# Practical Modern Enterprise Architecture at Saxo Bank

Simon Rohrer

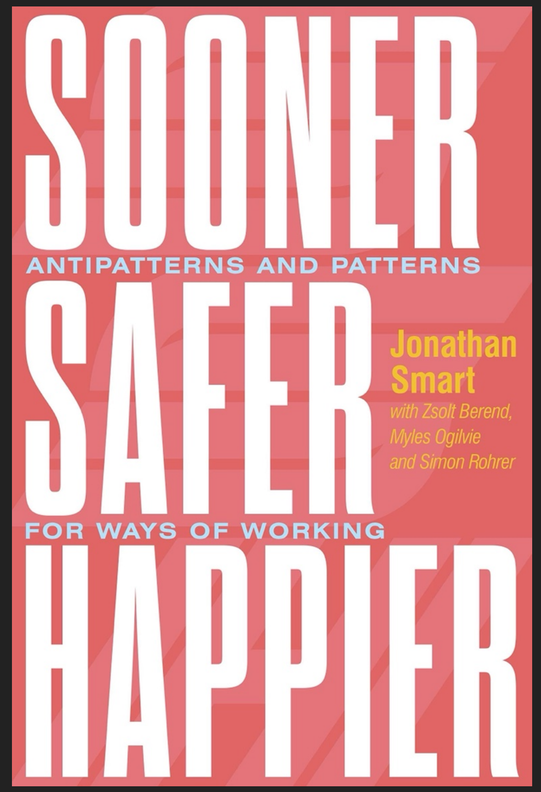
# My journey

Developer

Enterprise architect

Ways of working lead

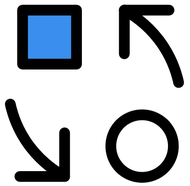
Head of tech arch & WoW



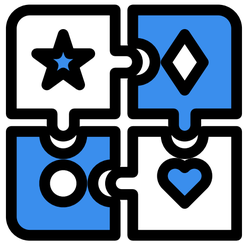
# Modern Enterprise

Teams-of-Teams-of-Teams: > 150 Employees

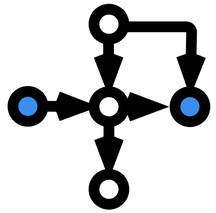
Interact with Your Customers Digitally



Change is a Constant



Heterogeneous Environment: Old/New, Big/Small, Slow/Fast, Monolith/Distributed Vendors/Internal



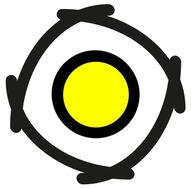
Systems of Systems of Systems

# Modern Enterprise Architecture

**A**

**A**ligning Value, People and Technology

- Conway’s law ++ and reverse Conway manoeuvre
- Removing complexity and increasing autonomy
- Enterprise architecture as politics and diplomacy



**B**

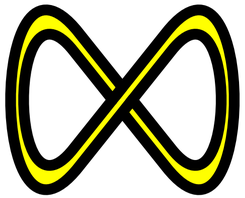
Architecting for Outcomes: **B**etter Value, Sooner, Safer, Happier

- Outcomes of quality, flow, safety to balance pure “value”
- Happier customers, colleagues, citizens & climate
- Continuous improvement - be the best at being better

**C**

Governing **C**ontinuously: Conversational and Automated

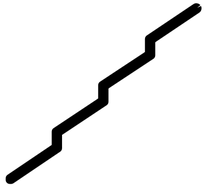
- A stream of continuous architecture decisions
- Moving decisions down into the right level of the org
- Automating common top-level system policies



**D**

Practicing **D**evOps at Enterprise Scale

- Learning by observing: intention & reflection
- Humility in architecture: all decisions are contingent



**E**

Curating the **E**volution of Enterprise Architecture and Organisation

- Fitness functions for socio-technical architecture at scale
- Evolving fast and slow


HUNTERS' COVE
SAXO
BE INVESTED

# Saxo is an industry pioneer with over 25 years of experience in FinTech

**1992 – 1998:**
Launch of brokerage

- Kim Fournais founded Saxo Bank – with €70,000 and one employee
- Creation of a platform for growth

**1998 – 2008:**
Becoming a FinTech

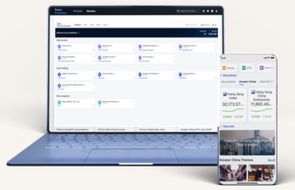
- Launch of the first online platforms
- Expansion of products
- Establishment of international presence

**2008 – 2017:**
Period of change

- Launch of SaxoTraderGO
- Focus on Trader segment
- Saxo Bank adapts to a rapidly changing world by becoming more digital and reducing business model risk
- Reduced number of local offices to streamline business, some sold to partners

**2017 – 2020:**
Redefining the new Saxo Bank

- Launch of SaxoInvestor and SaxoTraderPRO
- Deployment of a more Investor and Saxo Institutional focused strategy
- Gained scale with BinckBank acquisition
- Sale of Saxo Bank Privatbank and Banking Circle
- Set-up of Saxo FinTech, a joint venture with Geely Group



**2020 – 2023:**
Scaling Saxo Bank

- Launch of SaxoPartnerConnect
- New purpose - ‘We Get Curious People Invested in the World’
- Emphasis on Digital, Platform First engagement model
- Migration of BinckBank clients
- Saxo crosses 1 million clients and €95 billion in client assets
- Divestment of Saxo FinTech to Geely Group

SAXO
BE INVESTED

**The Future of Saxo**
Phase of constant improvement

- Saxo lowered prices in 2024 to be market leading
- Breakdown of silos and new organsiational structure focused on client centricity
- Focus on rolling out new products and services globally
- All Saxo Bank clients serviced from on one platform
- Strengthened focus on People and Culture



SAXO
BE INVESTED

# Saxo is a global, multi-asset facilitator

We deliver capital markets access, products, content and services through our platforms – **The Saxo Experience**
We run and develop **one global, multi-asset, multi-tenanted tech stack** and one set of global business processes

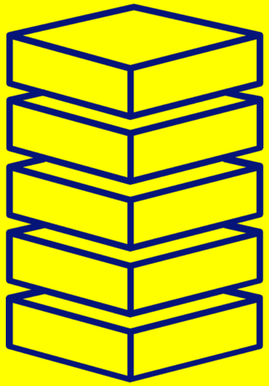
## Capital market products, services and liquidity

We unbundle the value chain through our open architecture

We source the best ideas, products, liquidity and services from the best providers

## Saxo facilitation

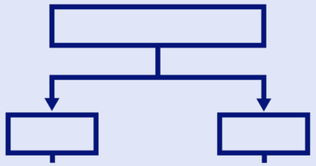
**Tech Stack**
Trading platforms
CRM
OMS / EMS
FIX API / Broker connectivity
Hosting services



Clearing and settlement
Client account structures
Margin & risk management
Market data connectivity
Custody
Regulatory reporting

*Integration Points*
EOD files, FSSO & TENS
Open API

**Processes**
Business management
Client management
Integration

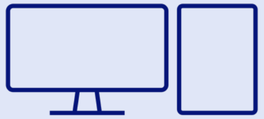


Execution and trading
Market data
Custody and post trade
Reporting
Asset Management

## Distribution to clients and partners

**Traders**
SaxoTraderPRO and SaxoTraderGO for self-directed traders

**Investors**
SaxoInvestor for self-directed and delegating investors



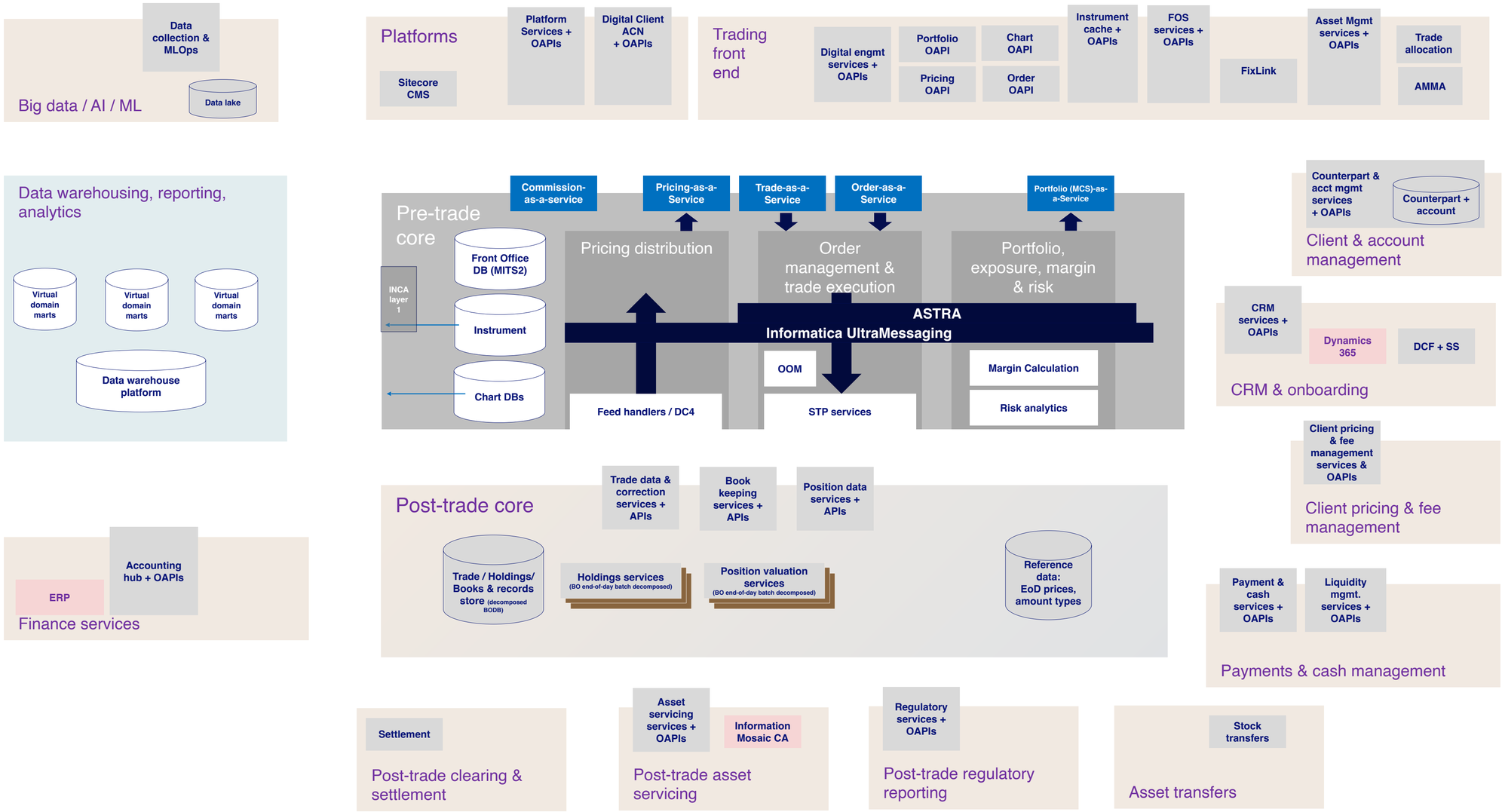
**Wholesale and Direct Institutional**
Outsourced capital markets infrastructure and client facing front ends incl. Open and Fix API

We distribute capital market and asset management products and services through our platforms tied together by the Saxo Experience



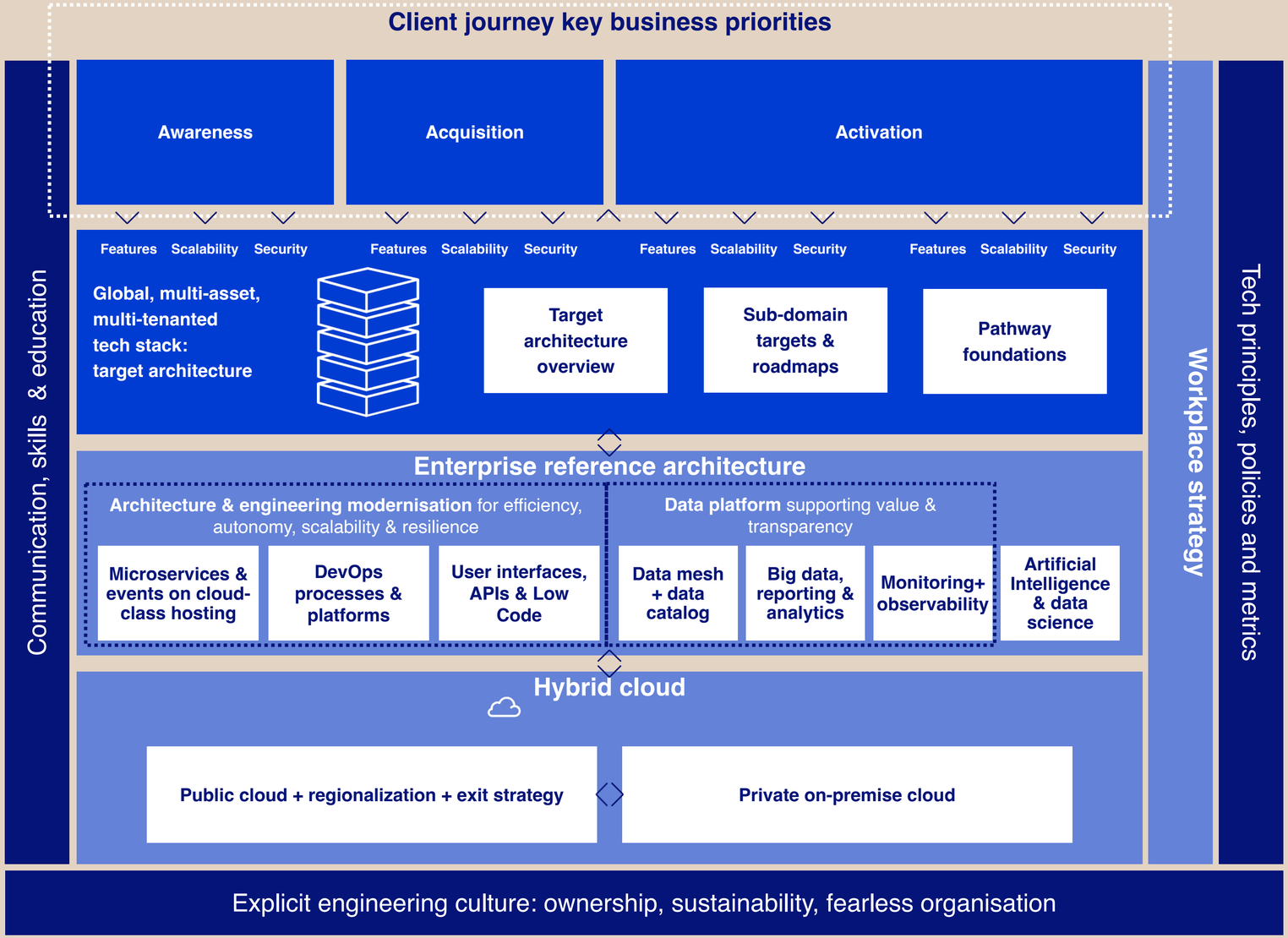
SAXO
BE INVESTED

# Our single global, multi-asset multi-tenanted tech stack is a system-of-systems-of-systems that we continue to develop and evolve




SAXO
BE INVESTED

# Saxo are executing on an explicit Technology Strategy to support our Business Strategy and to sustain and evolve our global tech stack

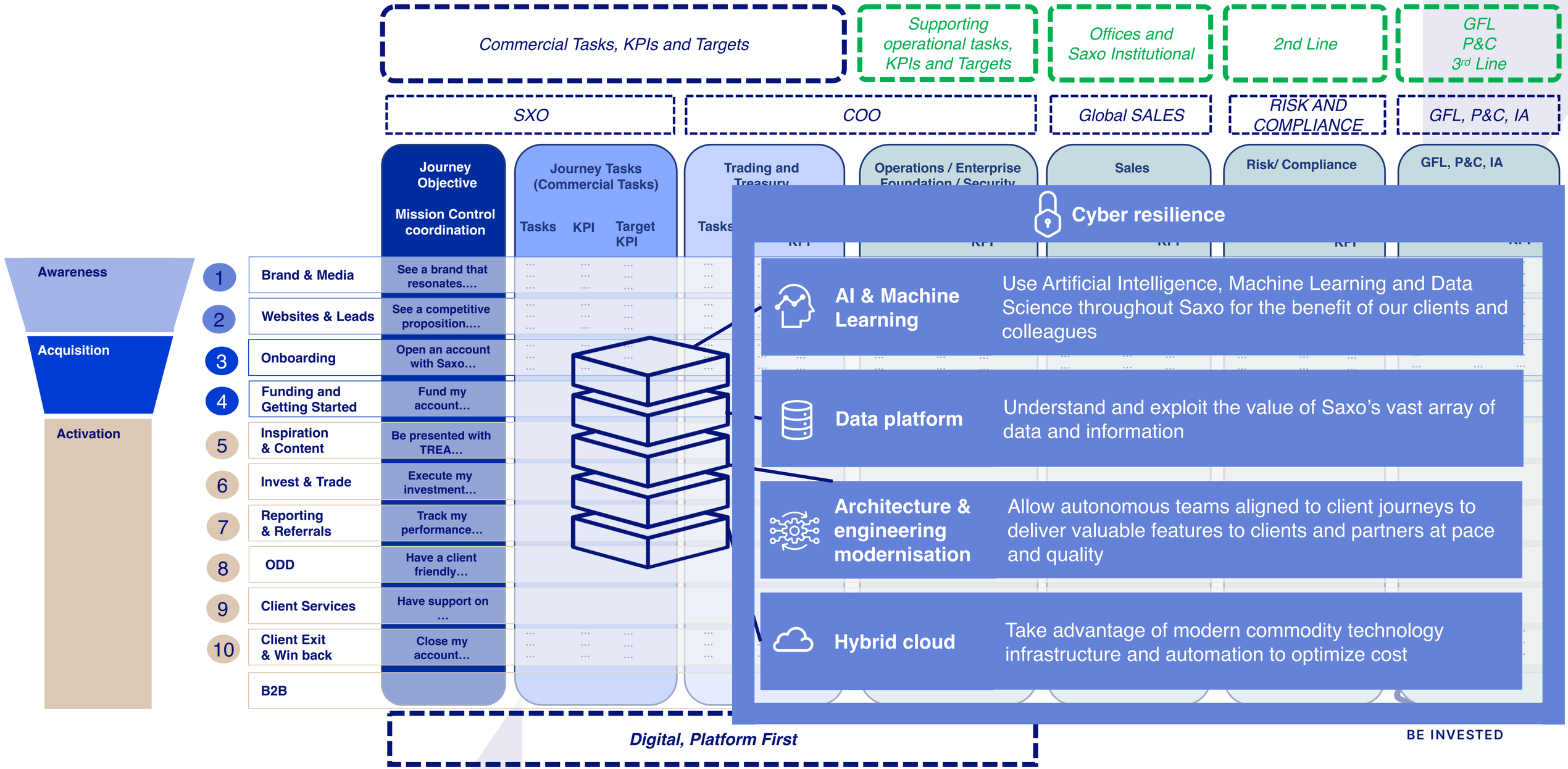



- We have a **hybrid cloud hosting strategy to rationally exploit** both private and public cloud capabilities
- We continue to modernise our application technology stack to a **microservices and event-driven architecture** to support efficient autonomous teams & a scalable and resilient platform
- We have invested in a **data platform** to support extracting value from the significant data sources we collect
- The data platform serves as a solid foundation for exploiting **artificial intelligence**

- Our technology strategy supports us in continuously **improving all our client journeys**: improving cyber security, stability, scalability, reducing cost and complexity, and reducing time to market

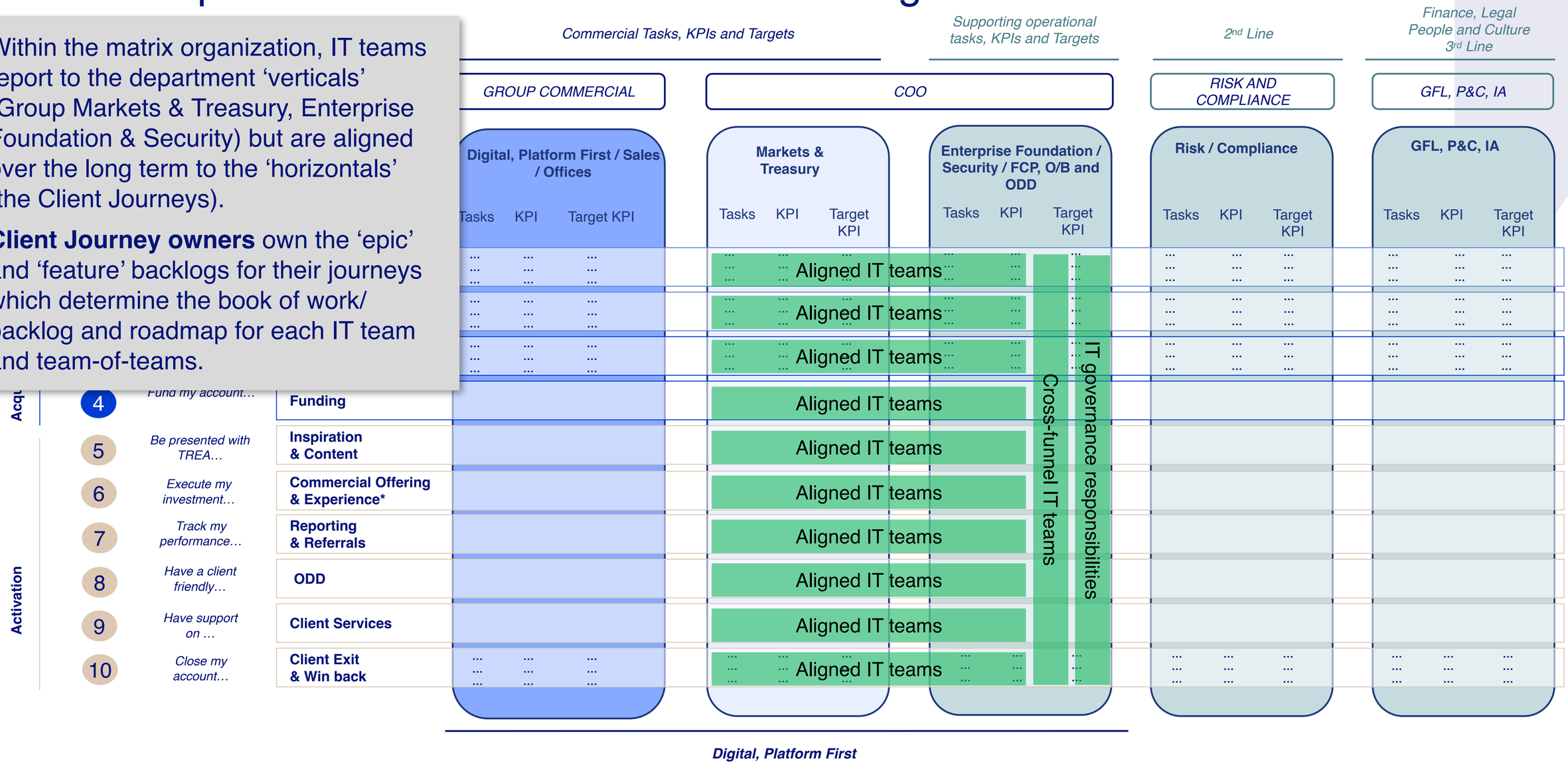
SAXO
BE INVESTED

# Our technology strategy exists to support our business strategy



# Roles & responsibilities within the Saxo matrix organisation

- Within the matrix organization, IT teams report to the department ‘verticals’ (Group Markets & Treasury, Enterprise Foundation & Security) but are aligned over the long term to the ‘horizontals’ (the Client Journeys).
- **Client Journey owners** own the ‘epic’ and ‘feature’ backlogs for their journeys which determine the book of work/ backlog and roadmap for each IT team and team-of-teams.

