

# **Riding the elevator: Domain Driven Design in the Penthouse**



**MICHAEL PLÖD**  
FELLOW

# Michael Plöd

Fellow at INNOQ

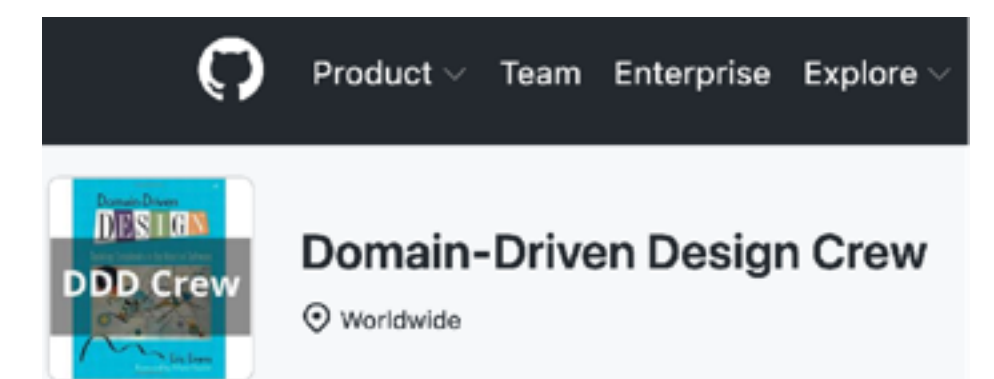
Mastodon (or Twitter): @bitboss@mastodon.social

LinkedIn: <https://www.linkedin.com/in/michael-ploed/>

Current consulting topics:

- Domain-Driven Design
- Team Topologies
- Transformation from IT Delivery to digital product orgs

Regular speaker at (inter-)national conferences and author of a book + various articles







<https://architectelevator.com/>

**Modern architects align organization and technology, reduce friction, and chart transformation journeys. In addition to working with UML and architecture styles, such architects ride the Architect Elevator from the penthouse, where the business strategy is set, to the engine room, where the enabling technologies are implemented. They shun popular buzzwords in favor of a clear strategy defined by conscious decision making.**

## **Gregor Hohpe**

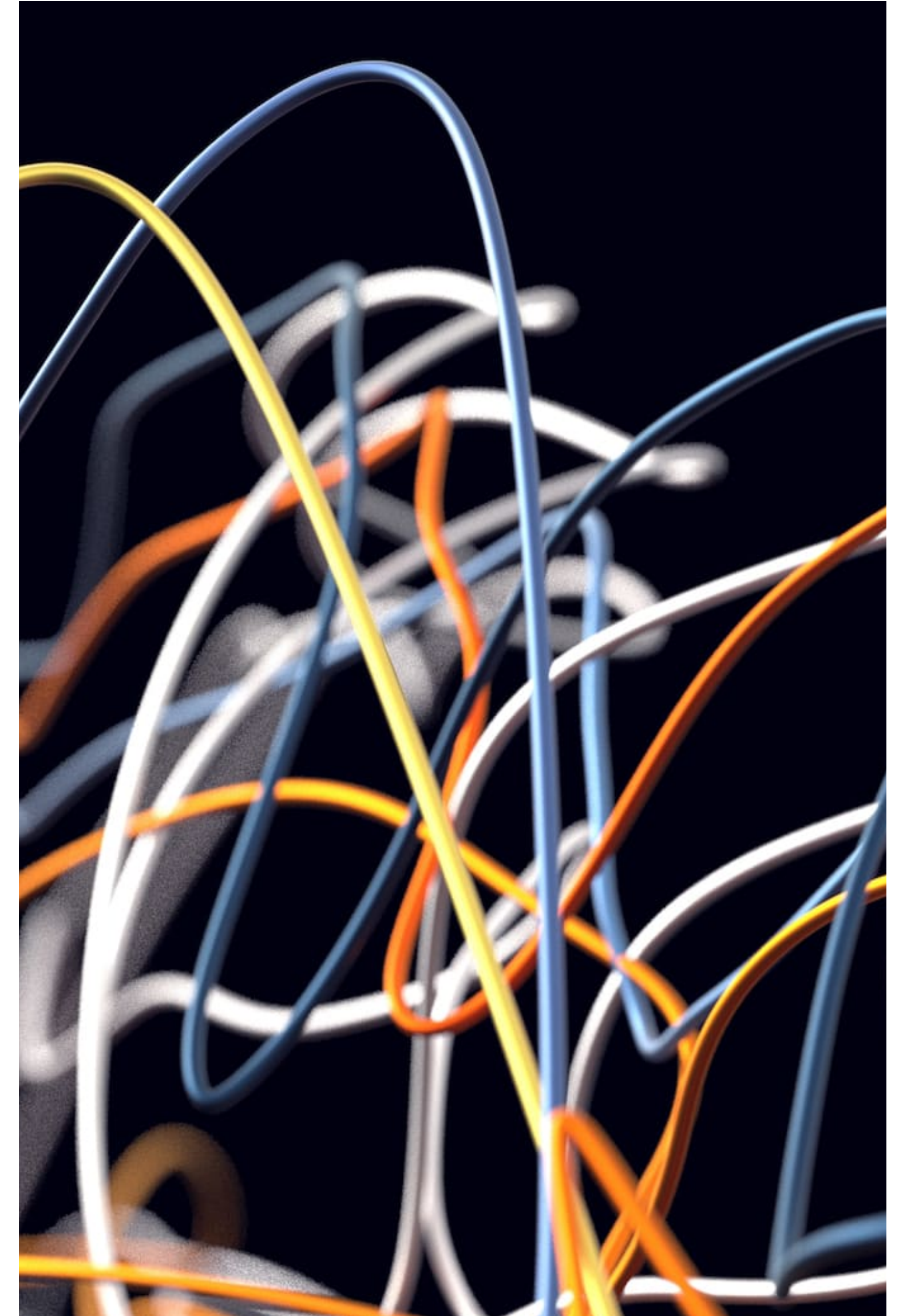
Author of „The Software Architect Elevator“





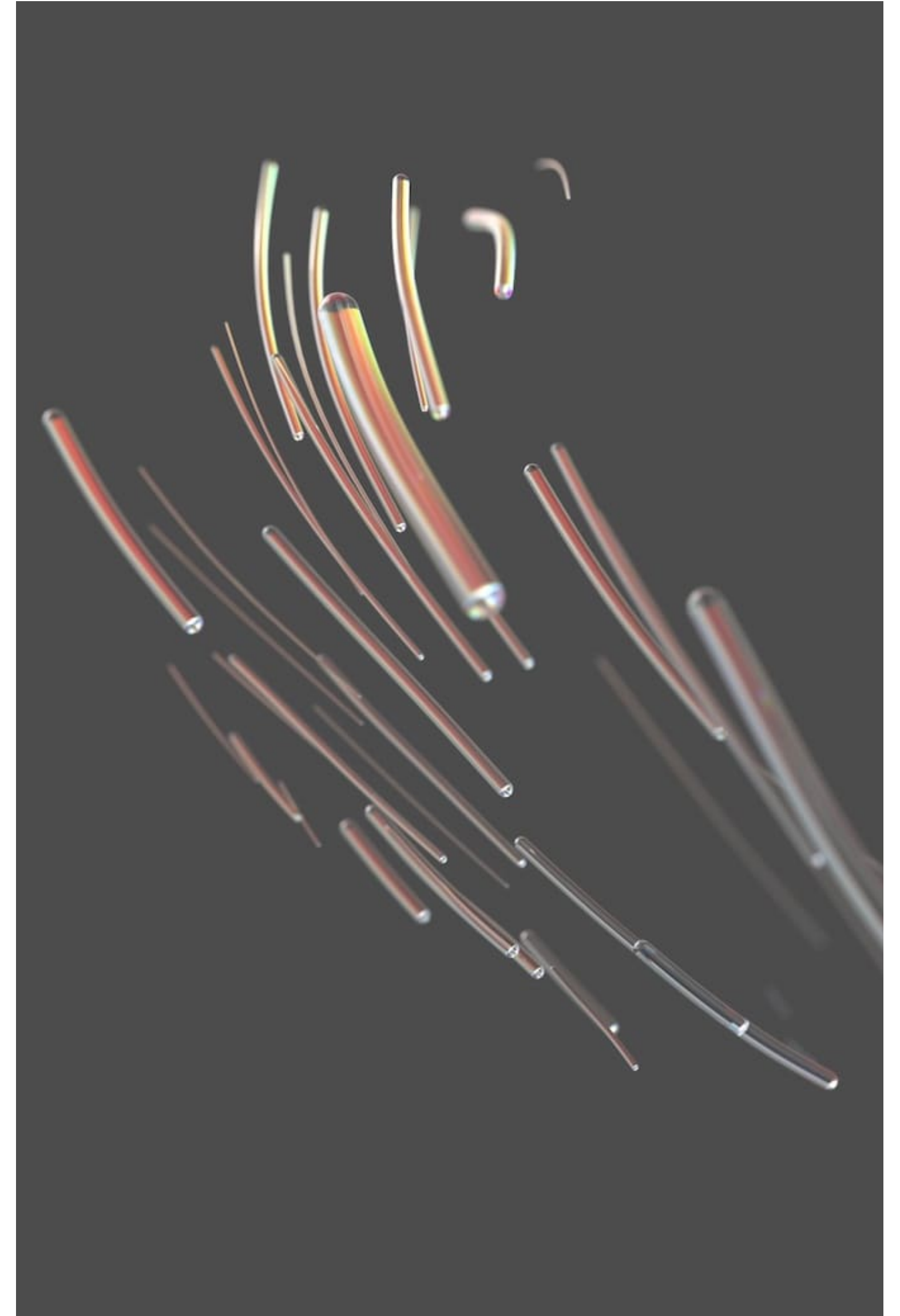
# Topics in the penthouse

- Make or Buy decisions
- How do I make my org more agile?
- How do I structure my teams?
- How can we transform our existing legacy software (to the cloud)?
- How can we become a product-driven org?
- Digitalization
- How do we differentiate in a digital market?
- How can our teams become more autonomous?

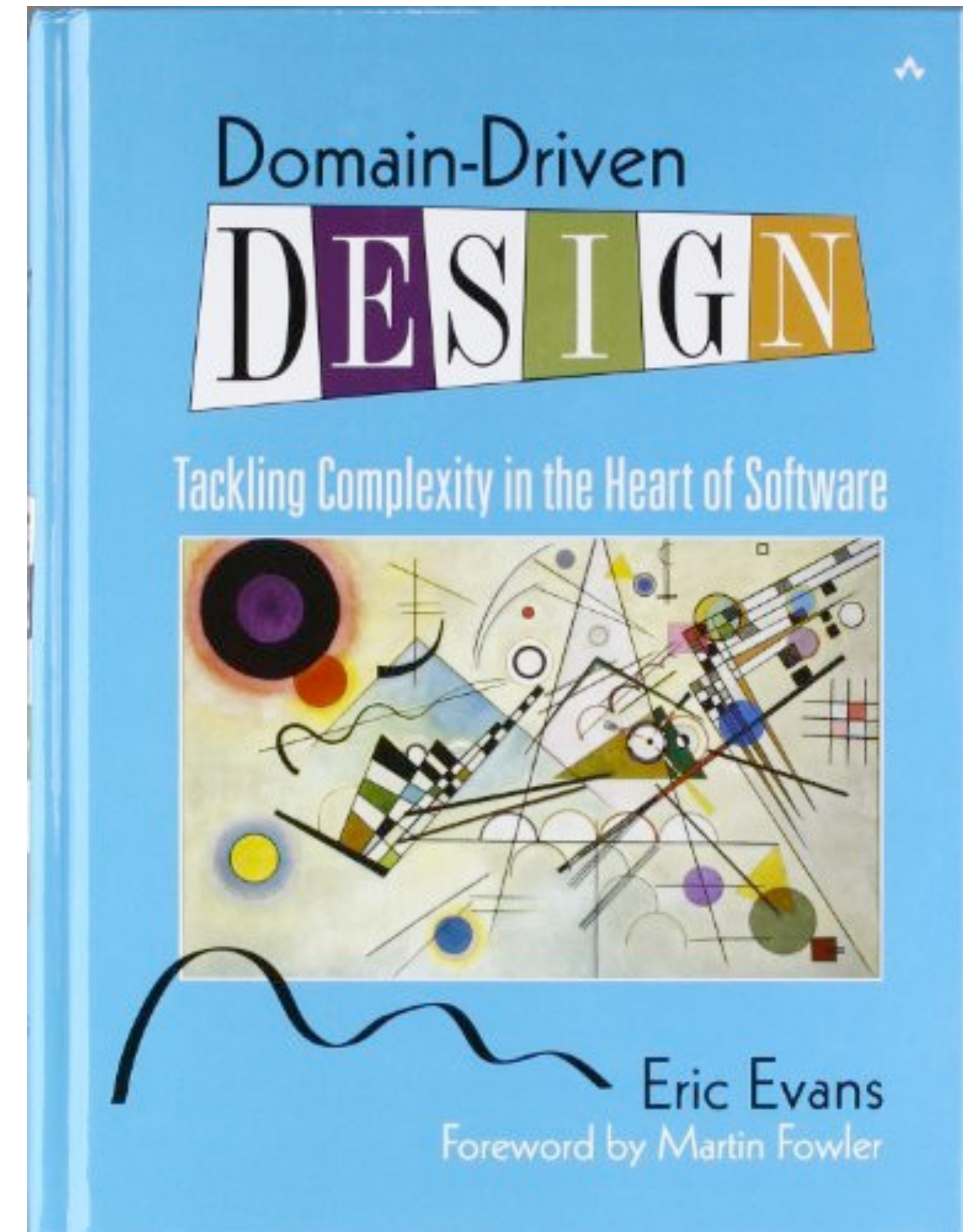


# DDD Topics

- Domain Modeling
- Identifying boundaries
- Shared understanding between domain experts and developers
- Categorizing Subdomains
- Context Mapping
- Iterative design work







**How could they complement each other?**



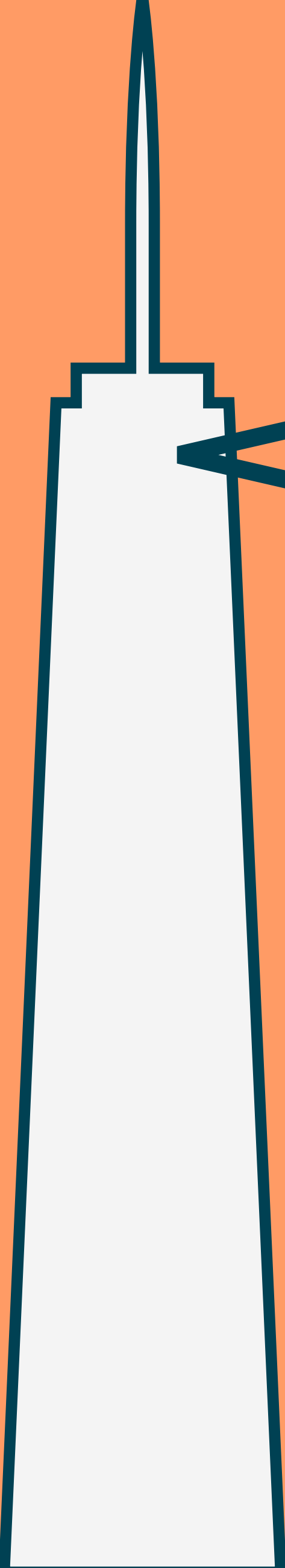
Modern architects **align organization and technology**, reduce friction, and chart transformation journeys. In addition to working with UML and architecture styles, such architects ride the Architect Elevator from the penthouse, where the **business strategy** is set, to the engine room, where the enabling technologies are **implemented**. They shun popular buzzwords in favor of a **clear strategy defined by conscious decision making**.

## Gregor Hohpe

Author of „The Software Architect Elevator“

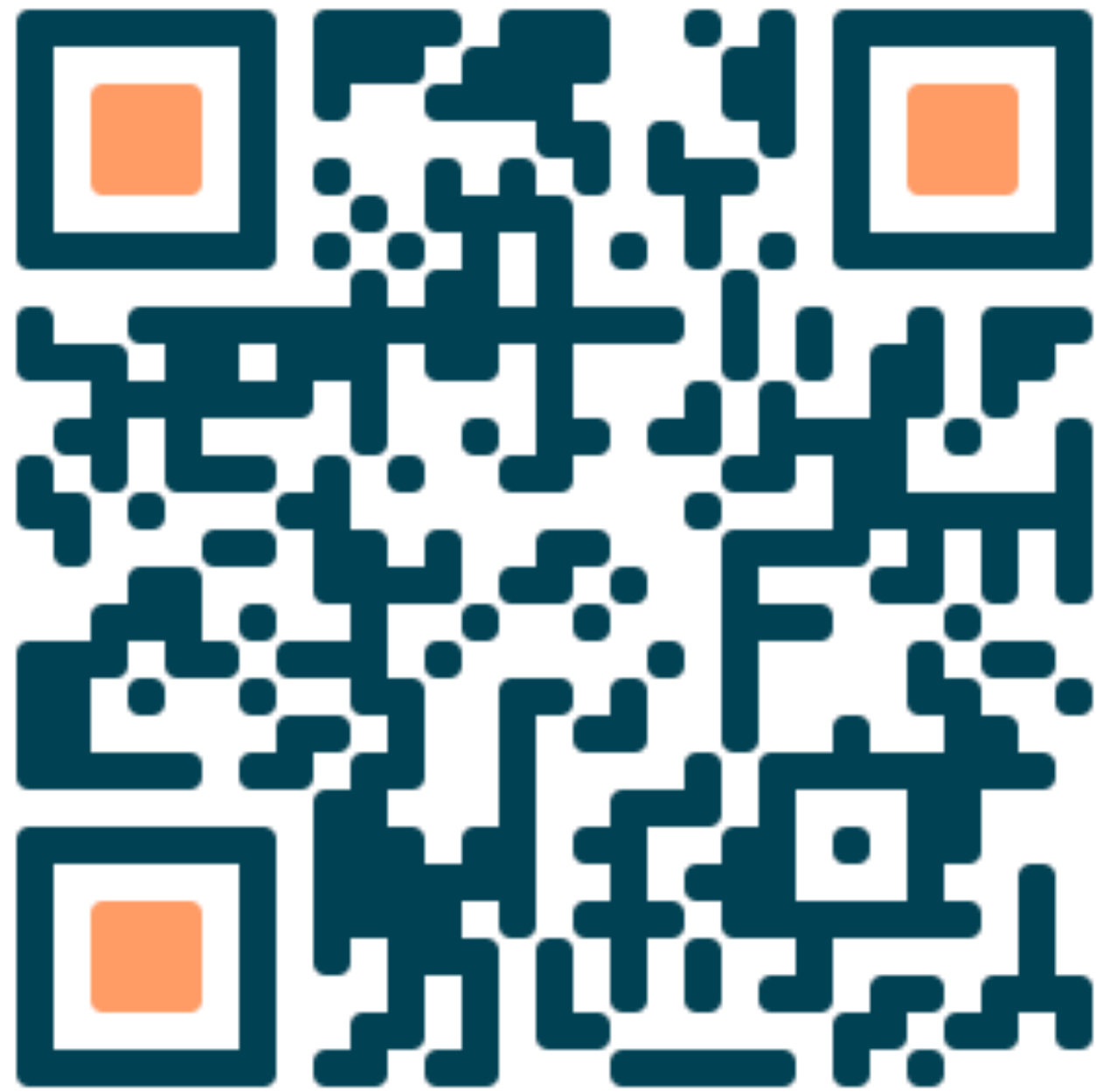






**„We start an agile transformation,  
how can we do modeling, design and  
requirements engineering in this scenario?“**

