
  - Mindestkontingente
  - Maximale Kontinente
  - potentielle Rabatte
- Werden jährlich verhandelt damit im Katalog ein Festpreis angeboten werden kann
- Simulation von Gewinnen
  - Preisannahmen <-> Bedarsplanung  
-> Umsatz
  - Bedarfsplanung -> Ratenstruktur  
-> Kosten
  - $\text{Gewinn} = \text{Umsatz} - \text{Kosten}$
  - Verschiedene Szenarien sollen simuliert werden können
- Benötigt Vier-Augen-Prinzip
  - Einkauf wird geplant und vorgelegt
  - Einkaufsmanager für Region muss akzeptieren oder ablehnen





Wozu braucht man eine Architektur  
für Software-intensive Systeme?



# Architekturen erfüllen Anforderungen!

Anforderungen = ???

= Funktionale Anforderungen

+

Nich-funktionale Anforderungen

Effizienz

Planbarkeit

...

Wartbarkeit

Konfigurierbarkeit

**Je nach Systemtyp und pro Kontext sehr unterschiedlich!**



# Agenda

- Software-intensive Systeme
- **(Software-)Architektur**
- Methoden für den Entwurf
- Architektur im Software-Entwicklungsprozess
- Rolle des Architekten
- Historisches



# Was ist Architektur?



lat. *architectura* "Baukunst"

„**handwerkliche** Beschäftigung und **ästhetische** Auseinandersetzung des Menschen mit dem gebauten Raum“

Quelle: <https://de.wikipedia.org/wiki/Architektur>

## Beispiele:



Architektur zur  
Verteidigung



Architektur zur  
Organisation



Architektur zur  
Mobilität



# Wie sieht Software-Architektur aus? So?



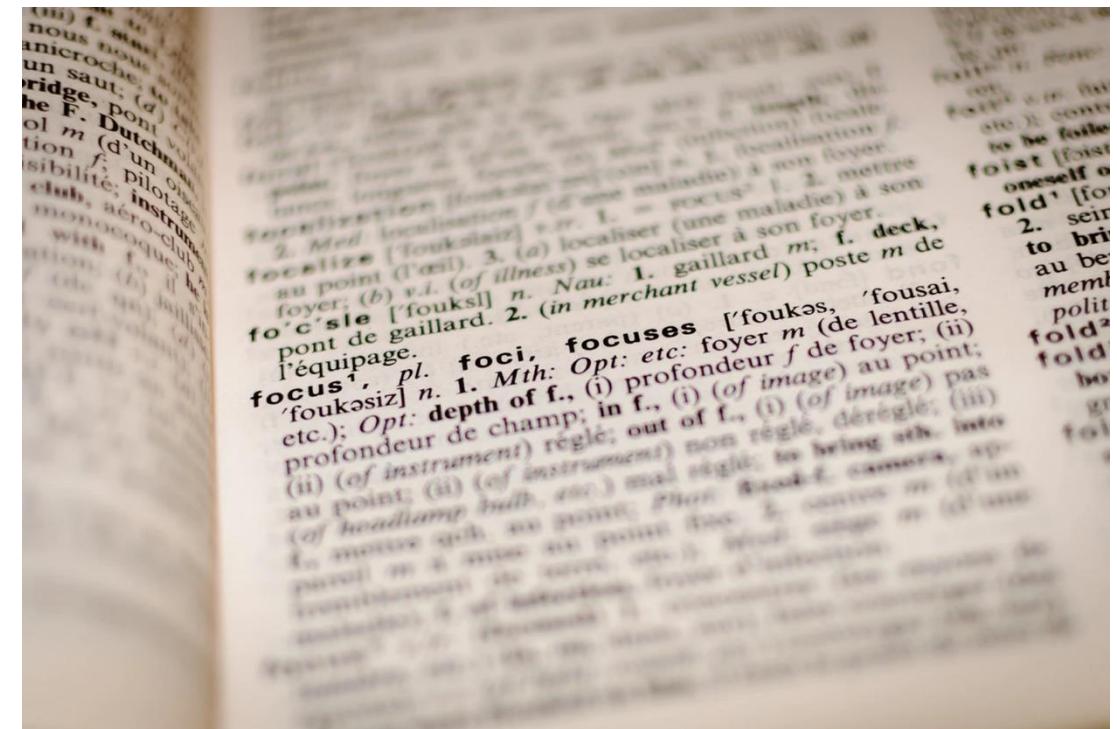
ERK

A large, solid dark blue rectangular box occupies the center of the slide. In its upper right quadrant, the letters "ERK" are displayed in a bold, white, sans-serif font.



# Was ist Software-Architektur?

- Prinzipien und Design Entscheidungen eines Systems
- Bei Informationssystemen insbesondere:
  - Komponenten / Bausteinen / Services
  - Schnittstellen
  - Abhängigkeiten
  - dargestellt in verschiedenen Sichten (Bauplänen)
- Guter Leitfaden: [arc42](#)





# Architektur als Artifakt (z.B. nach arc42)

1. Introduction and goals
2. Constraints
3. Context and scope
4. Solution strategy
5. Building block view
6. Runtime view
7. Deployment view
8. Crosscutting concepts
9. Architecture decisions
- 10.Quality
- 11.Risks and technical debt
- 12.Glossary





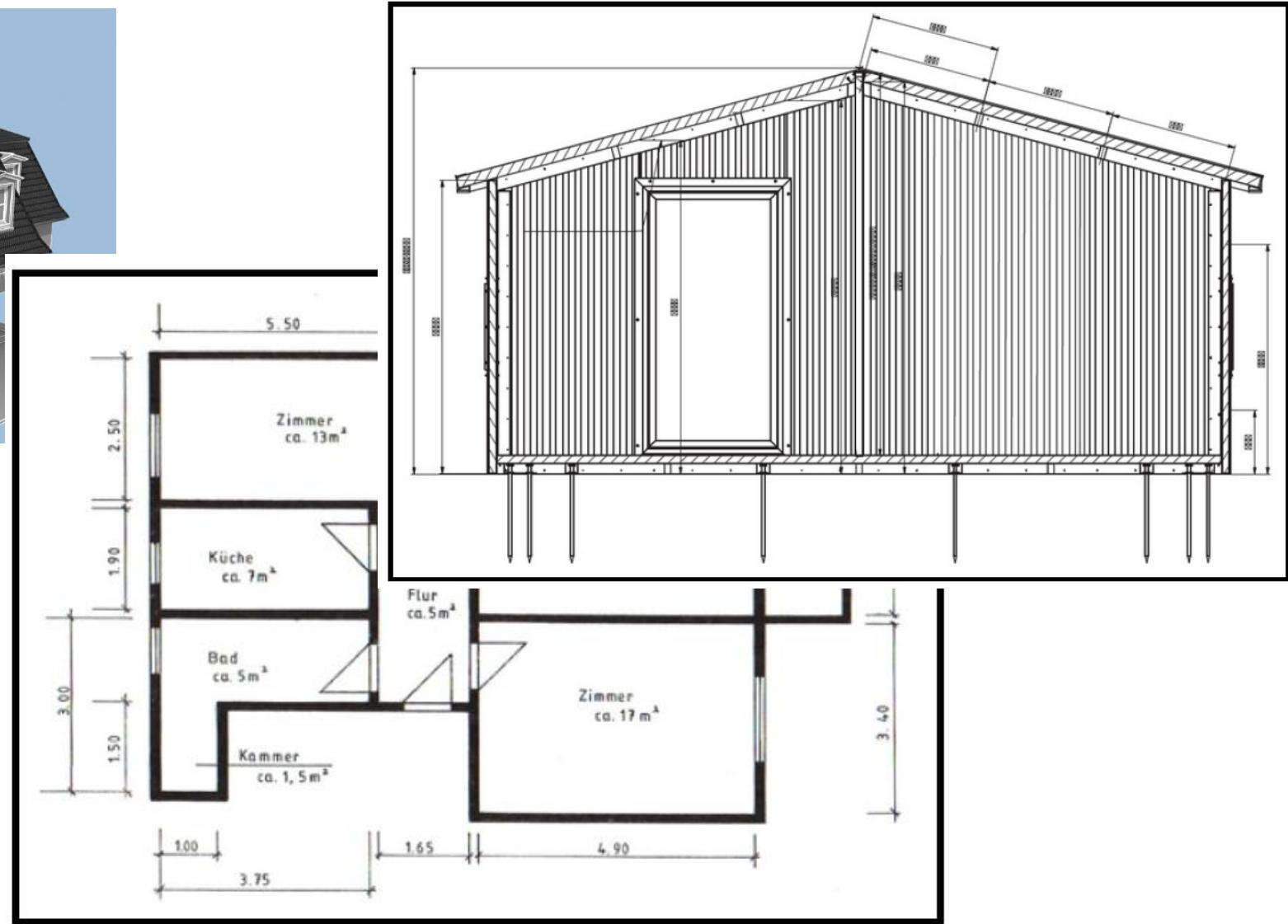
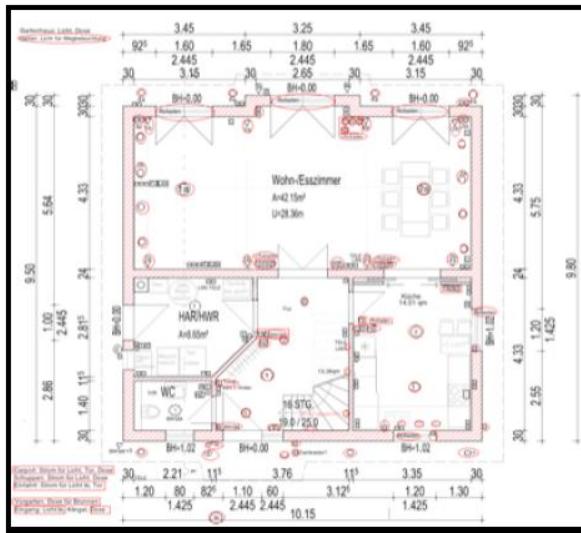
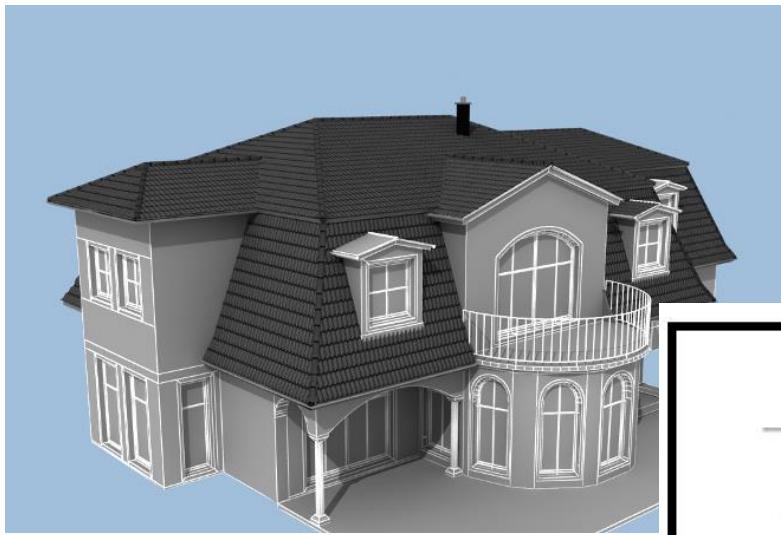
# Sichten beleuchten unterschiedliche Aspekte der Architektur

- Trennen von Zuständigkeiten
- Architektur aus unterschiedlichen Perspektiven
- Für unterschiedliche Stakeholder
- Standard-Sichten:
  - Kontextabgrenzung
  - Bausteinsicht
  - Laufzeitsicht
  - Verteilungssicht



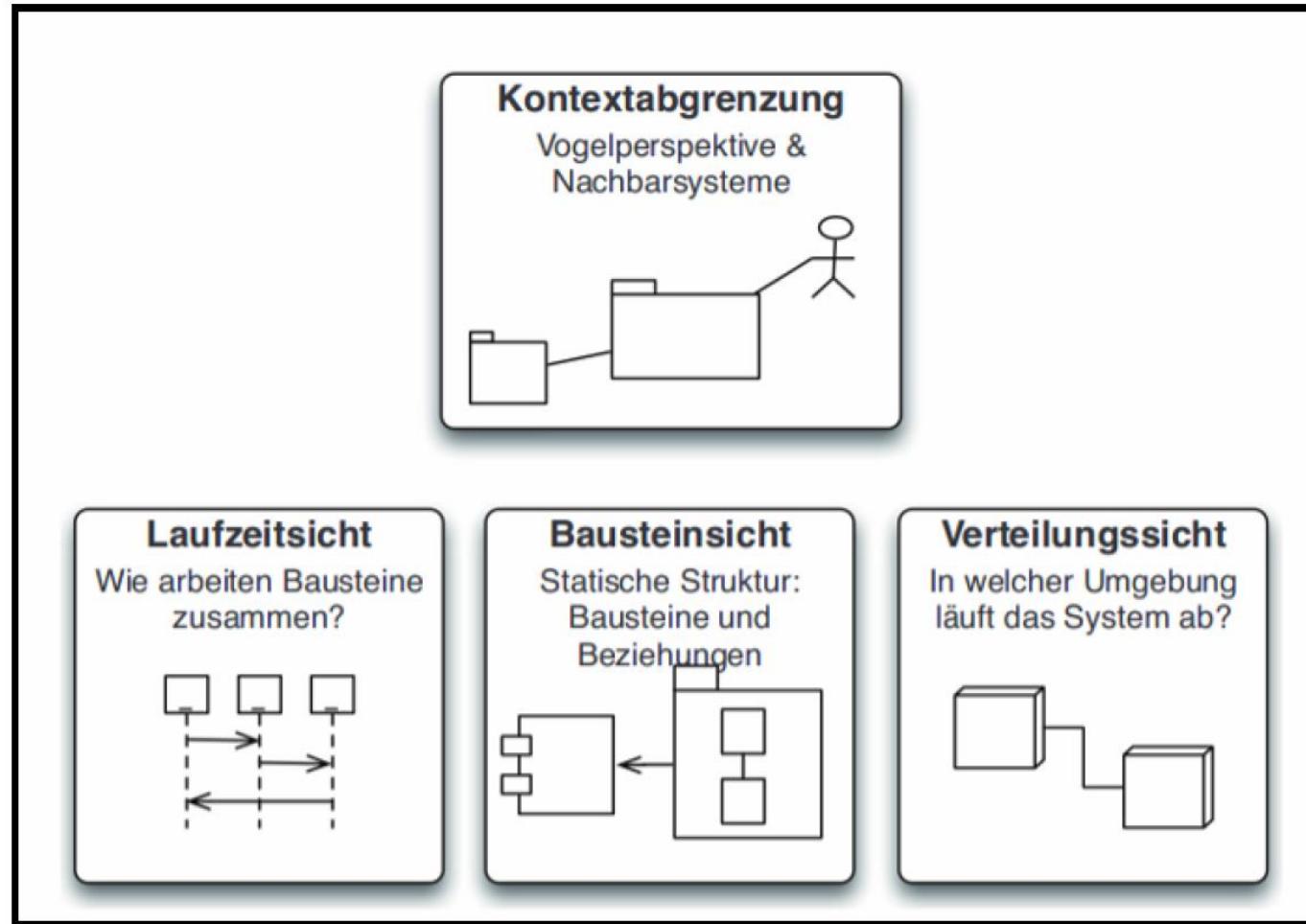


# Beispiele aus dem Gebäudebau





# Vier Arten von Sichten nach Gernot Starke



Quelle: "Effektive Softwarearchitekturen" von Gernot Starke

# Abgrenzung der alltäglichen Architektur zur Software Architektur



## Alltägliche Architektur:

- Schönes Äußeres
- Beeindruckende Strukturen
- Modern
- Etwas wertvolles, exklusives oder nützliches

## Software Architektur:

- Klarheit und Verständlichkeit
- Eleganz
- Funktionalität
- Einfachheit



# Weitere formale Definitionen

- "A software system's architecture is the set of principal **design decisions** about the system." (*Taylor*)
- Die Software-Architektur ist die grundlegende **Organisation eines Systems**, dargestellt durch dessen **Komponenten**, deren **Beziehungen** zueinander und zur **Umgebung** sowie die **Prinzipien**, die den Entwurf und die Evolution des Systems bestimmen. (*Reussner*)
- Eine Architektur ist die Menge der signifikanten **Entscheidungen** über die **Organisation** eines Systems:
  - Auswahl der **Strukturelemente** und ihrer **Schnittstellen**, aus denen sich ein System zusammensetzt
  - **Verhalten**, wie es in der Zusammenarbeit dieser Elemente spezifiziert ist
  - **Zusammenfassung** dieser Struktur- und Verhaltenselemente zu immer größeren Subsystemen
  - **Architekturstil**, der diese ganze Organisation beschreibt

(*Grady Booch*)

