OST
Ostschweizer
Fachhochschule

Collaborative Modeling Learning Lab: Overview

ITBO Learning Lab 2, Initiative 1 (ZIOL, KAPS)

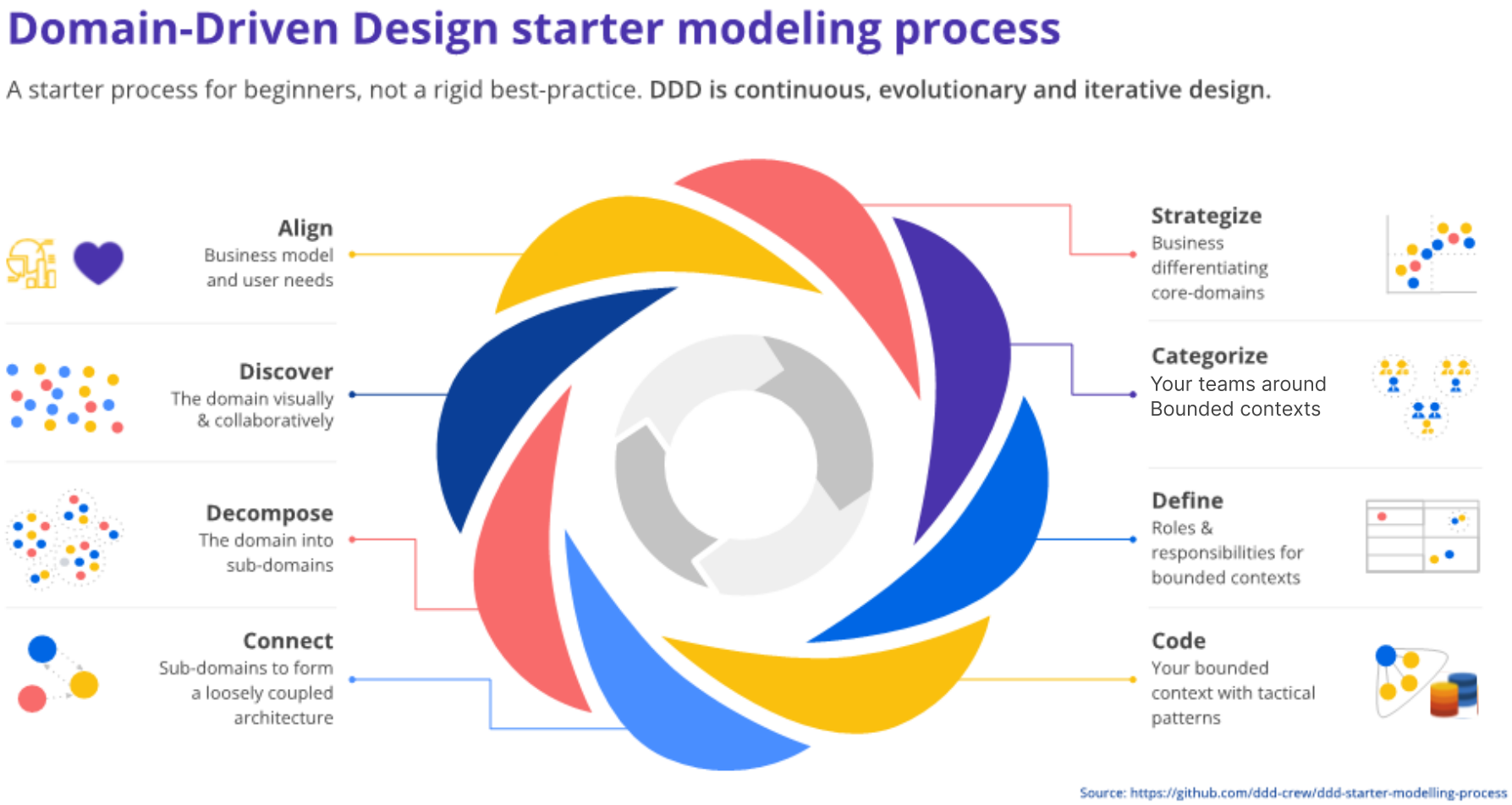
Spring Term 2025

Departement Informatik and ITBO

Agenda

- Motivation
  - Modeling
  - Collaboration
  - CoMo community and CoMo Camp (unconference)
- CoMo LL overview
  - Advance organizer
  - Lectures
  - 7 labs
- Tooling (CI/CD, Pandoc)

Context: Domain-Driven Design



Why Collaborative Modelling (CoMo)?

Motivation

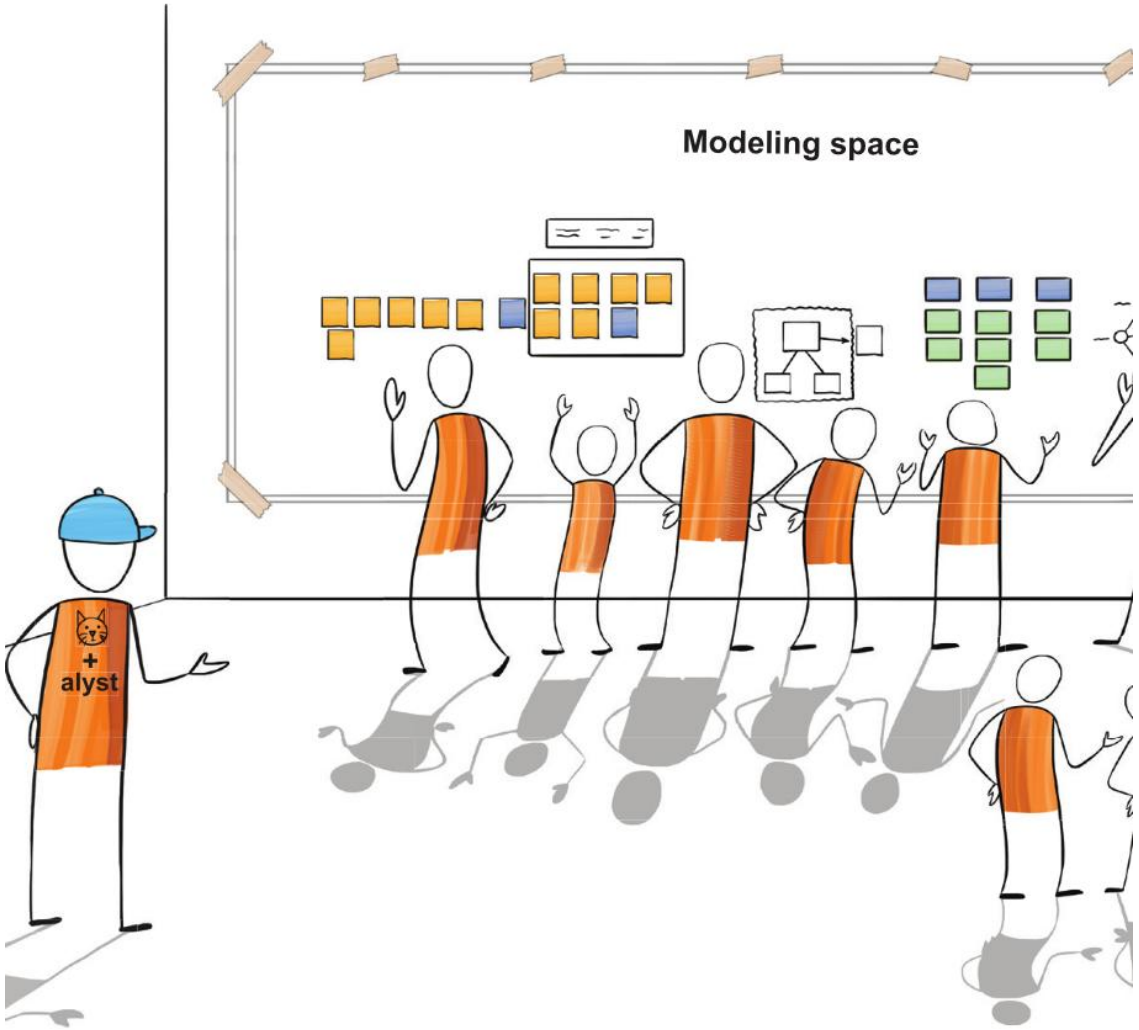
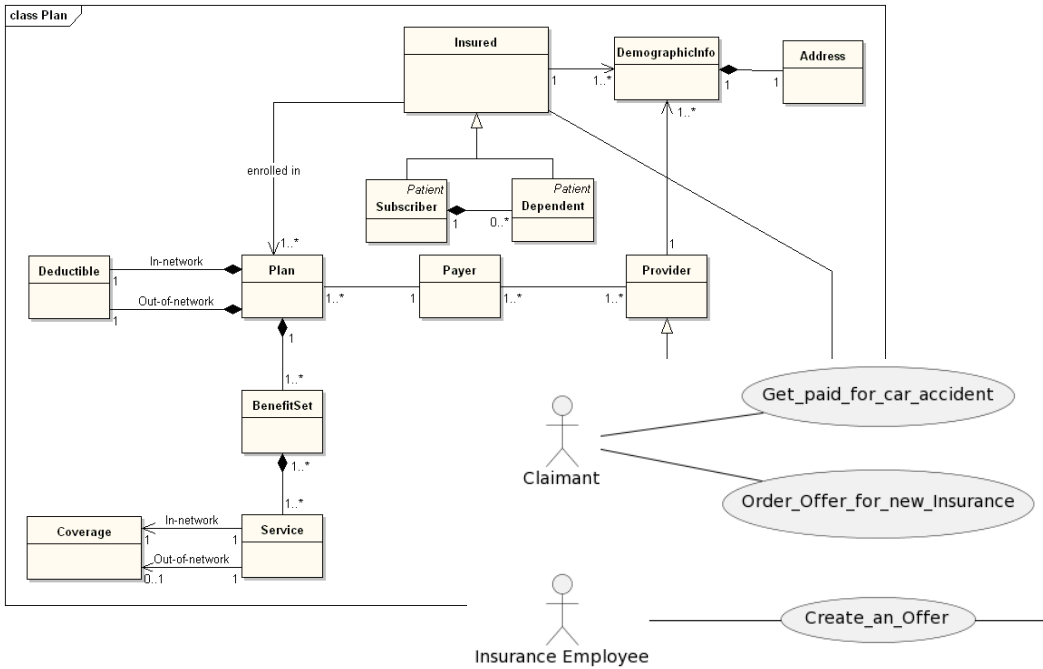
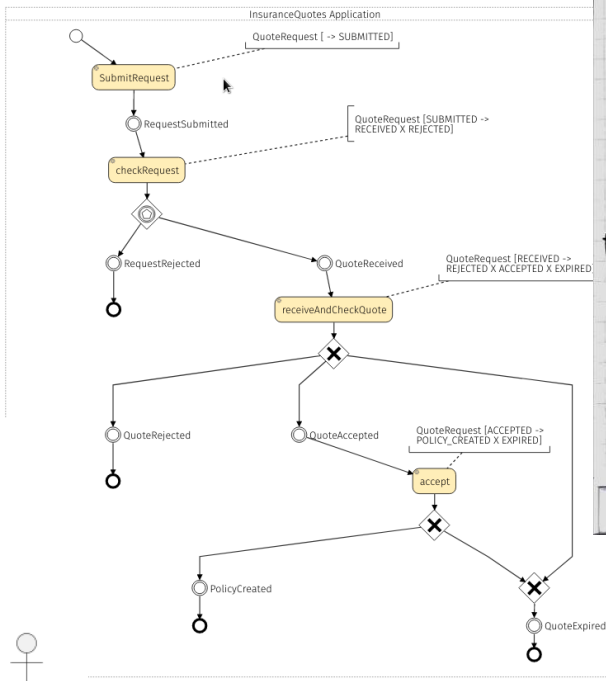