# Check out DDD-CREW on GitHub



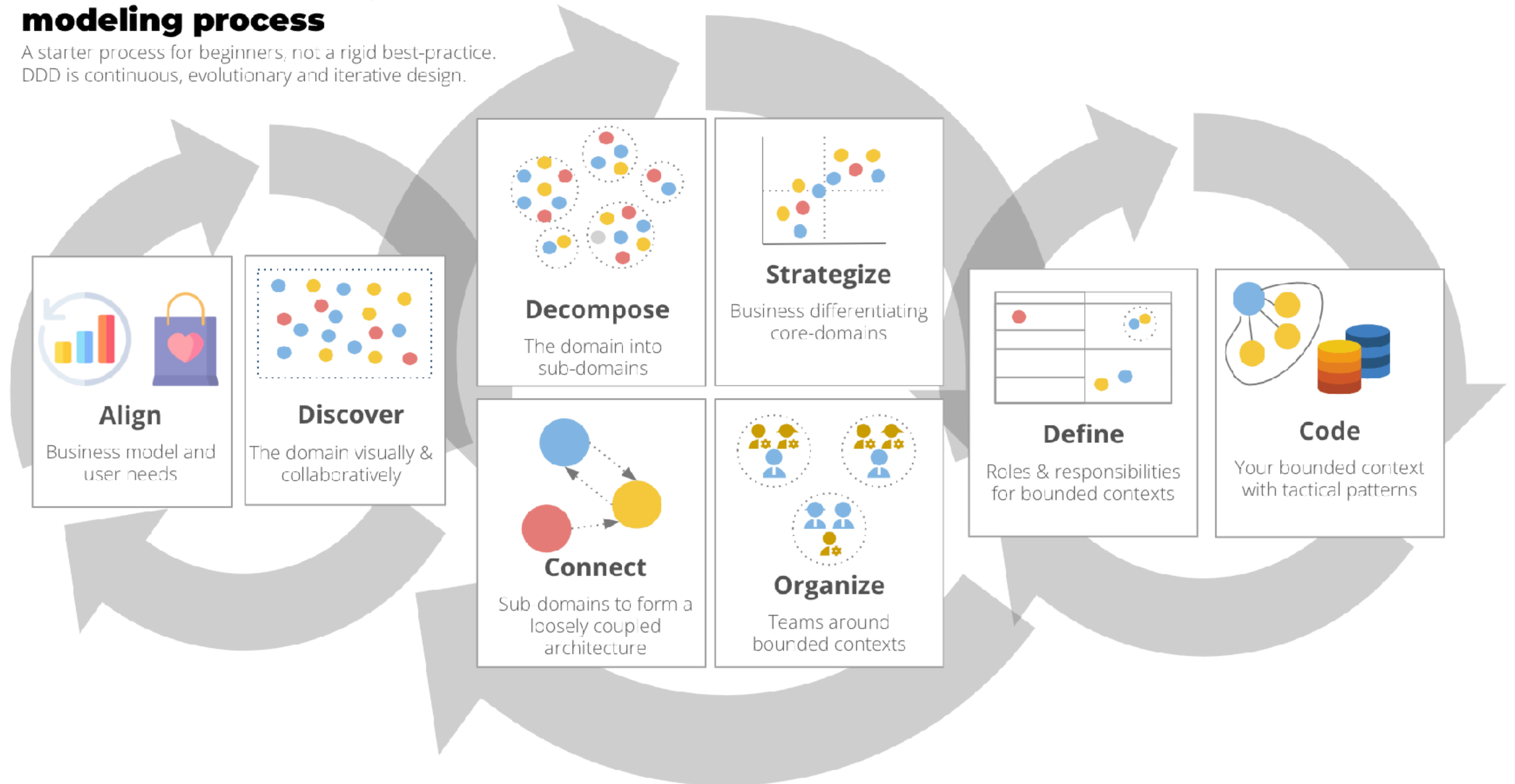
A screenshot of the GitHub profile page for 'Domain-Driven Design Crew'. The header shows the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The profile section includes the team's name, a 'Worldwide' location, and a 'Follow' button. Below this is a navigation bar with tabs for Overview, Repositories (16), Projects (1), Packages, Teams (1), and People (15). The 'Pinned' section features two repositories: 'welcome-to-ddd' (Public) with 759 stars and 43 forks, and 'ddd-starter-modelling-process' (Public) with 3.1k stars and 232 forks. The 'Repositories' section has a search bar and filters for Type, Language, and Sort, with a 'New' button. Two repositories are listed: 'eventstorming-glossary-cheat-sheet' (Public) with 403 stars, CC-BY-SA-4.0 license, 45 forks, 2 commits, 2 issues, and updated 11 days ago; and 'bounded-context-canvas' (Public) with 1,106 stars, CC-BY-SA-4.0 license, 120 forks, 1 commit, 0 issues, and updated 12 days ago. The right sidebar shows 'View as: Public', a note about viewing the README, a link to create a README file, a 'People' section with 15 member avatars, and a 'Top languages' section showing HTML.

<https://github.com/ddd-crew>



# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.



<https://github.com/ddd-crew/ddd-starter-modelling-process>

**Key Message for the penthouse:**

**Domain Driven Design is iterative  
and based on continuous  
improvement / learning**



**Let's check the agile manifesto against the cultural  
ideas of Domain Driven Design**



# Principles behind the Agile Manifesto

*We follow these principles:*

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Domain Driven Design is all about continuous learning through direct customer feedback

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Changing requirements may come from new insights and learnings. Something we appreciate in Domain Driven Design

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

This principle is not addressed directly by Domain Driven Design but most folks in the community agree with it

Business people and developers must work together daily throughout the project.

This is what Domain Driven Design is all about on a collaborative level

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Modern Domain Driven Design talks a lot about trust and safe environments. So: yes, there is a perfect fit



The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

This is 100% a core idea / principle in Domain Driven Design

Working software is the primary measure of progress.

Not directly addressed but appreciated

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Domain Driven Design does not address concepts like pace but most experts will agree that this principle is a good idea

Continuous attention to technical excellence and good design enhances agility.

There is a dedicated chapter in the blue book by Eric Evans: Supply Design

Simplicity--the art of maximizing the amount of work not done--is essential.

DDD does not talk about simplicity but about addressing / managing complexity

The best architectures, requirements, and designs emerge from self-organizing teams.

This is heavily addressed in modern sociotechnical Domain Driven Design

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Domain Driven Design does not mention retrospectives or team improvements but many experts fully agree with this.



**Key Message for the penthouse:**

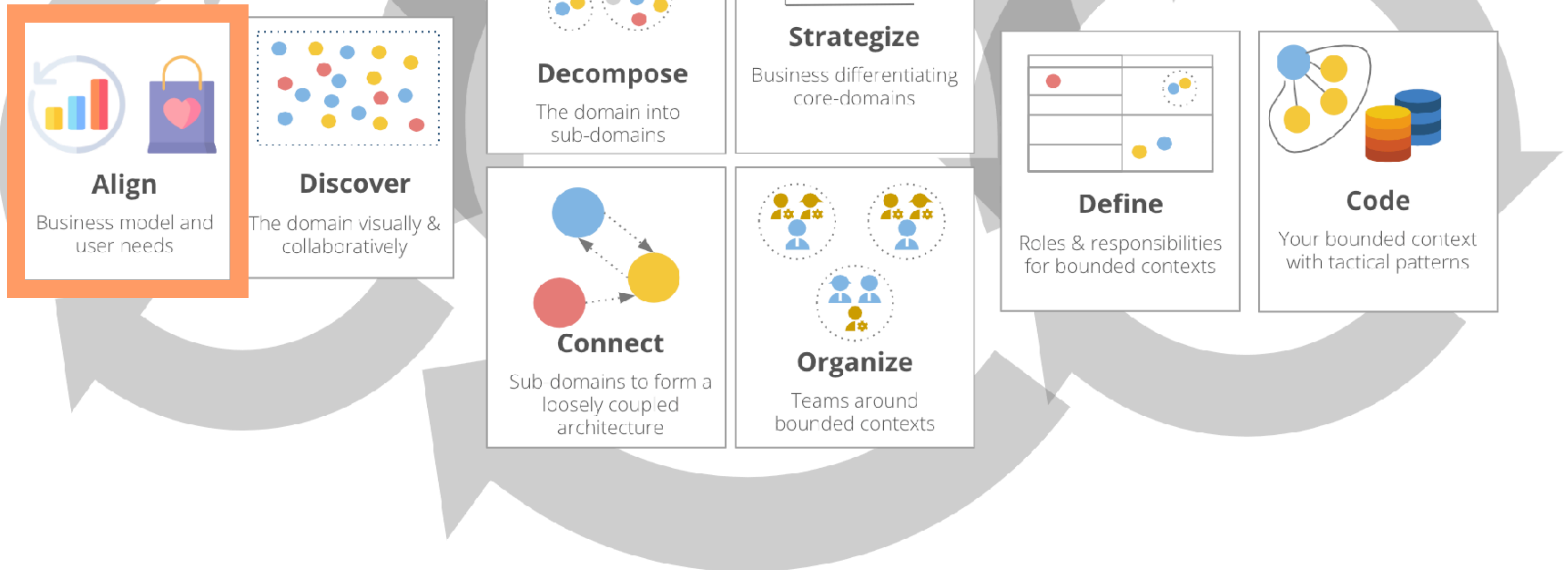
**Domain Driven Design thrives in an  
agile environment and suffers  
heavily in a waterfall**



**„We are business driven everyone knows  
this“**

# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.





**Do your architects  
understand the business  
model of their products?**

**Is there a shared  
understanding between  
various stakeholders  
regarding the business  
model?**

**Questions for the penthouse...**










# The Business Model Canvas

Designed for:

Designed by:

Date:

Version:

<div><h3>Key Partners</h3><p>Who are our Key Partners? Who are our key suppliers? Which Key Resources are we acquiring from partners? Which Key Activities do partners perform?</p><p><b>MOTIVATIONS FOR PARTNERSHIPS</b> Outstanding and economy Reduction of risk and uncertainty Acquisition of particular resources and activities</p></div>	<div><h3>Key Activities</h3><p>What Key Activities do our Value Propositions require? Our Distribution Channels? Customer Relationships? Revenue Streams?</p><p><b>CATEGORIES</b> Production Problem Solving Customization</p></div>	<div><h3>Value Propositions</h3><p>What value do we deliver to the customer? Which one of our customer's problems are we helping to solve? What bundles of products and services are we offering to each Customer Segment? Which customer needs are we satisfying?</p><p><b>CHARACTERISTICS</b> Newness Performance Customization "Get into the Job Done" Design Brand/Status Price Cost Reduction Risk Reduction Accessibility Convenience/Usability</p></div>	<div><h3>Customer Relationships</h3><p>What type of relationship does each of our Customer Segments expect us to establish and maintain with them? Which ones have we established? How are they integrated with the rest of our business model? How costly are they?</p><p><b>EXAMPLES</b> Personal assistance Dedicated Personal Assistance Self-Service Automated Services Communities Co-creation</p></div>	<div><h3>Customer Segments</h3><p>For whom are we creating value? Who are our most important customers?</p><p><b>EXAMPLES</b> Mass Market Niche Market Segmented Diversified Multi-sided Platform</p></div>																								
	<div><h3>Key Resources</h3><p>What Key Resources do our Value Propositions require? Our Distribution Channels? Customer Relationships? Revenue Streams?</p><p><b>TYPES OF RESOURCES</b> Physical Intellectual (Patent, patents, copyrights, etc.) Human Financial</p></div>		<div><h3>Channels</h3><p>Through which Channels do our Customer Segments want to be reached? How are we reaching them now? How are our Channels integrated? Which ones work best? Which ones are most cost-efficient? How are we integrating them with customer routines?</p><p><b>CHANNEL PHASES</b> 1. Awareness How do we make awareness about our company's products and services? 2. Evaluation How do we help customers evaluate our organization's Value Proposition? 3. Purchase How do we allow customers to purchase specific products and services? 4. Delivery How do we deliver a Value Proposition to customers? 5. After sales How do we provide post-purchase customer support?</p></div>																									
<div><h3>Cost Structure</h3><p>What are the most important costs inherent in our business model? Which Key Resources are most expensive? Which Key Activities are most expensive?</p><p><b>IS YOUR BUSINESS MODEL</b> Cost Driven (leanest cost structure, low price value proposition, maximum automation, extensive outsourcing) Value Driven (throughest value creation, premium value proposition)</p><p><b>EXAMPLE CHARACTERISTICS</b> Fixed Costs (salaries, rents, utilities) Variable costs Economies of scale Economies of scope</p></div>		<div><h3>Revenue Streams</h3><p>For what value are our customers really willing to pay? For what do they currently pay? How are they currently paying? How would they prefer to pay? How much does each Revenue Stream contribute to overall revenues?</p><table><tr><td><b>TYPE</b></td><td><b>FIXED PRICING</b></td><td><b>DYNAMIC PRICING</b></td></tr><tr><td>Asset sale</td><td>List Price</td><td>Negotiation (bargaining)</td></tr><tr><td>Usage fee</td><td>Product/feature dependent</td><td>Yield Management</td></tr><tr><td>Subscription fee</td><td>Customer segment dependent</td><td>Real-time Market</td></tr><tr><td>Lending/Renting/Leasing</td><td></td><td></td></tr><tr><td>Licensing</td><td>Volume dependent</td><td></td></tr><tr><td>Franchise fees</td><td></td><td></td></tr><tr><td>Advertising</td><td></td><td></td></tr></table></div>			<b>TYPE</b>	<b>FIXED PRICING</b>	<b>DYNAMIC PRICING</b>	Asset sale	List Price	Negotiation (bargaining)	Usage fee	Product/feature dependent	Yield Management	Subscription fee	Customer segment dependent	Real-time Market	Lending/Renting/Leasing			Licensing	Volume dependent		Franchise fees			Advertising		
<b>TYPE</b>	<b>FIXED PRICING</b>	<b>DYNAMIC PRICING</b>																										
Asset sale	List Price	Negotiation (bargaining)																										
Usage fee	Product/feature dependent	Yield Management																										
Subscription fee	Customer segment dependent	Real-time Market																										
Lending/Renting/Leasing																												
Licensing	Volume dependent																											
Franchise fees																												
Advertising																												



DESIGNED BY: Business Model Foundry AG  
The makers of Business Model Generation and Strategyzer

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit: <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