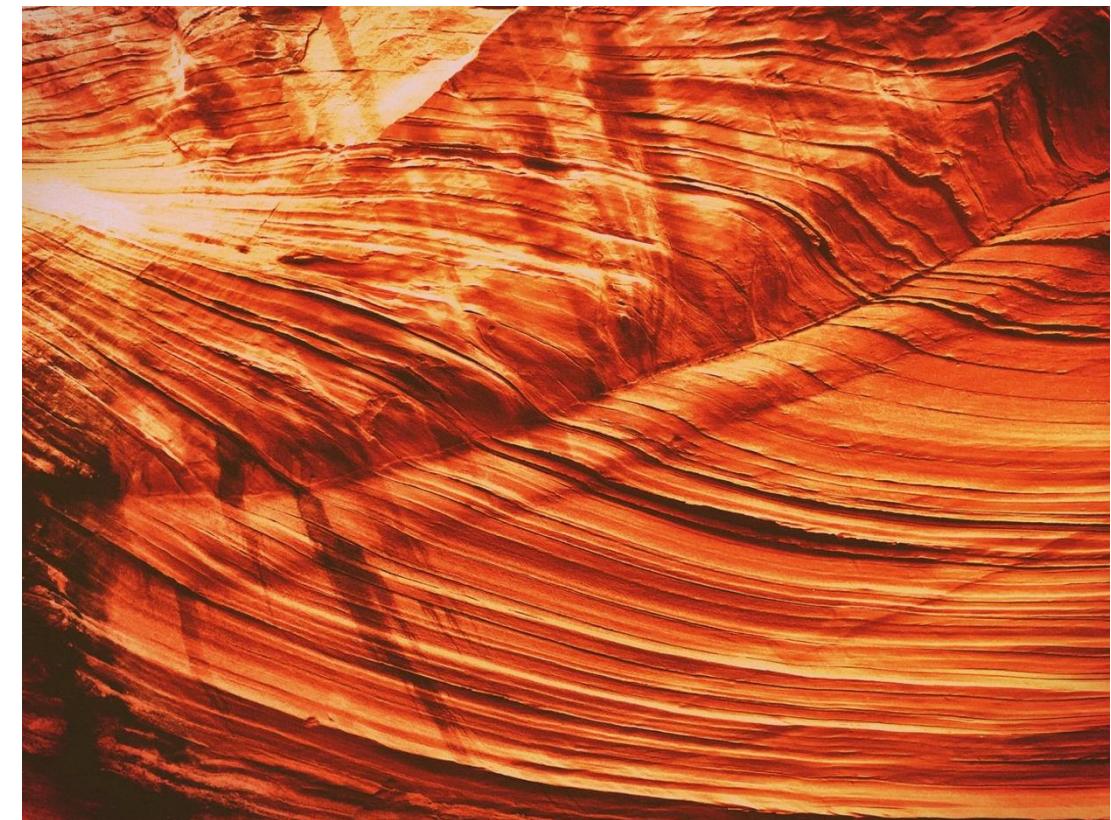
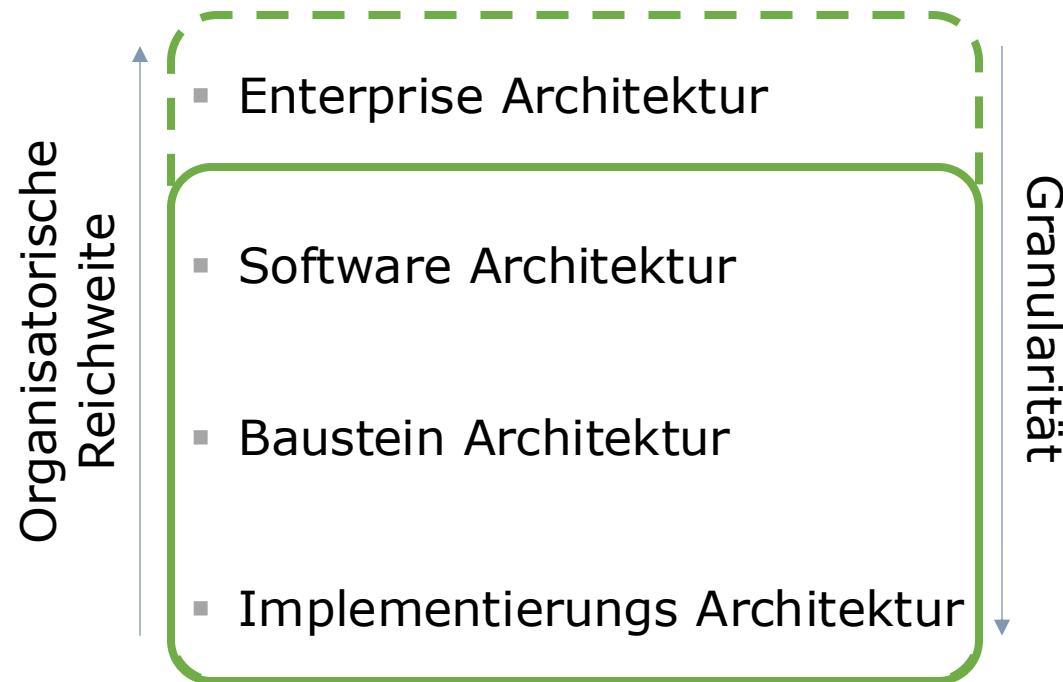


## „Catchy“ definitions – but maybe not so helpful...

- Software Architecture = { what, how, why } (*Perry and Wolf*)
- The software architecture of deployed software is determined by those aspects that are the hardest to change. (*Chris Verhoef*)
- The important stuff, whatever that is. (*Ralph Johnson*)
- Architektur ist die Menge der Entwurfsentscheidungen, die, wenn falsch getroffen, Ihr Projekt scheitern lassen können. (*Eoin Woods*)



# Architektur existiert auf unterschiedlichen Ebenen



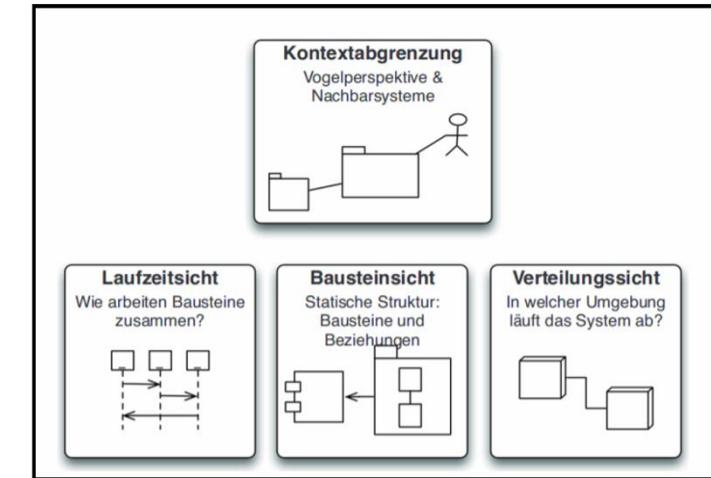
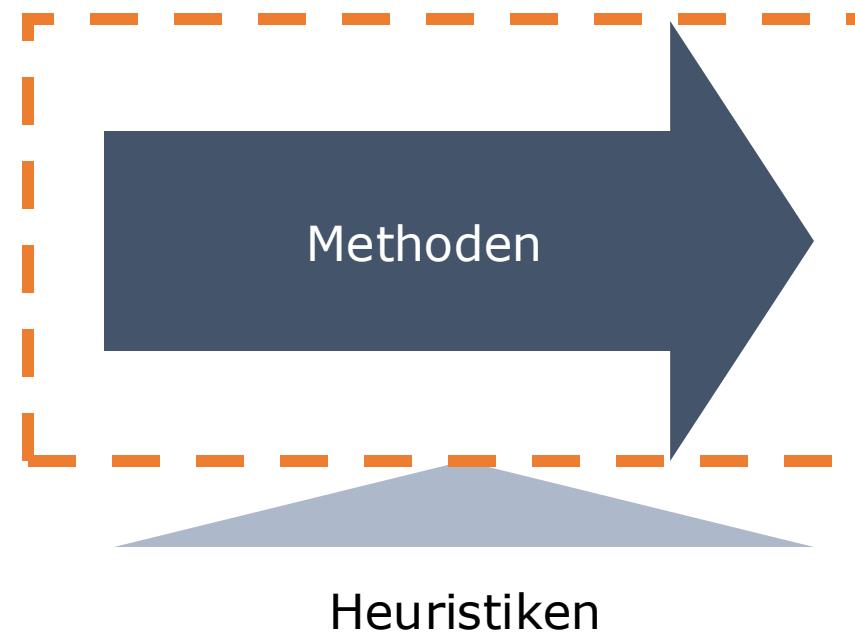


# Agenda

- Software-intensive Systeme
- (Software-)Architektur
- **Methoden für den Entwurf**
- Architektur im Software-Entwicklungsprozess
- Rolle des Architekten
- Historisches



# Fragestellung: Wie kommt man zu einer Architektur?

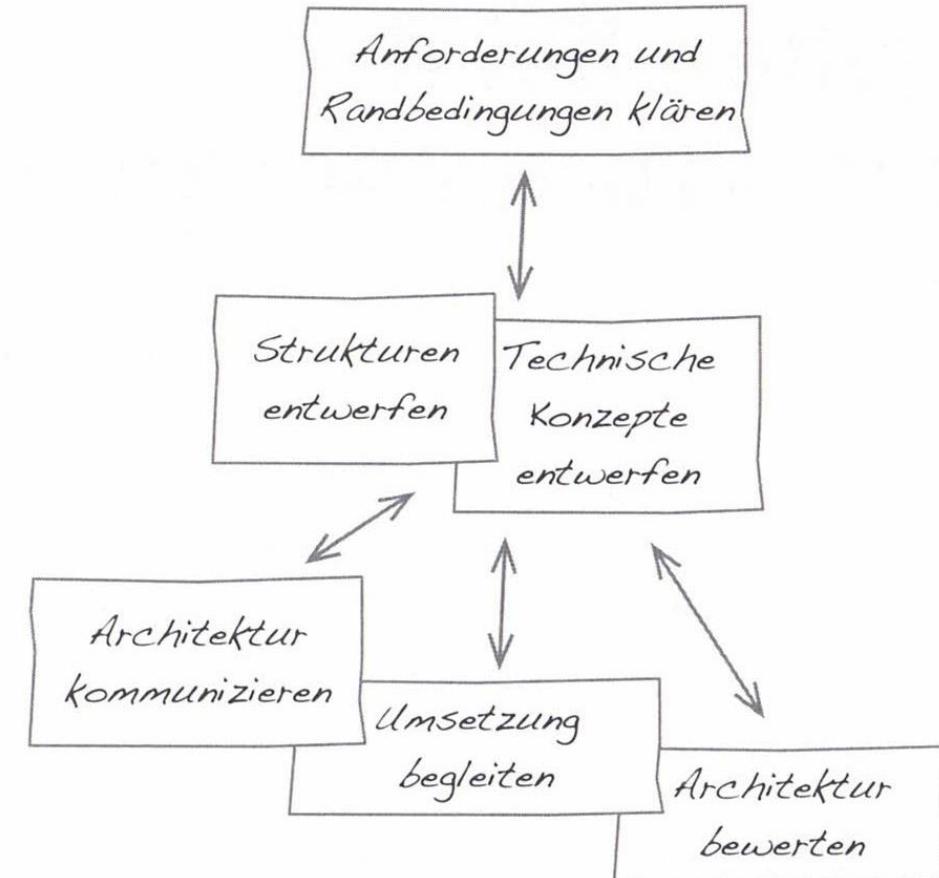


⚠ Es gibt keine silberne Kugel!



# Vorgehen nach Starke

- Verständnis erlangen
  - Richtiges Ansprechpartner finden
  - Anforderungsanalyse
  - Randbedingungen = Leitplanken
  - Risiken ableiten
- Entwurf
  - Strukturen schaffen (Bausteine, Ablauf, Muster, s. ff.)
  - Fachlich beginnen, technisch verfeinern
  - Parallel: technische Konzepte (Persistenz, Oberfläche)
- Bedarfsgerechte, kontinuierliche Kommunikation und Dokumentation
- Rückmeldung der Entwickler berücksichtigen



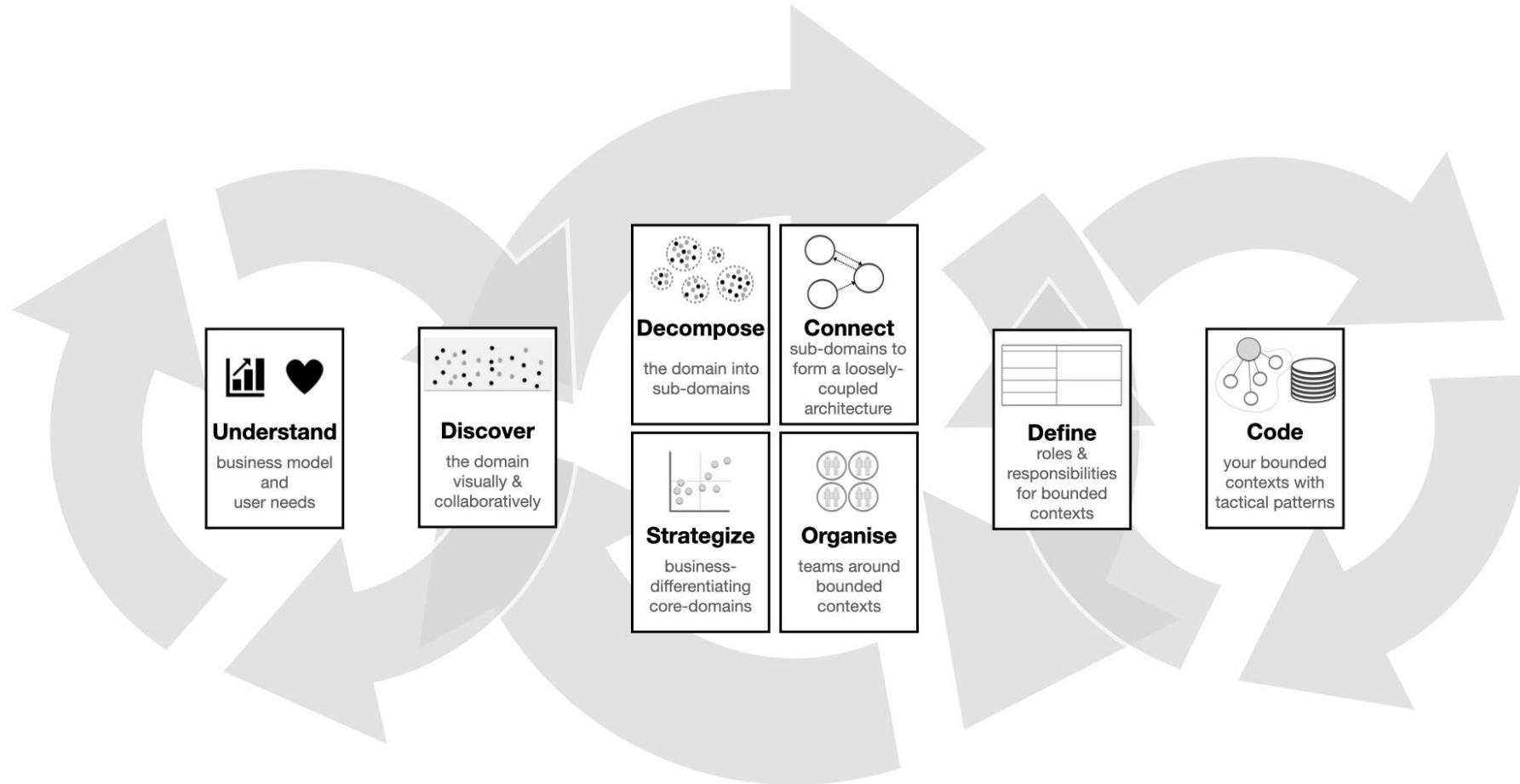
Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke



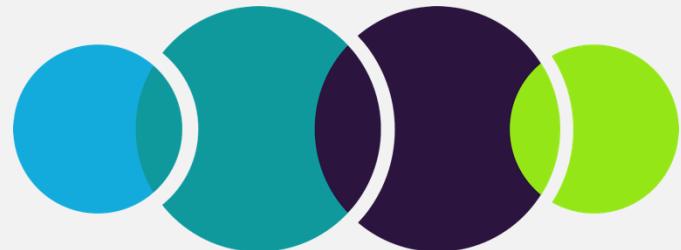
# Vorgehen aus der DDD Community

## Domain-Driven Design Starter Modelling Process

A starter process for beginners, not a rigid best-practice. DDD is continuous, evolutionary, and iterative design.



