Domain Driven Design

JUG Bulgaria

Nikolay Vasilev

Domain Driven Design

- Status Quo of building Enterprise Applications (EA)
- What is DDD?
- Strategic DDD
- Tactical DDD (a.k.a. DDD-Lite)
- Code / Example
- Questions
- Resources

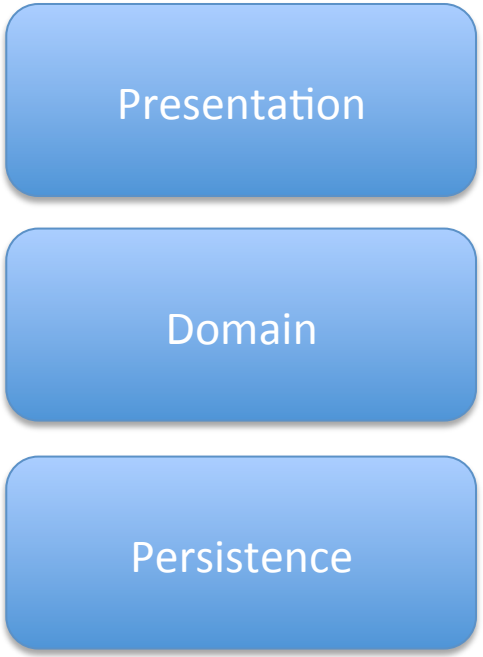
Status Quo of building Enterprise Applications (EA)

Status Quo of building EA

- How would you build an EA?

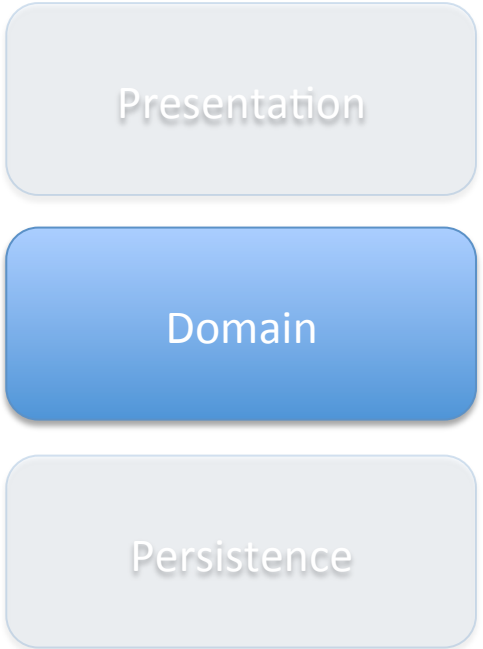
Status Quo of building EA

- How would you build an EA?
  - Layered Architecture



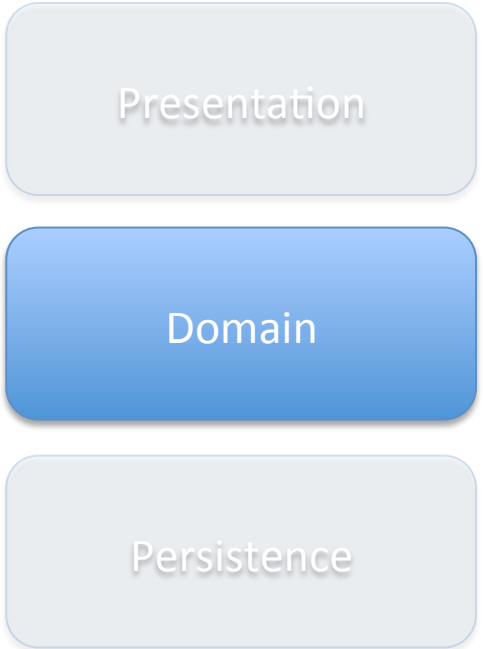
Status Quo of building EA

- How would you build an EA?
  - Layered Architecture
- Domain Layer: How to organize it?



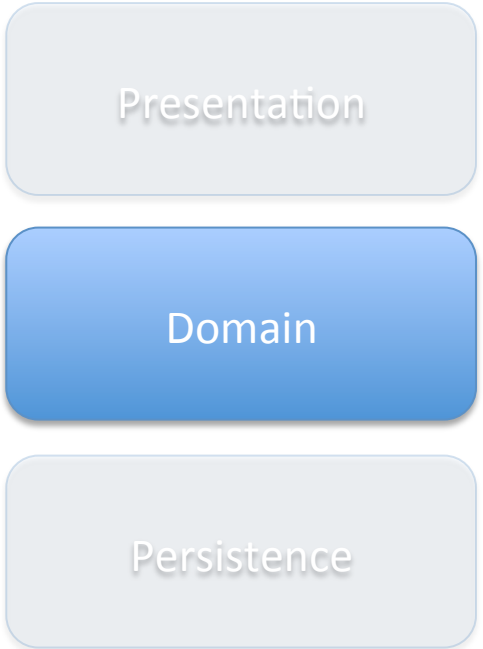
Status Quo of building EA

- How would you build an EA?
  - Layered Architecture
- Domain Layer: How to organize it?
- Three Standard Patterns

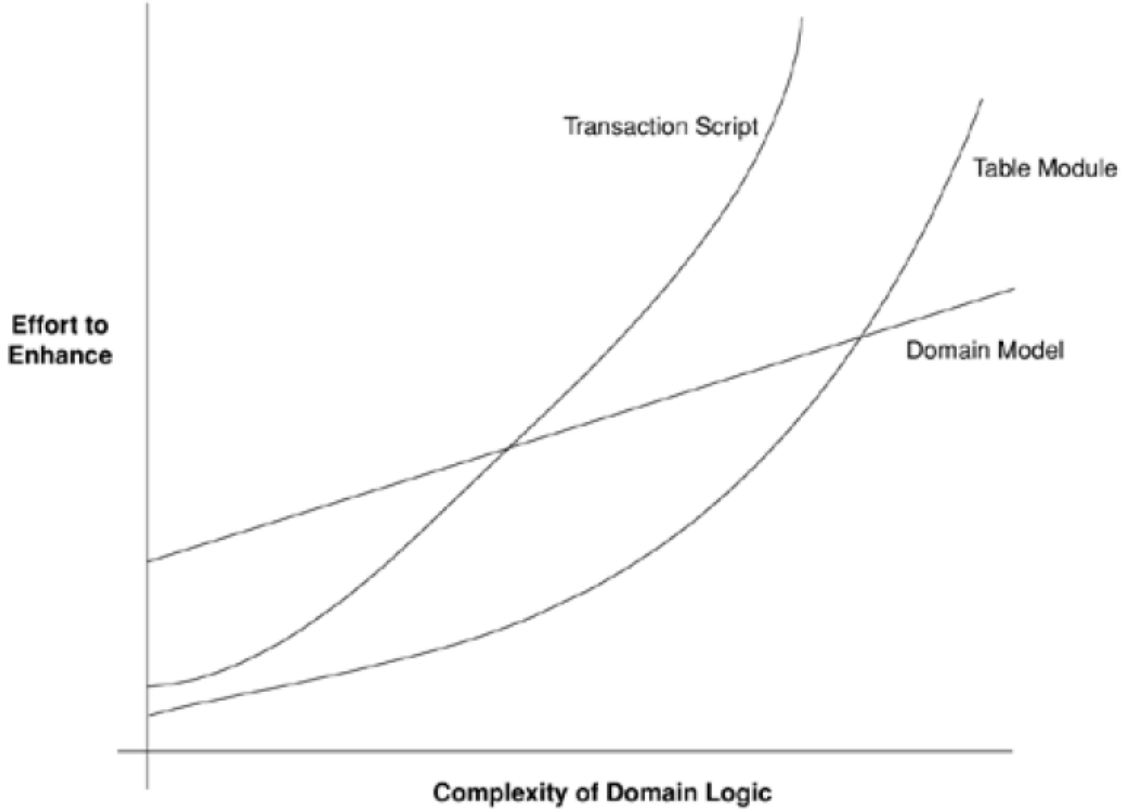


Status Quo of building EA

- Transaction Script
  - Express business logic via **procedures**
  - Easy to grasp and to avoid performance problems
- Domain Model
  - **Object model** of domain, having both – behaviour and data
  - Expressive but elaborated
- Table Module
  - middle ground between the two above
  - one class per db table with **procedures** over the data