**What is the**  
**purpose of digitalization**  
**from a business perspective**



# Purposes of digitalization

Digitalization

Improve an existing business model with tech

Create a whole new business (model) enabled by tech

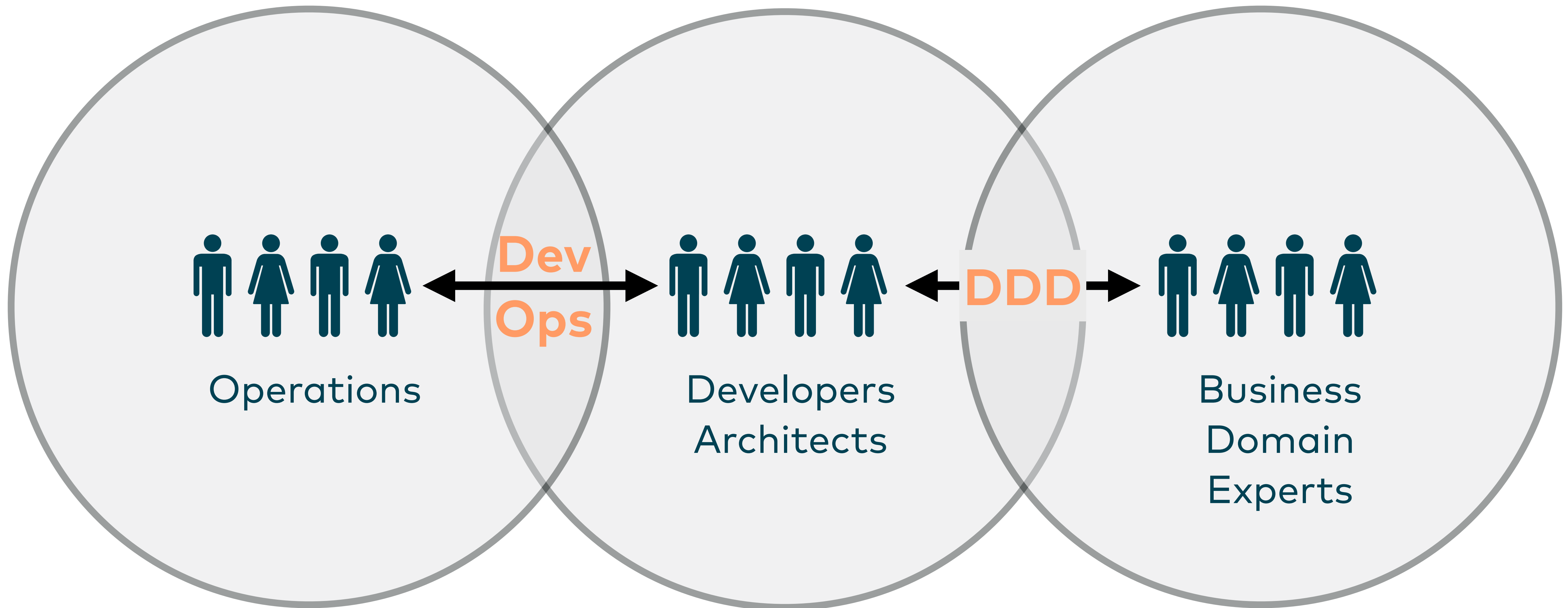
**Did you realize?**

**The words business & tech appeared together in both  
purposes**





**„Let's introduce DevOps!“**



You design it, you build it and you run it

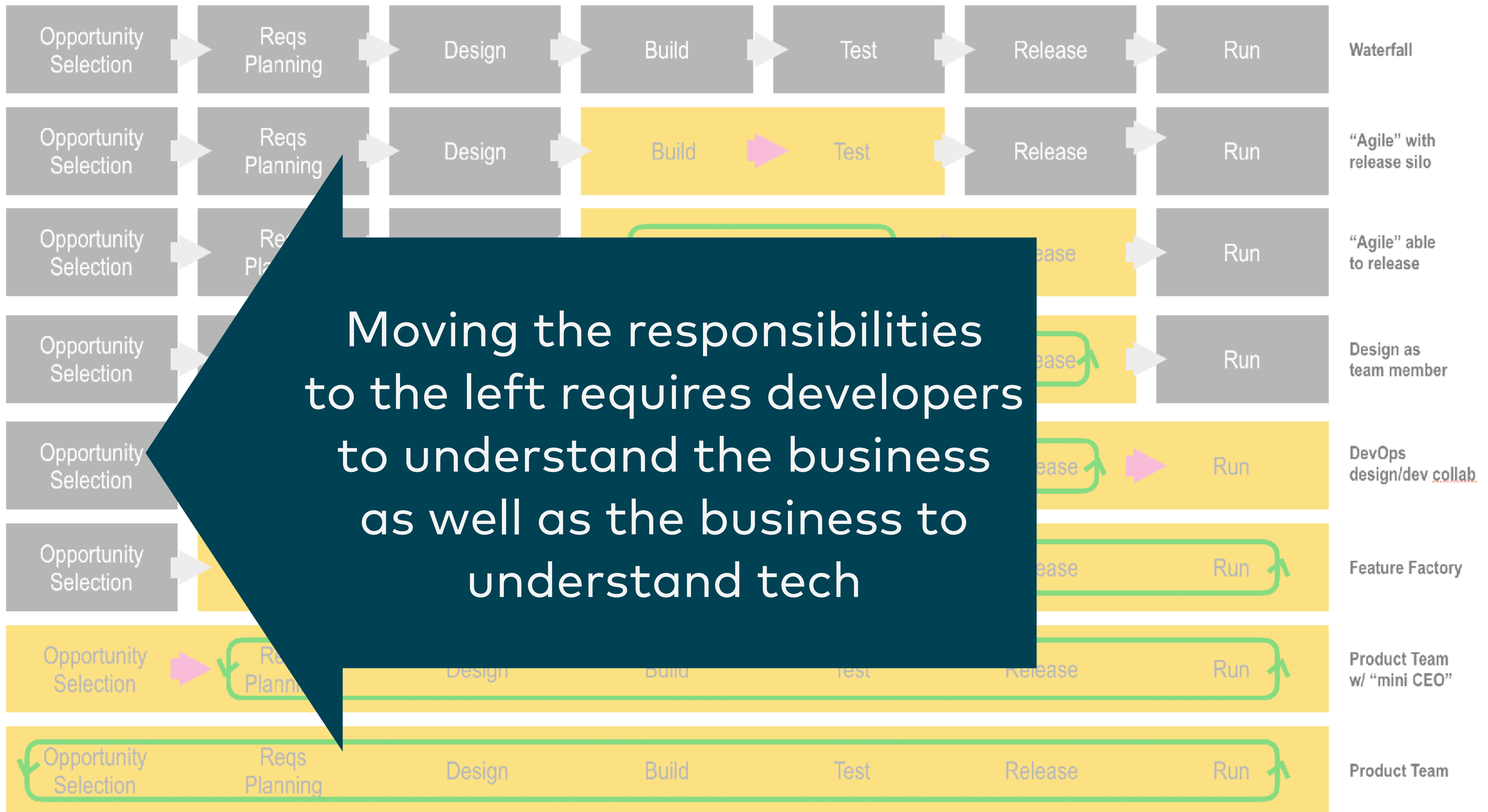
**Message to the Penthouse:**

**Cross-functional teams**

**Are about you design it, you build it and you run it**

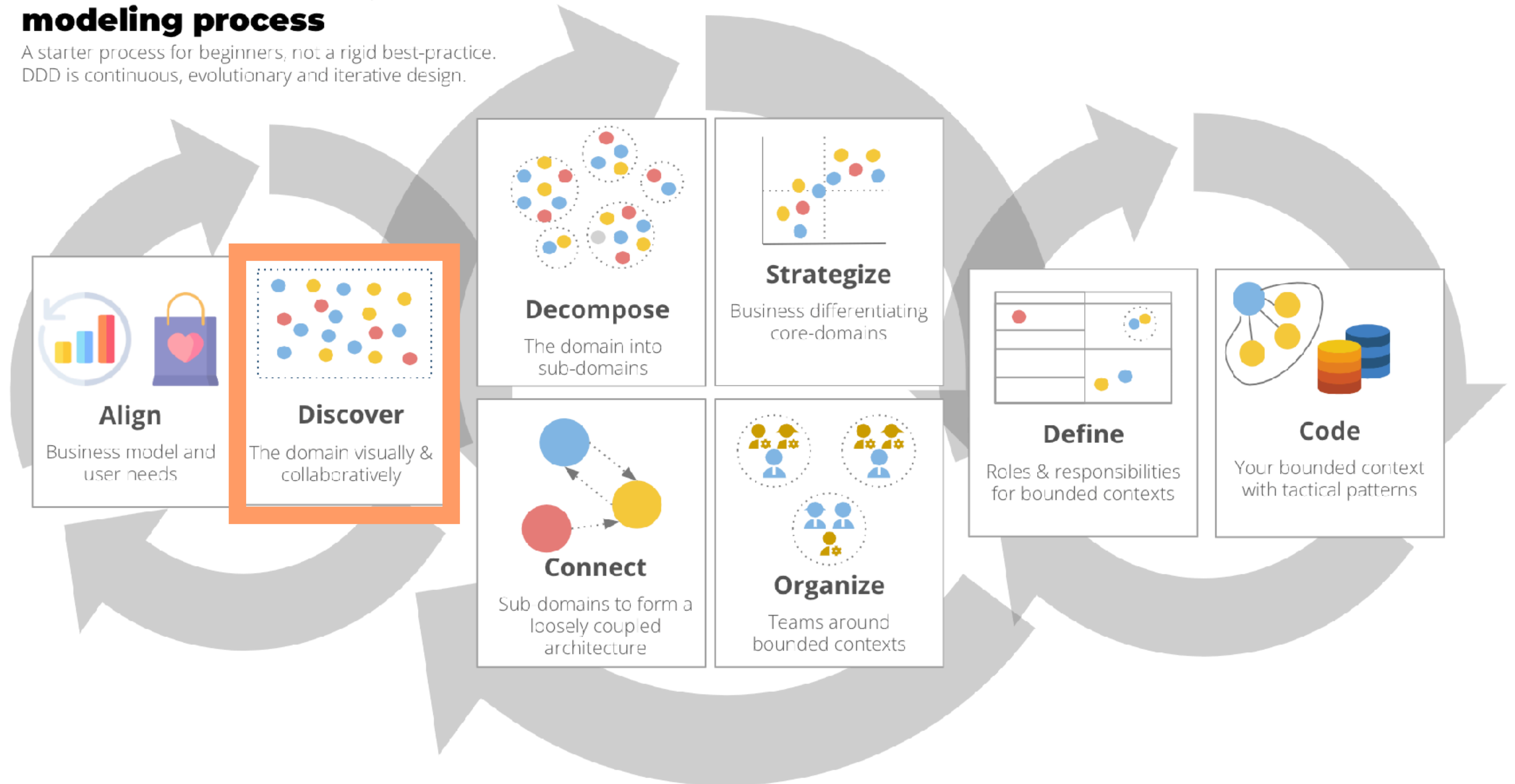






# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.





**„It is not the domain experts knowledge that goes into production, it is the assumption of the developers that goes into production“**

## **Alberto Brandolini**

Inventor of EventStorming





**WHAT COULD**

**POSSIBLY GO WRONG?**





How the business names things

What we see in code

Window

TransparencyFactory

Painting

DecoratorImpl

TV

EntertainmentProviderSingleton

Desk

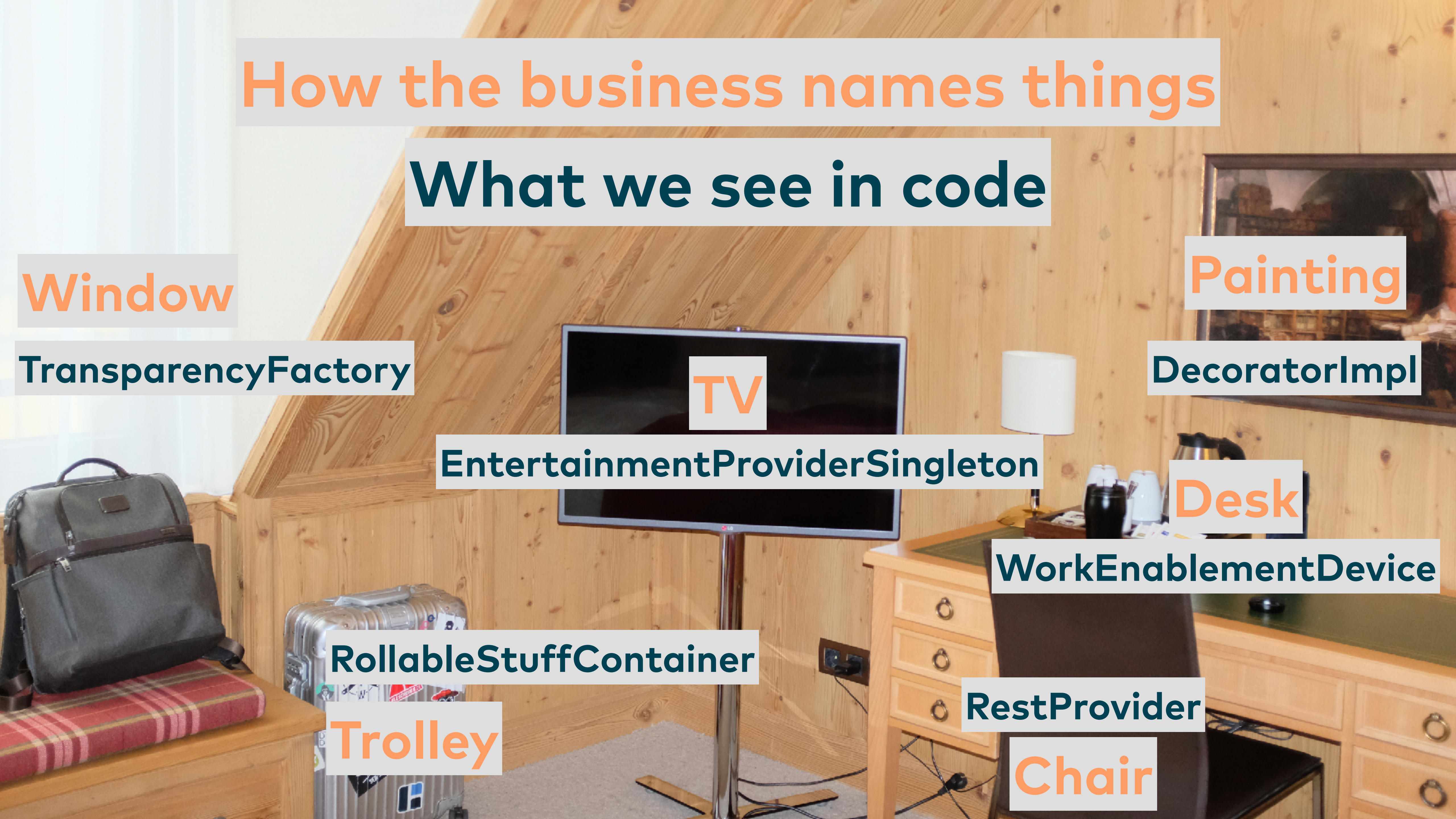
WorkEnablementDevice

RollableStuffContainer

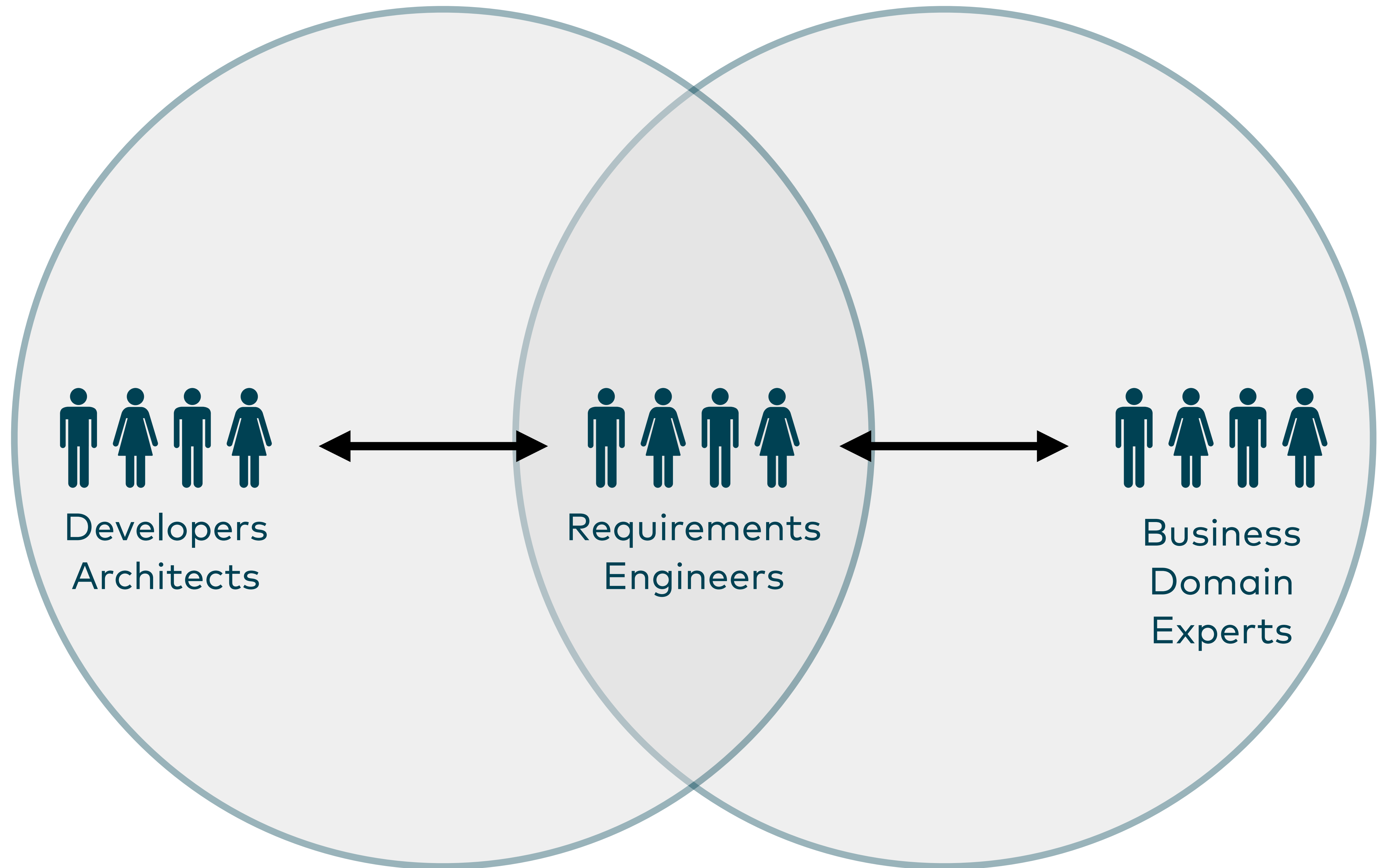
Trolley

RestProvider

Chair



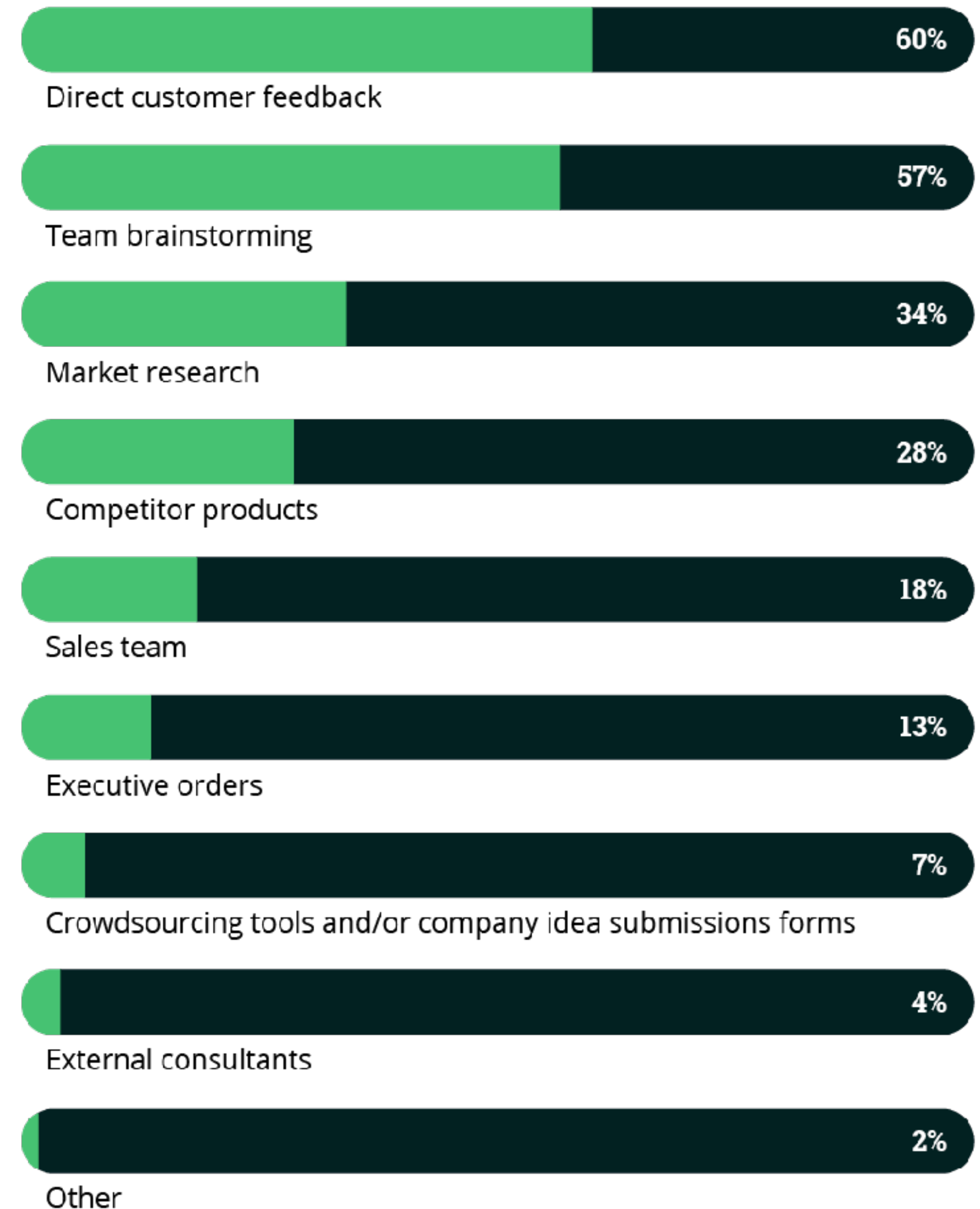




*"If you're just using your engineers to code, you're only getting about half their value."*

Marty Cagan

## Source of Best Product and Feature Ideas

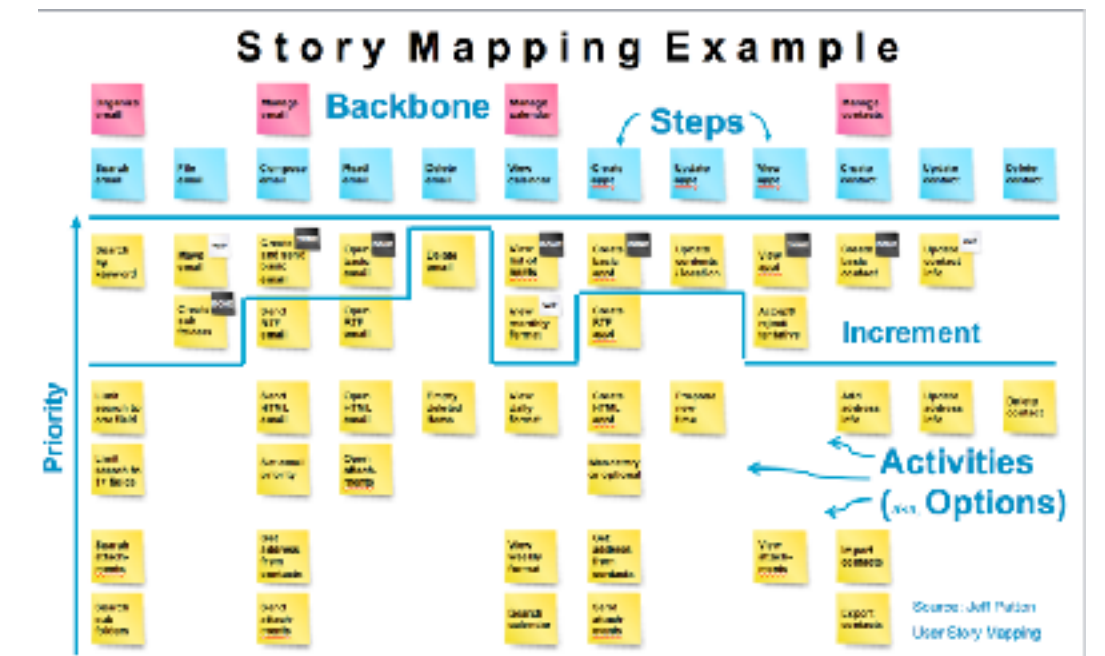


**What should you propose to  
the penthouse?**





# DOMAIN Storytelling



# Collaborative Modeling

# EVENT STORMING

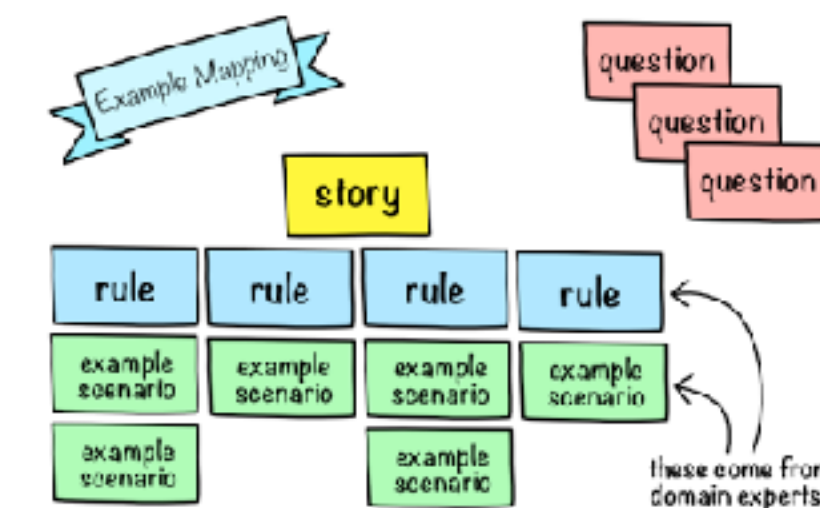


Image for example mapping taken from: <https://openpracticelibrary.com/practice/example-mapping/>

