Hochschule RheinMain

ARCHITEKTUREN FÜR VERTEILTE SYSTEME

Verteilte Systeme

Prof. Dr. Georg Hinkel
07.06.2024

GLIEDERUNG

Hochschule RheinMain

Datum	Vorlesung	Übungsblatt	Abgabe
19.04.2024	Einführung	HamsterLib	06.05.2024
26.04.2024	Netzwerkprogrammierung	Theorie	
03.05.2024	World Wide Web	HamsterRPC 1	20.05.2024
10.05.2024	Remote Procedure Calls	Theorie	
17.05.2024	Webservices	HamsterRPC 2	03.06.2024
24.05.2024	Fehlertolerante Systeme	Theorie	
31.05.2024	Transportsicherheit	HamsterREST	17.06.2024
07.06.2024	Architekturen für Verteilte Systeme	Theorie	
14.06.2024	Internet der Dinge	HamsterIoT	01.07.2024
21.06.2024	Namen- und Verzeichnisdienste	Theorie	
28.06.2024	Authentifikation im Web	HamsterAuth	15.07.2024
05.07.2024	Infrastruktur für Verteilte Systeme	Theorie	
12.07.2024	Wrap-Up	HamsterCluster (Bonus)	16.08.2024

Hochschule RheinMain

Agenda

- Client/Server, Peer-to-Peer
- 3-Tier, N-Tier
- Microservices

Lernziele

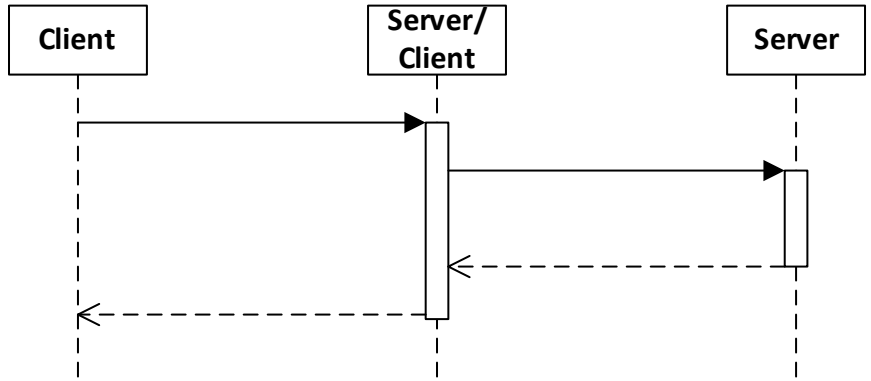
- Architekturstile abwägen können

ARCHITEKTURMODELLE FÜR VERTEILTE SYSTEME

Client/Server

Hochschule RheinMain

- Kommunikationsvorgang wird vom Client initiiert, basiert auf Request/Response
- Client kann mit verschiedenen Servern arbeiten
- Server kann Aufträge verschiedener Clients annehmen
- Server kann selbst wieder als Client auftreten



- Beispiel: Proxy
  - Zwischengeschaltete Instanz, übernimmt Server-Rolle gegenüber Client und delegiert Request
  - Zusätzliche Aufgaben (e.g., Authentifizierung, Caching, Virenscan, Lastverteilung, …)

ARCHITEKTURMODELLE FÜR VERTEILTE SYSTEME

Peer-to-Peer (P2P)

Hochschule RheinMain

- Dezentrale Kommunikation zwischen Gleichrangigen (~ Peer)
- Keine zusätzliche zentrale Infrastruktur ➔ Verringerte Angreifbarkeit

- Beispiele
  - File-Sharing, z.B. BitTorrent, Gnutella, eMule
  - Skype bis 2016

ARCHITEKTURMODELLE FÜR VERTEILTE SYSTEME

Multi-Tier

Hochschule RheinMain

- *Tier* = Reihe, Strang

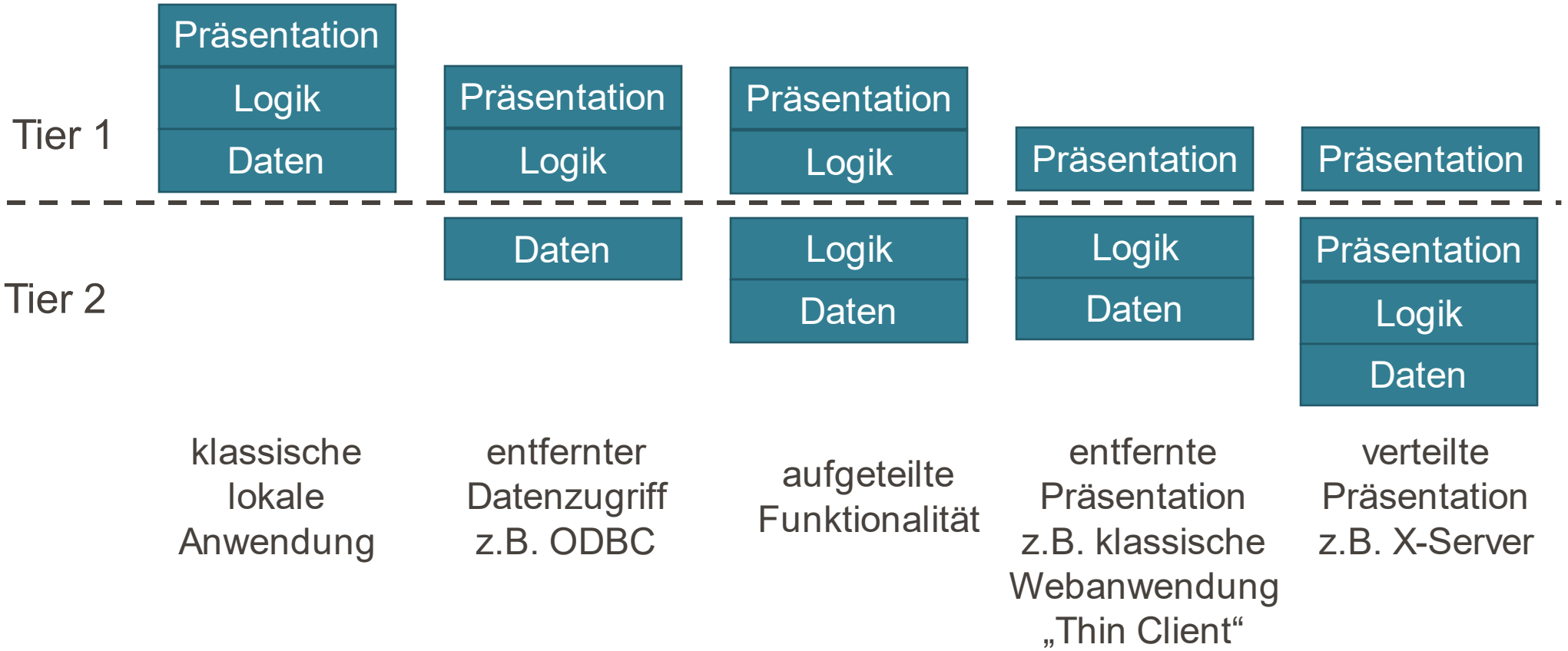
- Einteilung der Funktionalität in verschiedene Kategorien, physikalische Trennung
  - Präsentation / Benutzerschnittstellen
  - Funktion / Anwendungslogik
  - Datenhaltung