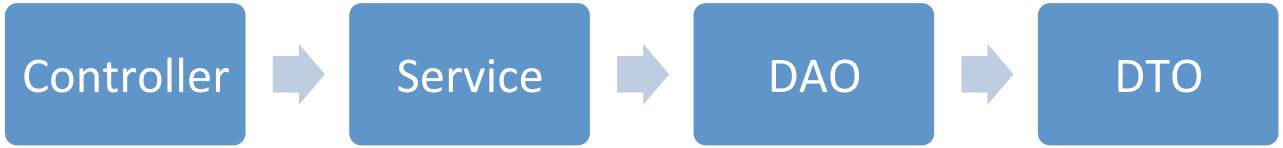
Status Quo of building EA



Source: [Fowler, PoEAA]

Status Quo of building EA

- How Spring MVC does the work?



- The result:
  - Thinking in technical terms, not domain's one
  - Anemic Domain Model (POJO's with no behaviour)
  - Boilerplate Services layer with too many responsibilities
  - No separation of (domain) concerns

- Accept it my friend, it's pure Transaction Script in action!
  - i.e. Procedural Programming disguised as OOP

What is DDD?

What is DDD?

- What is DDD?
  - Not a Technology or Methodology
  - Set of principles and patterns for focusing the design effort where it matters most

What is DDD?

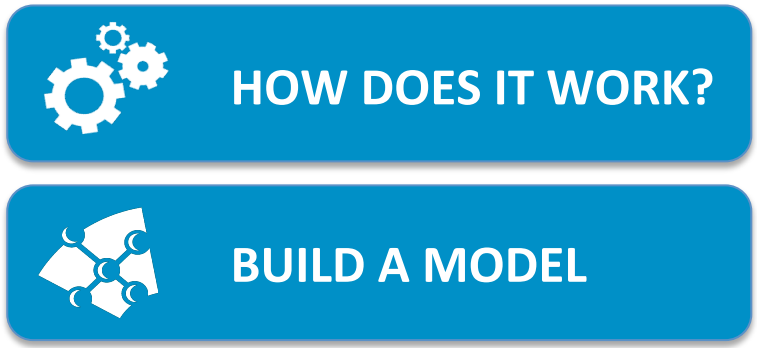
- What is DDD?
  - Not a Technology or Methodology
  - Set of principles and patterns for focusing the design effort where it matters most

- It's all about…
  - Understanding of the **domain** (subject area) where the software will be applied

What is DDD?

- What is DDD?
  - Not a Technology or Methodology
  - Set of principles and patterns for focusing the design effort where it matters most

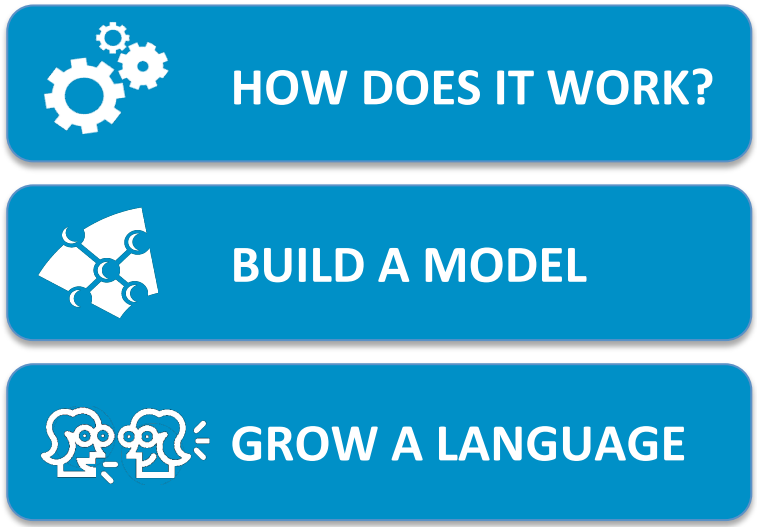
- It's all about…
  - Understanding of the **domain** (subject area) where the software will be applied
  - Create highly expressive **model** of that domain



What is DDD?

- What is DDD?
  - Not a Technology or Methodology
  - Set of principles and patterns for focusing the design effort where it matters most

- It's all about…
  - Understanding of the **domain** (subject area) where the software will be applied
  - Create highly expressive **model** of that domain
  - Distil **Ubiquitous Language**



What is a Model?

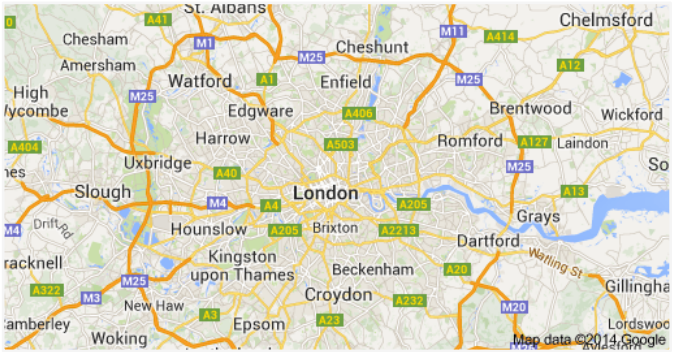
- What is a Model?

What is a Model?

- What is a Model?
  - Simplification of reality

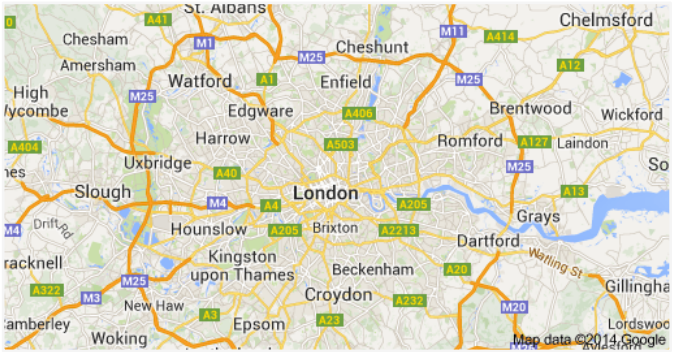
What is a Model?

- What is a Model?
  - Simplification of reality
  - Shows some aspect of reality…



What is a Model?

- What is a Model?
  - Simplification of reality
  - Shows some aspect of reality…
  - … or an idea that is of interest



What is a Model?

- How do you represent your Model?

(UML) DIAGRAMS?

What is a Model?

- How do you represent your Model?

Ingest engineer manages brands
As an ingest engineer
I want to ingest brands, series and episodes
So that users could browse them

TEXT?

Scenario 1: Episodes Ingestion
Given a brand with one series
When I ingest an episode
Then its gets ready for to be seen

Scenario 2: Episodes are seen
Given a brand with 1 series and an episode
When an user request episode content
Then she is able to see it

What is a Model?

- How do you represent your Model?

```
@Test
public void shouldValidateEpisodeAvailability() {
    AvailabilityWindow availabilityWindow =
                new AvailabilityWindow(yesterday, tomorrow);
    Episode episode = new Episode();
    episode.withAvailabilityWindow(availabilityWindow);
    assertTrue("Episode is not availabile.", episode.isAvailable(today));
}
```

AUTOMATED TESTS?

