

Q&A: QUALITÄT UND ARCHITEKTUR

JUG Saxony Day 2022

Dirk Mahler

BUSCHMAIS GbR

▲ Wer wir sind

- ▲ Dresdner IT-Beratungsunternehmen, gegründet im Jahre 2008

▲ Unsere Schwerpunkte

- ▲ Architekturberatung und Entwicklung moderner Geschäftsanwendungen
- ▲ Software-Qualitätsanalysen und -sicherung

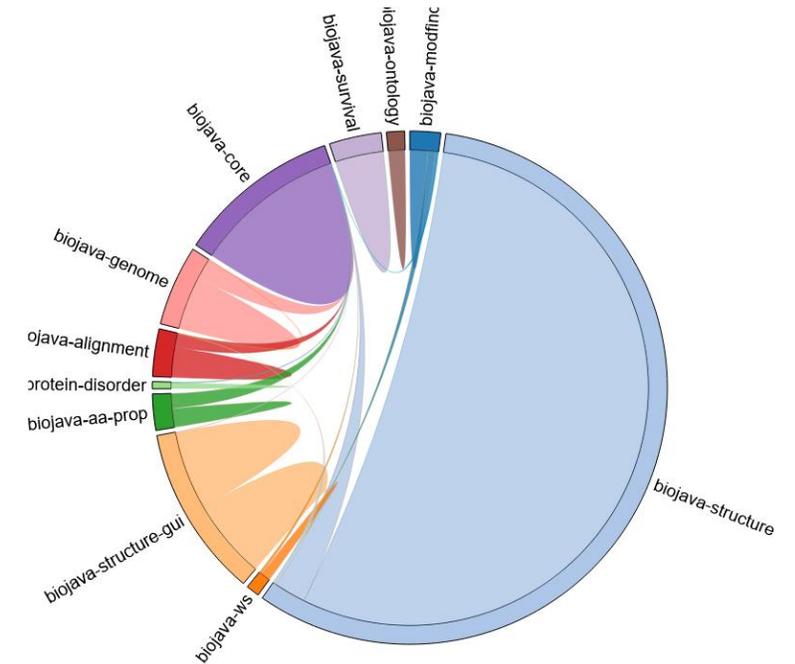
▲ Unsere Kunden

- ▲ Auswahl: ITZBund, Sächsische Aufbaubank-Förderbank, ASML, GlobalFoundries, Thyssenkrupp Steel, Deutsche Telekom, COOP Schweiz

SOFTWARE-METRIKEN

- ▲ Übersichten über fachliche und technische Strukturen des Systems
- ▲ Aussagen über Kopplung, Kohäsion, Kapselung, Komplexität uvm. von der Methoden- bis zur System-Ebene
- ▲ Einfache Identifizierbarkeit von Hot Spots im Gesamtkontext

BUSCHMAIS



Q&A: „Q“

- ▲ Wer von Euch beschäftigt sich mit Software-Qualität?
- ▲ Was versteht Ihr unter Software-Qualität?

Q&A: „A“

- ▲ Wer von Euch beschäftigt sich mit Software-Architektur?
- ▲ Was versteht Ihr unter Software-Architektur?

Q&A: „Q&A“

- ▲ Welchen Zusammenhang seht Ihr zwischen Software-Qualität und Architektur?

QUALITÄT, ARCHITEKTUR & ENTSCHEIDUNGEN



QUALITÄTSMERKMALE

^ Functional Suitability

- ^ Functional Completeness
- ^ Functional Correctness
- ^ Functional Appropriateness

^ Reliability

- ^ Maturity
- ^ Availability
- ^ Fault Tolerance
- ^ Recoverability

^ Performance Efficiency

- ^ Time Behavior
- ^ Resource Utilization
- ^ Capacity

^ Usability

- ^ Appropriateness Recognizability
- ^ Learnability
- ^ Operability
- ^ User Error Protection
- ^ User Interface Aesthetics
- ^ Accessibility

QUALITÄTSMERKMALE

^ Security

- ^ Confidentiality
- ^ Integrity
- ^ Non-repudiation
- ^ Accountability
- ^ Authenticity

^ Compatibility

- ^ Co-existence
- ^ Interoperability

^ Maintainability

- ^ Modularity
- ^ Reusability
- ^ Analysability
- ^ Modifiability
- ^ Testability

^ Portability

- ^ Adaptability
- ^ Installability
- ^ Replaceability

QUALITÄTSMERKMALE

▲ Abhängigkeiten und Überschneidungen

▲ Availability & Recoverability

▲ Mean Time To Recovery (MTTR)

▲ Functional Correctness, Operability, Modularity, Modifiability & Testability

▲ Fehlerdiagnose und -behebung

▲ User Interface Aesthetics & Time Behavior

▲ Response-Zeit (UI)

QUALITÄTSMERKMALE

- △ Konflikte & Trade-Offs
 - △ Time Behavior vs. Resource Efficiency
 - △ Replikation & Caching
 - △ Availability vs. Functional Correctness
 - △ Eventual Consistency (CAP)
 - △ Time Behavior vs. Interoperability
 - △ Binär-Protokolle

