Verteilte Systeme Master Lab

christian.zirpins@hs-karlsruhe.de

Einleitung und Überblick

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Struktur — Einleitung und Überblick

Veranstaltung

- Team
- Lehrestruktur

Einleitung

- Architekturen von Anwendungen auf Microservice-Basis

Überblick der Laborarbeit

- Aufgabenstellung
- Organisation

Einordnung in der Fakultät IWI (Informatik)

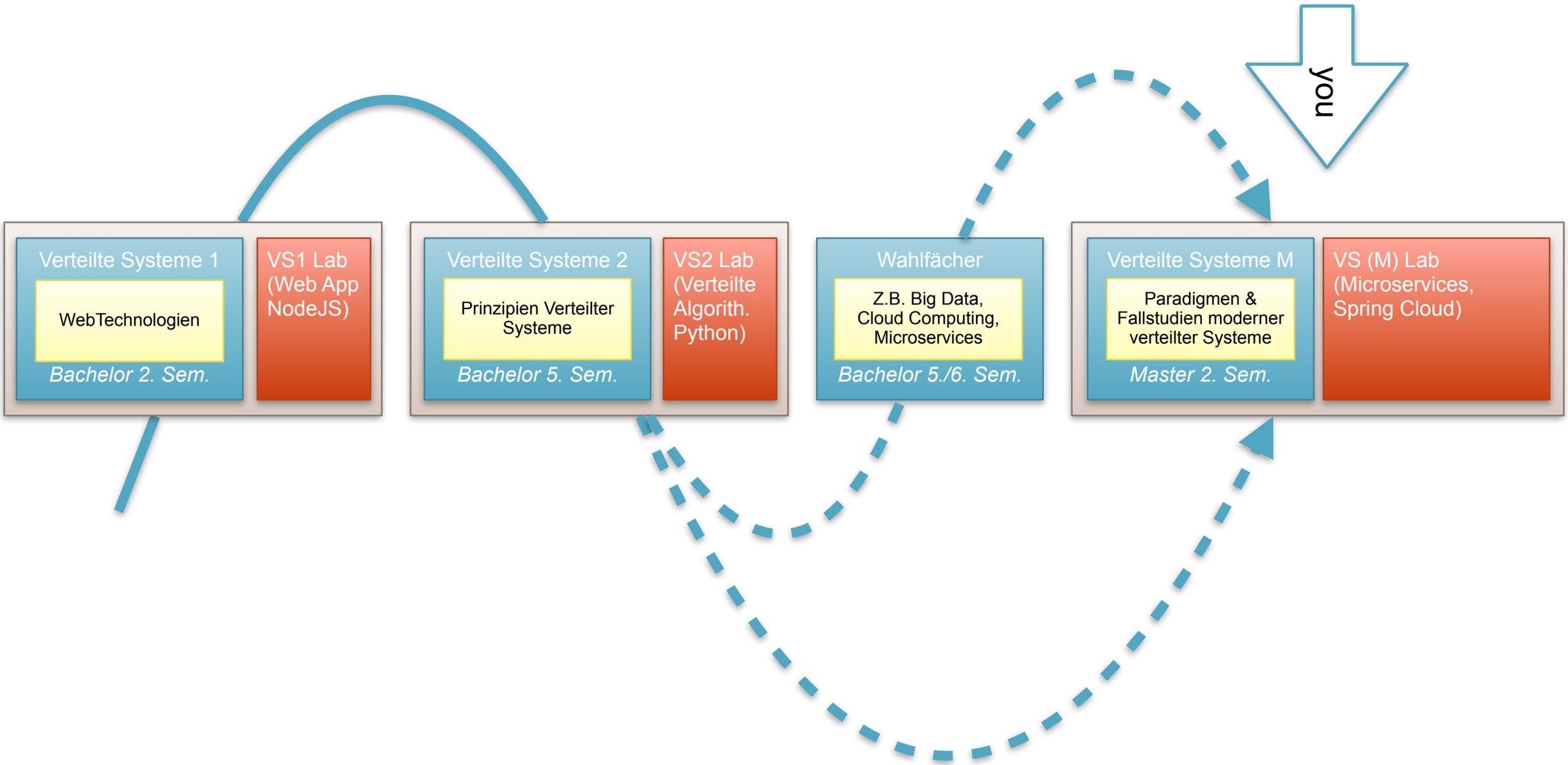
VSMLab-Team

- Dipl.-Inf. **Adelheid Knodel**
- Prof. Dr. **Christian Zirpins**

Bereich Verteilte Systeme (VSYS)

- **Schwerpunkte** in Forschung und Lehre:
  - *Web Engineering*
  - *Datenintensive Systeme ("Big Data")*
  - *Service Computing ("Microservices")*
- Mit **Bezügen** zu…
  - Cloud/Fog/Edge Computing, Internet of Things ("IoT")
  - Betriebliche und intelligente Informationssysteme ("Smart Systems")

Lehrpfade Verteilte Systeme



Nach dem Labor können Sie…

- …die **Konstruktion verteilter Informationssysteme** durch eigene Erfahrung *praktisch einschätzen*.
- …gegebene **Problemstellungen industrieller Forschung und Entwicklung** *eigenständig bearbeiten*.
- …konkrete **industrierelevante Plattformen und Frameworks** *fachgerecht anwenden*.

Konkrete Aufgabe: Refaktorisierung einer monolithischen Web Anwendung im Sinne des Microservice Architekturstils.

Struktur — Einleitung und Überblick

Veranstaltung

- Team
- Lehrestruktur

Einleitung

- Architekturen von Anwendungen auf Microservice-Basis

Überblick der Laborarbeit

- Aufgabenstellung
- Organisation

Was sind Microservices?

- **Microservices** kennzeichnen einen Architekturstil für komplexe Unternehmensanwendungen.
  - Idee und Begriff gehen auf Diskussionen der Software Engineering Community um ThoughtWorks (u.a. James Lewis, Fred George, Martin Fowler) und anderen in 2011/12 zurück.
  - Microservice Architekturen bündeln aktuelle Praktiken zur Steigerung von Agilität und Flexibilität bei der Entwicklung großer betrieblicher Informationssysteme.
- Die Microservice Idee beruht auf dem Entwurf großer, komplexer und langlebiger Anwendungen als
  - …einer Menge (relativ) kleiner, kohäsiver Services,
  - …die sich im Laufe der Zeit evolutionär entwickeln.

