WPS WORKPLACE SOLUTIONS

DOMAIN-DRIVEN DESIGN HANDS-ON

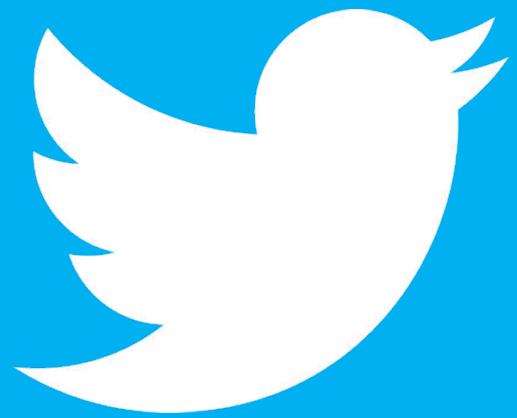
Henning Schwentner


2017
Karlsruher Entwicklertag
NOVATEC
SAP
Scrum.org
SEEBURGER


Schloss-
gärtnerei
Teehaus
Fasanengarten
Schlossgarten
Chinesische
Teehäuschen
Schloss
Botanischer
Garten
Fasanen-
schlösschen
BVerfG
Ehrenhof
Staatliche
Kunsthalle
Schlossplatz
KIT
Campus Süd
Denkmal

HENNING

SCHWENTNER



@hschwentner

speakerdeck.com/hschwentner

Quelle: http://www.schulbilder.org/malvorlage-informatiker-i10418.html

UNIVERSITÄT HAMBURG
1·9·1·9

Foto: Henning Schwentner

Fritz&
Franz&
Lothar&
Philipp.


WPS
WORKPLACE
SOLUTIONS


Die Legende von
Paul und
Paula
Mit den PUHDYS-Hits
"Wenn ein Mensch lebt"
"Geh zu ihr"

Wilhelm Paula Siegfried

Foto: LotseAWikipoedia

Inhalt

Workshop
Domain-Driven Design *konkret*

iSAQB International Software Architecture Qualification Board

wps.de/ddd

Tactical Design

Entity

Bounded Context

Building Blocks

Context Mapping

Domain Event

Aggregate

Factory

Domain Model

Value Object

Strategic Design

Ubiquitous Language

Domain Expert

Event Storming

Hexagonal Architecture

Microservices

CQRS

Event Sourcing

Kanban

Scrum

Agile

Extreme Programming

Design by Contract

lesetipps

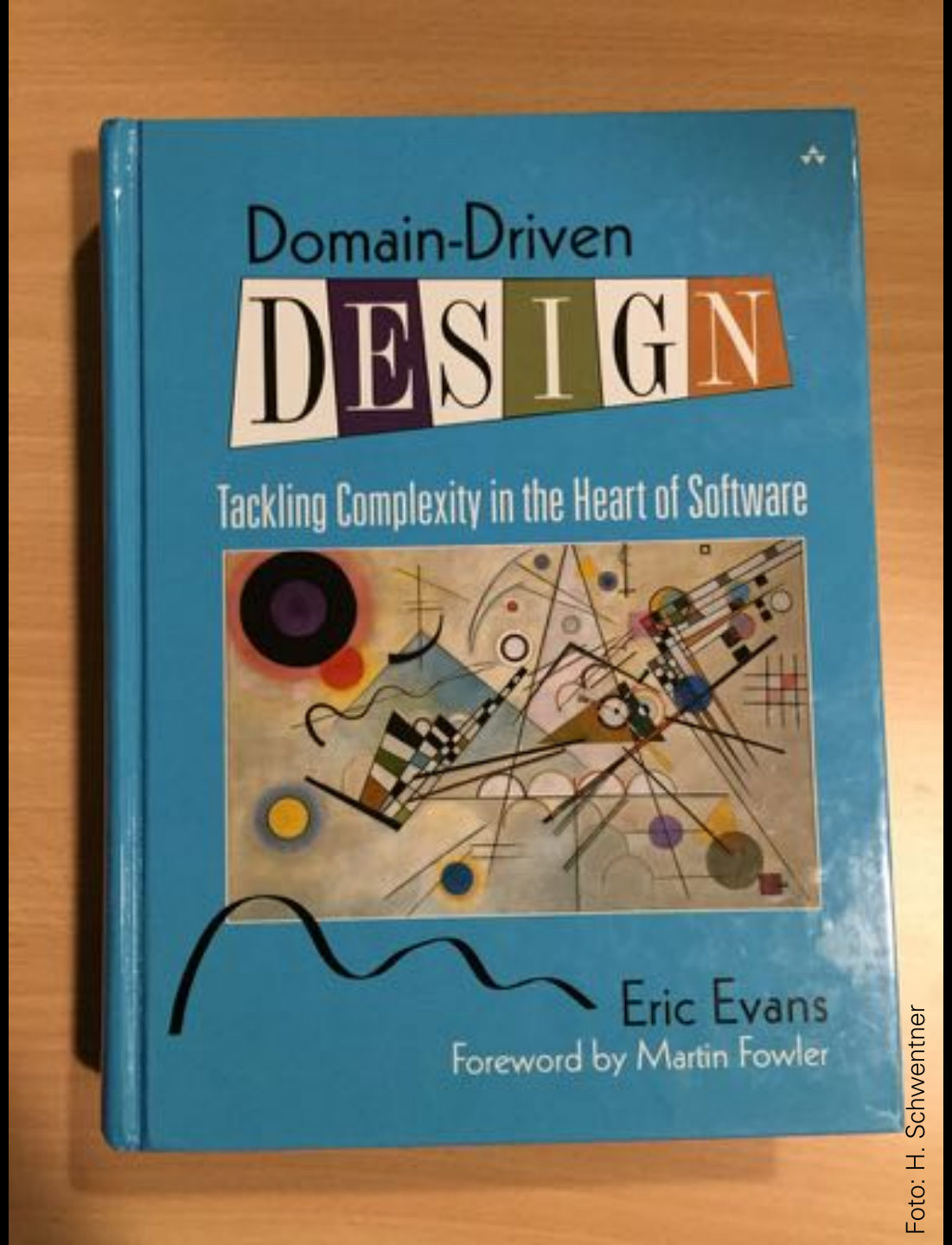


Foto: H. Schwentner

Foto: H. Schwentner

Foto: H. Schwentner

Foto: H. Schwentner

Vaughn Vernon

Domain-Driven Design kompakt

→ Aus dem Englischen übersetzt von Carola Lilienthal und Henning Schwentner

dpunkt.verlag

VORSTELLUNGSRUNDE

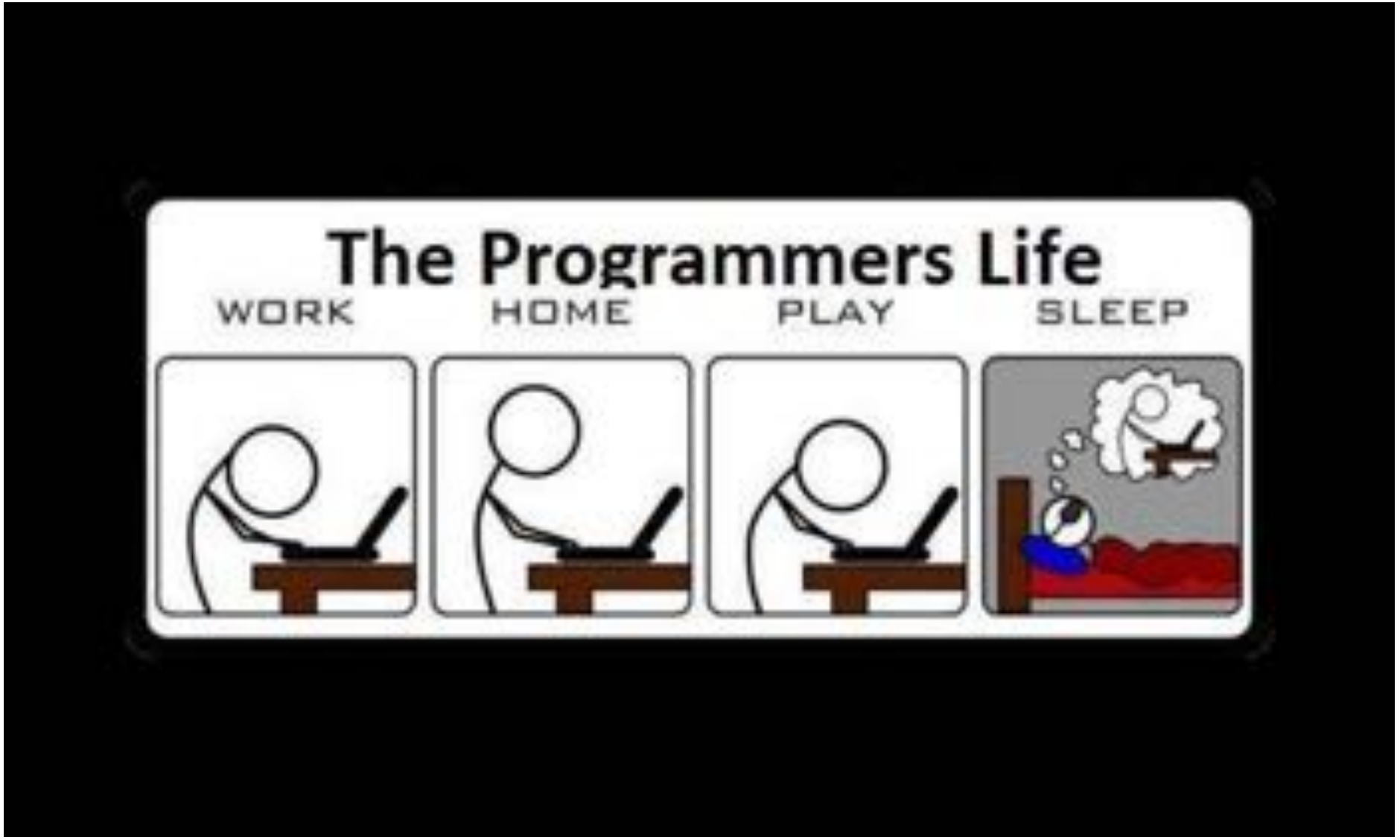
- Wer bin ich?
- Welche **Erfahrung als Entwickler** habe ich? **Programmiersprachen**?
- Was **wünsche** ich mir von der Schulung?
- Welche **Fragen** habe ich?
- Was **weiß ich schon** über DDD?

WPS WORKPLACE SOLUTIONS

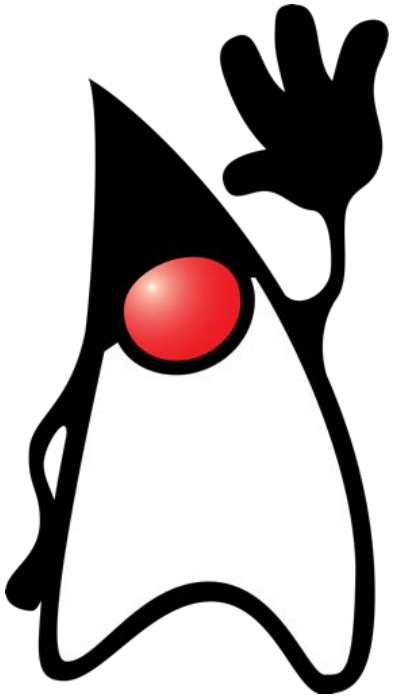
ÜBERBLICK – WAS IST DOMAIN-DRIVEN DESIGN?

Domain-Driven Design konkret

WIR SIND PROGRAMMIERER...

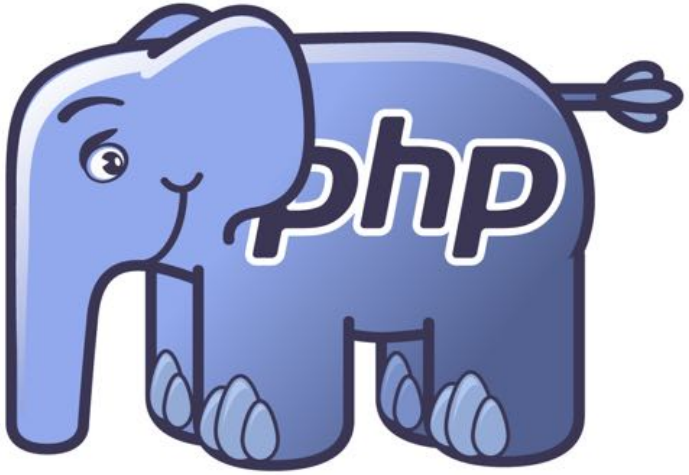


... UND WIR LIEBEN TECHNIK



spring

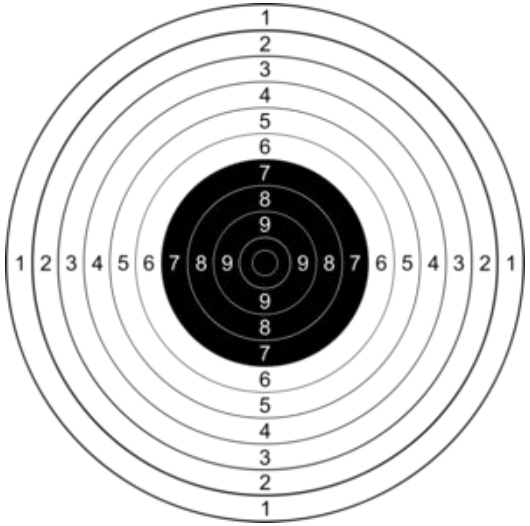
HIBERNATE



eclipse

EINFÜHRUNG

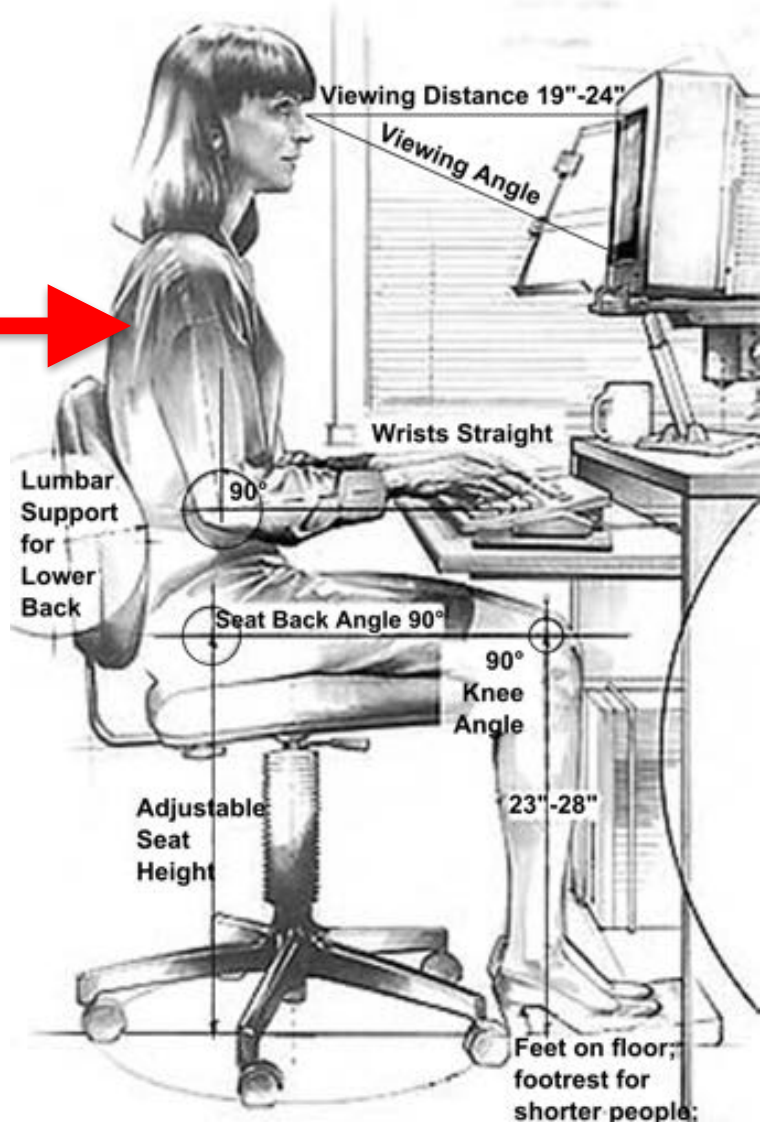
- Software ist ein **Mittel zum Zweck**
  - **Kein Selbstzweck**
  - Das Ziel ist das Ziel



- Eine produktive Software lässt sich nicht von ihrem Einsatzkontext trennen.
- Ein Softwareprodukt wird durch seine Schöpfer geprägt.

ANWENDER

WICHTIG!



IN SCHÖNHEIT STERBEN

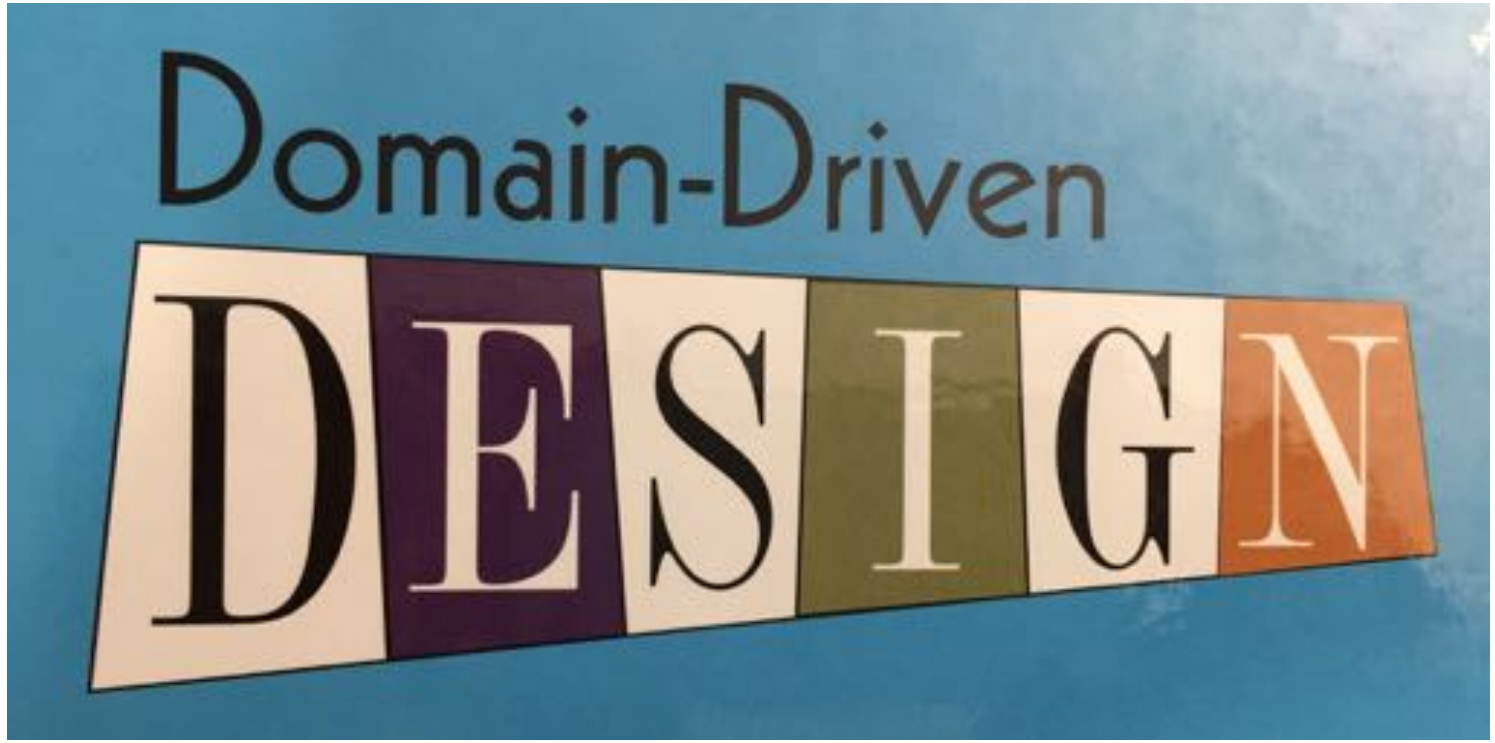
- Technische Schönheit wird uns nicht helfen, wenn wir unsere Fachexperten (*domain experts*) nicht unterstützen

Foto: Stahlkocher/Wikipedia

WAS IST DOMAIN-DRIVEN DESIGN?

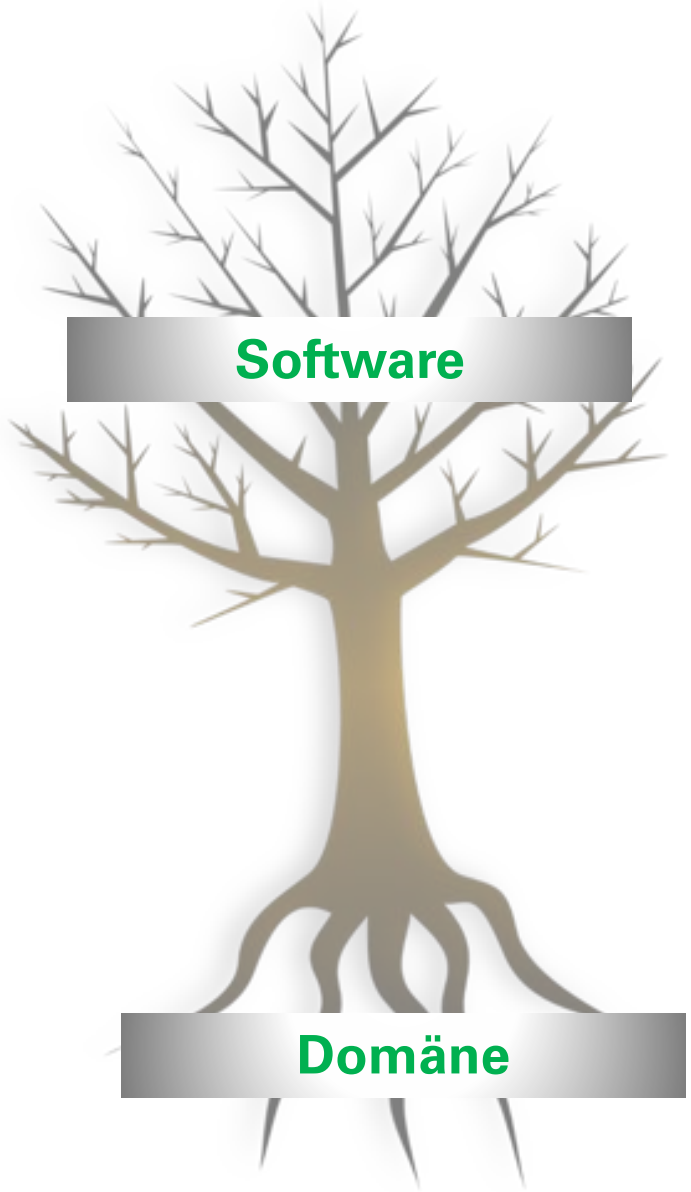
- Eine **Herangehensweise an die Entwicklung von Software**,
  - deren zentraler Bestandteil die **Implementierung eines Domänenmodells** ist.

- Es vereint
  - Entwurf
  - Entwicklungspraxis



SOFTWARE UND DOMÄNE

- Schreibe Software, die tief in der Domäne verwurzelt ist!

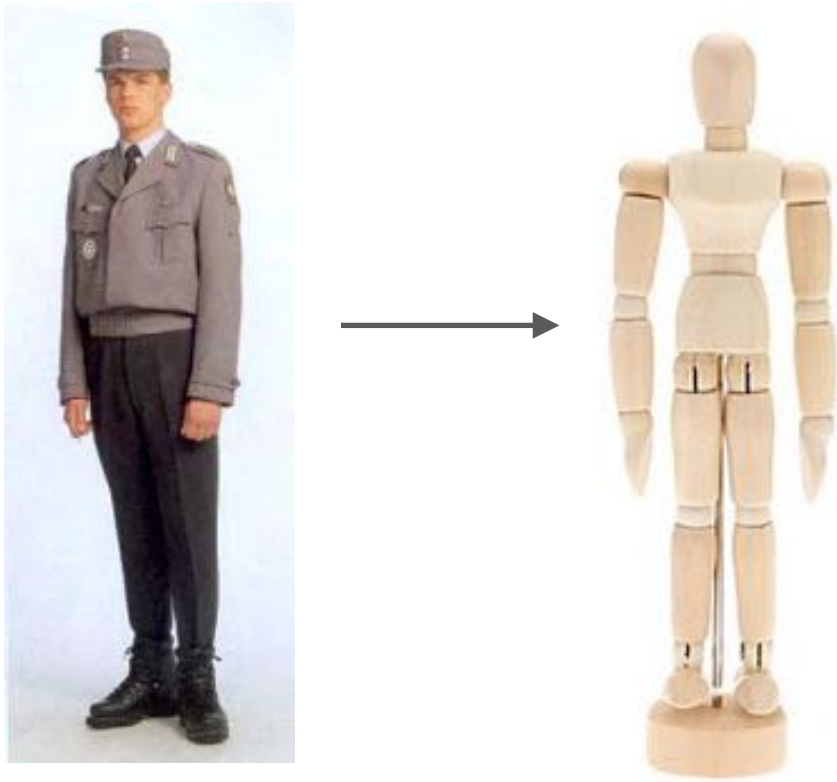


DOMAIN-DRIVEN DESIGN ANWENDEN

- Zu Beginn eines Softwareprojekts:
  - **Fokussiere die Domäne** in der die Software eingesetzt wird

- Wie erschafft man Software, die zu einer Domäne passt?
  - Begreife Software als **Reflektion** der Domäne
  - Lasse **Kernkonzepte** und **Elemente** der Domäne in die Software einfließen
  - Realisiere ihre **Zusammenhänge**

- Erstelle ein **Domänenmodell**



SOFTWARE ALS REFLEKTION DER DOMÄNE

- Die Dinge aus der Wirklichkeit werden zu Klassen
- Die Umgangsformen der Dinge werden zu Methoden

➔Wir wollen Fachliche Modelle mit reichem fachlichem Verhalten

DAS MODELL KOMMUNIZIEREN

- Das Modell kann **nicht nur in unseren Köpfen** bestehen

- Wir müssen Wissen und Informationen **verteilen**
  - Grafisch
  - Schriftlich
  - Mündlich

➔Eine gemeinsame Sprache ist notwendig, um das Modell ausdrücken zu können

➔In DDD wird diese **Allgegenwärtige Sprache** (***Ubiquitous Language***) genannt

TAKTISCHES UND STRATEGISCHES MODELLIEREN

- DDD gibt Anleitung für Modellierung **im Großen** und **im Kleinen**
- Strategisches Modellieren
  - **Teile die Domäne** in getrennte Bounded Contexts auf
  - Jeder BC hat sein eigenes Fachmodell und seine eigene Allgegenwärtige Sprache
  - Context Mapping hilft die Beziehungen zwischen Bounded Contexts zu verstehen
- Taktisches Modellieren
  - **Innerhalb** eines Bounded Context
  - **Building Blocks**: Entity,
    Value Object, Aggregate, Repository

WPS WORKPLACE SOLUTIONS

EVENT STORMING

Domain-Driven Design konkret

WAS IST EVENT STORMING?

