PROFESSIONAL SOFTWARE ENGINEERING

PSE SWE LE 4 und 5 - Domain Driven Design

DDD als Designaufgabe

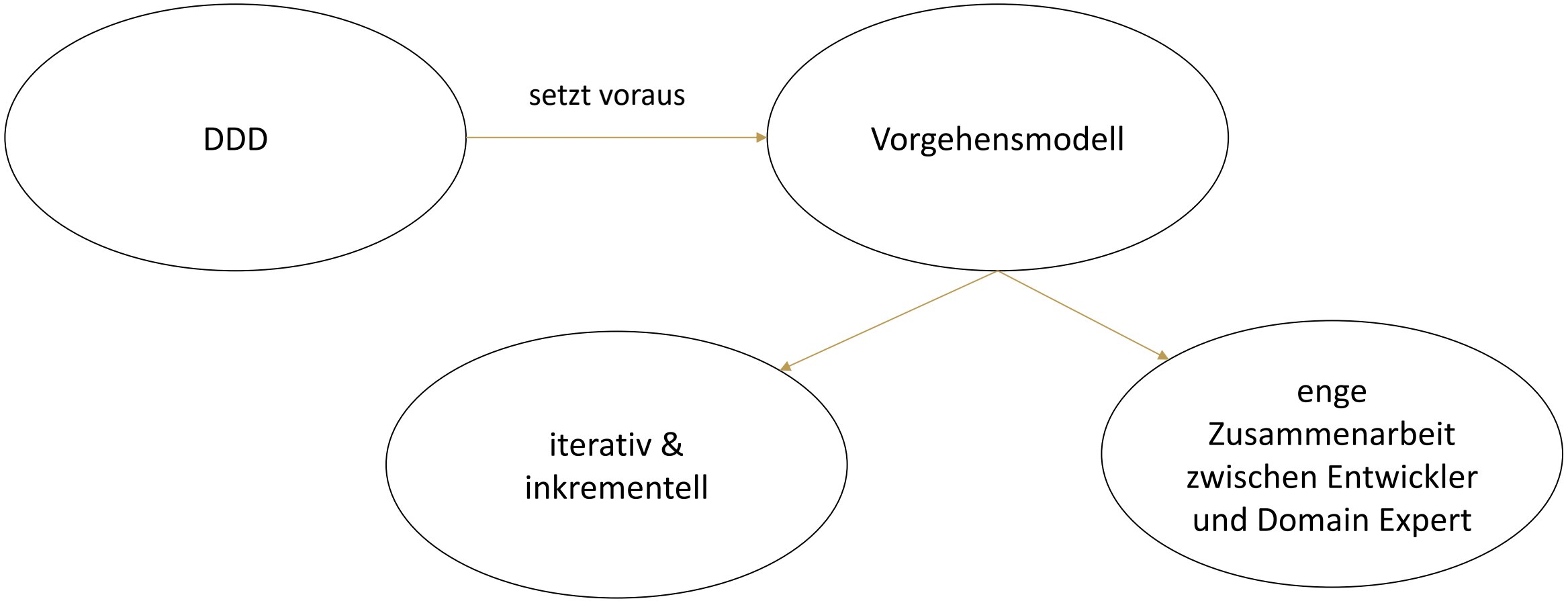
Dominik Neumann

DDD STELLT DAS MODELL IN DEN MITTELPUNKT

**Essentially, all models are wrong,
but some are useful.**

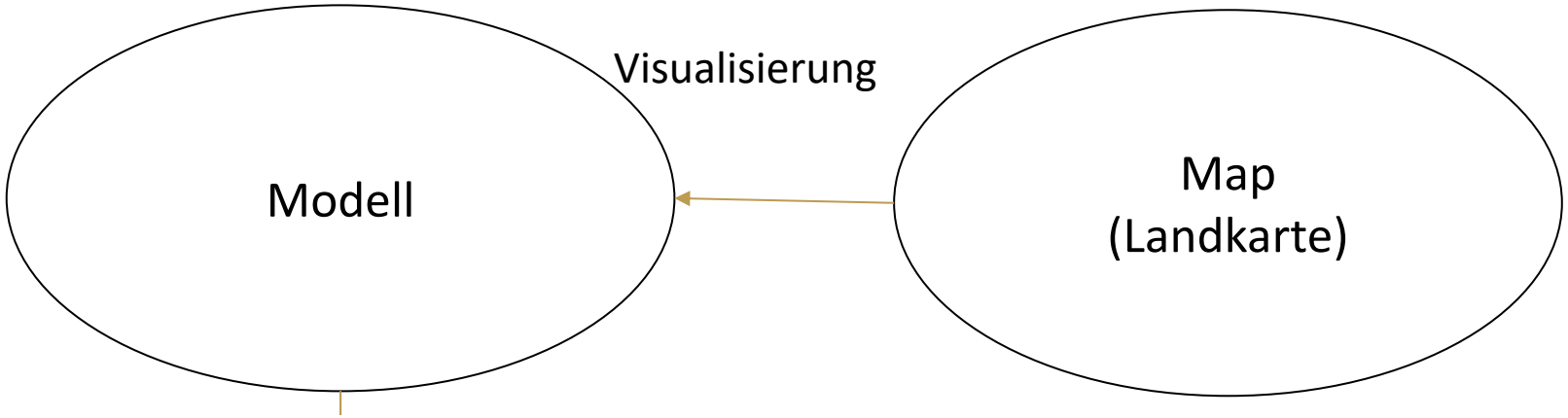
- George E. P. Box

VORAUSSETZUNGEN FÜR DDD



VON DER DOMÄNE (*DOMAIN*) ZUM *DOMAIN MODEL*

Ein Modell ist…
- Vereinfachung
- Abstraktion relevanter Aspekte
- schafft Strukturen



Ein Modell ist eine Vereinfachung der Realität oder einer Idee

Realität

Es repräsentiert Aspekte der Realität die von Interesse sind.

Es ist eine Abstraktion relevanter Aspekte mit dem Ziel ein Problem zu verstehen und darauf aufbauend eine Lösung zu entwickeln.

(Aspekte sind für die Lösung relevant)