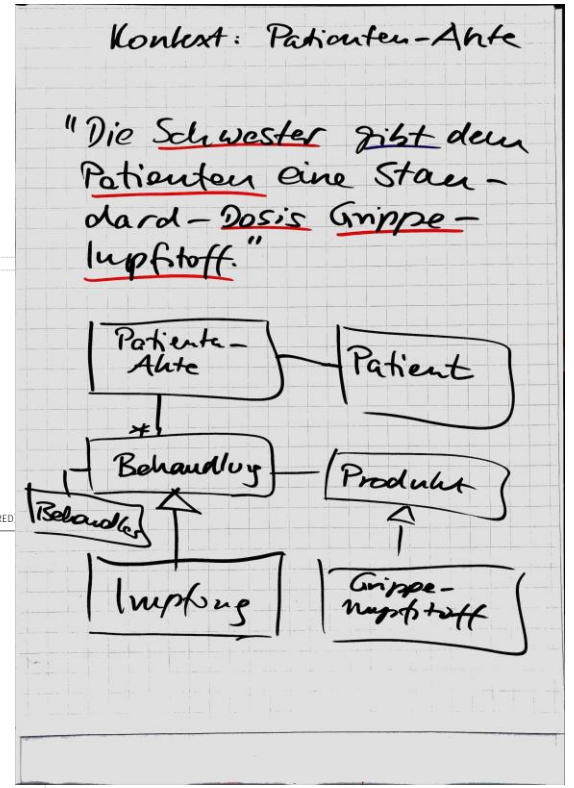
Image Reference: *Collaborative Software Design*, Evelyn van Kelle, Gien Verschatse, Kenny Baas-Schwegler

Motivation

Why do we model in software development?
Why creating domain models?







Motivation for Domain Modelling

- Capture the **essence** of the **problem domain**
- Establish a **shared understanding** (“ubiquitous language”)
- Align **mental models** through **visualization**
- Align **language** with **business** (domain experts)
  - Bridging technical and non-technical stakeholders

- And later of course also to:
  - Find the right **abstractions** and **simplifications**
  - Use domain knowledge as basis for **system design** and **architecture**

Avoid Misunderstandings …

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

-- Phil Karlton

Image Reference: Michael Plöd, *Aligning organization and architecture with strategic DDD*, Link to Slides

Align Solution and „Real World“

Business language	vs.	Developer language
Domain Know-How	vs.	Software Engineering Know-How
Problem space	vs.	Solution space (software design)
Object-oriented analysis (OOA)	vs.	Object-oriented design (OOD)
Subdomains	**vs.**	**Bounded Contexts**

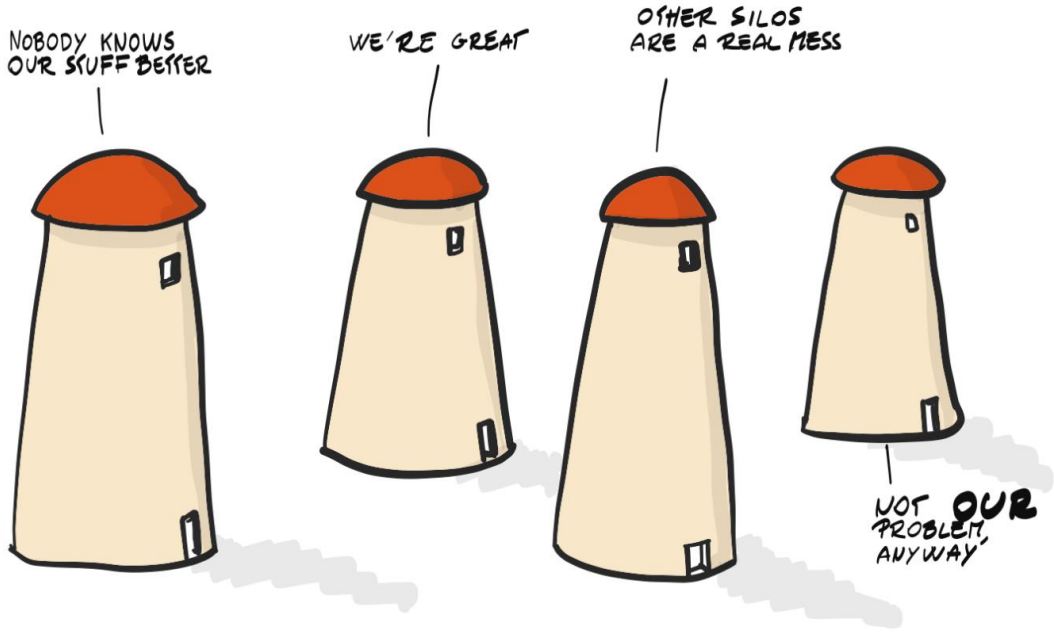
communication required!

Business Domain Experts

Software Engineers and Architects (“Techies”)

Stakeholder Involvement is Key

- Projects within complex domains need **shared unterstanding**
- Knowledge is typically **spread** …
  - … across **business experts**, **developers**, **designers**, **operations**, **management**, etc.
  - … and **silos** in organisations
- Without **everyone‘s perspective**, important goals, rules, expectations and exceptions might be overlooked
- Models must serve as **communication tools**, not just documentation!



