Domain Driven Design

in der Praxis

Michael Mirold, 22.10.2019

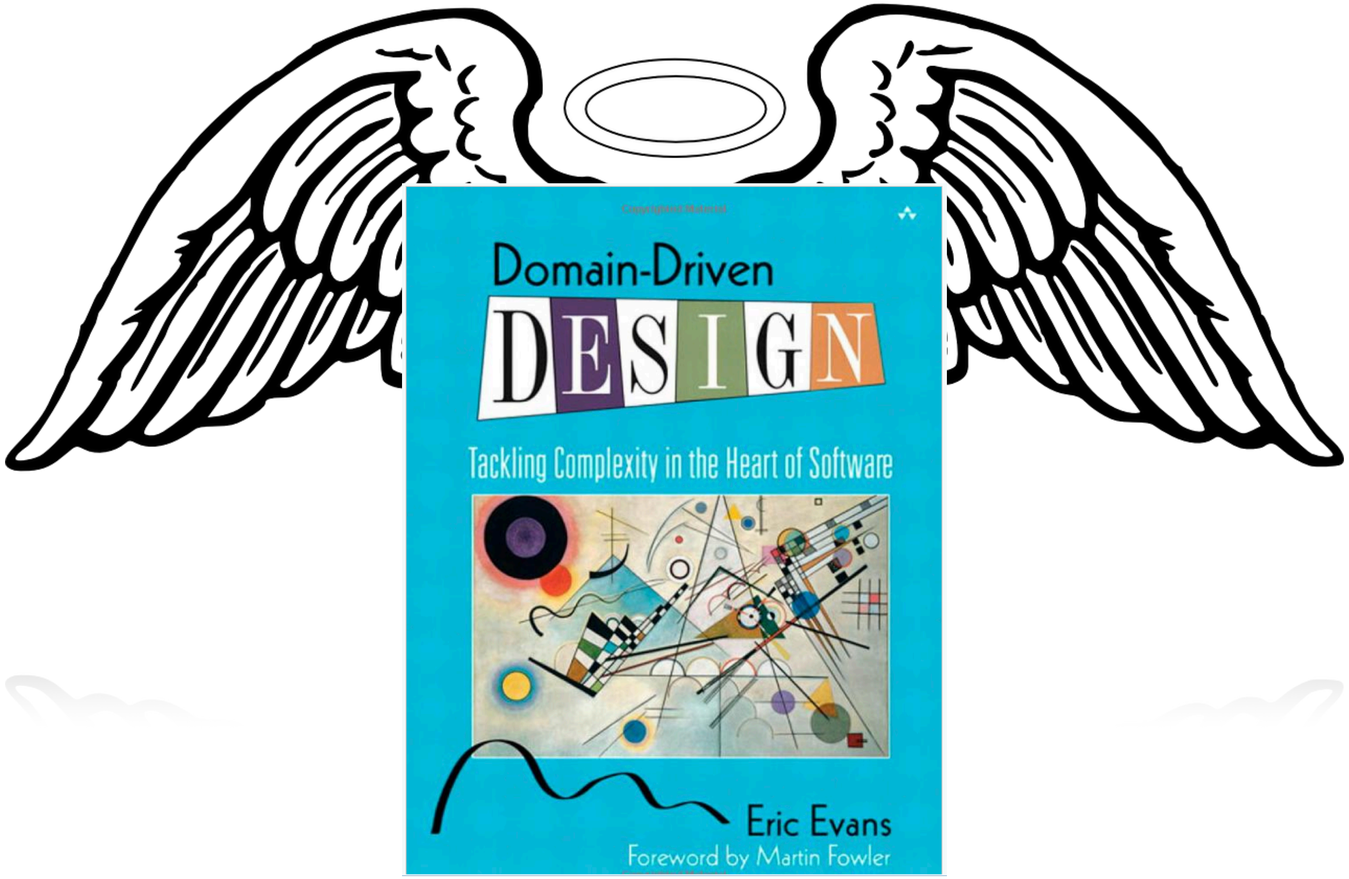
TESTFABRIK

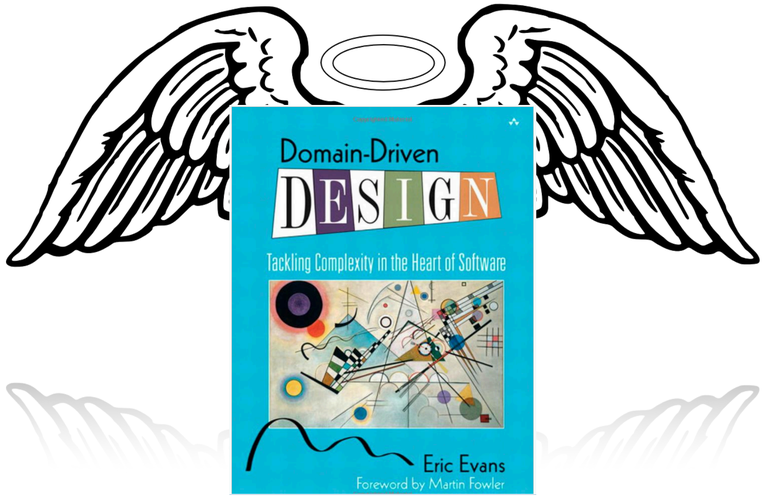
```
> $(".eventList-list").children.length
```

```
> $(".eventList-list").children.length
< 50
```


Adobe Stock


Adobe Stock
STORY TIME


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler



- "erfunden" 2004 durch Eric Evans
- Software-Modellierungs-Ansatz für fachlich komplexe Anwendungen
- etabliert Pattern-Sprache
- große Fan-Gemeinde

DDD Grundideen

Fokus auf
"Core Domain"

Kollaboration von
Fachexperten
und Entwicklern

Etablierung einer
gemeinsamen
Sprache

WAS IST DDD?

DDD

Process & Organization

DDD

Process &
Organization

Architecture &
Frameworks

DDD

Process &
Organization

Architecture &
Frameworks

DDD

Strategic
Design

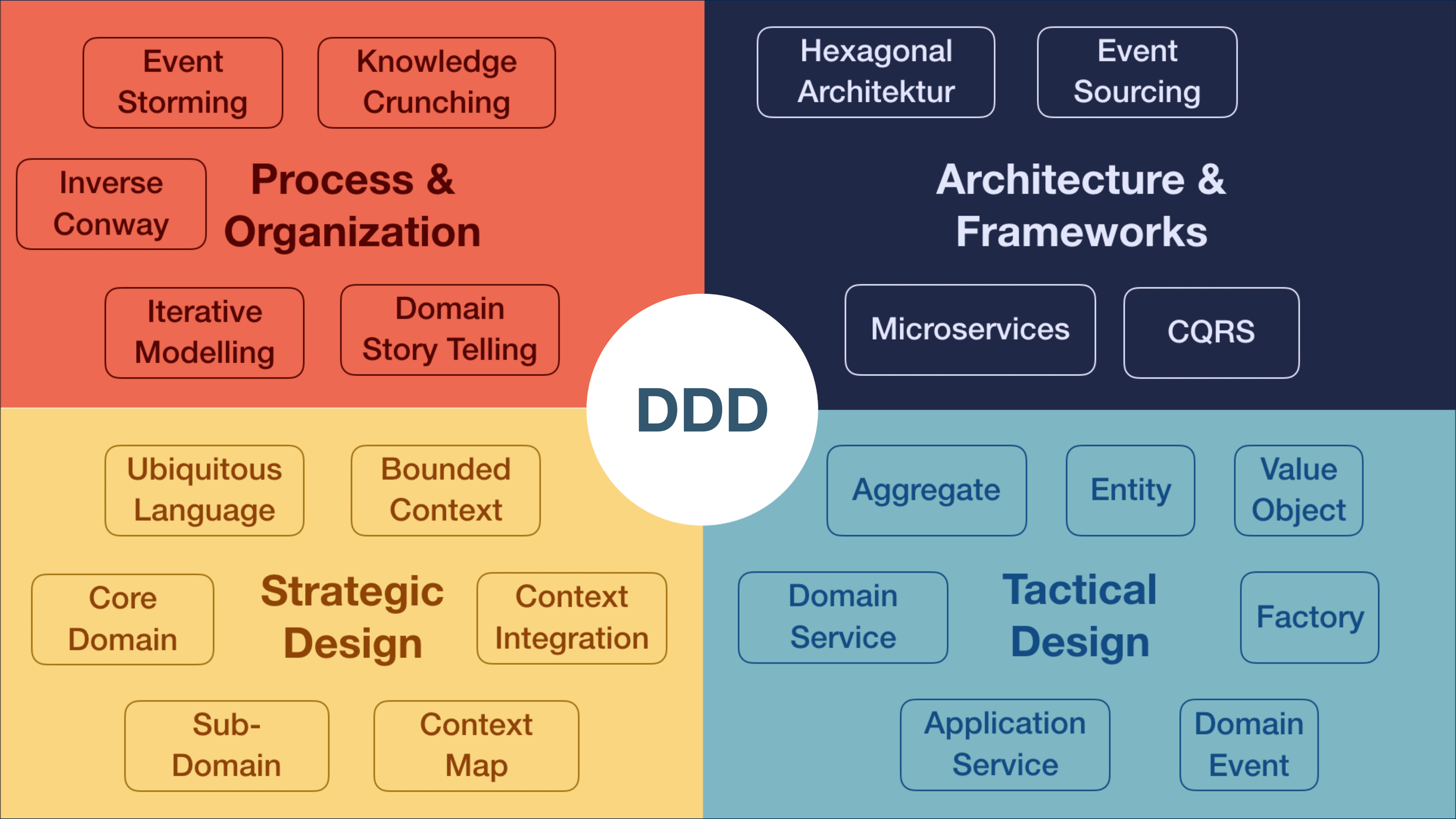
Process &
Organization

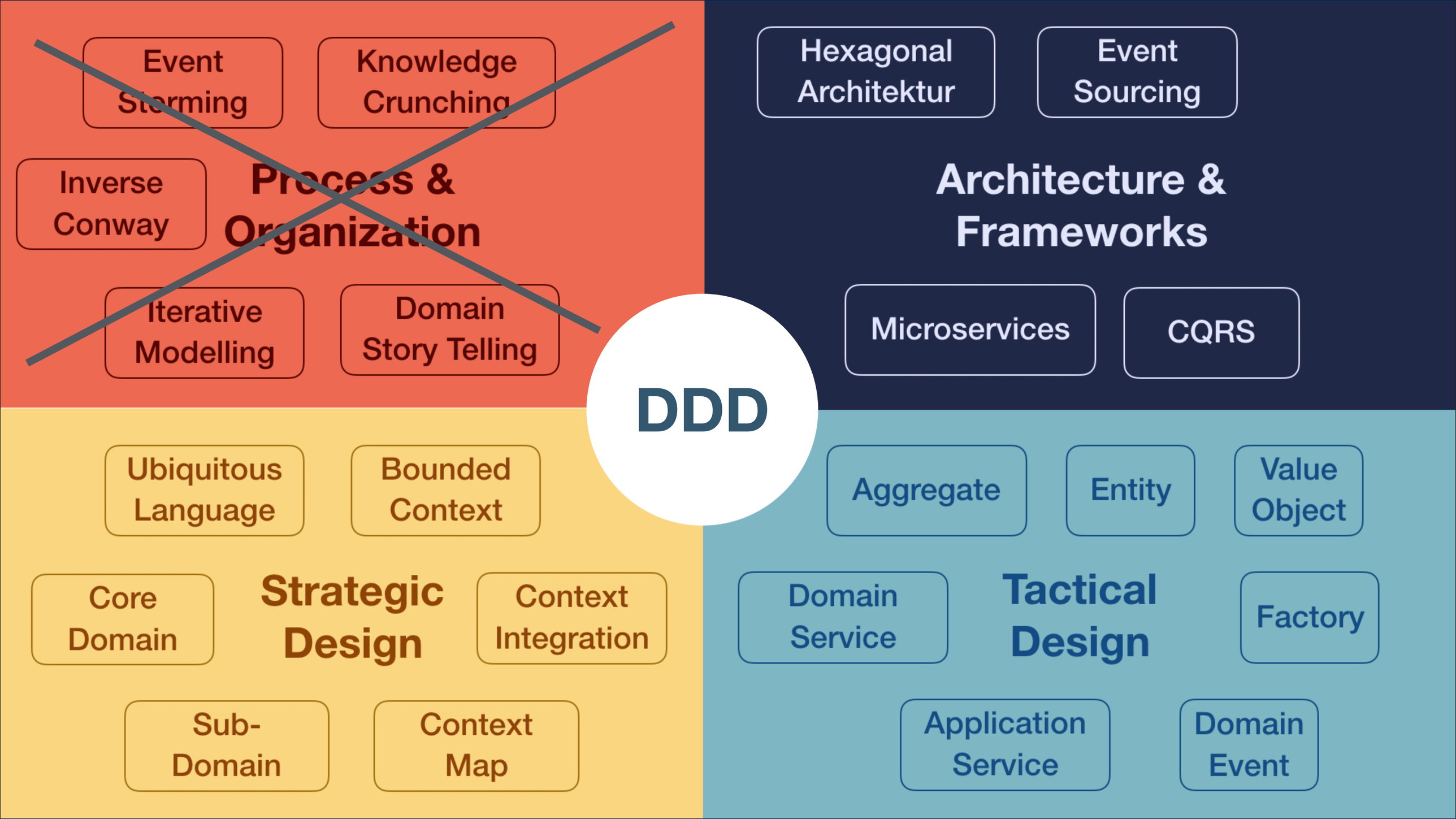
Architecture &
Frameworks

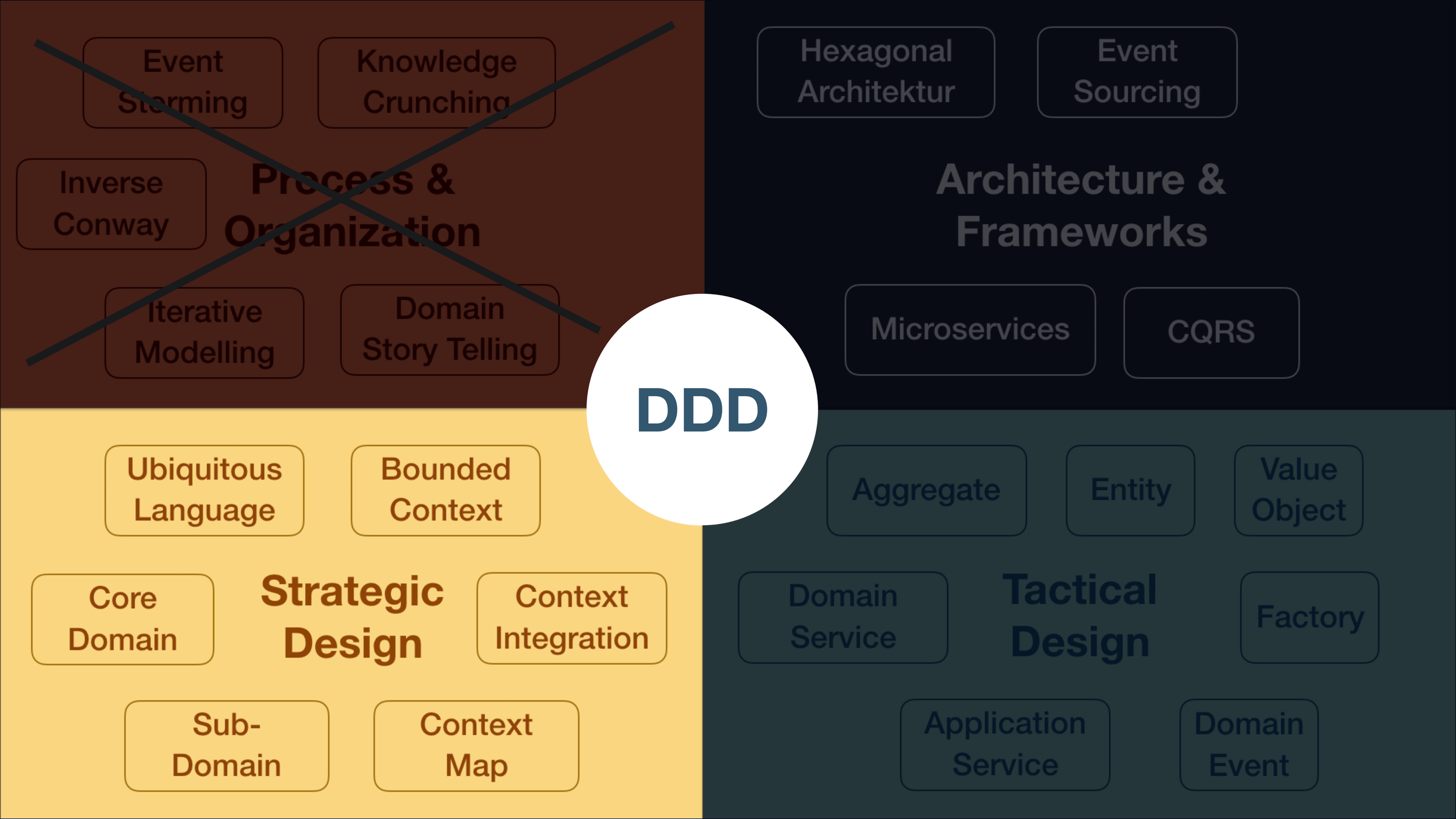
DDD

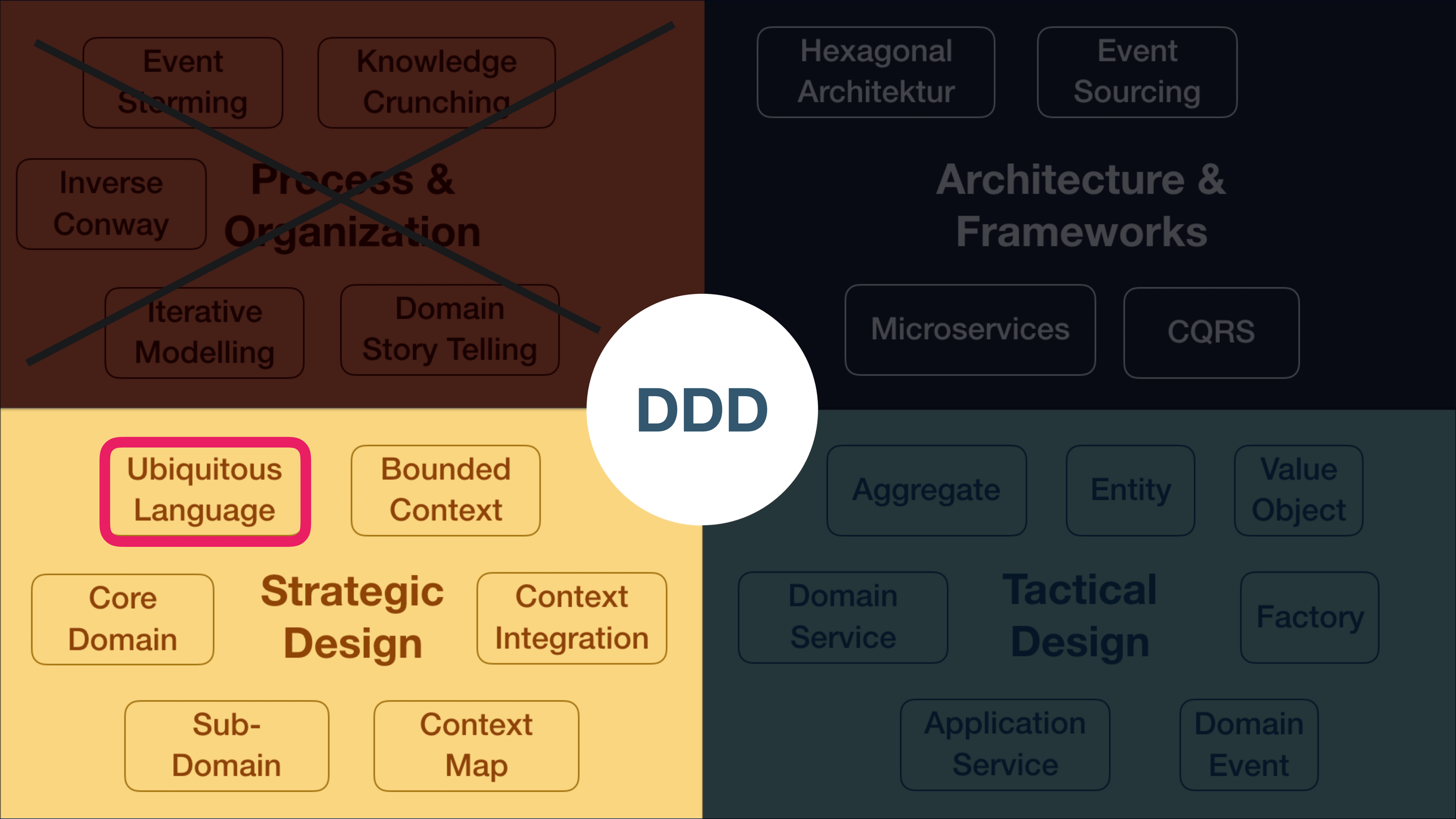
Strategic
Design

Tactical
Design


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

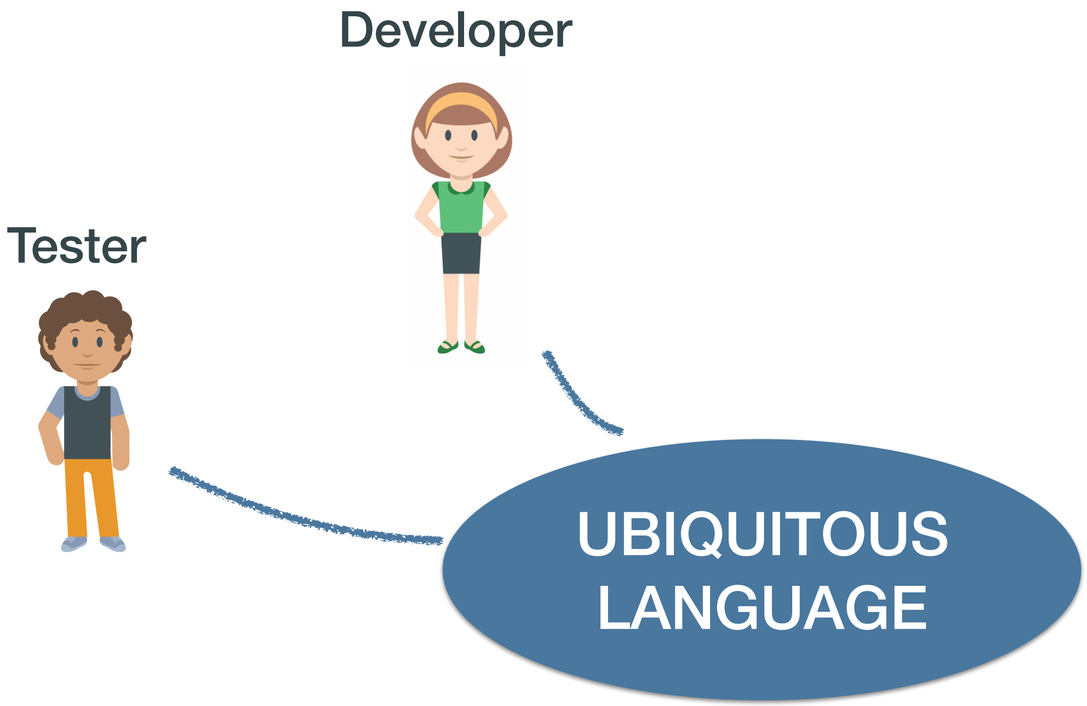
UBIQUITOUS LANGUAGE

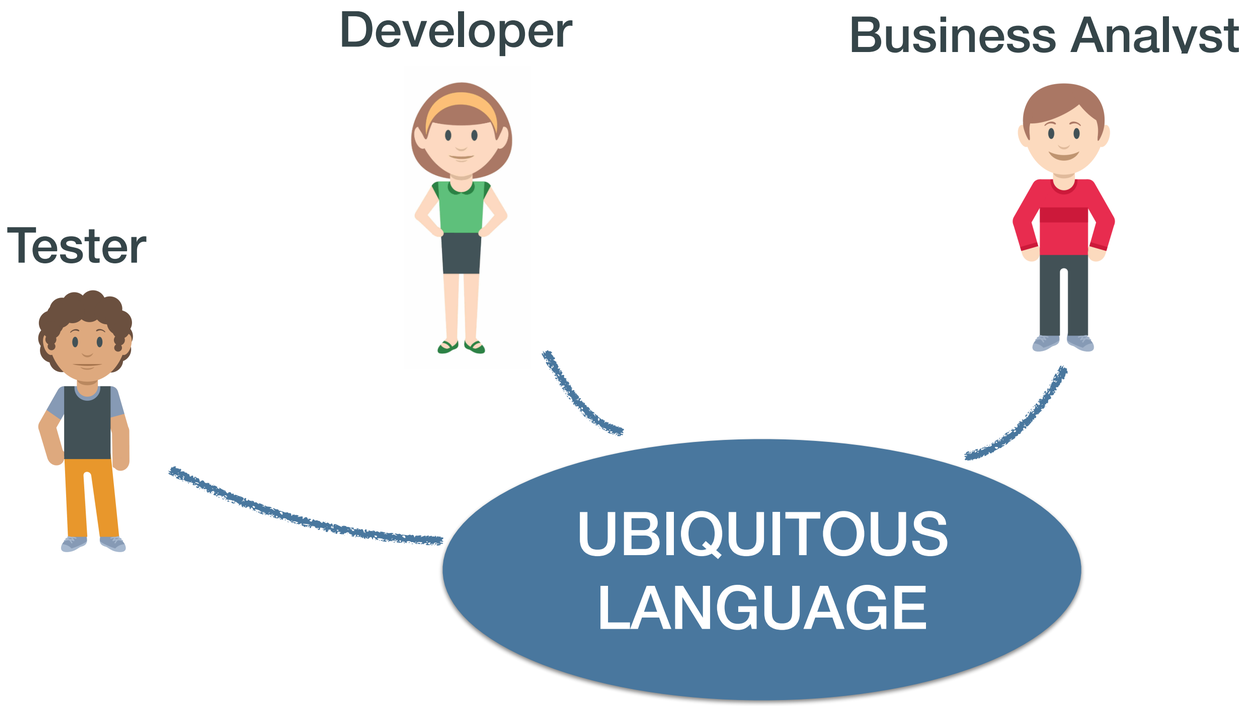
"A language structured around the domain model and used by all team members within a bounded context to connect all the activities of the team with the software."

