Fachhochschule Dortmund
University of Applied Sciences and Arts

Softwaretechnik 1(A)

Domain-Driven Design

Autoren: Prof. Dr. Sabine Sachweh
Unterlagen basieren auf Folien von
Prof. Dr. Sven Jörges

Fachhochschule Dortmund
University of Applied Sciences and Arts

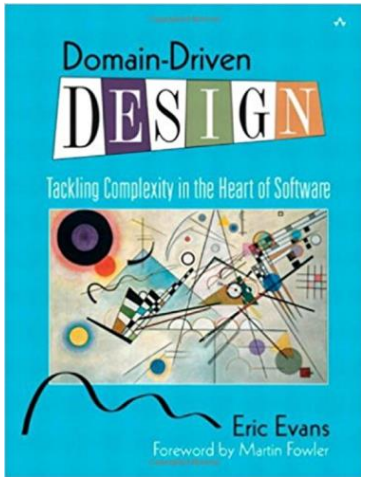
Domain-Driven Design

01: Einführung und Begriffsdefinitionen

Autoren: Prof. Dr. Sabine Sachweh
Unterlagen basieren auf Folien von
Prof. Dr. Sven Jörges

Domain-Driven Design (DDD)

- Erstmals 2003 von Eric Evans beschrieben *[Eva03]*



The Big Blue Book [Eva03]

[Ver17]

DDD-Konferenzen

- Domain-Driven Design Europe
  DDD Europe (seit 2016) → 4./5. Februar 21

- KanDDDinsky Conference (seit 2017)
  2020 wegen Covid19 ausgesetzt

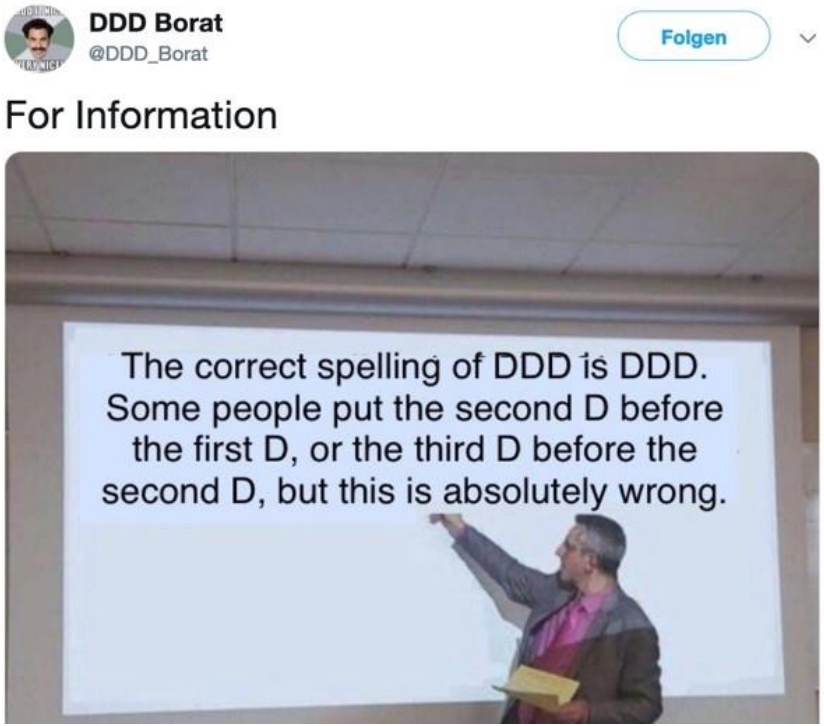
- Explore DDD (seit 2017)
  2020 Online (Ende Okt/Anfang Nov.)

Begriffsdefinition

- Methodik bzw.
  - Satz von Prinzipien,
  - Mustern und
  - Werkzeugen,

  die das Entwerfen und Entwickeln komplexer Software unterstützen

- Zentraler Begriff: Domäne



[Quelle: DDD Borat auf Twitter]

Begriff: Domäne

"A sphere of knowledge, influence, or activity. The subject area to which the user applies a program is the **domain of the software.**"

Eric Evans *[Eva14, S. vi]*

„Ein Wissens-, Einfluss- oder Aktivitätsbereich. Der Themenbereich, in dem ein Benutzer ein Programm/Anwendung einsetzt, ist die **Domäne der Software.**"

Beispiele

- Steuern
- Buchhaltung
- Lagerhaltung
- Personalmanagement
- Versicherungsgeschäfte

DDD: KERNIDEE

- Software wird stets im Kontext einer Domäne eingesetzt
- Software spiegelt dabei Konzepte und Elemente der Domäne wieder
  → sie modelliert die Domäne
- Zur Entwicklung der Sohware müssen die relevanten Konzepte und Elemente sowie deren Beziehungen identifiziert und beschrieben werden
- Das dazu notwendige Wissen besitzen typischerweise nicht die SohwareentwicklerInnen, sondern die FachexpertInnen (*domain experts*), die sich in der entsprechenden Domäne auskennen

DDD: KERNIDEE (2)

"DDD is about **designing software** *based on models of the* **underlying domain."** Martin Fowler

„Bei DDD geht es um Software-Design auf Basis der zugrundeliegenden **Domäne"**

- Die Domäne, in der eine Software eingesetzt wird, steht im Fokus von DDD (anstatt z.B. die Benutzeroberfläche oder das Datenbankmodell)
- Zentrale Tätigkeit: EntwicklerInnen erstellen gemeinsam mit den FachexpertInnen ein **Domänenmodell**

Begriff: Domänenmodell

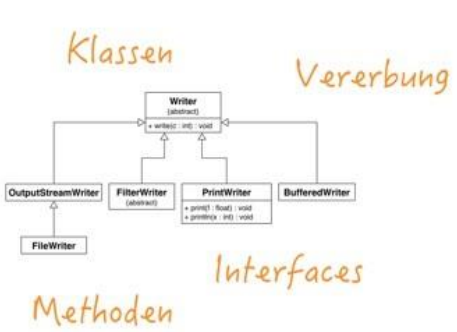
"A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain."

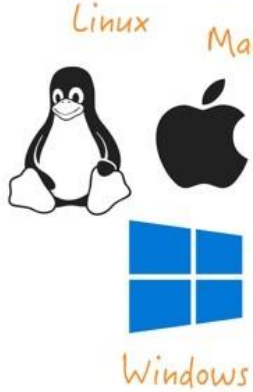
Eric Evans *[Eva14, S.vi]*

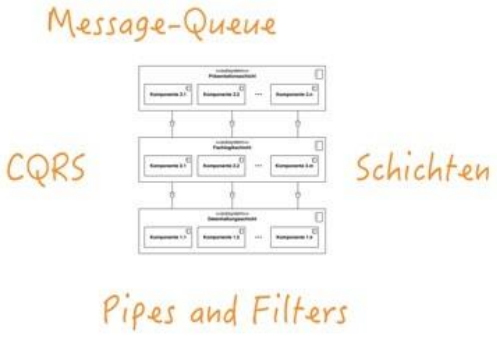
"Ein System von Abstraktionen, das ausgewählte Aspekte einer Domäne beschreibt und zur Lösung von Problemen in Bezug auf diese Domäne verwendet werden kann."