Everyone is a master in their silo

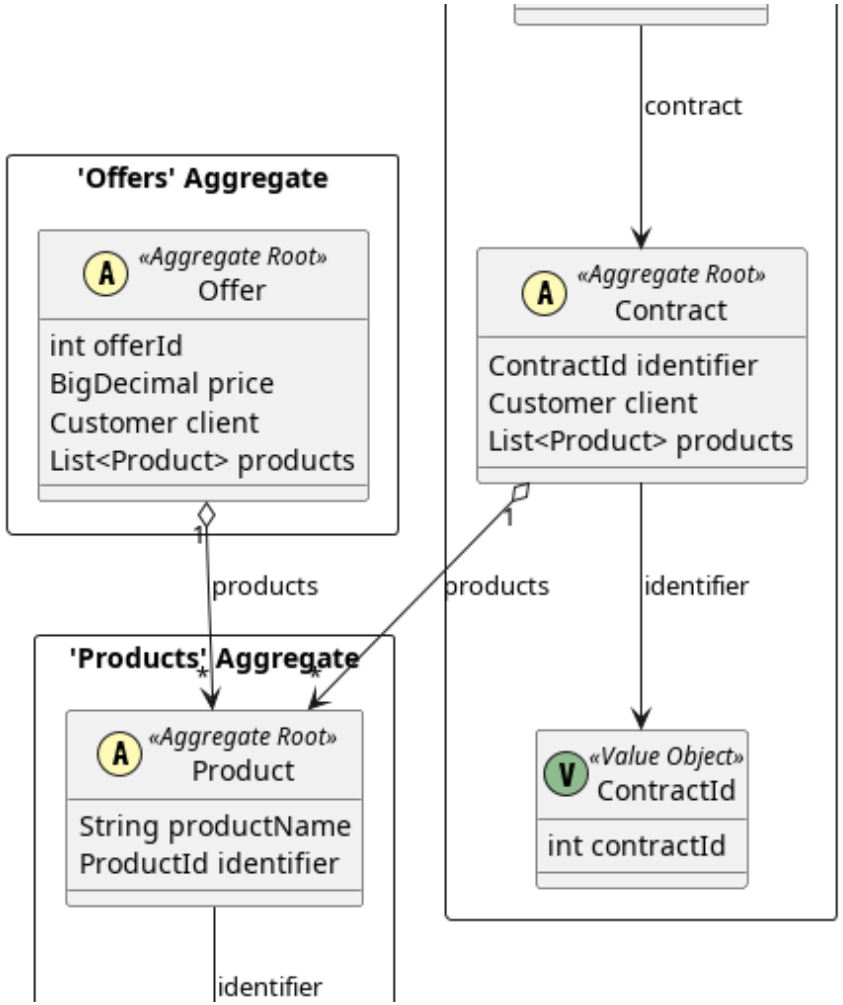
Image Reference: Introducing EventStorming, Alberto Brandolini, http://leanpub.com/introducing_eventstorming

Challenges of Traditional Modelling

Classical notations (such as UML, BPMN, etc.) are designed for **precision** and **technical clarity**…

But they…

- are **not accessible** to non-technical stakeholders (business and domain experts).
- create **„ivory tower“ diagrams**, only understood by analysts and „techies“.
- do **not help having the right discussion**.
  - “In fact, using UML severely harms the possibility of this discussion to happen” (Brandolini)
  - They often force discussions into arguing about technical details.
  - CoMo practices support the idea of having an “incremental notation”.



Key Ideas of „Collaborative Modelling“ (CoMo)

- **Pragmatic, Workshop-based, techniques** to **model collaboratively**
- Invite all team members, important stakeholders and domain experts
- No knowledge on notation required
- Ideally not much preparation by participants required
  - But by moderator/facilitator!
- **Benefits**: inclusion, engagement, speed, shared language, discovery

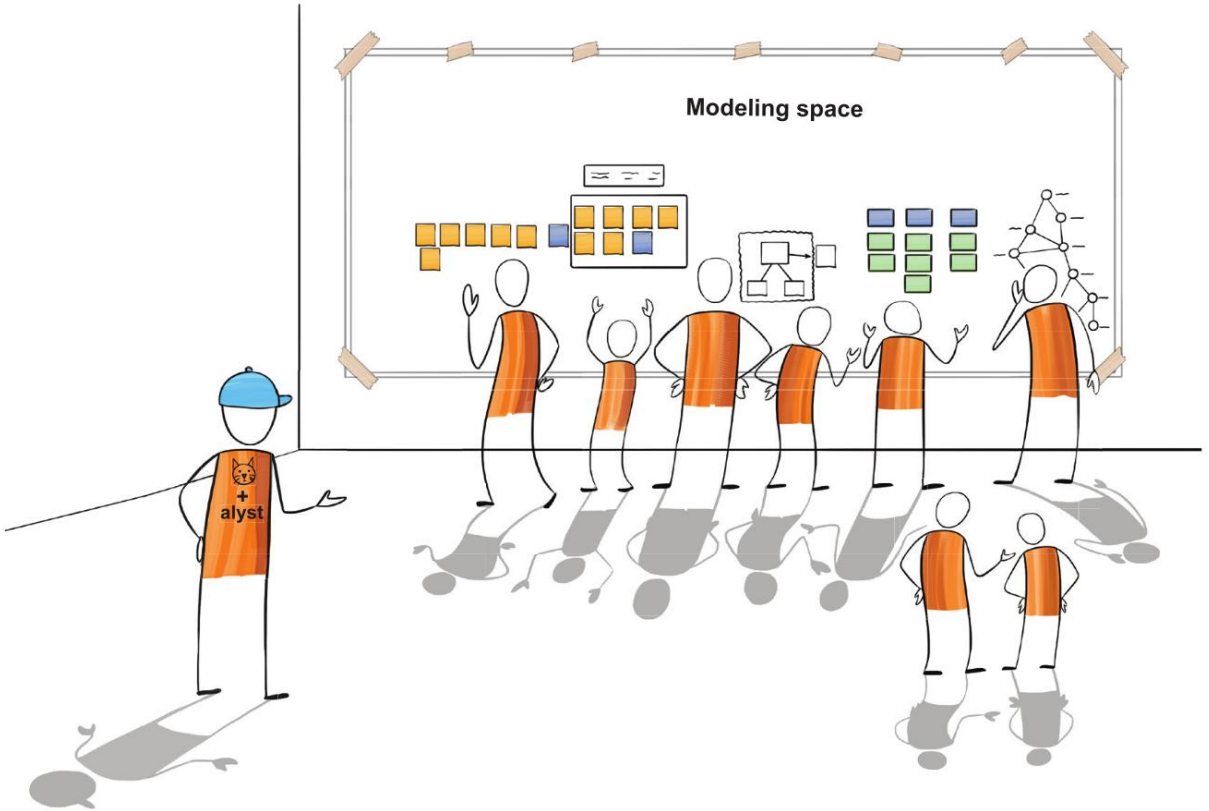
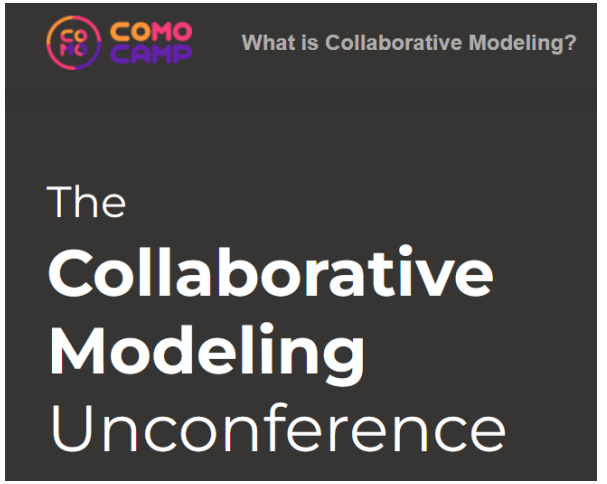


Image Reference: *Collaborative Software Design*, Evelyn van Kelle, Gien Verschatse, Kenny Baas-Schwegler

Motivation and Background

- https://comocamp.org/#como



What is Collaborative Modeling?

The essence of Collaborative Modeling lies in **cooperation** and **knowledge sharing**.
All participants bring in their different perspectives, expertise, and experiences.
This leads to a shared understanding of the problem, which helps to develop a shared software-related solution.
The goal is to achieve better and more innovative solutions through collaboration.

