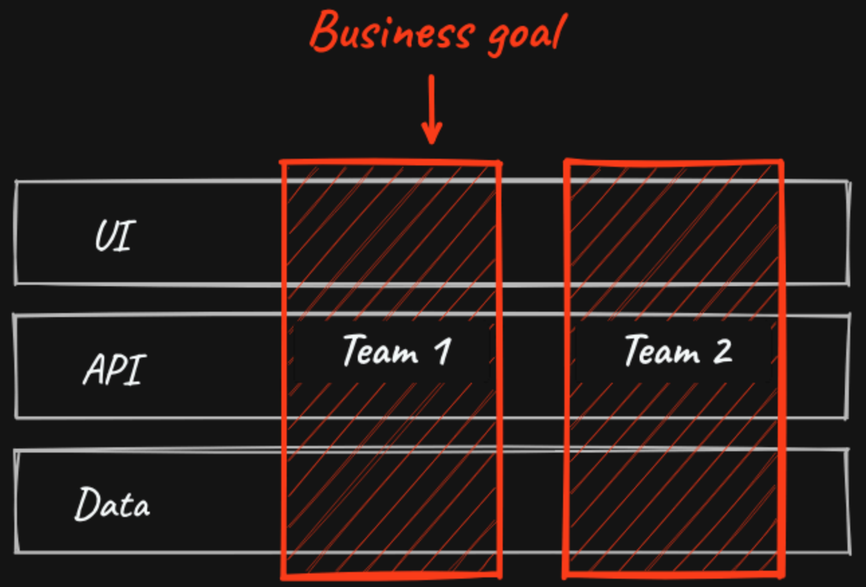
YaTalks · December 6, 2023

How to Form Team Structure for Business Demands

Five transformations from a project to end-to-end teams





The Goal Shapes Architecture and Team Boundaries

Core idea `Architecture ↔ teams`

- Start from the desired future
- Make the target architecture explicit
- Shape team boundaries around it
- Treat the transition as work of its own

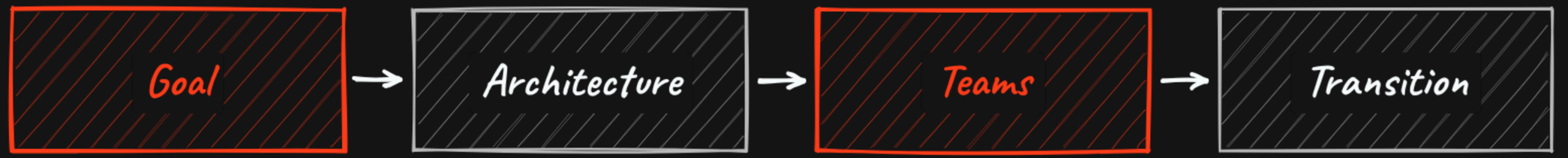
- Stabilize what is burning before redesigning the system

