Microservices im Zusammenspiel mit Domain-driven Design

-

Digicomp DevDay 2017

Arif Chughtai

Referent

- Arif Chughtai
  - IT-Consultant (Beratung, Schulung, Entwicklung)
  - mail@arifchughtai.org
  - http://www.arifchughtai.org

In Zusammenarbeit mit...

- Nicole Rauch
  - Softwareentwicklung & Entwicklungscoaching
  - info@nicole-rauch.de
  - @NicoleRauch
  - http://www.nicole-rauch.de

Inhalt

- Einleitung
- Microservices Einordnung
- Domain-driven Design (DDD) Einordnung
- Microservices mit DDD
- Ressourcen

Einleitung

Einleitung

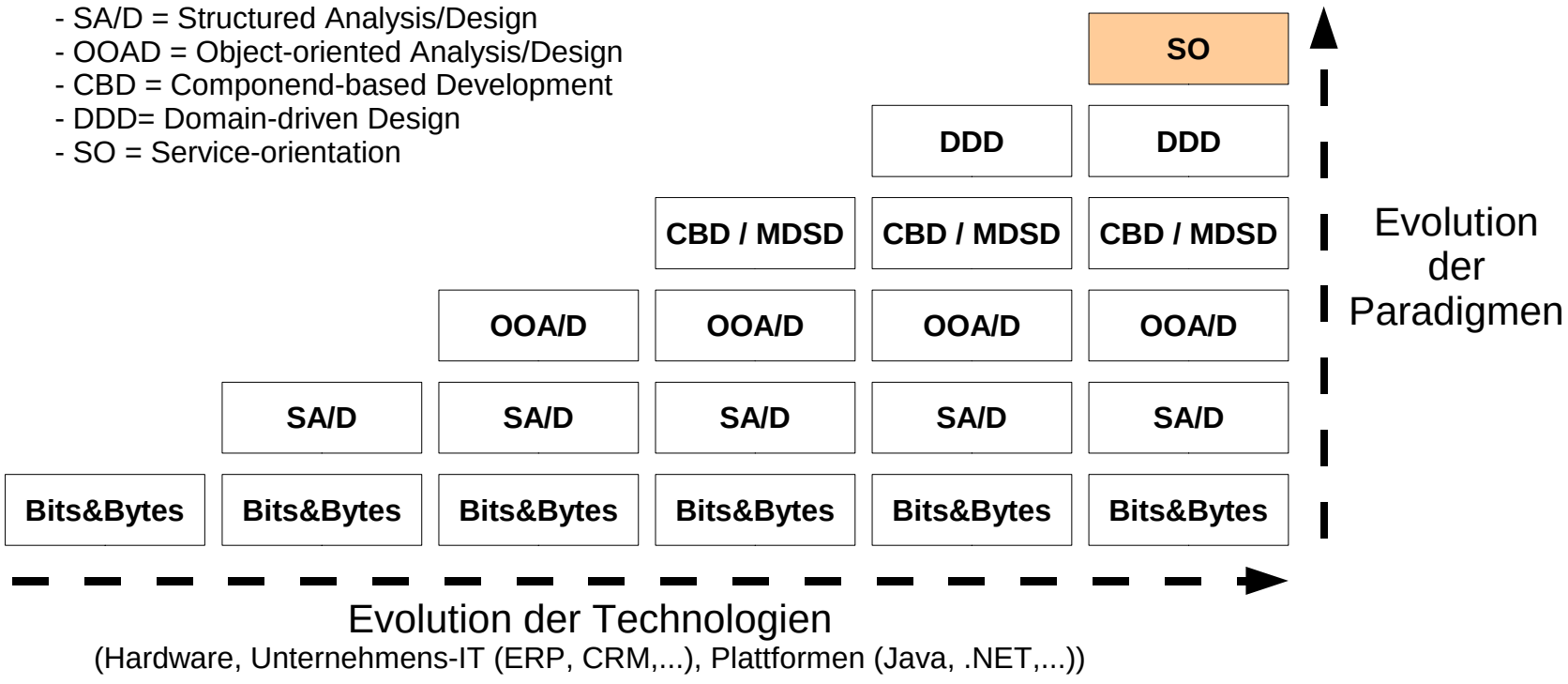
Alter Wein in neuen Schläuchen?

Einleitung

JA, aber...

Einleitung

- Evolution der IT
- Industrialisierung der IT



Einleitung

- Spreu vom Weizen trennen... prüfen, was neu ist
- Microservices und DDD ordnen und kombinieren Bekanntes, aber setzen andere/neue Schwerpunkte
  - Z.B. mehr Fokus auf Business/Fachseite

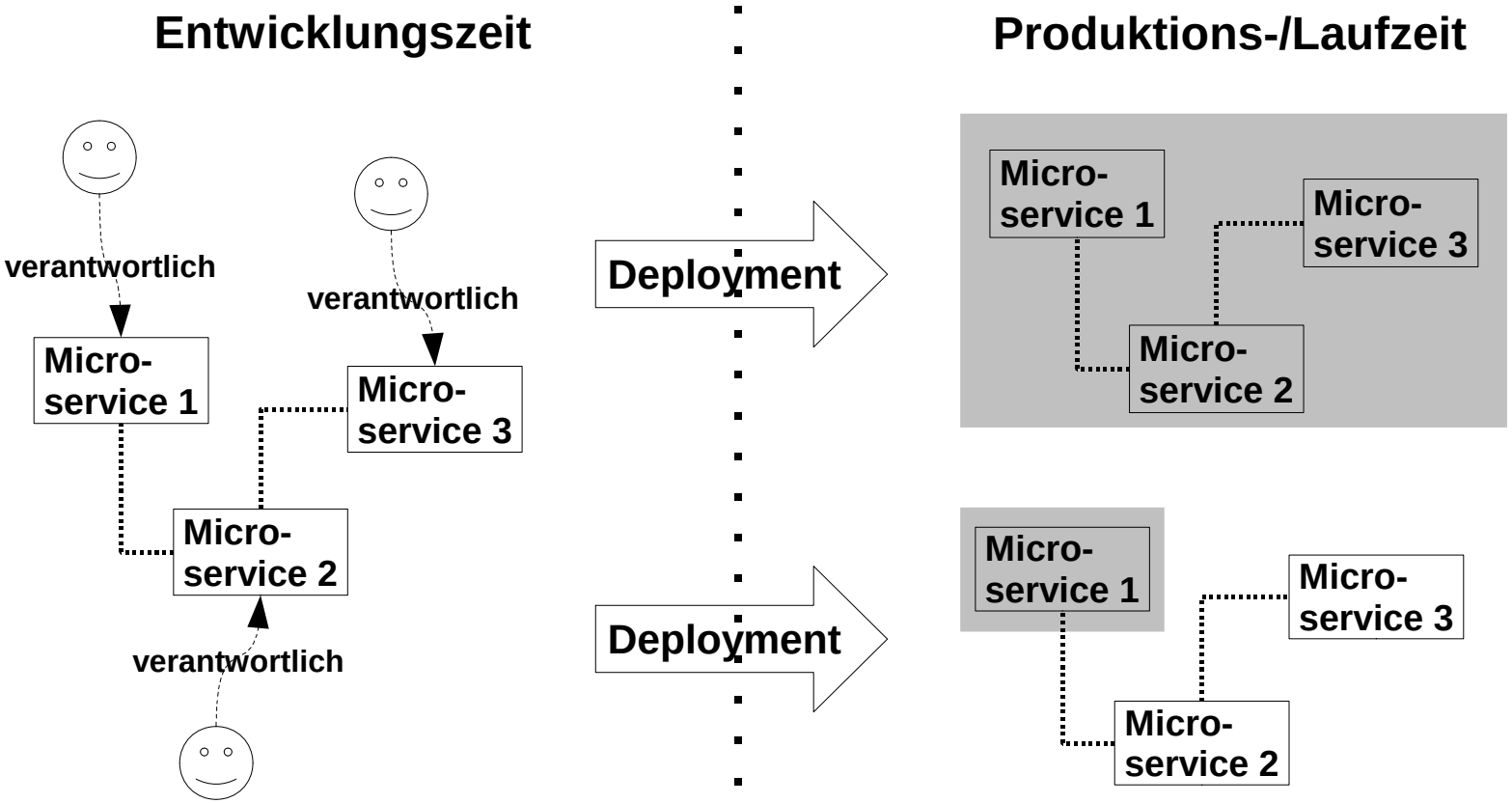
Microservices Einordnung

Microservices Einordnung

- Ansatz für Software-Entwicklung und -produktion
- Zweck: Modularisierung von Software
- Vieles nicht neu!
- Evolution der IT zur Serviceorientierung