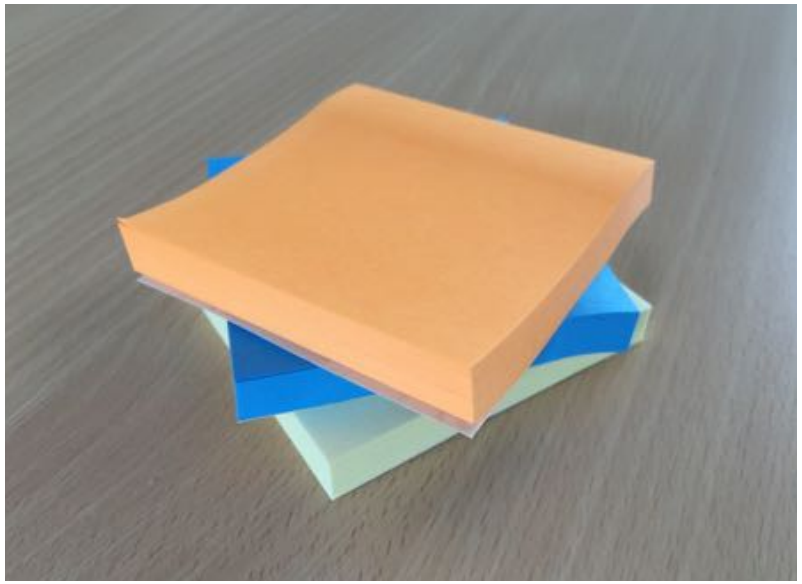
- Ein Workshopformat
- Eine Methode um
  - Die Domäne kennenzulernen
  - Geschäftsprozesse zu modellieren
- Die Spaß macht

➔Keine Diagramme
➔Keine Tische
➔Keine Stühle

@ziobrando

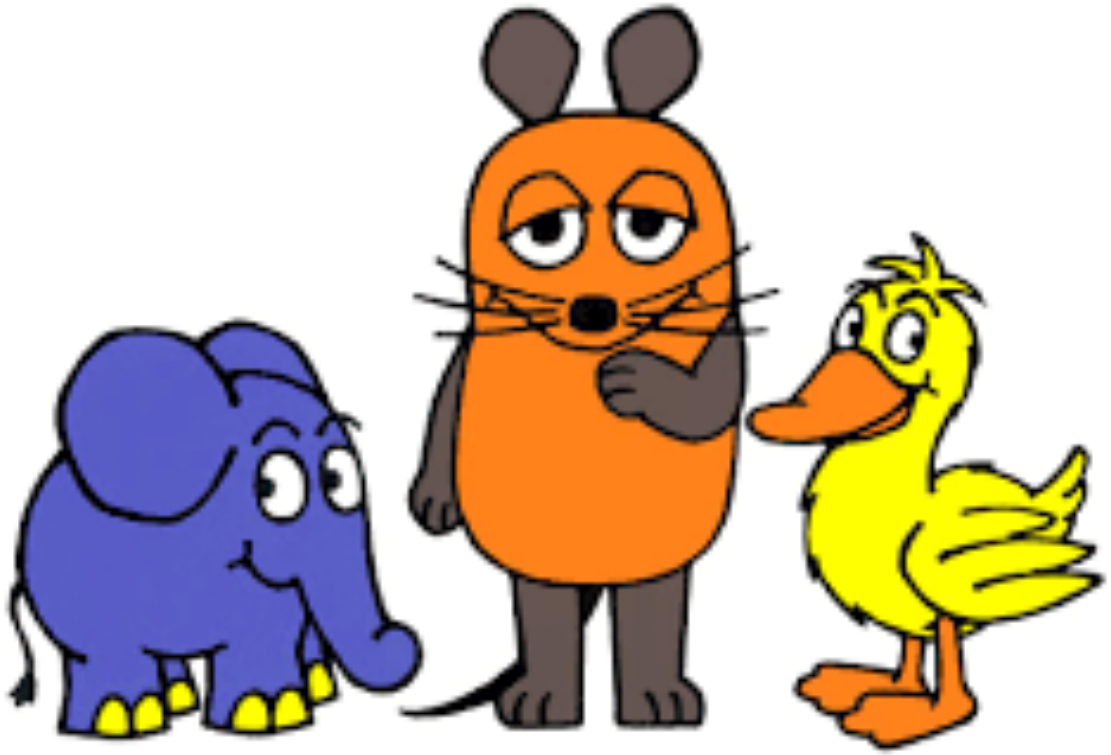
WAS MAN BRAUCHT

- Die richtigen Leute: Fachexperten und Entwickler
- Einen offenen Geist
- Eine Sammlung Sticker in verschiedenen Farben
- Eine leere Wand die ca. 10 Meter lang ist
- Eine lange Rolle Papier, die man auf der Wand befestigen kann



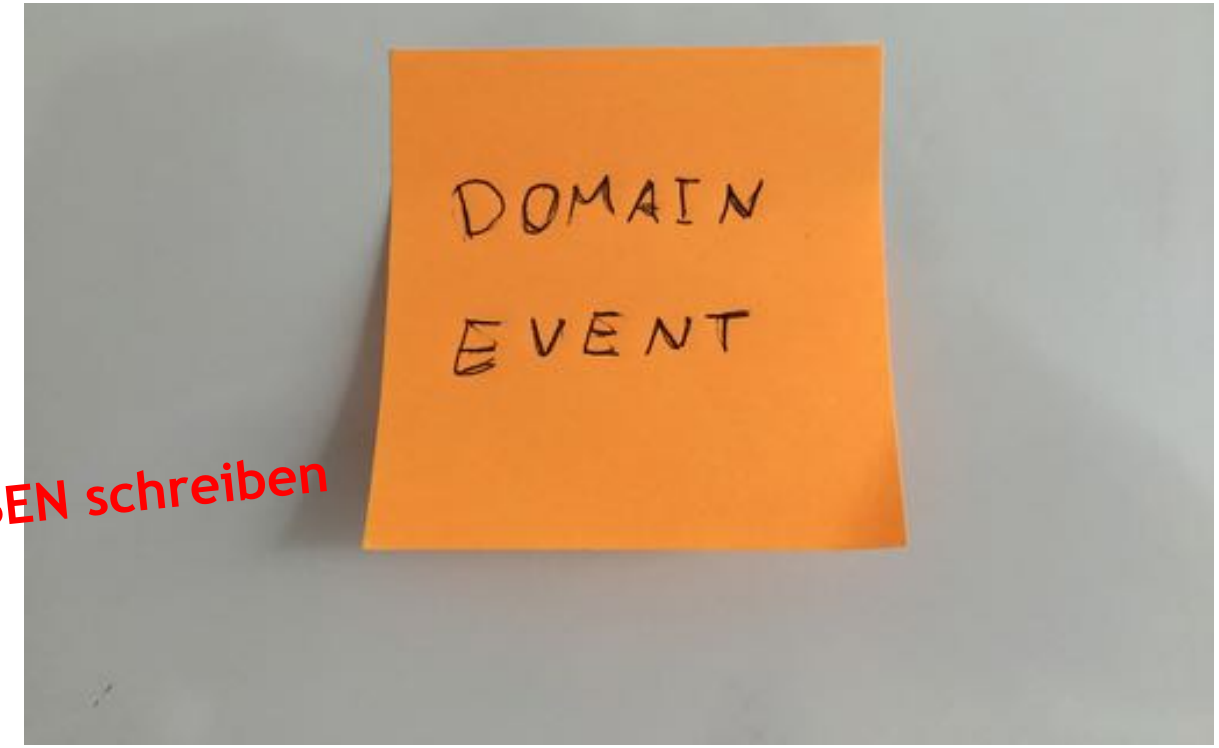
DIE DREI SCHRITTE

1. Erzeuge eine Serie von Fachlichen Ereignissen (Domain Events)
2. Erzeuge die Commands, die die Fachlichen Ereignisse verursachen
3. Füge die Entity/Aggregate hinzu, auf dem das Command ausgeführt wird



1. ERZEUGE EINE SERIE VON DOMAIN EVENTS

- Arbeite den Geschäftsprozess als eine Serie von Domänen Ereignissen aus
  - Orange Stickies
  - Verben in der **Vergangenheitsform**
  - **Relevant** für die Fachexperten

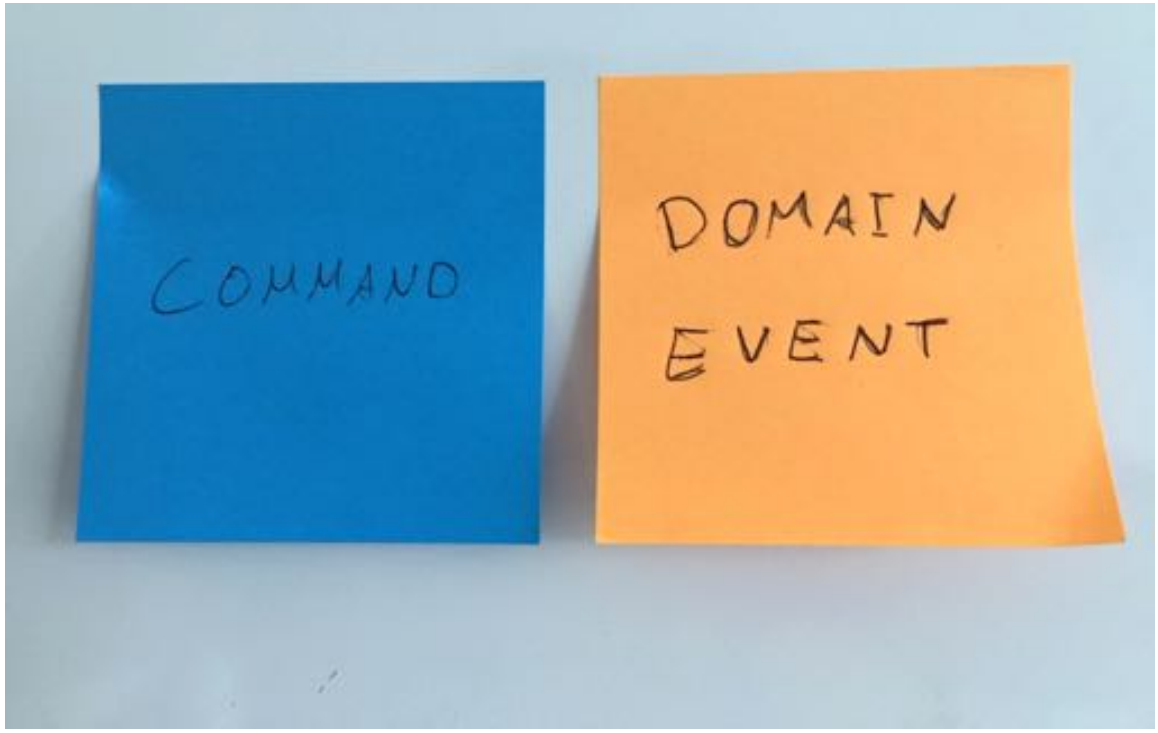


Tipp: in GROßBUCHSTABEN schreiben

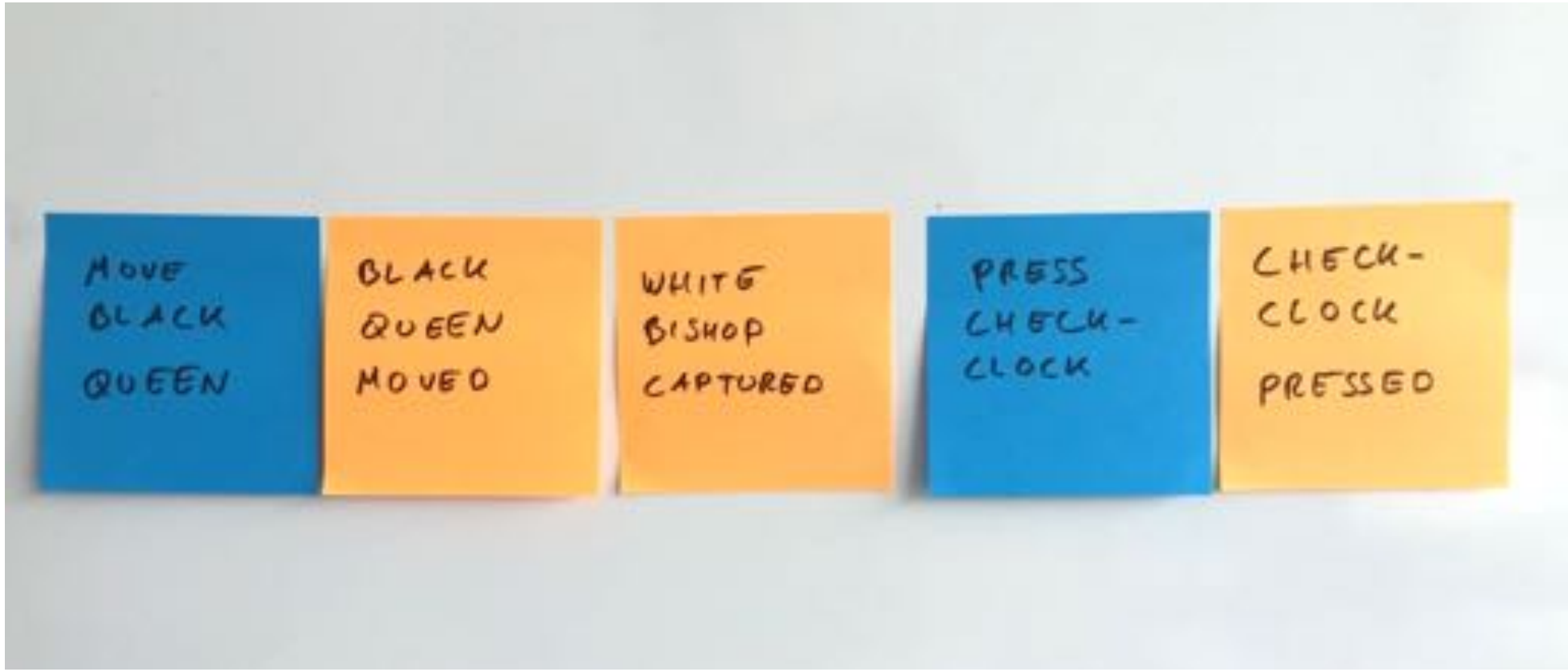
1. ERZEUGE EINE SERIE VON DOMAIN EVENTS – BEISPIEL

2. ERZEUGE DIE VERANTWORTLICHEN COMMANDS

- Für jedes Domain Event erzeugt man einen Command, der ihn verursacht
  - **Blaue** Stickies
  - Im **Imperativ**
  - Die Akteur kommen auf kleine gelbe Stickies



2. ERZEUGE DIE VERANTWORTLICHEN COMMANDS – BEISPIEL

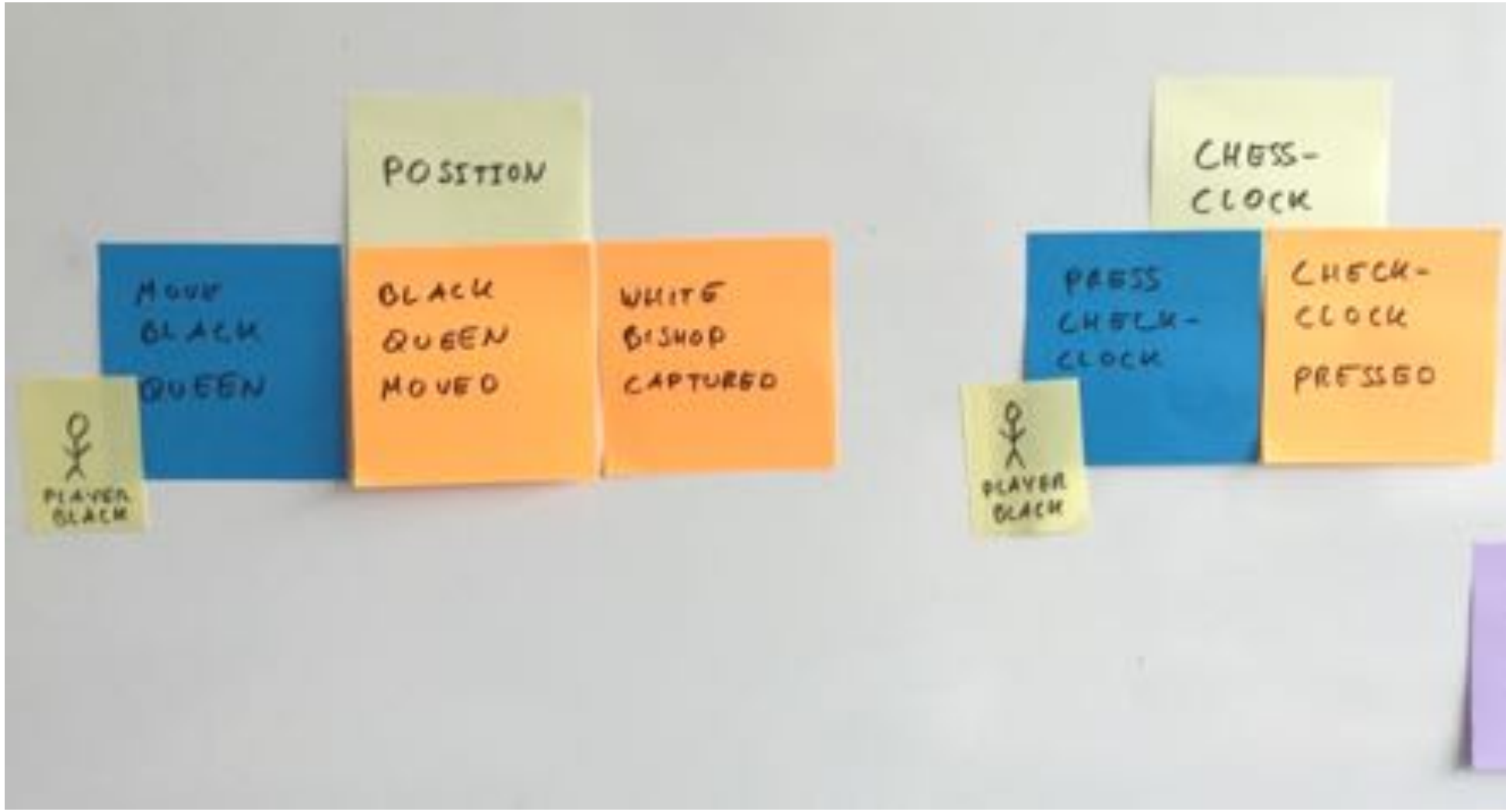


2. ERZEUGE DIE VERANTWORTLICHEN COMMANDS – MIT DEN AKTEUREN

3. DAZUGEHÖRIGE AGGREGATE

- Ordne das Aggregate oder die Entity dem Command zu, der an ihm ausgeführt wird, und dem Domain Event zu, das es produziert.
  - Helles gelb
  - Hinter den Stickies von Command und Domain Event

3. ORDNE ENTITY/AGGREGATE ZU – BEISPIEL



ÜBUNG: EVENT STORMING IM KINO

WPS WORKPLACE SOLUTIONS

UBIQUITOUS LANGUAGE

Domain-Driven Design konkret

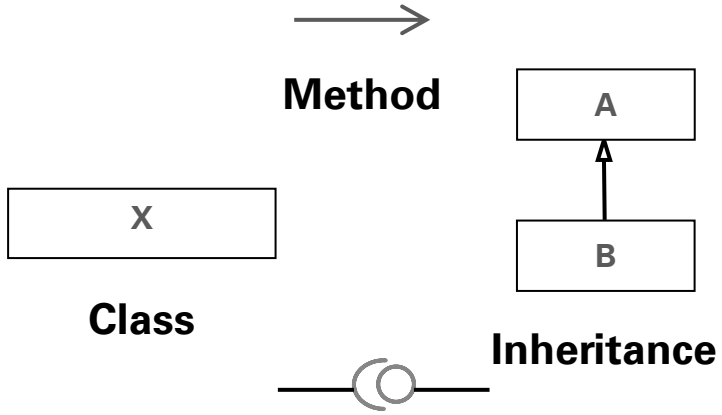
“There are only two hard things in Computer Science: cache invalidation and naming things.”

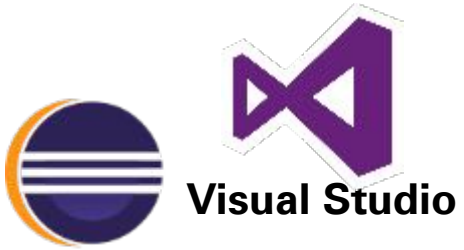
Phil Karlton

“There are only two hard things in Computer Science:
cache invalidation, naming things,
and off-by-one errors.”

