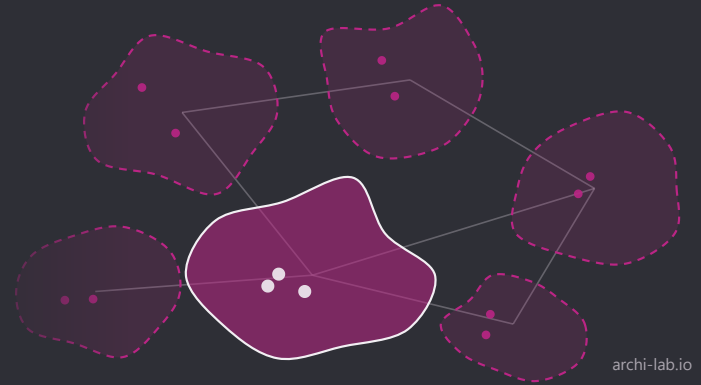


MASTER ELECTIVE · WINTER SEMESTER 2026/27

DDD

Domain-Driven Design of Large Software Systems

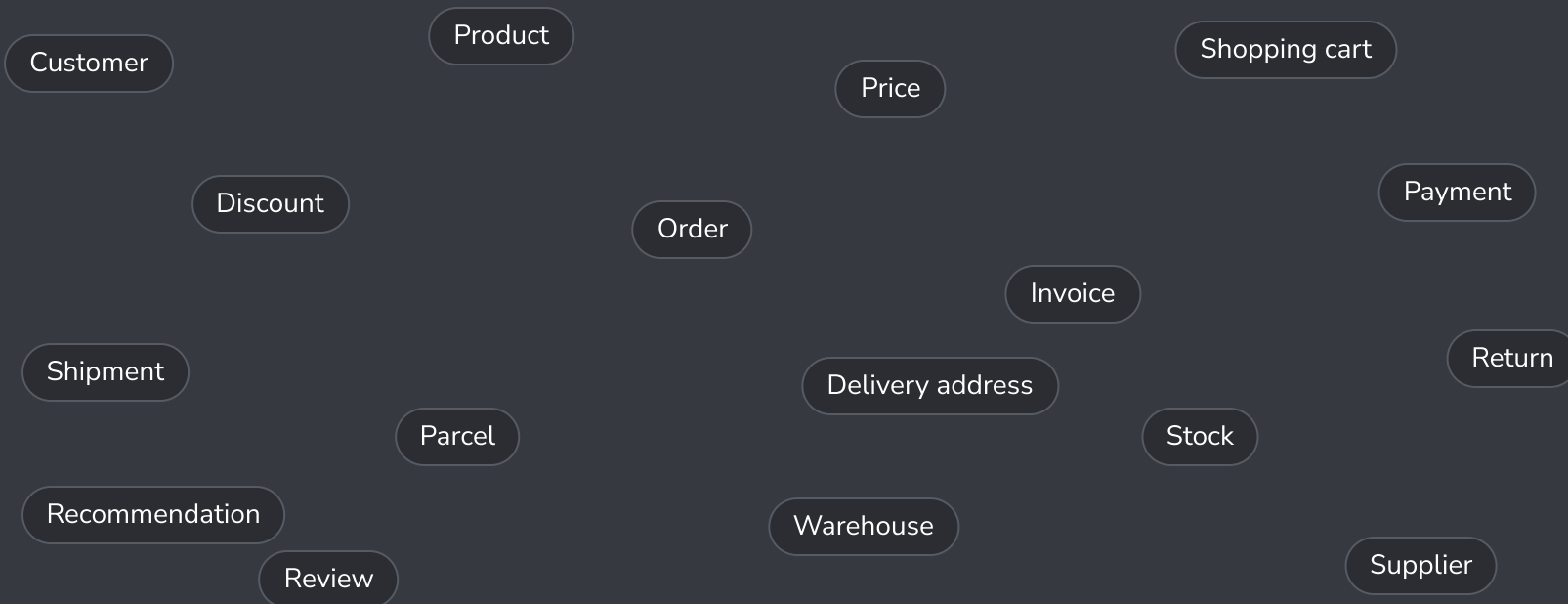


Why Domain-Driven Design?