00

Frame

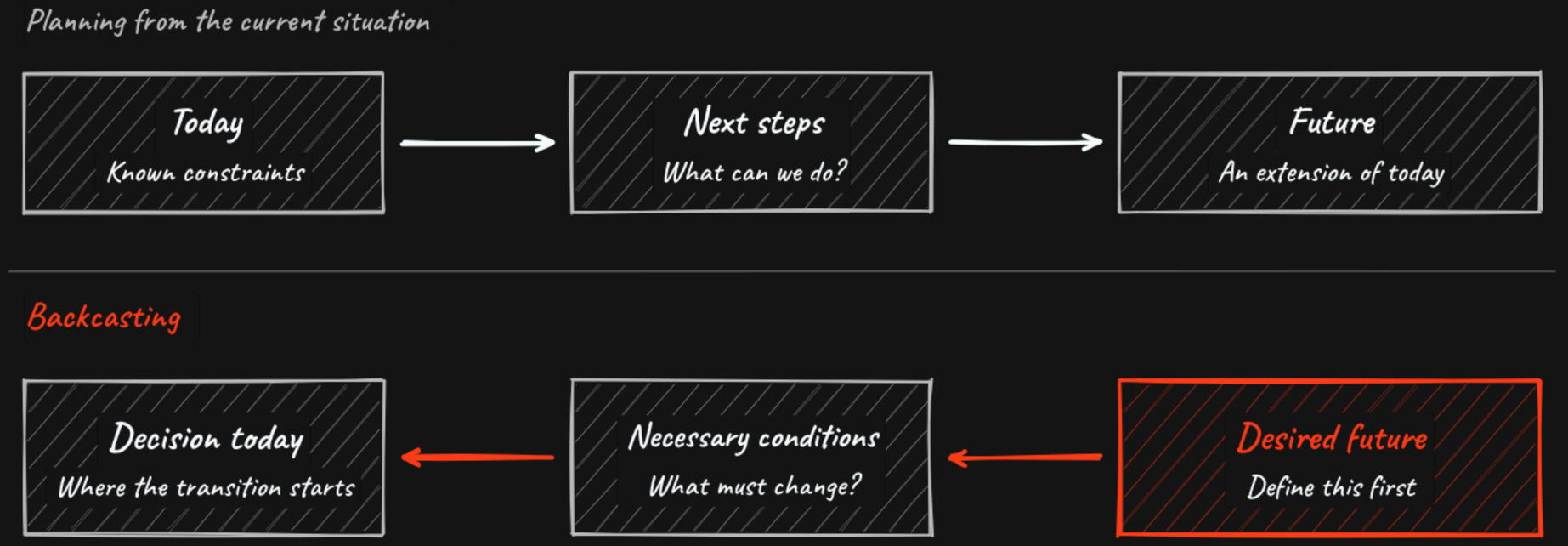
Tools and Algorithm

From the desired future to a managed transition



The Desired Future Shapes Today's Plan

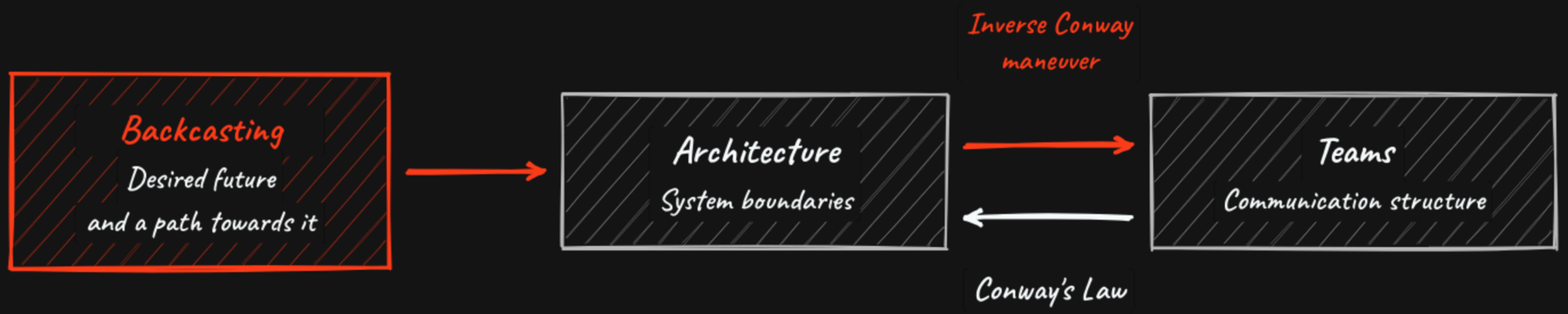
Planning | Backcasting



The goal defines a path back to today's decisions

Different Challenges Need Different Tools

Toolkit 6 approaches

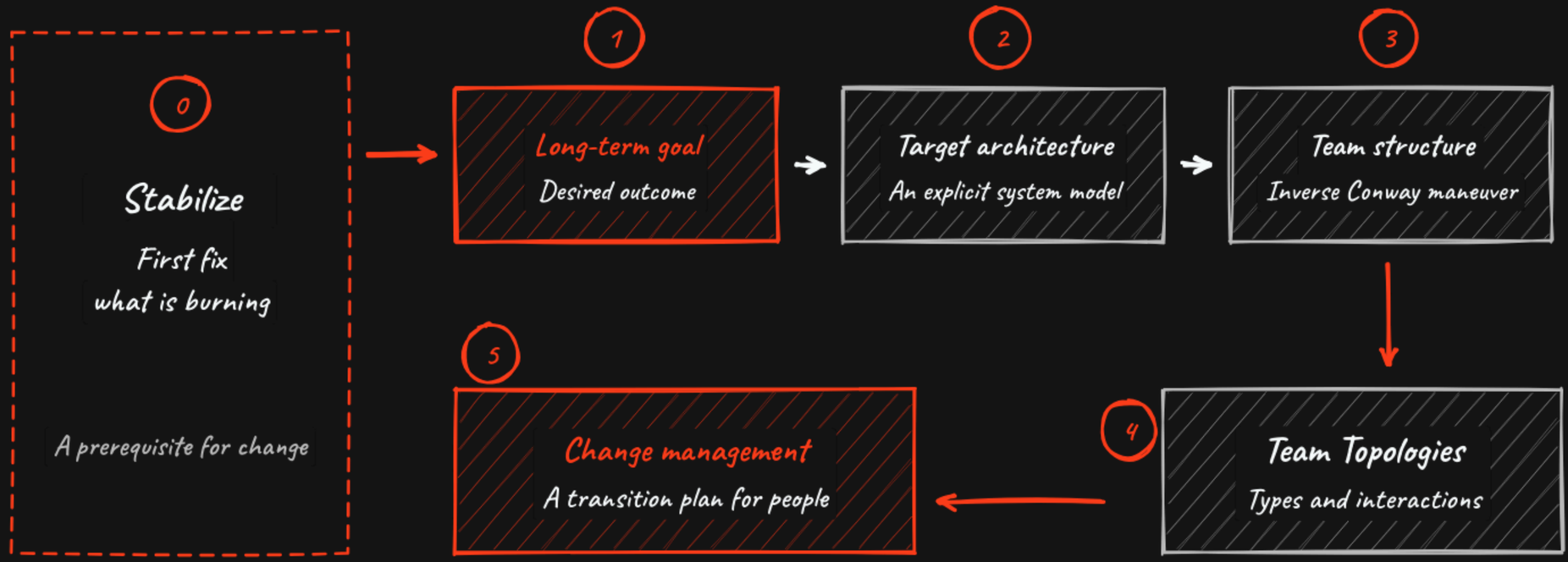


Tools for the transition



Stability First, Then a New Structure

Algorithm 0 + 5 steps



Structure follows the goal and architecture; transition takes deliberate work

01

Case 1

Project: Migrating Tinkoff.ru

Three months to fix critical acquisition problems



Four Problems Before the Migration

Situation — 3 months

3
months to migration

50+
people in the team

3
on acquisition

Problems on the path to launch

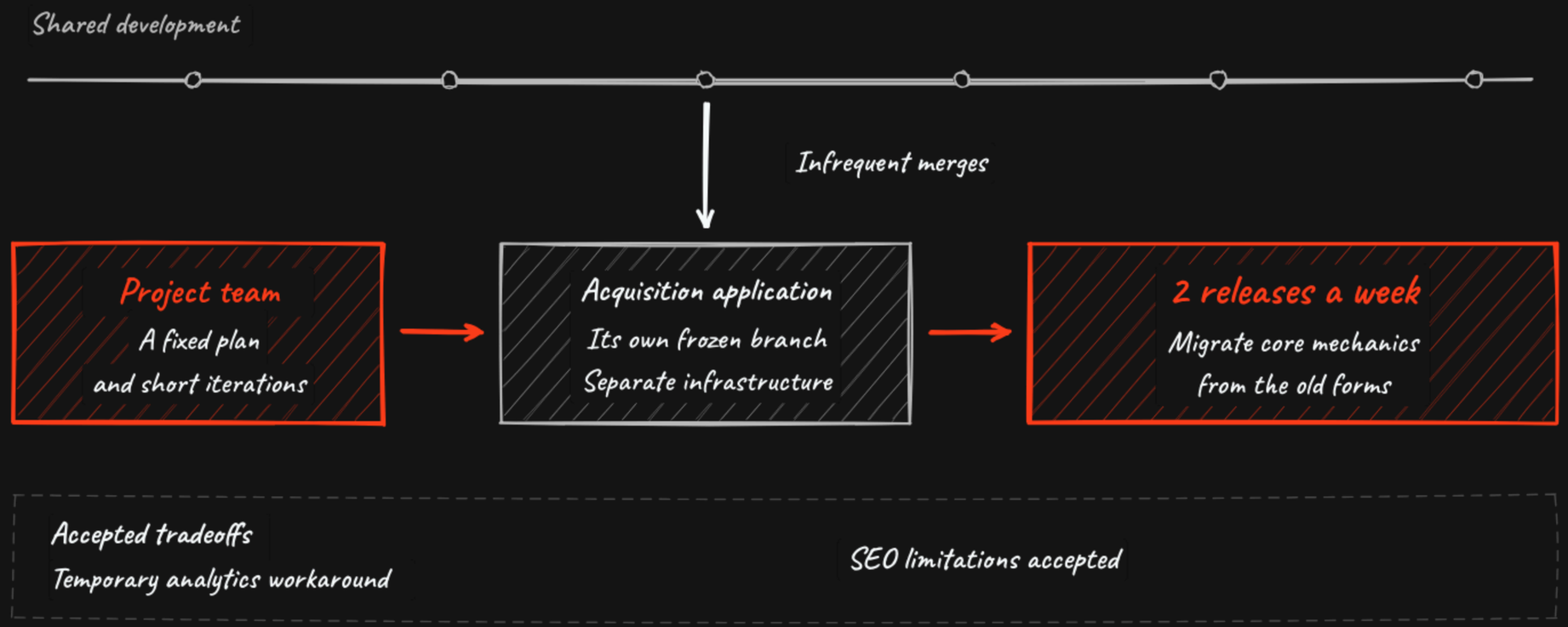
1. **SSR** — Not working ✗
2. **Analytics** — Not working ✗
3. **Acquisition mechanics** — Many were broken ✗
4. **Release** — About 3 weeks of manual regression ✗

Migration with these failures: hundreds of millions of rubles in extra costs

Additional spending on customer acquisition

Project Team: Two Releases a Week

Intervention Project mode



Forms Moved with Little Efficiency Loss

Outcome Project → product

- Core forms and pages moved on time