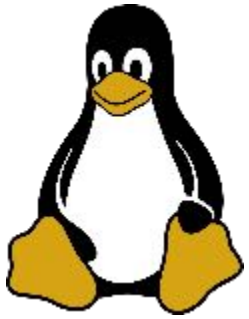
Phil Karlton

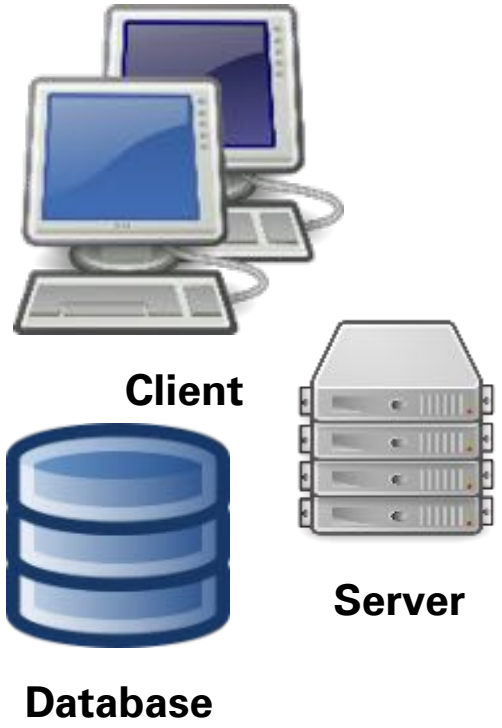
TECHNISCHE SPRACHE





O/R-Mapping





DOMÄNEN-SPRACHE – BEISPIEL SCHIFFAHRT

Kran

Container

Frachtbrief

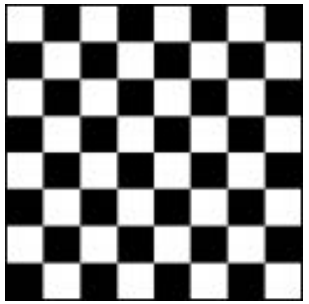
4300

Containernummer

Twistlock

DOMÄNEN-SPRACHE – BEISPIEL SCHACH

Figuren



Brett

Spieler

Schachuhr

König

DOMÄNEN-SPRACHE – BEISPIEL SCHACH II

Schachmatt

Spiel

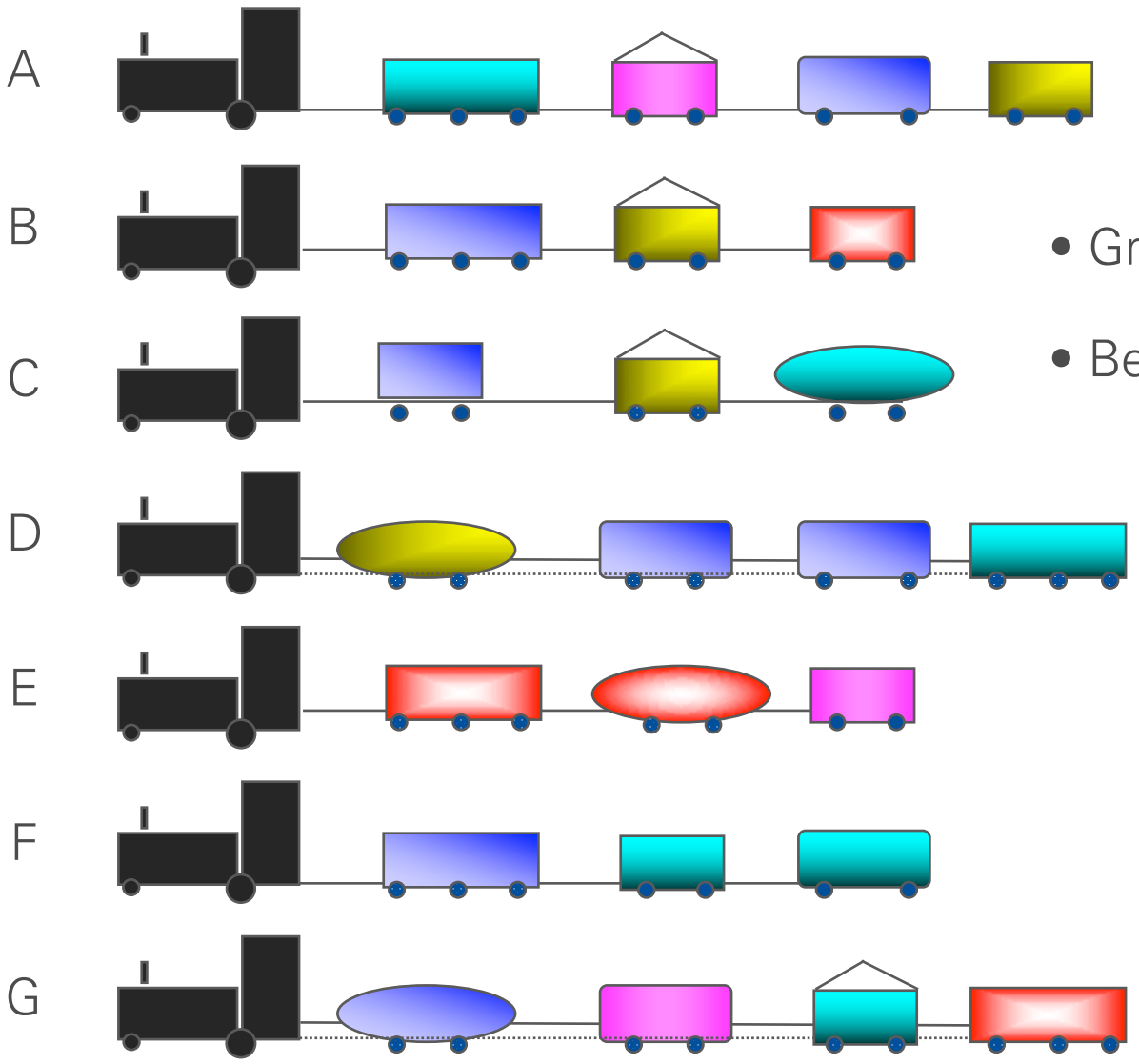
Remis

Zug

Rochade

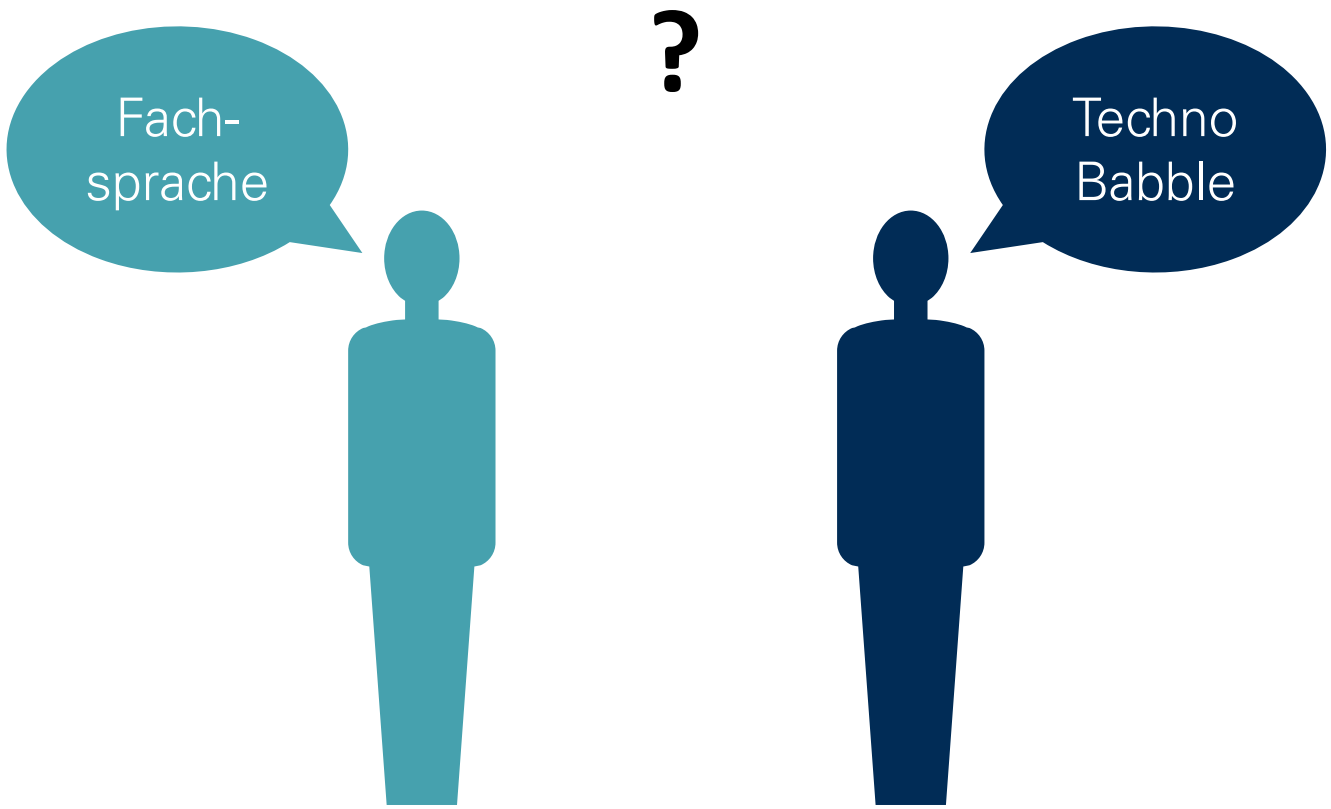
Zugzwang

KATEGORISIERUNG ANHAND EINES BEISPIELS

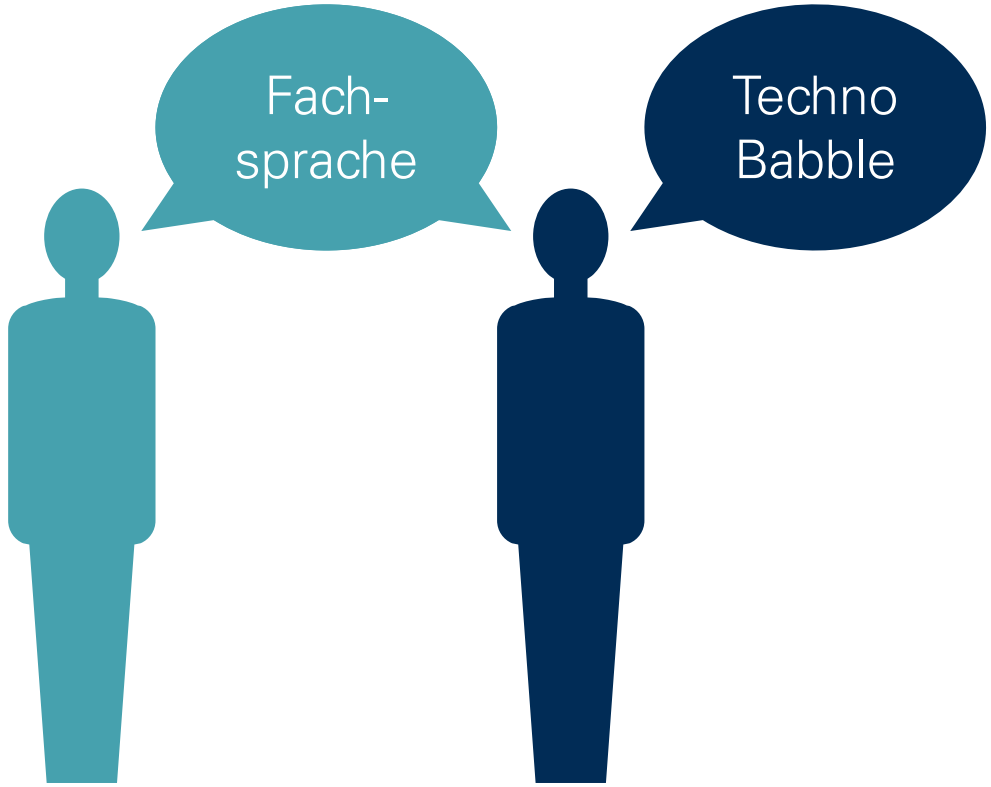


- Gruppiere
- Begründe die Gruppierung

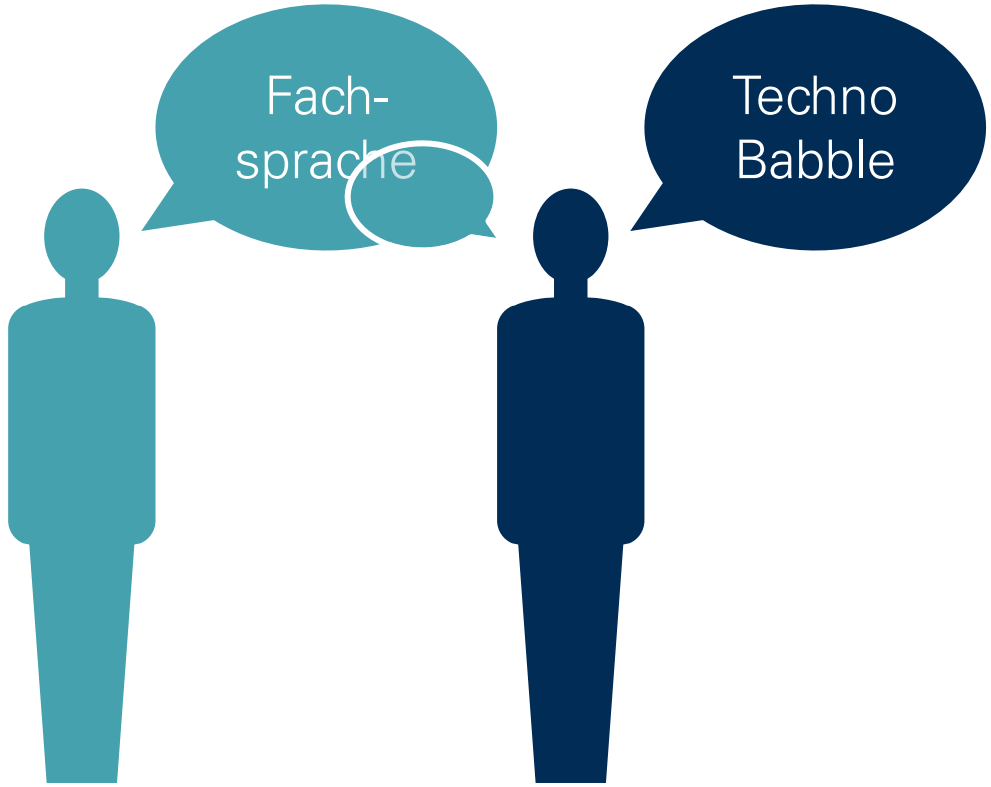
MOTIVATION ALLGEGENWÄRTIGE SPRACHE



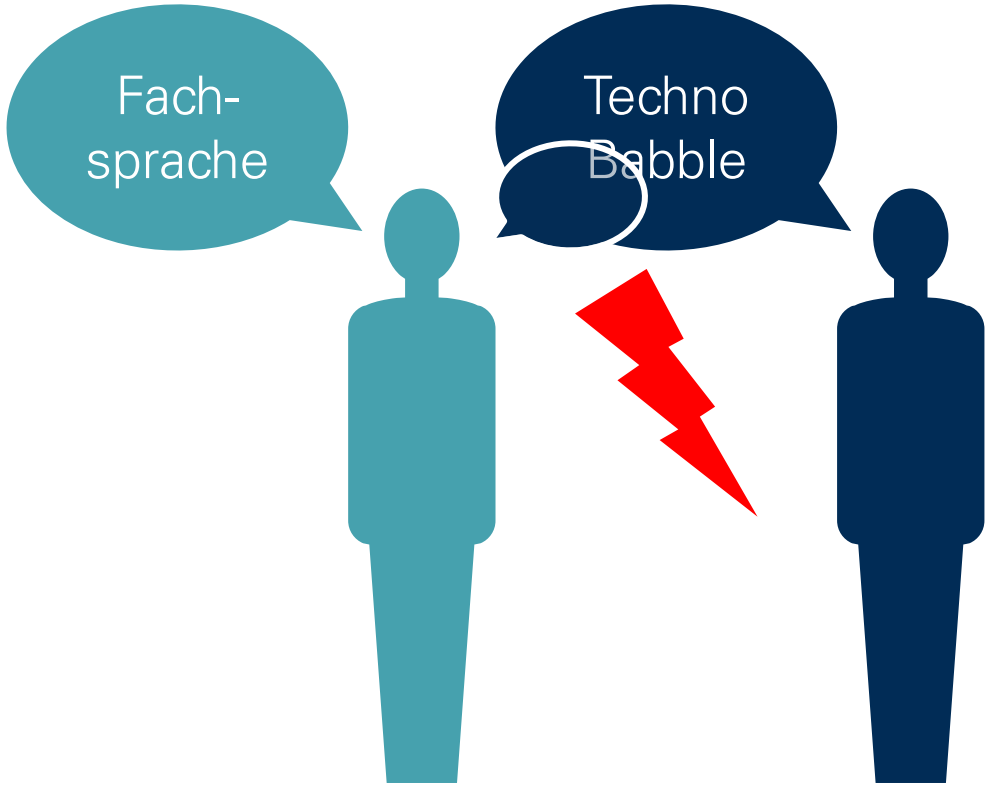
MOTIVATION ALLGEGENWÄRTIGE SPRACHE



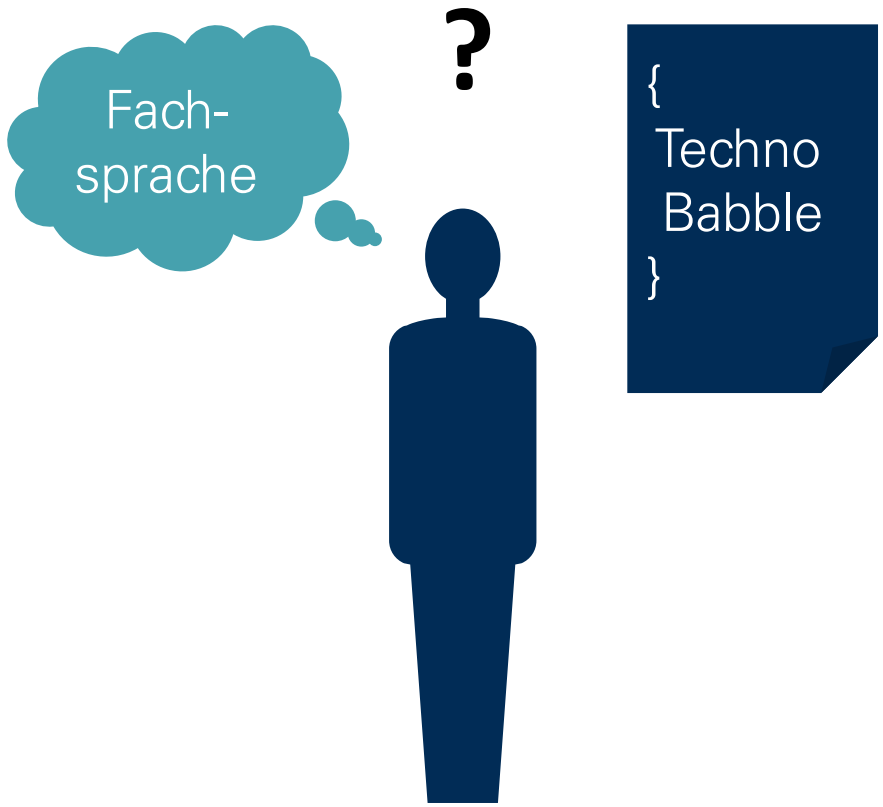
MOTIVATION ALLGEGENWÄRTIGE SPRACHE



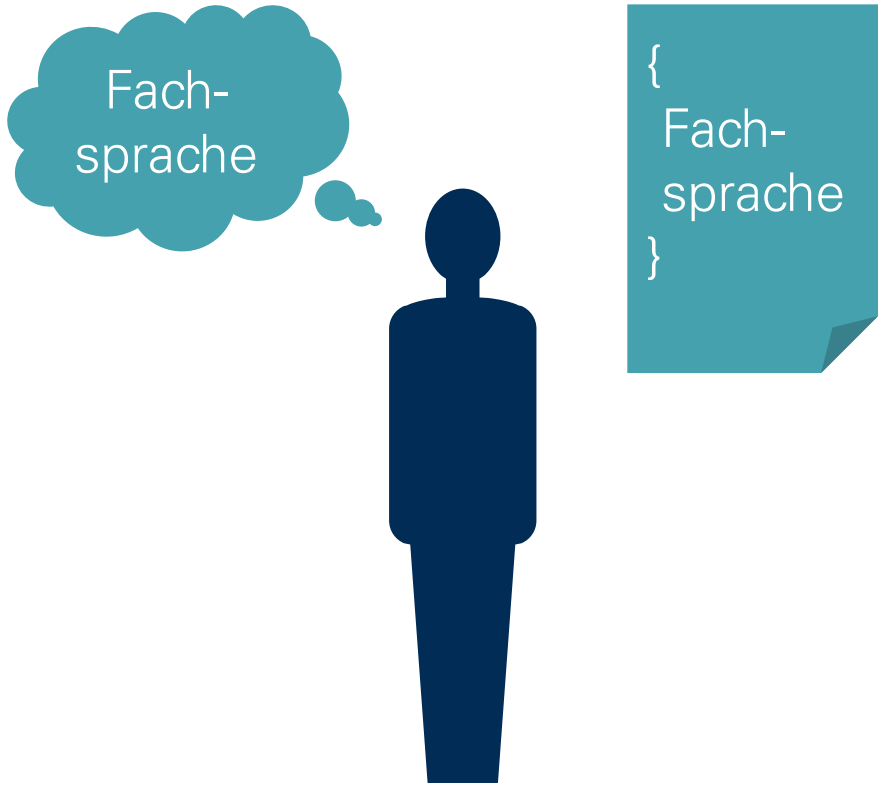
MOTIVATION ALLGEGENWÄRTIGE SPRACHE



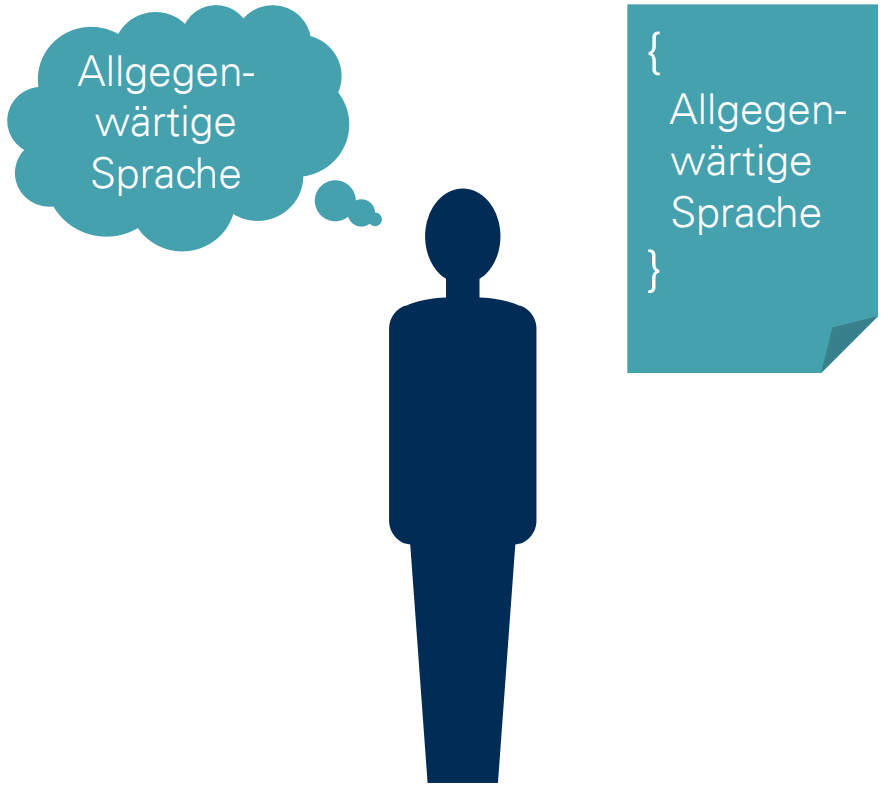
MOTIVATION ALLGEGENWÄRTIGE SPRACHE



MOTIVATION ALLGEGENWÄRTIGE SPRACHE



MOTIVATION ALLGEGENWÄRTIGE SPRACHE



GEMEINSAME SPRACHE

- Fachexperten verstehen keine Begriffe zu technischen Umsetzungen
- Fachexperten sprechen den Jargon ihrer Domäne, der für Außenstehende wiederum schwer verständlich sein kann

➔ Eine **gemeinsame Sprache** ist notwendig!

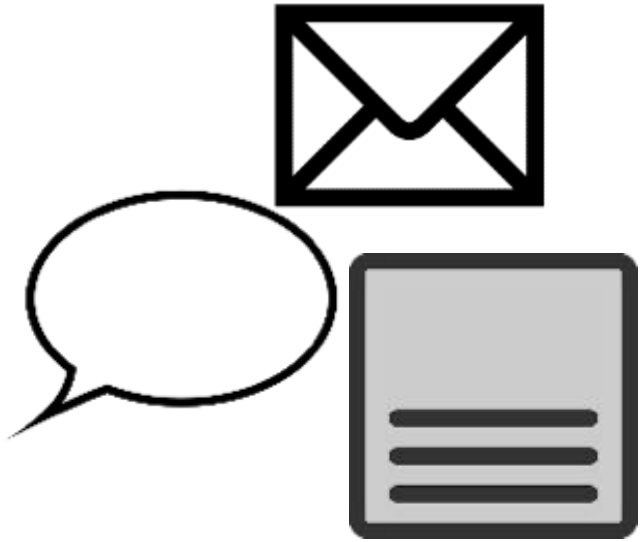
- Welche soll es sein?
  - Die der Entwickler?
  - Die der Fachexperten?
  - Etwas dazwischen?

➔Prinzip von DDD:

„Verwende **eine Sprache die auf dem Domänenmodell basiert**“

ALLGEGENWÄRTIGE SPRACHE

- Verwende die gemeinsame Sprache in..
  - der **Kommunikation**
    - mündlich
    - schriftlich
    - grafisch
  - im **Code**



  - **Im Grunde genommen überall**
    - ➔ Deswegen nennt sich die Sprache *allgegenwärtig*.

SPRACHE TAUCHT NICHT EINFACH AUF

- Es braucht **Wochen bis Monate...**
  - harter Arbeit
  - und scharfem Fokus
- … um die **Schlüsselkonzepte** offenzulegen.

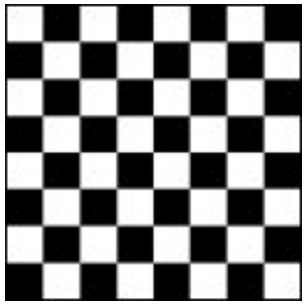
- Die ersten Wörter einer allgegenwärtigen Sprache kommen üblicherweise **direkt aus der Domäne**
- Im Laufe der Entwicklung werden **neue Begriffe** definiert und hinzugefügt

DOMÄNEN-SPRACHE

Figuren

Schachuhr

Spieler



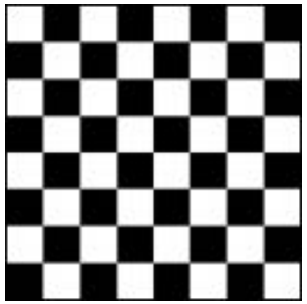
Brett

König

ALLGEGENWÄRTIGE SPRACHE

Figuren

Spieler

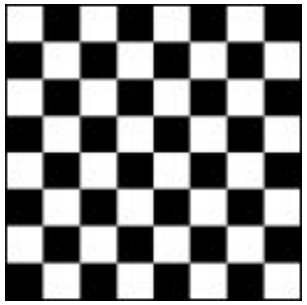


Brett

ALLGEGENWÄRTIGE SPRACHE – ITERATION 2

Figuren

Spieler



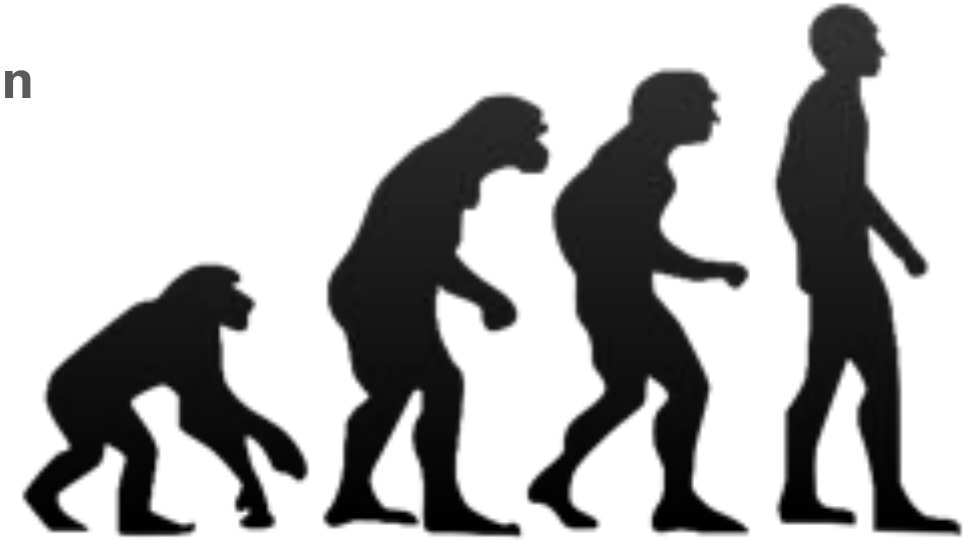
Brett

Nichtmenschlicher Spieler

SPRACHEN SIND LEBENDIG

- **Experimentiere** mit alternativen Ausdrucksformen
  - Das Modell und die Sprache **entwickeln sich weiter**
- **Überarbeite** dann den Code
  - **Benenne** Klassen, Methoden, Module
  - **Entspreche** dem neuen Modell

- Eine Sprache will gesprochen werden:
  - Beseitige Unklarheiten durch **Konversation**

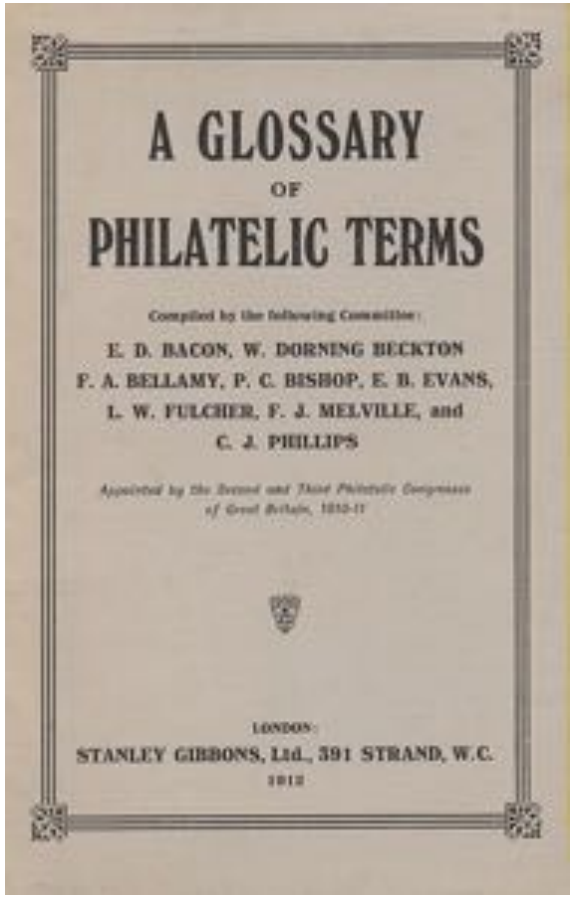


GLOSSAR

- Fachsprache der Benutzer/Ubiquitous Language
  - bereits existierende Begriffe
  - rekonstruierte Begriffe
  - neue Begriffsbildungen
- **Wer tut was damit wozu**?