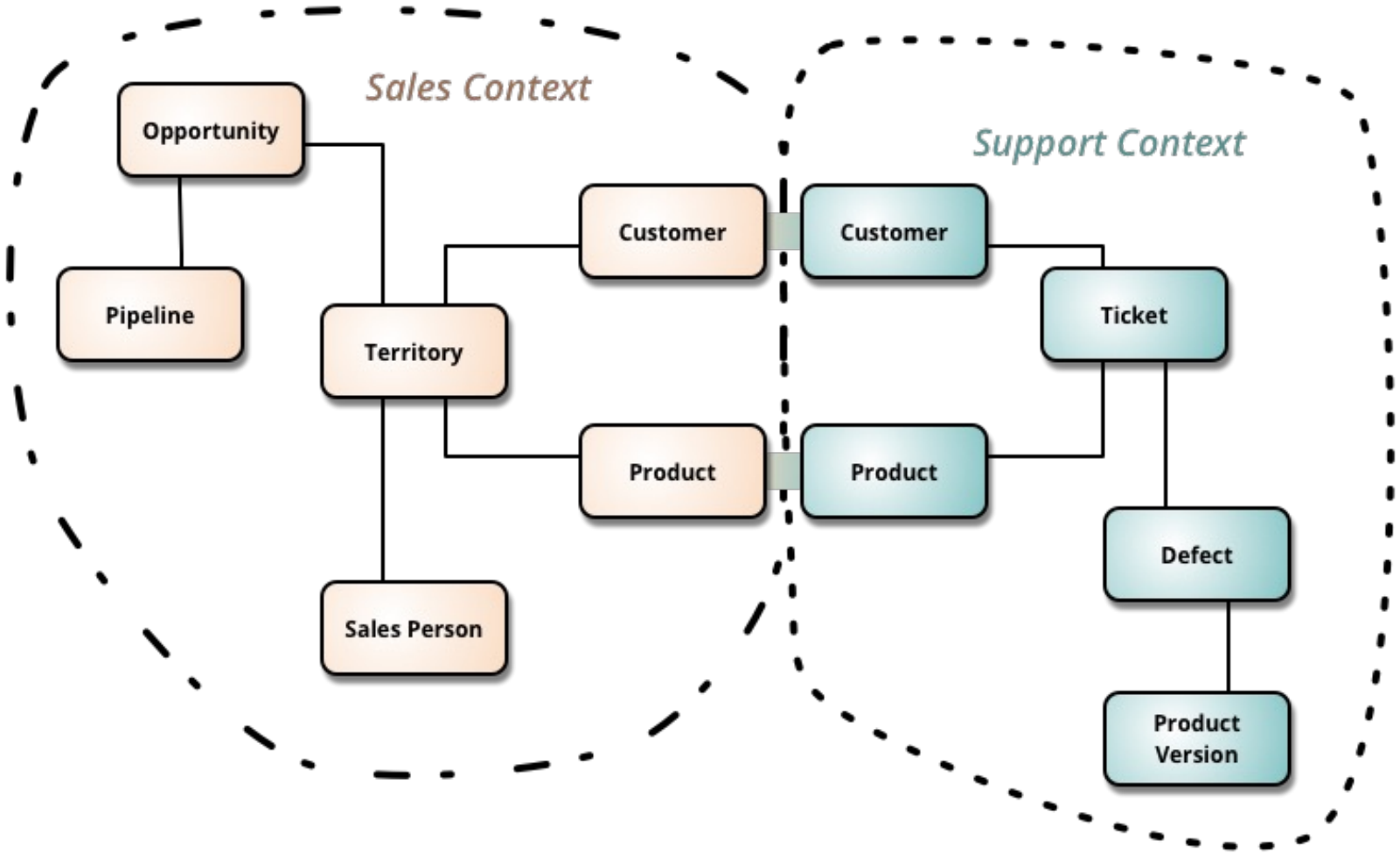
Idee

Problemraum

BEISPIELE FÜR MODELLE

BEISPIELE FÜR MODELLE

Two Domain Models

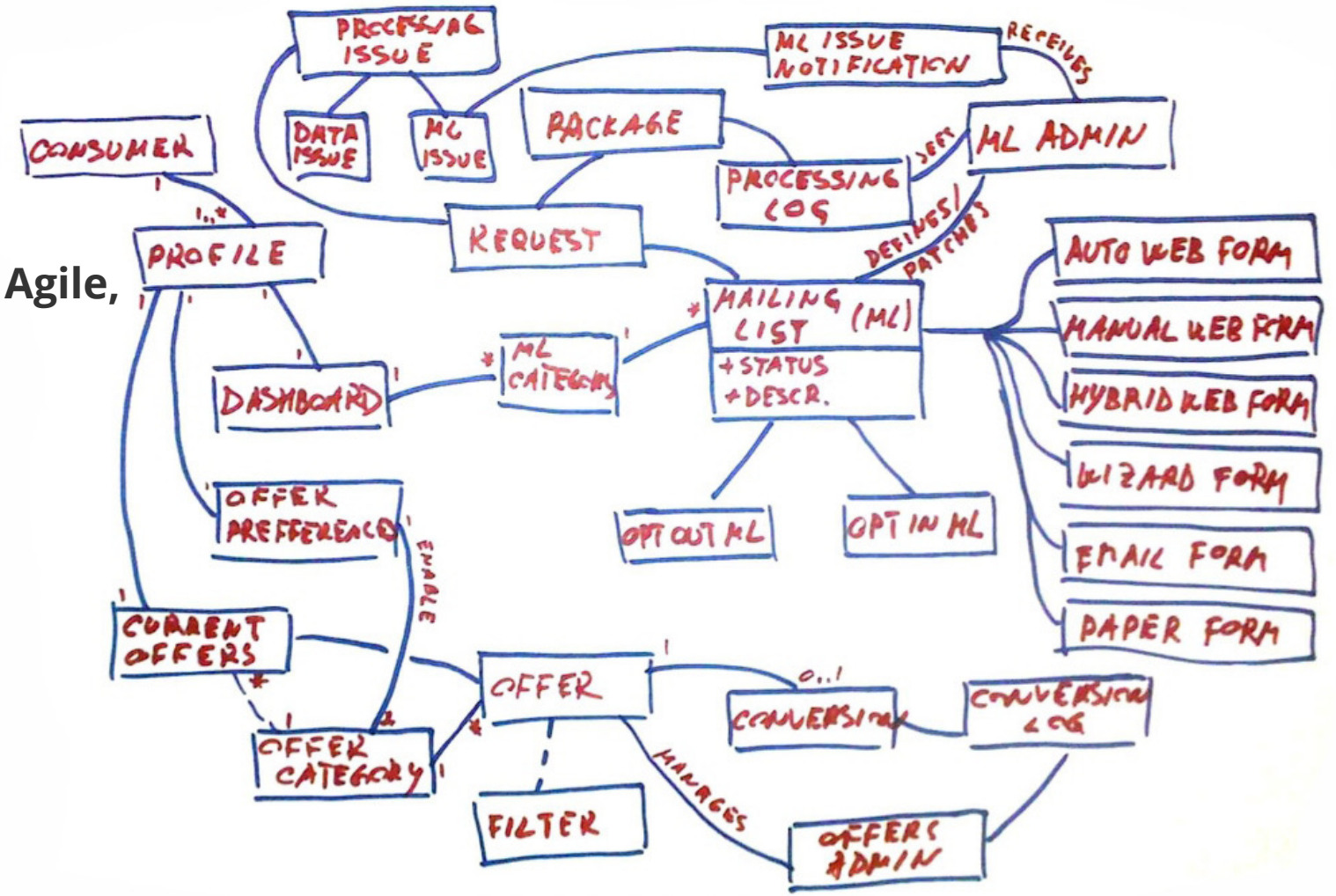


Quelle: Martin Fowler, https://martinfowler.com/bliki/BoundedContext.html

BEISPIELE FÜR MODELLE

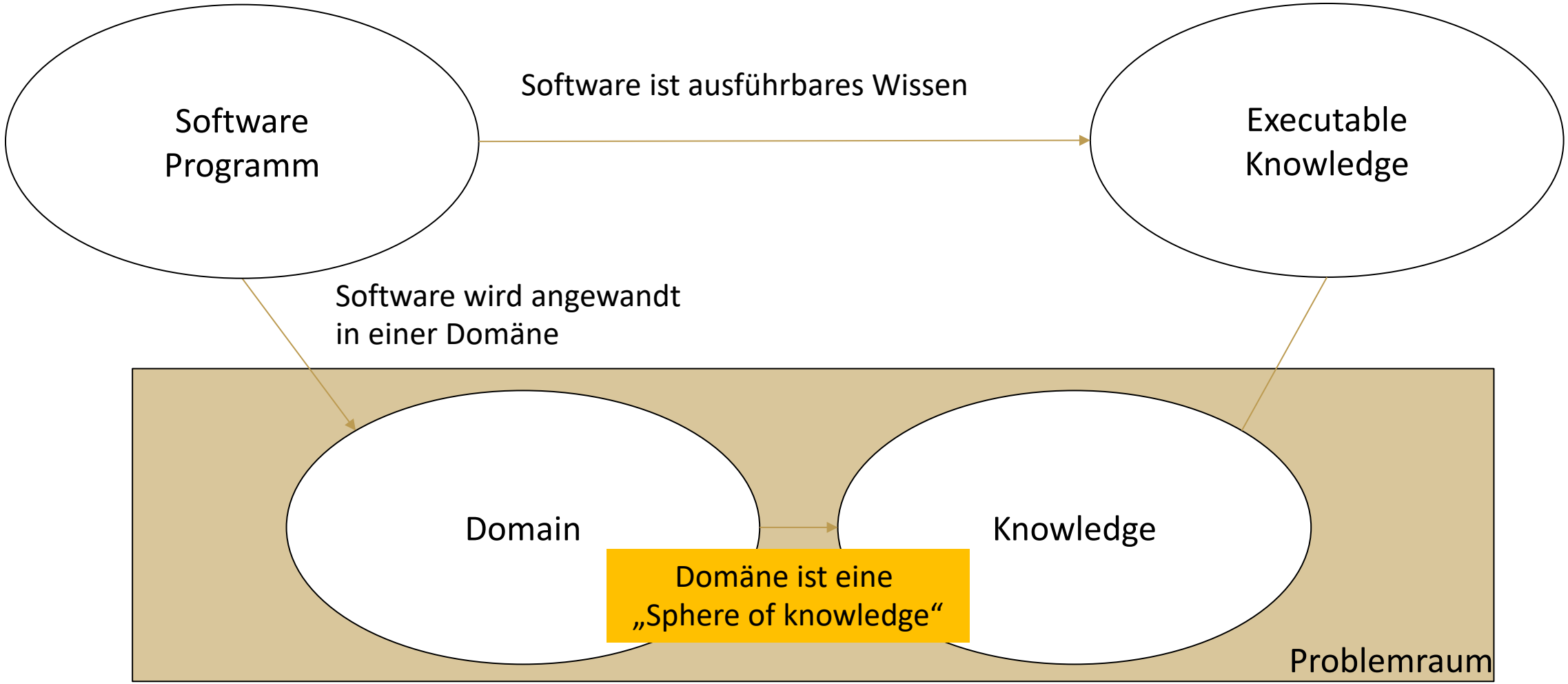
if you only model one thing in Agile, model the domain.

