UNIVERSITÄT DES SAARLANDES

Domain-Driven Design (DDD)

Architectural Thinking for Intelligent Systems

Winter 2020/2021

Prof. Dr. habil. Jana Koehler

Agenda

- Build an understanding of the business domain
- Capture this business understanding in models
- Separate business domain and application layer
- Domain elements and bounded contexts
  - Entities
  - Value objects
  - Domain events
  - Services
- DDD context maps and the big ball of Mud

Assignment for this Lecture

- We analyze the business domain in more detail and identify the essential elements of the domain.

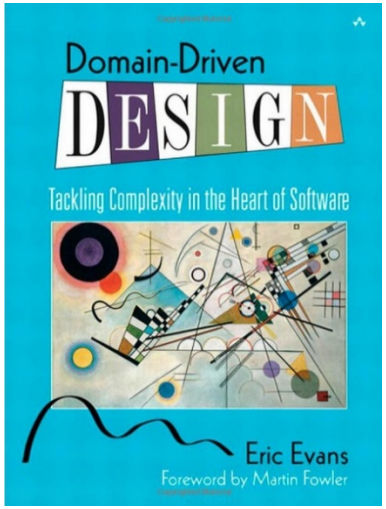
- We explore ways to structure the domain into different bounded contexts.

- For each context, we decide which
  - entities,
  - Value objects
  - domain events, and
  - services

  we need and begin to model these elements.

Recommended Reading

- Chapter 1 (pages 7-21)
- Chapter 5 (pages 81-122)



- For a larger example, consult Chapter 7 (pages 163-186)

- Optional Reading
  - Chapter 8 (pages 193-203)
    - Another nice example illustrating the method
  - Chapter 2 (pages 23-43)
  - Chapter 4 (pages 67-79)

From Business Case to System Structure

- Understand the business domain and reflect it in the architecture
- Speak the language of the business
- Find out about potential conflicts in the understanding of what needs to be done among different stakeholders
- Address risks related to data and services
  - find compromises

Understanding the Business Language

- Adopt the business language and use it in architecture, design, and code
  - Business languages are always spoken in a context
  - Clarify the meaning of specific terms in this context
- Create models to capture key concepts of the application domain at an adequate level of abstraction
- Code implements the model
  - *«express the model through the code»*

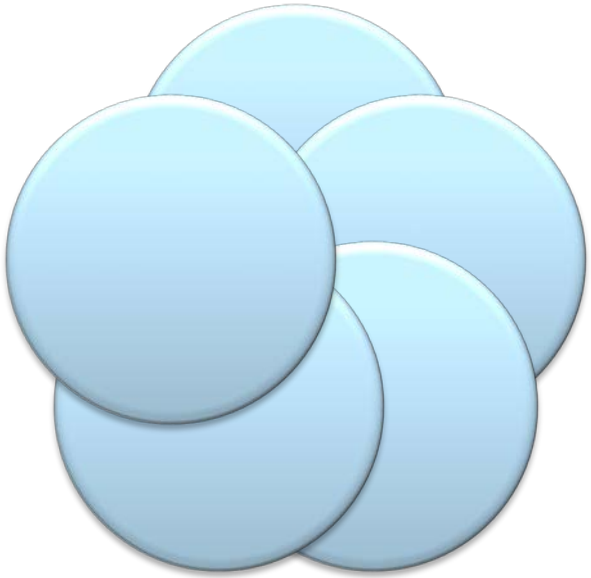
- Continuous Integration means above all ensuring a common understanding of the business language among the team!

Same Language Everywhere …

- The stronger the agreement on the meaning of business concepts, the more viable the system!

