Systemic Event Discovery Approach

Driving Software Transformation Through Domain-Driven Design & Systems Thinking

March 2025
Munich, Germany

Masoud Chelongar
Hands-on Software Architect

Overview

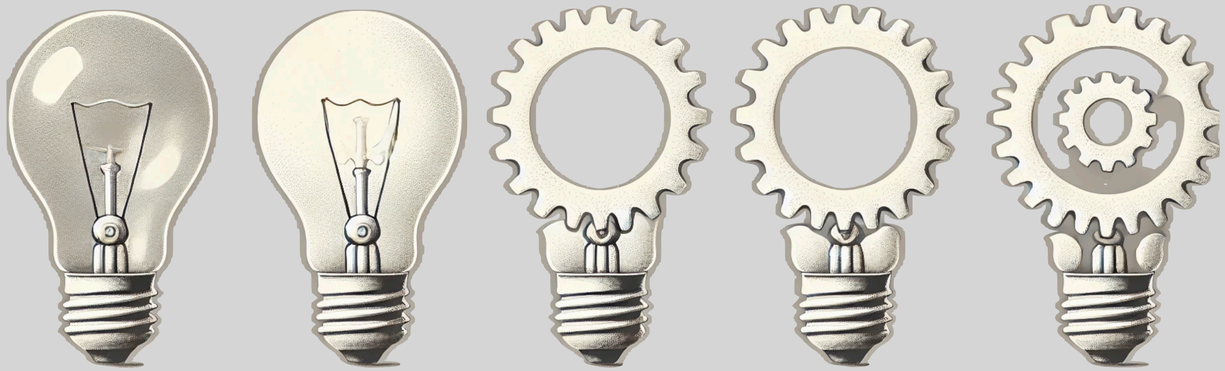
Problem Space (Sociotechnical Systems)

Tools & Methodologies

SEDA Framework

Problem Space
(Sociotechnical Systems)

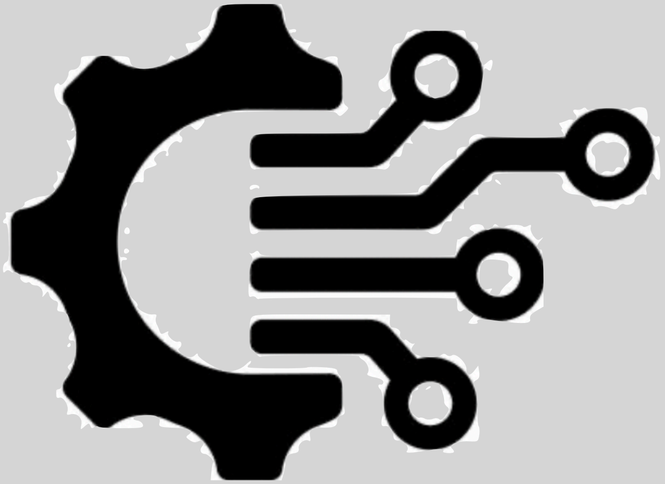
Modernizing
Legacy Ecosystem



Aligning
The Software With Business Goals & Requirements

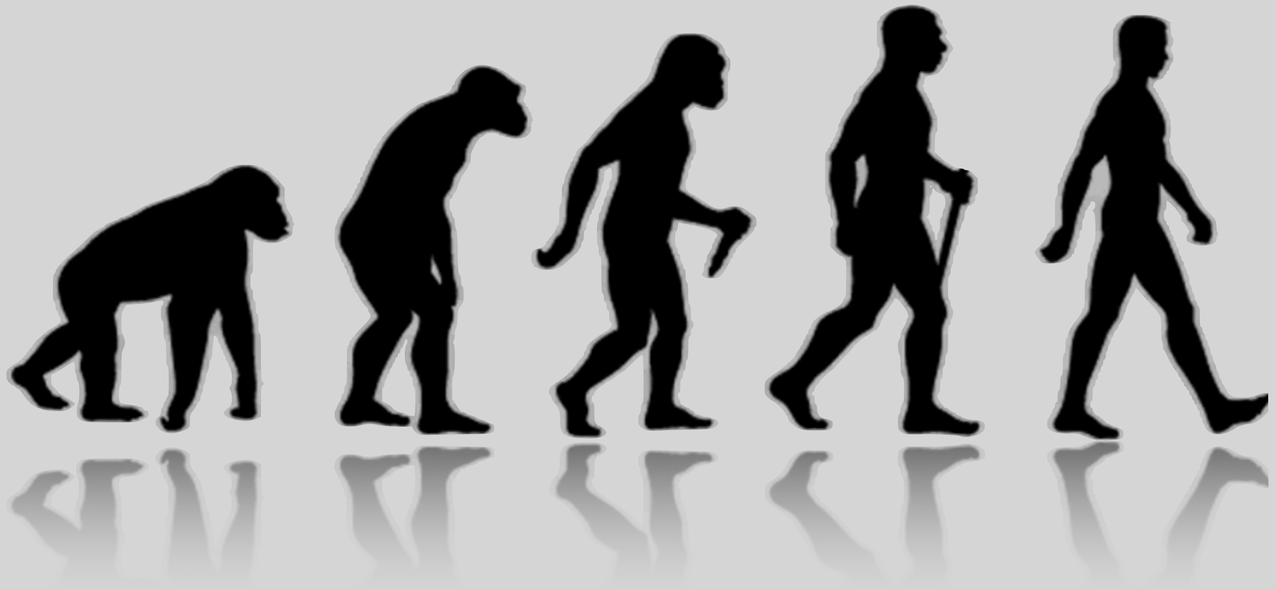
https://kit8.net/freebies/business-icons-2/target-and-arrow-icon-goal-achievement/