- Keine Festlegung über die verwendete Middleware

- Weit verbreitet
  - Two-Tier-Architektur
  - 3-Tier-Architektur
  - N-Tier-Architektur

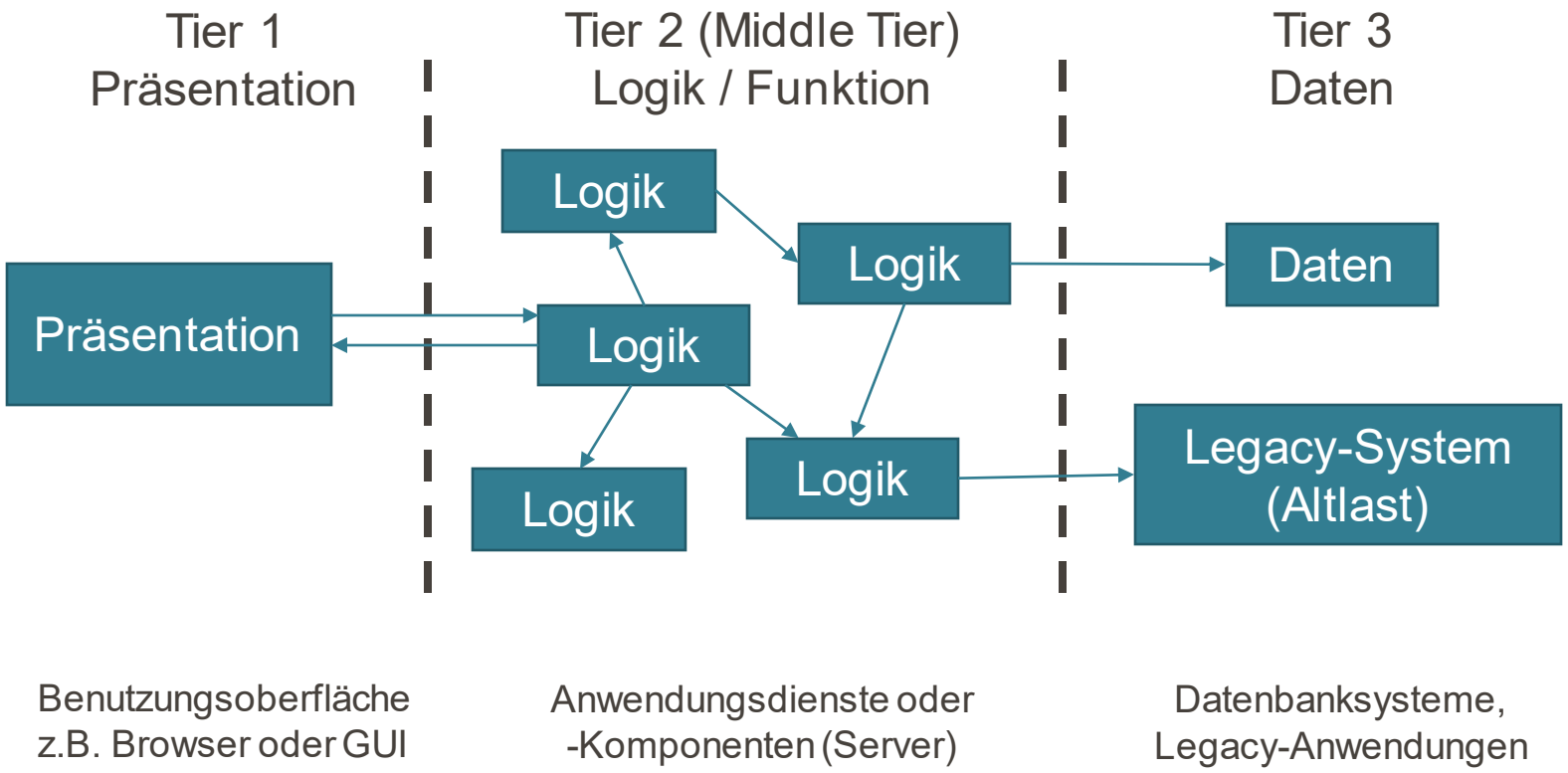
TWO-TIER ARCHITEKTUREN

Hochschule RheinMain



3-TIER ARCHITEKTUR

Hochschule RheinMain

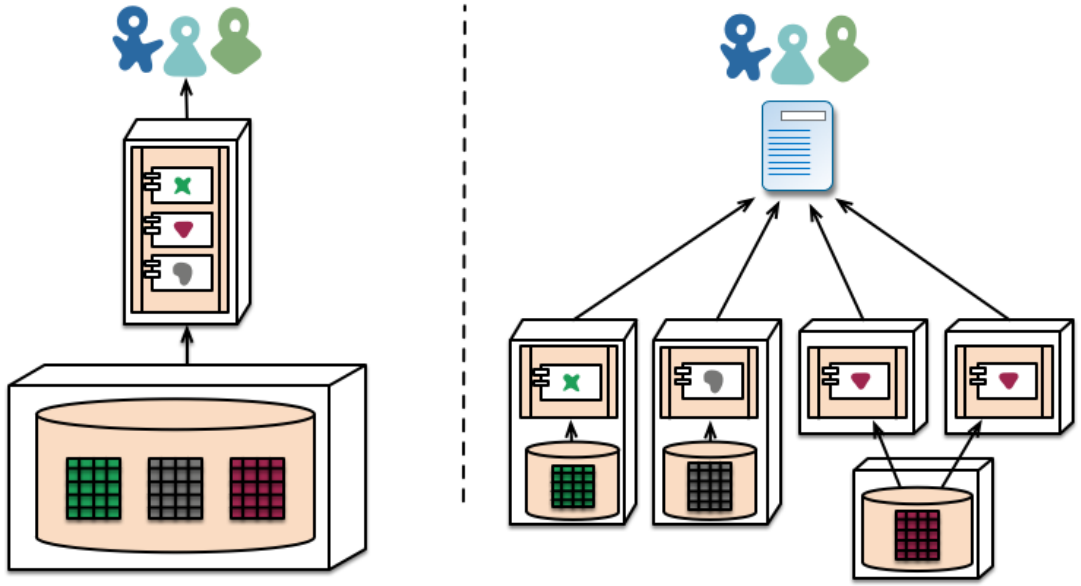


- Verallgemeinerung in N-Tier
  - Typischerweise Auftrennen des Logik-Strangs

MICROSERVICE-ARCHITEKTUREN
Einführung

Hochschule RheinMain

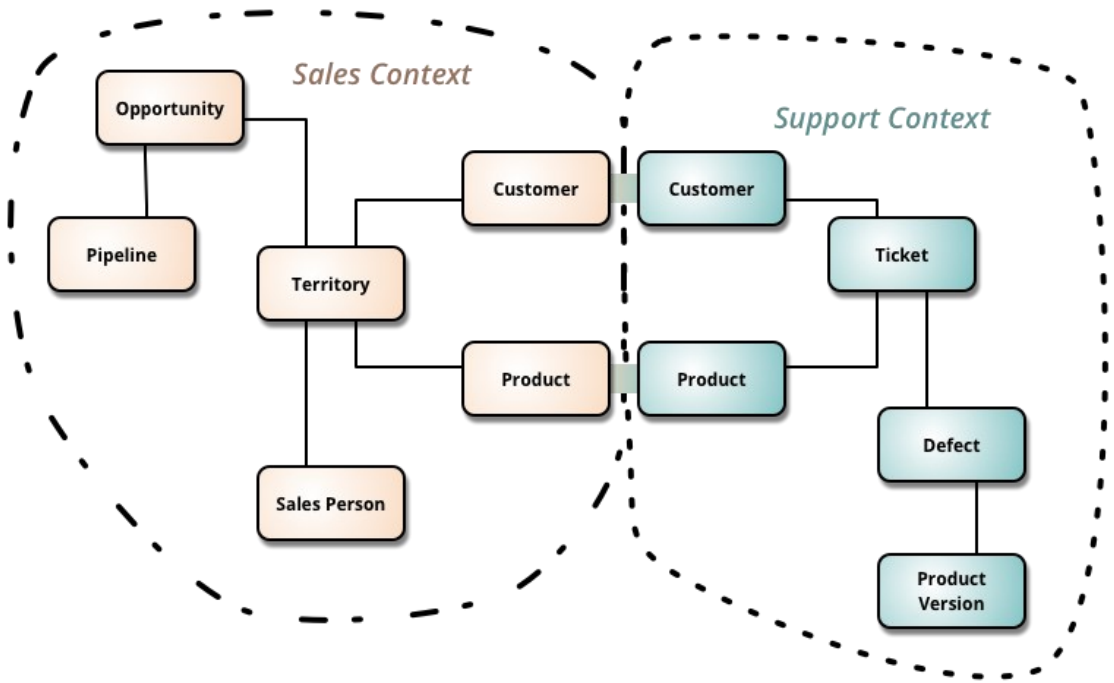
- Anwendung wird aufgeteilt in mehrere Services, die unabhängig Teile der Funktionalität erbringen
  - Fachliche Aufteilung des Systems, „Do one thing and do it right“



[https://martinfowler.com/articles/microservices.html]

MICROSERVICE-ARCHITEKTUREN
Bounded Contexts

