Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Vorlesung 9 – Bounded Context

Foto: Quirin Leppert

Themenübersicht

Data Integration

- Data Formats (csv, XML, json)
- Extract, Transform, Load
- Object Relation Mapper (ORM)
- Staging

Data Processing

- Relationale Datenbanken
- nicht-relationale Datenbanken
- Resource Description Framework (RDF)
- Ontologien
- Data Warehouse

Data Modelling

- Serialisierung
- OPC UA
- MQTT
- Pub/Sub
- **Data pipelines**
  - Apache Airflow
  - **gRPC**

Web-Service Architektur

- Front-End
- Backend for Frontend (BFF)
- **Micro Services**
- Docker Container
- **Bounded Context**
- **Domain-Driven Design**

Security

- Security ist wichtig in allen Phasen der Softwareentwicklung und Datenbereitstellung.

Übung

- Erstellung eines Daten-Modells einer prozesstechnischen Anlage
- Statische Daten
- Dynamische Daten
- Auswertung der Daten

Microservices – Literaturstudie

Domain-Driven Design

- Herangehensweise zur Modellierung komplexer Software
- Iterative Softwareentwicklung
- Enge Zusammenarbeit zwischen Entwicklern und Fachexperten

Bounded Context (Kontextgrenzen)

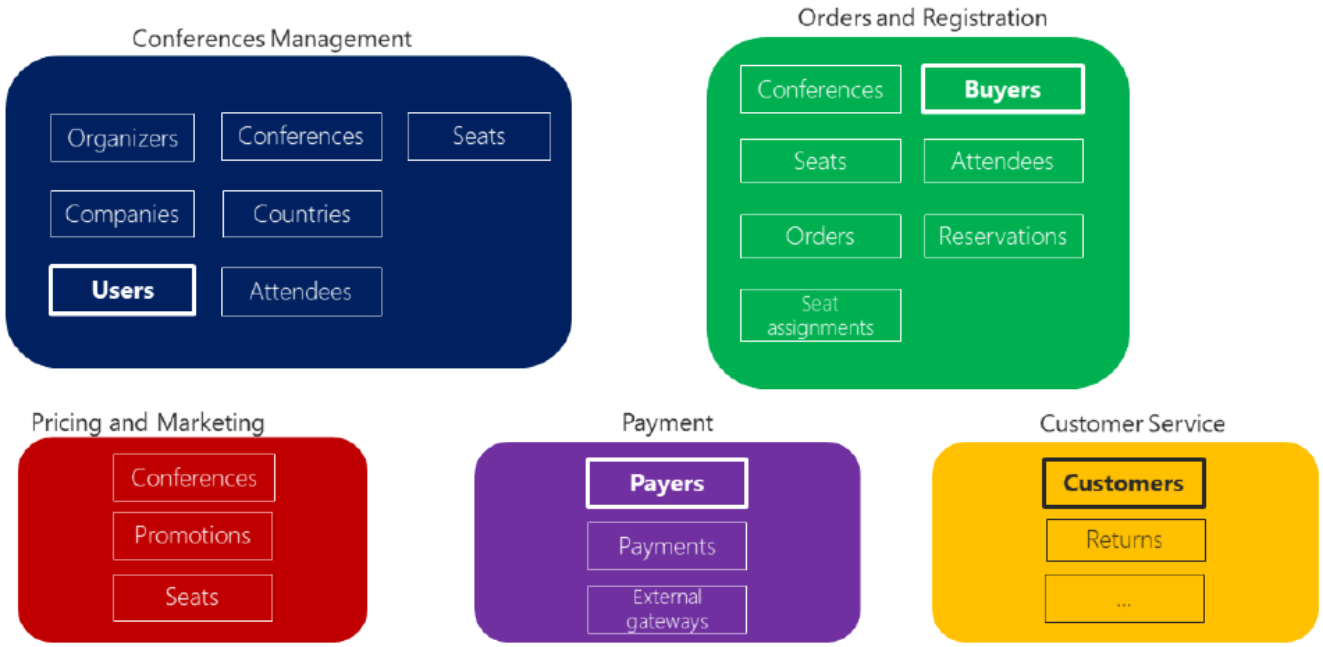
- beschreiben die Grenzen jedes Kontexts in vielfältiger Hinsicht wie beispielsweise Teamzuordnung, Verwendungszweck, dahinter liegende Datenbankschemata.

Quelle: https://dotnet.microsoft.com/download/e-book/microservices-architecture/pdf; https://de.wikipedia.org/wiki/Domain-driven_Design

Microservices – Literaturstudie

Definition der Grenzen von Microservices (Bounded Context)

+ Ein Ansatz besteht darin, Grenzen entlang von Business terms zu ziehen
+ Beispiel: Unterschiedliche Bezeichnungen für einen User
- User im Identitätsservice
- Customer in Customer-Relationship-Management-Service
- Buyer im Bestell-Kontext
+ Grenzen von Microservices können durch das Bounded Context pattern gefunden werden (Sam Newman)
+ Applikationen reflektieren die sozialen Grenzen der Organisiation, durch die sie entwickelt wurden (Conway‘s law).