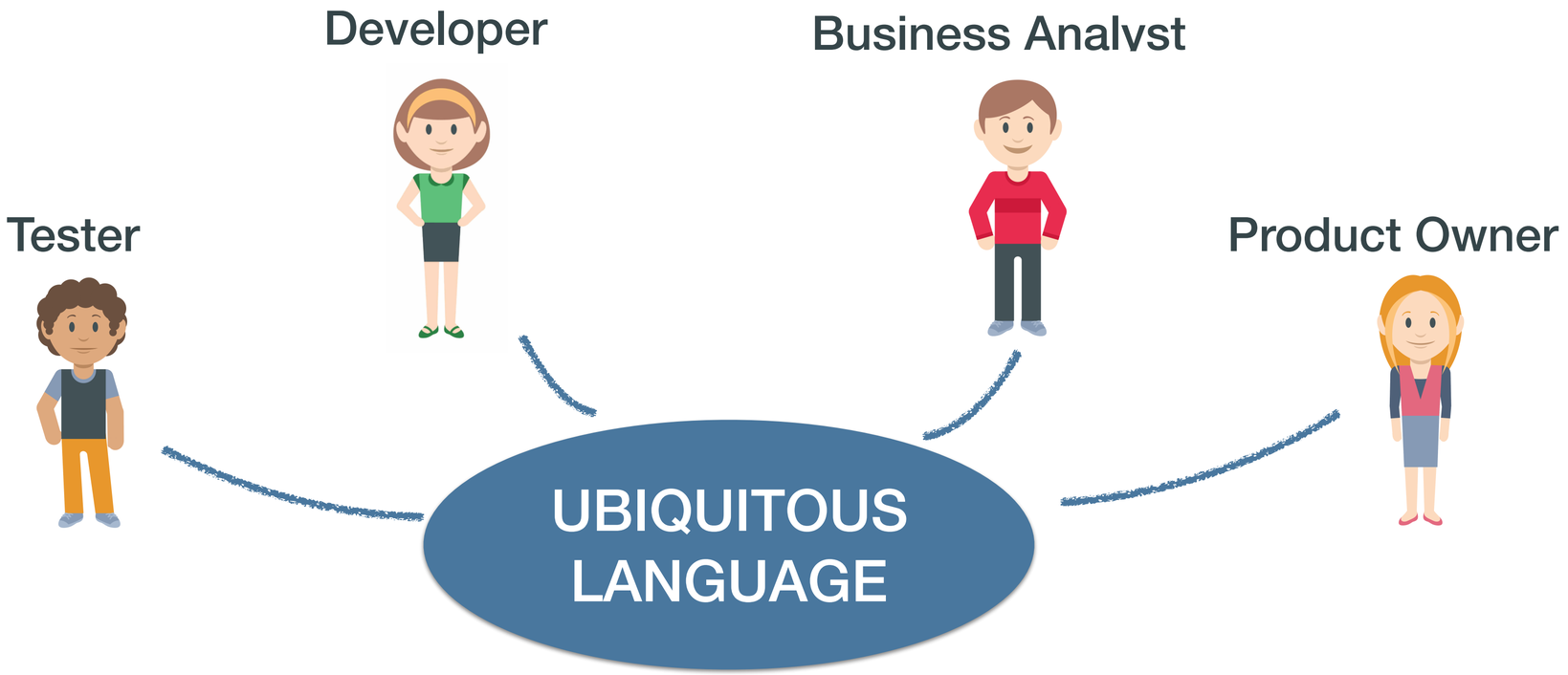

UBIQUITOUS
LANGUAGE

Tester

UBIQUITOUS LANGUAGE


Developer
Tester
UBIQUITOUS
LANGUAGE


Developer
Business Analyst
Tester
UBIQUITOUS
LANGUAGE


Developer
Business Analyst
Tester
Product Owner
UBIQUITOUS
LANGUAGE

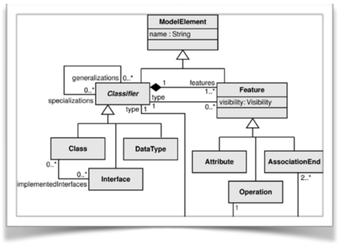
Developer

Business Analyst

Tester

Product Owner

UBIQUITOUS LANGUAGE



Software-Dokumentation

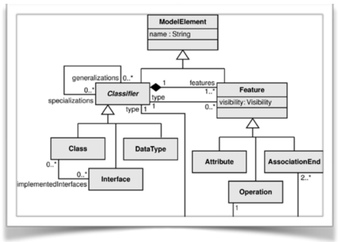
Developer

Business Analyst

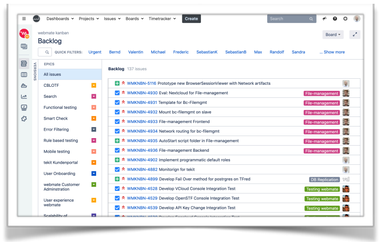
Tester

Product Owner

UBIQUITOUS LANGUAGE



Software-Dokumentation



Requirements

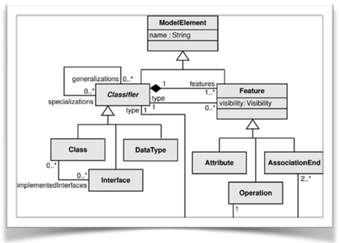
Developer

Business Analyst

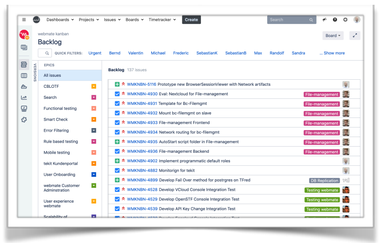
Tester

Product Owner

UBIQUITOUS LANGUAGE



Software-Dokumentation



Requirements

Team-Kommunikation

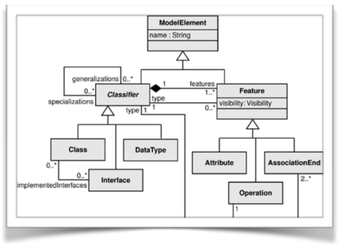
Developer

Business Analyst

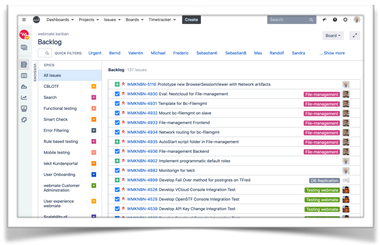
Tester

Product Owner

UBIQUITOUS LANGUAGE

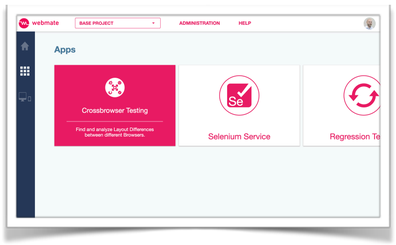


Software-Dokumentation



Requirements

Team-Kommunikation



User Interfaces + Dokumentation

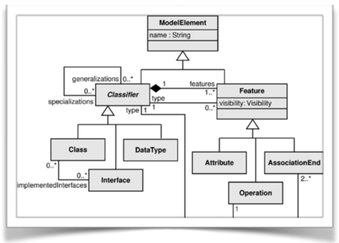
Developer

Business Analyst

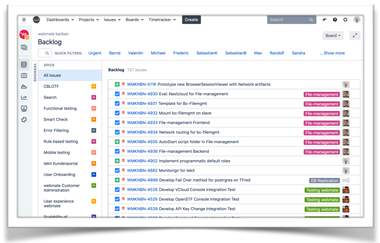
Tester

Product Owner

UBIQUITOUS LANGUAGE

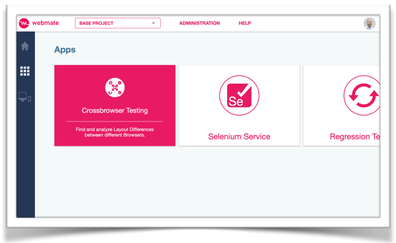


Software-Dokumentation

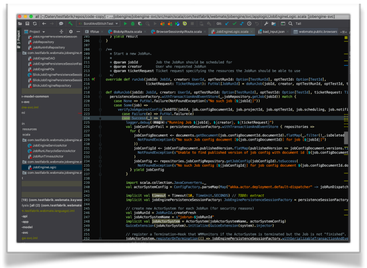


Requirements

Team-Kommunikation



User Interfaces + Dokumentation



Source-Code

!!!

IDEEN FÜR DIE PRAXIS

Aufbau eines Glossars

Vokabular für User Stories

Verwendung im Source-Code

- Möglichst eindeutige Begriffsdefinitionen
- Beziehungen zu anderen Begriffen erläutern
- nicht nur Substantive, auch Verhalten und Eigenschaften
- wie CRC-Karten

IDEEN FÜR DIE PRAXIS

Aufbau eines Glossars

Vokabular für User Stories

Verwendung im Source-Code

- **Schon bei Anforderungsaufnahme "neue Sprache" verwenden**
- **Aufkommen anderer Begriffe hinterfragen und diskutieren**

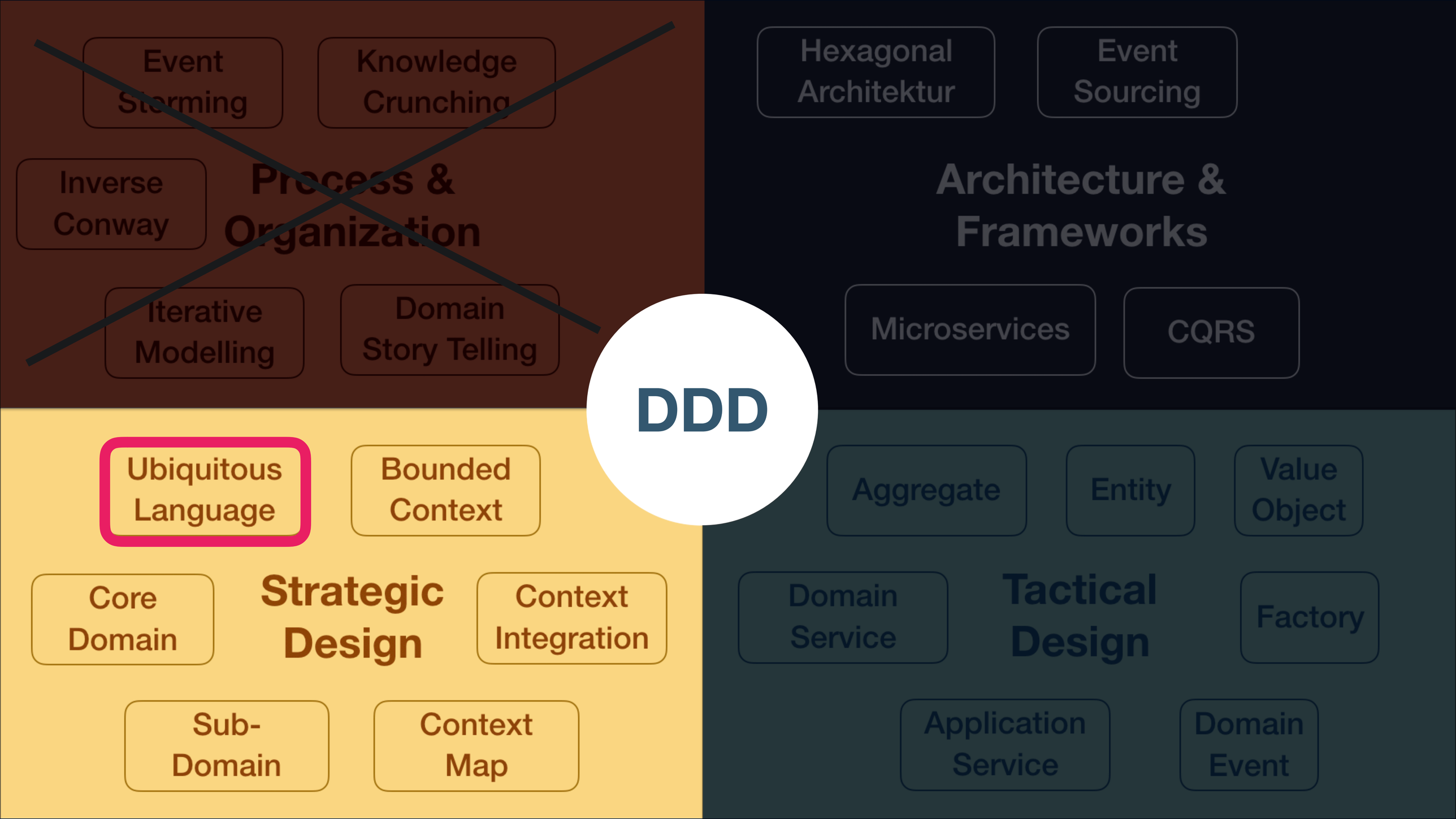
IDEEN FÜR DIE PRAXIS

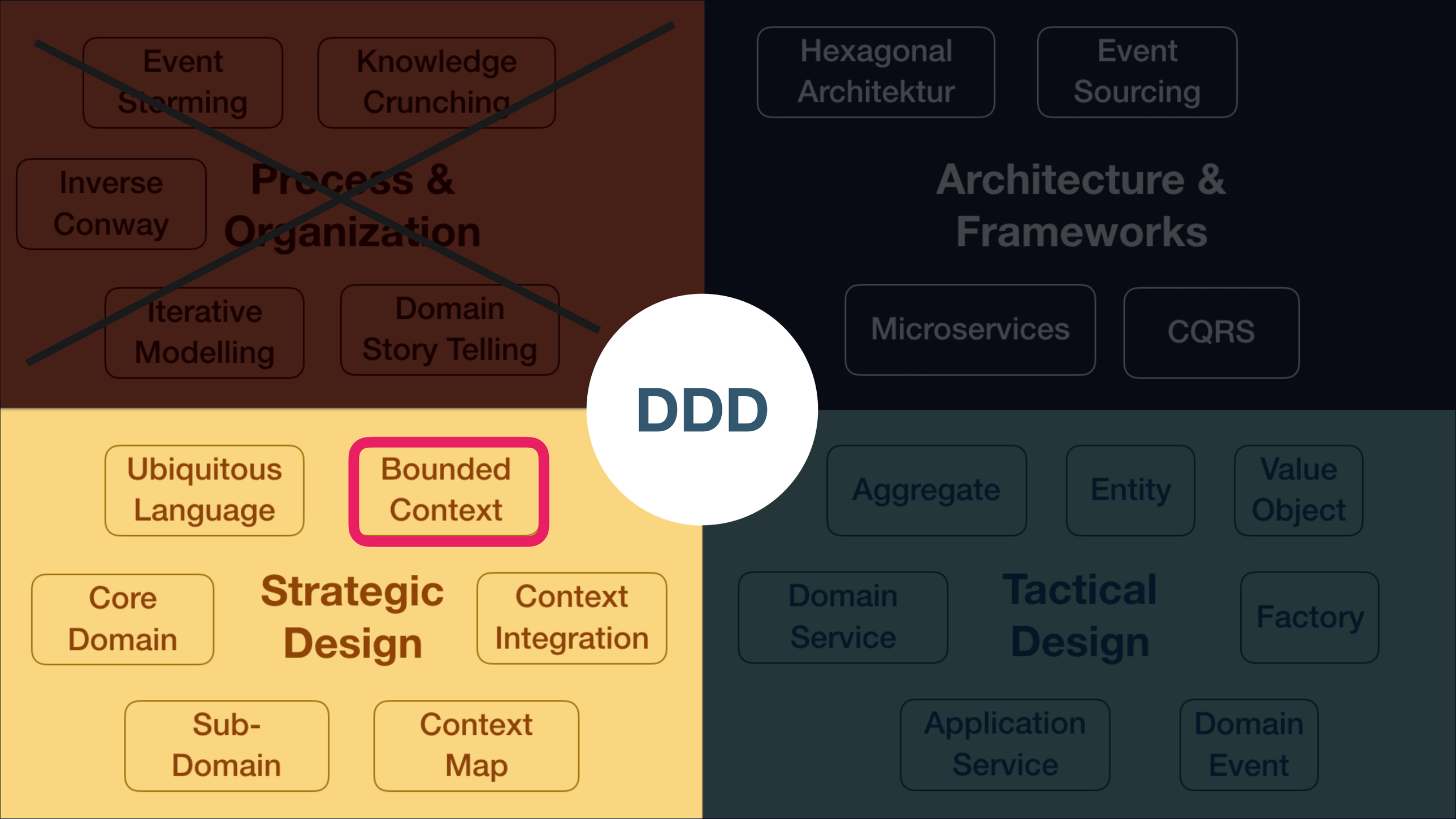
Aufbau eines Glossars

Vokabular für User Stories

Verwendung im Source-Code

- ▸ Verwendung in
  - Variablen- und Methodennamen
  - Klassennamen
  - APIs
  - CSS-Klassen und HTML-Ids
- ▸ Kontrolle in Code-Reviews


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

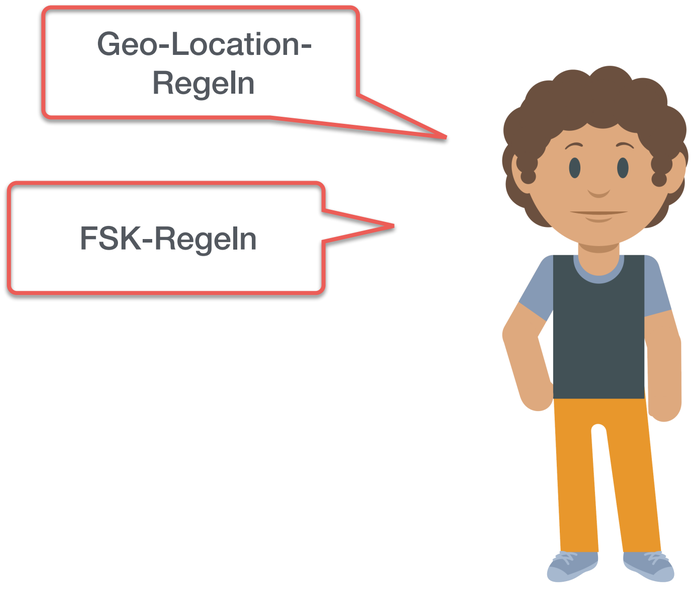
WAS IST EIN STREAM?

Streaming-Management

WAS IST EIN STREAM?

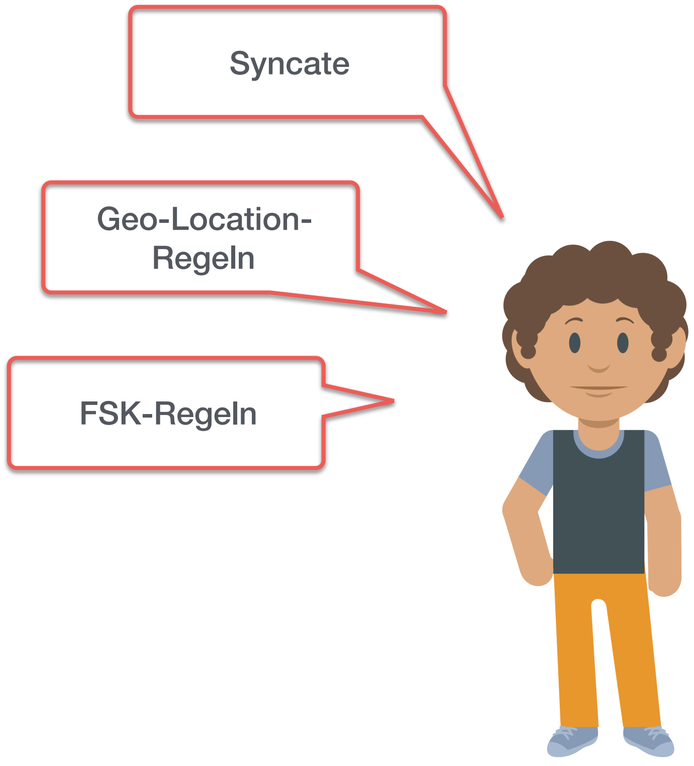
Streaming-Management

WAS IST EIN STREAM?



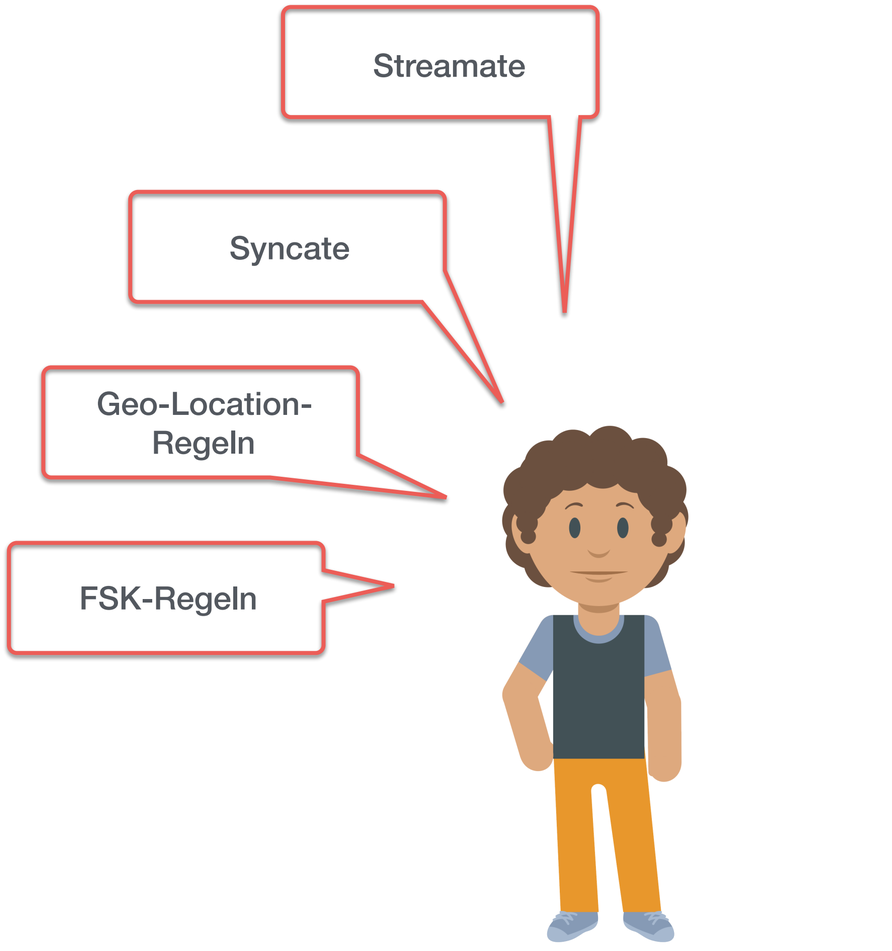
Streaming-Management

WAS IST EIN STREAM?



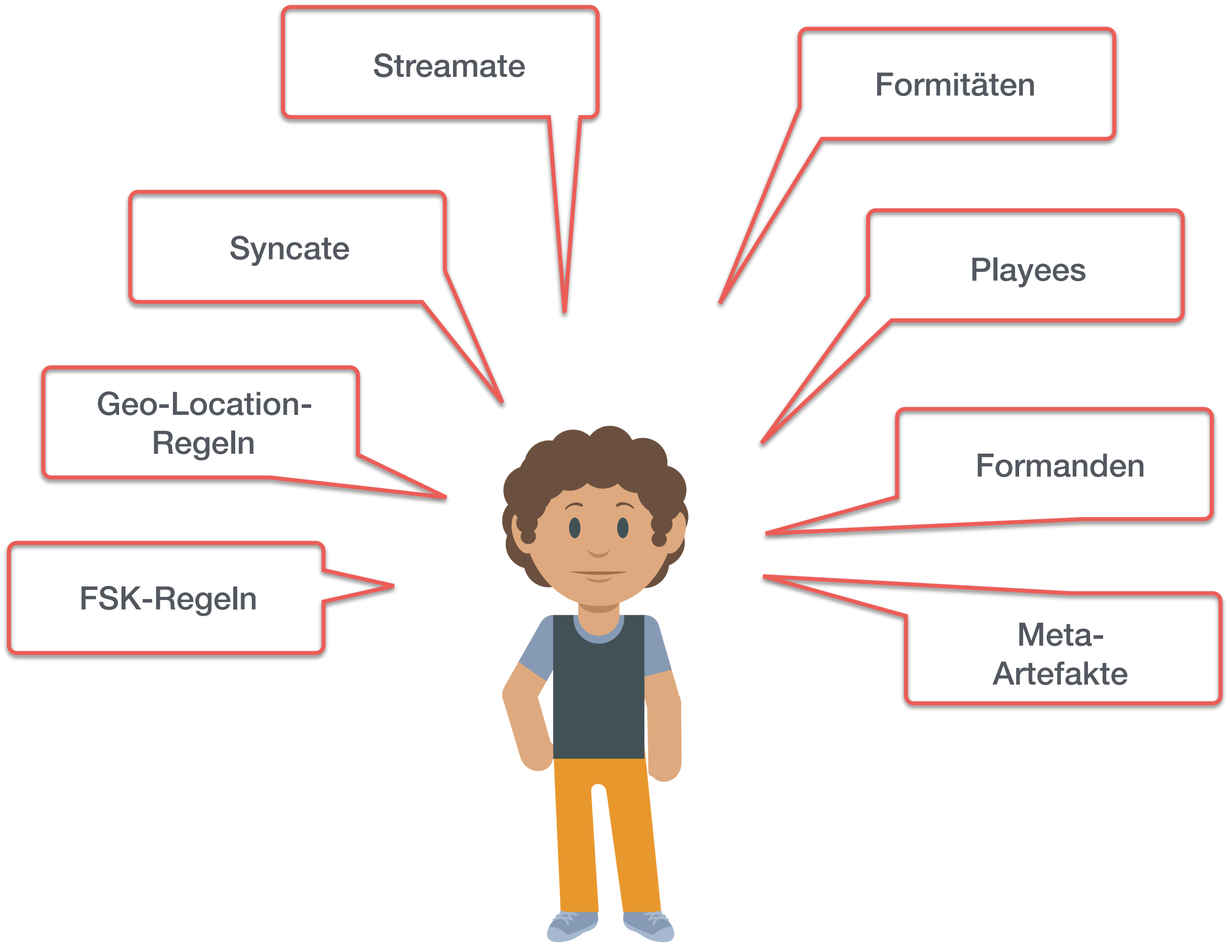
Streaming-Management

WAS IST EIN STREAM?



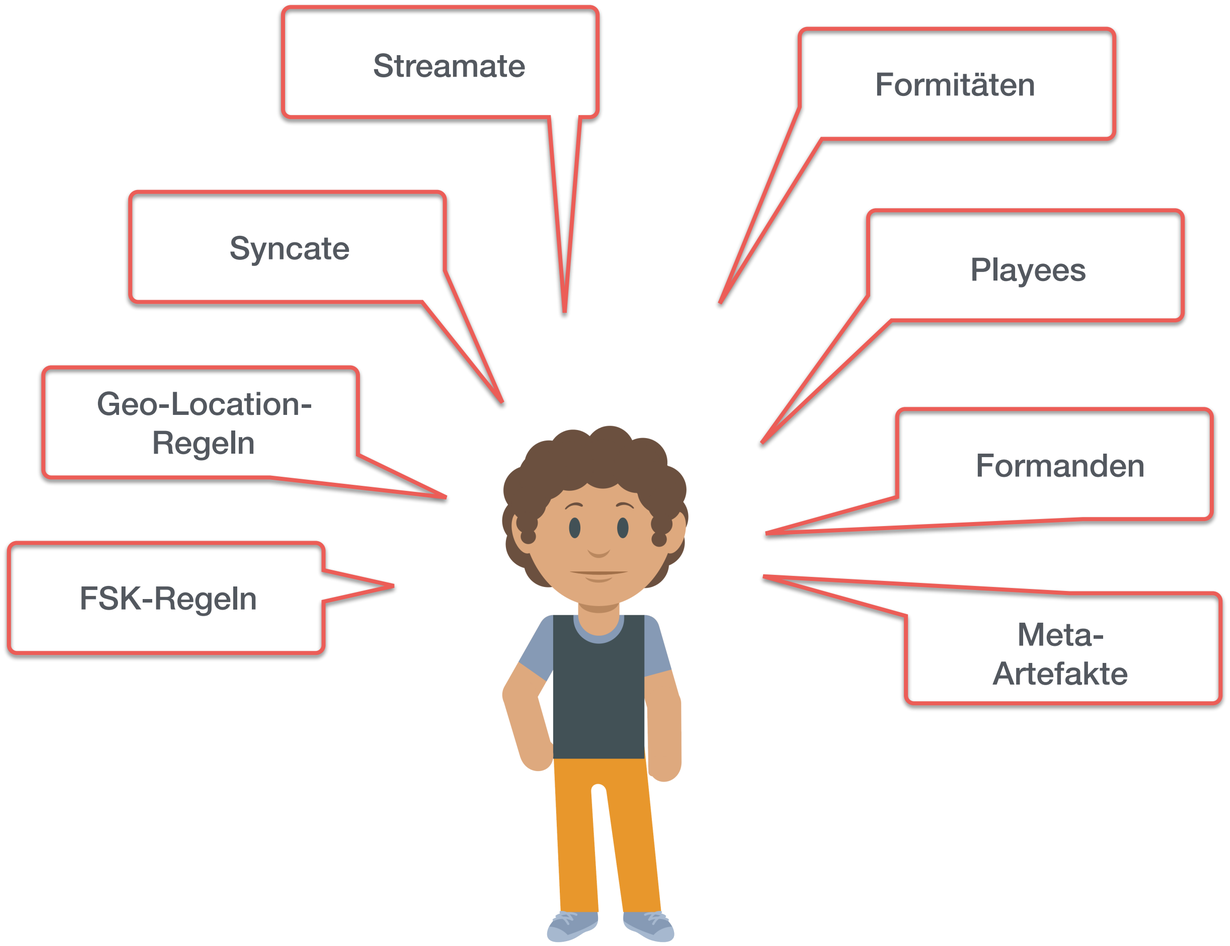
Streaming-Management

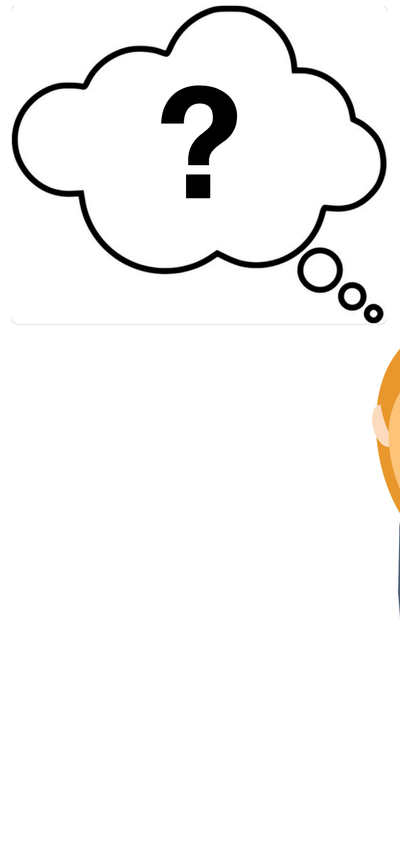
WAS IST EIN STREAM?