Hochschule RheinMain



[https://martinfowler.com/bliki/BoundedContext.html]

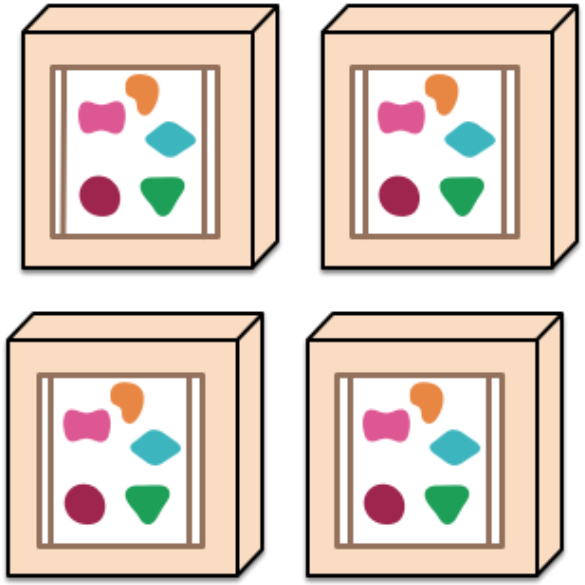
- Idee: Unterteile Fachlichkeit einer Anwendung gemäß Domain-Driven-Design (DDD) in abgeschlossene Bereiche (Bounded Contexts)
  - Viele Anfragen benötigen mehrere Daten aus dem Kontext (hohe Kohäsion)
  - Wenige Anfrage benötigen Daten aus mehreren Kontexten (geringe Kopplung)
- Für jeden Bounded Context sollte dann ein eigener Microservice angelegt werden
- Jeder Microservice ist dann selbst für seine Daten verantwortlich
  - Daten werden ggf. repliziert

MICROSERVICE-ARCHITEKTUREN
Skalierung

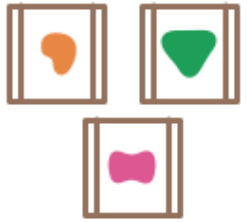
Hochschule **RheinMain**

A monolithic application puts all its functionality into a single process...