What is a Model?

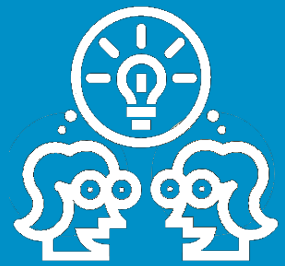
- How do you represent your Model?

```
class Episode {
    private AvailabilityWindow availabilityWindow;
    // ...
    public void withAvailabilityWindow(AvailabilityWindow availabilityWindow) {
        this.availabilityWindow = availabilityWindow;
    }
    public boolean isAvailable(DateTime date) {
        return availabilityWindow.isWithin(date);
    }
}

class AvailabilityWindow {
    private DateTime startDate;
    private DateTime endDate;

    public AvailabilityWindow(DateTime startDate, DateTime endDate) {
        this.startDate = startDate; this.endDate = endDate;
    }
    public boolean isWithin(DateTime date) {
        return startDate.isBefore(date) && endDate.isAfter(date);
    }
}
```

CODE?

What is a Model?

- The model is…



THE MENTAL REPRESENTATION

- Everything else is…

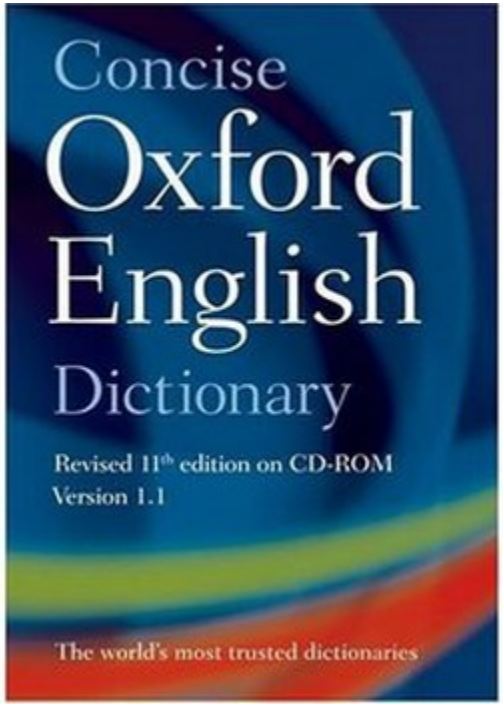
JUST COMMUNICATION TOOL

Elaborate the Model

A COLLABORATIVE EXERCISE

Elaborate the Model

BASED ON A COMMON LANGUAGE



Elaborate the Model

Ingest engineer manages brands
As an ingest engineer
I want to ingest brands, series and episodes

```
@Test
public void shouldValidateEpisodeAvailability() {
    AvailabilityWindow availabilityWindow =
                new AvailabilityWindow(yesterday, tomorrow);
    Episode episode = new Episode();
    episode.withAvailabilityWindow(avail
    assertTrue("Episode is not availabil
}
```

```
class Episode {
    private AvailabilityWindow availabilityWindow;
    // ...
    public void withAvailabilityWindow(AvailabilityWindow availabilityWindow) {
        this.availabilityWindow = availabilityWindow;
    }
    public boolean isAvailable(DateTime date) {
        return availabilityWindow.isWithin(date);
```