Quelle: <https://github.com/ddd-crew/ddd-starter-modelling-process>

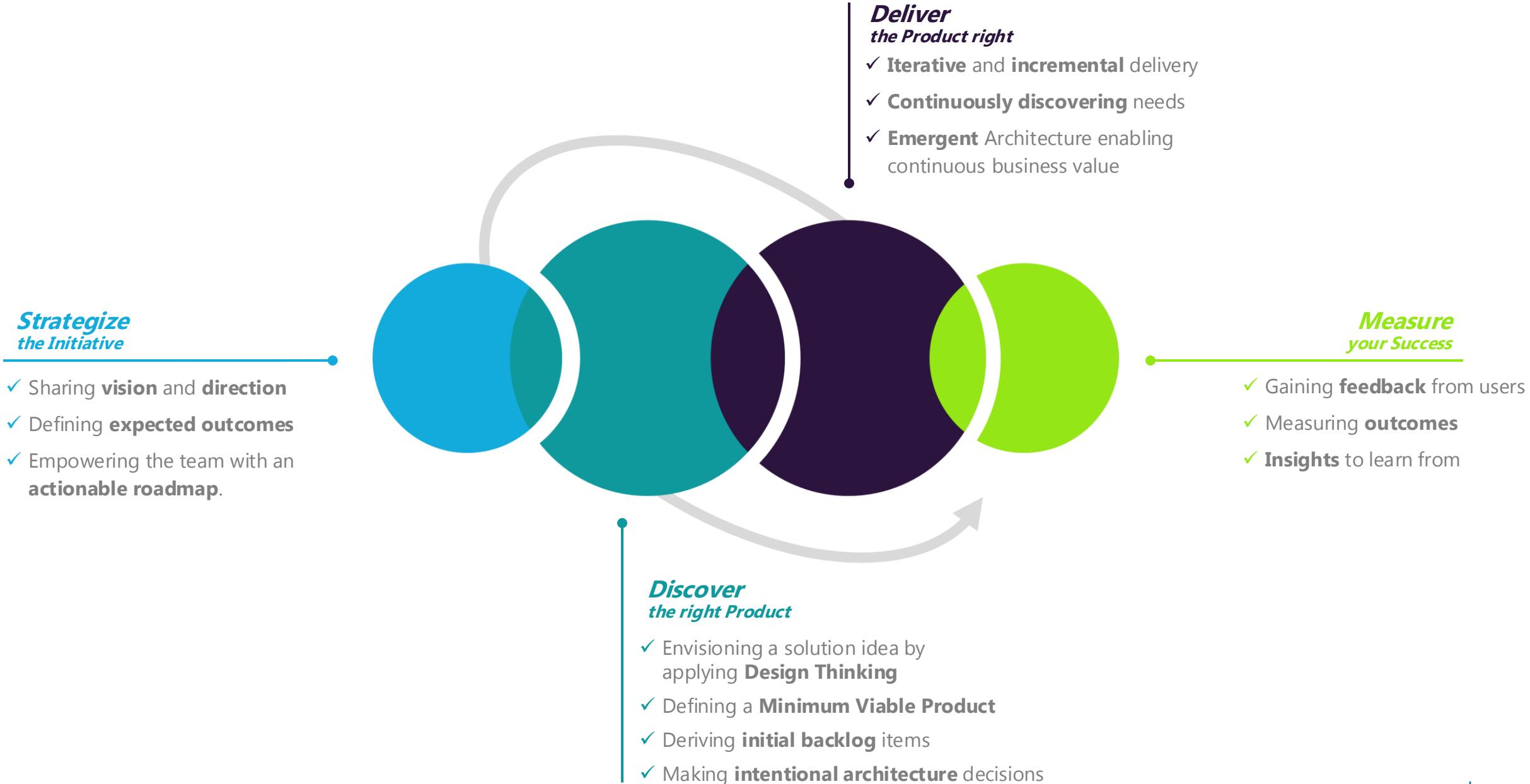


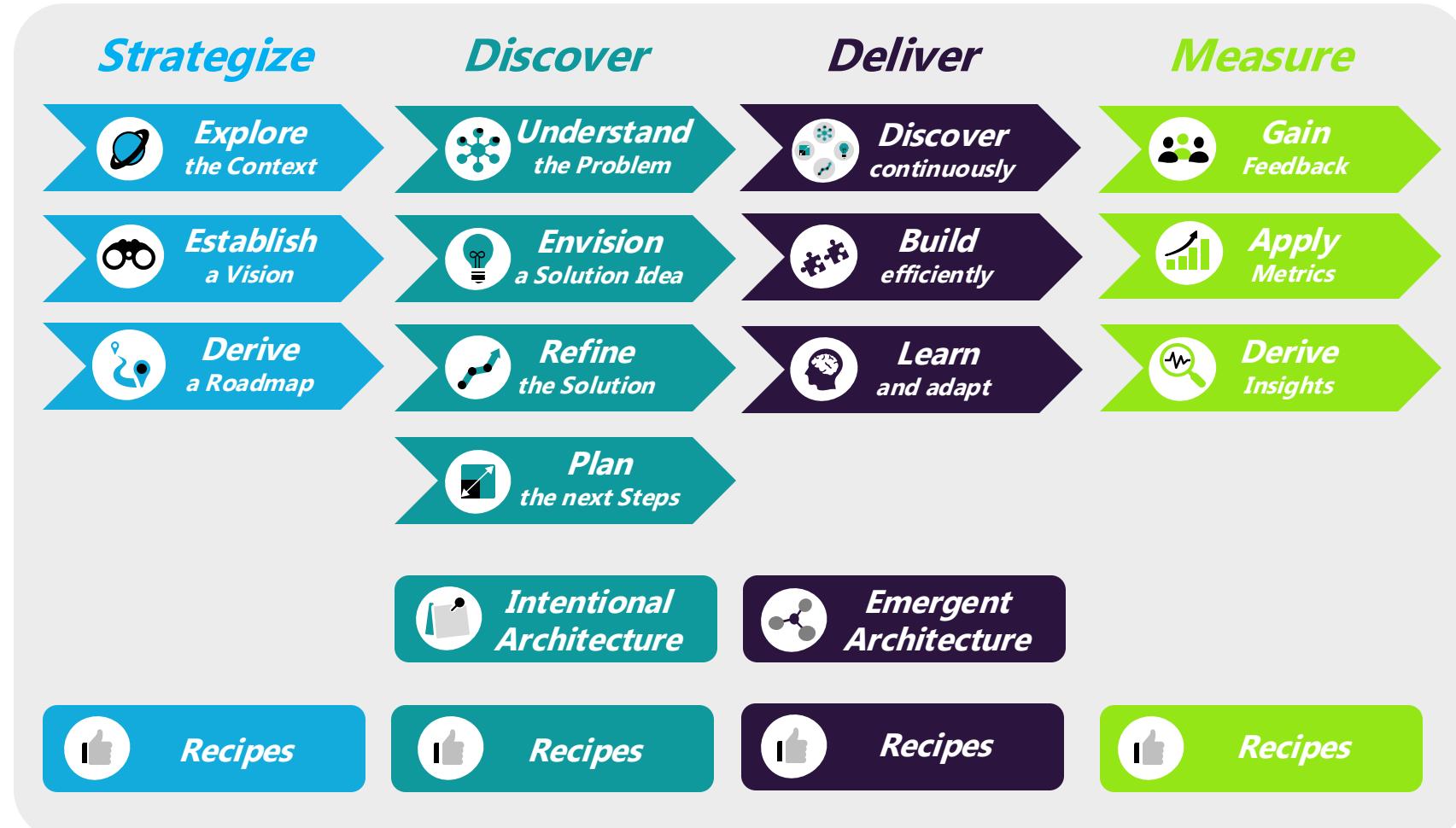
# A·D·A·M

AGILE DESIGN APPLIED METHOD

## **Beispiel-Methodik Baukasten von Capgemini**

- Kein vordefiniertes Vorgehen, aber Anleitungen für die Erstellung eines solchen
- Organisiert in vier Hauptmodule/Bausteine
- End-to-end Betrachtung, aber ohne Technik!
- Fokus auf Fachlichkeit und fachliche Architektur
- Bündelung, Konsolidierung und Aufbereitung von bekannten Methoden aus der Industrie





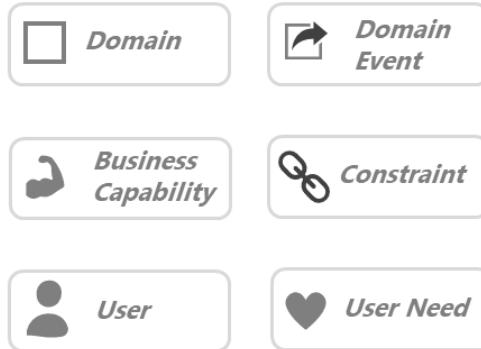
# Content Model



## Product Strategy



## ? Problem Space

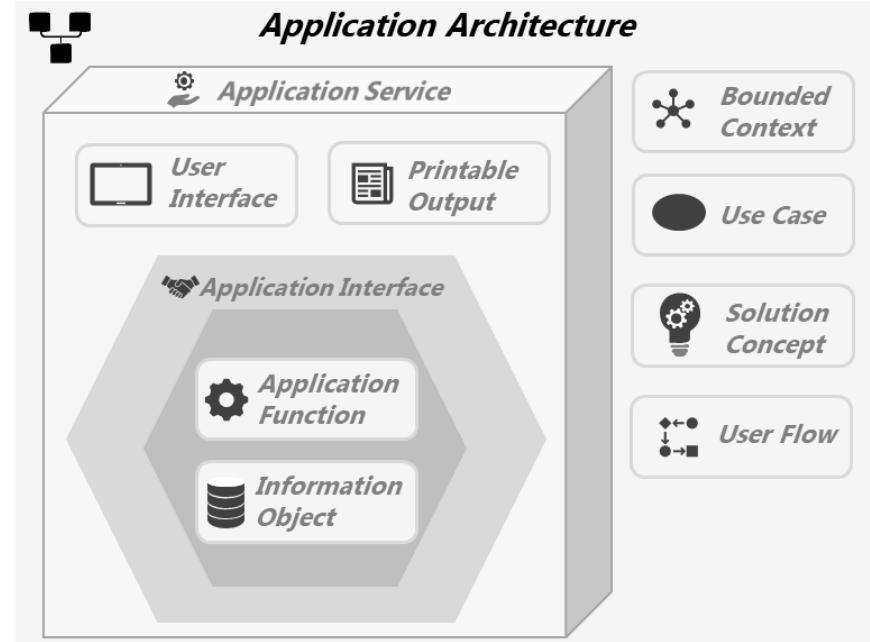


## Solution Space

### Ideas & Concepts

- Solution Prototype
- Solution Sketch
- MVP Definition

### Application Architecture



### Design System

- Bounded Context
- Use Case
- Solution Concept
- User Flow
- Design Language
- Components & Patterns
- Practices

expressed by

## Backlog Items





## Einschub: Modellierung





# Einführung in die Modellierung

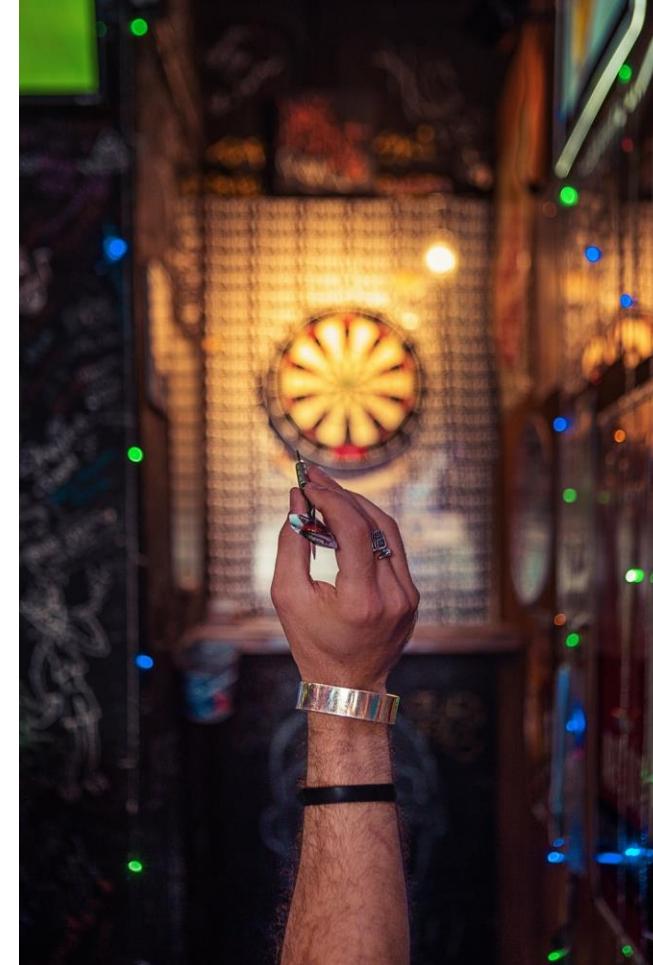
- **Basis:** Modelle + Modellierung
- Rahmenwerk definiert:
  - Eigene Systematik
  - Eigene Techniken
  - Eigene Theorie
- Dient der Visualisierung
- Verfolgt Modelzweck





# Typische Modellzwecke

- Erkenntnis
- Erklärung/Demonstration
- Beherrschung des Originals
- Indikation
- Variation/Optimierung/Reorganisation
- Simulation
- Planung
- Konstruktion
- Verifikation
- Steuerung/Kontrolle
- Ersatzfunktion
- ...





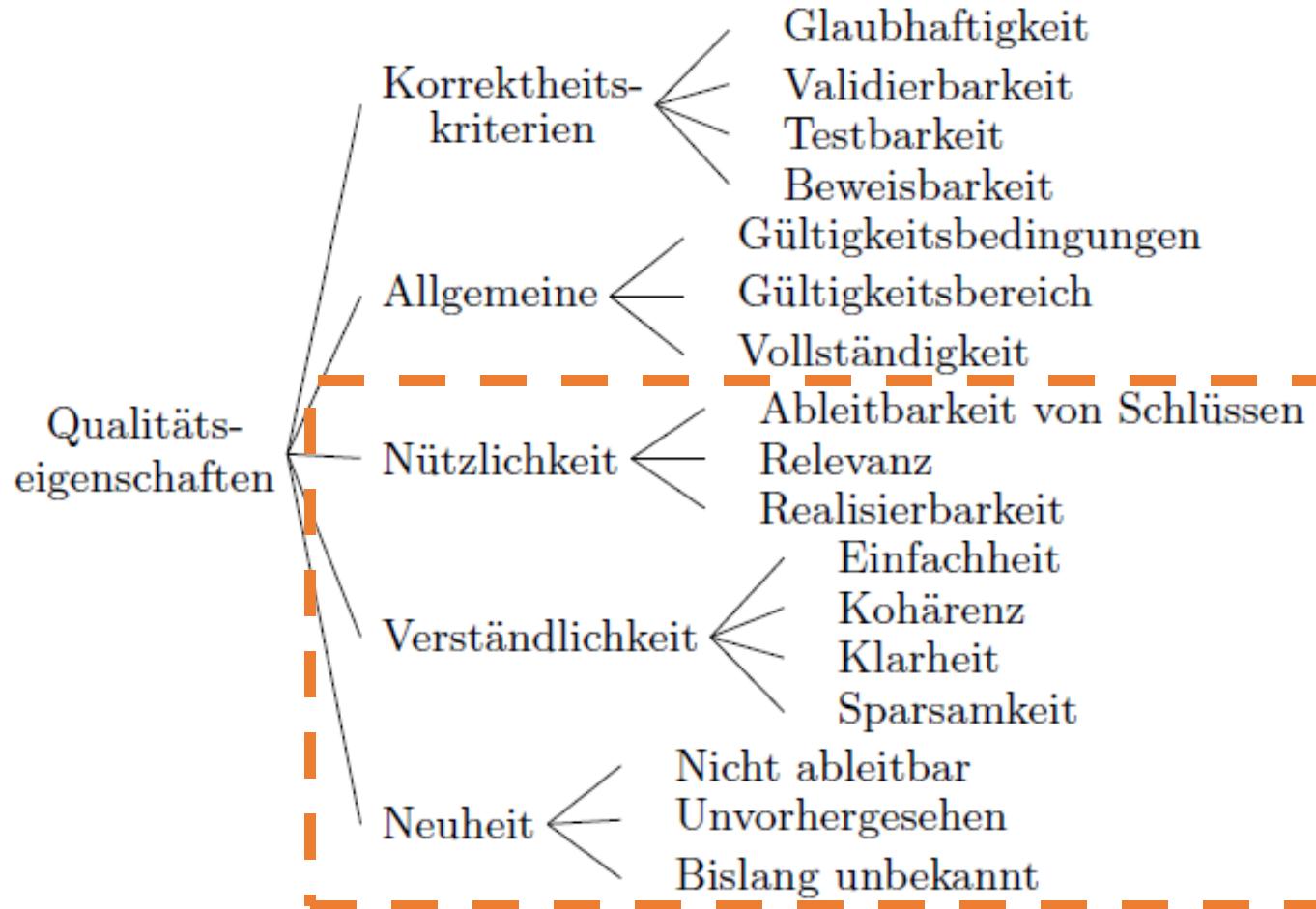
# Typische Verwendung von Modellen

- *deskriptiv* – zur Beschreibung
- *präskriptiv* – als Vorlage
- *konzeptuell* – zum Verständnis und als Vermittler
- *exemplarisch* – Muster / Exemplar (z.B. im Museum)
- *experimentell* – mit einem Prototypen
- *explikativ* – mit einer Erklärung
- *normativ* – mit Annahmen
- *prognostisch* – als Vorhersagemodell
- *metaphorisch* – Bezug zu anderen Semantikfeldern
- *hypothetisch* – als empirisch-belegter Gegenstand
- *substituierend* – als Ersatz für ein Teilsystem
- *gestalterisch* – als Entwurf oder Plan
- *nachprüfend* – als Testfall/-szenario
- *anschaulich* – als visueller Gegenstand
- *repräsentativ* – als Überbringer einer Nachricht oder Kultur
- *anleitend* – als Arbeitsvorschrift
- etc ...