Microservices Einordnung

- Neu: Modularisierung auch zur Produktionszeit
  - Unabhängiges Deployment statt Deployment-Monolithen



Microservices Einordnung

- Grund: Notwendigkeit für Industrialisierung der Software-Produktion (DevOps)
- Komplexe IT-Landschaften
- Komplexe Anforderungen
- Schnelle Umsetzung von Anforderungen/Innovationen

Microservices Einordnung

- Hypephase, Entwicklung ist im Fluss
- Haupteigenschaften
  - Lose gekoppelt
    - Strikte Trennung über Schnittstellen
    - Nach Fachlichkeit getrennte Entwicklung / unabhängige Teams
    - "Unabhängiges" Deployment
    - "Unabhängig" skalierbar
  - Eigener Technologiestack
  - Ersetzbar
  - Robust
  - Verteiltes System

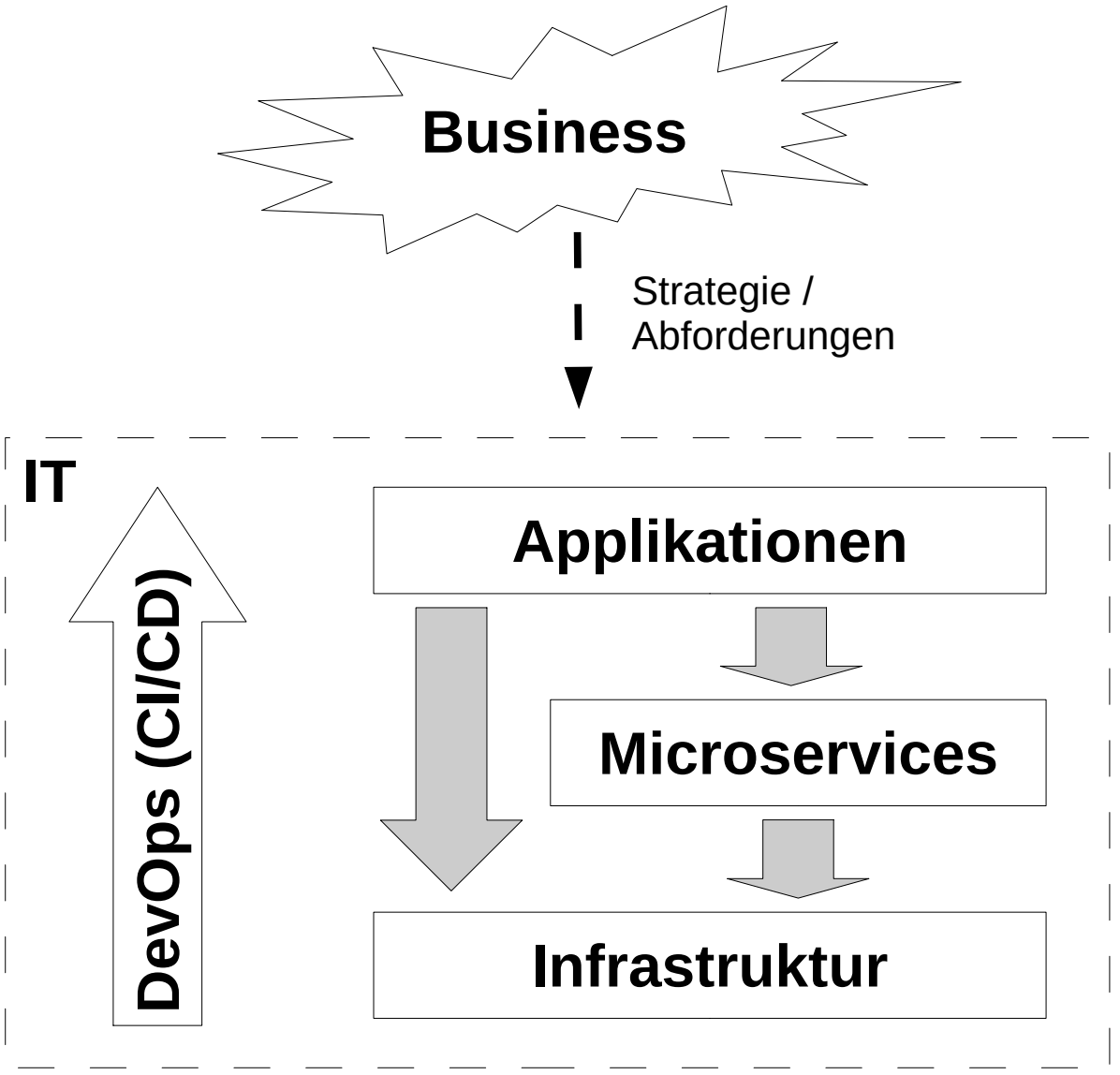
Microservices Einordnung

- Fachliche oder technische Verantwortlichkeit
- Granulariät
  - Kleinere ("Wegwerf"-Software, Nano-Services)
    - Einzelne Funktionen/Operationen (z.B. Online-Marketing)
  - Grössere (Reuse von Software)
    - Geschäftsfunktionalitäten (Netflix)
    - Self-Contained Systems (Integration von Webcontent)

Microservices Einordnung
Herausforderungen/Tradeoffs

- Unabhängigkeit der Teams
- Deployment und Test
- Modularisierung
- Ersetzbarkeit
- Organisationsstrukturen