EXPRESSED AT ALL LEVELS

Elaborate the Model

EVOLUTIONARY PROCESS

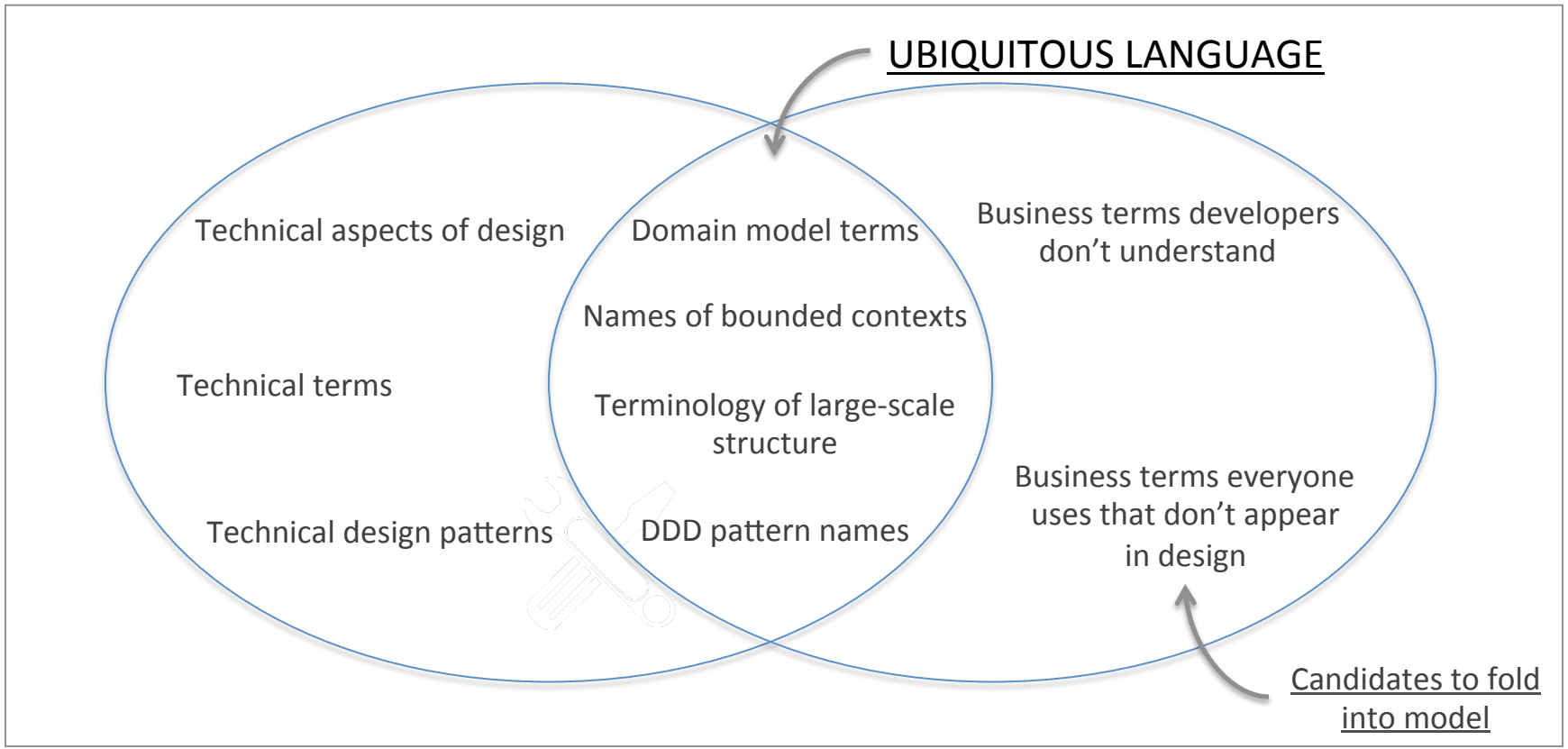
Elaborate the Model

EXPERIMENTING IS ESSENTIAL

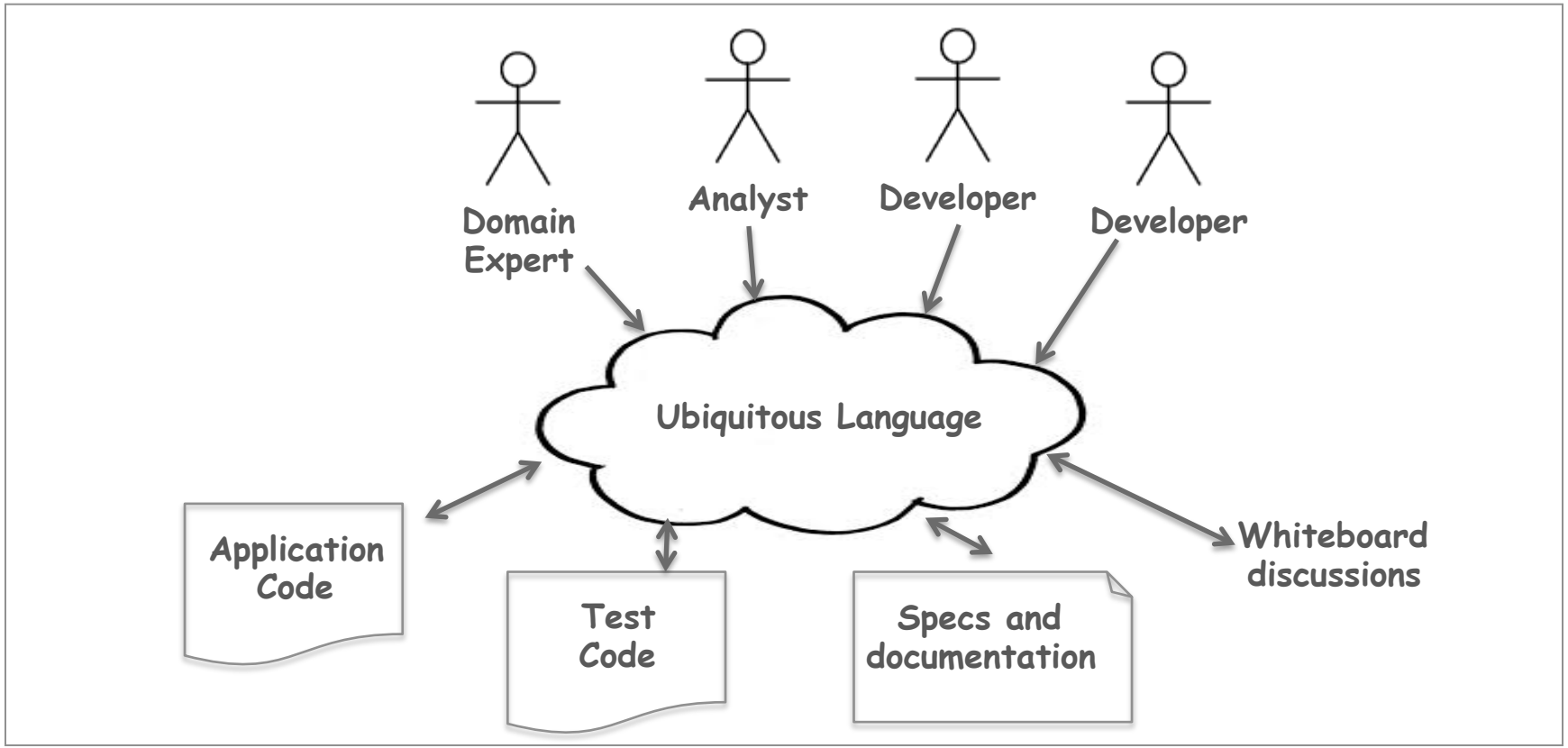
Elaborate the Model

SO DO AUTOMATED TESTING

Ubiquitous Language



Ubiquitous Language



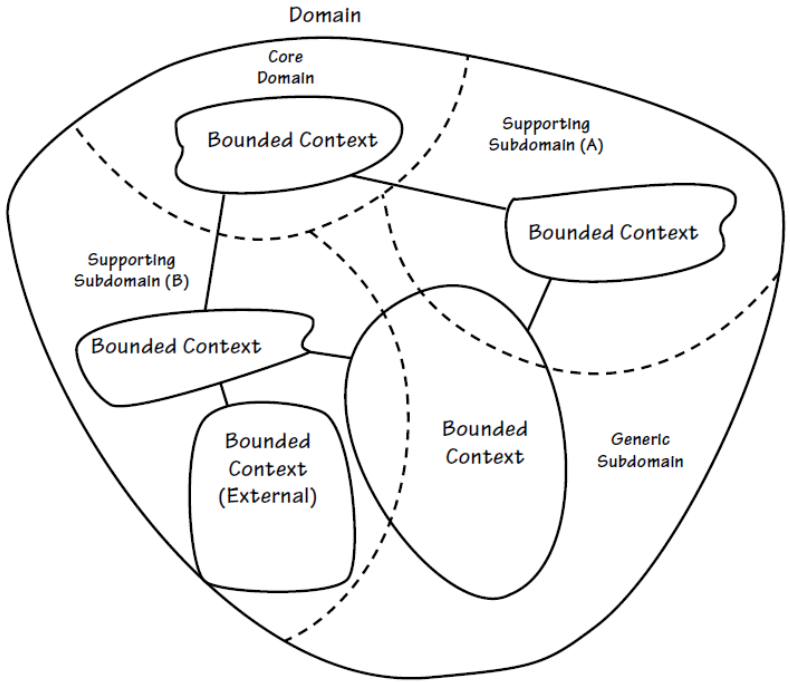
Strategic DDD

Domain Modelling

- Typical system today
  - Consist of a few subsystems, responsible for multiple functions
  - Software/Domain concerns are not clearly separated
- Which functions make business successful?
  - Think about each of them separately
  - Do **NOT** create one model of the whole domain
  - Otherwise everything will be connected to and depend on everything else

Domain Modelling – Basic Terms

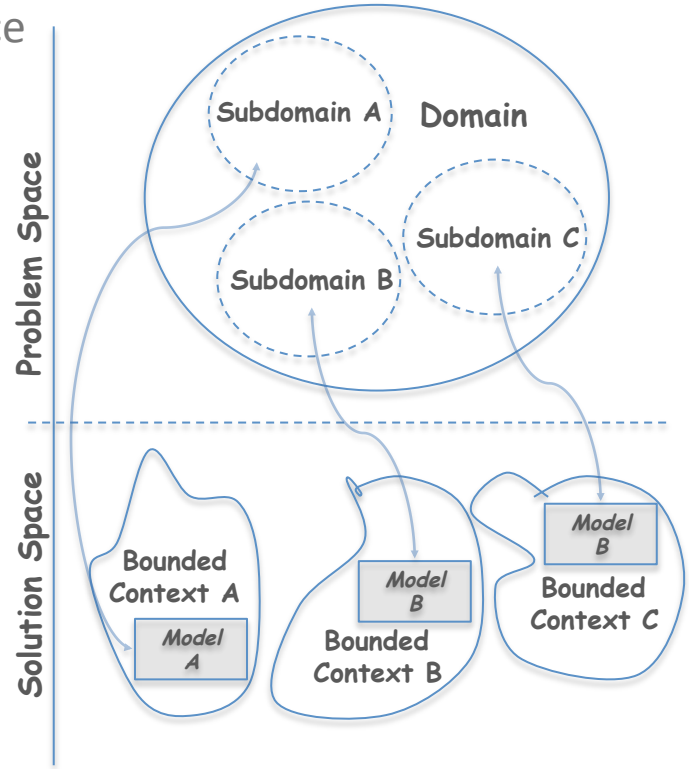
- Domain
  - Subject area where the software will be applied
  - Example: VOD Domain
- Subdomain
  - Logically separated part of the Domain
  - Example: Ingestion, Streaming, Geo Locatio
- Domain Model
  - Software model for solution of a domain problem
- Bounded Context
  - Explicit boundary where Domain Model lives



Source: [Vernon, DDD]