Streaming-Management

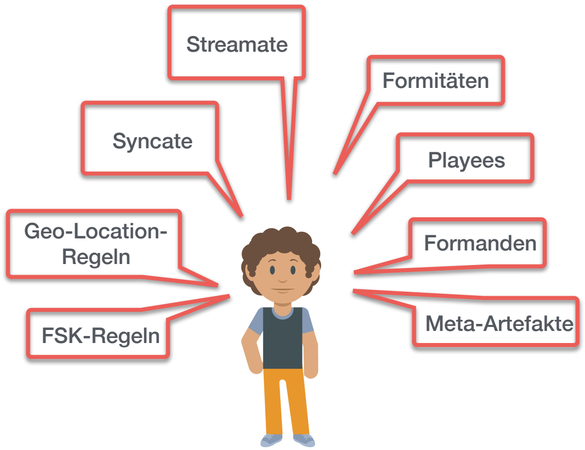
WAS IST EIN STREAM?





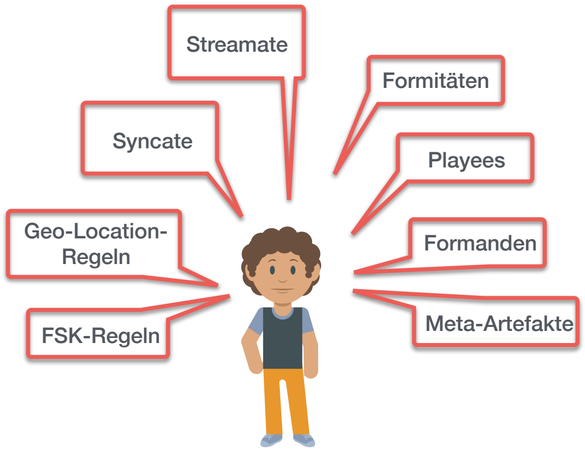
Streaming-Management

WAS IST EIN STREAM?

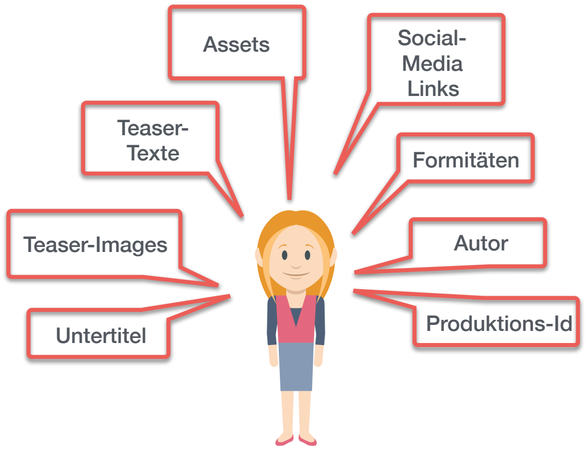


Streaming-Management

WAS IST EIN STREAM?

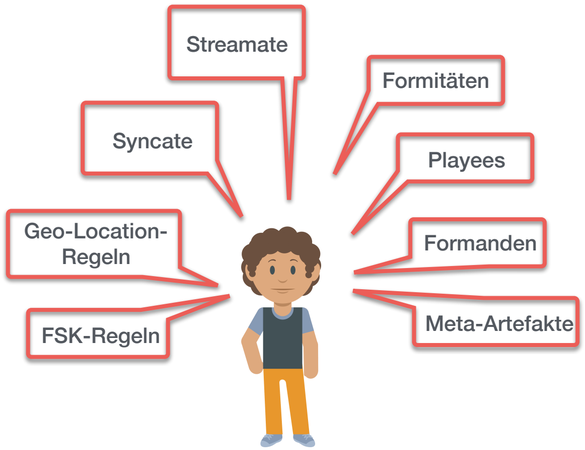


Streaming-Management

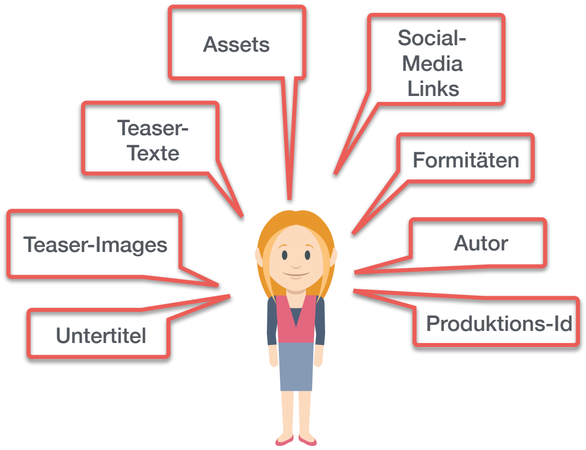


Mediathek CMS

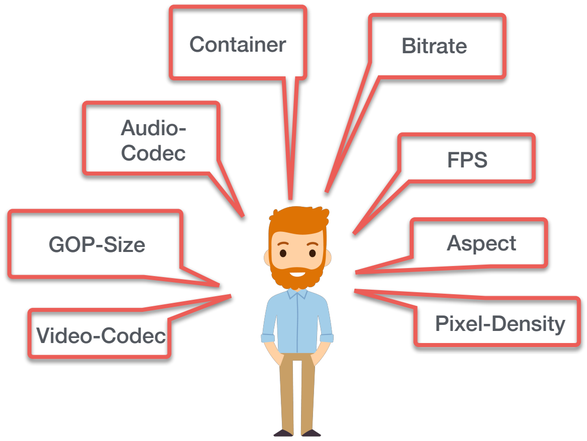
WAS IST EIN STREAM?



Streaming-Management



Mediathek CMS

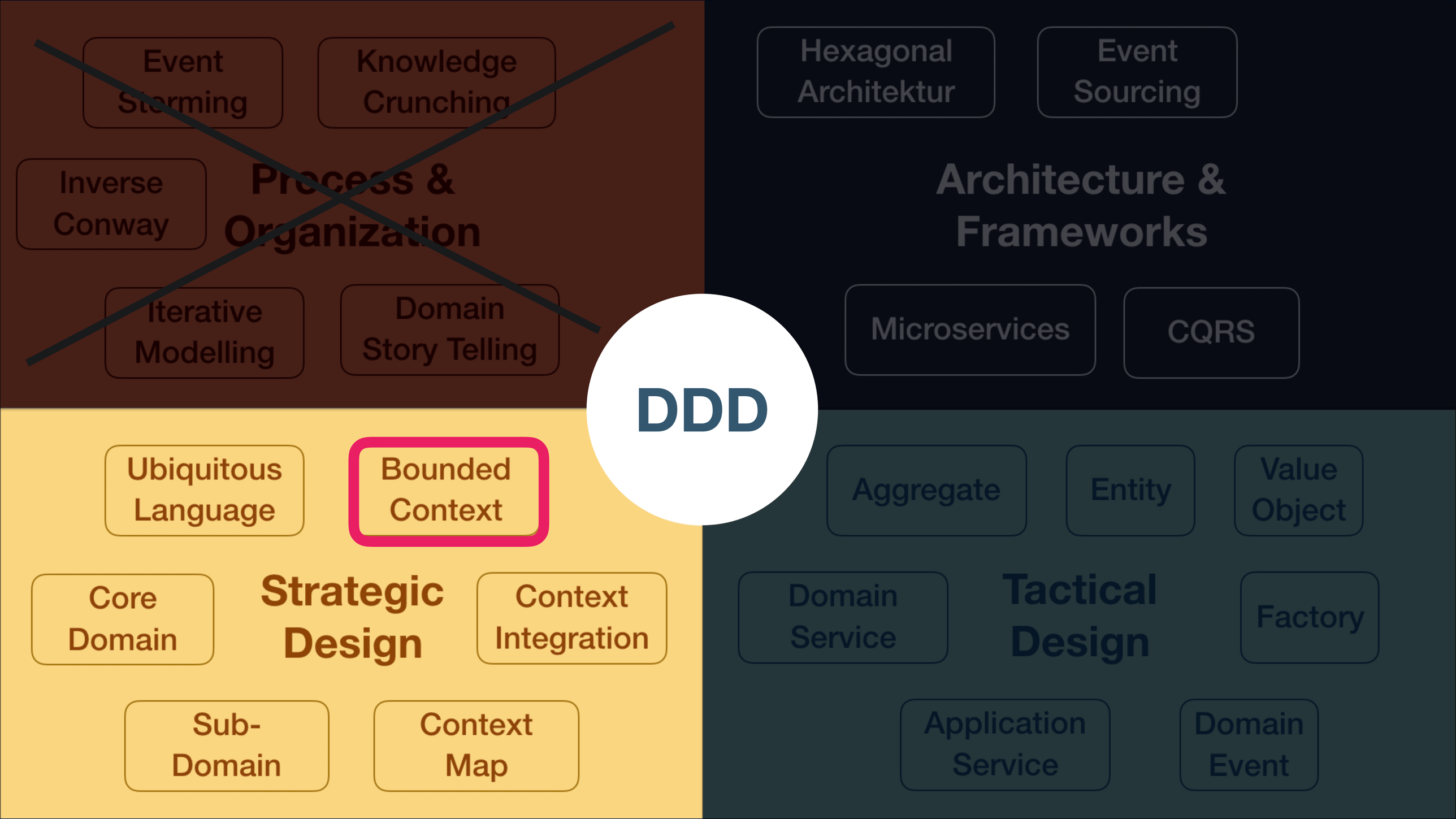


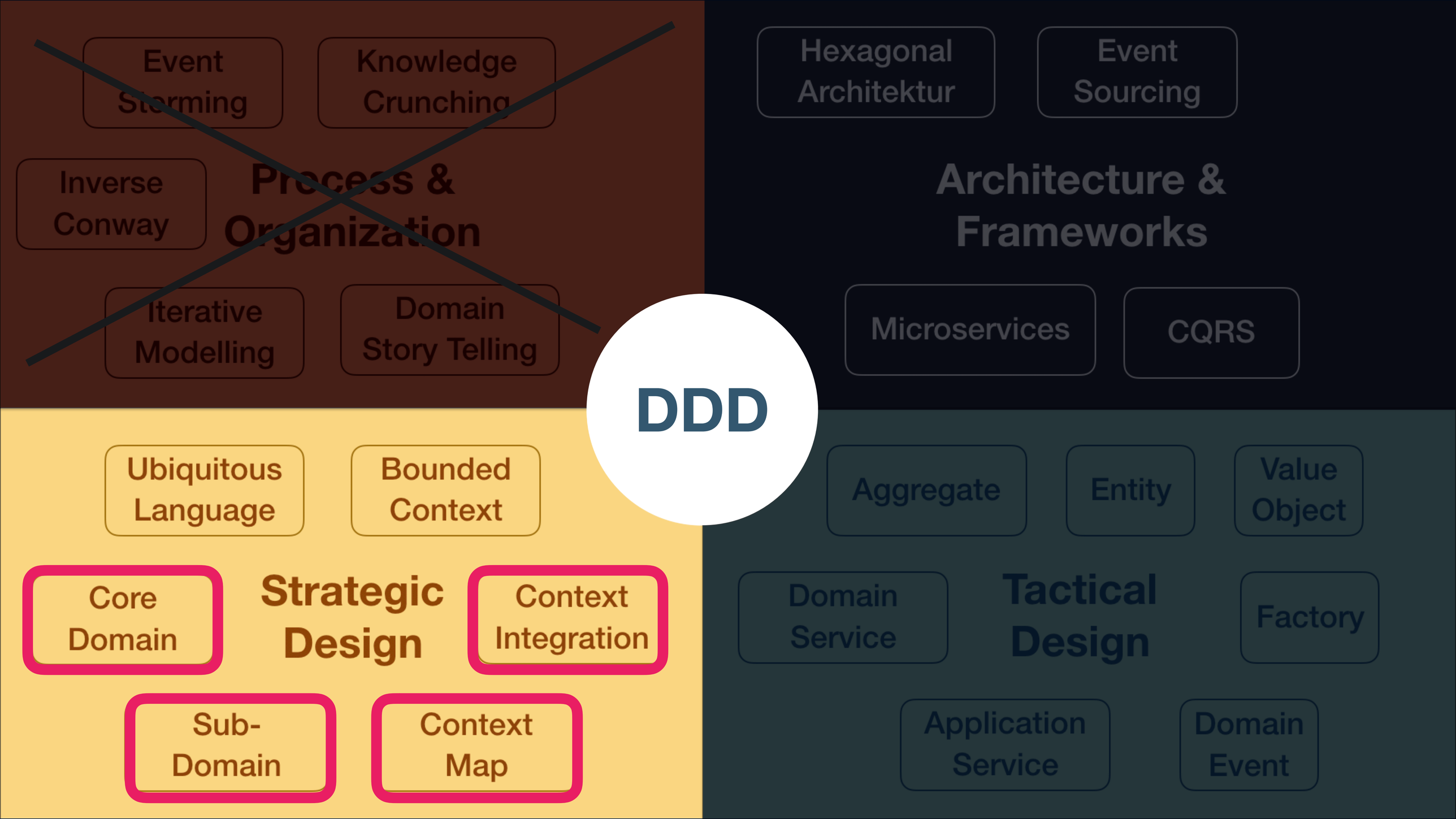
Encoding Cluster

WAS IST EIN STREAM?

BOUNDED CONTEXT

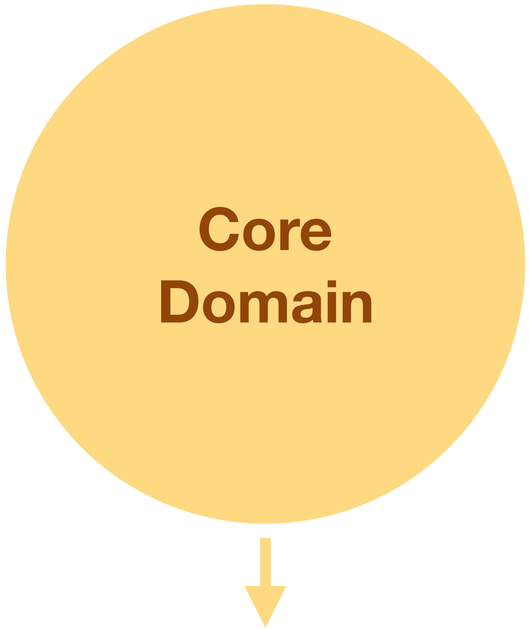
"A description of a boundary (typically a subsystem, or the work of a particular team) within which a particular model is defined and applicable."


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

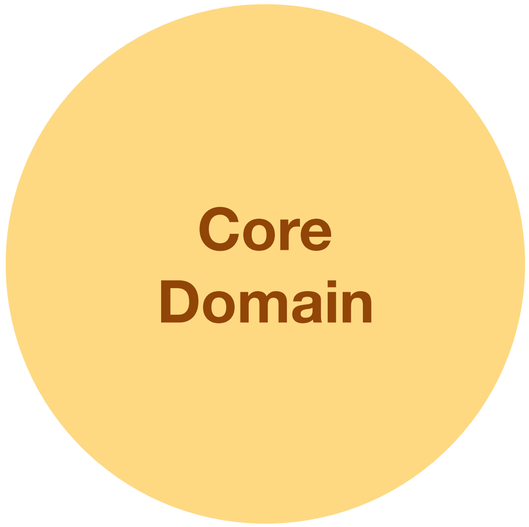
STRATEGIC DESIGN - SUB DOMAINS

Der Teil des Unternehmens,
um den sich alles dreht



**Volle DDD-Power,
volle "Exzellenz"**

Der Teil des Unternehmens,
um den sich alles dreht

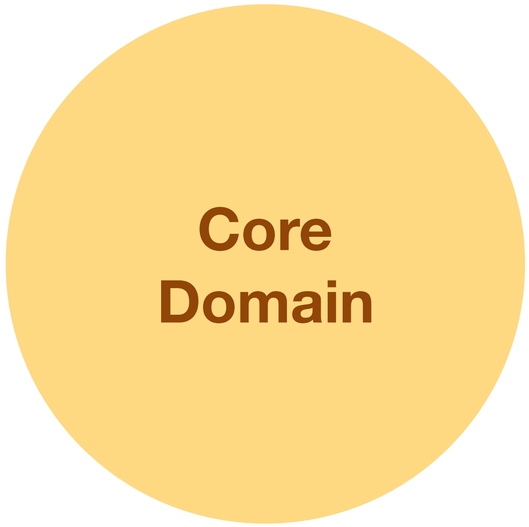


**Volle DDD-Power,
volle "Exzellenz"**

Weniger wichtig als
Core Domain

**"Muss funktionieren"
(egal wie)**

Der Teil des Unternehmens,
um den sich alles dreht

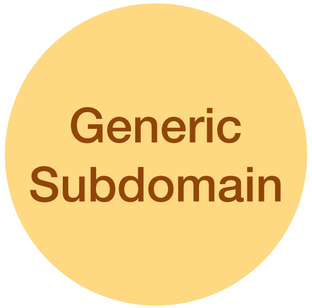


**Volle DDD-Power,
volle "Exzellenz"**

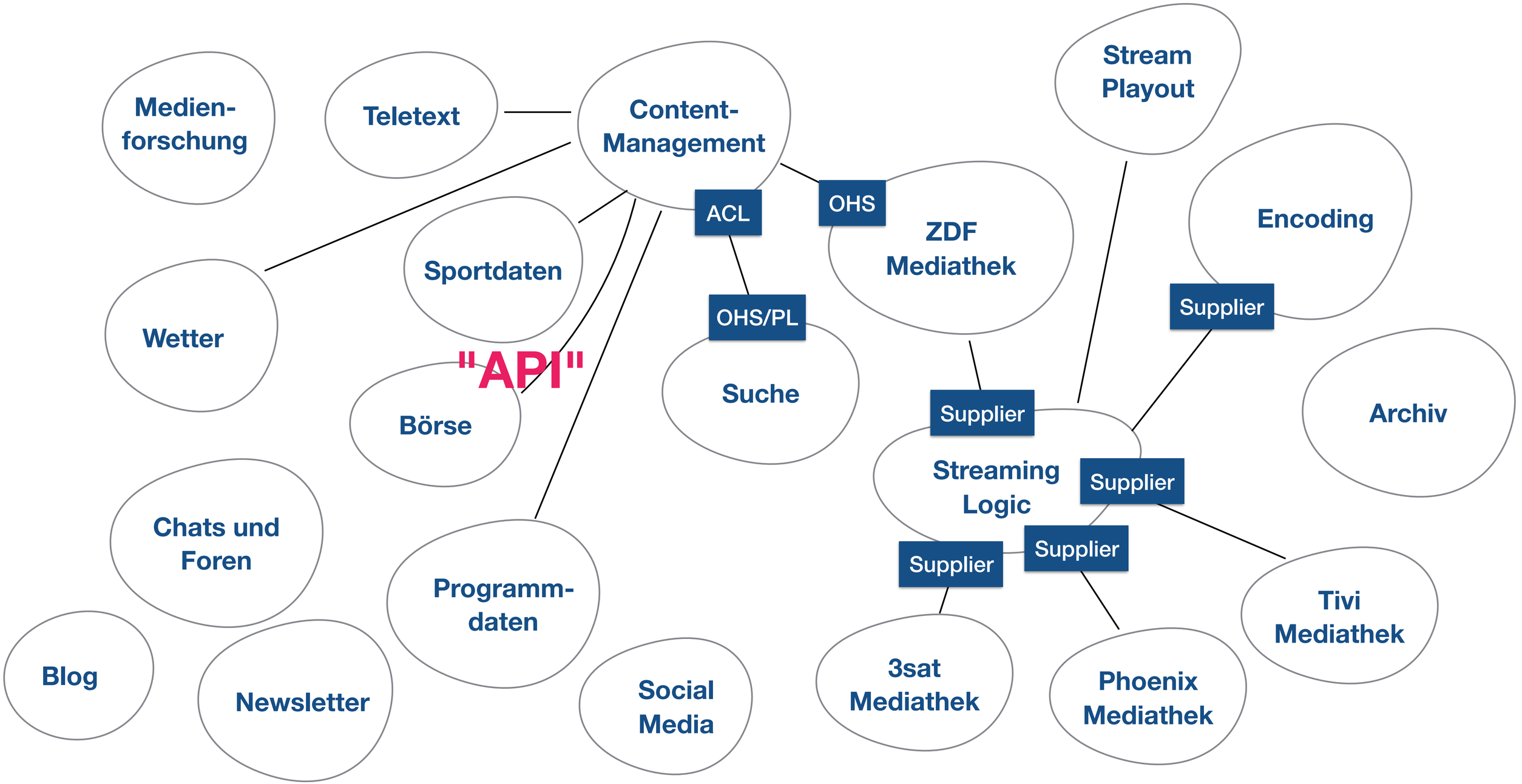
Weniger wichtig als
Core Domain

**"Muss funktionieren"
(egal wie)**

Generisch, wird
aber gebraucht



**Wenn möglich,
zukaufen**


Medien-
forschung
Teletext
Content-
Management
Stream
Playout
ACL
OHS
ZDF
Mediathek
Encoding
Sportdaten
Wetter
OHS/PL
Supplier
"API"
Börse
Suche
Archiv
Supplier
Streaming
Logic
Supplier
Chats und
Foren
Supplier
Supplier
Programm-
daten
Tivi
Mediathek
Blog
Newsletter
Social
Media
3sat
Mediathek
Phoenix
Mediathek

IDEEN FÜR DIE PRAXIS

Pragmatisch bleiben

Nicht zu früh an Technik denken

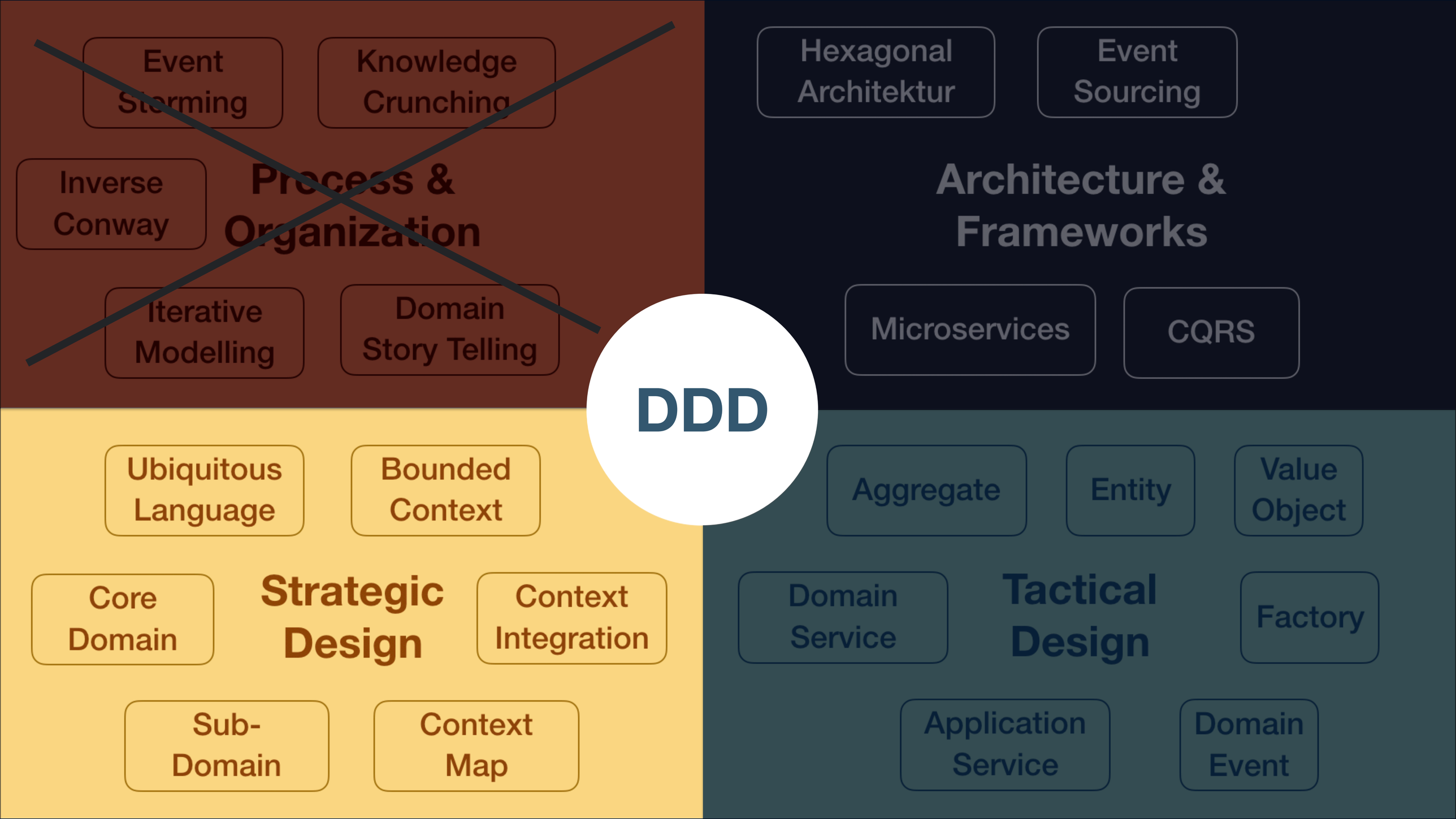
- DDD Tactical Design nur, wo es notwendig ist
- Support Domains auch per CRUD oder Transaction Script