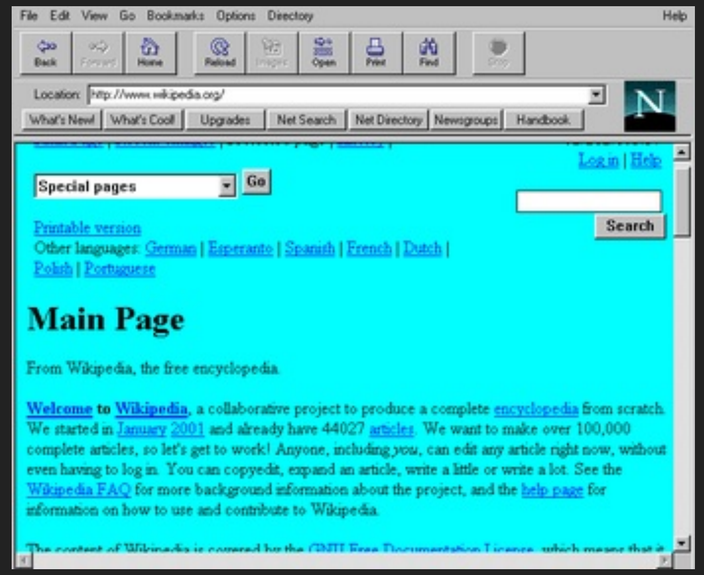



* Includes Products, Services, Price and Direct and Institutional Platforms

# Where are we?
# How did we get here?

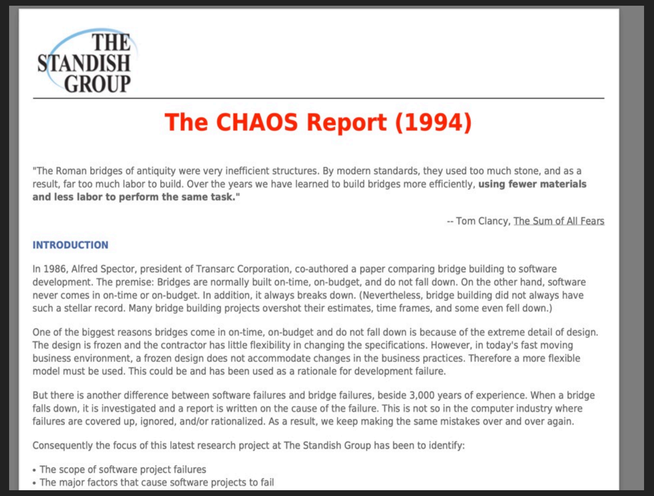
# 1990s

1994




1999

1994



THE STANDISH GROUP

## The CHAOS Report (1994)

"The Roman bridges of antiquity were very inefficient structures. By modern standards, they used too much stone, and as a result, far too much labor to build. Over the years we have learned to build bridges more efficiently, **using fewer materials and less labor to perform the same task.**"

-- Tom Clancy, The Sum of All Fears

**INTRODUCTION**

In 1986, Alfred Spector, president of Transarc Corporation, co-authored a paper comparing bridge building to software development. The premise: Bridges are normally built on-time, on-budget, and do not fall down. On the other hand, software never comes in on-time or on-budget. In addition, it always breaks down. (Nevertheless, bridge building did not always have such a stellar record. Many bridge building projects overshot their estimates, time frames, and some even fell down.)

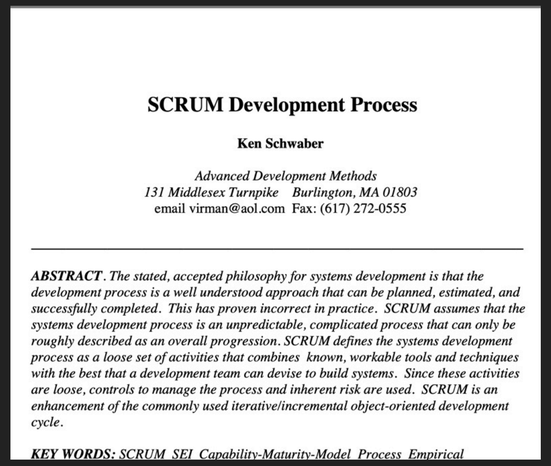
One of the biggest reasons bridges come in on-time, on-budget and do not fall down is because of the extreme detail of design. The design is frozen and the contractor has little flexibility in changing the specifications. However, in today's fast moving business environment, a frozen design does not accommodate changes in the business practices. Therefore a more flexible model must be used. This could be and has been used as a rationale for development failure.

But there is another difference between software failures and bridge failures, beside 3,000 years of experience. When a bridge falls down, it is investigated and a report is written on the cause of the failure. This is not so in the computer industry where failures are covered up, ignored, and/or rationalized. As a result, we keep making the same mistakes over and over again.

Consequently the focus of this latest research project at The Standish Group has been to identify:

- The scope of software project failures
- The major factors that cause software projects to fail

1995



## SCRUM Development Process

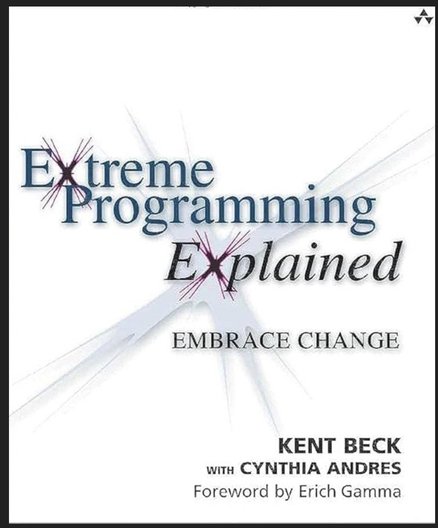
**Ken Schwaber**

*Advanced Development Methods*
*131 Middlesex Turnpike Burlington, MA 01803*
email virman@aol.com Fax: (617) 272-0555

***ABSTRACT**. The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

***KEY WORDS:** SCRUM, SEI Capability-Maturity-Model, Process, Empirical*

1999



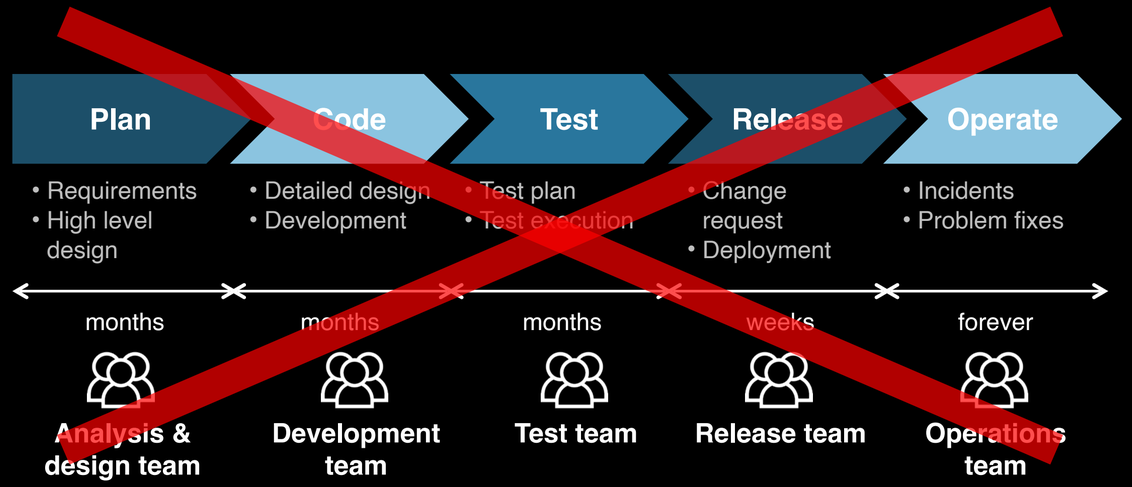
Extreme Programming Explained

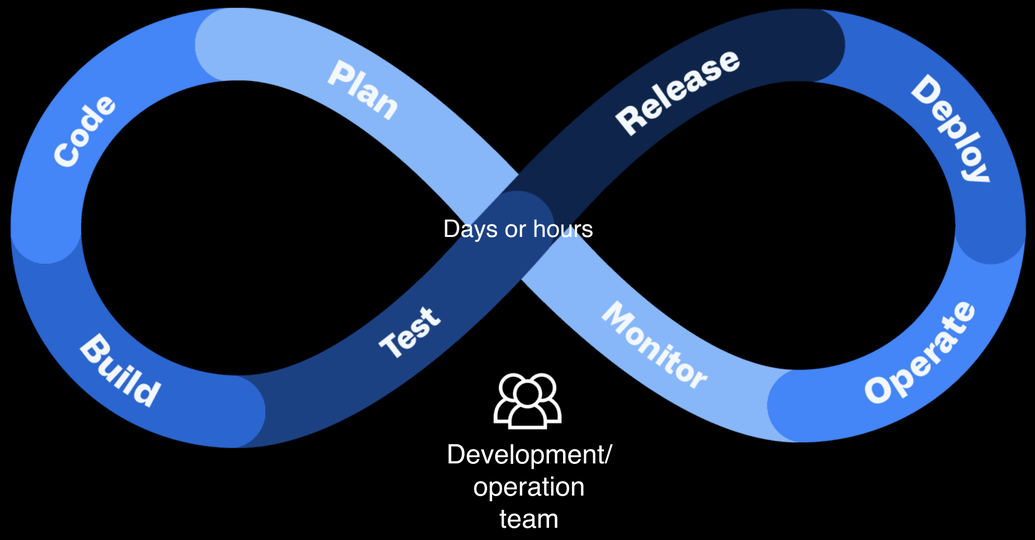
EMBRACE CHANGE

KENT BECK
WITH CYNTHIA ANDRES
Foreword by Erich Gamma

# Software ate the world

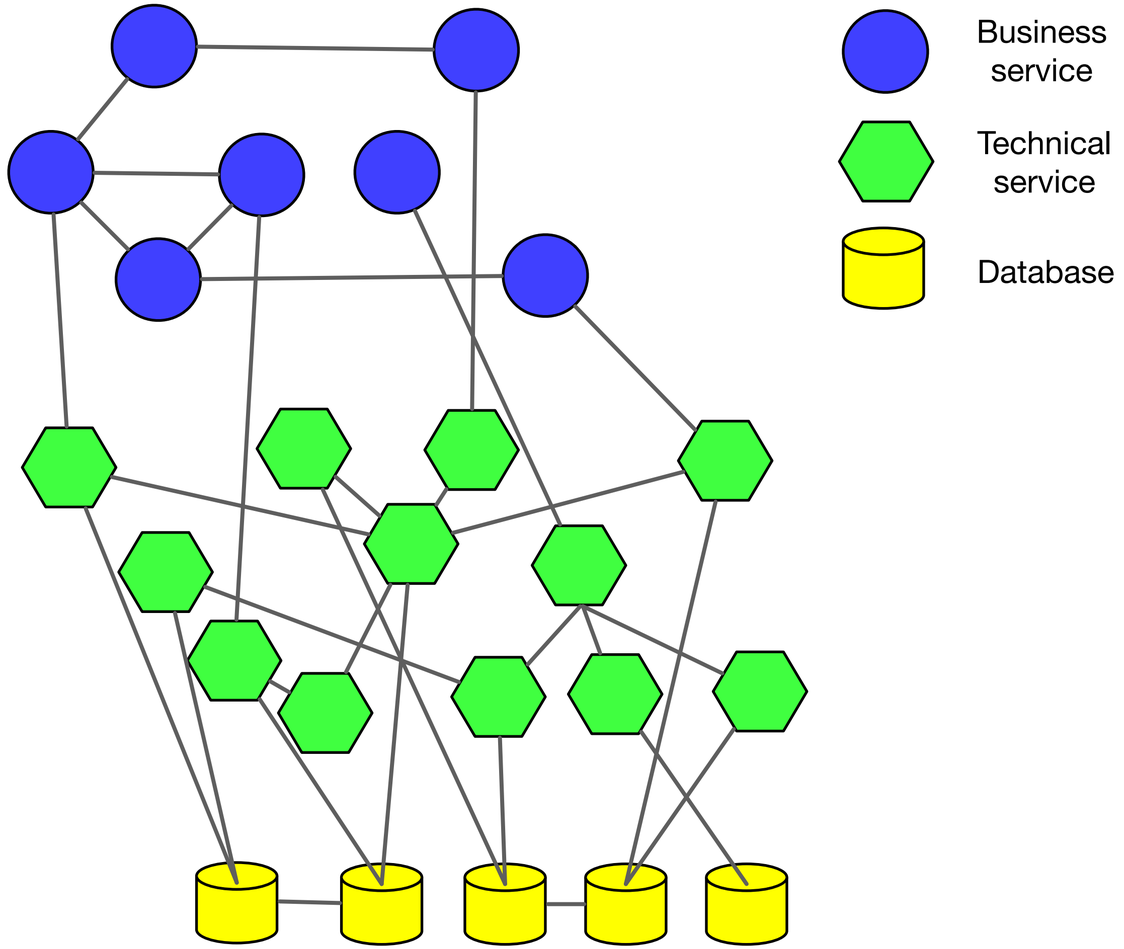
# A paradigm shift happened in software development



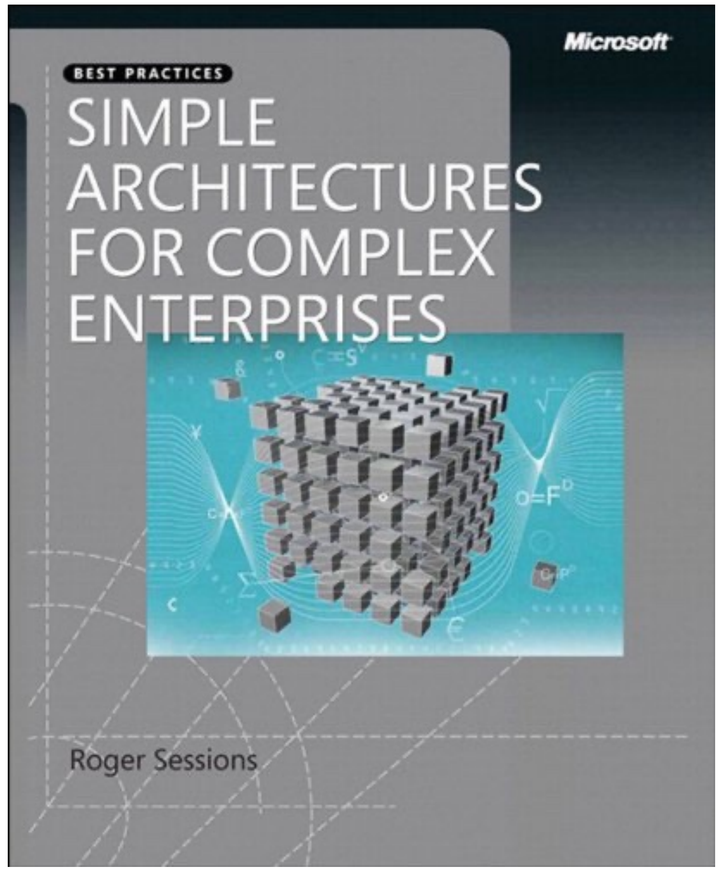



# Not everyone noticed

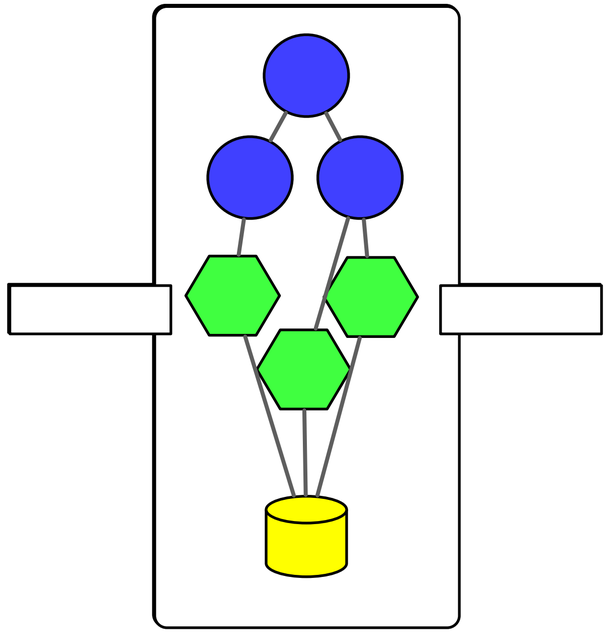
# Complexity in the existing (“legacy”) system landscape started to dominate the pace of change



# Patterns were documented to manage complexity but not everyone used them. They still don't.




2008



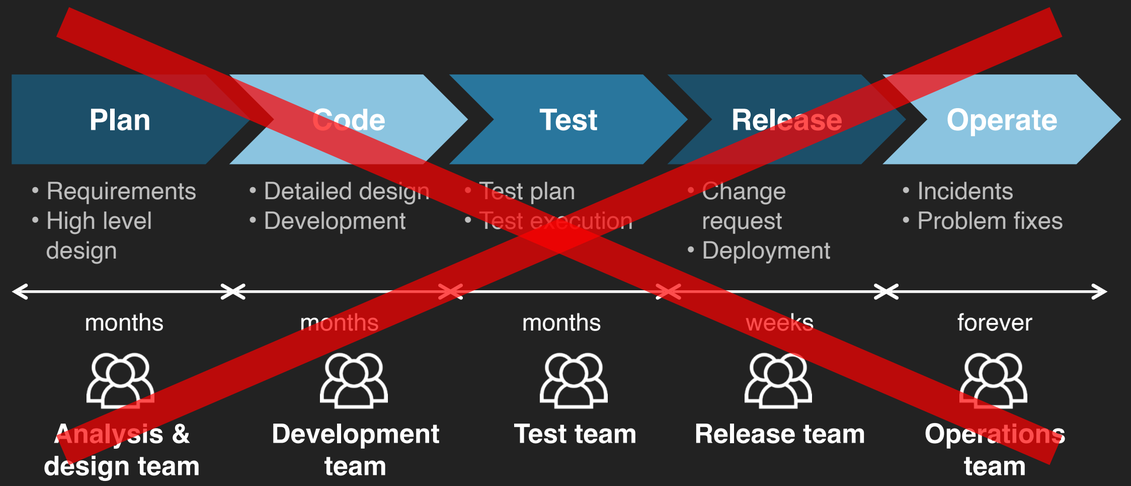
Autonomous business capability

1. Autonomy
2. Explicit boundaries
3. Partitioning of functionality
4. Dependencies defined by functionality
5. Asynchronicity [by default]
6. Partitioning of data
7. No cross-fortress transactions
8. Single point security
9. Inside trust
10. **Keep it simple**

# Our problems are similar but different to those in the 1990s

## 1990s

Civil engineering processes are the best way to deliver any software change




## 2020s

The enterprise software landscape is so complex we need to use project management techniques to coordinate multiple teams for simple features

An organization that lacks a viable program in enterprise architecture will pay a severe cost in IT complexity.

**Complexity is the enemy.**

Enterprise Architecture is the solution.

The only solution.

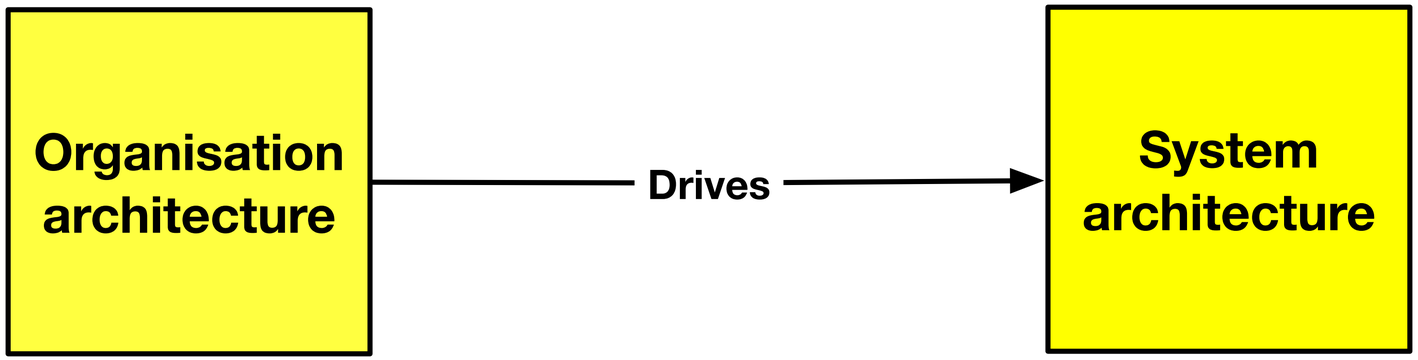
Roger Sessions, 2008

https://web.archive.org/web/20211021220626/https://simplearchitectures.blogspot.com/2008/11/job-of-enterprise-architect.html

# A

**A**ligning Value, People and Technology

# Conway’s law (very roughly)



If the architecture of the system and the architecture of the organization are at odds, the architecture of the organization wins.

Ruth Malan, 2008

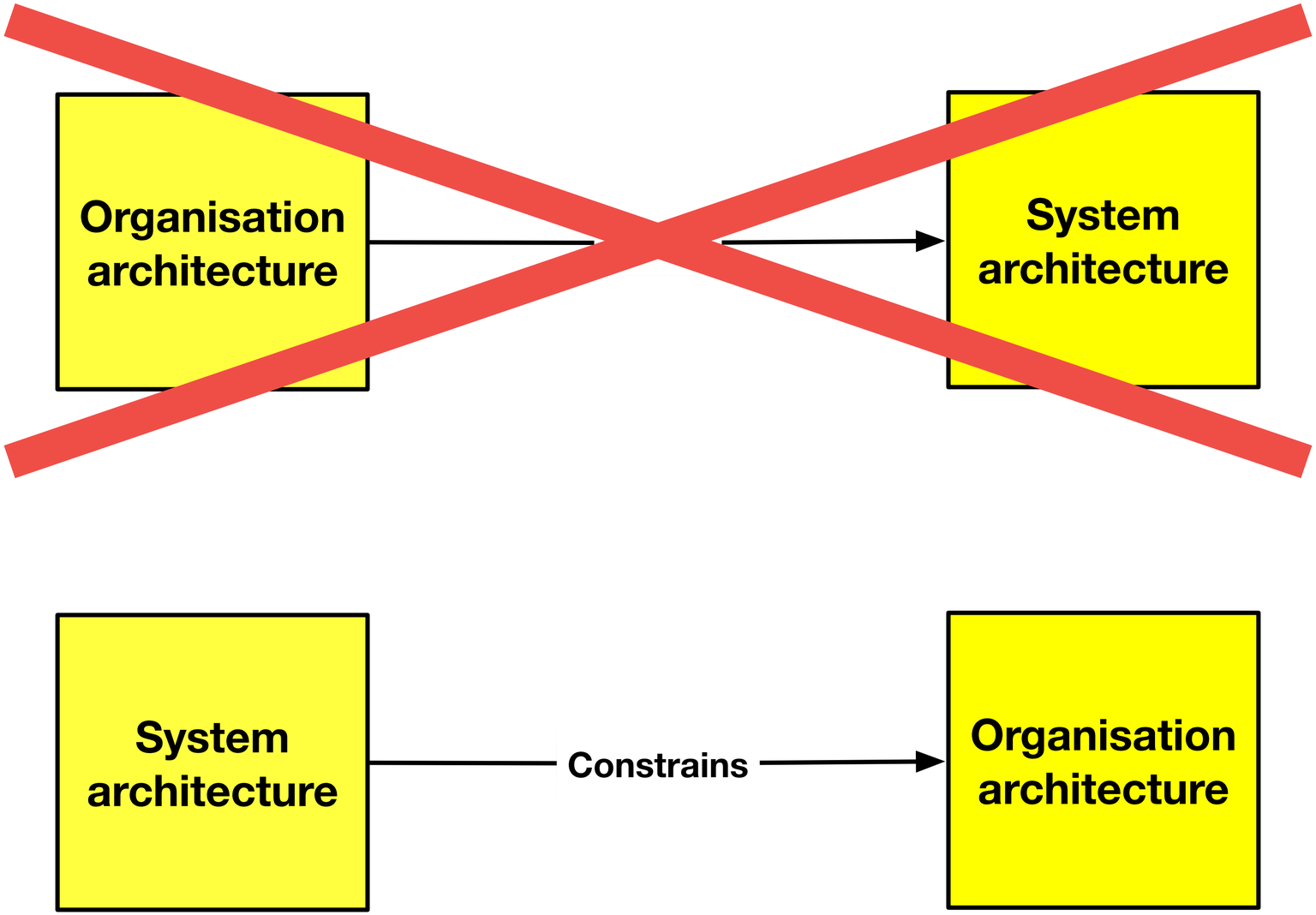
https://web.archive.org/web/20221205155201/https://ruthmalan.com/traceinthesand/conwayslaw.htm

# BUT!

Where all hell really breaks loose is when management decides to reorganise things.
… if you try to change the organisation, the software won't let it happen.