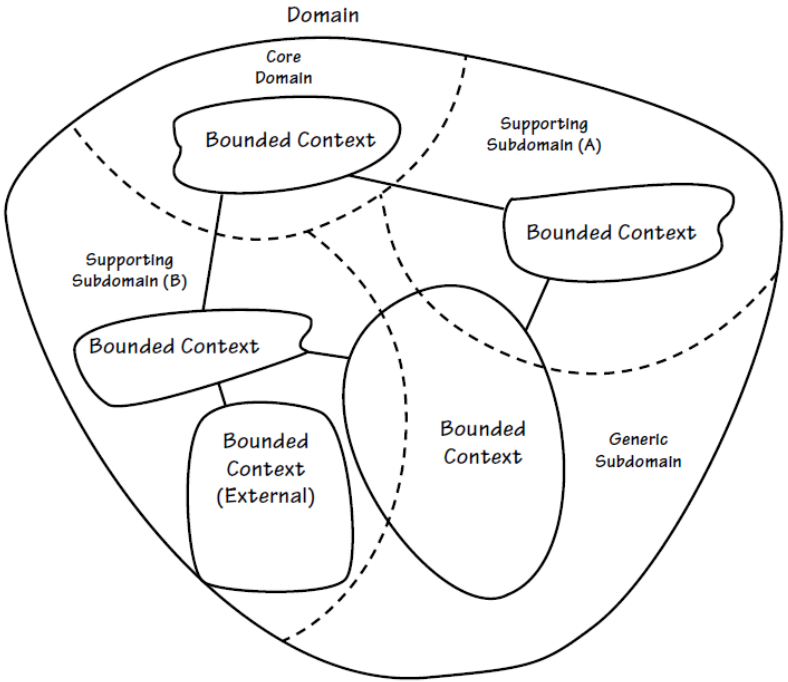
Problem-Solution Perspective

- Domains have problems space and solution space
- Problem Space
  - Strategic business challenge to be solved
  - Separate it into Subdomains
- Solution Space
  - Implementation of the software to solve the business problem
  - Bounded Context is a specific solution
  - The model lives in the boundaries of Bounded Context
- When you have good understanding of the problem space, turn it into solution space



Subdomains

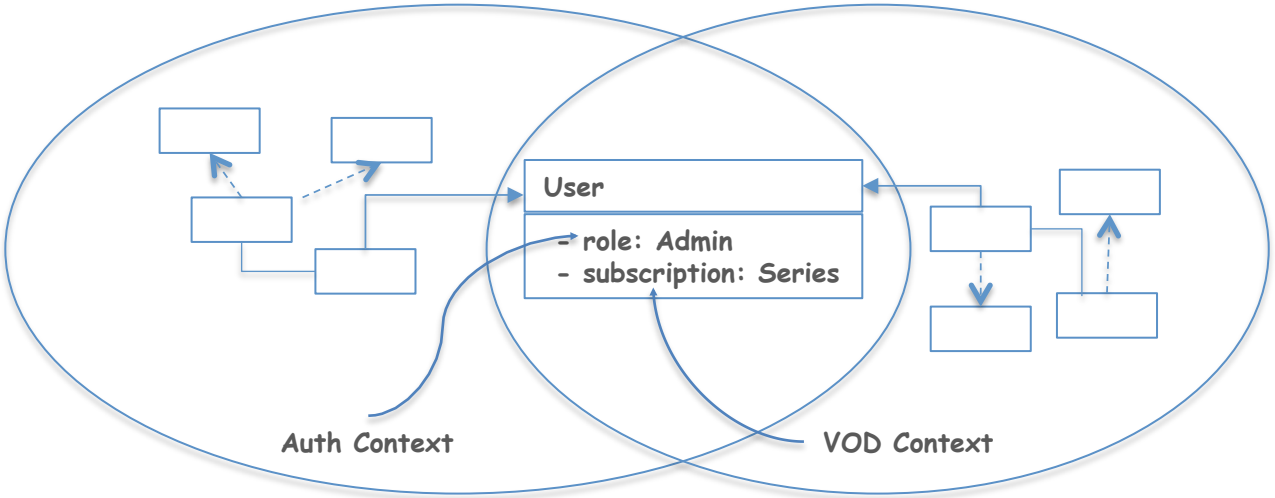
- Core Domain
  - part of the business that is of primary importance
  - Example: Asset Metadata Delivery
- Supporting Domain
  - models some aspect of the business that is essential, yet not Core
  - Example: Asset Streaming Info
- Generic Domain
  - captures nothing special to the business, yet is required
  - Example: GeoLocation



Source: [Vernon, DDD]

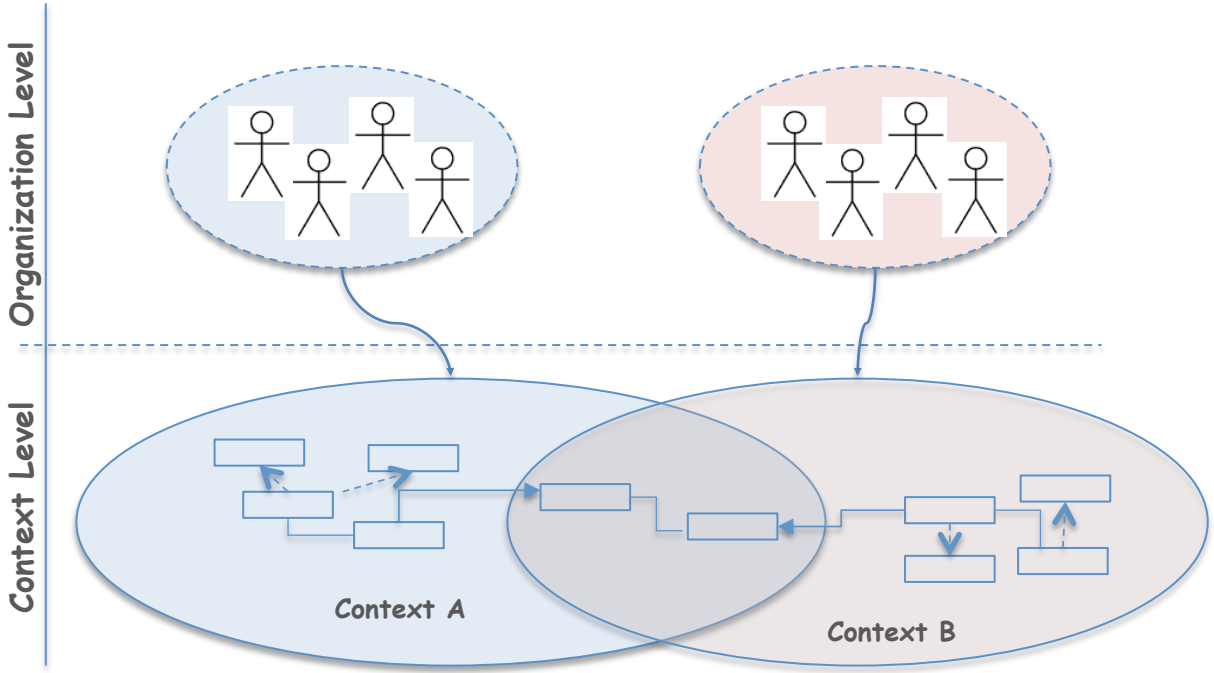
Bounded Contexts

- Models are developed in Bounded Contexts
- Linguistic boundary in which the model exists
  - Mark off where the meaning of every term used by the domain model is well understood
  - Example: User (Auth Context: Role) and User (VOD Context: Content Customer)