# Qualitätseigenschaften von Modellen





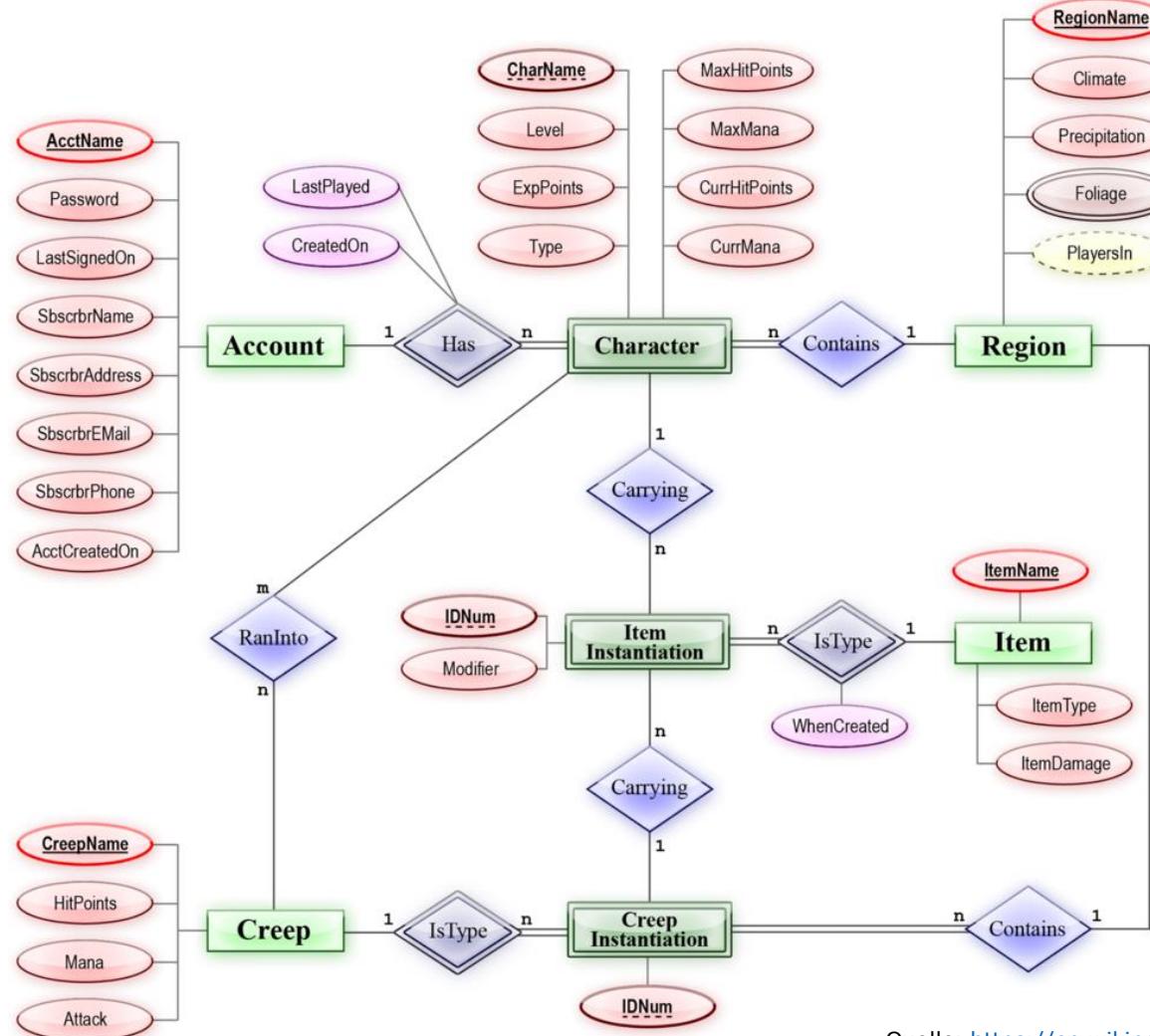
# Bekannte Modellierungssprachen

- **ERM** – Entity-Relationship-Modeling
- **BPMN** – Business Process Model and Notation
- **EPK** – Ereignisgesteuerte Prozesskette
- **UML** – Unified Modeling Language
- etc.





# Entity-Relationship Model



Quelle: [https://en.wikipedia.org/wiki/Entity-relationship\\_model](https://en.wikipedia.org/wiki/Entity-relationship_model)



# Unified Modelling Language

- OMG Standard
- Viele Diagrammarten
  - Verhaltensdiagramme
  - Strukturdiagramme
  - Interaktionsdiagramme
- Viele Tools am Markt
- Wenig bis keine Automatisierung
- Keine definierte Ausrichtung





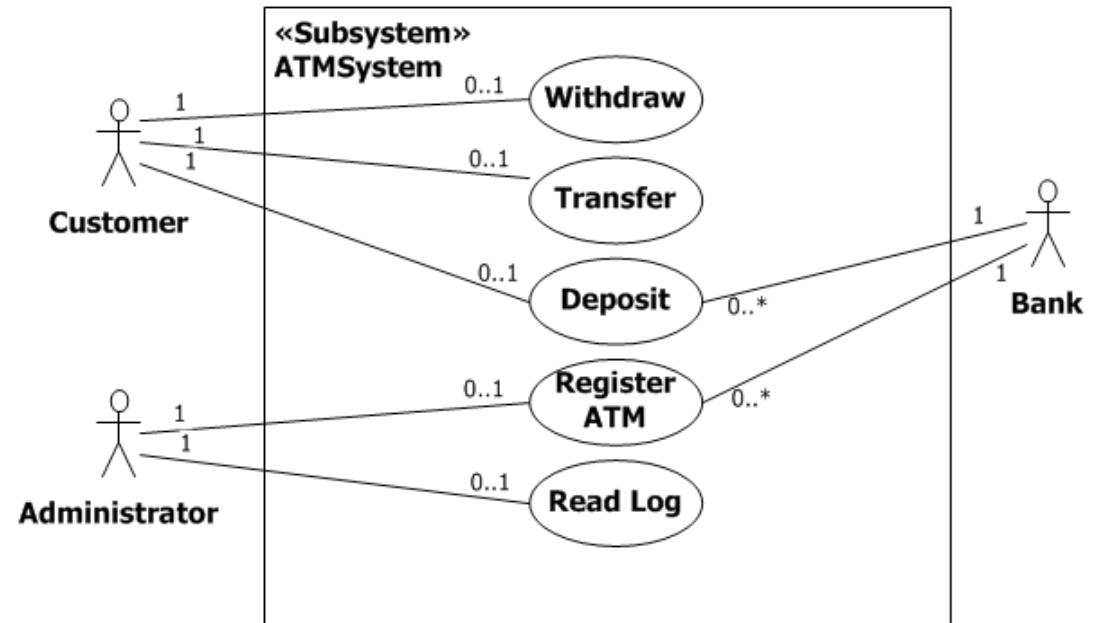
# UML Software

- Online Tools:  
<http://www.umlet.com/>  
<https://online.visual-paradigm.com/>  
<https://www.lucidchart.com/>  
<https://diagrams.net>  
...
- Desktop Tools:  
<https://sparxsystems.com/products/ea>  
<https://www.eclipse.org/papyrus/>  
<http://argouml.tigris.org/>  
Microsoft Visio  
Archi
- IDE Plugins: Diverse



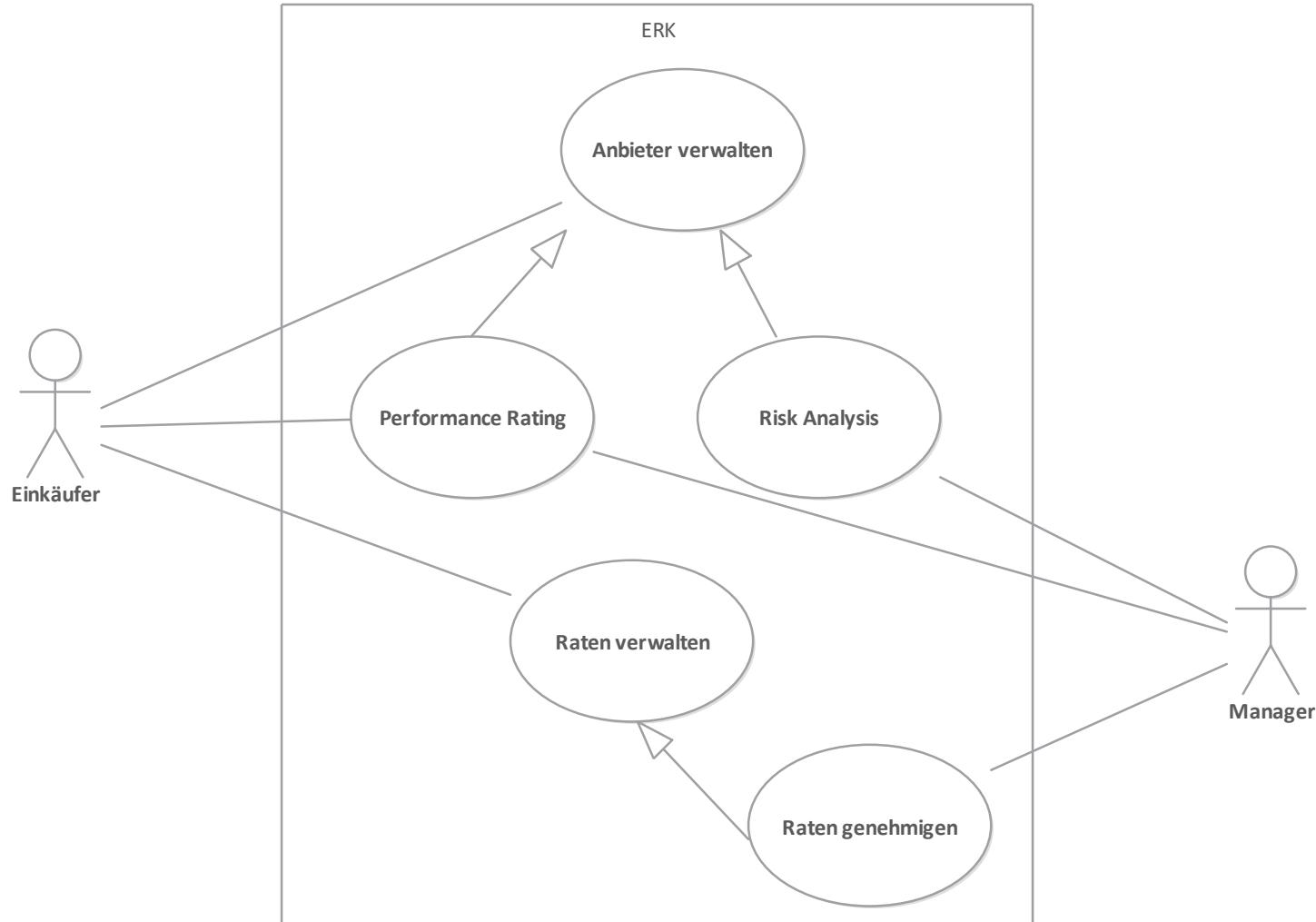
# Verhaltensdiagramme – Use Case Diagramm

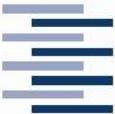
- Benutzerinteraktionen mit dem System
- Anwendungsfall = Leistung eines Systems für einen Anwender
- Enthält:
  - Anwendungsfälle
  - Akteure (Mensch oder System)
  - Beziehungen zwischen den Elementen
- Anwendungsfälle können andere erweitern oder inkludieren





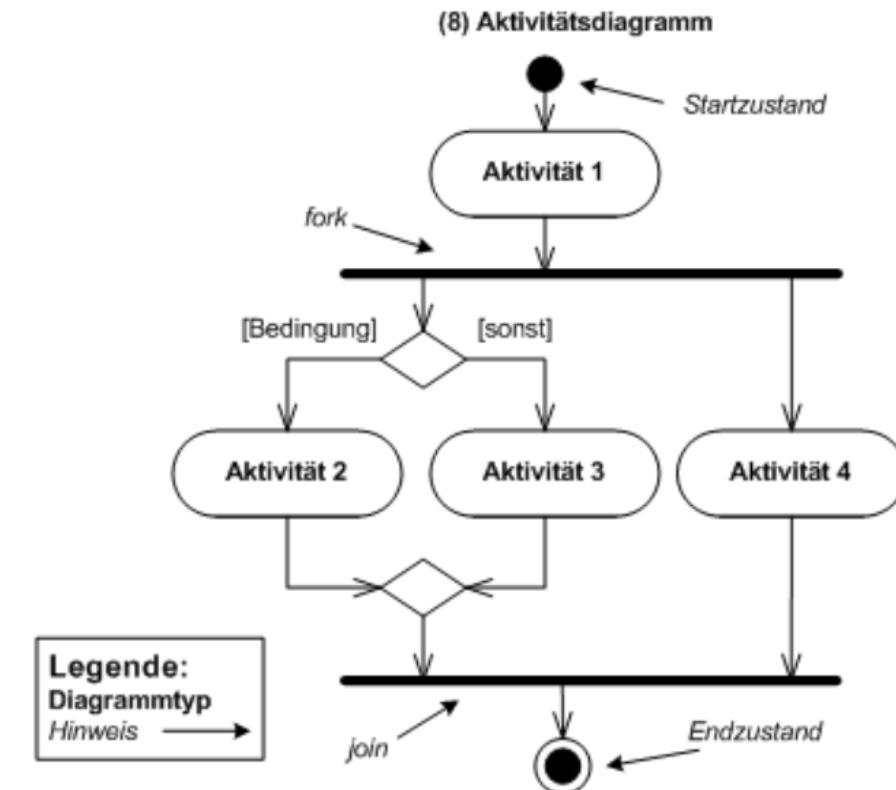
# Beispiel für das ERK System





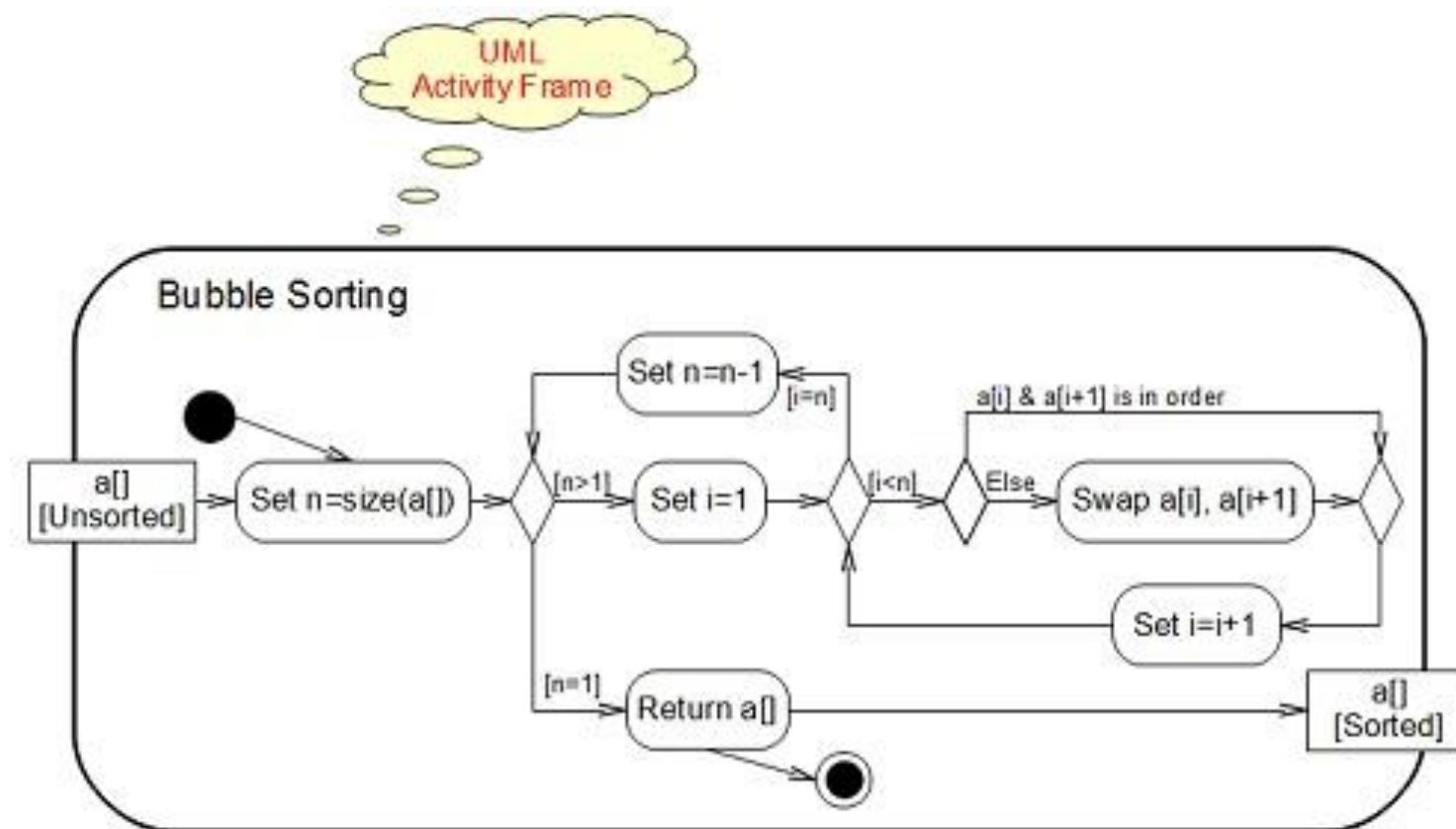
# Verhaltensdiagramme - Aktivitätsdiagramm

- Ablaufverhalten in einem System
- Verhaltensaspekte von mehreren Objekten
- Häufig: Geschäftsprozessbeschreibung
- Parallelisierung und Synchronisation möglich
- Bedingte Verzweigungen über Wächter





# Beispiel: Bubble Sort Algorithmus



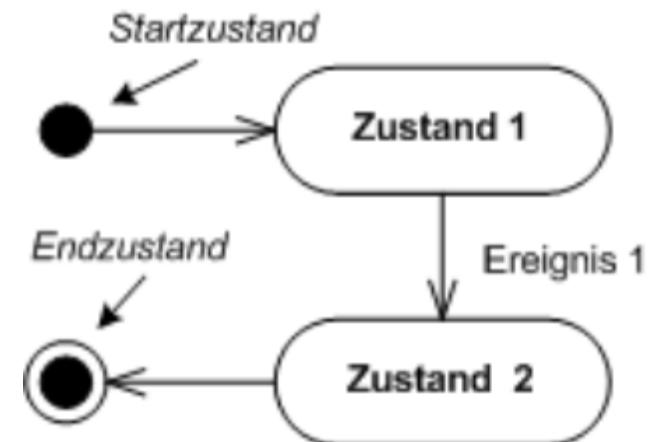
Quelle: <http://www.herongyang.com/UML/Activity-Diagram-Frame-Notation-and-Parameter.htm>