(Business) Language

Architecture



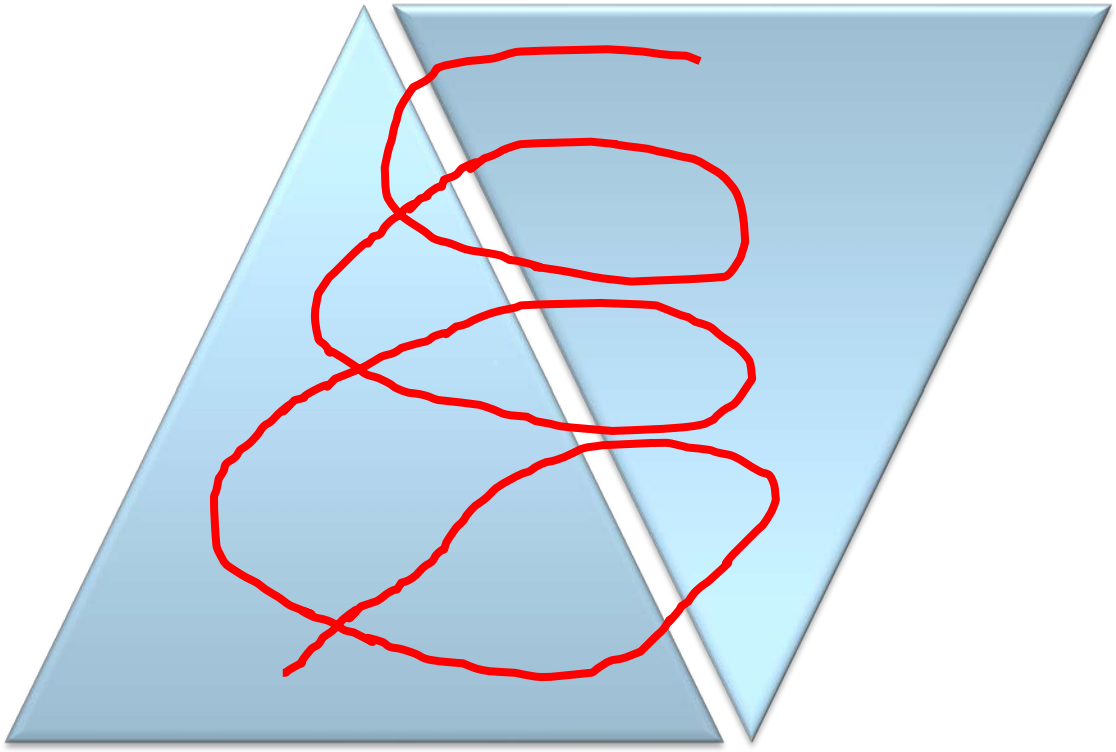
Design

Code

Model

Interleaving of Understanding – Modeling – Implementing

Documenting and understanding important concepts of the business domain

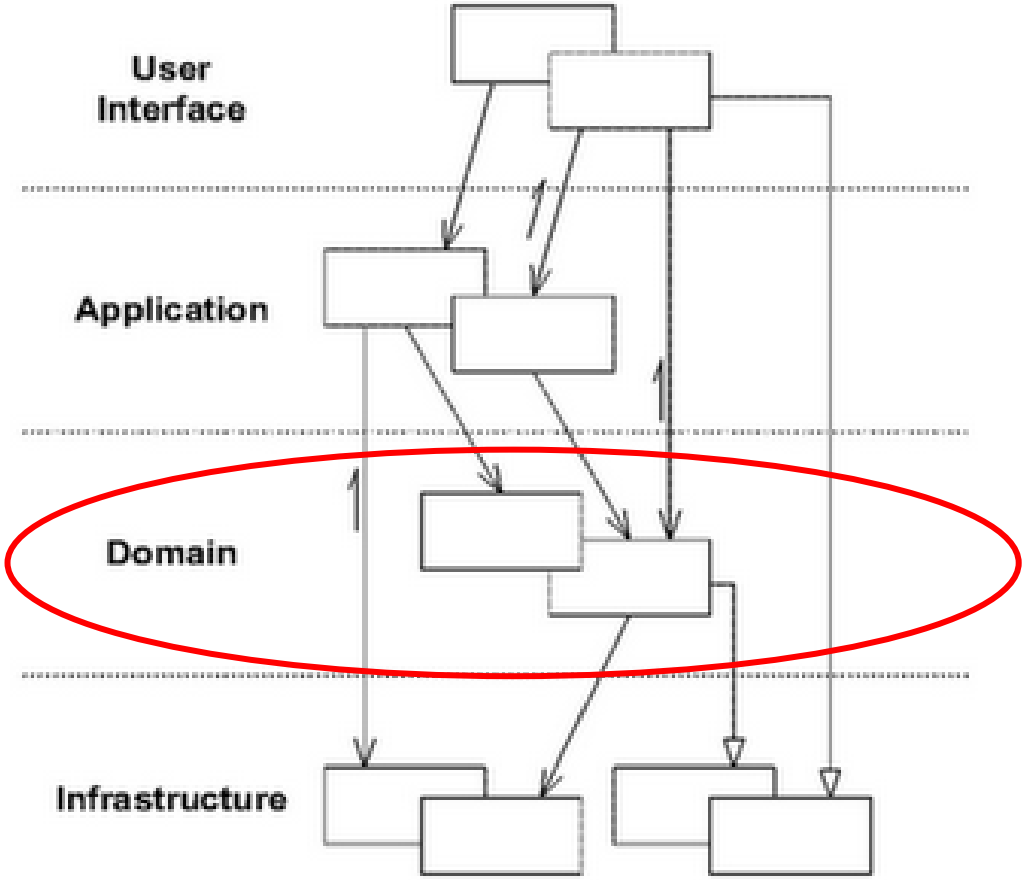


"Sophisticated domain models seldom turn out useful except when developed through an iterative process of refactoring, including close involvement of the domain experts with developers interested in learning about the domain."

Designing a part of the software to reflect the understanding of the business domain

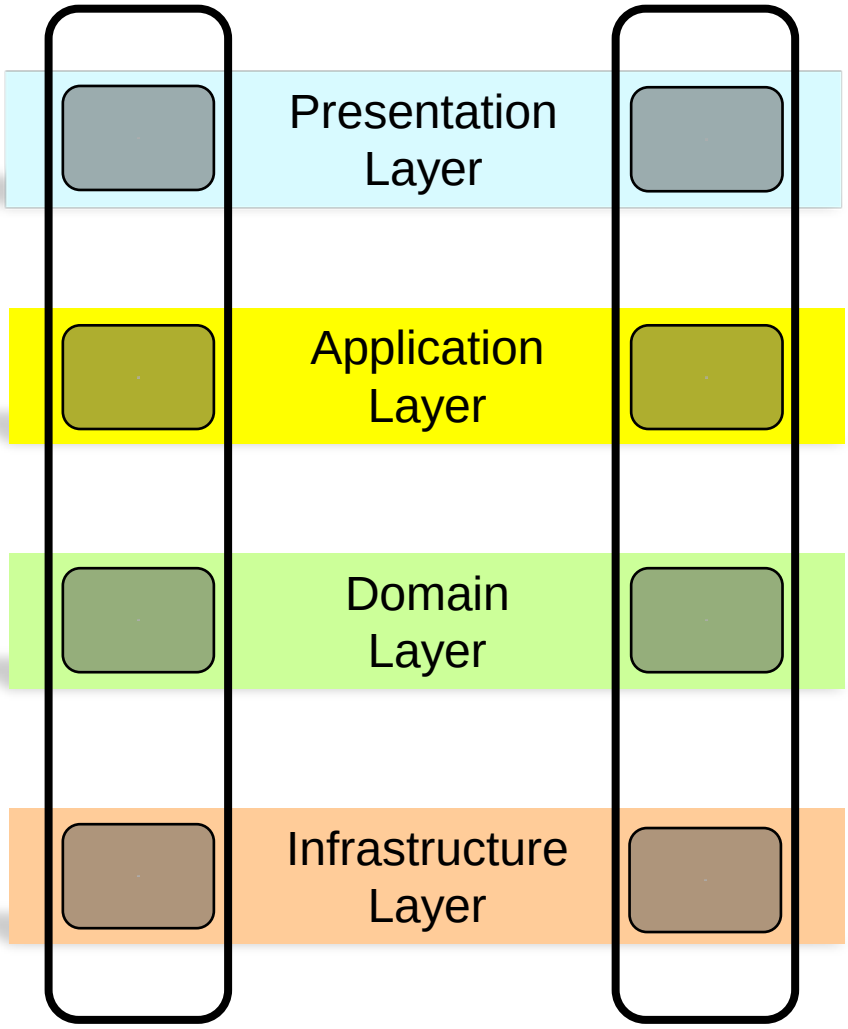
Separating the Domain from other Architectural Layers

- Essential Elements of any Application Domain
  - Entities
  - Value Objects
  - Domain Events
  - Services



Domain and application exist <u>independently</u> of IT TECHNOLOGY!

Divide and Conquer – Build Layers and Components



- Each layer provides clearly defined interfaces and uses services from underlying layers
- Each component implements a specific business or technical function
- **Understanding the business domain is crucial for architecture viability!**

How to Proceed?

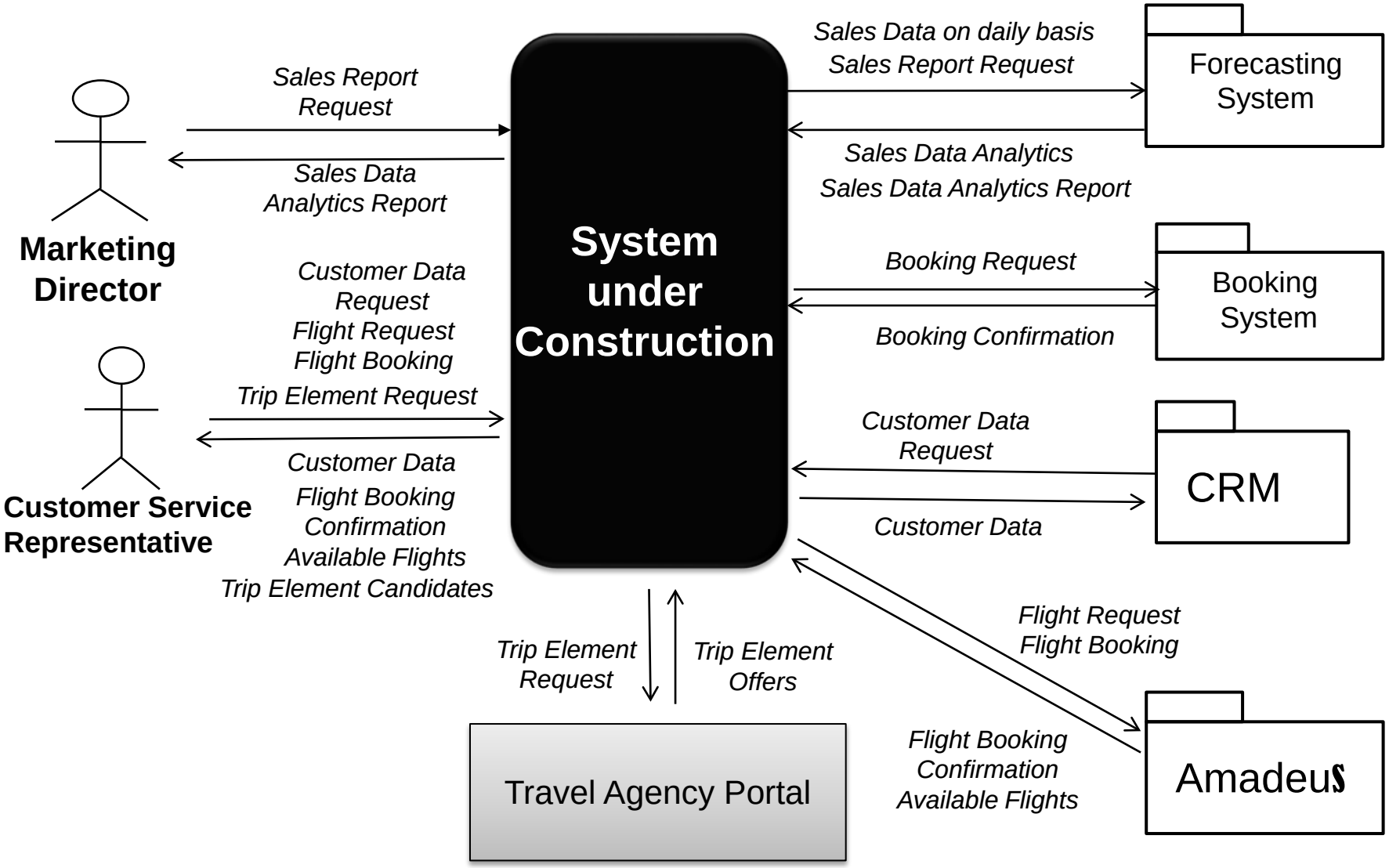
- Identify the main business objects
- Do not establish dependencies on
  - Infrastructure
  - User interfaces
  - Application logic