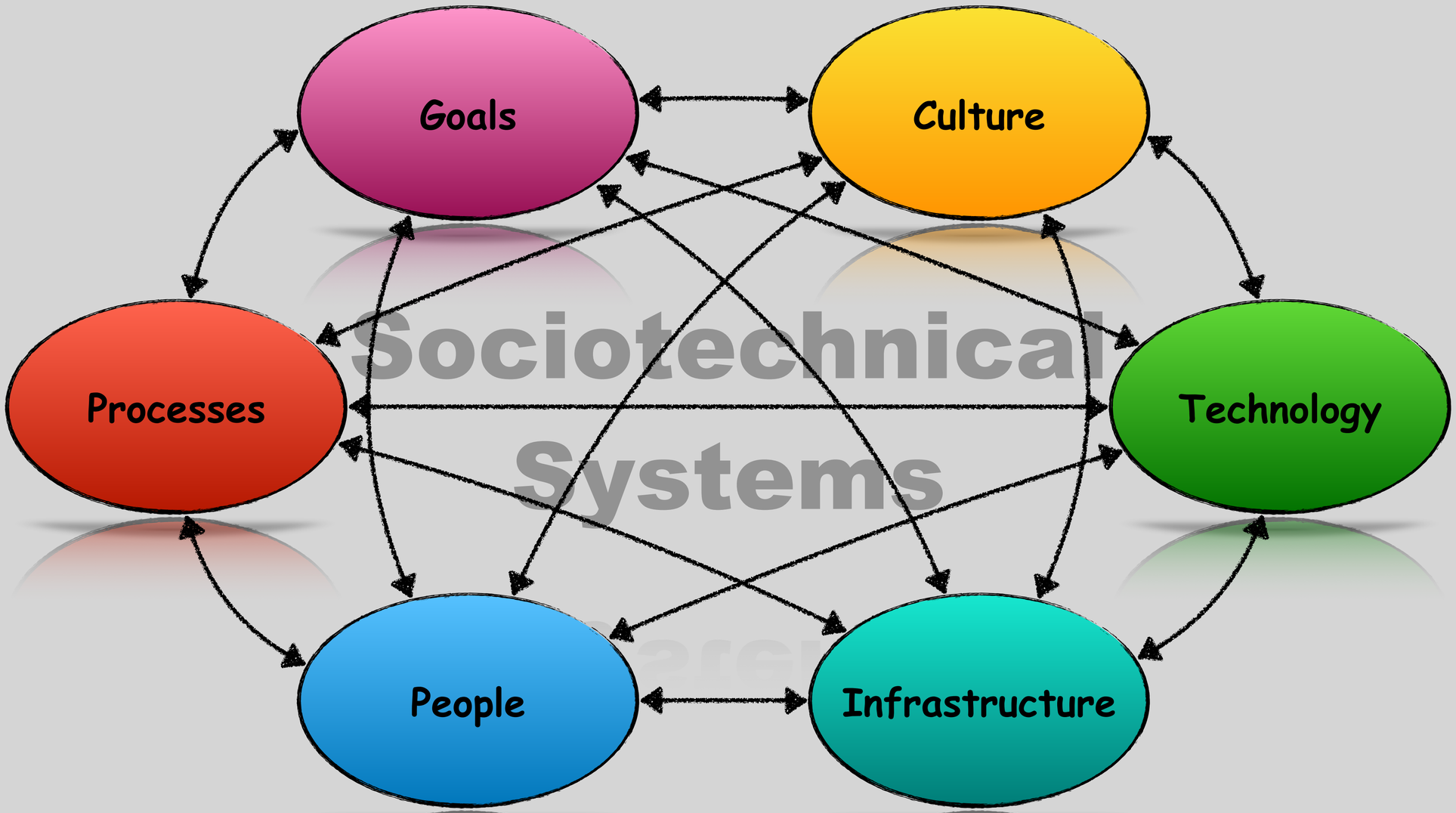
Embracing Digital Transformation Practices

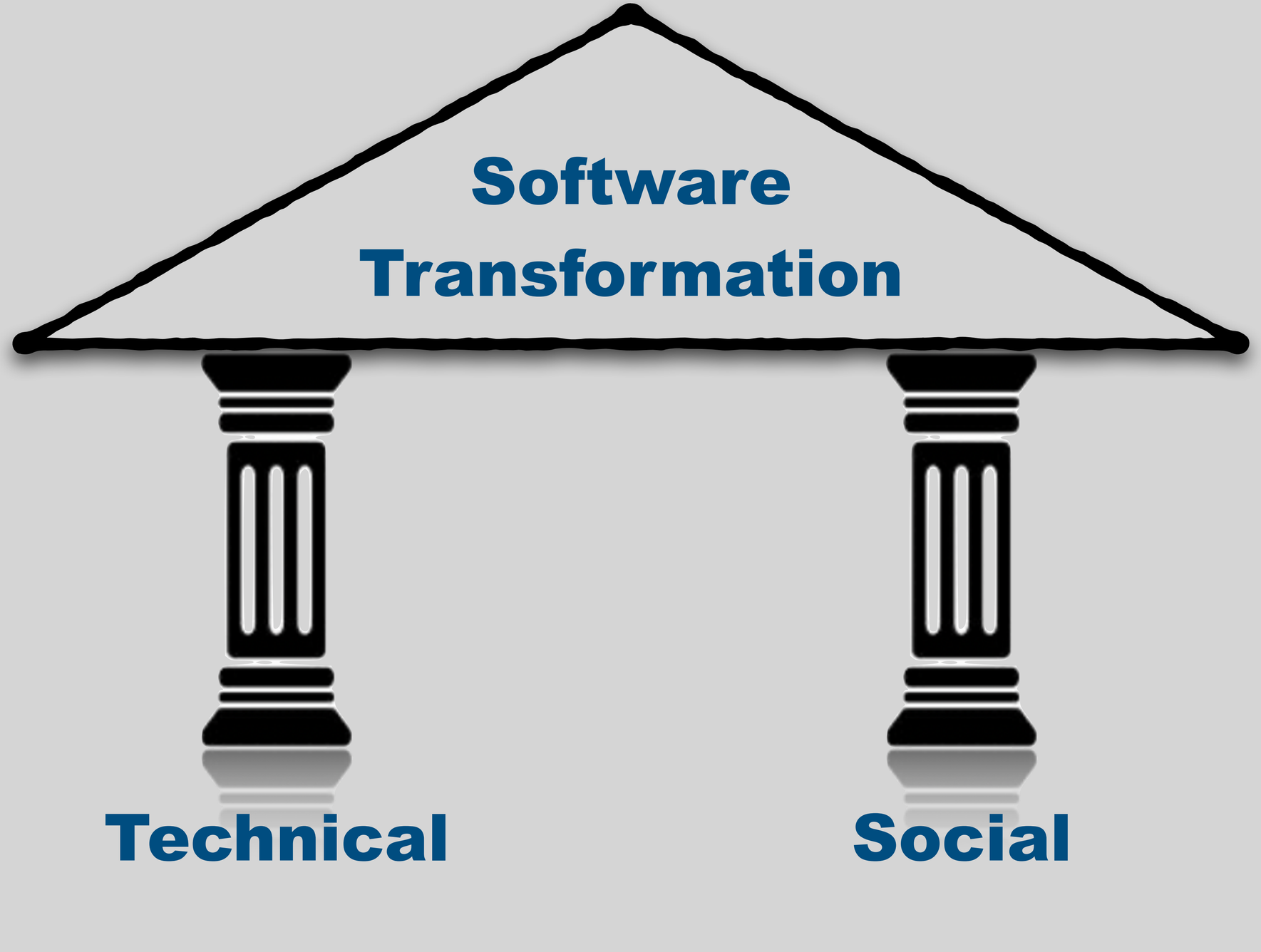


https://icons-for-free.com/communication+digital+digital+world+technology+world+icon-1320190779839110503/