"Some Collaborative Modeling methods and mashups are:

Common CoMo practices are listed on the CoMo Unconference website:
https://comocamp.org/#como

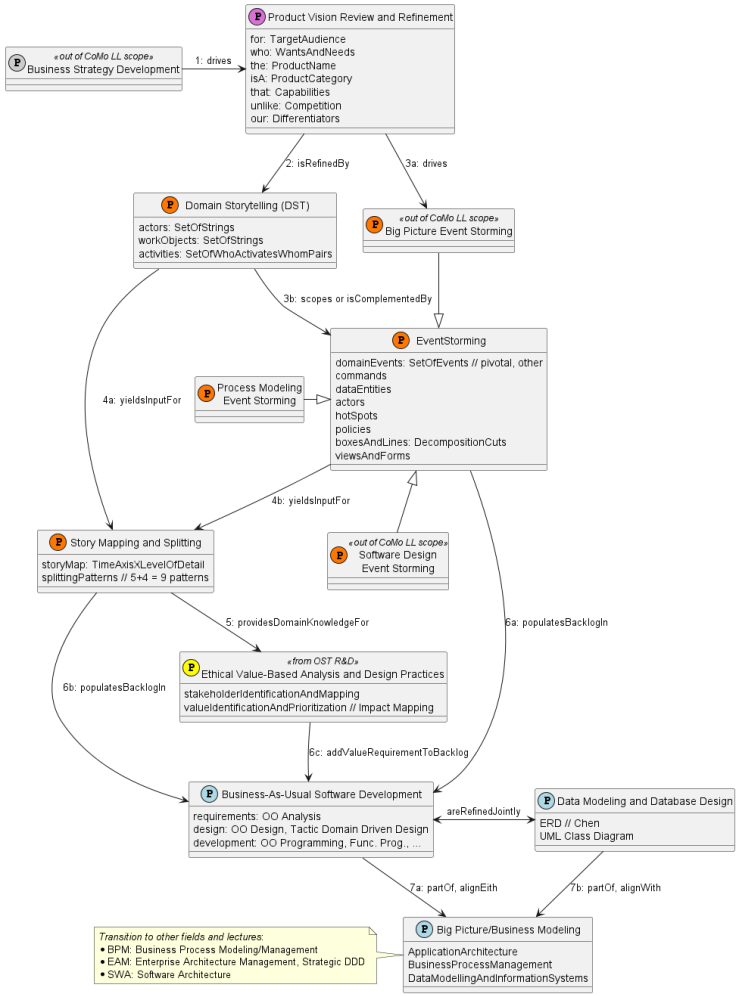
- EventStorming
- User Story Mapping
- Impact Mapping
- Domain Storytelling
- Storystorming
- Context Mapping
- Example Mapping
- Business Model Canvas
- Bounded Context Canvas
- Scenario Casting
- Event Modeling
- Heuristics Mapping”

CoMo Practices: Event Storming, Domain Storytelling

- **Event storming: "agile Business Process Modeling (BPM)"**
  - Concepts: events, commands, entities/aggregates
  - Collaborative, often on whiteboard such as miro
  - Output: e.g., process model, candidate services (APIs)
  - https://www.eventstorming.com/

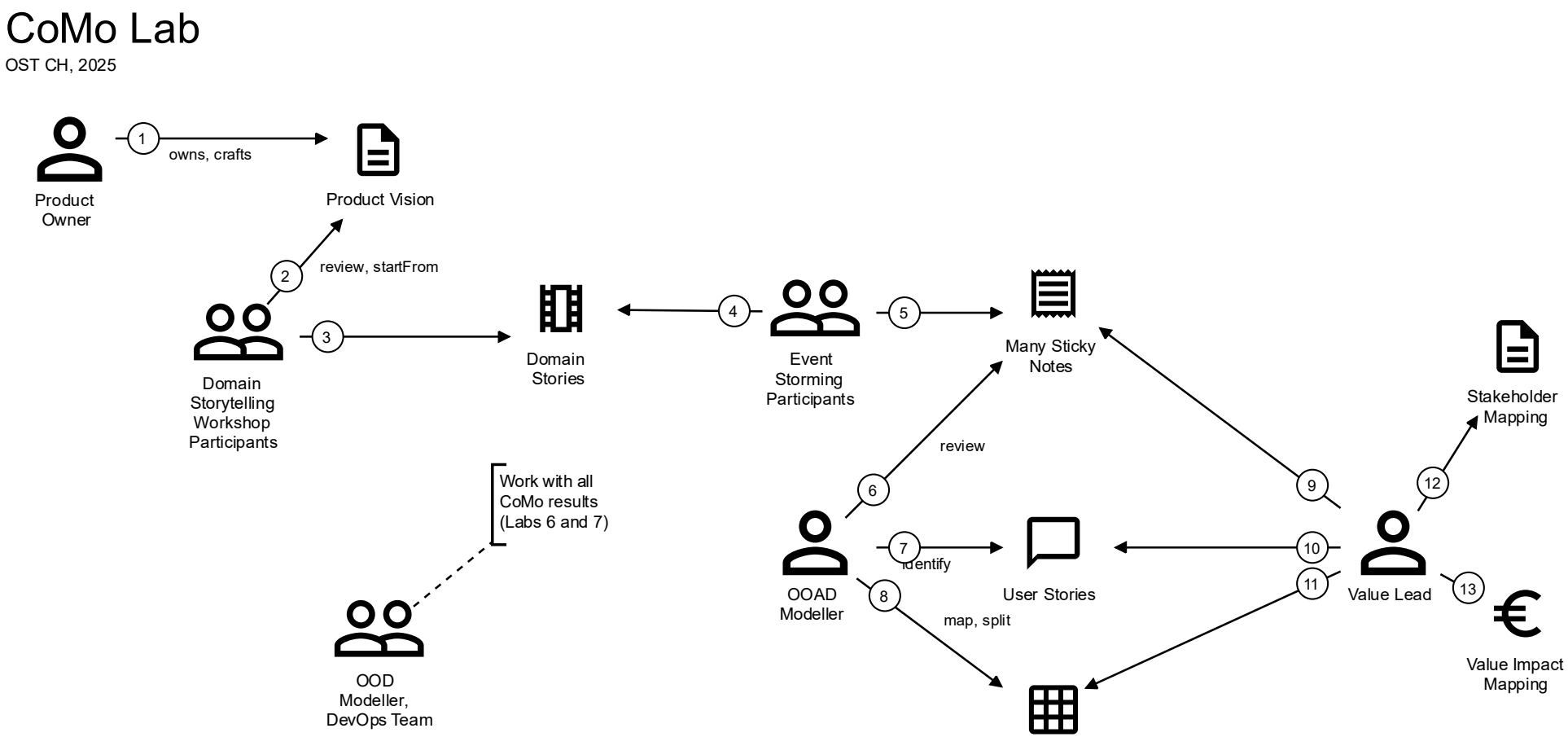
- **Domain storytelling: "agile OOAD"**
  - Pictographic language readable for domain experts (rich pictures)
  - Collaborative; modeling canvas, general-purpose or specific tools
  - User stories and domain model as output; additional usage scenarios
  - https://domainstorytelling.org/

CoMo LL Organization: 5+2 = 7 Labs



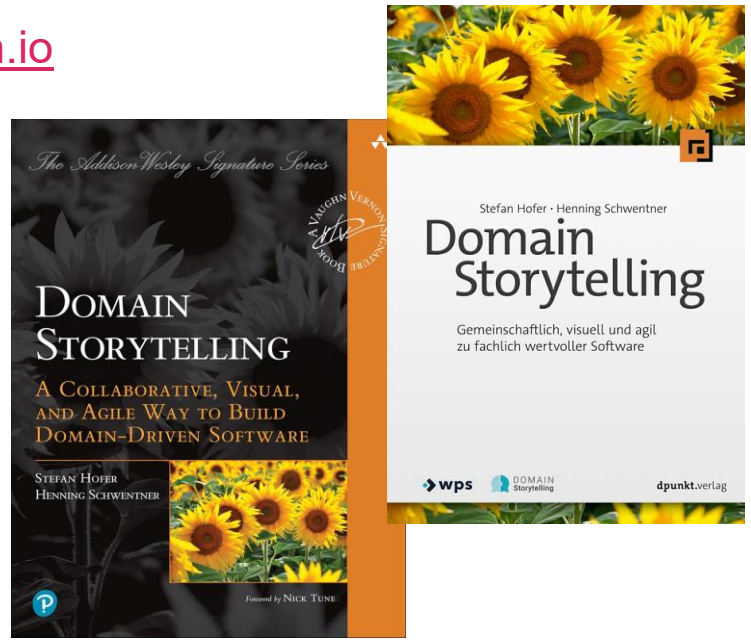
- Lab 1: Product Vision
- Lab 2: Domain Storytelling
- Lab 3: EventStorming
- Lab 4: Story Mapping and Splitting
- Lab 5: VDAD Stakeholder Mapping and Value-Impact Mapping
- Lab 6: Transition to OOAD, OOP and other Business As Usual (BAU)
- Lab 7: Return to Big Picture (Architecture) and Handover to Other Lectures
  - AppArch, BPM, DMIS, Software Engineering

