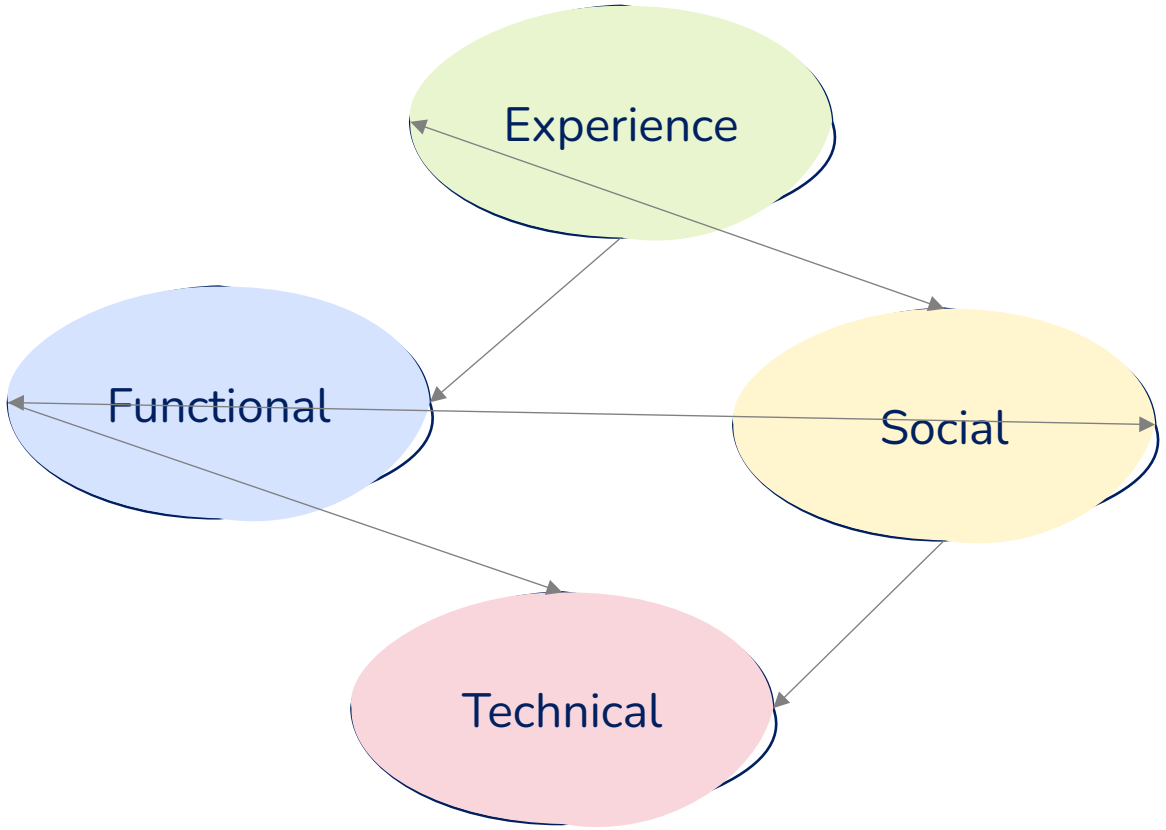


Distibuted Wars

These are not the boundaries you are looking for

Cédric Pontet

CTO @ Agile Partner, Founder @ #play14, Agile Coach, Software & Cloud Architect, EventStormer, and Sketchnoter

Luxembourg · 500+ connections · Contact info

Agile Partner S.A.

Université Henri Poincaré (Nancy I)

How can we split our system into microservices ?

What language will we code with?

What message broker will we pick?

What orchestrator will we use?

Should we use REST or gRPC?

agilepartner

How small should a microservice be?

“La bonne longueur pour les jambes, c'est quand les pieds touchent bien par terre.

The right length for legs is when feet are touching the floor.

— Coluche”

CC BY-NC

These are all important questions

But they are not the questions you should start with

These are not the ~~droids~~ boundaries
you are looking for...

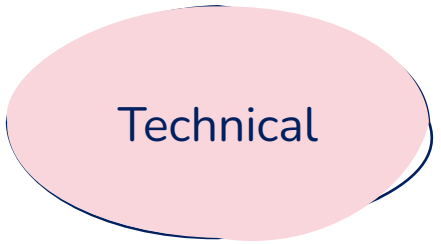
You should start by asking what, why and for whom

Four types of boundaries

Yes! There is more than one

Technical boundaries

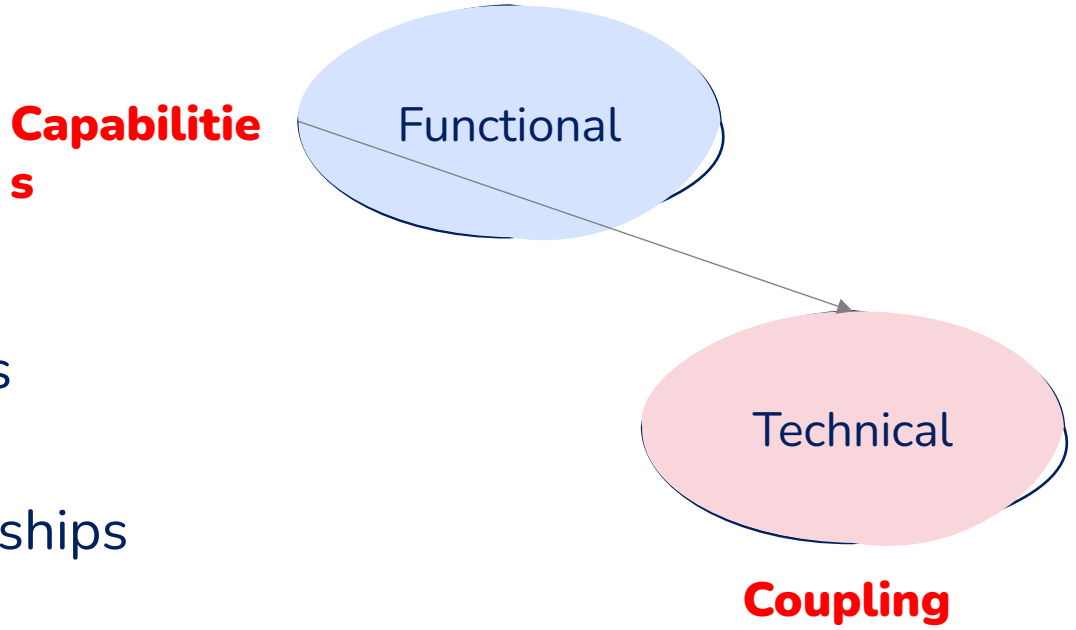
- Technology & tools
- Coding language
- Independent deployment units
- Frequency of change
- Communication
  - Ingress
  - API call vs Messaging
  - Sync vs async
- Technical dependencies
- Scalability & resilience
- Security



Coupling

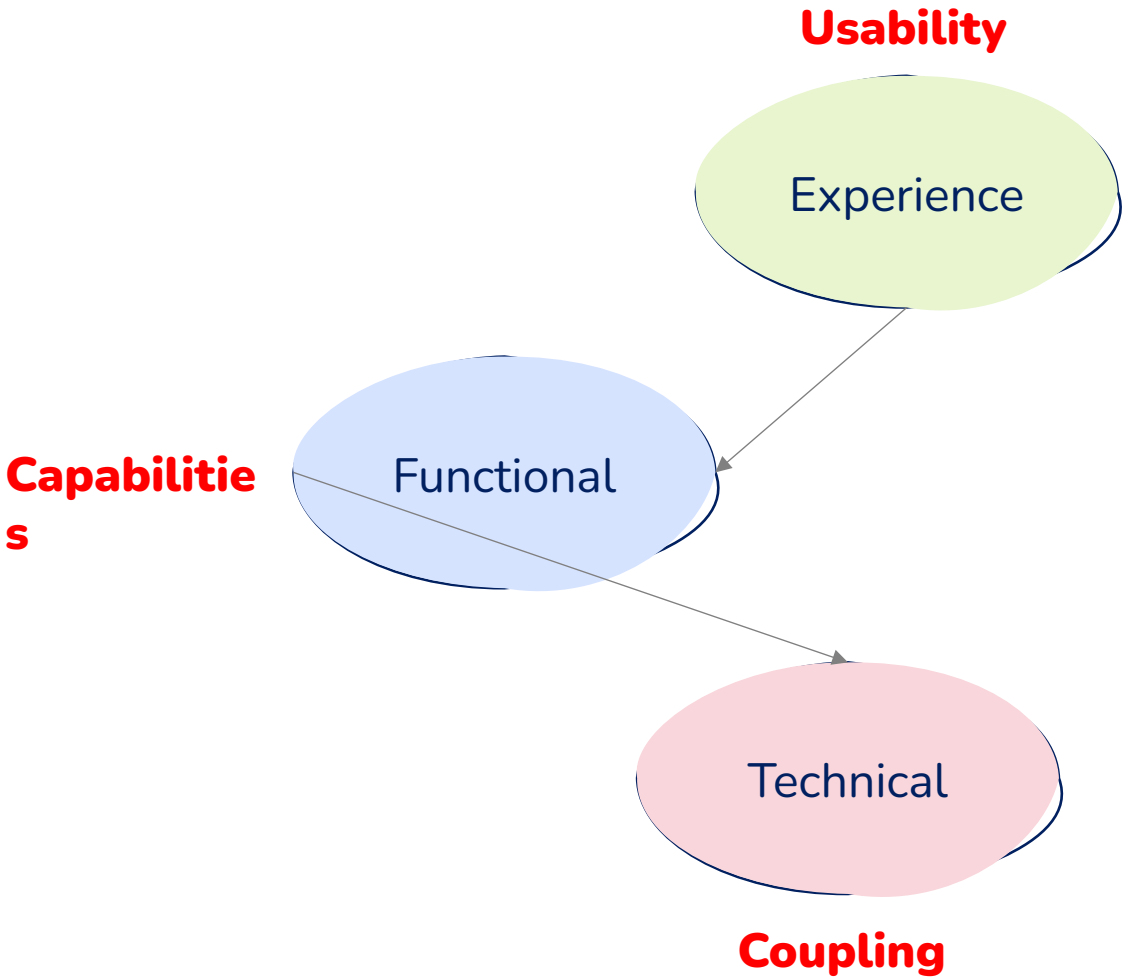
Functional boundaries

- a.k.a. Domain in Domain-Driven Design
- Domains are hierarchical
- Type of sub-domains
  - Core
  - Supporting
  - Generic
- Domain Language
- Consistency
- Entities & business rules
- Processes & policies
- Collaboration & relationships
- Legal & compliance