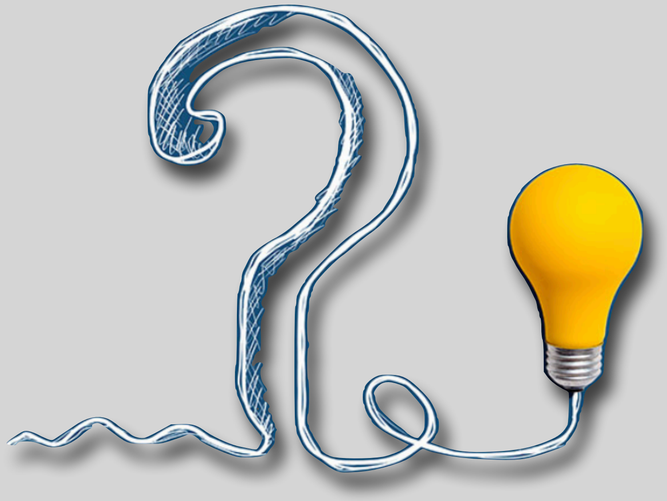
Failing to act evolutionary compels revolutionary action.




Goals
Culture
Processes
Technology
People
Infrastructure
Sociotechnical Systems


Software
Transformation
Technical
Social

Tools & Methodologies

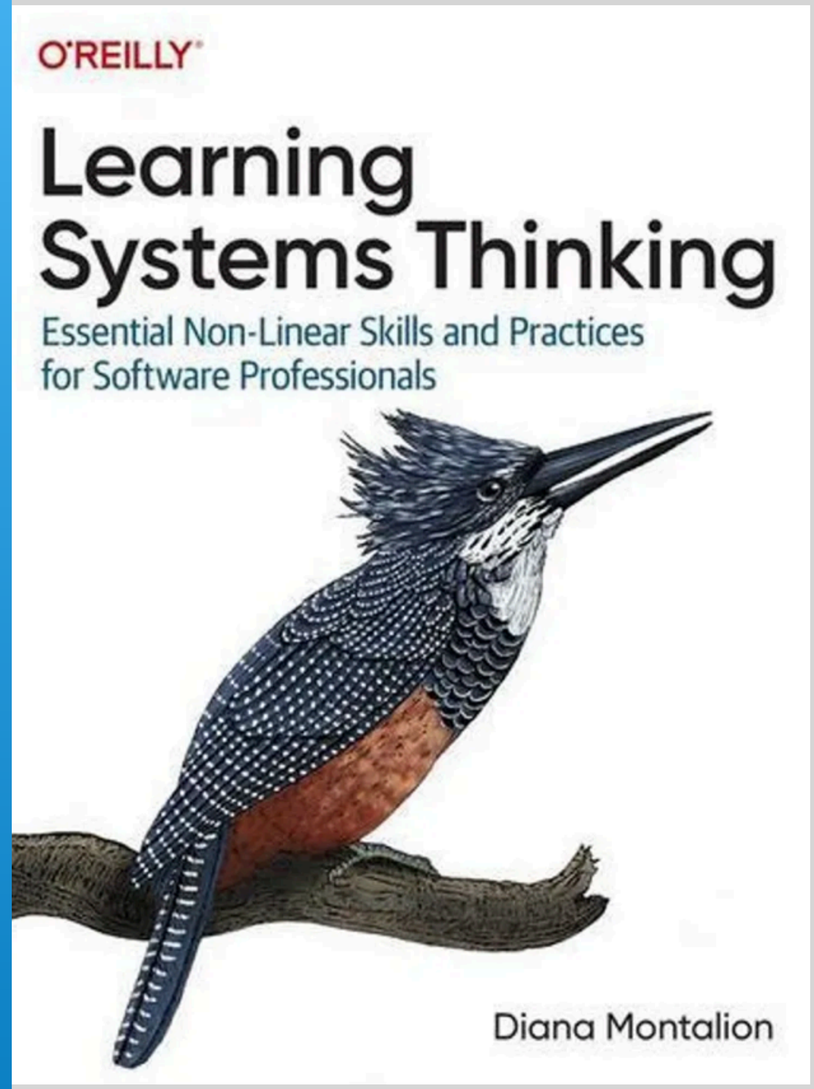


Cross-Disciplinary Thinking

- From Reactive Problem-Solving to Proactive Problem-Solving
- Thinking Critically and Creatively
- Promote Transparency and Cultivate a Collaborative Mindset

Systems Thinking expands our toolsets as knowledge workers. It steps us outside the constant, pointless culture war about architecture versus engineering as a practice.

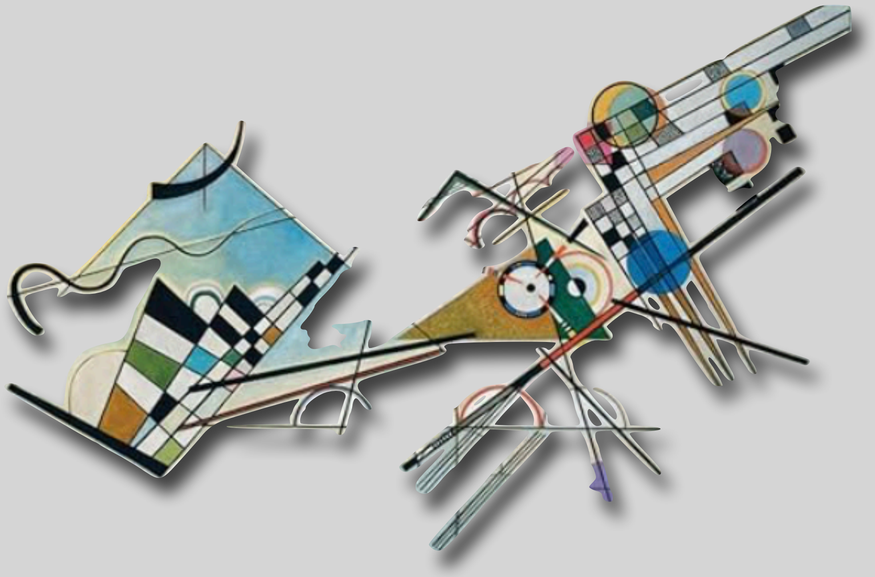
—Diana Montalion, Learning Systems Thinking