- Kernkonzepte

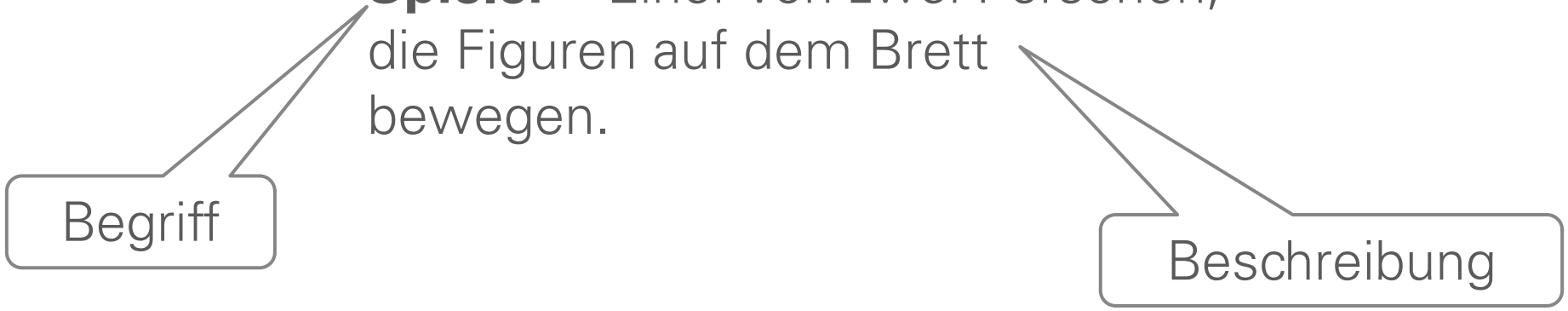
- Wichtiger am Anfang des Projektes
- Oft Wegwerfprodukt


A GLOSSARY
OF
PHILATELIC TERMS
Compiled by the following Committee:
E. D. BACON, W. DORNING BECKTON
F. A. BELLAMY, P. C. BISHOP, E. B. EVANS,
L. W. FULCHER, F. J. MELVILLE, and
C. J. PHILLIPS
Appointed by the Second and Third Philatelic Congresses of Great Britain, 1910-11
LONDON:
STANLEY GIBBONS, Ltd., 391 STRAND, W.C.
1912

GLOSSAR – BEISPIELDOMÄNE SCHACH

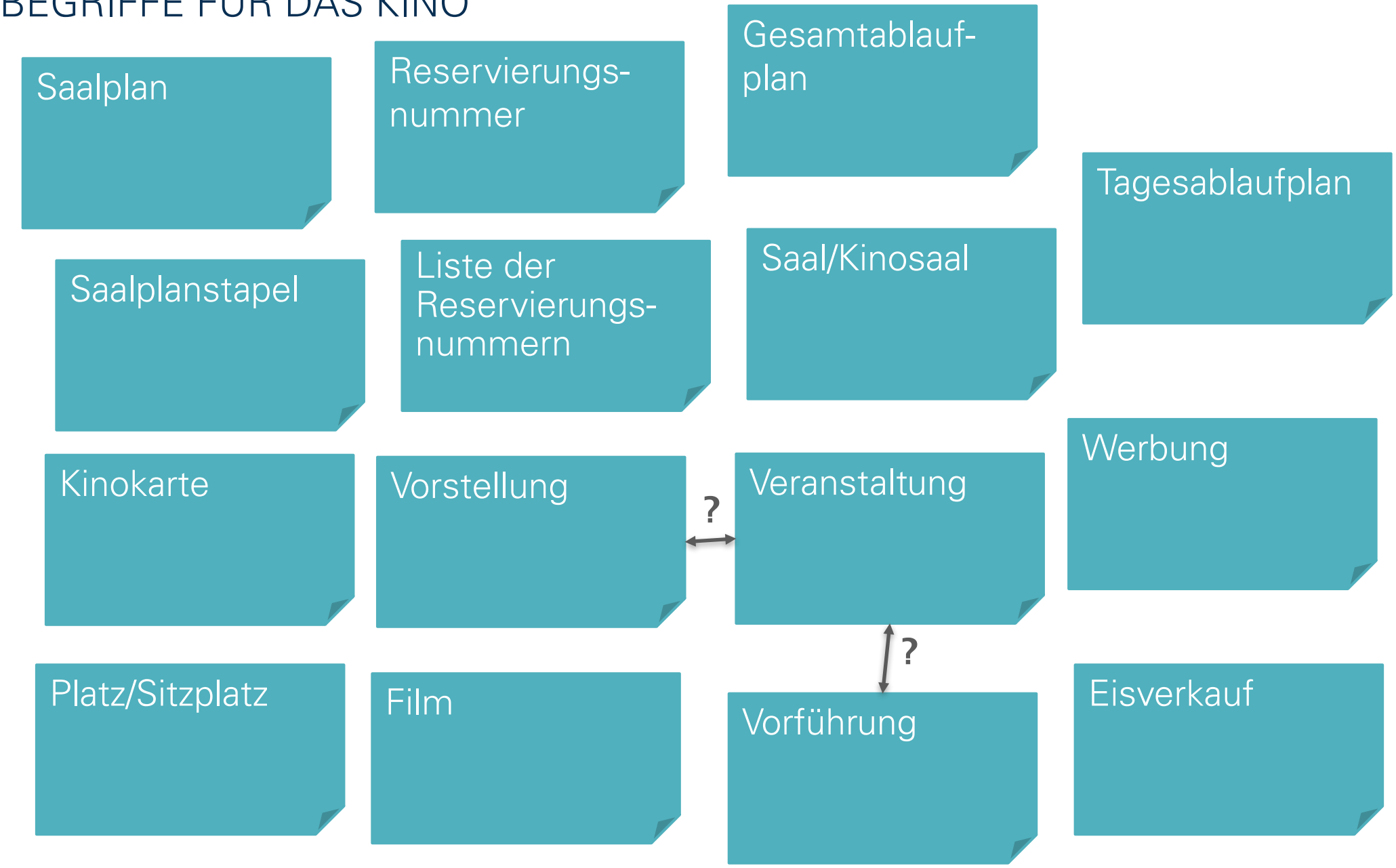
König – eine besondere Figur, die nur ein Feld pro Zug bewegt werden kann. Wenn der König schachmatt gesetzt wurde, ist das Spiel vorbei.

Spieler – Einer von zwei Personen, die Figuren auf dem Brett bewegen.



FACHLICHE KERNBEGRIFFE IM IST IDENTIFIZIEREN

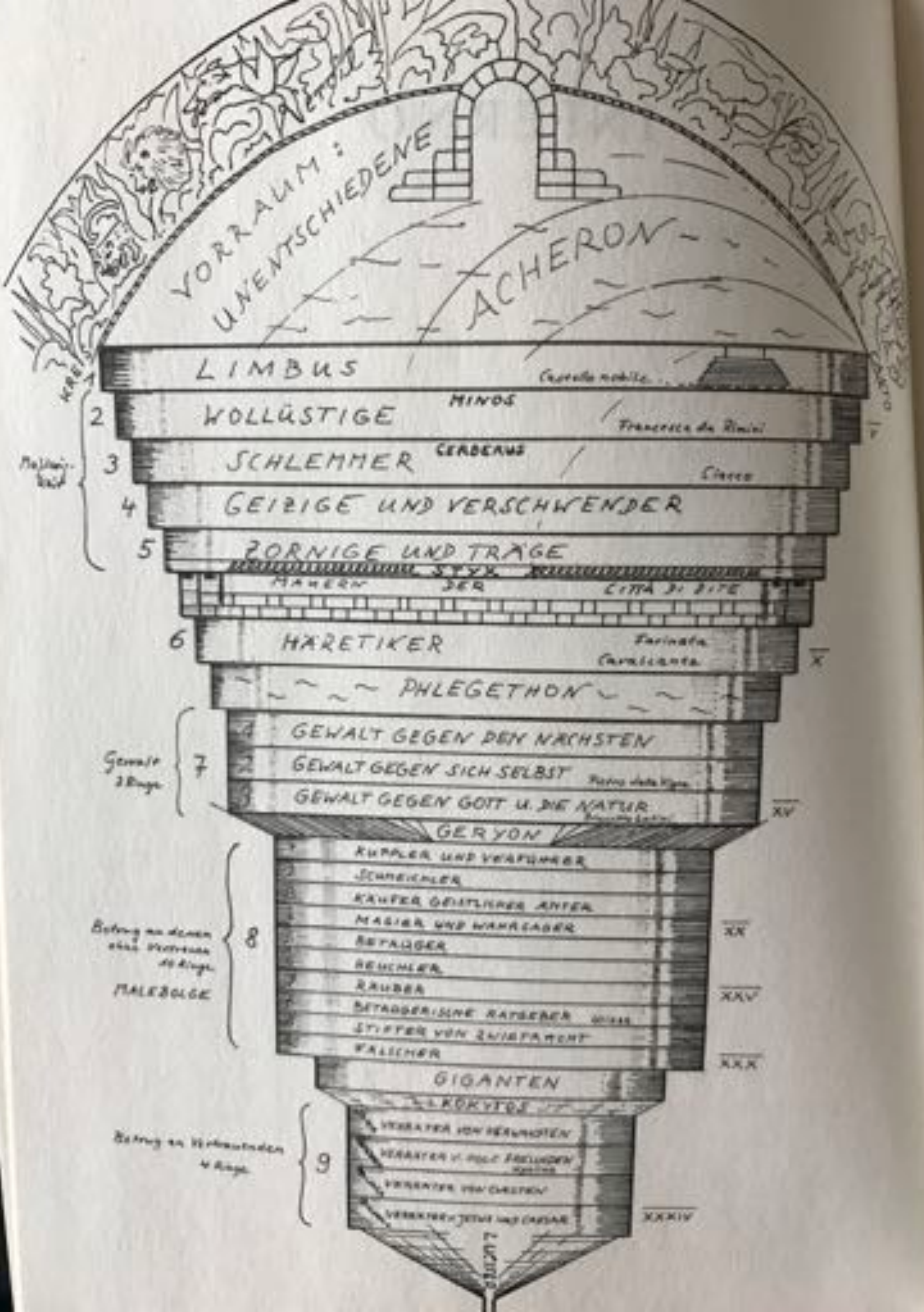
KERNBEGRIFFE FÜR DAS KINO



WPS WORKPLACE SOLUTIONS

STRATEGISCHES DESIGN

Domain-Driven Design konkret



VORRAUM :
UNENTSCHIEDENE
ACHERON
LIMBUS
Castello nobile
2
WOLLÜSTIGE
MINOS
Francesca da Rimini
3
SCHLEMMER
CERBERUS
Ciacco
4
GEIZIGE UND VERSCHWENDER
5
ZORNIGE UND TRÄGE
STYX
MAUERN DER CITTÀ DI DITE
6
HÄRETIKER
Farinata
Cavalcante
X
PHLEGETHON
Gewalt
3 Ringe
7
GEWALT GEGEN DEN NÄCHSTEN
GEWALT GEGEN SICH SELBST
Pietro della Vigna
GEWALT GEGEN GOTT U. DIE NATUR
XV
GERYON
Betrug an denen ohne Vertrauen
10 Ringe
MALEBOLGE
8
KUPPLER UND VERFÜHRER
SCHMEICHLER
KÄUFER GEISTLICHER ÄMTER
MAGIER UND WAHRSAGER
XX
BETRÜGER
HEUCHLER
RÄUBER
XXV
BETRÜGERISCHE RATGEBER
STIFTER VON ZWIETRACHT
FÄLSCHER
XXX
GIGANTEN
KOKYTOS
Betrug an Vertrauten
4 Ringe
9
VERRÄTER VON VERWANDTEN
VERRÄTER V. POLIT. FREUNDEN
VERRÄTER VON GÄSTEN
XXXIV
LUCIFER

MODELLKONSISTENZ

- Grundsätzliche Anforderung an jedes Domänenmodell: **Konsistenz**
  - **Begriffe** haben immer und überall die **selbe Bedeutung**
  - **Keine Widersprüche**

- **Vereinheitlichung** = Die **interne** Konsistenz eines Modells

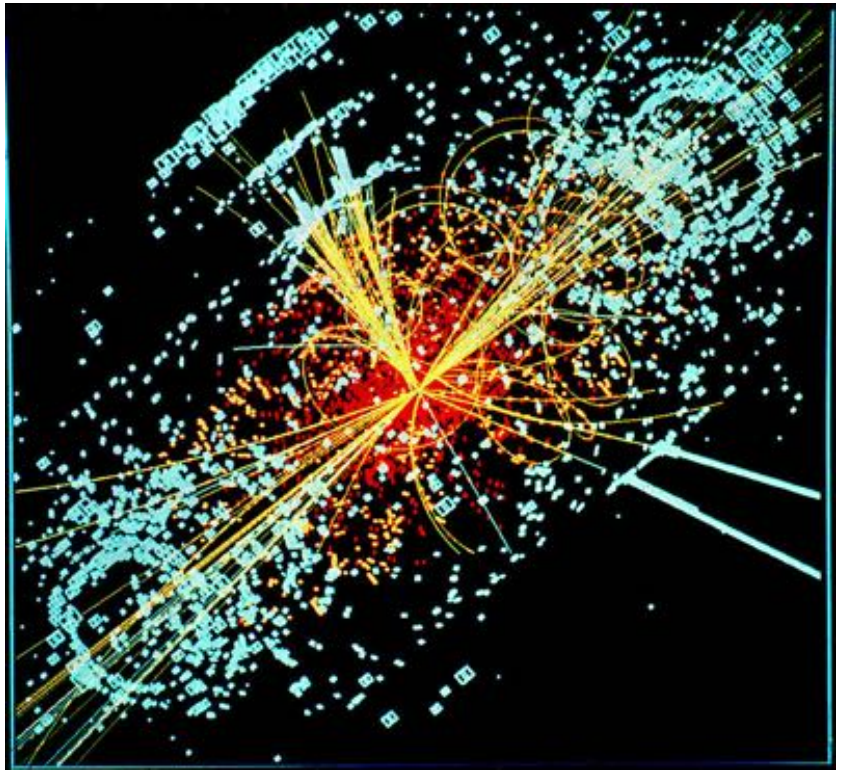
DAS GROSSE VEREINHEITLICHTE MODELL

- Umfasst die **vollständige Domäne** eines Unternehmens

➔Viel zu kompliziert

➔Hoher Koordinationsaufwand in Teams

➔Führt oft zu einem **big ball of mud**



DIE WIRKLICHE WELT

- Die vollständige Domäne ist **zu groß** für ein einzelnes Modell

➔ Bewusste Aufspaltung in Teilmodelle

- Separate Modelle
  - können **unabhängig entwickelt** werden
  - Müssen den an sie gestellten **Anforderungen genügen**
  - sollten **klar abgegrenzt** sein

➔ Jedes Modell sollte **klein genug** sein, so dass man es **einem Team** zuweisen kann

KONTEXTGRENZEN (*BOUNDED CONTEXT*)

- Jedes Modell hat einen Kontext
- Kontext = Grundsätzliche Voraussetzungen, damit Begriffe eine bestimmte Bedeutung erhalten
- Wird ein Modell aufgespalten, so ist für jedes Teilmodell eine Kontextdefinition erforderlich

- **Setze explizite Grenzen** in der…
  - Organisation von Teams
  - Benutzung von Teilen der Applikation
  - Codebasis
  - Entwicklung von Datenbank-Schemas

LIVING IN A BOX

- Erhalte die Konsistenz innerhalb der Grenzen
- Keine Ablenkung durch äußere Angelegenheiten

- Freie Gestaltung eines Teilmodells durch das zugehörige Team
  - Kenne die Restriktionen
  - Bleibe innerhalb der Modellgrenzen

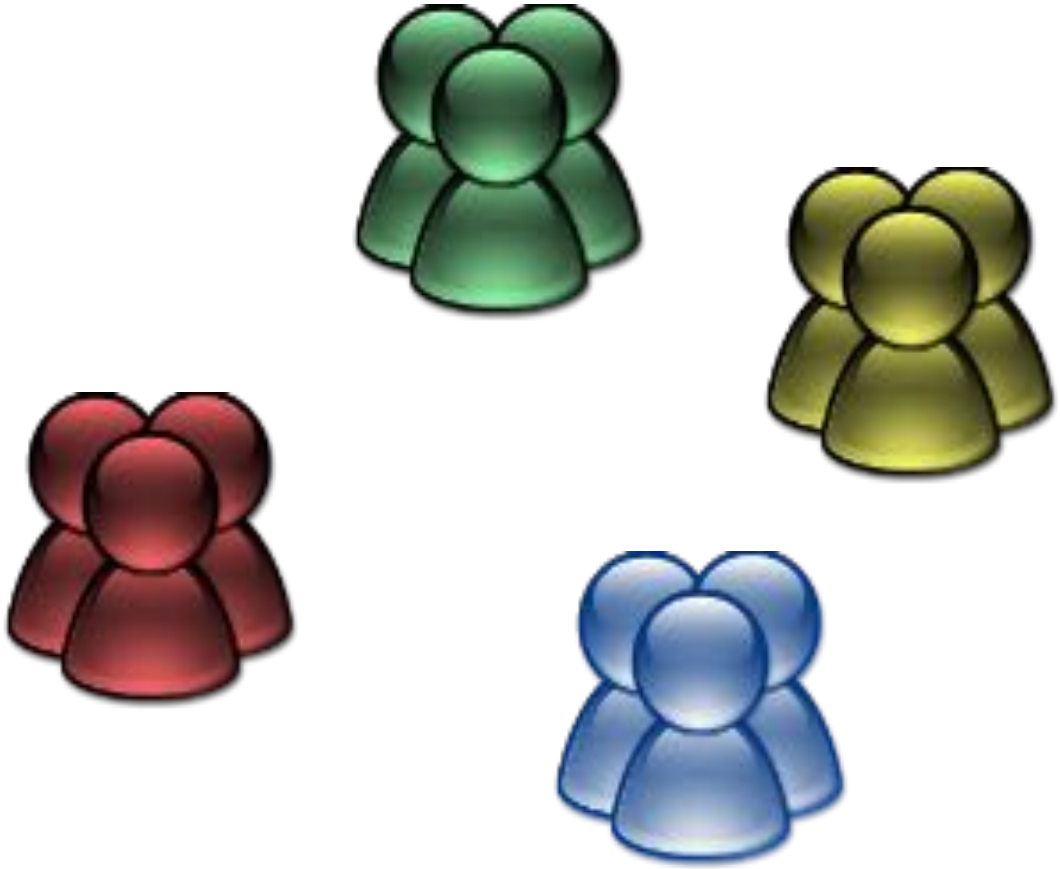
- Häufig werden Wertobjekte für die Kontext-Interkommunikation verwendet

JEDER BOUNDED CONTEXT HAT SEIN EIGENES MODELL

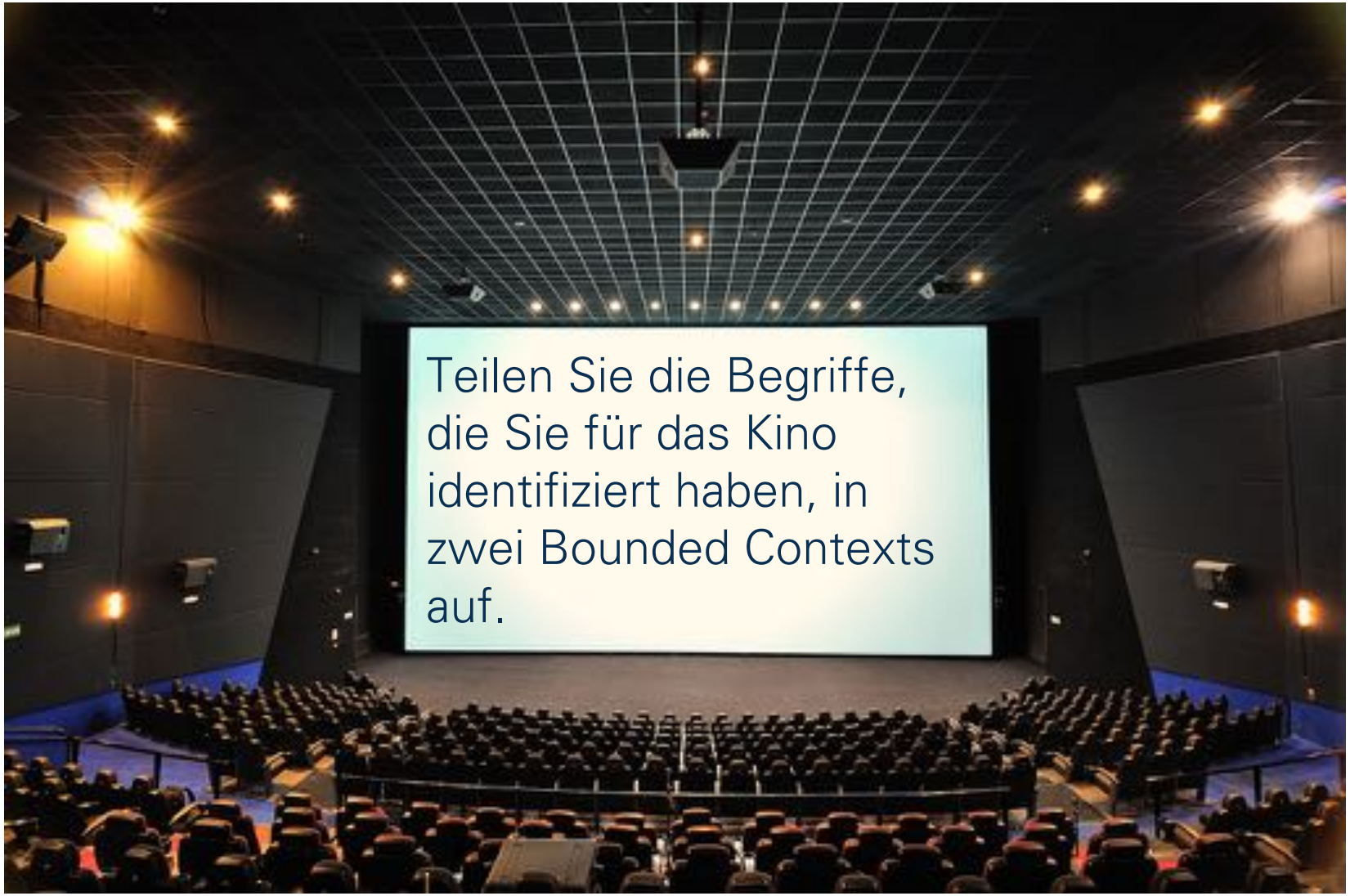
JEDER BOUNDED CONTEXT KANN EIN EIGENES TEAM HABEN

- Entwicklung findet parallel statt
- Jedes Team korrespondiert mit einem Teil des Modells

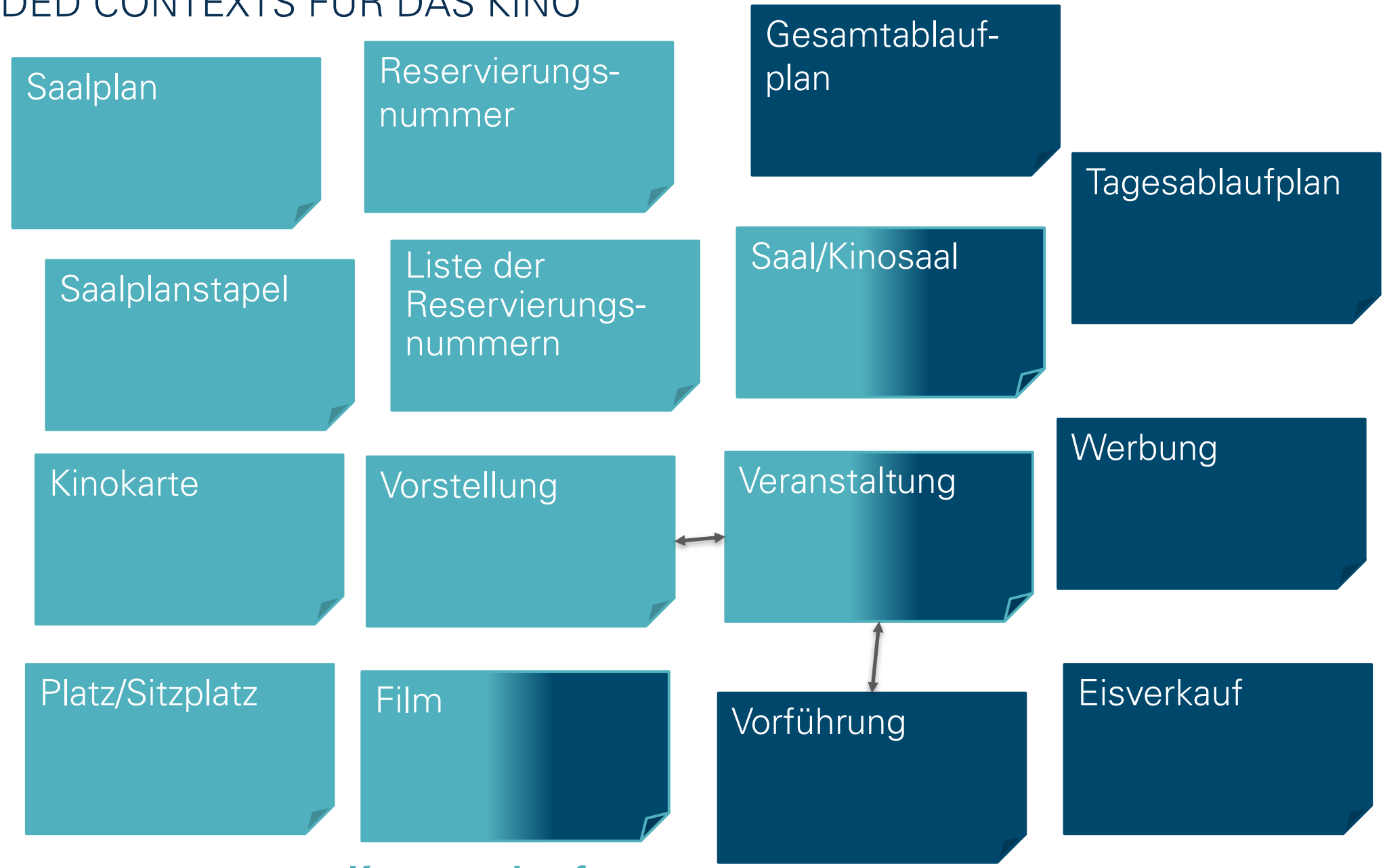
- → Microservices



BOUNDED CONTEXTE SCHNEIDEN



BOUNDED CONTEXTS FÜR DAS KINO



STRATEGISCHES DESIGN – WEITERE BEGRIFFE

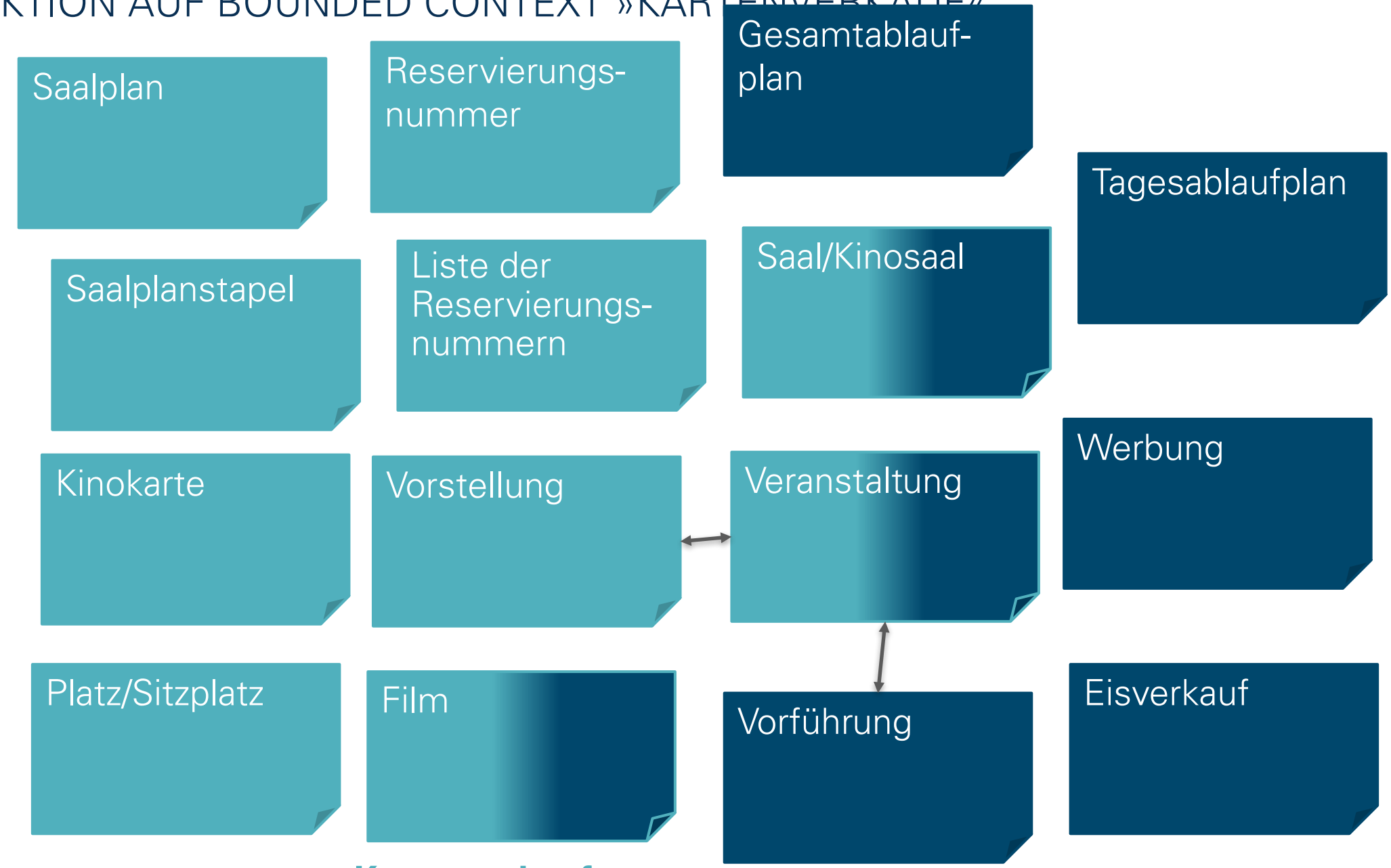
- Subdomänen
  - Kern (Core Domain)
  - Unterstützende (Supporting Domain)
  - Allgemeine (Generic Domain)