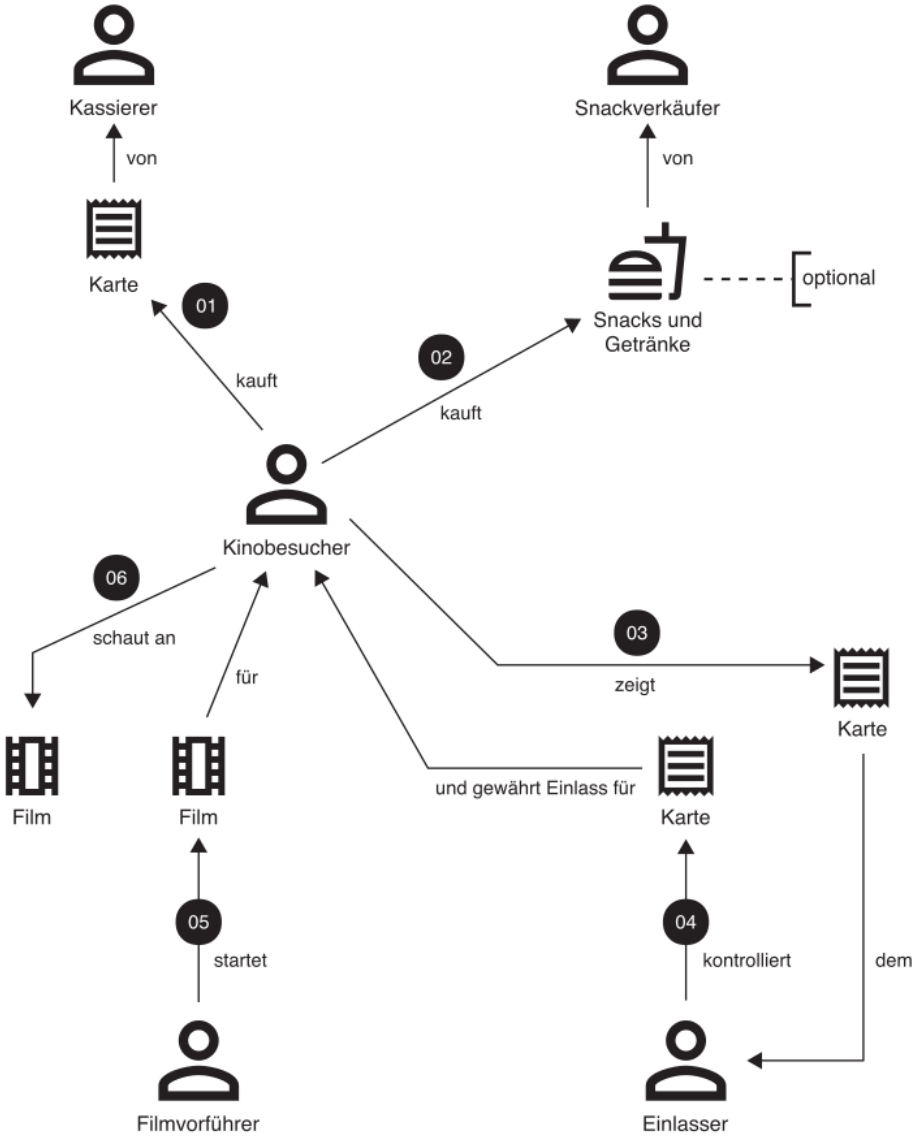
CoMo LL Organization: Input/Output Labs 1 to 5



Domain Storytelling

- **Pictographic language** readable for domain experts (rich pictures)
- domainstorytelling.org
- Collaborative; modeling canvas, general-purpose or specific tools
  - egon.io





Example Source: *Domain Storytelling*, Stefan Hofer, Henning Schwentner

EventStorming

- „The smartest approach to collaboration beyond silo boundaries“
- Flexible workshop format for collaborative exploration of complex business domains
- eventstorming.com

Image Source: Introducing EventStorming, Alberto Brandolini, leanpub.com/introducing_eventstorming

„Bridge“ to Strategic Domain-Driven Design (DDD)

- CoMo practices can be useful tools to **decompose** your domain into **Bounded Contexts**!

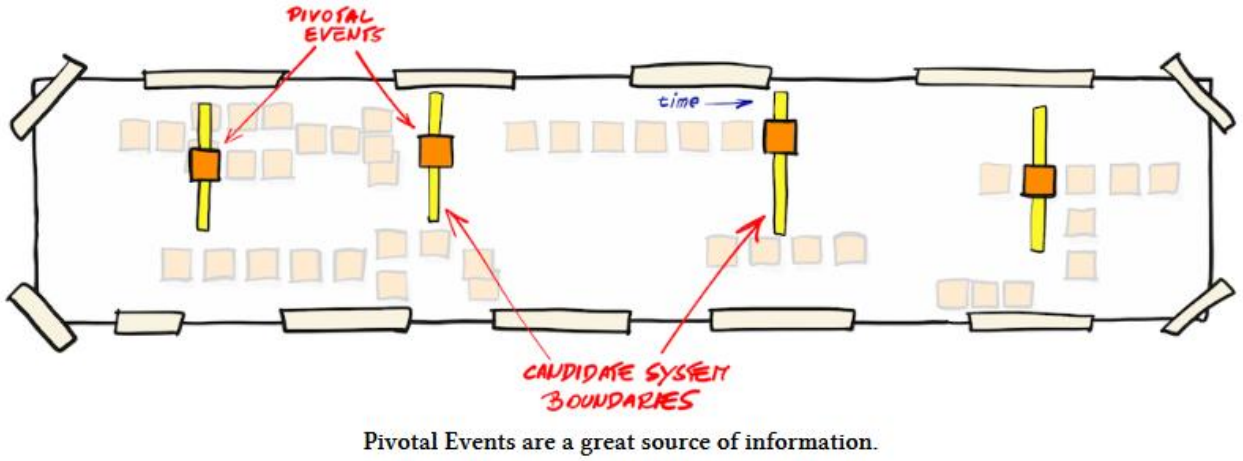


Image Source: Introducing EventStorming, Alberto Brandolini, leanpub.com/introducing_eventstorming

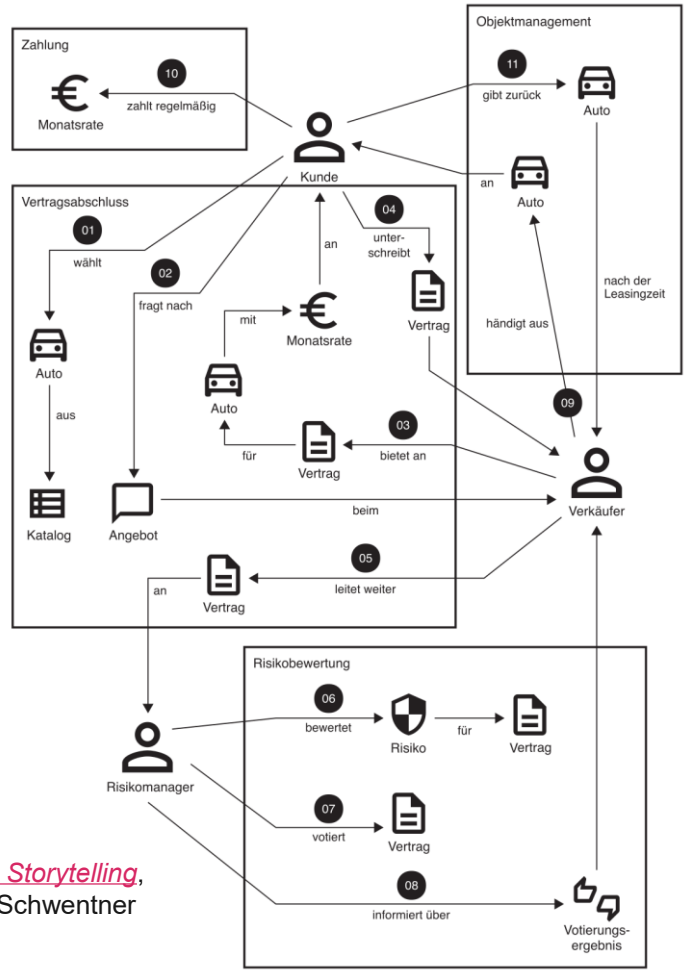


Image Source: *Domain Storytelling*, Stefan Hofer, Henning Schwentner

Strategic vs. Tactical Domain-driven Design

- **Strategic** DDD
  - Bounded Contexts
  - Relationships between Bounded Contexts (Context Mapping)
  - How do Bounded Contexts integrate
- **Tactical** DDD
  - Design of domain model inside a Bounded Context
  - Patterns such as: Aggregate, Entities, Value Objects, Domain Events, etc.