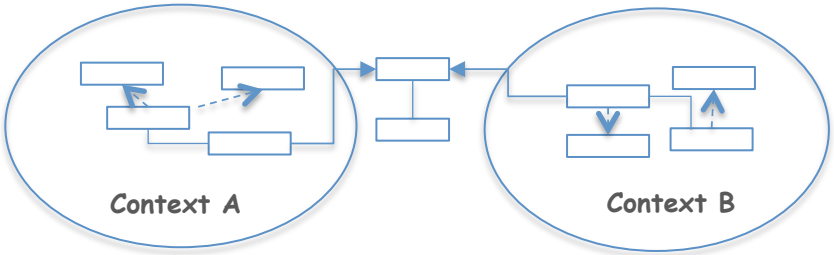
Bounded Contexts

- Attempt to align subdomains one-to-one with Bounded Contexts
- Could influence how teams are organized

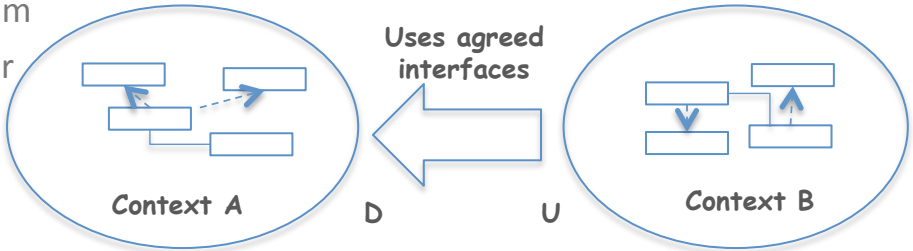


Context Mapping

- Mapping the contact points and translations between Bounded Contexts
- Context Mapping Patterns
  - Shared Kernel
    - Shared Subset of Domain
    - Example: Money

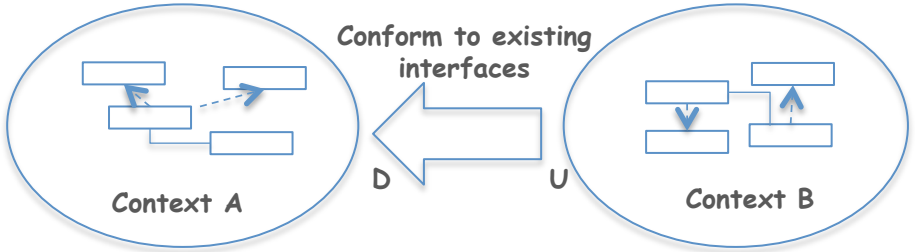


  - Customer/Supplier
    - Upstream Project – Supplier Team
    - Downstream Project – Consumer Team
    - Acceptance test at Upstream context

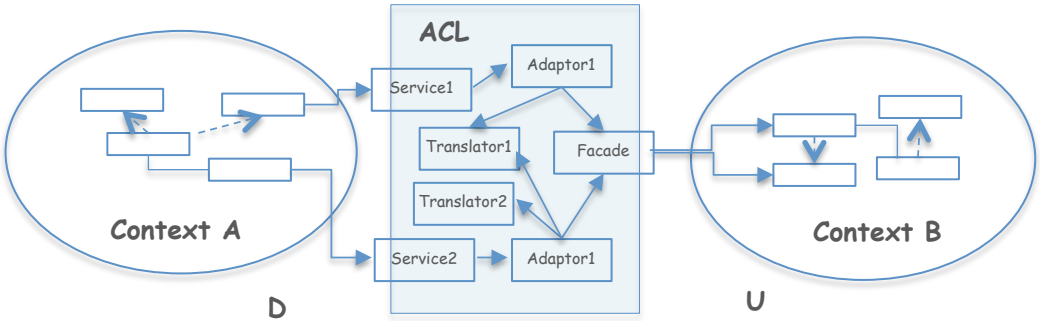


Context Mapping

- Context Mapping Patterns
  - Conformist

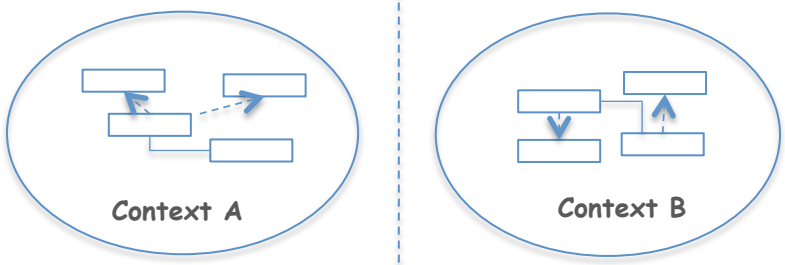


  - Anti-Corruption Layer (ACL)
    - Isolating Layer



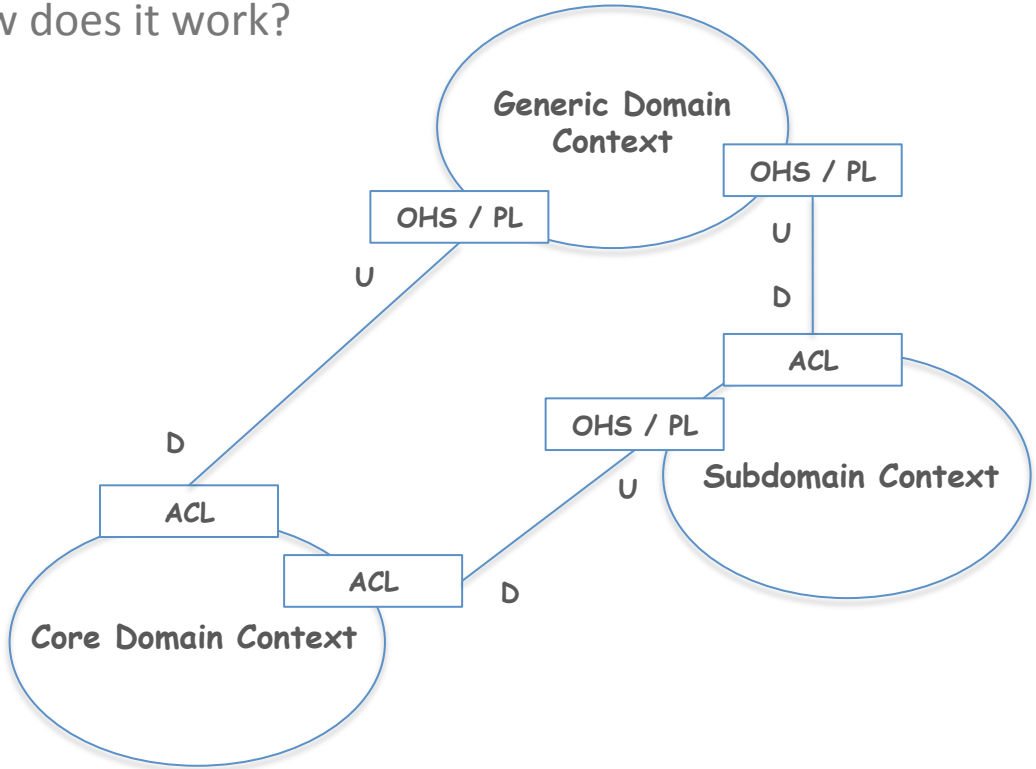
Context Mapping

- Context Mapping Patterns
  - Separate Ways
  - Open Host Service (OHS)
    - a.k.a. “Remote Façade”
    - Example: REST interface
  - Published Language (PL)
    - Example: XML, JSON
  - Big Ball of Mud