Microservices Einordnung
Kontext

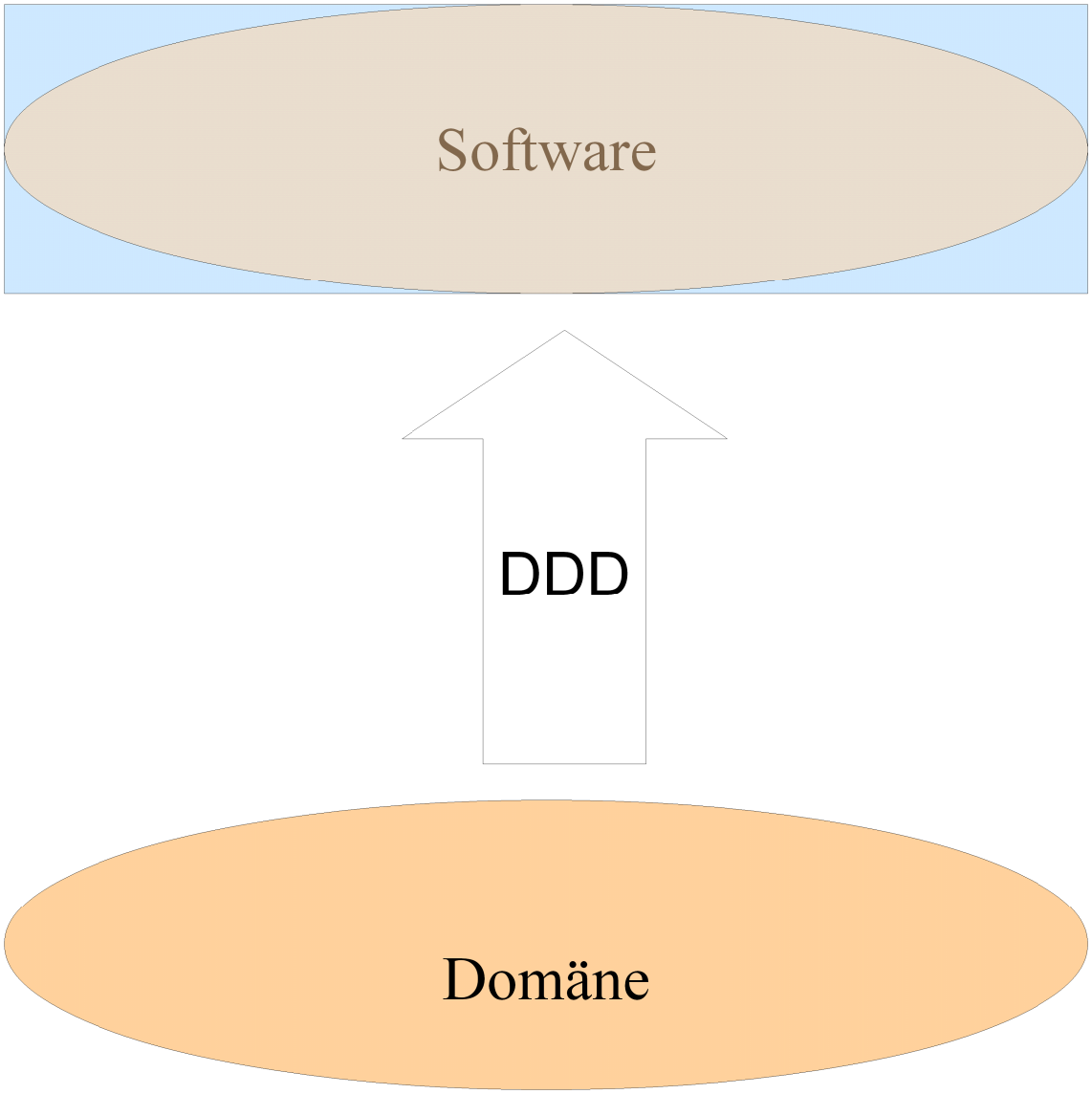


DDD Einordnung

DDD Einordnung

- DDD ist mehr als eine Sammlung von Patterns etc.
  - Denkschule
  - Bringt Ordnung in Konzepte und Techniken der Software-Entwicklung
    - Setzt diese in fachlichen Kontext
  - Etabliert Vokabular

DDD Einordnung
Ziele



DDD Einordnung
Ziele

- Tiefes Verständnis der Domäne
  - Reduzierte Komplexität
  - Besserer Business-Nutzen von Software (IT-Business-Alignment)
    - Technik (IT) folgt Domäne
- Geschmeidige Software als Ergebnis
  - Lose Kopplung, hohe Kohäsion, wenig Redundanz
  - Separation of Concerns
  - Bessere Flexibilität und Wartbarkeit

DDD Einordnung
Voraussetzungen für DDD

- Fachkompetenz
- Zusammenarbeit zwischen Domänenexperten und Entwicklern
- Lernbereitschaft aller Beteiligten
- Domäne ist ausreichend kompliziert
- Iteratives Vorgehen bei Modellierung und Implementierung
- Refactoring / Evolutionäre Entwicklung

Microservices mit DDD

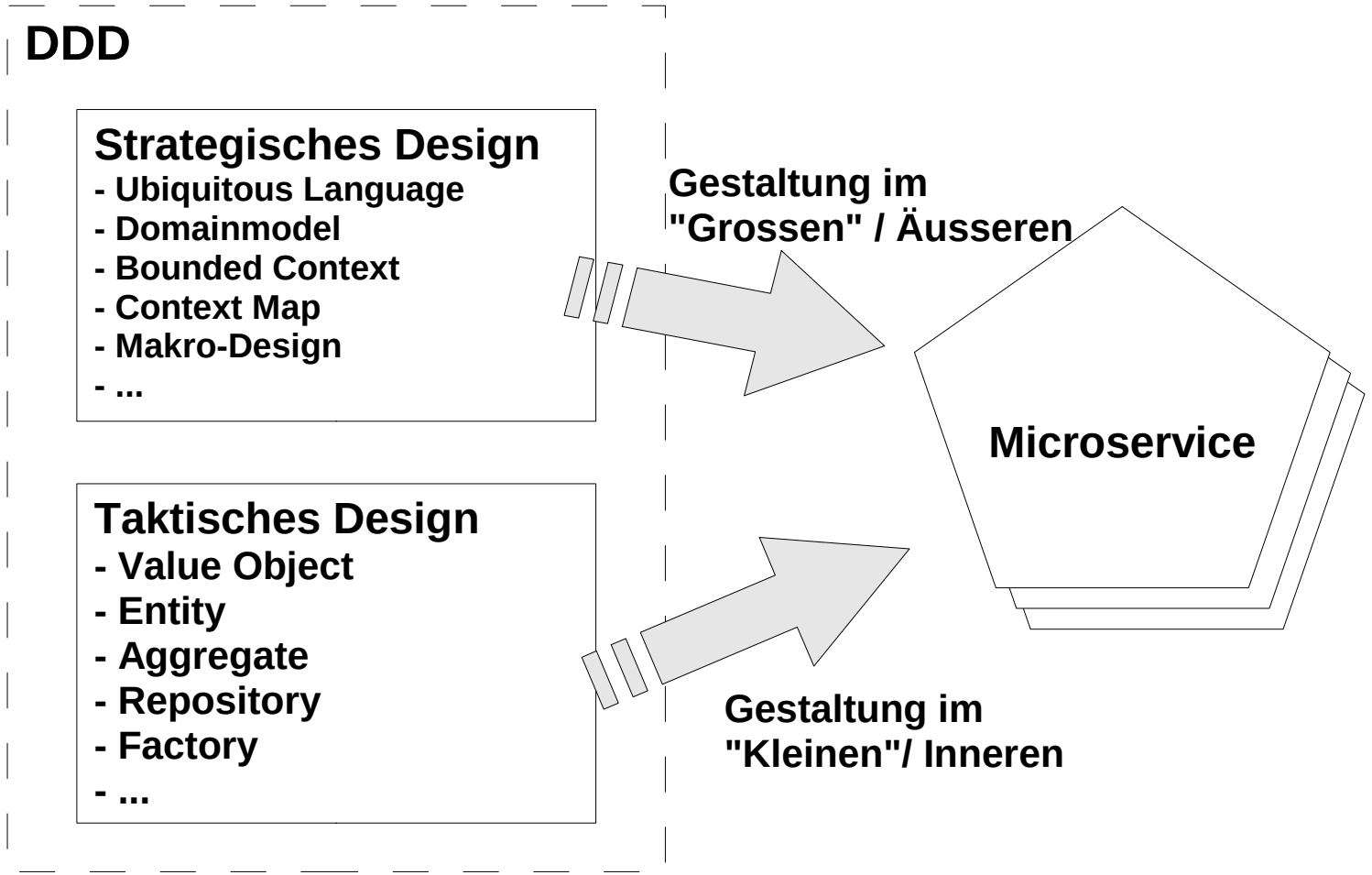
Microservices mit DDD

- Modularisierung muss organisiert werden
- Modularisierung muss zweckmässig sein
- Modularisierung alleine genügt nicht... was ist mit
  - Verständnis für Domäne?
    - Modell
    - Gemeinsame Sprache von Fachbereich und IT
  - Architektur?
    - Trennung von Domäne und Technik?
    - Nachhaltigkeit (Wartbarkeit,...)?