The question every architect has to answer first

You are the architect. Where do you cut?

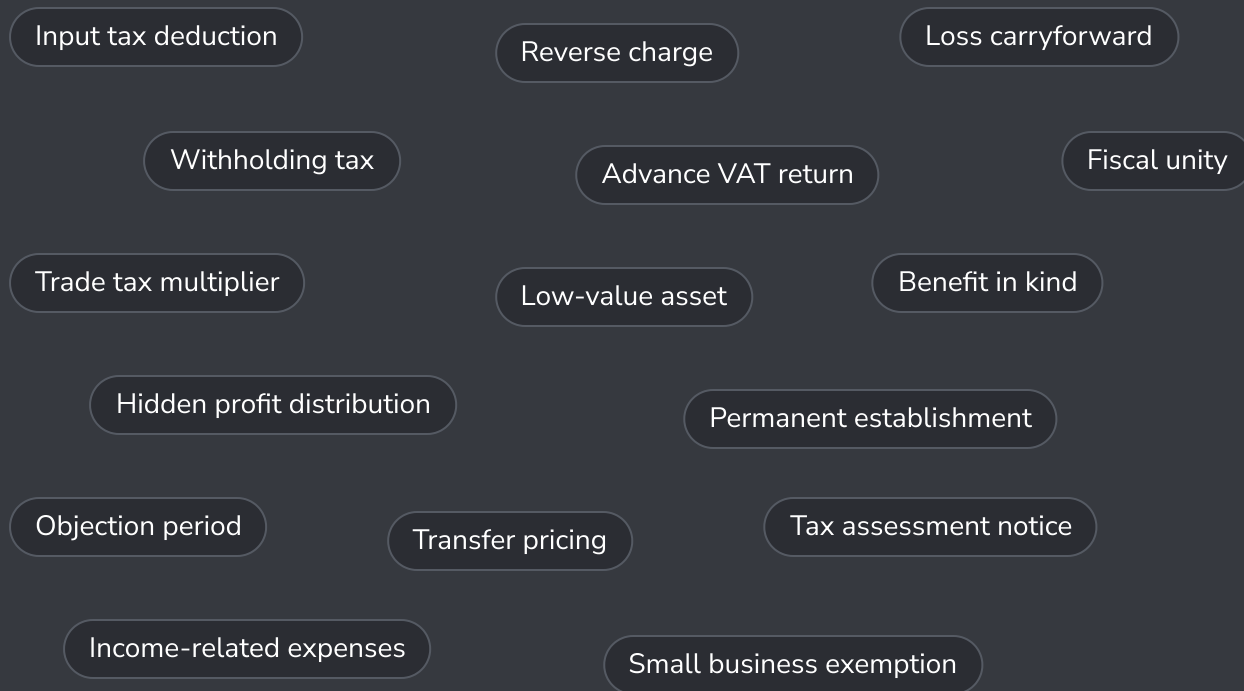
An online shop, like the one you last ordered from. Thirty developers, five teams, one system.



Which team owns what? Where do the boundaries run?

And now this one. Where do you cut?

Software for a tax consultancy: income tax, VAT, corporate clients. Same team size, same question.



AUF DEUTSCH

Vorsteuerabzug · Reverse-Charge-Verfahren · Verlustvortrag · Kapitalertragsteuer · Umsatzsteuer-Voranmeldung · Organschaft · Gewerbesteuer-Hebesatz · geringwertiges Wirtschaftsgut · geldwerter Vorteil · verdeckte Gewinnausschüttung · Betriebsstätte · Einspruchsfrist · Verrechnungspreise · Steuerbescheid · Werbungskosten · Kleinunternehmerregelung

You don't understand half of these words. Where does your intuition take you now?