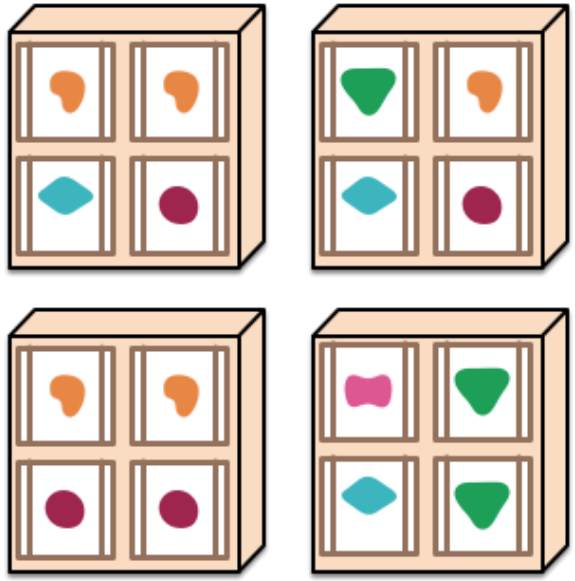
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



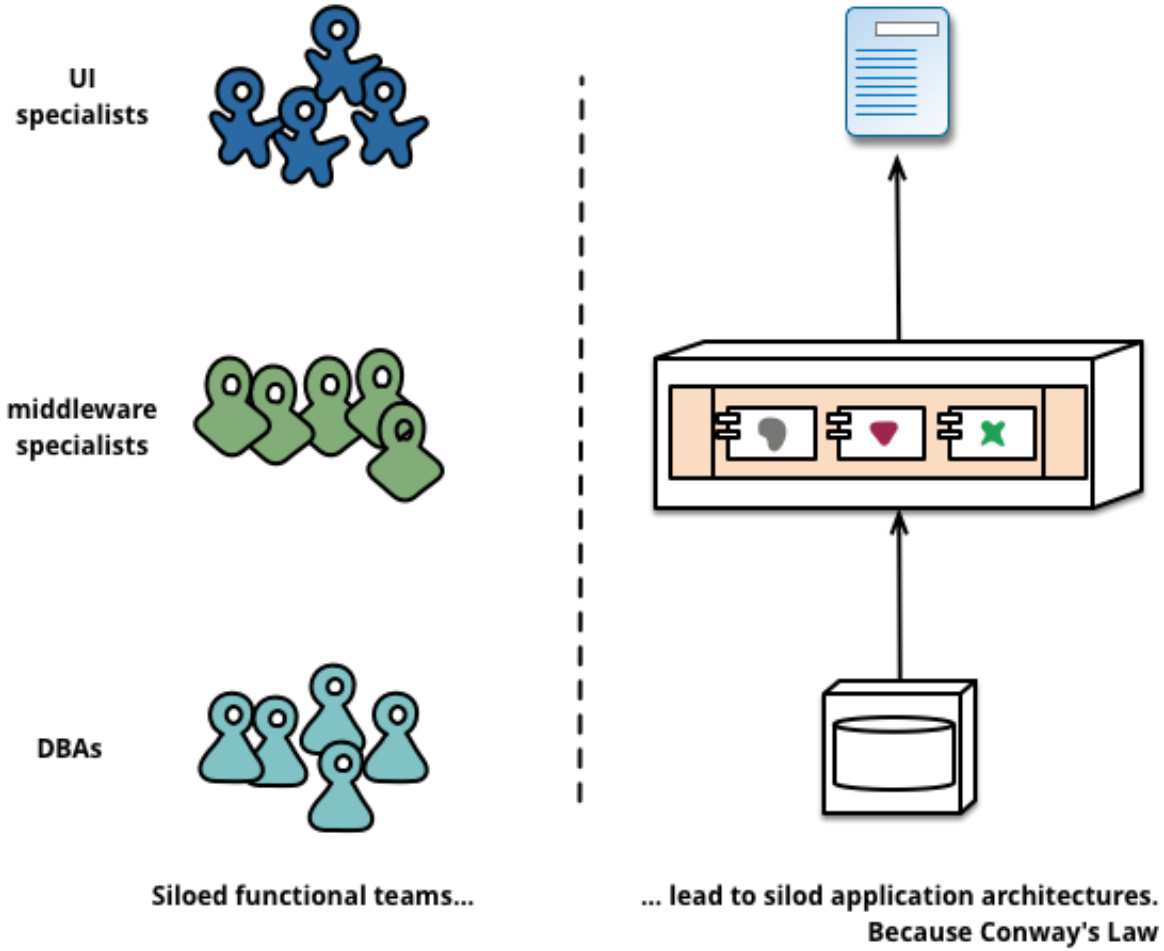
[https://martinfowler.com/articles/microservices.html]

ANY ORGANIZATION THAT DESIGNS A SYSTEM […] WILL PRODUCE A DESIGN WHOSE STRUCTURE IS A COPY OF THE ORGANIZATION'S COMMUNICATION STRUCTURE.