Microservices mit DDD

- DDD macht Domänen handhabbar
- DDD liefert Mittel für (zweckmässige!) "Zerlegung", Entwicklung und Wartung von Software
- Microservices erfordern fachliches "Zerlegen" einer Software

Microservices mit DDD



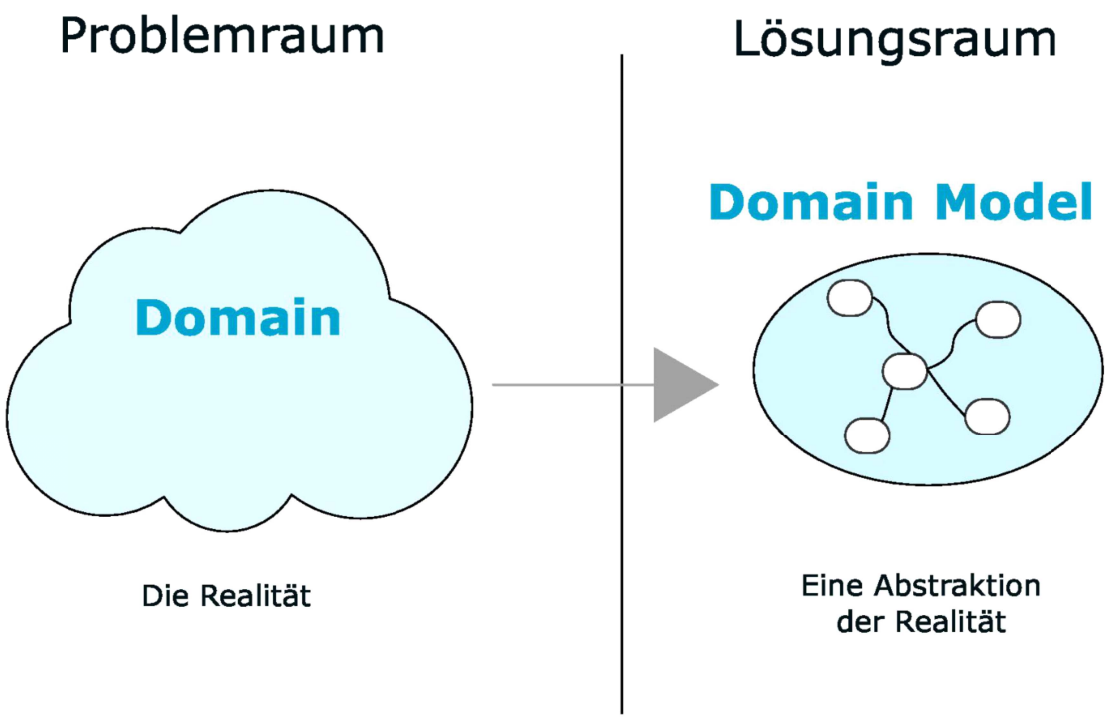
Microservices mit DDD: Strategisches Design
DDD-Mittel für Microservices

- Granularität von Microservices
  - Knowledge Crunching
  - Ubiquitous Language (gemeinsame Sprache)
  - Domain Model (Domänenmodell)
    - Destillation
    - Subdomain
    - Core Domain
    - Supporting Domain
    - Generic Domain
    - ...
  - Bounded Context

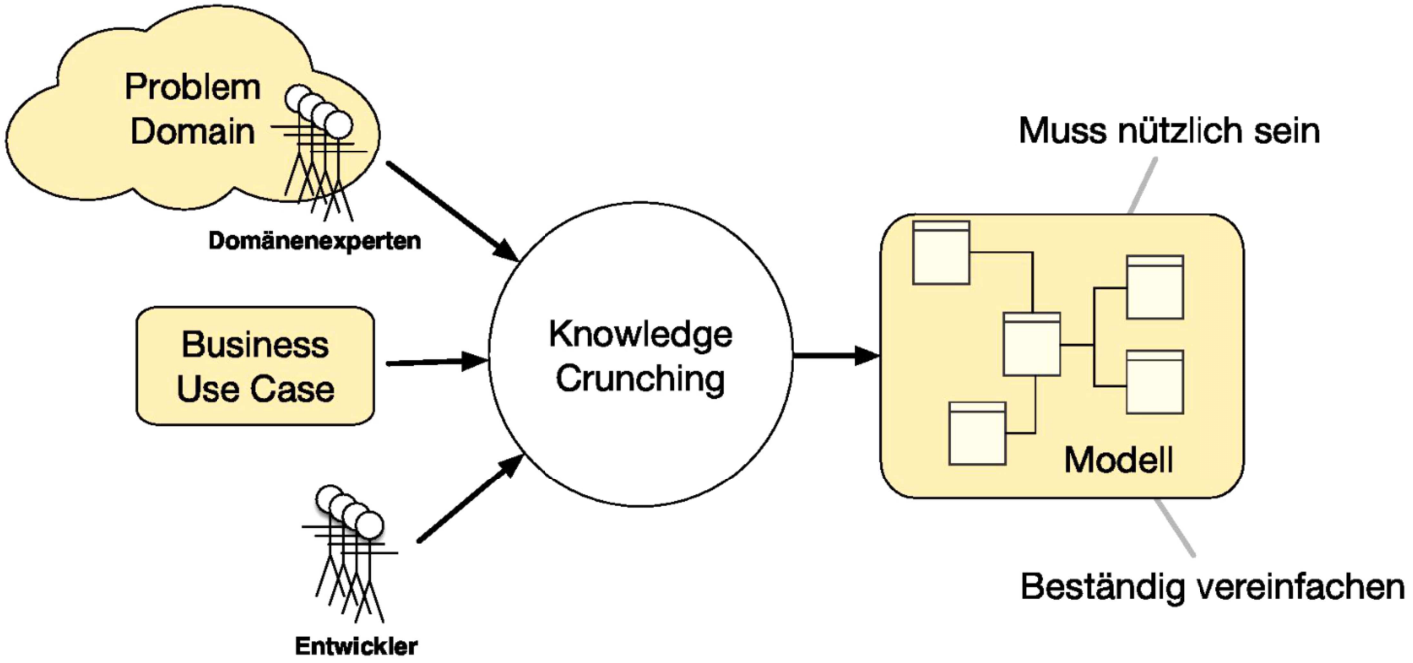
Microservices mit DDD: Strategisches Design
DDD-Mittel für Microservices

- Abhängigkeiten zwischen Microservices
  - Context Map
- Innerer Aufbau von Microservices
  - Internal Building Blocks (Taktisches Design)
  - ...
- Evolution von Microservices
  - Evolving Order
  - Responsibility Layer
  - ...