- Context Mapping
  - Shared Kernel
  - Customer/Supplier
  - Open-Host-Service
  - Published Language
  - Separate Ways
  - Anticorruption Layer
  - Conformist

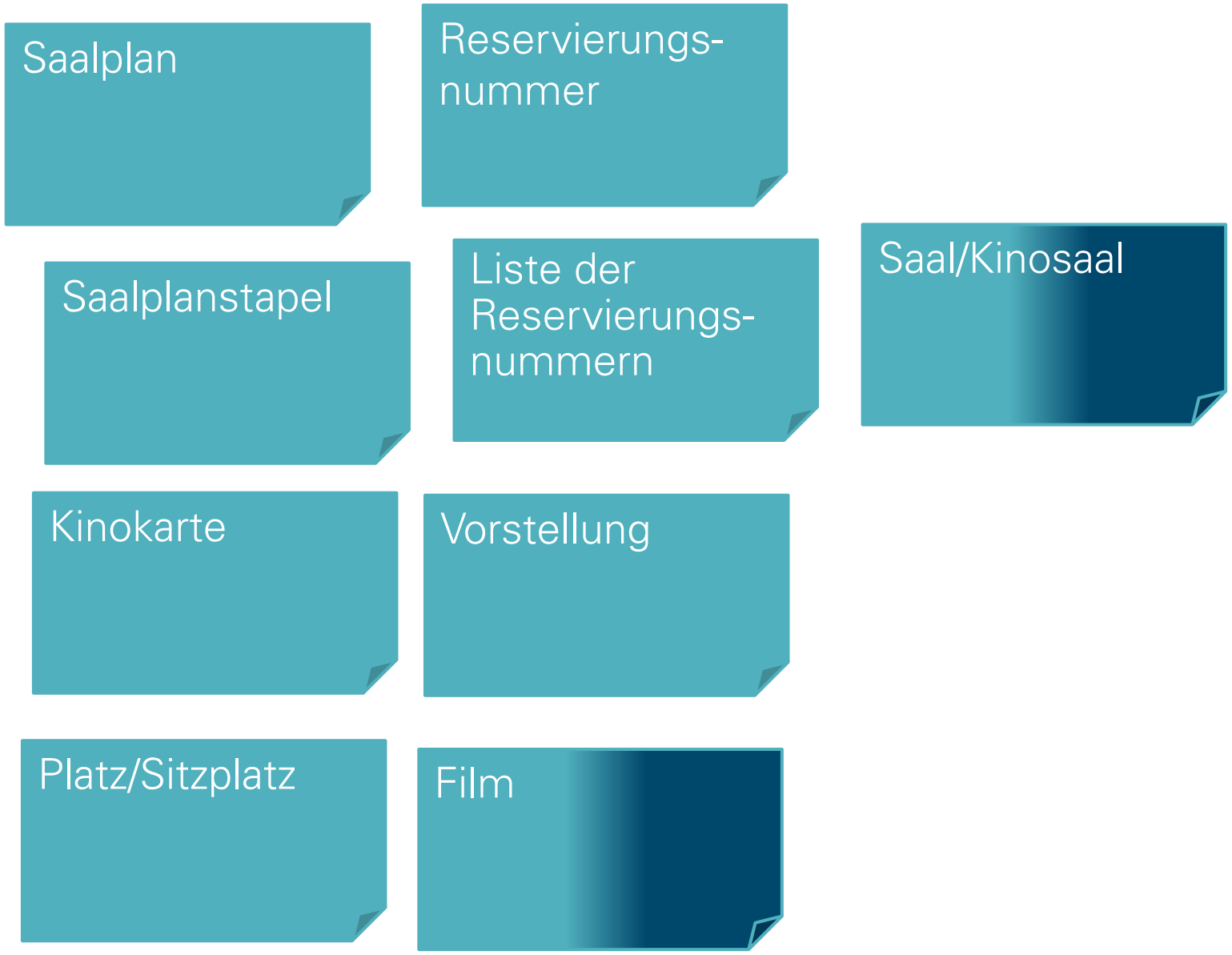
DER PREIS FÜR DOMÄNENMODELLE

- Zusätzliche Arbeit
  - Definition der Modellgrenzen
  - Definition der Beziehungen zwischen verschiedenen Modellen (*Context Mapping*)
- Abbildungen zwischen verschiedenen Modellen
- Objekte können nicht zwischen den BC transportiert werden

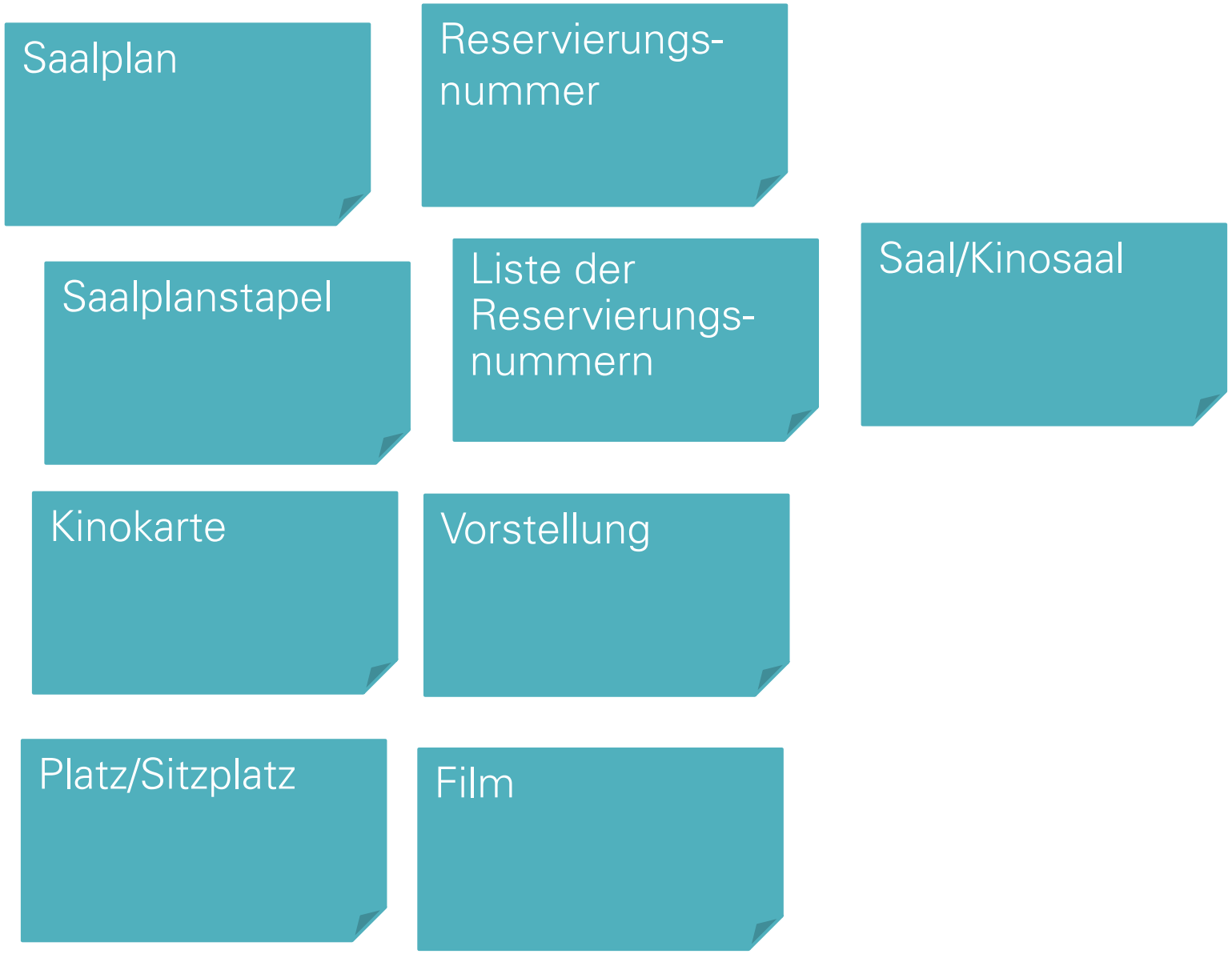
REDUKTION AUF BOUNDED CONTEXT »KARTENVERKAUF«



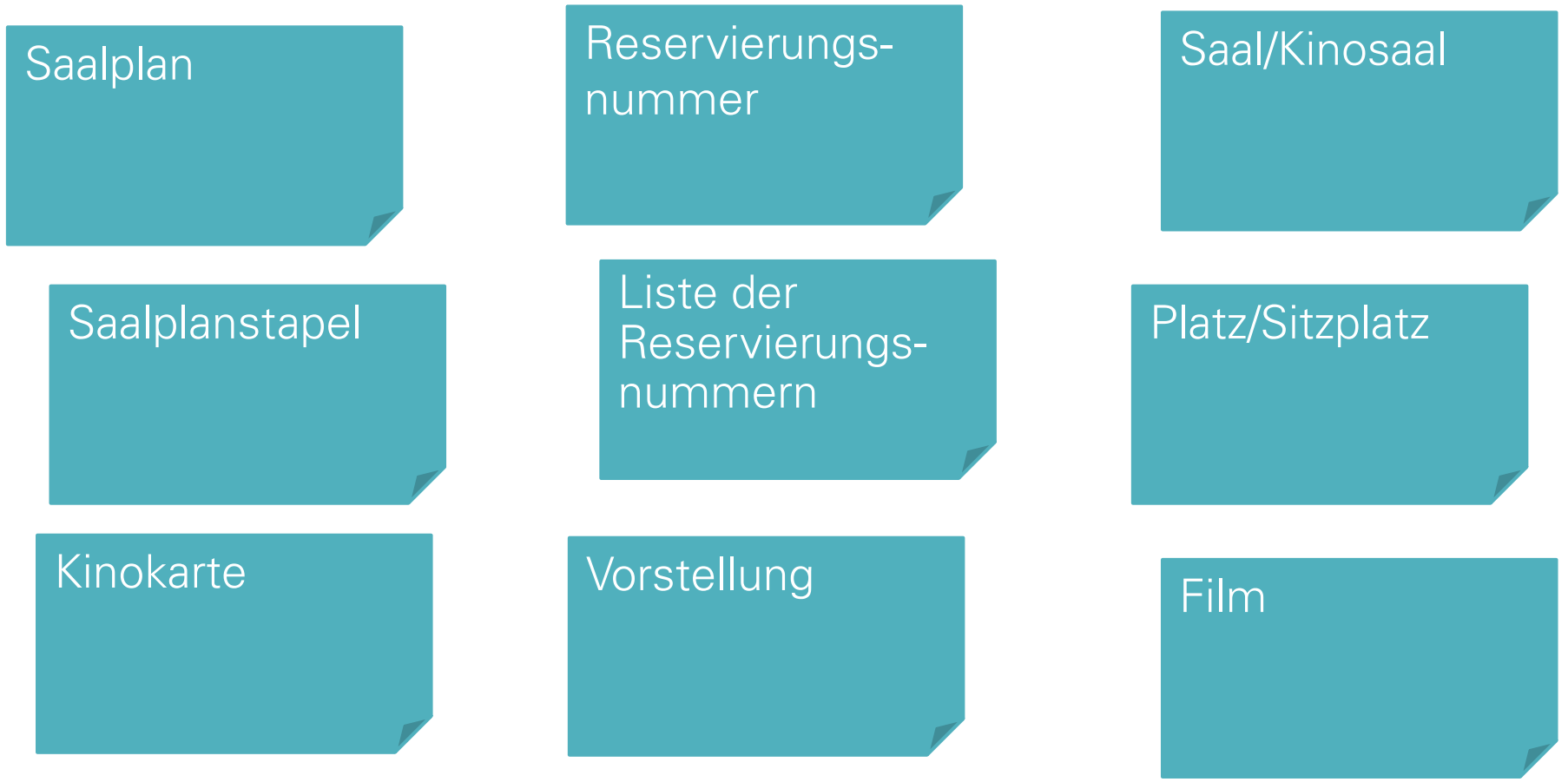
REDUKTION AUF BOUNDED CONTEXT »KARTENVERKAUF«



REDUKTION AUF BOUNDED CONTEXT »KARTENVERKAUF«



REDUKTION AUF BOUNDED CONTEXT »KARTENVERKAUF«



WPS WORKPLACE SOLUTIONS

FACHLICHE HANDLUNGEN

Domain-Driven Design hands-on

WIE LERNEN WIR DIE FACHLICHKEIT?

- Durch die **Fachexperten** (***domain experts***)
- Sie wissen
  - Worum es bei ihrer Arbeit geht
  - Wo die Software sie unterstützen kann

➔Wer die Fachlichkeit nicht versteht, kann ihr nicht helfen

➔Softwareentwickler und Fachexperten bilden zusammen das Team

Foto: Brandon Raile/Wikipedia

WIE KOMMEN WIR AN DAS WISSEN DER FACHEXPERTEN?

Techniken:

- Interviews
- Szenarios
- Event Storming
- Domain Storytelling

Das wollen wir extrahieren!

Foto: Brandon Raile/Wikipedia

WORKSHOPS MIT ANWENDERN UND PRODUCT OWNERS

- Teilnehmer aus verschiedenen Bereichen (Business, IT, Management, …)
- ein Moderator für den Workshop
  → direktes Feedback von allen Beteiligten

EVENT STORMING

- Eine Methode um Geschäftsprozesse zu modellieren

- Modelliert werden:
  - Domain Events
  - Commands
  - Akteure
  - Aggregates

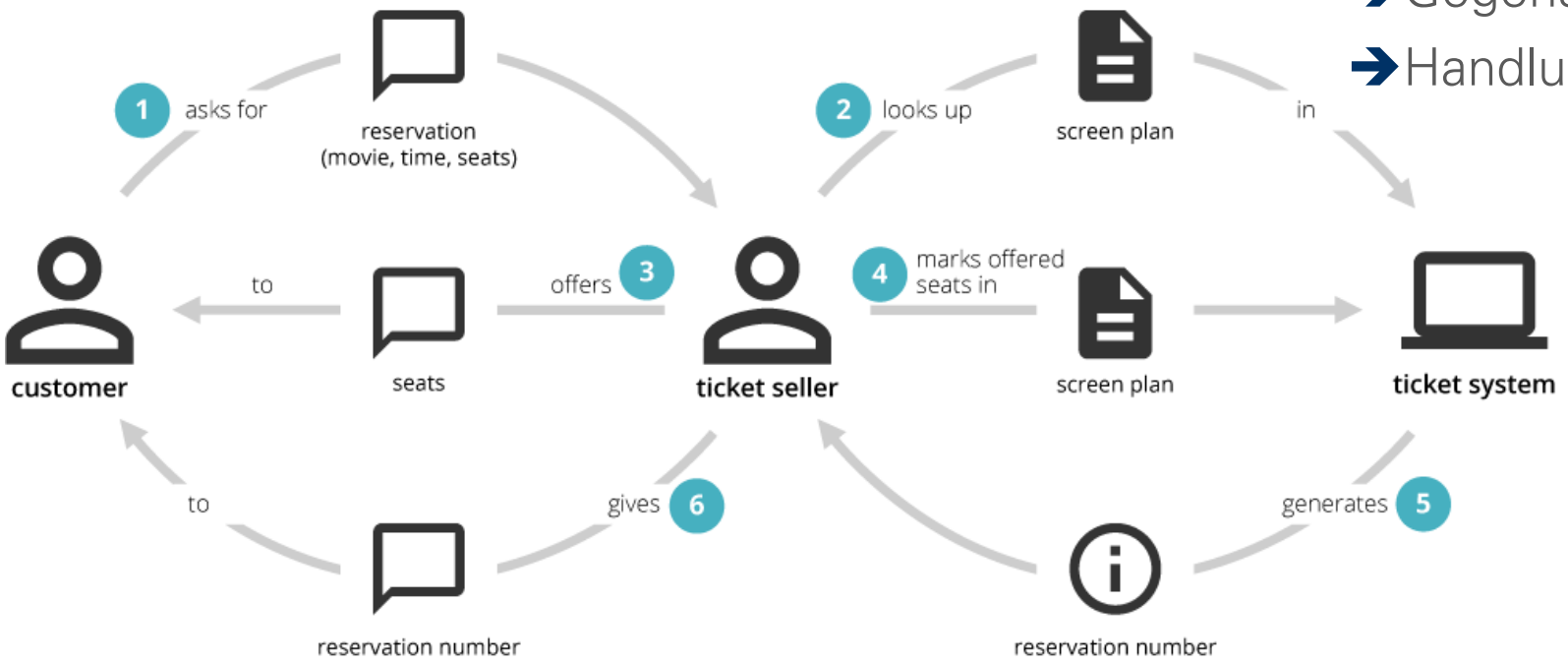
DOMAIN STORYTELLING

Modelliert werden:

➔Akteure

➔Gegenstände

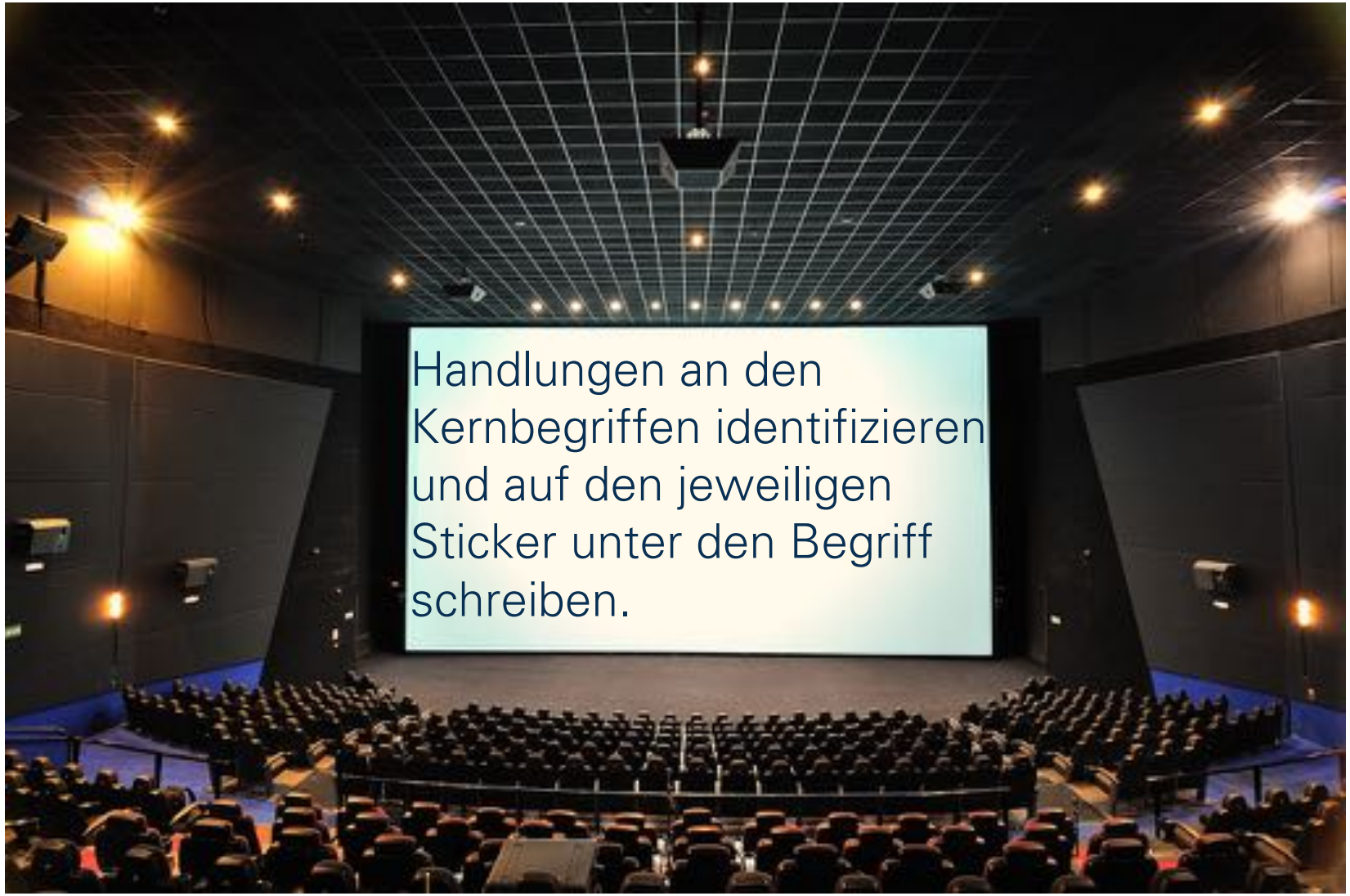
➔Handlungen



http://www.domainstorytelling.org

Domain Storytelling

FACHLICHE HANDLUNGEN IM IST IDENTIFIZIEREN



FACHLICHE HANDLUNGEN IM KARTENVERKAUF

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Reservierungs-nummer

Saal/Kinosaal

Saalplanstapel
- Saalplan zu Vorst. Holen
- Saalplan zurücklegen

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Platz/Sitzplatz

Kinokarte
- Mit Platz beschriften

Vorstellung

Film

EINIGE DATEN ERGÄNZEN

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Reservierungs-nummer

Saal/Kinosaal
- Kapazität
- Anzahl Reihen

Saalplanstapel
- Saalplan zu Vorst. Holen
- Saalplan zurücklegen

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Platz/Sitzplatz
- Reihe
- Nummer

Kinokarte
- Mit Platz beschriften
- Vorstellung

Vorstellung
- Datum
- Zeitraum
- Film

Film
- Titel
- Regisseur, Schauspieler
- Freigabe
- Spieldauer

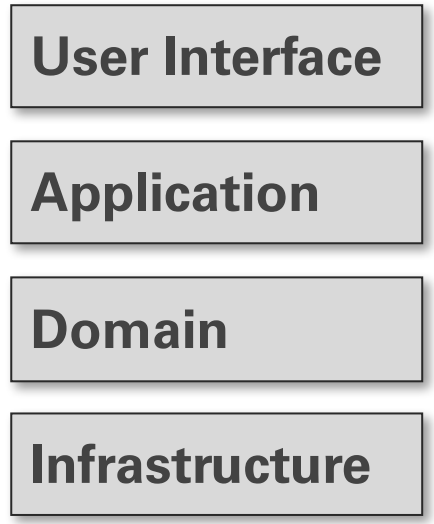
WPS WORKPLACE SOLUTIONS

TAKTISCHES DESIGN

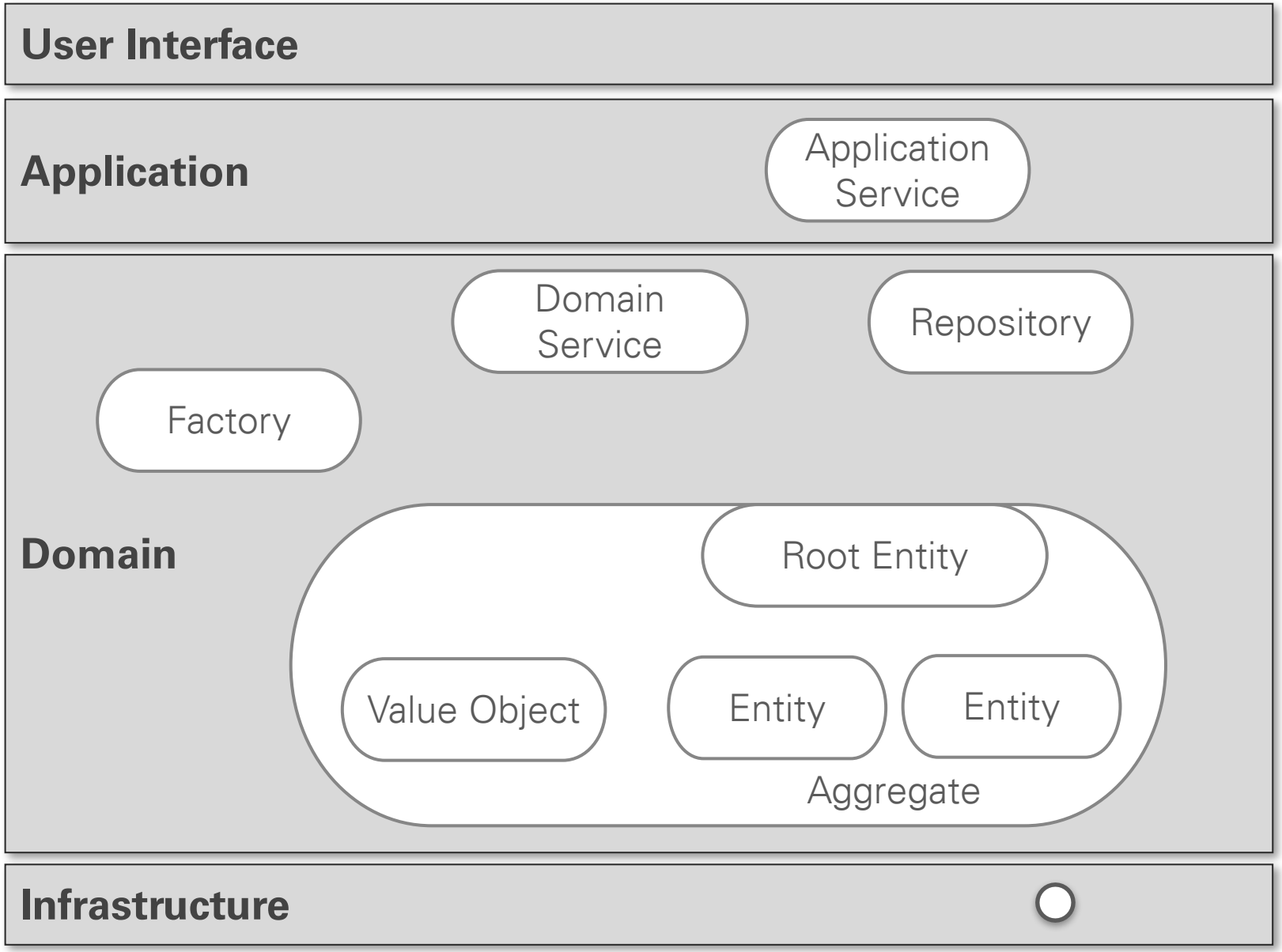
Domain-Driven Design konkret

DOMAIN-DRIVEN DESIGN – SCHICHTENARCHITEKTUR

- **User-Interface-Schicht**
  - Nimmt Eingaben und Benutzerkommandos entgegen und stellt Informationen dar.
- **Application-Schicht**
  - Beschreibt und koordiniert Geschäftsprozesse.
  - Kann in vielen Anwendungen entfallen, weil die Domain-Schicht ihre Aufgabe wahrnimmt.
- **Domain-Schicht**
  - Repräsentiert die Fachdomäne.
- **Infrastruktur-Schicht**
  - Bietet technische Dienste, wie beispielsweise Persistenz oder die Kommunikation mit anderen Systemen.