Allan Kelly, 2018

https://www.youtube.com/watch?v=Cu0AU8vw3xw


Organisation architecture
System architecture
System architecture
Constrains
Organisation architecture

???

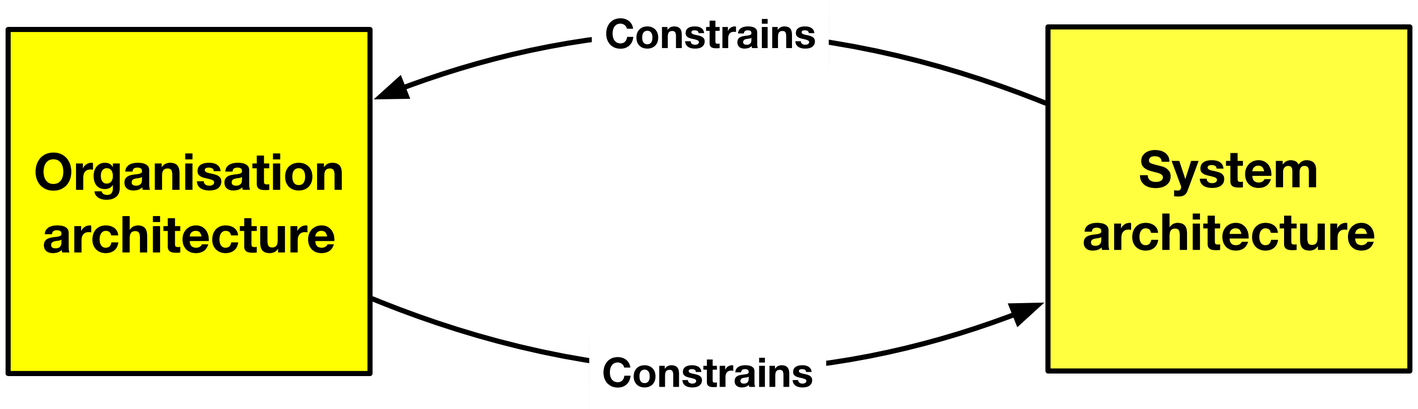
We shape our buildings; thereafter they shape us

Winston Churchill, 1943

We shape our architecture and then the architecture shapes us

Gene Kim, 2022

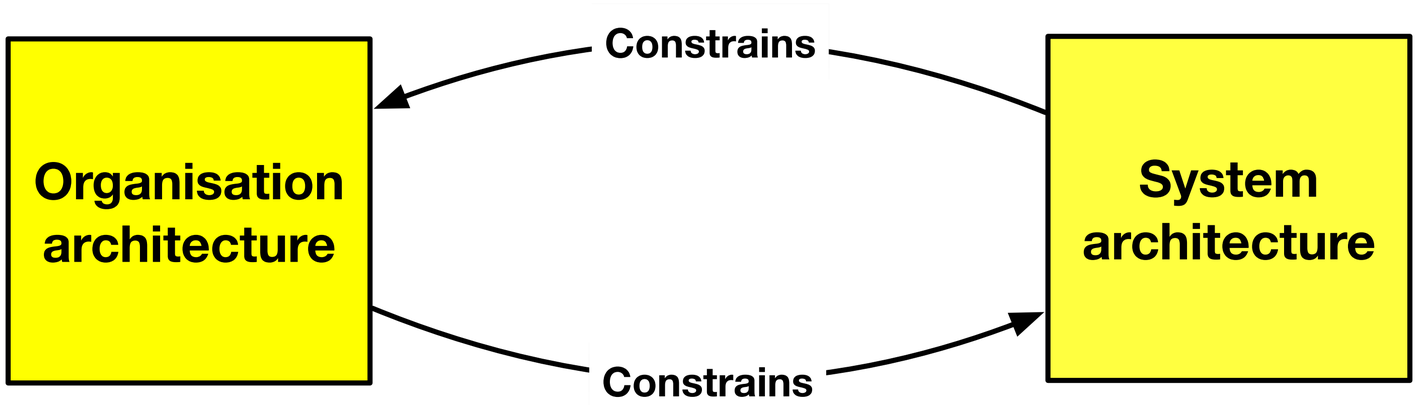
https://web.archive.org/web/20231015100627/https://twitter.com/lbmkrishna/status/1547447927567953920?s=20


Organisation architecture
Constrains
Constrains
System architecture

System architects (who we call architects) and business/organization architects (who we call managers) should not work as if one has no impact on the other.

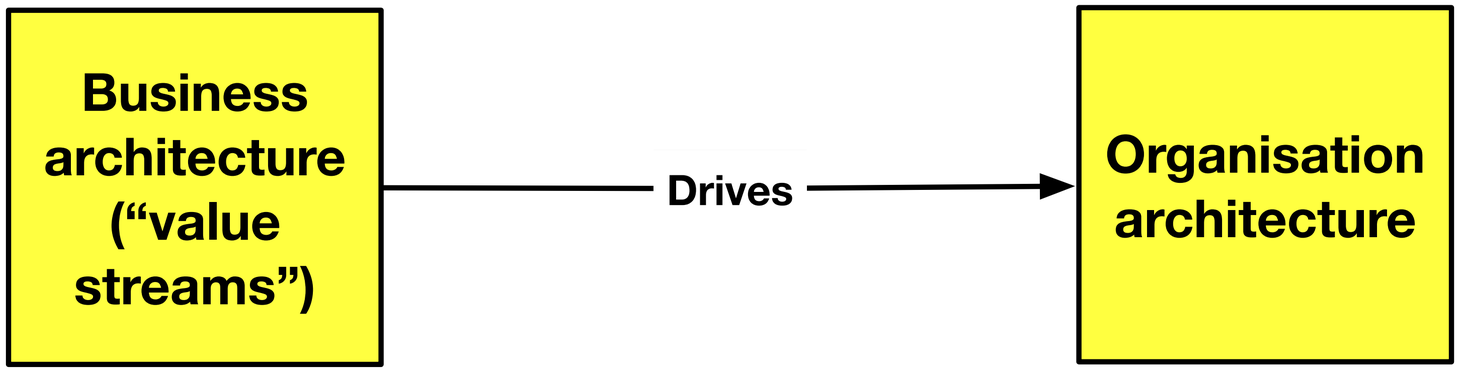
Ruth Malan, 2008

https://web.archive.org/web/20221205155201/https://ruthmalan.com/traceinthesand/conwayslaw.htm




Organization architects
(“managers”)

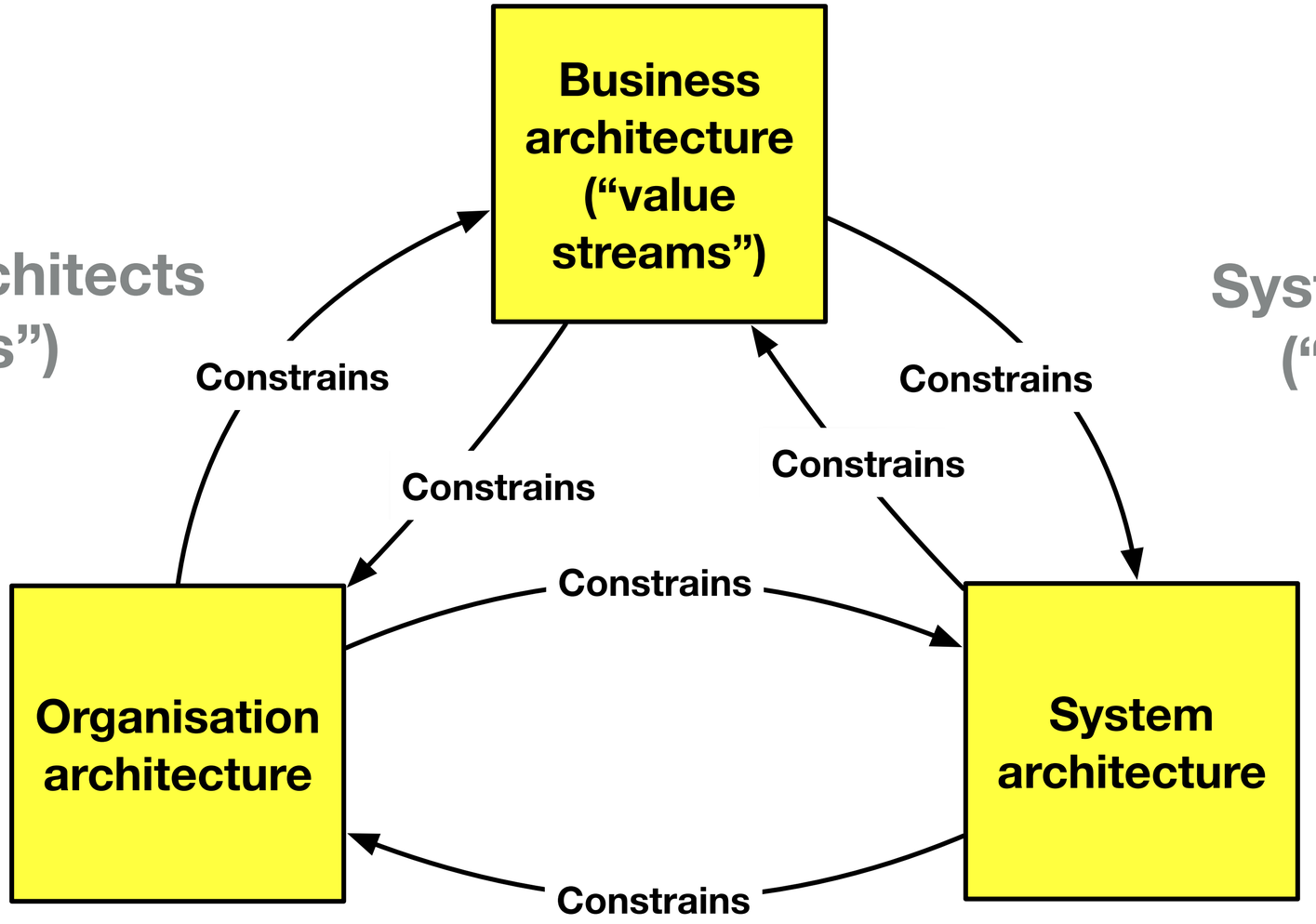
System architects
(“architects”)




Organization architects
(“managers”)

Organization architects
(“managers”)

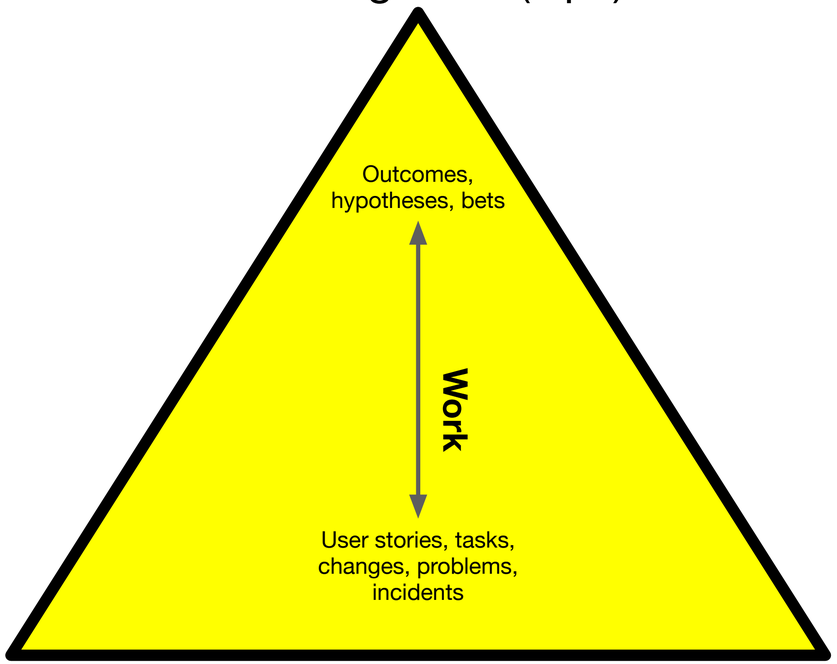
System architects
(“architects”)



# Aligning all 3 for sustainable flow

## Value

OKRs looking forward (Dev)
KPIs & OLIs right now &
looking back (Ops)




## Organization

People, teams,
teams-of-teams,
departments

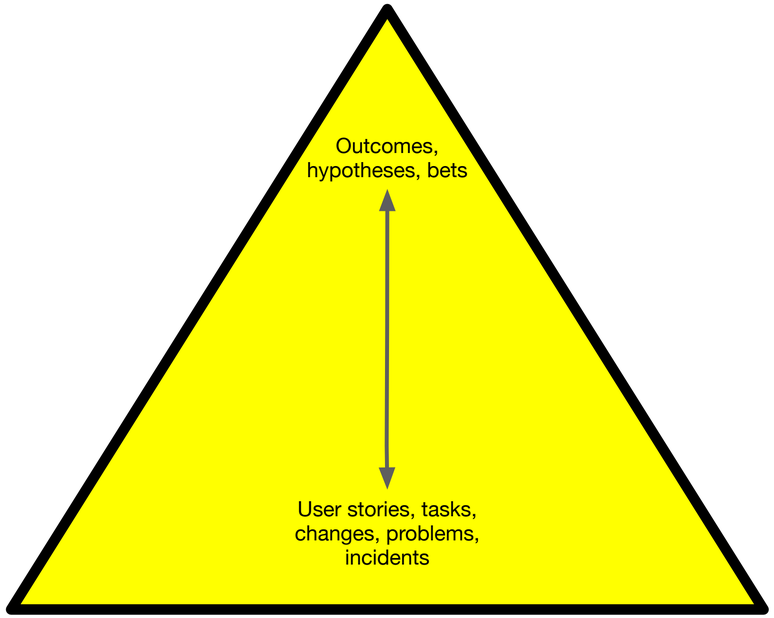
## Technology & processes

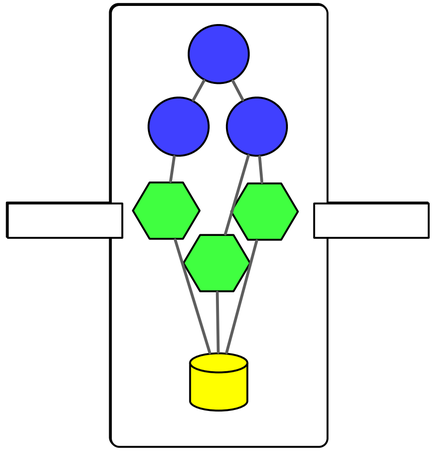
Business services &
business processes,
IT services & IT
components

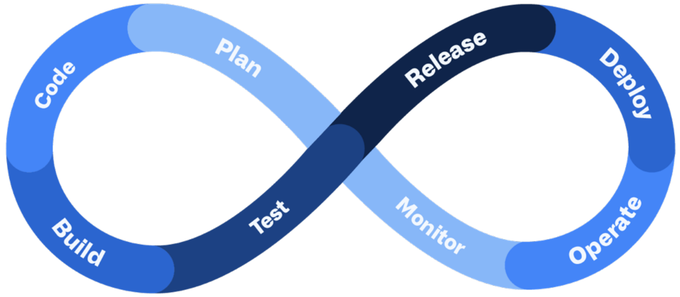
# Level 0: an ideal 2-pizza stream-aligned team

## Value

OKRs looking forward (Dev)
KPIs & OLIs right now &
looking back (Ops)






## 2 pizza team - <?20 people

Full stack, full lifecycle, full burrito, T-shaped people - YBIYRI
Self-organizing

## Software that fits inside the team's head

DDD
Eventually consistent
Independently deployable & testable
“Monolith vs Microservice is missing the point”
Emergent design & evolutionary* architecture

design

Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

# Who Needs an Architect?

**Martin Fowler**

Wandering down our corridor a while ago, I saw my colleague Dave Rice in a particularly grumpy mood. My brief question caused a violent statement, "We shouldn't interview anyone who has 'architect' on his resume." At first blush, this was an odd turn of phrase, because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry's standards, "architect" and "architecture" are terribly overloaded words. For many, the term "software architect" fits perfectly with the smug controlling image at the end of *Matrix Reloaded*. Yet even in firms that have the greatest contempt for that image, there's a vital role for the technical leadership that an architect such as Dave plays.

## What is architecture?

When I was fretting over the title for *Patterns of Enterprise Application Architecture* (Addison-Wesley, 2002), everyone who reviewed it agreed that "architecture" belonged in the title. Yet we all felt uncomfortable defining the word. Because it was my book, I felt compelled to take a stab at defining it.

chitect.) However, as so often occurs, inside the blighted cynicism is a pinch of truth. Understanding came to me after reading a posting from Ralph Johnson on the Extreme Programming mailing list. It's so good I'll quote it all.

A previous posting said

> The RUP, working off the IEEE definition, defines architecture as "the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Johnson responded:

> I was a reviewer on the IEEE standard that used that, and I argued uselessly that this was clearly a completely bogus definition. There is no highest level concept of a system. Customers have a different concept than developers. Customers do not care at all about the structure of significant components. So, perhaps an architecture is the highest level concept that developers have of a system in its environment. Let's forget the developers who just understand their little piece. Architecture is the highest level concept of the expert developers. What makes a component



# Is Design Dead?

*For many that come briefly into contact with Extreme Programming, it seems that XP calls for the death of software design. Not just is much design activity ridiculed as "Big Up Front Design", but such design techniques as the UML, flexible frameworks, and even patterns are de-emphasized or downright ignored. In fact XP involves a lot of design, but does it in a different way than established software processes. XP has rejuvenated the notion of evolutionary design with practices that allow evolution to become a viable design strategy. It also provides new challenges and skills as designers need to learn how to do a simple design, how to use refactoring to keep a design clean, and how to use patterns in an evolutionary style.*

May 2004

Martin Fowler

POPULAR

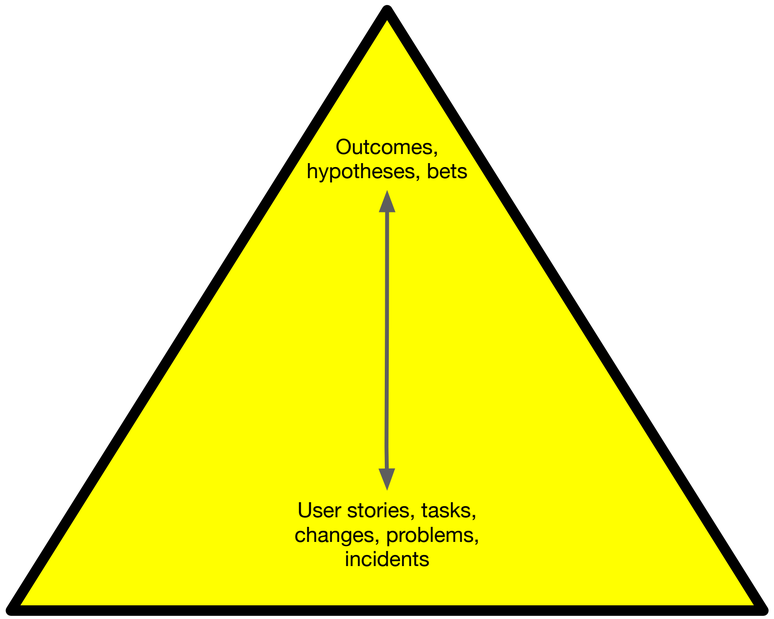
DESIGN

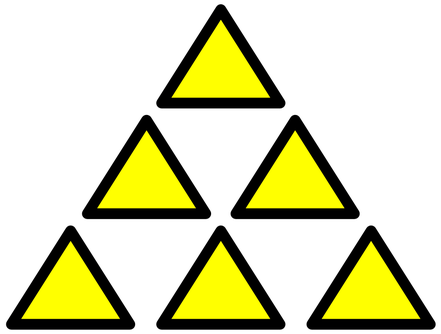
AGILE

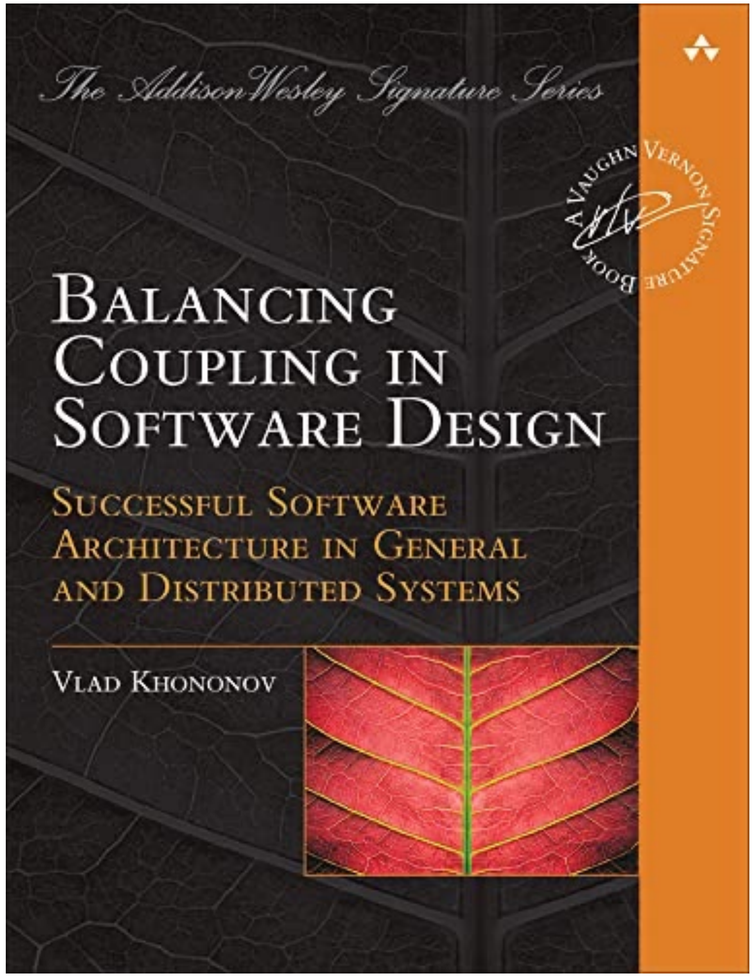
EXTREME PROGRAMMING

EVOLUTIONARY DESIGN

**CONTENTS**


Outcomes,
hypotheses, bets
User stories, tasks,
changes, problems,
incidents

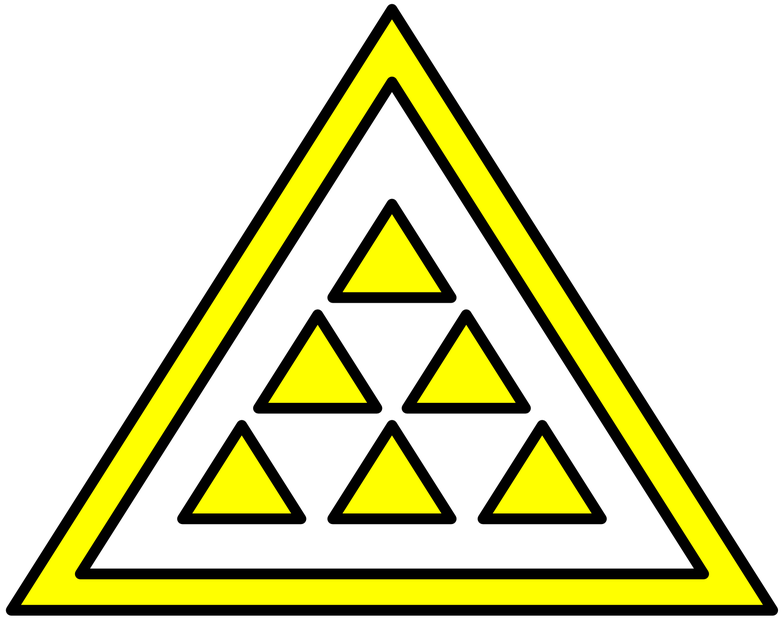


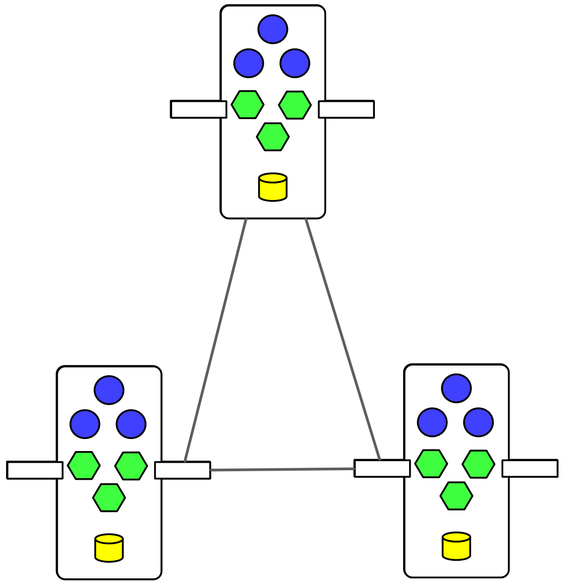

The Addison Wesley Signature Series
A VAUGHN VERNON SIGNATURE BOOK
BALANCING COUPLING IN SOFTWARE DESIGN
SUCCESSFUL SOFTWARE ARCHITECTURE IN GENERAL AND DISTRIBUTED SYSTEMS
VLAD KHONONOV