- Scaled Agile, Inc.



Quelle: https://www.scaledagileframework.com/domain-modeling/

VON DER DOMÄNE (*DOMAIN*) ZUM *DOMAIN MODEL*

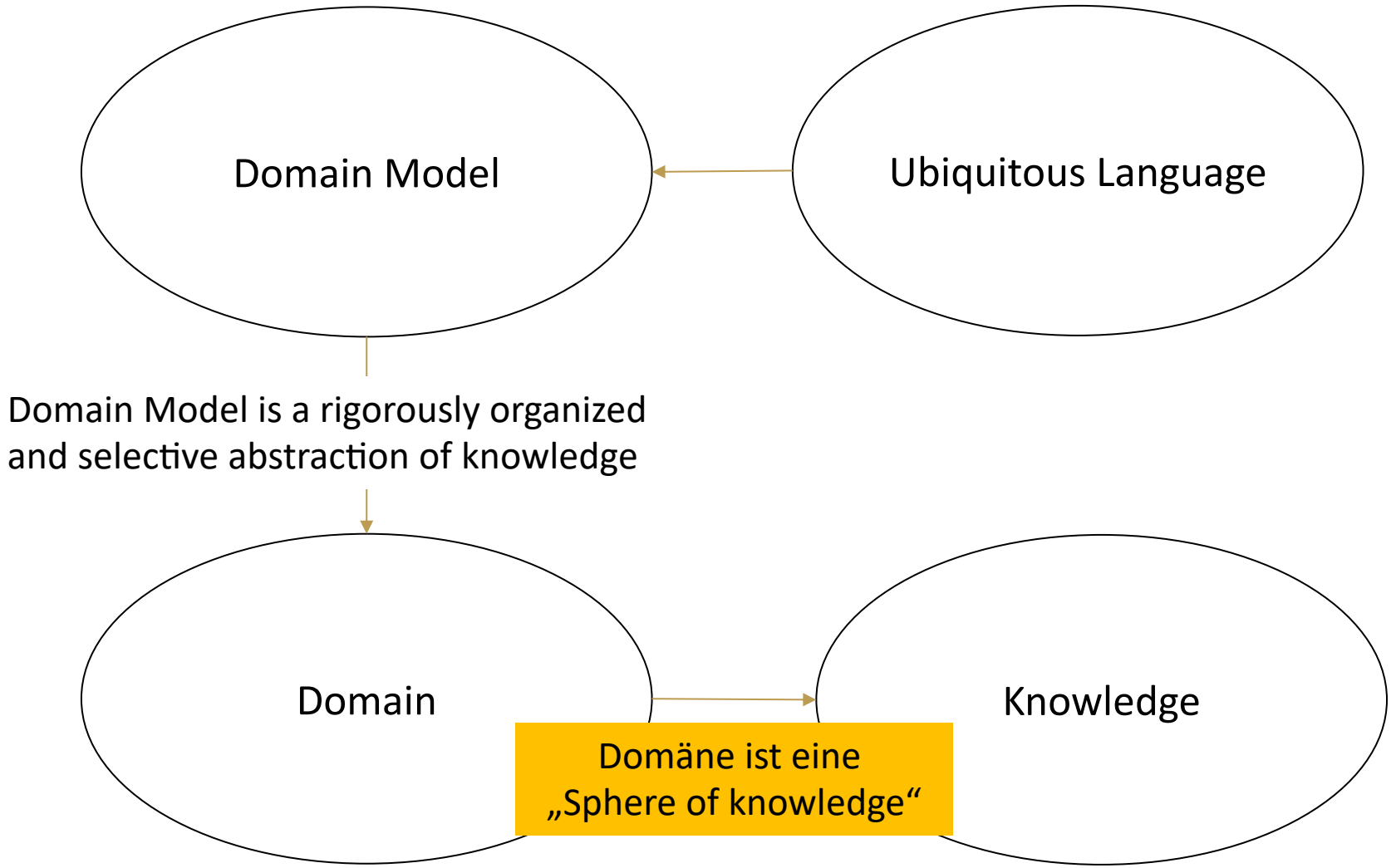


VON DER DOMÄNE (*DOMAIN*) ZUM *DOMAIN MODEL*

Das Domain Model ist eine Abstraktion der Realität.

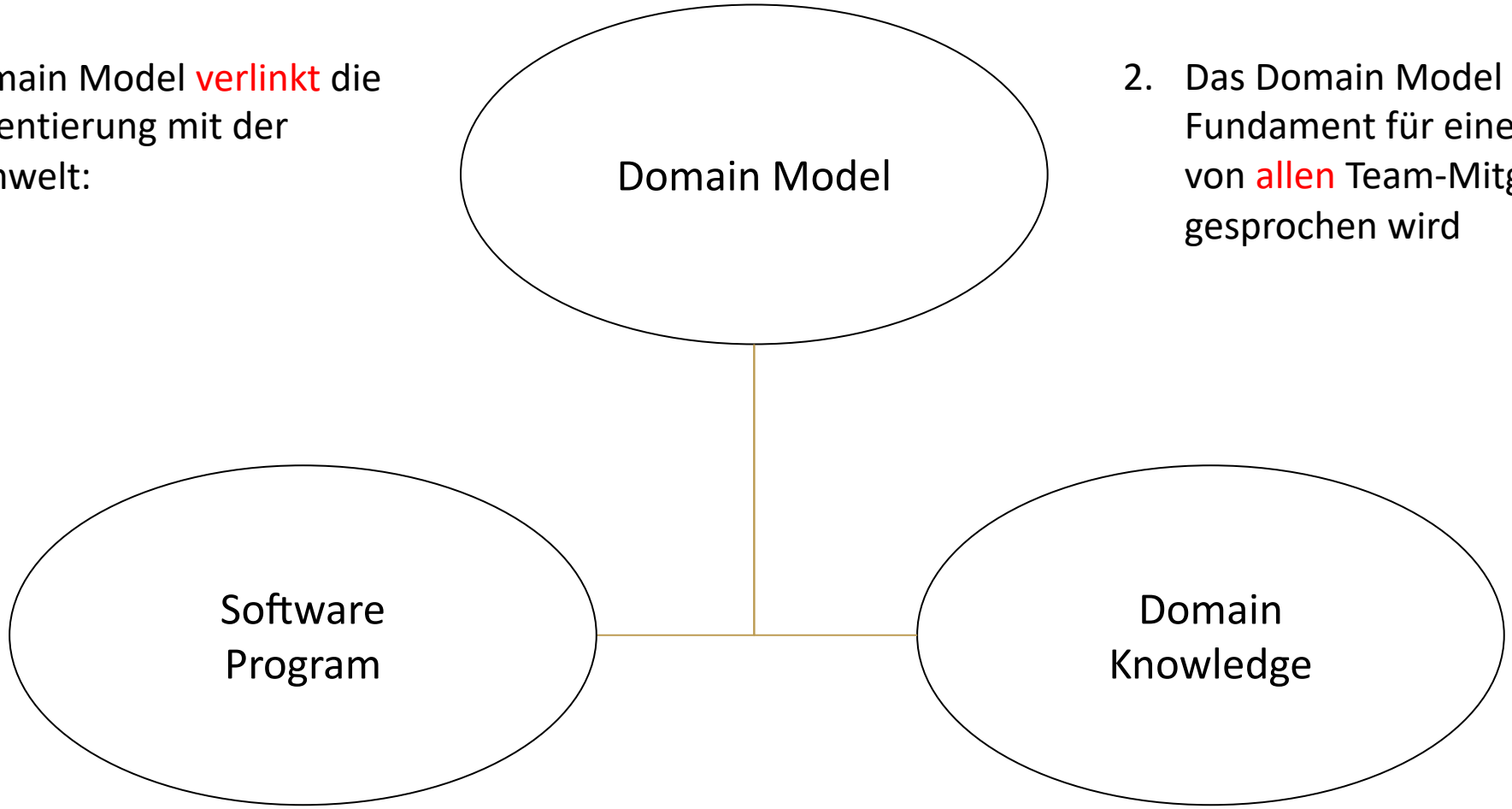
Verschiedene Repräsentationen eines Domain Model müssen auf einer gemeinsamen Sprache aufsetzen: Ubiquitous Language.