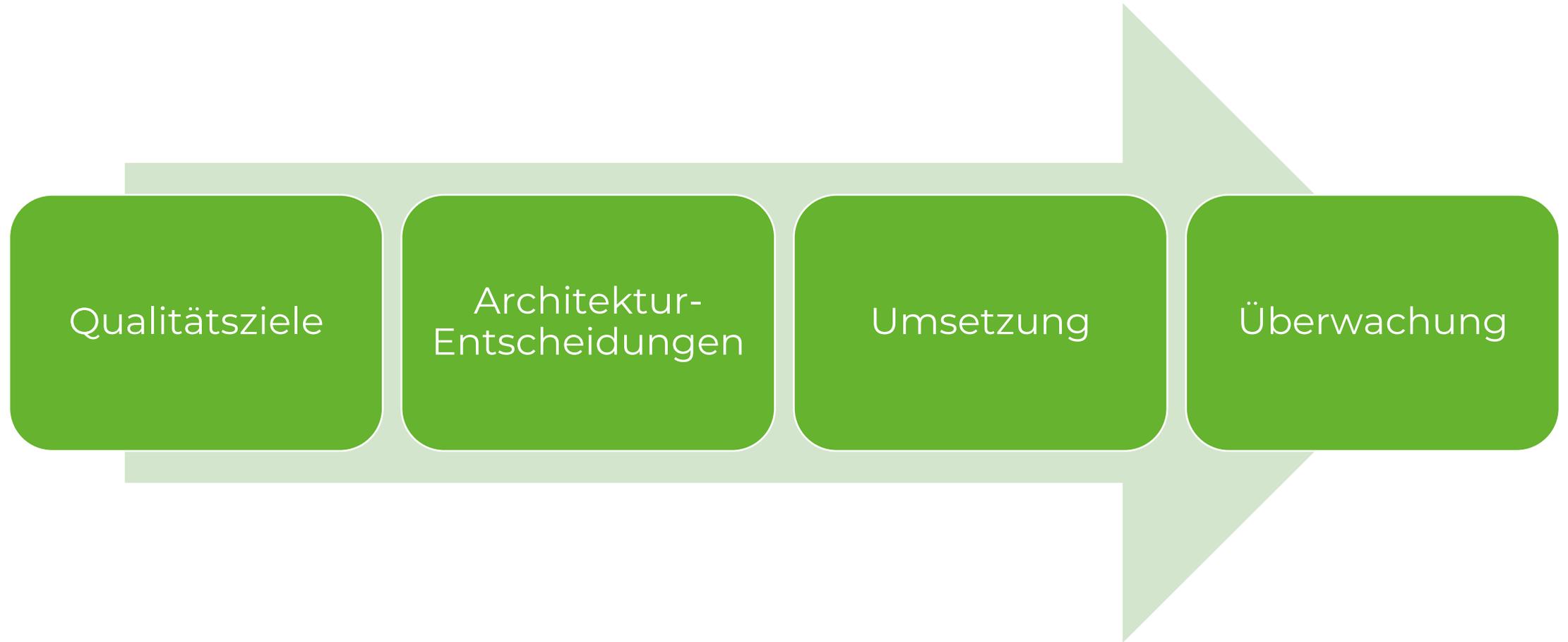
QUALITÄTSZIELE

- ^ Anforderungen sind System-spezifisch
 - ^ „Instanzen von Qualitätsmerkmalen“
- ^ Ausrichtung an Geschäftszielen
 - ^ Priorisiert und mit Stakeholdern abgestimmt
- ^ Konkretisiert durch überprüfbare Qualitätsszenarien
 - ^ Verfügbarkeit
 - ^ Kapazität und Antwortzeiten
 - ^ Zeit bis zur Fehlerbehebung

QUALITÄTSZIELE

- △ ...können zu Projektbeginn vollständig definiert werden?
- △ ...sind über die Lebensdauer des Systems unveränderlich?
- △ ...gelten gleichermaßen für alle Bestandteile des Systems?

SOFTWARE-ARCHITEKTUR



ARCHITEKTUR-ENTSCHEIDUNGEN

- ^ Werkzeuge und Rahmenbedingungen
 - ^ Technologisch
 - ^ Architekturstil und -muster
 - ^ Technologieauswahl
 - ^ Strukturierung & Schnittstellen
 - ^ Querschnittsaspekte
 - ^ Prozessual
 - ^ Entwicklung & Qualitätssicherung
 - ^ Organisatorisch
 - ^ Team-Strukturen
 - ^ Vorgehensmodell(e)

ARCHITEKTUR-ENTSCHEIDUNGEN

- ▲ „Architektur ist die Menge wichtiger Entscheidungen, die später nur schwer zu ändern sind“ (Martin Fowler)
- ▲ schwer = Kosten & Risiken

SOFTWARE-ARCHITEKTUR

▲ Architektur ist mehr als „Technologie“

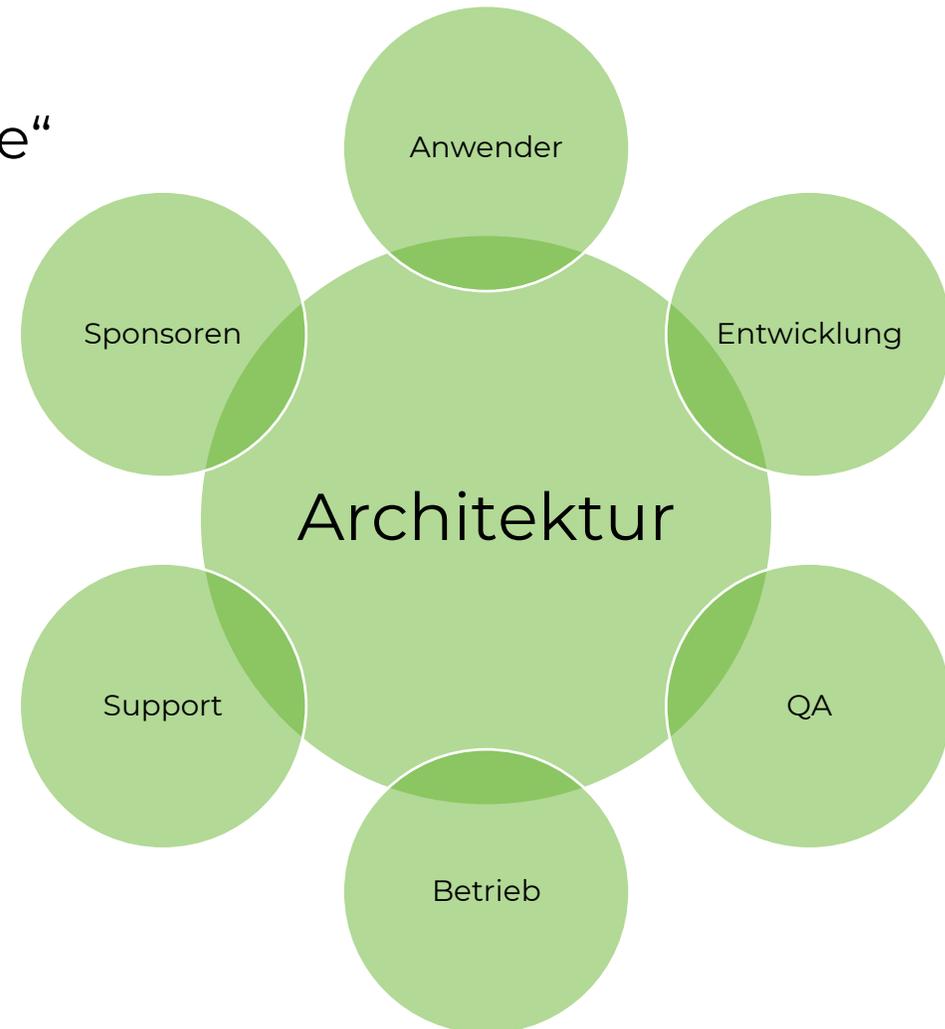
▲ interdisziplinäre Stakeholder

▲ „Der Architekt“

▲ Moderator

▲ Designer

▲ Kommunikator



DOKUMENTATION

^ Ist (guter) Code selbsterklärend?

→ Beschreibung des „Wie?“

^ Aber „Warum?“

^ Architekturstil

^ Strukturierung des Systems

^ Technologie- und & Framework-Auswahl

^ ...