Every framework tells you to cut

arc42

"the static decomposition of the system into building blocks ... as well as their dependencies"

Section 5, building block view

"decisions about the top-level decomposition of the system"

Section 4, solution strategy

iSAQB, the architects' curriculum

"make structural decisions regarding system decomposition and building-block structure"

CPSA Foundation Level, LG 03-02, level R1: "being able"

ISO/IEC/IEEE 42010, the standard

"fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution"

Definition of architecture, 42010:2011

Rightly so: the cut is the first thing an architect has to get right.

... but none of them tells you where

THE ARCHITECTURE PROCESS CHAIN (arc42 method, six activities)



Yields: the building block view

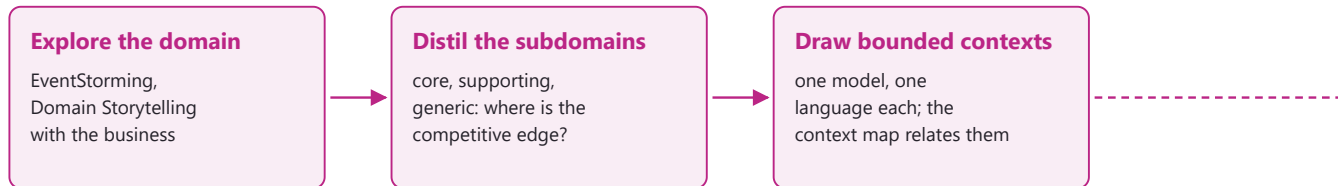
"the static decomposition of the system into building blocks (modules, components, subsystems, ...) as well as their dependencies"



But how do I find where the boundaries run?

Template, method and standard describe and document the cut. None of them says how to find it.

DOMAIN-DRIVEN DESIGN: A METHOD FOR THE CUT



the boundaries feed the building block view

Even the curriculum says so

iSAQB CPSA FOUNDATION LEVEL, CURRICULUM 2025.1

R1

Being able

to put into practice independently

"make structural decisions regarding system decomposition and building-block structure" (LG 03-02)

R2

Understanding

to understand in principle

the "what" is a must-do,
the "how" a nice-to-know

R3

Knowing

can enhance understanding and motivate the topic

"domain-driven design", listed among the approaches and heuristics for architecture development (LG 03-03)

Source: iSAQB CPSA-F curriculum 2025.1-rev2 (public.isaqb.org/curriculum-foundation), LG 03-02 and LG 03-03. Diagram: own illustration.



"In two decades as an architect, every framework asked me to decompose the system. Domain-Driven Design was the first method that told me **how**."

If we don't know better, we cut by technology

Frontend team

Customer · Product · Order · Payment · Shipment · Stock · Return · ...

Backend team

Customer · Product · Order · Payment · Shipment · Stock · Return · ...

Database team

Customer · Product · Order · Payment · Shipment · Stock · Return · ...

- A new feature touches every layer
- So it touches every team
- And every release has to be coordinated

"Once any of these layers gets too big you should split your top level into domain oriented modules which are internally layered."

Martin Fowler, 2015

Every business concept lives in every layer.

Why the cut matters

What changes together

"... one begins with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such a decision from the others."

David L. Parnas, 1972: On the Criteria To Be Used in Decomposing Systems into Modules

Who talks to whom

"... organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

Melvin E. Conway, 1968: How Do Committees Invent?

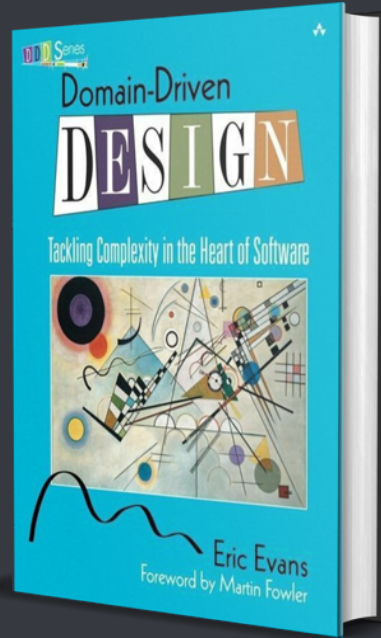
What it costs to change

"Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."