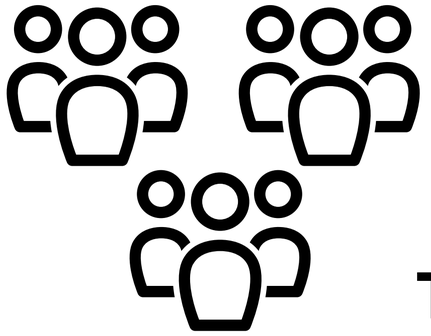
# Level 1: an ideal team-of-teams

**Value**

OKRs looking forward (Dev)
KPIs & OLIs right now &
looking back (Ops)







**Team-of-teams**
**Dunbar number - <150**

"Tribe"
Nested value
Self-organizing?

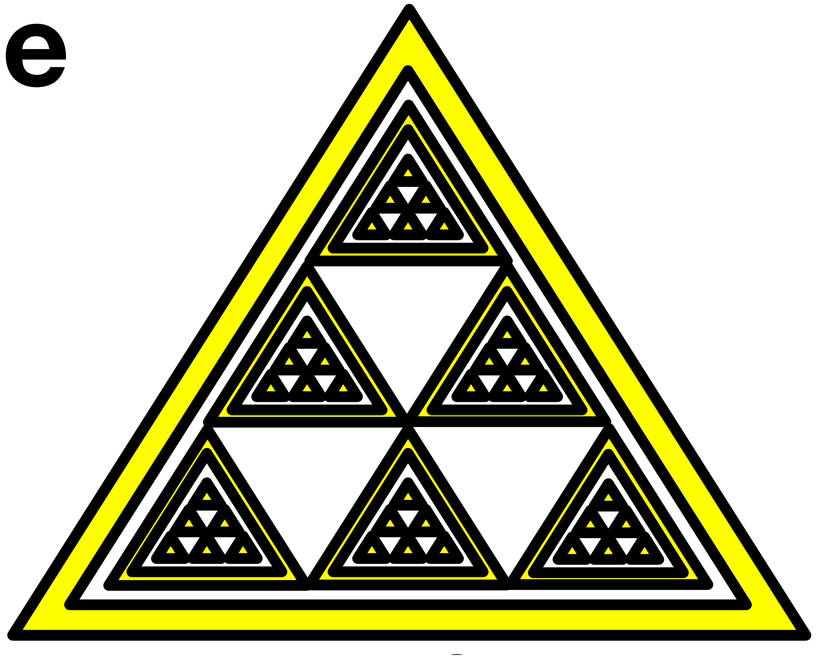
**System-of-systems & domain that fits inside the team's head**

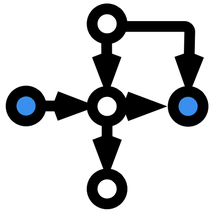
Nested DDD
Shared data model?
Eventually consistent
Each component independently deployable & testable

# Level 2: an ideal business line

**Value**

OKRs looking forward (Dev)
KPIs & OLIs right now &
looking back (Ops)





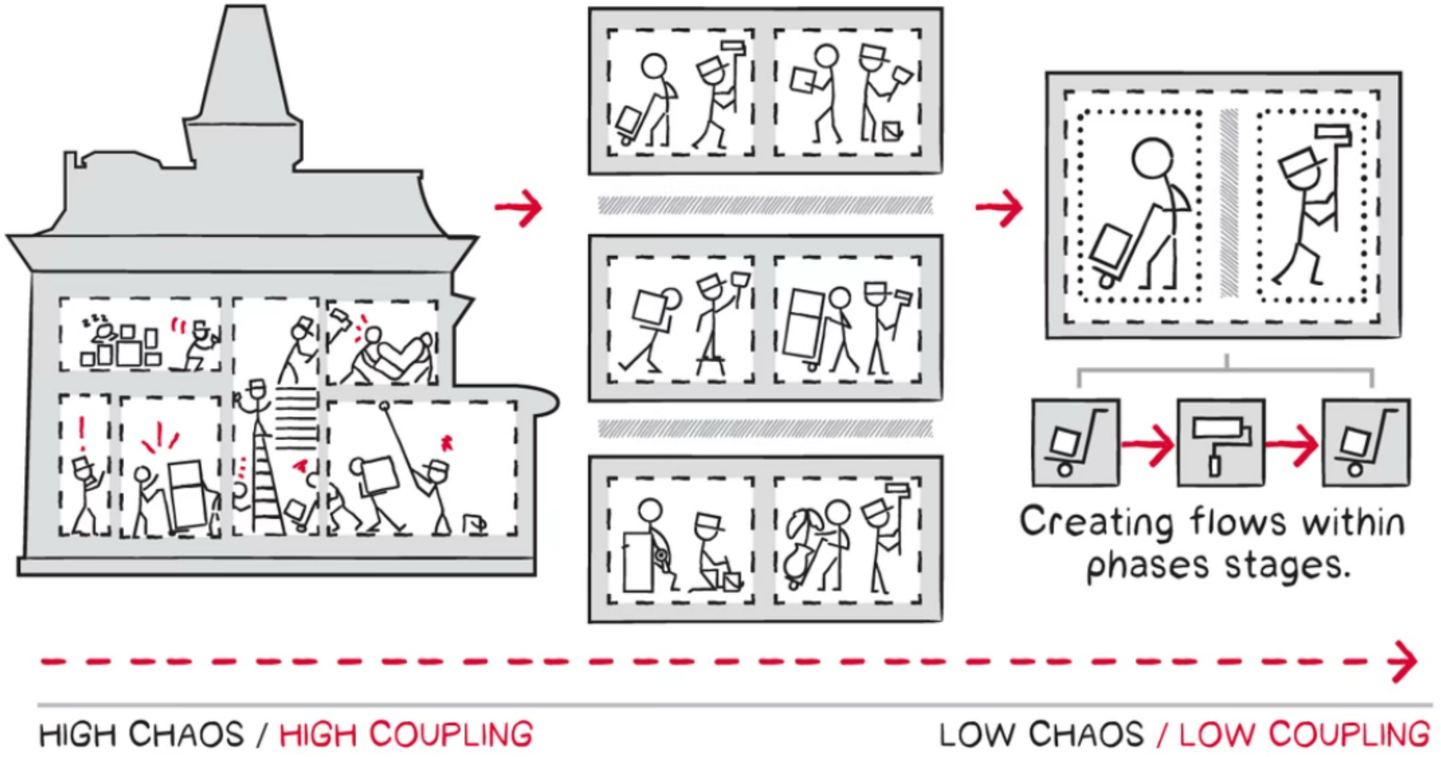
**Team-of-team-of-teams**
**1000–2000**

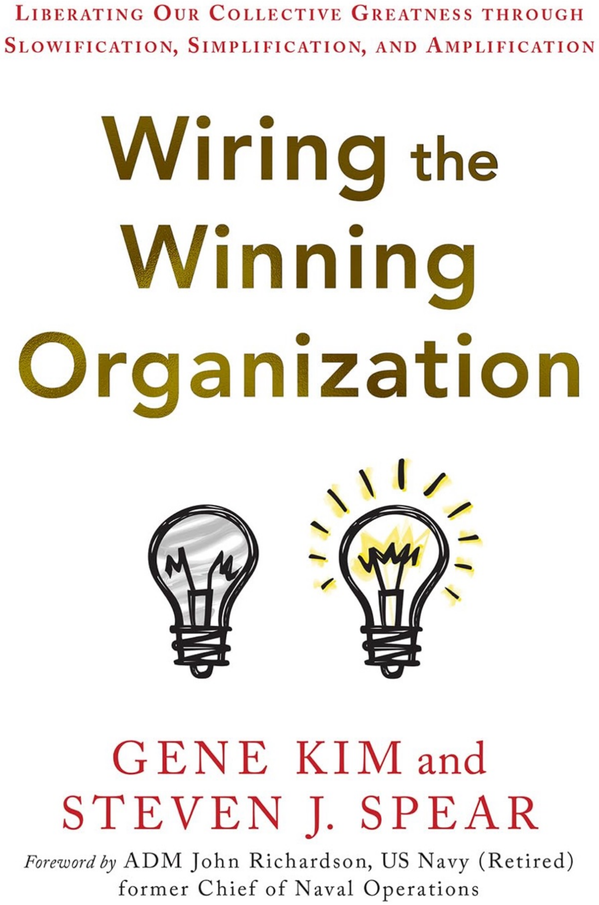
Department /
business line
Nested value
Self-organizing???

**System-of-system-of-systems**
**Does this fit in anyone's head?**

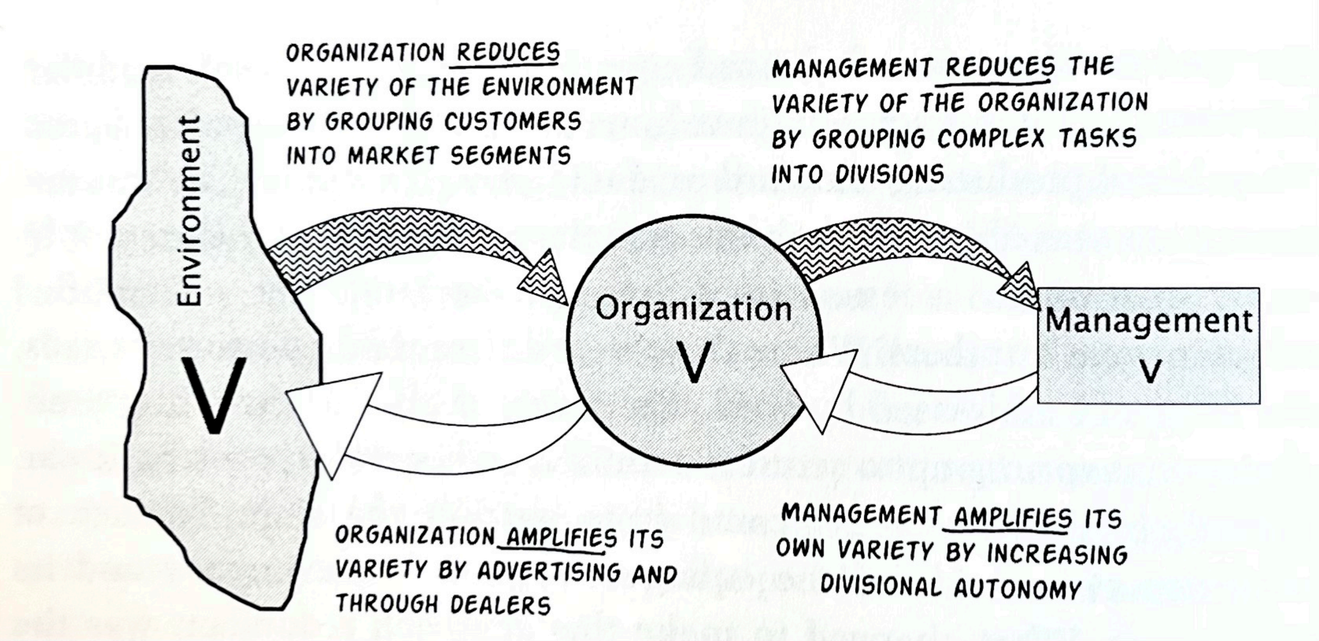
Further nested DDD
Minimally shared data model - just keys?
Eventually consistent between subdomains
Key paths and failure modes should be known

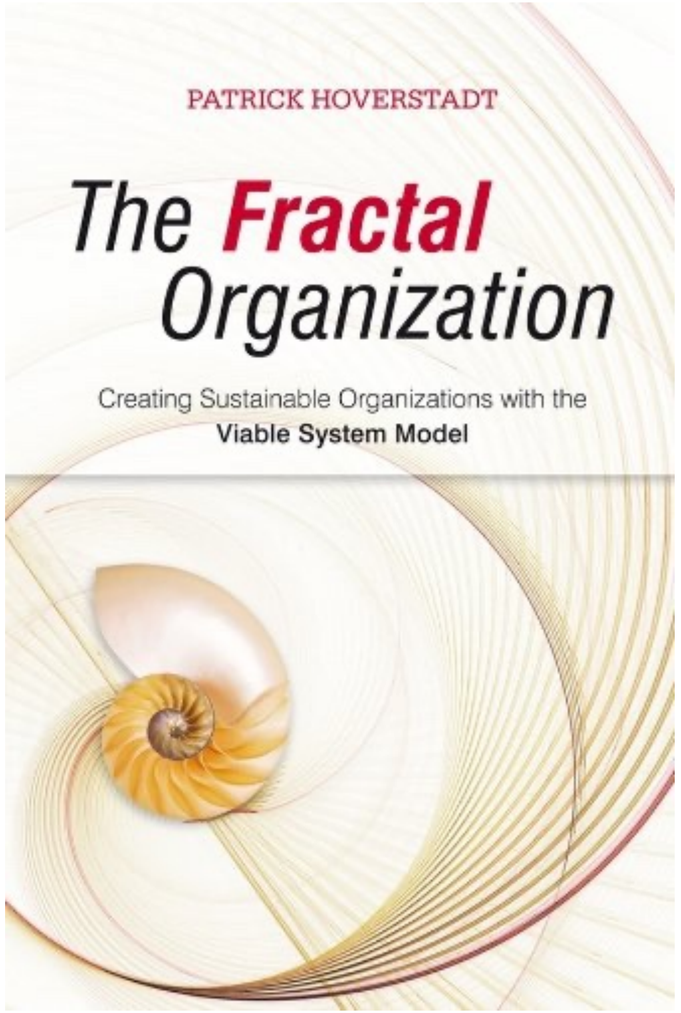
# Simplification through modularity



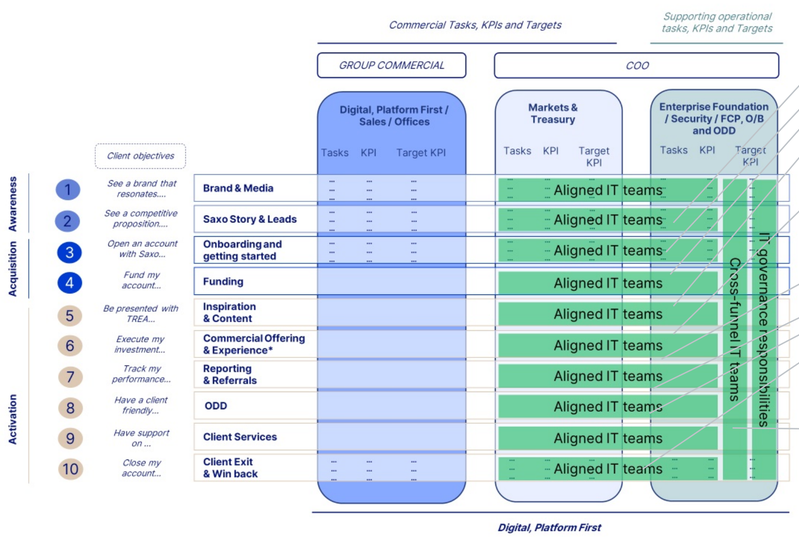
# The fractal organisation

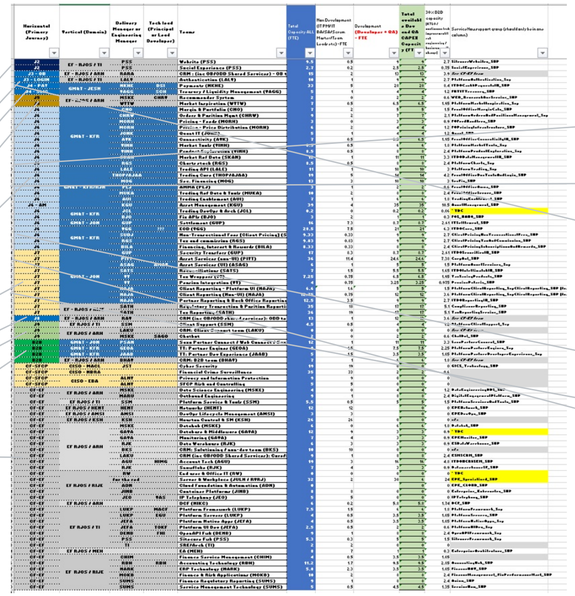


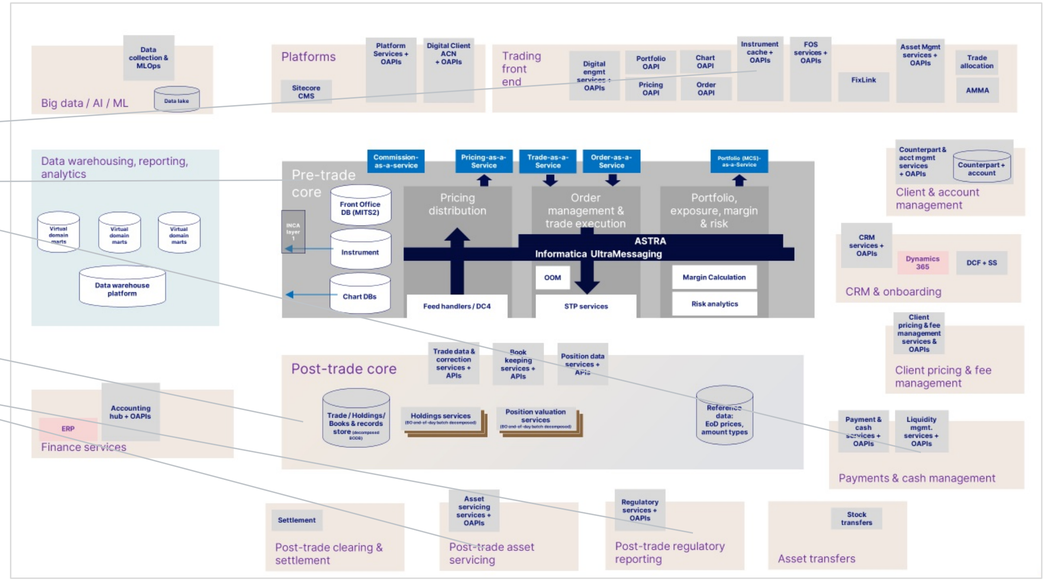




- Control theory & cybernetics
- Stafford Beer's Viable Systems Model
- Ashby's Law of Requisite Variety: "only variety can absorb variety"
- Attenuation and amplification

# The fractal organisation at Saxo Bank

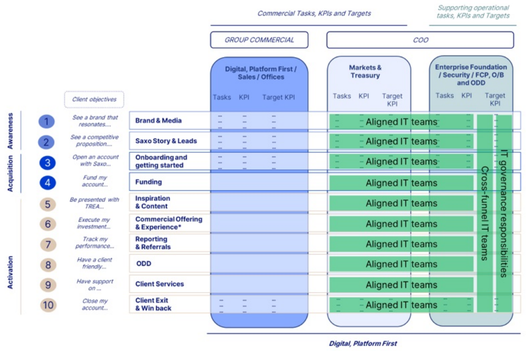





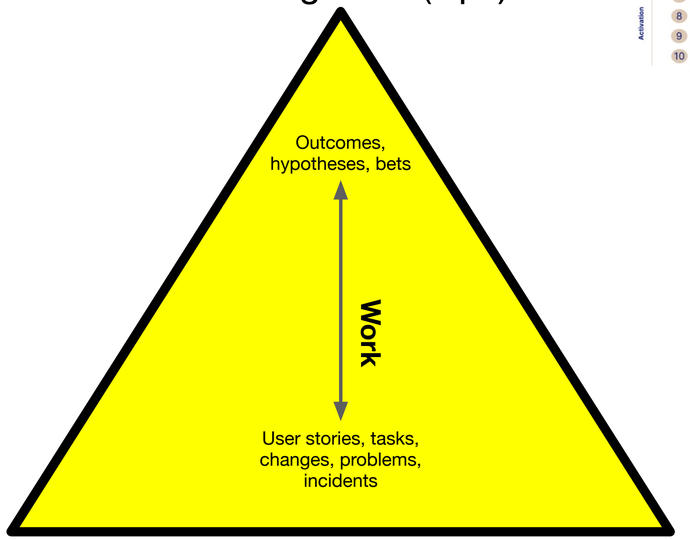

SAXO
BE INVESTED

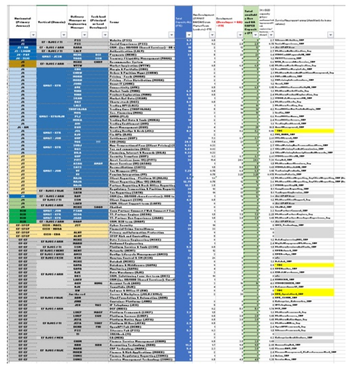
# The fractal organisation at Saxo Bank




## Value

OKRs looking forward (Dev)
KPIs & OLIs right now &
looking back (Ops)