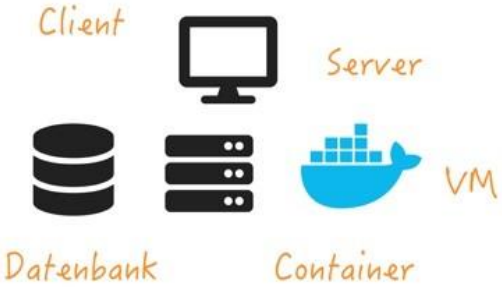
- **Repräsentiert das Domänenwissen** (z.B. Daten, Abläufe, Regeln etc.) der FachexpertInnen in abstrahierter und strukturierter Form
- **Keine festgelegte Notation, kein einzelnes Diagramm**
- **Form** des Domänenmodells **muss geeignet sein, um die Kommunikation** zwischen FachexpertInnen und EntwicklerInnen **zu unterstützen**

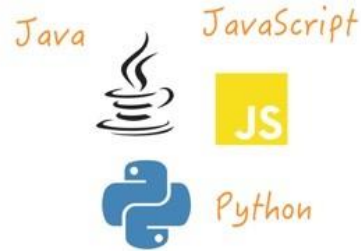
Technische vs. Fachliche Sprache

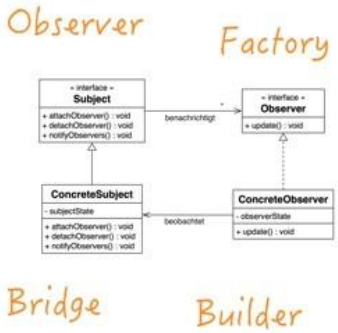












Technische vs. Fachliche Sprache

Beispieldomäne: **Industrie- und Handelskammern (IHK)**

Technische vs. Fachliche Sprache

- FachexpertInnen und EntwicklerInnen verwenden unterschiedliche (Fach-)Sprachen
- Es wird eine gemeinsame Sprache benötigt, welche auf dem Domänenmodell basiert
- DDD spricht hier von der **Ubiquitous Language** (*allgegenwärtigen Sprache*)

Ubiquitous Language

" A language structured around the domain model and used by all team members to connect all the activities of the team with the soßware." **Eric Evans** *[Evo03, S.514]*

„ Eine Sprache, die um das Domänenmodell herum strukturiert ist und von allen Teammitgliedern verwendet wird, um alle Aktivitäten des Teams mit der Software zu verbinden."

- "Allgegenwärtig", da diese gemeinsame Sprache
  - von **allen beteiligten Personen** sowie
  - **an allen Stellen** (d.h. Dokumentation, Diagramme, gesprochenes Wort, Code etc.)

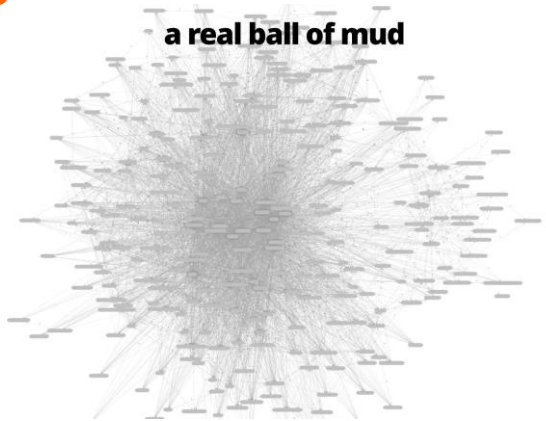
  verwendet wird.
- Das Domänenmodell sowie die darauf basierende **Ubiquitous Language** werden von FachexpertInnen und EntwicklerInnen gemeinsam entwickelt

Gemeinsamer Wissensaufbau

- Verschiedene Techniken und Methoden können zur gemeinsamen Erschließung des Domänenmodells angewendet werden, z.B.:
  - Erstellung eines Glossars zur Sammlung und Erläuterung der Kernkonzepte ✓
  - Use-Case-Diagramme ✓
  - User-Stories (✓)
  - Beschreibung von Szenarien *[Ver17, S.34]*
  - Event Storming*
  - Domain Storytelling*

*Dazu später mehr!

Domänenmodell: Herausforderungen

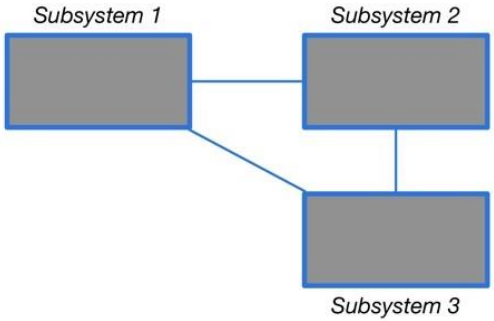


- **Domänen können sehr groß sein.**
  Abstraktion ist schwierig: Was ist relevant für das Domänenmodell und was nicht?

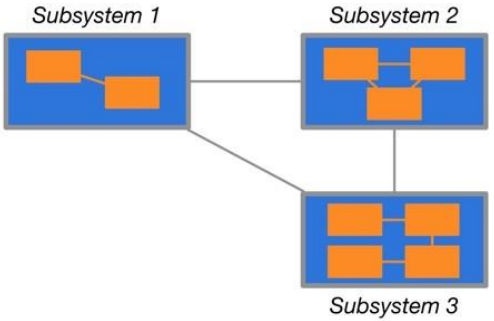
- **Ein einheitliches Domänenmodell für die gesamte Domäne?**
  - Schwer zu erreichen und potentiell sehr komplex [Ver17, S. 20-23]
  - Bei mehreren Teams: Jedes Team ist zuständig für einen Teil des Modells
    → hoher Koordinationsaufwand, Gefahr eines Big Ball of Mud (englisch für „große Matschkugel“)

Strategisches vs. Taktisches Design

- DDD unterscheidet zwischen der Modellierung
  - im Großen (Strategisches Design) und
  - im Kleinen (Taktisches Design)



Makroarchitektur



Mikroarchitektur

Strategisches Design

- Aufteilung des Domänenmodells in sogenannte **Bounded Contexts** (begrenzte Kontexte)
- Jeder Bounded Context hat ein **eigenes Domänenmodell** und seine **eigene Ubiquitous Language**
- Zusammenhänge/Beziehungen zwischen Bounded Contexts werden mittels **Context Maps** modelliert
- Zusätzliche Strukturierung komplexer Domänen (z.B. bei Altsystemen) in **Subdomains** (Teildomänen, Subdomänen)

Taktisches Design

- Befasst sich mit der **Modellierung innerhalb eines Bounded Contexts**
- DDD definiert eine Menge von **Basisbausteinen** (auch: *Building Blocks, Tactical Patterns*) für Entwurf und Implementierung
- → SWT 2

Fachhochschule Dortmund
University of Applied Sciences and Arts

Literaturquellen

[Eva03] Evans E.; Domain-Driven Design: Tackling Complexity in the Heart of Sohware. Addison-Wesley; 2003

[Eva14] Evans E.; Domain-Driven Design Reference: Definitions and Pattern Summaries. Dog Ear Publishing; 2014; Kostenloser Download auf domainlanguage.com