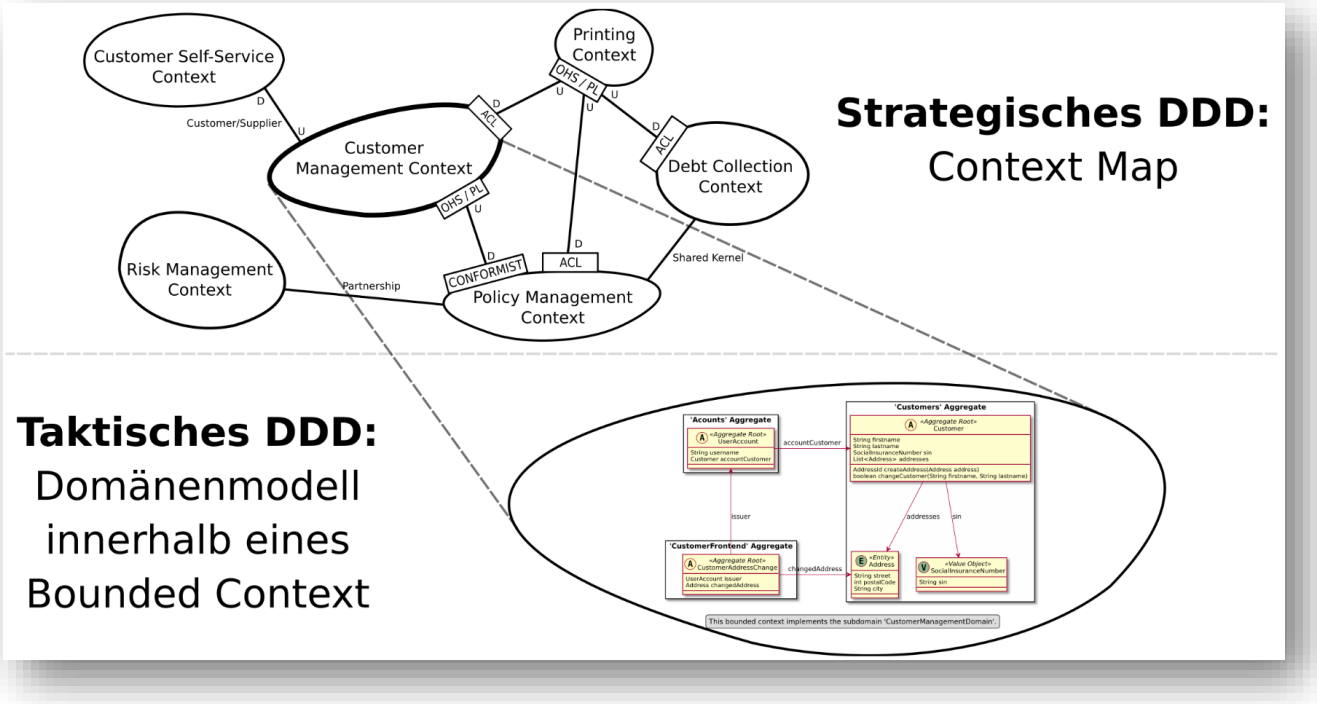
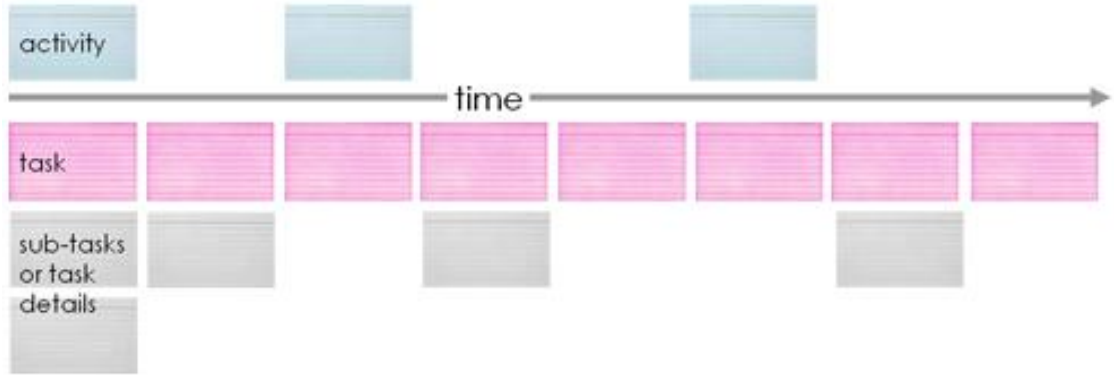


Image Source: Stefan Kapferer & Olaf Zimmermann, *«Domain-Driven Design in der Praxis - Erfahrungen mit dem Open-Source-Tool Context Mapper»*, JavaSPEKTRUM Artikel

Story Mapping & Splitting

- **Create**, **split** and **prioritize** user stories
- Ensure the meet **INVEST** properties: (agilealliance.org/glossary/invest)
  - „**I**“ndependent (of all others)
  - „**N**“egotiable (not a specific contract for features)
  - „**V**“aluable (or vertical)
  - „**E**“stimable (to a good approximation)
  - „**S**“mall (so as to fit within iteration)
  - „**T**“estable (in principle, even if there isn‘t a test for it yet)



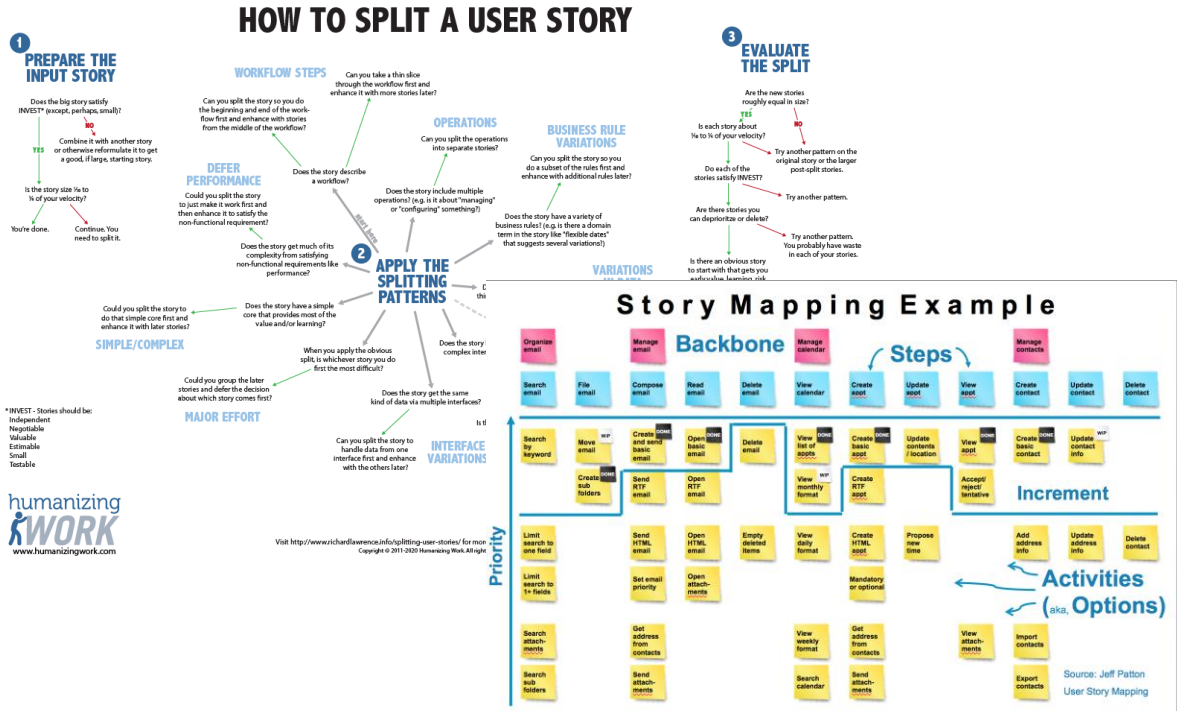


Image Sources: pattonassociates.com/the-new-backlog, humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories, cmforagile.blogspot.com/2015/07/story-telling-with-story-mapping.html

Value-Driven Analysis and Design (VDAD)

https://ethical-se.github.io/value-driven-analysis-and-design/

VDAD Process | Practices | Why VDAD? (User Stories) | Tools | Background | Glossary

Value-Driven Analysis and Design (VDAD)

License CC BY 4.0

TL;DR: Technology should support humanity and avoid harming people and their values; creators create solutions that promote positive values and aim to reduce harm to our soc Design (VDAD) process aims to combine value-driven approaches with state-of-the-art s Driven Design (DDD).