Image for user story mapping taken from: [https://www.hanssamios.com/dokuwiki/how\\_do\\_we\\_build\\_and\\_maintain\\_context\\_when\\_all\\_we\\_have\\_is\\_a\\_backlog\\_list](https://www.hanssamios.com/dokuwiki/how_do_we_build_and_maintain_context_when_all_we_have_is_a_backlog_list)

# **Questions for the penthouse**

**Are there already Design Thinking initiatives in the business?**

**Is your organization prepared for direct collaboration?**



**„We want to become a tech-enabled  
product organization“**



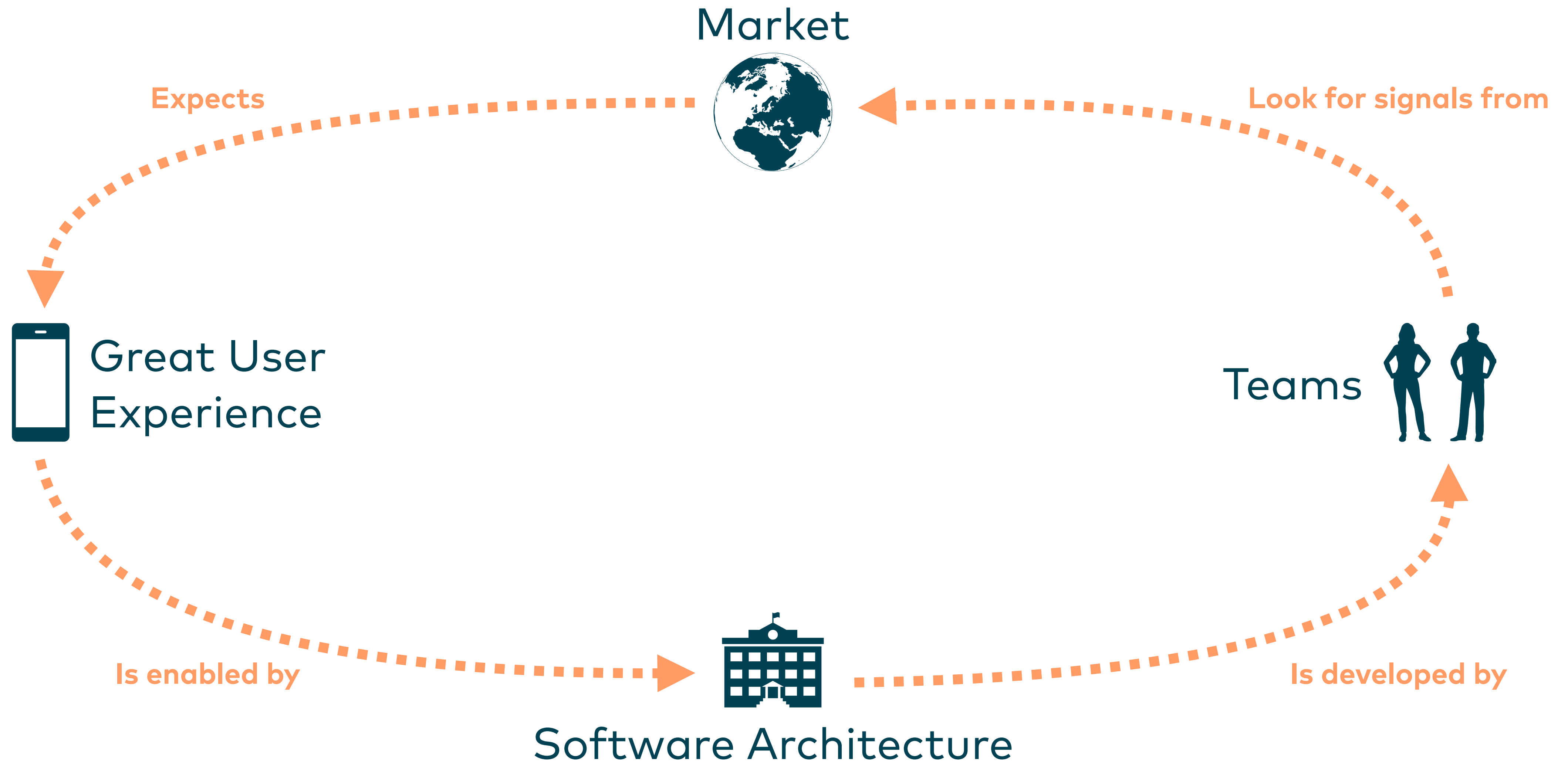


**Shift  
from  
Projects  
to  
Products**

# **Tech-enabled Organizations**

- Tech is core strategic asset in product and not a cost center
- No feature factories
- Strong aim to insourcing of development of functionality which is core to the business

# Innovation Cycles



**„the key to incremental architecture is to build on a framework that can accommodate change... that framework is the domain.... By modeling the domain, you can more easily handle changes to the domain“**

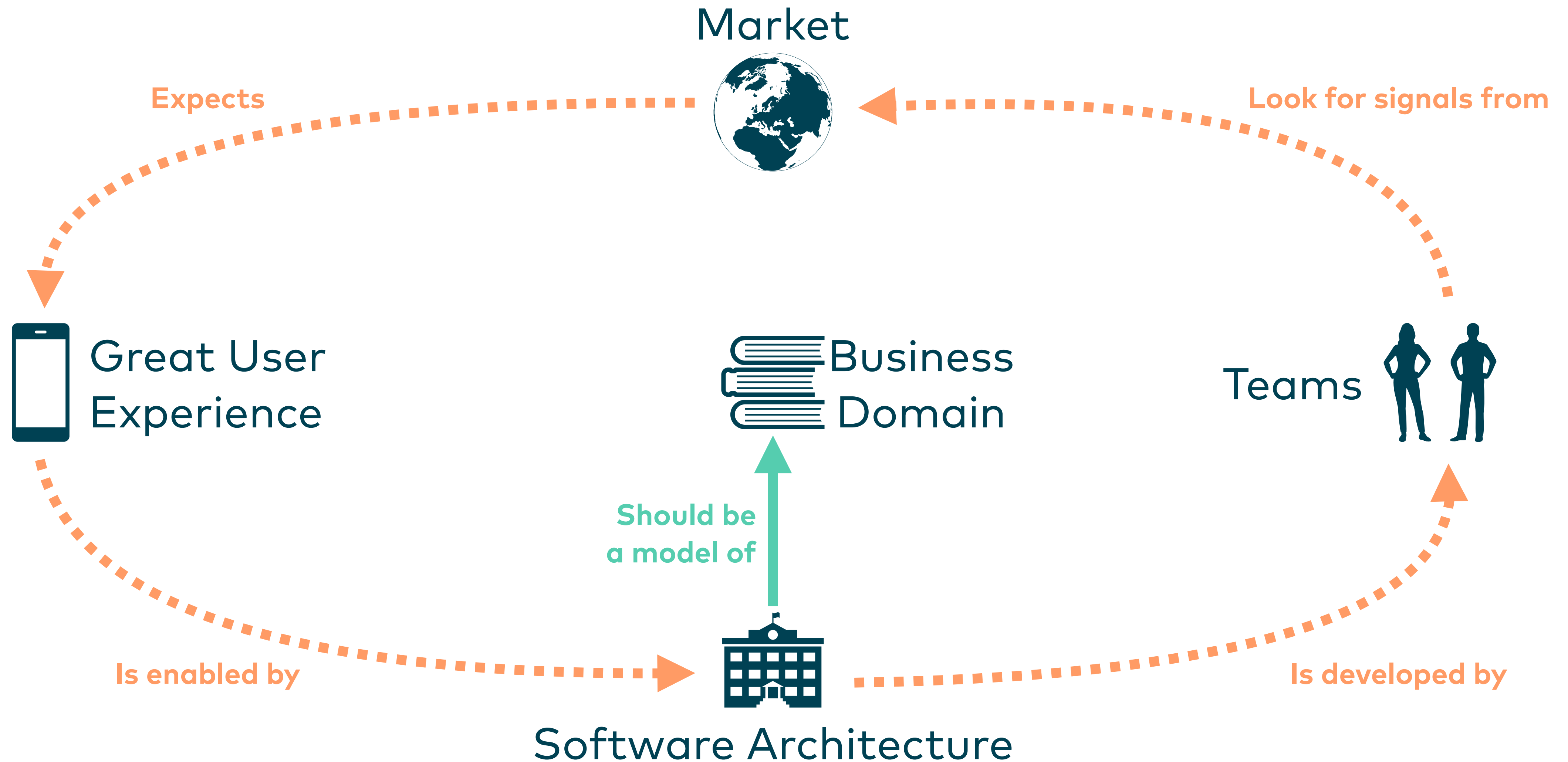
**Allen Holub**

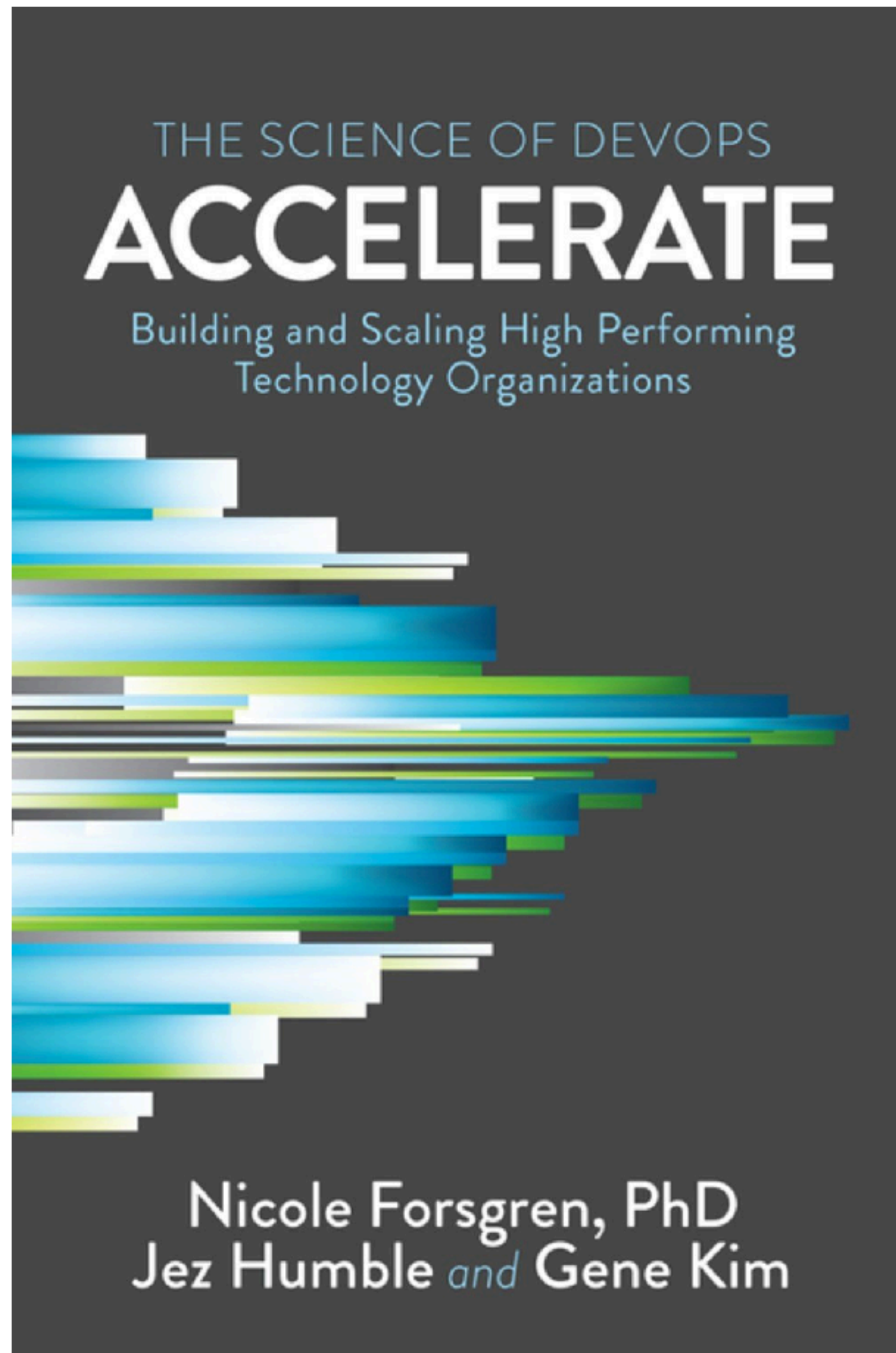
**<https://holub.com>**





# Innovation Cycles

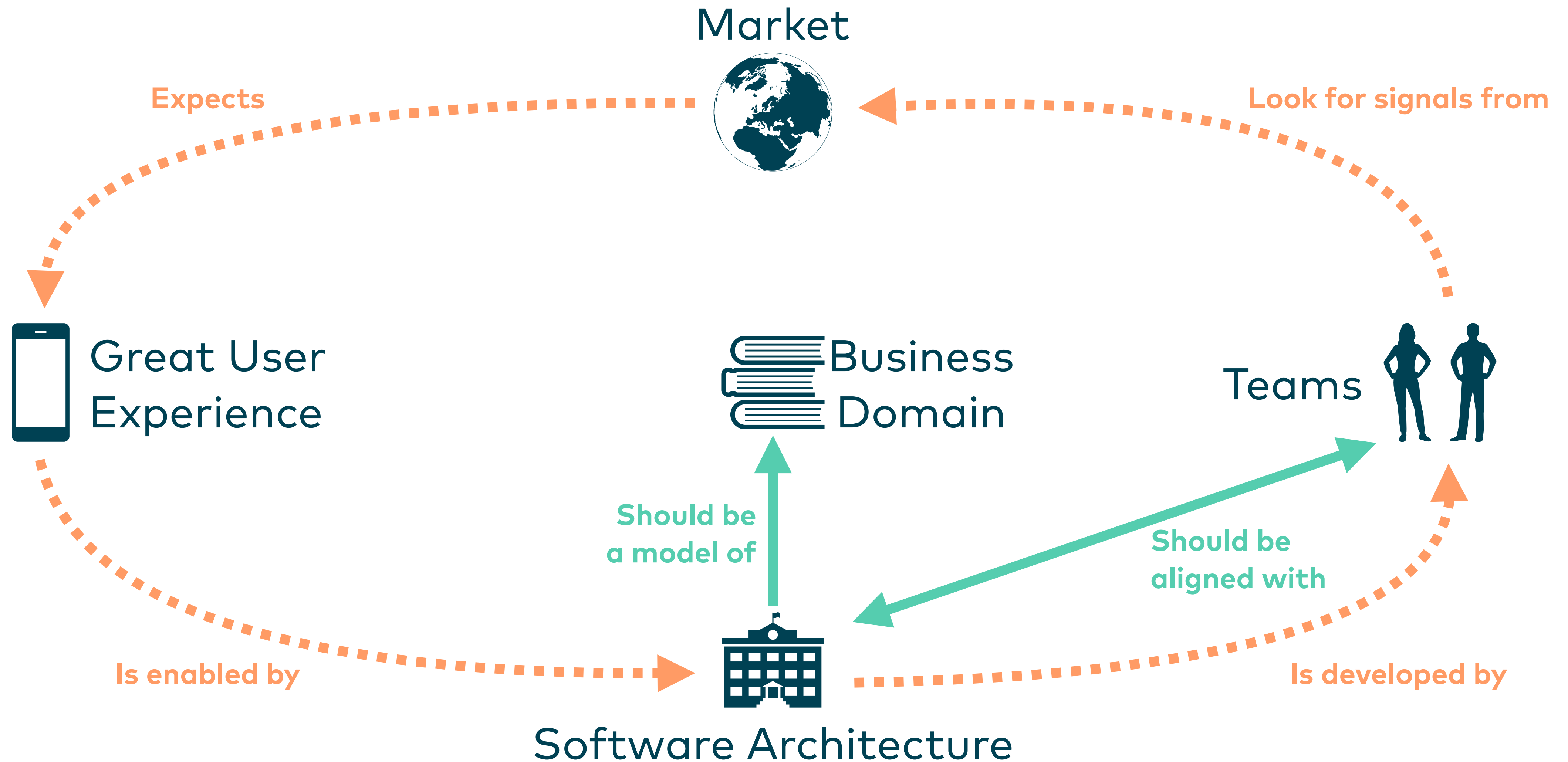




**"A loosely coupled software architecture and org structure to match" is a key predictor of:**

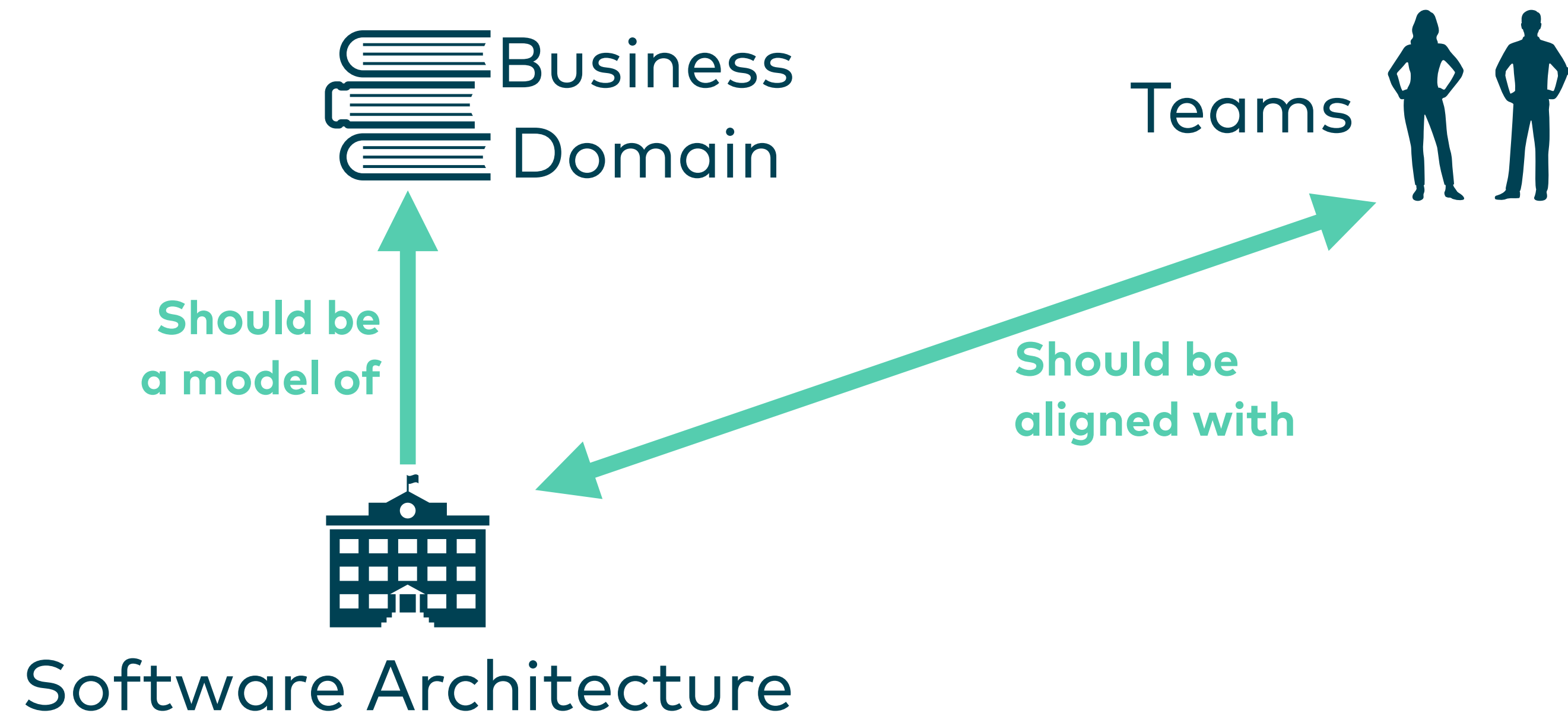
- Continuous Delivery Performance
- Ability to scale organization and increase performance linearly