Architektur und Architektur-Stil

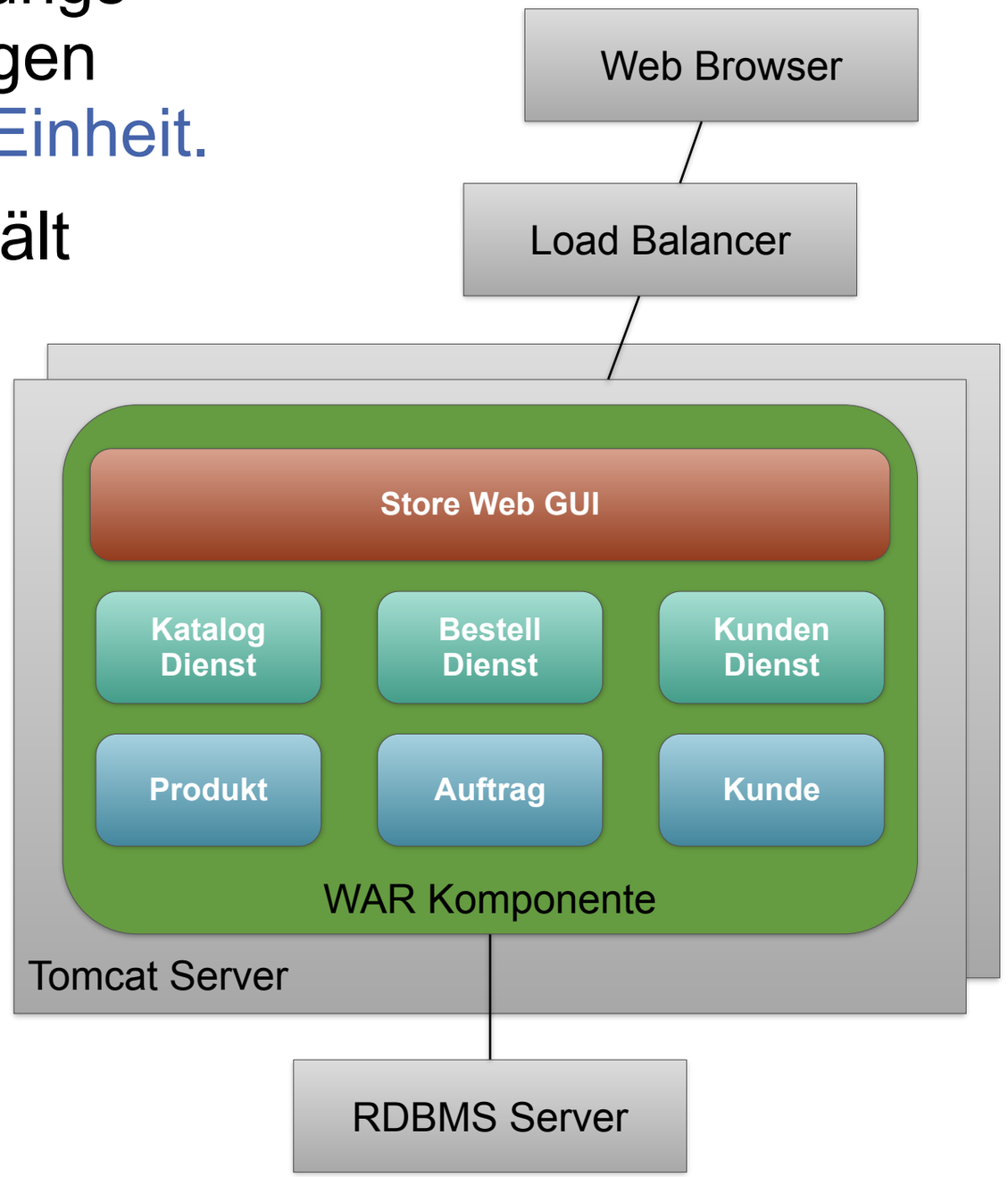
- Architektur: Entwurf einer Problemlösung in Bezug auf gegebene Bedingungen/Einschränkungen.
- Architektur-Stil: Generelle Prinzipien, die die Gestaltung von Architekturen beeinflussen.

- Architektur: Louvre
- Architektur-Stil: Barock

- Architektur: Villa Savoye
- Architektur-Stil: Moderne

Monolithische Anwendungen als Anti-Pattern

- Die gängigste betriebliche Anwendungs-architektur verpackt alle serverseitigen Komponenten in eine Deployment-Einheit.
- Ein typischer J2EE Web-Shop enthält z.B. eine Web-GUI,Geschäfts-funktionen und ein Domänen-modell mit Geschäftsdaten.
- Das Deployment all dieser Komponenten erfolgt mittels eines WAR-Archivs auf einer Menge von Servern.
- Zur Leistungssteigerung werden die Anwendungen repliziert und Anfragen per Load-Balancer verteilt.