Melvin Conway, 1968

MICROSERVICE-ARCHITEKTUREN

Silos führen zu 3-Tier-Architekturen



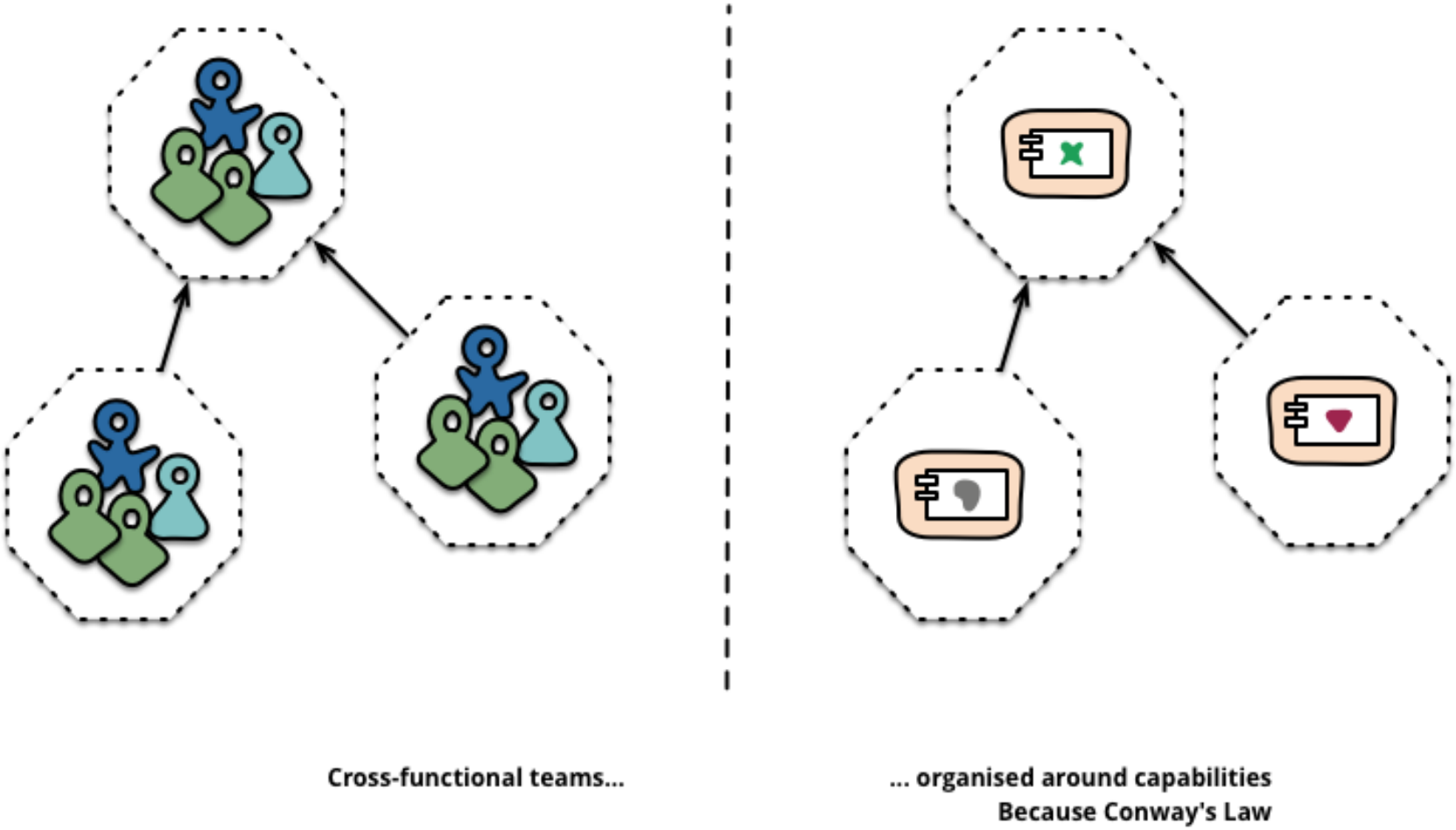
[https://martinfowler.com/articles/microservices.html]

- Teams werden häufig eher anhand von Kompetenzen zusammengestellt
  - Einfacheres Management
- Führt zu Architektur entlang Kompetenzen anstatt Fachlichkeiten
  ➔ Conway‘s Law
- Führt zu Abhängigkeiten zwischen Teams für einfache Aufgaben
  ➔ Erhöhter Kommunikationsaufwand

MICROSERVICE-ARCHITEKTUREN

Cross-funktionale Teams führen zu fachlicher Teilung

Hochschule RheinMain



[https://martinfowler.com/articles/microservices.html]

MICROSERVICE-ARCHITEKTUREN
Diskussion

Hochschule RheinMain

Vorteile

- Klare Modulgrenzen
  - Erzwingt klare Softwarearchitektur
  - Kaum Workarounds möglich
- Entwicklung der Services ist unabhängig
  - Unabhängigkeit der Entwicklungszyklen
  - Änderungen können separat deployt werden
  - Autonomie der Entwicklungsteams
- Diversität der Software Plattformen
  - Wahl der Technologie basierend auf Anforderungen für jeden Service separat