Value-Driven Analysis and Design: Applying Domain-Driven Practices in Ethical Software Engineering

Stefan Kapferer
Eastern Switzerland University of Applied Sciences (OST)
Switzerland
stefan.kapferer@ost.ch

Olaf Zimmermann
Eastern Switzerland University of Applied Sciences (OST)
Switzerland
olaf.zimmermann@ost.ch

Mirko Stocker
Eastern Switzerland University of Applied Sciences (OST)
Switzerland
mirko.stocker@ost.ch

Abstract

Business goals and economic values typically drive decisions in digitalization efforts and software development projects. There seems to be a lack of awareness that other values, such as ethical ones, should be respected to produce systems that do not harm human beings. Ethical values often are subjective; different stakeholders have different interests and priorities. So how can we make the values of all stakeholders of a software-intensive system transparent, especially the negative and positive impacts of the system on those stakeholders and their values, so that adequate decisions can be made and tradeoffs be found? The process pattern described in this paper suggests combining domain-driven analysis and design practices (modeling the domain, the software design, as well as its impacts and consequences) with value-based systems engineering (eliciting and prioritizing values and deriving value requirements from them) to make values and value requirements first-class citizens of the engineering process that go through the same refinement steps as business requirements and technical quality attributes and receive the same attention during design and implementation. We exemplify our process pattern in an online shop scenario that wants to add a same-day delivery feature, which has different ethical ramifications for several groups of system stakeholders.

CCS Concepts

• **Software and its engineering** → **Patterns**; **Requirements analysis**; **Software design tradeoffs**; • **Social and professional topics** → **Codes of ethics**.

Keywords

architectural decisions, domain modelling, ethical software engineering, requirements engineering, software design, value-driven systems engineering

ACM Reference Format:
Stefan Kapferer, Olaf Zimmermann, and Mirko Stocker. 2024. Value Driven Analysis and Design: Applying Domain Driven Practices in Ethical Software Engineering. In *29th European Conference on Pattern Languages of Programs, People, and Practices (EuroPLoP 2024), July 03–07, 2024, Irsee, Germany.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3698322.3698332

EuroPLoP 2024, July 03–07, 2024, Irsee, Germany

ACM ISBN 979-8-4007-1683-6/24/07
https://doi.org/10.1145/3698322.3698332

1 Introduction

Many recent developments in software and technology, such as artificial intelligence (AI) and robotics, have shown that systems that are released to the market not only have positive consequences on human and ethical values but negative ones as well. This is, however, not a recent phenomenon. The systems provided by the big tech companies that we all use on a daily basis have been reported to promote addictive behavior [4], make us lonely [24, 28], spy on us [6, 27] and deal with our stolen personal data [12, 20]. Codes of conduct, standards and guidelines towards ethical and value-based engineering exist [1, 15, 34] but do not always receive the amount of attention they deserve.

Software development is driven by requirements – functional and non-functional ones. Software engineers and architects need to know what the software should be able to do and which important qualities it should have. These requirements can include what the users should be able to achieve with the software. Still, they can also just focus on how the producers or operators of the software can achieve their business interests and economic goals. Often, functional requirements are based on the knowledge of the company or business experts who claim to know what the users and stakeholders need and want from the system. The Non-Functional Requirements (NFRs) usually focus on qualities such as *security*, *maintainability*, *scalability*, etc. that have to be respected in order to make the system work. When initiating software development projects, we have to consider whether digitalizing domains and processes is worth it; if the negative impacts of the system outweigh the positive ones, it might be in order not to go forward.

Development teams applying Domain-Driven Design (DDD) [14] and domain-driven requirements engineering practices, including Domain Storytelling [19] and Event Storming [8], aim to understand and model domain knowledge in order to find software architectures and designs that align well with the problem domain and take organizational structures into account (e.g., team topologies during development and operations). DDD focuses on communication between software engineers and business experts to establish that knowledge. A shared understanding of the problem and a so-called *ubiquitous language* shall be established. The idea of analyzing and modeling the business domain and then finding technically fitting software designs was already described by earlier Object-Oriented Analysis and Design (OOAD), e.g., [7]. Placing people and communication at the center of development processes was later promoted by the Manifesto for Agile Software Development as well.[1] We

[1] https://agilemanifesto.org

Featured Practices

- Product Vision
- Domain Story
- Event Storming
- Story Mapping and Splitting
- Stakeholder Mapping, Value Impact Mapping

Concepts in the Practices

For

whose

[product]

[genre]

that

[competition]

our product

Actors

Activities

Work Objects

Groupings

Annotations

Events, Timeline

Commands

Data

Processes, Systems

Subdomains, Contexts

Hot Spots

Policies

Views and Forms

Role

Feature

Benefit

Story Map

Workflow

Operations

Biz. Rule

Data Var.

Interface Variation

Stakeholder

Value

Impact

Concept Mapping

Concept vs. Lab	1: PV	2: DS	3: ES	4: SM, SS	5: VDAD
User	"For"	Actor	User	"As a..." part of story	Stakeholder
Goals	"whose", "that"	(annotations)	(hot spots)	"so that" part of story	Values
Feature	"that", "our product"	Activity	Command	"I want to" part of story	n/a
Artifact	n/a	Work Object	Data (Entity, Aggregate)	nouns in "I want to" part, (data variation splitting pattern)	n/a
UI	n/a	n/a	View/form (read model)	(channel splitting pattern)	n/a
Logic	n/a	(Activity)	Policy	(business rule/operation splitting patterns)	
Aggregation	n/a	Process	Grouping	(vertical axis in story map)	(Step 7 in VDAD)
Timeline	n/a	Sequence numbers of activities	Horizontal axis	Horizontal axis of story map	n/a

Application Example: Fintech Startup

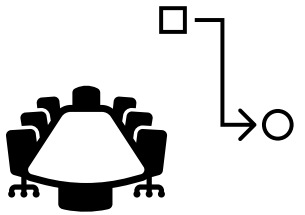
- https://ozimmer.ch/modeling/2022/11/23/ContextMapperInsights.html:
  - “DDD and event storming were initially used for familiarization with the domain and the existing systems and then for joint creative work. ”

Wearonize AG, a fintech start-up for future-oriented wearable payment solutions for banks and producers alike, has raised more than EUR 1.6 million via the investment and crowdfunding platform Companisto. Busines angels, commercial and private investors participated in the round.

Artifacts (from Lab to Lab)

3 EventStorming Output
(Process Model)

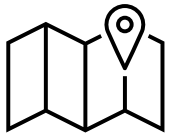
5: VDAD Output
(Stakeholder Map,
Value-Impact Map)

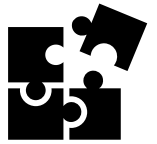


2: Domain Story

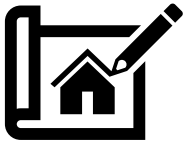
1: Product Vision

Epics & User Story Backlog
(Role-Feature-Benefit Structure)





4: Story Map, Story Splits

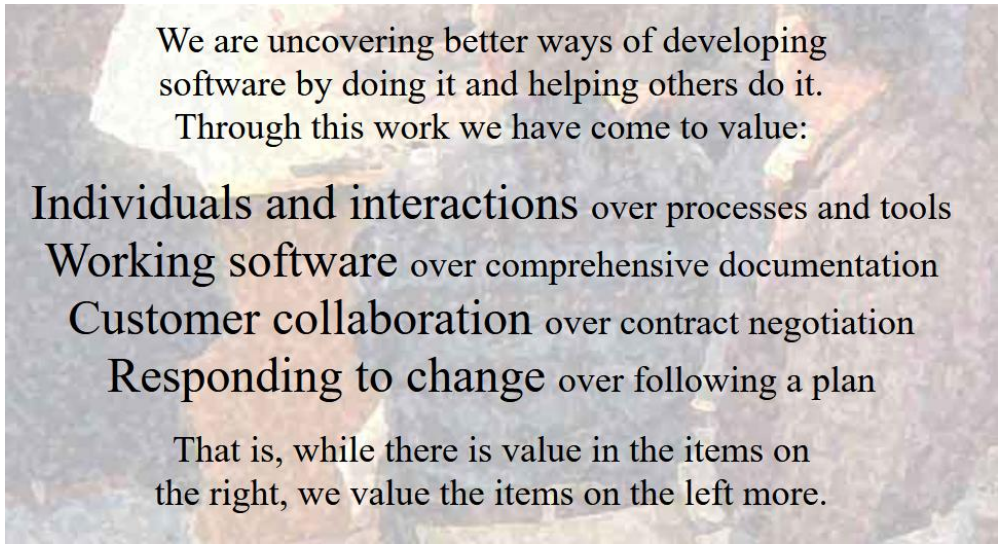


7: Big Picture Models
(Architecture, Data, …)

6: Realization Prototypes
(Domain-Driven Design,
Development, Operations)

Summary: Collaborative Modelling is…

- Driven by industry, not academia
- Modern, agile touch on well established engineering practices
- Emphasis on **people and interactions** rather than automation and correctness
  - In line with values in Agile Manifesto (note that items on the right do have value!)
  - **Formal models** with UML, BPMN, etc. **are still valueable** for precision and towards implementation.