# Innovation Cycles





# Strategic DDD helps with the alignment of the business domain with software architecture and teams



**„Product thinking is the journey from the  
problem space of the users to the  
solution space of the business.**

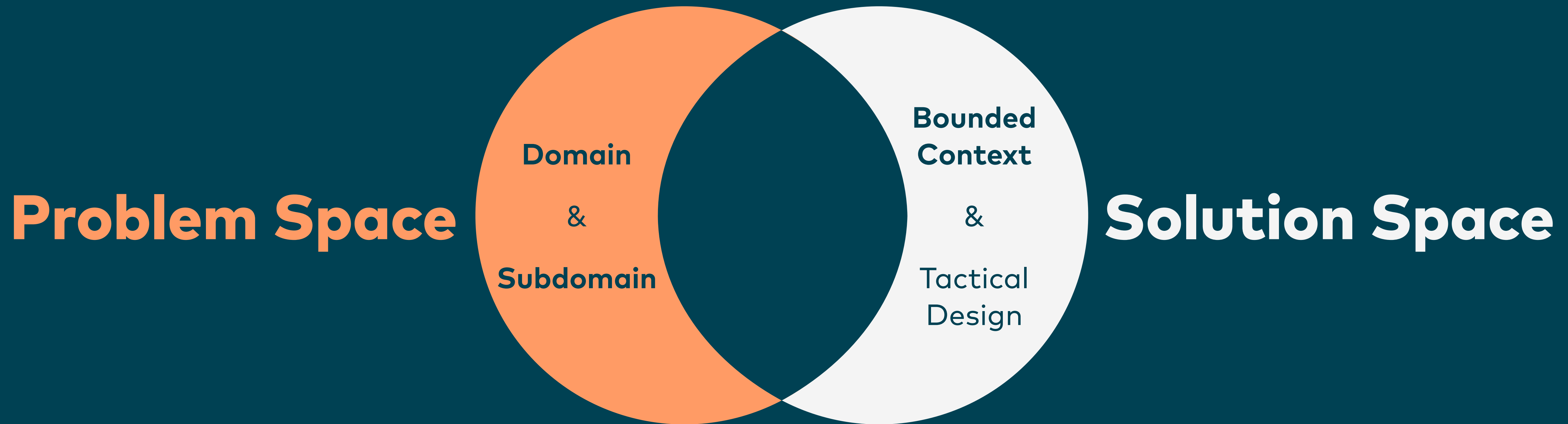
**The goal of this journey is to reduce the  
gap between users and the business."**

Naren Katakam



# Problem vs Solution Space

Strategic Design starts with the **problem space**, which represents the **business architecture** and which includes problem domains and (categorized) subdomains. The **solution space** represents the **software architecture** and contains the bounded context. There must be an overlap between the two.





**„Domains live in the problem space. They are how an organization perceives its areas of activity and expertise.“**



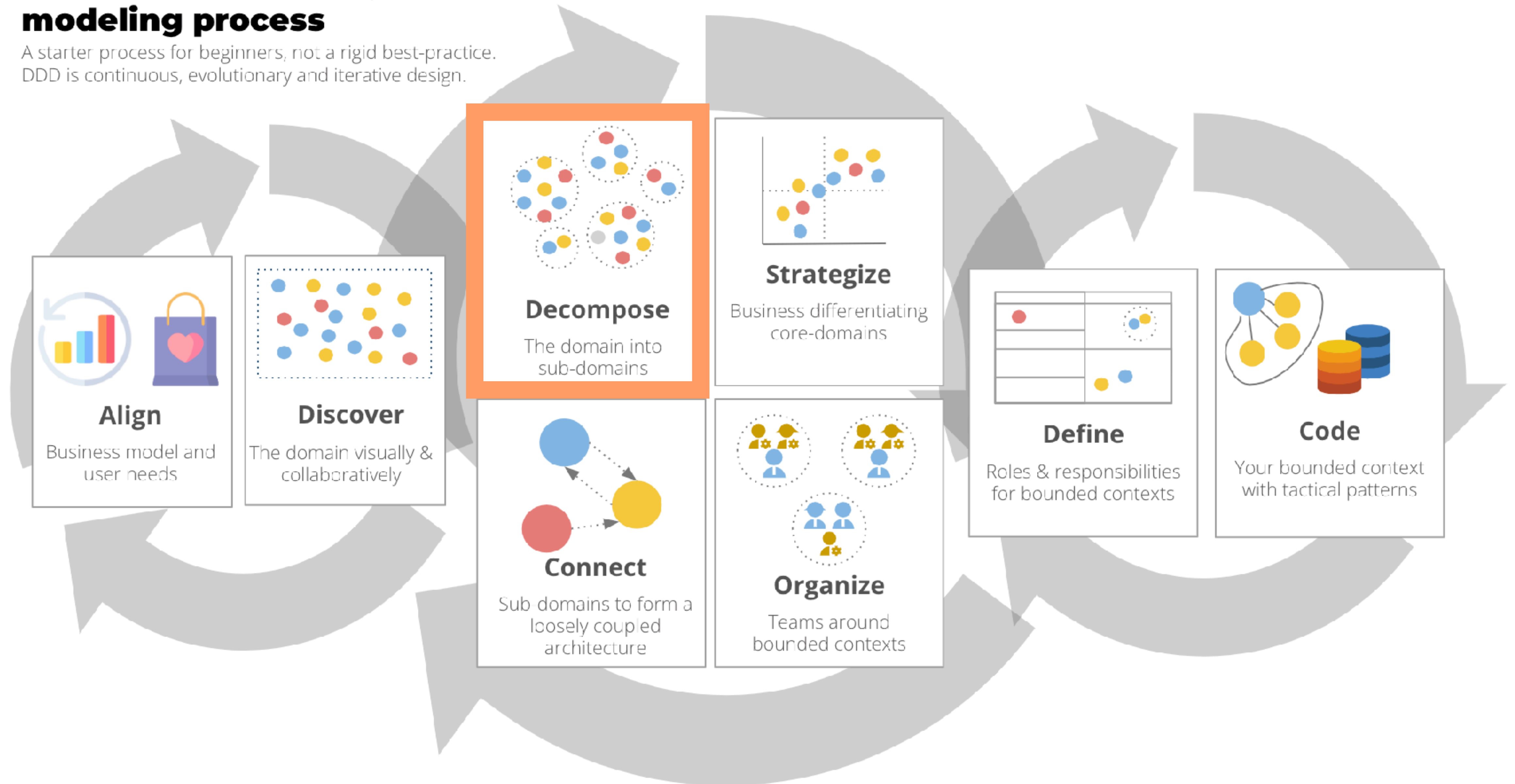
## **Mathias Verraes and Rebecca Wirfs-Brock**

Quote from „Splitting a Domain Across Multiple Bounded Contexts“  
<https://verraes.net/2021/06/split-domain-across-bounded-contexts/>



# Domain-Driven Design starter modeling process

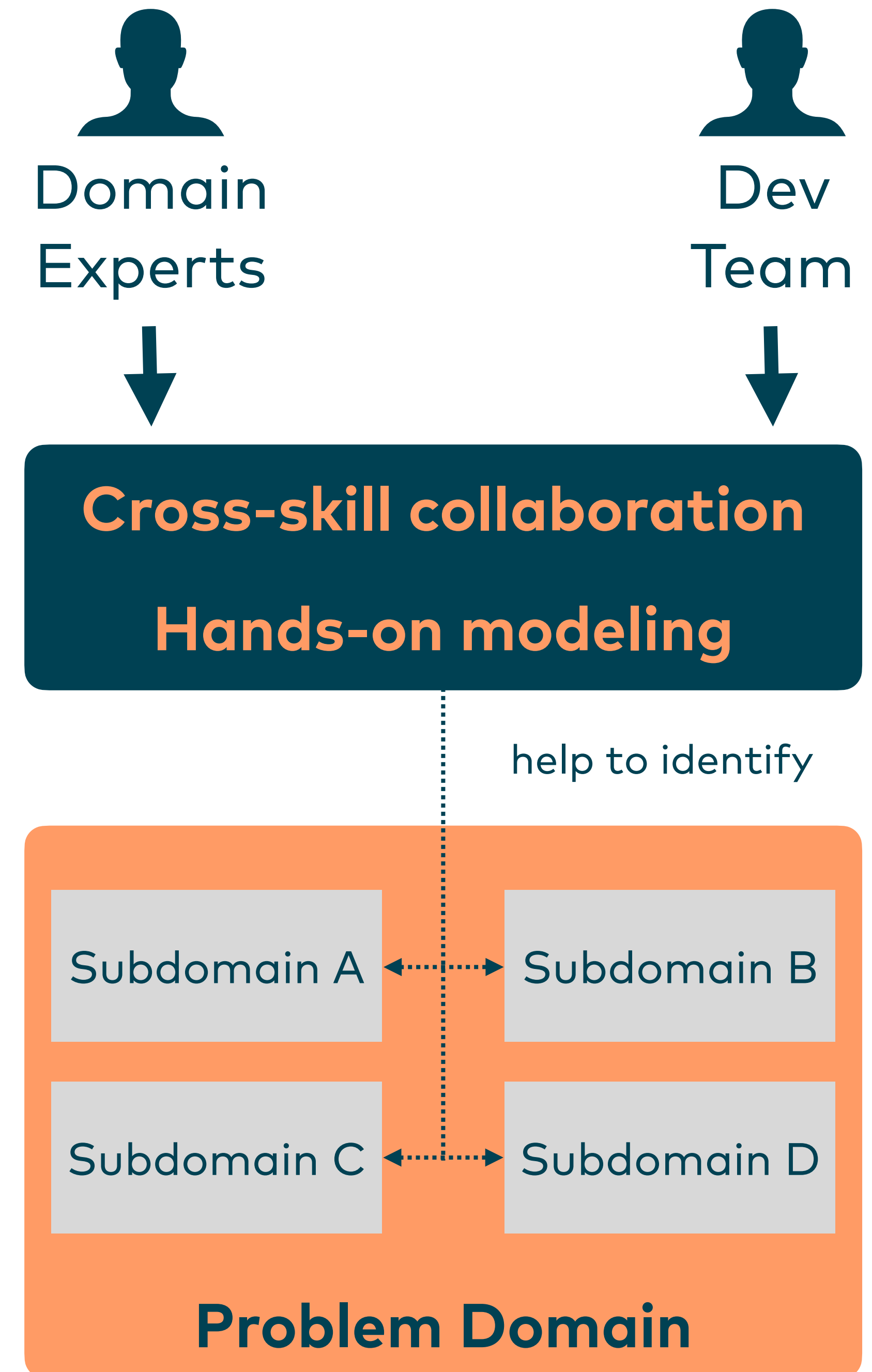
A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.





# Subdomains

- Each problem domain can contain some subdomains
- These subdomains are lower level business capabilities than the ones of a domain
- The identification of subdomains is usually performed by collaborative modeling within an integrated teams

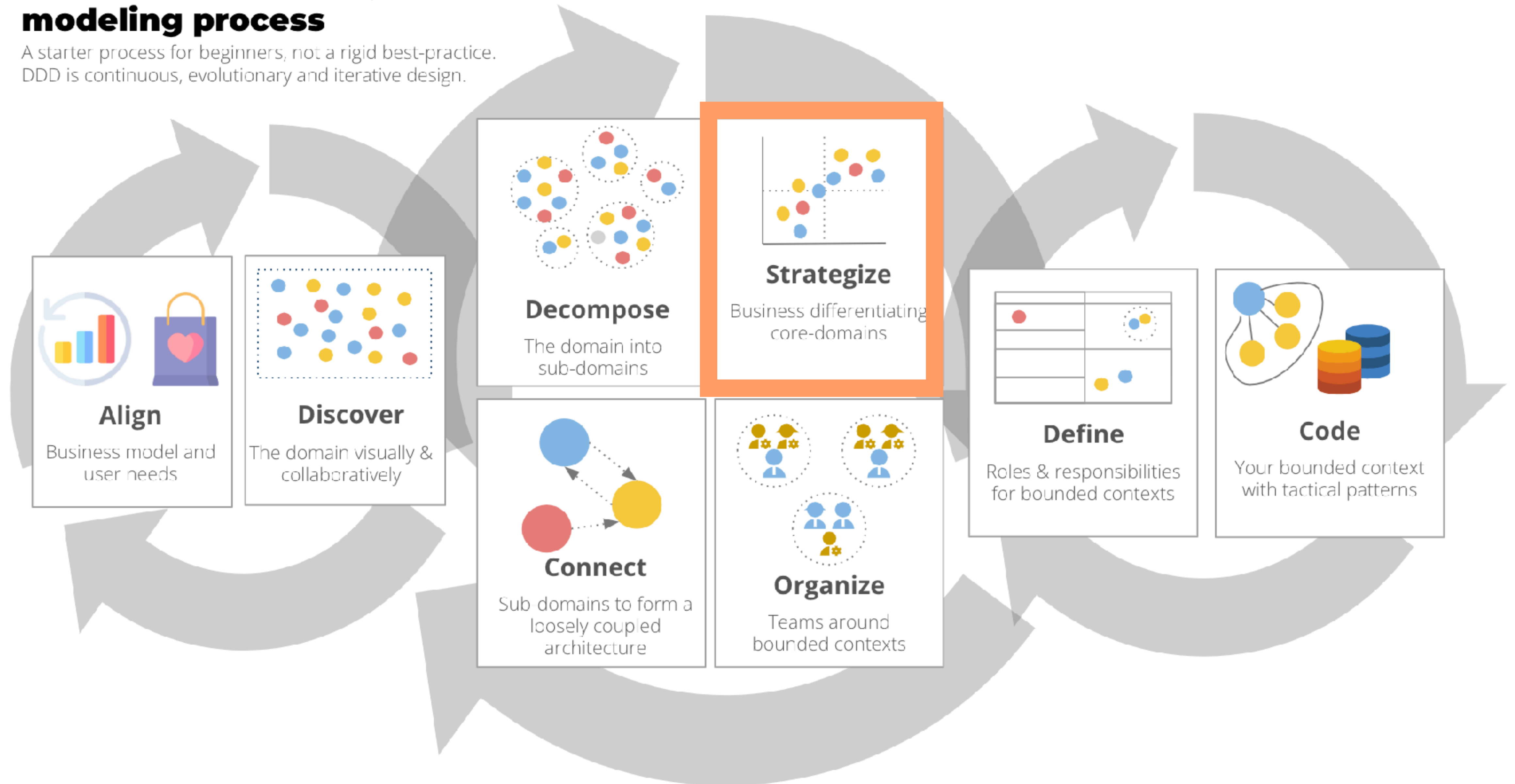




***„Let's go all in on SaaS and the cloud“***

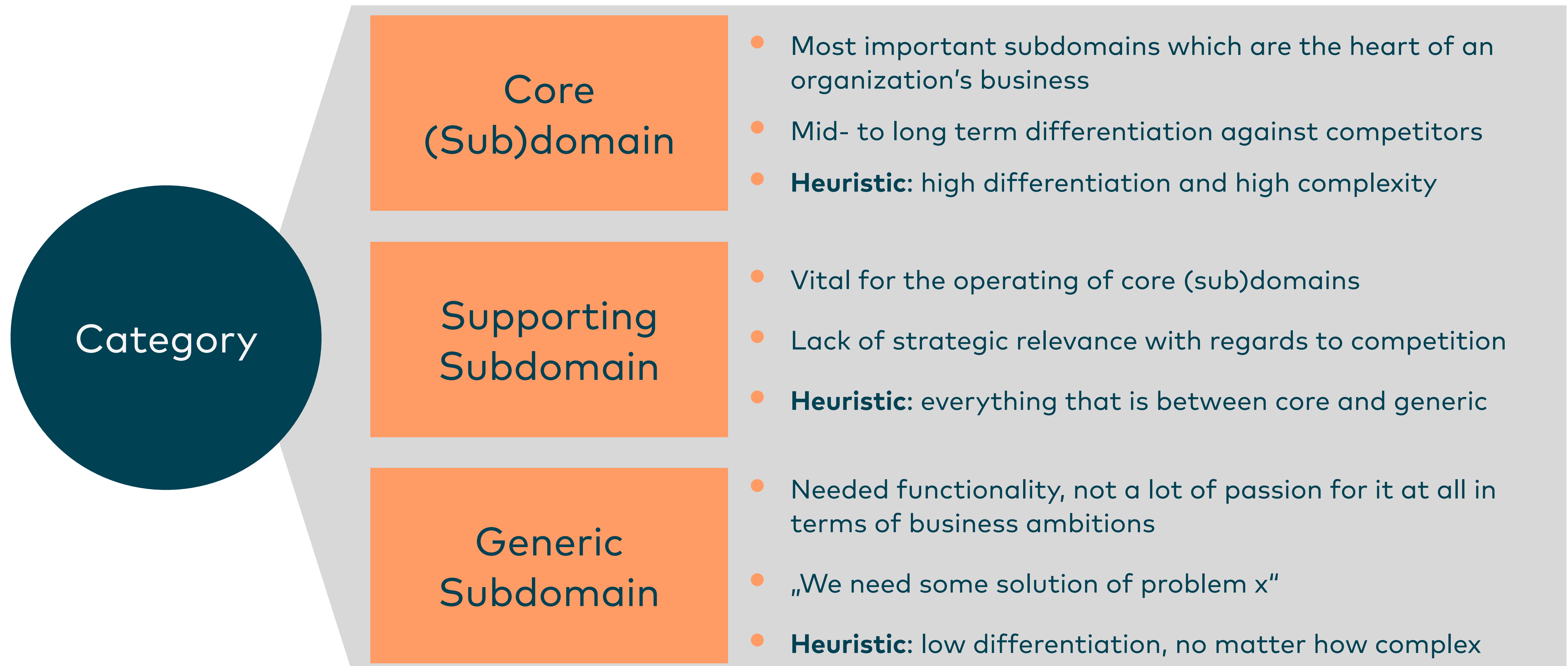
# Domain-Driven Design starter modeling process