NonLinear Thinking

understanding how components in a system interact dynamically

- Iterative Processes
- Incremental Development
- Parallel Problem Solving
- User-Centric Development



Business-Centric Development

Business Alignment

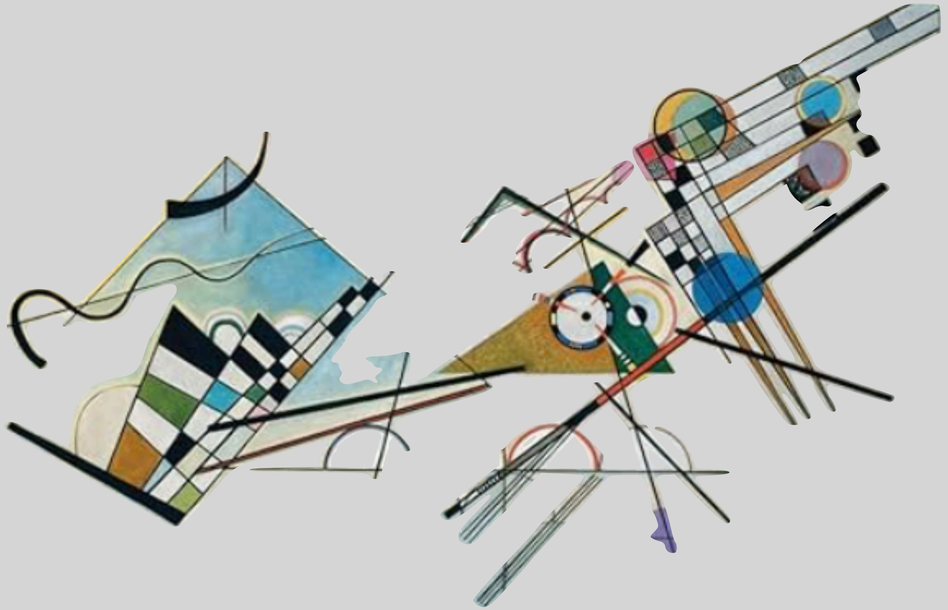
A successful software project is one where business goals are clear, and the development process is closely aligned with achieving those goals.

—Martin Fowlers, Thoughtworks

https://martinfowler.com/

Domain-Driven Design is both a way of thinking and a set of priorities, aimed at accelerating software projects that have to deal with complicated domains.

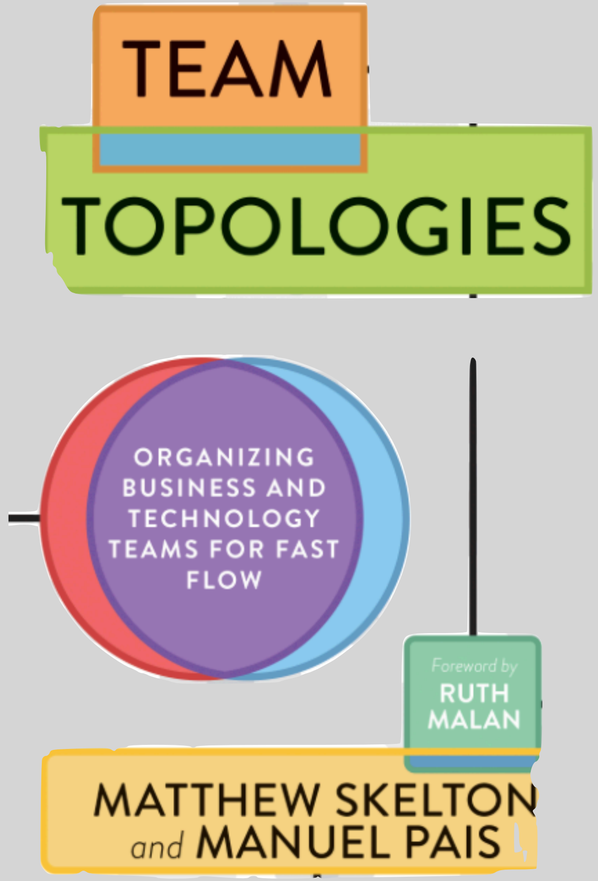
—Domain-Driven Design, Eric Evans



Team Dynamics

Team Topologies is an approach to designing team-of-teams organisations for fast flow of value.