- Split complex domains into subdomains
  - «Bounded Contexts»
- The structure and behavior of domain objects emerges directly from the business domain

Where to Start From?

- Review the set of business terms from
  - Requirement descriptions in scenarios and use cases
  - Context diagram
  - Vision statement
- Derive key data objects from business terms
- Group objects into contexts
- Create data models focusing on properties and services
  - Avoid information duplication

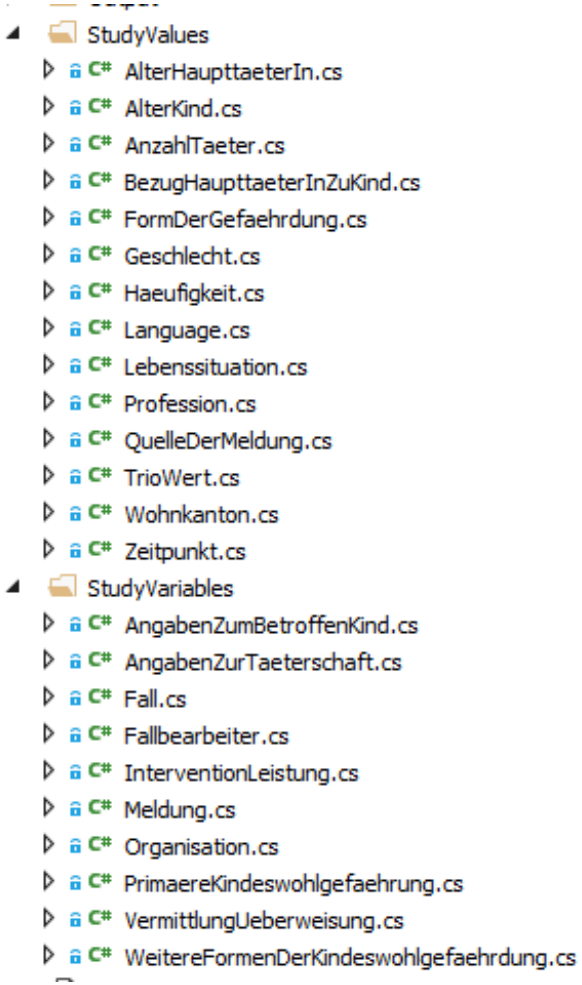
Example: Analyzing a Context Diagram



Example

- «Case»
- «Study variables»
- «Variable (study) values»

"Tightly relating the code to an underlying model gives the code meaning and makes the model relevant."



	Angaben zur fallbearbeitenden Person					Angaben zur Meldung der Kindeswohlgefährdung			Angaben zur primären Kindeswohlgefährdung			Weitere Formen der Kindeswohlgefährdung			
Fall	Alter	Geschlecht	Profession	Anzahl der Jahre im Berufsfeld	Anzahl Jahre in Organisation	Quelle der Erstmeldung	Datum	frühere Meldung	Form der Kindeswohlgefährdung	Beginn	Häufigkeit	psychische Gewalt	körperliche Gewalt	Vernachlässigung	sexuelle Gewalt
1	16	weiblich	Soziale Arbeit	4	1	Gesundheitswesen	13.11.2016	nein	Münchhausen Stellvertretersy	in diesem Monat	mehrmals regelmässig	nein	nein	nein	ja
2	15	weiblich	Pädagogik	3	2	Eltern(teil)	17.10.2016	ja	körperliche Gewalt	in diesem halben ja	mehrmals (unspezifisch)	nein	nein	nein	ja
3	75	männlich	Pädagogik	4	3	Verwandte und soziales Um	19.09.2016	nein	sexuelle Gewalt	vor mehr als einem !	mehrmals unregelmässig	ja	nein	ja	nein
4	15	männlich	Medizin	6	4	Pflegeeltern, Heim, Kinderti	15.09.2016	nein	Gewalt zw. Bezugspersonen	vor mehr als einem !	mehrmals unregelmässig	ja	nein	ja	nein
5	16	weiblich	Pädagogik	10	5	Schule und schulische Diens	20.11.2016	nein	sexuelle Gewalt	in dieser Woche	mehrmals regelmässig	ja	nein	nein	nein
6	55	weiblich	Pädagogik	35	6	KESB	25.11.2016	nein	sexuelle Gewalt	in dieser Woche	einmal	ja	nein	nein	nein
7	19	männlich	Medizin	3	1	Sozialbereich	30.11.2016	nein	Gewalt zw. Bezugspersonen	in diesem Monat	mehrmals unregelmässig	nein	nein	nein	nein
8	20	männlich	Polizei	20	2	Gesundheitswesen	05.10.2016	nein	Vernachlässigung	in diesem halben ja	mehrmals regelmässig	nein	ja	nein	nein
9	25	männlich	Feuerwehr	3	3	Strafverfolgung	13.10.2016	nein	psychische Gewalt	vor mehr als einem !	mehrmals (unspezifisch)	nein	nein	nein	nein
10	35	männlich	Pädagogik	5	4	anonyme Meldung	17.10.2016	ja	Gewalt zw. Bezugspersonen	vor mehr als einem !	einmal	nein	nein	ja	ja
11	45	weiblich	Psychiatrie	7	5	andere Organisation, Fachp	17.09.2016	nein	sexuelle Gewalt	in dieser Woche	einmal	nein	nein	ja	ja

