Softwaretechnik 2 (ST2)

Informatik Bachelor, SoSe 2025

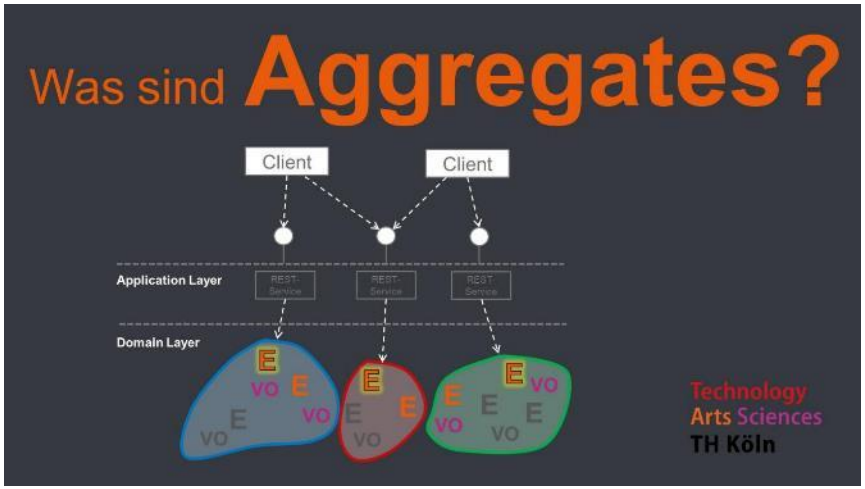
Prof. Dr.-Ing. Stefan Bente

Workshop M1:

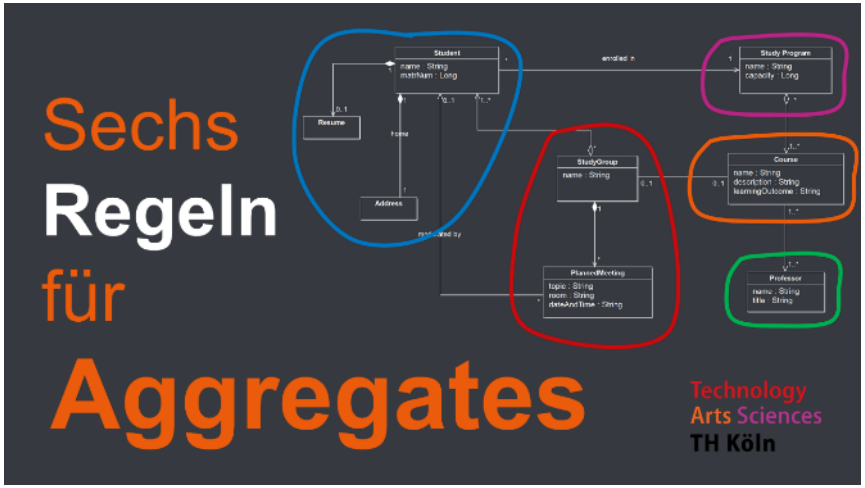
Clean Code, SOLID, und Aggregate-Konventionen