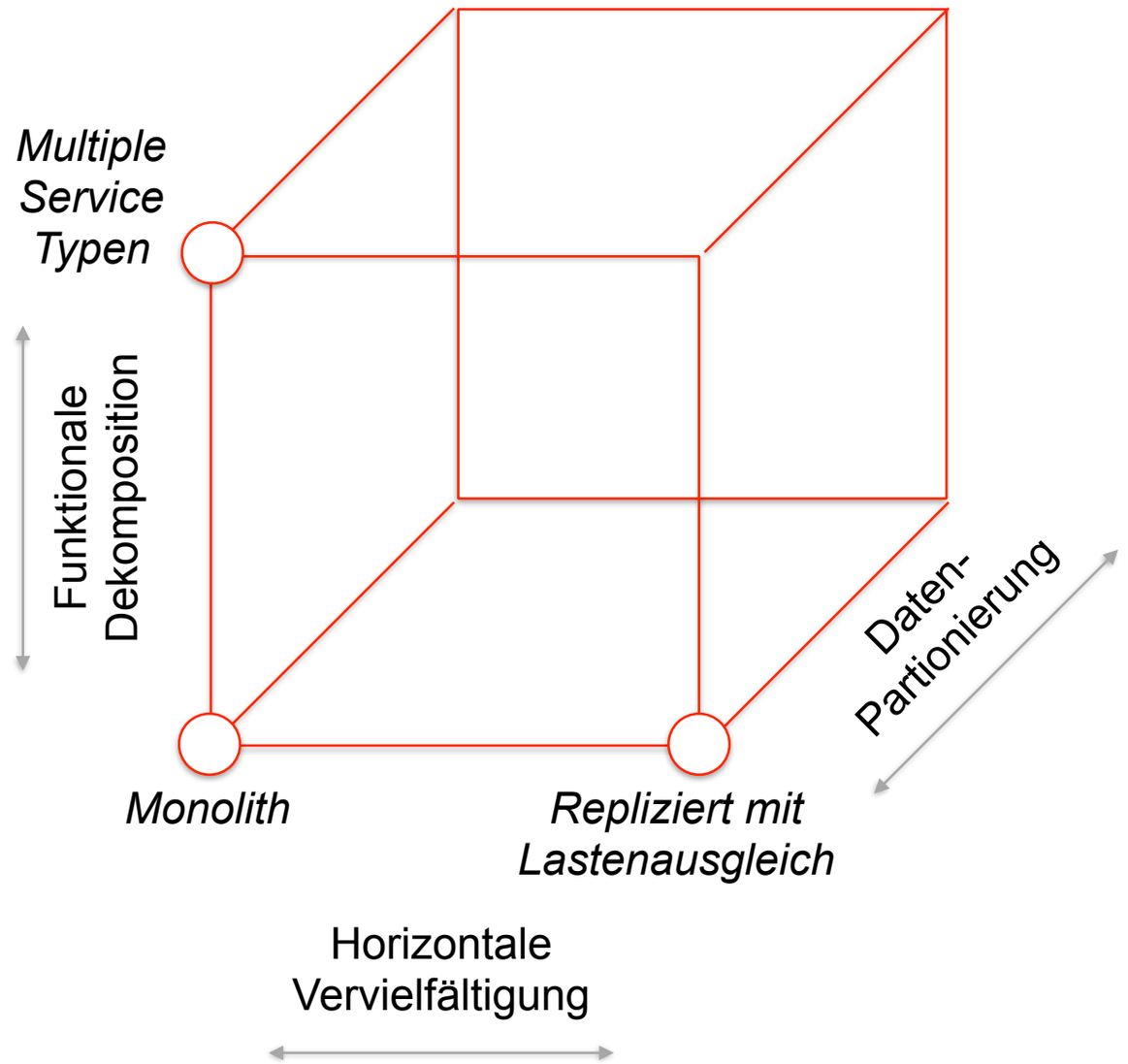
Monolithische Architektur

Eigenschaften monolithischer Architekturen

- Monolithische Architekturen sind vorteilhaft für die Entwicklung kleiner Anwendungen.
  - IDEs sind für die Entwicklung einzelner Anwendungen optimiert.
  - Zum Testen muss nur eine Anwendung gestartet werden.
  - Zum Deployment reicht das Kopieren einer Einheit (z.B. einer WAR-Datei) auf eine Maschine mit passendem Server.
- Sie stoßen aber bei großen, langlebigen Anwendungen an die Grenzen der Skalierbarkeit.
  - Komplexe Anwendungen sind schlecht verständlich und wartbar.
  - Bei häufigem Deployment muss immer der gesamte Monolith gebaut und installiert werden - auch für kleinste Änderungen.
  - Test und Verwendung neuer (Infrastruktur) Techniken erfordert meist massive Änderungen der ganze Anwendung.

Wie verteilte Anwendungen skalieren

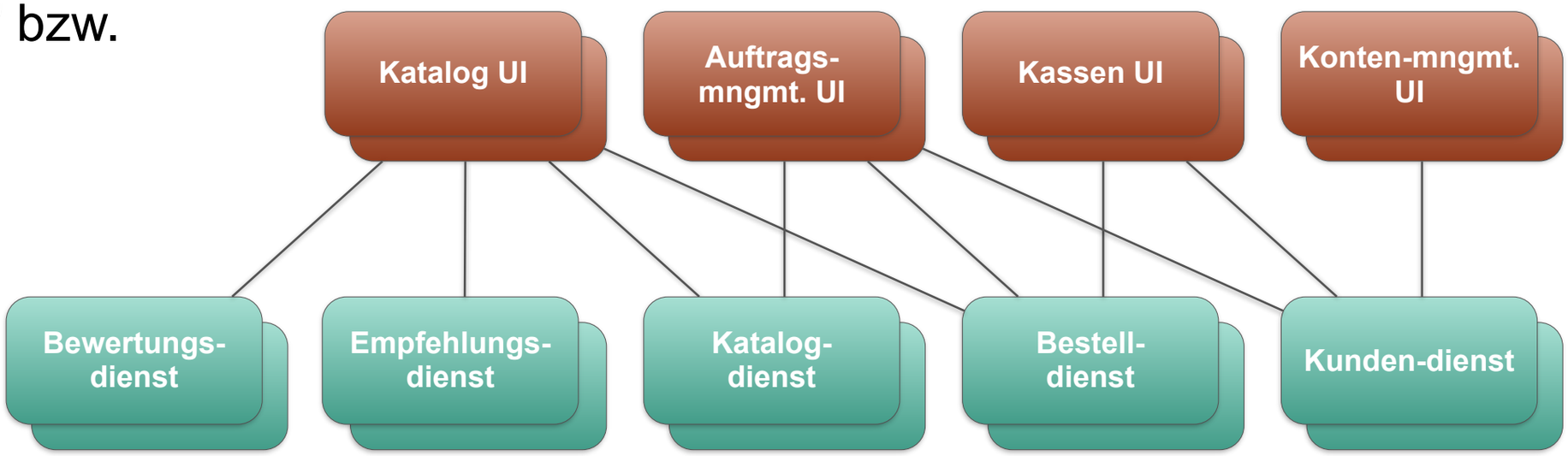
- Skalierung einer Anwendung durch mehrere identische Kopien hinter einem Load Balancer (X-Achsen Skalierung) erhöht deren Kapazität und Verfügbarkeit.
- Dies kann noch verstärkt werden, wenn die identischen Kopien jeweils nur für einen Teil der Daten zuständig sind (Z-Achsen Skalierung), z.B. basierend auf dem Primärschlüssel.
- Nur funktionale Dekomposition (Y-Achsen Skalierung), kann der Komplexität von Anwendungen und Entwicklungsprozessen Entgegenwirken.