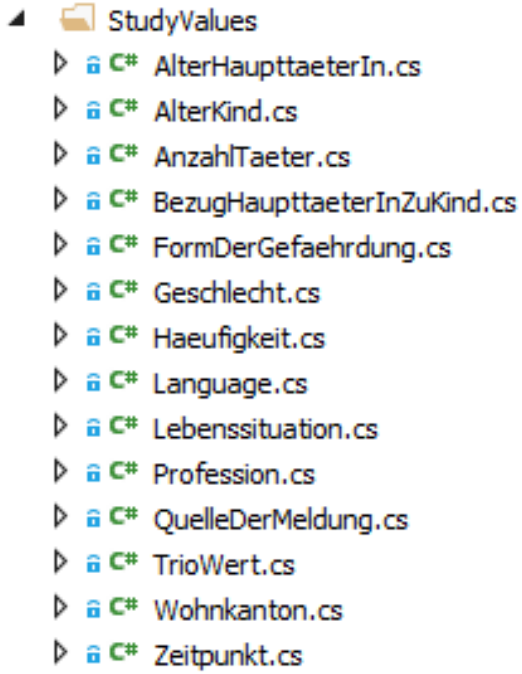
Entities

- Things, people who have an identity = business objects
- Clearly defined (longer) life cycle
  - States relevant in the business domain
    - remember BPMN events
  - Rules control state transition
- Attributes may vary
  - Need for discussion and clarification
- One of the entities plays a particularly important role
  - Examples
    - Restaurant: bill
    - Bank: account
    - Social work (case management): case

Value Objects

- Things that describe the state of entities
- No own identity, never composed of entities, no life cycle that would be meaningful and necessary from a business point of view

```
public class PrimaereKindeswohlgefaehrung
{
    0 references | Fux Etienne tcfux, 147 days ago | 1 author, 1 change
    public FormDerGefaehrdung FormDerGefaehrdung { get; set; }
    0 references | Fux Etienne tcfux, 147 days ago | 1 author, 1 change
    public Zeitpunkt Beginn { get; set; }
    0 references | Fux Etienne tcfux, 147 days ago | 1 author, 1 change
    public Haeufigkeit Haeufigkeit { get; set; }
```



Domain Events

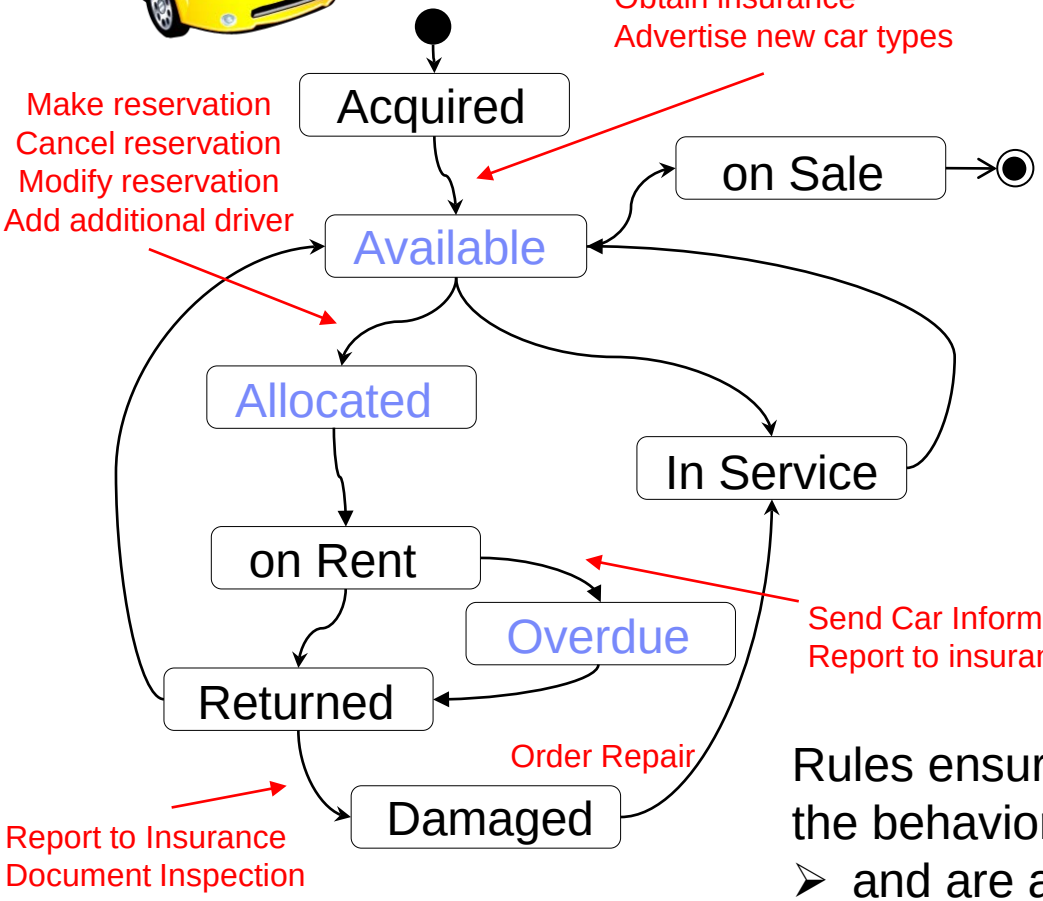
- Which events of the domain determine the life cycle of entities?
- Which events cause state changes?
  - who/what triggers the event?
  - when does the trigger occur?
  - which properties characterize the event?

- Important: Separate business events from application-specific (architecture-specific) events
  - a case of child abuse is suspected
  - a case of child abuse has been edited in the data base by an agency

Services

- Procedures and processes, which change the state (values of properties) of entities
- Understanding business processes: business services/business capabilities
- Service contract
  - Who can use the service when and how?
  - What does the service guarantee?
- Examples:
  - Make table reservation in restaurant, pay by card, …

Derive Services from the Lifecycle of Entities



"A car from another branch may be allocated, if there is a suitable car available and there is time to transfer it to the pick-up branch."

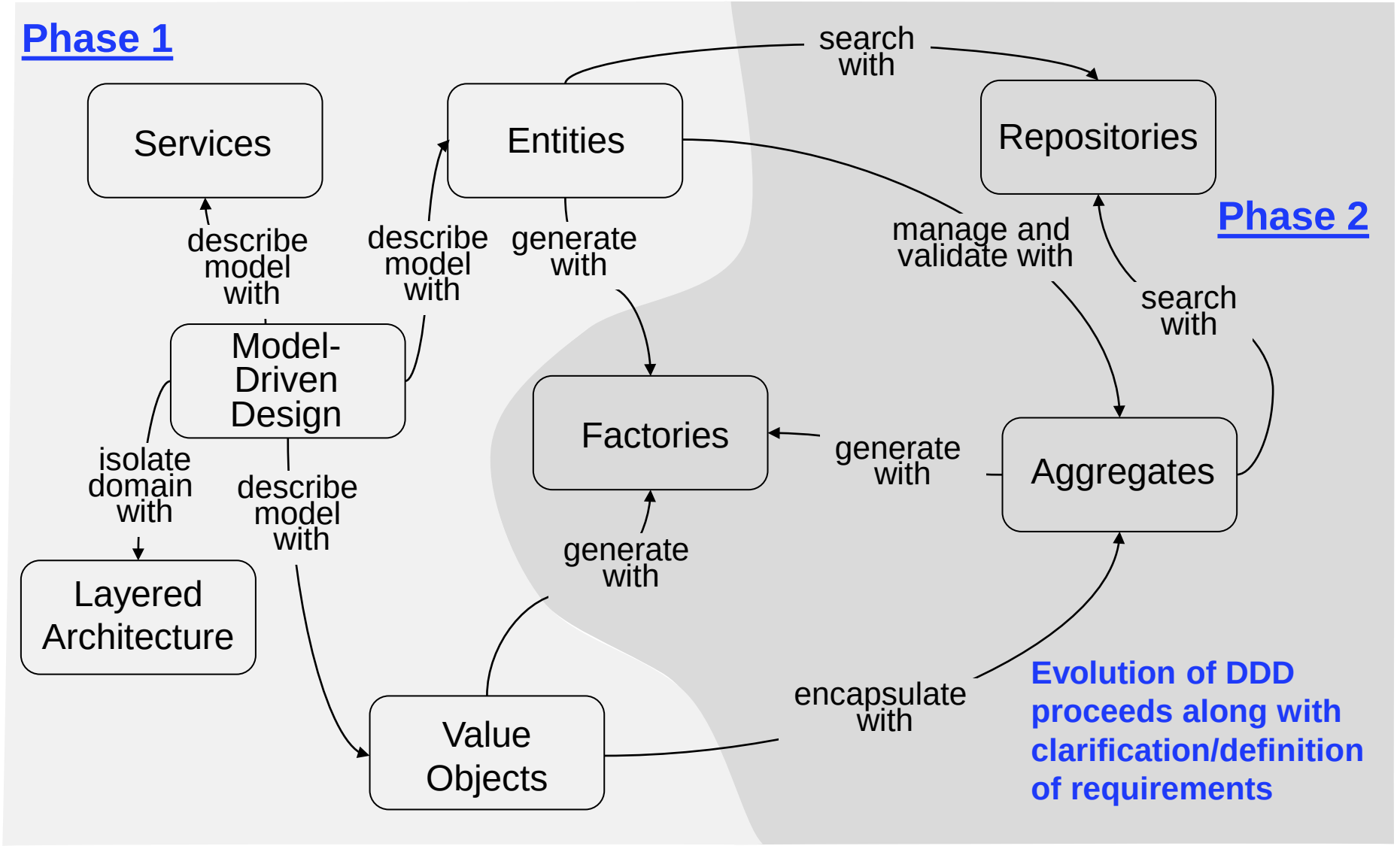
"If a car is three days overdue and the customer has not arranged an extension, insurance cover lapses and the police must be informed."