Die endgültige Wahrheit
Ist die Repräsentation als Code!



VON DER DOMÄNE (*DOMAIN*) ZUM *DOMAIN MODEL*

1. Das Domain Model verlinkt die Implementierung mit der Problemwelt:



2. Das Domain Model ist das Fundament für eine Sprache die von allen Team-Mitgliedern gesprochen wird

3. Das Domain Model ist der Weg, wie das Team das Wissen (*domain knowledge*) aufbereitet und strukturiert:
   „the model is distilled knowledge“ – „the software is the executable knowledge“

Quelle: Evans: p. 3 The Utility of a Model in DDD

KNOWLEDGE CRUNCHING – DDD ALS ZIELGERICHTETER WISSENSERWERB

When we set out to write software, we never know enough.
… we don‘t realize how much we don‘t know. (Eric Evans)

**It‘s developers‘ (mis)understanding, not domain experts‘ knowledge,
that gets released in production**

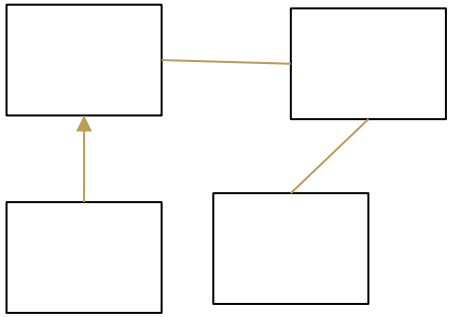
Software development is a learning process, working code is a side effect.

(Alberto Brandolini)

KNOWLEDGE CRUNCHING

Domain Knowledge

Mental Model



Solution Model

```
for method_invocation in method['invocations']:
    if method['identity'] != method_invocation:
        invoked_method = exxfer2_methods[method_invocation]
        target_fqn = self.get_full_qualified_class_name(invoked_method['package'], invoked_method
        relations.extend(self.build_relations(source, [target_fqn], RELATION_CALL, exxfer2_entit
return relations

staticmethod
ef inspect_relation_trace(relations: List[dict], entities: List[dict]) -> str:
ep = {
    entity['entityId']
    for entity in entities if entity['is_entrypoint']
}
```

Code

Discovery

Design

Implementation

Wie stellen wir sicher, dass bei der Transformation des Wissens in Code nicht verloren geht oder Rauschen dominiert?

Quelle: Vlad Kohonen: Figure 2.2

KNOWLEDGE CRUNCHING

Problemstellung:

- Das Verstehen von Anforderungen setzt das Verstehen der Terminologie der Problemdomäne, deren Konzepte und Beziehungen voraus. Domänen können kompliziert sein und es kann Jahre dauern, bis man sie vollständig durchdrungen hat und beherrscht.

- Wie kann ein Entwickler schnell ausreichendes Wissen über eine Anwendungsdomäne erlangen?

KNOWLEDGE CRUNCHING

Lösung:

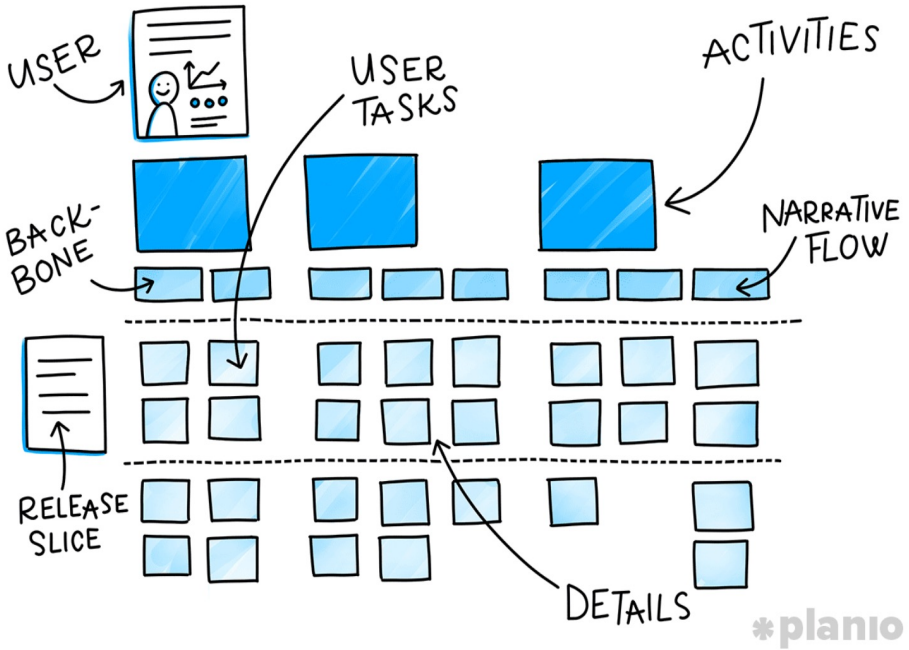
- Knowledge Crunching - Methoden für die Zusammenarbeit mit Domain-Experten um schnell relevantes Wissen zu erlangen.

- Im Vordergrund steht der Lernprozess und die Selektion relevanten Wissens zur Wissensvertiefung
  - Effizientes Lernen und Wissenserwerb (Fokus auf relevantes Wissen)
  - Wissen wird im Team verteilt und im Modell konserviert und dann in die Software gegossen.
  - Kombination aus Active Listening (Aktives Zuhören) und Facilitation (Moderation)

KNOWLEDGE CRUNCHING

User Story Mapping (Jeff Patton *)

- Story Maps sind Landkarten für das Product Backlog
- Moderation: Ausgehend vom User wird das Big Picture der Anwendung übersichtlich und strukturiert herausgearbeitet.
- Vorteile:
  - Fokus auf User Experience und Impact sowie Minimierung von Release-Umfängen.
    → min Output und max Outcome / Impact
  - Übersichtliche Darstellung des Big Picture.
  - Fördert Gruppendiskussion und Wissensverteilung



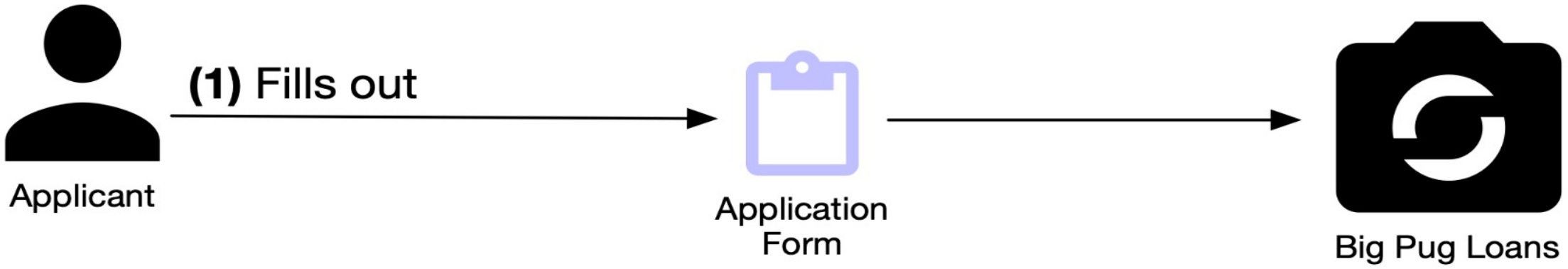
*After a first article in 2005 and a blog post in 2008 in 2014 Jeff Patton published the user-story mapping technique

KNOWLEDGE CRUNCHING

Domain Storytelling

- Interview- und Modellierungs-Technik: Während der Domain Experte in Erzähltechnik über seine Domäne spricht, wird in Echtzeit mittels einer visuellen Sprache die Erzählung dokumentiert.