- Acquisition efficiency was largely preserved
- The team learned to resolve acute problems
- The resulting product was ready to scale

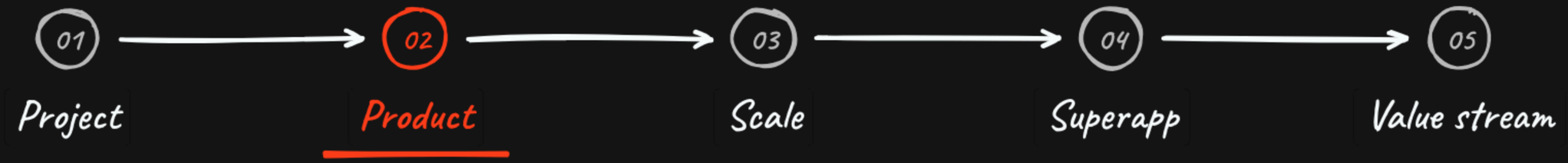
- After migration, the team could plan product development

02

Case 2

Product: Deliberate Development

Explicit ownership, APIs, and Kanban



Architecture Defines Team Responsibilities

Target system: Server-driven UI



Distinct flows reveal distinct areas of ownership

APIs and Pull Connect the Teams

Operating model

Ownership + API + Kanban

Ownership follows the parts of the system



Kanban

Visible work

Work is pulled as capacity permits



APIs connect teams; Kanban makes the flow manageable

Visible Work and Predictable Throughput

Outcome **Predictability**

- Teams understand their areas of ownership
- Work is visible to teams and customers
- Throughput became predictable
- The process gained a path for improvement

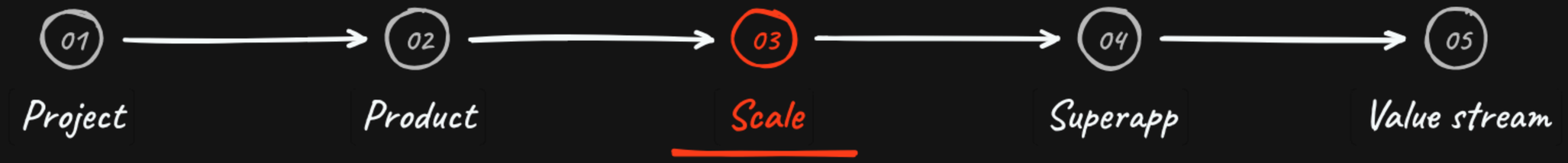
- The product teams were ready to scale

03

Case 3

Scale: Product and Platform

Feature-team autonomy without duplicating shared solutions



Team Types Combine with Interaction Modes

Team Topologies · 4 types · 3 modes

Team type

- Stream-aligned team
- Enabling team
- Complicated-subsystem team
- Platform team