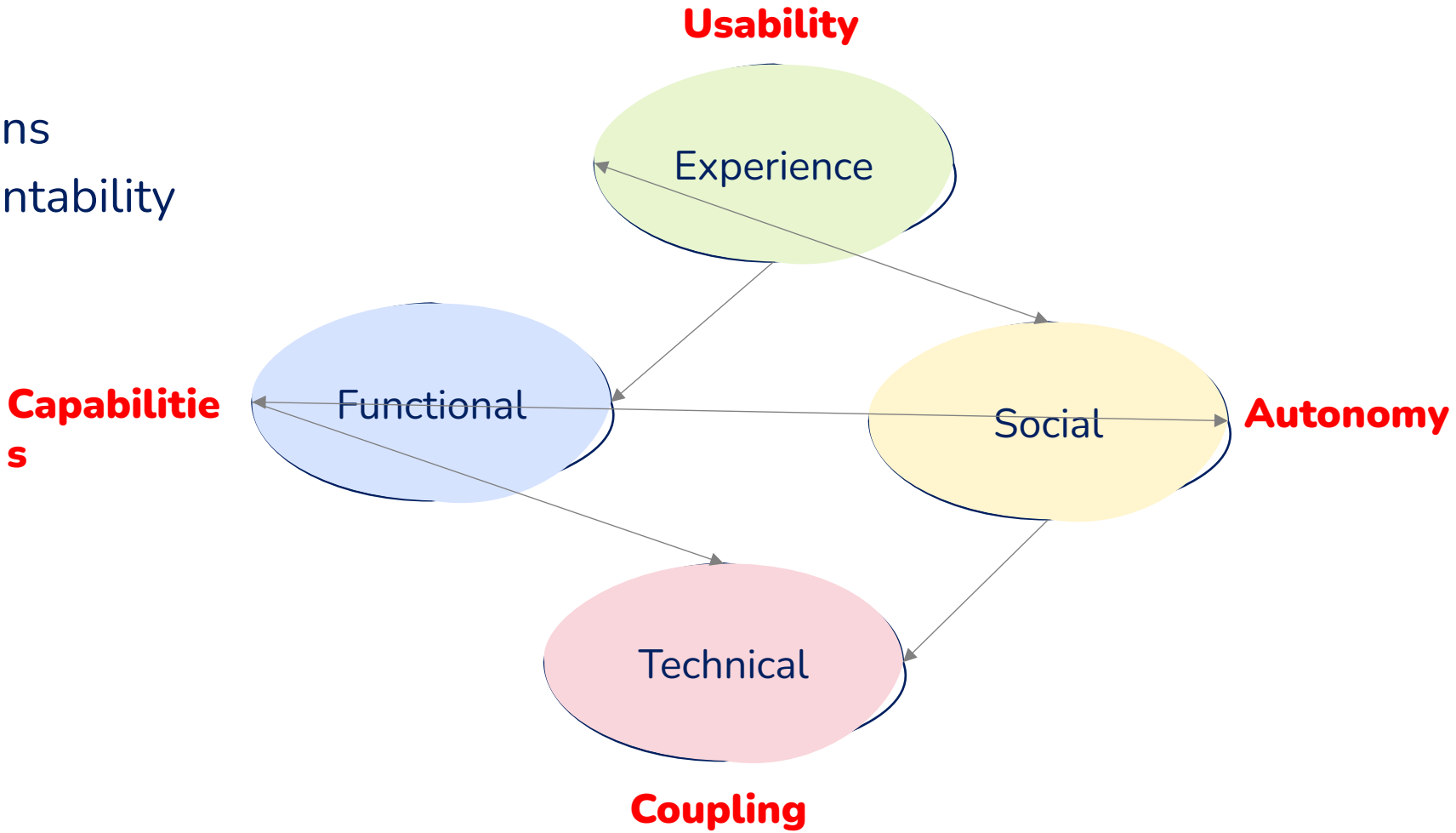
Experience boundaries

- Cohesive set of features
- Customer journey
- Utility
- Ease of use
- Efficiency
- Accessibility



Social boundaries

- People & their interactions
- Responsibilities & accountability
- Collaboration
- Skills
- Culture
- Learning
- Mental load
- Ability to change
- Evolution over time



Technical boundaries

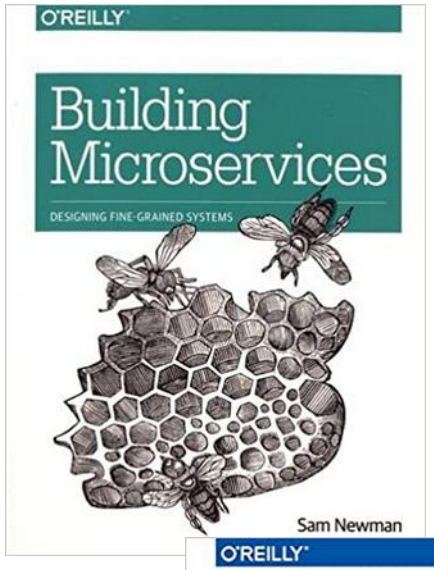
Microservices?

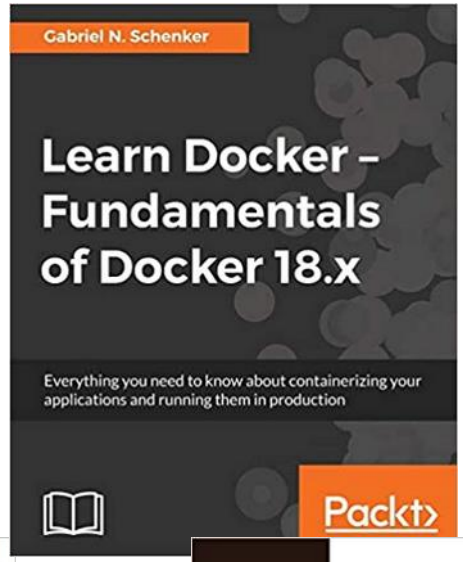
Welcome to the world of buzz words

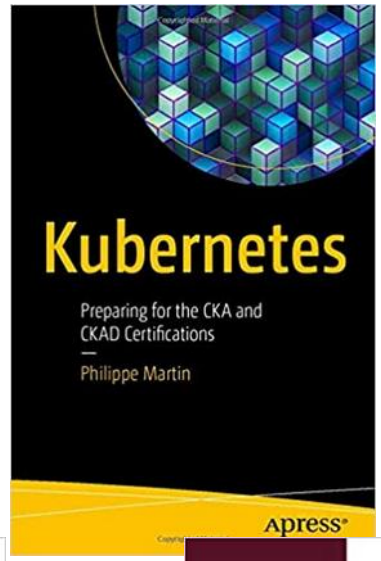
- Container runtime
- Orchestrator
- Volumes
- API Gateway
- Service Mesh
- Observability
- Control plane
- Pods
- Ingress
- Sidecar proxy
- Messaging system
- Distributed tracing
- Monitoring & alerting

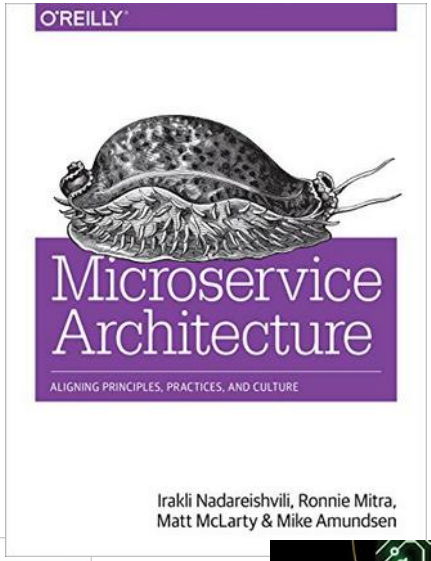
Prometheus

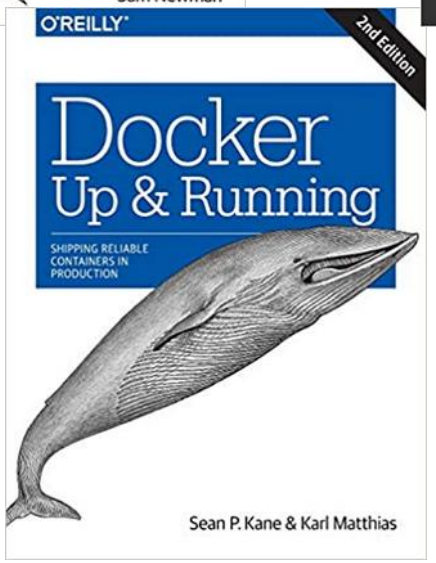
AWS Lambda

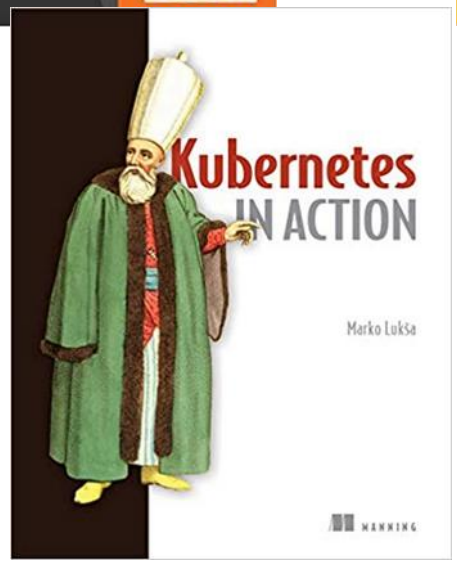

O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman

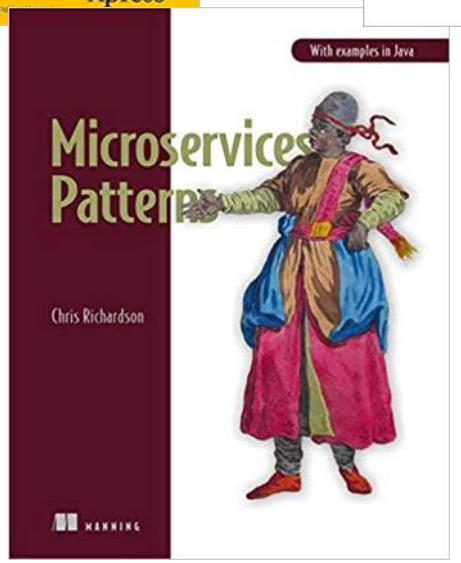

Gabriel N. Schenker
Learn Docker – Fundamentals of Docker 18.x
Everything you need to know about containerizing your applications and running them in production
Packt

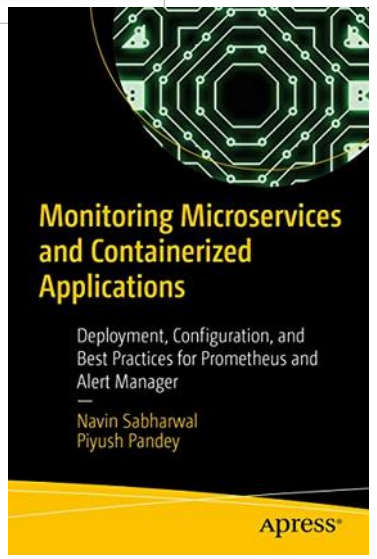

Kubernetes
Preparing for the CKA and CKAD Certifications
Philippe Martin
Apress


O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Irakli Nadareishvili, Ronnie Mitra, Matt McLarty & Mike Amundsen

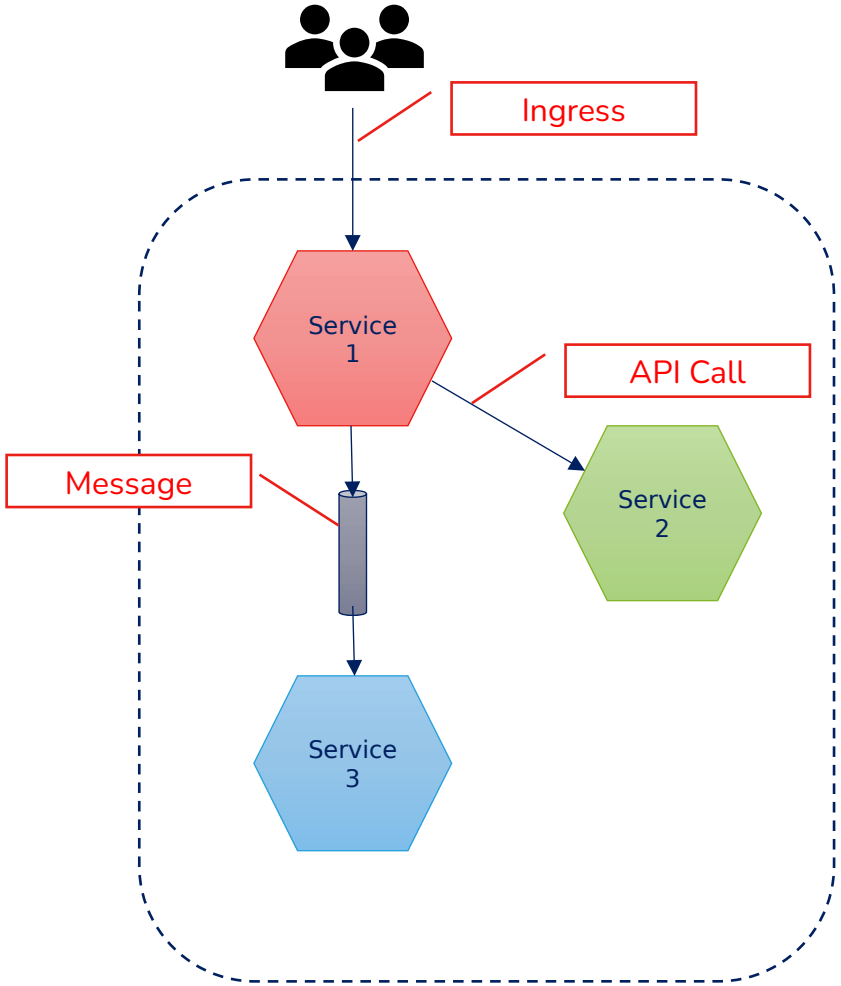

O'REILLY
2nd Edition
Docker Up & Running
SHIPPING RELIABLE CONTAINERS IN PRODUCTION
Sean P. Kane & Karl Matthias