- Einfache Sätze ohne Verzweigungen: Subjekt - Prädikat - Objekt

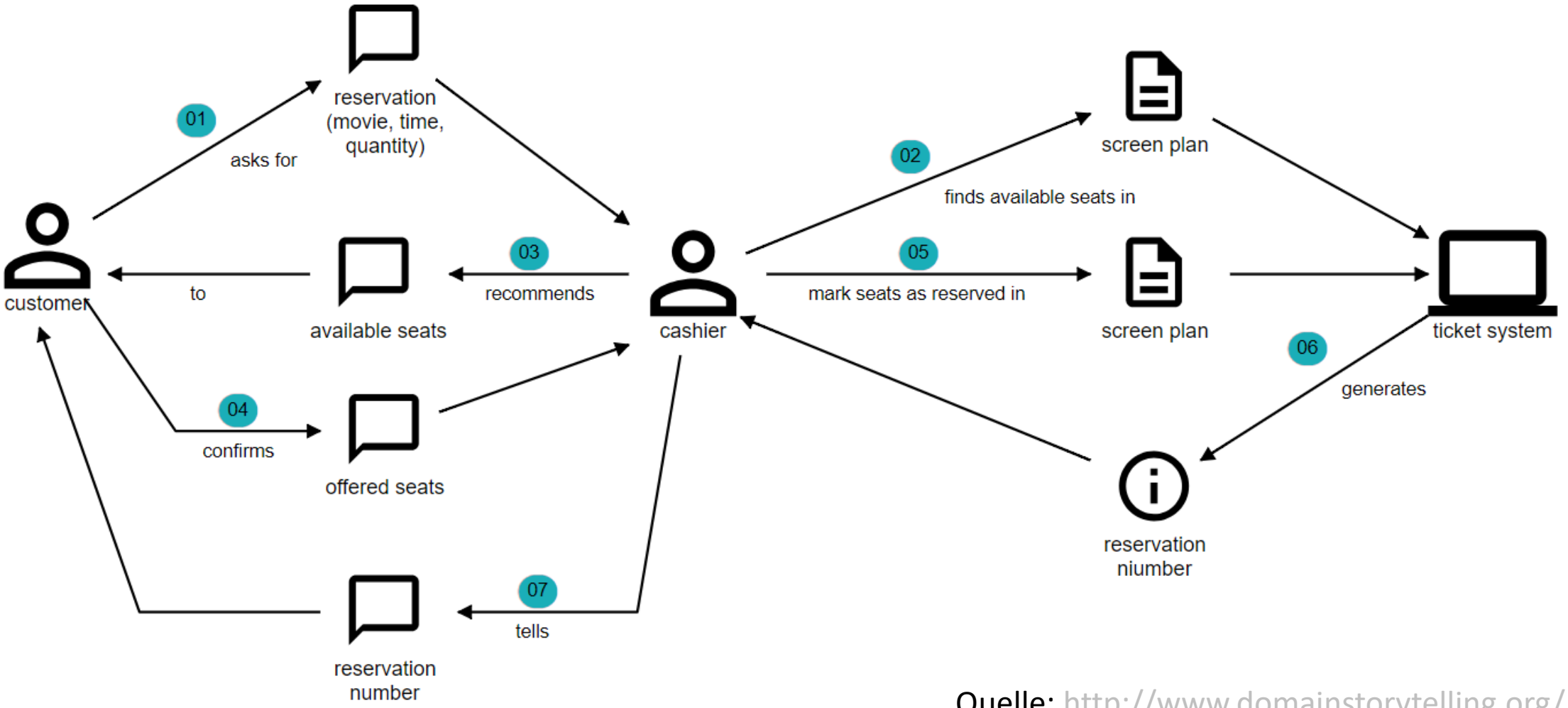


Simple Domain Storytelling example

Quelle: http://www.domainstorytelling.org/

KNOWLEDGE CRUNCHING

Domain Storytelling



Quelle: http://www.domainstorytelling.org/

KNOWLEDGE CRUNCHING

EventStorming (Alberto Brandolini)

- Brainstorming-Technik, in der ein Geschäftsproblem mit Hilfe von Domain Events untersucht wird. Domain Events sind Ereignisse, die für Domain Experten von Interesse sind: relevante Ereignisse.

- Ereignisse werden immer in der Vergangenheitsform notiert. (Sie haben sich ja bereits ereignet.)

- Domain Events sind dann Ausgangspunkt von allem anderen (einem Ereignis geht in der Regel eine Aktion voraus, diese hat eventuell einen User oder ein System, der die Aktion durchführt, ….)

- Wir schauen also von Hinten auf das Problem (backward-looking view)

Quelle: http://www.eventstorming.org/

KNOWLEDGE CRUNCHING

EventStorming (Alberto Brandolini)

- Big Picture Workshop
- Design Level
- IT Modernization
  - Concept Location: Services identifizieren, die dann mittels Strangler Pattern externalisiert warden
  - To-Be Modularisierung

Quelle: http://www.eventstorming.org/

KNOWLEDGE CRUNCHING

Was das Knowlede Crunching schwierig macht:

Five Orders of Ignorance (P. Armour)

(1) Ich weiß etwas und kann es gut und greifbar darstellen bzw. erklären.

(2) Ich weiß etwas nicht, bin mir dessen aber bewusst und auch bewusst, wie ich es mir aneigne.

(3) Ich weiß nicht, dass ich etwas nicht weiß bzw. ich weiß nicht genug, um zu wissen, was ich nicht weiß.

(4) Ich kenne keinen Weg, um herauszufinden, dass ich nicht weiß, dass ich etwas nicht weiß.

(5) Ich kenne die „Five Orders of Ignorance“ nicht bzw. ich weiß nicht, dass Software Entwicklung die Aktivität ist Wissen aufzubauen und weiß auch nicht welche Stufen mein Wissen hat.

- Lack of Ignorance
- Lack of Knowledge
- Lack of Awareness
- Lack of Process
- Meta-Ignorance

KNOWLEDE CRUNCHING - SCARF-MODELL

Knowledge Foundation
@ Reutlingen University

- ***S****tatus*: Relative Stellung zwischen Mitarbeitern, sozialer Status
- ***C****ertainty (Vorhersehbarkeit)*: Vorhersehbarkeit von zukünftigen Situationen
- ***A****utonomy (Autonomie)*: Beeinflussung, Kontrolle und Gestaltung des eigenen Umfelds
- ***R****elatedness (Soziale Beziehungen)*: Zugehörigkeit zu einer Gruppe;
- ***F****airness*: Gerechtigkeit.

- Das SCARF Modell beschreibt basierend auf den Erkenntnissen der modernen Hirnforschung elementare Grundbedürfnisse des Menschen.
- Werden diese Grundbedürfnisse erfüllt, können Menschen kooperativ und vertrauensvoll miteinander arbeiten.
- Das SCARF Modell ist ein zentrales Konzept des Neuroleadership (Rock 2008).