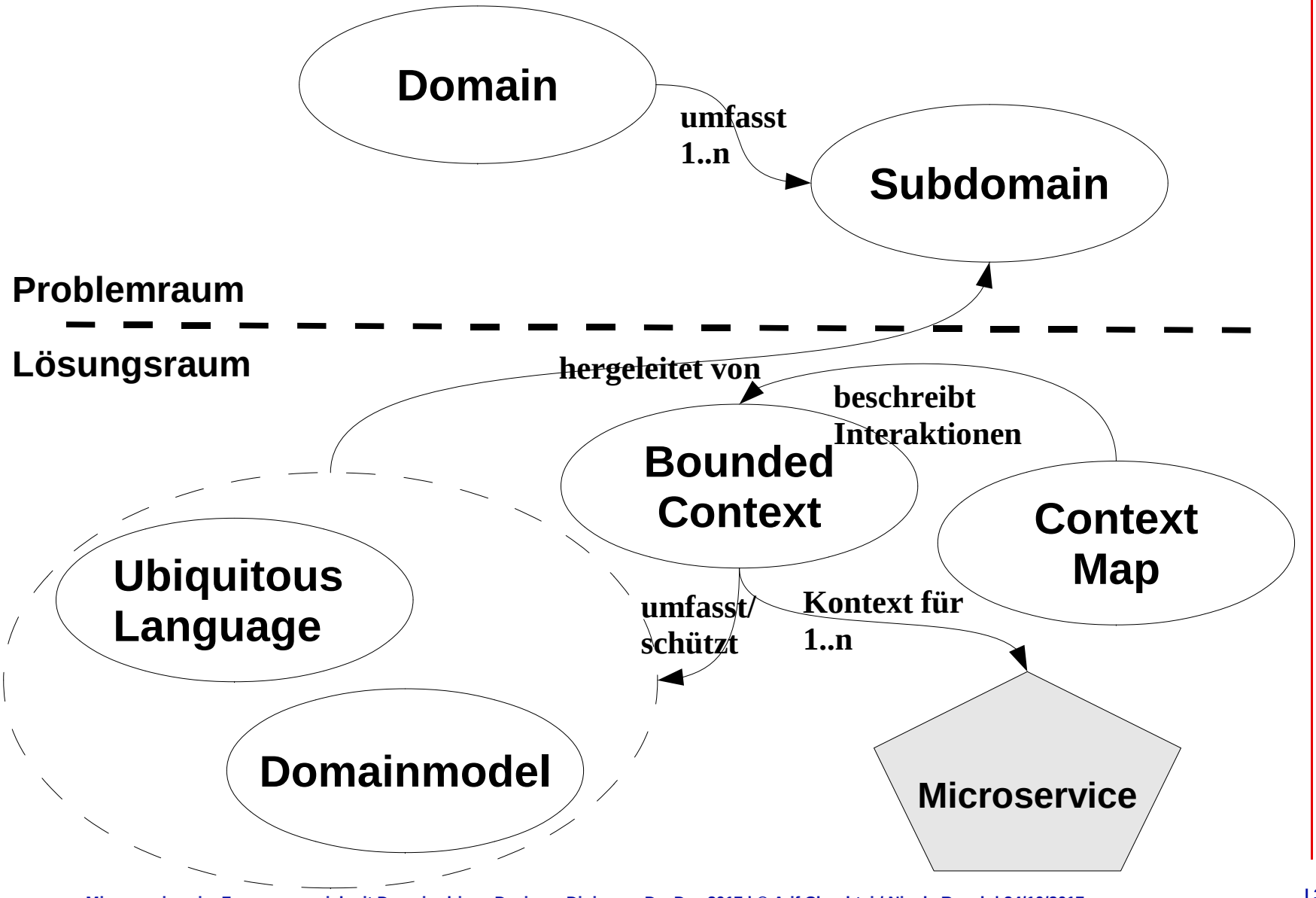
Microservices mit DDD: Strategisches Design
Problemraum/Lösungsraum



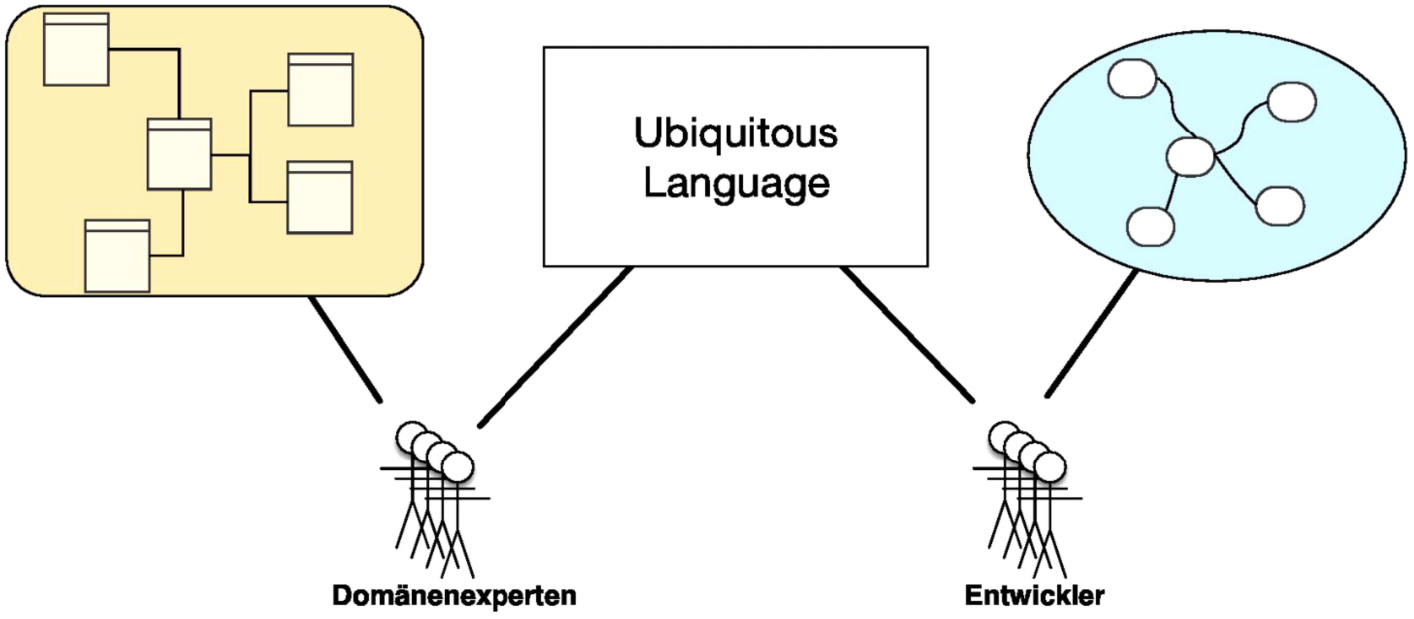
Microservices mit DDD: Strategisches Design
Knowledge Crunching



Microservices mit DDD: Strategisches Design
Domain, Ubiquitous Language,...



Microservices mit DDD: Strategisches Design
Ubiquitous Language



Microservices mit DDD: Strategisches Design
Ubiquitous Language

- Lebt in der Sprache des Teams und Arbeitsergebnissen
- Entsteht aus der Kollaboration zwischen Team und Domänenexperten
- Muss ständig verbessert und verfeinert werden
- Schafft Klarheit und Konsistenz
- Ohne technische Einflüsse
- Wird wahrscheinlich länger leben als die Software selbst
- Das Entwickeln der Ubiquitous Language ist der wichtigste Teil von Domain-driven Design

Microservices mit DDD: Strategisches Design
Ubiquitous Language

- Begriffe, die der Fachbereich standardmäßig verwendet, sind manchmal zu generisch
- Neue Begriffe müssen gemeinsam gefunden und definiert werden
- Auch bei der Implementierung können neue Konzepte entdeckt werden
- Diese impliziten Konzepte müssen explizit gemacht und benannt werden
- Die Ubiquitous Language muss allgegenwärtig sein – auf allen Abstraktionsebenen (Analyse, Design/Code) und in Gesprächen
- Technische Begriffe und irrelevante Fachbegriffe müssen draußen bleiben

Microservices mit DDD: Strategisches Design
Domänenmodell

- Was: Herausdestillieren der Domäne
- Warum:
  - Zum Entwickeln eines Verständnisses für die Fachlichkeit
  - Zum Schließen von Wissenslücken
- Wie: In enger Zusammenarbeit mit den Domänenexperten:
  - Product Owner / Stakeholder
  - Fachexperten / Fachlich versierte Nutzer der Software
  - Kennen die Regeln, Arbeitsabläufe, die Lästigkeiten und
  - Besonderheiten der Domäne

Microservices mit DDD: Strategisches Design
Domänenmodell

- Auf das Wichtigste fokussieren
- Man kann nicht überall gleich viel Energie investieren und muss dasauch nicht
- Dasselbe gilt für die Softwarequalität
- Distillation erlaubt die wertvollsten Bereiche zu erkennen