Kubernetes IN ACTION
Marko Lukša
MANNING

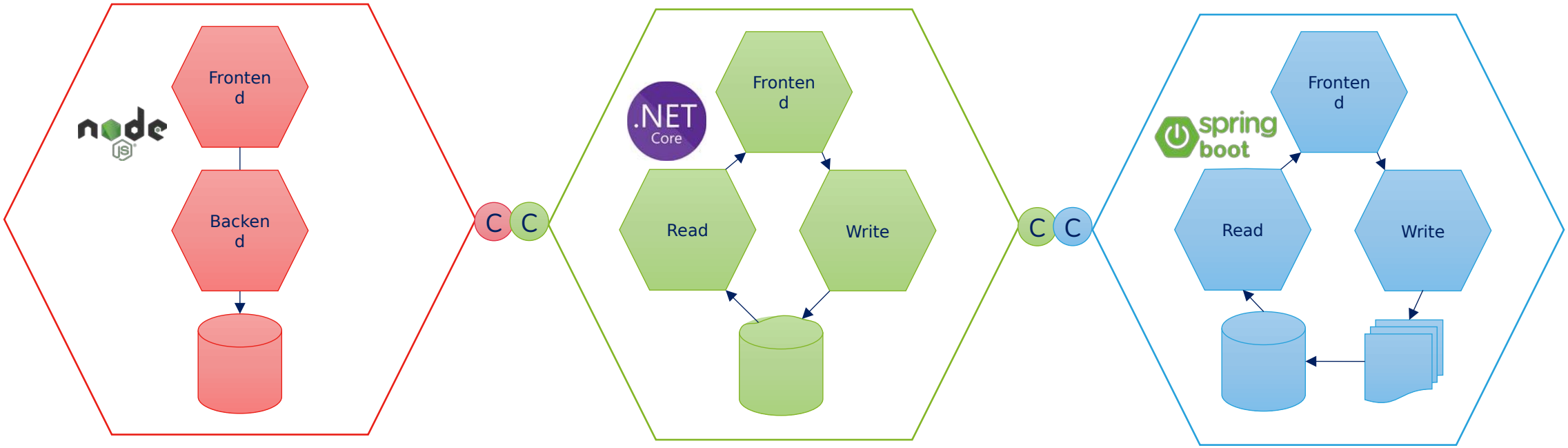

With examples in Java
Microservices Patterns
Chris Richardson
MANNING


Monitoring Microservices and Containerized Applications
Deployment, Configuration, and Best Practices for Prometheus and Alert Manager
Navin Sabharwal
Piyush Pandey
Apress

Let’s talk about coupling



Loose-coupling

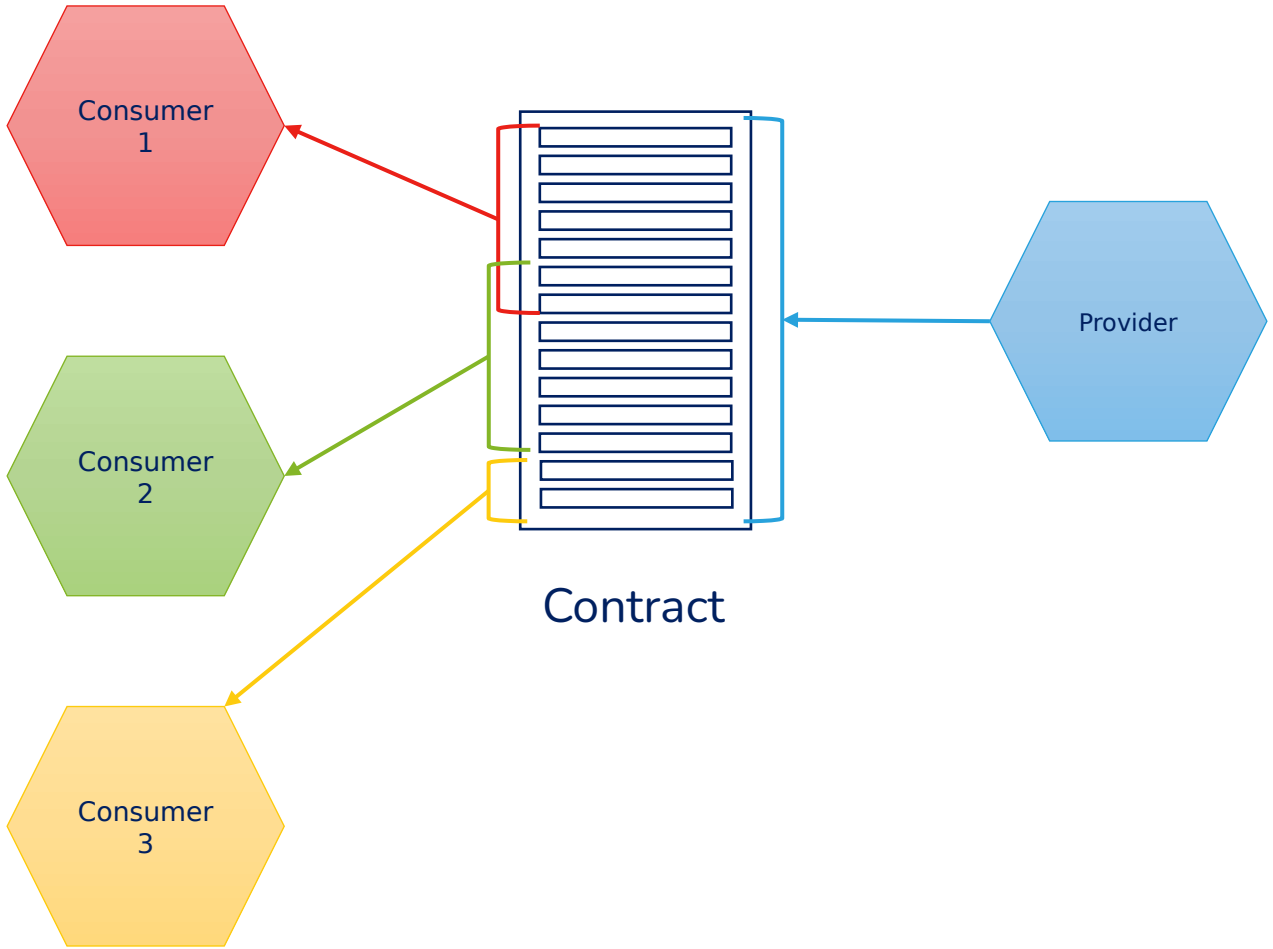


CRUD

CQRS

Event Sourced

Consumer-Driven Contract



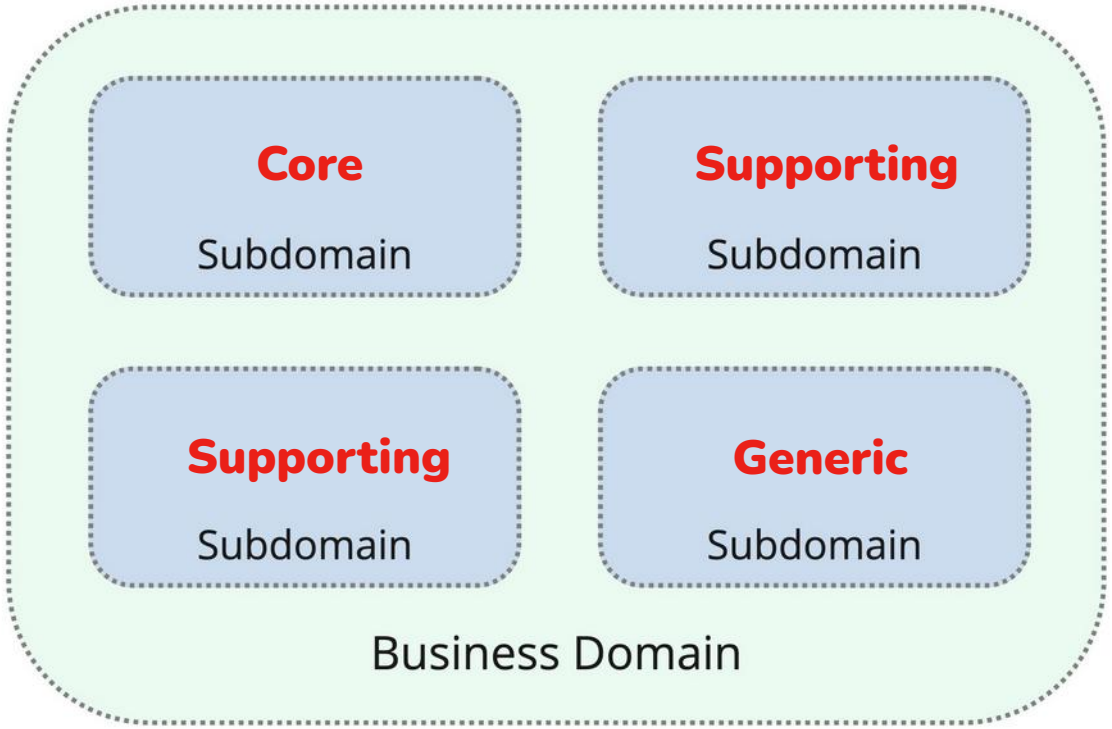
Standards

Functional boundaries

a.k.a domain boundaries

agilepartner

Hierarchical Domain



Core

Key differentiator for the business and the most valuable part of the application