# Verhaltensdiagramme - Zustandsautomat

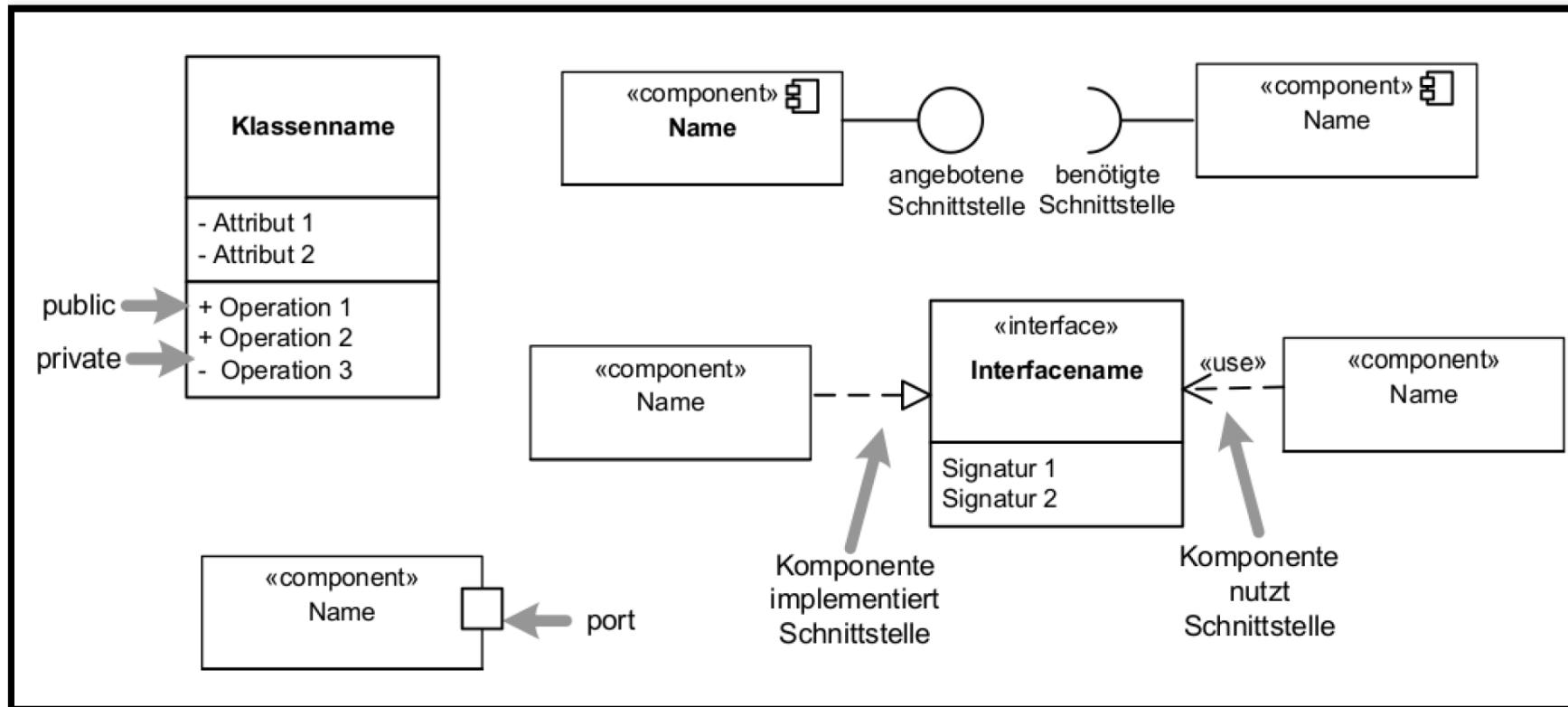
- Ereignisse und Zustände eines Objekts
- Transition als Wechsel von Zuständen
- Ereignisse = Auslöser
- Sonderzustände für Start und Ende

(9) Zustandsautomat





# Klassen und Schnittstellen in UML

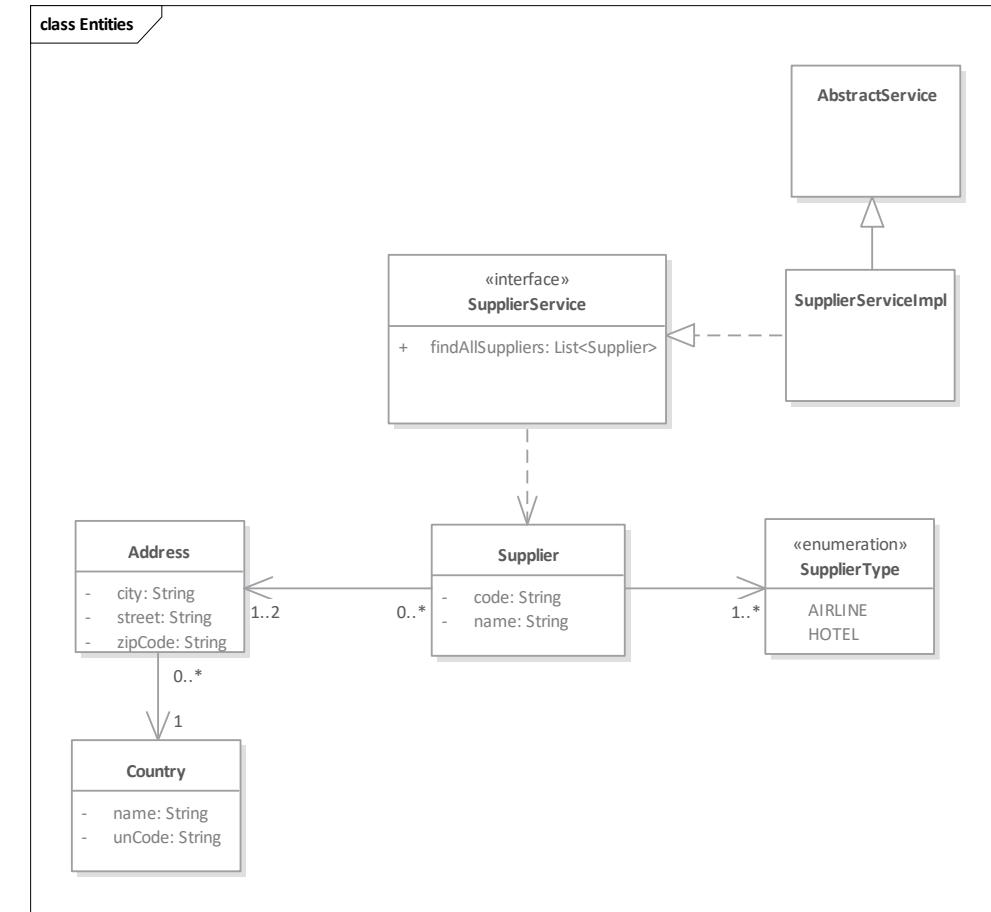


Bildquelle: "Effektive Softwarearchitekturen" von Gernot Starke



# Struktur-Diagramme - Klassendiagramm

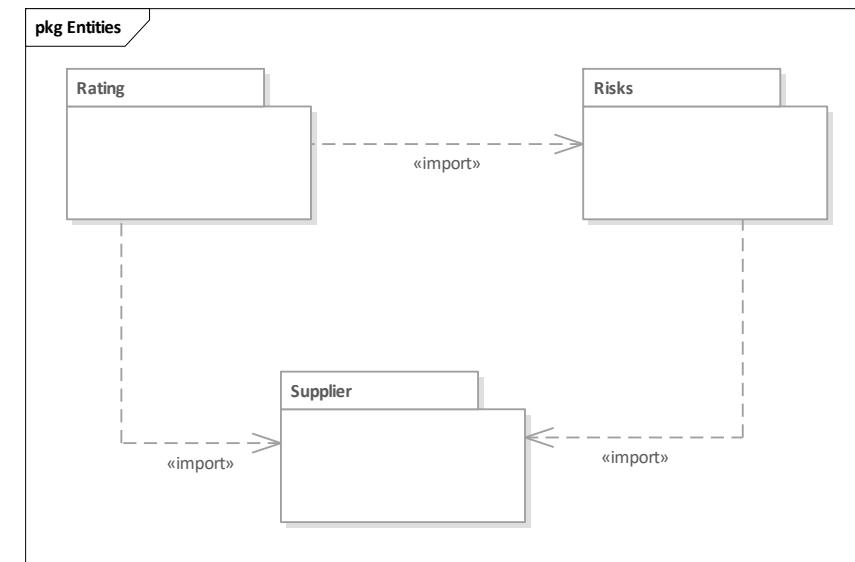
- Struktur der Klassen eines Systems
- Enthält:
  - Klassen mit Attributen und Methoden
  - Beziehungen zwischen den Klassen:
    - Assoziationen mit Kardinalitäten
    - Vererbung
    - Implementierung
    - Abhängigkeiten





# Strukturdiagramme - Paketdiagramm

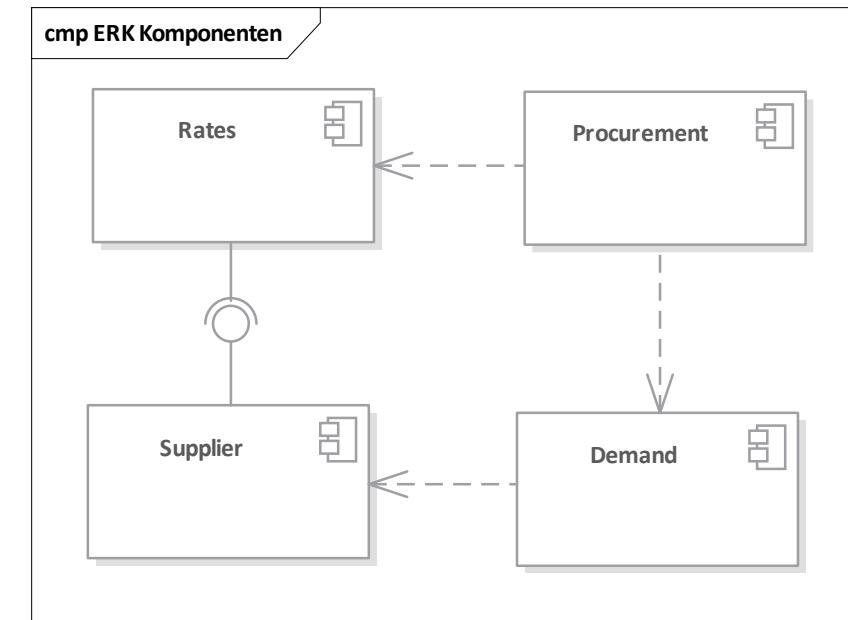
- Beschreibt die Pakete eines Systems
- Enthält:
  - Pakete
  - Beziehungen zwischen Paketen





# Strukturdiagramme – Komponentendiagramm

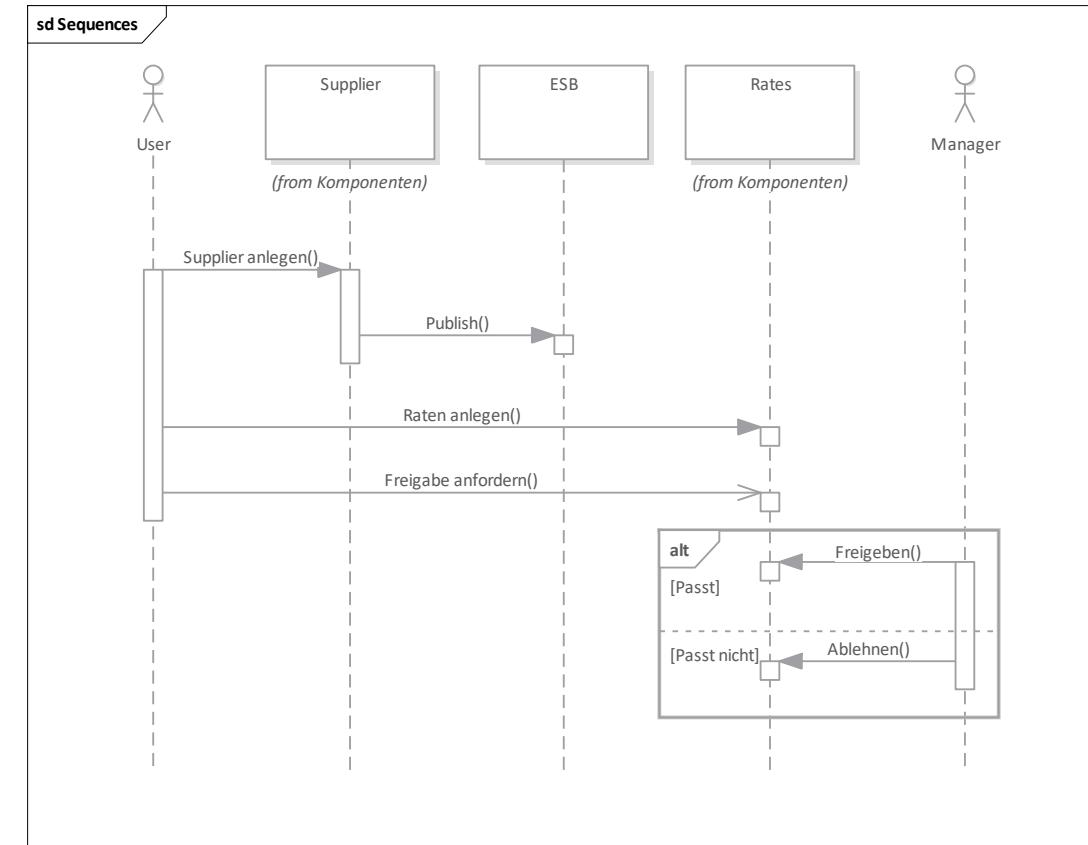
- Beschreibt die Komponenten eines Systems
- Komponenten können alles sein:
  - Bausteine
  - Schichten
  - Deployment Units
  - Systeme
- Definiert Abhängigkeiten





# Interaktionsdiagramme - Sequenzdiagramm

- Beschreibt Ausschnitt eines Lebenszyklus von Elementen
- Ablauf wird durch Aufrufe dargestellt
- Unterscheidung Asynchron/Synchron
- Fragmente zur Strukturierung

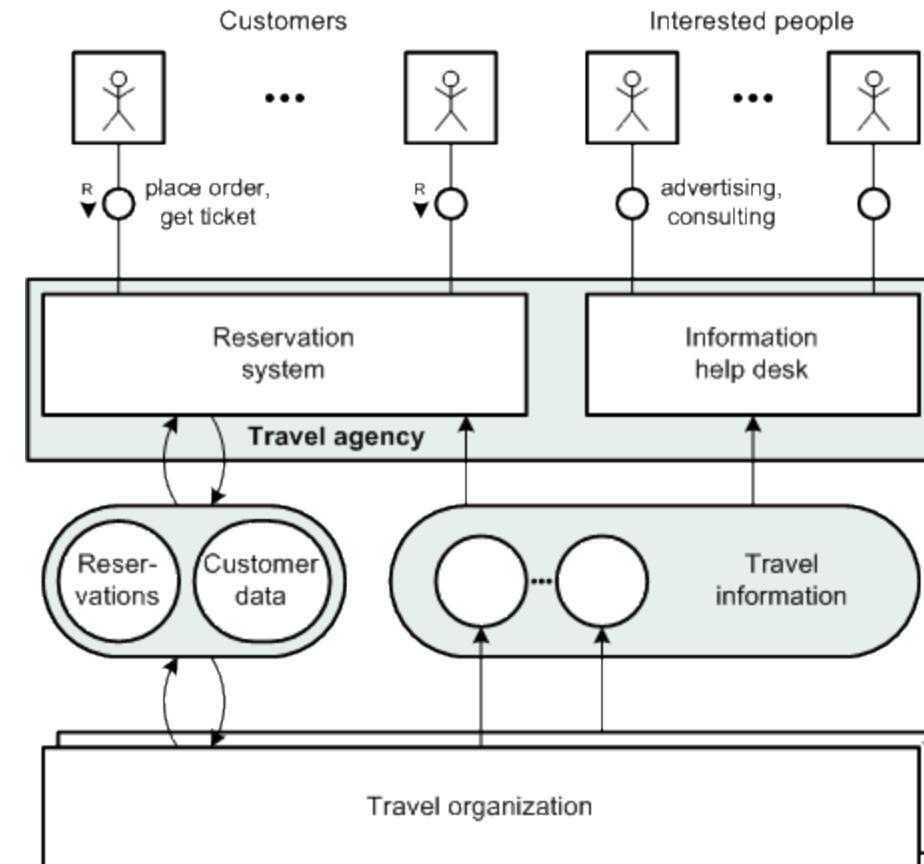




# Fundamental Modelling Concepts (FMC)

- Beliebt im SAP Umfeld
- Getrieben von Hasso Plattner im Hasso Plattner Institut
- Einfache Darstellung
- Leicht verständlich