Technology
Arts Sciences
TH Köln

E1 - Aggregates

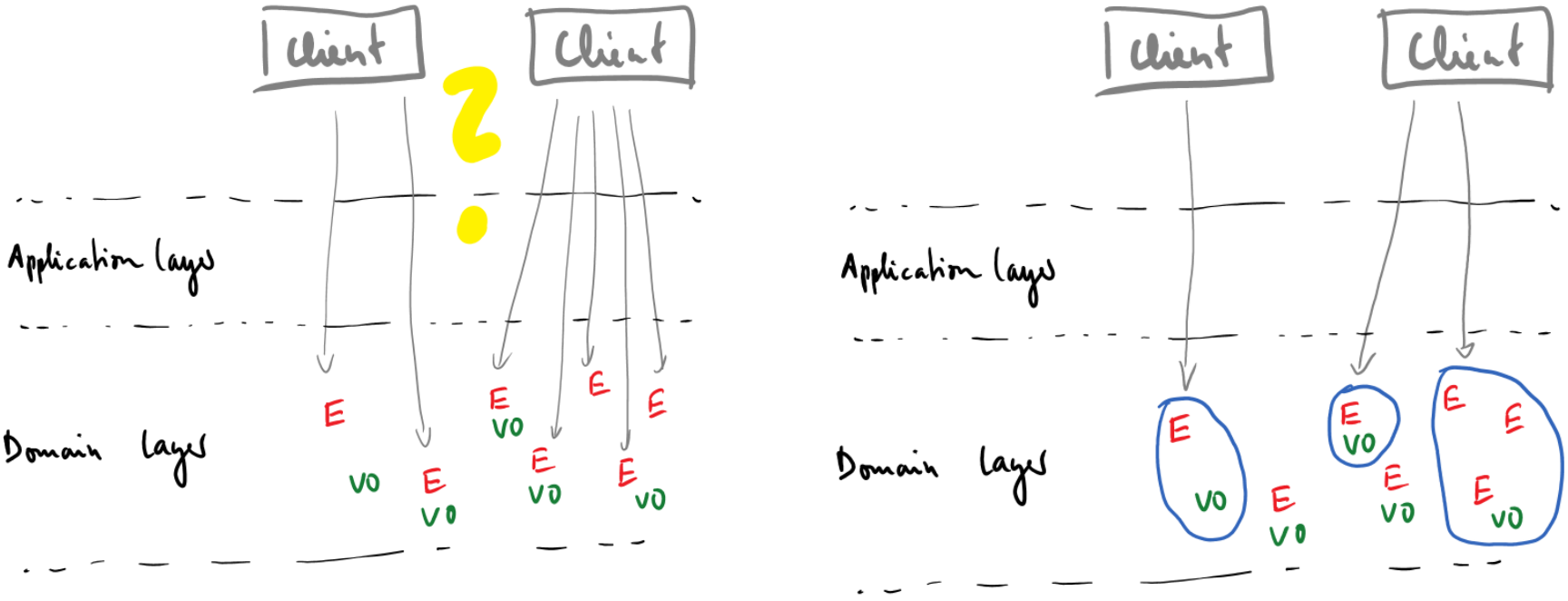


https://www.youtube.com/watch?v=KywRgZpLb5w



https://www.youtube.com/watch?v=WTau26feewU

Big Picture: **Warum Aggregates wichtig sind**



- Entities, Value Objects und Repositories nutzen Sie, um Ihren Domain Layer – also die elementare Geschäftslogik im Backend – zu implementieren.
- Das alles machen Sie am Ende aber i.d.R. nur, um mit Clients via Schnittstellen (z.B. REST) auch darauf zugreifen zu können. Wenn Sie das machen – möchten Sie dann wahllos einfach ***ALLES*** nach außen sichtbar (und änderbar) machen (siehe Bild links)?
- Vermutlich nein. Und wenn Sie nur selektiv bestimmte Teile Ihrer Funktionalität über Schnittstellen nach außen sichtbar machen, welche Teile? Und welche Struktur haben dann die Schnittstellen? Aggregates (Bild rechts) sind bei dieser Frage eine sehr große Hilfe.