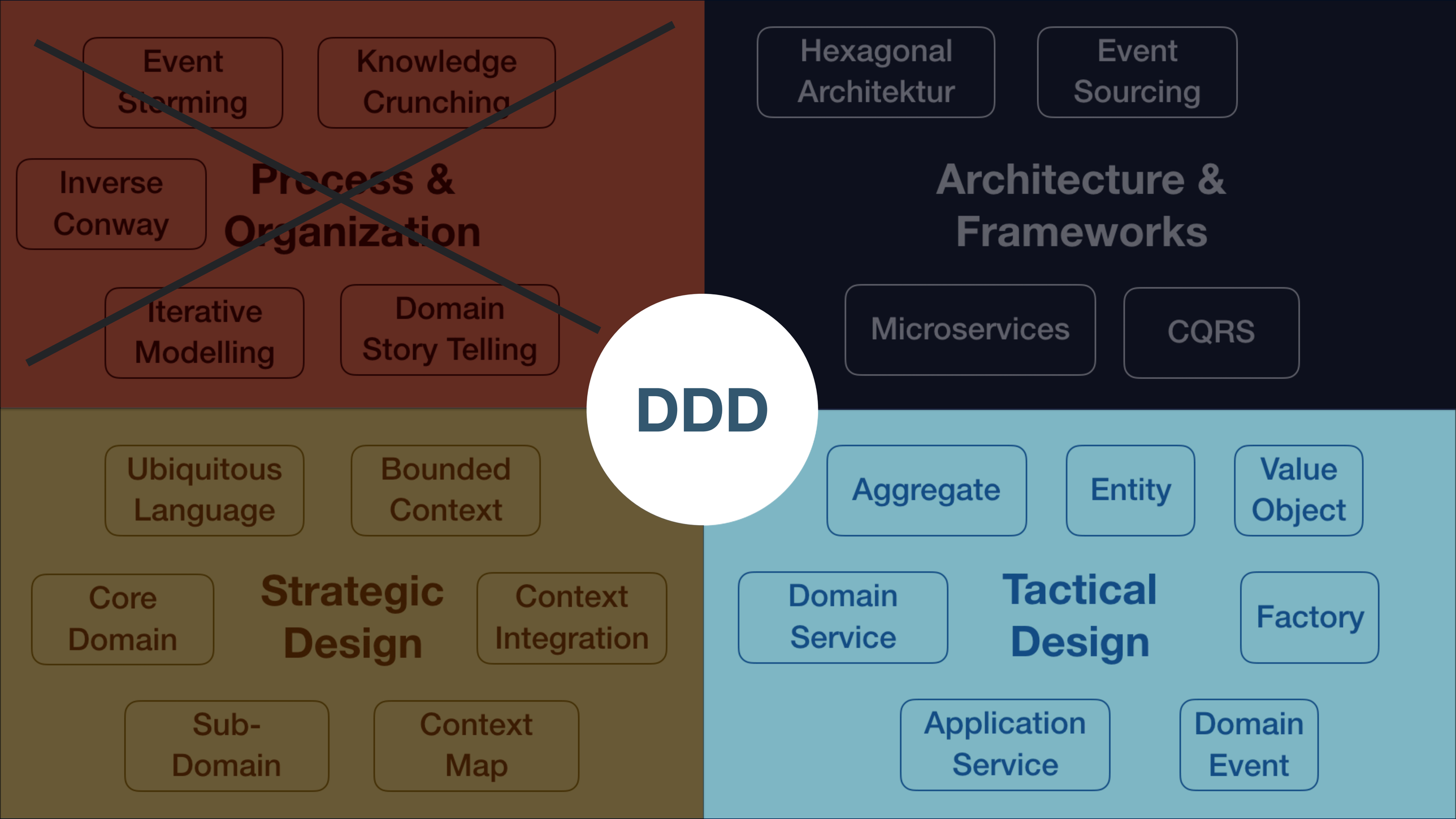
IDEEN FÜR DIE PRAXIS

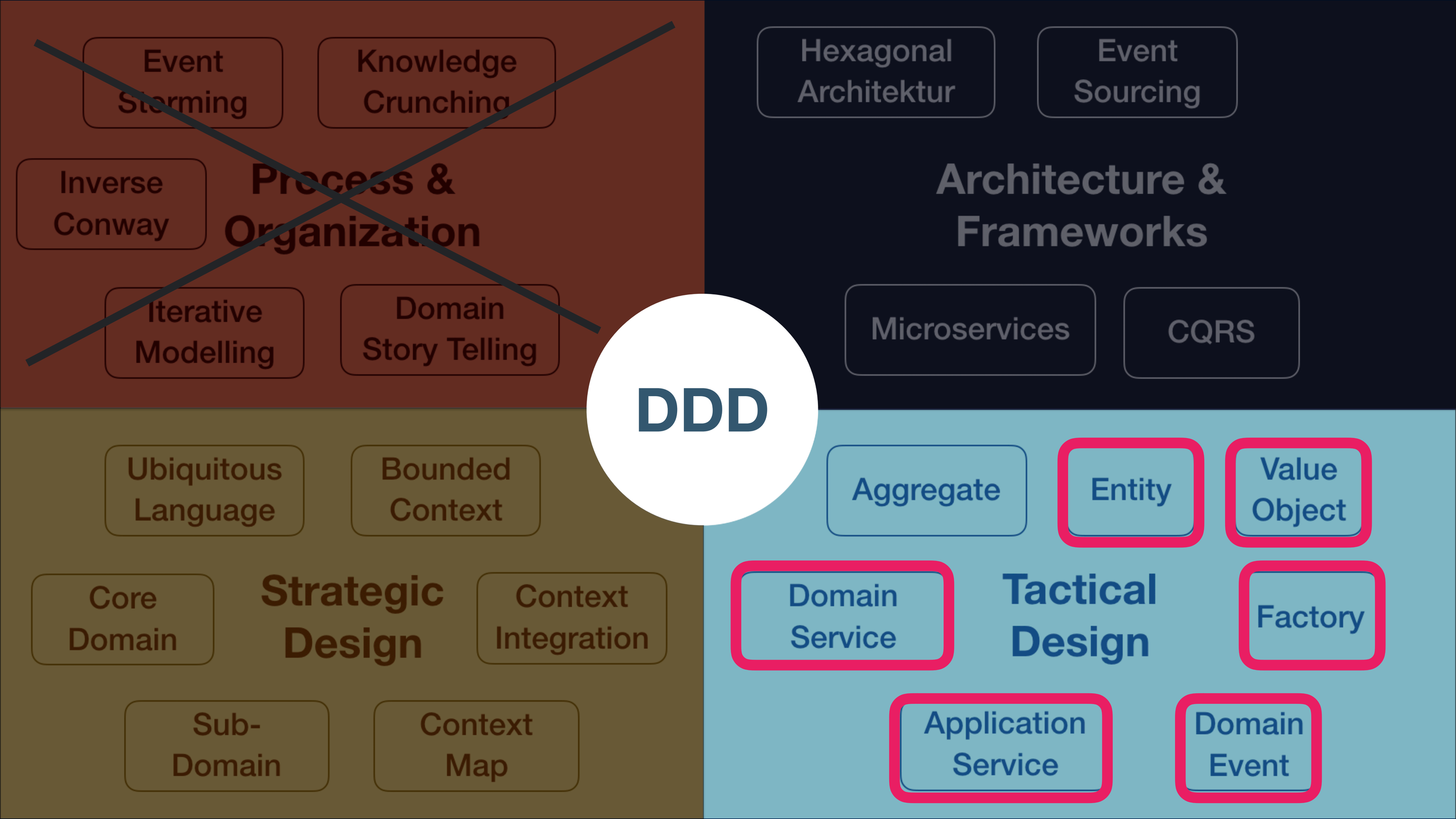
Pragmatisch bleiben

Nicht zu früh an Technik denken

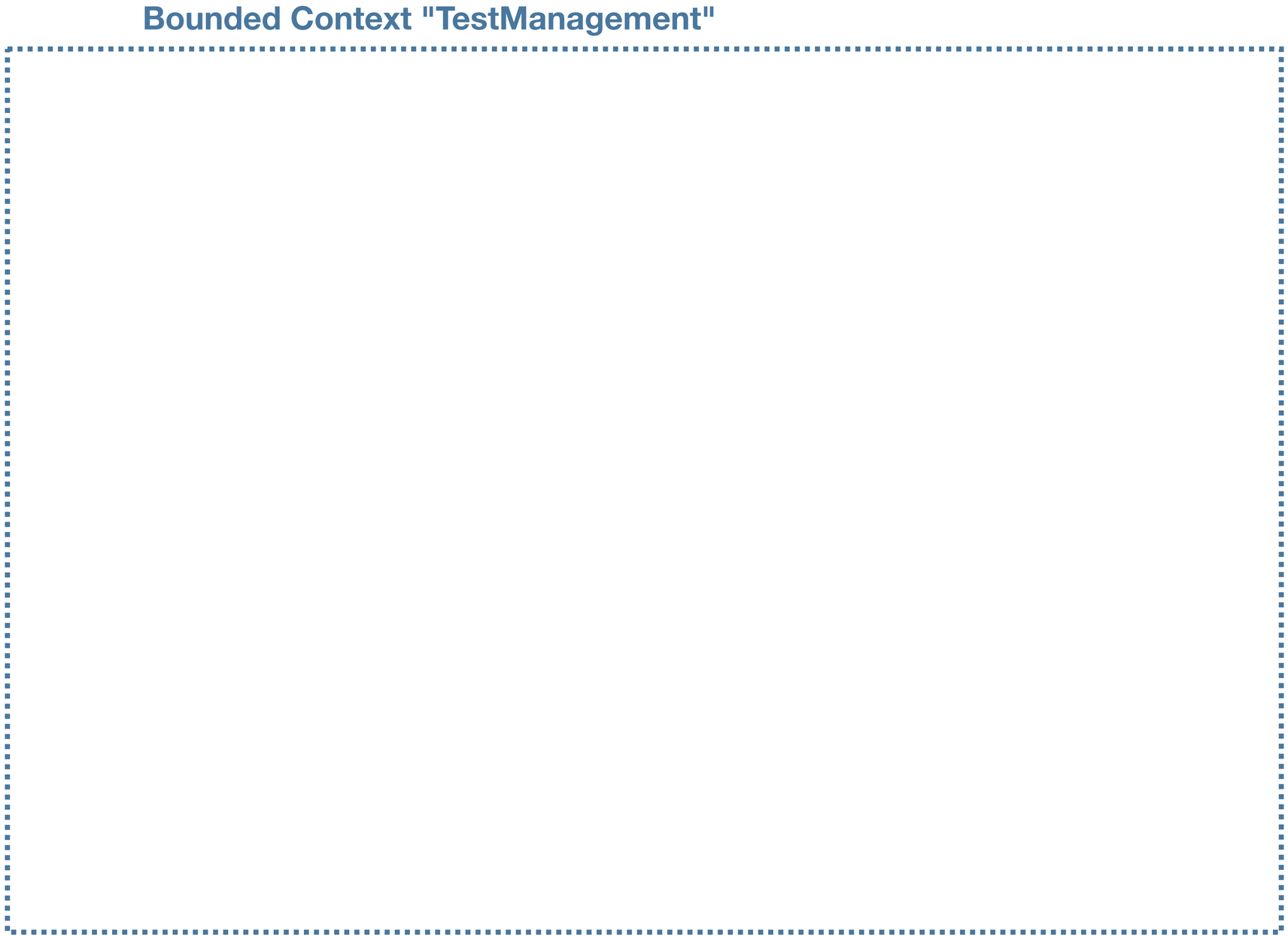
- Sub-Domains beschreiben "Problem Space", d.h. das Business
- Bounded Contexts beschreiben "Solution Space", d.h. die Realisierung
- Business-Anforderungen sind die Grundlage


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

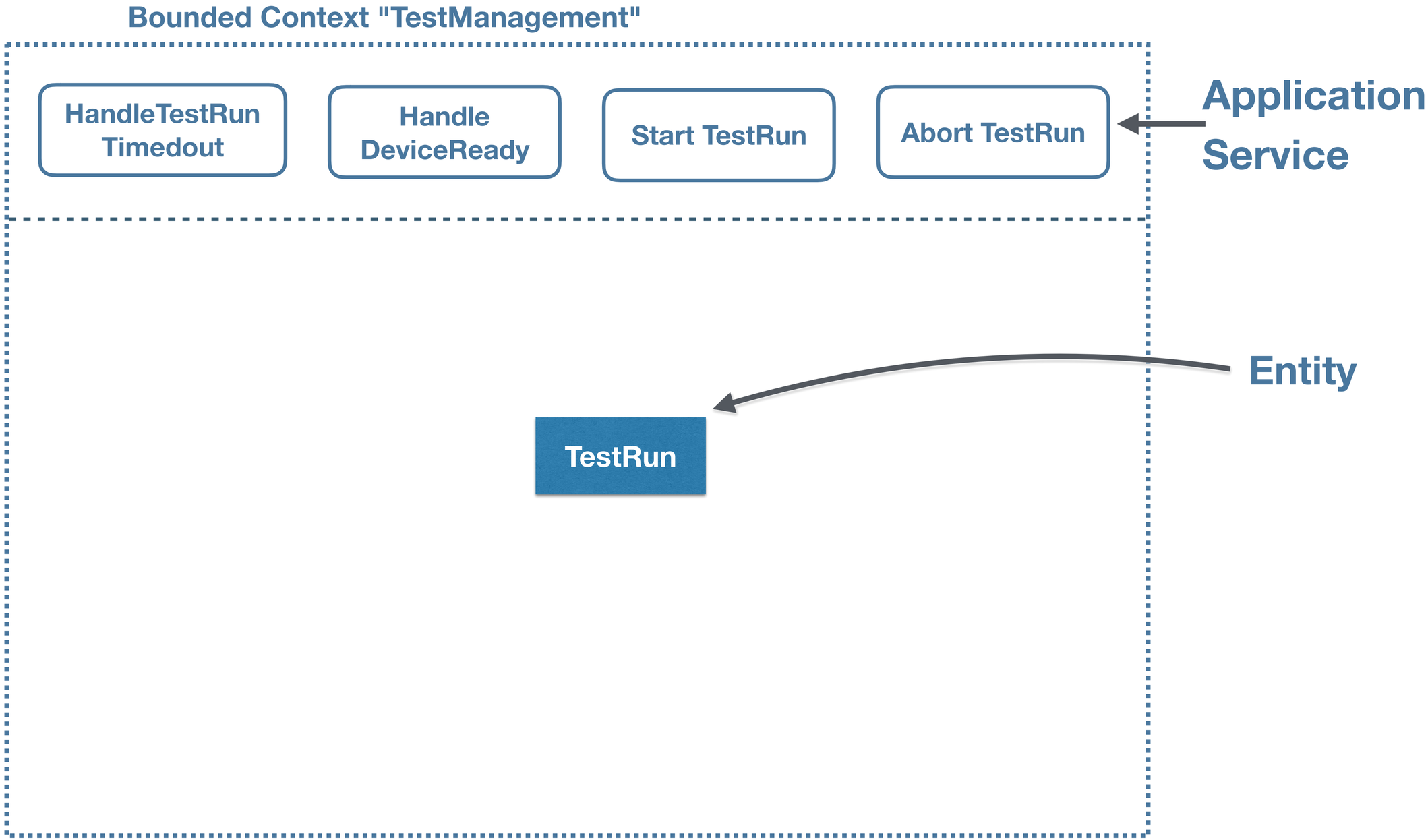

Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

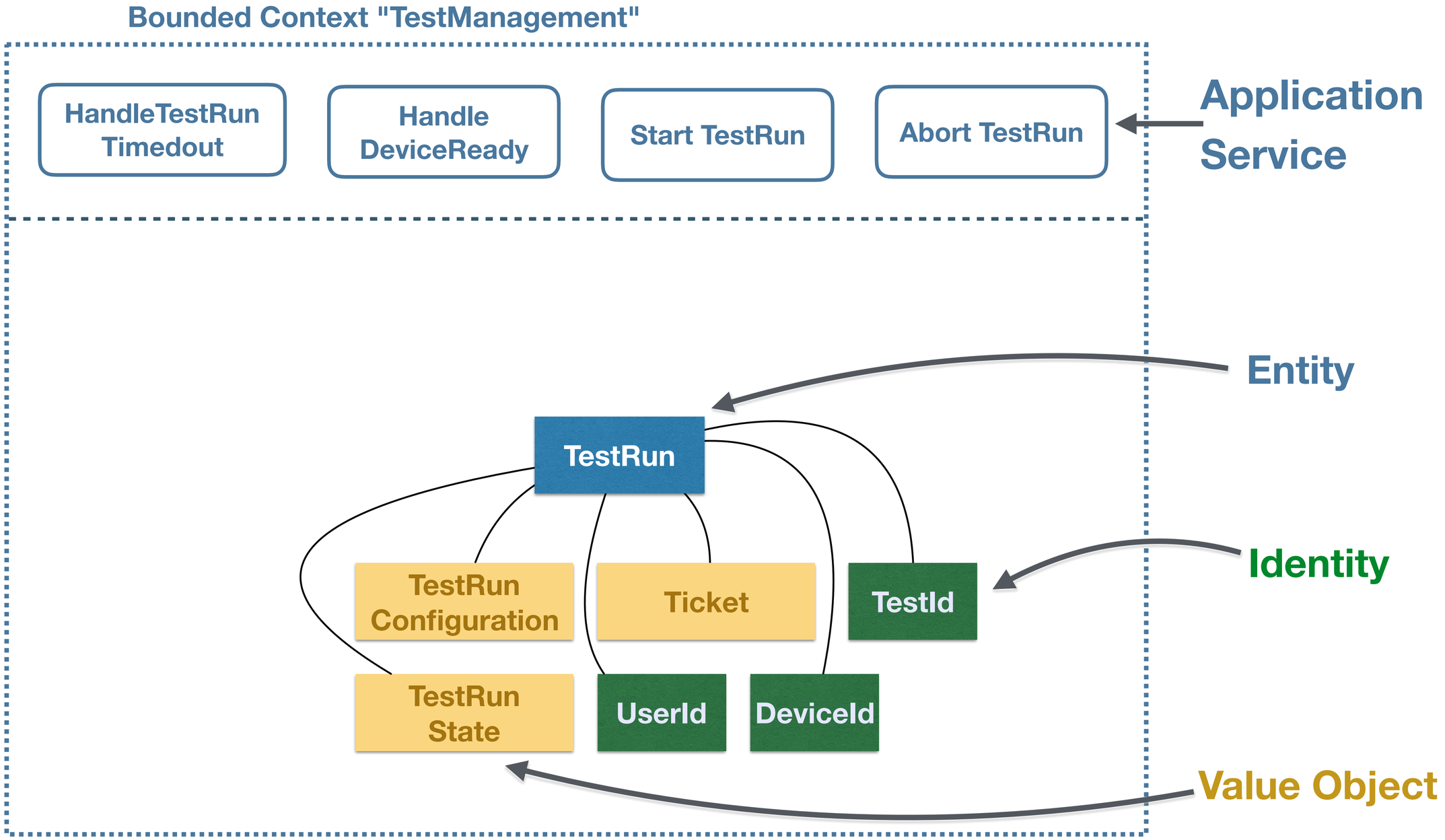

Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

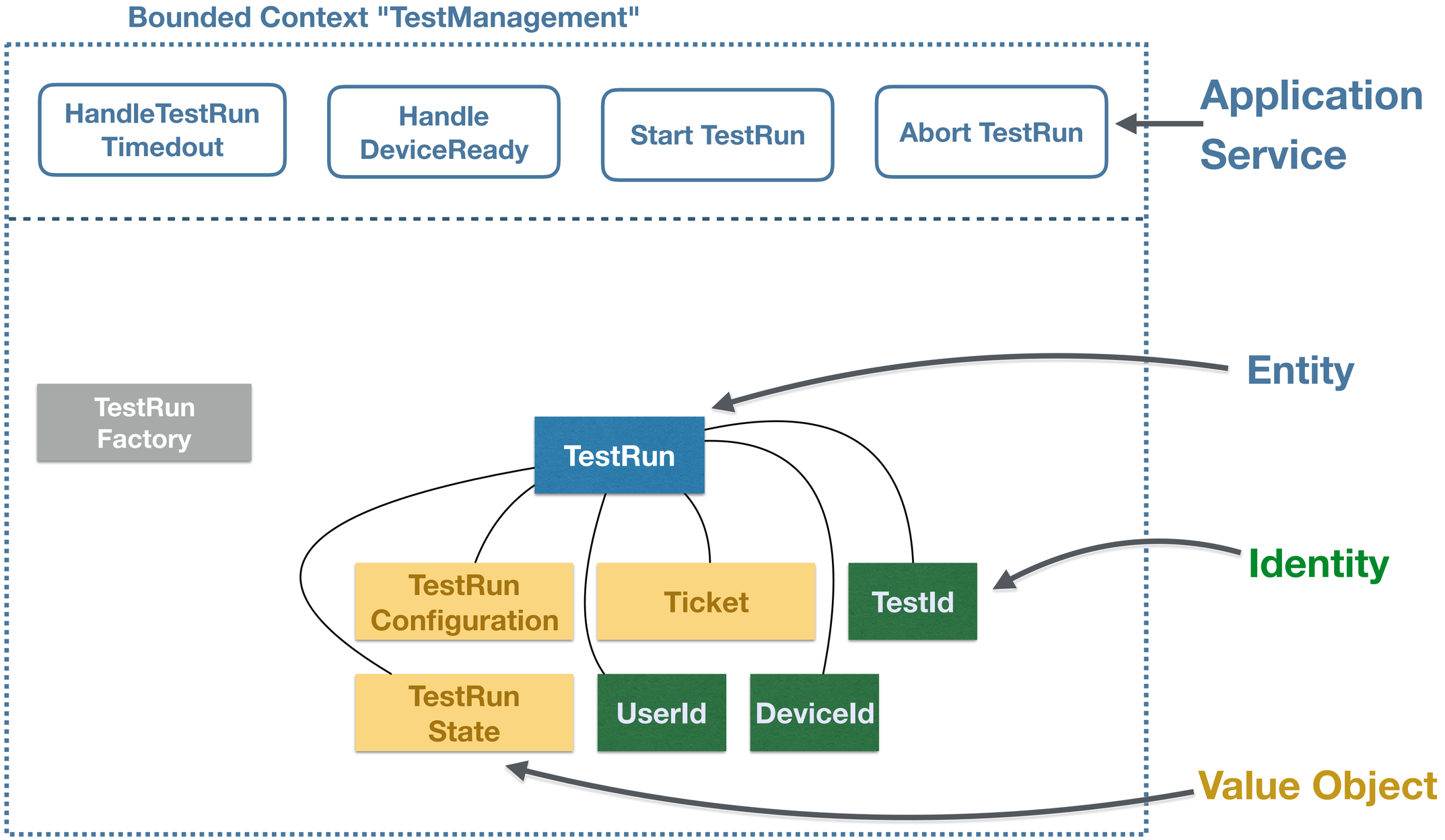
TACTICAL DESIGN - ÜBERSICHT

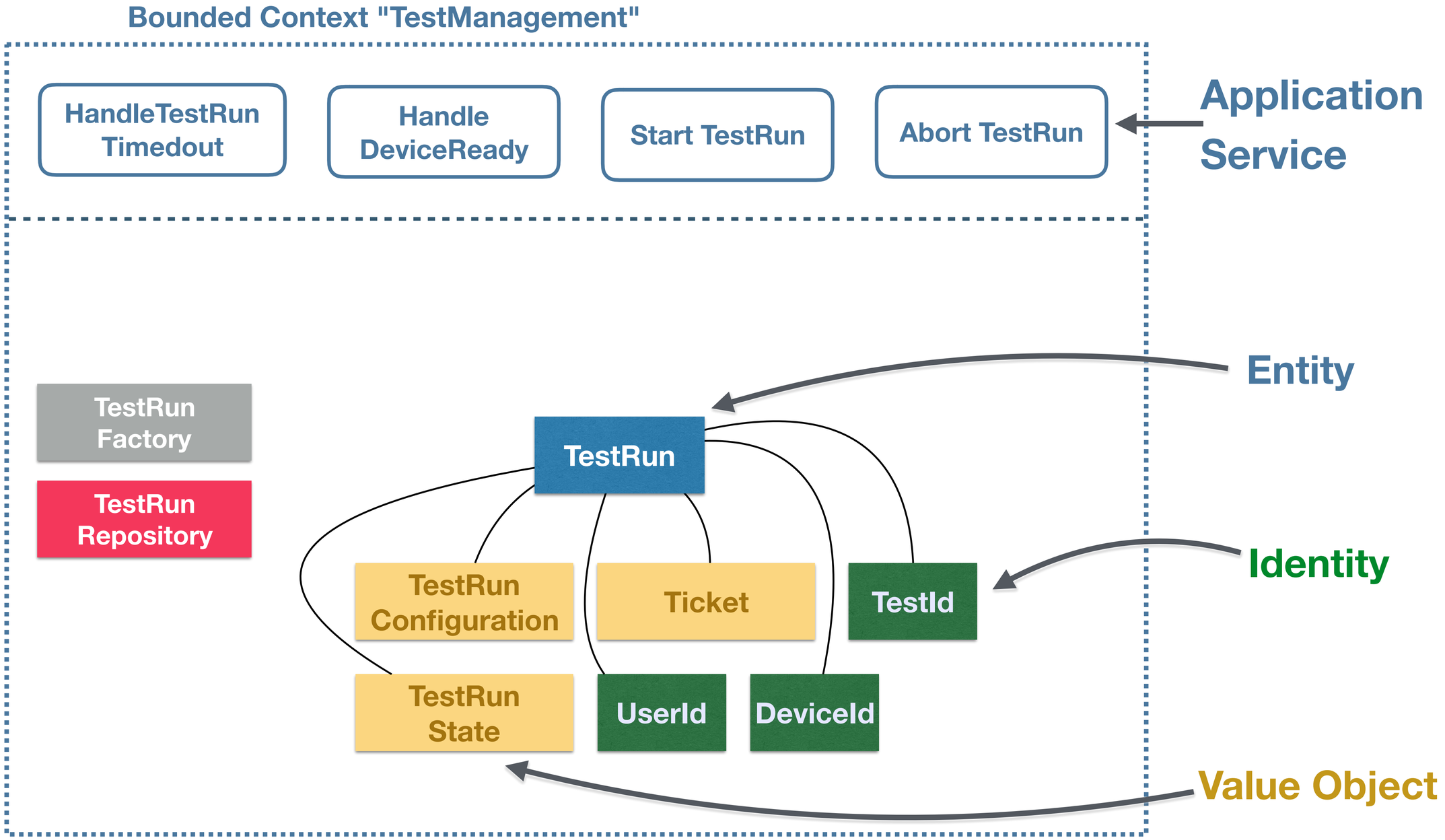

Bounded Context "TestManagement"