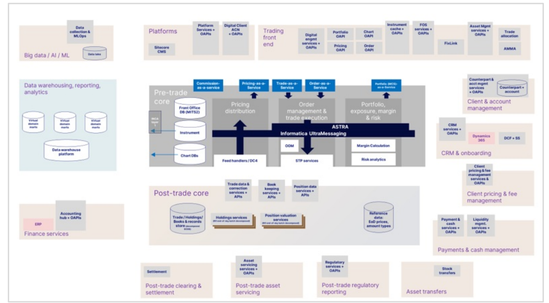



## Organization

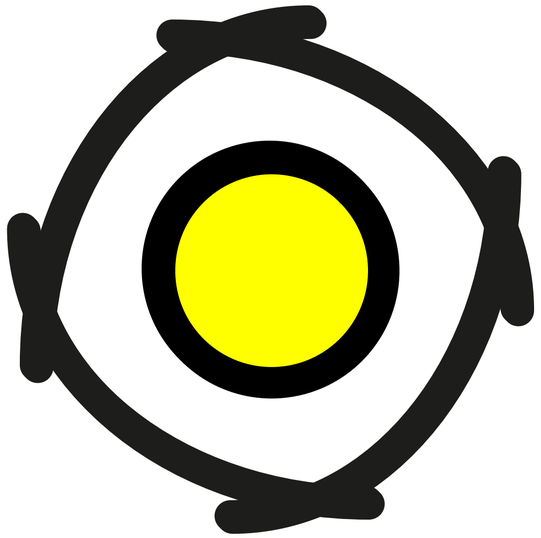
People, teams,
teams-of-teams,
departments

## Technology & processes

Business services &
business processes,
IT services & IT
components

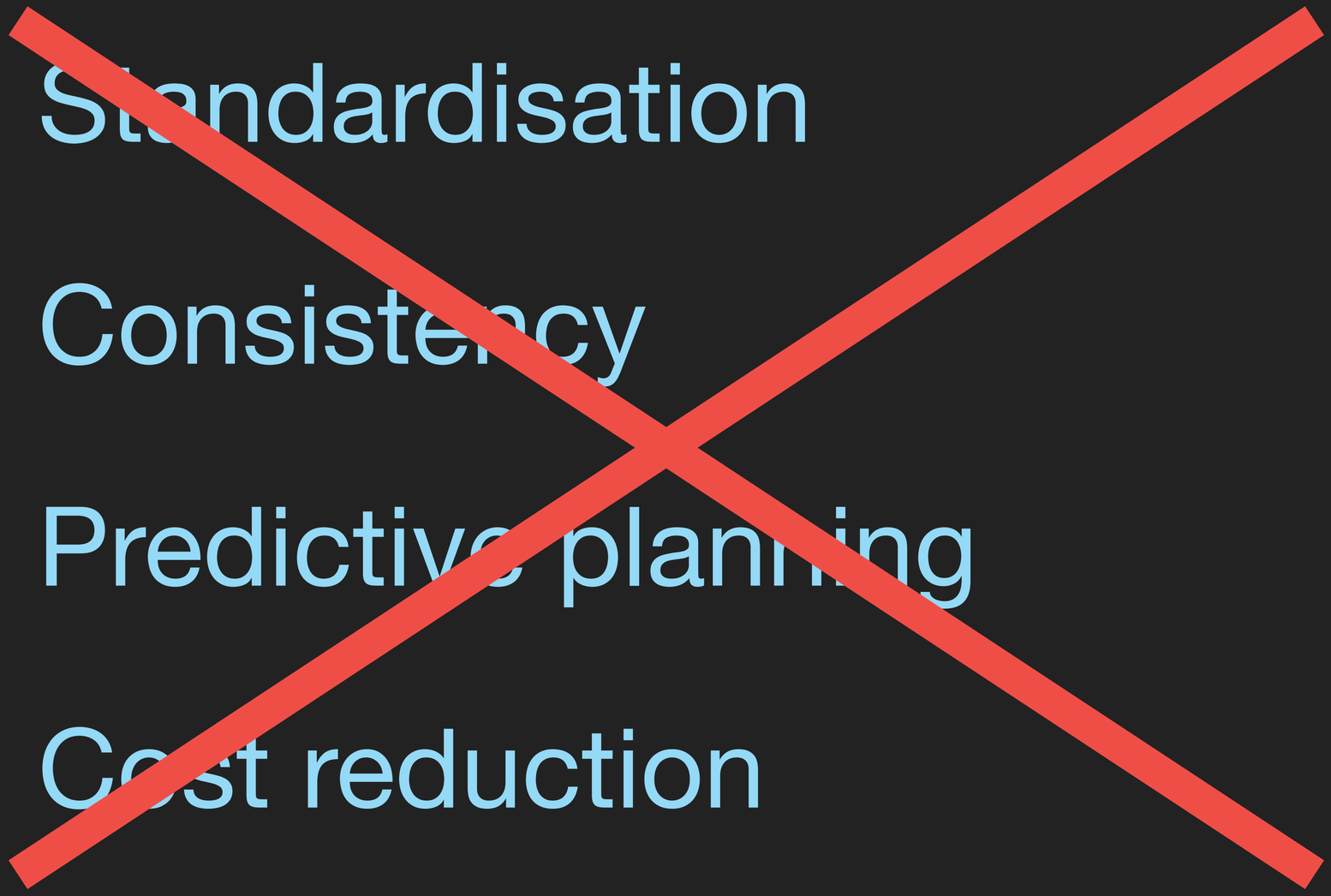



SAXO
BE INVESTED



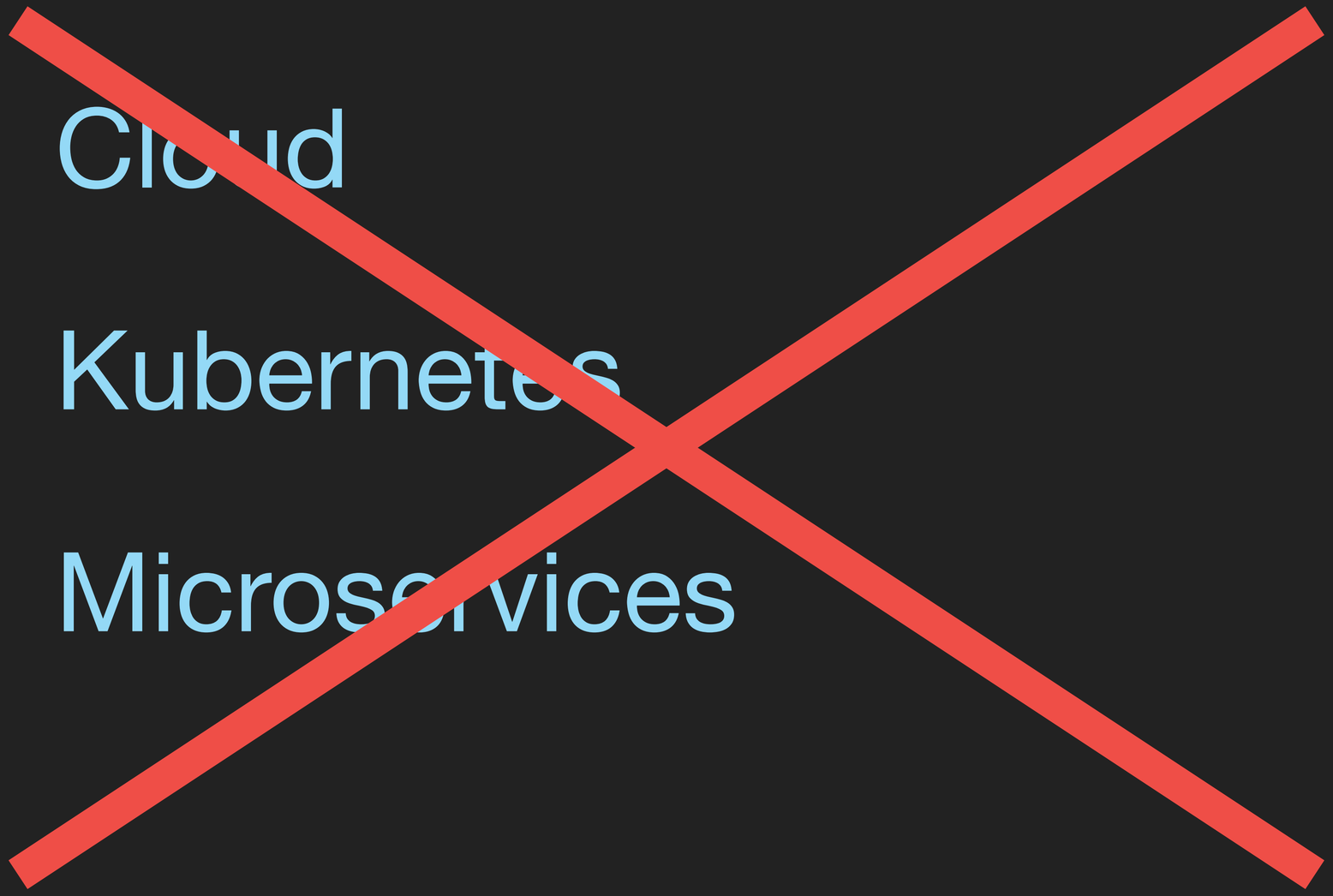
# Architecting for Outcomes: **B**etter Value, Sooner, Safer, Happier

# Antipattern: traditional EA outcomes




These are not the outcomes you're looking for

# Antipattern: “newer” EA outcomes




These are not the outcomes you’re looking for

# Pattern: align to business agility

Quality

Flow


Customers

Colleagues

Citizens

Climate

Security

Privacy

Minimum viable compliance

We shape our architecture and then the architecture shapes us

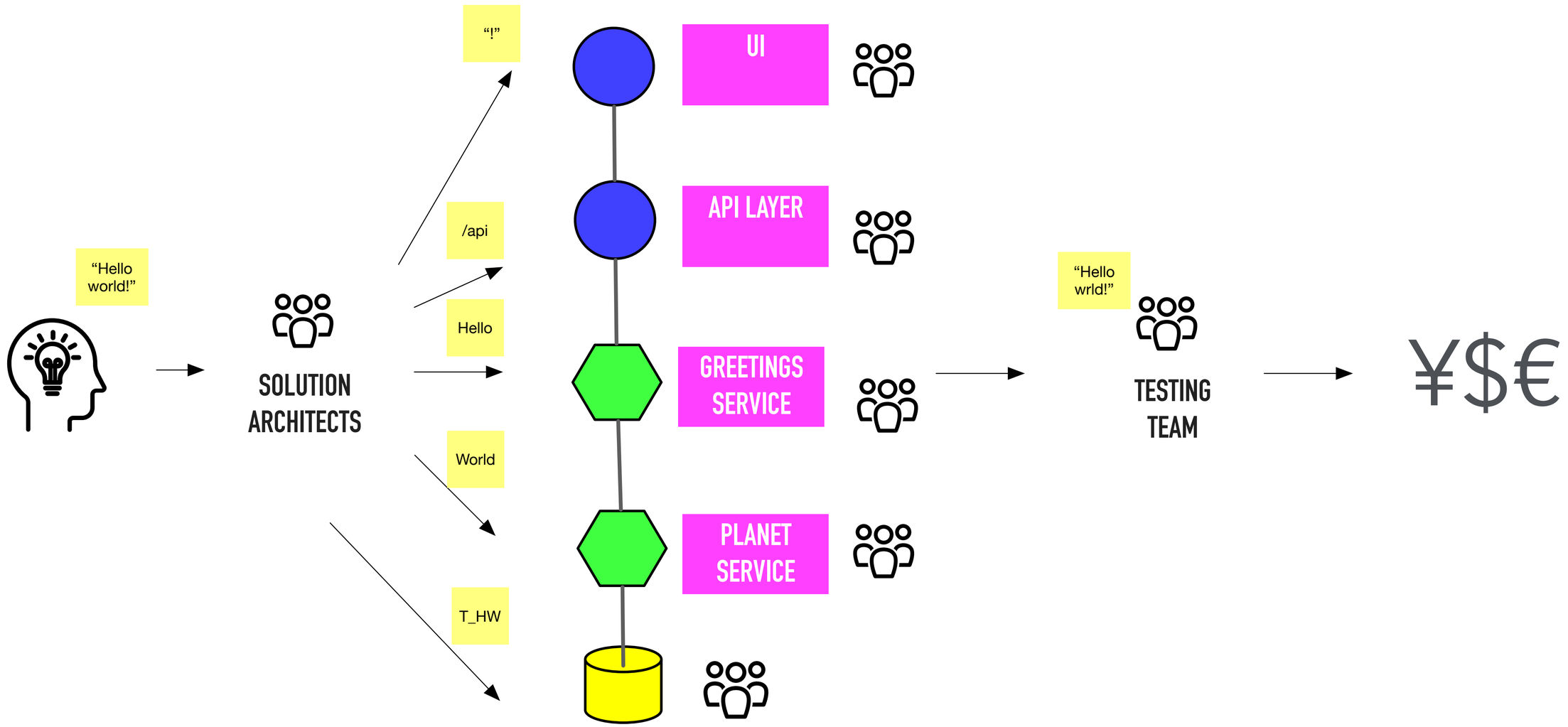
As the systems we build become larger, the coordination increases…

…it can grow so so large that **all our time and energy is spent coordinating** - it is at the expense of our value creating activities

Gene Kim

https://web.archive.org/web/20231015100627/https://twitter.com/lbmkrishna/status/1547447927567953920?s=20

# “Hello world” with tightly coupled complex architecture



A beautiful layer cocktail

# Layering (sometimes) Considered Harmful


**Gregor Hohpe**

IT Strategist firmly entrenched in the cloud engine room. Author, Speaker, former Singapore Smart Nation Fellow

74 articles ✓ Following

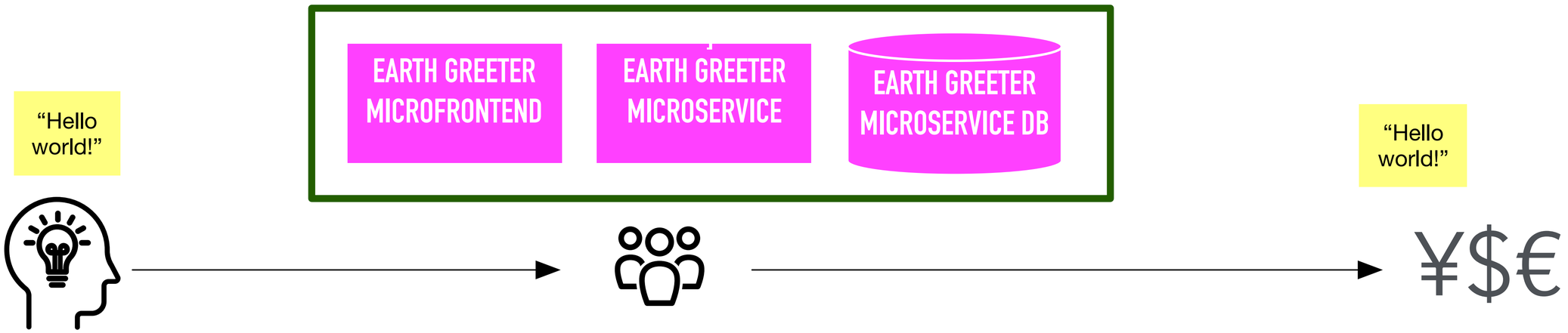

November 13, 2016

Release trains are an iterative and incremental lifecycle, but they are not agile.

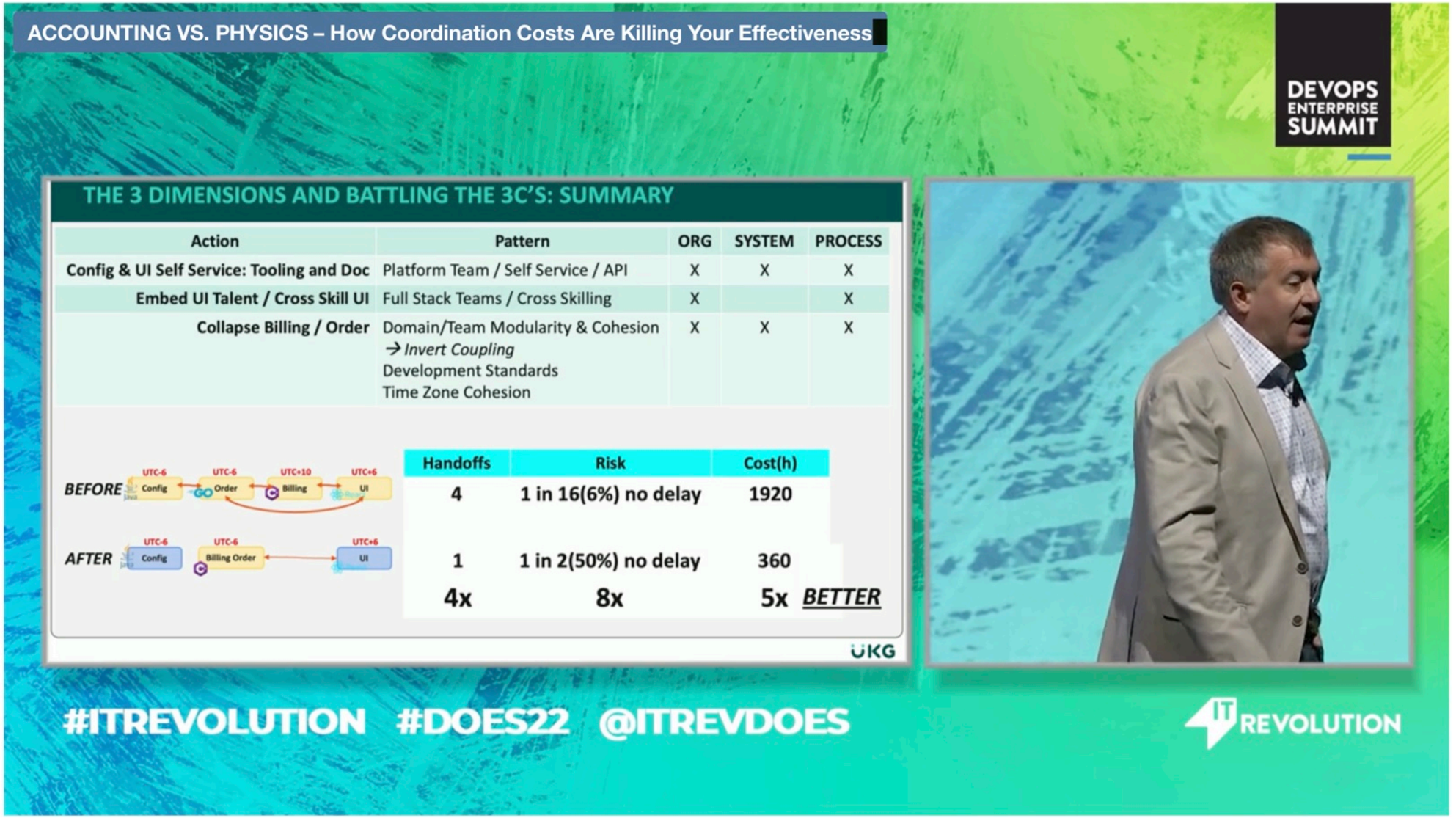
Johanna Rothman, 2011

https://www.jrothman.com/articles/2011/01/not-ready-for-agile-start-your-journey-with-release-trains/

# Autonomous team with their own valuable business capability




~~Refactored~~ Restructured organization and architecture

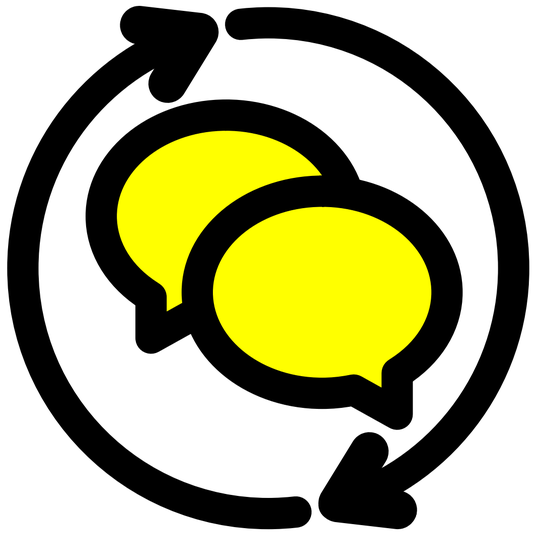



Las Vegas 2022

NO SLIDES AVAILABLE

# ACCOUNTING VS. PHYSICS – How Coordination Costs Are Killing Your Effectiveness

ACCOUNTING VS. PHYSICS – How Coordination Costs Are Killing Your Effectiveness



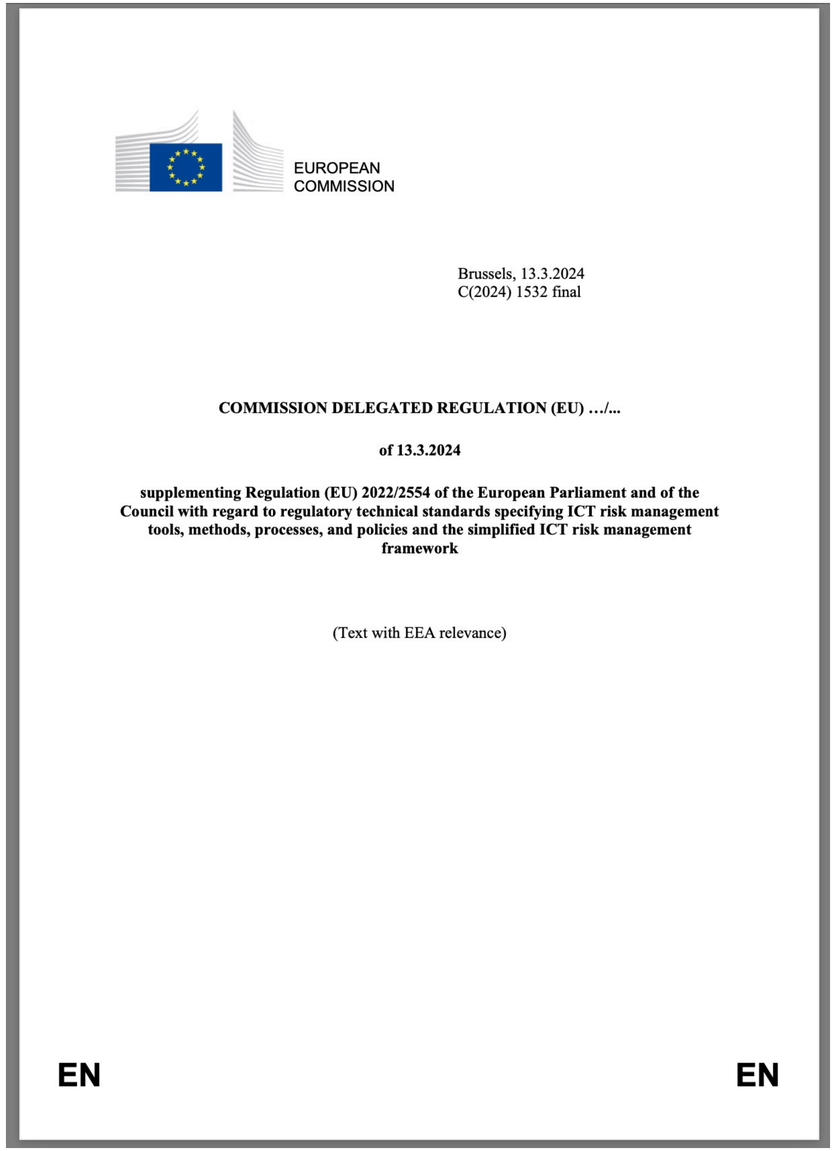
**C**

Governing
**C**ontinuously:
Conversational
and Automated

Creating and reducing complexity may sound like perfect opposites. But in fact fundamental asymmetries exist between the two.