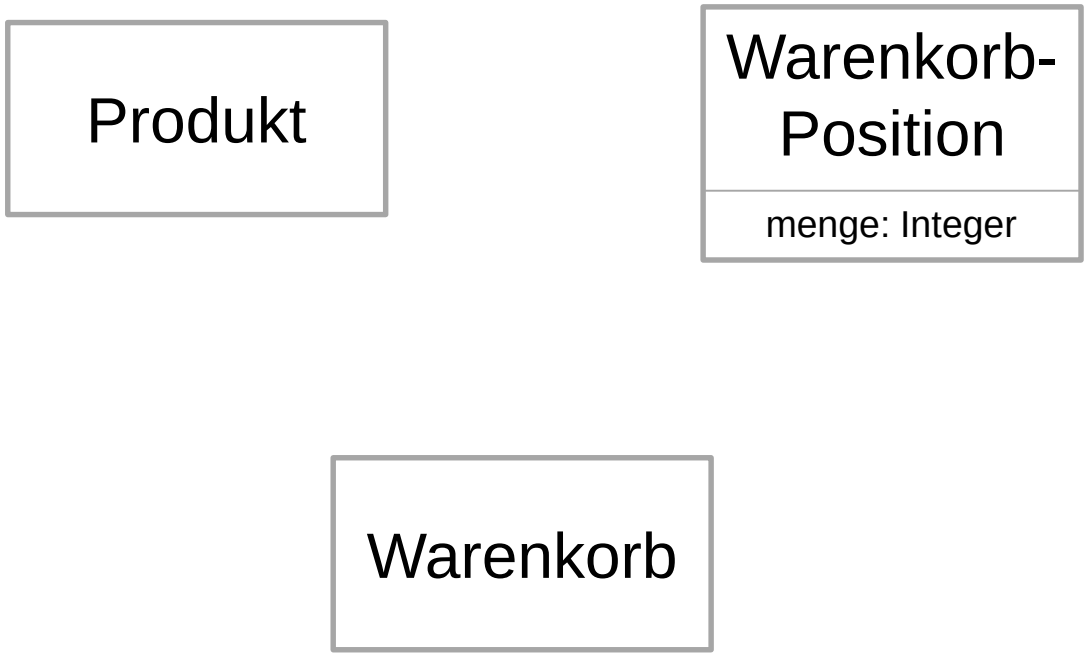
Definition eines Aggregate

- **Aggregate =**
  Geflecht von Entities mit fachlichem Zusammenhang

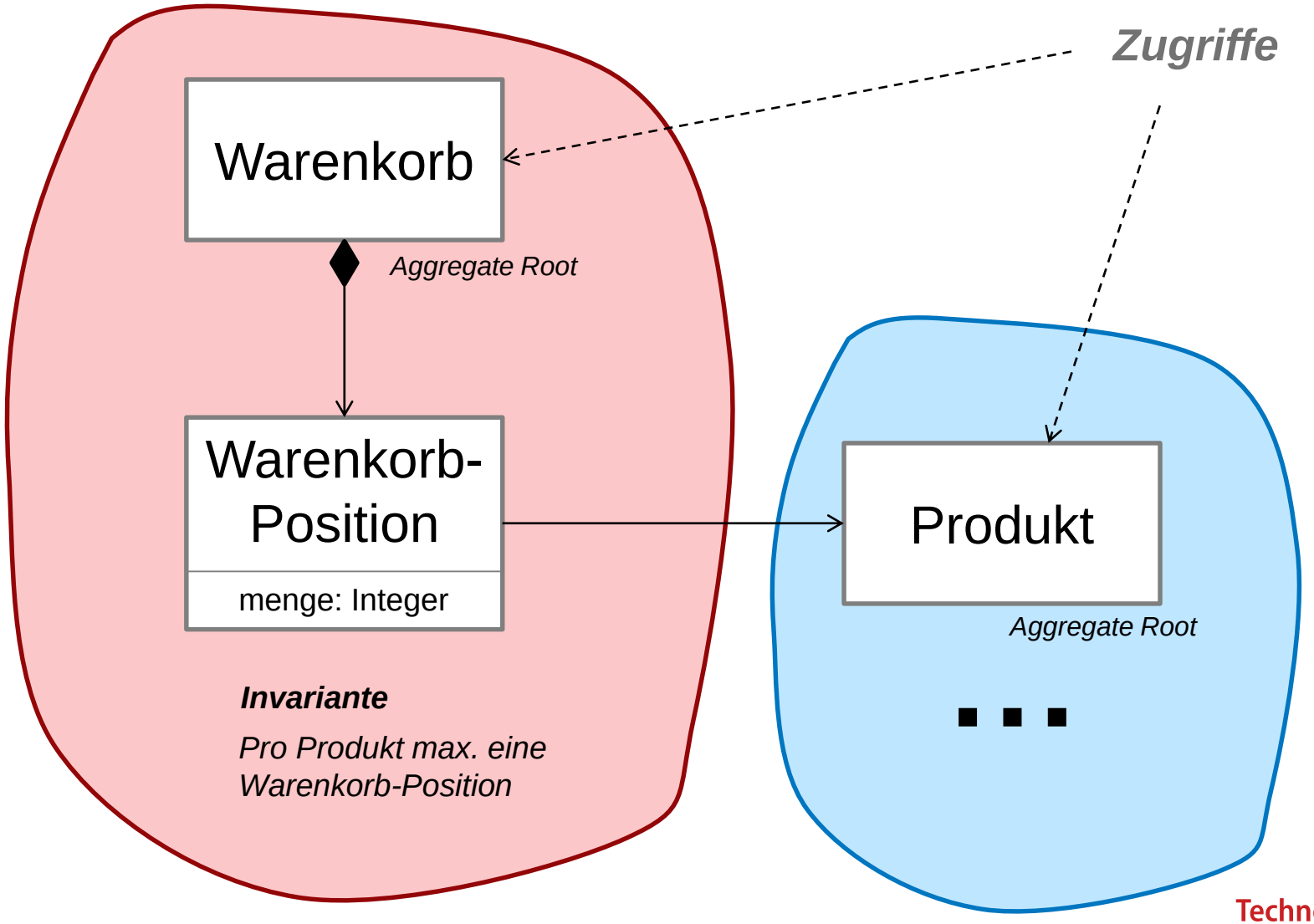
- Die Bestandteile des Aggregate haben Invarianten
  - Kann sich auf Attribute verschiedener inneren Entities beziehen
- Die **Wurzel-Entity** hat eine **globale** Identität
  - Identitäten der inneren Entities sind lokal zum Aggregate
- Nur diese Wurzel-Entity kann durch eine Query gesucht werden
  - keine Suche nach inneren Entities
- Referenzen von außen: nur auf die Wurzel-Entity
  - keine Referenzen auf innere Entities
- Lösch-Operationen löschen das gesamte Aggregate

Quelle: [Evans], S. 128 f.

Motivation für Aggregates am konkreten Beispiel



Motivation für Aggregates am konkreten Beispiel



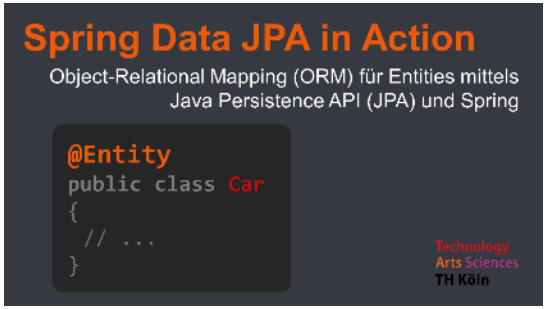
Regel für Aggregates

1. Referenzen von außen auf ein inneres Entity sind verboten.
   - Alle Zugriffe auf innere Entities müssen über Methoden des Aggregate Root erfolgen.
2. Nur für Aggregate Roots gibt es Repositories
3. Aus dem Aggregate heraus darf es Referenzen auf andere Entities geben.
4. Entities können in höchstens einem Aggregat enthalten sein.
5. Value Objects können in mehreren Aggregaten enthalten sein.
6. Ein „alleinstehendes“ Entity ist immer auch ein Aggregate Root.
7. Beim Löschen des Aggregate Root werden alle inneren Entities auch gelöscht.