https://wirtschaftslexikon.gabler.de/definition/scarf-54113

STRATEGISCHES UND TAKTISCHES DESIGN

STRATEGISCHES UND TAKTISCHES DESIGN IM ÜBERBLICK

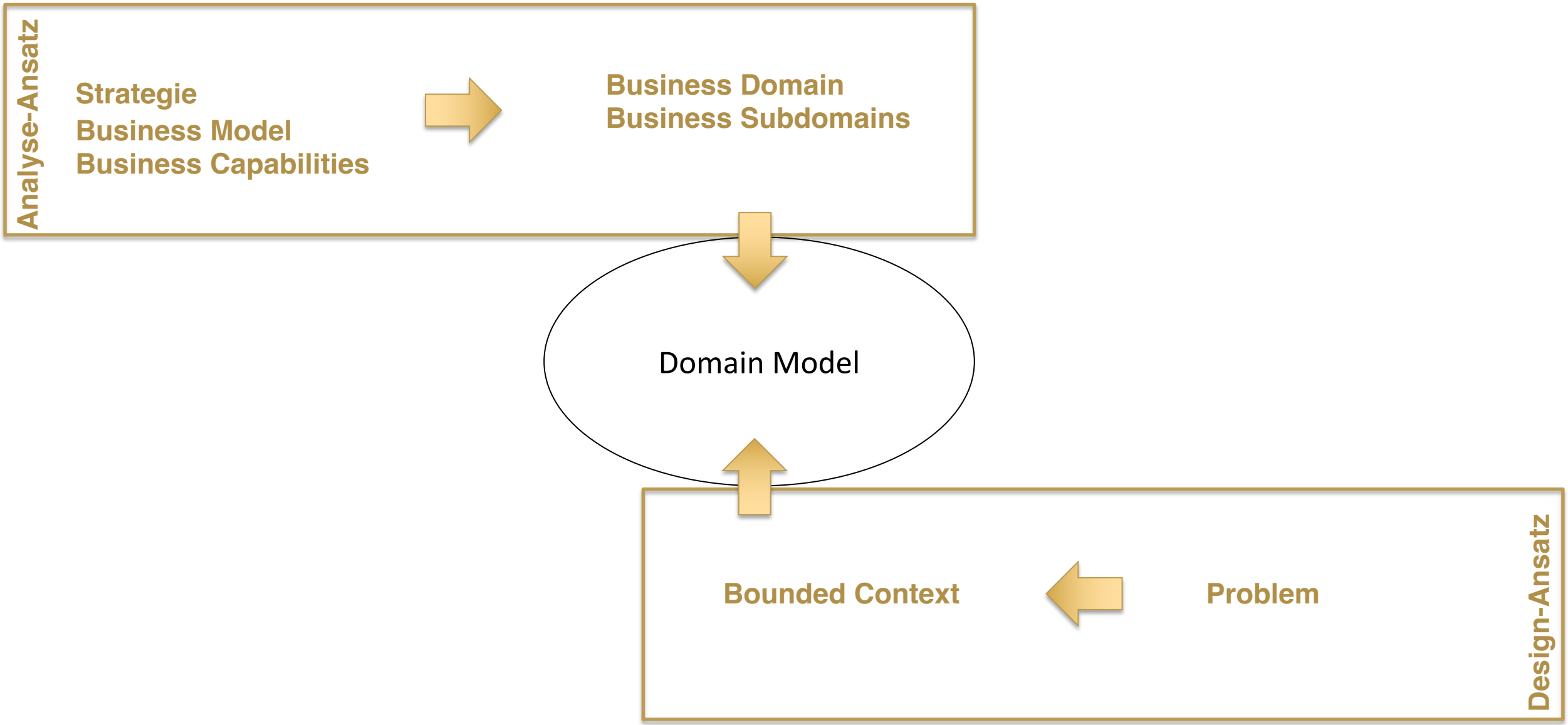
- Strategisches Design definiert die übergeordnete Sichtweise, die groben Leitplanken - was strategisch für das Geschäft wichtig ist.
- Es hilft uns ein Problem zu strukturieren. Es in Teilprobleme zu zerlegen, die wir unabhängig voneinander analysieren können.
- Dazu lernen wir 2 verschiedene Zerlegungsmuster kennen:
  - Subdomain
  - Bounded Context

STRATEGISCHES UND TAKTISCHES DESIGN IM ÜBERBLICK

- **<u>Top-Down-Ansatz (Analyse Ansatz)</u>**
- Um die Strategie des Business besser zu verstehen, wird die Business Domain
- in ***Subdomains*** gegliedert. Eine Subdomain hat ihr eigenes Domänen Modell.
- Wir analysieren unterschiedliche Arten von Subdomains. Diese geben Aufschluss darüber, wie das Unternehmen funktioniert.

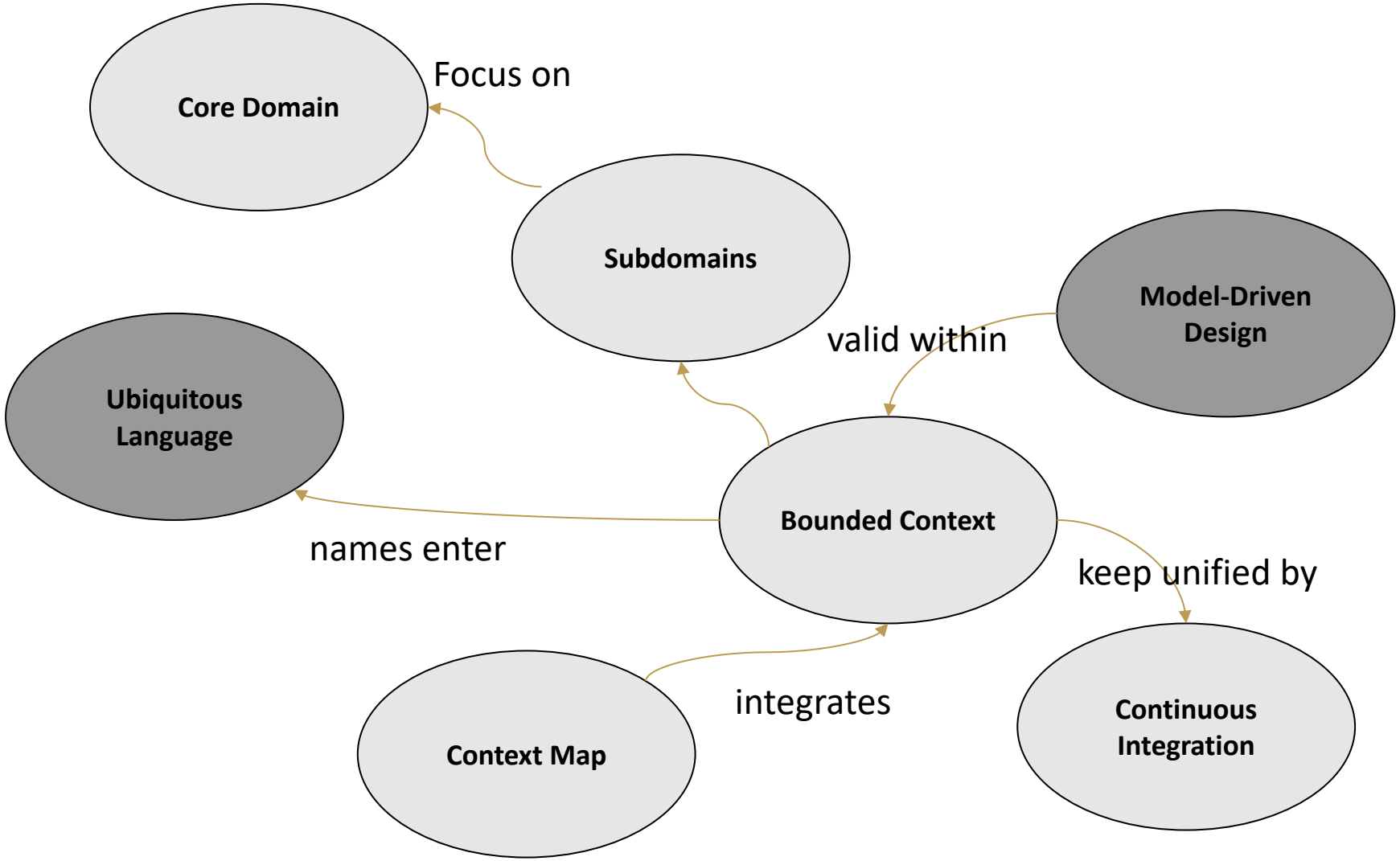
- **<u>Bottom-Up-Ansatz (Design Ansatz)</u>**
- Für jeden Problembereich (Problem Domain) entwickeln wir ein Domain Model.
- Mit ***Bounded Context*** *werden* Domänen Modelle voneinander abgrenzt.
- Die Wahl für die Modellgrenze ist eine strategische Design-Entscheidung. Wir als Team treffen diese Entscheidung.

STRATEGISCHES UND TAKTISCHES DESIGN IM ÜBERBLICK




Knowledge Foundation
@ Reutlingen University
STRATEGISCHES DDD

STRATEGISCHE MUSTER IM ÜBERBLICK



„**Tightly relating the code to an underlying model** gives the code meaning and makes the model relevant.“ Eric Evans

Das ist nicht unser Fokus

MDA, DSL, generative Softwareentwicklung

STRATEGISCHE MUSTER IM ÜBERBLICK

- Für jeden Problembereich (Problem Domain) entwickeln wir ein **Domain Model**.
- Mit ***Bounded Context*** *werden* Domänen Modelle voneinander abgrenzt.
- Innerhalb eines Domänen Modells wird eine ***Ubiquitous Language*** entwickelt.

Analogie:

- Jedes Modell ist wie ein Land durch seine Grenze (Bounded Context) definiert. und es wird im Land eine eigene Sprache (Ubiquitous Language) gesprochen.
- Diese Sprache ist allgegenwärtig, wird vom gesamten Team gesprochen. Sie dominiert die gesamte Kommunikation und durchdringt die Software.