"Scale Cube" aus *M. Abbott, M. Fisher, "The Art of Scalability", Pearson Education, 2009*

Dekomposition von Anwendungen in Services

- **Funktionale Dekomposition** dient der Teilung einer monolithischen Anwendung in eine Menge unabhängiger Services.
- Jeder Service implementiert eine kohäsive Menge von Anwendungsfunktionen (z.B. Order Management) oder Anwendungsdaten (z.B. Customer Service).
- Die Dekomposition einer monolithischen Anwendung in Services kann nach vielfältigen Kriterien erfolgen.
  - Z.B. auf Basis von "Verben" bzw. Anwendungsfällen,
  - oder auf Basis von "Substantiven" bzw. Ressourcen.



Microservice Architektur

Microservice Architekturen: *skalierbar und flexibel!*

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

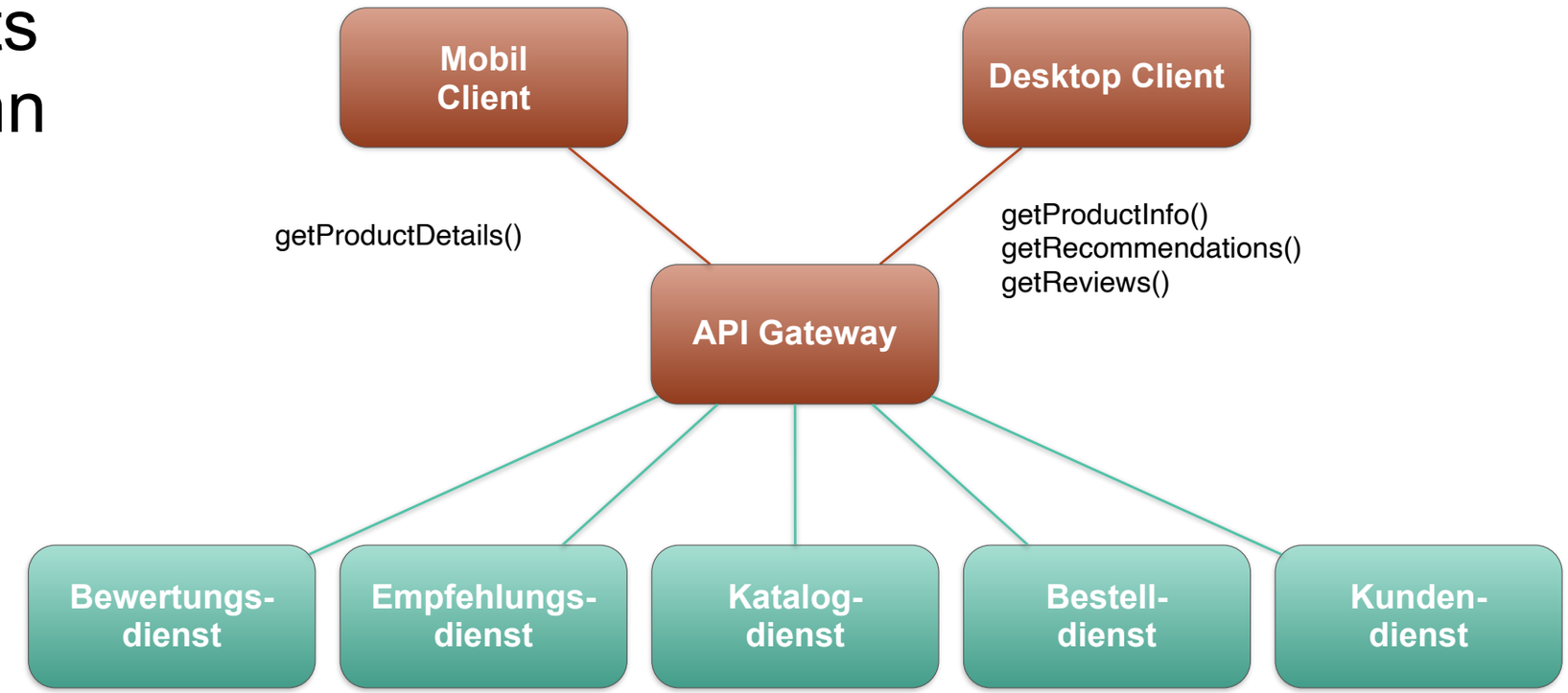
- Microservices sind relativ klein, überschaubar, in der IDE gut zu handhaben und für Tests schnell zu starten.
- Microservices erlauben individuelles Deployment unabhängig von anderen Teilen einer Anwendung.
- Durch Replikation oder Datenpartionierung kann für jeden Microservice eine individuelle Skalierung erfolgen.
- Microservices eines Anwendungsbereichs können bzgl. Entwicklung und Betrieb auf kleine Teams verteilt werden.
- Für einzelne Microservices können verschiedene Technologien wie Sprachen, Systemplattformen oder Hardware verwendet werden.

Microservice Architekturen: *komplex und aufwändig!*

- Mikroservice Architektur bedingt erhöhte Komplexität bei der Konstruktion verteilter Systeme - etwa Kommunikation und Datenmanagement zwischen Services.
- IDEs und Entwicklungswerkzeuge fokussieren Monolithen und auch das Testen mehrerer Services ist schwieriger.
- Der Betrieb von Anwendungen bedingt Automatisierung da viele Services überwacht und gesteuert werden müssen.
- Beim Deployment abhängiger Services müssen sich die Teams bezüglich der Reihenfolge abstimmen.
- Der richtige Zeitpunkt der Umstellung von monolithischen auf Microservice Architekturen ist oft schwer zu bestimmen.

Kommunikation mit Microservices

- Wie kommunizieren z.B. mobile Clients oder Web Browser mit den vielen APIs einer Microservice Anwendung?
  - Amazon z.B. benötigt z.T. 100+ Serviceaufrufe pro Seite!
- API-Gateways aggregieren Service APIs zur Effizienzoptimierung für verschiedene Client-Typen.
- Sie kapseln Service APIs zudem nach außen, so dass diese sich unabhängig von externen Clients ändern lassen, wenn das API-Gateway konstant gehalten wird.