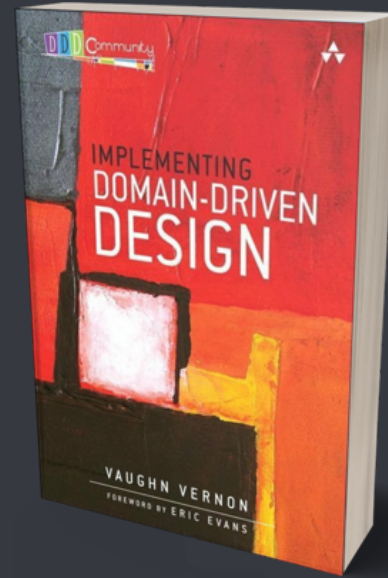
Grady Booch, 2006

The cut is one of "the things that people perceive as hard to change". Martin Fowler, 2003

Enter Domain-Driven Design



Eric Evans, 2003
"the blue book"



Vaughn Vernon, 2013
"the red book"



"The crucial point that I am concerned with is also the reason why the concept is called domain-driven design: When we develop software, our focus should not be primarily on the **technologies** we use. Instead, our main focus should be on the business side, i.e. on the specialist area that we want to support with our software, in short: on the **domain**. That's what I meant by domain-driven design."

Eric Evans

From an interview published in German by entwickler.de, translated back into English for this slide (English, German, English), so the wording is not Evans' own.
entwickler.de/ddd/microservices-konnen-teams-dabei-helfen-ddd-zu-praktizieren-interview-mit-eric-evans

Technical complexity vs. domain complexity

Technical complexity

Frameworks, cloud, messaging, databases. Our home turf: we learn it once and reuse it everywhere.

Domain complexity

The business itself: its rules, exceptions and language. New in every project, and owned by someone else.

"Car insurance is easy, that's always the same, we'll manage this.

But life insurance is tricky, can we skip this?"

Domain-Driven Design in four ideas

- 1 The domain is the starting point**
for the whole design process, not the technology
- 2 Complexity is handled in iterations**
you will not get the model right the first time; DDD expects you to refine it as you learn
- 3 A Ubiquitous Language, based on the domain model**
one language shared by business and developers, from the workshop wall to the code
- 4 Many models, each in its own Bounded Context**
every context has its own language, model, business rules and decisions. We give up non-redundancy on purpose, so that each context is self-contained

DDD: cut along the business

1

Explore the domain

Together with the business:
EventStorming, Domain Storytelling.
What happens, in whose words?

2

Distil the subdomains

Core, supporting, generic. Where is the competitive edge, and where do we buy off the shelf?

3

Draw bounded contexts

One model and one language per context. The context map says how they relate.

Not a template for documenting the cut, but a method for **finding** it.

What you gain

Fast, agile development



Autonomous teams



Decoupled business contexts

Bounded Contexts

In the words of the people who shaped it

"Developers have to steep themselves in the domain to build up knowledge of the business."

Eric Evans, Domain-Driven Design, 2003

"... total unification of the domain model for a large system will not be feasible or cost-effective."