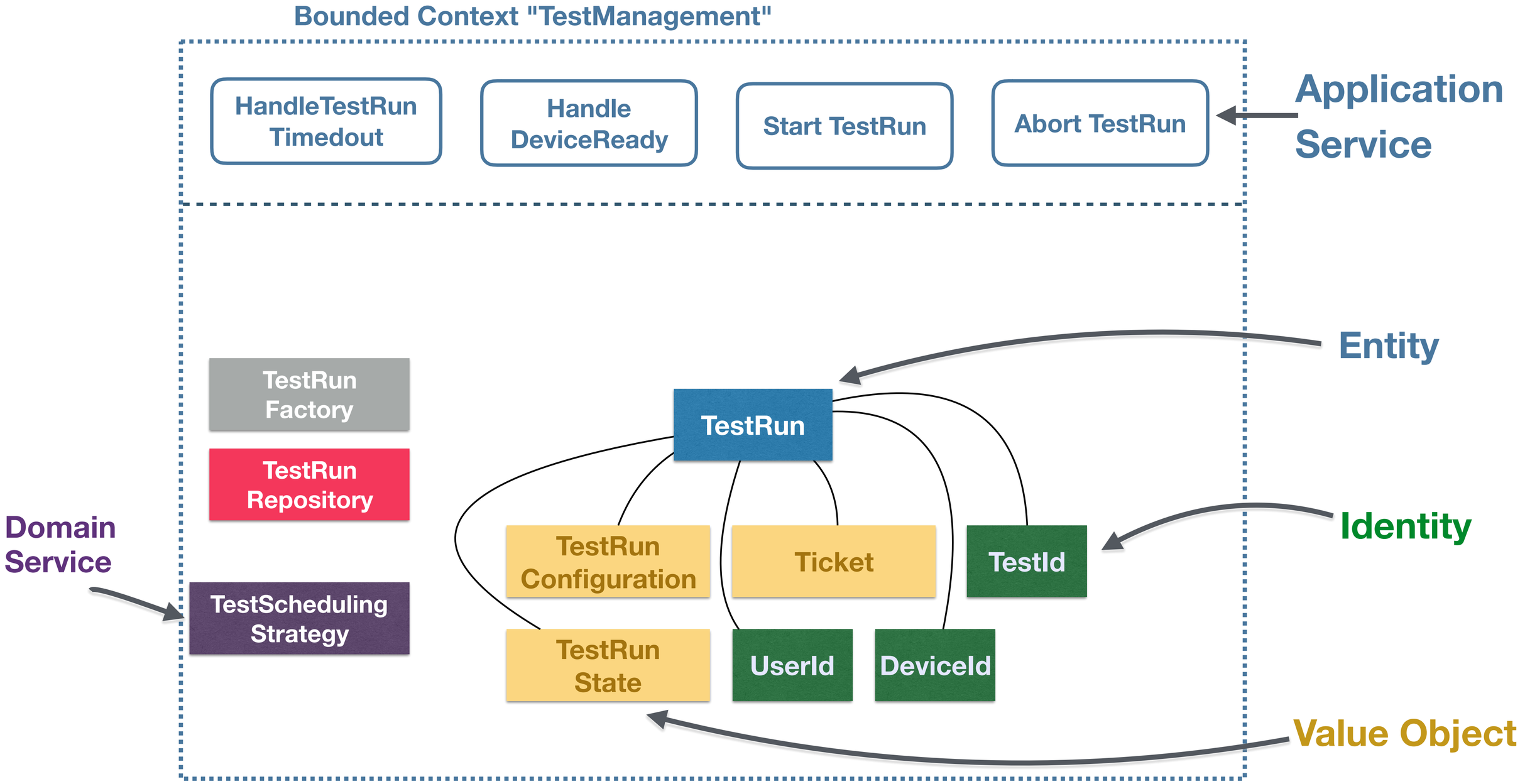

Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service

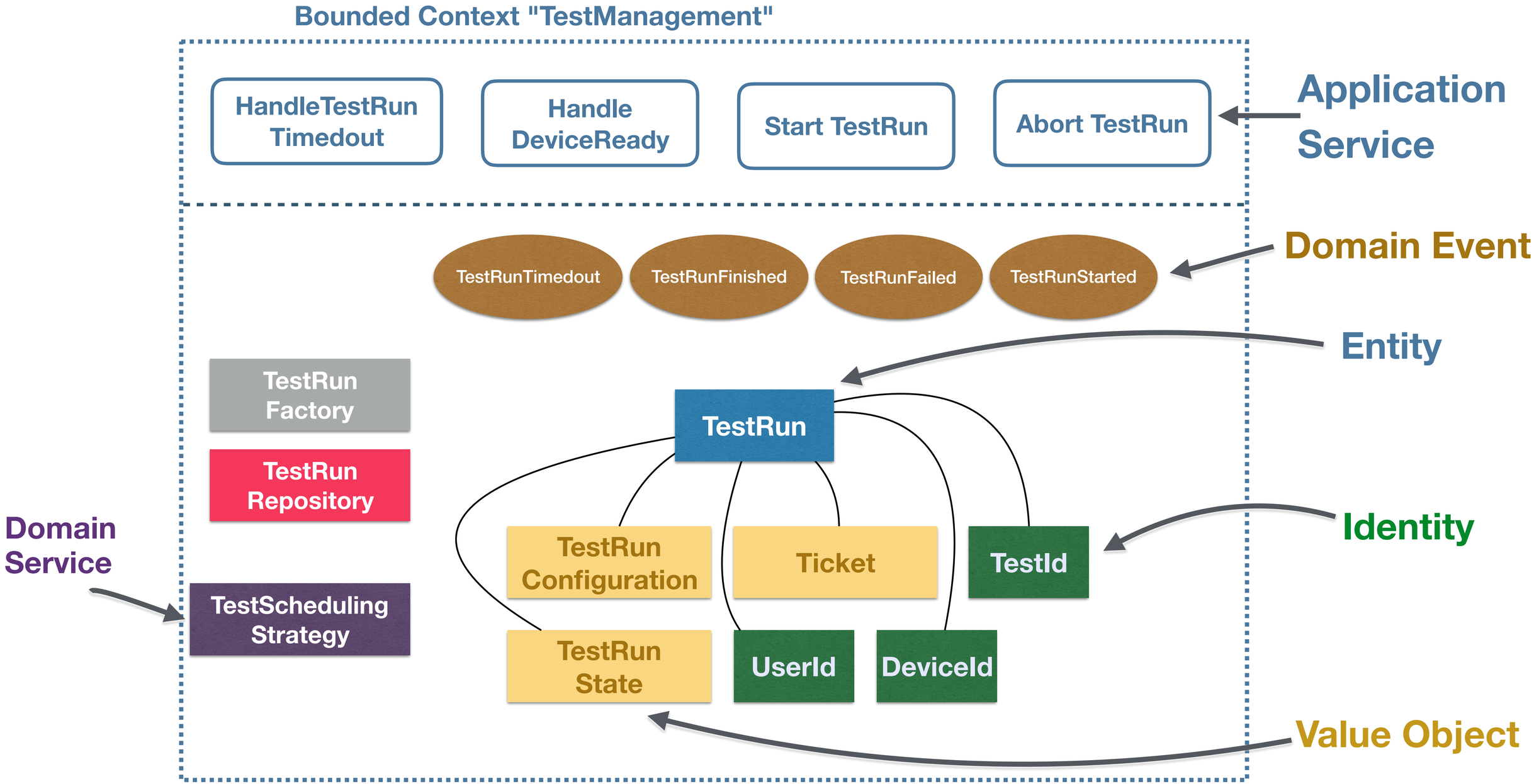

Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
Entity
TestRun

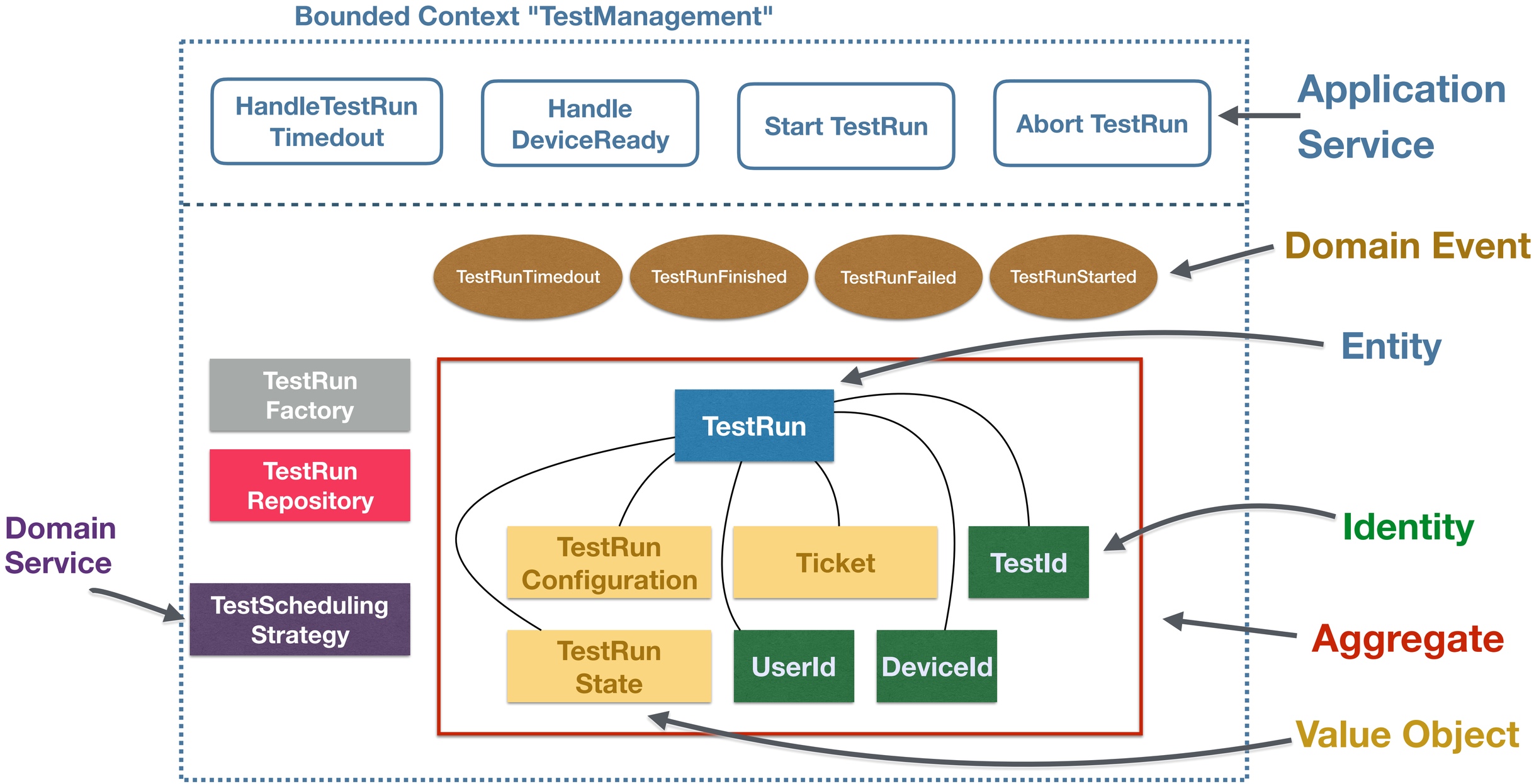

Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
Entity
TestRun
TestRun Configuration
Ticket
TestId
Identity
TestRun State
UserId
DeviceId
Value Object


Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
TestRun Factory
TestRun
TestRun Configuration
Ticket
TestId
TestRun State
UserId
DeviceId
Entity
Identity
Value Object

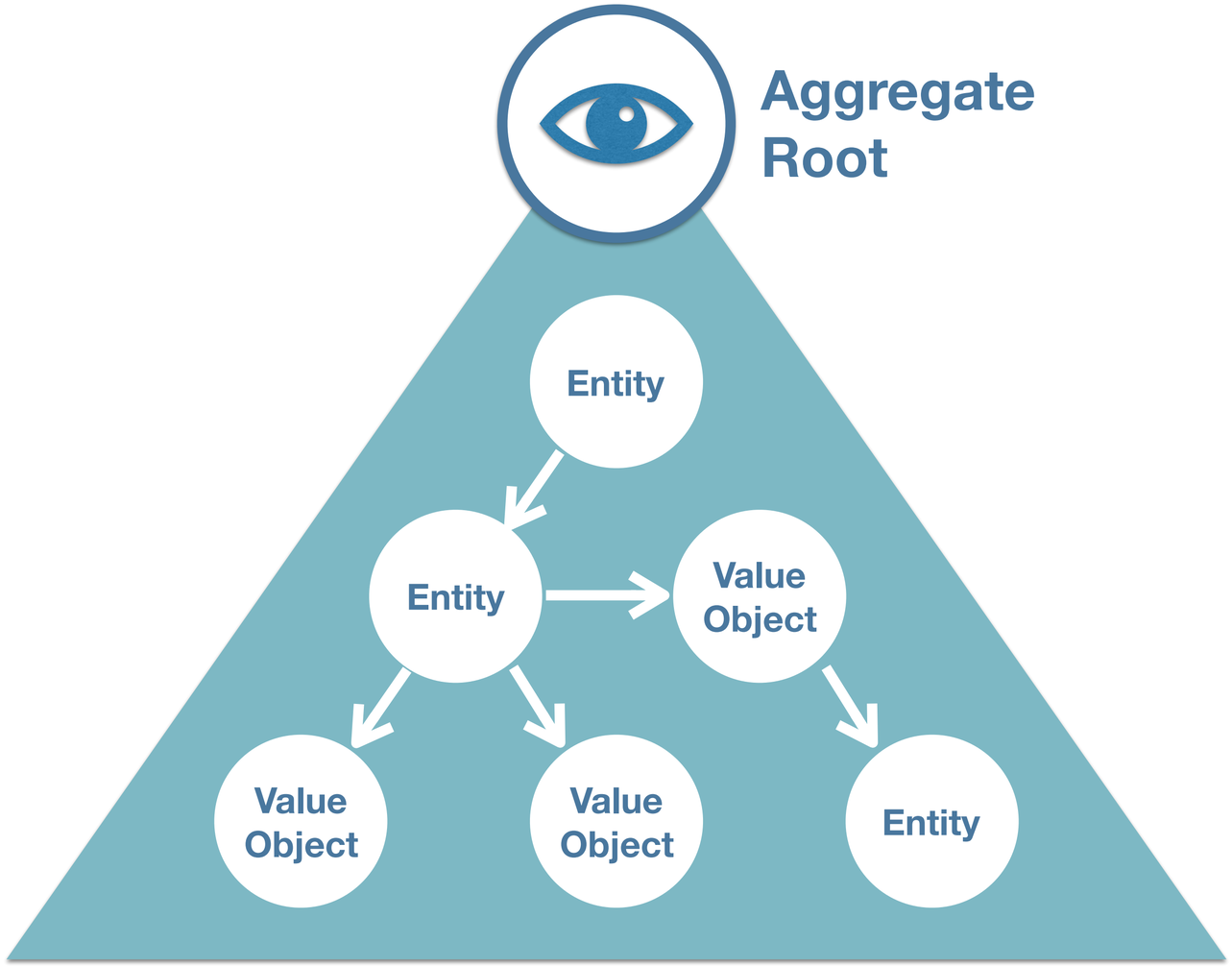

Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
TestRun Factory
TestRun Repository
TestRun
TestRun Configuration
Ticket
TestId
TestRun State
UserId
DeviceId
Entity
Identity
Value Object


Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
TestRun Factory
TestRun Repository
TestScheduling Strategy
Domain Service
TestRun
TestRun Configuration
Ticket
TestId
TestRun State
UserId
DeviceId
Entity
Identity
Value Object

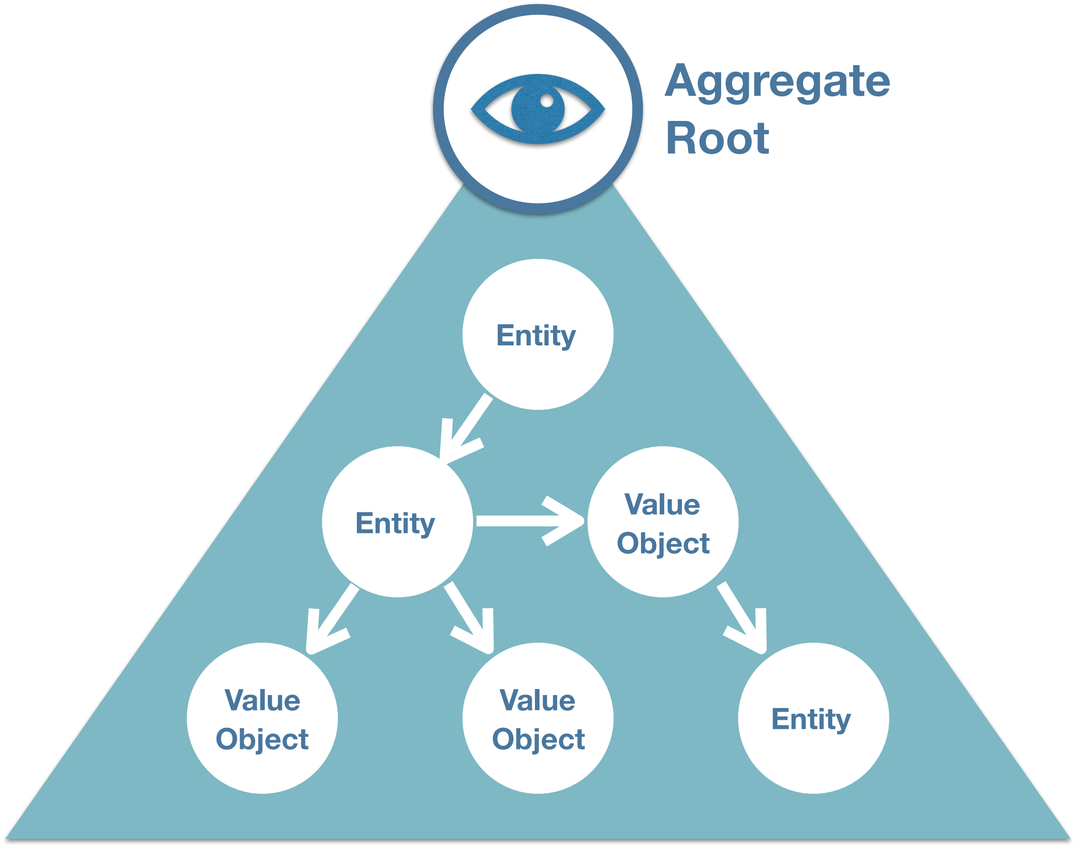

Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
TestRunTimedout
TestRunFinished
TestRunFailed
TestRunStarted
Domain Event
Entity
TestRun Factory
TestRun Repository
TestScheduling Strategy
Domain Service
TestRun
TestRun Configuration
Ticket
TestId
TestRun State
UserId
DeviceId
Identity
Value Object


Bounded Context "TestManagement"
HandleTestRun Timedout
Handle DeviceReady
Start TestRun
Abort TestRun
Application Service
TestRunTimedout
TestRunFinished
TestRunFailed
TestRunStarted
Domain Event
Entity
TestRun Factory
TestRun Repository
TestScheduling Strategy
Domain Service
TestRun
TestRun Configuration
Ticket
TestId
TestRun State
UserId
DeviceId
Identity
Aggregate
Value Object

AGGREGATES: Zentrales Konzept im Domänenmodell



AGGREGATE: Zentrales Konzept im Domänenmodell



- Aggregat: zusammenhängende Menge an Objekten
- Aggregat muss zu jedem Zeitpunkt valide und **fachlich konsistent** sein

  ➡ **"Consistency Boundary"**
- auf Aggregat kann von außen nur über "Aggregate Root" zugegriffen werden
  - innere Objekte von außen nicht einzeln ansprechbar
- Aggregat ist Basis für Transaktionalität

TIPPS AGGREGATE DESIGN

Aggregate gut wählen

Aggregate klein halten

Eine Transaktion - ein Aggregat

Aggregat-Referenz via Id

- Aggregate richtig schneiden ist schwierig
- Falsche Entscheidung kann u.a. Skalierbarkeit verschlechtern

TIPPS AGGREGATE DESIGN

Aggregate gut wählen

Aggregate klein halten

Eine Transaktion - ein Aggregat

Aggregat-Referenz via Id

- Sind Aggregate zu groß, drohen Concurrency-Probleme
- Oftmals ein Zeichen für falsche Modellierung
- an die (wesentlichen) Usecases denken

TIPPS AGGREGATE DESIGN

Aggregate gut wählen

Aggregate klein halten

Eine Transaktion - ein Aggregat

Aggregat-Referenz via Id

- Verbessert Skalierbarkeit
- Spiegelt die Grundidee von Aggregaten wider
- "Daumenregel"

TIPPS AGGREGATE DESIGN

Aggregate gut wählen

Aggregate klein halten

Eine Transaktion -
ein Aggregat

Aggregat-Referenz via Id

- Ids sind fachliche Modellobjekte
- keine Java-Referenzen
- keine Foreign-Key-Beziehungen in der DB

TIPPS AGGREGATE DESIGN

Aggregate g

Aggregate k

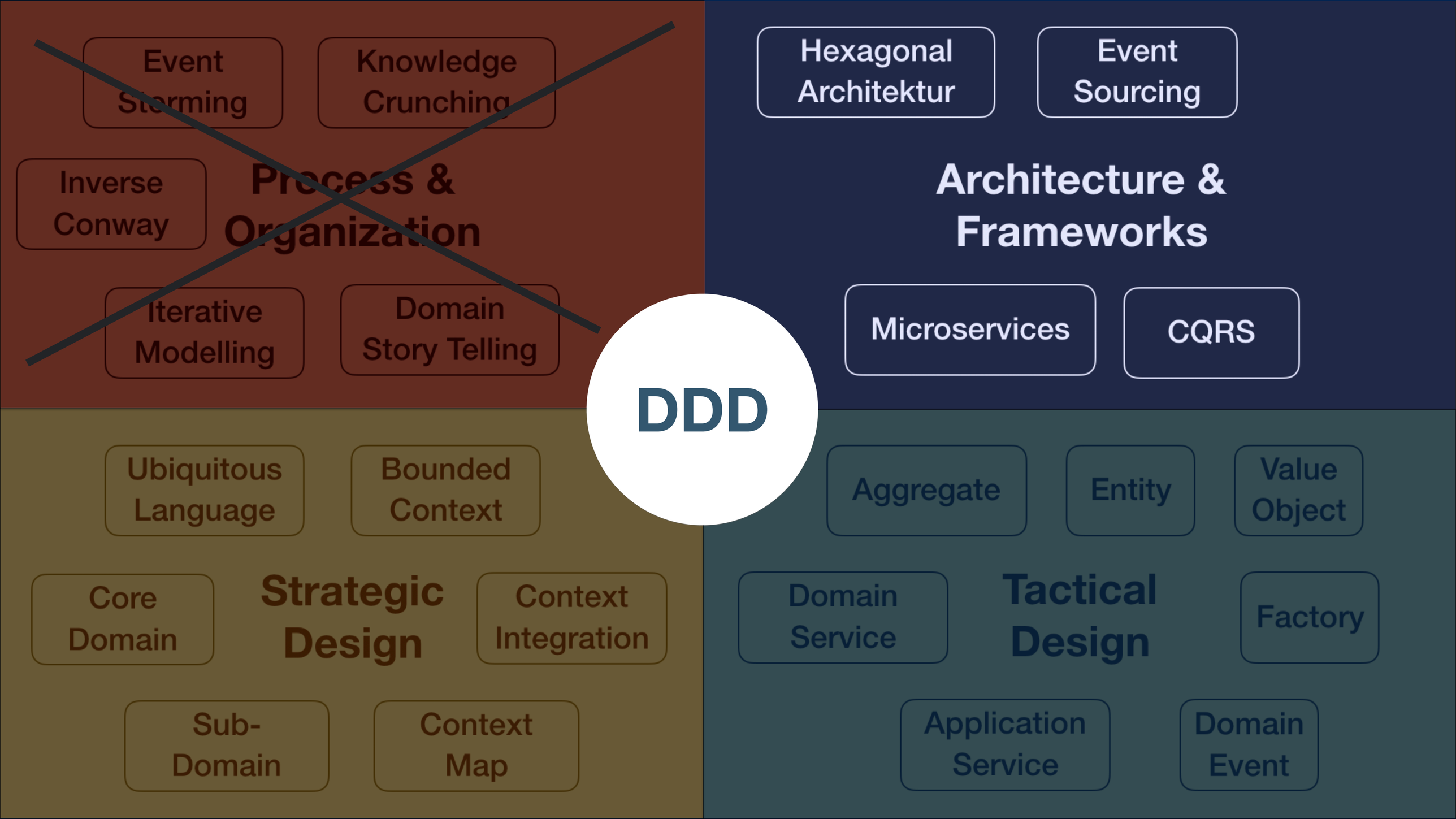
Eine Tran
ein Agg

Aggregat-Referenz via Id

**Aggregat-Persistenz
durch Document-Stores
bietet sich an**

ekte

gen in der DB


Event Storming
Knowledge Crunching
Inverse Conway
Process & Organization
Iterative Modelling
Domain Story Telling
Hexagonal Architektur
Event Sourcing
Architecture & Frameworks
Microservices
CQRS
DDD
Ubiquitous Language
Bounded Context
Core Domain
Strategic Design
Context Integration
Sub-Domain
Context Map
Aggregate
Entity
Value Object
Domain Service
Tactical Design
Factory
Application Service
Domain Event