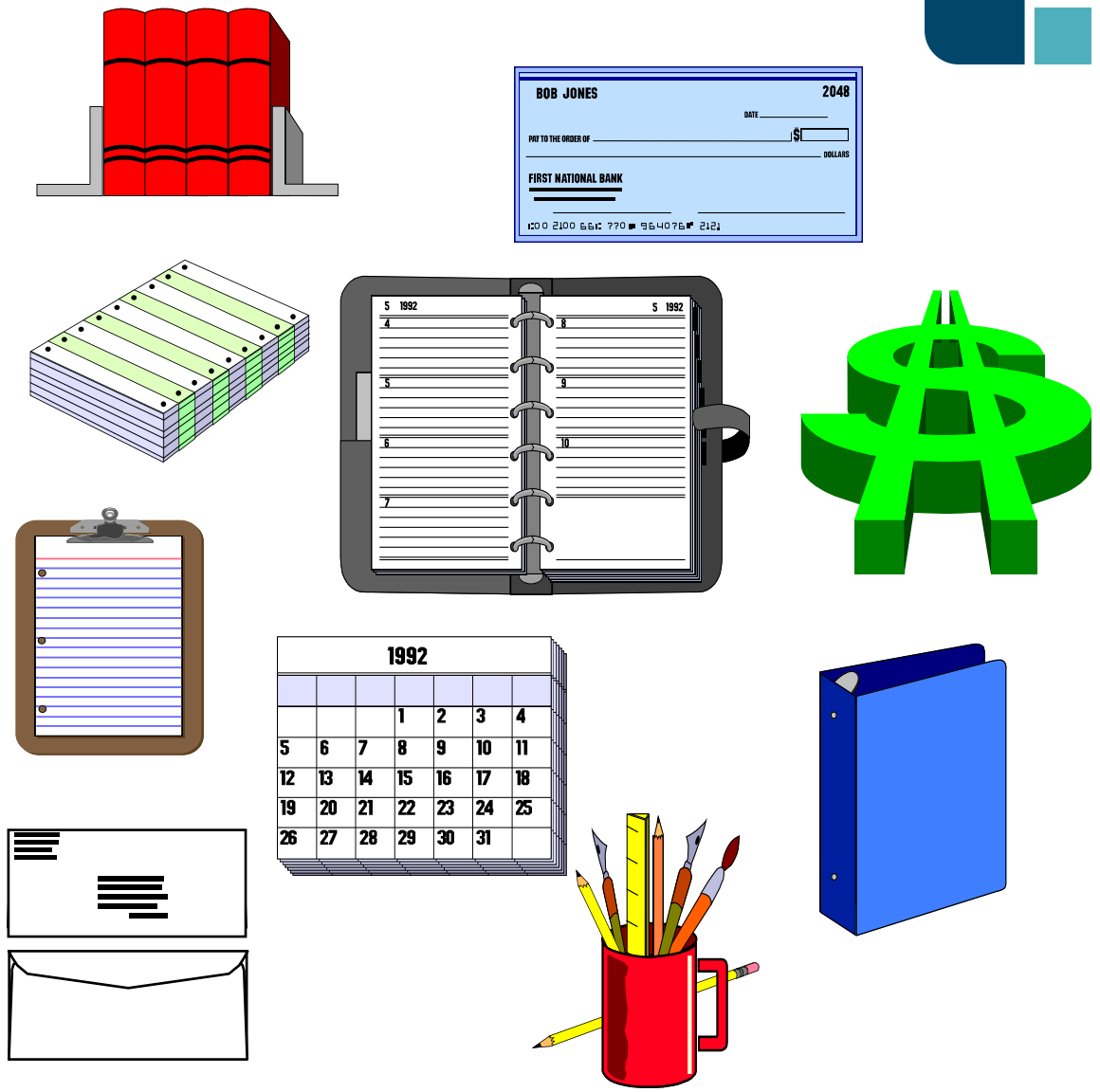
ELEMENTE DER DOMAIN-SCHICHT



ENTITIES

- Sind die **Kernobjekte** einer Fachdomäne.
- Besitzen eine zustandsunabhängige, unveränderliche **Identität**.
- Haben einen klar definierten **Lebenszyklus**.
- Besitzen einen (meist veränderlichen) **Zustand**.
- Beschreiben ihren Zustand mithilfe von Value Objects.
- Sind praktisch immer persistent.

- Auch: *Business Objects / Domain Objects*



→ NICHT ZU VERWECHSELN mit dem Begriff "ENTITY" aus dem Entity-Relationship-Modell!

ENTITIES – BEISPIELE

Baumaßnahme

Urlaubsantrag

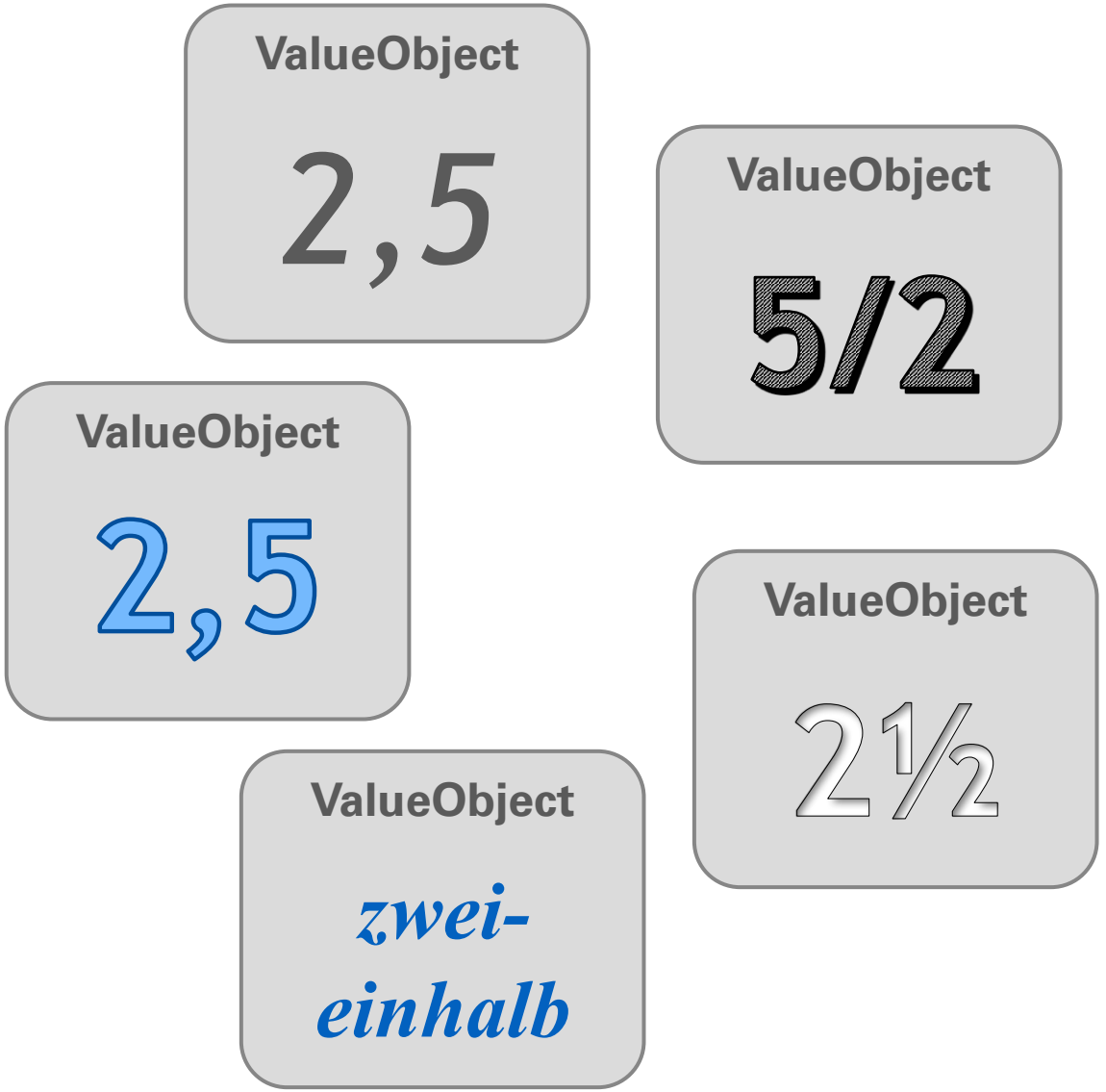
Versicherungspolice

Kaufvertrag

Fahrauftrag

VALUE OBJECTS

- Symbolisieren Werte eines bestimmten Typs der Fachdomäne.
- Symbolisieren bei Gleichheit denselben Wert.
- Sind unveränderlich.
- Können ggf. (aus anderen Value Objects) berechnet werden.
- Können aus anderen Value Objects bestehen, aber nie aus Entitäten!



VALUE OBJECTS – BEISPIELE

Postleitzahl

IBAN

IATA-Code

GPS-Koordinate

Containernummer

ENTITES, VALUE OBJECTS

ENTITIES UND VALUE OBJECTS IM KARTENVERKAUF

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Entity

Reservierungs-nummer

VO

Saal/Kinosaal
- Kapazität
- Anzahl Reihen

Entity

Saalplanstapel
- Saalplan zu Vorst. Holen
- Saalplan zurücklegen

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Entity

Platz/Sitzplatz
- Reihe
- Nummer

Entity

Kinokarte
- Mit Platz beschriften
- Vorstellung

Vorstellung
- Datum
- Zeitraum
- Film

Entity

Film
- Titel
- Regisseur, Schauspieler
- Freigabe
- Spieldauer

Entity

ENTITIES UND VALUE OBJECTS IM KARTENVERKAUF

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Entity

Reservierungs-nummer

VO

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Entity

Saal/Kinosaal
- Kapazität
- Anzahl Reihen

Entity

Platz/Sitzplatz
- Reihe
- Nummer

Entity

Vorstellung
- Datum
- Zeitraum
- Film

Entity

Film
- Titel
- Regisseur, Schauspieler
- Freigabe
- Spieldauer

Entity

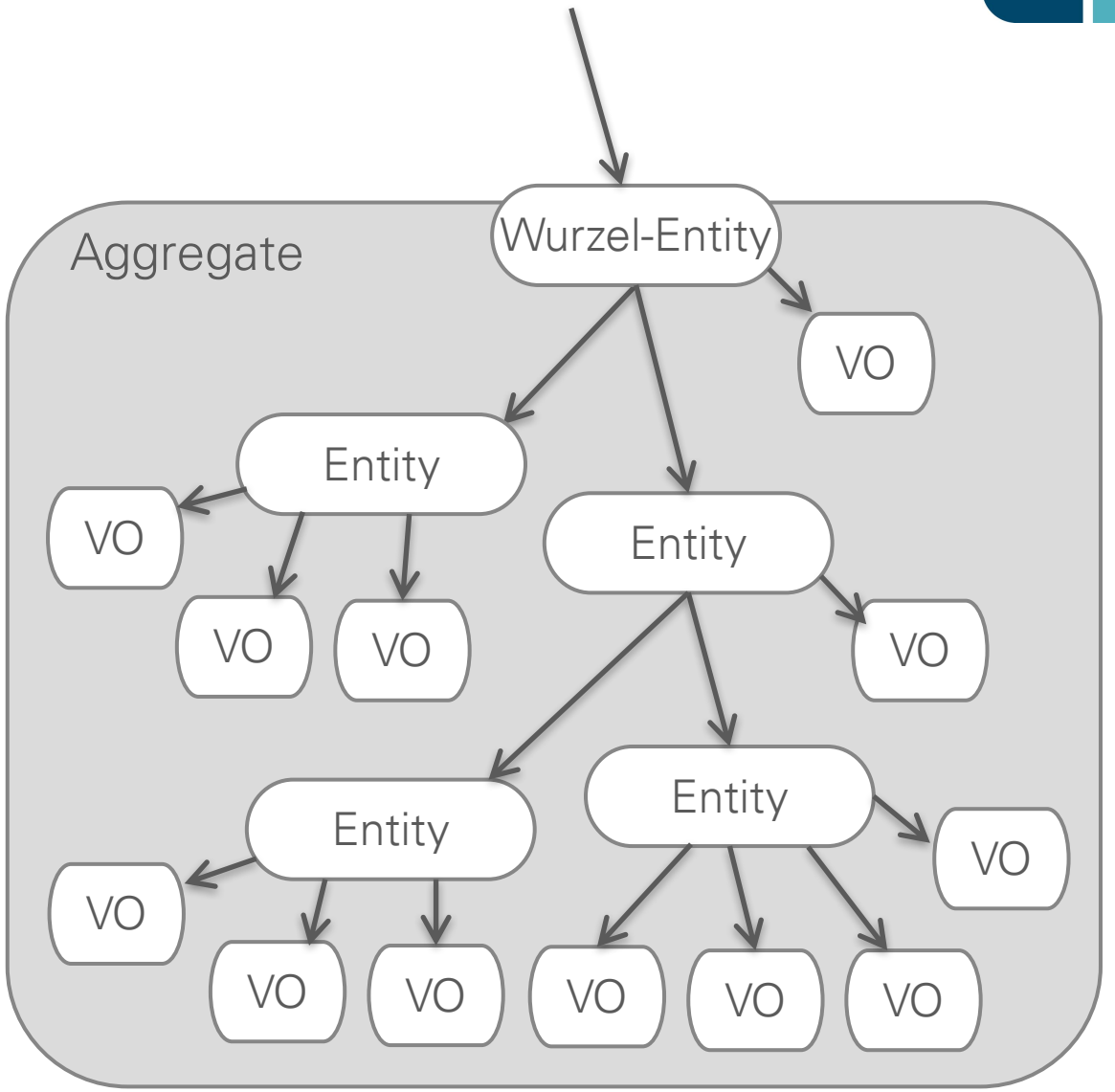
WPS WORKPLACE SOLUTIONS

AGGREGATES

AGGREGATE

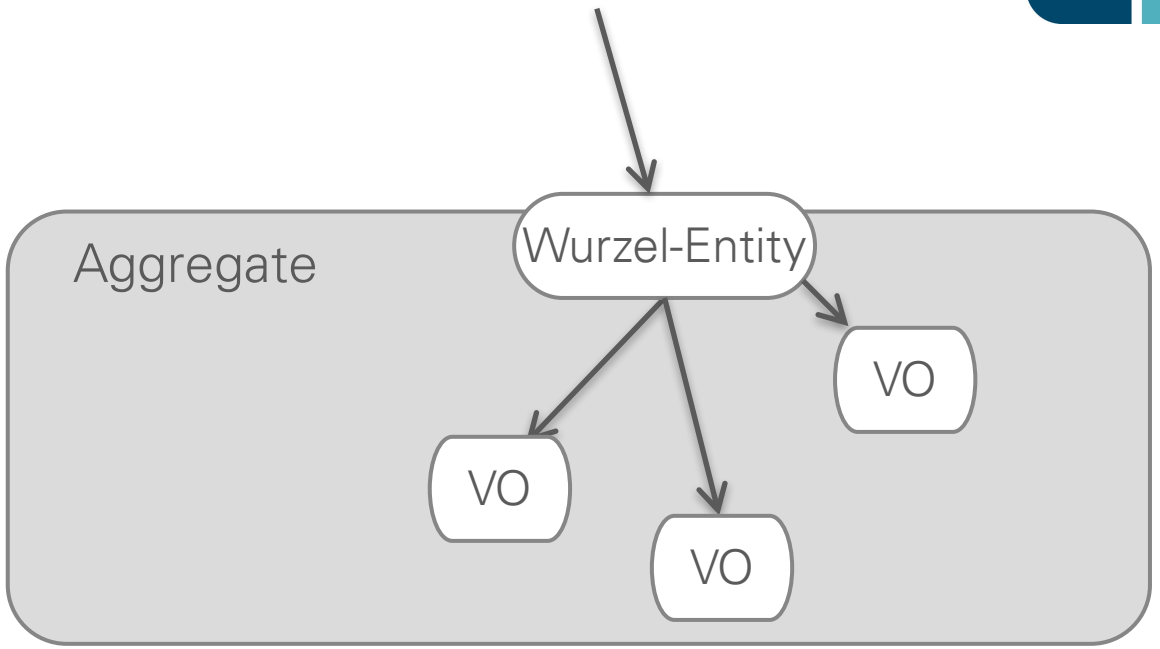
- Bildet eine Klammer um eine Struktur von Entities.
- Besitzt grundsätzlich eine ausgezeichnete Entität als Einstiegspunkt (Wurzel).
- Wird fachlich als eine zusammenhängende Einheit betrachtet.
- Schützt Konsistenz und Integrität ihrer innerer Entities (muss sie dazu aber nicht notwendigerweise verbergen!), sowie die eigene Konsistenz und Integrität!

- Technik:
  - Wird üblicherweise in der Datenbank gespeichert.
  - Als Ganzes!



AGGREGATE

- Besteht oft auch nur aus einer Entity
- Oft gleicher Name wie die Wurzel-Entity



ENTITIES UND VALUE OBJECTS IM KARTENVERKAUF

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Entity

Reservierungs-nummer

VO

Saal/Kinosaal
- Kapazität
- Anzahl Reihen

Entity

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Entity

Platz/Sitzplatz
- Reihe
- Nummer

Entity

Vorstellung
- Datum
- Zeitraum
- Film

Entity

Film
- Titel
- Regisseur, Schauspieler
- Freigabe
- Spieldauer

Entity

AGGREGATES SCHNEIDEN

ENTITIES UND VALUE OBJECTS IM KARTENVERKAUF

Aggregate

Saalplan
- Anzahl Plätze suchen
- Verkaufte Plätze mark.
- Reserv. Plätze mit RN markieren

Entity

Aggregate

Reservierungs-nummer **VO**

Liste der Reservierungsnummern
- Reservierungsnummer abholen
- Name+Vorst. Vermerken
- Vorst. Mit RN heraussuchen

Entity

Aggregate

Saal/Kinosaal
- Kapazität
- Anzahl Reihen

Wurzel-Entity

Platz/Sitzplatz
- Reihe
- Nummer

Entity

Aggregate

Vorstellung
- Datum
- Zeitraum
- Film

Entity

Film
- Titel
- Regisseur, Schauspieler
- Freigabe
- Spieldauer

Entity

Aggregate

WPS WORKPLACE SOLUTIONS

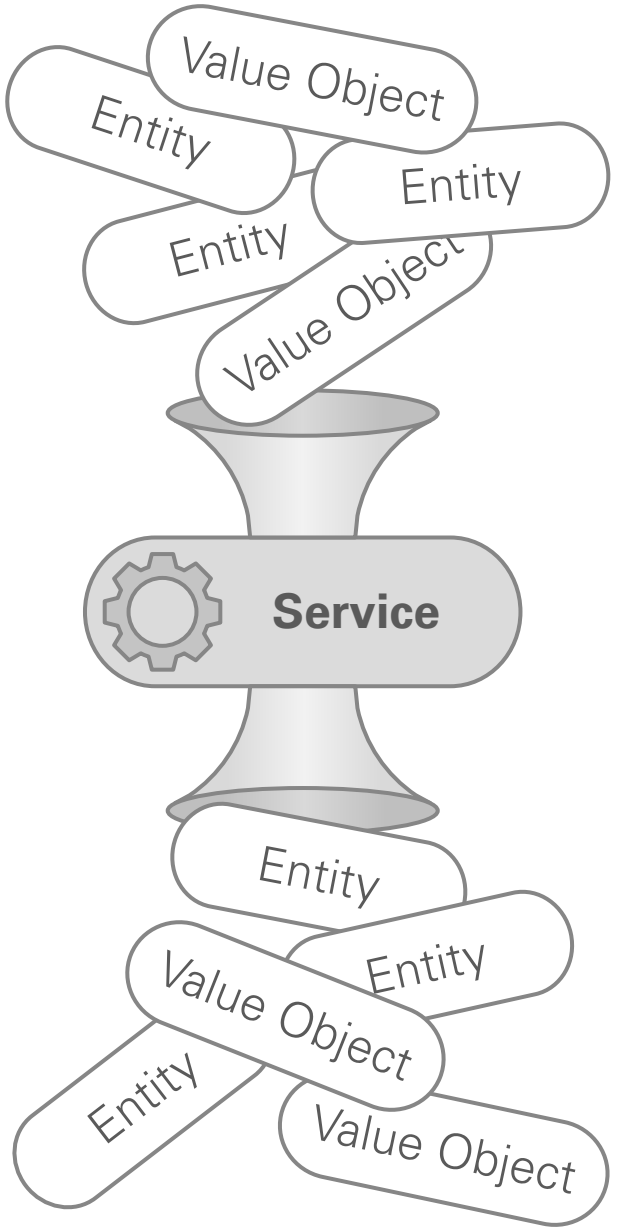
MEHR BAUSTEINE

SERVICES

- Stellen Abläufe oder Prozesse der Fachdomäne dar, die nicht von Entitäten ausgeführt werden können.
- Sind **zustandslos**!
- Parameter und Ergebnisse ihrer Operationen sind Entities und Value Objects.

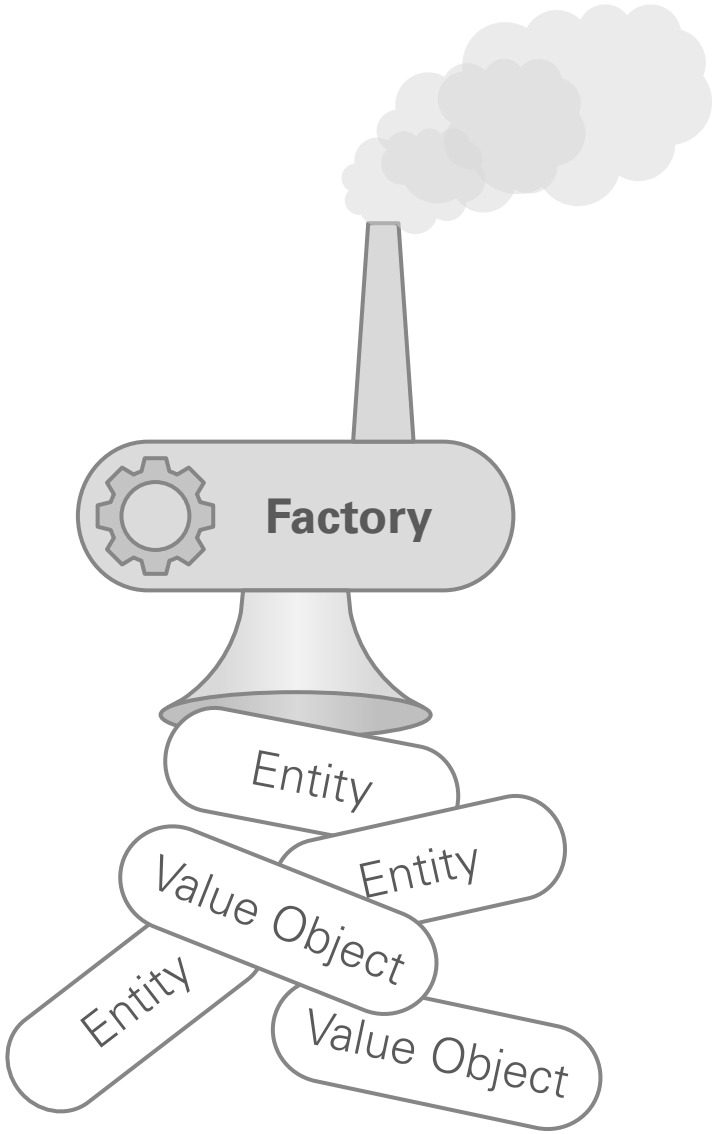
- Auch: Fachlicher Service

- *NICHT ZU VERWECHSELN mit den "technischen Services" der Application-Schicht!*



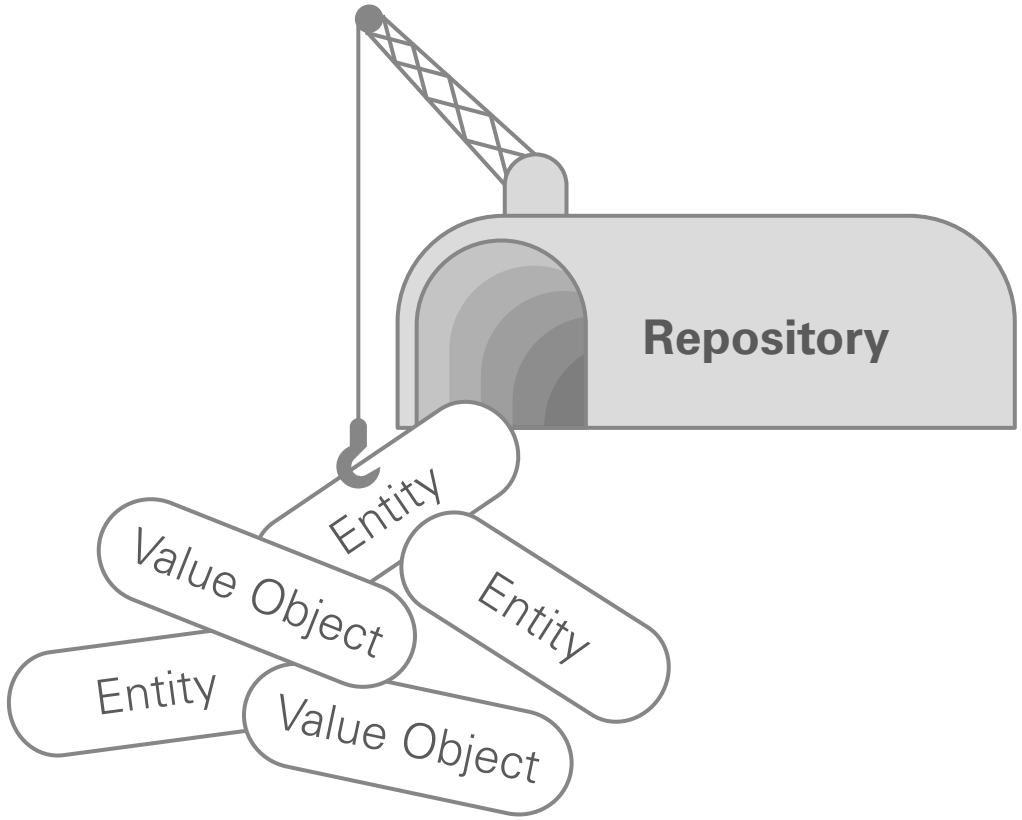
FACTORIES

- Erzeugen Aggregate, Entities (und ggf. auch Value Objects).
- Arbeiten ausschließlich innerhalb der Domain-Schicht.