https://hbr.org/2020/01/taming-complexity

# Complexity and compliance



EUROPEAN
COMMISSION

Brussels, 13.3.2024
C(2024) 1532 final

**COMMISSION DELEGATED REGULATION (EU) …/...**

**of 13.3.2024**

**supplementing Regulation (EU) 2022/2554 of the European Parliament and of the Council with regard to regulatory technical standards specifying ICT risk management tools, methods, processes, and policies and the simplified ICT risk management framework**

(Text with EEA relevance)

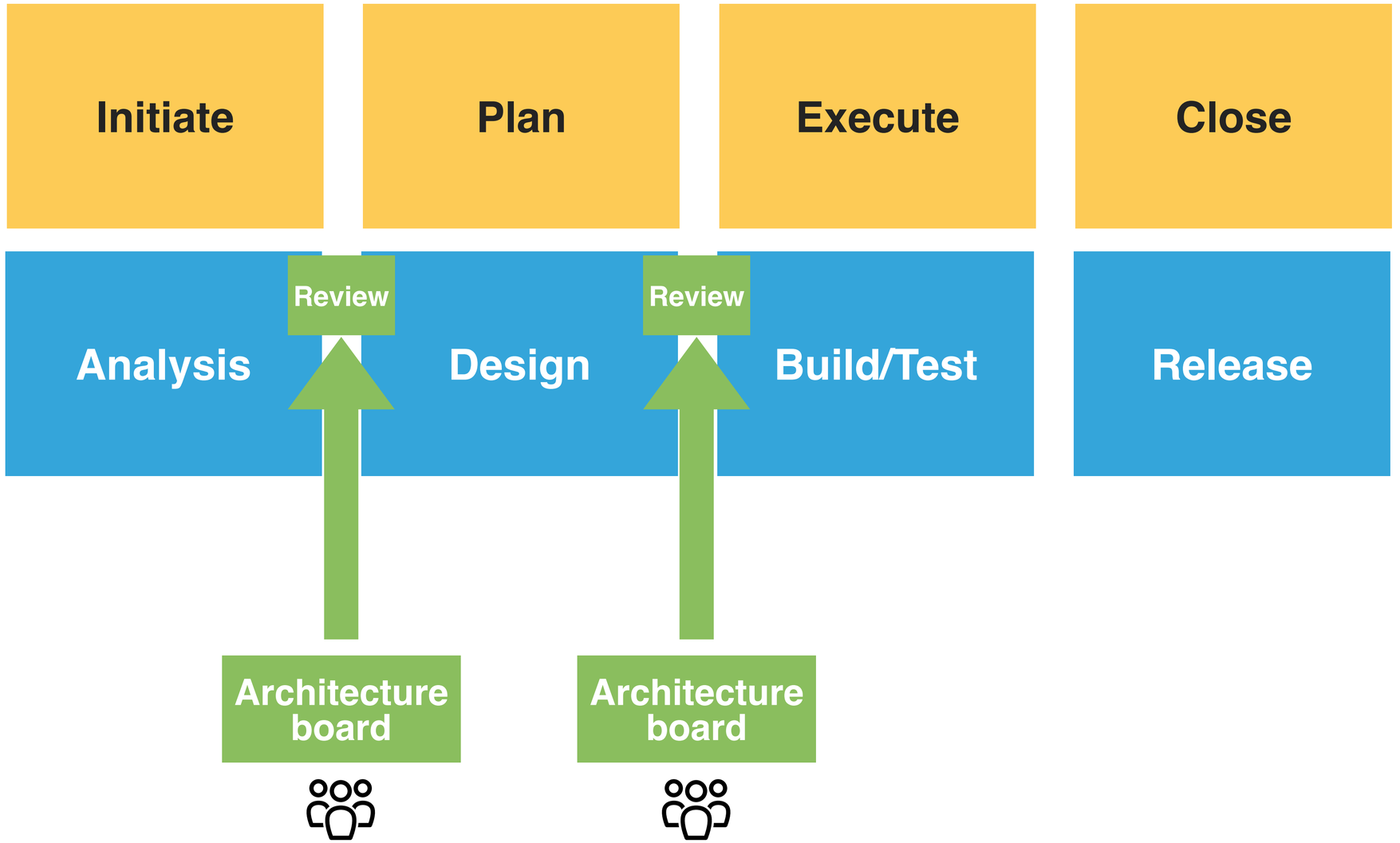
**EN** **EN**

# Architecture governance to the rescue?

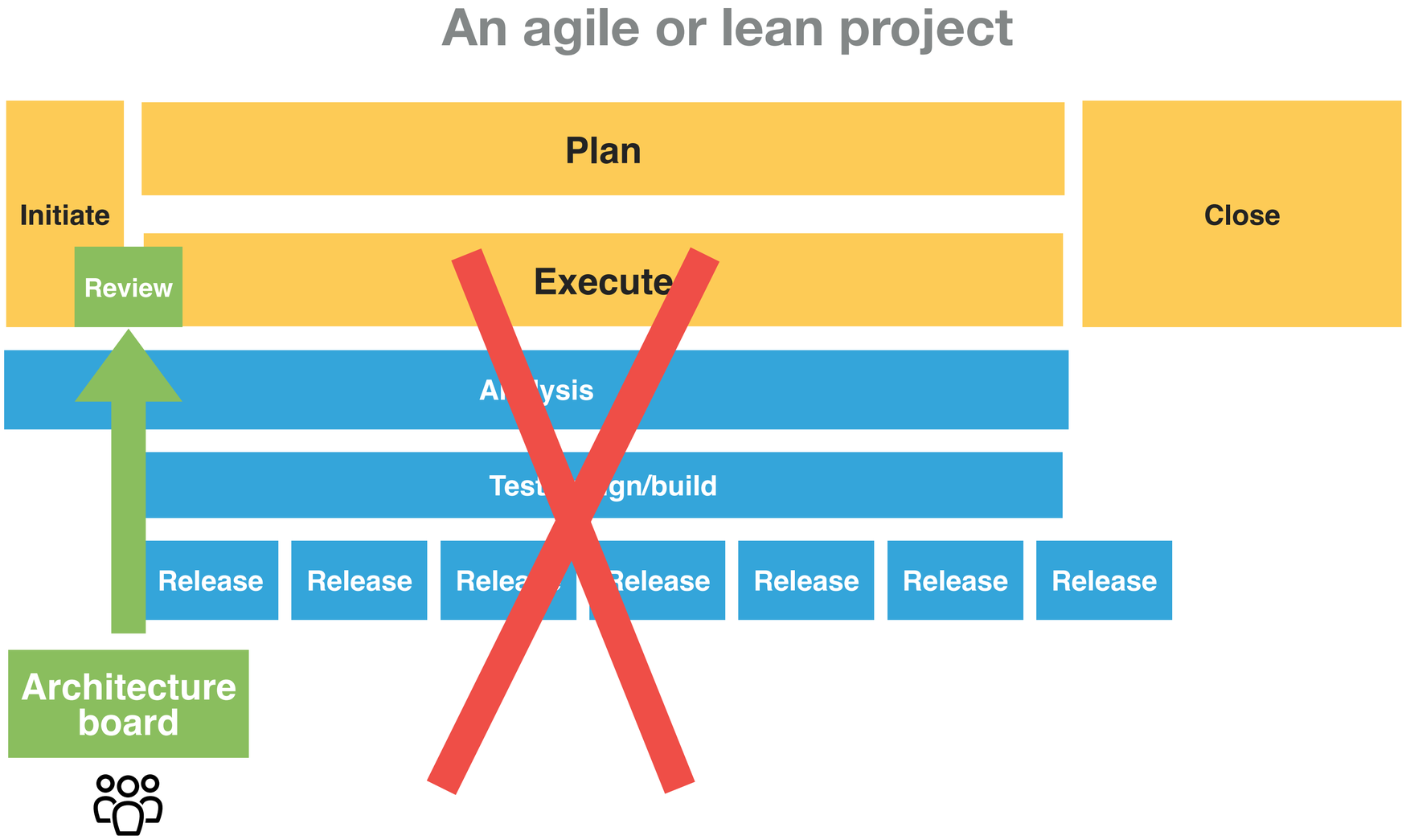
# Antipattern:
# Architecture Review Boards

# How we used to govern architecture

A project: waterfall / wagile / water-scrum-fall / “hybrid”

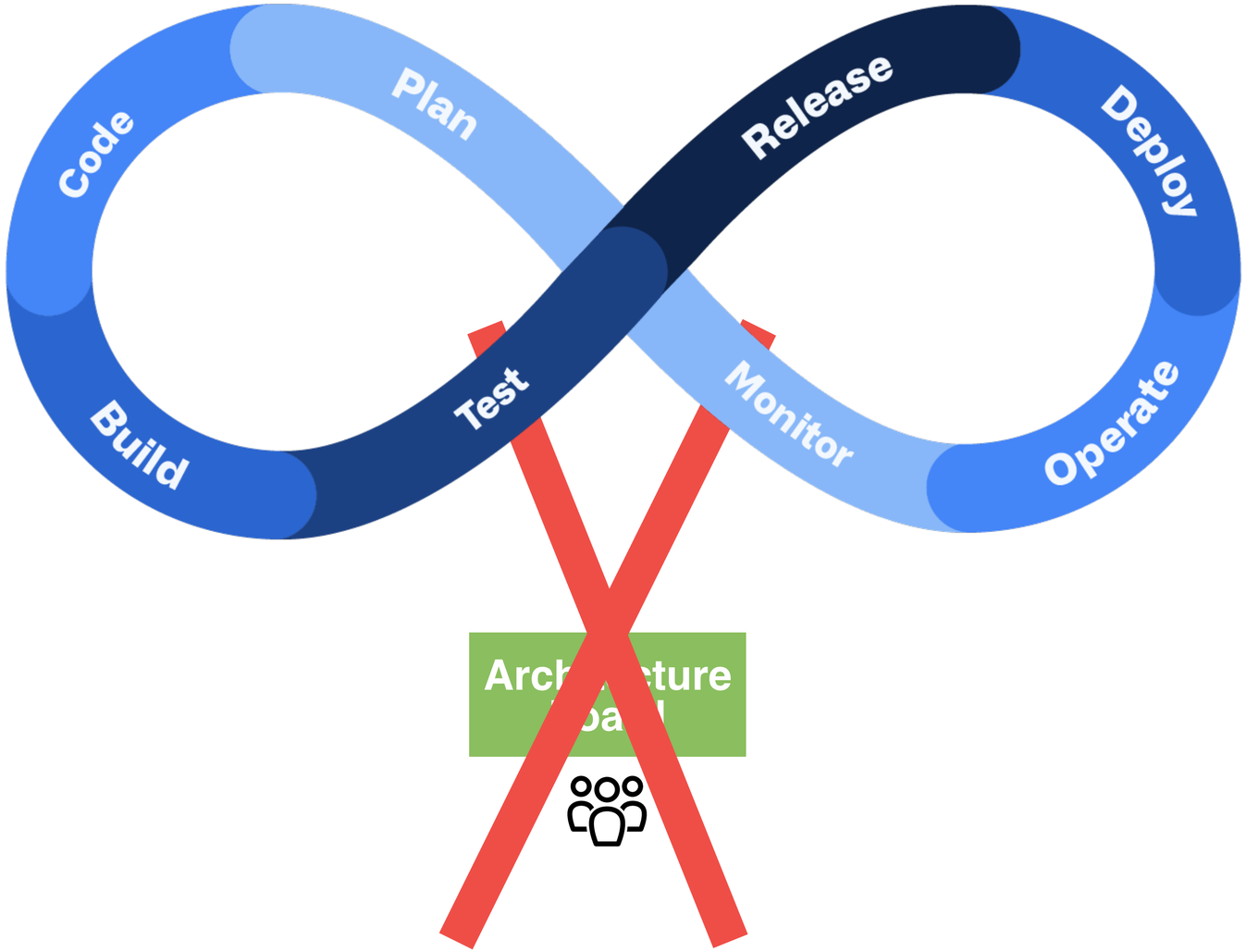


# How does this work with modern delivery?



# How does this work with modern delivery?

A product



# Pattern:
# Black box architecture governance

# Govern realised architectures, not (just) designs on paper

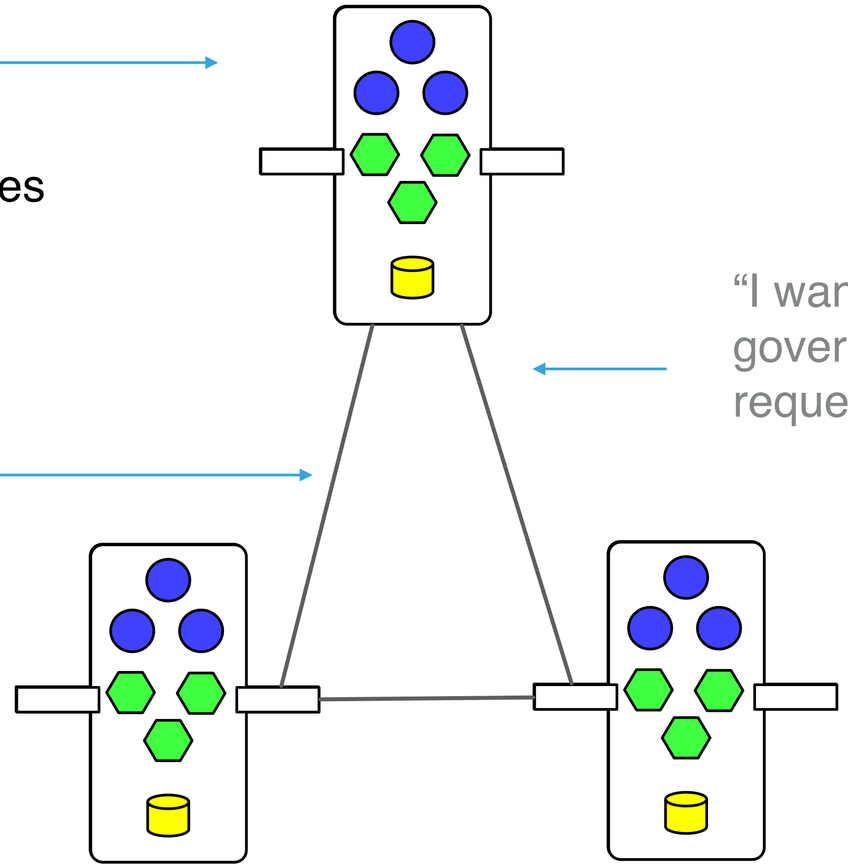
"I think I need an new service"

EA approve new services, new namespaces in Kubernetes (etc.)

"I think my service needs to access data or functionality from another service"

EA approve:

- Service-to-service IAM
- Firewalls



"I want an exception to automated governance - e.g. synchronous request/response"

EA approve:

- Exceptions
- Tech debt

Enterprise Architecture as GitOps

# Pattern:
# Platform Engineering as automated governance

Enterprise architecture was often too high-handed and bureaucratic to support rapid innovation

But sound architectural principles and Agile processes aren’t mutually exclusive

“In a **properly executed platform strategy**, major architecture and security checks **have happened**,”

“That 50-item checklist diminishes by 90%.”

Charles Betz, 2024

https://www.ciodive.com/news/enterprise-cios-elevate-architecture-skills-tech-debt-sprawl/722696/

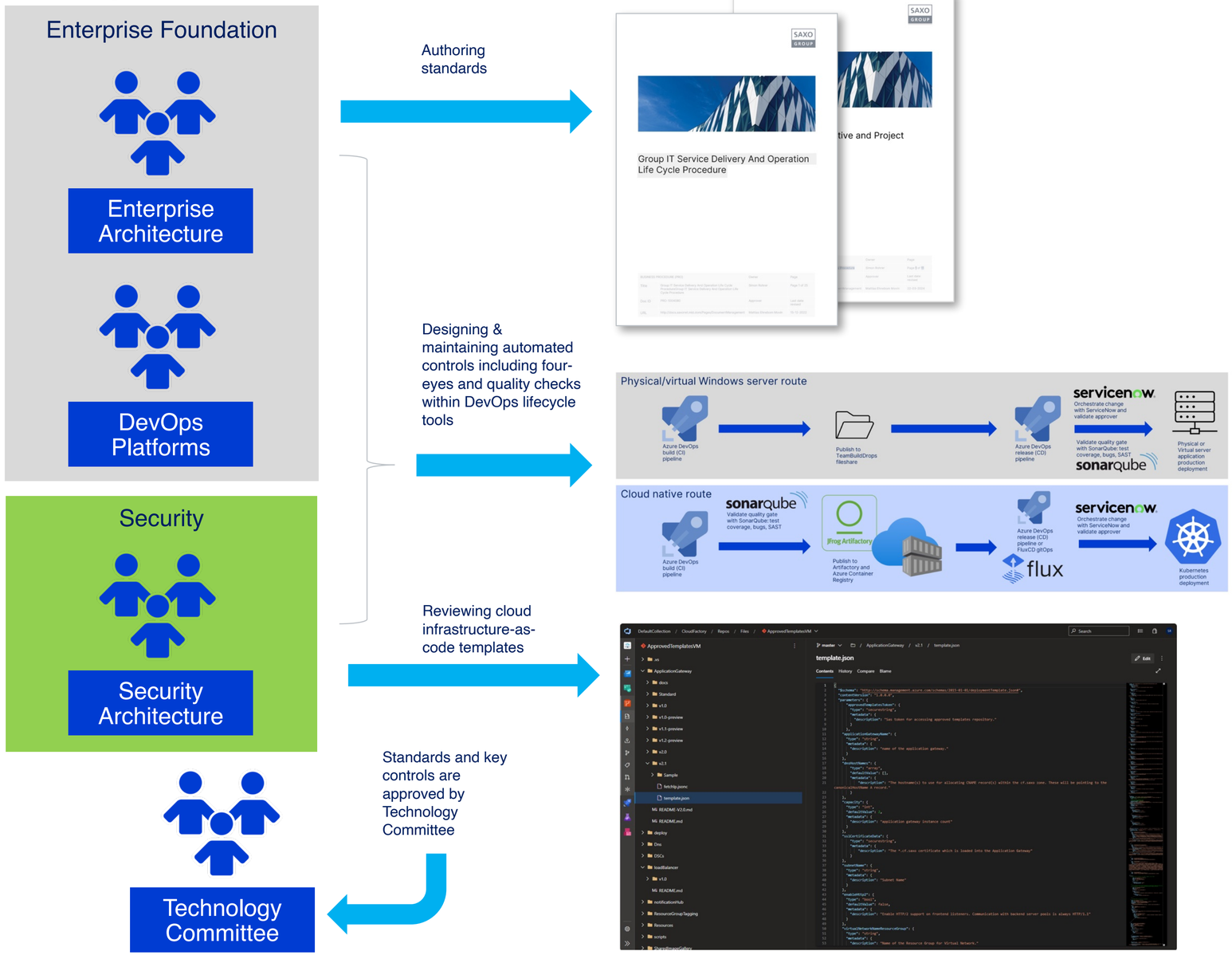
> The path forward is a two-way street, with stakeholders accepting a level of standardization and architects not standing in the way of progress.

Charles Betz, 2024

https://www.ciodive.com/news/enterprise-cios-elevate-architecture-skills-tech-debt-sprawl/722696/

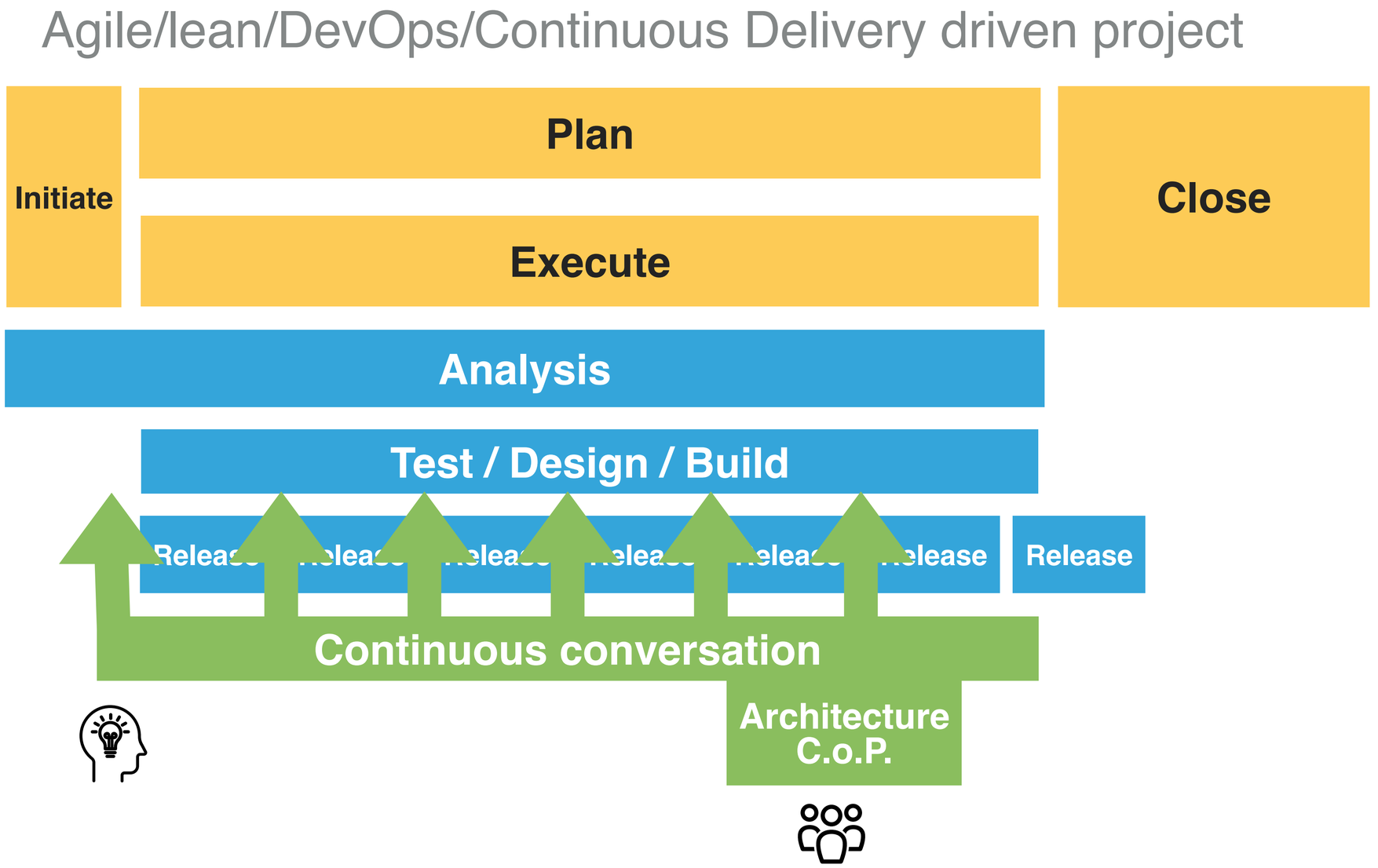
# IT Governance roles & responsibilities at Saxo Bank