→ Architekturdokumentation

DOKUMENTATION

- ^ Dokumentation von Architekturentscheidungen
 - ^ Kontext
 - ^ Abwägungen und Entscheidungen
 - ^ Konsequenzen

- ^ Startpunkt: arc42
 - ^ <https://arc42.org>
 - ^ Template für Architekturdokumentation
 - ^ verschiedene Formate

ARC42

Einführung und Ziele

Aufgabenstellung, Qualitätsziele, Stakeholder

Kontextabgrenzung

Fachlicher und Technischer Kontext

Lösungsstrategie

Bausteinsicht

Laufzeitsicht

Verteilungssicht

Querschnittliche Konzepte

Architekturentscheidungen

Architecture Decision Records

Qualitätsanforderungen

Qualitätsbaum und Qualitätsszenarien

Risiken und technische Schulden

Glossar

→ Grundlegendes Verständnis für das „Big Picture“ des Systems

Warum und Wie sind Entscheidungen gefällt worden?

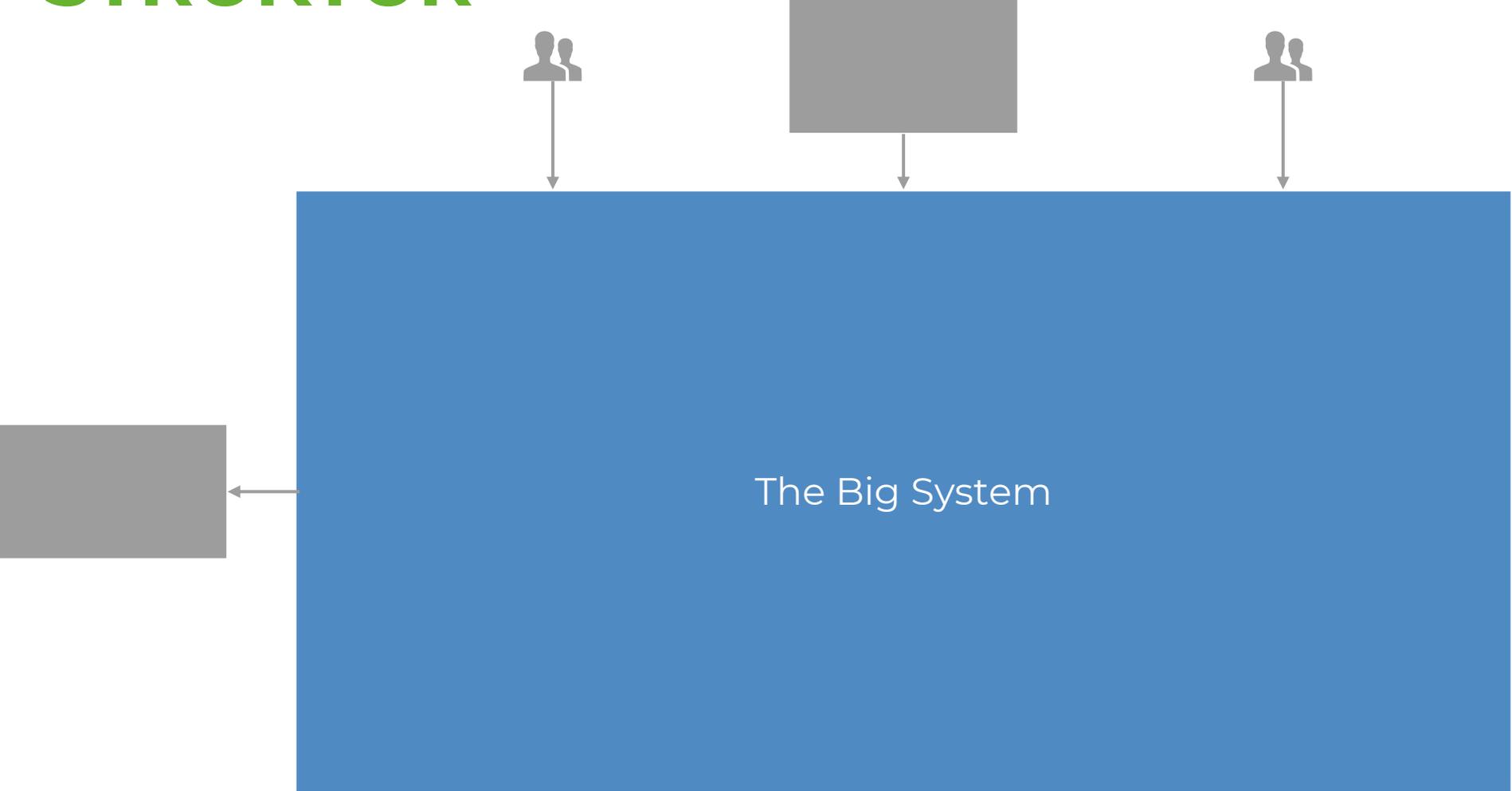
ENTSCHEIDUNGEN & KONSEQUENZEN

STRUKTUR

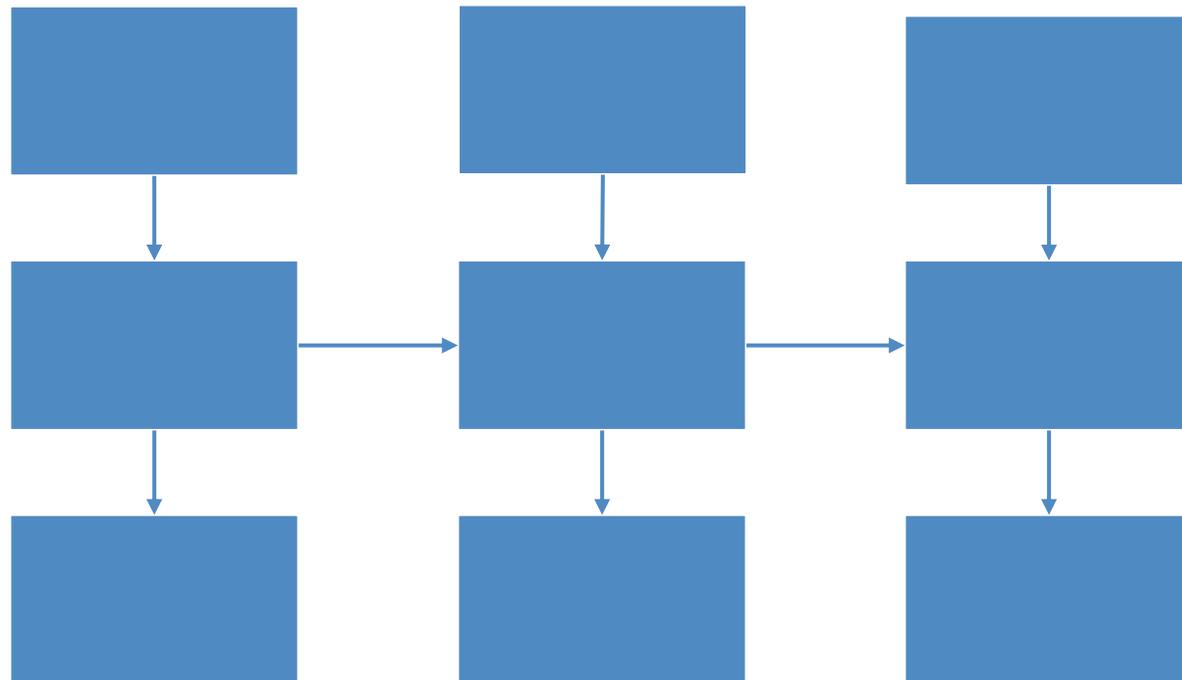
„Wir machen Microservices und DDD!“

Warum eigentlich?

STRUKTUR



STRUKTUR



Fachlich oder
technisch?

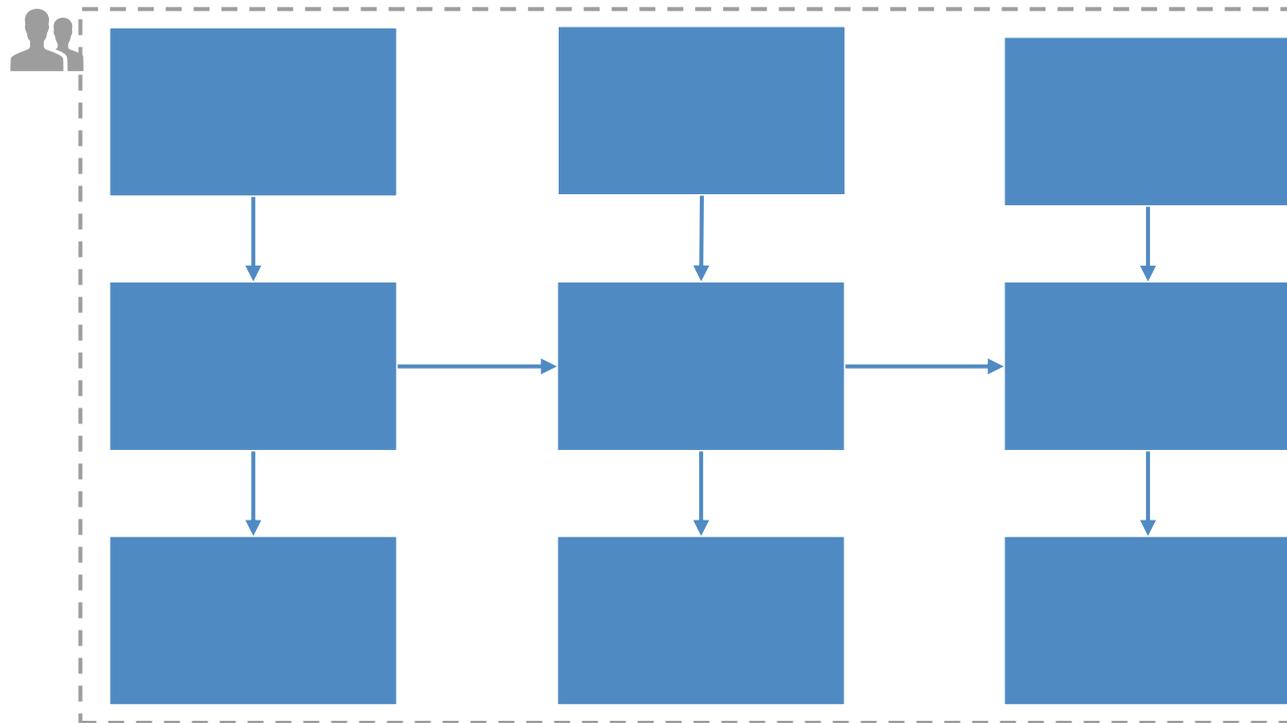
Monolith oder
Microservices?

STRUKTUR

- ^ Strukturierung = Schnitte
 - ^ Ziel: Isolation & Entkopplung
 - ^ Kontexte: Entwicklung & Laufzeit
 - ^ Nebenwirkungen: Aufwände (und Risiken)

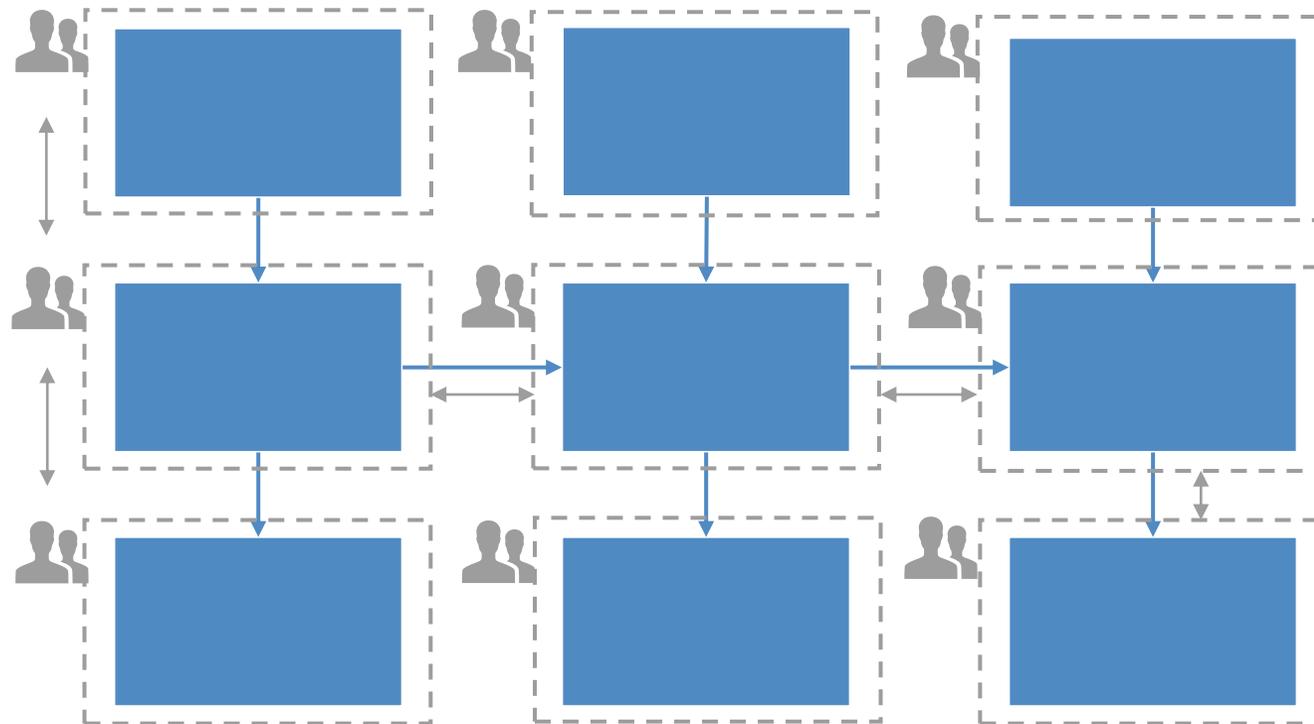
- ^ Getrieben durch Qualitätsziele, z.B.
 - ^ Wartbarkeit
 - ^ Performance und Effizienz
 - ^ Sicherheit

STRUKTUR



Umfang
und Komplexität?