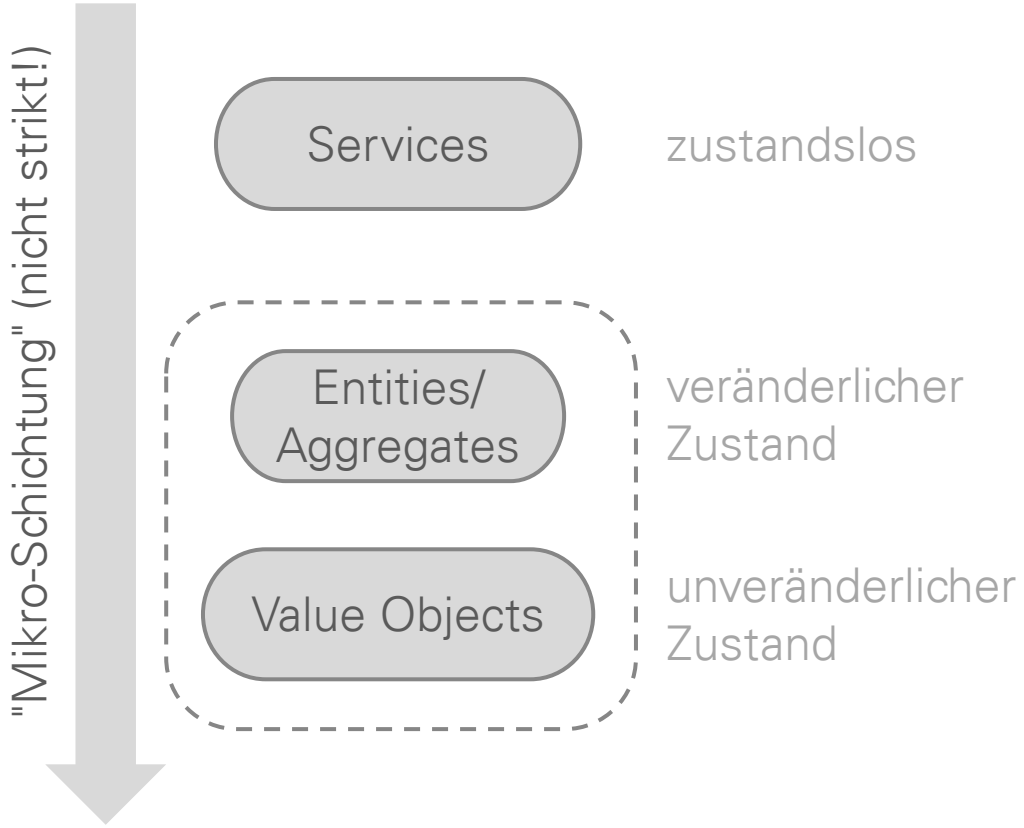


REPOSITORIES

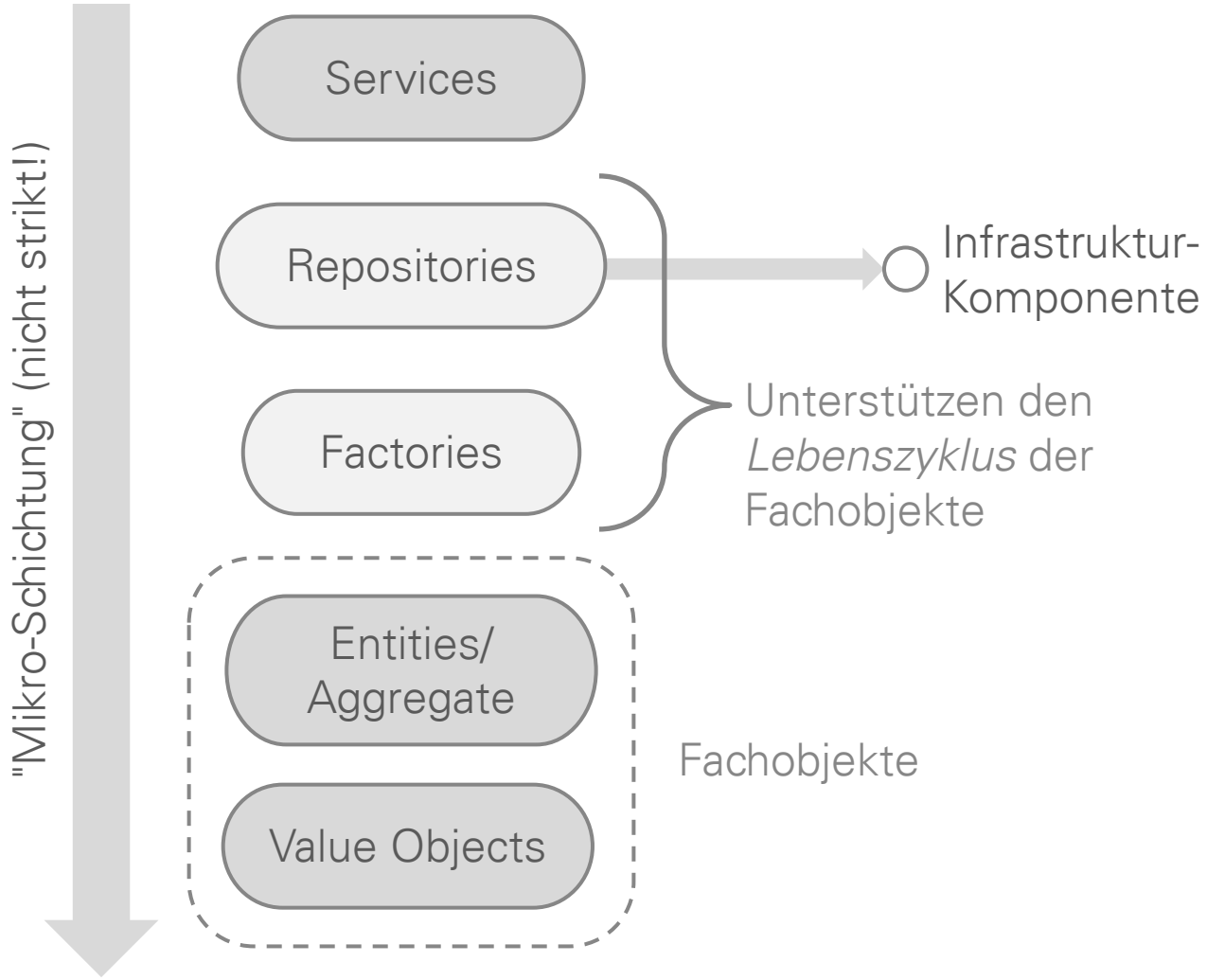
- Kapseln die technischen Details der technischen Infrastrukturschicht…
- … und bilden deren Daten auf fachliche Entities und Value Objects ab.
- Persistieren Entities z.B. in Datenbanken.



DIE »KERNELEMENTE«

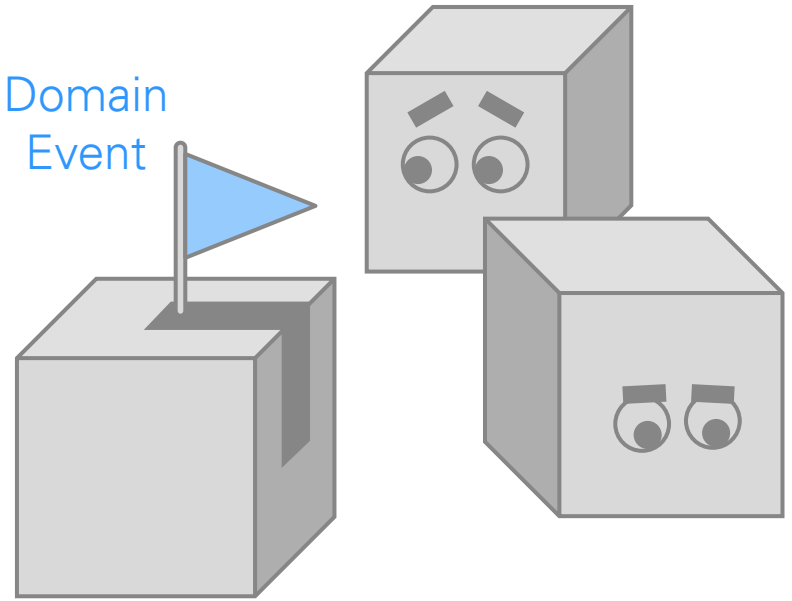


ELEMENTE DER DOMAIN-SCHICHT



DOMAIN EVENTS

- Signalisieren Ereignisse, die aus fachlicher Sicht relevant sind
- Signalisieren Ereignisse die geschehen sind (NICHT solche, die passieren sollen).
- Werden zur Kommunikation zwischen Bounded Contexts verwendet und…
  - … fördern so deren Entkoppelung,
  - … erleichtern asynchrone Prozesse.
- Bewirken Reaktionen der Subsysteme, die sich für diese fachlichen Ereignisse interessieren.



DOMAIN EVENTS – BEISPIELE

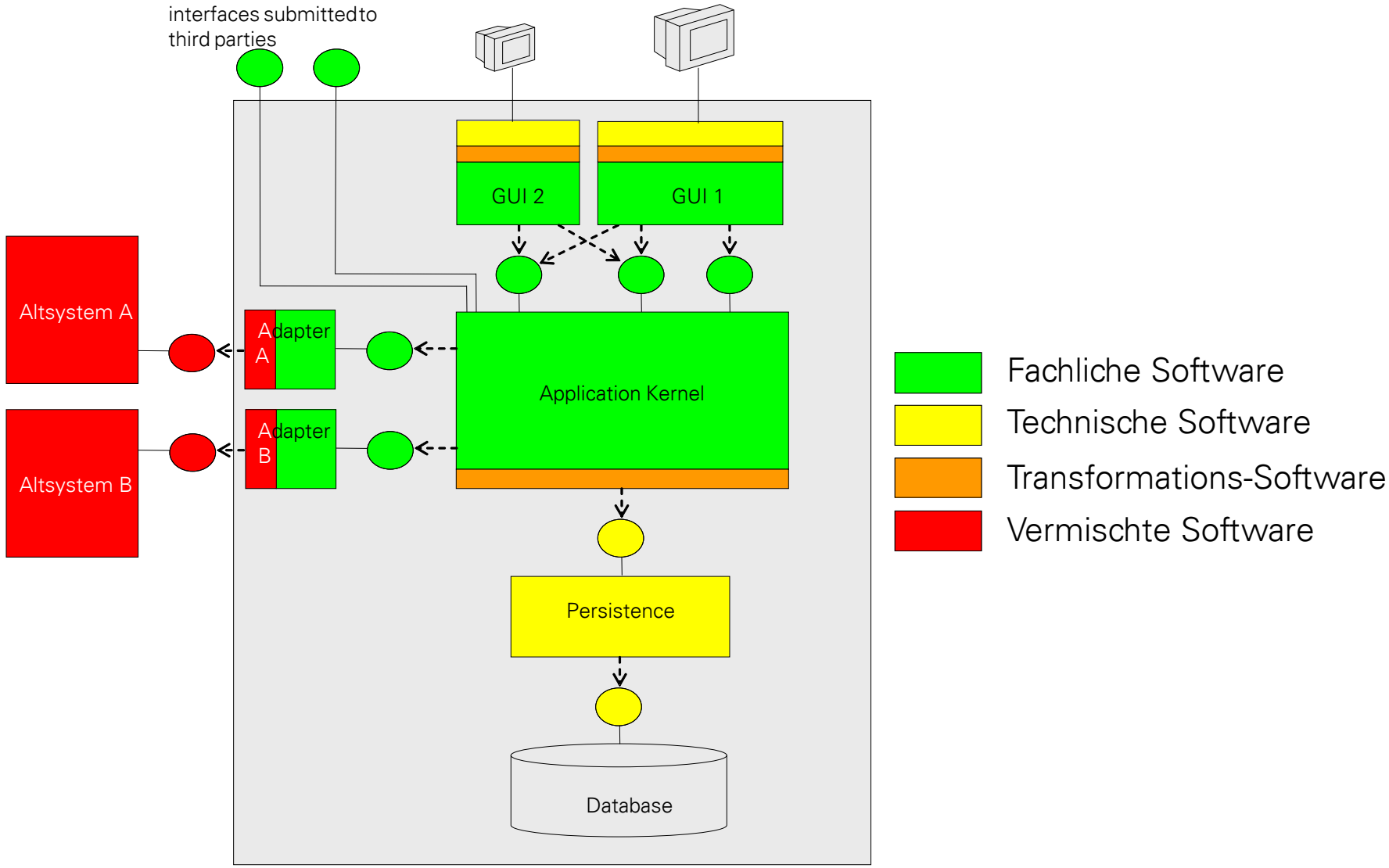
RechnungAusgestellt

ÜberweisungErfolgt

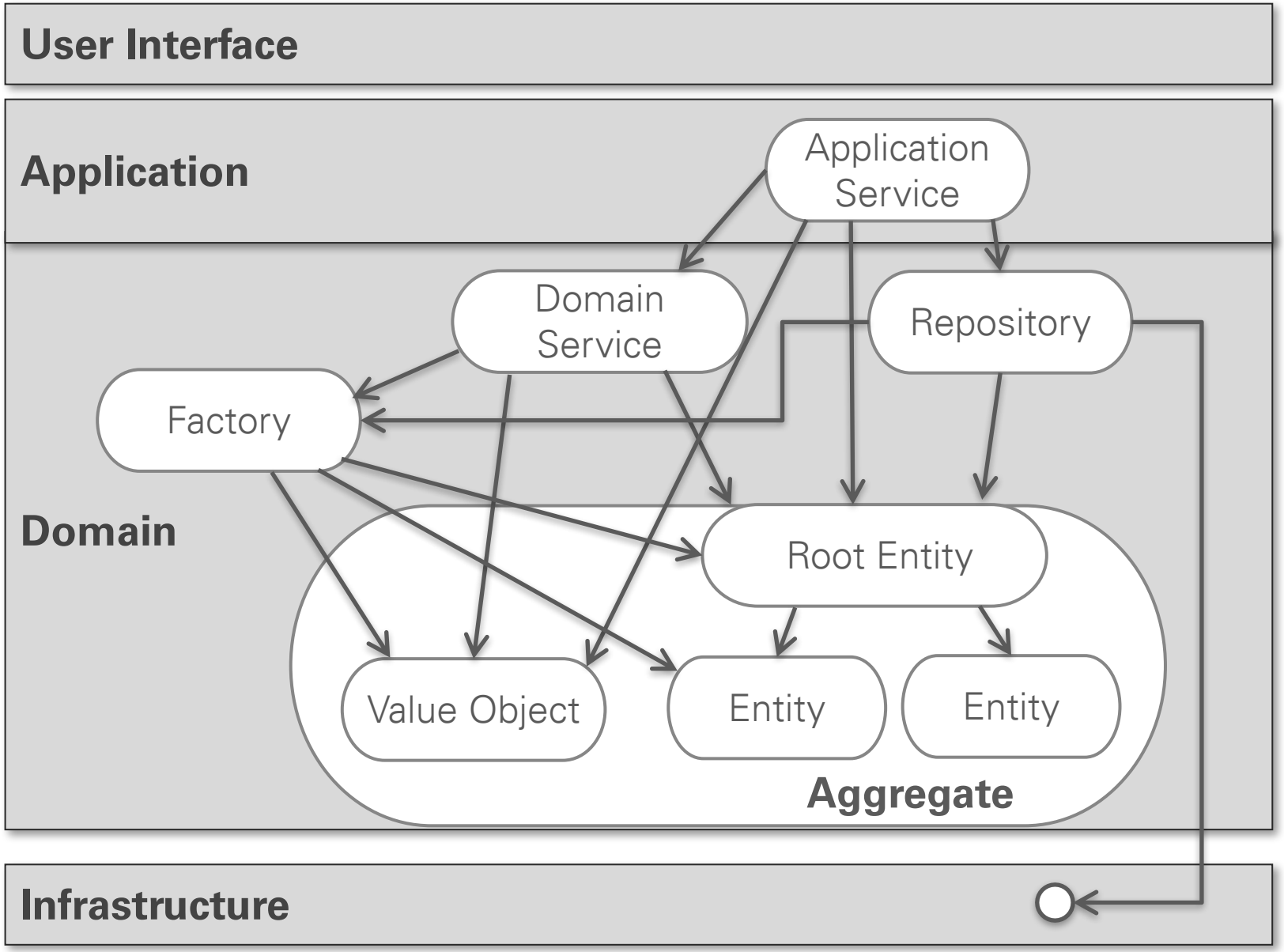
BuchVerkauft

KontoEröffnet

TRENNUNG VON FACHLICHER + TECHNISCHER SOFTWARE

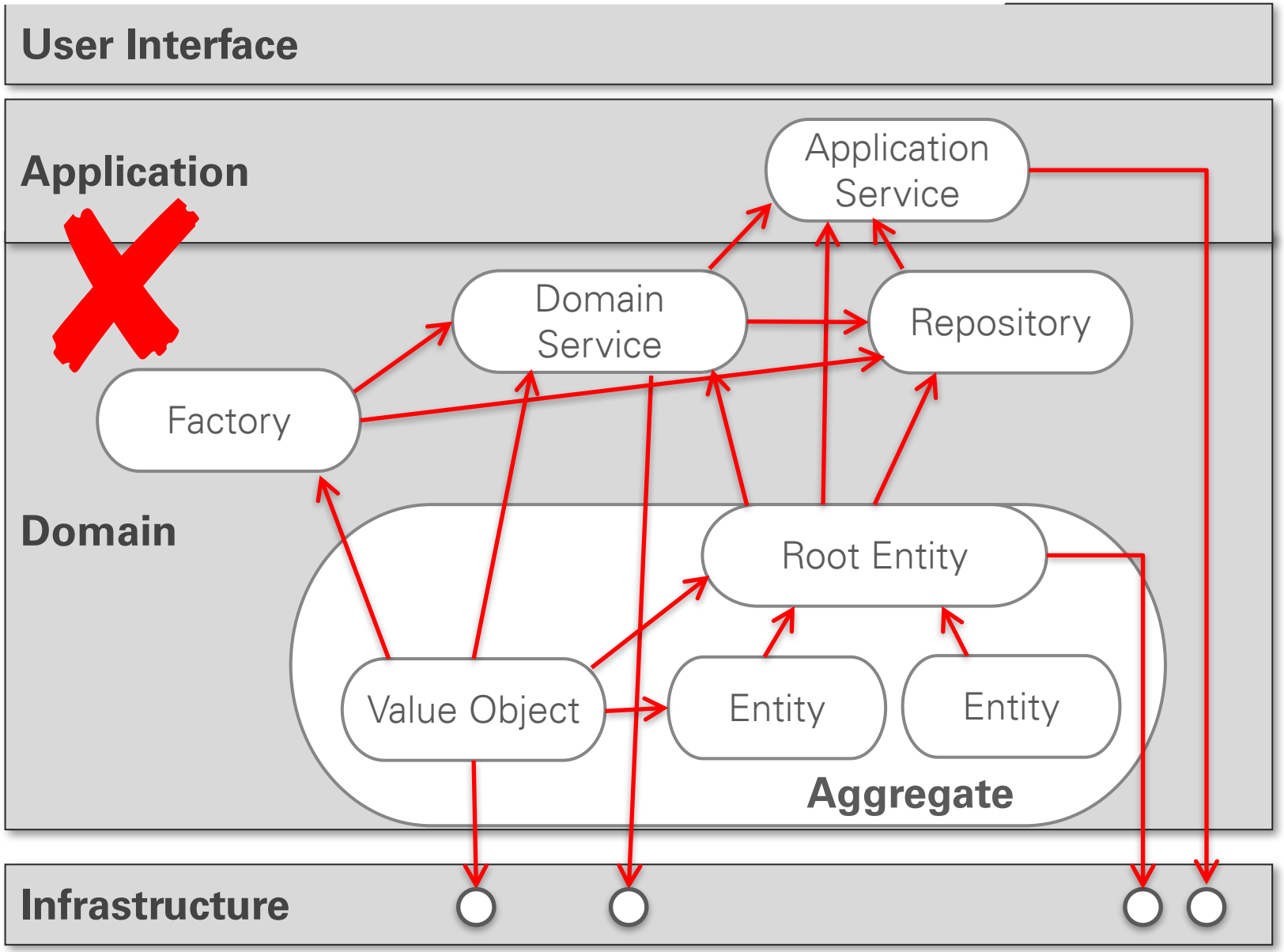


ELEMENTE DER DOMAIN-SCHICHT



ELEMENTE DER DOMAIN-SCHICHT

Nicht erlaubte Beziehungen (typische Beispiele)



SERVICES, REPOSITORIES, FACTORIES

WPS WORKPLACE SOLUTIONS

DIE BAUSTEINE IMPLEMENTIEREN

Domain-Driven Design konkret

EIN BANKKONTO – ERSTER ENTWURF

```
public class Account {
    private int _balance;

    public int getBalance() {
        return _balance;
    }

    public void setBalance(int balance) {
        _balance = balance;
    }
}
```

EIN BANKKONTO – ERSTER ENTWURF

```
public class Account {
    private int _balance;

    public int getBalance() {
        return _balance;
    }

    public void setBalance(int balance) {
        _balance = balance;
    }
}
```

Schlecht: Der Kontostand kann auf beliebigen Wert gesetzt werden

EIN BANKKONTO – ZWEITER ENTWURF

```
public class Account {
    private int _balance;

    public int getBalance() {
        return _balance;
    }

    public void deposit(int amount) {
        _balance += amount;
    }

    public void withdraw(int amount) {
        _balance -= amount;
    }
}
```

Besser:

Methoden mit fachlichem Verhalten und Namen

EIN BANKKONTO – DRITTER ENTWURF

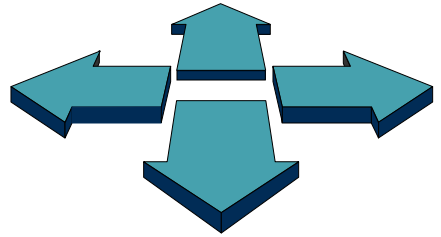
```
public class Account {
    private int _balance;

    public int getBalance() {
        return _balance;
    }

    public void deposit(int amount) {
        _balance += amount;
    }

    public void withdraw(int amount) {
        if (amount >= getBalance()) {
            throw new IllegalArgumentException("Amount too big");
        }
        _balance -= amount;
    }
}
```

Noch besser:

Vorbedingungen können geprüft werden

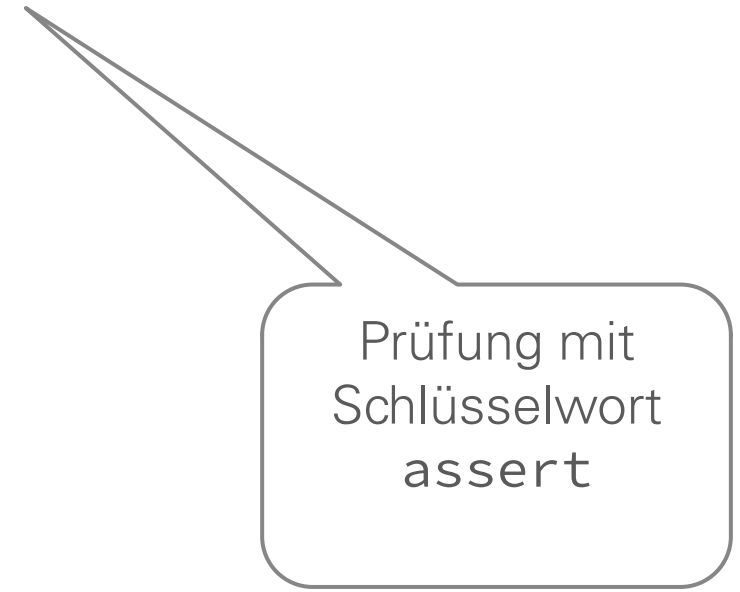
ÜBERSICHT

- Entities Implementieren
- Beispiel-Entity
- Vertragsmodell (Design by contract)
- Beispiel-Value-Object
- Identität von Entities



EIN BANKKONTO – VERTRAGSMODELL MIT ASSERT

```
public class Account {

    // ...

    public void withdraw(int amount) {
        assert amount >= getBalance();
        _balance -= amount;
    }
}
```



EIN BANKKONTO – VERTRAGSMODELL MIT VALID4J

```
import static org.valid4j.Assertive.*;

public class Account {

    // ...

    public void withdraw(int amount) {
        require(amount <= getBalance());
        _balance -= amount;
    }
}
```

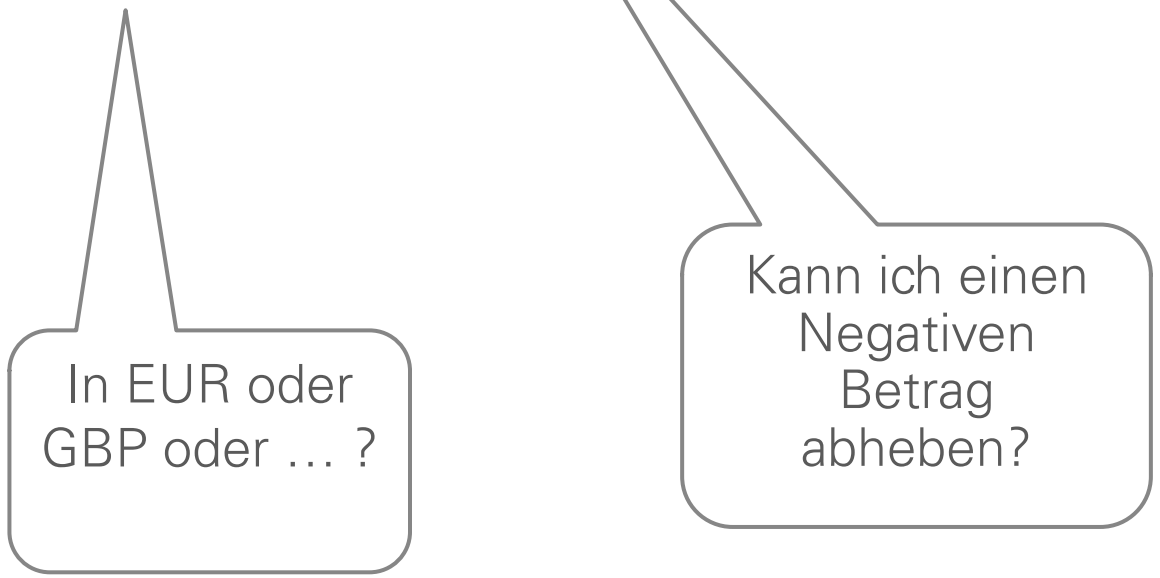
Hamcrest Matchers können verwendet werden