Microservices mit DDD: Strategisches Design
Domänenmodell

- Ein Modell für die gesamte Domäne ist oft...
  - viel zu groß, zu abstrakt, zu nichtssagend
- Daher: Domain in Subdomains zerlegen
  - Z.B. Domain Online-Handel mit Subdomains Bestellungen, Lagerverwaltung, Produktekatalog etc.
- Subdomains
  - Beschreiben Geschäftsprozesse
  - Repräsentieren die Funktionalitäten eines Systems
- Technische Aspekte außen vor lassen
- Zerlegung reduziert Komplexität (Divide and Conquer)
- Für jede Subdomain kann ein Modell erstellt werden

Microservices mit DDD: Strategisches Design
Domänenmodell: Core Domain

- Wie erhalten?
  - Welche Teile des Produkts werden ihm zum Erfolg verhelfen?
  - Warum sind diese Teile wichtig?
  - Warum kann man sie nicht von der Stange kaufen?
- Strategien
  - Die besten Entwickler damit beauftragen
  - Hier muss die meiste Energie investiert werden
  - Die Core Domain wie ein Produkt behandeln (nicht wie ein Projekt)
  - Entwicklung ist niemals abgeschlossen
  - Muss flexibel und erweiterbar sein

Microservices mit DDD: Strategisches Design
Domänenmodell: Generic Domains

- Viele Subdomains sind generisch
  - Z. B. Mailversand, Rechnungsstellung, Reporting
- Nicht sinnvoll, hier viel zu investieren
- Kaufen oder von Junior-Entwicklern erstellen lassen

Domänenmodell: Supporting Domains

- Supporting Domains unterstützen die Core Domain
- Sind notwendig, tragen aber nicht zum fachlichen Kern bei
- Hier ebenfalls nicht viel investieren
- Nach Möglichkeit fertige Lösungen kaufen
- Kann evtl. auch durch manuellen Prozess gelöst werden

Microservices mit DDD: Strategisches Design
Bounded Context

- Domänen umfassen oft mehrere Modelle
- Jedes Domänenmodell verkörpert einen bestimmten Bereich der Domäne
  - Z.B. Domain Online-Handel mit Subdomains Bestellungen, Lagerverwaltung, Produktekatalog etc.
  - Einheitsmodelle führen zu unnötigen organisatorischen und technischen Anhängigkeiten
- Jede Implementierung nutzt die am besten geeigneten Methoden und Patterns
- Idealerweise ein Domänenmodell pro Subdomain (passt nicht immer)

Microservices mit DDD: Strategisches Design
Bounded Context

- Domänenmodelle müssen miteinander kommunizieren
- Gefahr der Vermischung und Verwässerung
- Wichtig: Integrität jedes Domänenmodells bewahren
- Klare Verantwortlichkeitsgrenzen definieren
- Wird erreicht durch Binden eines Domänenmodells an seinen Kontext, den “Bounded Context”
  - Logische Klammer
- Setzt Sprachgrenze
- Implementation findet im Bounded Context statt
  - Jede Implementierung nutzt die am besten geeigneten Methoden, Techniken und Technologien

Bounded Context

Warum nicht ein einziges Domänenmodell?

- Verwechslungsgefahr, wenn es ähnliche Konzepte in verschiedenen
- Bereichen gibt
- Versehentliche Kopplung dieser ähnlichen Konzepte
- Enthält viele Fachkonzepte und Use Cases
- Komplexität ist viel höher
- Es ist leichter, Fehler zu machen
- Mehr Overhead bei der Kollaboration über Teamgrenzen hinweg

Bounded Context

Vorteile mehrerer Domänenmodelle!

- Unabhängigkeit mehrerer Entwicklungsteams
- Teams arbeiten effektiver, da isoliert
- Abhängigkeiten zwischen Modellteilen sind aufgehoben
- Duplizierte Elemente sind in Wirklichkeit nicht dupliziert (unterschiedliche Konzepte)

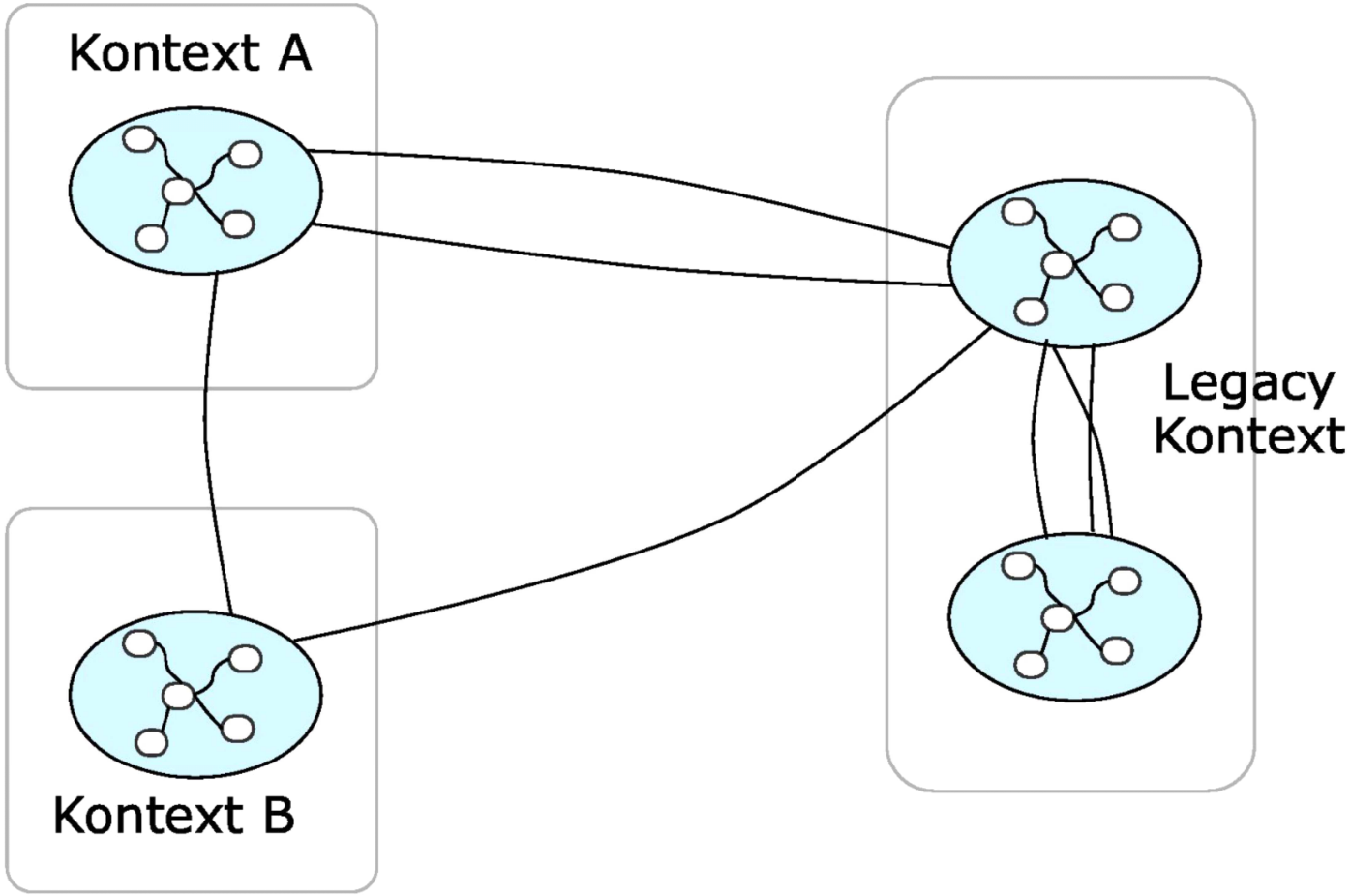
Microservices mit DDD: Strategisches Design
Bounded Context

- Ein Bounded Context enthält die zugehörige Funktionalität quer durch alle Teile einer Applikation:
  - Präsentation (UI)
  - Domänenlogik
  - Persistenz
  - Datenbank
- Jeder Bounded Context kümmert sich um alle diese
- Verantwortlichkeiten selbst
- Jeder Bounded Context enthält quasi ein “Mini-Produkt”

Microservices mit DDD: Strategisches Design
Bounded Context

- Unerwünschte Effekte auf andere Bounded Contexts sind ausgeschlossen
- Konzepte, die in mehreren Bounded Contexts vorkommen, werden mehrfach getrennt implementiert
- Jeder Bounded Context verwendet das geeignetste Architektur-Pattern (OO, Layer, Hexagon, CRUD,...)