Quelle: https://dotnet.microsoft.com/download/e-book/microservices-architecture/pdf

Microservices – Literaturstudie

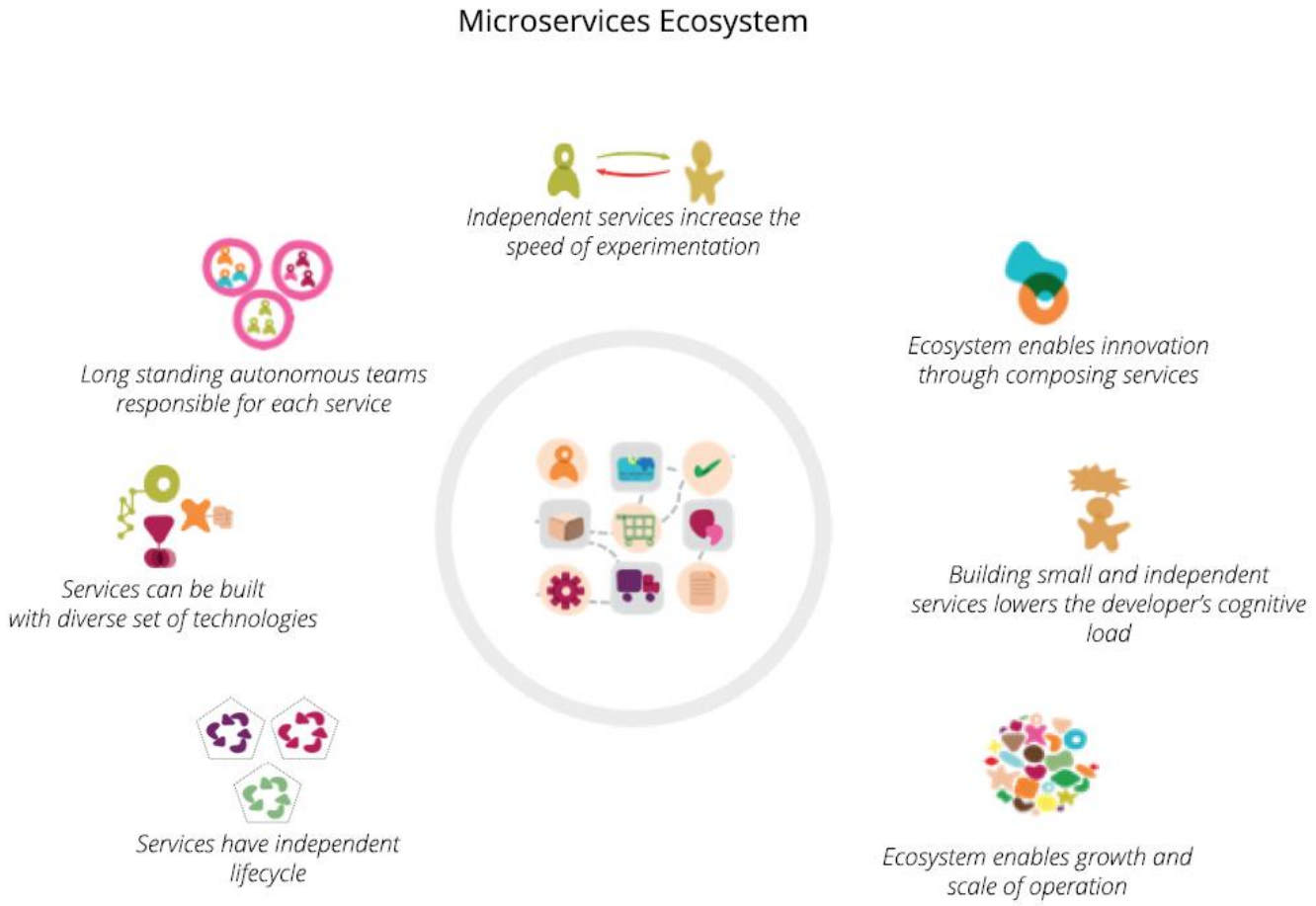
Literatur

- How to break a Monolith into Microservices; Zhamak Dehghani; 24.04.2018; https://martinfowler.com/articles/break-monolith-into-microservices.html, abgerufen am 19.11.2021
- Design of a reference architecture for developing smart warehouses in industry 4.0; van Geest et.al.; Computers in Industry 124 (2021) 103343
- Benchmarking Microservice Systems for Software Engineering Research; Zhou et.al.; 2018 ACM/IEEE 40th International Conference on Software Engineering: Companion Proceedings
- A model-driven approach for continuous performance engineering in microservice-based systems; Cortellessa et.al.; The Journal of Systems & Software 183 (2022) 111084
- Does Domain-Driven Design Lead to Finding the Optimal Modularity of a Microservice?; Vural et.al; IEEEAccess, Volume 9; DOI 0.1109/ACCESS.2021.3060895