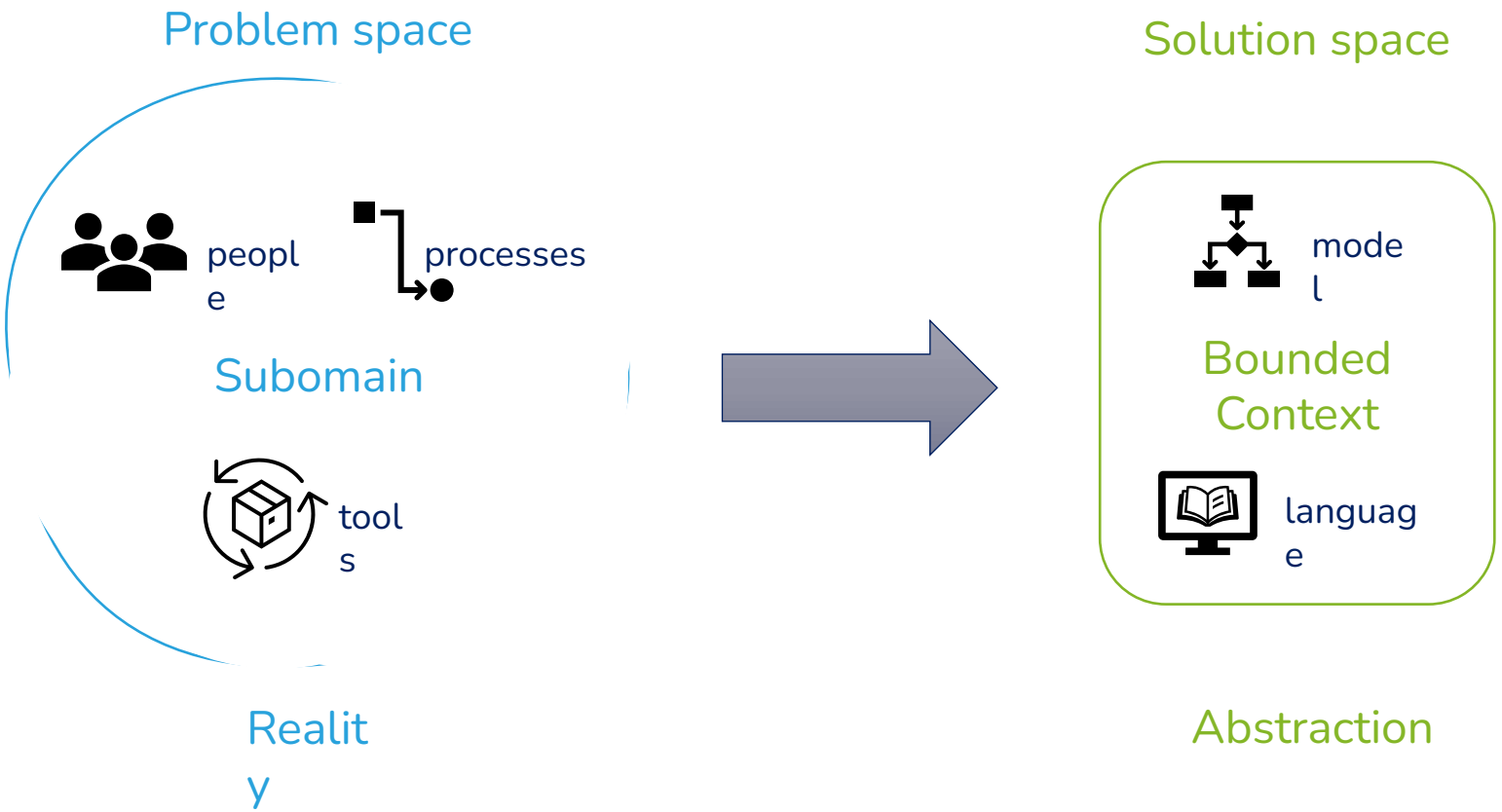
Supporting

Related to what the business does but not a differentiator

Generic

Not specific to the business and are ideally implemented using off the shelf software

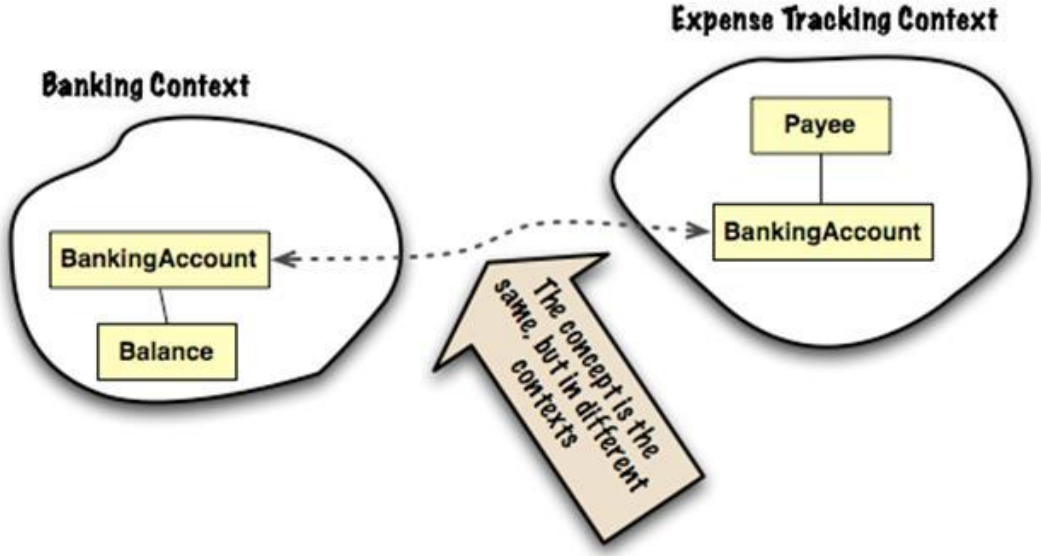
From subdomain to Bounded Context



What is a Bounded Context?

DDD deals with **large models** by **dividing** them into different Bounded Contexts and being **explicit** about their **interrelationships**.

A bounded context is a sub-system in a software architecture aligned to a part of your domain.



Bounded Context are like Pokémons

- They are many
- They have different shapes
- They have different sizes
- They are sometimes fuzzy

They are difficult to catch

CC BY-SA-NC

Example

Sales

- Customer relationship
- Booking
- Quotes
- Marketing

Global Logistics

- Cargo Management

Flight Ops

- Flight Safety
- Flight Planning
- Onboard equipment
- Crew Management

Messaging

- SITA
- Customs

Maintenance & engineering

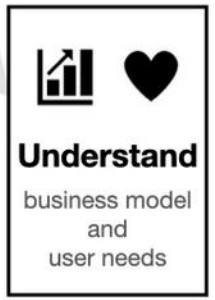
- Aircraft maintenance
- MBF/MPT Requests
- Fleet Support

Finance

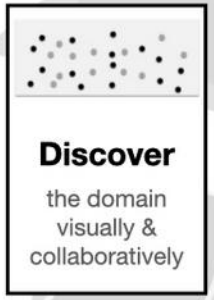
- Accounting
- Analytics

Domain-Driven Design Starter Modelling Process

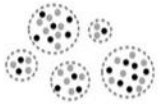
A starter process for beginners, not a rigid best-practice. DDD is continuous, evolutionary, and iterative design.



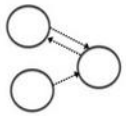
Understand
business model and user needs



Discover
the domain visually & collaboratively



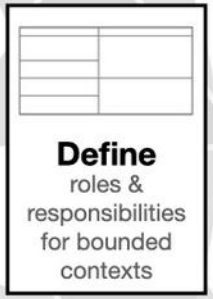
Decompose
the domain into sub-domains



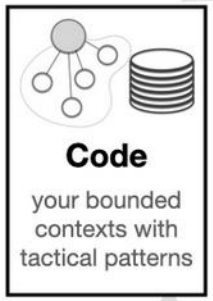
Connect
sub-domains to form a loosely-coupled architecture