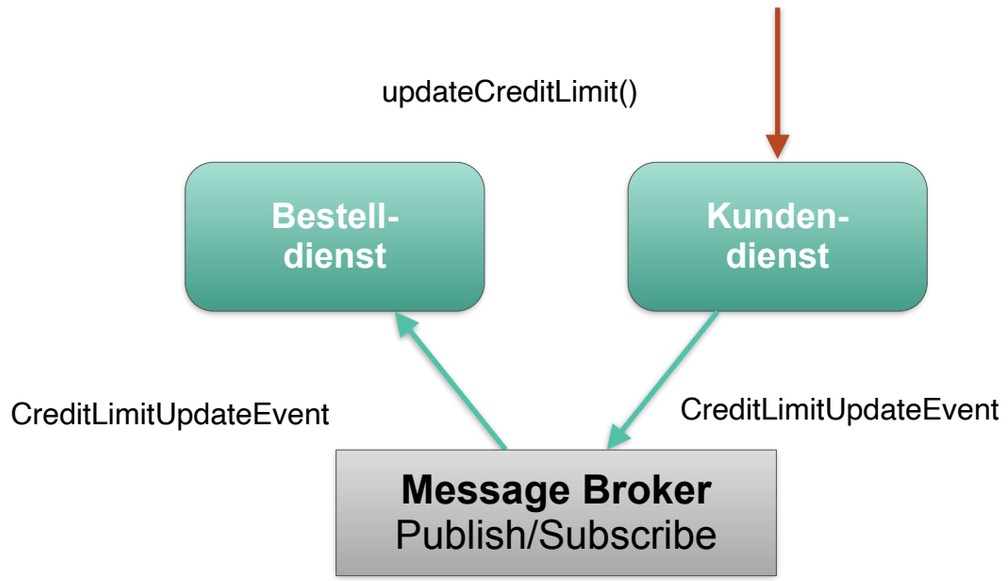
API Gateway Architekturmuster

Kommunikation zwischen Microservices

- Komponenten monolithischer Anwendungen nutzen reguläre Methodenaufrufe, Microservices hingegen Mechanismen zur Kommunikation zwischen Prozessen.
- In Microservice Architekturen kommen meist bewusst einfache Kommunikationsmechanismen zum Einsatz:
- Synchrone Mechanismen (REST/HTTP)
  - HTTP ist einfach, Internet-fähig, Firewall-freundlich und bildet synchrone Request-Reply Kommunikation direkt ab.
  - Die Kommunikationsmuster sind aber eingeschränkt und es müssen Adressen vermittelt werden (Service Discovery).
- Asynchrone Messaging Mechanismen (AMQP)
  - Nachrichten Queues entkoppeln Clients von Services und bieten vielfältige Kommunikationsmodi (zB Publish-Subscribe).
  - Sie benötigen aber meist weitere Komponenten (Queue bzw. Broker) und bilden Request-Reply Kommunikation nicht direkt ab.

Datenmanagement in Microservice Architekturen

- Die Zerlegung von Anwendungen in Microservices führt zu einer Partitionierung der Datenbasis.
- Zur Steigerung der Unabhängigkeit hat jeder Service seine eigene Datenbank und nutzt dabei ggf. individuelle Technologien (Polyglot Persistence).
- Ein Problem besteht in der Unterstützung von Anfragen, die Daten von mehreren Services zugreifen.
  - Microservices können Daten anderer Services lesen indem sie diese dynamisch abfragen oder als Kopie vorhalten.
  - Asynchrone ereignisgetriebene Updates vermeiden verteilte Transaktionen.
  - Services koordinieren Transaktionen als Anwendungsprozesse, nutzen verschiedene Kommunikations-muster und kompensieren im Falle von Fehlschlägen.



Ereignisbasierte asynchrone Updates

Diskussion und Ausblick

- Nach heutiger Erfahrung scheinen große Anwendungen, die skalieren müssen, etwa Web oder SaaS Anwendungen, von Mikroservice Architekturen zu profitieren.
- Bekannte Anbieter wie eBay, Amazon.com und Groupon haben ihre Anwendungen von monolithischen zu Mikroservice Architekturen entwickelt.

Wie geht die Entwicklung weiter?

Reaktive Architekturen: *…systems that are Responsive, Resilient, Elastic and Message Driven.* (**`www.reactivemanifesto.org`**)

Struktur — Einleitung und Überblick

Veranstaltung

- Team
- Lehrestruktur

Einleitung

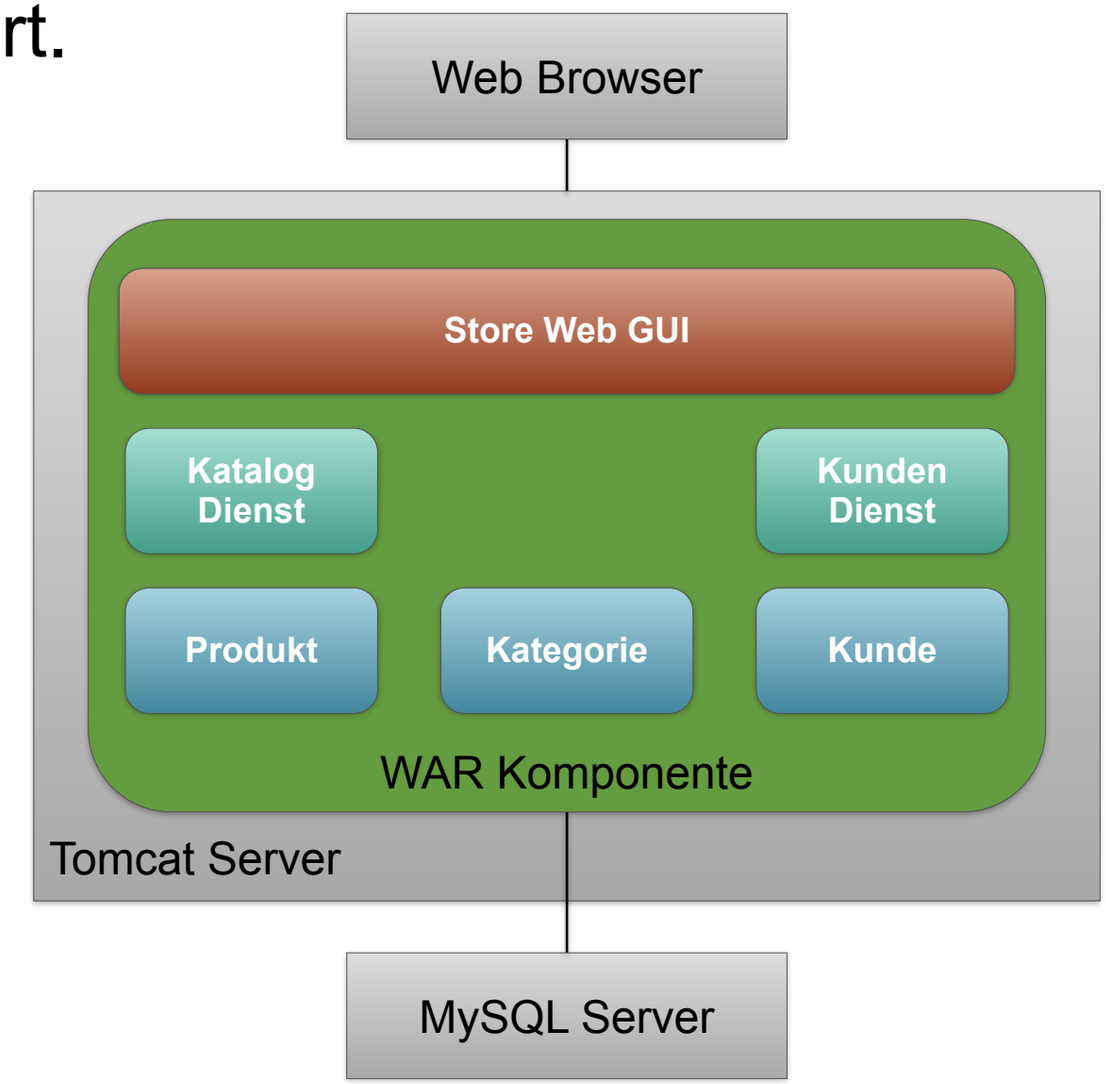
- Architekturen von Anwendungen auf Microservice-Basis