Event Storming

Knowledge Crunching

Inverse Conway

Process & Organization

Iterative Modelling

Domain Story Telling

Hexagonal Architektur

Event Sourcing

Architecture & Frameworks

Microservices

CQRS

DDD

Ubiquitous Language

Bounded Context

Core Domain

Strategic Design

Context Integration

Sub-Domain

Context Map

Aggregate

Entity

Value Object

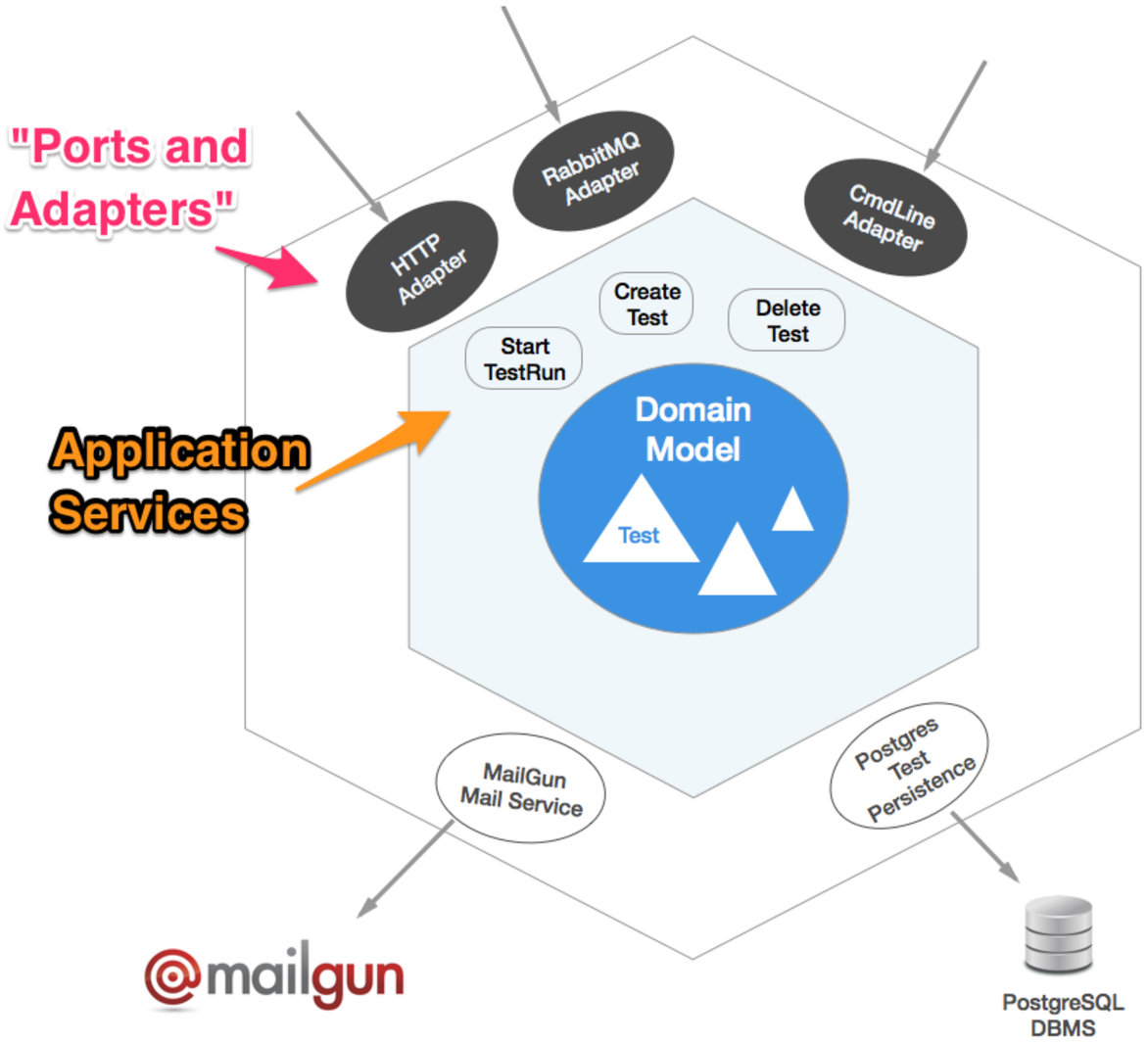
Domain Service

Tactical Design

Factory

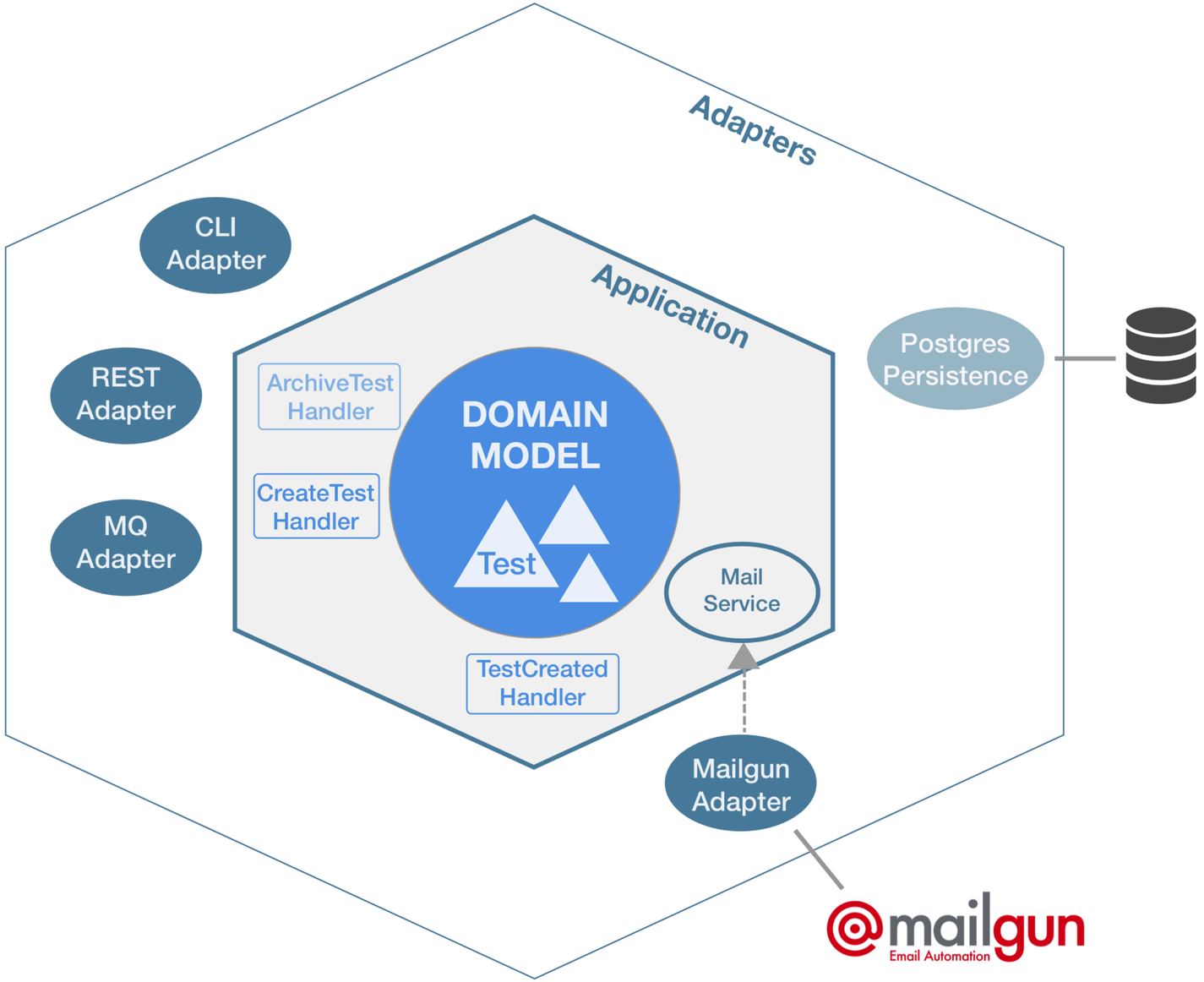
Application Service

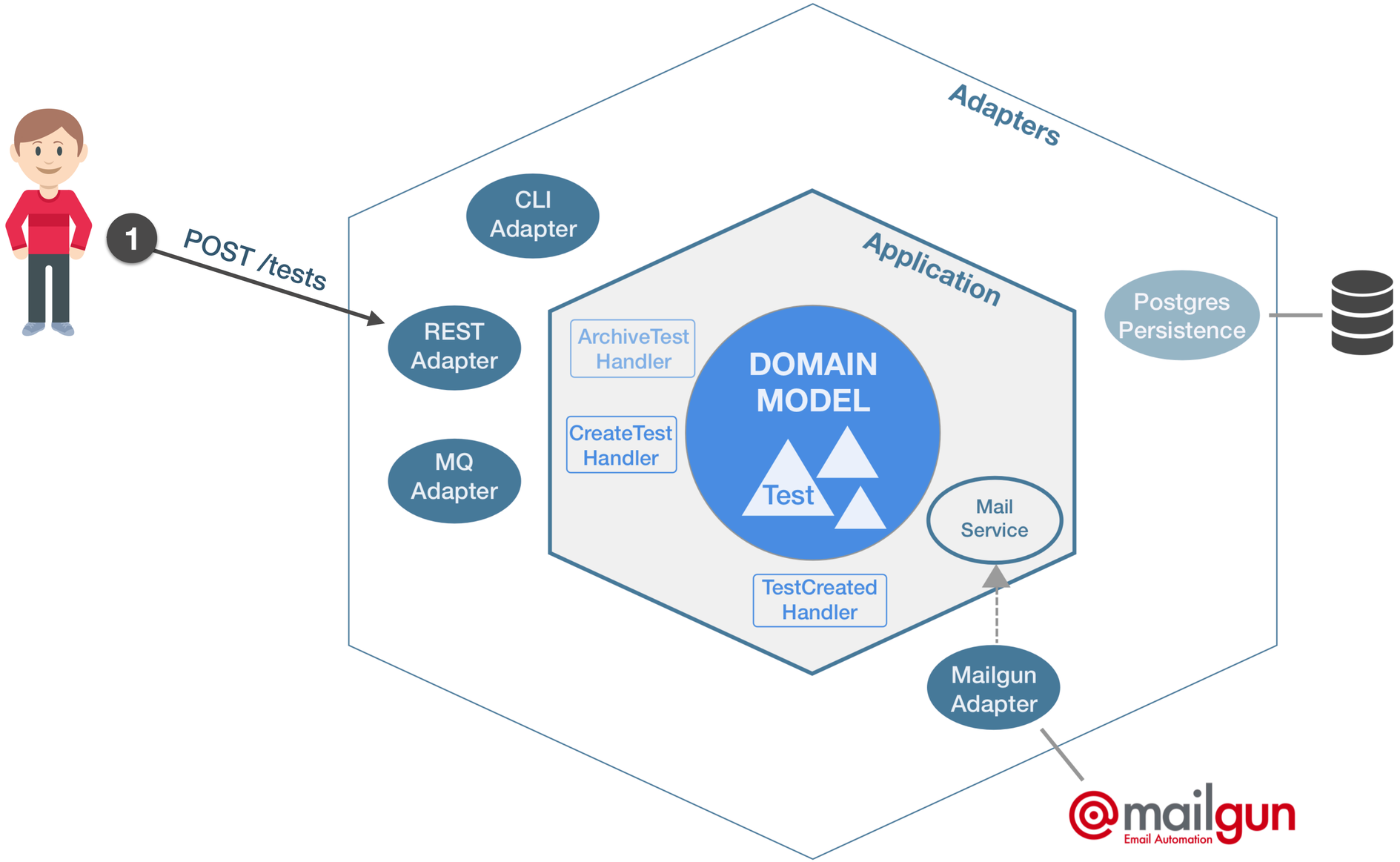
Domain Event

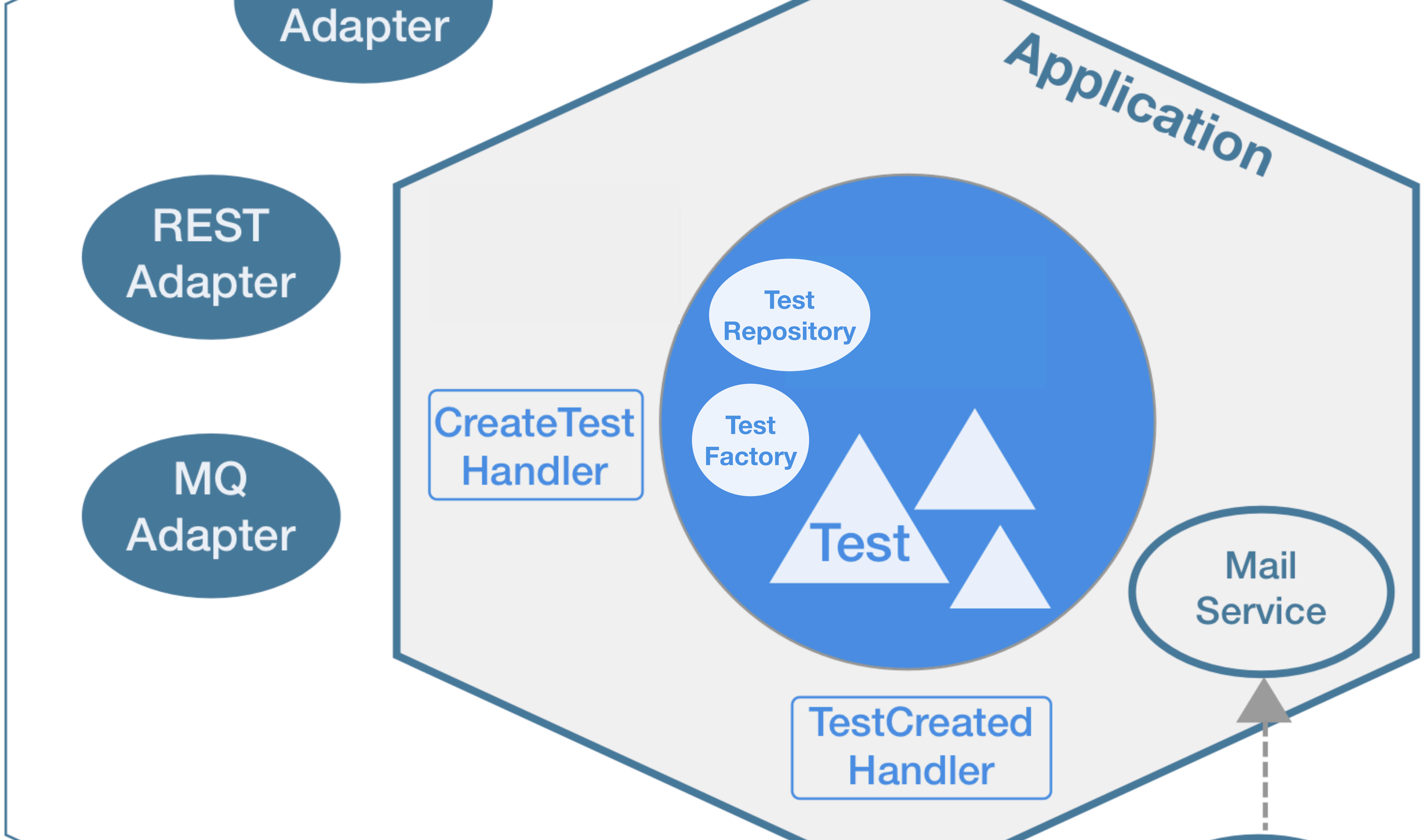


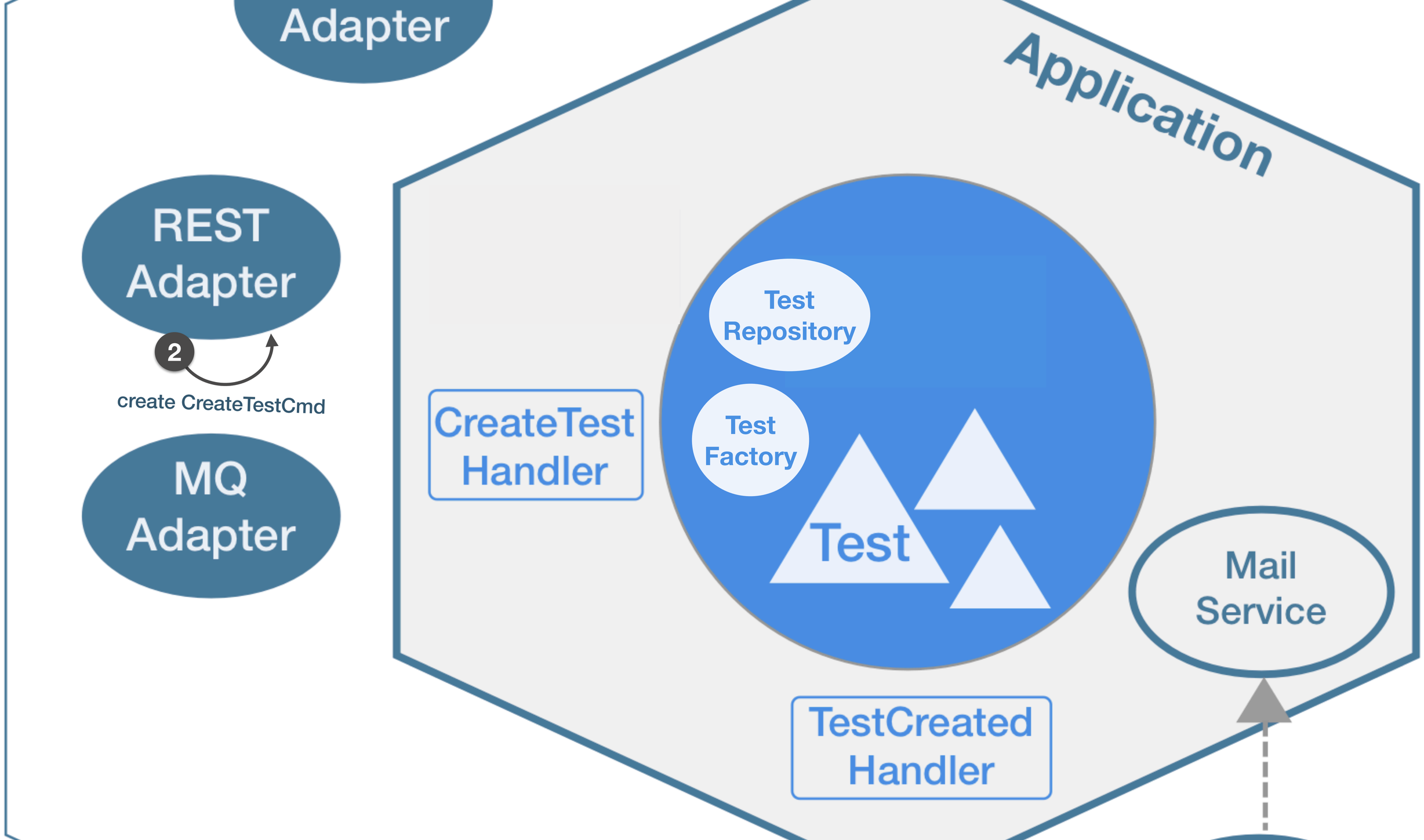
- Hexagon realisiert **Bounded Context**
- Alternative zu Schicht-Architektur
- Dependency-Inversion-Prinzip (Bob Martin):

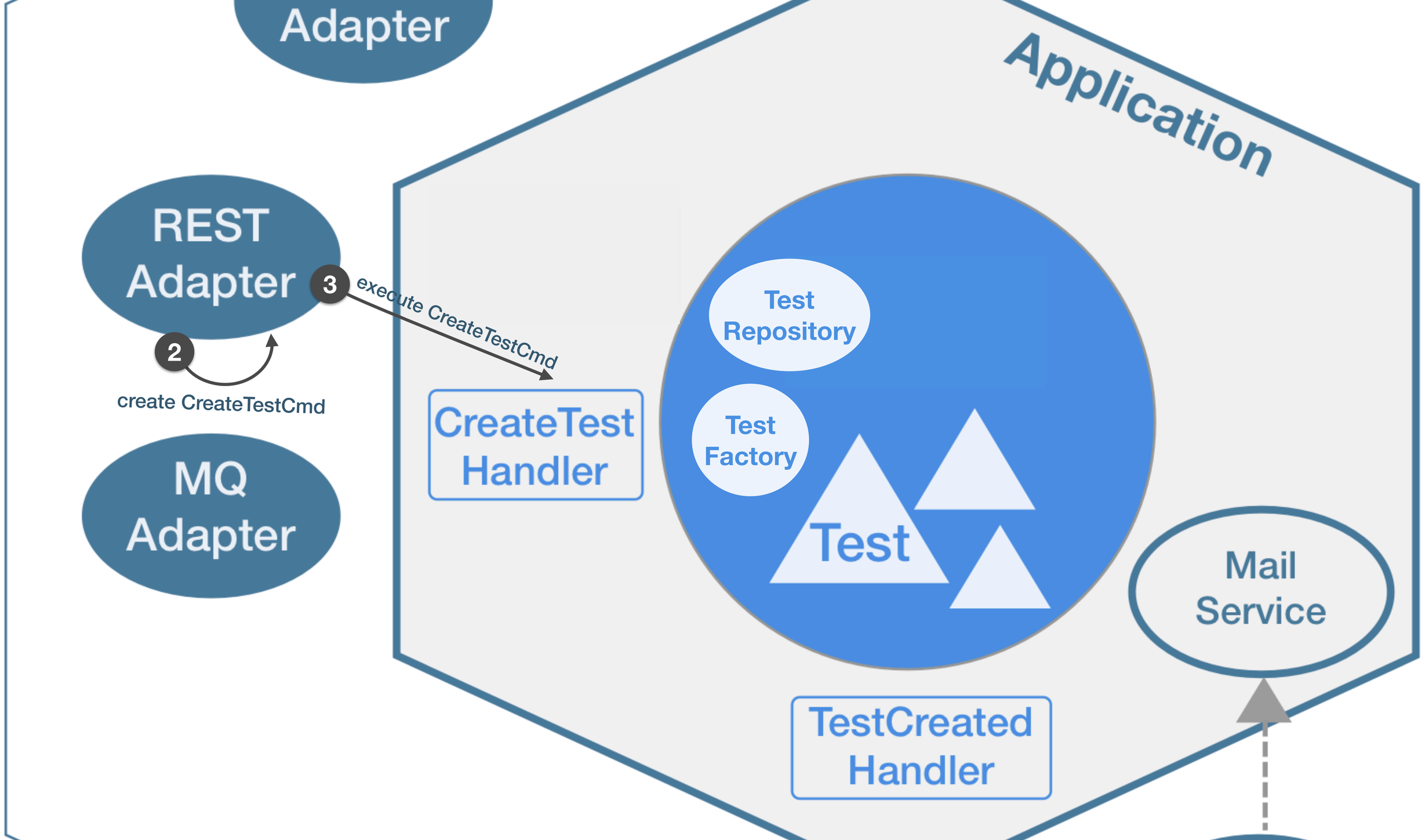
> "High-level modules should not depend on low-level modules. Both should depend on abstractions (e.g. interfaces)."
>
> "Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions."

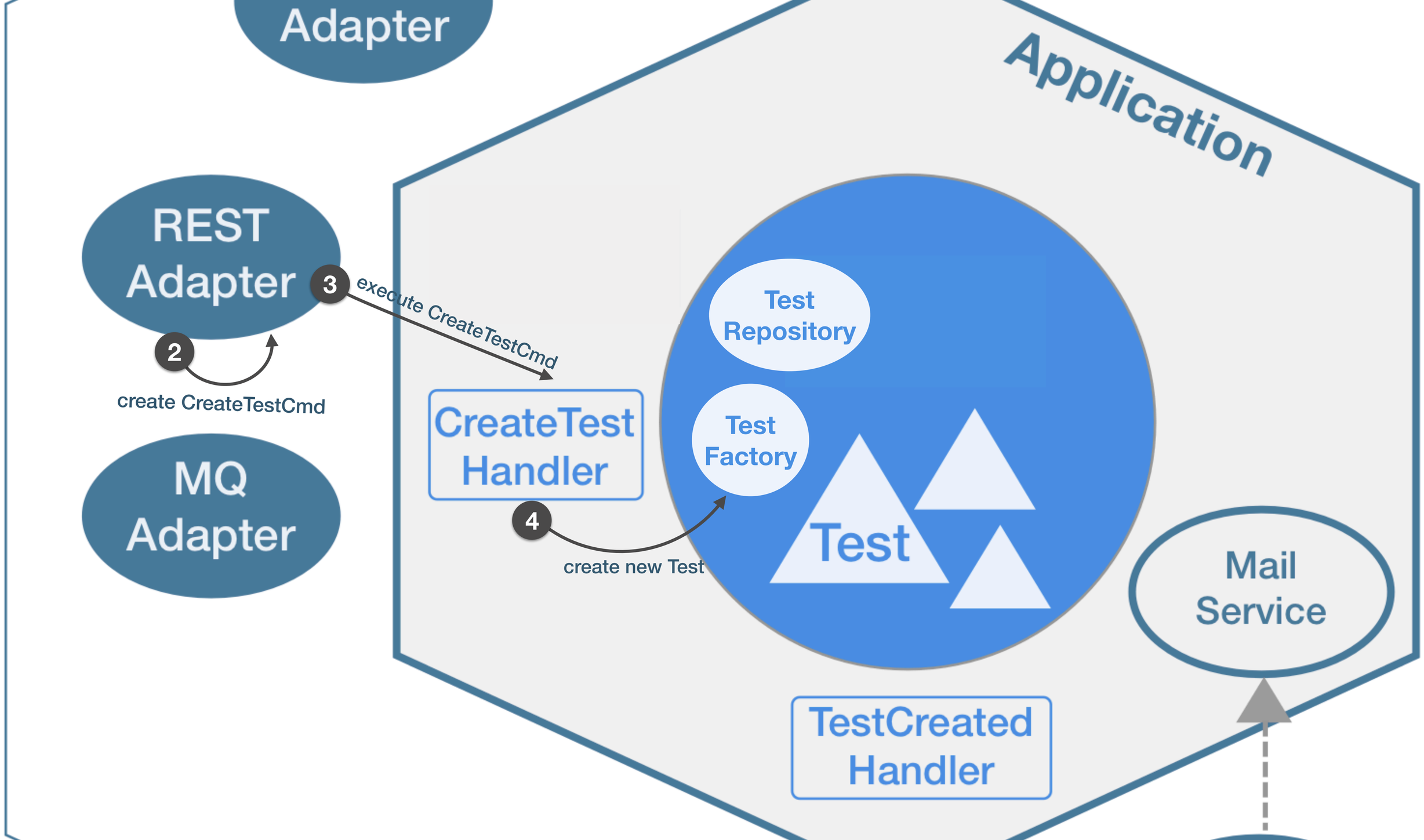

Adapters
CLI Adapter
REST Adapter
MQ Adapter
Application
ArchiveTest Handler
CreateTest Handler
DOMAIN MODEL
Test
TestCreated Handler
Mail Service
Postgres Persistence
Mailgun Adapter
@mailgun
Email Automation

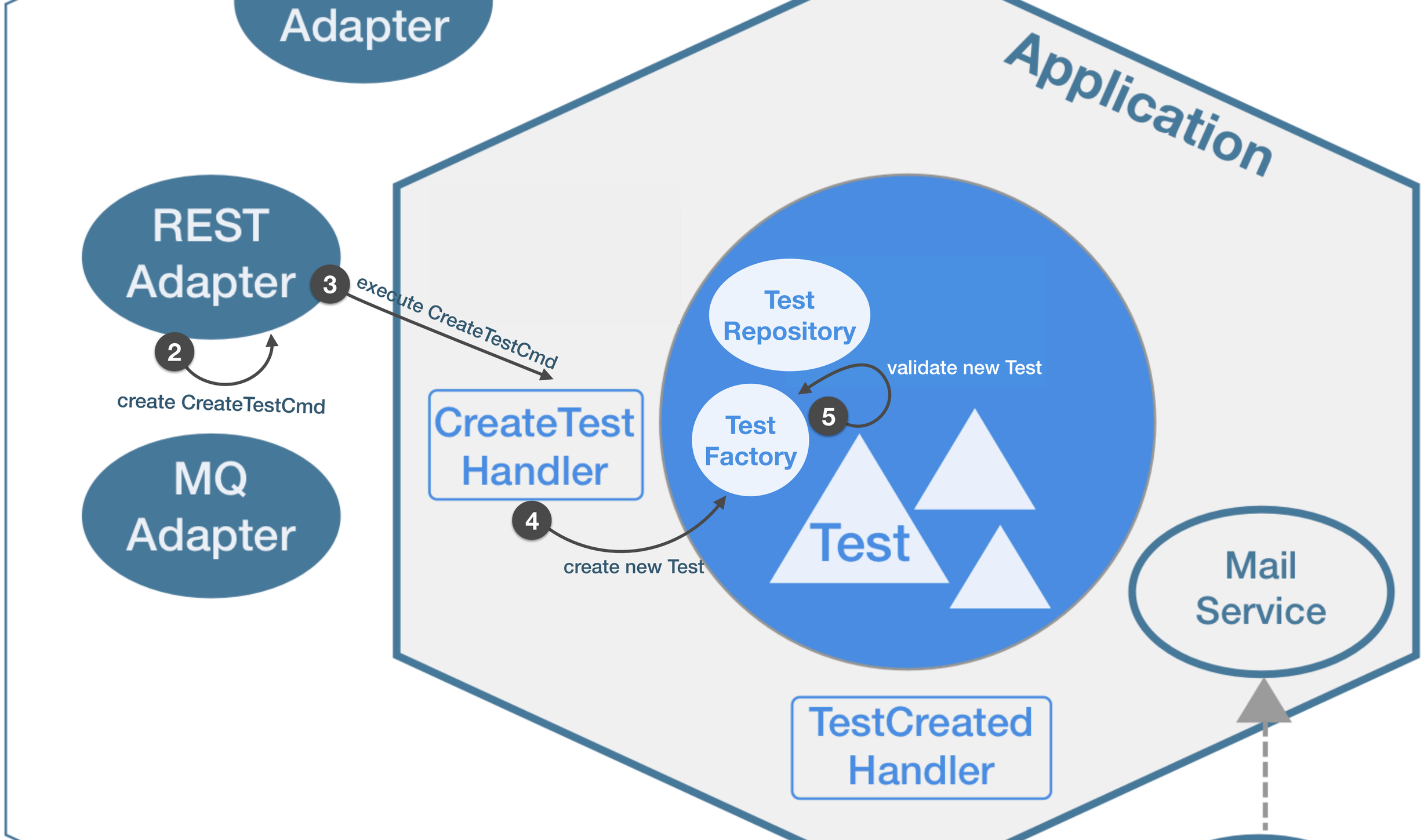

Adapters
CLI Adapter
1
POST /tests
REST Adapter
MQ Adapter
Application
ArchiveTest Handler
CreateTest Handler
DOMAIN MODEL
Test
TestCreated Handler
Mail Service
Postgres Persistence
Mailgun Adapter
@mailgun
Email Automation

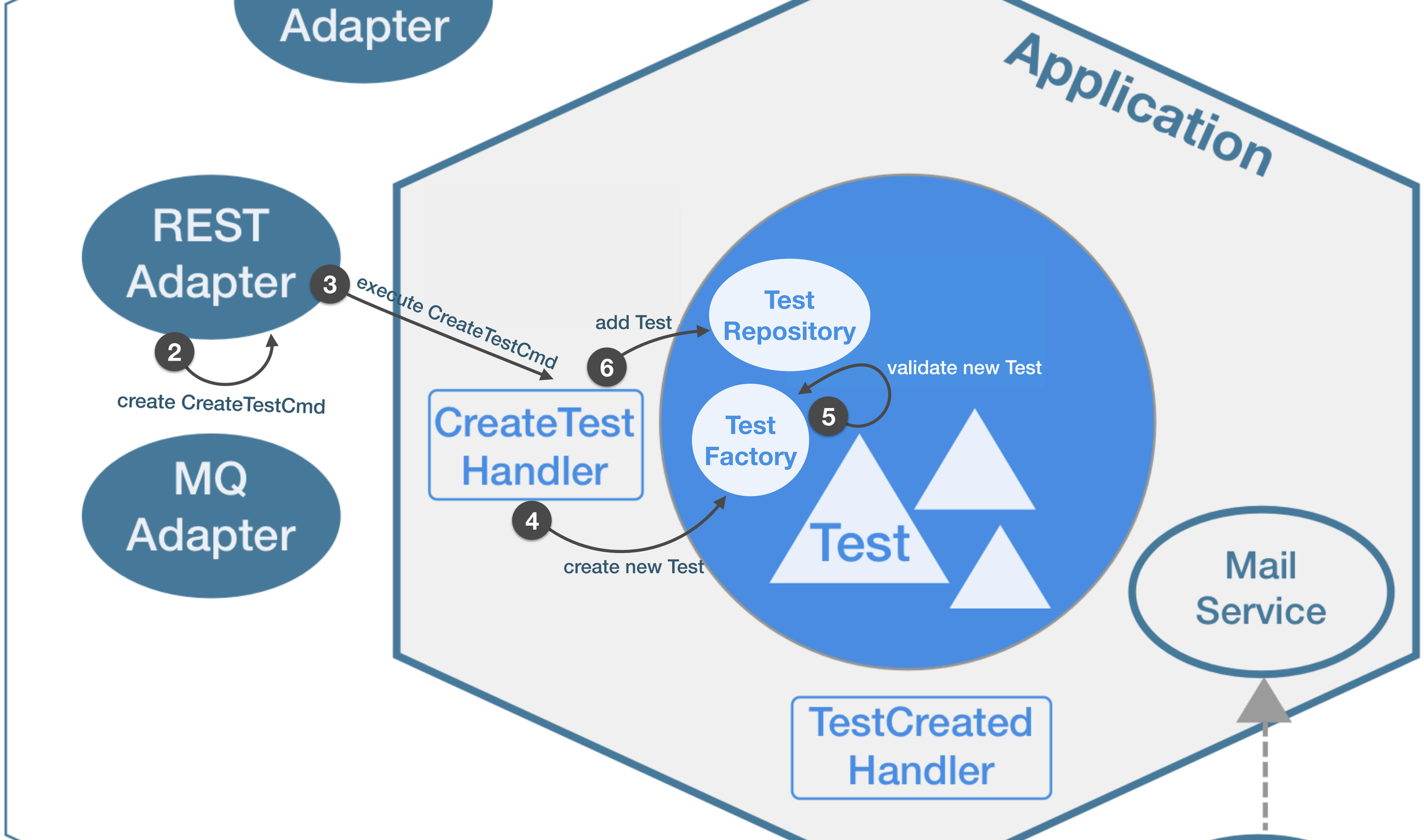

Adapter
REST
Adapter
MQ
Adapter
Application
Test
Repository
Test
Factory
Test
CreateTest
Handler
TestCreated
Handler
Mail
Service