Strategize
business-differentiating core-domains

Organise
teams around bounded contexts

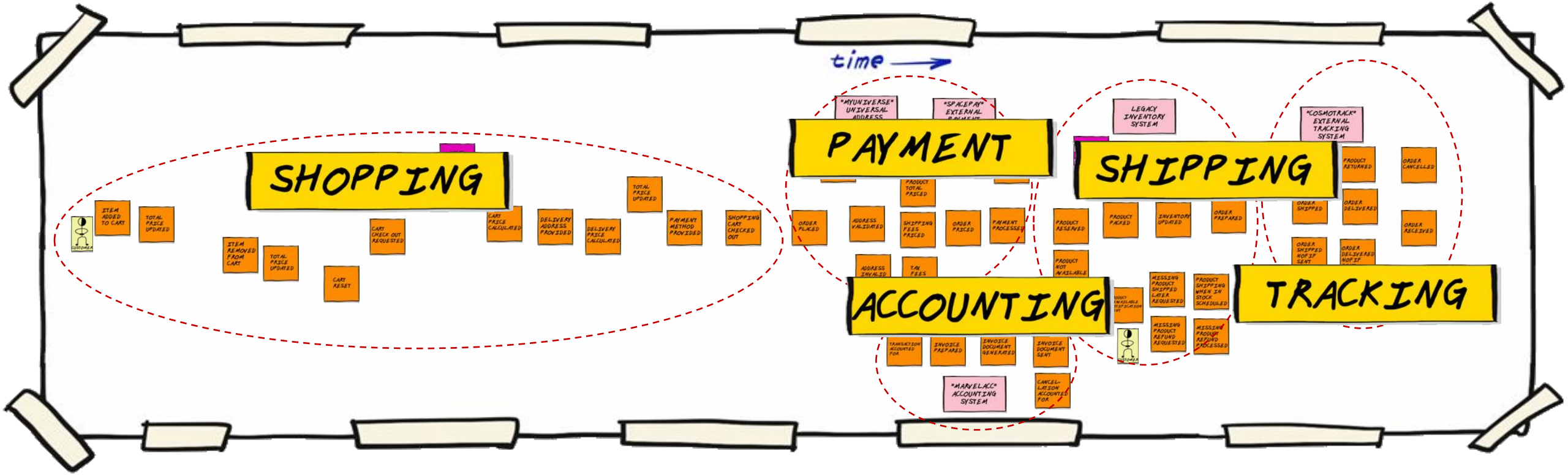


Define
roles & responsibilities for bounded contexts

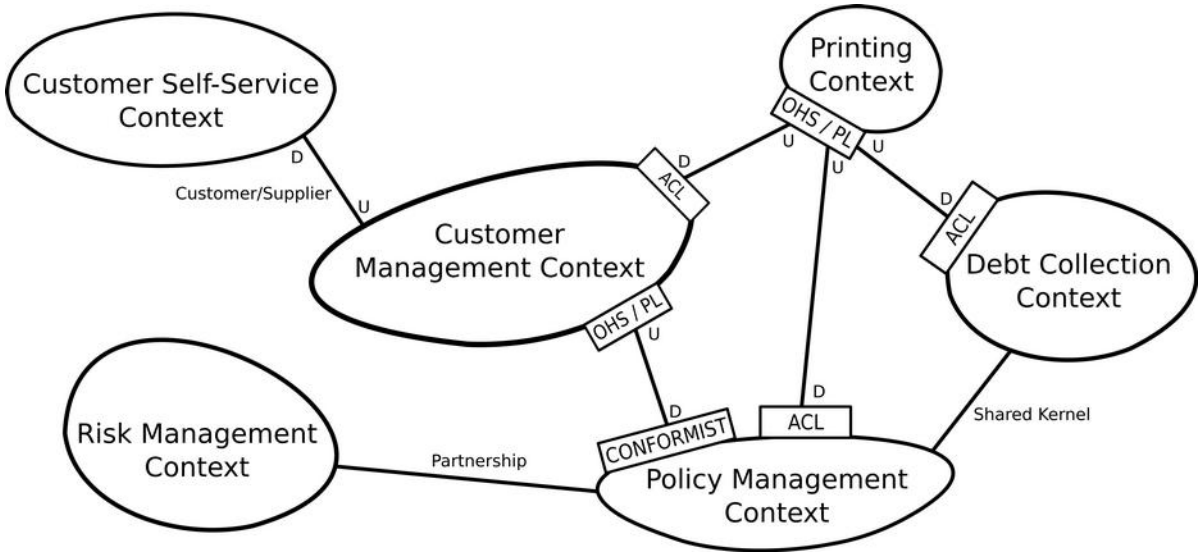


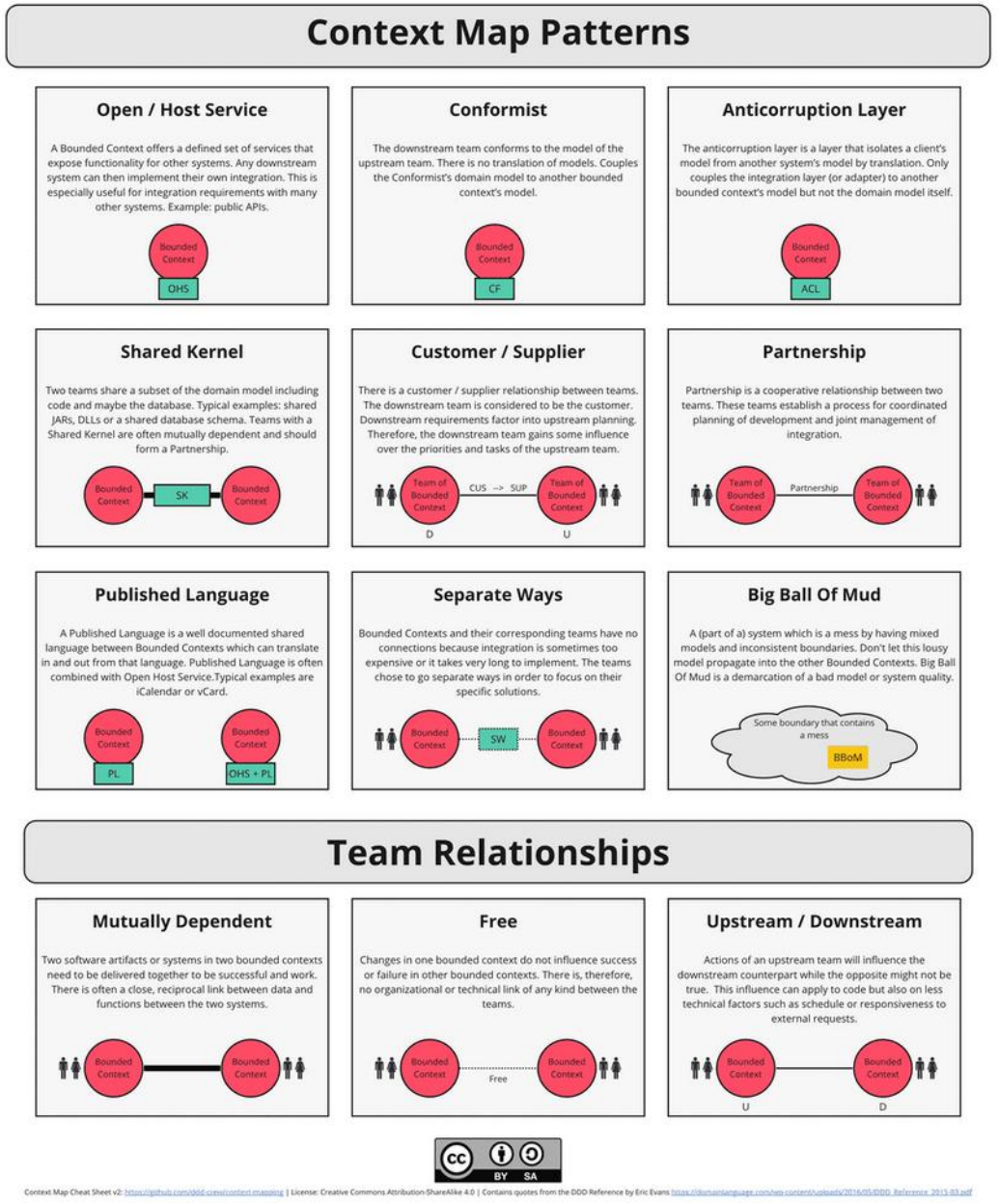
Code
your bounded contexts with tactical patterns

EventStroming Big Picture



Context mapping





Context Mapper: A DSL for Service Decomposition – Stefan Kapferer

Virtual Domain-Driven Design - A community of practise (virtualddd.com)

Bounded Context Canvas

Nick Tune

Name	Model Traits draft, execute, audit, enforcer, interchange, gateway, etc.
Description Summary of purpose and responsibilities	**Messages Consumed and Produced** Messages Consumed: Commands Handled; Events Handled; Queries Handled Messages Produced: Commands Sent; Events Published; Queries Invoked
Strategic Classification Domain: Core, Supporting, Generic, Other Business Model: Revenue, Engagement, Compliance, Cost reduction Evolution: Genesis, Custom built, Product, Commodity	
Business Decisions Key business rules, policies, and decisions	**Dependencies and Relationships** Message Suppliers: Name, Relationship Message Consumers: Name, Relationship
Ubiquitous Language Key domain terminology	

Bounded Context Canvas V3 Nick Tune | ntcoding.co.uk

Bounded Context and microservices

Bounded Contexts are built around the ***purpose*** of the model, **Microservices** around the ***deployment boundaries***.

The driving forces can be similar in some scenarios,
but they're not the same.

> ***Keep your boundaries logically clean, enforce physical separation when it's worth the cost***
>
> — ***Alberto Brandolini***

Bounded Contexts and teams

The delimited applicability of a particular model.

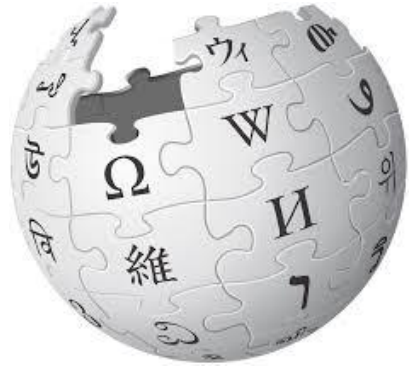
> ***BOUNDING CONTEXTS gives team members a clear and shared understanding of what has to be consistent and what can develop independently***
>
> — ***Eric Evans***

Experience boundaries

Let’s think about our users for a minute

User Experience

User experience (UX or UE) is about how a user interacts with, and experiences, a particular **product**, **system** or **service**.
It includes a person's perceptions of **utility**, **ease of use**, and **efficiency**.



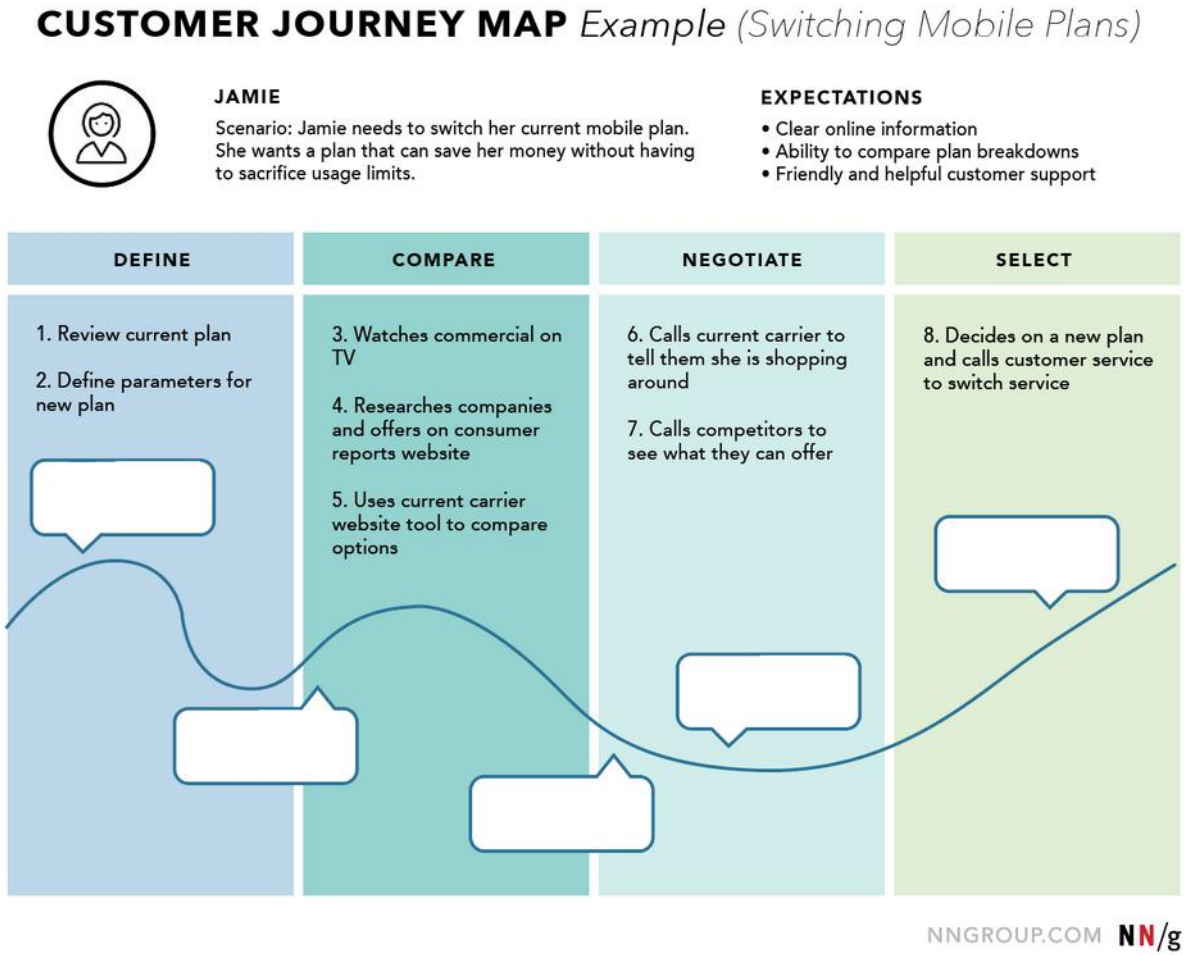
User experience is an important concern to many companies when creating products, as negative user experience **may decay profitability**.
User experience is subjective.

Customer journey

Visualization of the process that a person goes through in order to accomplish a goal tied to a specific business or product

Why use it

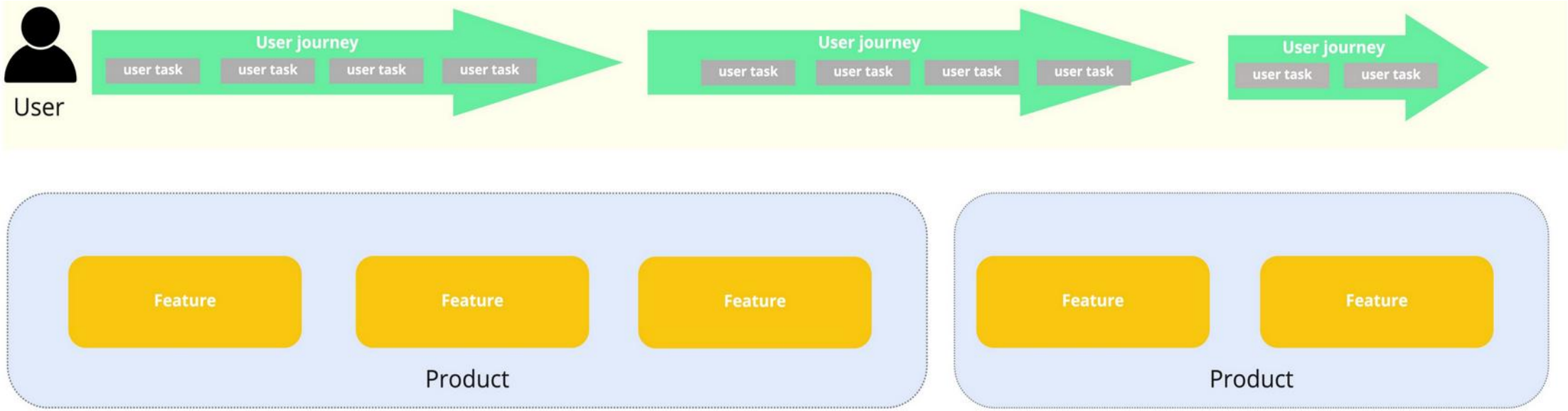
- To pinpoint **specific customer journey touchpoints** that cause pain or delight
- To break down silos to **create one shared, organization-wide understanding** of the customer journey
- To **assign ownership** of key touchpoints in the journey **to internal departments**



UX Mapping Methods Compared: A Cheat Sheet (nngroup.com)

Think outside-in

A user experiences the product by carrying out user journeys interacting with one or more features across one or more products.