- Recommendation or „**best practice**“: **Start collaboratively**, then **refine** into more formal models as needed.

Resources (Links, Books, etc.)

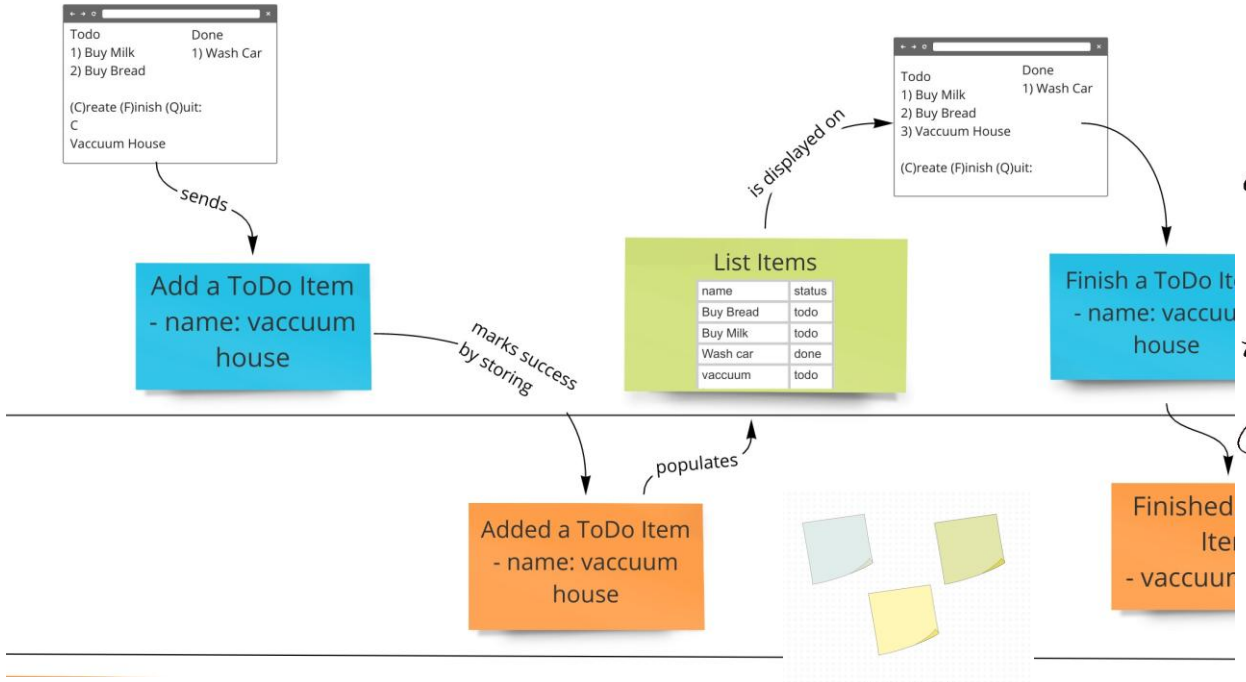
- *See CoMo LL Module Overview PDF*

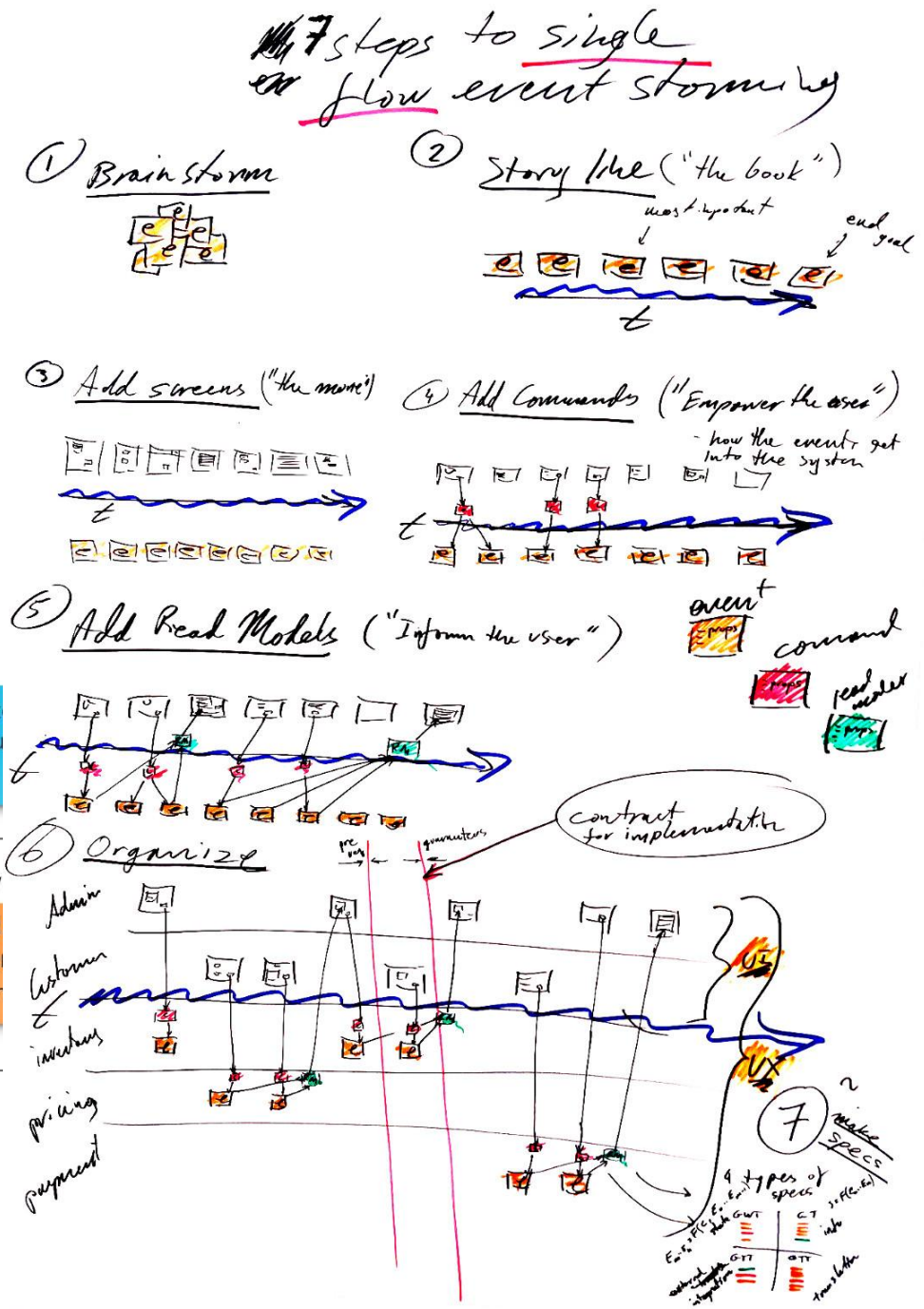
Questions (Repetition, Practice)

- *See PDFs of Labs 0 to Lab 7*

Outlook: Event Modeling

- https://eventmodeling.org/





OST
Ostschweizer
Fachhochschule

Collaborative Modelling: Lesson 0
BACKGROUND INFORMATION

ITBO Learning Lab 2, Initiative 1 (ZIOL, KAPS)

Spring Term 2025

Departement Informatik

Lab 6: From Collab. Modeling to OOAD/DDD, OOP

- *See sample solutions to Lab 6 and Lab 7 (PDFs)*

	OOA	**OOD**	**DDD**	**OOP**
PV				
DST				
ES				
SM, SS				
VDAD				

Lab 7: Big Picture and Transition (Putting it All Together)

- *See sample solutions to Lab 6 and Lab 7 (PDFs)*

	Software Architecture	Software Engineering	BPM(N)	Data Modeling
PV				
DST				
ES				
SM, SS				
VDAD				

Pros and Cons

- *See questions and sample answers in steps of Labs 0 to Lab 7, as well as recap/reflection sections at the end of each PDF*

- See DST book
- See ES articles

- See “DDD Distilled” book, later chapters
- See “User Story Mapping” book

Frequently Asked Questions (FAQ)

- *See PDFs of Labs 0 to Lab 7*
  - *Student version 1: task assignments, questions*
  - *Student version 2: version 1 plus sample solutions*
  - *Instructor version: student version 2 plus lecturer notes*