Send Car Information to Police
Report to insurance

Rules ensure the integrity of data and regulate the behavior of actors

➢ and are a source for acceptance tests!!!

Further Elements of Domain-Driven Design

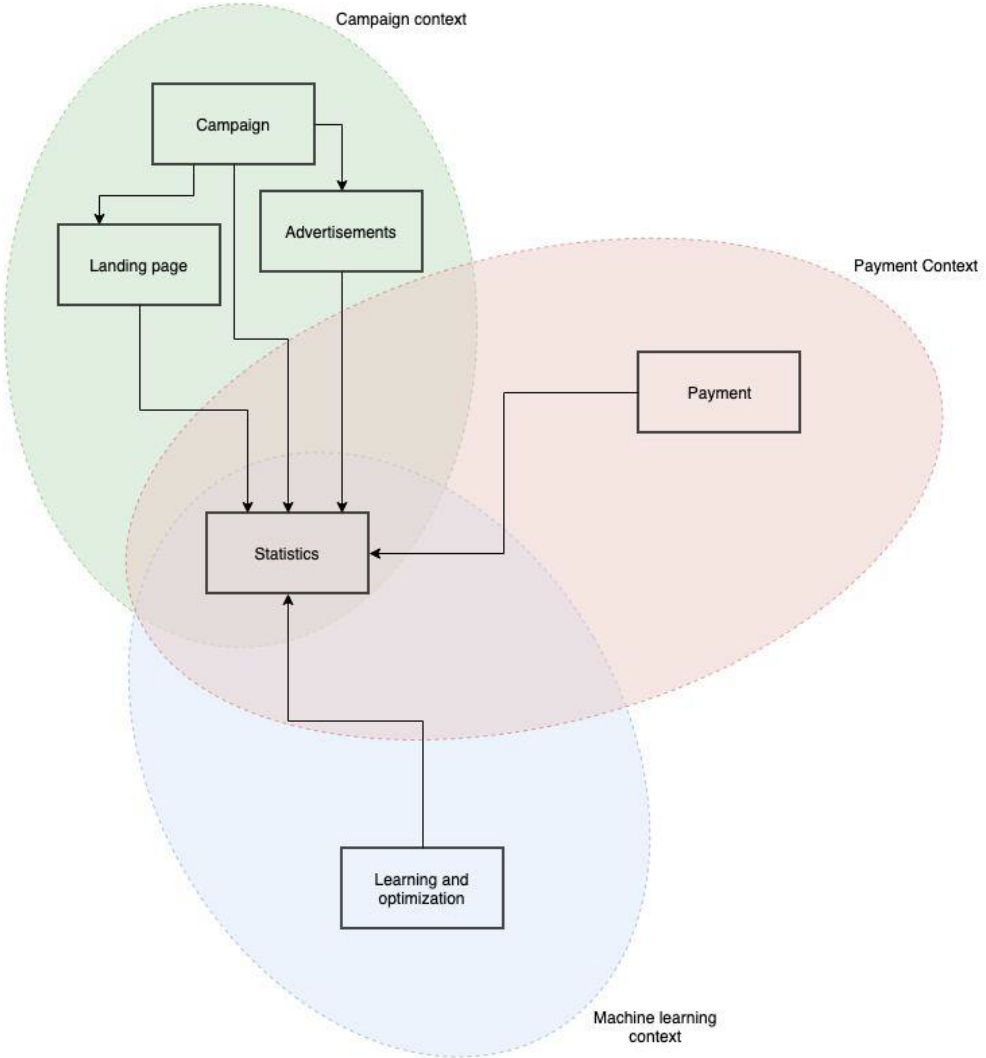


Bounded Contexts

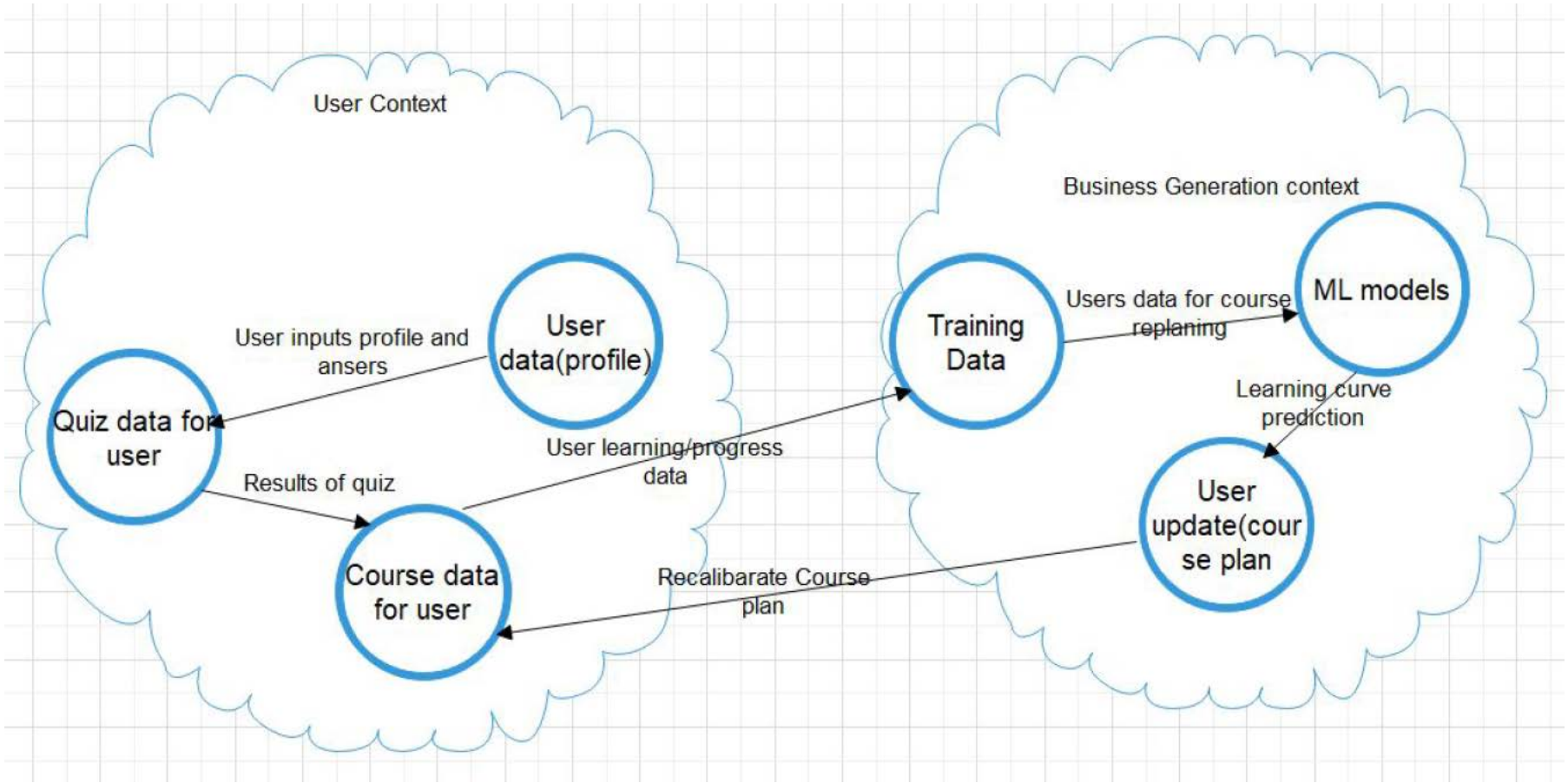
- Remember: the context view separates system and environment
- Within the system, we can distinguish conceptually related domains, which play different roles and which can be separated from each other
- Outside the system, we can identify contexts with whitch the system interacts
- The goal of DDD is to identify and define these bounded contexts, which also help us to separate and refine data models (internally and for the interfaces for external systems)
- Each model applies to one or more bounded contexts
  - Make the boundary explicit