Microservices mit DDD: Strategisches Design
Context Map



Microservices mit DDD: Strategisches Design
Context Map

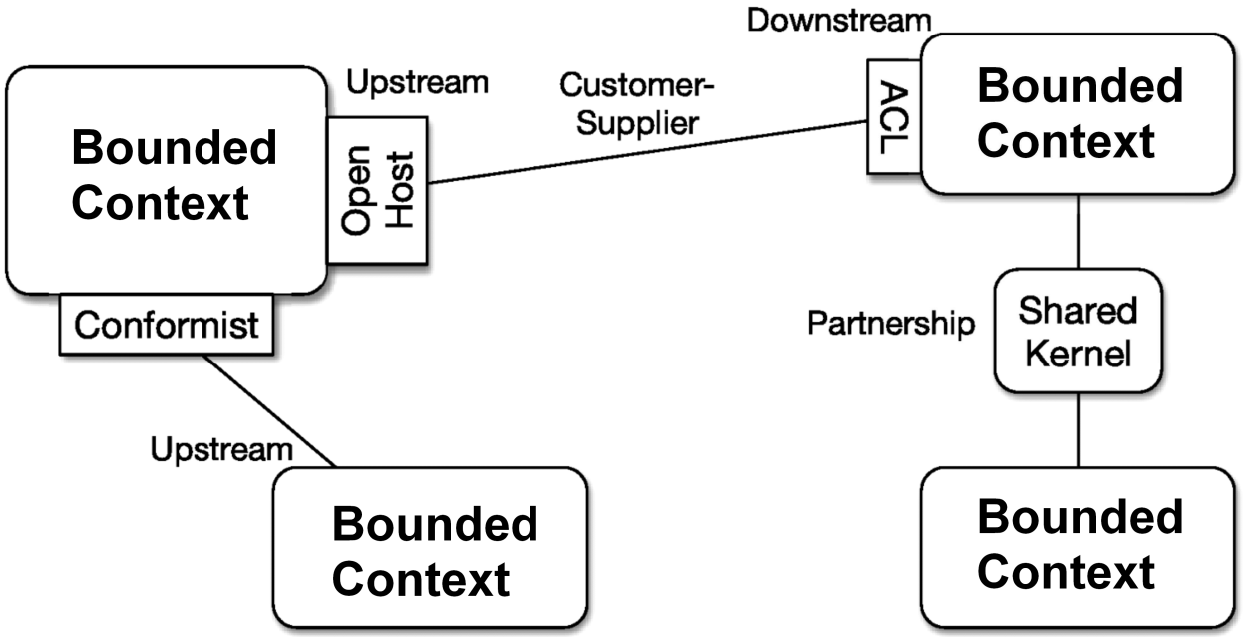
- Teams brauchen einen Überblick über die Bounded Context und ihre Beziehungen
- Domänenmodelle der Kontexte kollaborieren
- Technische Details müssen bekannt sein, damit sie berücksichtigt werden können
- Auch organisatorische Einflüsse sind wichtig zu verstehen
- Manchmal gibt es Funktionalität “zwischen den Kontexten”, oft ohne Verantwortliche
- Context Map stellt diese technischen und organisatorischen Beziehungen zwischen den Bounded Contexts dar

Microservices mit DDD: Strategisches Design
Context Map

- Beschreibt den Istzustand des Software (so hässlich er sein mag)
  - Organisatorisch und Technisch
- Definiert die Grenzen zwischen Bounded Contexts
- Beschreibt die Berührungspunkte verschiedener Bounded Contexts
- Details der Domänenmodelle selbst sind irrelevant
  - High-Level Darstellung ohne unnötige Details
- Einfach genug, um von Entwicklern und Domänenexperten verstanden zu werden
- Stellt klar, welche Bereiche von den Teams gut verstanden werden und welche nicht

Microservices mit DDD: Strategisches Design
Context Map: Beziehungsmuster

- Beispiele



Microservices mit DDD: Strategisches Design
Context Map: Beziehungsmuster

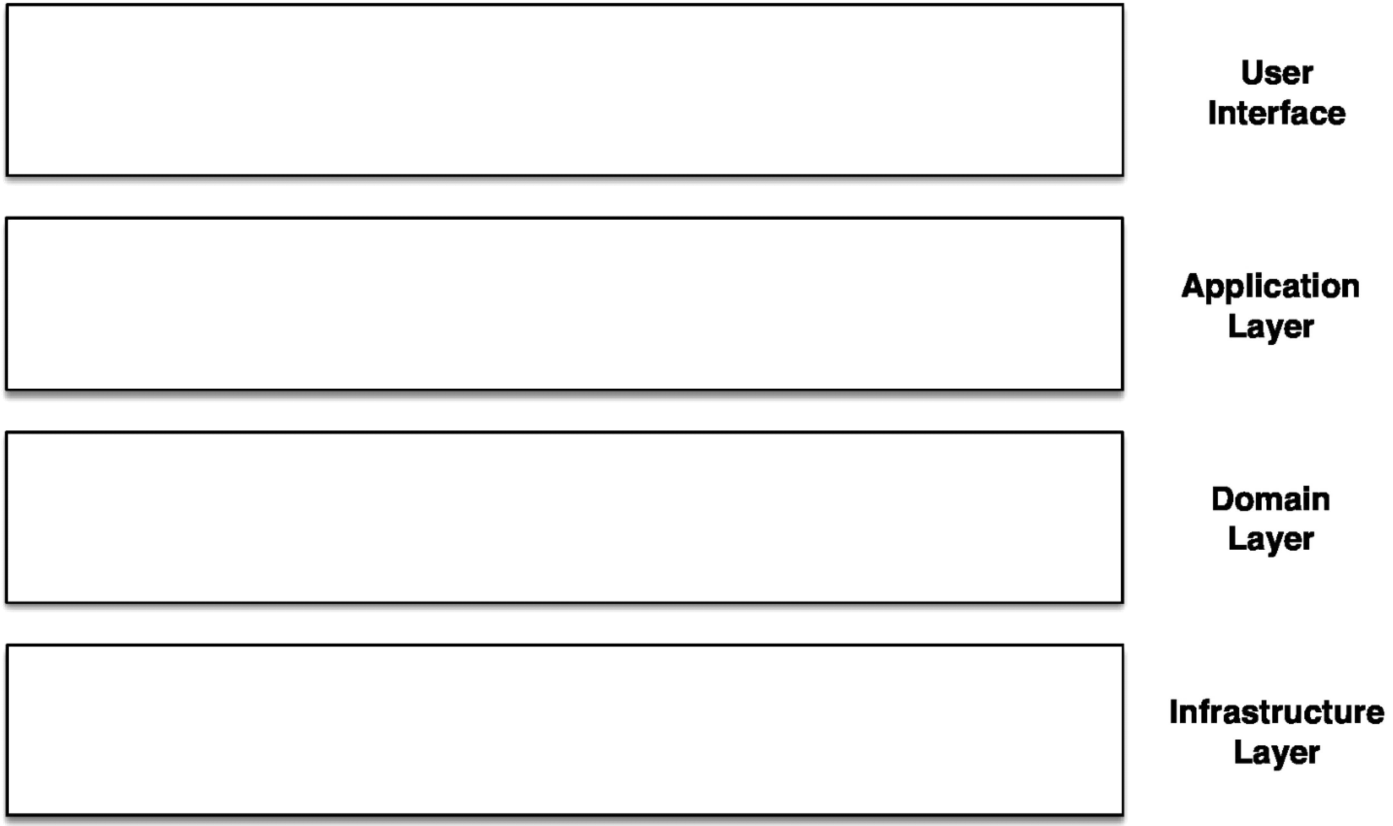
- Anti-Corruption Layer (ACL)
  - Verhindert “Verschmutzen” des Domänenmodells
- Shared Kernel
  - Gemeinsames Teilmodell
- Open Host Service
  - Bounded Context bietet Modell-Transformation für andere Bounded Contexts an