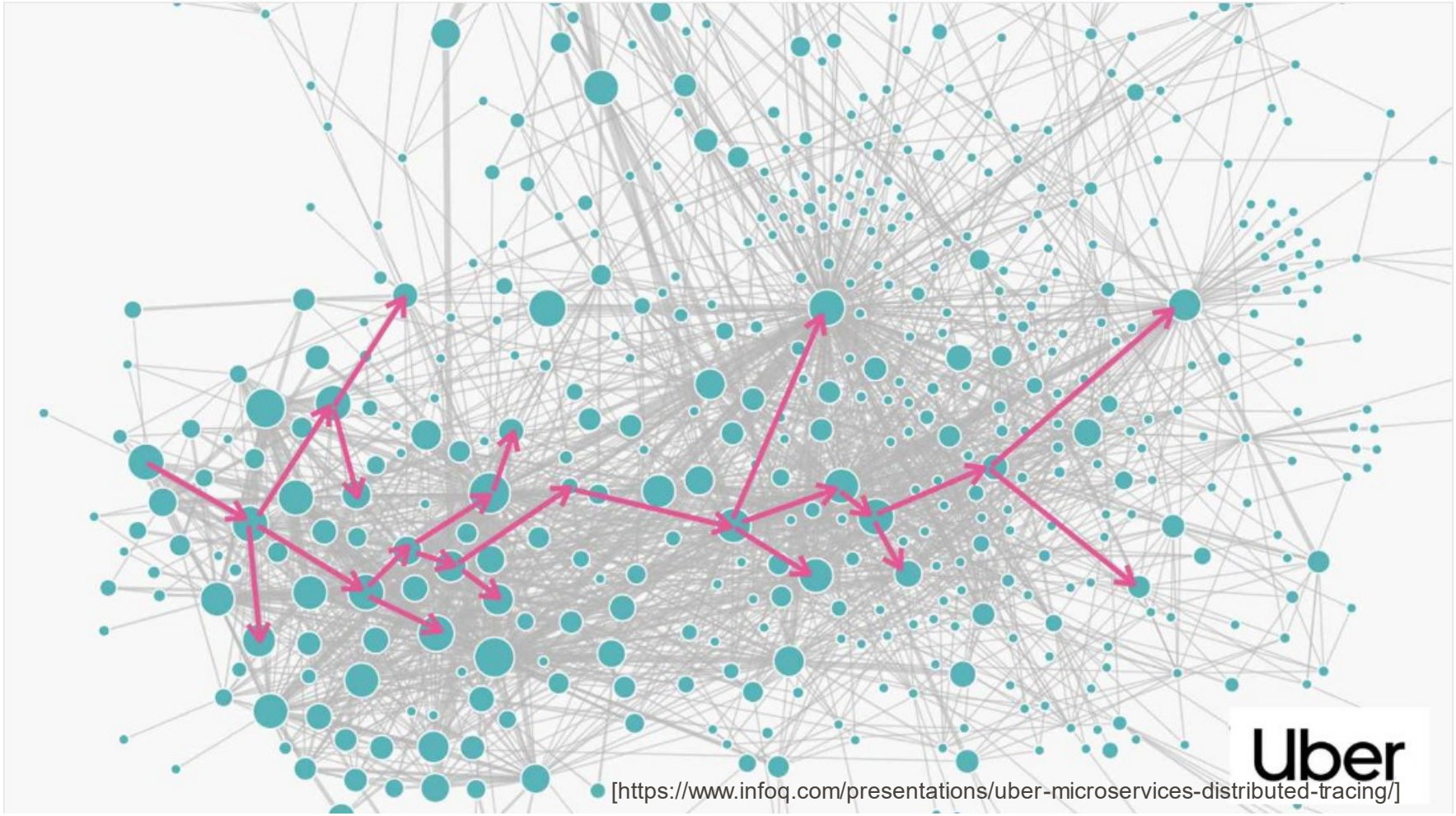
Nachteile

- Verteilung
  - Mehr Angriffsfläche
  - Performance-Einbußen durch häufiges Konvertieren
- Komplexität
  - Betrieb ist komplexer
  - Fehlersuche ist komplizierter
- Konsistenz
  - Deutlich mehr Fehlerquellen
  - Sicherstellen der Konsistenz schwieriger

MICROSERVICES BEI UBER

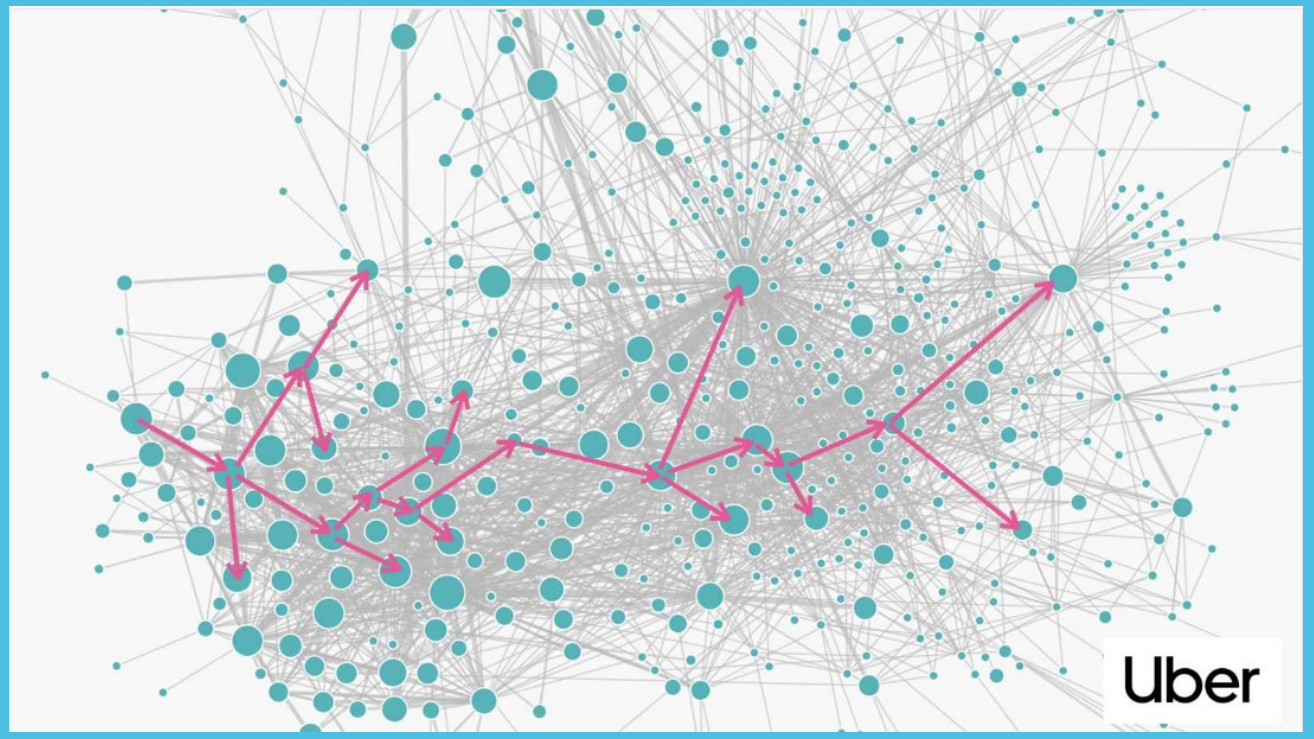
Übertrieben?