Eric Evans, Domain-Driven Design, 2003

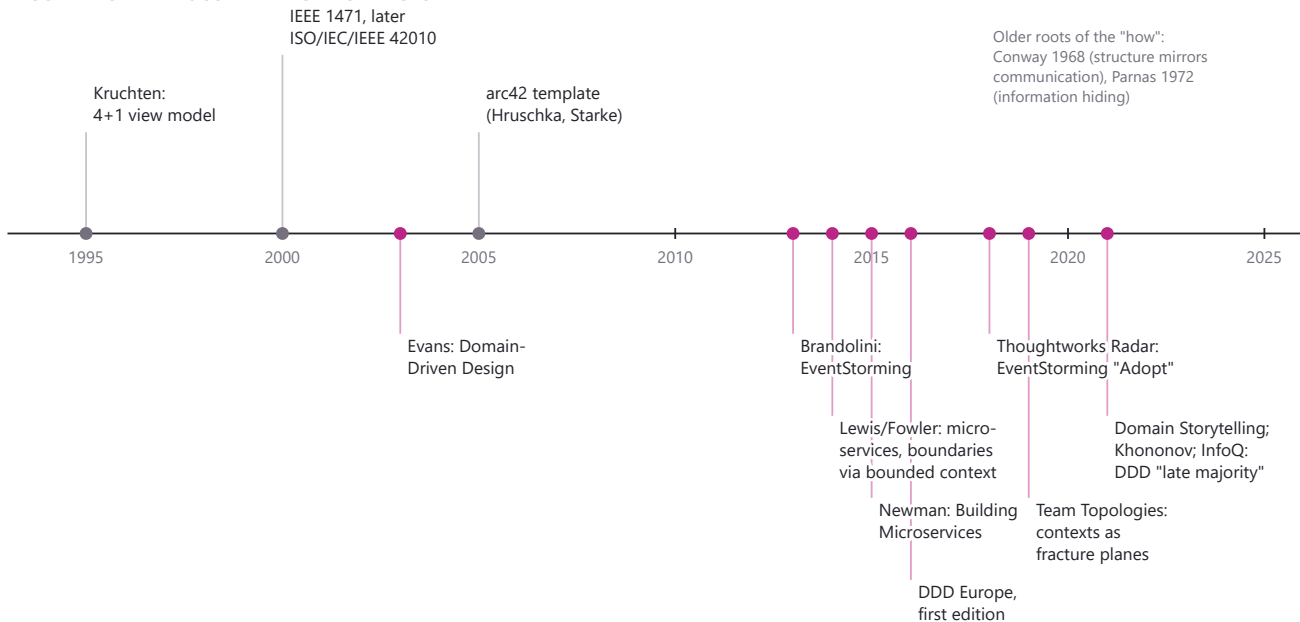
"It's developers' (mis)understanding, not domain experts' knowledge, that gets released in production."

Alberto Brandolini, creator of EventStorming

Learn the domain first, accept several models, and make the learning a team activity.