STRATEGISCHE MUSTER IM ÜBERBLICK

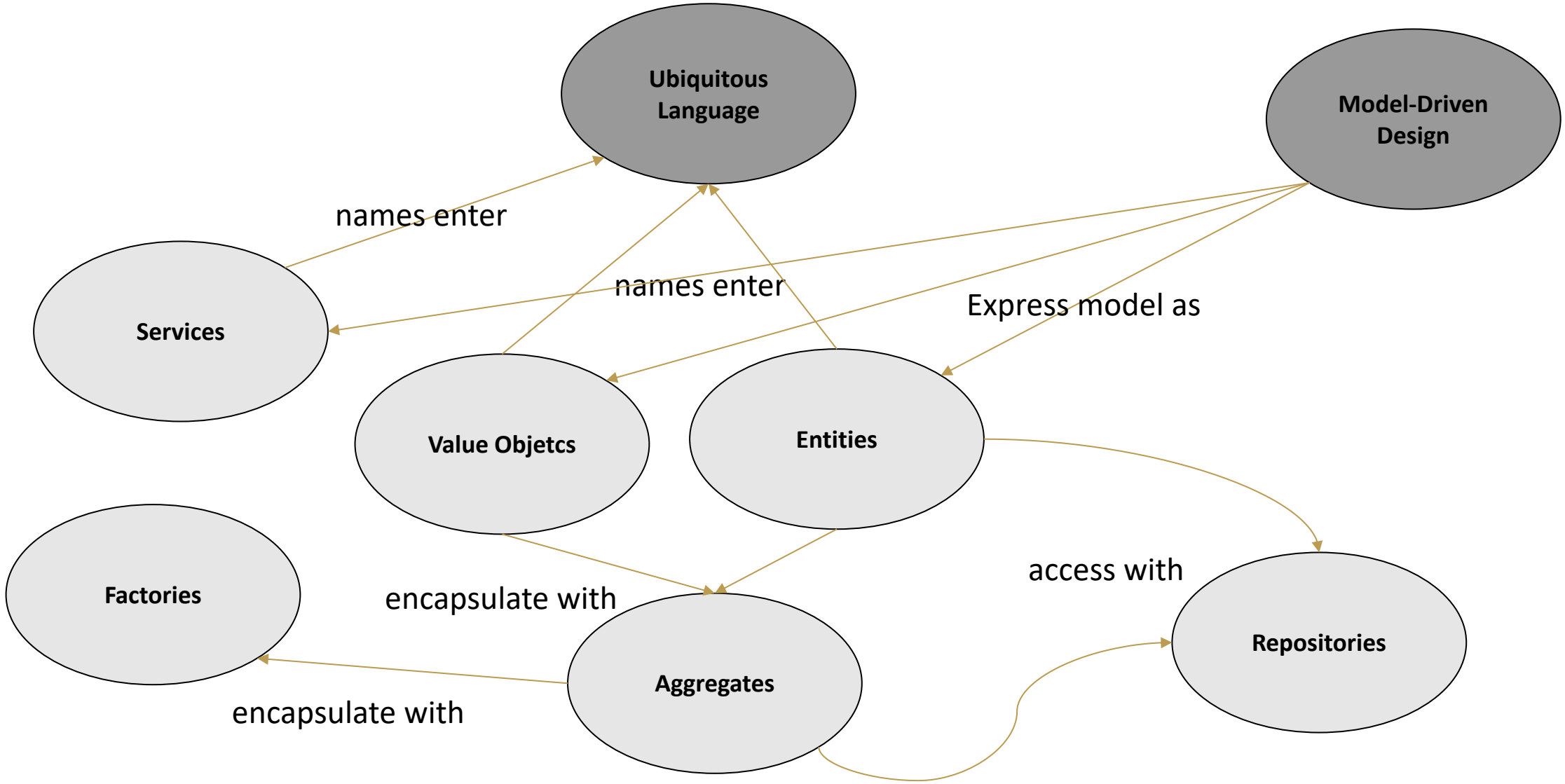
- Mittels ***Context Mapping*** werden zwei Bounded Context (die jeweils ein eigenes Domänenmodel beinhalten) integriert.

<u>Analogie:</u>

- Damit man sich über Länder hinweg verständigen kann und Handel treiben kann, bedarf es einer Übersetzung und Regeln. Diese sind in durch das Context Mapping implementiert.

TAKTISCHES DDD

TAKTISCHE MUSTER IM ÜBERBLICK (AUSSCHNITT)



TAKTISCHE MUSTER IM ÜBERBLICK

Taktisches Design definiert die Details des Domänen Modells:

- ***Value Objects***

A value object is uniquely identified by the data ist holds.

```
class Color:
        RGB: str
```

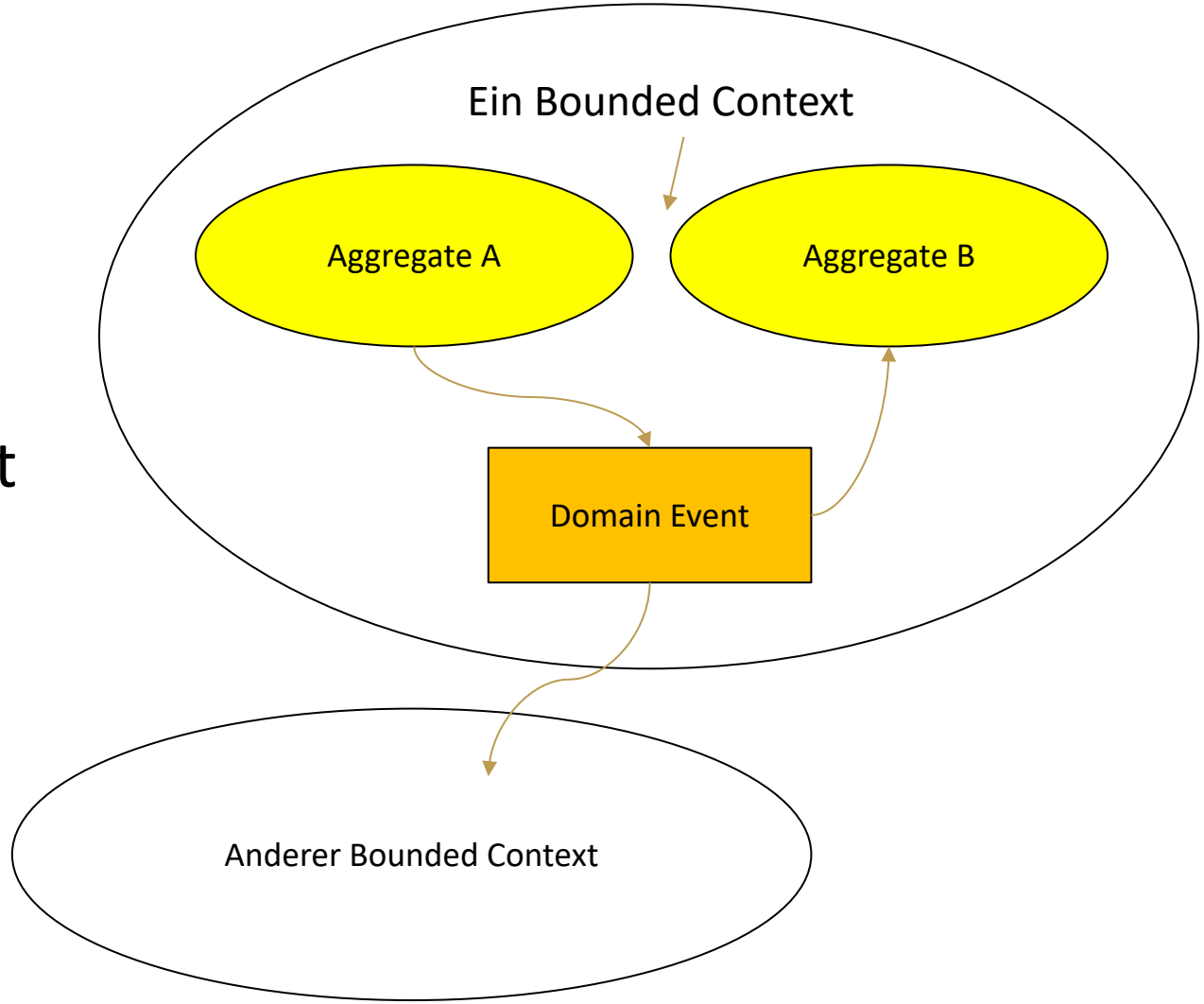
If we change RGB in Color, we have a new color. That's the definition of a value object: any object that is identified only by its data and doesn't have a long-lived identity.