[Ver17] Vernon V.; Domain-Driven Design kompakt. dpunkt; 2017

Fachhochschule Dortmund
University of Applied Sciences and Arts

Domain-Driven Design

02: Strategisches Design – Bounded Contexts

Autoren: Prof. Dr. Sabine Sachweh
Unterlagen basieren auf Folien von
Prof. Dr. Sven Jörges

Strategisches Design

- Aufteilung des Domänenmodells in sogenannte **Bounded Contexts** (begrenzte Kontexte)
- Jeder Bounded Context hat ein **eigenes Domänenmodell** und seine **eigene Ubiquitous Language**
- Zusammenhänge/Beziehungen zwischen Bounded Contexts werden mittels **Context Maps** modelliert
- Zusätzliche Strukturierung komplexer Domänen (z.B. bei Altsystemen) in **Subdomains** (Teildomänen, Subdomänen)

Fachhochschule Dortmund
University of Applied Sciences and Arts

Begriff: Bounded Context

"A description of a boundary (typically a subsystem, or the work of a particular team) within which a particular model is defined and applicable." **Eric Evans** *[Eva14, S.vi]*

„Eine Beschreibung einer Grenze (normalerweise eines Subsystems oder der Arbeit eines bestimmten Teams), innerhalb derer ein bestimmtes Modell definiert und anwendbar ist. "

- "Strategisches Entwurfsmuster" [Ver17, S. 7]
- Fasst fachlich eng zusammengehörende Konzepte und Komponenten in einem eigenen Domänenmodell zusammen (→ *hohe Kohäsion*!)
- Besitzt eine eigene Ubiquitous Language, die innerhalb des Bounded Contexts eindeutig und konsistent ist

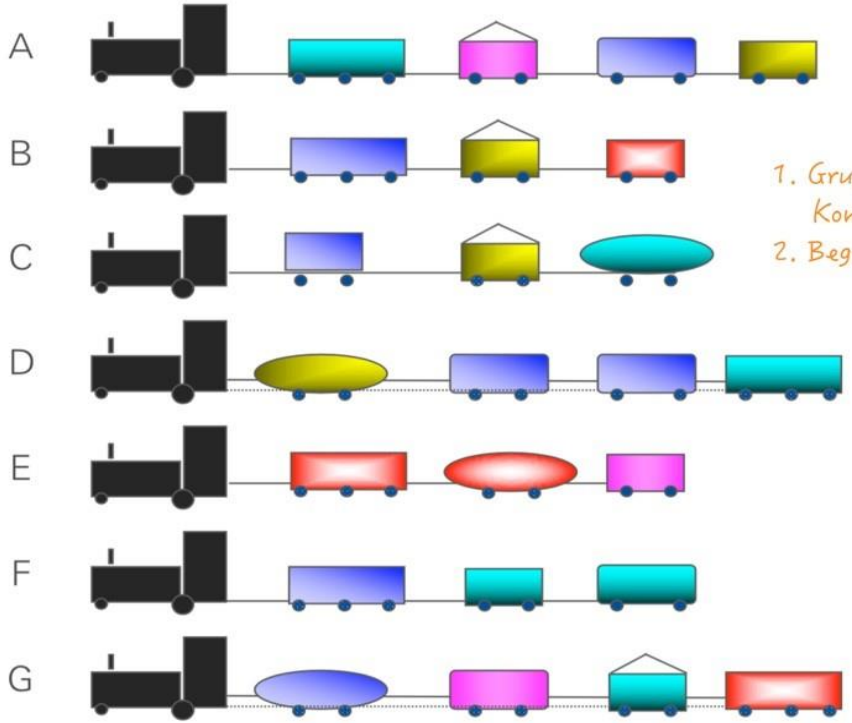
Begriff: Bounded Context (2)

Die durch einen Bounded Context definierten Grenzen haben direkten Einfluss auf:

- **Teams**
  - Für jeden Bounded Context sollte genau ein Team zuständig sein [Ver17, S. 14]
  - Innerhalb des Teams wird die Ubiquitous Language des Bounded Contexts verwendet und entwickelt
- **Entwurf der Software**
  - Für jeden Bounded Context sollte eine eigene Quellcode-Basis, ein eigenes Datenbankschema etc. existieren [Ver17, S. 14]
  - Das zuständige Team definiert die Schnittstellen für die Benutzung des Bounded Context (→ Kapselung!)

Fachhochschule Dortmund
University of Applied Sciences and Arts

Aufgabe: Konzepte gruppieren



1. Gruppieren Sie die dargestellten Konzepte A-G.
2. Begründen Sie die Gruppierung.

Quelle: Carola Lilienthal, The Core of Domain-Driven Design, Software Architecture Summit 2017

Beispieldomäne: IHK

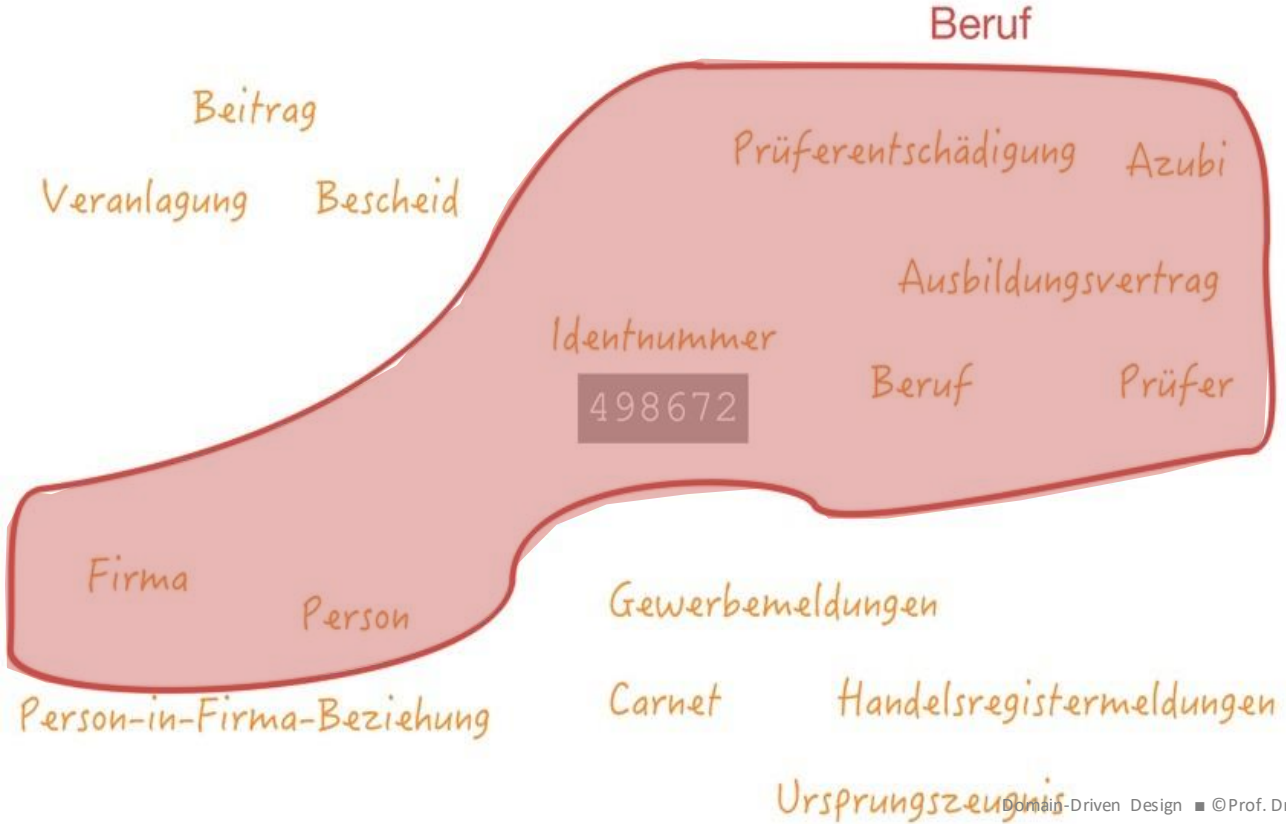
Beispieldomäne: Aufteilen in Bounded Contexts

Nach welchen Kriterien kann die Domäne "geschnitten" werden?