Microservices mit DDD: Strategisches Design
Context Map: Beziehungsmuster

- Partnership
  - Organisation der Koordination zwischen verschiedenen Bounded Contexts zum Zwecke der Integration
- Upstream/Downstream: Customer/Supplier
  - Abhängigkeit zwischen zwei Bounded Contexts nur in eine Richtung ist
- Upstream/Downstream: Conformist
  - Wenn es keine Möglichkeit zur Kollaboration gibt
  - Der Downstream Kontext muss nehmen, was er bekommt

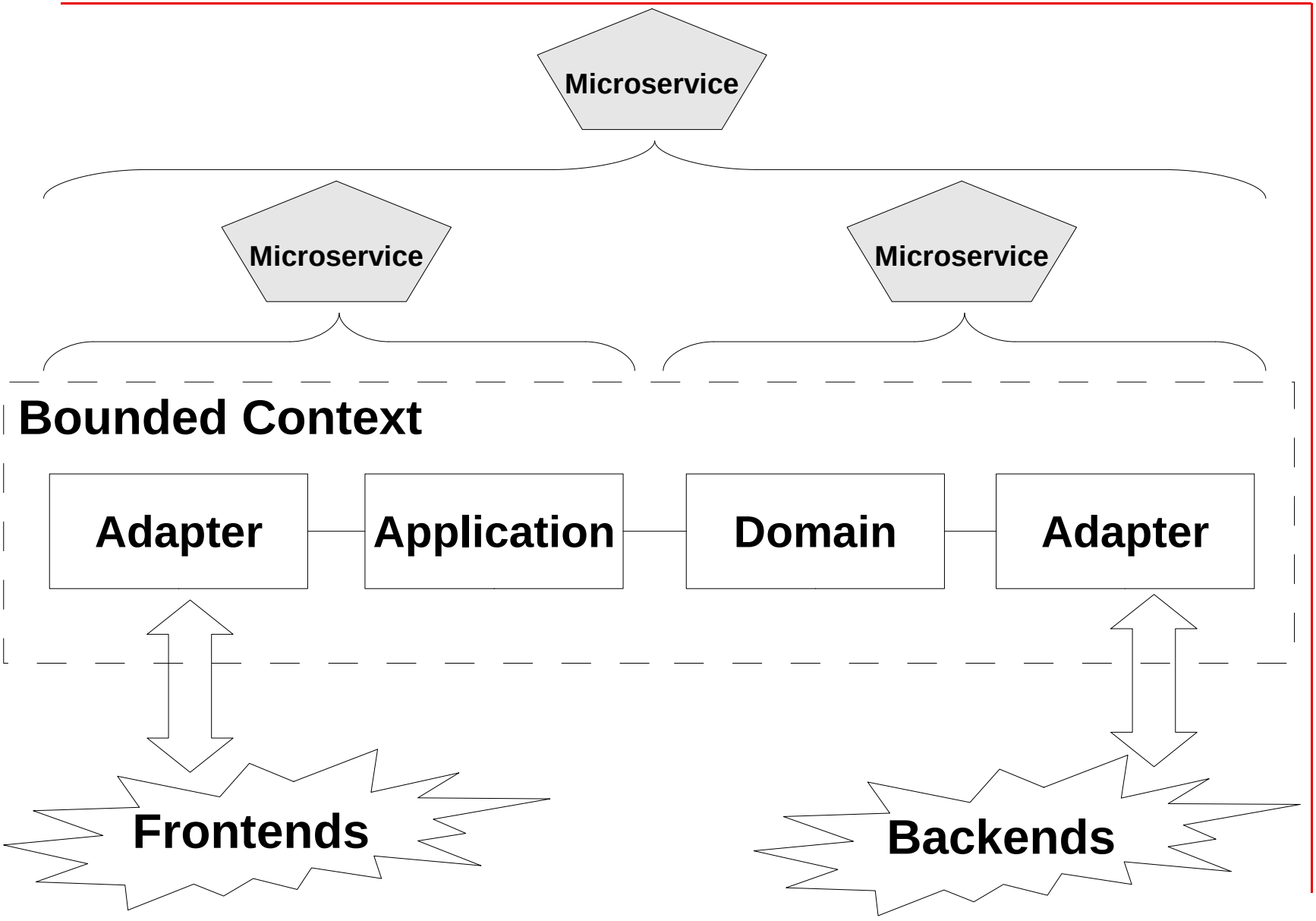
Microservices mit DDD: Strategisches Design

Marko-Design: Layer



Microservices mit DDD: Strategisches Design

Marko-Design: Layer



Microservices mit DDD: Taktisches Design
Domänenobjekte

- Entity
  - Identität mit Lebenszyklus
- Value Object
  - Wert ohne Identität
- Aggregate
  - Kontrolle des Lebenszyklus von Objektnetzen
- Factory
  - Erzeugung komplexer Objekte (Aggregate)
- Repository
  - Fachliche Schnittstelle zu Daten

Microservices mit DDD: Taktisches Design
Domänenobjekte

- Domain Services
  - Element-übergreifende Logik
- Domain Event
  - Fachliche Nachricht
  - Entkopplung von Bounded Contexts
- ...

Ressourcen

Ressourcen
Bücher: Microservices

- Eberhard Wolff (2015): Microservices; dpunkt; ISBN: 9783864903137

Ressourcen
Bücher: DDD

- Vaughn Vernon (2016): Domain-Driven Design Distilled; Addison-Wesley; ISBN: 0134434420
- Scott Millett, Nick Tune (2015): Patterns, Principles, and Practices of Domain-Driven Design; John Wiley & Sons; ISBN: 1118714709
- Vaughn Vernon (2013): Implementing Domain-Driven Design; Addison-Wesley; ISBN: 0321834577
- Eric Evans (2003): Domain-Driven Design: Tackling Complexity in the Heart of Software; Addison-Wesley; ISBN: 0321125215

Ressourcen
Bücher: Software Engineering

- Martin Fowler (2003): Patterns of Enterprise Application Architecture; Addison-Wesley; ISBN; 0321127420
- Frank Buschmann et al. (1996): Pattern-Oriented Software Architecture - A System of Patterns; Wiley; ISBN: 0471958697
- Erich Gamma, et al. (1995): Design Patterns - Elements of Reusable Object-Oriented Software; Addision-Wesley; ISBN: 0201633612
- Oliver Vogel et al. (2011): Software Architecture: A Comprehensive Framework and Guide for Practitioners; Springer; ISBN: 3642197353

Ressourcen
Web: Microservices

- Chris Richardson
  - http://www.microservices.io

- Martin Fowler
  - http://martinfowler.com/articles/microservices.html

Ressourcen
Web: DDD

- DDD Community
  - http://domaindrivendesign.org
- Martin Fowler
  - https://martinfowler.com/tags/domain%20driven%20design.html