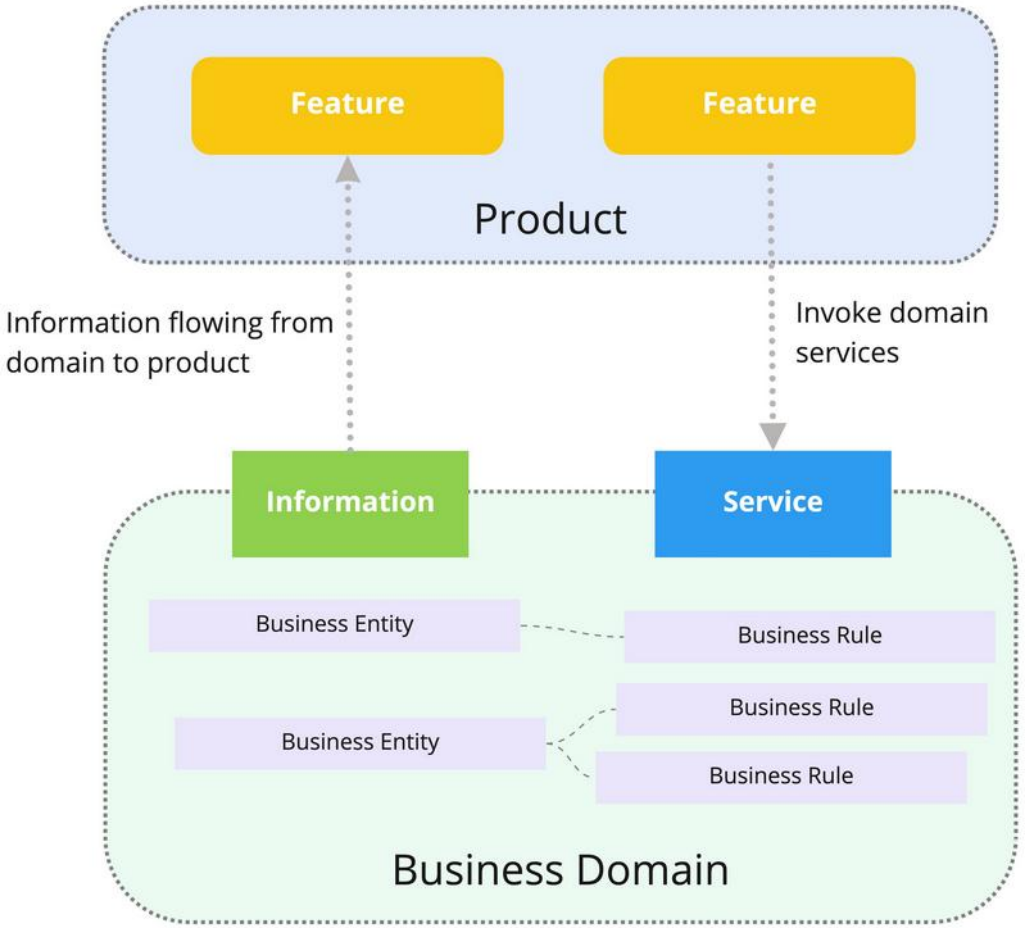


User journeys and experiences are what the user does and feels.
They are outside of the software.

Product variants

A product may have multiple variants such as web, android, and iOS apps which all offer **identical or similar functionality** delivering the **same value propositions** to the **same customer segments**.

Product using services from the domain



A product feature may display information such as a bank balance or perform actions like adding funds to a bank balance.

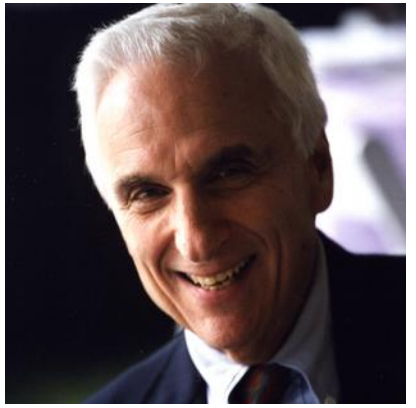
These pieces of information and actions are part of a business domain.

Essentially, product features expose domain information and capabilities.

Social boundaries

What we can learn from Conway's Law

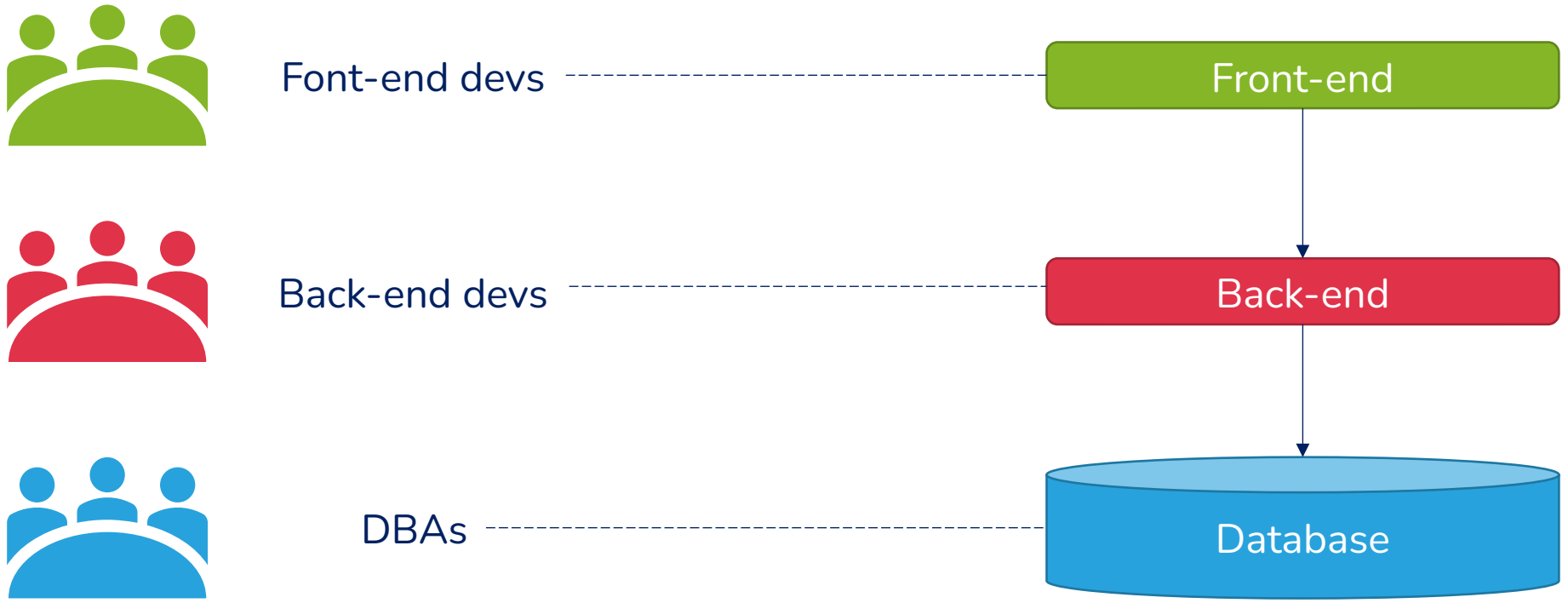
Conway’s law



> **Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.**
>
> ***— Melvin E. Conway***

Software ends up "shaped like" the organizational structure they are designed in or designed for