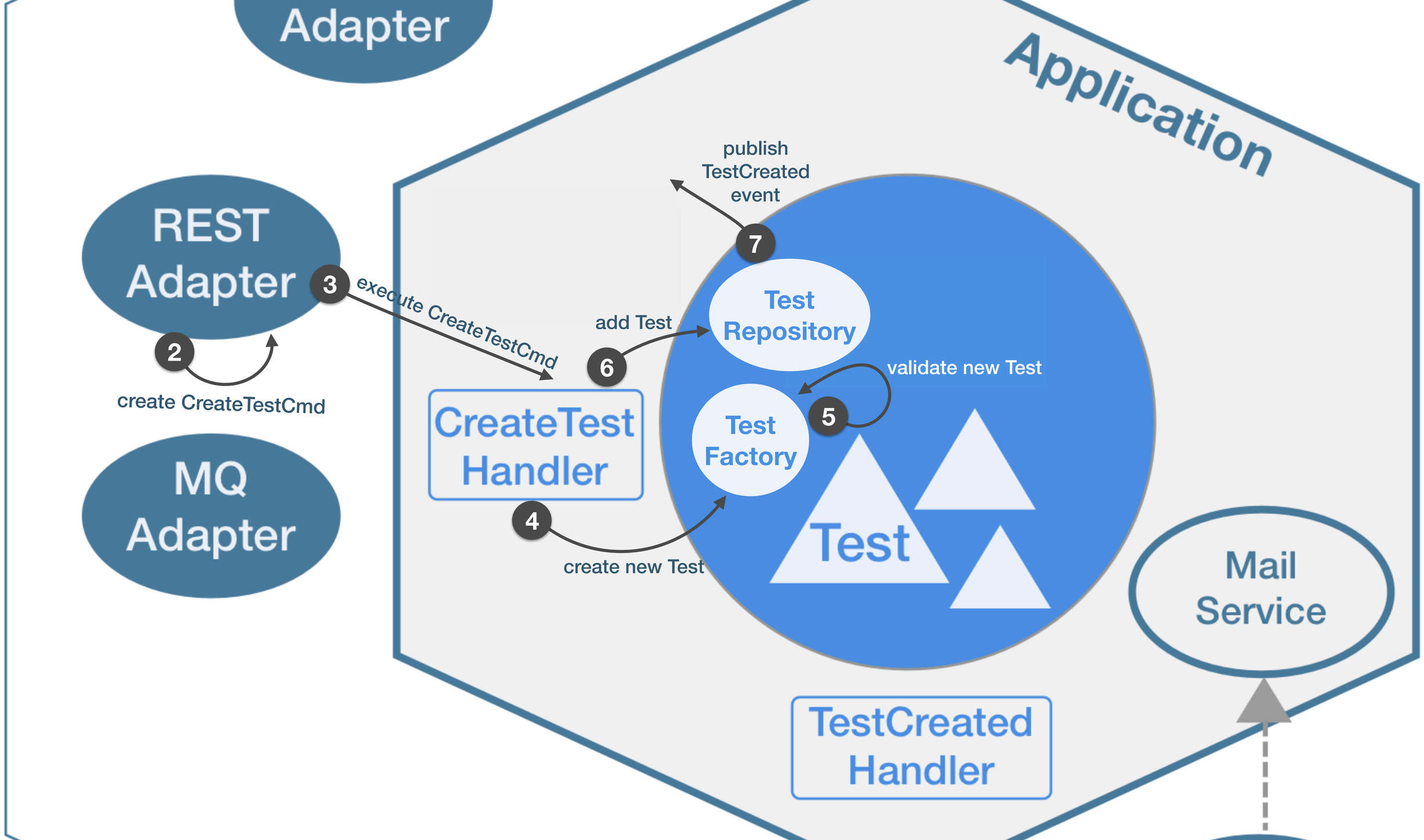

Adapter
REST
Adapter
2
create CreateTestCmd
MQ
Adapter
Application
Test
Repository
Test
Factory
Test
CreateTest
Handler
Mail
Service
TestCreated
Handler

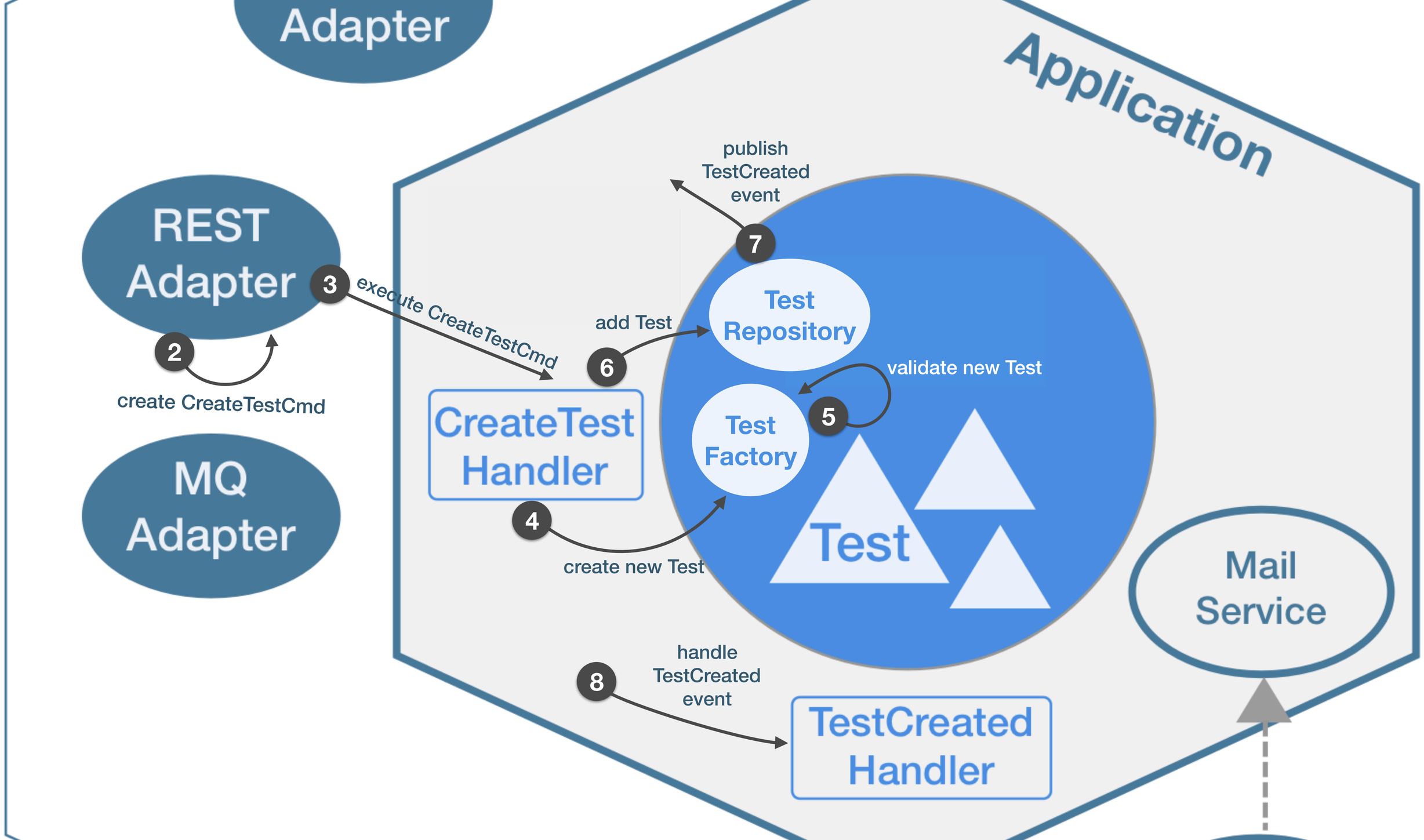

Adapter
Application
REST
Adapter
3
execute CreateTestCmd
2
create CreateTestCmd
MQ
Adapter
CreateTest
Handler
Test
Repository
Test
Factory
Test
Mail
Service
TestCreated
Handler

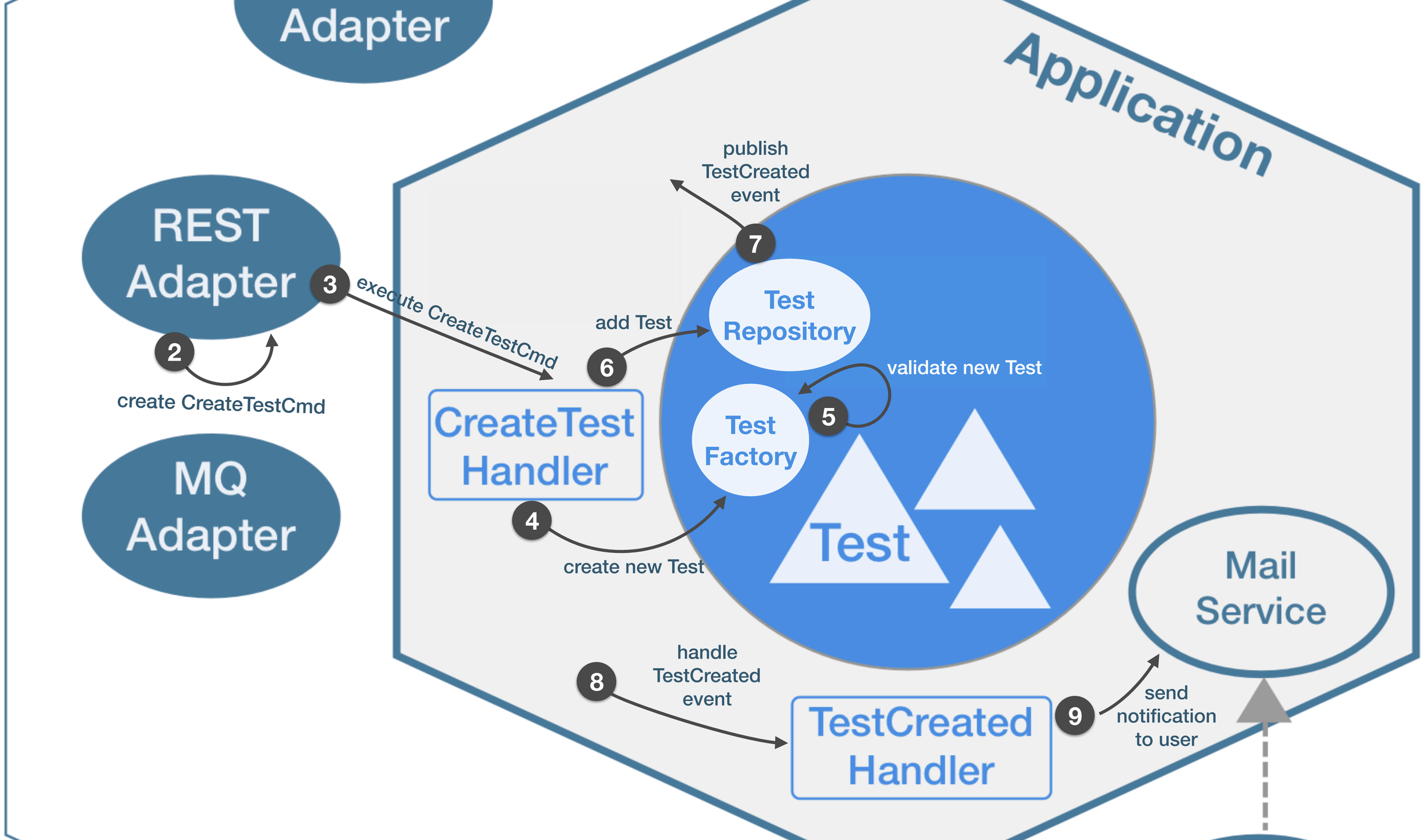

Adapter
Application
REST
Adapter
3
execute CreateTestCmd
2
create CreateTestCmd
MQ
Adapter
CreateTest
Handler
4
create new Test
Test
Repository
Test
Factory
Test
Mail
Service
TestCreated
Handler


Adapter
Application
REST
Adapter
3
execute CreateTestCmd
2
create CreateTestCmd
MQ
Adapter
CreateTest
Handler
4
create new Test
Test
Repository
validate new Test
Test
Factory
5
Test
Mail
Service
TestCreated
Handler


Adapter
Application
REST
Adapter
3
execute CreateTestCmd
2
create CreateTestCmd
MQ
Adapter
add Test
6
Test
Repository
validate new Test
5
Test
Factory
CreateTest
Handler
4
create new Test
Test
Mail
Service
TestCreated
Handler

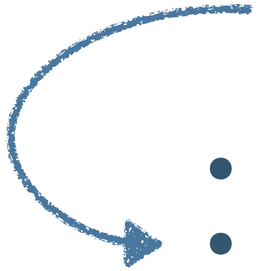

Adapter
Application
REST
Adapter
MQ
Adapter
2
create CreateTestCmd
3
execute CreateTestCmd
CreateTest
Handler
4
create new Test
add Test
6
publish
TestCreated
event
7
Test
Repository
validate new Test
5
Test
Factory
Test
Mail
Service
TestCreated
Handler


Adapter
Application
REST
Adapter
MQ
Adapter
2
create CreateTestCmd
3
execute CreateTestCmd
CreateTest
Handler
4
create new Test
add Test
6
publish
TestCreated
event
7
Test
Repository
validate new Test
5
Test
Factory
Test
Mail
Service
8
handle
TestCreated
event
TestCreated
Handler


Adapter
Application
REST Adapter
MQ Adapter
2
create CreateTestCmd
3
execute CreateTestCmd
CreateTest Handler
4
create new Test
add Test
6
publish TestCreated event
7
Test Repository
validate new Test
5
Test Factory
Test
8
handle TestCreated event
TestCreated Handler
9
send notification to user
Mail Service

Bounded Context + Hexagonal-Architektur

Gute Basis für Microservices



- **hohe organisatorische Isolation (Conway's Law)**
- **hohe technische Isolation (Aggregate als technische Transaktionsinseln)**

Event Storming

Knowledge Crunching

Inverse Conway

Process & Organization

Iterative Modelling

Domain Story Telling

Hexagonal Architektur

Event Sourcing

Architecture & Frameworks

Microservices

CQRS

DDD

Ubiquitous Language

Bounded Context

Core Domain

Strategic Design

Context Integration

Sub-Domain

Context Map

Aggregate

Entity

Value Object

Domain Service

Tactical Design

Factory

Application Service

Domain Event

Event Storming

Knowledge Crunching

Inverse Conway

Process & Organization

Iterative Modelling

Domain Story Telling

Hexagonal Architektur

Event Sourcing

Architecture & Frameworks

Microservices

CQRS

DDD

Ubiquitous Language

Bounded Context

Core Domain

Strategic Design

Context Integration

Sub-Domain

Context Map

Aggregate

Entity

Value Object

Domain Service

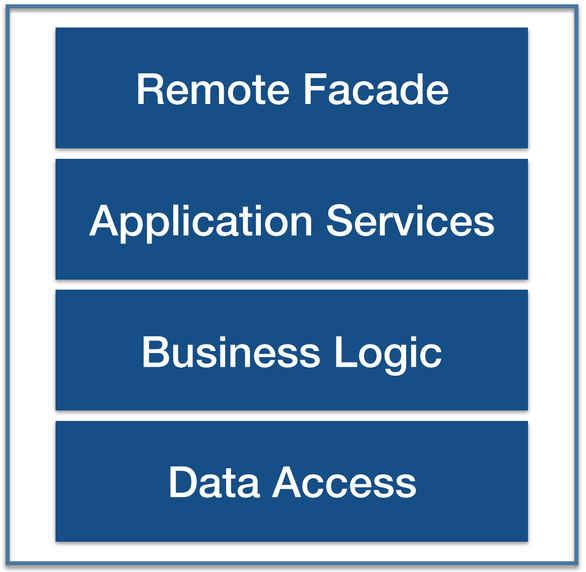
Tactical Design

Factory

Application Service

Domain Event

ANWENDUNG MIT "NORMALEM" SCHICHTENMODELL

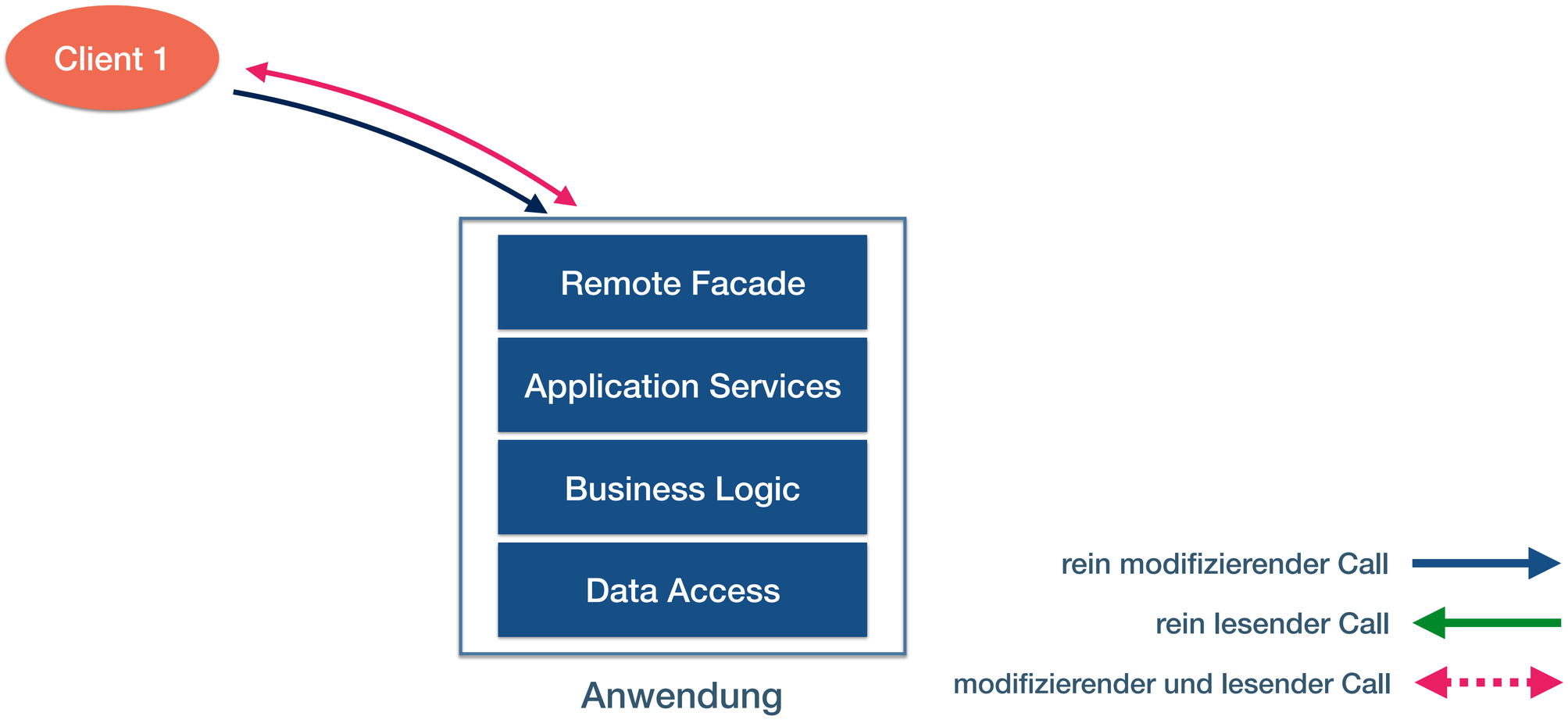


rein modifizierender Call

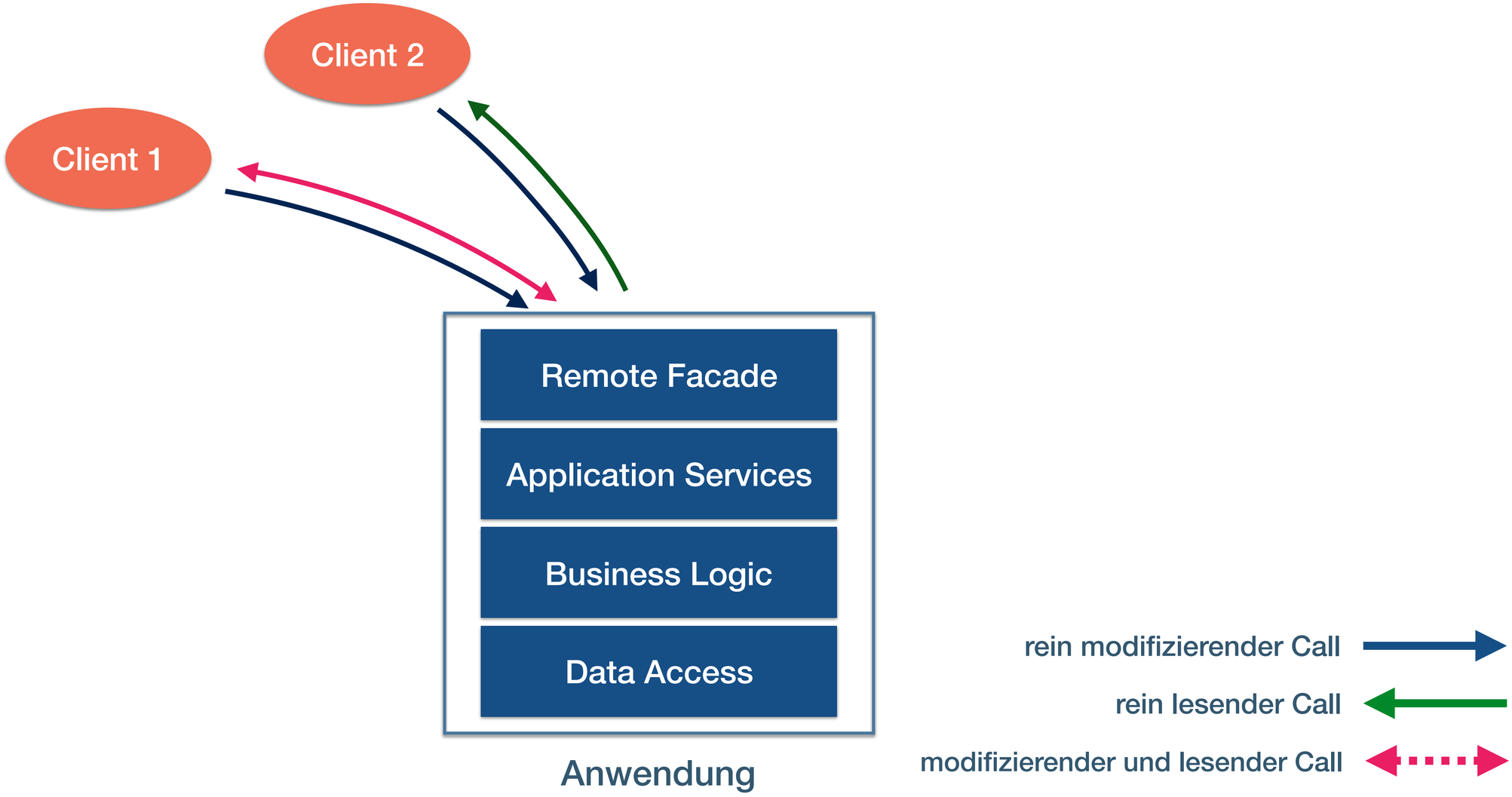
rein lesender Call

modifizierender und lesender Call

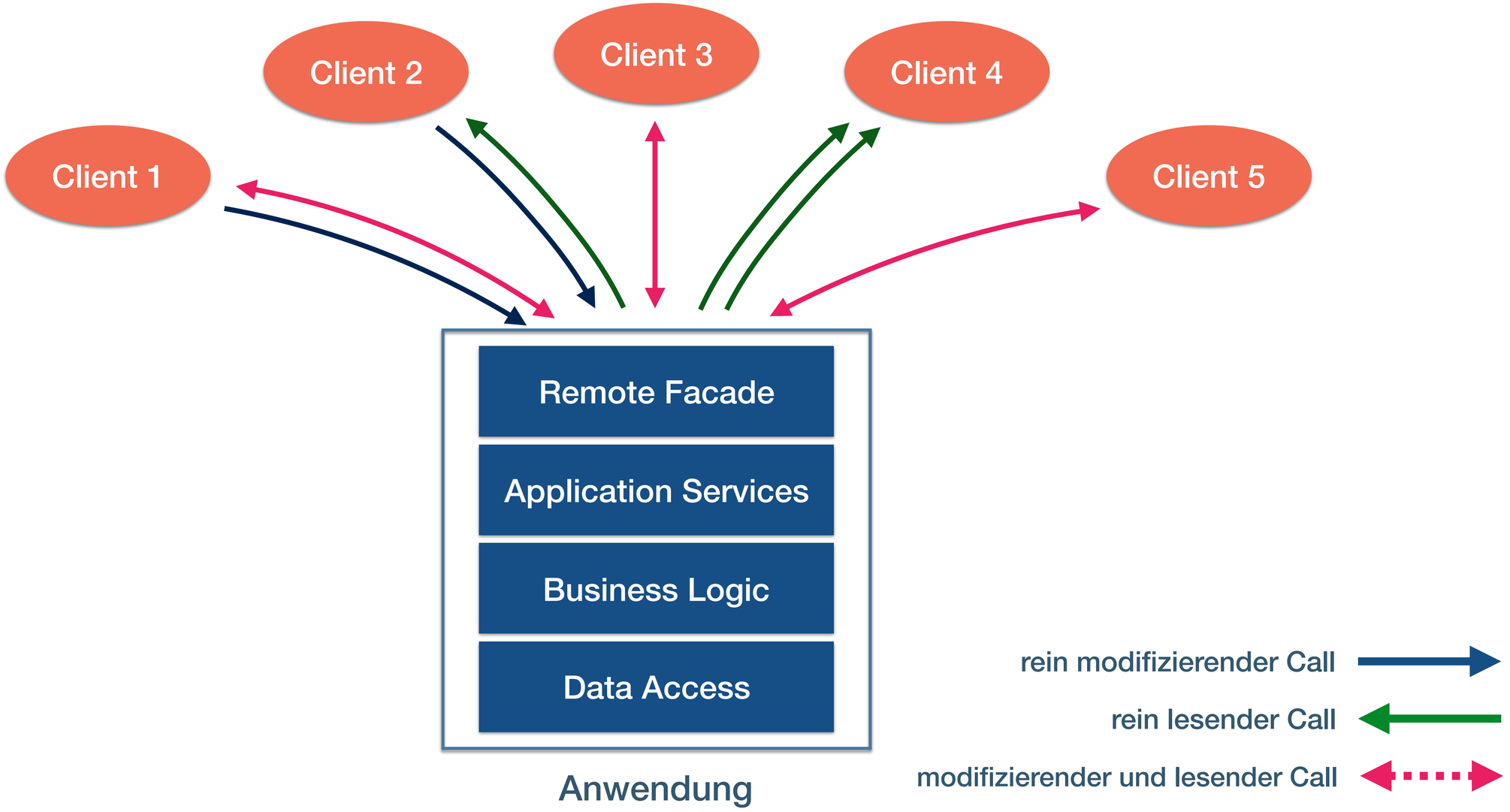
ANWENDUNG MIT "NORMALEM" SCHICHTENMODELL



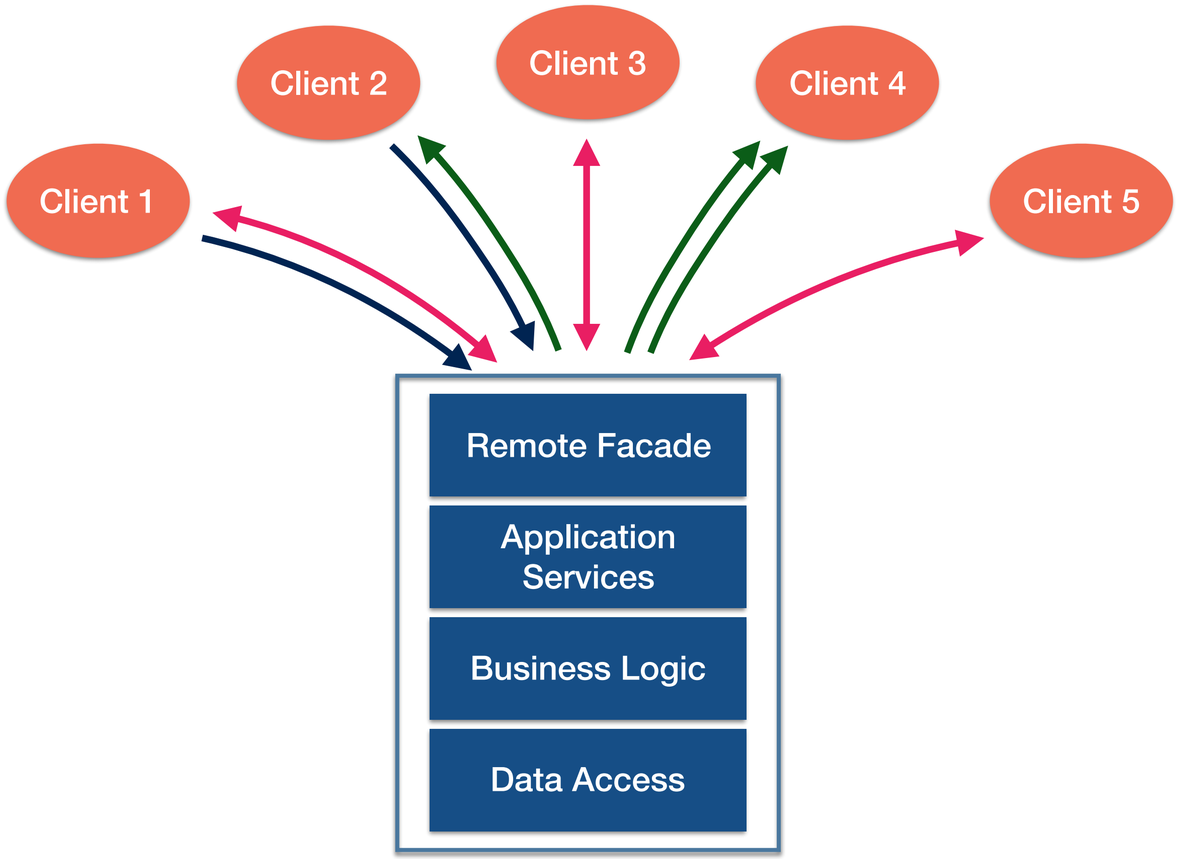
ANWENDUNG MIT "NORMALEM" SCHICHTENMODELL