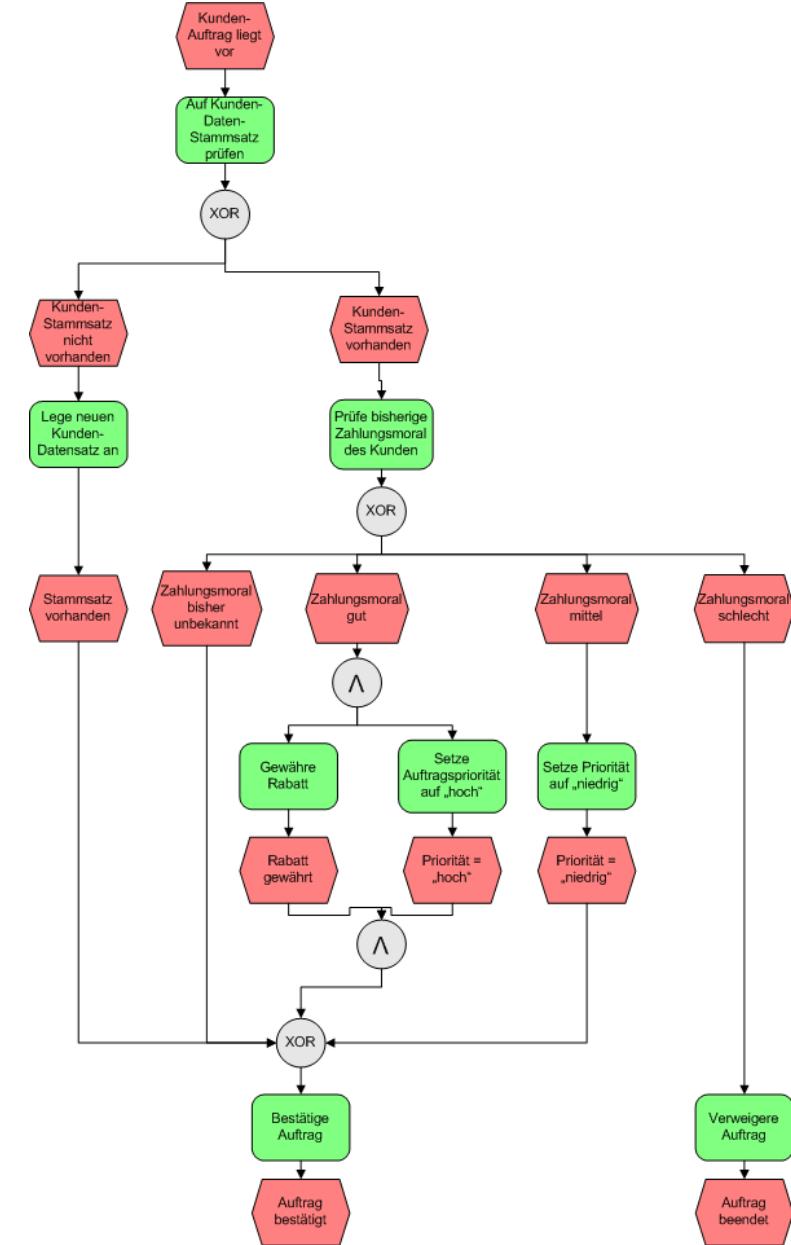
<http://www.fmc-modeling.org/>





# Geschäftsprozessmodellierung

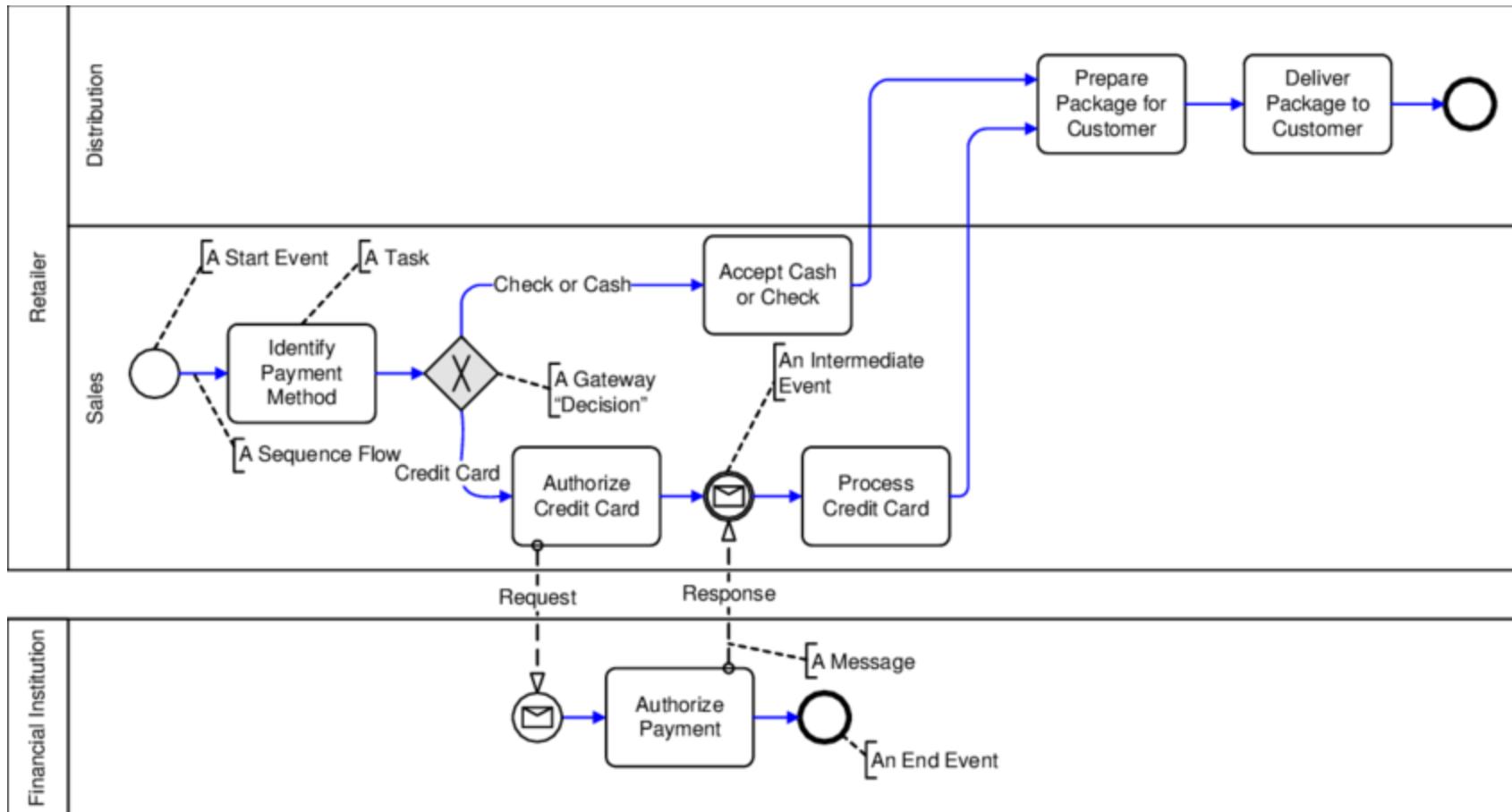
- Historisch: Ereignis Prozess Ketten (SAP)
- Aktueller Standard: Business Process Modelling Notation
- Neu: Case Management Modelling Notation
  - Besser geeignet für nicht-strikt sequentielle Prozesse
  - Ermöglicht viel mehr Flexibilität
  - Kann BPMN Prozessen inkludieren



Quelle: <https://www.der-wirtschaftingenieur.de/index.php/ereignisgesteuerte-prozesskette-epk/>



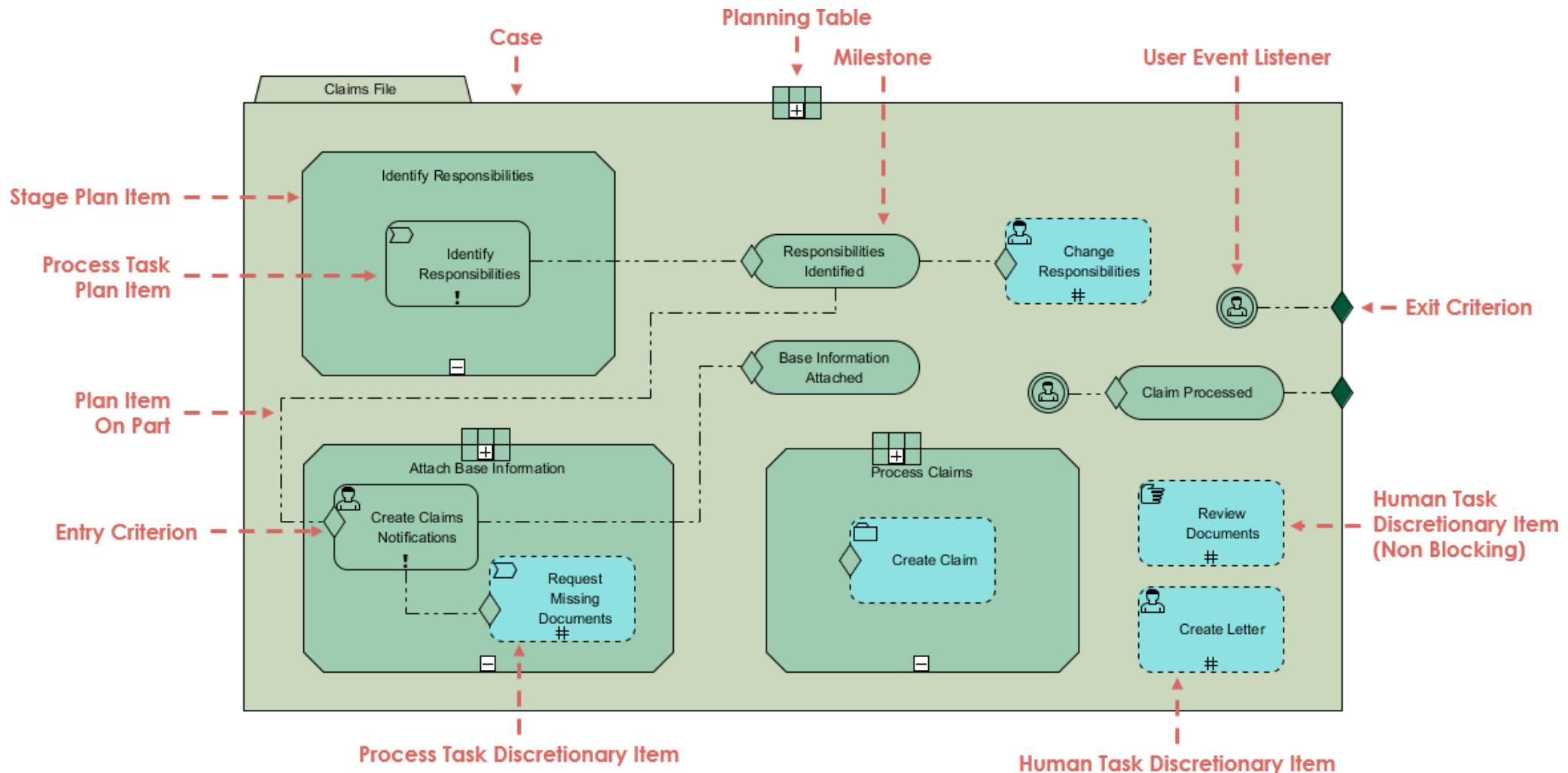
# Business Process Modelling Notation



Quelle: [https://www.researchgate.net/figure/BPMN-example-Payment-process\\_fig1\\_27478001](https://www.researchgate.net/figure/BPMN-example-Payment-process_fig1_27478001)



# Case Management Modelling Notation



Quelle: <https://www.visual-paradigm.com/guide/cmmn/cmmn-example/>



# Agenda

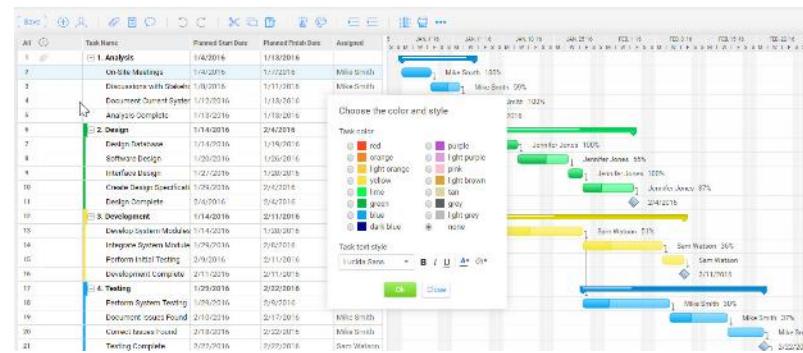
- Software-intensive Systeme
- (Software-)Architektur
- Methoden für den Entwurf
- **Architektur im Software-Entwicklungsprozess**
- Rolle des Architekten
- Historisches



# Übersicht über Vorgehens- und Projektmodelle

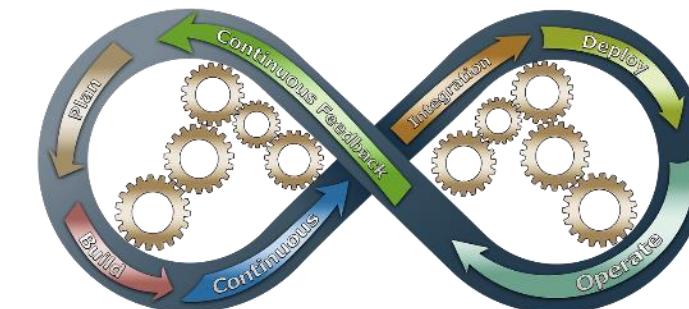
## Klassische / Traditionelle Modelle

- Wasserfallmodell
- Iteratives Wasserfallmodell
- V-Modell
- Rational Unified Process



## Agile Modelle

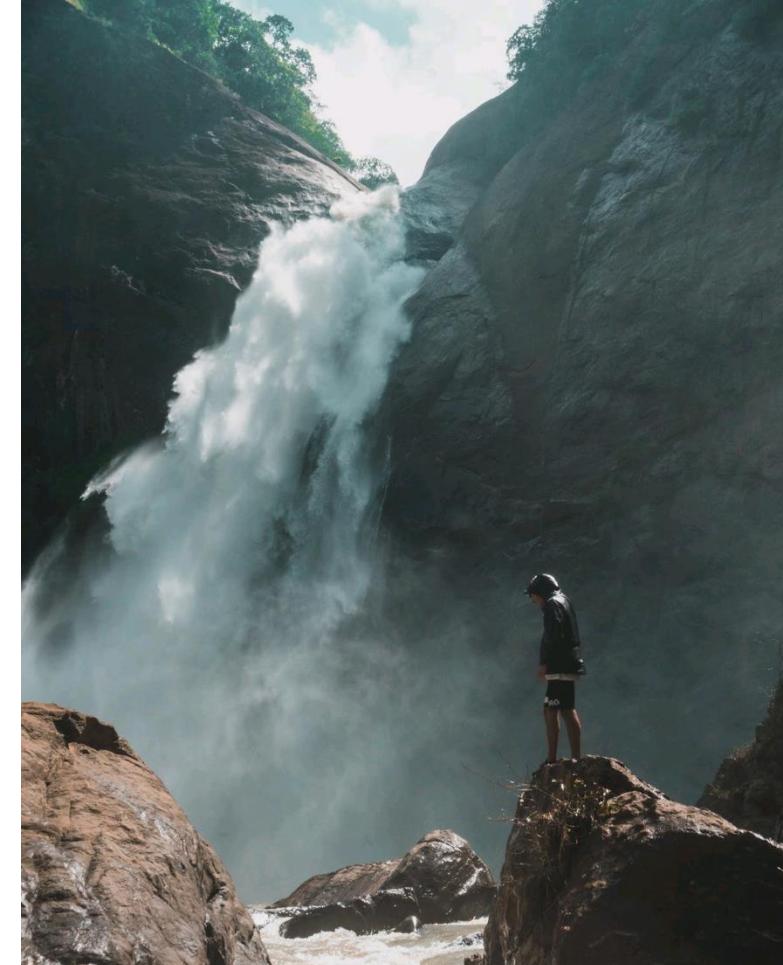
- Lean
- Extreme Programming
- Kanban
- SCRUM





# Architektur im klassischen Projekt

- **Basis:** Requirements Engineering
  - Vollständig
  - konsistent
- **Input:** Spezifikation
- **Vorgehen:**
  - Entwurf in verschiedenen Sichten
  - Vom Grob- zum Fein-Entwurf
- **Output:**
  - Architektsichten
  - Designentscheidungen
  - Arbeitspakete für die Projektplanung
  - Richtlinien
  - Risiken



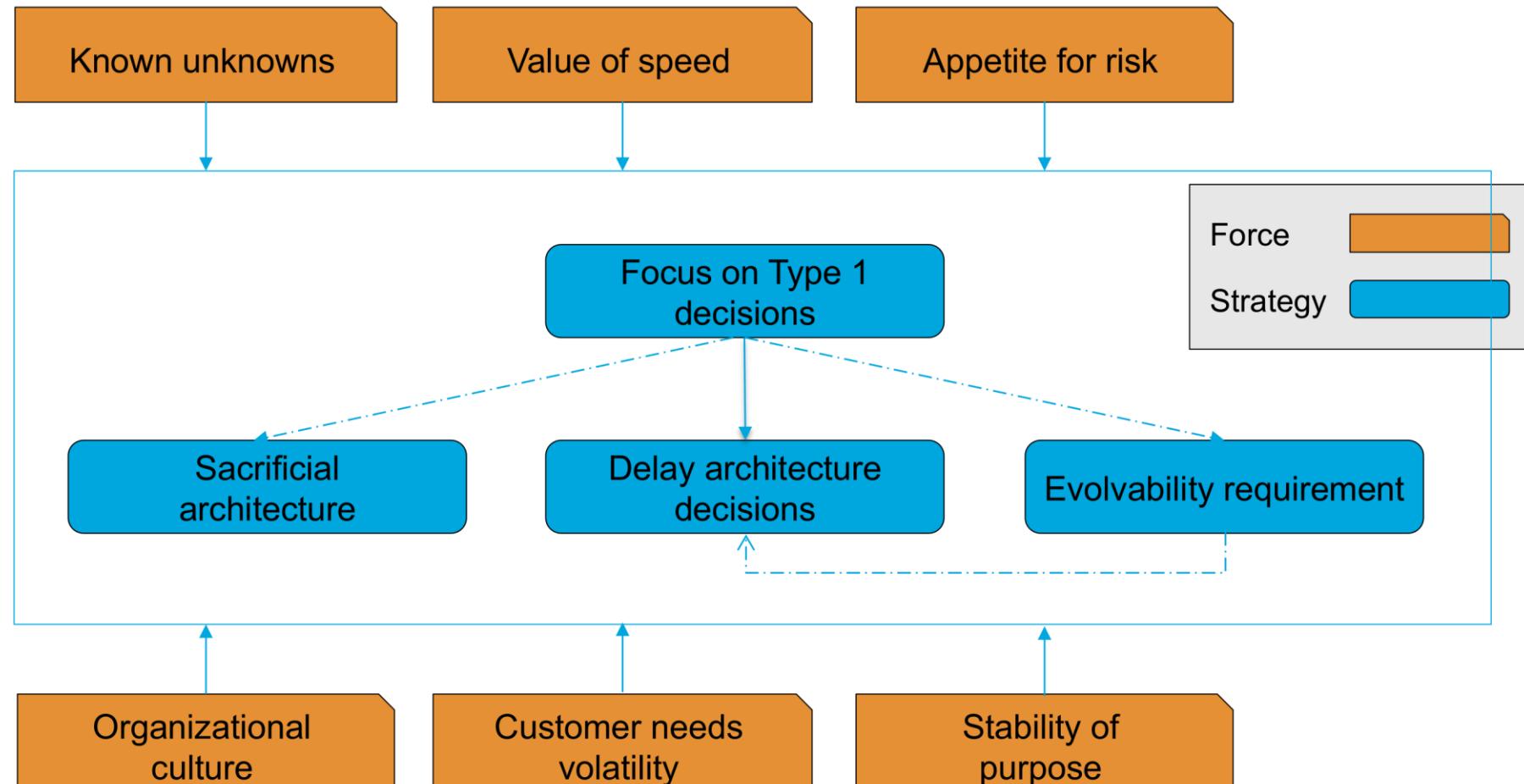


# Architektur im agilen Projekt

- **Frage:** Gibt es Architektur?
- Kein Big Design Up Front (BDUF)!
  - Änderung der Anforderungen
  - Keine Garantie für Umsetzbarkeit
  - Time-to-market
- Gibt es trotzdem Architektur?
  - Nicht garantiert
  - Aber: „Architecture happens“
  - Jedoch: kein Architekt
  - Evolutionäre Architektur
- **Vorgehen:**
  - Passende Architektur pro Inkrement
  - Trade-off: Goldener Henkel vs. Refactoring
  - Minimum Viable Architecture
- Und was macht man bei agilen Großprojekten?