- **Beispiele**
  - Nach **(Fach-)Abteilungen** bzw. **Arbeitsgruppen** in der Organisation
    (z.B. Marketing, Buchhaltung, Support)
  - Nach **Geschäftsprozessen**
    (z.B. Versicherung: "Risikobewertung bei Neuverträgen", "Schadensregulierung")
  - Nach kontextbezogenen **Unterschieden in der Verwendung von Begriffen**
    (siehe Beispiel auf den folgenden Folien)

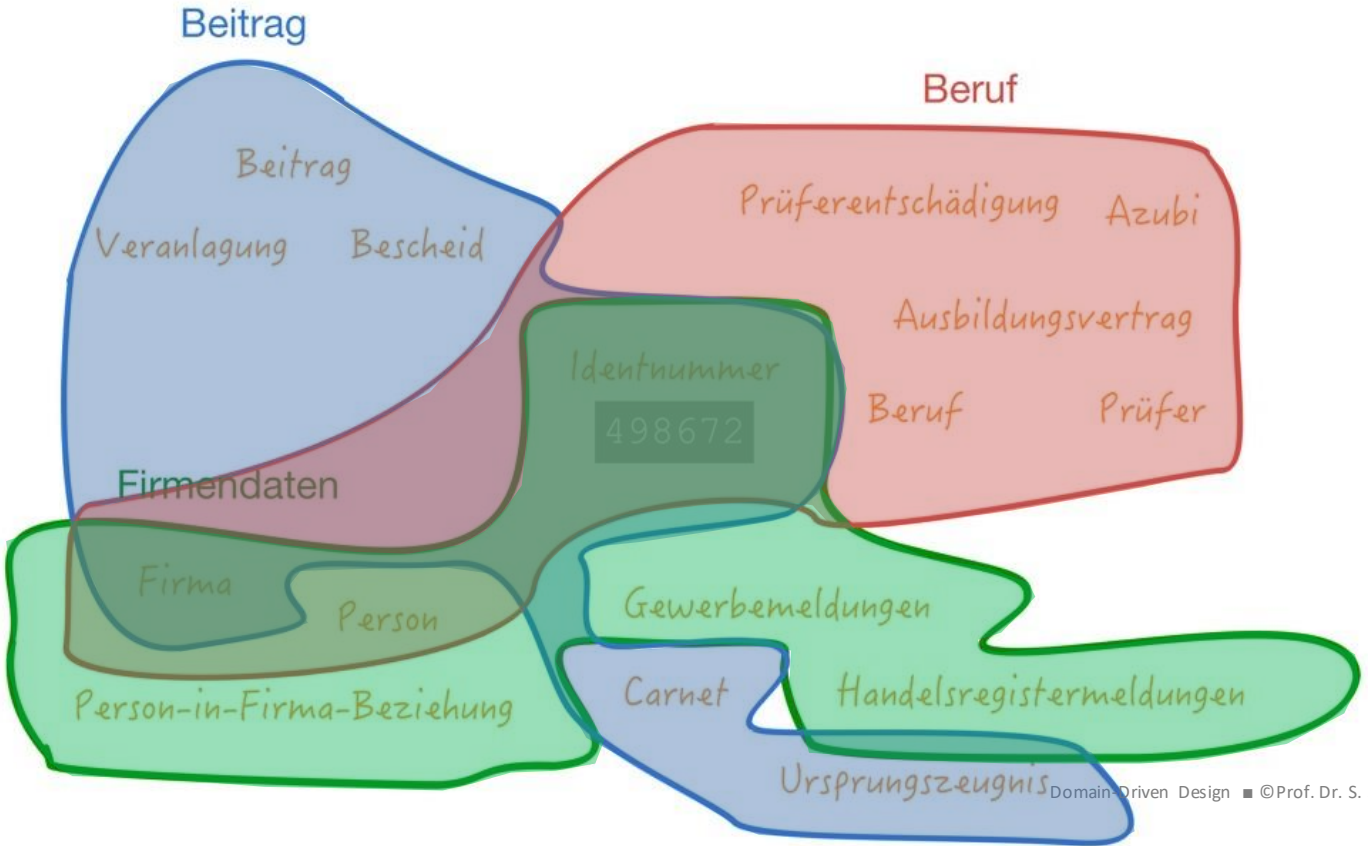
Beispieldomäne: IHK

Beispieldomäne: IHK



Beispieldomäne: IHK

Beispieldomäne: IHK



Beispieldomäne: IHK

- Bounded Contexts können **überlappende Konzepte** enthalten:
  - **„Firma“**
    - in Abteilung „Beitrag“ als Beitragszahler,
    - in Abteilung “Beruf“ als Ausbildungsstätte
  - **„Person“**
    - in Abteilung „Firmendaten“ als Geschäftsführer,
    - in Abteilung “Beruf“ als Prüfer

Fachhochschule Dortmund
University of Applied Sciences and Arts

Literaturquellen

[Eva14] Evans E.; Domain-Driven Design Reference: Definitions and Pattern Summaries. Dog Ear Publishing; 2014; Kostenloser Download auf domainlanguage.com

[Ver17] Vernon V.; Domain-Driven Design kompakt. dpunkt; 2017

Fachhochschule Dortmund
University of Applied Sciences and Arts

Domain-Driven Design

03: Strategisches Design – Context Mapping

Autoren: Prof. Dr. Sabine Sachweh
Unterlagen basieren auf Folien von
Prof. Dr. Sven Jörges

Strategisches Design

- Aufteilung des Domänenmodells in sogenannte Bounded Contexts (begrenzte Kontexte)
- Jeder Bounded Context hat ein eigenes Domänenmodell und seine eigene Ubiquitous Language
- Zusammenhänge/Beziehungen zwischen Bounded Contexts werden mittels **Context Maps** modelliert
- Zusätzliche Strukturierung komplexer Domänen (z.B. bei Altsystemen) in **Subdomains** (Teildomänen, Subdomänen)

Begriff: Context Mapping

- Context Mapping befasst sich mit den Beziehungen zwischen bzw. mit der Integration von verschiedenen Bounded Contexts

  ! Folglich auch: Wechselseitige Beziehungen zwischen Teams!