Context Map Example

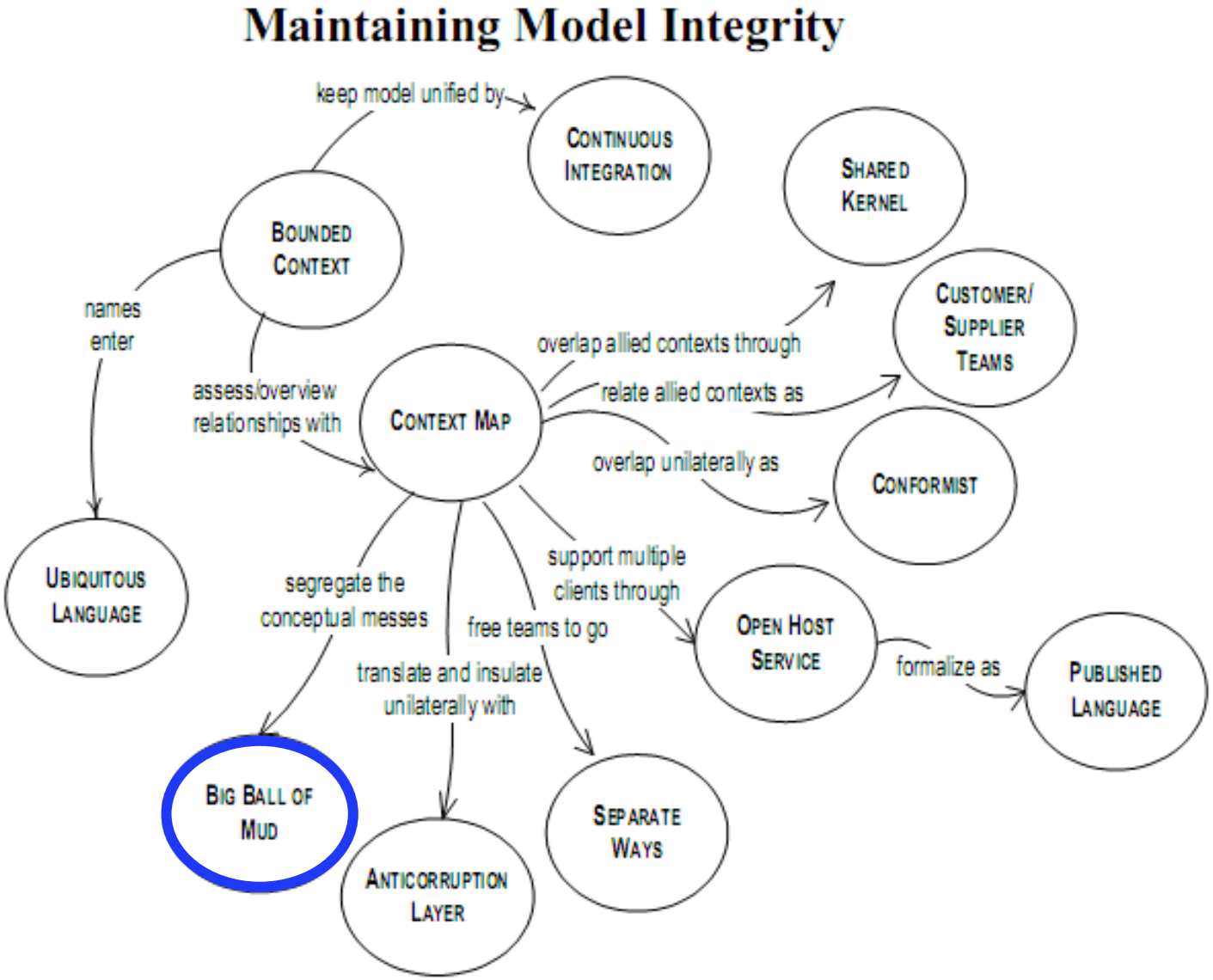
- The first separation of the context shows an overlap in the statistics
- May domain elements create, update, or consume statistics values
- It seems to make sense to separate this into another context and look into the services offered by the statistics entity to the entities in the other contexts



Contexts in a Language-Learning Application (Duolingo)