When the cut got expensive, DDD took off

DESCRIBING AND DOCUMENTING ARCHITECTURE

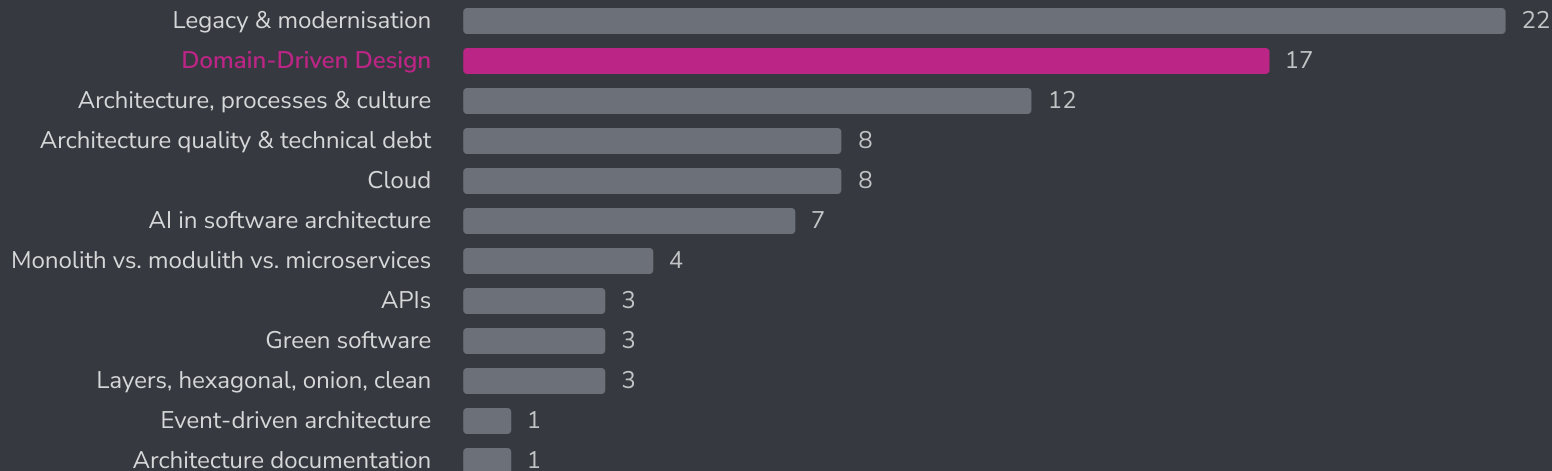


FINDING THE BOUNDARIES

Sources: see motivation-slides.md. Diagram: own illustration.

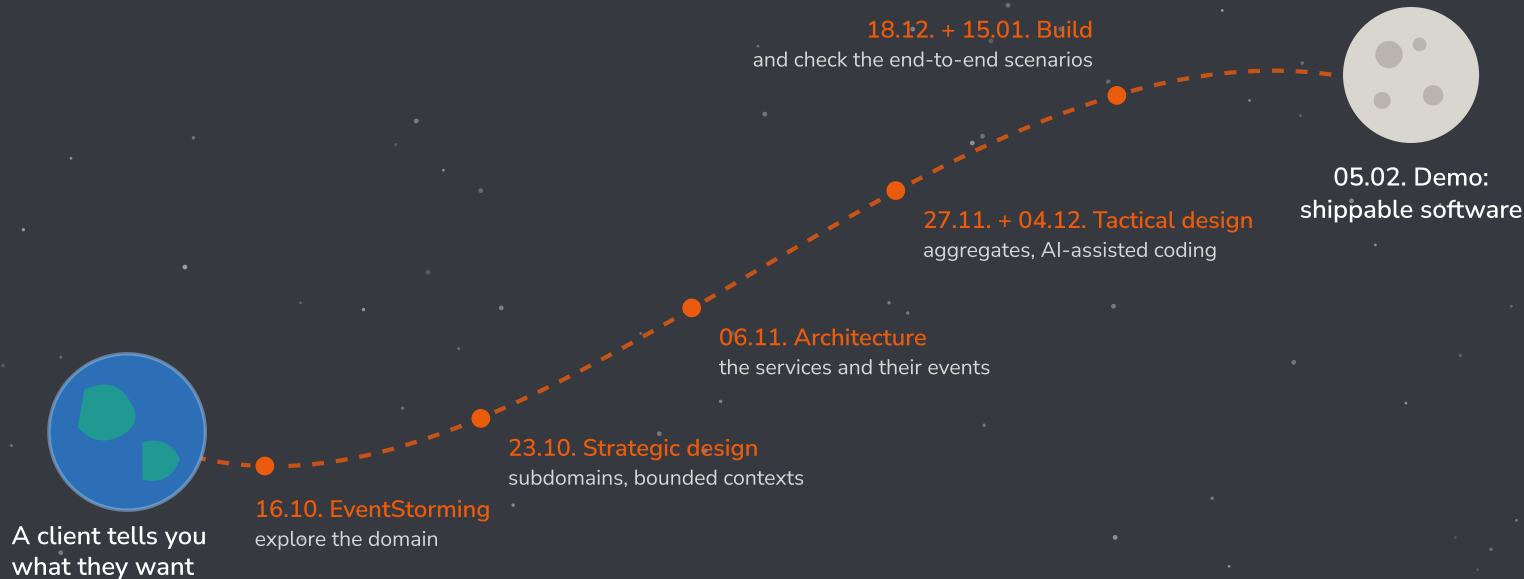
... and it is still there: OOP 2025

OOP in Munich: one of Europe's oldest software conferences (since 1992), about 2,000 participants. 61 of its 158 talks were about software architecture. What were they about?



Own evaluation of the OOP 2025 programme; a talk can cover several aspects.

What this means for you: a journey



A blueprint you can take to any project: from a client telling you what they want, to software you can ship. And you will make the cut yourselves.