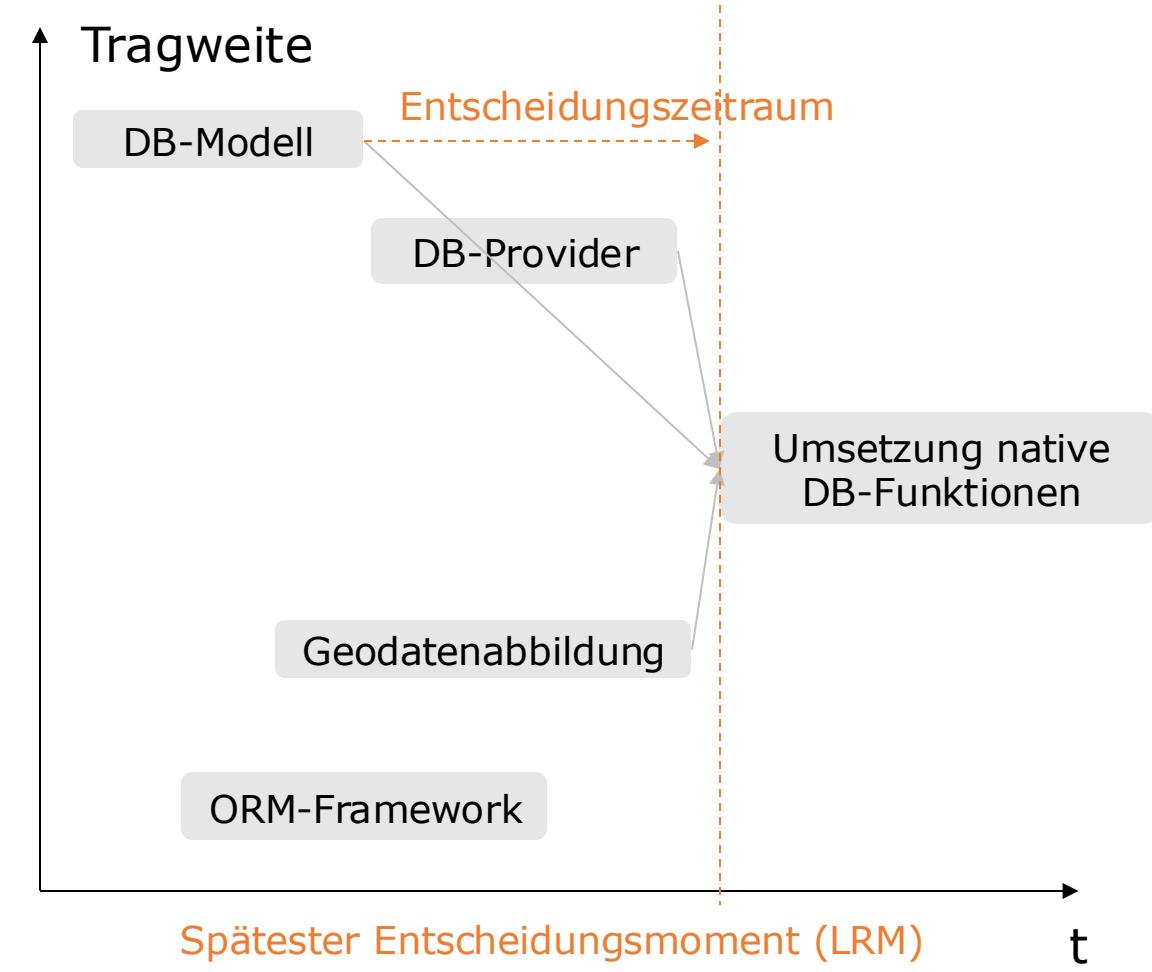
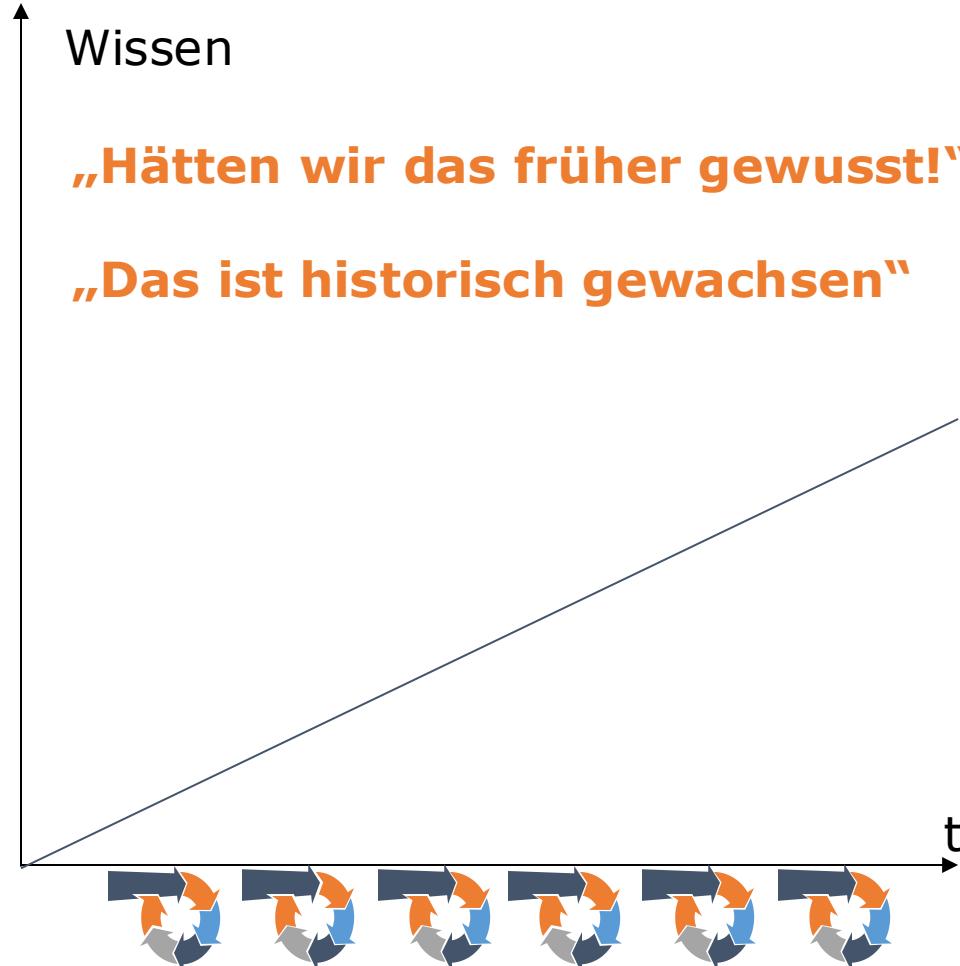


# MVA nach dem Agile Architecture Framework von der Open Group



Quelle: [https://pubs.opengroup.org/architecture/o-aaf/snapshot/Agile\\_Architecture\\_Framework.html#minimum-viable-architecture](https://pubs.opengroup.org/architecture/o-aaf/snapshot/Agile_Architecture_Framework.html#minimum-viable-architecture)

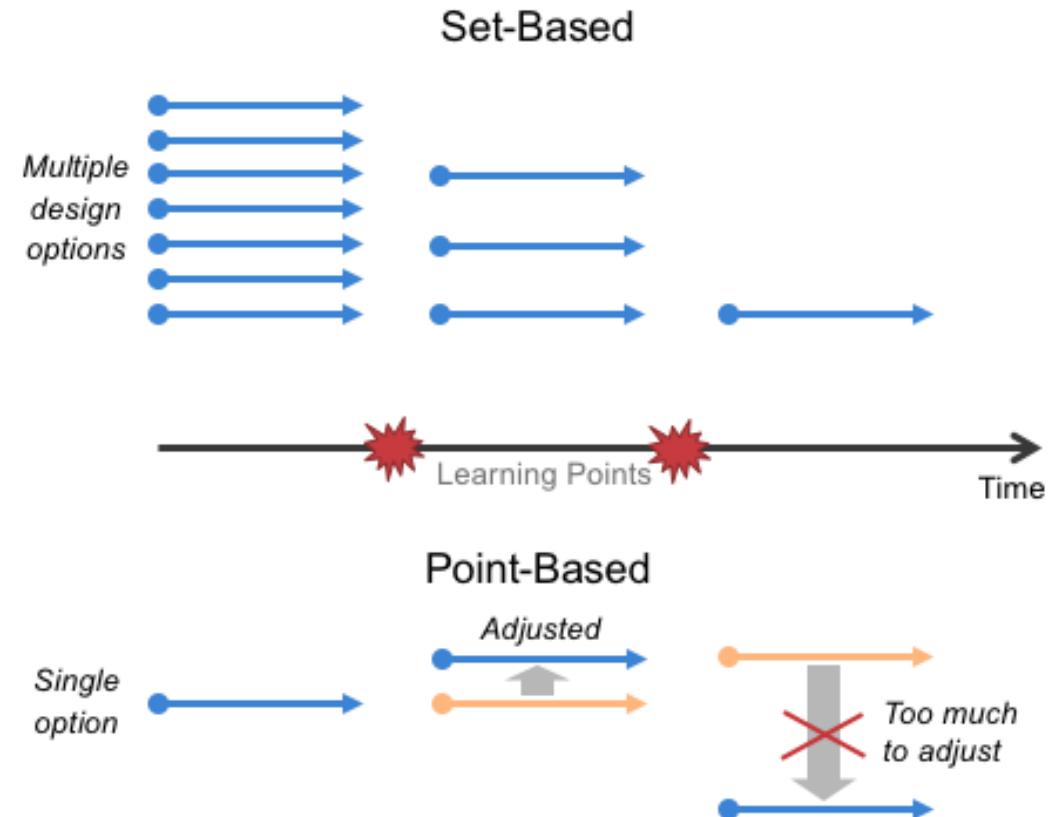
# Beispiel für Vorgehen beim agilen Ansatz: Last Reasonable Moment



# Ähnlich aber etwas anders: Set-based Concurrent Engineering (SBCE)



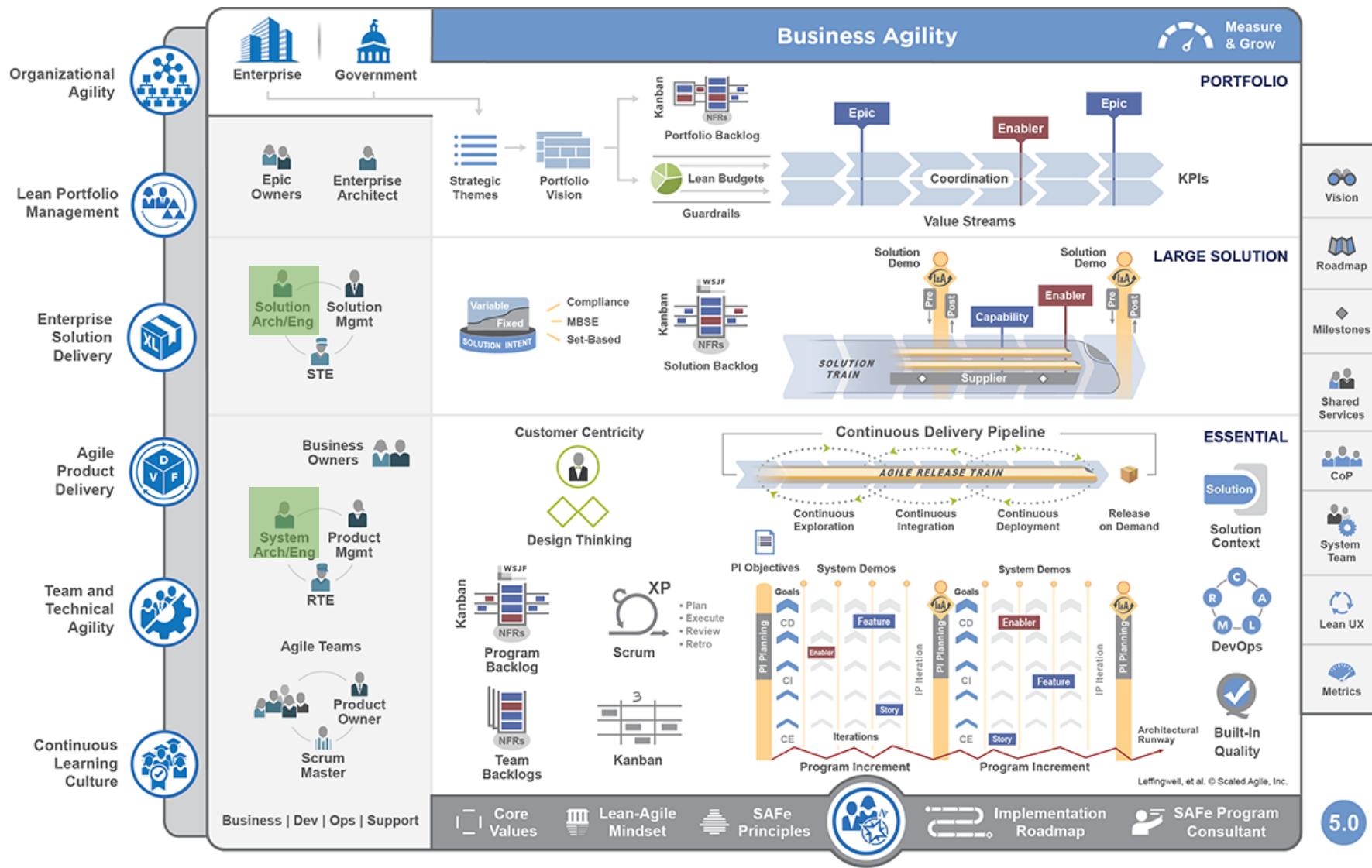
- Mehrere Optionen werden parallel evaluiert
- Findet auch bei klassischen Methoden statt, meist aber nicht in der Tiefe
- Kann in verschiedenen Teams / Subsystemen stattfinden



© Scaled Agile, Inc.



# Architektur im Scaling Agile Framework



Lean-Agile Leadership



# Agenda

- Software-intensive Systeme
- (Software-)Architektur
- Methoden für den Entwurf
- Architektur im Software-Entwicklungsprozess

## ▪ **Rolle des Architekten**

- Historisches



# Verantwortlichkeiten und Aufgaben eines Architekten

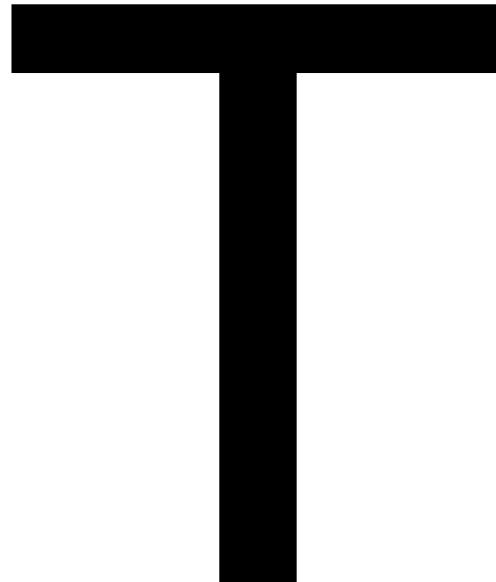
- Analyse und Definition der Qualitäts-Anforderungen
- Entwurf und Konstruktion von Optionen
- Treffen von Entscheidungen
- Erfüllung der Qualitätsanforderungen
- Dokumentation
- Beratung
- Kommunikation
- Risiken im Auge behalten
- Wirtschaftlichkeit der Gesamtlösung
- Bewertung der Architektur