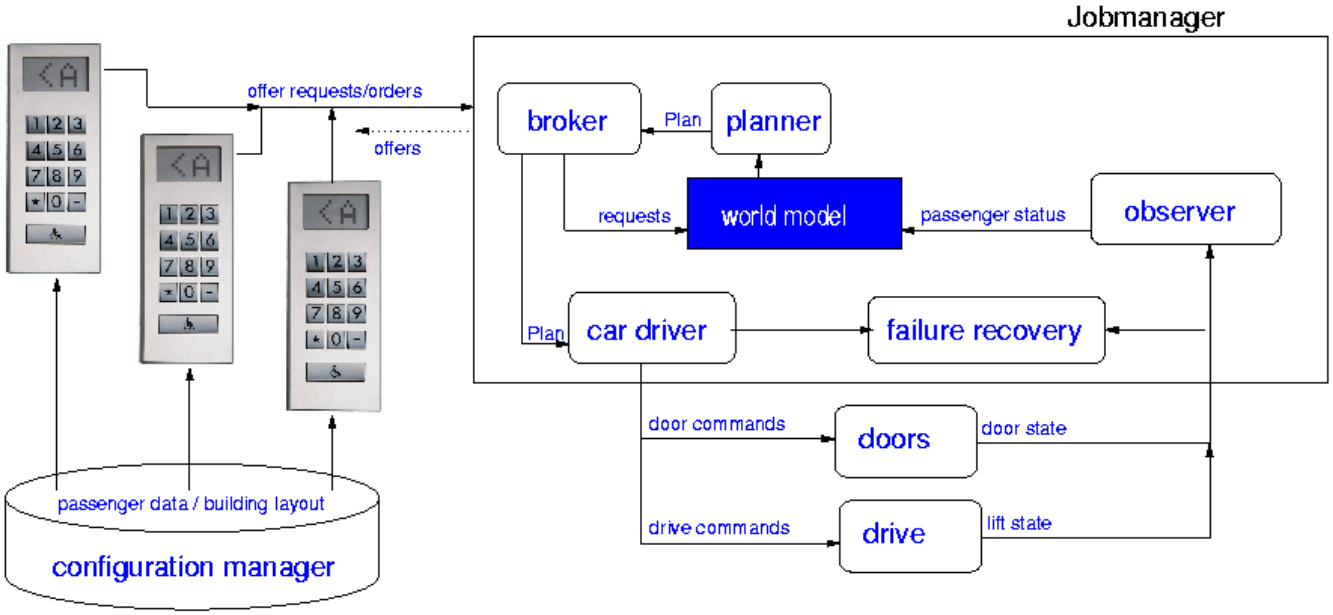


Using DDD in Large Project



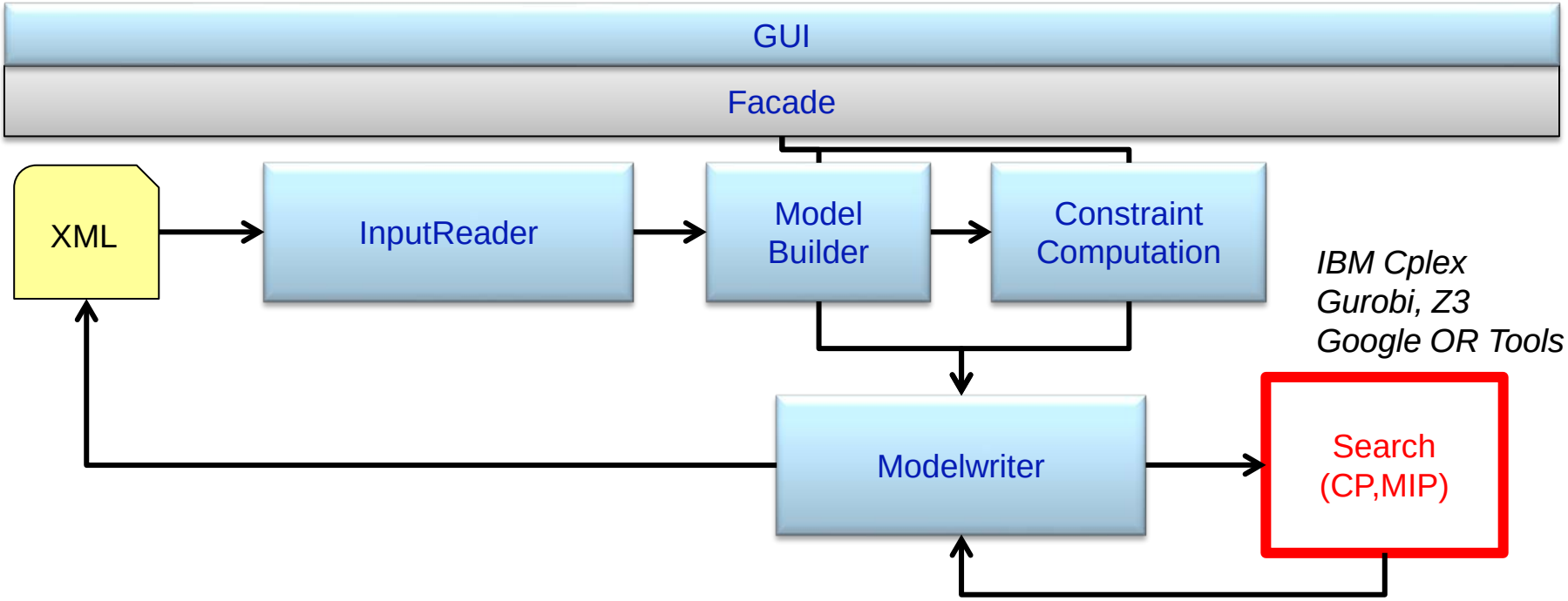
DDD for Elevator Destination Control

- Entities: building, car, drive, door, passenger
- Value objects: geometry of building, behavior of cars (speed, position, jerk) and passengers (origin, destination, boarding state
- Bounded contexts: execution, planning, user communication

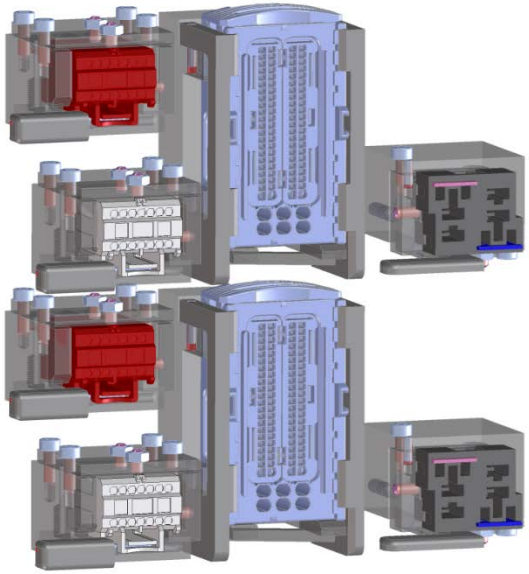


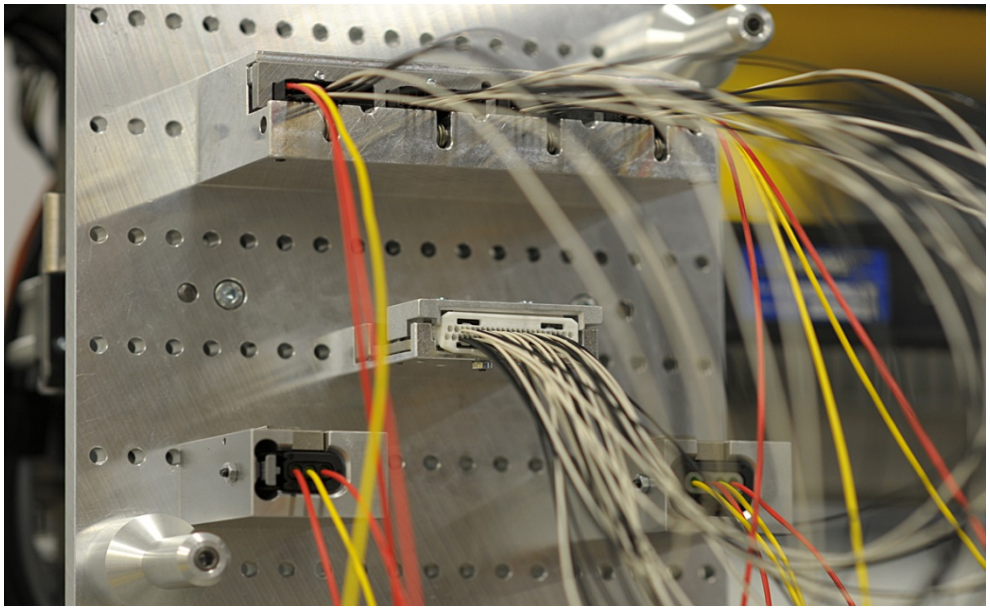
AI System for Cable Wiring

- 10 Code Contributors
- Lines of C# Code: 4500 core system, 5300 GUI, 1800 tests
- > 1200 commits 2012-2016, test coverage 65%
- Yearly major refactoring, maintainability index > 85%