STRUKTUR



Abstimmungs-
Aufwände?

STRUKTUR

- ▲ Cross-Team-Interaktionen sind aufwändig
 - ▲ Räumliche & zeitliche Trennung
 - ▲ Handshakes an Sprint-Grenzen
 - ▲ Prioritätskonflikte

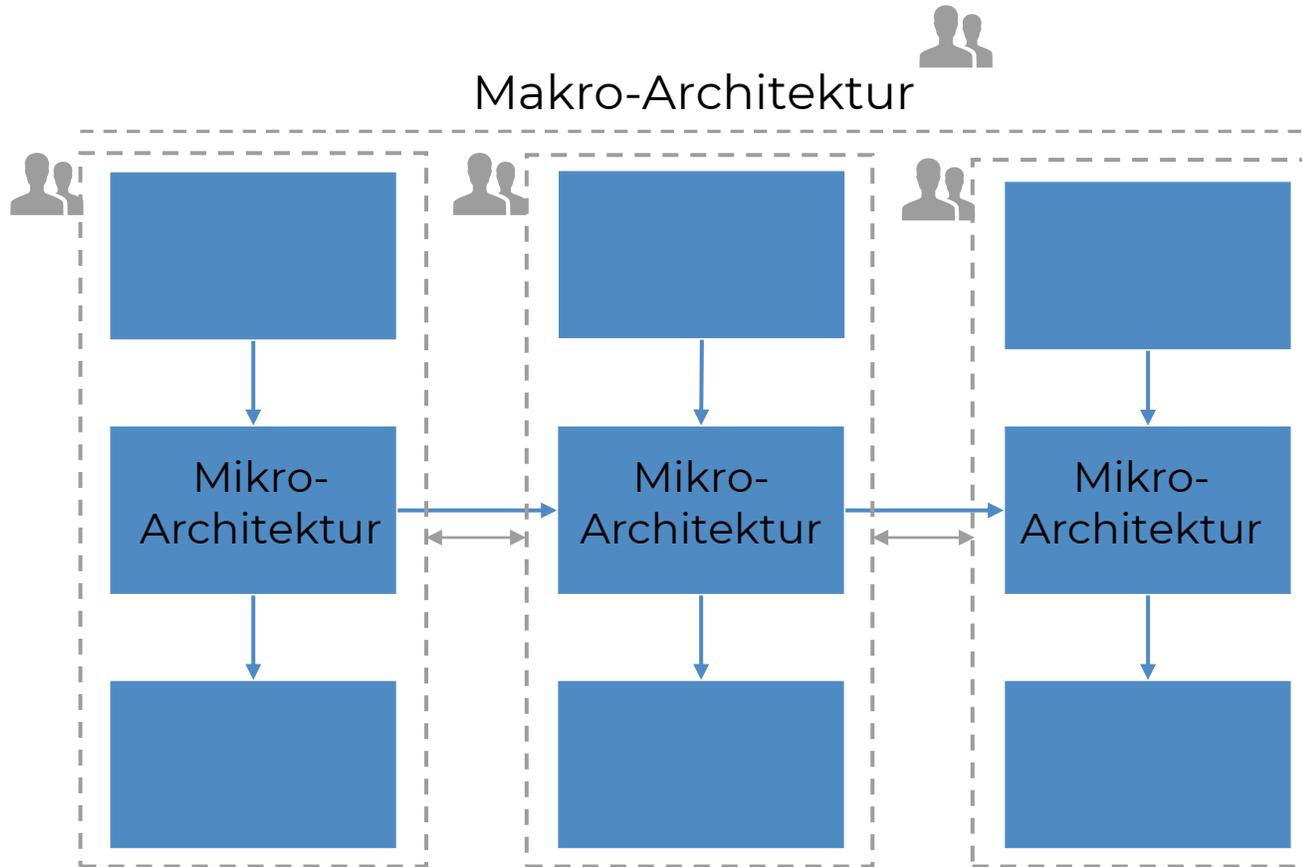
- ▲ Wie schnell können Änderungen ausgeliefert werden, wenn x Teams involviert sind?
 - ▲ Abstimmung von Schnittstellen
 - ▲ Integration und Qualifizierung