E2.1 - DDD building blocks (entities, value objects)

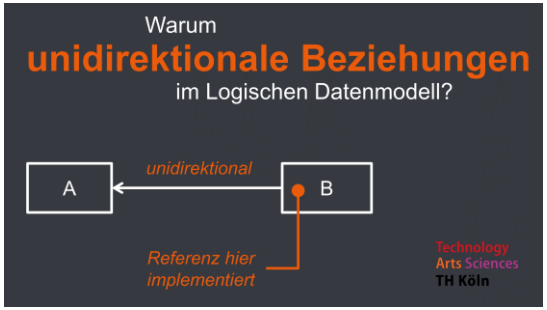
E2.1 - DDD building blocks (entities, value objects)

- Referenzen zwischen Entitäten sind NIEMALS bidirektional, sondern IMMER unidirektional
- Value Objects
  - sind immutable (unveränderlich)
  - sind gleich, wenn alle ihre Attribute gleich sind



https://www.youtube.com/watch?v=iCo4Gj_h5xQ

https://www.youtube.com/watch?v=BGyfj2ixPy8



https://www.youtube.com/watch?v=RufBXPTt1SA

E2.2 - DDD Conventions and proper Aggregate Packages (as in E1)

Services und Domain-Klassen

Application Layer

- *zustandslos*
- *regelt Domain Lifecycle (CRUD)*
- *ermöglicht Interaktionen & Workflows*