- Context Maps skizzieren diese Beziehung
  → keine formal definierte Notation

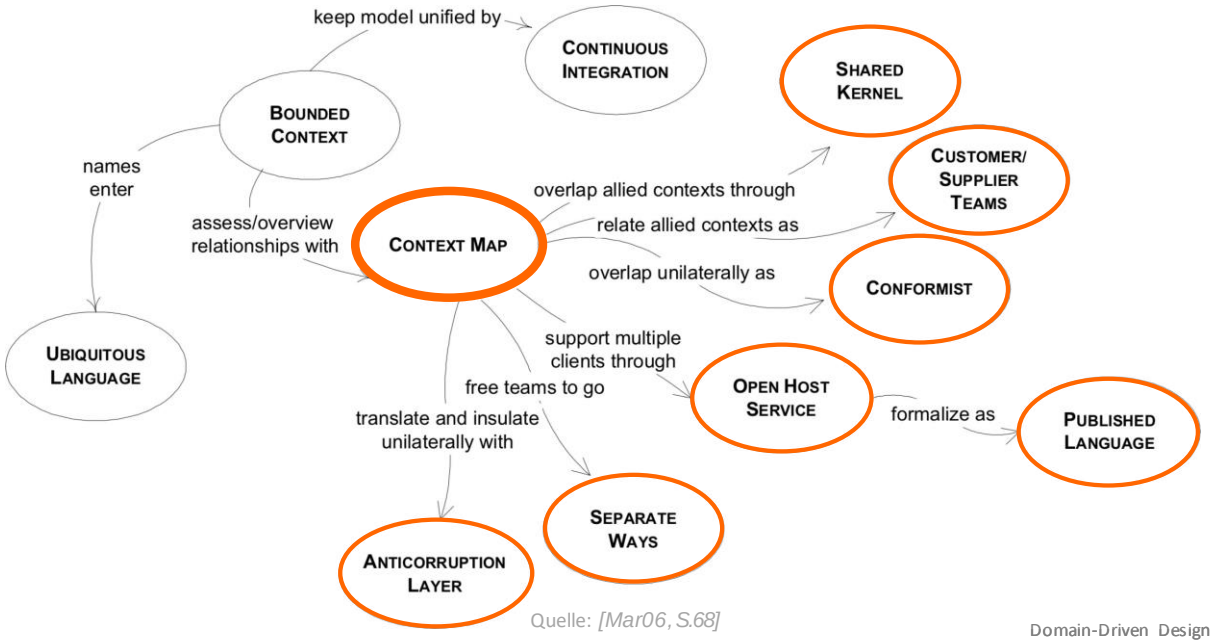
  (im Folgenden wird die Notation aus [Ver17] verwendet)

- Ziel: Klare Grenzen und Verträge zwischen Bounded Contexts definieren

  [Ver17, S.51]

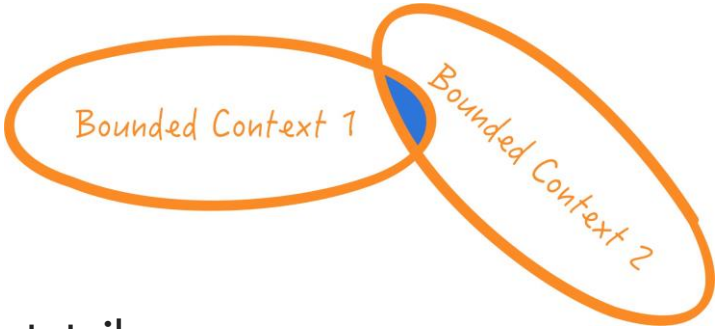
Arten von Context Mappings

DDD definiert verschiedene Arten von Beziehungen zwischen ***Bounded Contexts***:



Quelle: *[Mar06, S.68]*

Shared Kernel



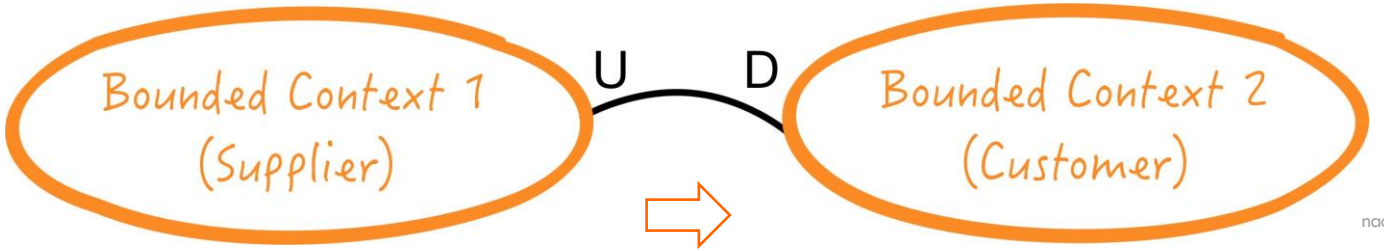
nach [Ver17]

- Shared Kernel = Kleines gemeinsames Domänenmodell, das sich mehrere Bounded Contexts teilen
- z.B. in Form einer Bibliothek Vermeidet Duplizierung
- Führt zu einer engen Kopplung zwischen den Teams → bei Änderung/Weiterentwicklung des geteilten Modells müssen alle beteiligten Teams einbezogen werden

Shared Kernel – Beispiel IHK

Modellelement "Identnummer" als Shared Kernel

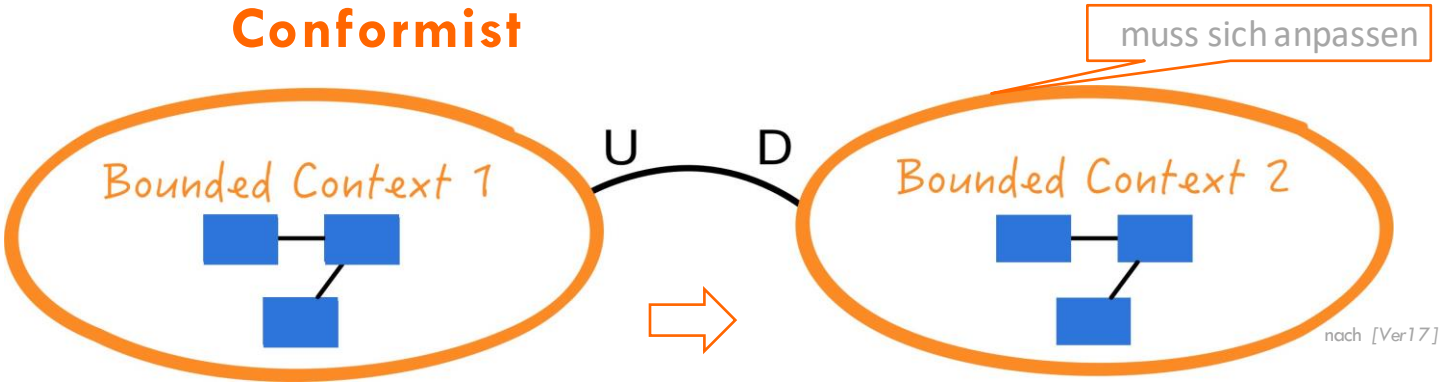
Customer-Supplier



nach [Ver17]