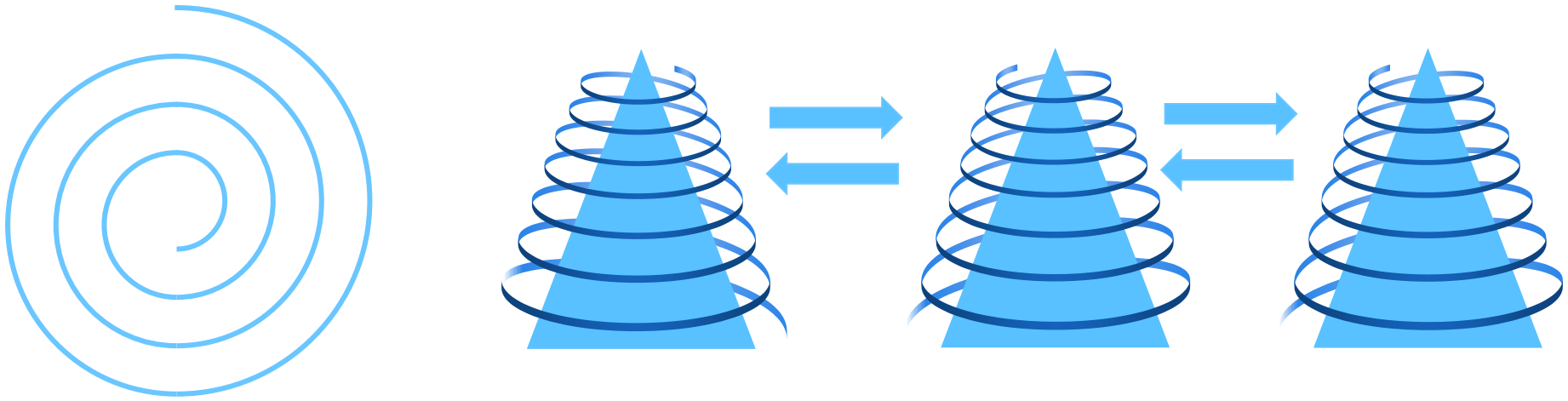
DDD for Cable Tree Wiring





- Entities: cable tree, harness, cavity, cable, palette
- Value objects: positioning of harnesses on palette, geometry of cavities, harnesses, cables
- AI entities: constraints, insertion order, layout

Where are we in Architectural Thinking?



Goal Hierarchies, Scenarios, System Vision Context View

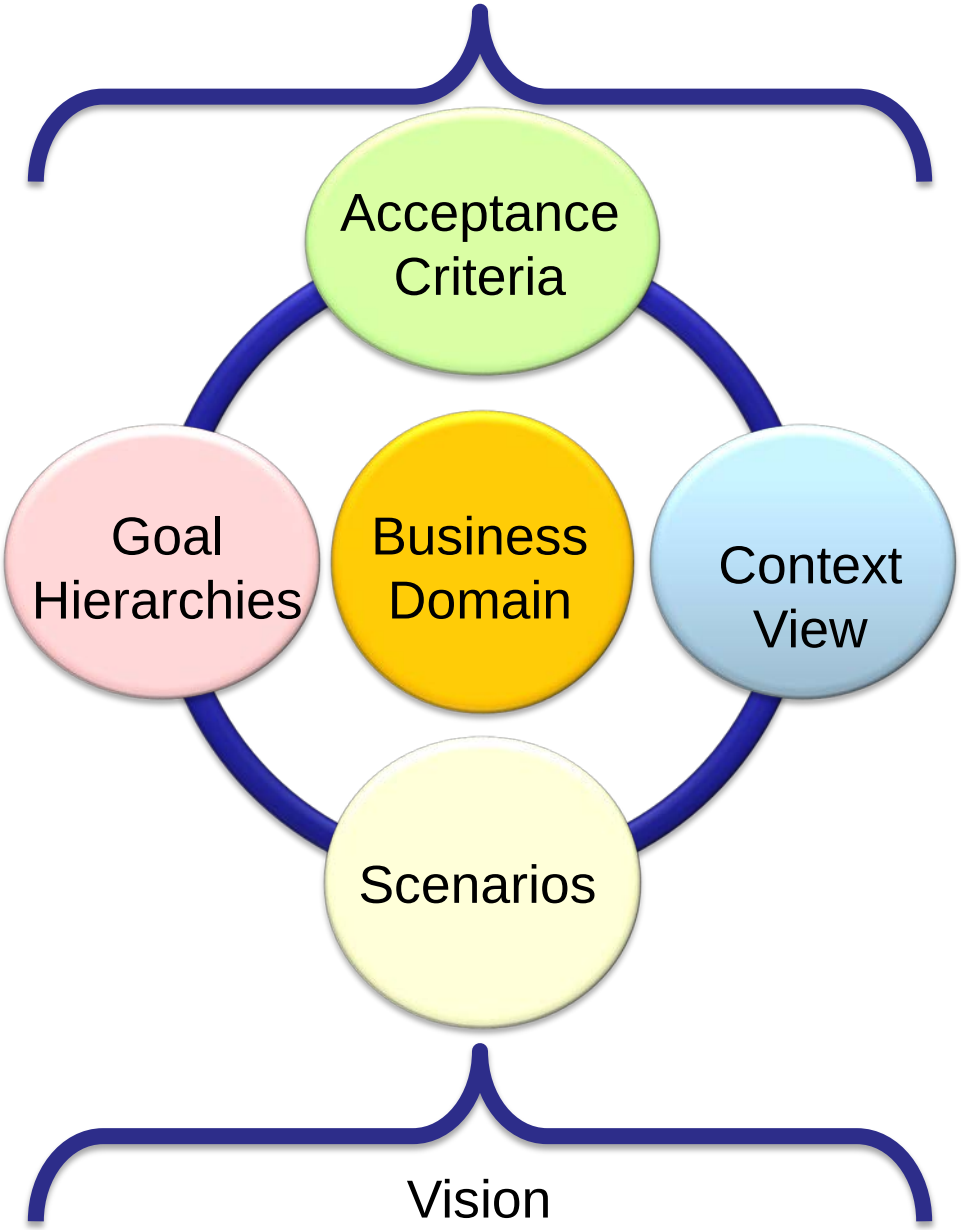
Requirements Constraints

Architectural Draft System Idea

Design

Models of the Business Domain

System Idea



Summary

- Adequate understanding of the business domain is crucial for architectural thinking
- Capture business understanding in models – create views for communication
- Separate business domain from application domain
- Clarification of business terms is critical and time-consuming to build a uniform understanding among stakeholders
- Undetected ambiguities or different semantic understandings are a source of major risks
- DDD focuses on business objects (entities) and their life cycle
  - clear separation of entities - value objects - events - services
- Business model should never be influenced by the infrastructure or technology used
- Large domains are broken into bounded contexts and linked via a context map

Working Questions

1. What do we understand by Domain-Driven Design (DDD)?
2. What is the most important goal of DDD?
3. Why should one strictly separate Domain Layer and Application Layer?
4. What role does Continuous Integration play in DDD?
5. What is a context in DDD?
6. Give examples of entities, value objects, domain events, and services in a domain of your choice.
7. What role does the context map play and how do contexts relate to the Big Ball of Mud?
8. Why do we discuss DDD although our topic is software architecture?