Überblick der Laborarbeit

- Aufgabenstellung
- Organisation

Hintergrund

- Ein Unternehmen bietet seine Produkte über das Internet in Form eines **elektronischen Shops (eShop)** an.
- Der Shop ist als *Webanwendung* mit *Struts2* und *Hibernate* implementiert.
- Die Anwendung ist als *Drei-Tier-Architektur* verteilt.
  - Web GUI und eShop-Funktionen sind nach *Model-View-Controller (MVC)* Muster auf dem Web-server zusammengefasst.
  - Eine relationale Datenbank läuft auf einem separaten Server und wird von der Anwendung über eine o*bjektrelationale Abbildung und JDBC* angebunden.



monolithischer eShop

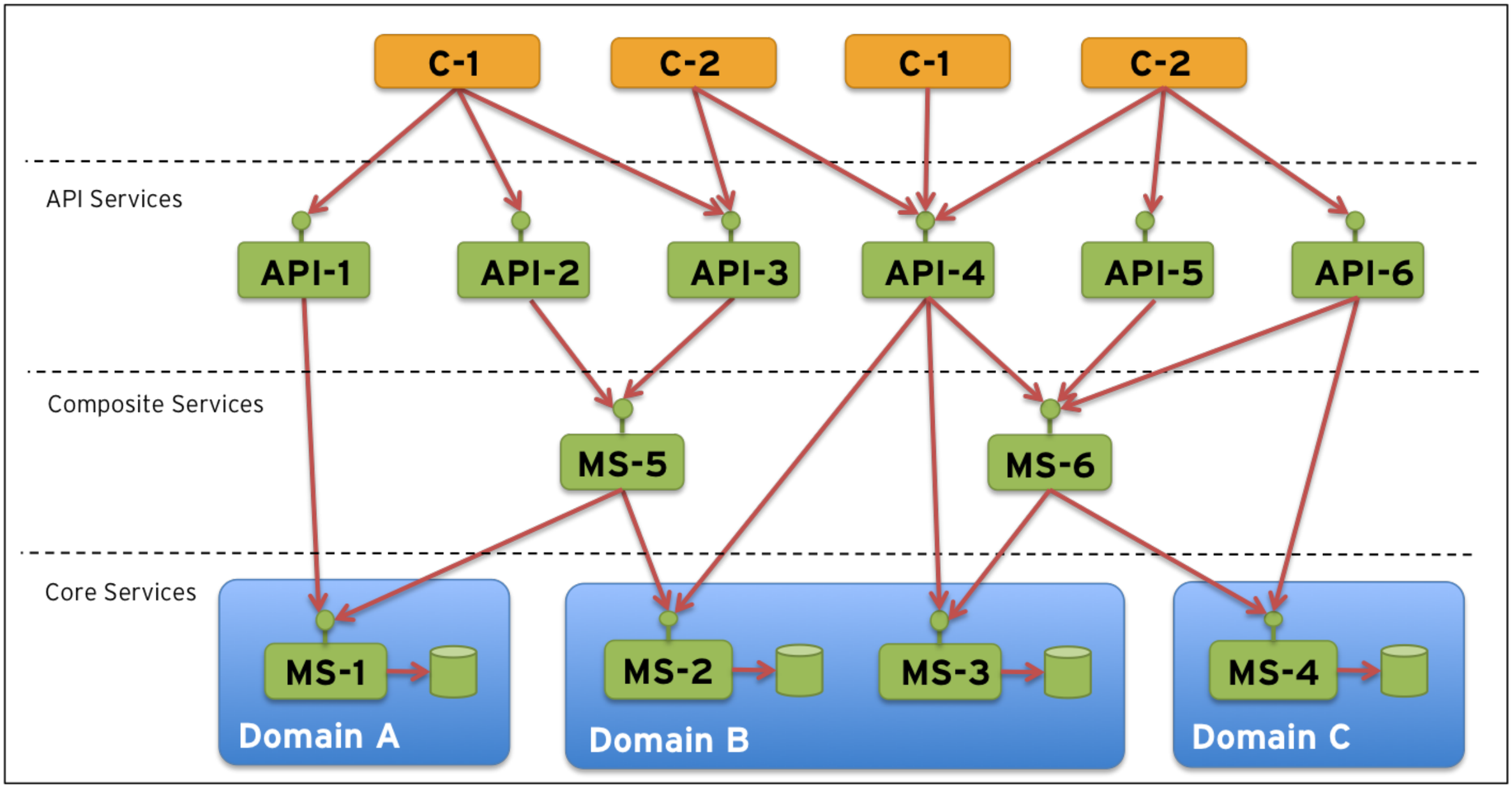
Aufgabenstellung

- Das Unternehmen möchte seinen eShop flexibler, robuster und leistungsfähiger machen. Hierzu soll ein **Refactoring als Microservice Architektur** vorgenommen werden.