Microservices – Literaturstudie

Literatur: How to break a Monolith into Microservices (1)

+ Die Autorin diskutiert die Frage, wie ein monolithisches Software-System in einzelne Microservices unterteilt werden kann.
+ Gründe für eine Aufteilung können
- Skalierbarkeit
- höhere Geschwindigkeit für Anpassungen an Änderungen und
- Reduzierung der Kosten für Änderungen sein.
+ Microservices ermöglichen den Einsatz parallel arbeitender Teams.
+ Die Autorin schlägt einen inkrementellen Ansatz zum Aufbrechen eines monolithisches Software-Systems vor.
+ Jeder Microservice kapselt eine Eigenschaft eines Geschäftsfelds.
+ Microservices bieten ein API an, die Lebenszyklen der Microservices sind unabhängig.



Microservices – Literaturstudie

Literatur: How to break a Monolith into Microservices (2)

- Die Aufteilung verursacht Kosten, viele Iterationen sind in der Regel notwendig.
- Am Anfang sollte die Auslagerung eines einfachen, klar getrennten Service stehen, z.B. die Authentifizierung.
- Die Abhängigkeiten zwischen Microservices sollen auf ein Minimum reduziert werden.
- Abhängigkeiten des neuen Microservices zum monolithischen Bestand sollten im monolithischen Teil über ein neues API ermöglicht werden.
- Neben der Bereitstellung von API durch die Microservices muss auch die Datenbankstruktur angepasst werden.
- Bestehender Sourcecode sollte nicht wiederverwendet werden. Der Sourcecode sollte mit Fokussierung auf die Eigenschaften/Fähigkeiten des Microservices neu erstellt werden.

Microservices – Literaturstudie

Literatur: Design of a reference architecture for developing smart warehouses in industry 4.0

- Die Autoren vergleichen die Architektur eines traditionellen Warehouses mit einem „Smart Warehouse“.
- Der Unterschied besteht in der Aufnahme moderner Hilfsmittel wie Tagging mit RFIDs, Bewegung von Gütern mit Automated Guided Vehicles (AGVs), Unterstützung der Mitarbeiter durch Augmented Reality (AR) und Anbindung externer Services wie z.B. Zustand des Straßenverkehrs.
- Die Aufteilung einzelner Bereiche (Decomposition View) wird diskutiert.

- Anmerkung: Die Feature/Packages/Sub-Packages des Smart Warehouses können als Unterteilung für Microservices (Bounded Context) genutzt werden.

Microservices – Literaturstudie

Literatur: Benchmarking Microservice Systems for Software Engineering Research

+ Die Autoren entwickeln eine öffentlich verfügbare Microservice-Architektur zur Nutzung als Benchmark-System.

Microservices – Literaturstudie

Literatur: A model-driven approach for continuous performance engineering in microservice-based systems

+ Einführung: Microservices stellen lose Kopplung sicher.
+ Microservices unterstützen Produktivität in der Entwicklung, Skalierbarkeit und Wartbarkeit.
+ Zur Handhabung der Softwarekomplexität kann Model-Driven-Engineering (MDE) genutzt werden.
+ Performanz, Energieverbrauch und Speicherbedarf sind relevant für Software-Applikationen.
+ Die Autoren stellen einen model-getriebenen Ansatz zur Entwicklung von Microservices unter Beachtung von Performance-Analysen vor.
+ Traces (logs) der Microservices werden gesammelt und analysiert (distributed tracing).
+ Performance antipattern werden eingeführt.
+ Continuous performance engineering loop besteht aus vier Schritten:
- Runtime data acquisition
- Design-runtime traceability generation
- Model analysis and refactoring
- System refactoring
+ Abschließend werden die Performance-Unterschiede ausgiebig diskutiert.

Microservices – Literaturstudie
Übungsaufgabe 13

Literaturstudie

+ Beantworten Sie zur Veröffentlichung *Does Domain-Driven Design Lead to Finding the Optimal Modularity of a Microservice?* folgende Fragen:

- Was bedeuten die Begriffe *Afferent Coupling* und *Efferent Coupling*?
- Welche Aussage machen die Autoren zur optimalen Granularität der Microservices?
- Case Study 1 bezieht sich auf Microservices für Frachtgut (cargo example). Beschreiben Sie die Unterschiede zwischen dem Original-Zustand (Abbildung 1) und den beiden Vorschlägen zur Abwandlung (MGM und LGM, Abbildungen 2 und 3).
- Welche Aufteilung würden Sie vornehmen? Begründen Sie ihre Entscheidung.

Hochschule Karlsruhe
University of
Applied Sciences

www.h-ka.de