nutzt für CRUD

nutzt für Workflows

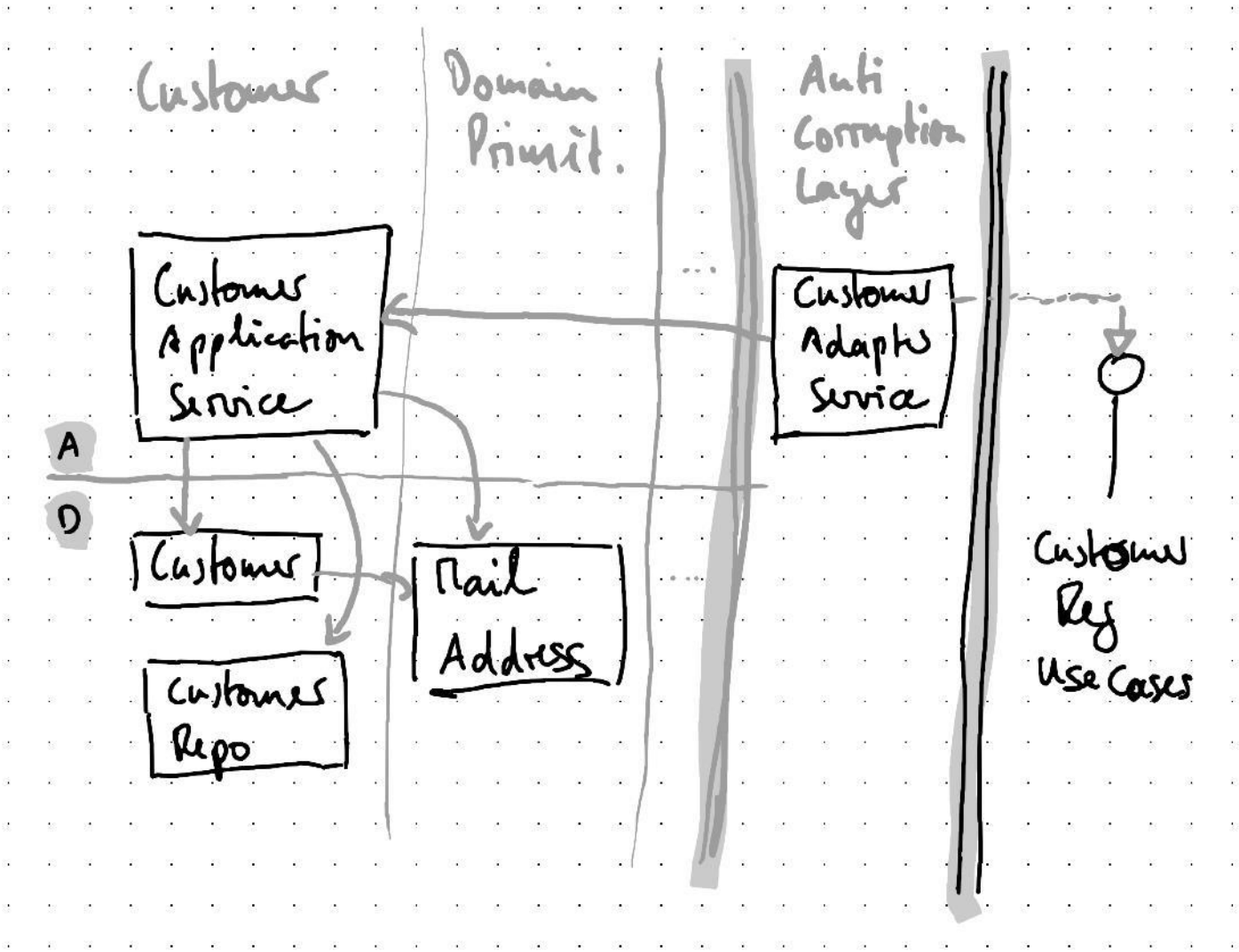
Domain Layer

- *stellt Standard-Operationen für CRUD zur Verfügung*
- *definiert Queries*

Value Object

- *haben Zustand*
- *werden auf Datenbank abgebildet*

… und das für das Beispiel „Kunde“ …



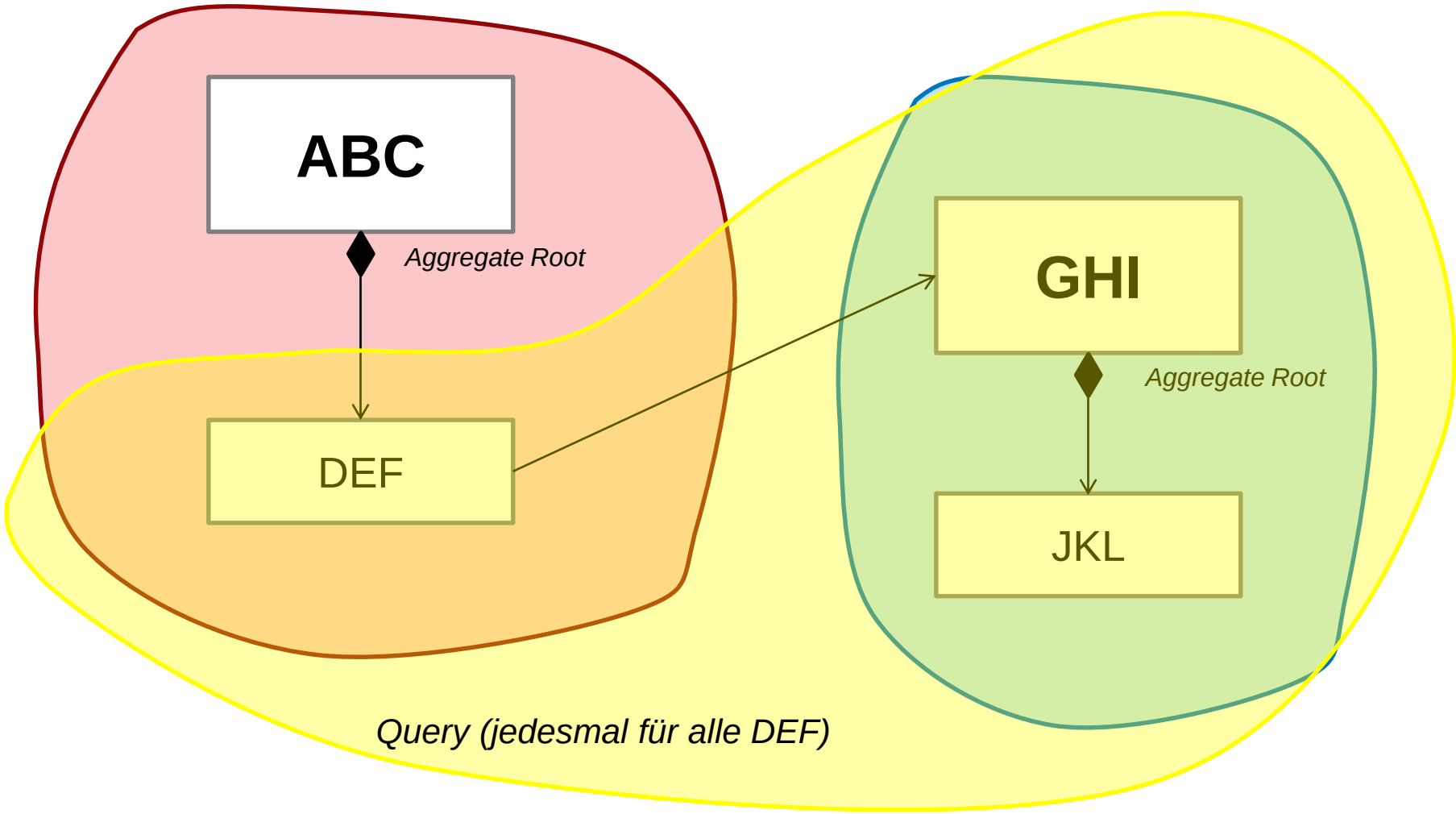
ArchiLab-Infoseiten

- Package- und Namens-Konventionen
- https://www.archi-lab.io/infopages/ddd/package-convention-ddd.html