Hochschule RheinMain



[https://www.infoq.com/presentations/uber-microservices-distributed-tracing/]

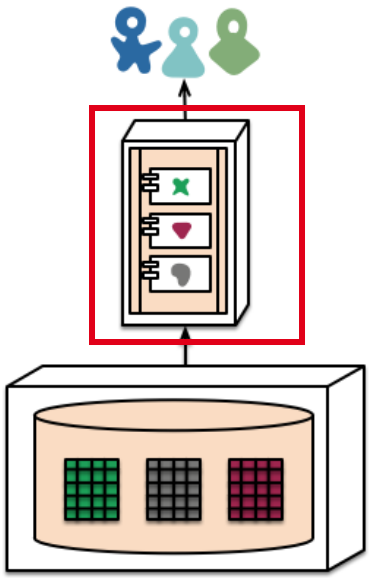
WOHER WEIß MAN EIGENTLICH WELCHE DIENSTE VON EINEM REQUEST AUFGERUFEN WERDEN?



MONOLITH FIRST

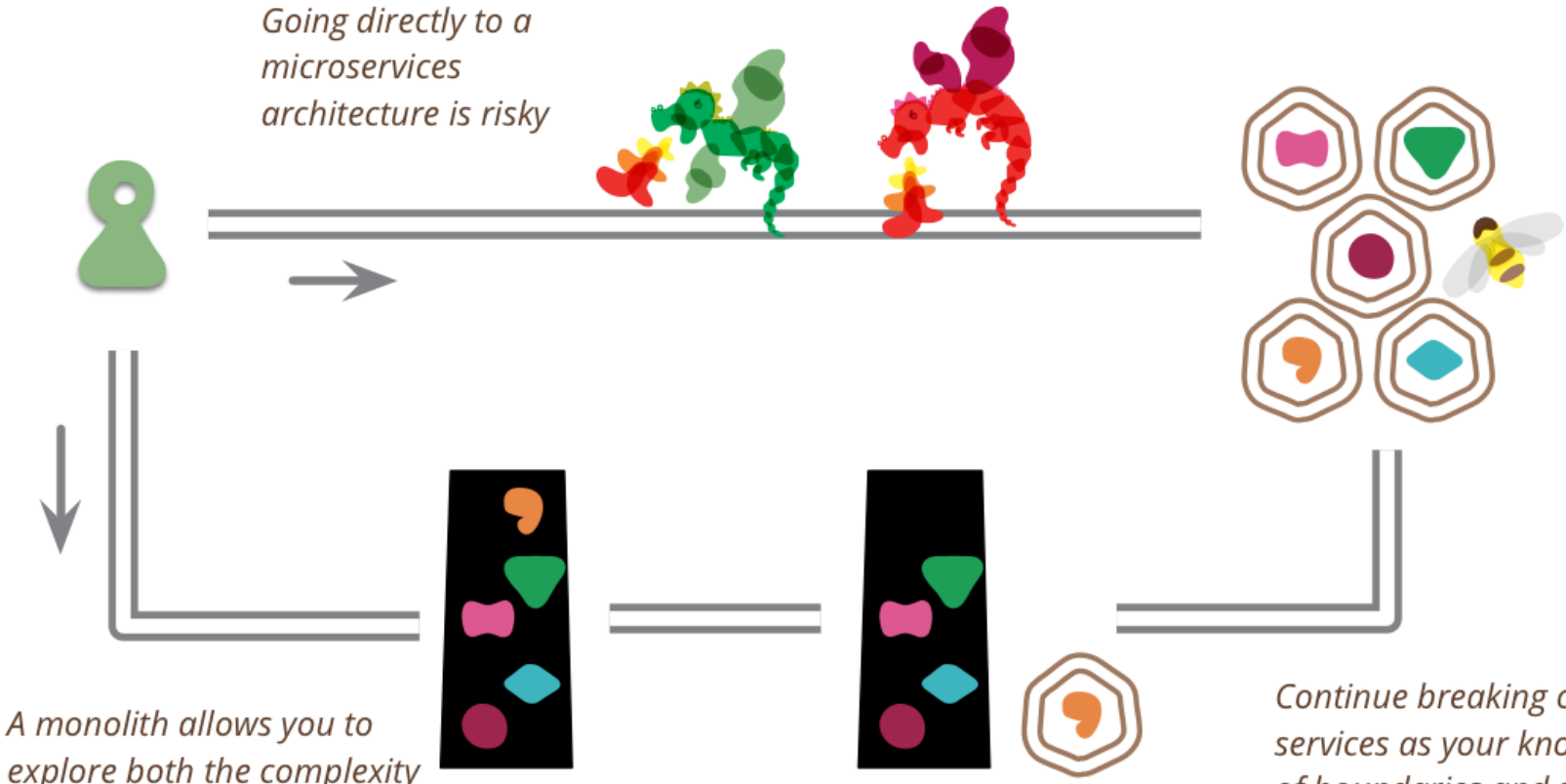
Hochschule RheinMain

- Beobachtung: Die meisten Unternehmen haben nicht so viele Aufrufe wie Uber
  - Aufteilung in viele Services führt daher nur zu erhöhter Komplexität
- Idee „Monolith First“ (M. Fowler): Starten Sie mit modular aufgebautem Deployment-Monolithen
  - Dienst lässt sich später in mehrere Dienste aufteilen, sobald die Anforderungen es erfordern
  - Skalierung graduell
  - Vorteil: geringere Komplexität, solange der Dienst nicht stark verwendet wird
- Ausnahme: Programmiersprachengrenze
  - Beispiel: Data Science Dienst wird mit Python umgesetzt, restliche Anwendung mit Java oder .NET