Interaction mode



Collaboration
Shared discovery and delivery



X-as-a-Service
Consume an available service



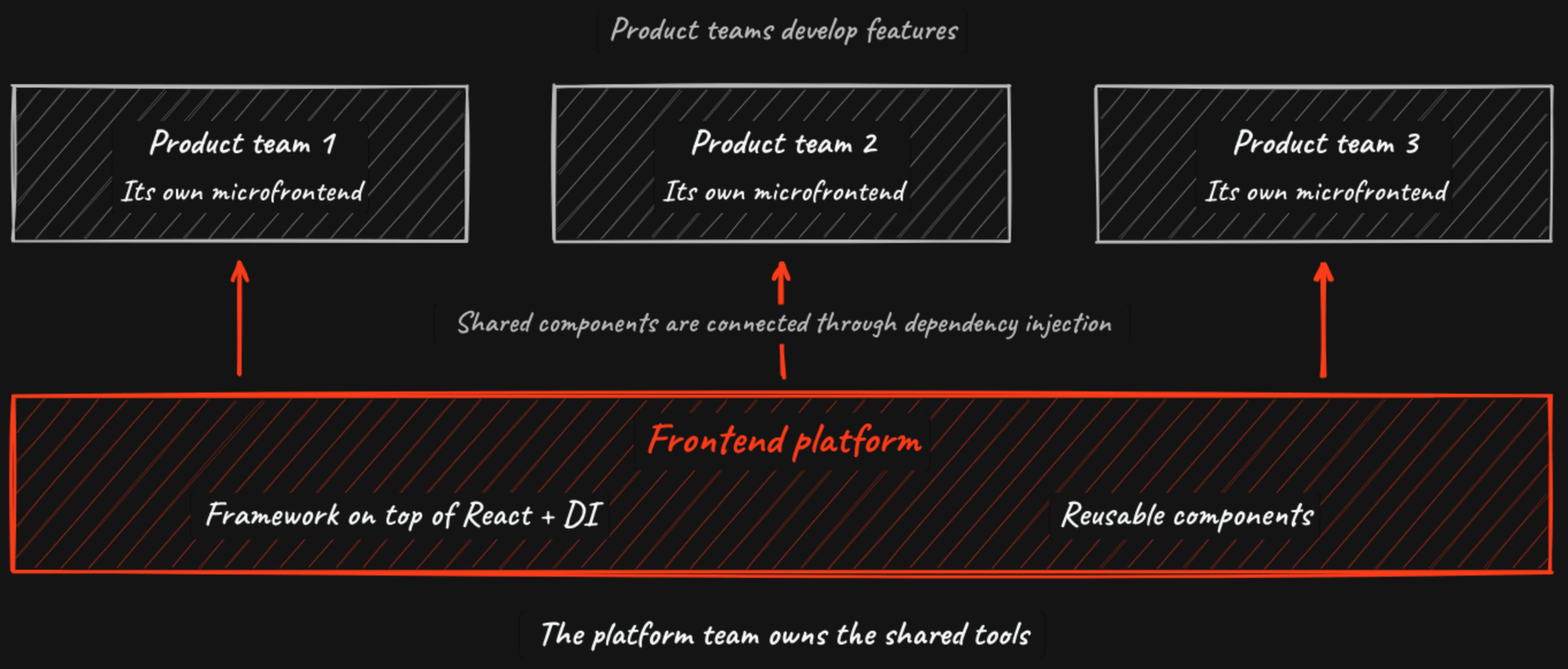
Facilitation
Help a team develop capability

Choosing a team type and an interaction mode are separate decisions

A Shared Platform for Autonomous Teams

Frontend platform

Autonomy + standards



Products Own Features, Platforms Own Shared Tools

Outcome Shared tools

- Product teams ship features
- The platform team develops shared tools
- Shared React components connect through DI
- Teams moved to microfrontends

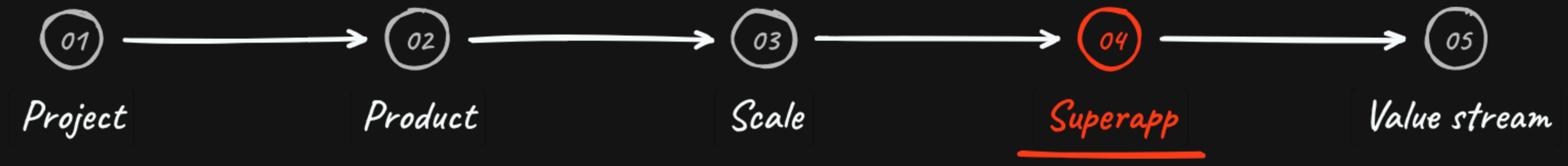
- The teams develop the unauthenticated part of ↗tinkoff.ru