- Key governance responsibilities sit under **Enterprise Architecture** and **DevOps Platforms** departments in **Enterprise Foundation**
- Primary standards for the IT delivery lifecycle are owned and developed by **Enterprise Architecture**
- **Enterprise Architecture** and **DevOps Platforms** collaborate on automated controls and their setup and implementation within automated build and release pipelines
- **Security Architecture** contribute to designing security rules within pipelines and approve cloud infrastructure-as-code templates
- Oversight for these standards and policies lives with Technology Committee



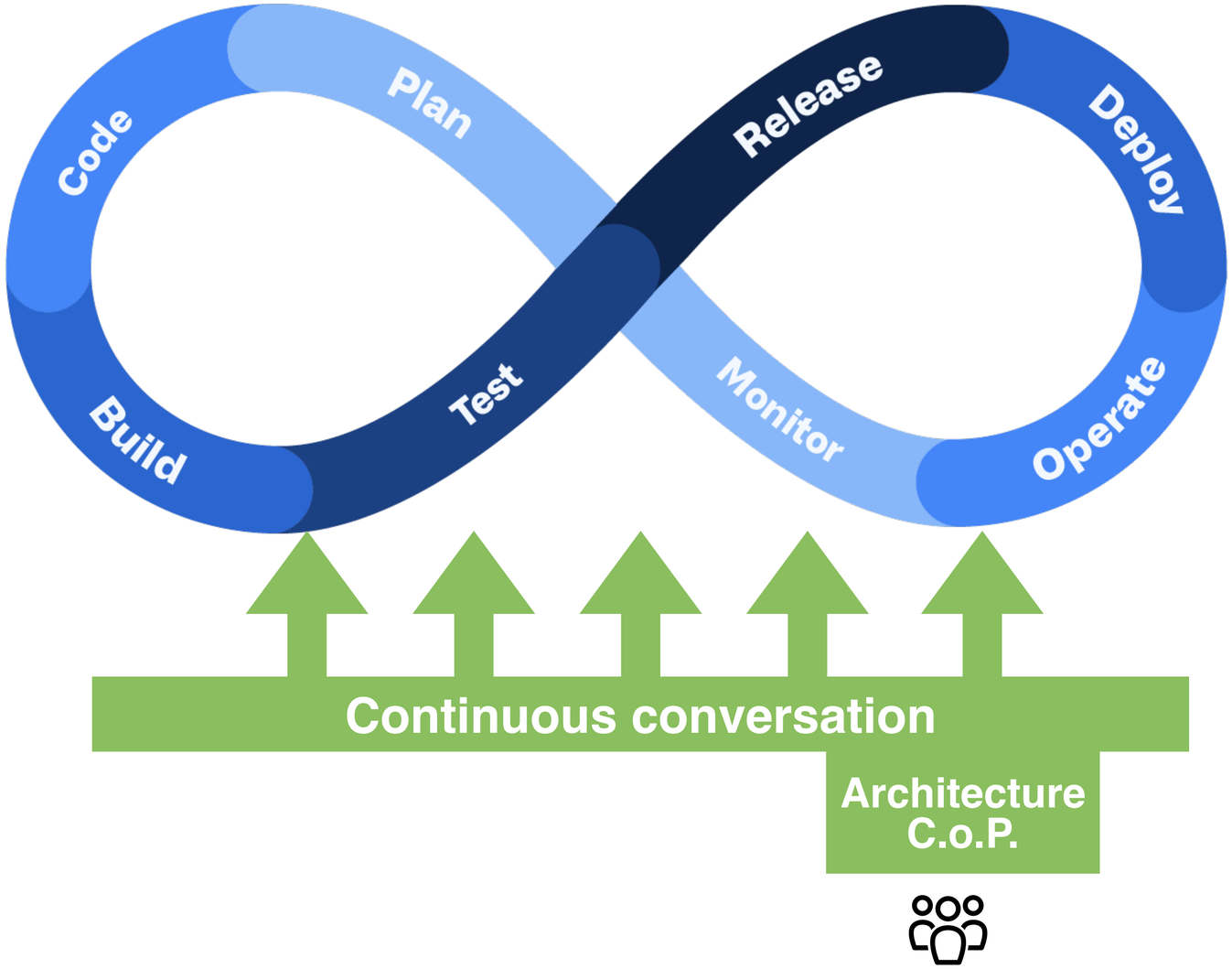
# Pattern:
# Continuous conversational “governance”

# Continuous conversational governance

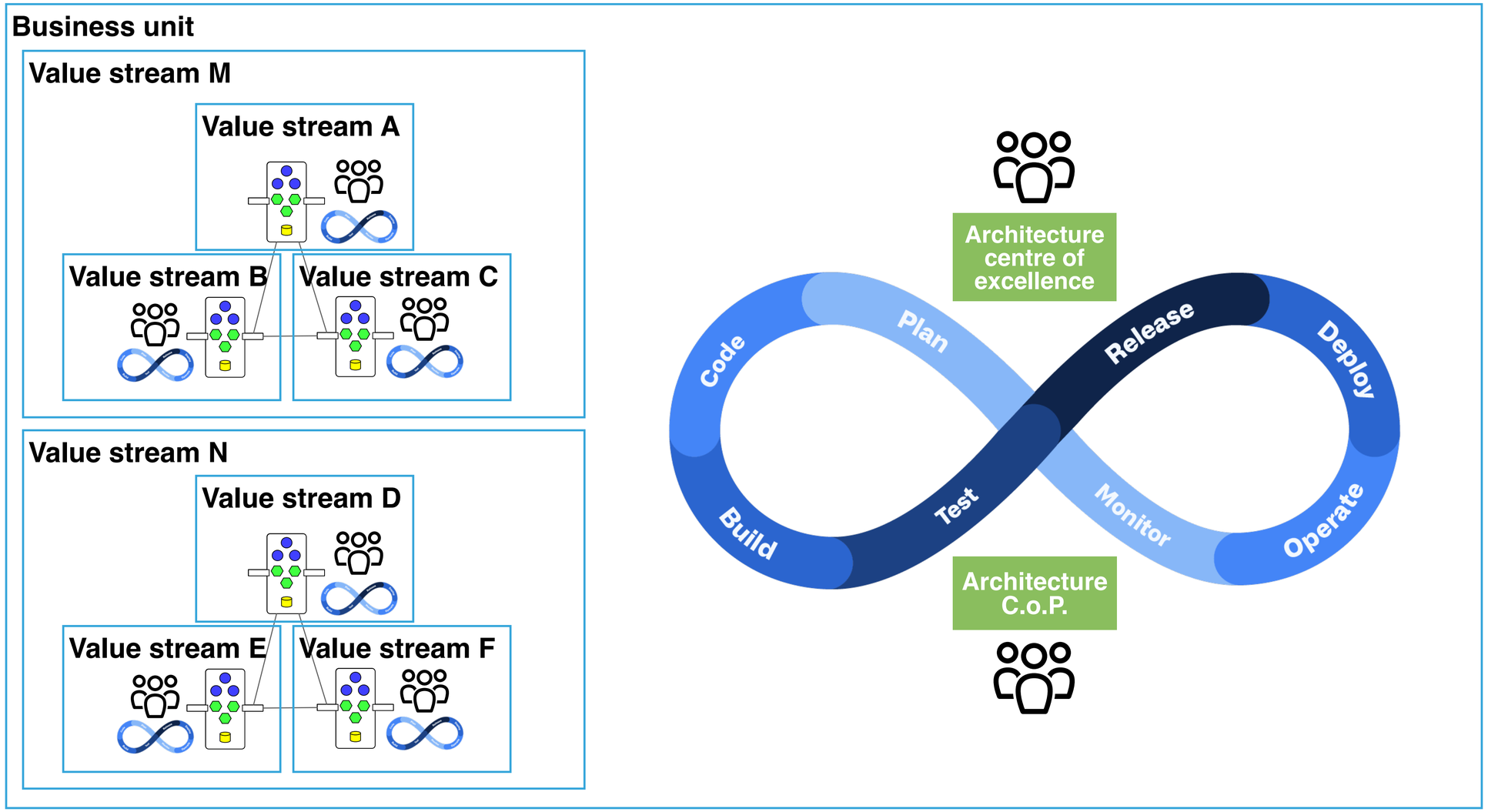


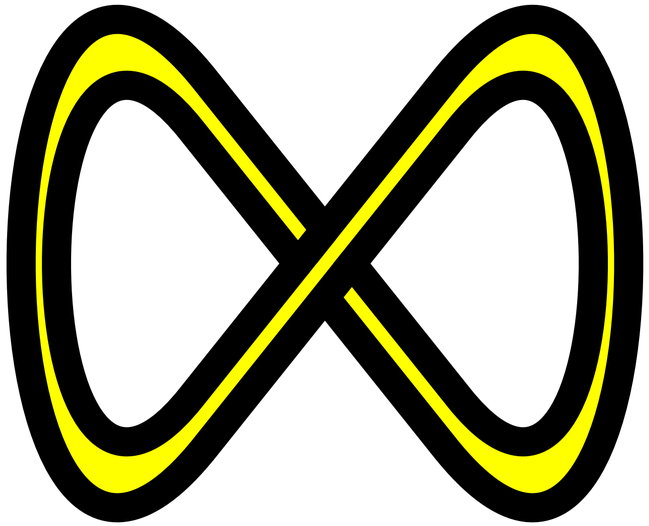
# Continuous conversational governance

A single product



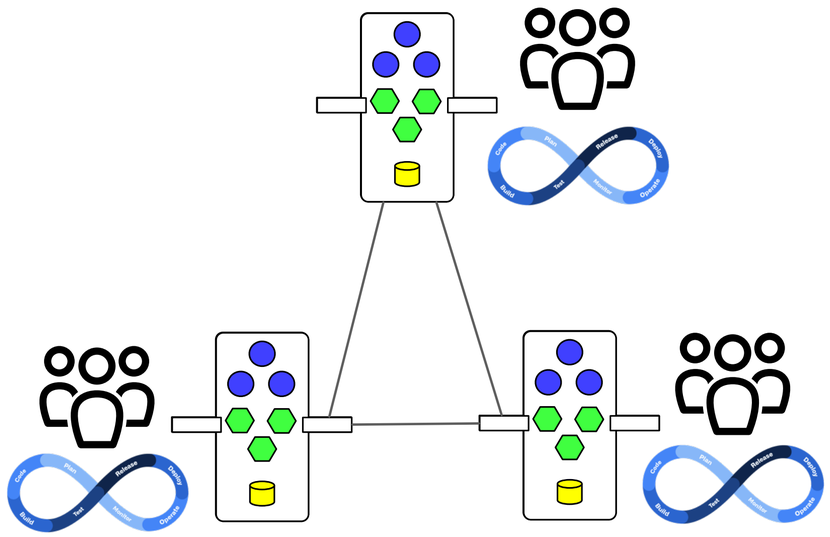
# Continuous conversational governance across value streams