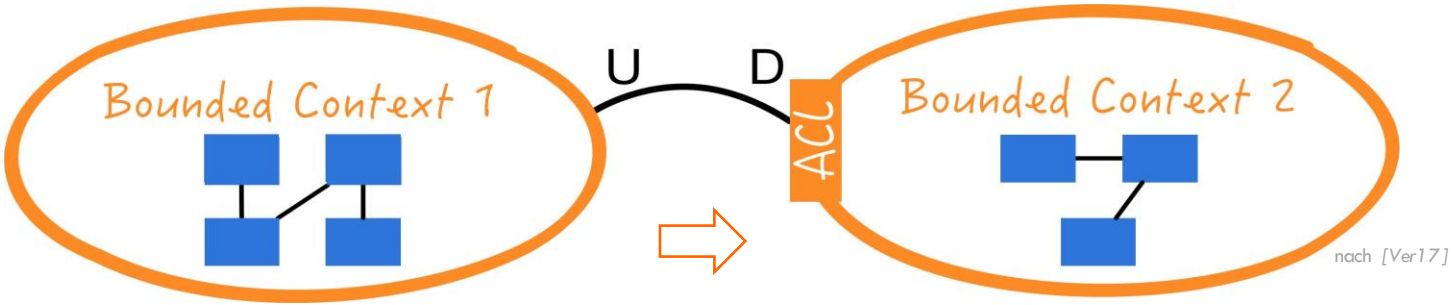
- Ein Bounded Context stellt etwas zur Verfügung (Supplier, Lieferant), was ein anderer Bounded Context benötigt (Customer, Kunde)
- Der Supplier ist vorgeschaltet (**u**pstream), der Customer ist nachgeschaltet (**d**ownstream)
- Customer stellt Anforderungen an Supplier, der diese umsetzt
- Supplier bestimmt, wann und wie die Anforderungen umgesetzt werden

Conformist



- Im Gegensatz zu Customer-Supplier nimmt der Upstream-Context keine Rücksicht auf die Anforderungen des Downstream-Context
- Der Downstream-Context passt sich (*engl. conforms to*) dem Modell des Upstream-Kontextes an
- Beispiel: Apple-Partner müssen sich dem Apple-Modell anpassen

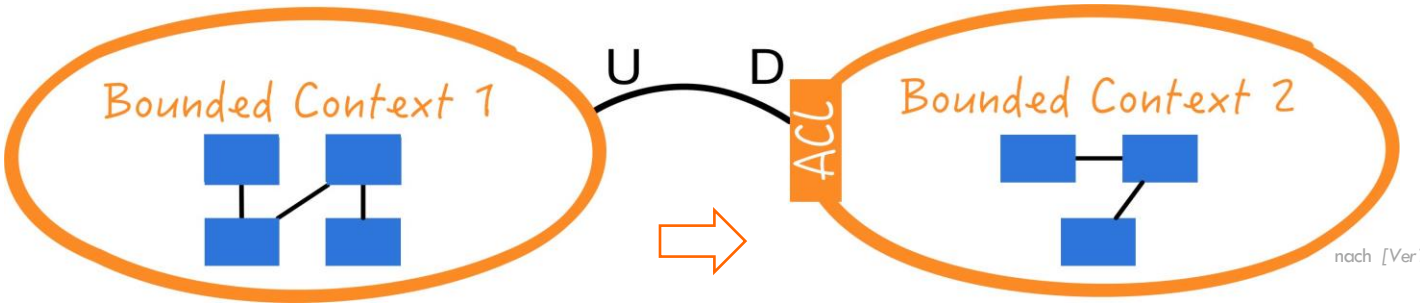
Anticorruption Layer (ACL)



nach [Ver17]

- Anticorruption Layer (ACL) = Übersetzungsschicht zwischen Upstream- und Downstream-Modell
- Von Downstream-Context zur Verfügung gestellt → Umsetzungsaufwand!
- **Ziele:**
  - Entkopplung auf Seiten des Downstream-Contexts
  - Abstraktion des Upstream-Modells
    → höhere Flexibilität bei der Gestaltung des eigenen Domänenmodells

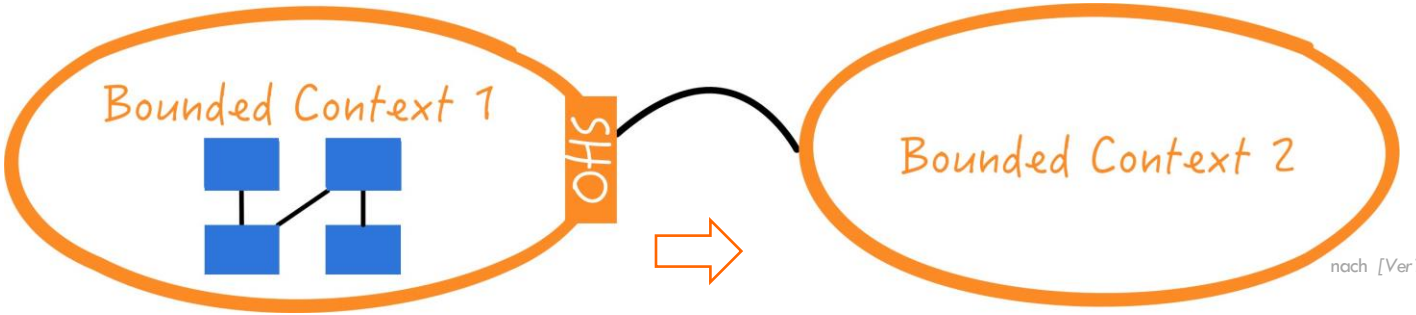
Anticorruption Layer (ACL) (2)



nach [Ver17]

- GoF-Entwurfsmuster zur Umsetzung: z.B. Facade, Adapter
- (Technische) Beispiele: Java Persistence API (JPA) , SLF4J

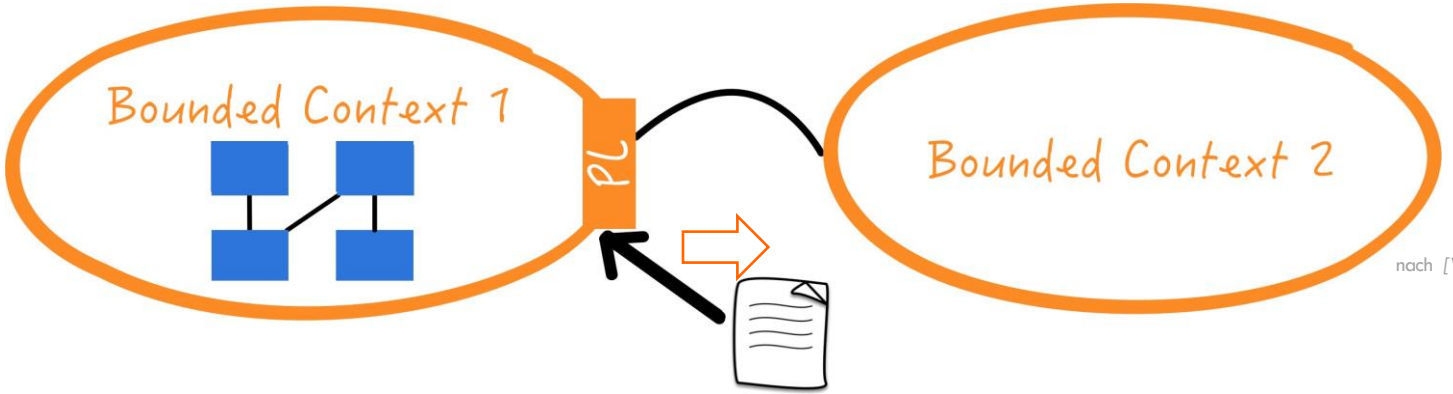
Open Host Service (OHS)



nach [Ver17]