04

Case 4

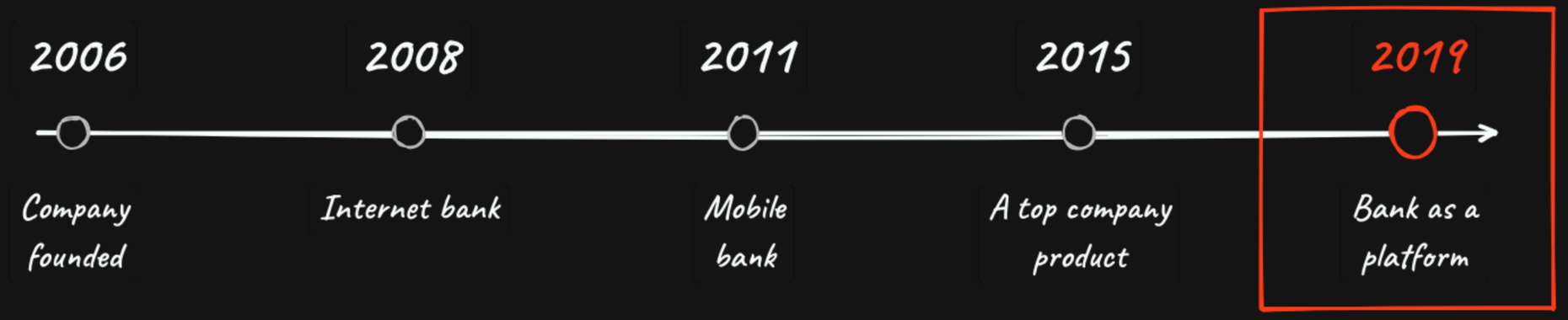
Mobile Bank as a Platform

Autonomous business domains and platform teams in one application



The Mobile Bank Becomes a Platform

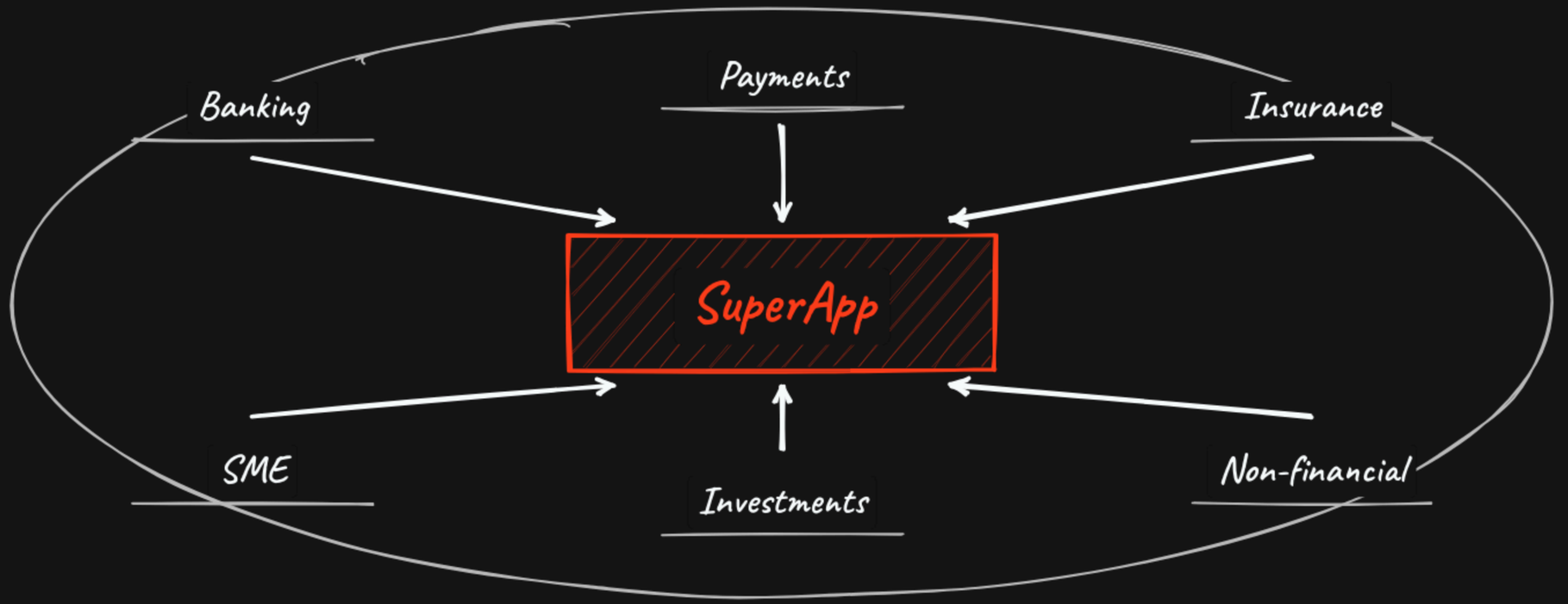
Product evolution 2006 → 2019



Product growth changes the teams' task

One App Connects Different Businesses

SuperApp Not a monolith

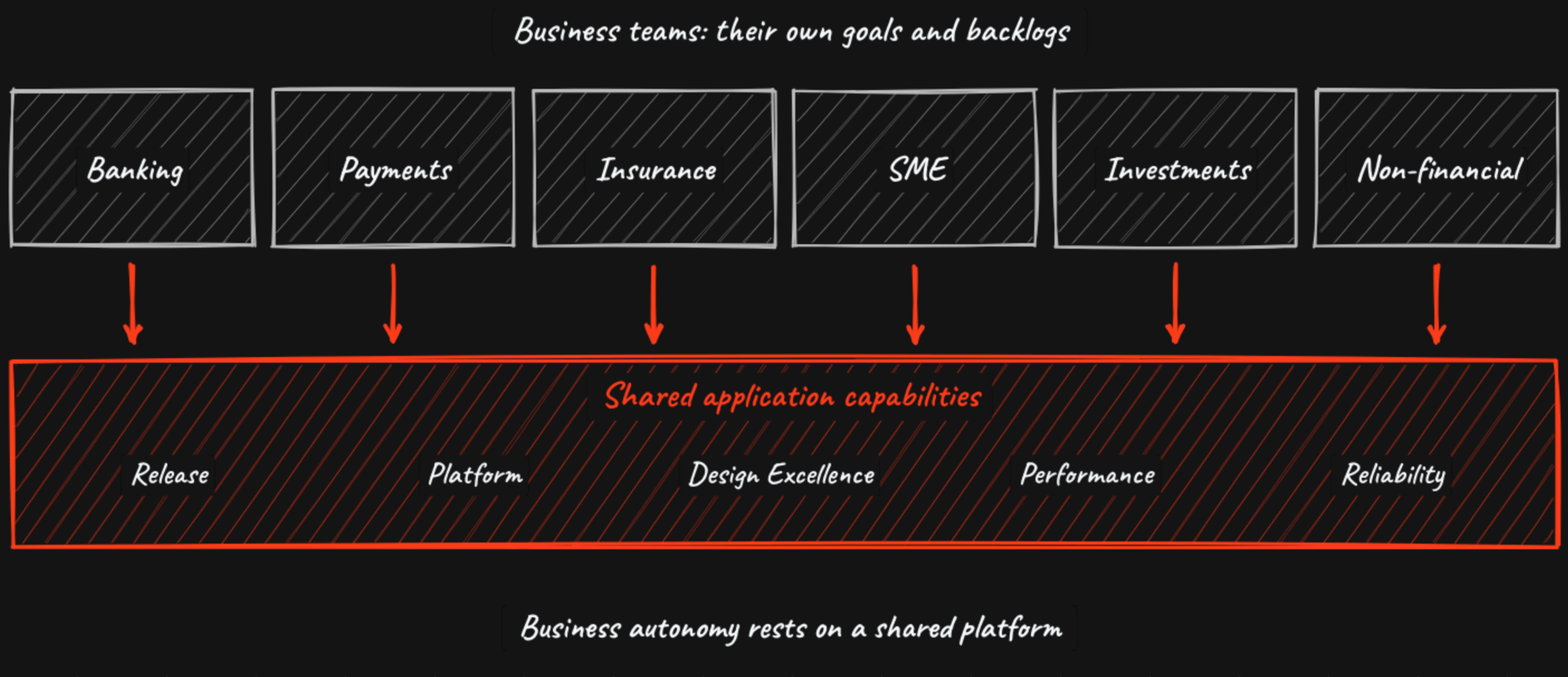


One interface, businesses developing in parallel