STRUKTUR

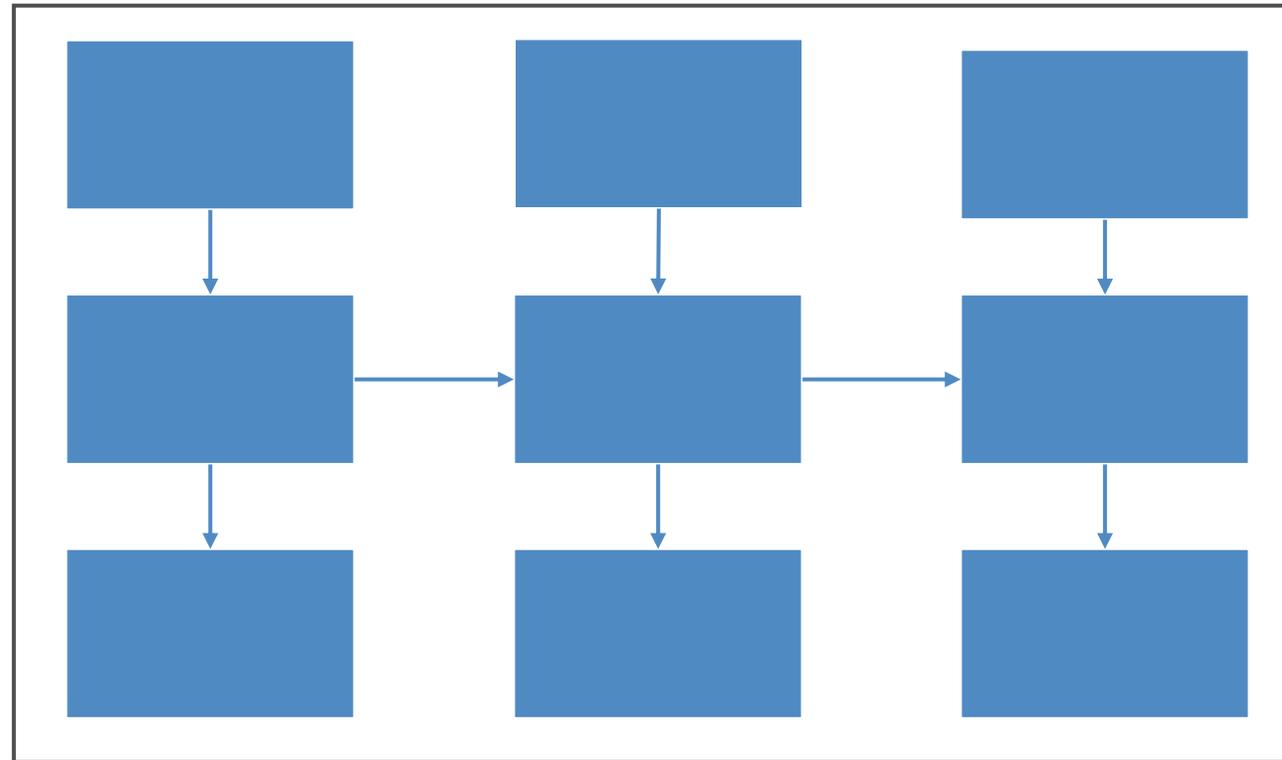
- ▲ Ownership nach Wahrscheinlichkeit gemeinsamer Änderungen
 - ▲ üblicherweise fachlich getrieben
 - ▲ „vertikal“, durch technische Schichten hindurch
 - ▲ Kohäsion

- ▲ Strategic Domain Driven Design (DDD)
 - ▲ Fokus auf fachliche Strukturierung
 - ▲ „Bounded Contexts“
 - ▲ Nähe zu fachlichen Stakeholdern

STRUKTUR

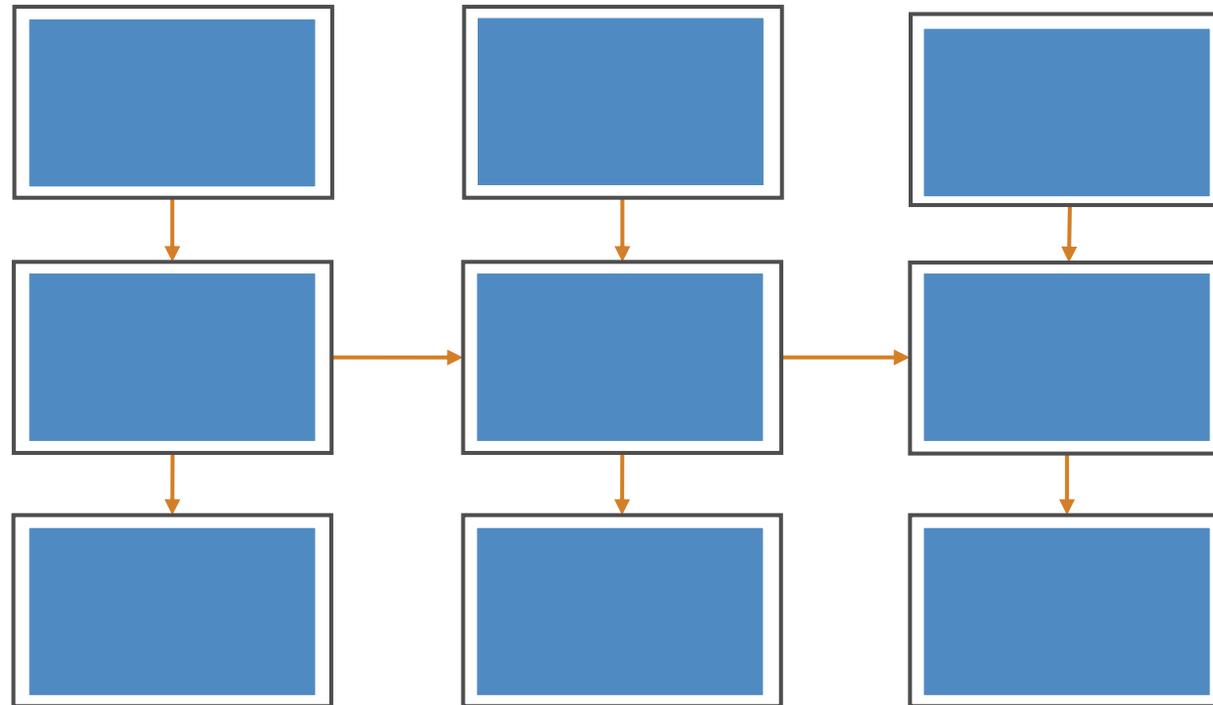


STRUKTUR



Schwerfälliger,
Ressourcen-hungriger
Laufzeit-Monolith?

STRUKTUR



Microservices mit
langen
Antwortzeiten &
Fehlerkaskaden?