ANWENDUNG MIT "NORMALEM" SCHICHTENMODELL

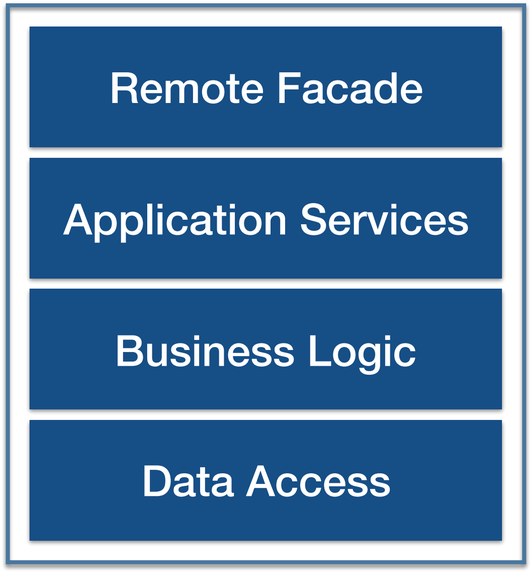


EIN "NORMALES" SCHICHTENMODELL: PROBLEME

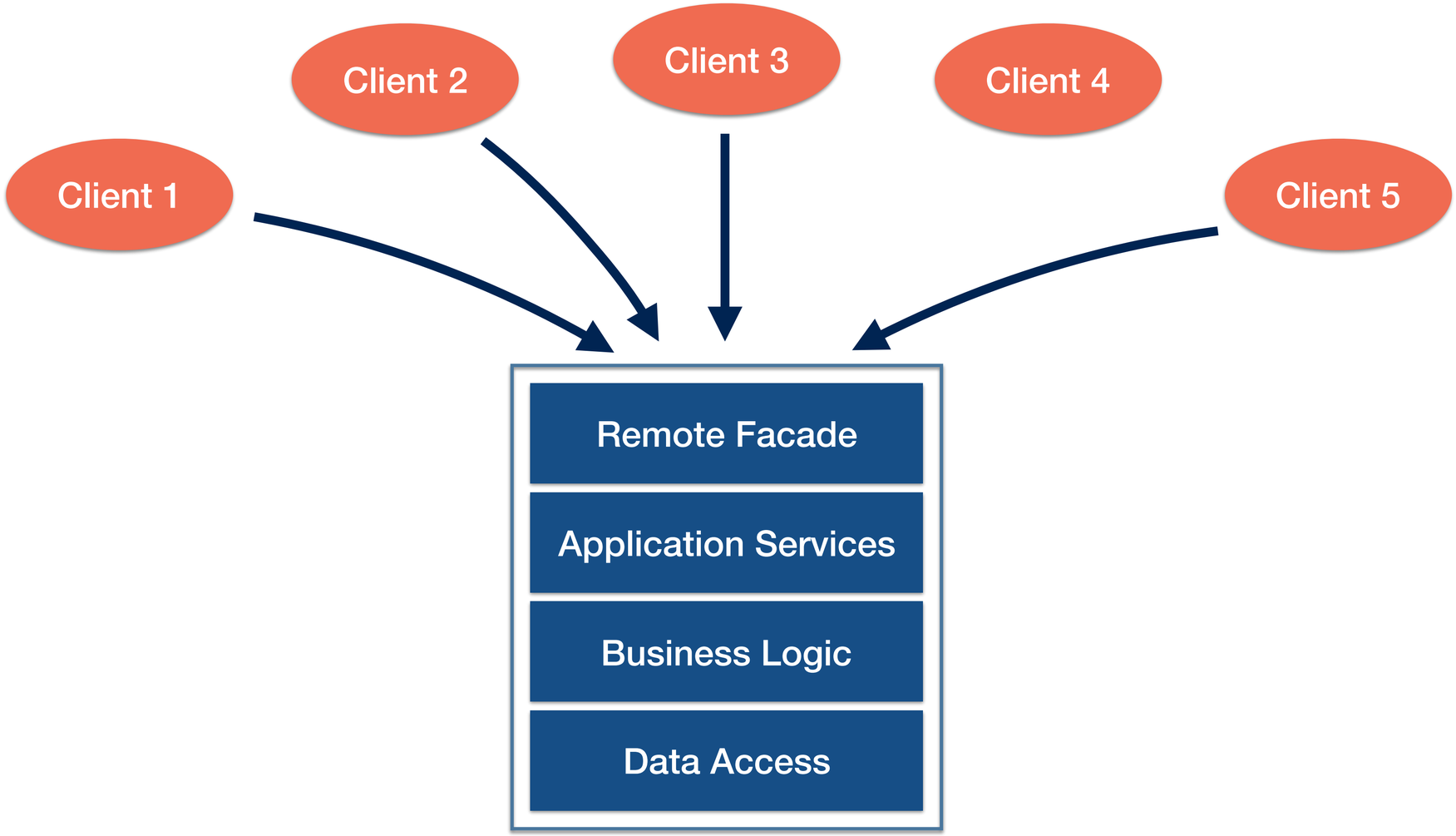


- "One size fits it all"
- Neue Clients benötigen oft neue Sichten auf Daten
- Inflation von DTOs und Schnittstellen
- Performance-Optimierung immer schwieriger wegen unterschiedlicher Anforderungen
- Changes am Model immer aufwändiger

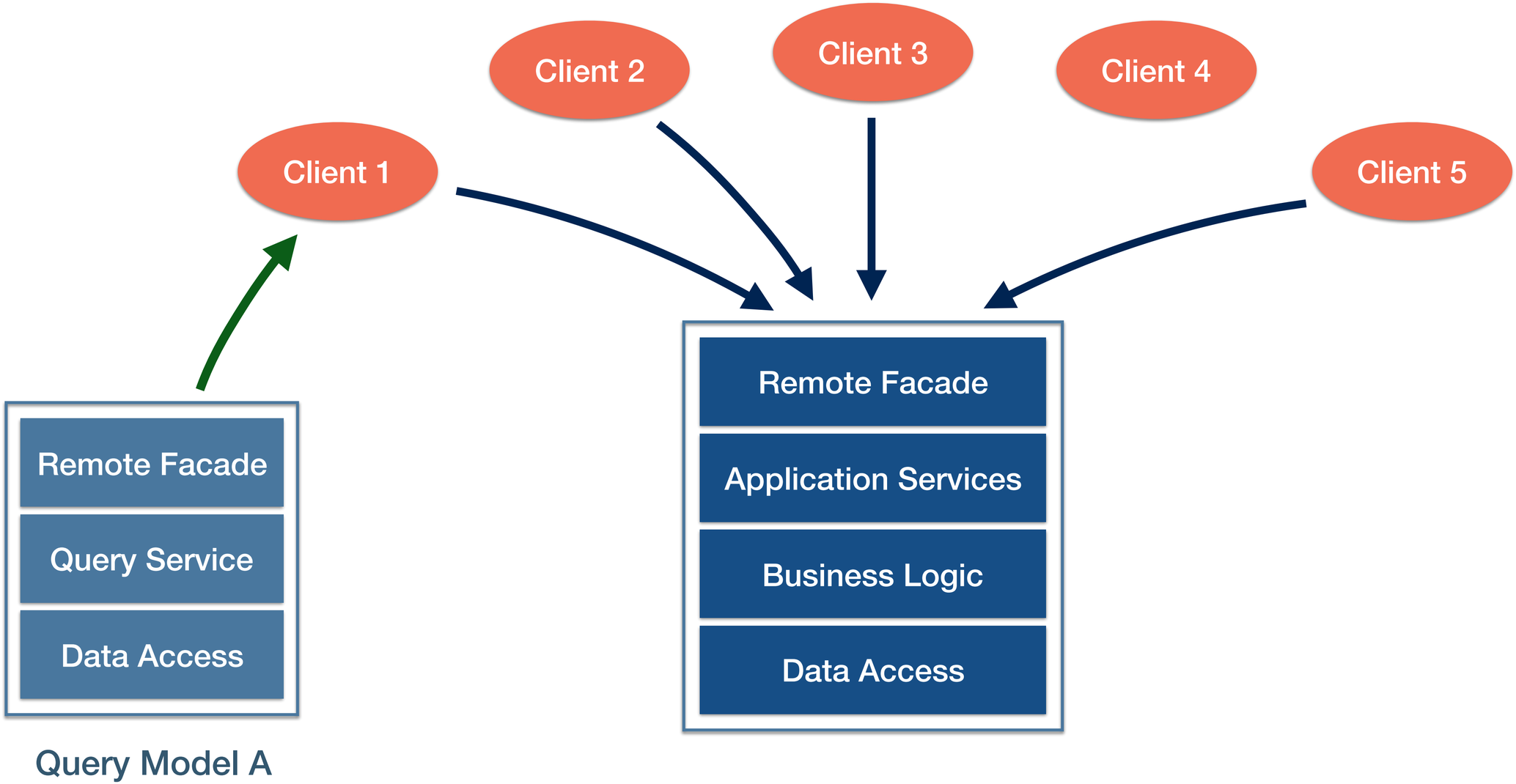
CQRS: Trennung von Command- und Query-Models



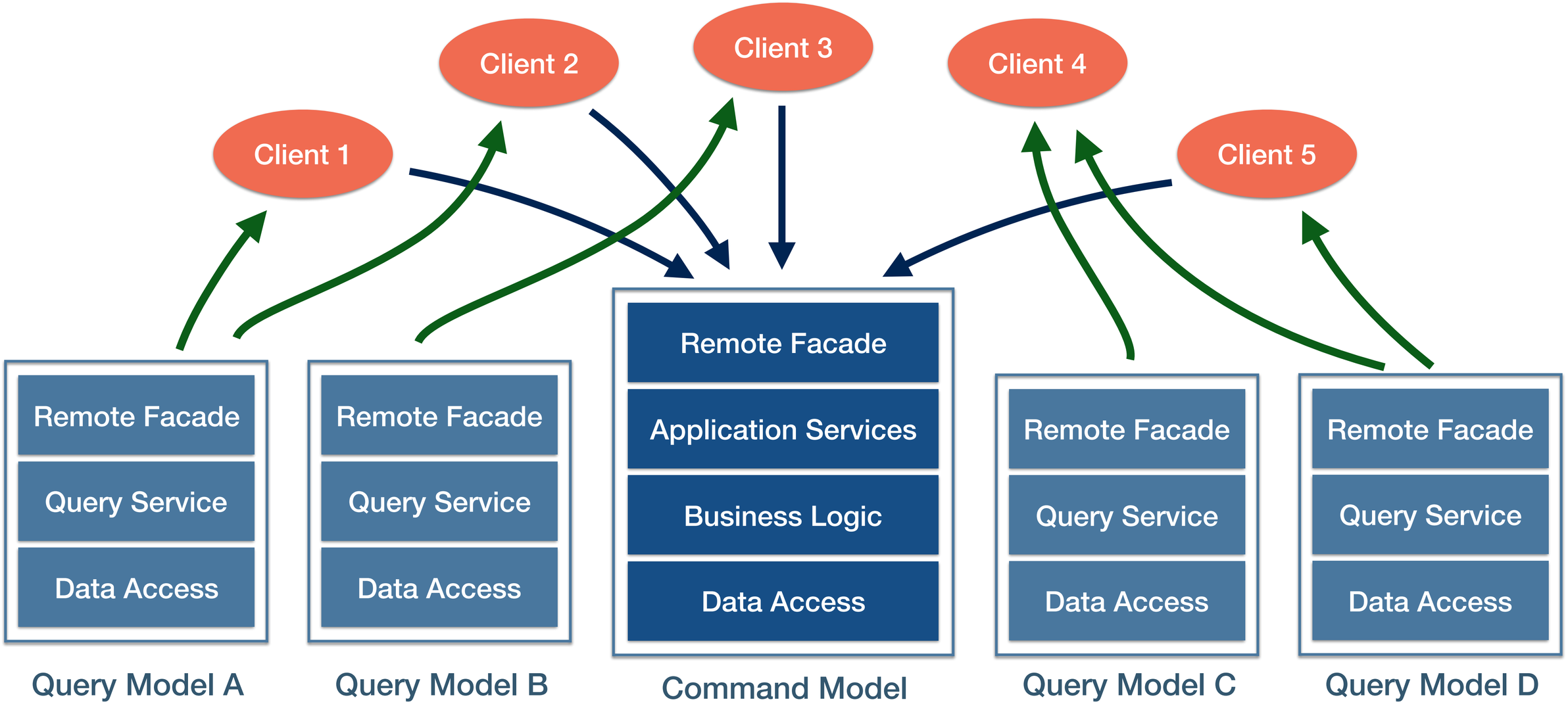
CQRS: Trennung von Command- und Query-Models



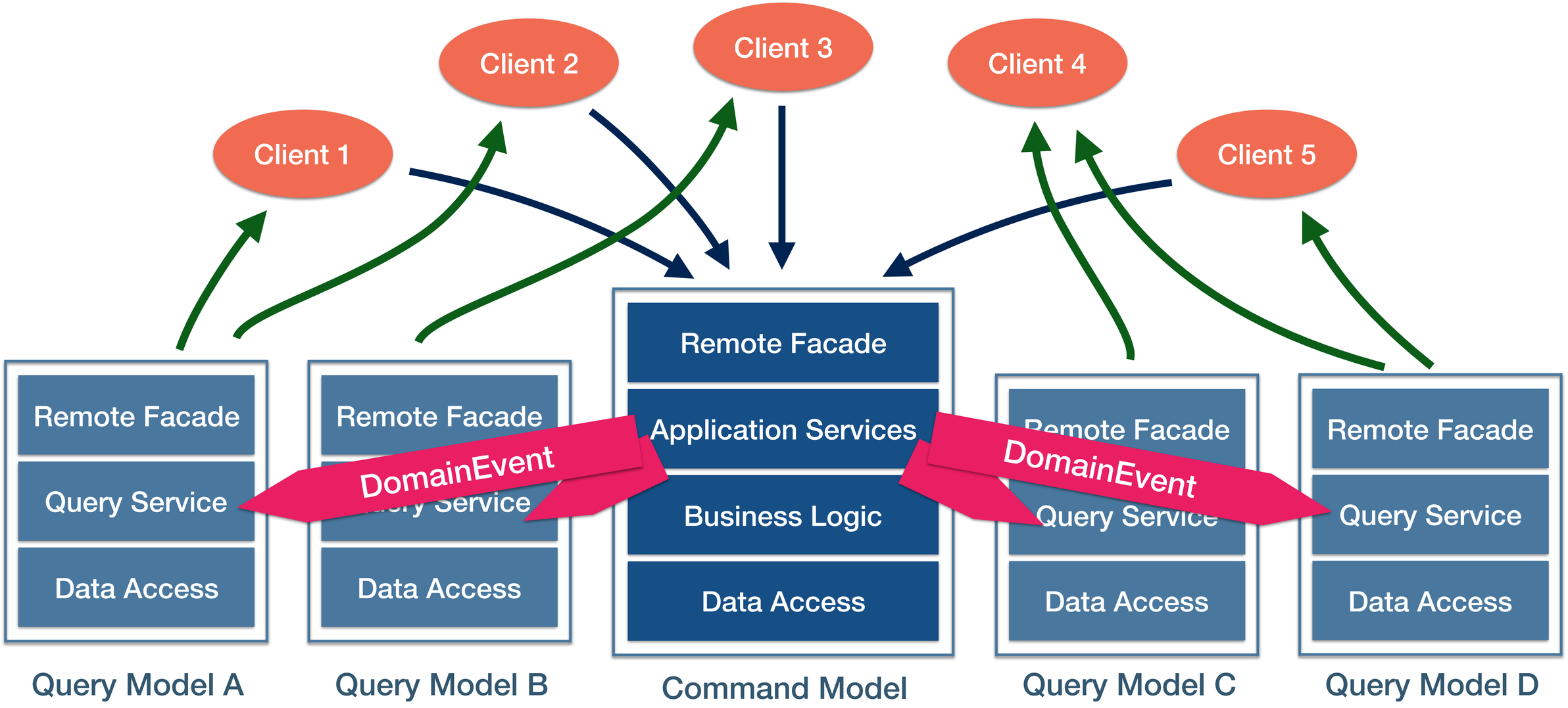
CQRS: Trennung von Command- und Query-Models



CQRS: Trennung von Command- und Query-Models



CQRS: Trennung von Command- und Query-Models



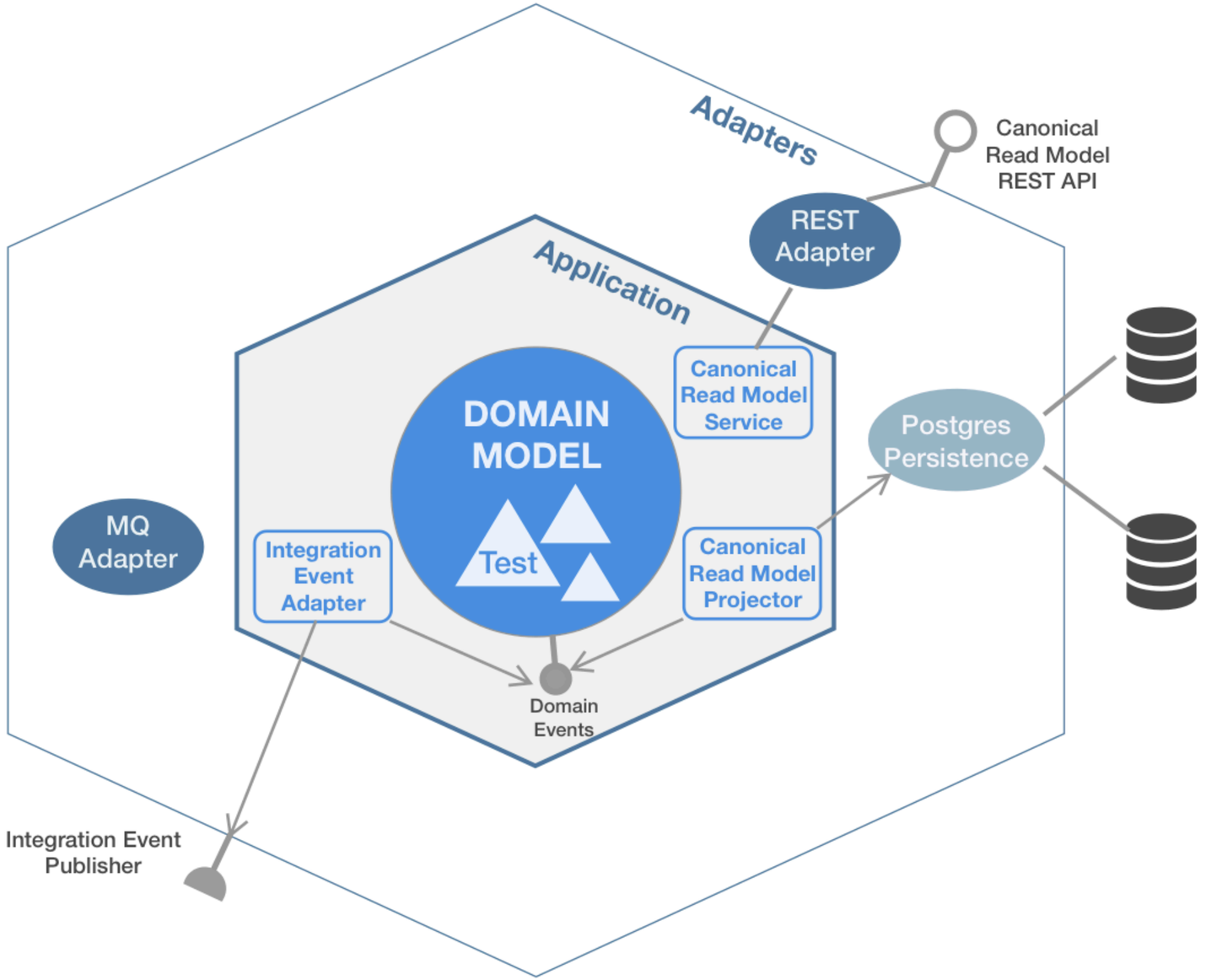
CQRS: Command / Query Responsibility Segregration

Vorteile

- Read-Models können extrem einfach sein, z.B.
  - vollständig **denormalisierte** SQL-Tabellen (bzw. Materialized Views)
  - ... oder Document Stores
- Read-Models erlauben **einfache Skalierung** und Performance-Optimierung

Nachteile

- möglicherweise **komplexe Mechanik**: Messaging, Apache Kafka etc.
- nur "**eventual consistency**": muss auf Applikations-Ebene kompensiert werden
- Full-Rebuild von Query-Models muss eingeplant werden und oft teuer (analog zu Seach-Feeder)

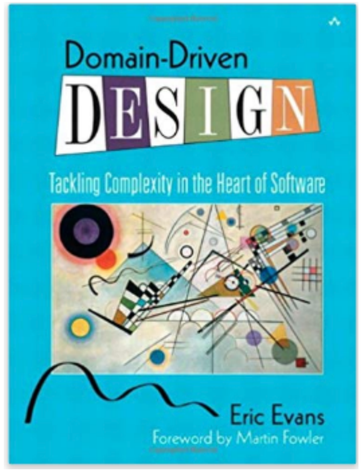

Adapters
Canonical
Read Model
REST API
REST
Adapter
Application
DOMAIN
MODEL
Test
Canonical
Read Model
Service
Canonical
Read Model
Projector
Postgres
Persistence
MQ
Adapter
Integration
Event
Adapter
Domain
Events
Integration Event
Publisher

WIE GEHTS WEITER?

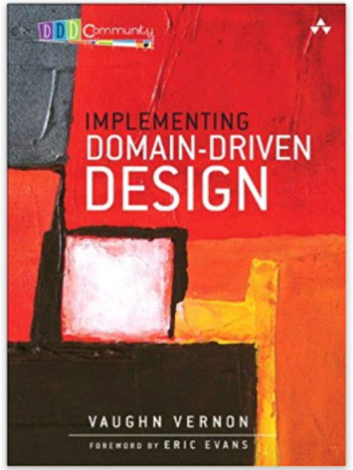
Functional DDD

DDD und Prozess

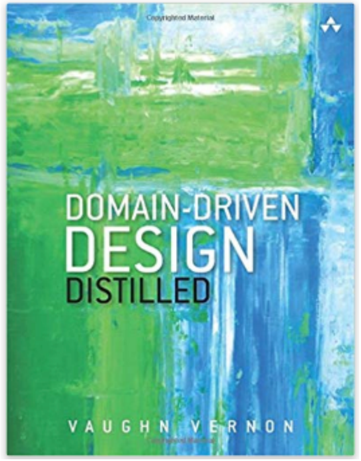
Event Sourcing


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

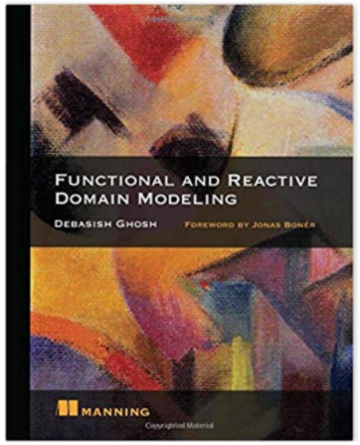
🤦‍♂️ 👨‍🔬


DDD Community
IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS

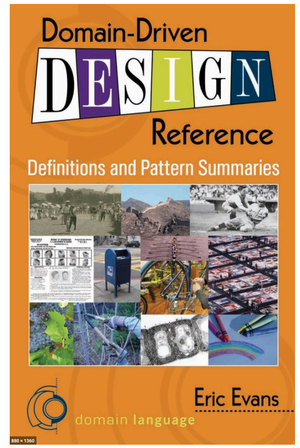
👍 🔧


DOMAIN-DRIVEN
DESIGN
DISTILLED
VAUGHN VERNON

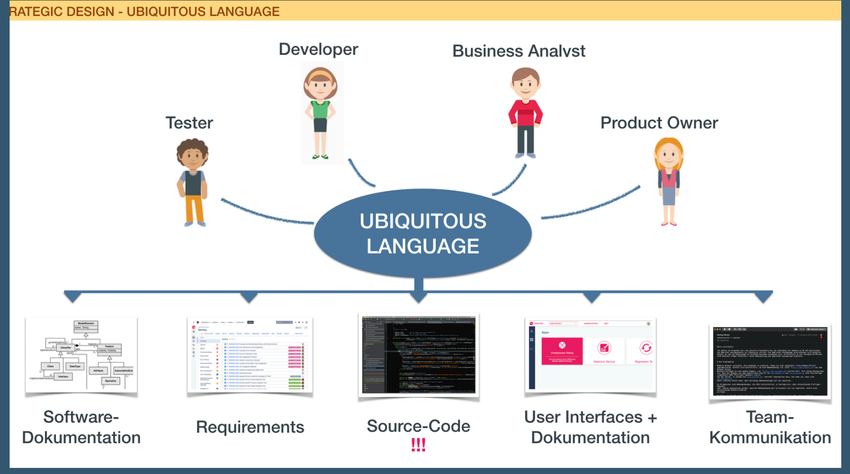
⏲️ 👍

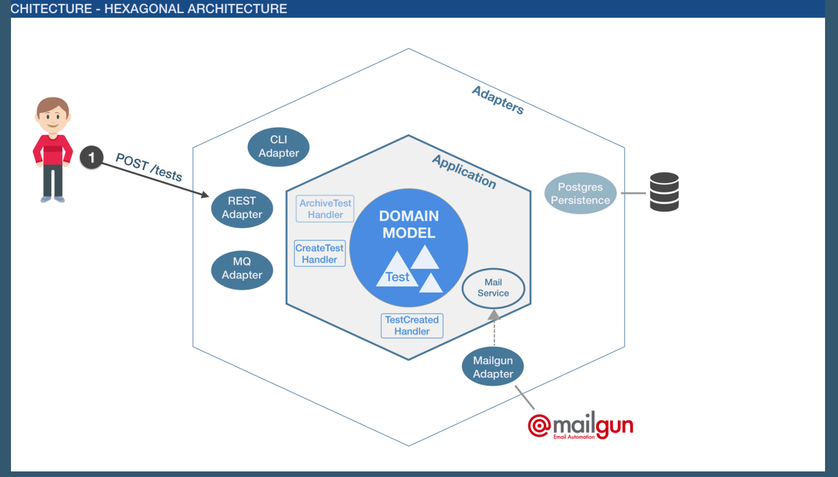

FUNCTIONAL AND REACTIVE
DOMAIN MODELING
DEBASISH GHOSH
FOREWORD BY JONAS BONÉR
MANNING

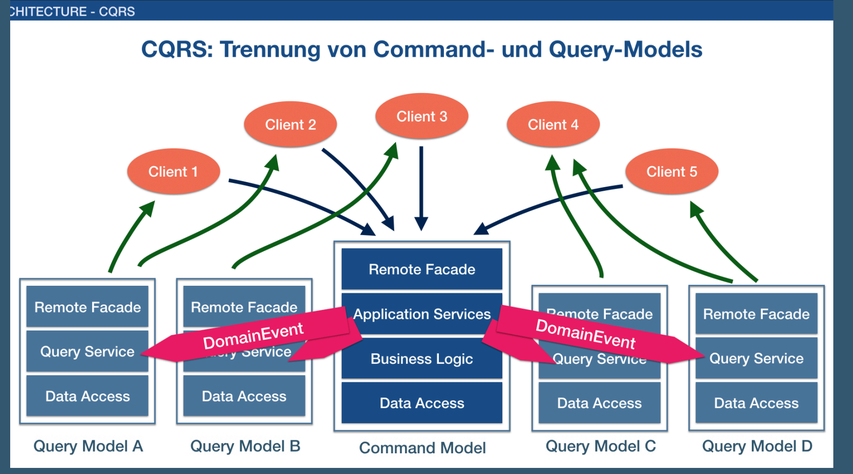
😍 👨‍🔬


Domain-Driven
DESIGN
Reference
Definitions and Pattern Summaries
Eric Evans
domain language

⏲️ 👍







DANKE FÜR EURE AUFMERKSAMKEIT :-)