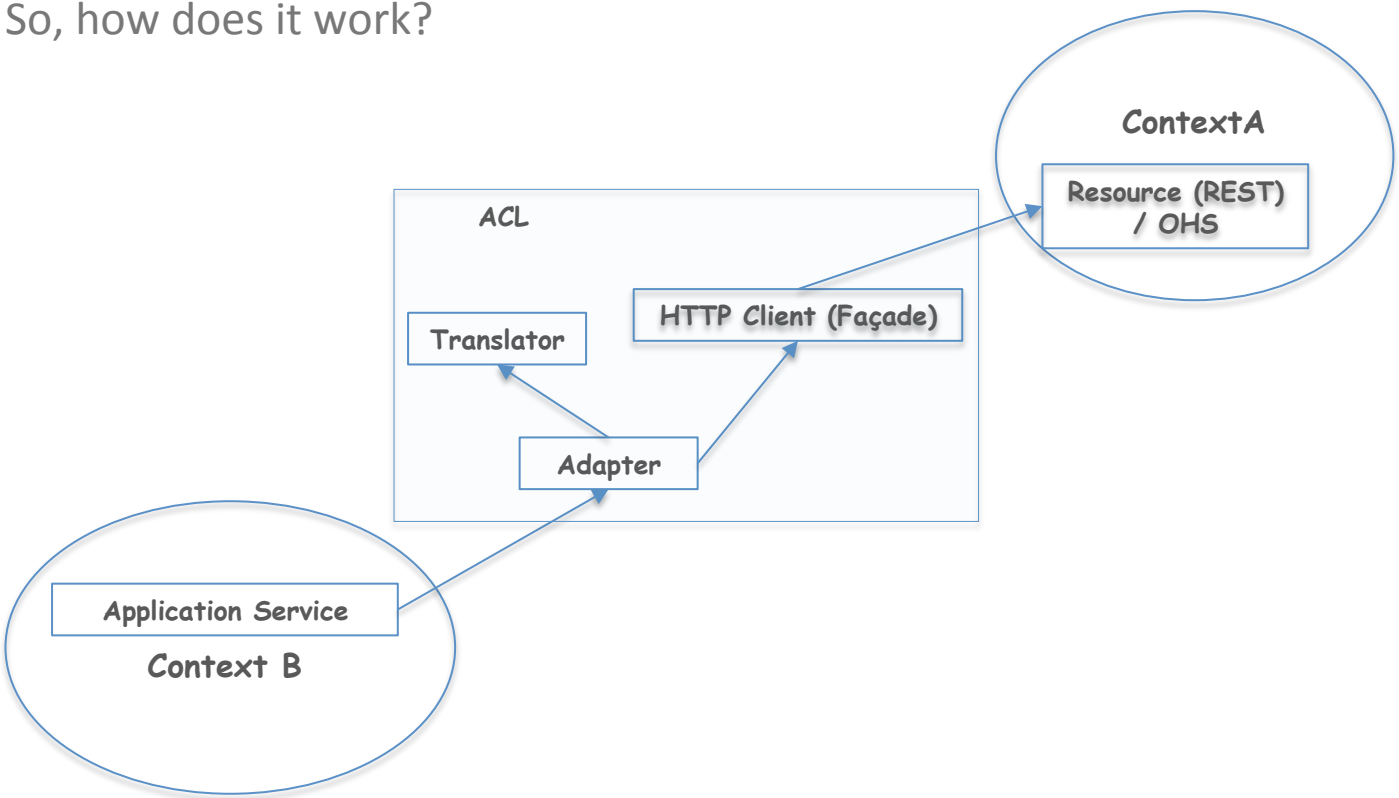
Context Mapping

- So, how does it work?



Context Mapping

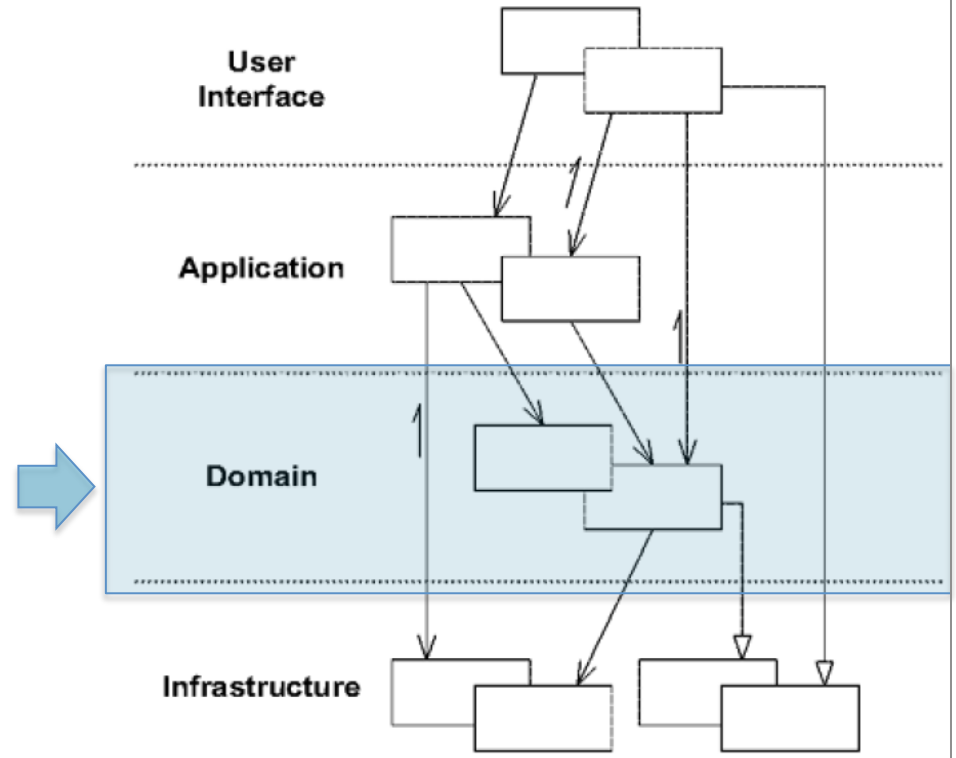
- So, how does it work?



Tactical DDD (DDD-Lite)

Layered Architecture

The DDD-Lite is applied in Domain Layer



Source: [Evans, DDD]

Entity

- A.k.a. “Reference Object”
- Has Identity and State
- May have multiple representations
  - Example: book through a publishing process
- Not defined primarily by their attributes

Value Object

- Have no identity
- Defined by its attributes
- Used to describe the state of Entity
  - as its attribute
- Immutable
  - Replacement rather than modification
- Could be implemented as Flyweight [GoF]
- Example: Colours, Money, Time

Services

- Operation which does not conceptually belongs to an Entity or Value Object
- Fine-grained domain operation over domain objects
  - Usage: Calculation, transformation etc.
- Domain level – no application, infrastructure etc.
- Stateless
- Usually Singletons [GoF]
- Example:
  - Triggers Entity consistency validation

Domain Event

- Event is something that happened in the domain
- Purpose: to makes a change of state explicit
- Immutable (it happened in the past)
- Data holding structure (POJO)
- Class naming – command happened in the past
  - Example: BrandRenamed
- Publish-Subscribe (a.k.a. Observer [GoF])

Module

- A.k.a. packages
- Divide code by concepts (not by technical criteria)
- There is limit to how many things a person can think about at once (low coupling)
- Choose Modules that tell the story of the system
- Example:

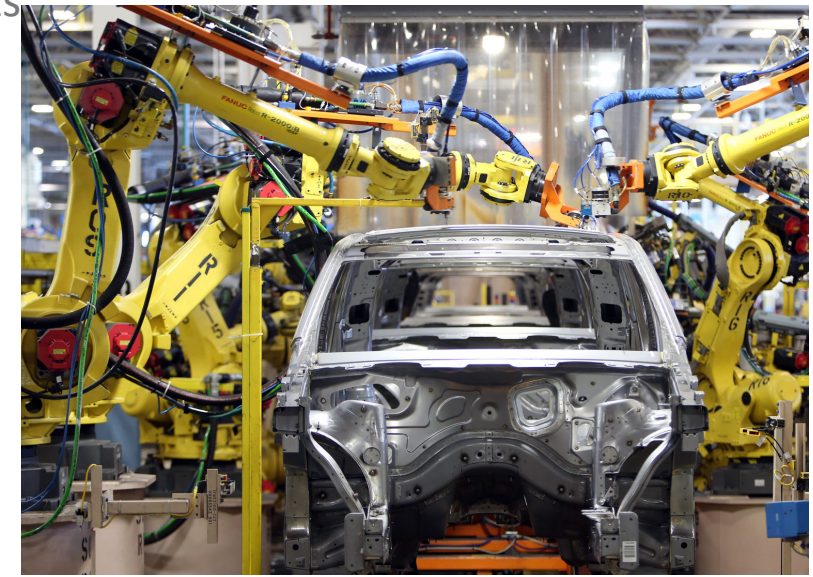
```
[tld].[bounded-context-name].domain.[model | service|…].[conceptname]

com.piksel.vod.domain.model
com.piksel.auth.domain.model

com.piksel.vod.domain.model.assets

com.piksel.vod.domain.service

com.piksel.vod.resources.view
```

Aggregate

- A cluster of associated objects (Entities and Value Objects usually) treated as a unit for the purpose of data changes
- Root (Entity)
- Boundary (what is inside the Aggregate)
- Enforces invariants (business rules which should be always consistent)
- Only Aggregate's root could be referenced from outside objects
- Domain Events emission
  - On change for the Aggregate
- Aggregate Stores
  - NoSQL Key-Value/Document Stores

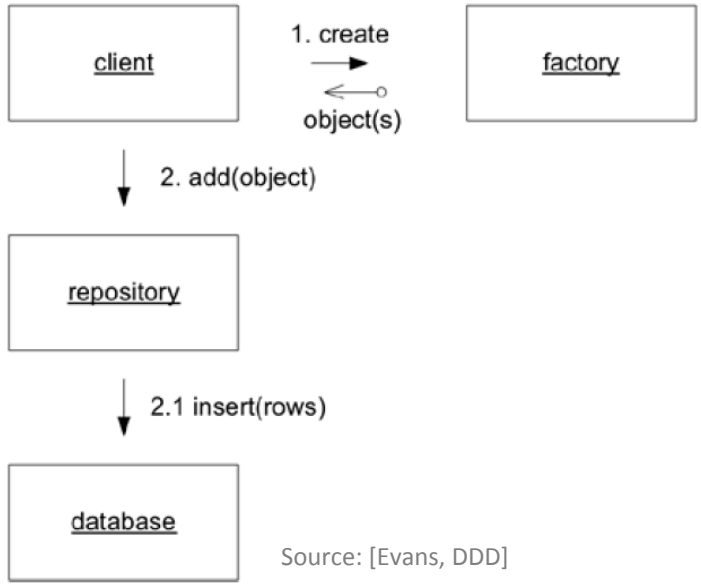
Factory

- Constructors - “primitive level of instance creation… for the programming language”, Evans
  - use only for very simple objects - straightforward construction!
- Creation of complex Entities/Value Objects
- Implementation - various creational (GoF) patterns
  - Factory Method
  - Abstract Factory
  - Builder



Repository

- A contract between application and data storage, that speaks UL
- Repository per Aggregate
- Pretends to be a collection of Aggregate Roots
- Isolates Domain Layer from persistence details
- Uses Factory when storing new object



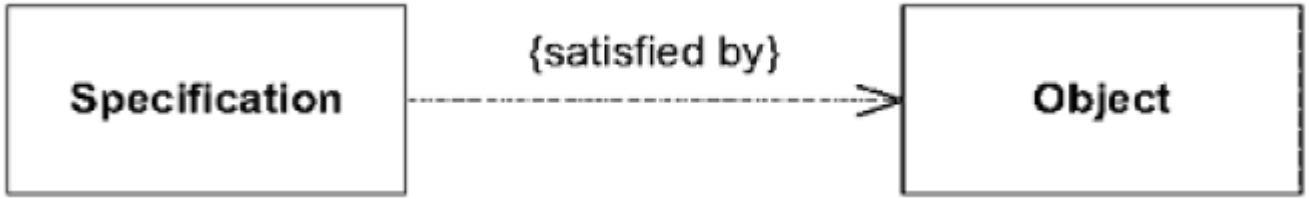
Source: [Evans, DDD]

Repository

- Uses Factory when reconstitutes preexisting object

Source: [Evans, DDD]

Specification



Source: [Evans, DDD]

- Makes implicit business rules explicit
- It's a predicate
- Allows creation of Compound Specifications (using NOT, AND, OR, etc.)
- Used for:
  - Validation
  - Selection / Queries (by Repository)
  - Building (Generation)

Code / Example

Code / Example

- Example project
  - DDD-CQRS Sample, (v.02)
  - HLA: ***http://prezi.com/akrfq7jyau8w/ddd-cqrs-leaven-v20/***
  - Code base: ***https://github.com/BottegaIT/ddd-leaven-v2***
  - User Group: ***https://groups.google.com/forum/#!forum/ddd-cqrs-sample***

- More projects on official CQRS site
  - http://cqrs.wordpress.com/examples/

Questions?

Thank you for your attention!

Resources

Resources

- [Evans, DDD] "Domain-Driven Design: Tackling Complexity in the Heart of Software", Eric Evans, Addison Wesley, 20/08/2003, ISBN: 0-321-12521-5
- [Vernon, DDD], "Implementing Domain-Driven Design", Vaughn Vernon, 2013, ISBN-10: 0-321-83457-7, ISBN-13: 978-0-321-83457-7
- [Fowler, PoEAA] "Patterns of Enterprise Application Architecture", Martin Fowler, Dave Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford), 05/11/2002, ISBN: 0-321-12742-0
- [Fowler, NoSQL] "NoSQL Distilled", Martin Fowler, 08/11/2012, ISBN-10: 0-321-82662-0, ISBN-13: 978-0-321-82662-6
- [GoF, DP] "Design Patterns: Elements of Reusable Object-Oriented Software", Erich Gamma , Richard Helm , Ralph Johnson , John Vlissides,10/11/1994, ISBN-10: 0201633612, ISBN-13: 078-5342633610
- [Fowler, NoSQL talk] "NoSQL matters Cologne 2013 - Key Note: NoSQL Distilled to an hour - Martin Fowler", Martin Fowler, 2013, ***http://vimeo.com/66052102***

Resources

- [Khan, DDD] "Getting Started with Domain-Driven Design", Aslam Khan, ***http://refcardz.dzone.com/refcardz/getting-started-domain-driven***
- [Brandolini, DDD ContextMapping] "Strategic Domain Driven Design with Context Mapping", Alberto Brandolini, 25/11/2009, ***http://www.infoq.com/articles/ddd-contextmapping***
- [Riley, DDD], "Domain Driven Design", Ryan Riley, 22/10/2009, ***http://www.slideshare.net/panesofglass/domain-driven-design***
- [Ferguson , DDD], "Domain Driven Design", John Ferguson Smart, 24/08/2011, ***http://www.slideshare.net/wakaleo/introduction-toddd***
- [Sizovs, DDD] "Introduction to DDD", Eduards Sizovs, 25/10/2012, ***http://www.slideshare.net/eduardsi/introduction-to-ddd***
- Domain Driven Design Website, ***http://domaindrivendesign.org/***
- DDD User Group, ***http://tech.groups.yahoo.com/group/domaindrivendesign/***
- DDD Community, ***http://dddcommunity.org/***


JUG
Bulgaria