Conway's law applied



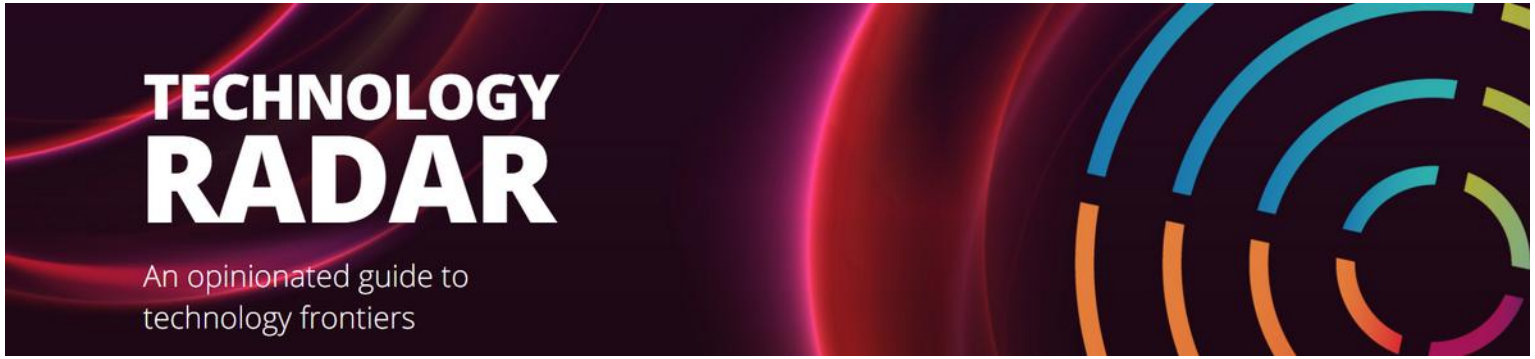
Specialized teams

Layered monolith

Inverse Conway maneuver

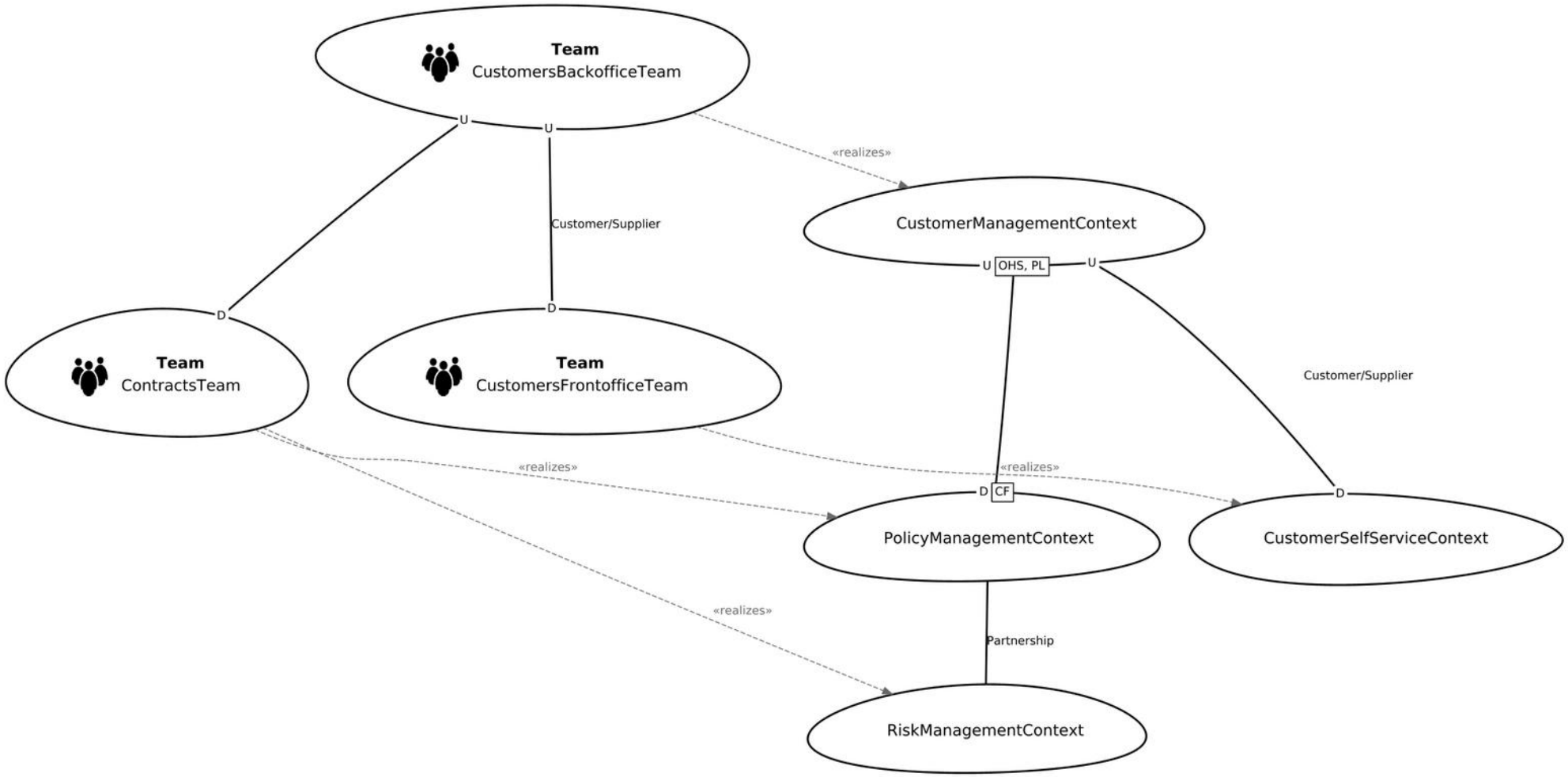
Cross-functional teams

Loosely-coupled services



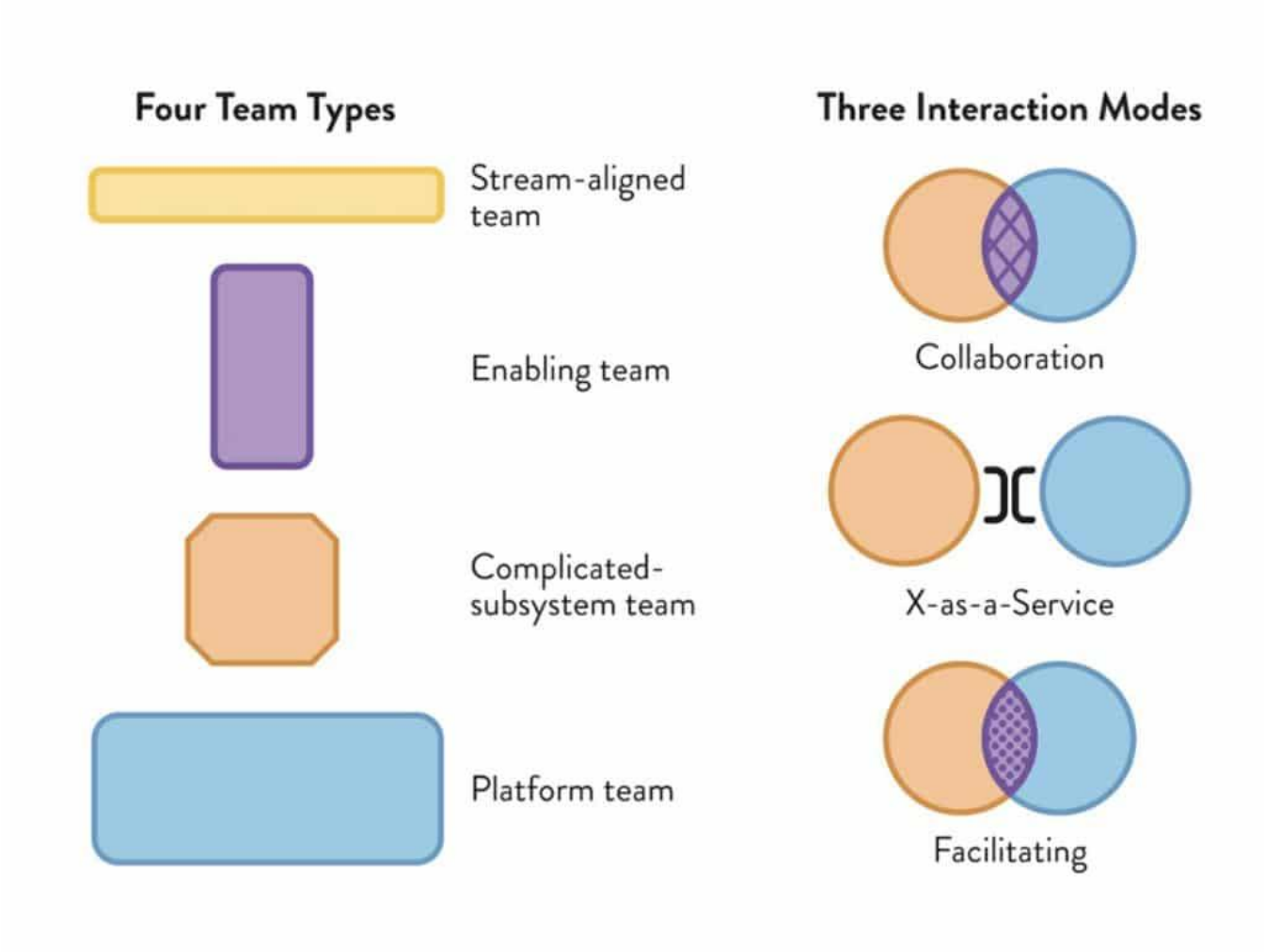
> **Evolve your team and organizational structure to promote your desired architecture**
>
> — ***ThoughtWorks Technology Radar***

Each team's responsibilities must be clear



Context Map Generator | Context Mapper

Team topologies



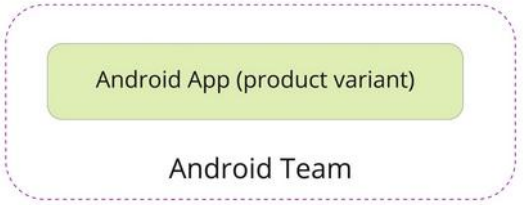
Matthew Skelton

Manuel Pais

Patterns for teams

Product Ownership Patterns

Product Variant Team



Product Area Team

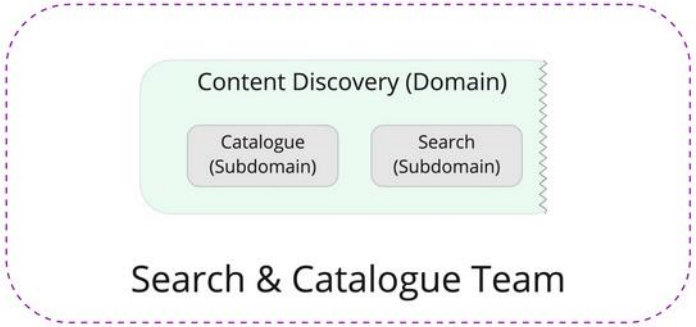
Multi Product Variant Team

User Journey Team

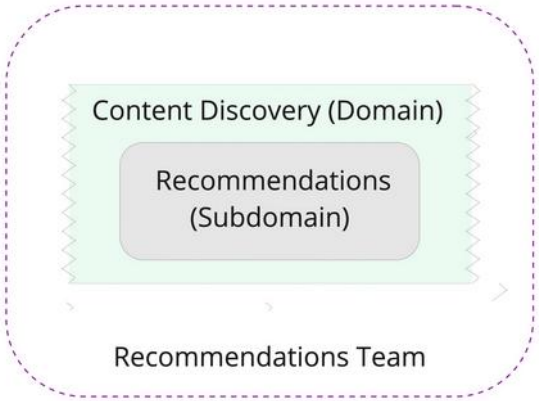
Domain Ownership Patterns

Domain Team

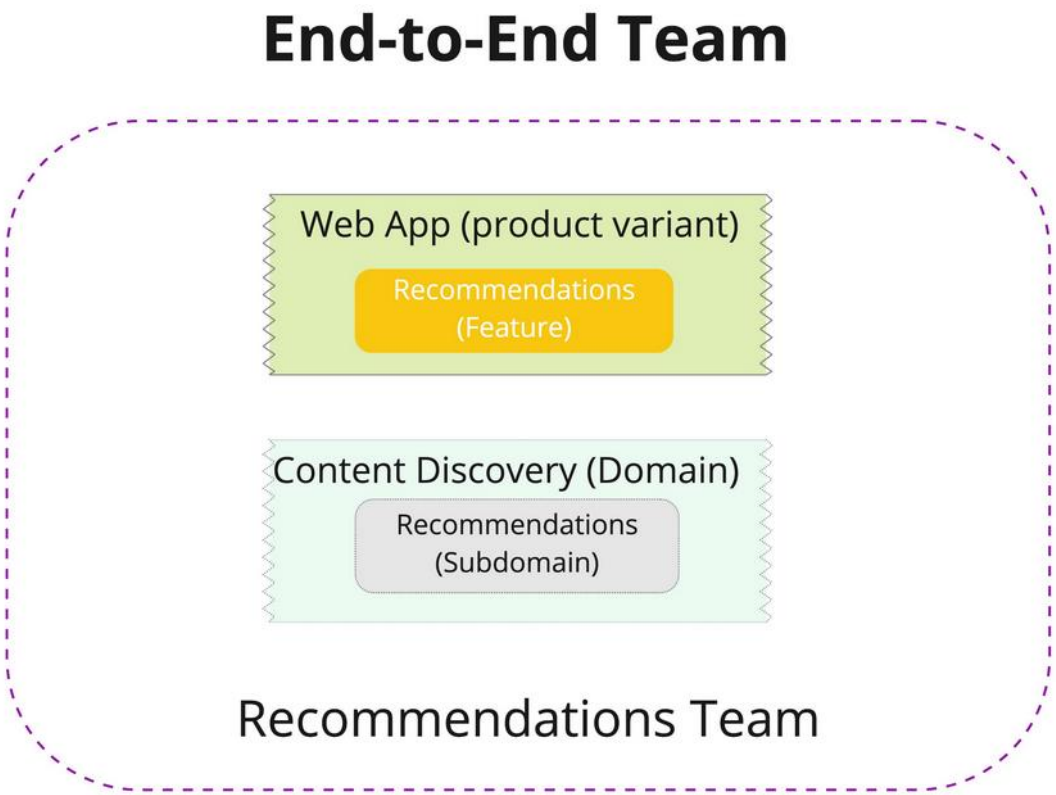
Subdomain Group Team



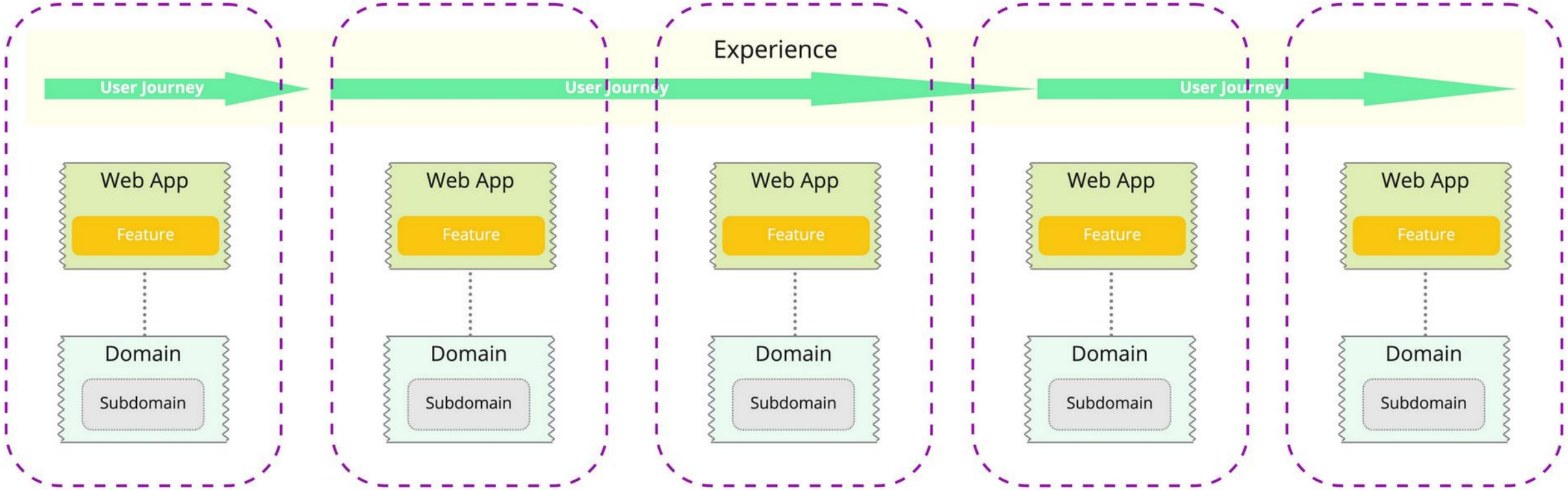
Single Subdomain Team



The Holy Grail?