Product and Platform Teams Split Responsibilities

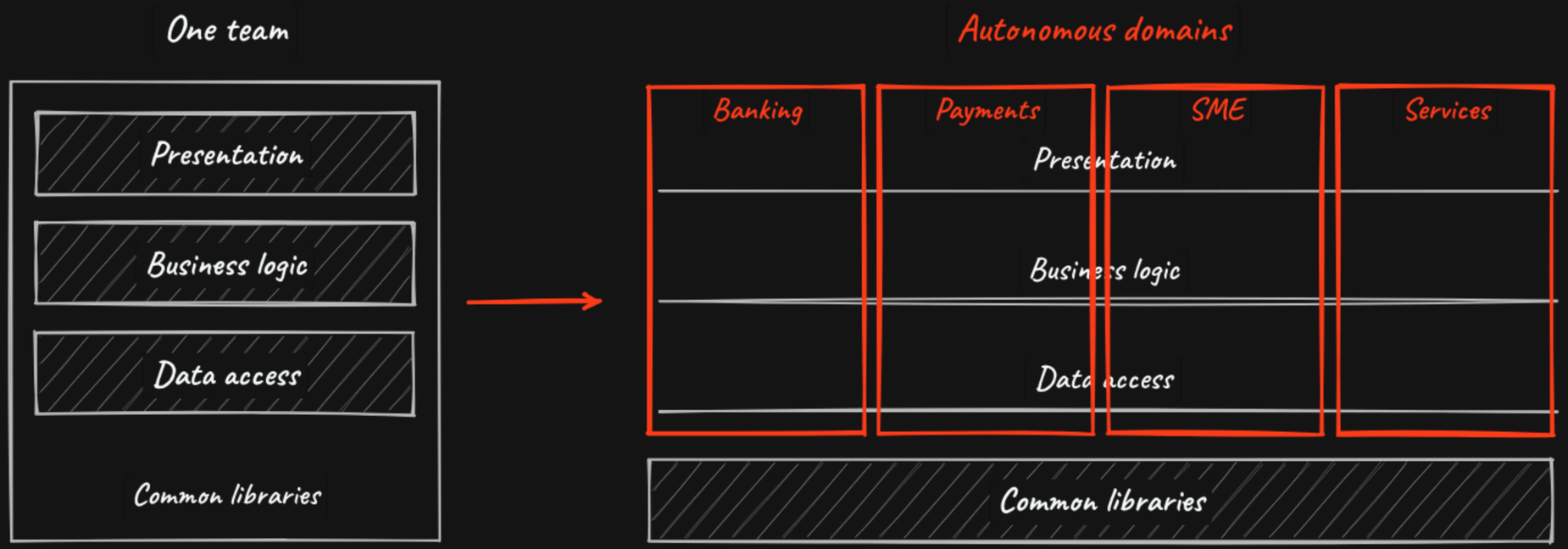
Inverse Conway

Business + platform



Shared Architecture Supports Autonomous Domains

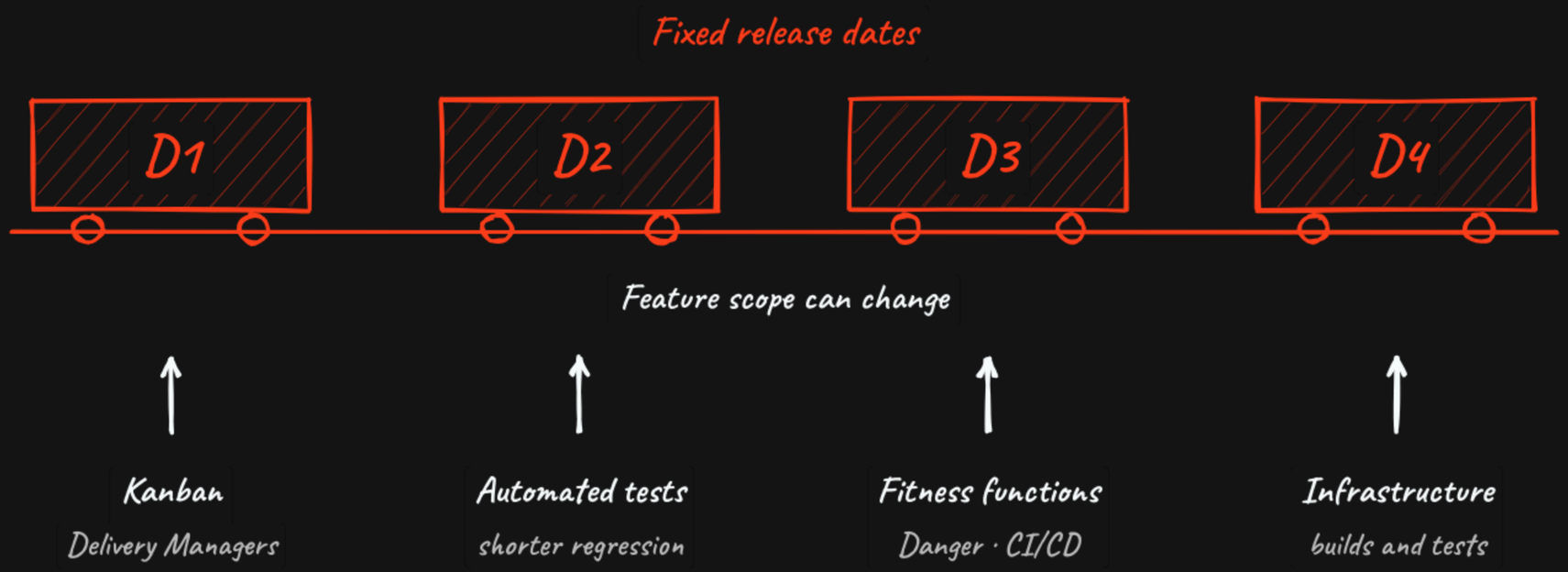
Architecture | Layers → domains



Ownership boundaries first, within the shared application

Autonomous Teams, Scheduled Releases

Delivery | Release rhythm



Autonomous teams share a delivery rhythm

Each Business Has Its Own Mobile Team

Outcome **Domain autonomy**