—Manuel Pais & Matthew Skelton

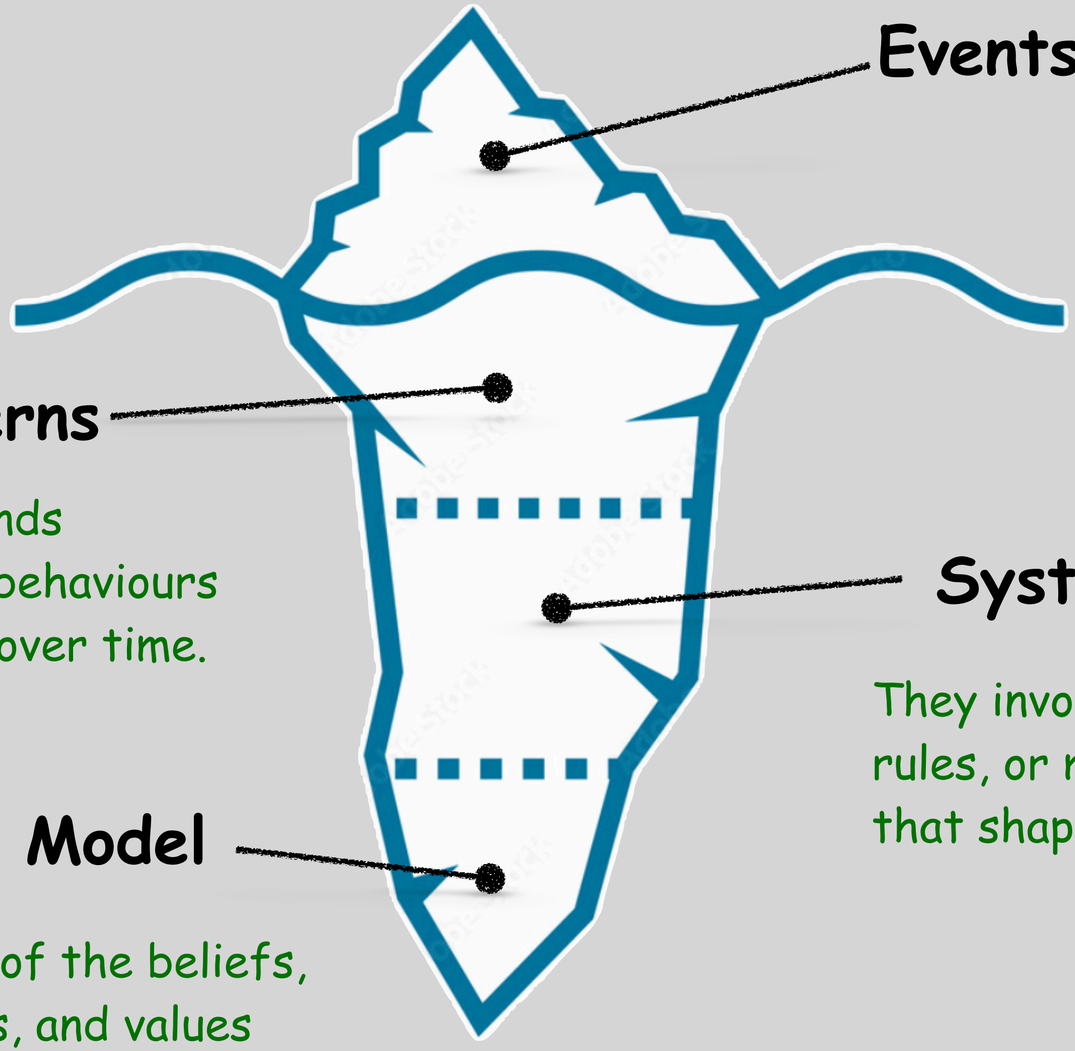


SEDA Framework

Step 1

Understanding Current State of System (Holistic Analysis)

System Exploration via Iceberg Model



Events

They are the immediate,
visible occurrences or
symptoms that we can observe.

Patterns

They are trends
or recurring behaviours
that emerge over time.

System Structure

They involves the underlying structures,
rules, or mechanisms
that shape the patterns we observe.

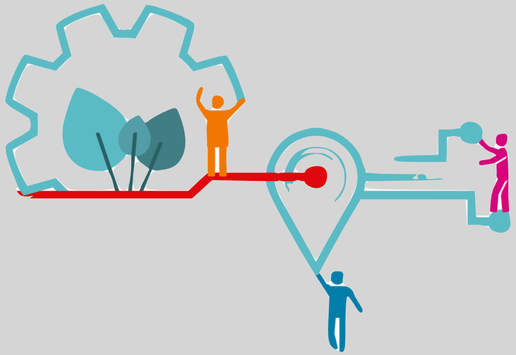
Mental Model

It consists of the beliefs,
assumptions, and values
that inform the structures within a system.

https://stock.adobe.com/th/images/iceberg-model-blank-outline-diagram-clipart-image-isolated-on-white-background/353437078

Step 1

Understanding Current State of System (Holistic Analysis)

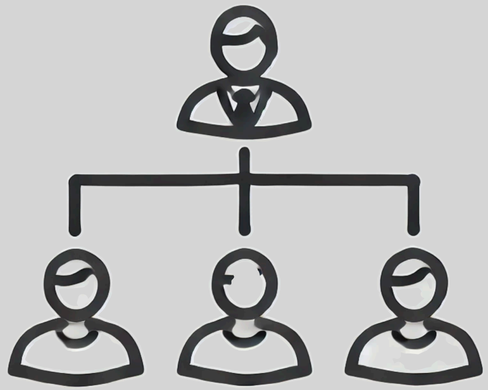


Principle Vision Investigation

Conway's Law

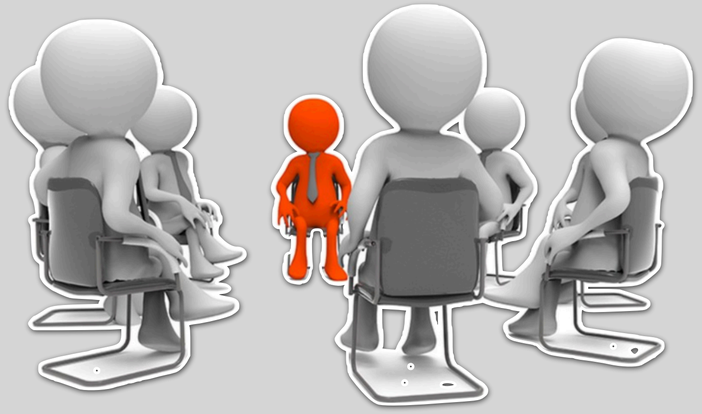
Any organization that designs a system (defined broadly)

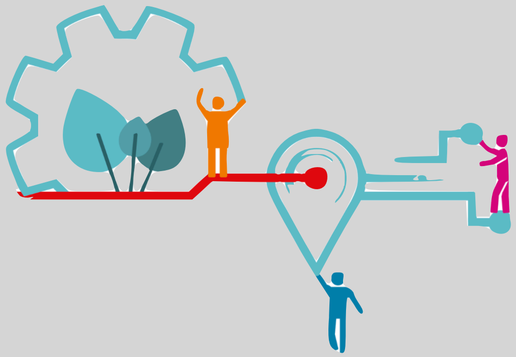
will produce a design whose structure is a copy of the organization's communication structure.



Brook's Law

The process of adding more manpower to a late-stage software project that is already behind schedule can push it to become even more delayed and behind schedule.