- Open Host Service (OHS) = Wohldefinierte Schnittstelle (API, Protokoll), die ein Bounded Context zum Zugriff anbietet
- Schnittstelle wird in Form von Services zur Verfügung gestellt (z.B. REST-Service)
- **Open**, d.h. jeder kann die Schnittstelle verwenden [Ver17, S. 55]
- **Unterschied zur Conformist-Beziehung**
  OHS wird explizit mit dem Ziel einer leichten Benutzbarkeit entworfen
- Beispiel: Google Maps API

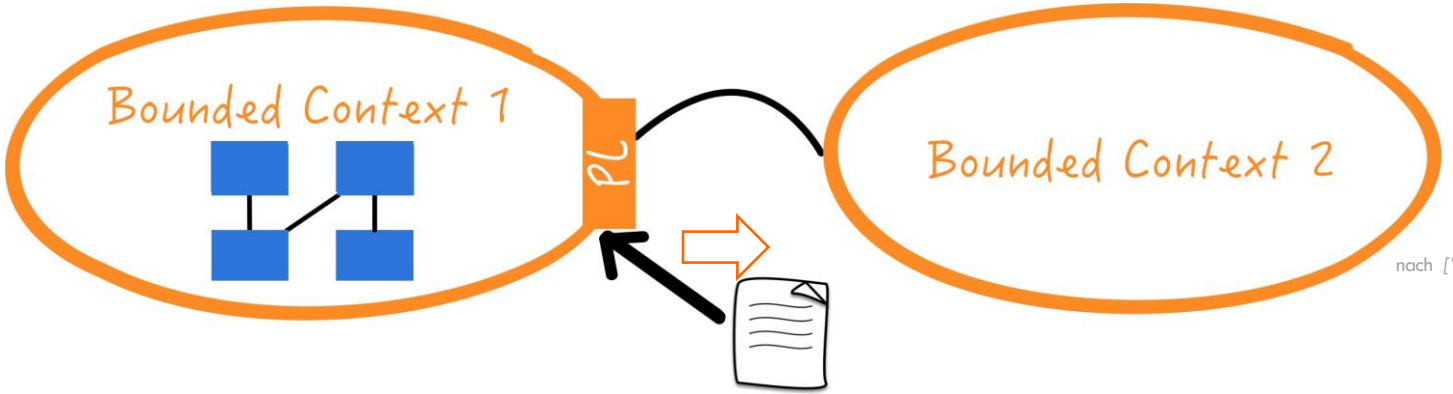
Published Language (PL)



nach [Ver17]

- Published Language (PL) = Wohldokumentierte Sprache zum Informations-austausch [Ver17, S. 56]
- Formal spezifiziert, z.B. mittels XML Schema oder JSON Schema
- Ziel: Erleichterung einer korrekten Integration bzw. Übersetzung verschiedener Domänenmodelle

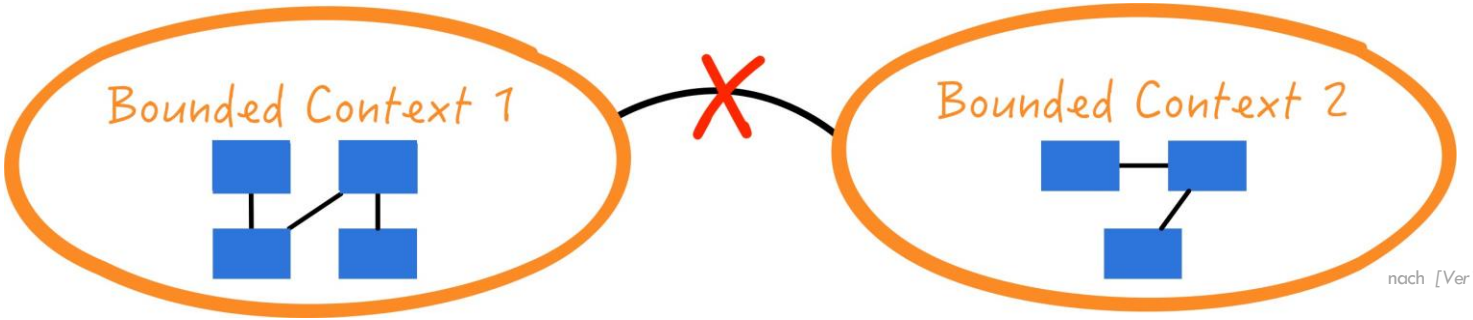
Published Language (PL) (2)



nach [Ver17]

- Oft bietet ein Open Host Service seine Schnittstelle in Form einer PL an
- Beispiele für Published Languages:
  - XJustiz: XML-basiertes Format für elektronischen Rechtsverkehr (z.B. zwischen Anwälten, Notaren etc.)
  - GeoJSON: JSON-basiertes Format zum Austausch geografischer Daten

Separate Ways



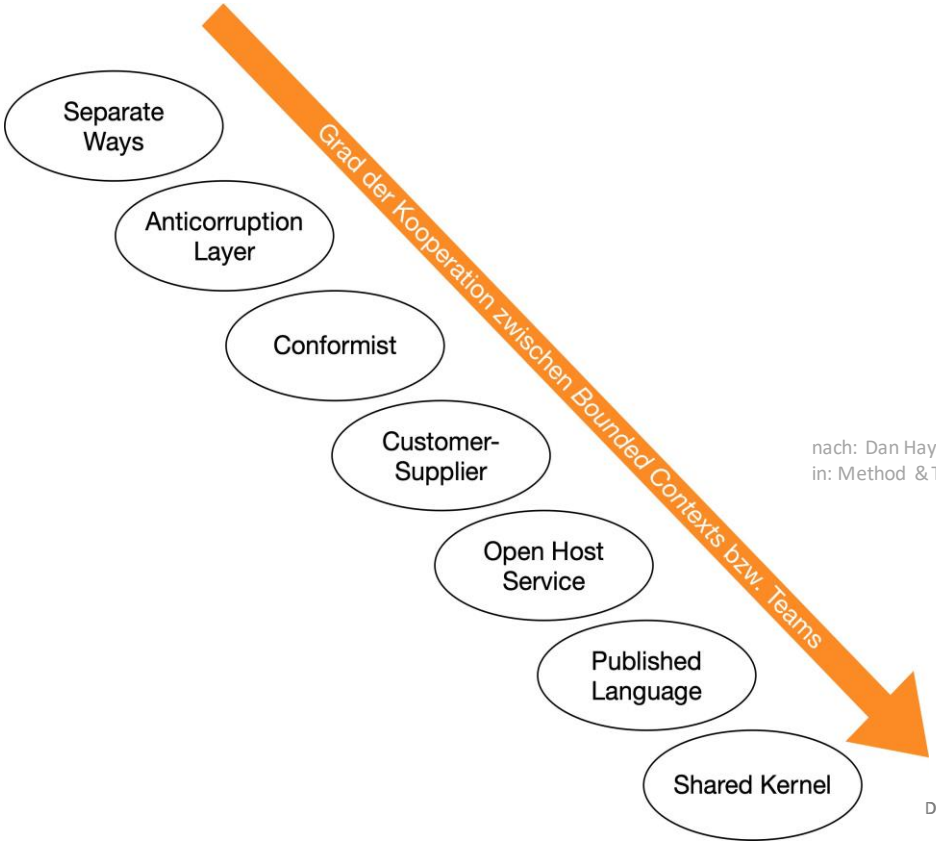
nach [Ver17]