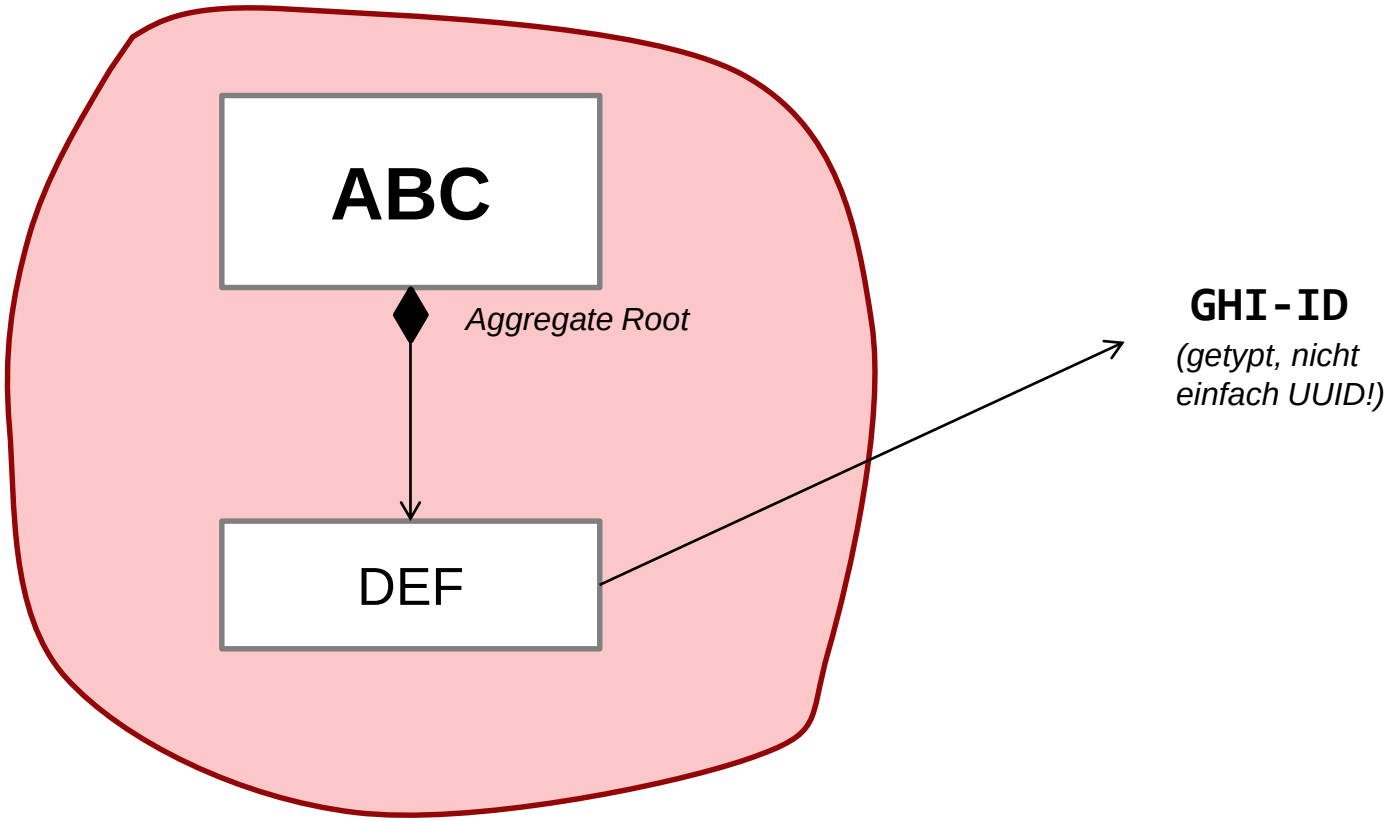
- Adapter-Pattern / Anti-Corruption-Layer
- https://www.archi-lab.io/infopages/coding/adapter-pattern.html

E2.3 - References to other Aggregates only via ID Reference

Keine Objektreferenzen außerhalb des Aggregates …



… stattdessen Referenz via (getypter) ID



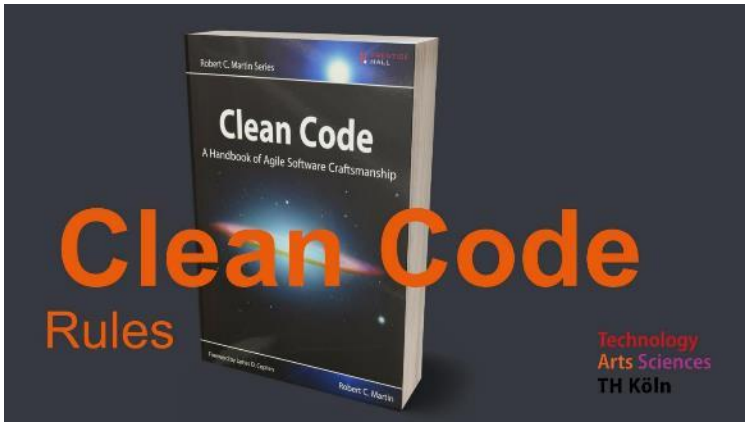
ArchiLab-Infoseiten

- Vaugn Vernon‘s 4 Rules for Aggregates
  https://www.archi-lab.io/infopages/ddd/aggregate-design-rules-vernon.html

- Verwendung getypter IDs
  https://www.archi-lab.io/infopages/spring/aggregate-references-via-ids.html