Identifying Transformation Vision

- Examining Structure with Conway’s Law
- Analyzing Excessive Handoffs
- Identifying Decision-Making Bottlenecks

Step 2

Redesign Around Domains (Button-Up Design)

Leverage Points Discovery via Domain-Driven Design

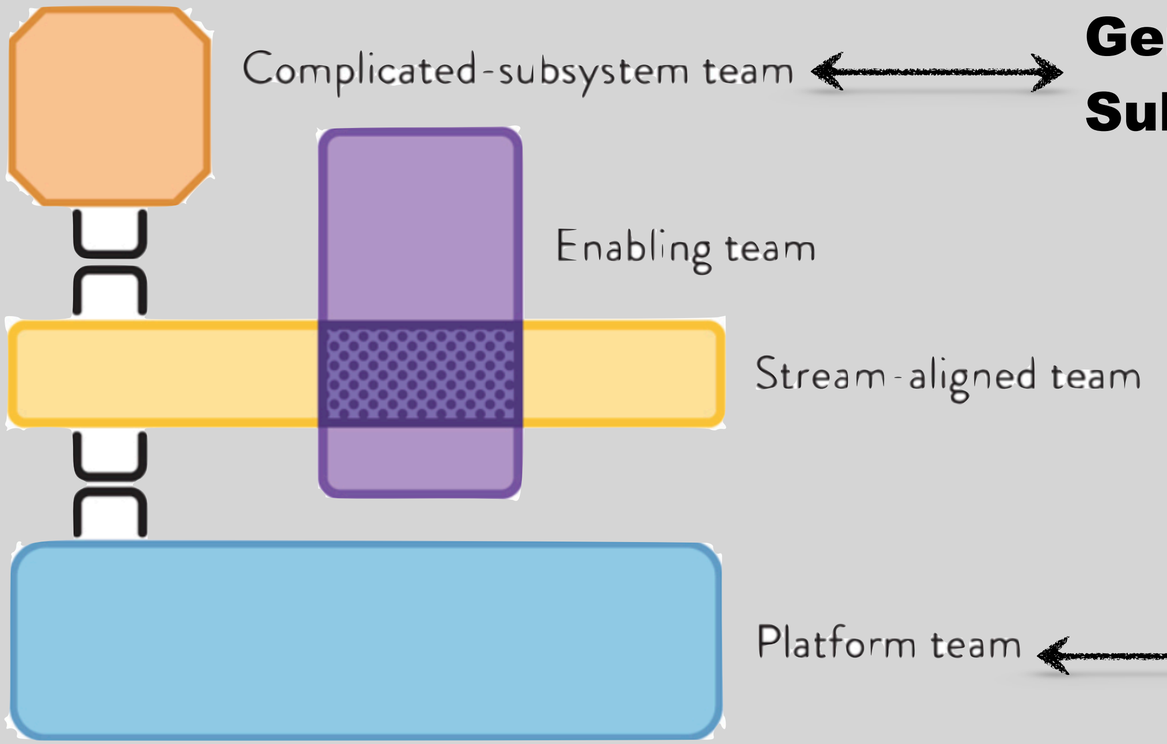
Leverage Points Discovery

- Identify Key Domain Events via EventStorming Workshop
- Define System Boundaries & Bounded Contexts
- Map Command-Event Interactions
- Pinpoint the Core Domains as Leverage Points

Step 2

Redesign Around Domains (Button-Up Design)

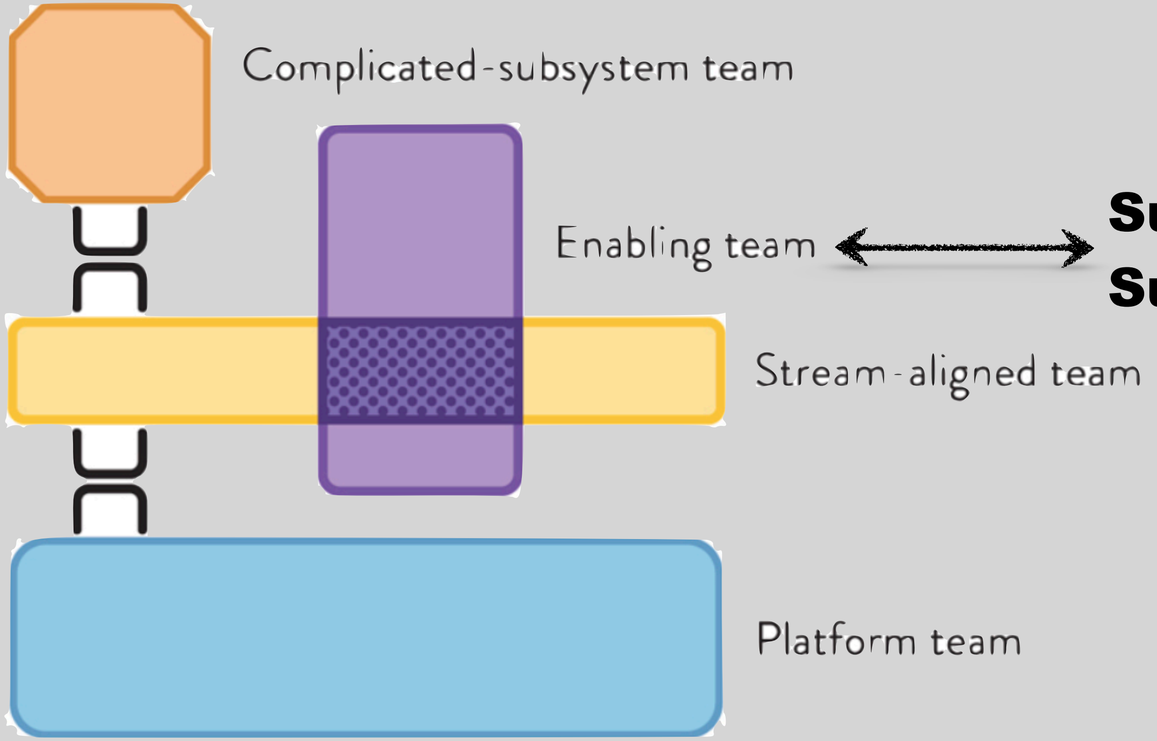
Team Topologies Explanation



Generic Subdomain

Generic subdomains refer to essential system functions that, while necessary, do not provide unique value or competitive edge.