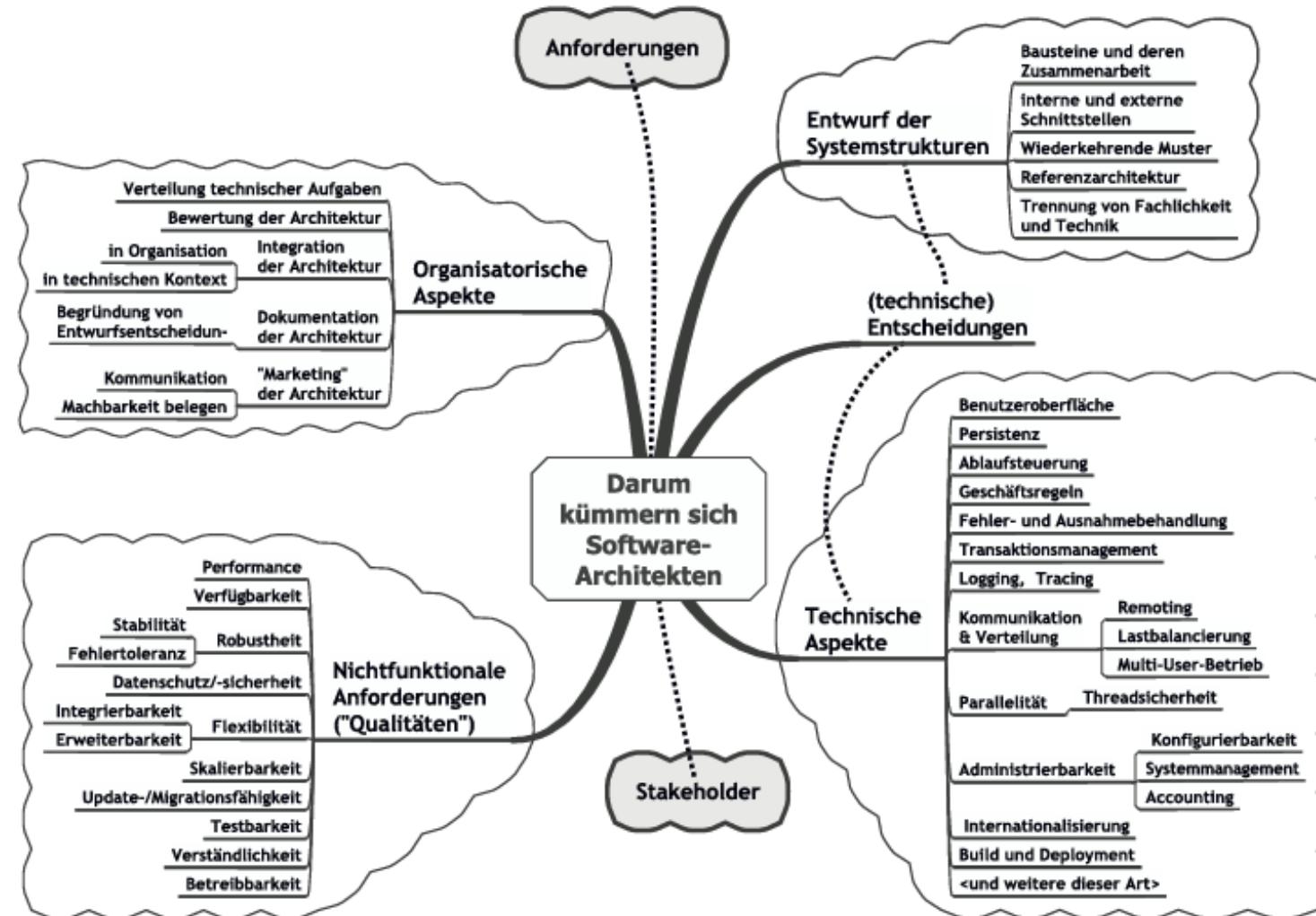
# Fähigkeiten eines Architekten

- Breites Technologiewissen
- Analytisches Denken
- Abstraktionsfähigkeit
- Kritisches Denken
- Kreativität
- Erfahrung
- Systematische Methodik
- Kommunikation und Social Skills



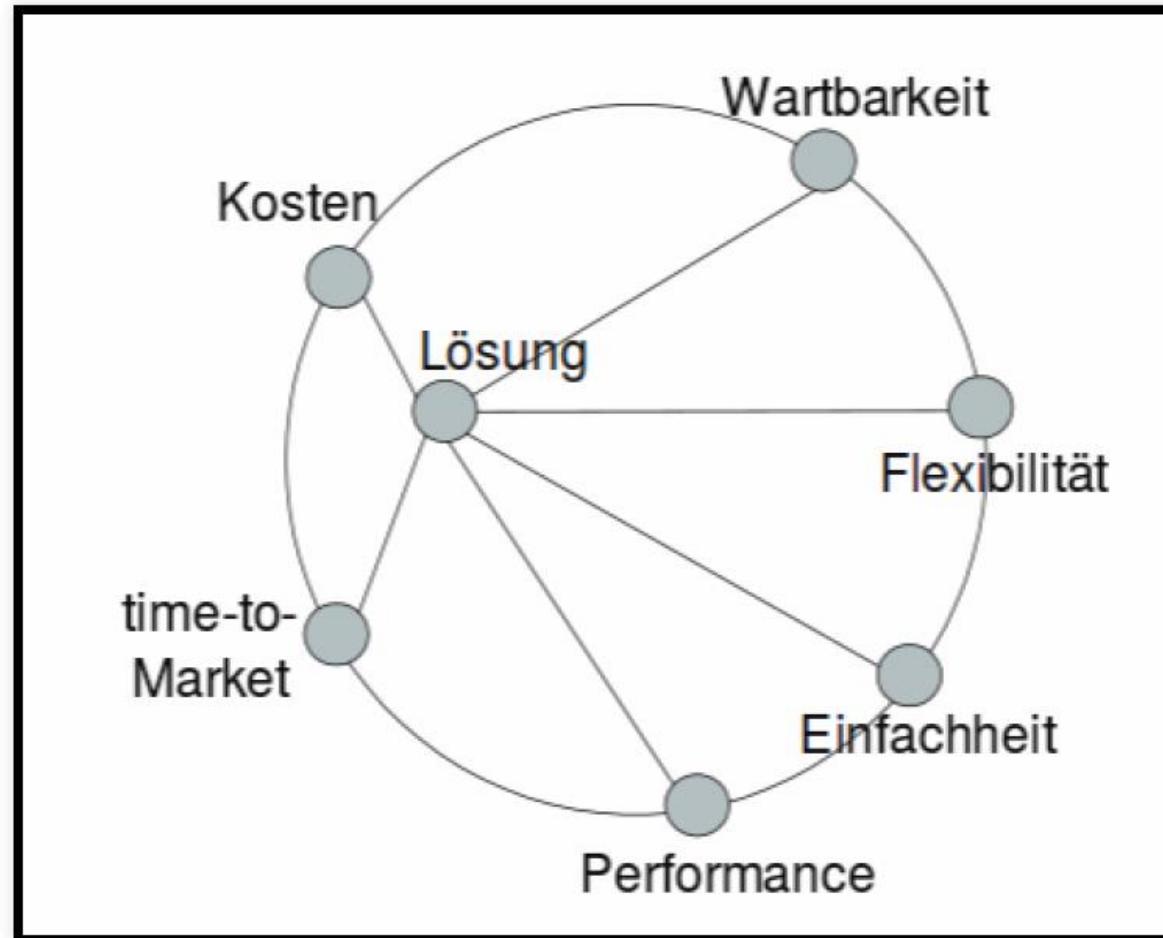


# Aufgaben eines Architekten nach Starke



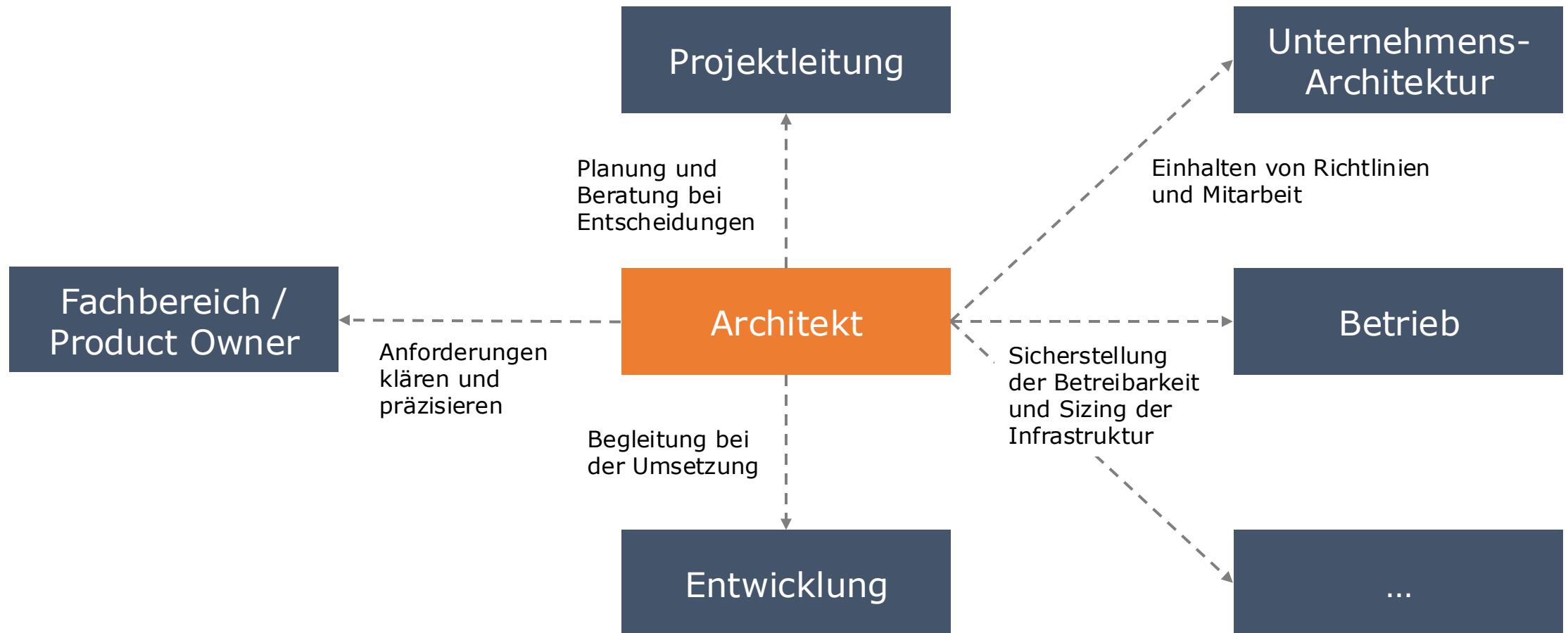


# Der Architekt als Diplomat und Akrobat



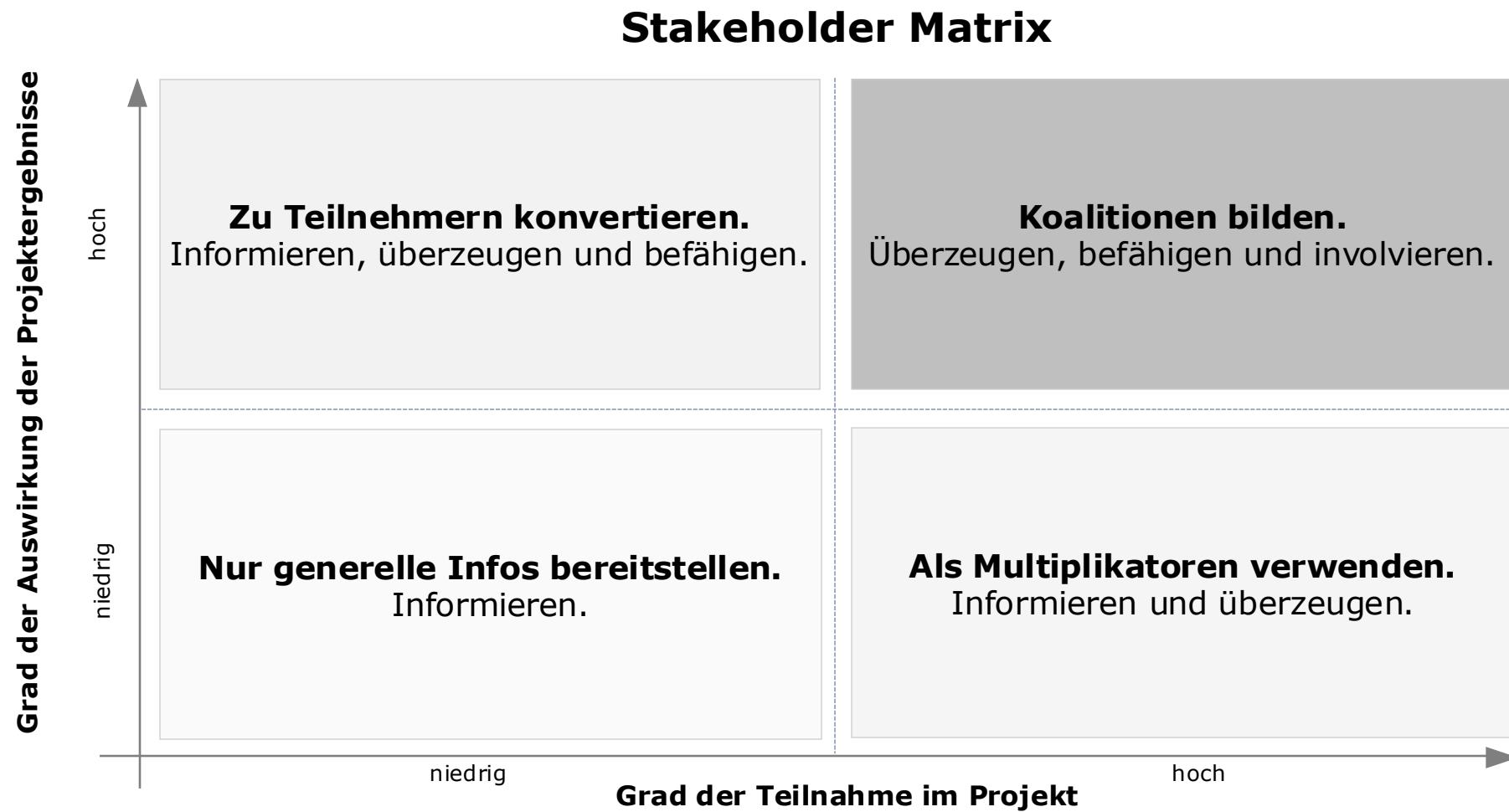


# Beziehungen zu Stakeholdern in einem Software Projekt





# Stakeholder Management 101





# Die Rolle des Architekten im Projekt

## Klassisches Projekt

- Verantwortung für Architektur und Entscheider
- Bei kleineren Projekten: einzelner Architekt
- Bei größeren Projekten: Führung eines Teams von Architekten
- Regeltermine mit dem Projektleiter
- Definition von Regeln und Richtlinien
- Übergreifende QA auf die Architektur-Artefakte
- Schnitt der Arbeitspakete
- Entwurf der Arbeitspakete

## Agiles Projekt

- Mitarbeit im Team ("Coding Architect")
- Berater des Teams
  - Teilnahme bei den Meetings
  - Coaching der Entwickler
  - Sparringspartner
  - Risiken aufzeigen
  - Bewertung von Optionen
- Enabler des Teams
  - Methoden
  - NFR Dashboard
  - Vermittlung von Wissen

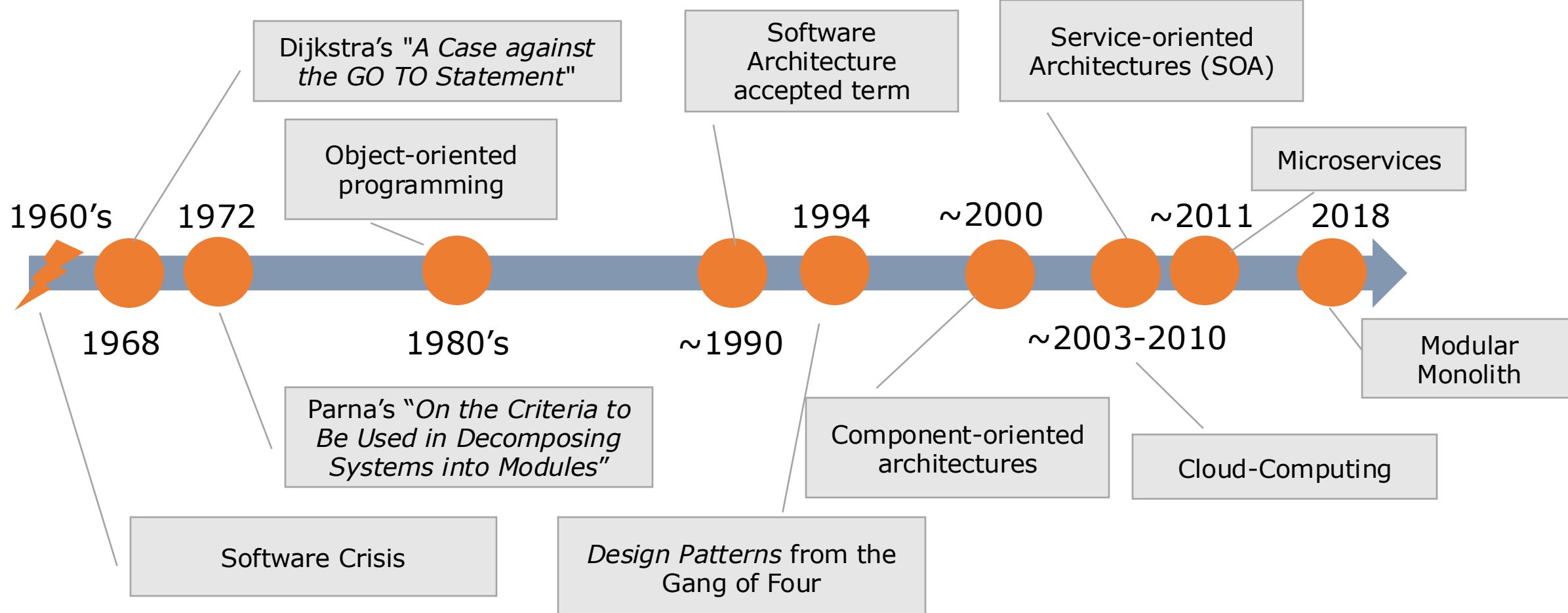


# Agenda

- Software-intensive Systeme
- (Software-)Architektur
- Methoden für den Entwurf
- Architektur im Software-Entwicklungsprozess
- Rolle des Architekten
- **Historisches**



# Eine kleine Historie zur Software Architektur





# Zusammenfassung

- Architektur zur Bewältigung der Komplexität
- Beruht auf Anforderungen
- Verschiedene Definitionen, häufig aber:
  - Komponenten/Bausteine
  - Abhängigkeiten
  - Schnittstellen
- Rolle des Architekten als Akrobat und Diplomat
- Unterschied zwischen Architektur im agilen und klassischen Projekt
- Überblick über den Entwurf
- (Wichtige Zeitpunkt in der Historie)