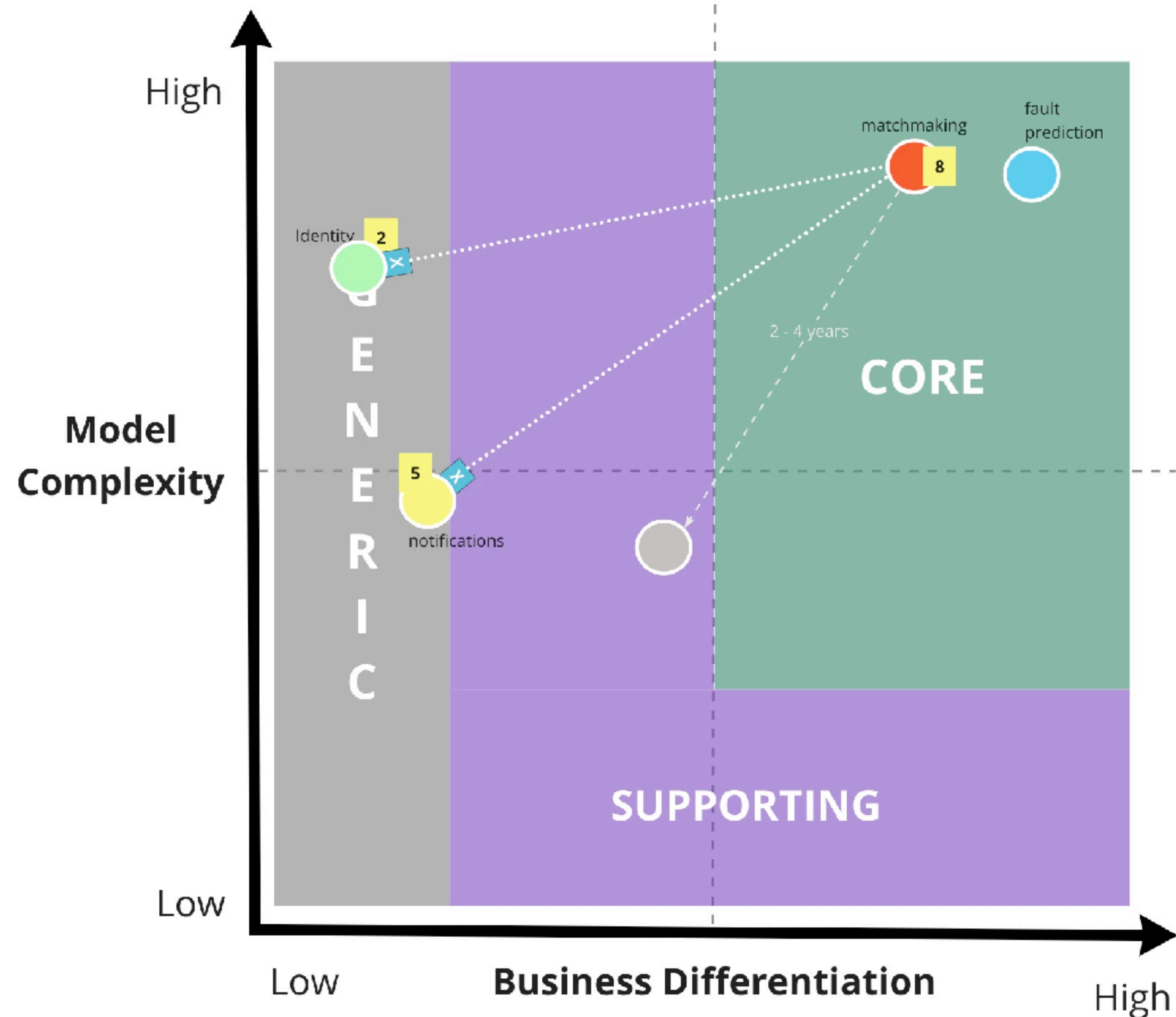
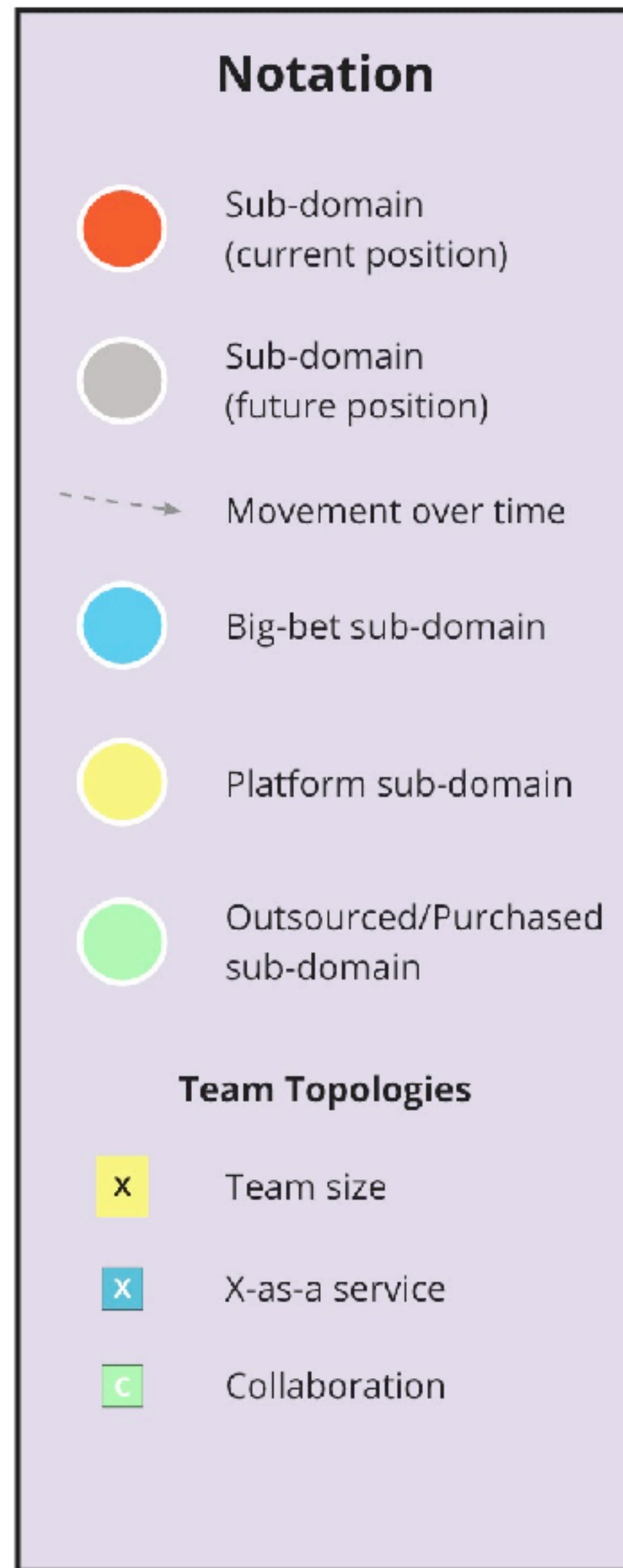
A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.





# Categories for Subdomains in DDD





<https://github.com/ddc-crew/core-domain-charts>



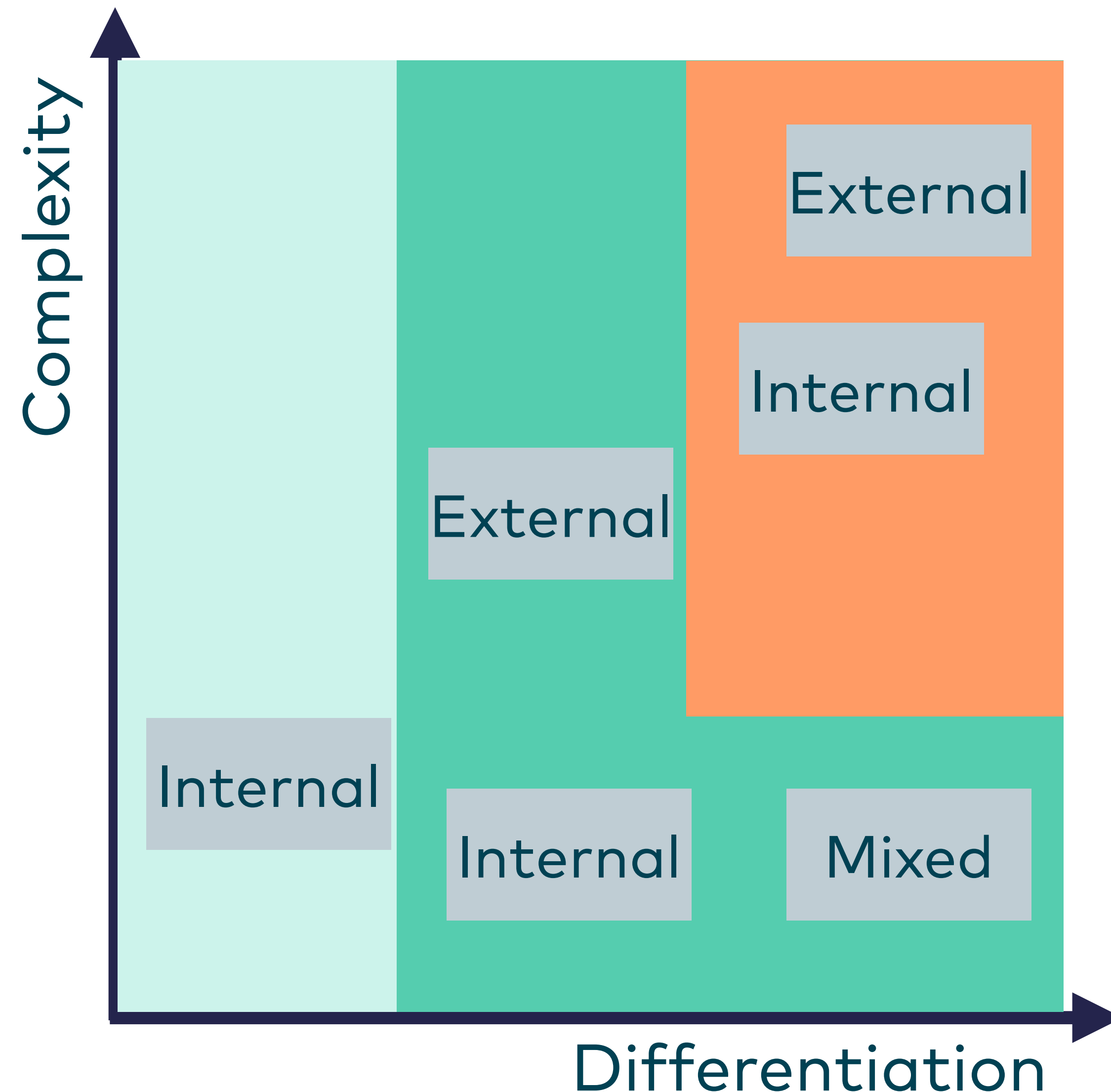
**„We are a bank, not a software shop“**

I really heard that quote from a C-Level person ... I'm not kidding



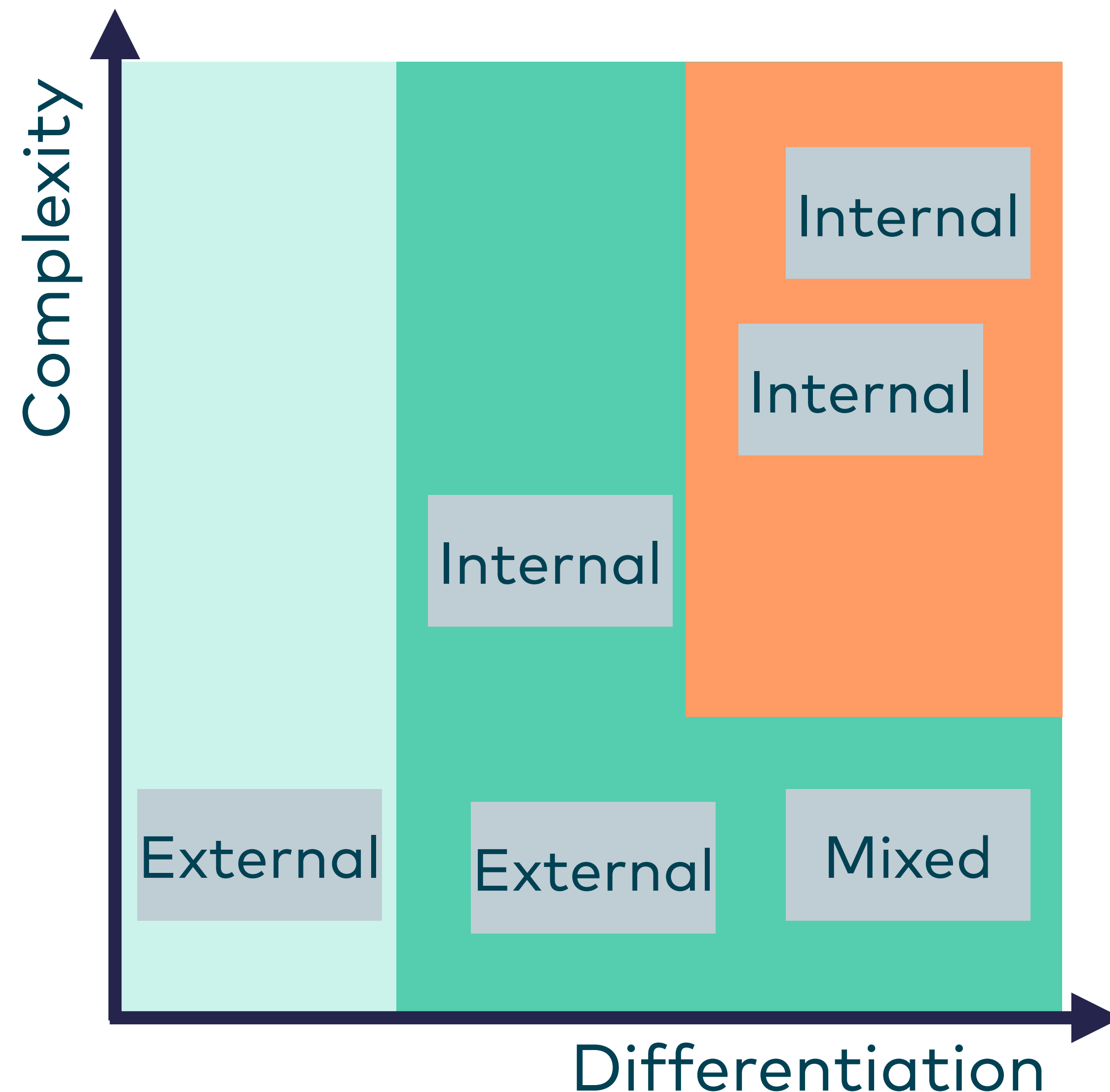
**Some explicit questions for the penthouse**

# How are your teams staffed?



- This is a highly critical situation
- You want to own everything in your core domains
- External teams should be the exception in those areas
- All T

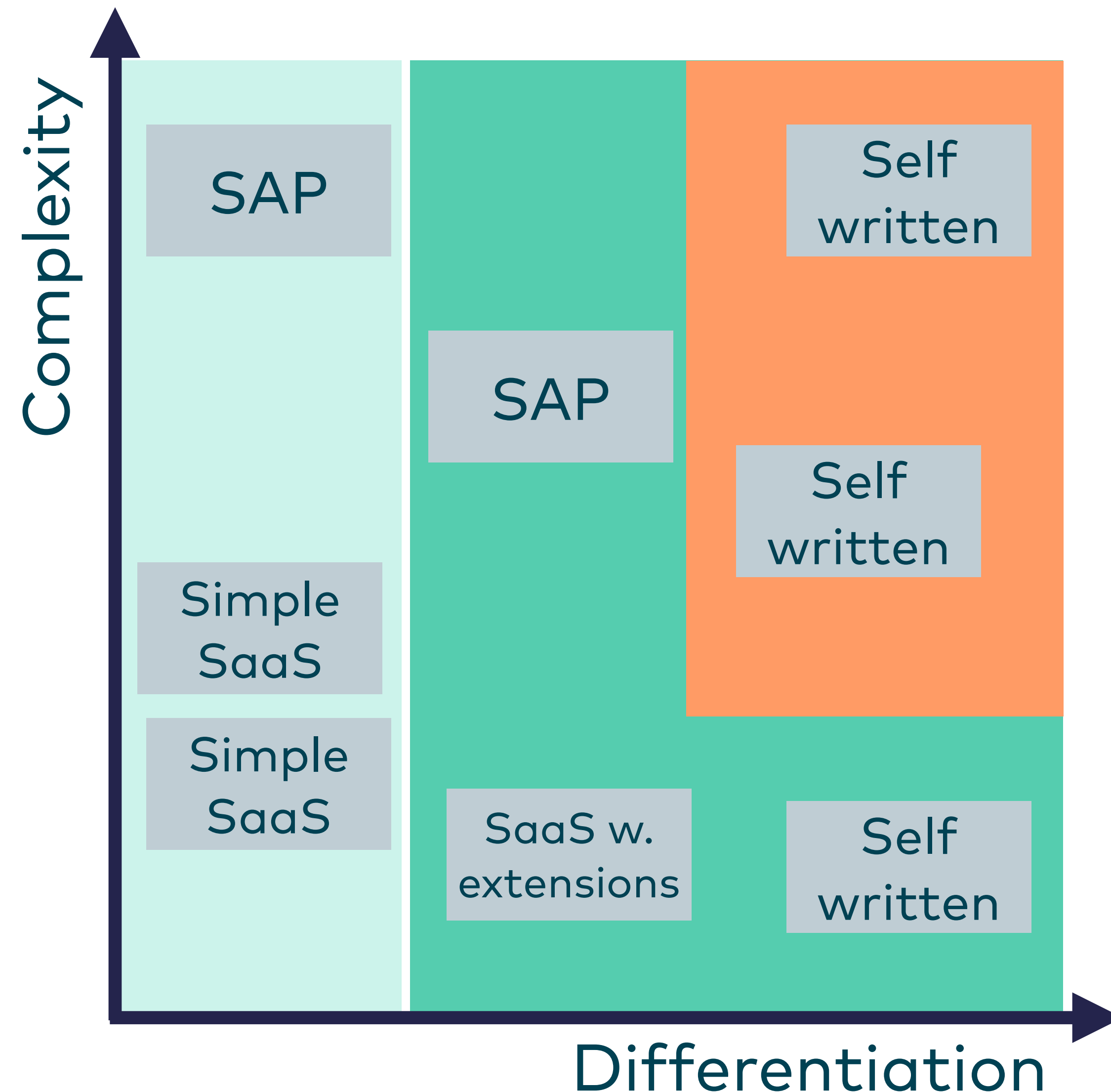
# Better:



- Follow a clear staffing strategy with regards to domain classifications
- Core: only really good internal teams
- Supporting: External ok, areas of high differentiation may work with mixed teams
- Generic: see next slide