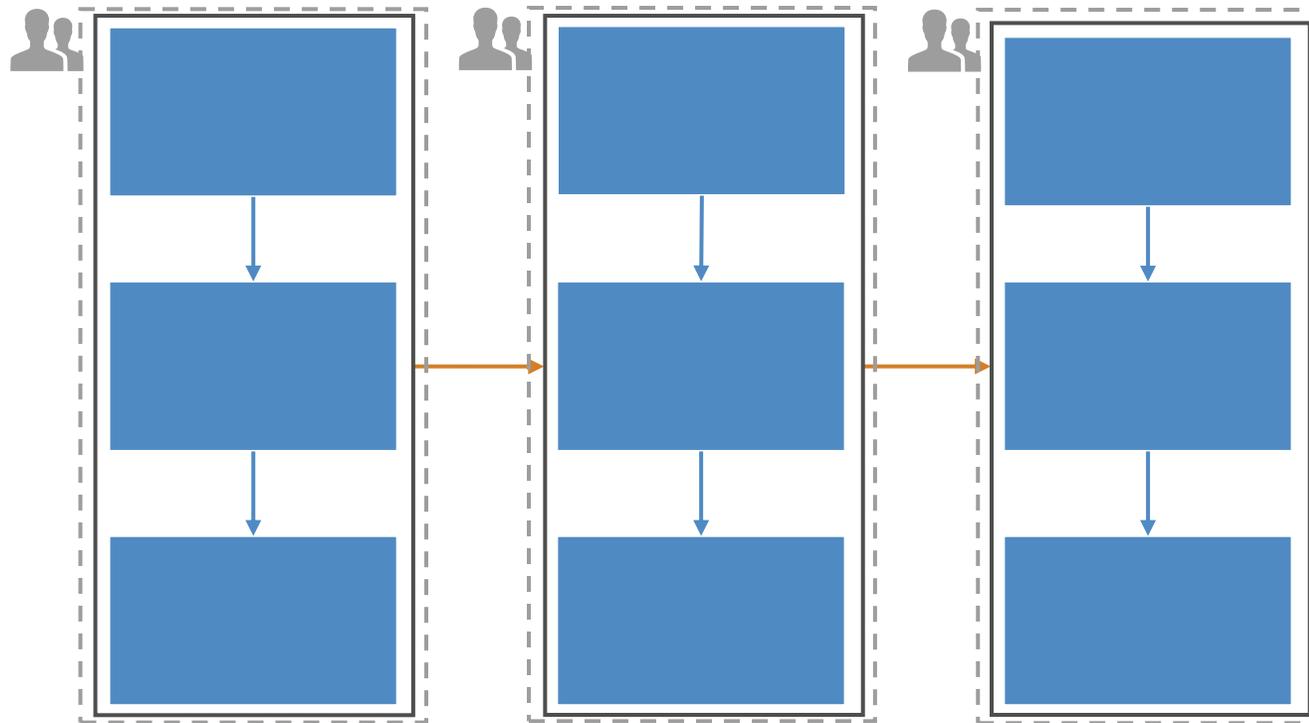
STRUKTUR

- ▲ Microservices sind Verteilte Systeme = „Anspruchsvoll“
 - ▲ Inter-Prozess-Kommunikation
 - ▲ Netzwerk-basiert
 - ▲ Laufzeitverhalten, Fehlerbehandlung, Konsistenz
 - ▲ Höherer Aufwand für Infrastruktur (Monitoring, Tracing)
- ▲ Deployment-Monolith mit „Nebenwirkungen“
 - ▲ Mangelnde Ressourcen-Isolation
 - ▲ Unnötiger Ressourcenverbrauch bei Skalierung

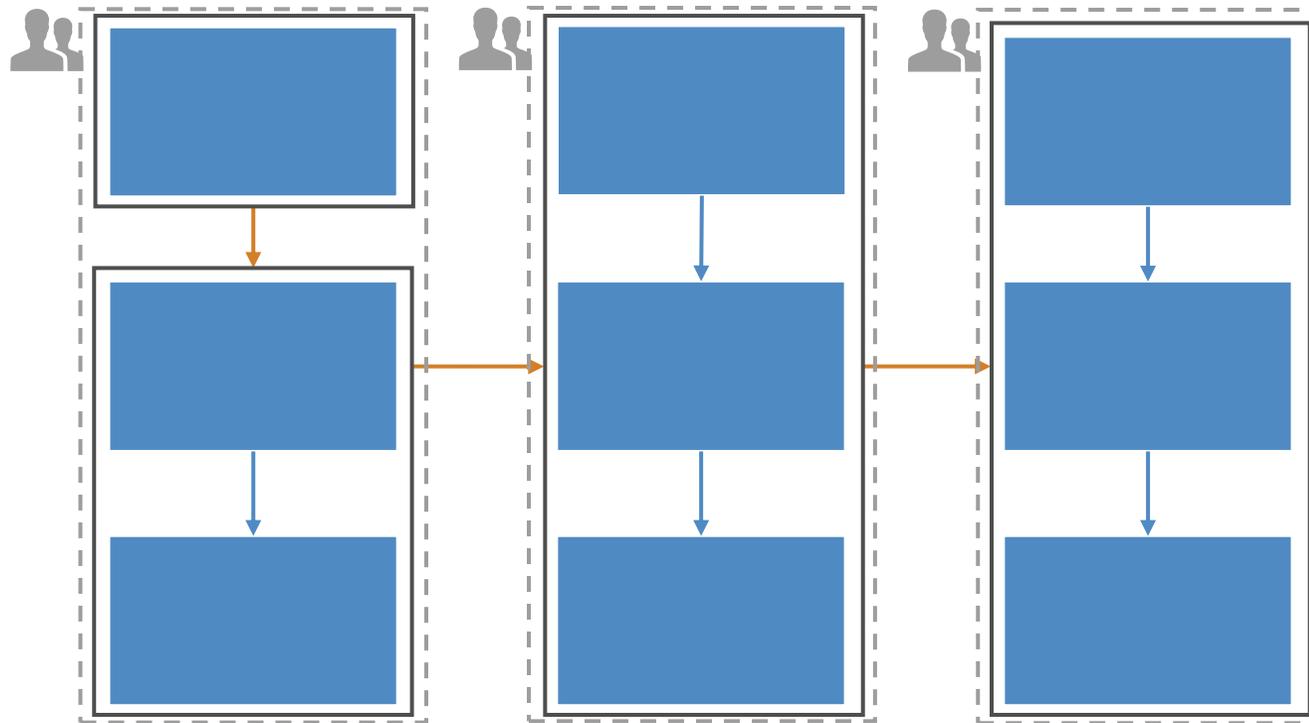
STRUKTUR

- ▲ Qualitätsziele als Treiber für Monolith vs. Microservices
- ▲ Divergierende Qualitätsanforderungen für Funktionalitäten, z.B.
 - ▲ Skalierbarkeit
 - ▲ Verfügbarkeit
- ▲ Wartbarkeit?
 - ▲ Ziel: Unabhängige Entwicklung durch Teams
 - ▲ Microservices sind keine hinreichende Bedingung
 - ▲ „Verteilter Monolith“