EIN BANKKONTO – VERTRAGSMODELL MIT CODE CONTRACTS

```
using System.Diagnostics.Contracts;

public class Account {

    // ...

    public void Withdraw(int amount) {
        Contract.Requires(amount <= Balance);
        _balance -= amount;
    }
}
```

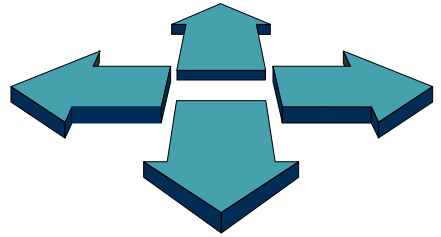
JA, ABER…

```
public class Account {

    // ...

    public void withdraw(int amount) {
        assert amount >= getBalance();
        _balance -= amount;
    }
}
```



ÜBERSICHT

- Entities Implementieren
- Beispiel-Entity
- Vertragsmodell (Design by contract)
- Beispiel-Value-Object
- Identität von Entities



EIN EIGENER BETRAGS-TYP

```
public class Amount {
    private final int _amount;
    private final Currency _currency;

    public Amount(int amount, Currency currency) {
        _amount = amount;
        _currency = currency;
    }
}
```

EIN EIGENER BETRAGS-TYP

```
public class Amount {
    private final int _amount;
    private final Currency _currency;

    public Amount(int amount, Currency currency) {
        _amount = amount;
        _currency = currency;
    }

    public Amount add(Amount otherAmount) {
        return new Amount(_amount + otherAmount._amount, _currency);
    }
}
```

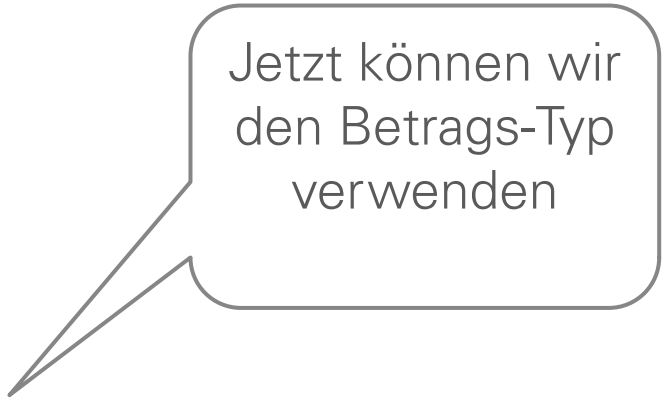
EIN EIGENER BETRAGS-TYP

```
public class Amount {
    private final int _amount;
    private final Currency _currency;

    public Amount(int amount, Currency currency) {
        _amount = amount;
        _currency = currency;
    }

    public Amount add(Amount otherAmount) {
        assert hasSameCurrency(otherAmount);

        return new Amount(_amount + otherAmount._amount, _currency);
    }

    public boolean hasSameCurrency(Amount otherAmount) {
        return otherAmount._currency == _currency;
    }
}
```

… und Verträge, die vor falschen Währungen schützen

EIN BANKKONTO – VIERTER ENTWURF

```
public class Account {
    private Amount _balance;

    public Amount getBalance() {
        return _balance;
    }

    public void deposit(Amount amount) {
        _balance.add(amount);
    }

    public void withdraw(Amount amount) {
        assert amount.isLessOrEqual(getBalance());
        _balance.subtract(amount);
    }
}
```



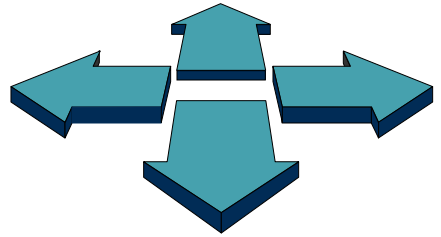
EIN BANKKONTO – VIERTER ENTWURF

```
public class Account {
    private Amount _balance;

    public Amount getBalance() {
        return _balance;
    }

    public void deposit(Amount amount) {
        _balance = _balance.add(amount);
    }

    public void withdraw(Amount amount) {
        assert amount.hasSameCurrency(getBalance());
        assert amount.isLessOrEqual(getBalance());
        _balance = _balance.subtract(amount);
    }
}
```

… und neue Verträge formulieren

ÜBERSICHT

- Entities Implementieren
- Beispiel-Entity
- Vertragsmodell (Design by contract)
- Beispiel-Value-Object
- Identität von Entities



IDENTITÄT VON ENTITIES

- In Java hat jedes Objekt eine Identität
  - Nämlich seine **Referenz**
  - Das ist eine **technische Identität**

- Eine Entität hat eine fachliche Identität
  - **Verschiedene Objekte** können **dieselbe Entität** repräsentieren
  - In verschiedenen Prozessen: z.B. auf dem Client/auf dem Server
  - Wenn sie gespeichert und später wieder aus der Datenbank gelesen werden.

- Die fachliche Identität kann **dem Benutzer bekannt** sein auf zwei Arten:
  - **Explizit** (z.B. IBAN)
  - **Implizit** (z.B. GUID)

EIN BANKKONTO – MIT FACHLICHER IDENTITÄT

```
public class Account {

    // ...

    private final IBAN _iban;

    public Account(IBAN iban) {
        _iban = iban;
    }

    @Override
    public boolean equals(Object other) {
        return _iban.equals(((Account)other)._iban);
    }
}
```

Die Identität ist unveränderlich

Der Typ der Identität ist typischerweise ein Value Object

Die Implementation von equals() basiert auf der Identität

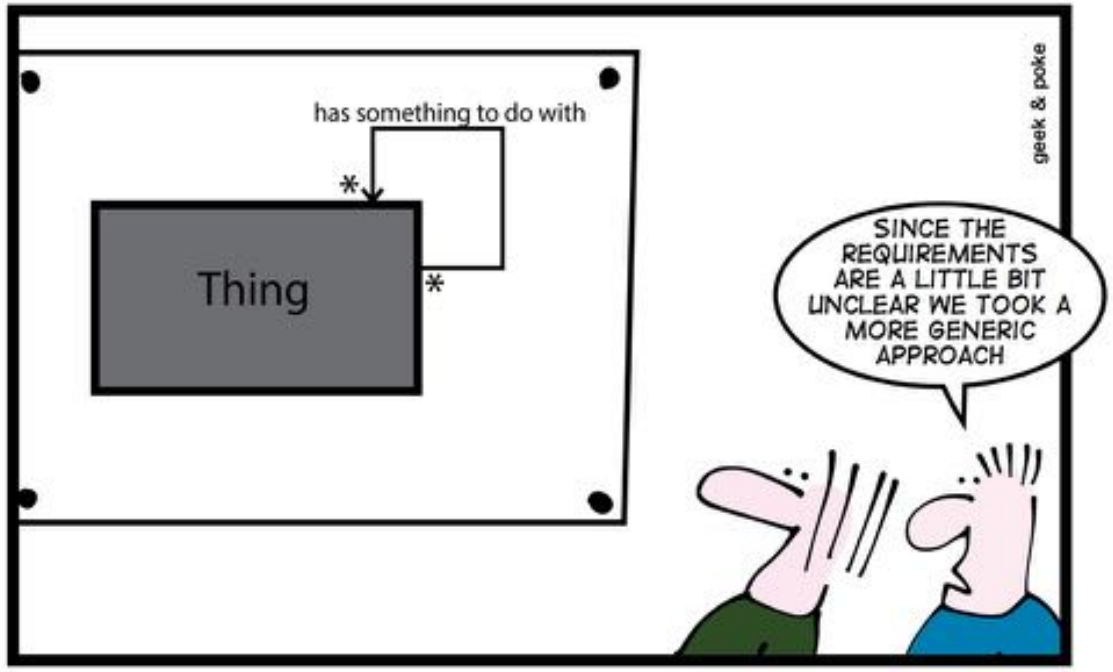
WER ERZEUGT EINE NEUE IDENTITÄT?

- Die Datenbank
  - Pro: Zentral
  - Kann manchmal »lazy« erzeugt werden
- Eine systemweite Fabrik
  - Für dem Benutzer bekannte Formate
  - Oft in dem Repository, das zum passenden Entity-Typ gehört
- Ein GUID-Generator
  - Pro: kann auf dem Client ohne Verbindung zum Server erzeugt werden
  - Con: String in einem besonderen Format

ÜBUNG: MOBPROGRAMMING

WAS WIR NICHT BAUEN WOLLEN

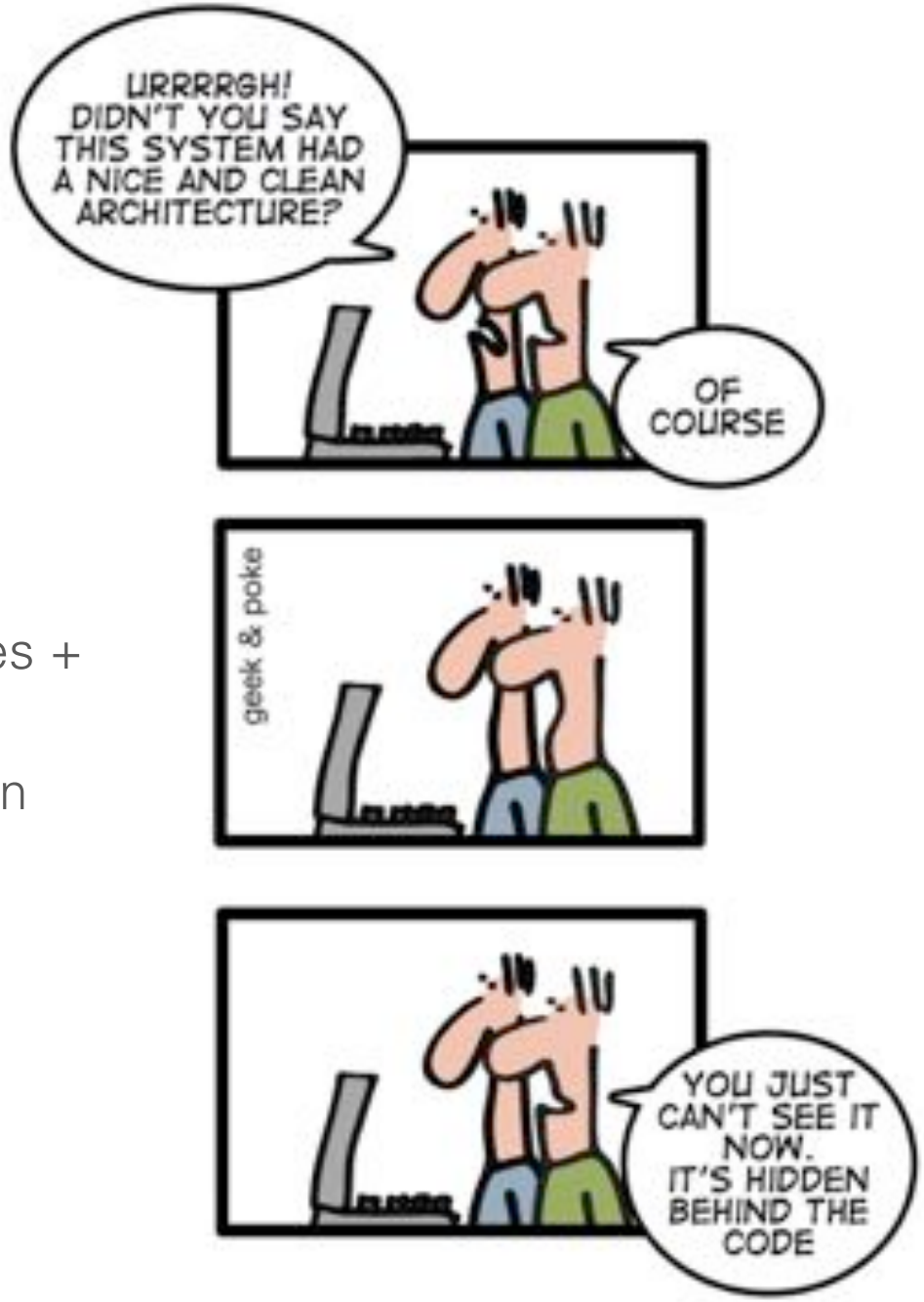
ANEMIC DOMAIN MODEL



ANEMIC DOMAIN MODEL

- „blutarme“ fachliche Objekte
- Schnittstelle ohne Aussagekraft
  - aus Gettern/Settern
  - Viele String Parameter

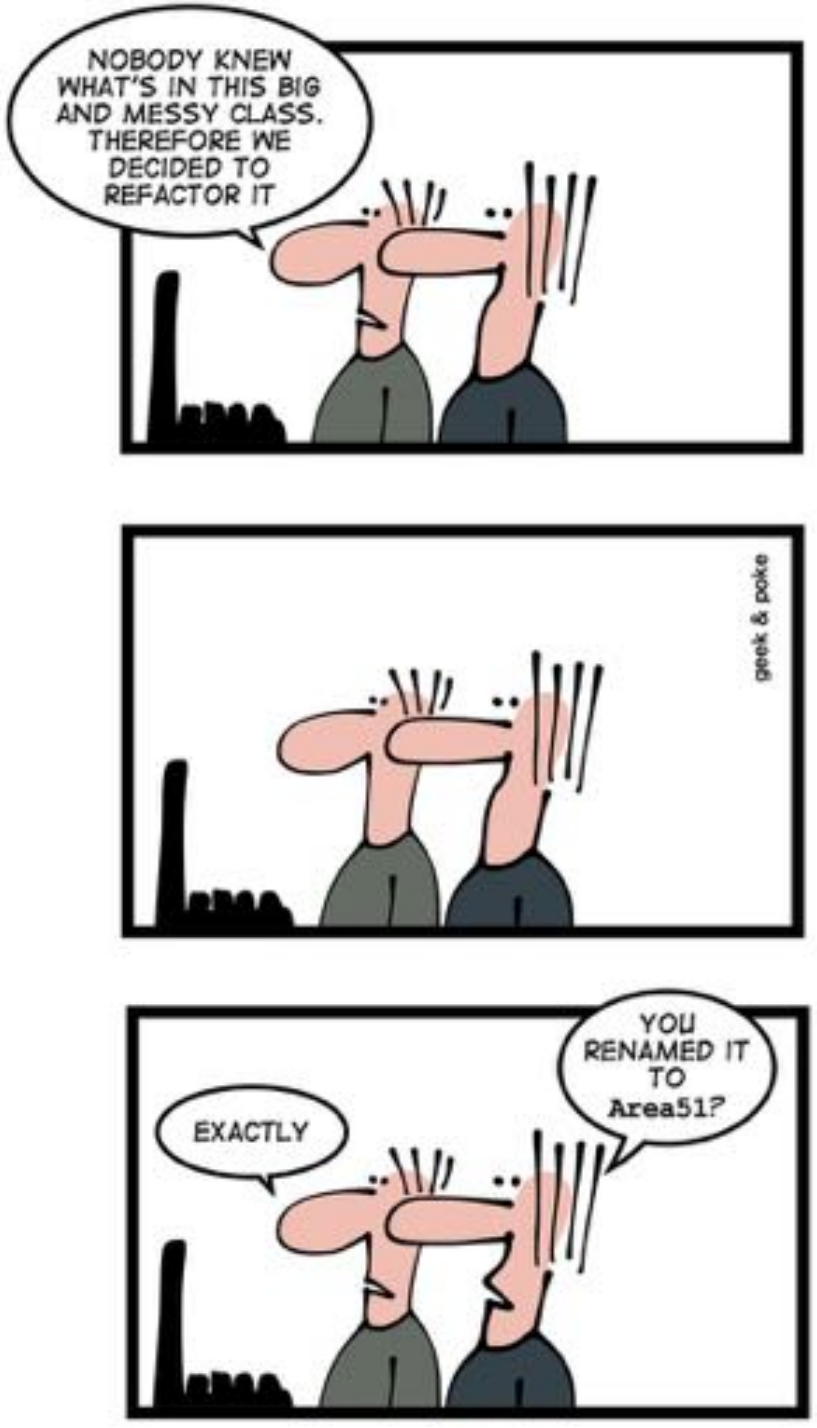
- Eigentliche Fachlichkeit außerhalb Entities + Value Objects in Services oder im UI
- Oberhalb des Modells findet man Klassen mit dupliziertem Code zur
  - Konsistenzprüfung
  - Konvertierung
  - Validierung
- Viele Util, Helper und Manager Klassen



ANEMIC → FEHLENDE ROBUSTHEIT

- Designschulden
  - Unklarer, schwer verständlicher Entwurf
  - Verteilte Fachlichkeit
  - Copy&Paste-Programmierung

→ Teure Wartung
→ Duplizierter Code
→ Viele Refactorings
→ Schlechte Testbarkeit



literatur

speakerdeck.com/hschwentner

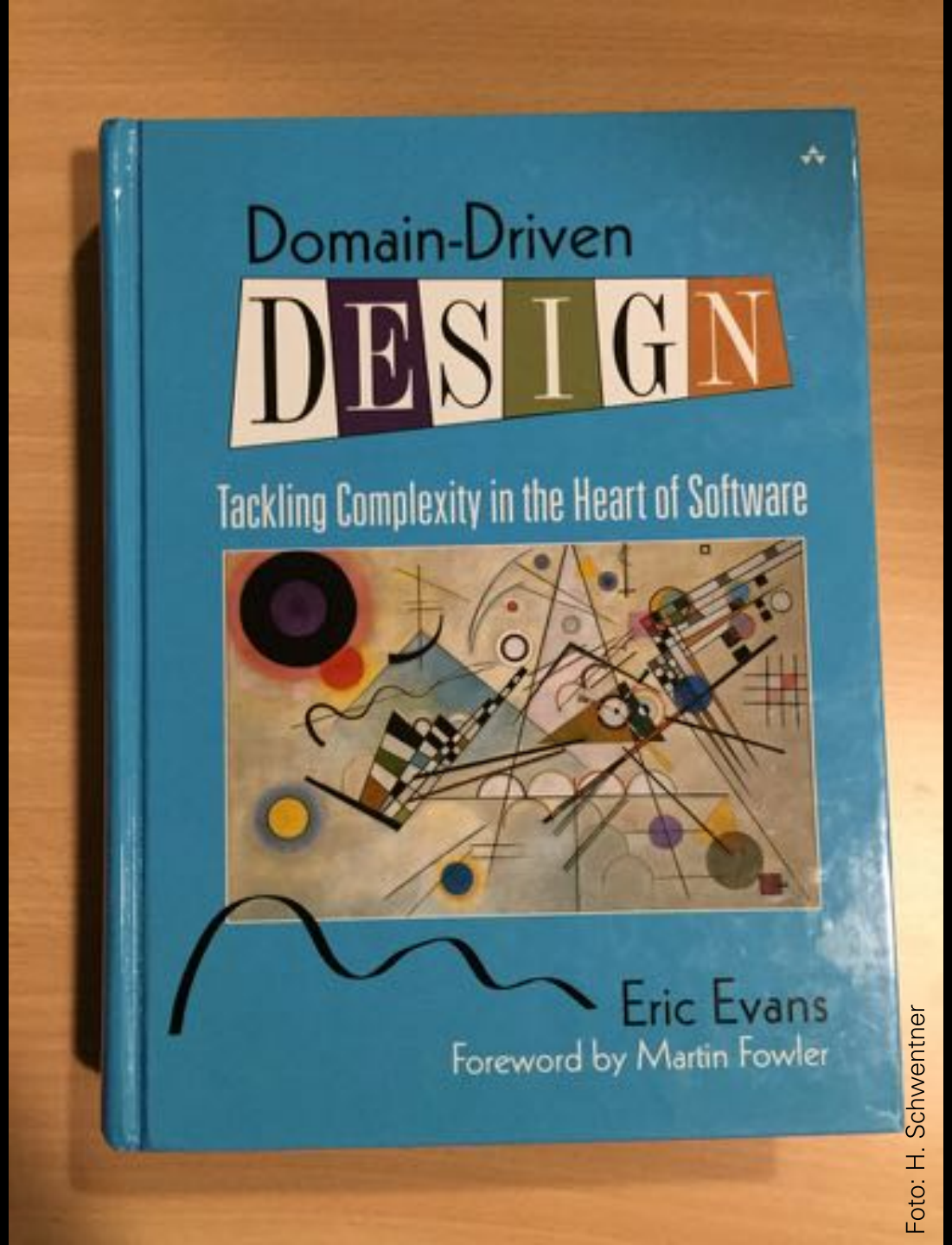


Foto: H. Schwentner

Vaughn Vernon

Domain-Driven Design kompakt

→ Aus dem Englischen übersetzt
von Carola Lilienthal und Henning Schwentner

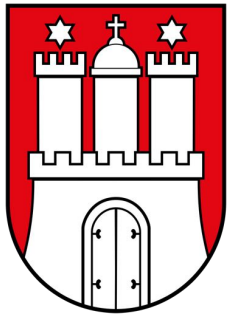
dpunkt.verlag

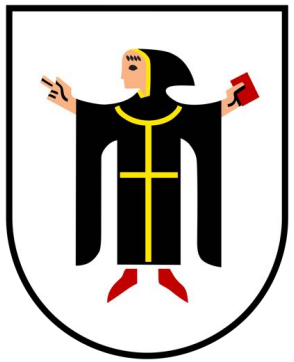
EVENT STORMING

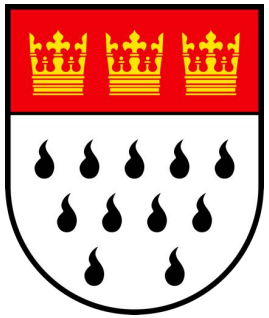
Alberto Brandolini

MEETUPS

@DDDger







KONFERENZEN

DOMAIN DRIVEN DESIGN EUROPE

Kandinsky

Germanys first DDD conference

Workshop
Domain-Driven Design *konkret*

iSAQB International Software Architecture Qualification Board

wps.de/ddd

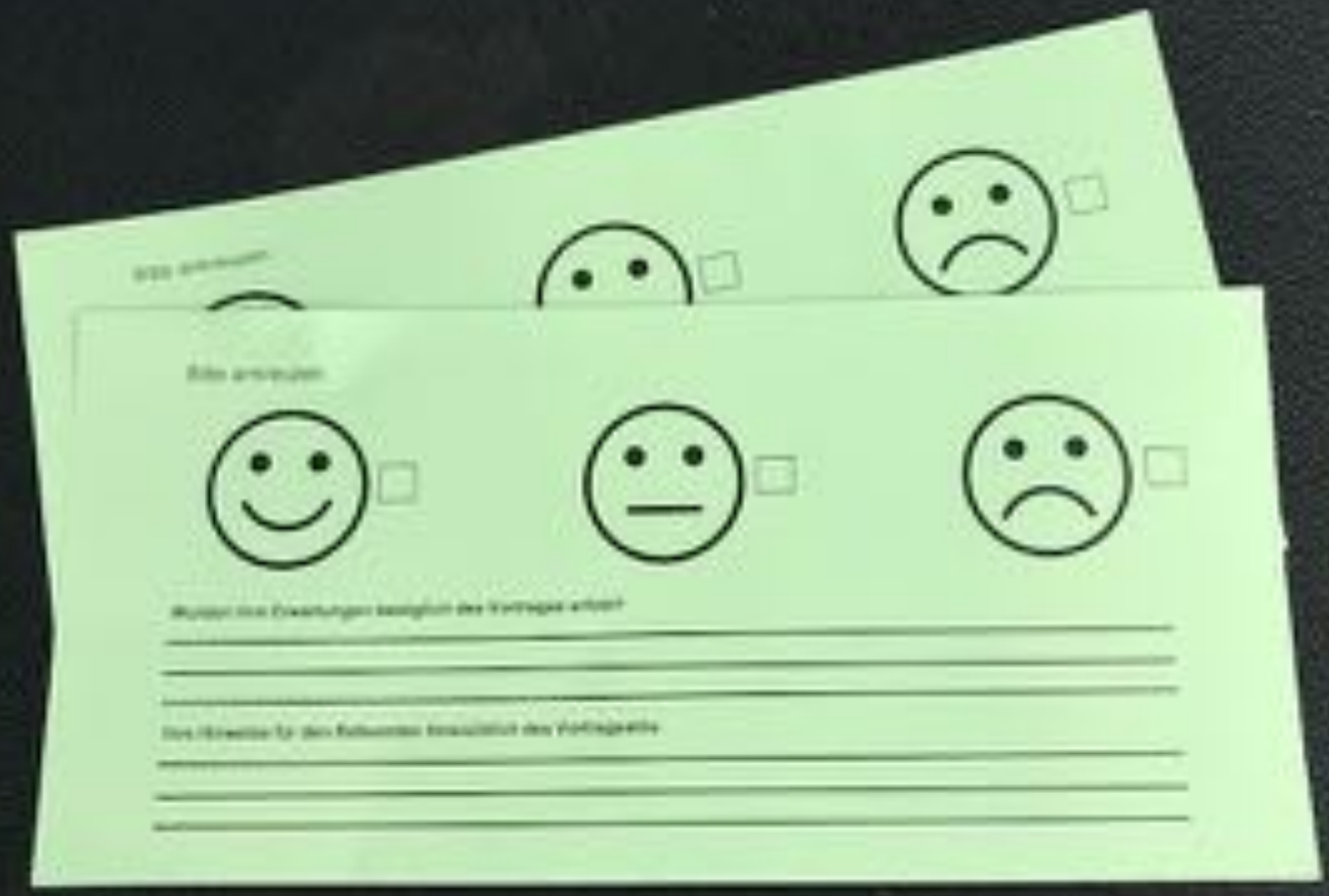
Schulung
Domain-Driven Design
Advanced Level
CREDIT POINTS
20
10
Anmeldung unter www.wps.de/ddd
WORKPLACE SOLUTIONS
iSAQB

Rabatt!

FEEDBACK

@hschwentner

@hschwentner




Happy End
Recycling
öffnen
3
aus
Recycling-
papier
aus
Recyclingpapier
200
8x

Henning Schwentner

hs@wps.de

@hschwentner

DDD

Foto: A. Markiewicz