# Make or Buy



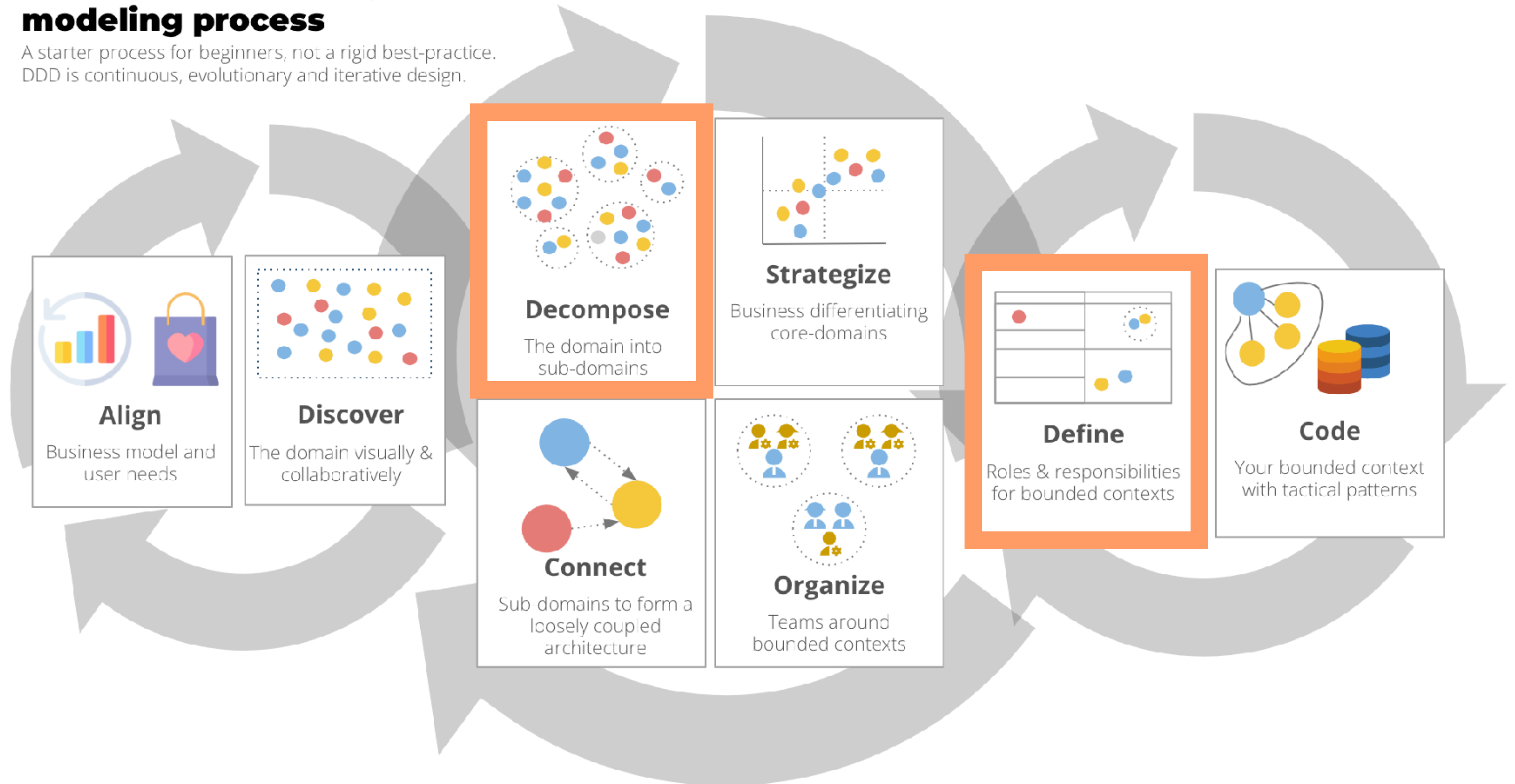
- Don't write your own software in non differentiating areas
- The categories may change over time. Something that was differentiating 15 years ago may be generic nowadays
- Mind this when modernizing your IT



**„We want agile and cross-functional teams  
which are autonomous and which can  
deliver fast“**

# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.



<https://github.com/ddd-crew/ddd-starter-modelling-process>



„Domains live in the problem space. They are how an organization perceives its areas of activity and expertise. **Bounded Contexts are part of the solution space; they are deliberate design choices. As a systems designer, you choose these boundaries to manage the understandability of the system,** by using different models to solve different aspects of the domain.“



## Mathias Verraes and Rebecca Wirfs-Brock

Quote from „Splitting a Domain Across Multiple Bounded Contexts“  
<https://verraes.net/2021/06/split-domain-across-bounded-contexts/>



**The bigger the alignment between  
problem and solution space, the better**

**But don't aim for ultimate perfection**

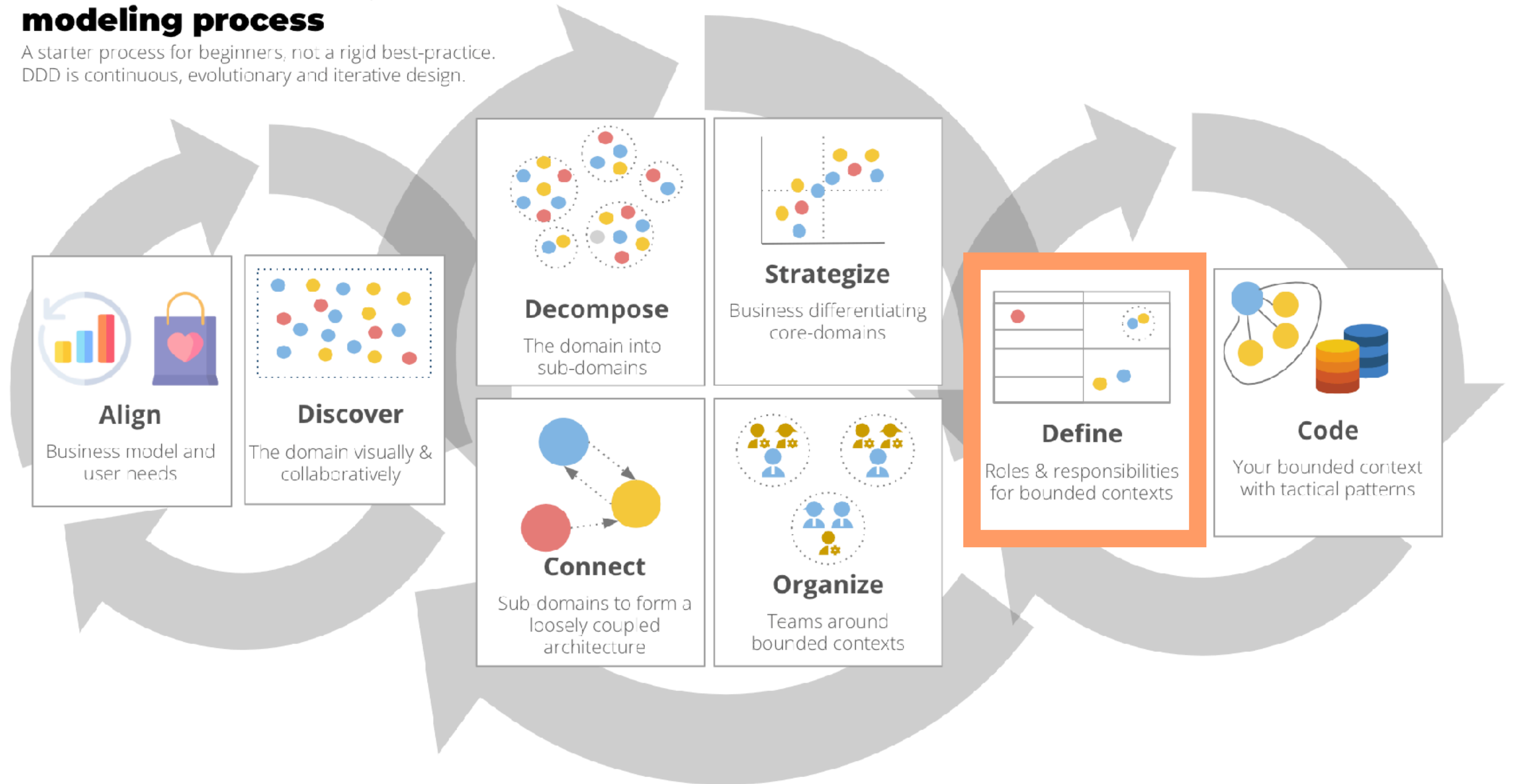
**A Bounded Context is a boundary for a model expressed in a consistent language tailored around a specific purpose**

**Bounded Context**







# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.



# Bounded Context Design Canvas

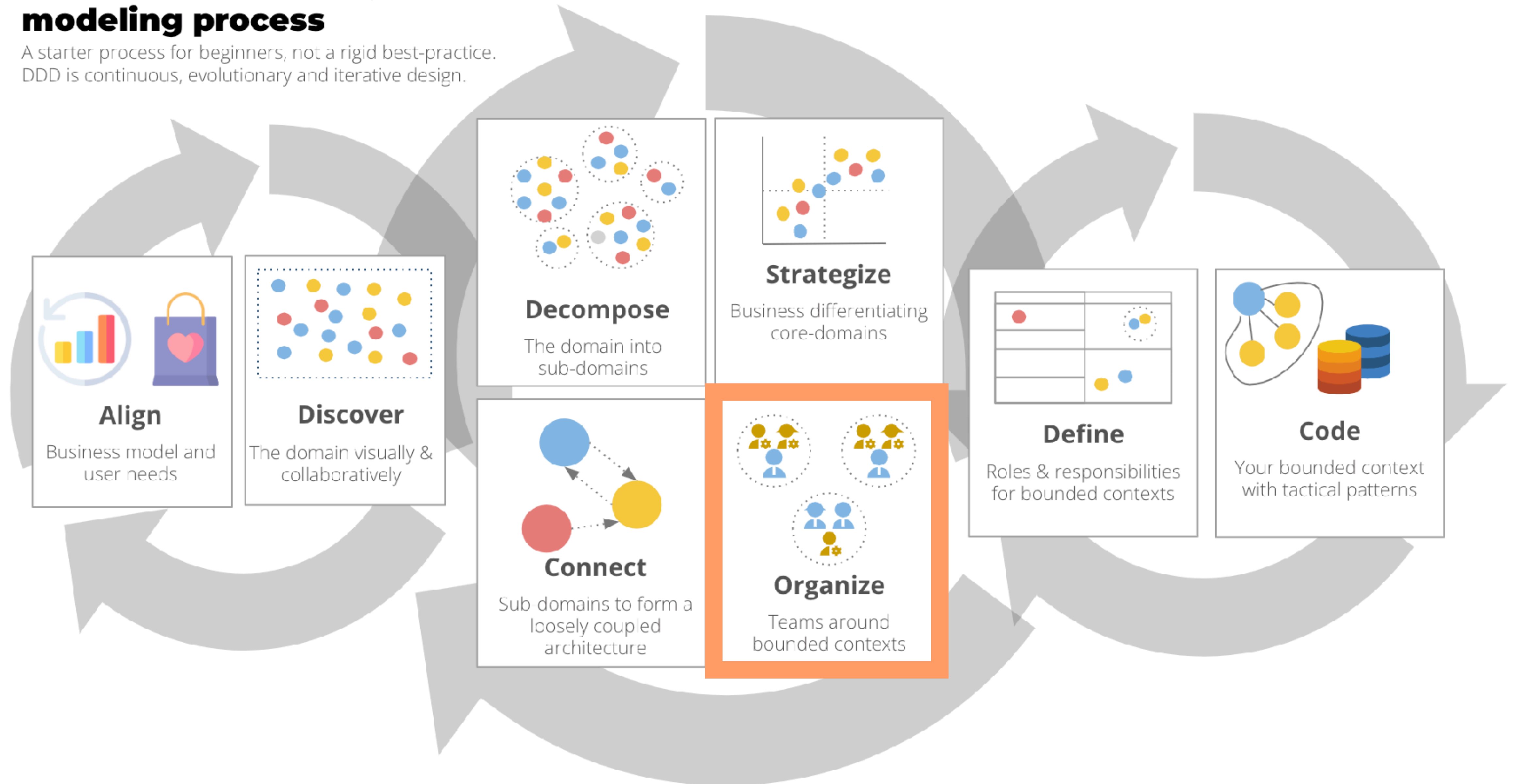
The canvas guides you through the process of designing a bounded context by requiring you to consider and make choices about the key elements of its design, from naming to responsibilities, to its public interface and dependencies.

<b>Name:</b>		V5 github.com/ddd-crew/bounded-context-canvas				
<b>Purpose</b> What benefits does this context provide, and how does it provide them? Describe the purpose from a business perspective	<b>Strategic Classification</b> <table border="1"><tr><td><b>Domain</b> - core - supporting - generic - other?</td><td><b>Business Model</b> - revenue - engagement - compliance - cost reduction</td><td><b>Evolution</b> - genesis - custom built - product - commodity</td></tr></table>		<b>Domain</b> - core - supporting - generic - other?	<b>Business Model</b> - revenue - engagement - compliance - cost reduction	<b>Evolution</b> - genesis - custom built - product - commodity	<b>Domain Roles</b> <b>Role Types</b> - draft context - execution context - analysis context - gateway context - other
<b>Domain</b> - core - supporting - generic - other?	<b>Business Model</b> - revenue - engagement - compliance - cost reduction	<b>Evolution</b> - genesis - custom built - product - commodity				
<b>Inbound Communication</b> <b>Collaborator</b> <b>Messages</b>  		<b>Ubiquitous Language</b> Context-specific domain terminology    <b>Business Decisions</b> Key business rules, policies, and decisions  	<b>Outbound Communication</b> <b>Messages</b> <b>Collaborator</b>  			
<b>Assumptions</b> Describe which currently unverified assumptions went into this bounded context design. Make those assumptions explicit by documenting them here	<b>Verification Metrics</b> Describe metrics which can be used to (in)validate the current structure of this bounded context?		<b>Open Questions</b>			

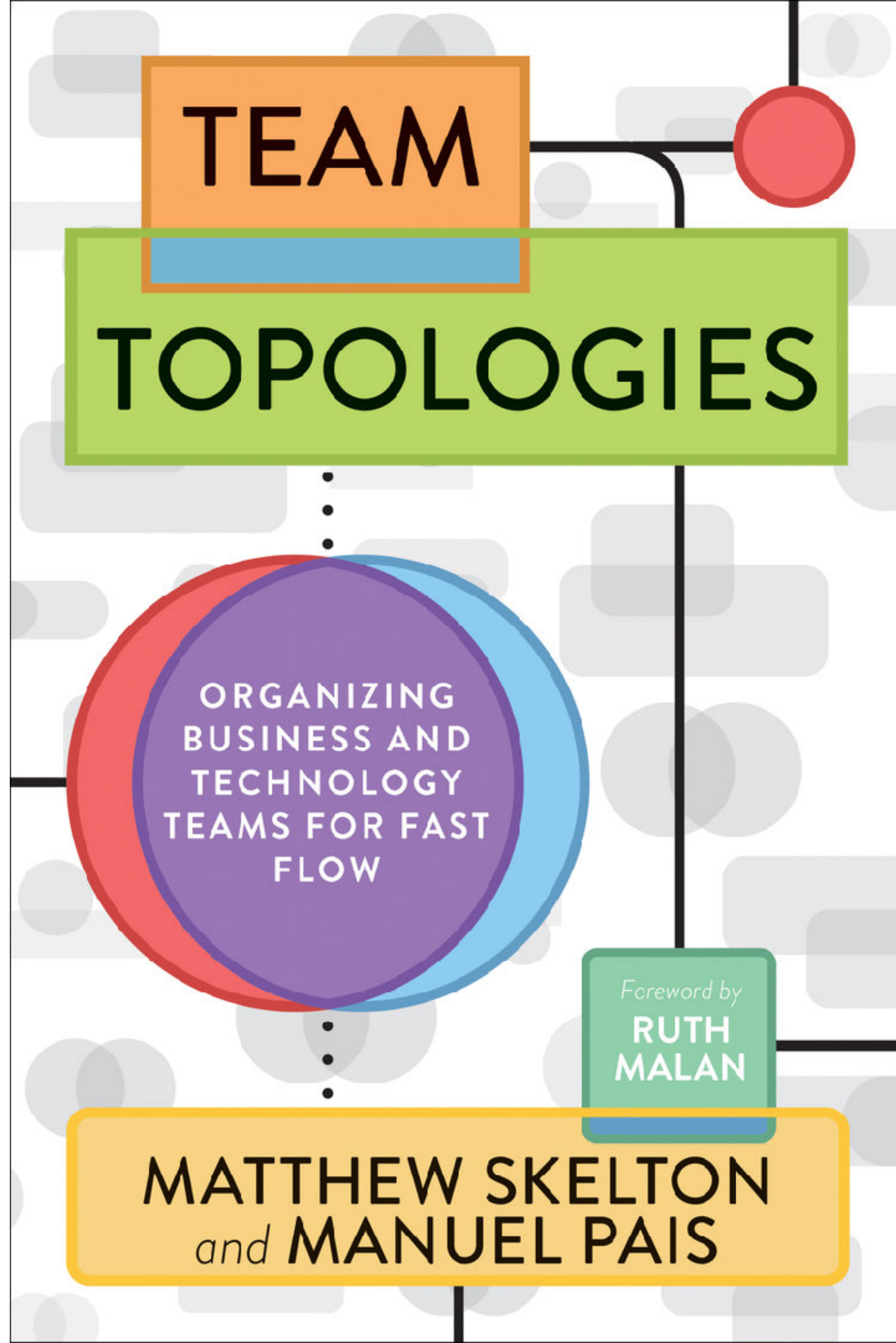
Source: <https://github.com/ddd-crew/bounded-context-canvas>

# Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.  
DDD is continuous, evolutionary and iterative design.