Der Umbau des eShop umfasst folgende Schritte:

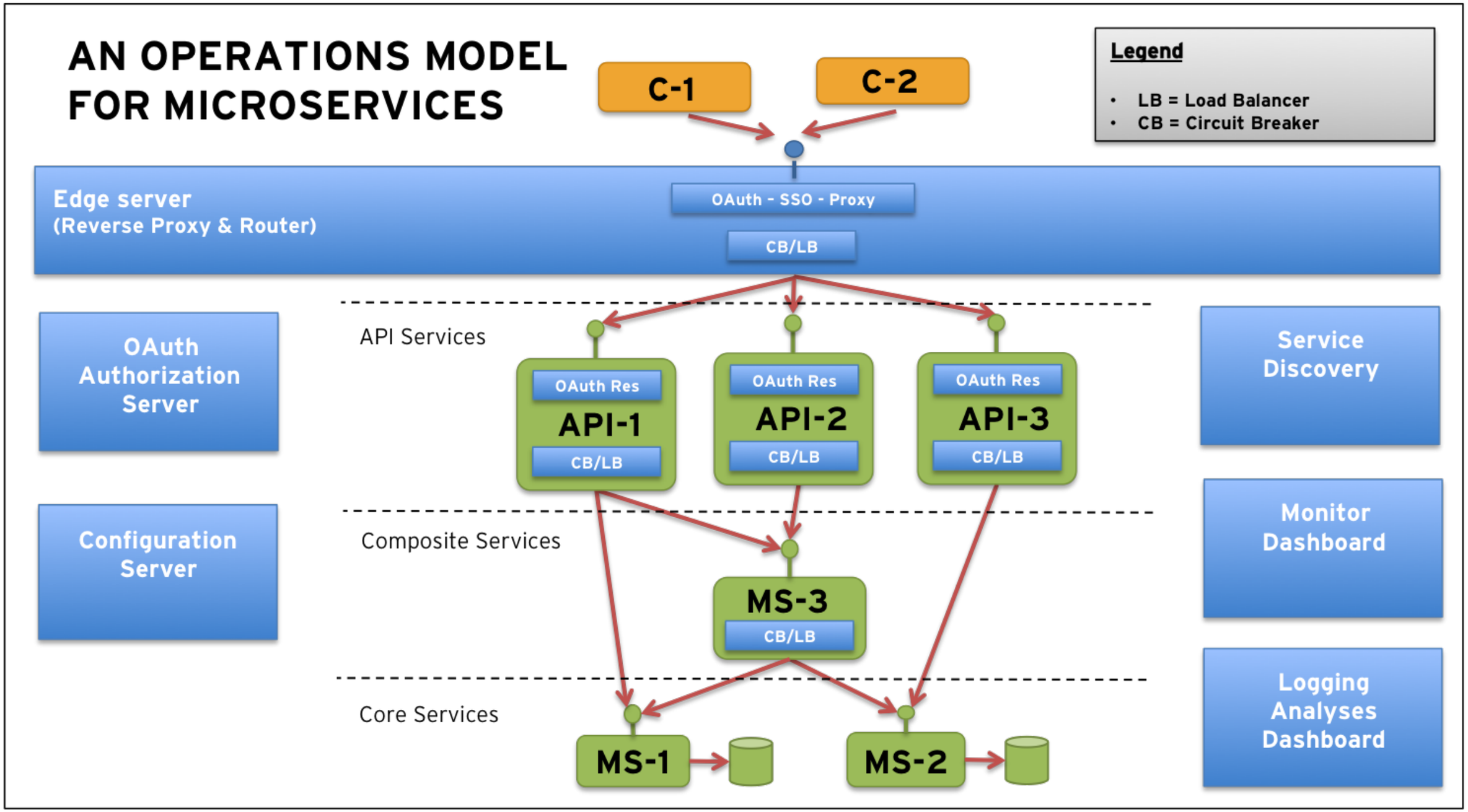
1. Analysephase
   - Test der bestehenden Anwendung und Analyse der momentanen Software Architektur
2. Entwurfsphase
   - Entwurf einer serviceorientierten Zielarchitektur und entsprechender REST Schnittstellen
3. Implementierungsphase
   - Konstruktion von Software Service Komponenten
   - Integration der Services über Middleware Dienste
   - Anbindung des eShop an die Service Infrastruktur

Serviceorientierte Zielarchitektur



Beispiel einer geschichteten Zielarchitektur (aus [Larson 2015a])