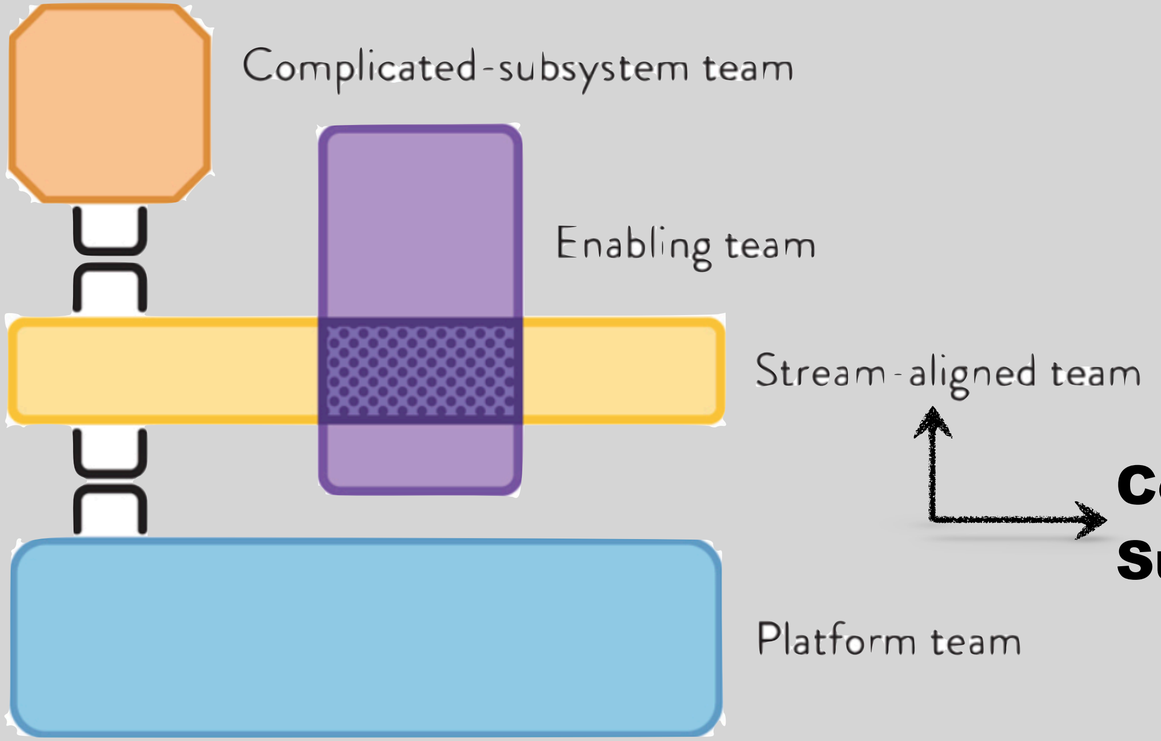
https://learning.oreilly.com/library/view/team-topologies/9781098157234/15-Ch05.xhtml#fig5-1



Supporting Subdomain

They offer essential functionalities that enhance core subdomains, although they do not directly create a competitive advantage.

https://learning.oreilly.com/library/view/team-topologies/9781098157234/15-Ch05.xhtml#fig5-1



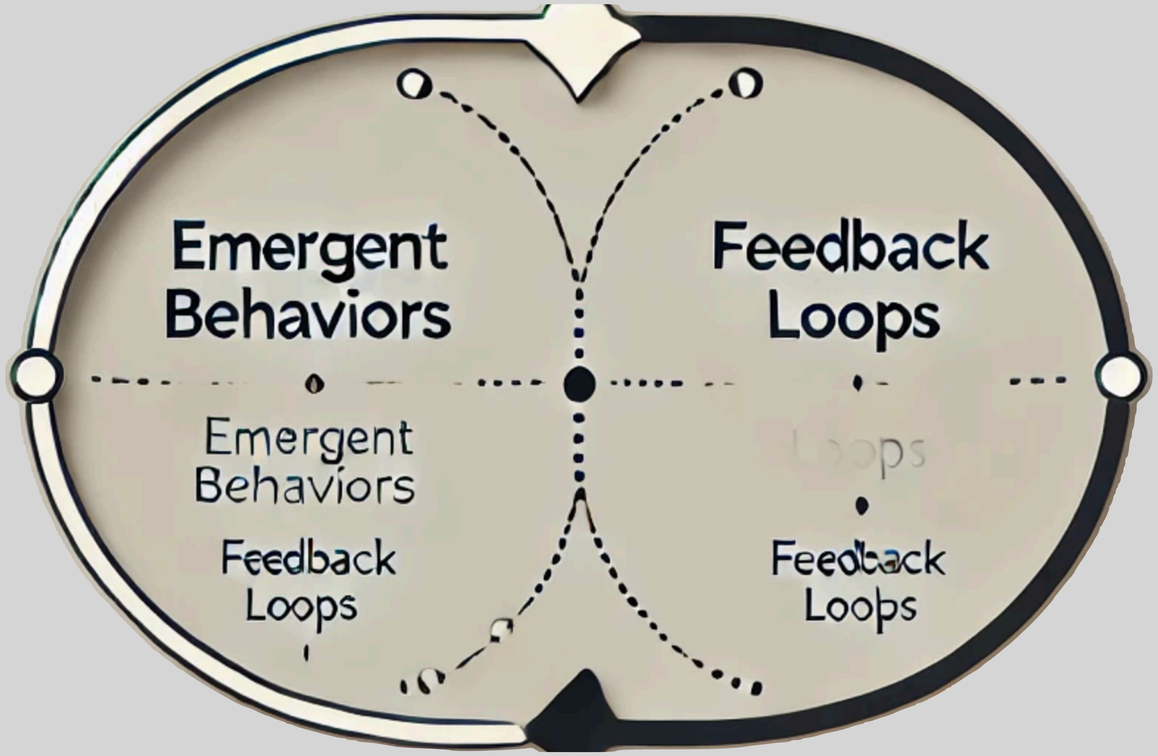
Core Subdomain

They are the fundamental components of a business that directly enhance its competitive advantage.

https://learning.oreilly.com/library/view/team-topologies/9781098157234/15-Ch05.xhtml#fig5-1

Step 3

Addressing Unexpected Outcomes



Vigilance Over Emergent Behaviors

To understand and address behaviors that arise unexpectedly from the interaction of system components