MONOLITH FIRST

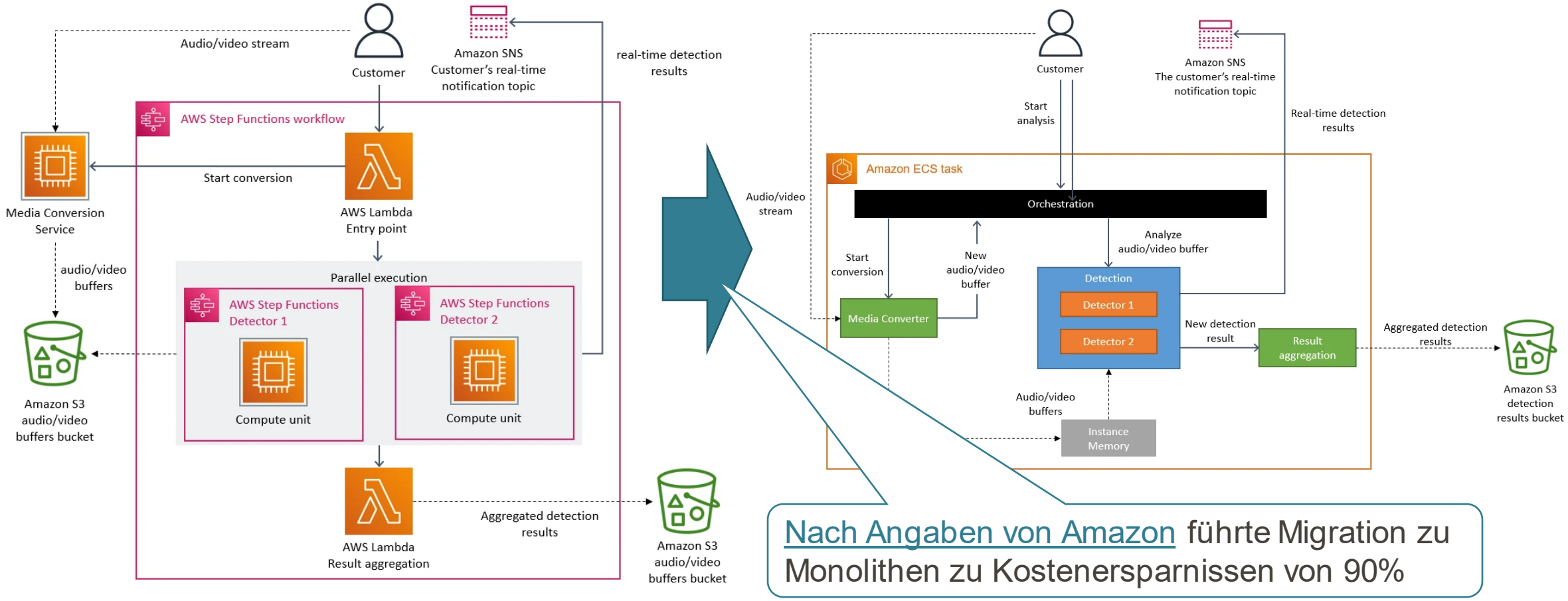
Hochschule **RheinMain**



[https://martinfowler.com/bliki/MonolithFirst.html]

GEGENBEISPIEL: AMAZON PRIME MOVES TO „MONOLITH“

Hochschule RheinMain

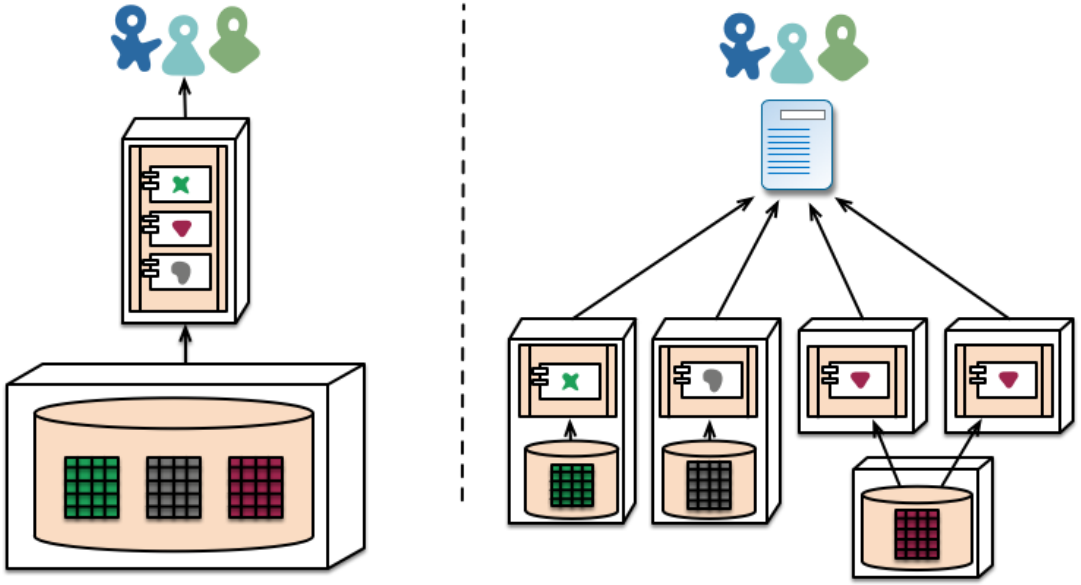


Nach Angaben von Amazon führte Migration zu Monolithen zu Kostenersparnissen von 90%

ZUSAMMENFASSUNG

Hochschule RheinMain

- Architekturen Verteilter Systeme
  - Client/Server
  - Peer-to-peer
  - N-Tier
  - Microservices



MÖGLICHE PRÜFUNGSAUFGABEN

Hochschule RheinMain

- Welches Architekturmodell würden Sie für eine gegebene Anwendung verwenden und warum?
- Welches Architekturmodell wird nach Conway von technologisch sortierten Teams befördert?
- Erläutern Sie die Vor- und Nachteile einer Microservice-Architektur!