E2.4 - Clean Code and SOLID Principles

Videos // ArchiLab-Infoseiten zu Clean Code



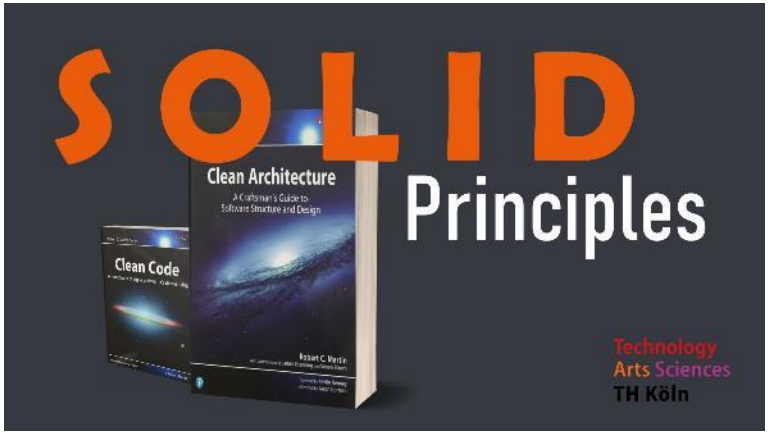
https://www.youtube.com/watch?v=2G20nqeHAn8

ArchiLab Infoseiten

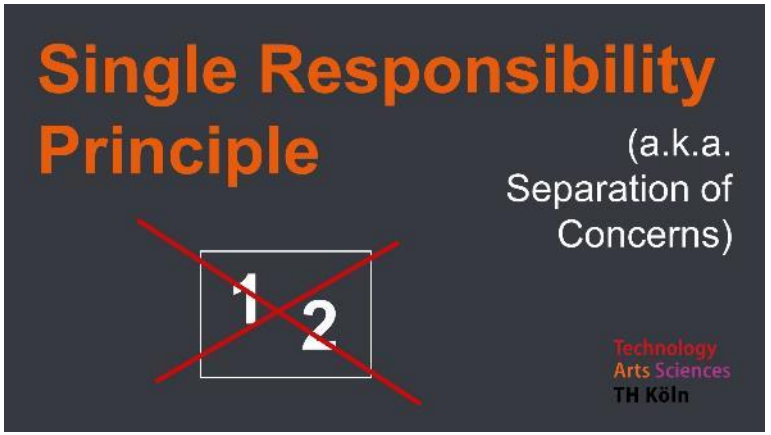
- PMD-Regel lokal in IntelliJ testen
  https://www.archi-lab.io/infopages/material/pmd-plugin.html

- Checkliste Clean Code (und SOLID)
  https://www.archi-lab.io/infopages/material/checklist-clean-code-and-solid.html

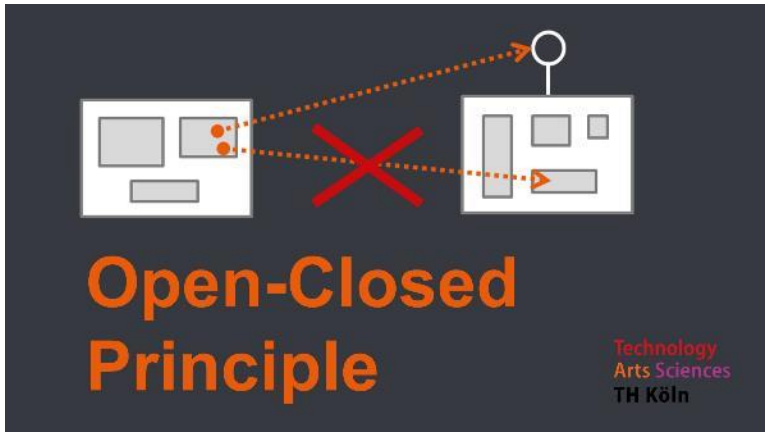
SOLID-Videos



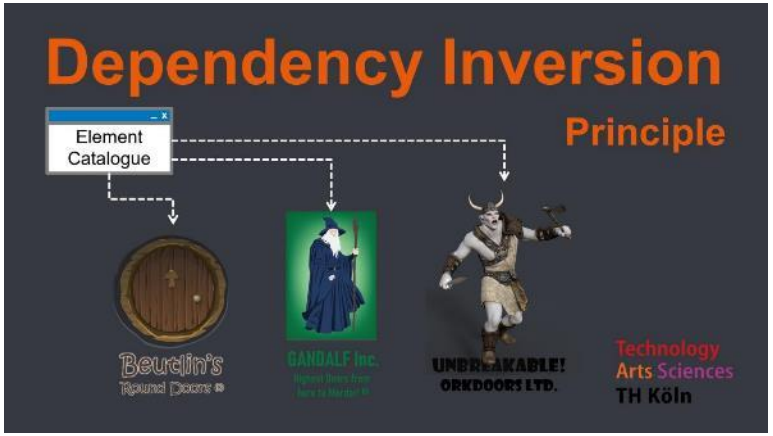
https://www.youtube.com/watch?v=kQDStoasH-Q



https://www.youtube.com/watch?v=BFVLXYFlNXg



https://www.youtube.com/watch?v=udPiJpvPsMQ s



https://www.youtube.com/watch?v=swax9LubOec

ArchiLab-Infoseiten zu SOLID

- Zusätzliche Übung (mit Beispiel-Repo) https://www.archi-lab.io/exercises/st2/bauzeicher20-cleancode-solid.html

E2.5 – No Cycles

Warum sind Zykel schlecht?