- **Cross-Domain Interaction**
- **Post-Transformation Evolution**

Step 3

Cultivate Feedback-Driven Culture

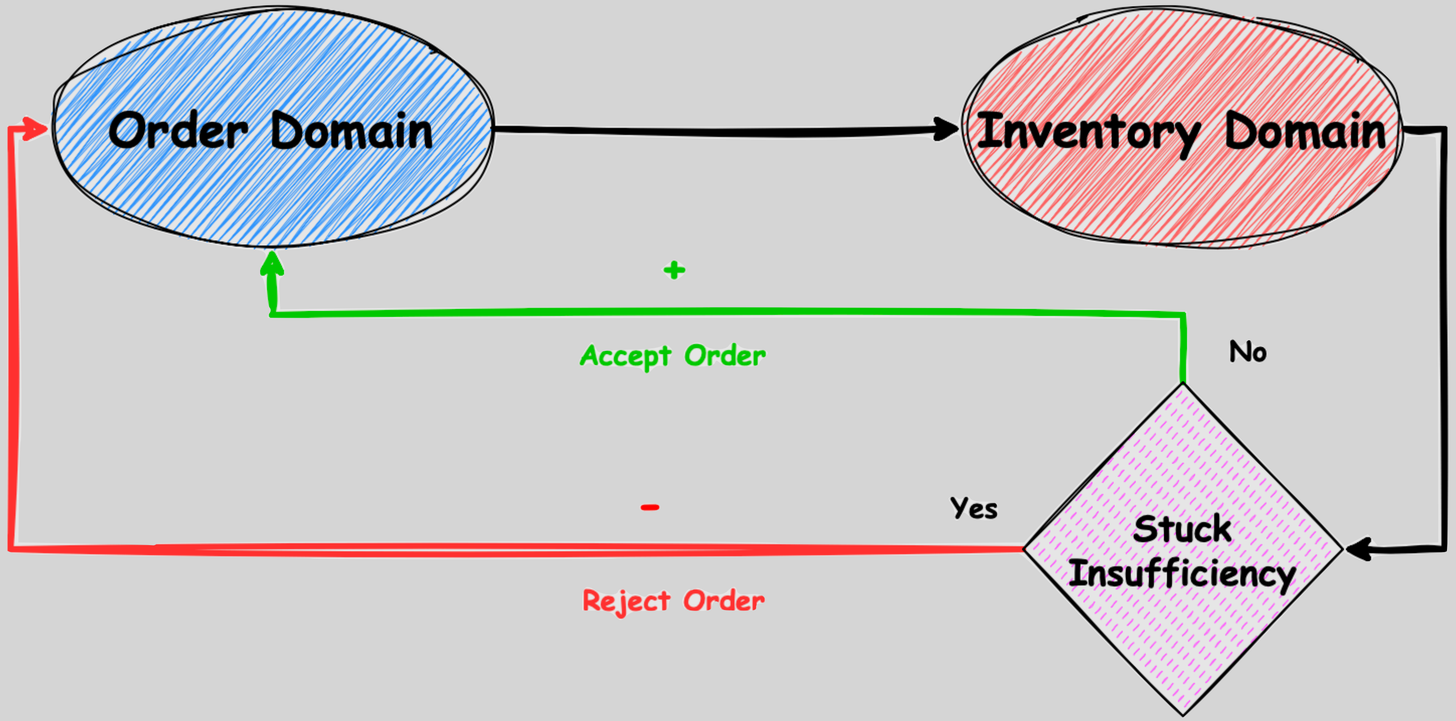
System Dynamic Adoption

System Dynamic Adoption with Feedback Loops

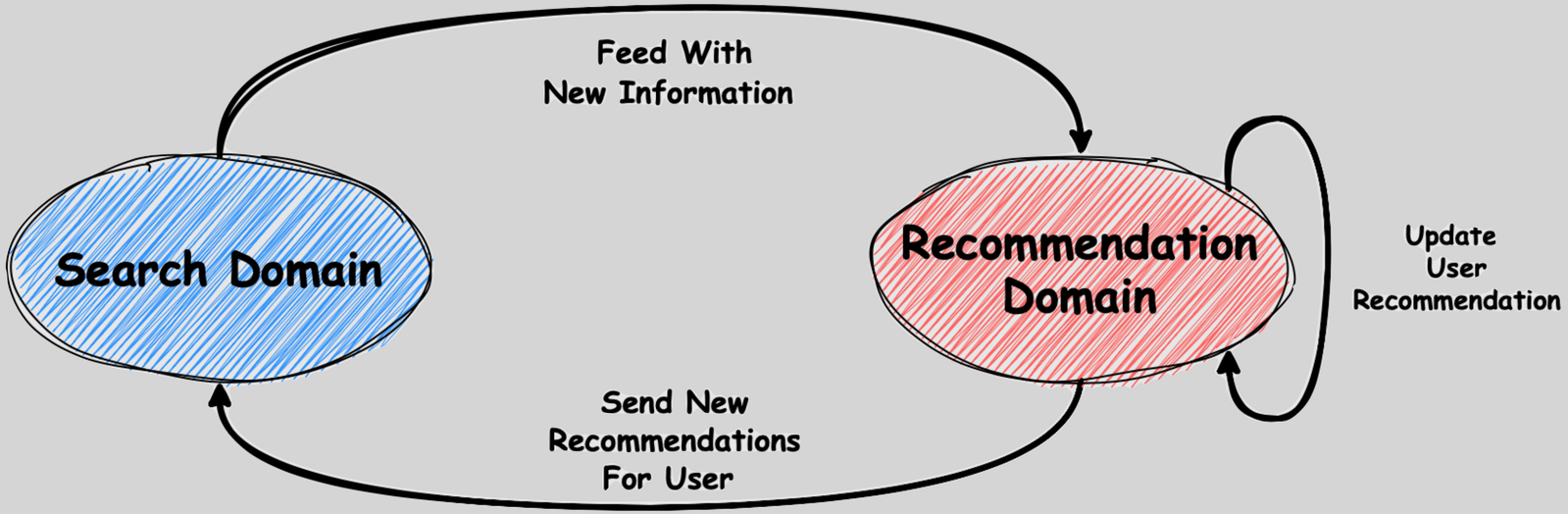
To monitor and adapt to dynamic changes in system behavior, during and after the Transformation

- Infrastructure Feedback Loops Between Domains
- Real-time Monitoring of Business Metrics
- Gathering User Insight

Negative Feedback Loops (System Stabilization)



Positive (Reinforcement) Feedback Loops



Step 4

Implement Incremental Changes

Global System Alignment

Global System Alignment With Fitness Functions

To measure how well the Transformation Project and its components align with business goals

- Continuous Integration & Continuous Delivery
- System Design & Architecture Definition
- Post-Deployment Monitoring (System Metrics)

Software Transformation is not a Final Destination
It is an Ongoing Journey

Thanks

Practical Envisioning of
Software Architecture

Masoud Chelongar
Hands-on Software Architect

msd@chelongar.com

https://www.chelongar.com