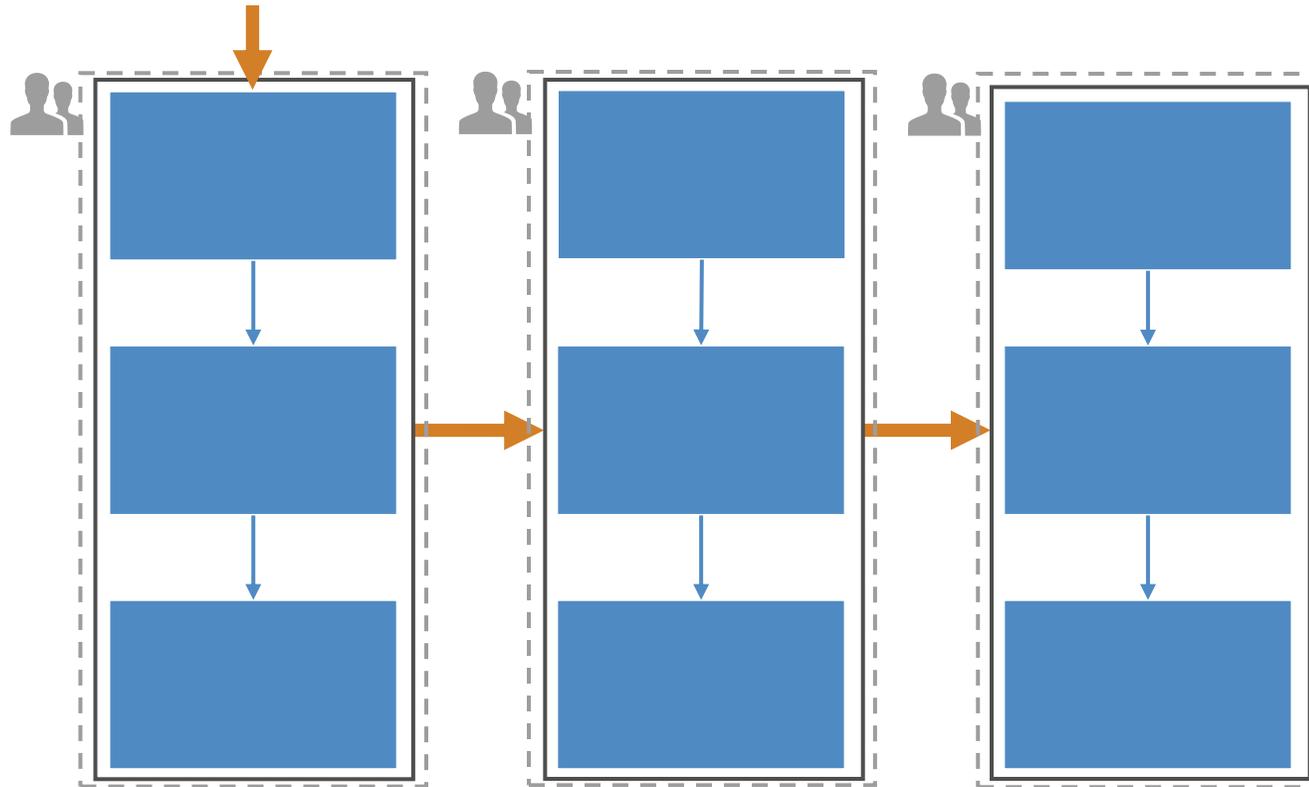
STRUKTUR



STRUKTUR



SCHNITTSTELLEN



SCHNITTSTELLEN

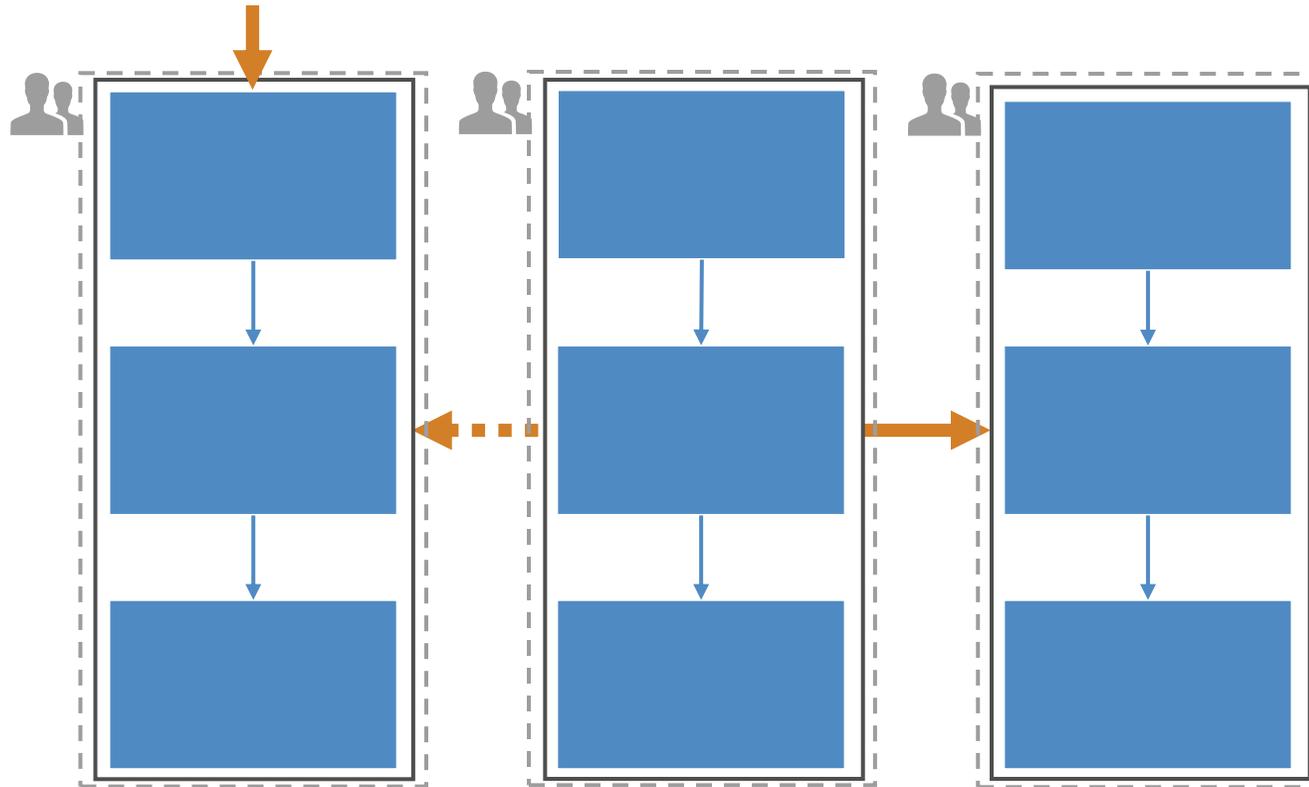
^ Anforderung

- ^ Antwortzeiten
 - ^ Echtzeitverarbeitung
 - ^ Usability

^ Verteilte Systeme

- ^ Verfügbarkeit von Services und Netzwerkinfrastruktur
- ^ Latenz & Durchsatz
- ^ Kaskadierende Effekte

SCHNITTSTELLEN



SCHNITTSTELLEN

^ Integrationsstil

- ^ Umkehrung der Kommunikationsrichtung
 - ^ gezielte Replikation von Daten
- ^ Synchron → Asynchron

^ Trade-Off

- ^ Entkopplung einzelner Funktionalitäten
 - ^ Performanz (u.a. Antwortzeiten)
 - ^ Verfügbarkeit → „Graceful Degradation“
- ^ Einschränkungen der Konsistenz

VIELEN DANK!

Q&A: Questions & Answers