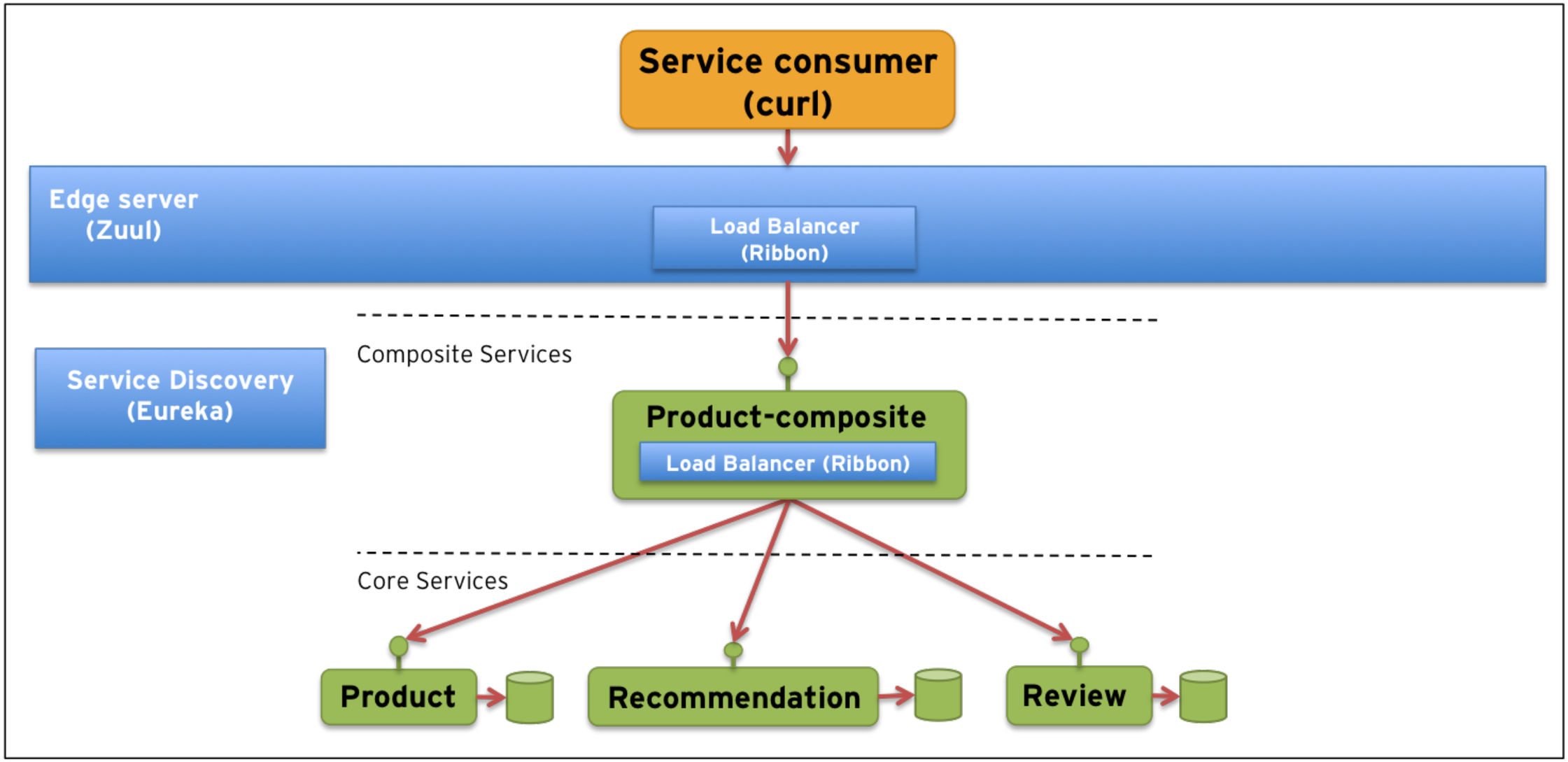
Microservice Infrastruktur



Typische Middleware Dienste einer Microservice Architektur (aus [Larson 2015a])

Technologien

- **REST-basierte Microservices**: *Spring MVC / REST templates*
- **Microservice Middleware**: *Spring Cloud, Netflix OSS*
- **Container Virtualisierung:** *Docker*



Beispiel: Microservice-basiertes Systems mit Netflix OSS Komponenten (aus [Larson 2015b])

Anforderungen

Anforderungen an die serviceorientierte Zielarchitektur

- Alle Anwendungsfunktionen und -daten sollen zu generischen **Core- und Composite Microservices** umstrukturiert werden.
- Zur Nutzung der Microservice Infrastruktur durch die eShop Web Anwendung sollen spezifische **API-Services** entworfen werden.

Hinweis: vollständige Spezifikation und Aufgabenstellung als PDF im ILIAS

Zeitplan

VERTEILTE SYSTEME MASTER LABOR - SOMMERSEMESTER 2020

				Präsenztermine jeweils Mi 8:00 Uhr im LKIT (LI137)	
#	**KW**	**Tag**	**Laboraufgabe**	**Termin**	**Abgabe**
1	*12*	*18.3.*			
2	*13*	*25.3.*			
3	*14*	*1.4.*	**A1+A2** Analyse der Legacy Anwendung und Microservice Architekturentwurf	**Seminar**: Microservice Architekturen (Mediacast)	
4	*15*	*8.4.*		**Seminar**: Web Shop Basistechnologien (Mediacast)	
5	*16*	*15.4.*		**Seminar**: Microservice Basistechnologien (Mediacast)	
6	*17*	*22.4.*		**Lab:** Analyse Legacy App / Microservice Entwurf	
7	*18*	*29.4.*		**Meetup**: Abgabe von Analysen und Entwürfen	System- und REST-Modelle
8	*19*	*6.5.*	**A3** Microservice Implementierung	**Seminar**: Microservice Middleware	
9	*20*	*13.5.*		**Lab**: Web Shop REST Microservices	
10	*21*	*20.5.*		**Lab**: Microservice Infrastruktur mit Netflix OSS	
11	*22*	*27.5.*		**Meetup**: Web Shop Microservices	Web Shop V2
	23	*3.6.*	*Pfingsten*		
12	*24*	*10.6.*	**A4** Web Shop Implementierung	**Seminar**: Microservice Security	
13	*25*	*17.6.*		**Lab**: Microservice Security mit OAuth	
14	*26*	*24.6.*		**Lab**: Frontend integration	
15	*27*	*1.7.*		**Meetup**: Web Shop V3	Web Shop V3

Ressourcen und Support

Foliensätze / Begleitmaterial etc. auf ILIAS

- **Anmeldung**

`http://bit.ly/2cSzube`

Unterlagen / Material

- Aufgabenblatt im ILIAS
- Demo Projekte auf Github laden

Email Support

- adelheid.knodel@hs-karlsruhe.de
- christian.zirpins@hs-karlsruhe.de

Zum Nach- und Weiterlesen

Literatur

[Newman 2015] Sam Newman, "Microservices - Konzeption und Design", mitp, 2015

[Wolff 2015] Eberhard Wolff,"Microservices, Grundlagen flexibler Softwarearchitekturen", dpunkt, 2015

Aus dem Web

[Fowler 2014] Martin Fowler, James Lewis, "Microservices", http://martinfowler.com/articles/microservices.html

[Larson 2015a] Magnus Larson,“An operations model for Microservices“, http://callistaenterprise.se/blogg/teknik/2015/03/25/an-operations-model-for-microservices/

[Larson 2015b] Magnus Larson, „Building microservices with Spring Cloud and Netflix OSS, part 1“, http://callistaenterprise.se/blogg/teknik/2015/04/10/building-microservices-with-spring-cloud-and-netflix-oss-part-1/

[Spring 2016a] Spring MVC, http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html

[Spring 2016b] Spring Cloud http://projects.spring.io/spring-cloud/

[Netflix 2016] Netflix Open Source Software Center, https://netflix.github.io