- Business verticals gained teams and backlogs
- Platforms strengthened modularity, SRE, performance, and CI/CD
- Release frequency increased several times
- Teams own process improvement themselves

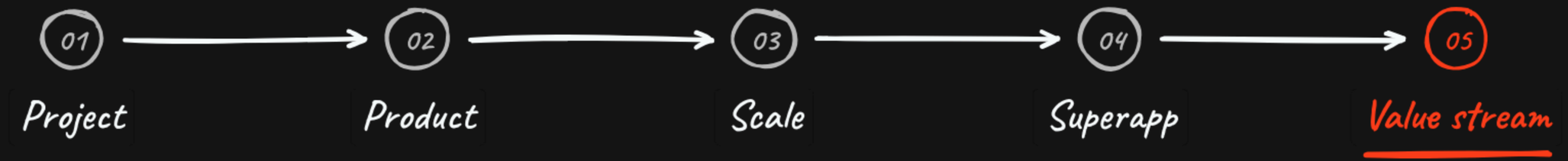
- Autonomy works when shared qualities have owners

05

Case 5

Teams Across the Value Stream

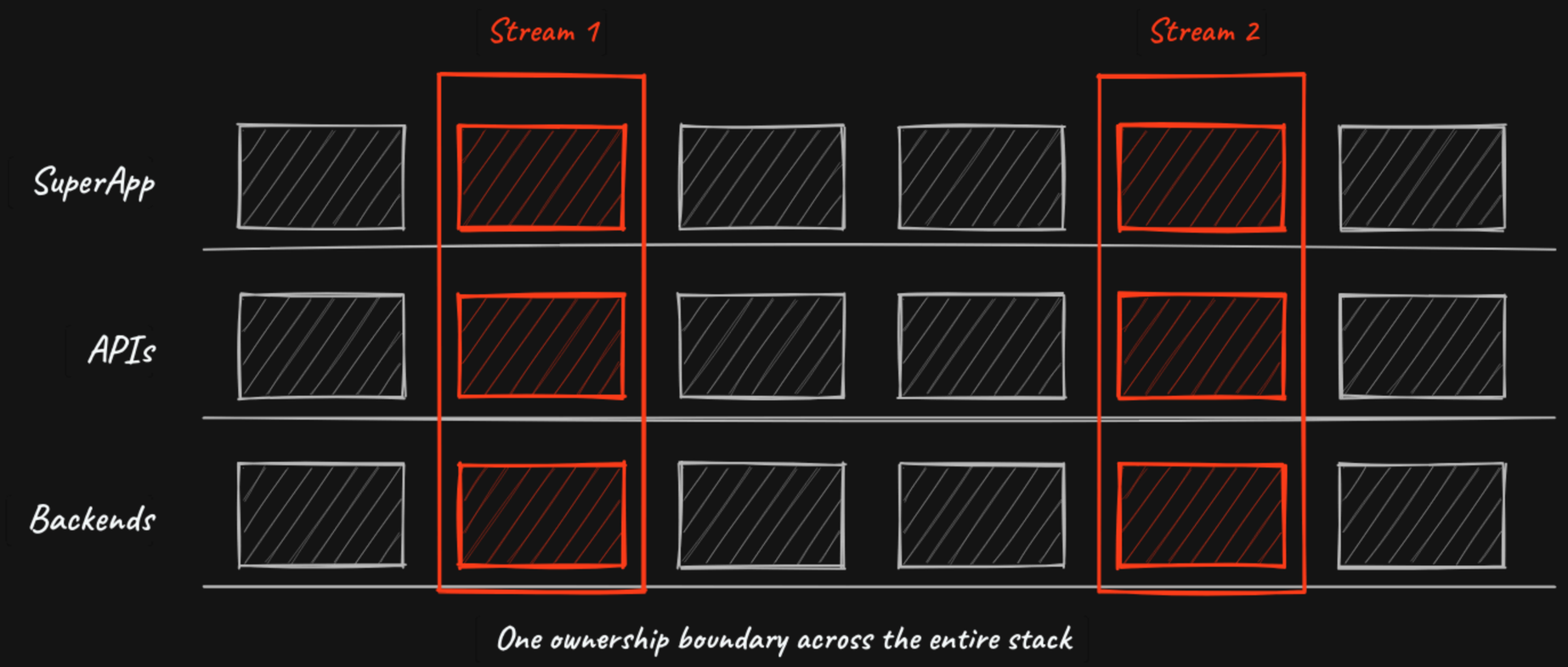
Reduce time to market and cognitive load, improve end-to-end architecture