Autonomous teams?



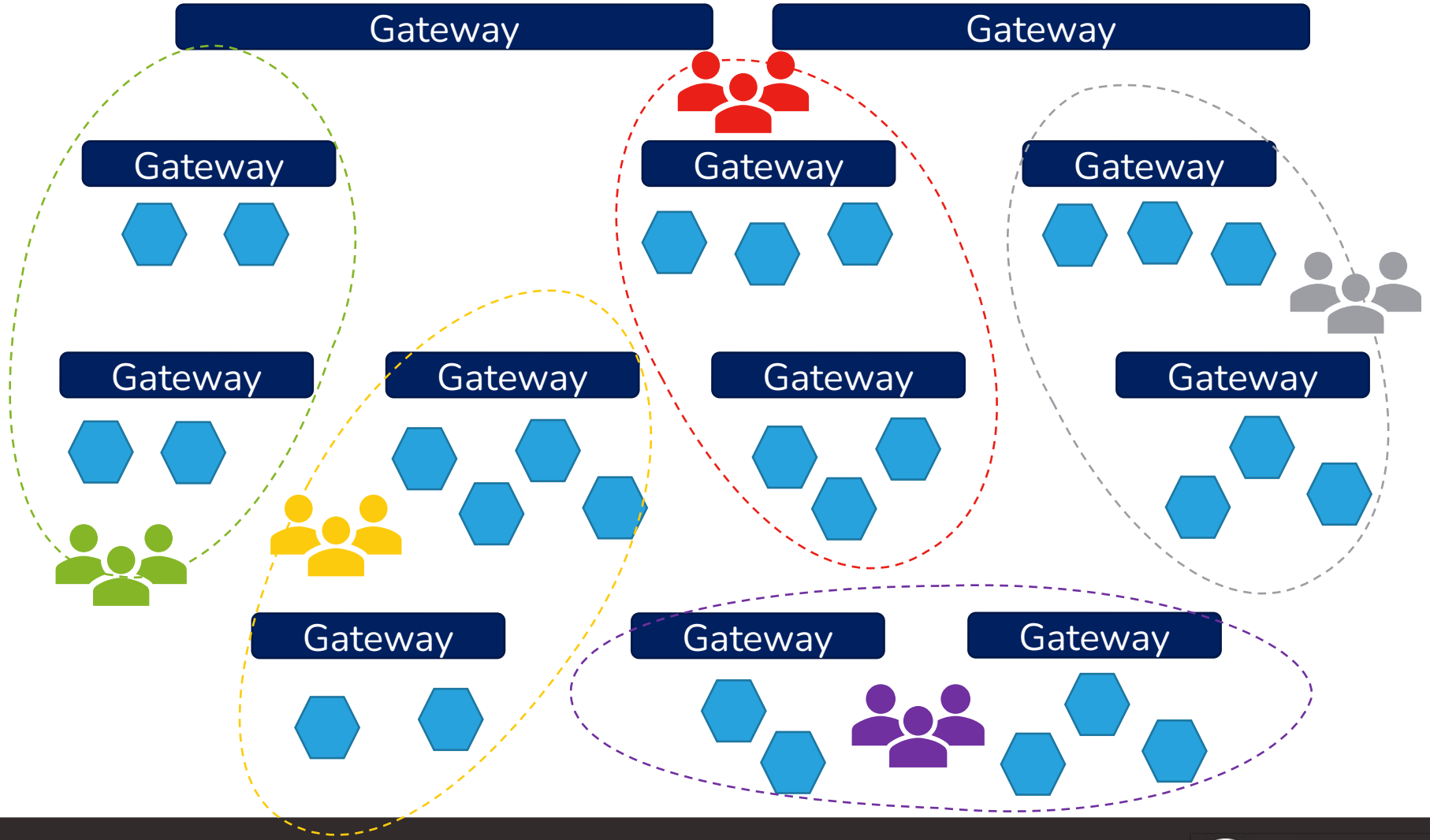
Domain-Oriented Microservices Architecture (DOMA)

Presentation layer

Product layer

Business layer

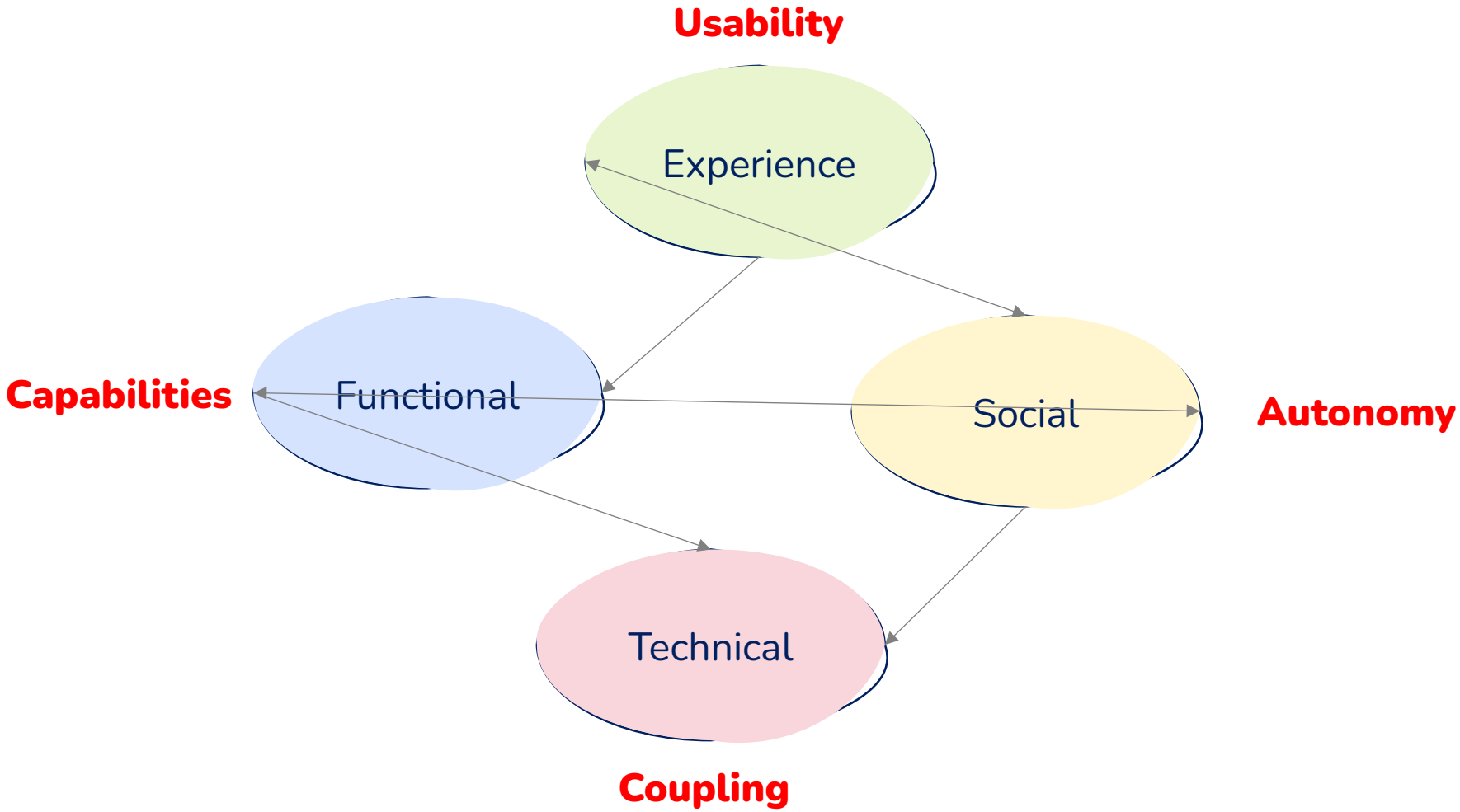
Infrastructure layer

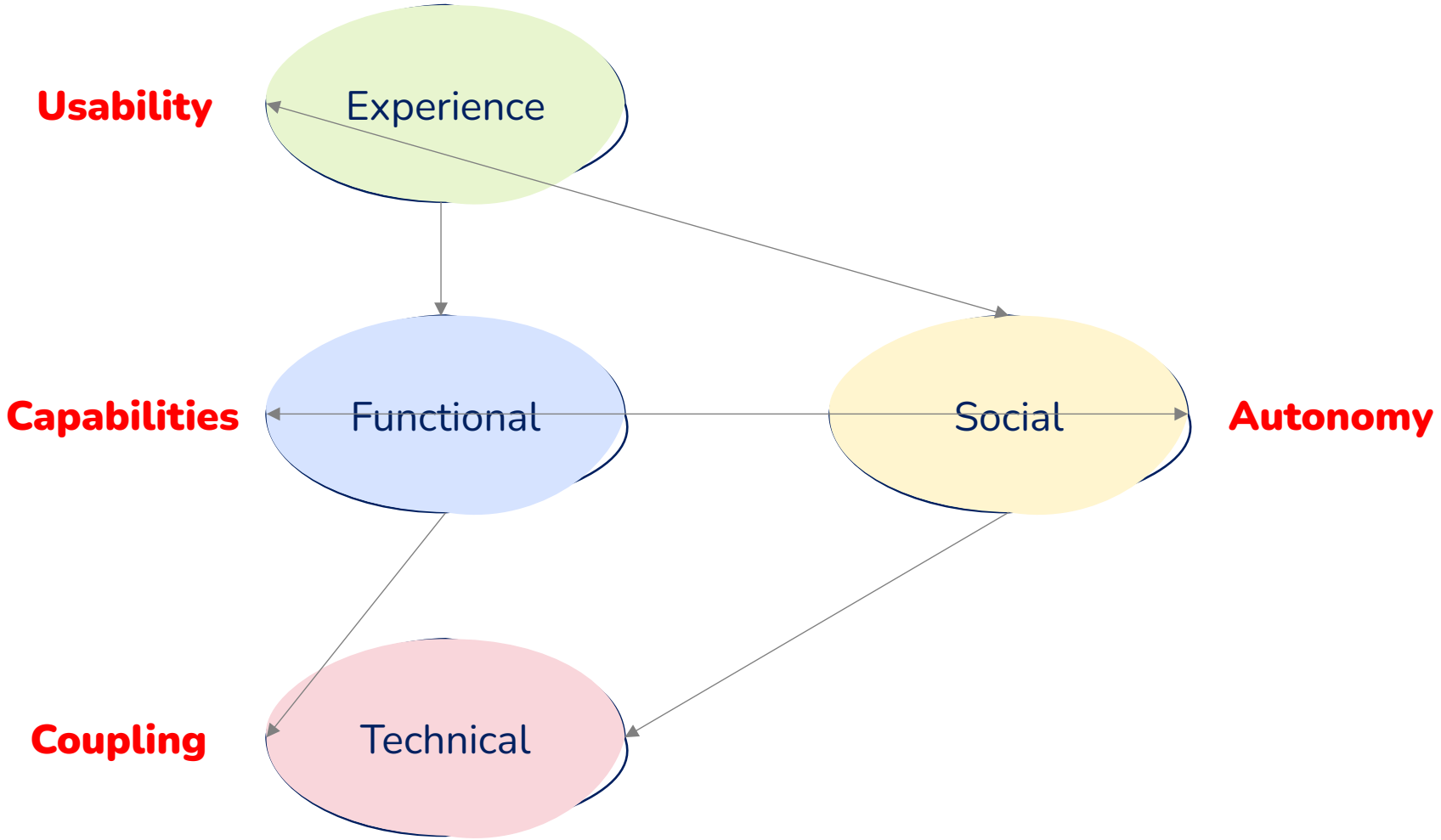


Introducing Domain-Oriented Microservice Architecture - Uber Engineering Blog

A new vision for boundaries

What have we learned?


Usability
Experience
Capabilities
Functional
Social
Autonomy
Technical
Coupling


Usability
Experience
Capabilities
Functional
Social
Autonomy
Coupling
Technical