⇒

- Situationen:
  - Die Unterschiede zwischen den Bounded Contexts sind zu groß (fachlich, technisch, organisatorisch)
  - Der durch eine Integration entstehende Nutzen ist nicht groß genug

⇒ Auf eine Integration wird vollständig verzichtet

Context Mappings und Kooperation



nach: Dan Haywood, An Introduction to Domain Driven Design, in: Method & Tools

Literaturquellen

[Mar06] Marinescu F., Avram A.; Domain-Driven Design Quickly. Lulu Press; 2006; Kostenloser Download auf InfoQ

[Ver17] Vernon V.; Domain-Driven Design kompakt. dpunkt; 2017

Fachhochschule Dortmund
University of Applied Sciences and Arts

Domain-Driven Design

04: Strategisches Design – Subdomains

Autoren: Prof. Dr. Sabine Sachweh
Unterlagen basieren auf Folien von
Prof. Dr. Sven Jörges

Strategisches Design

- Aufteilung des Domänenmodells in sogenannte Bounded Contexts (begrenzte Kontexte)
- Jeder Bounded Context hat ein eigenes Domänenmodell und seine eigene Ubiquitous Language
- Zusammenhänge/Beziehungen zwischen Bounded Contexts werden mittels Context Maps modelliert
- Zusätzliche Strukturierung komplexer Domänen (z.B. bei Altsystemen) in **Subdomains** (Teildomänen, Subdomänen)

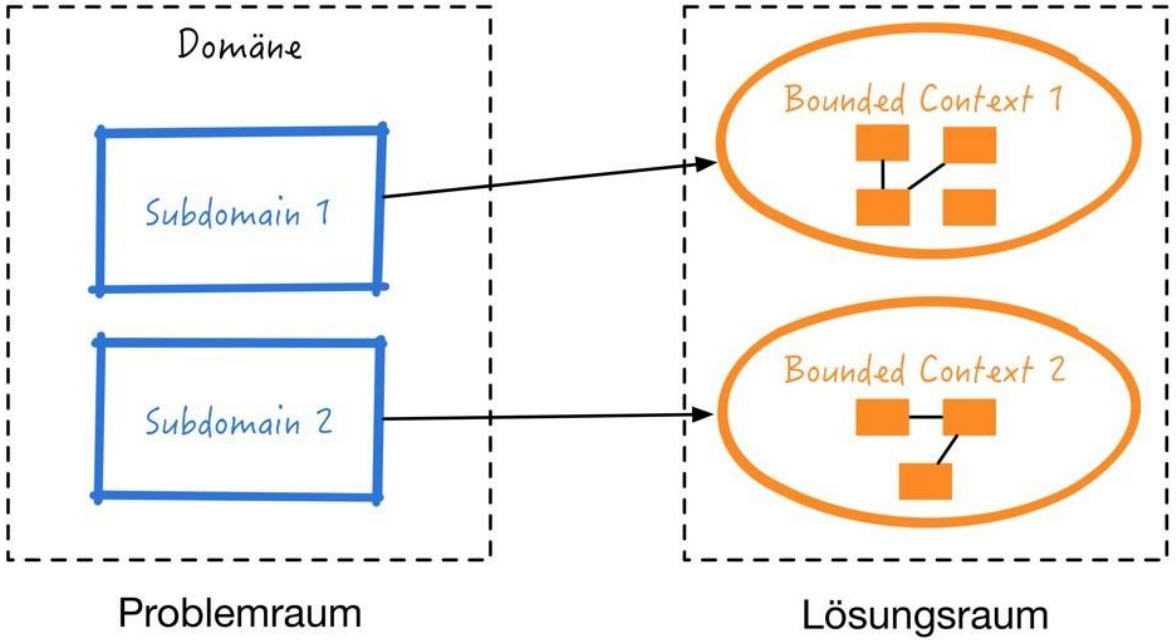
Subdomains

- **Subdomains** dienen der Aufteilung der Domäne in kleinere, beherrschbare Teile
- Insbesondere nützlich, wenn der Ausgangspunkt ein Big Ball of Mud ist (→ *Divide and Conquer*)

- **Unterschied zu Bounded Contexts:**
  - **Subdomains** dienen der Aufteilung der Domäne im **Problemraum**
  - **Bounded Contexts** dienen der Aufteilung des Domänenmodells im **Lösungsraum**

Subdomains vd. Bounded Contexts

- **Erstrebenswerter Idealfall:**
  1:1-Entsprechung zwischen Subdomains u. Bounded Contexts



Arten von Subdomains

- Nicht alle Subdomains sind für ein Unternehmen gleich wichtig

- DDD definiert eine Klassifikation von Subdomains, um Fokussierung zu ermöglichen:
  1. Core Domain
  2. Supporting Subdomain
  3. Generic Subdomain

Arten von Subdomains

- **Core Domain**
  - Zentrale Domäne s Alleinstellungsmerkmal des Unternehmens
  - Höchste Priorität, höchste Bindung von Ressourcen
  - Beispiel "ILIAS": Verwaltung von Lehrveranstaltungen und –materialien

- **Supporting Subdomain**
  - Wichtig und notwendig, aber nicht das Kerngeschäft des Unternehmens
  - Geringerer Einsatz eigener Ressourcen, ggf. Auslagern per Outsourcing
  - Beispiel "ILIAS": Chat-System

Arten von Subdomains (2)

- **Generic Subdomain**
  - Ebenfalls notwendig, aber durch Standardsoftware abdeckbar ("von der Stange")
  - Einkaufen, nicht selbst entwickeln
  - Beispiel "ILIAS": Authentifikationskomponente

Fachhochschule Dortmund
University of Applied Sciences and Arts

Literaturquellen (alle)

[Eva03] Evans E.; Domain-Driven Design: Tackling Complexity in the Heart of Sohware. Addison-Wesley; 2003

[Eva14] Evans E.; Domain-Driven Design Reference: Definitions and Pattern Summaries. Dog Ear Publishing; 2014; Kostenloser Download auf domainlanguage.com

[Mar06] Marinescu F., Avram A.; Domain-Driven Design Quickly. Lulu Press; 2006; Kostenloser Download auf InfoQ

[Ver17] Vernon V.; Domain-Driven Design kompakt. dpunkt; 2017

www.fh-dortmund.de

facebook.com/fhdortmund instagram.com/fhdortmund twitter.com/fh_dortmund youtube.com/FachhochschuleDO