One Team Spans All Technology Layers

New boundary

SuperApp → API → Backends



One Team Owns the Entire Stack

Team composition

Single ownership



Product, technology and delivery belong to one team

Team Boundaries Reshape End-to-End Work

End-to-end work 3 concerns

Design

CTO

↓

User story

Requirements across the stack

Process

Delivery Manager

↓

End-to-end flow

TTM · lead time · cycle time

upstream + downstream

Architecture

APIs: assemble the data

↓

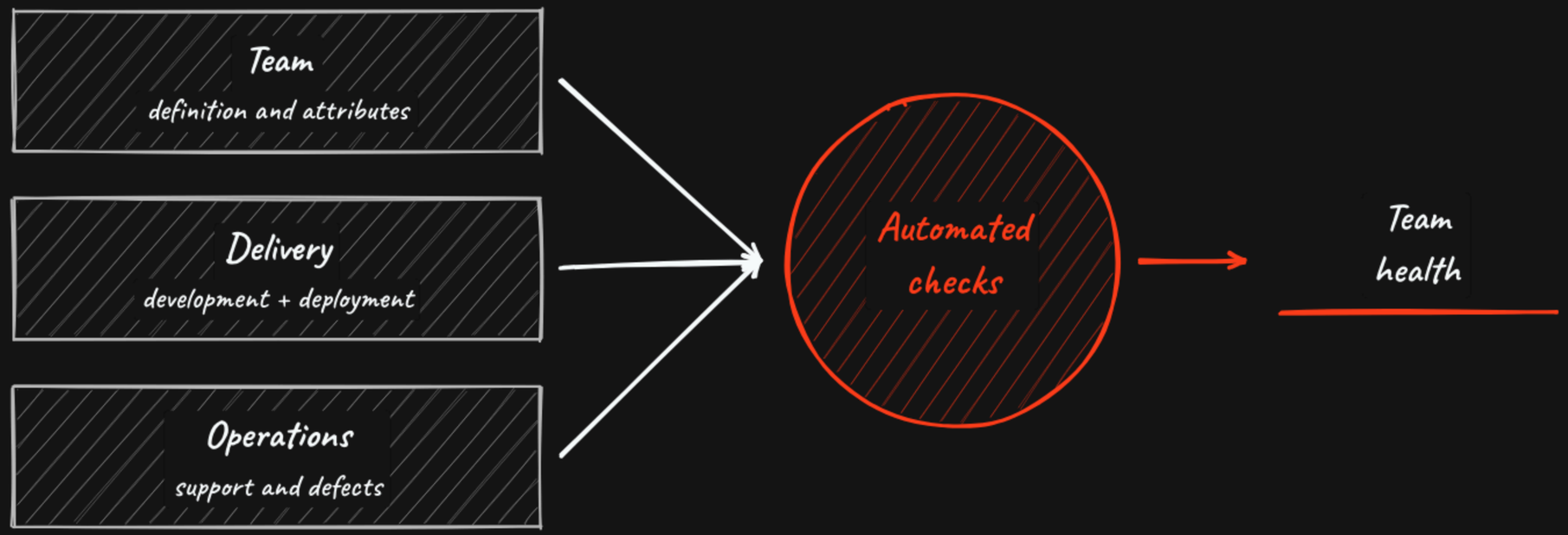
BFF

Event-based read models

transition under consideration

Shared Rules Are Checked Automatically

Technical governance | Rules as code



The platform sets verifiable rules for autonomous teams

Metrics Show Platform Impact on Teams

Platform leadership

3 metric groups

Impact of platform teams and process changes



Different Challenges Require Different Team Structures

Five cases

Signal → boundary → outcome

Context	Demand	Structure	Enables
Project	Deadline and fire	Project team	Migration
Product	Flow of work	Ownership + Kanban	Predictability
Scale	Many teams	Product + platform	Shared tools
SuperApp	Different businesses	Autonomous domains	Parallel development
Value stream	Layer handoffs	End-to-end team	End-to-end delivery

Conclusion

Architecture and organization

Teams Follow the Target Architecture

- Start from the long-term business goal
- Derive boundaries from the target architecture
- Choose team types and interaction modes
- Automate shared rules and measure outcomes

- Changing structure takes deliberate work with people

What the Model Builds On

Sources Methods and material

- Author PDF for YaTalks 2023 · 172 pages
- Backcasting, Kanban and the inverse Conway maneuver
- Team Topologies and change management
- Five cases from ↗ tinkoff.ru and the Superapp

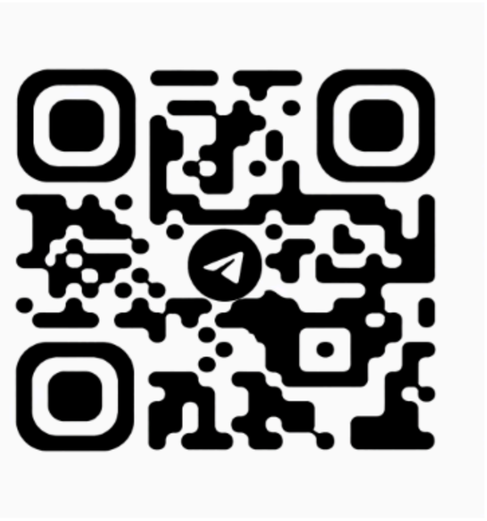
Knizhny kub

Thank you!

Continue the discussion: ↗ t.me

Alexander Polomodov

Technical Director,
Tinkoff



@book_cube