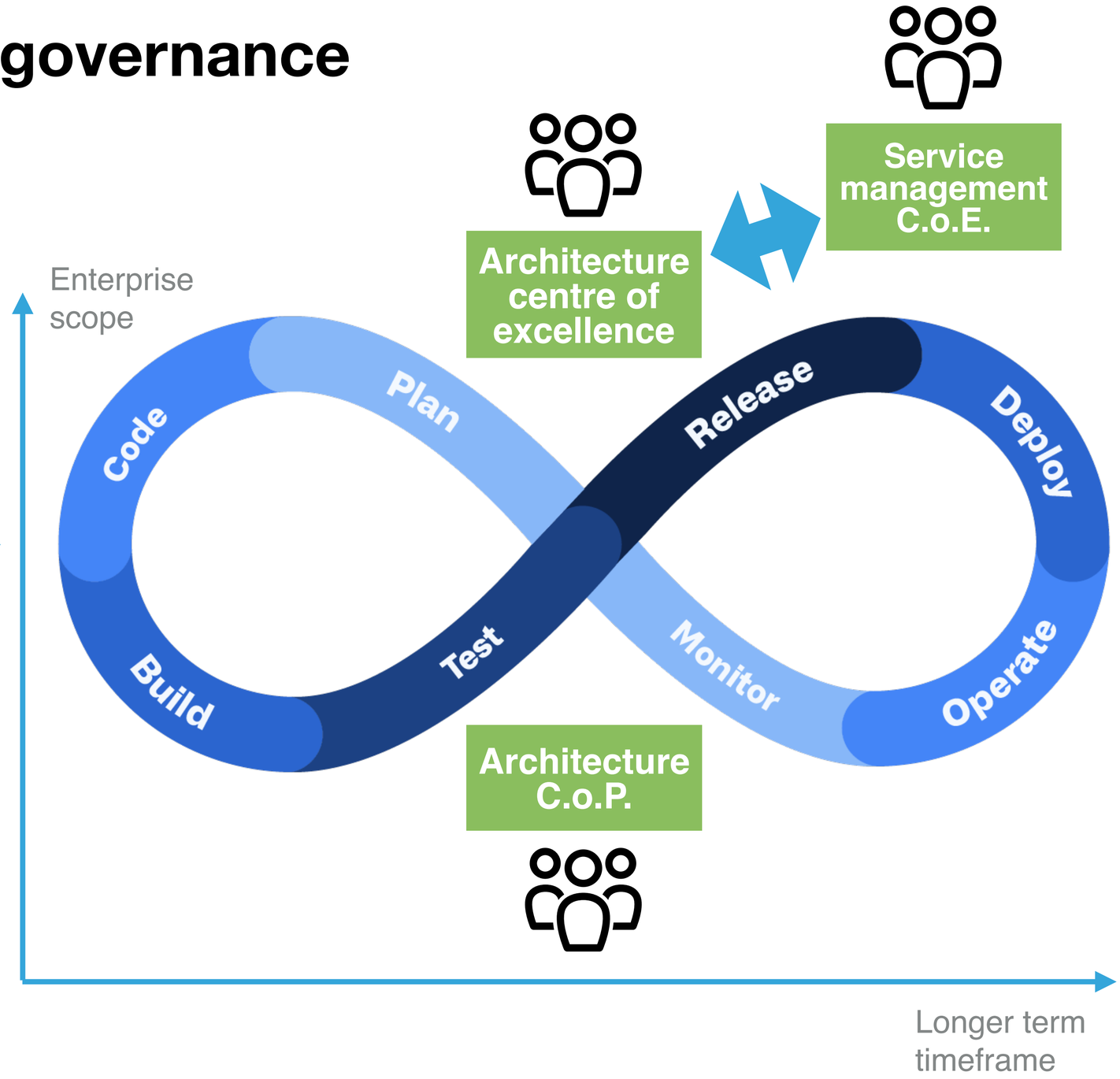




# DevOps at Enterprise Scale

# Continuous conversational governance

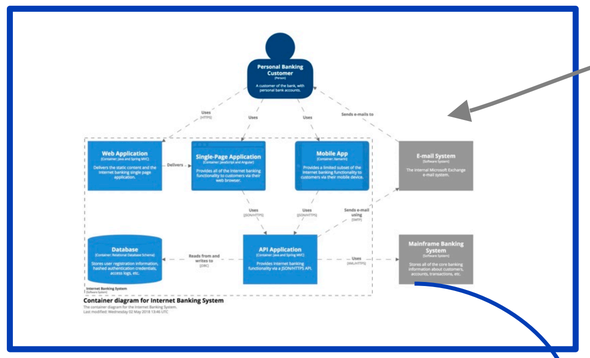


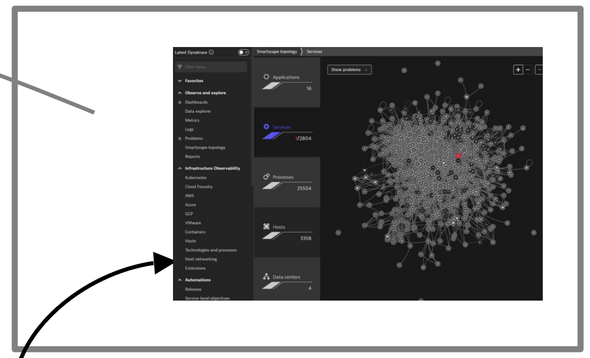


# Continuous conversational governance (after Ruth Malan)

## Intention

## Reflection



Whiteboard sessions

Model storming

Continuous refactoring

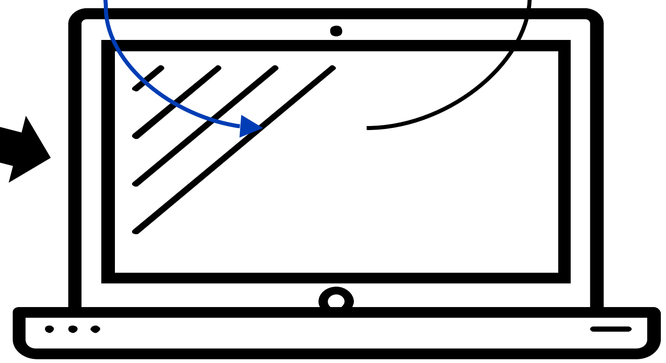
Test-driven design

Architecture decision records

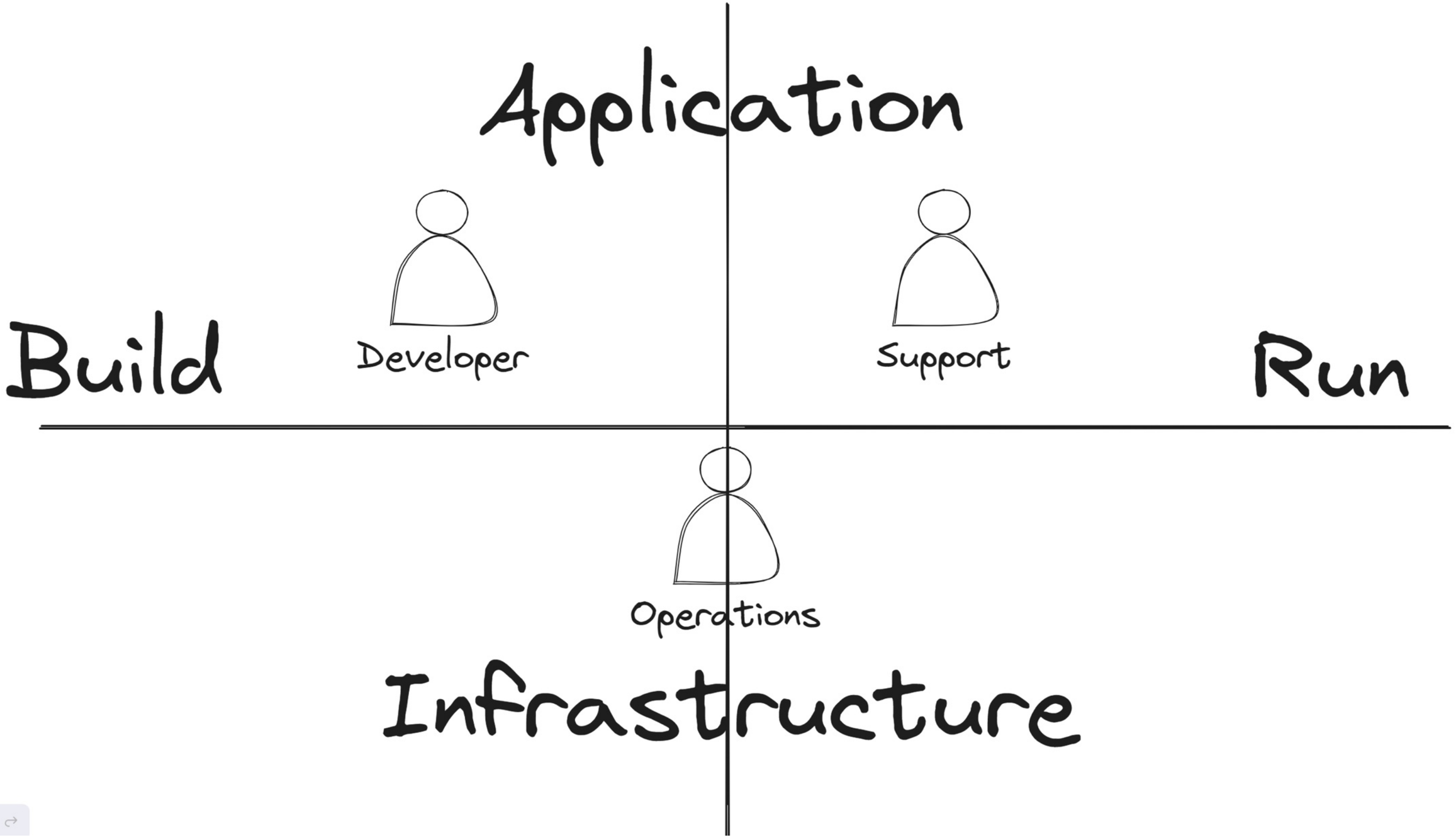
Emergent design

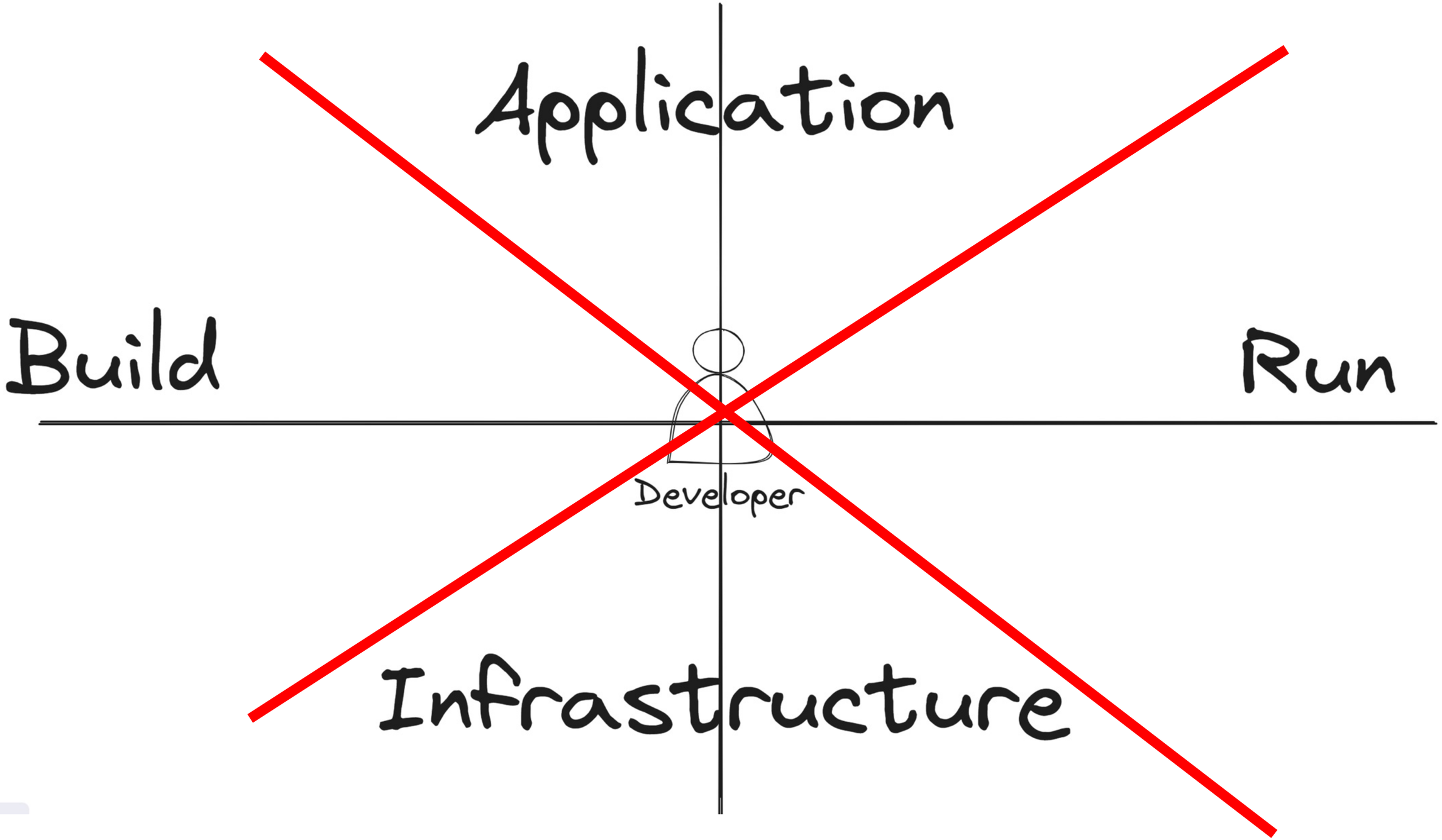
Discoverability & observability tooling

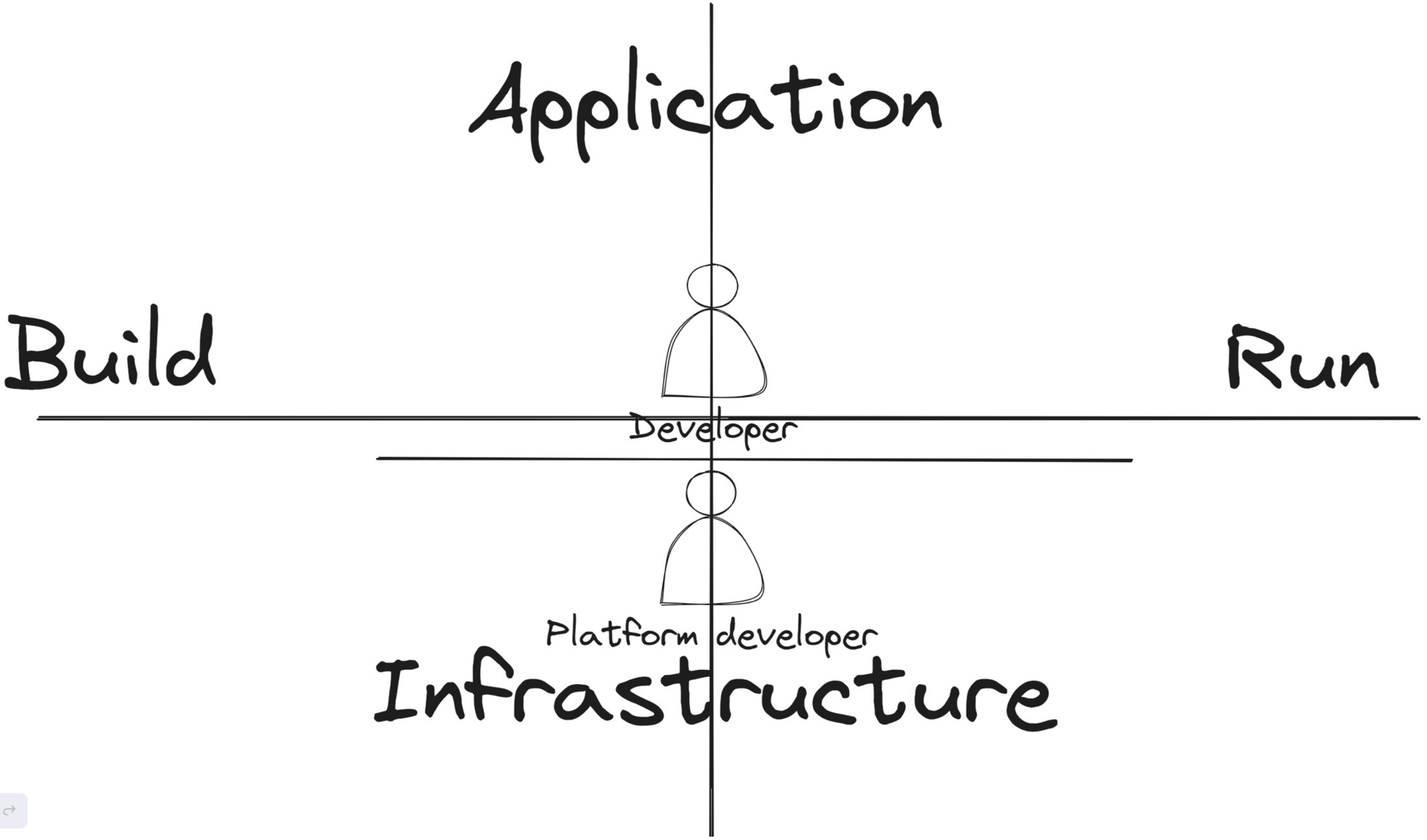
Infrastructure-as-code

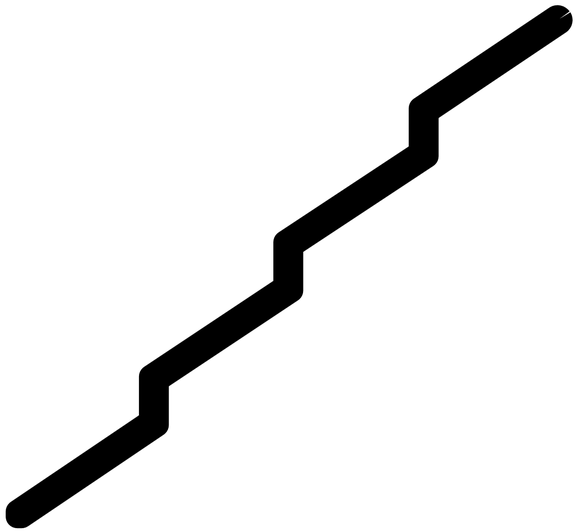


**Working code deployed as running system**


Application
Build
Developer
Support
Run
Operations
Infrastructure


Application
Build
Run
Developer
Infrastructure


Application
Build
Run
Developer
Platform developer
Infrastructure



# E

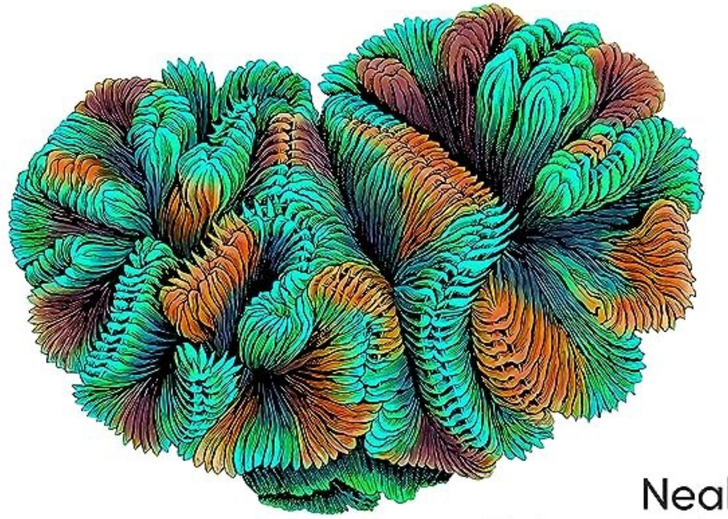
Curating the
**E**volution of
Enterprise
Architecture and
Organisation

O'REILLY®

2nd Edition

# Building Evolutionary Architectures

## Automated Software Governance



Neal Ford,
Rebecca Parsons,
Patrick Kua & Pramod Sadalage

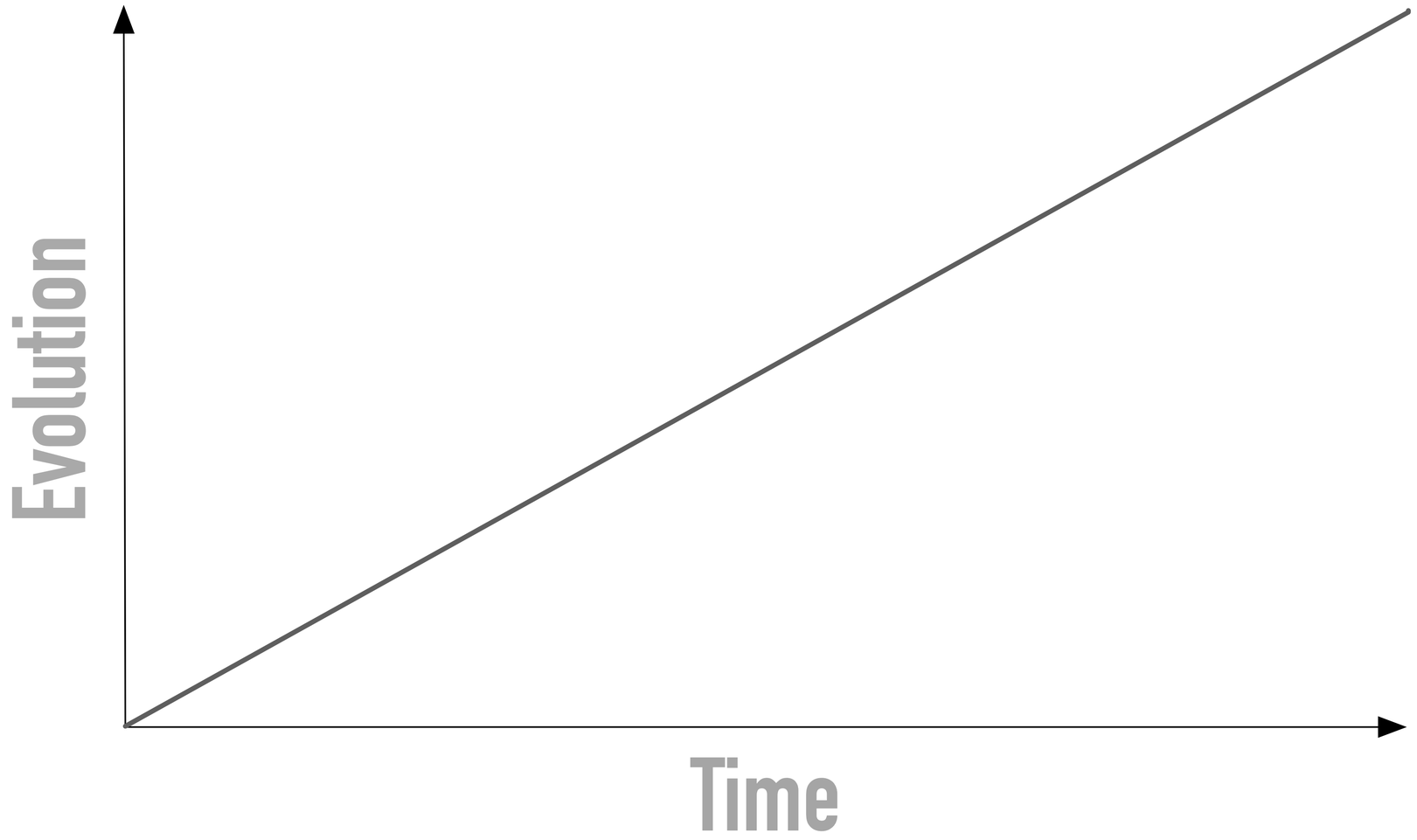
Forewords by Mark Richards & Martin Fowler

# Evolutionary architecture fitness functions at enterprise level (“macro architecture”)

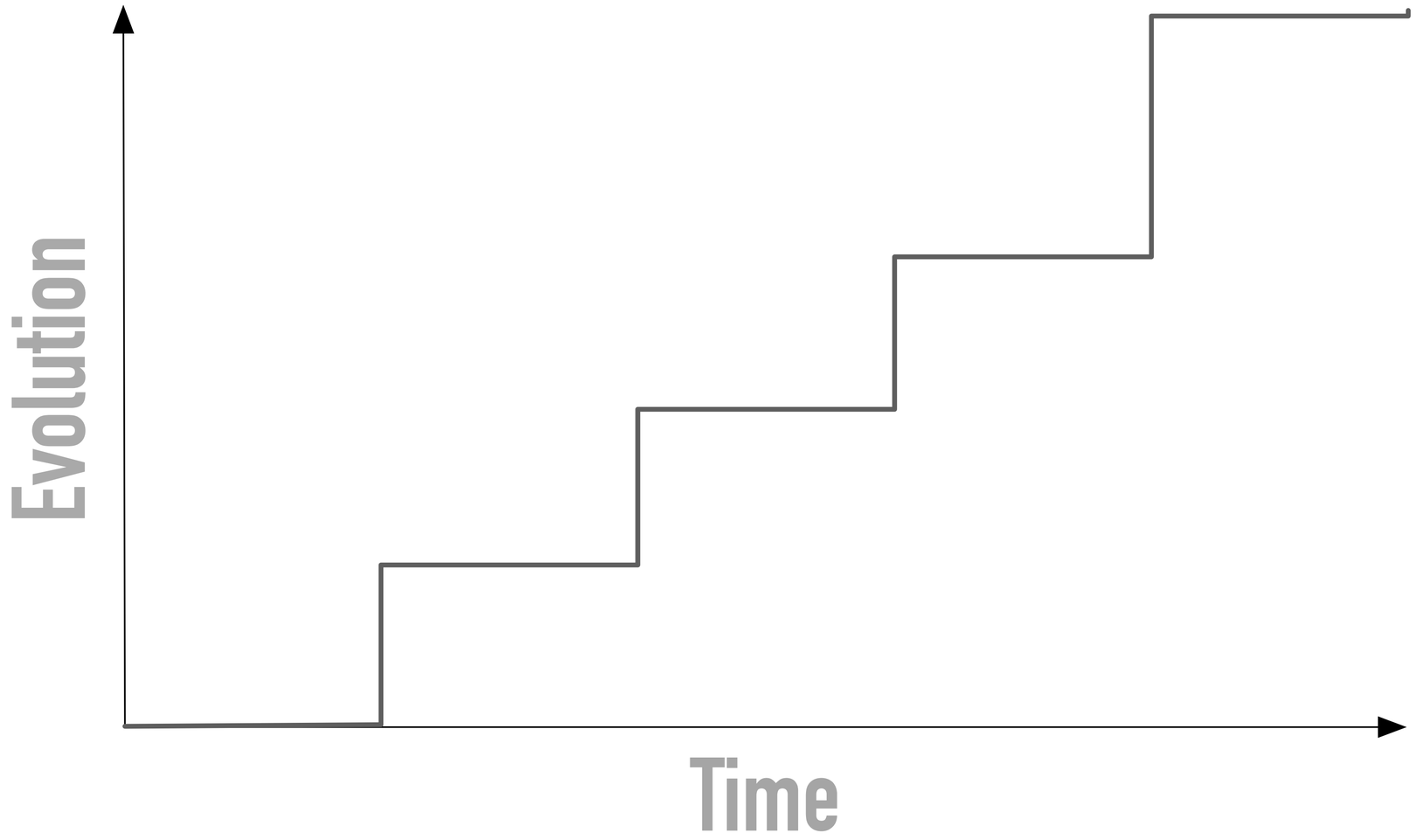
- Number of 2-pizza teams required to implement a typical business feature - sociotechnical coupling & cohesion

- Asynchronous connections ÷ synchronous connections

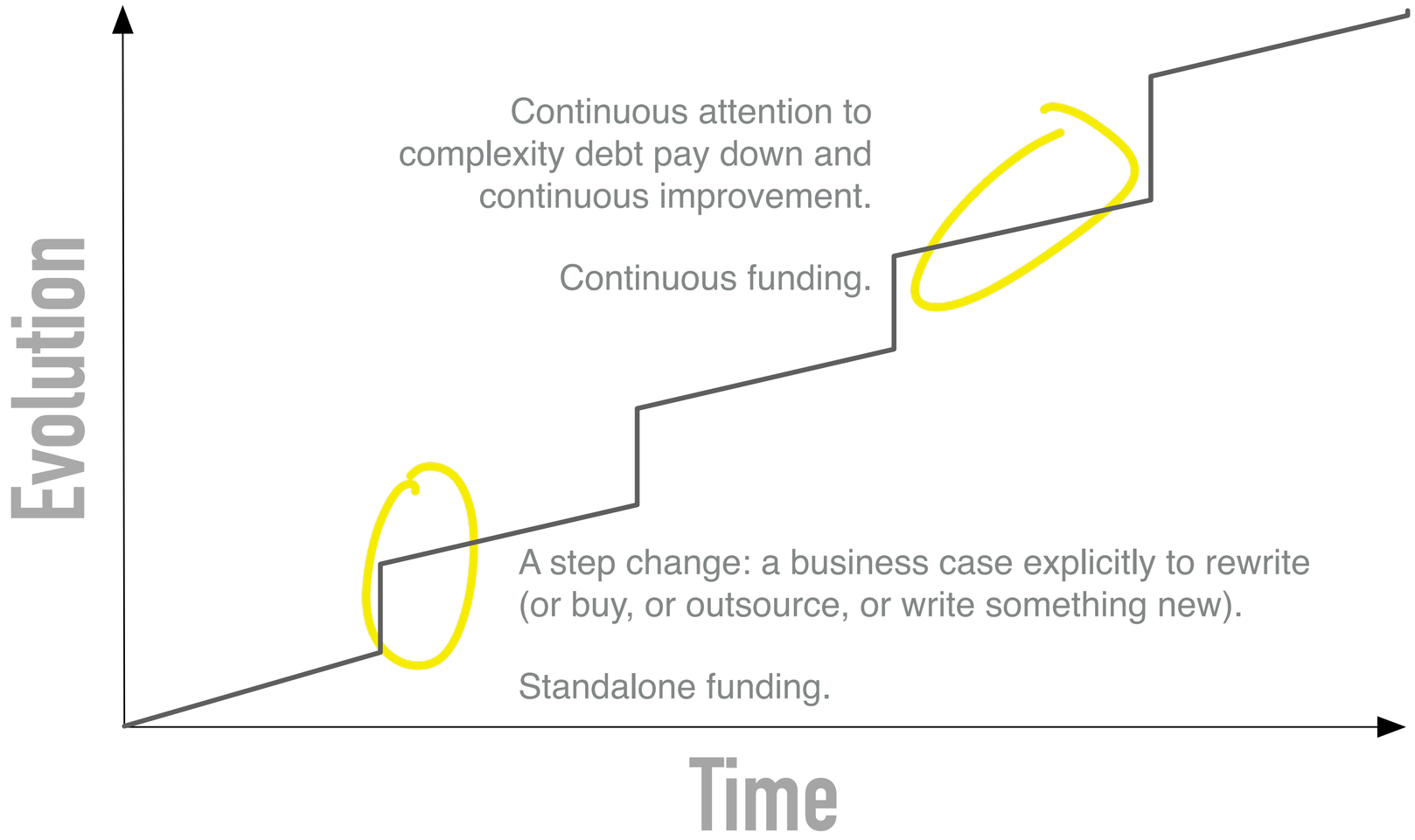
# Theory #1: gradual evolution



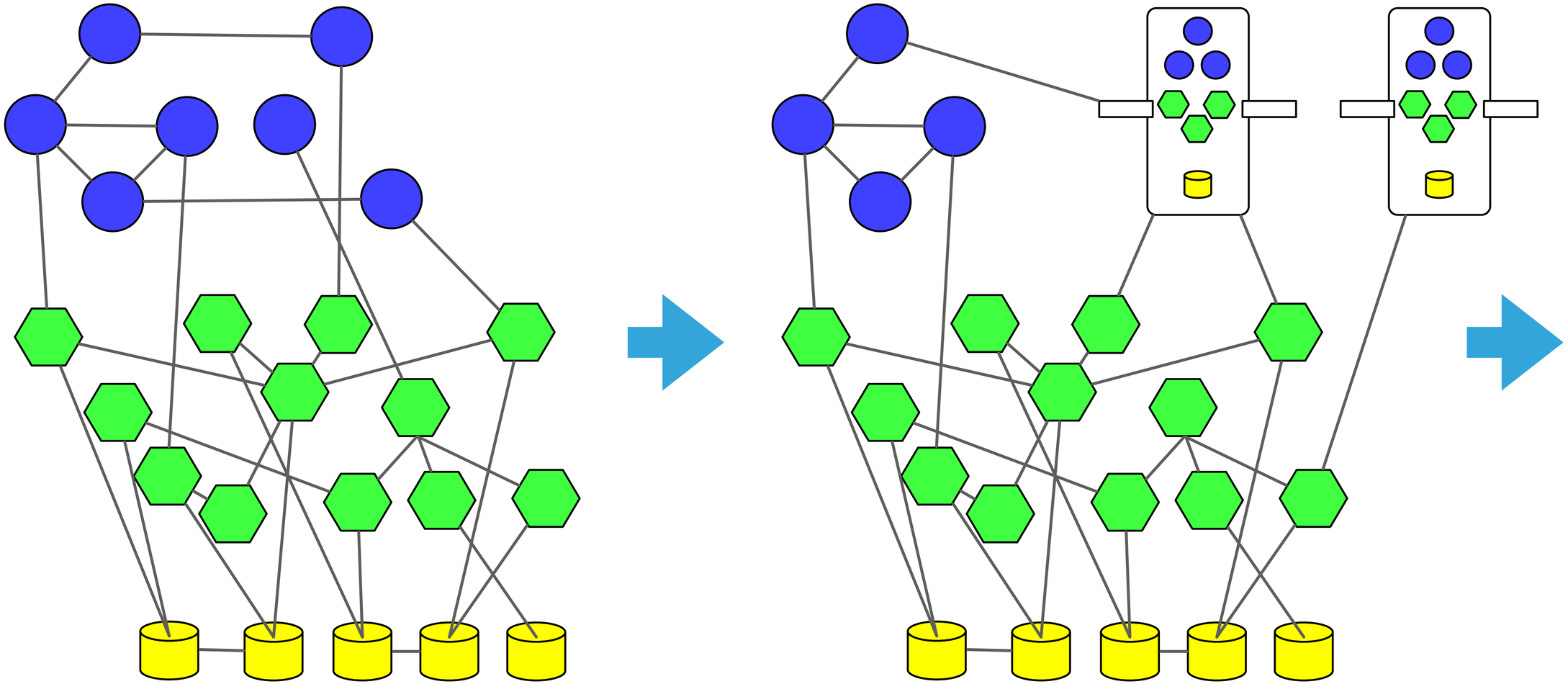
# Theory #2: punctuated equilibria



# Reconciled: punctuated gradualism




Strangler fig pattern

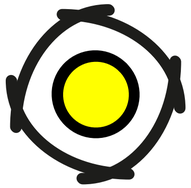


# Modern Enterprise Architecture

**A**

**A**ligning Value, People and Technology

- Conway’s law ++ and reverse Conway manoeuvre
- Removing complexity and increasing autonomy
- Enterprise architecture as politics and diplomacy



**B**

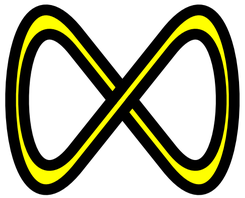
Architecting for Outcomes: **B**etter Value, Sooner, Safer, Happier

- Outcomes of quality, flow, safety to balance pure “value”
- Happier customers, colleagues, citizens & climate
- Continuous improvement - be the best at being better

**C**

Governing **C**ontinuously: Conversational and Automated

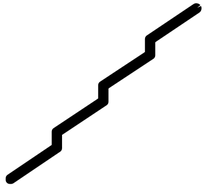
- A stream of continuous architecture decisions
- Moving decisions down into the right level of the org
- Automating common top-level system policies



**D**

Practicing **D**evOps at Enterprise Scale

- Learning by observing: intention & reflection
- Humility in architecture: all decisions are contingent



**E**

Curating the **E**volution of Enterprise Architecture and Organisation

- Fitness functions for socio-technical architecture at scale
- Evolving fast and slow

# Thank you!

@sirohrer