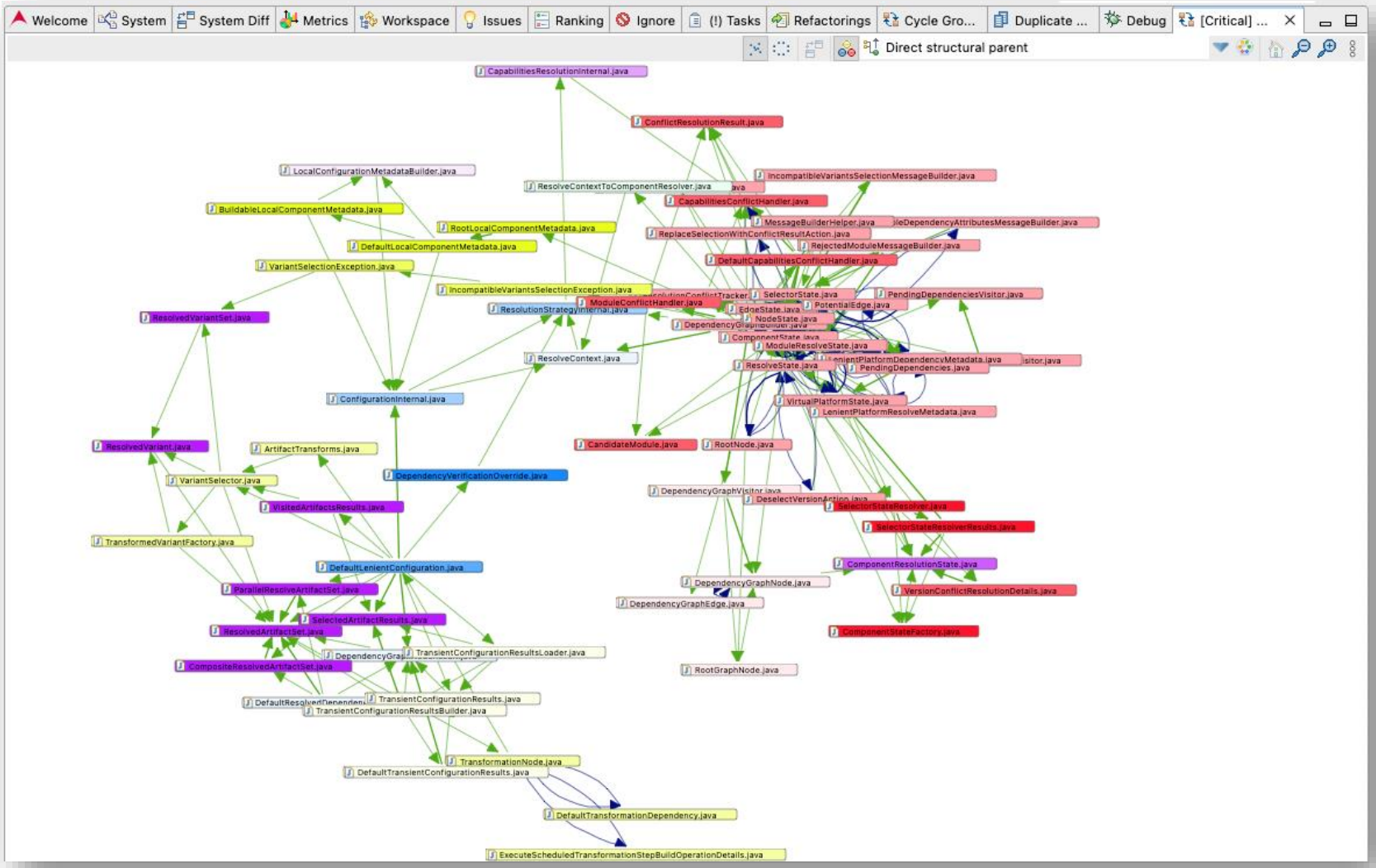
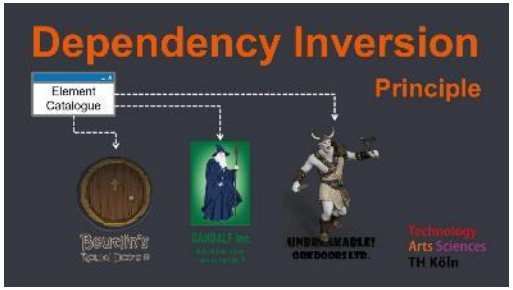


Bild: © Hello2Morrow, https://www.hello2morrow.com/products/sonargraph/architect

ArchiLab-Infoseiten zu Zykel-Auflösung



https://www.youtube.com/watch?v=swax9LubOec

Zu dem Video gibt es eine Infoseite zum Vorgehen & ein Beispielrepo „Bauzeichner 2.0“ wie im Video

- Infoseite zum Vorgehen (Entities und Services wie im Video) https://www.archi-lab.io/infopages/coding/zykel-aufloesen-mit-dip.html

- Zusätzliche Übung (mit Beispiel-Repo) https://git.archi-lab.io/public-repos/bauzeichner2.0-cleancode-solid