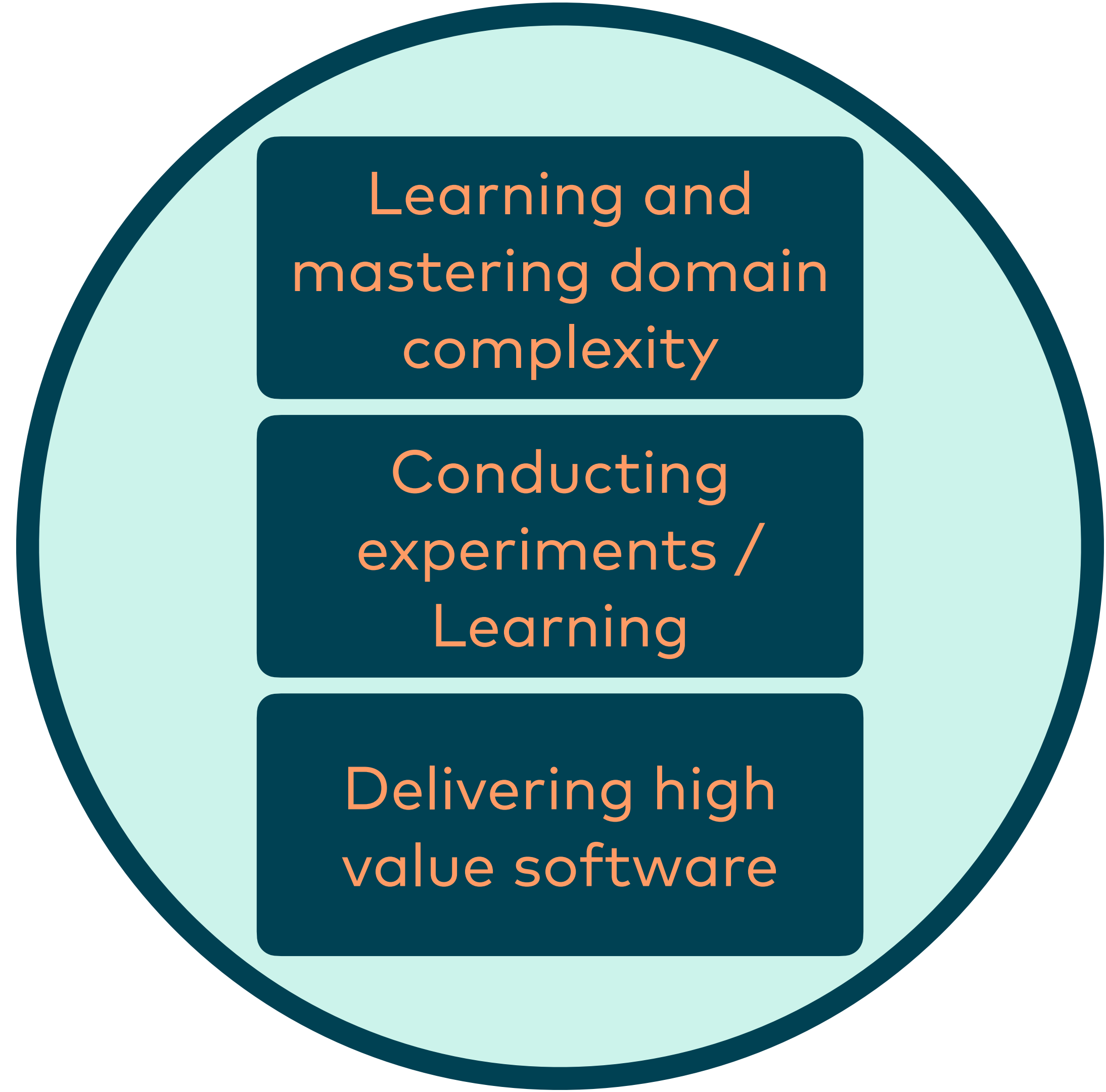


The  
Bounded Context  
is a

team first  
boundary

# Mind the **COGNITIVE LOAD**

of the team  
which is  
responsible for  
the bounded  
context



NEW YORK TIMES BESTSELLER

"Provocative and fascinating." —MALCOLM GLADWELL

Daniel H. Pink

author of *A Whole New Mind*

DRIVE

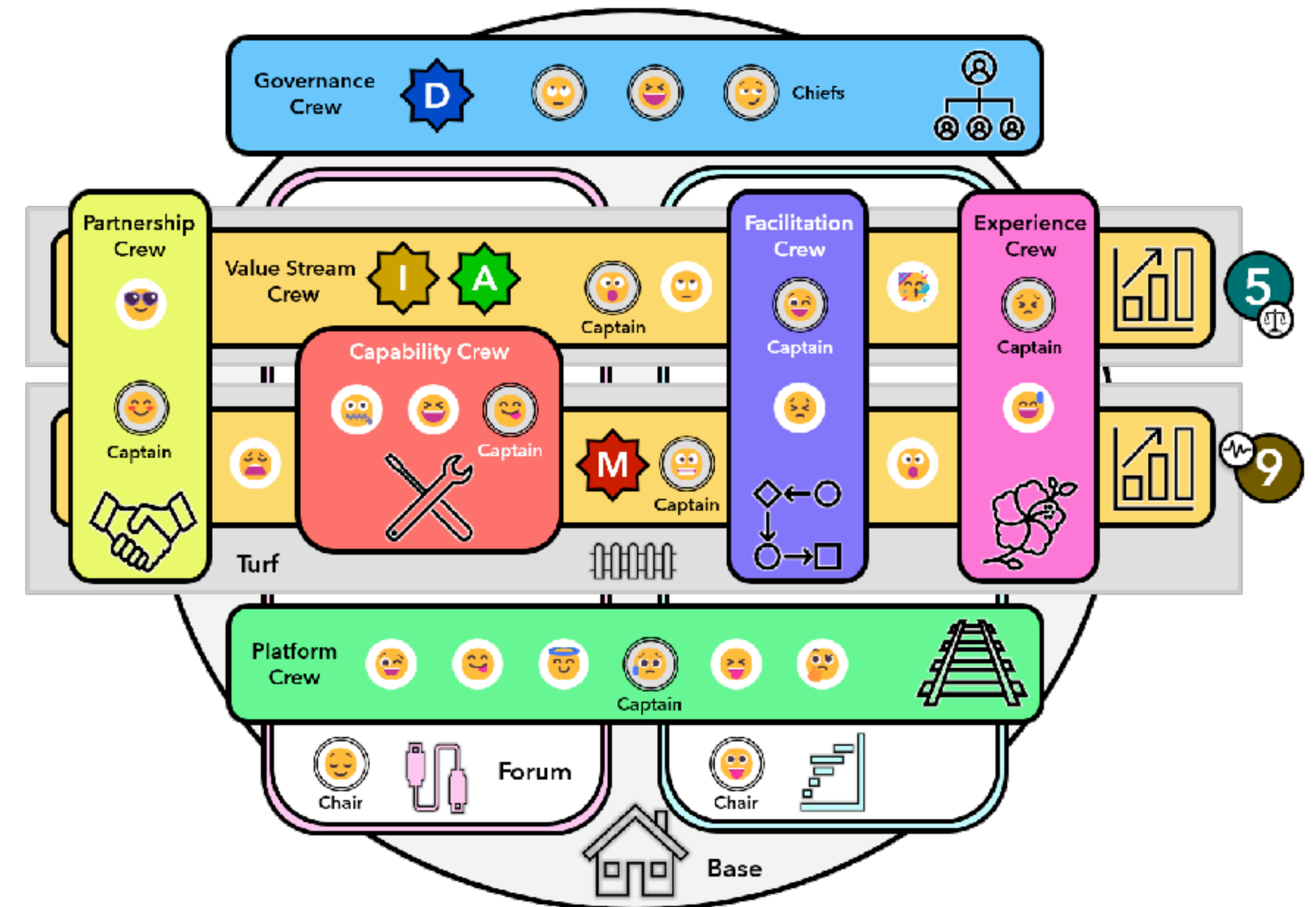
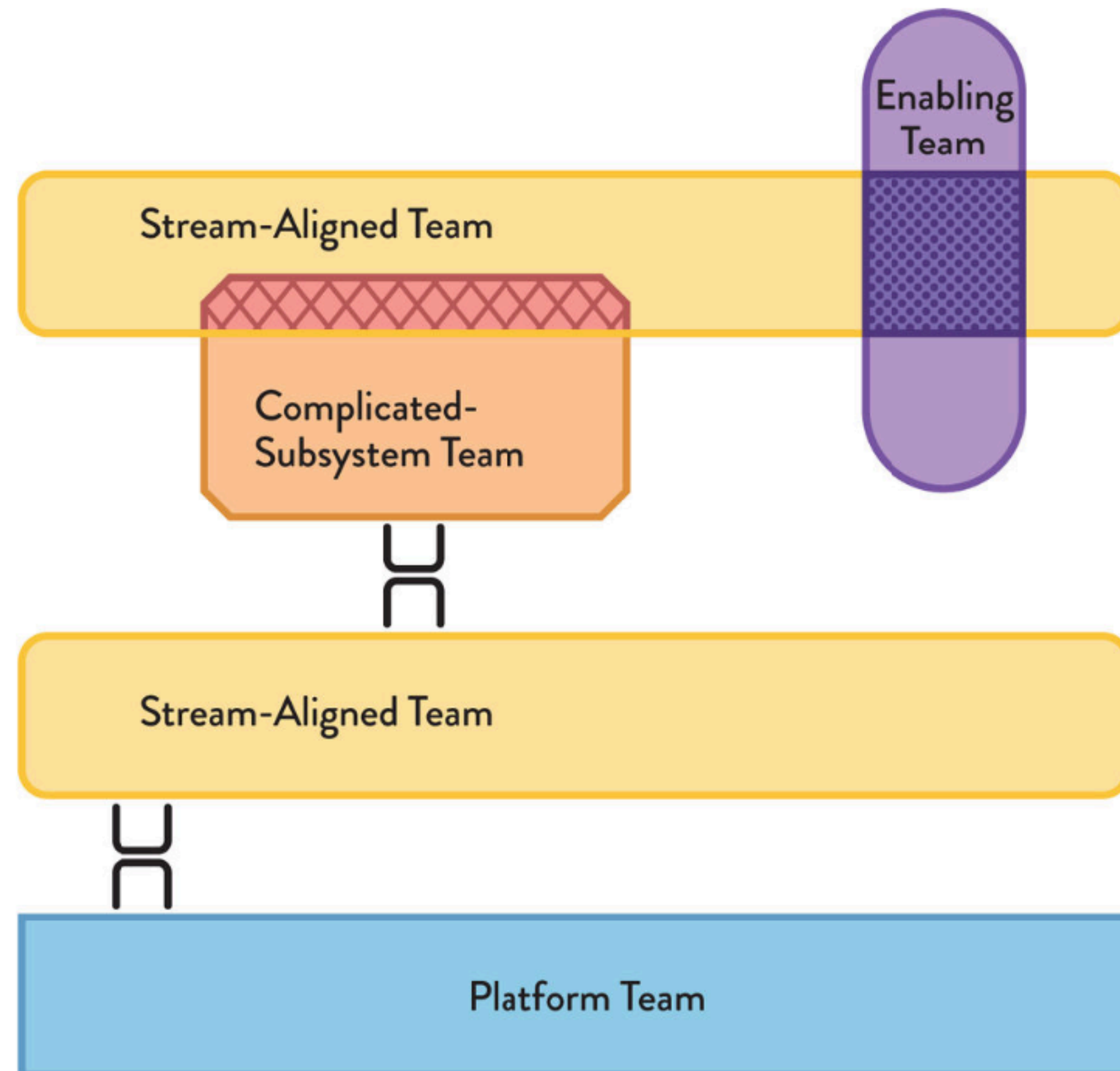
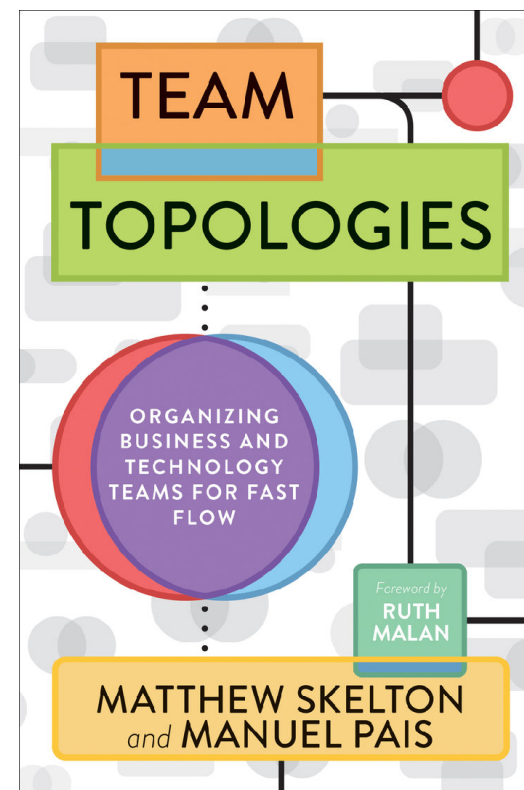
The Surprising Truth  
About What Motivates Us

**We need good  
boundaries in which  
teams can achieve**

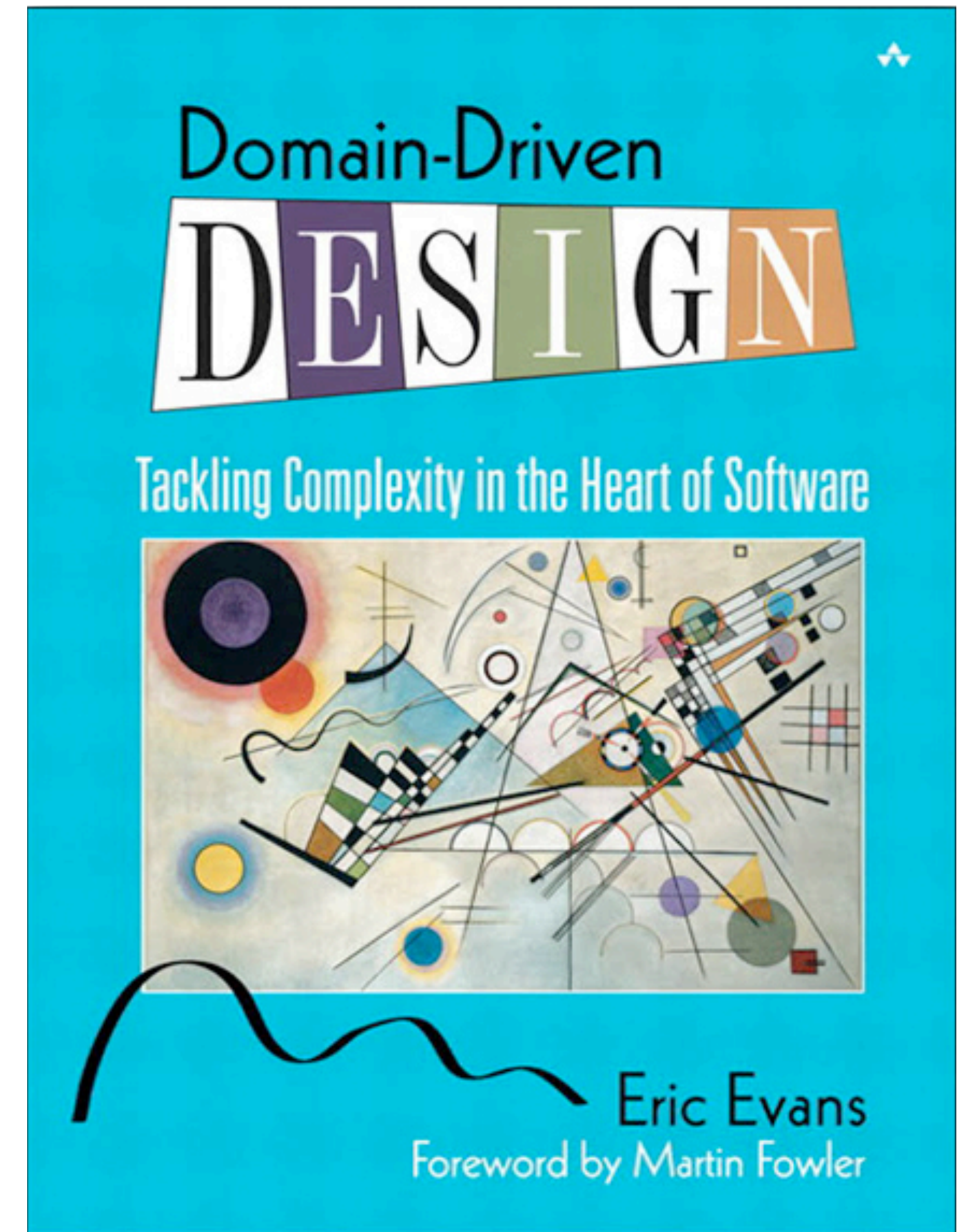
**Autonomy  
Mastery  
Purpose**







**Strategic Domain  
Driven Design**  
also has a technique  
to visualize  
sociotechnical  
relationships:  
**CONTEXT MAPS**



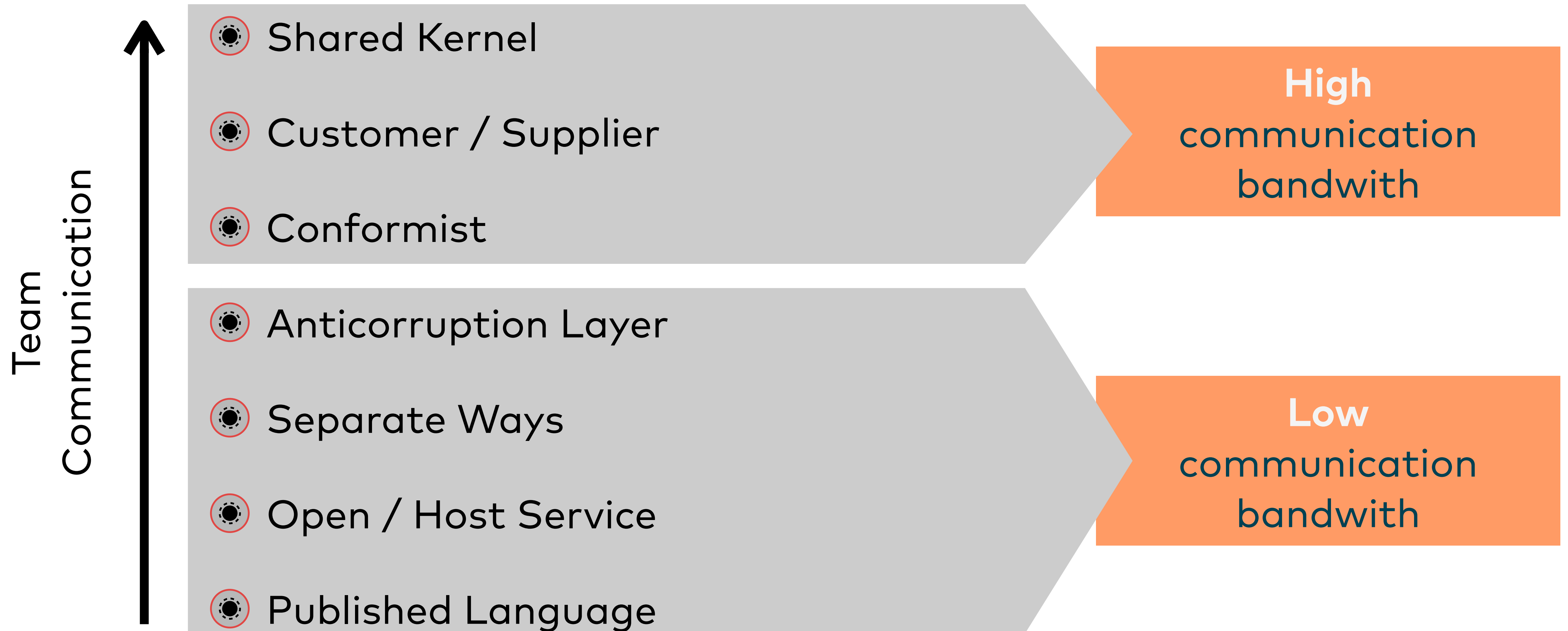


# The patterns address various aspects

	Team Relationships	Model Propagation	API / „technical“
Open-host Service	(✓)		✓
Anticorruption Layer		✓	
Conformist		✓	
Shared Kernel		✓	(✓)
Partnership	✓		
Customer-Supplier	✓		
Separate Ways	✓	(✓)	
Published Language	✓	(✓)	
Big Ball Of Mud			✓



# Mind team communication





**„Great, Domain Driven Design sounds like a  
silver bullet that solves everything.**

**Let's start a DDD-initiative"**



**Noooooooooo!**

**SORRY!**

**Nothing is this talk was a silver-bullet**

**You have to find out what works for you**



# Thank you!



Michael Plöd

E-Mail: [michael.ploed@innoq.com](mailto:michael.ploed@innoq.com)

Socials: [@bitboss@mastodon.social](https://mastodon.social/@bitboss)

LinkedIn: <https://www.linkedin.com/in/michael-ploed/>

## innoQ Deutschland GmbH

Krischerstr. 100  
40789 Monheim  
+49 2173 3366-0

Ohlauer Str. 43  
10999 Berlin

Ludwigstr. 180E  
63067 Offenbach

Kreuzstr. 16  
80331 München

Hermannstrasse 13  
20095 Hamburg

Erftstr. 15-17  
50672 Köln

Königstorgraben 11  
90402 Nürnberg