- ***Entities***

We use the term *entity* to describe a domain object that has long-lived identity.
An entity is the opposite of a value object. It requires an explicit identification field.

TAKTISCHE MUSTER IM ÜBERBLICK

- Mit ***Aggregate*** *werden* ***Entities*** und ***Value Objects*** zusammengefasst.
- Ein ganz zentrales Konzept sind ***Domain Events***. Es wird in Ereignissen gedacht. Sie dienen auch als Kommunikationsmittel. Mit Domain Events werden wichtige Ereignisse festgehalten und andere Bounded Context darüber informiert.



ZUSAMMENFASSUNG

- DDD ermöglicht ein gemeinsames Verständnis des Problembereichs. In DDD wird zwischen strategischem und taktischem DDD unterschieden.
- Es gibt komplexe und einfache Problembereiche. DDD sollte auf komplexe angewandt werden. Wir unterscheiden in essentielle und akzidentielle Komplexität.
- Im Fouks von DDD stehen Techniken zur schnellen Wissenaufnahme (Knowledge Crunching)
- DDD hilft Problembereiche zu strukturieren (Subdomain, Bounded Context). Die richtige Strukturierung hilft die Komplexität beherrschbar zu machen.
- Der Problembercih wird durch ein Modell repräsentiert. Jedes Modell wird in der Sprache des Fachbereich beschrieben. Mittels BC werden Modelle voneinander abgegrenzt.

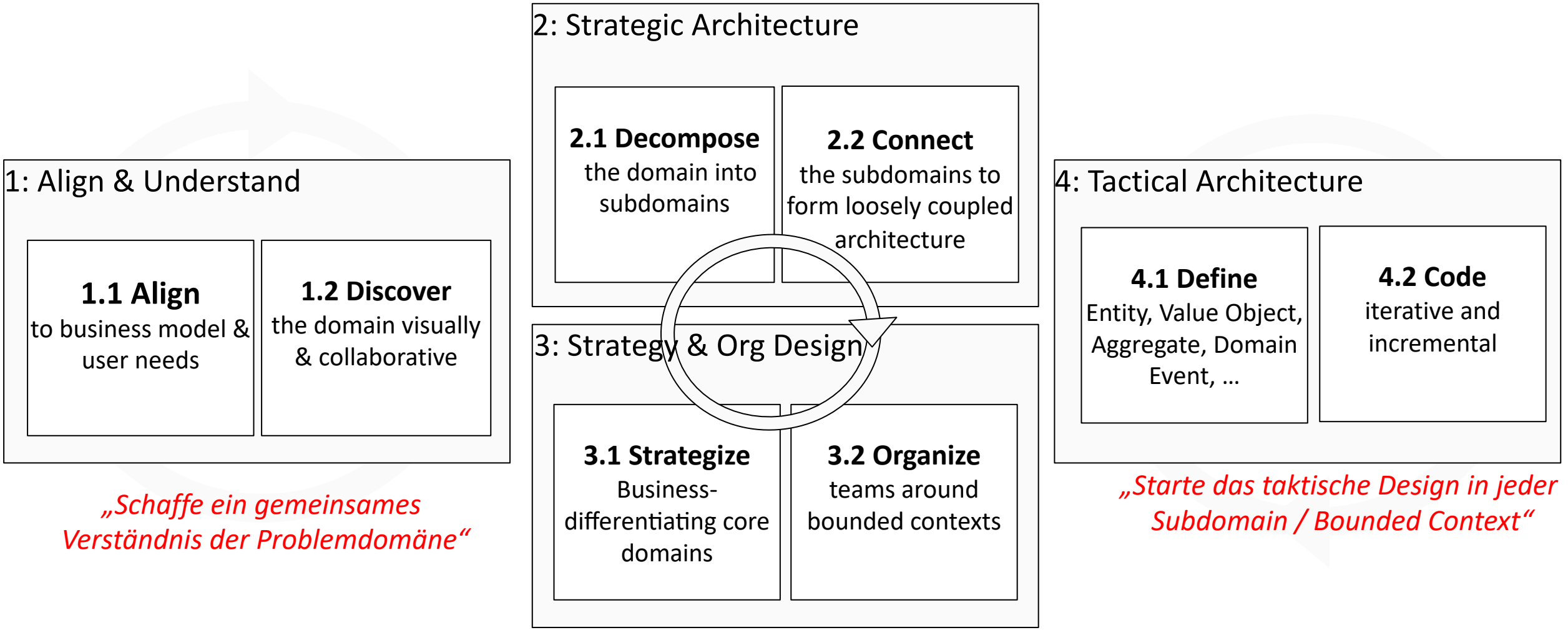
DDD EINGEBETTET IN EINEN SOZIO-TECHNISCHEN DESIGN-PROZESS

Quellen:
https://github.com/ddd-crew
https://github.com/ddd-crew/ddd-starter-modelling-process

SOCIO-TECHNICAL ARCHITECTURE DESIGN

Knowledge Foundation
@ Reutlingen University

„Erzeuge Struktur und verstehe die Abhängigkeiten: teile und herrsche“



„Schaffe ein gemeinsames Verständnis der Problemdomäne“

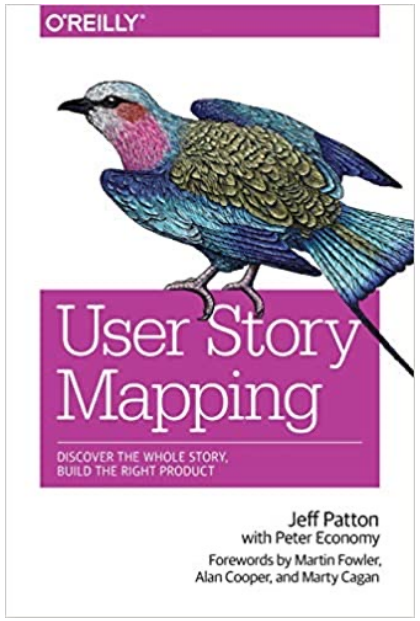
„Treffe strategische und organisationale Entscheidungen“

„Starte das taktische Design in jeder Subdomain / Bounded Context“

FURTHER READING

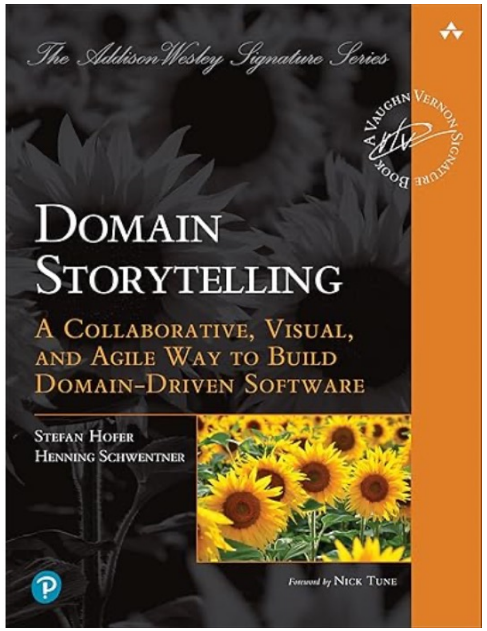
KNOWLEDGE CRUNCHING LITERATURE

User Story Mapping



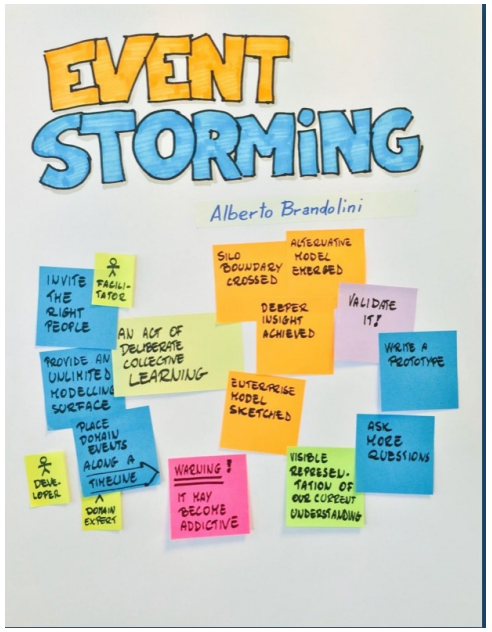
User Story Mapping
Jeff Patton

Domain Story Telling



Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software
Hofer, Schwentner

Event Storming



Event Storming
Alberto Brandolini