Refactoring Socio-Technical Systems

Streamlining Your Structures & Flows

Dr. Franziska Horn

Umm…what?

- **Socio-Technical (ST) System**:
  an organization where people and technology interact
  → focus here: software companies

- **Refactoring**:
  in computer science: changing how code is structured while
  preserving its purpose and functionality

⇒ Improving *how* a company does what they do

What brings me here

- PhD in Machine Learning (ML)

- Started out as a Data Scientist & Python Developer

- Team Lead at a startup: improved internal processes 💡

Refactoring in practice

At that startup, many things were great 🤩

- Modular software architecture (multiple microservices)

- Deploy to production multiple times per day

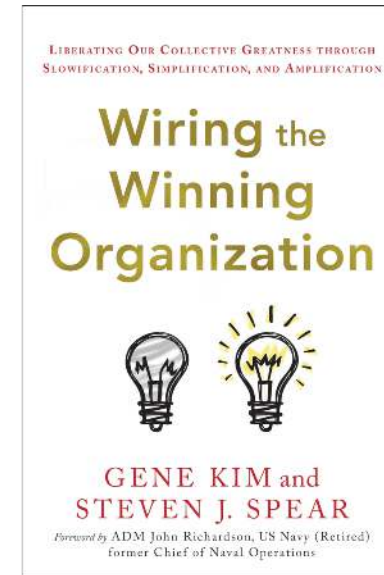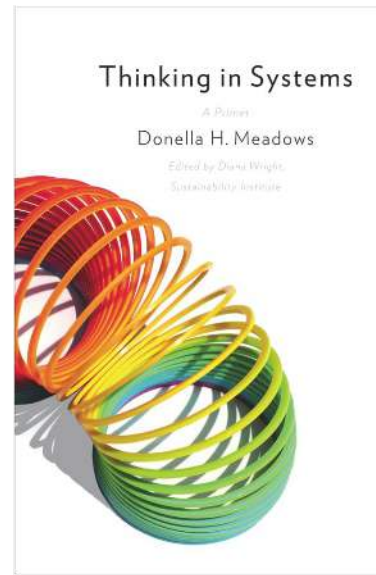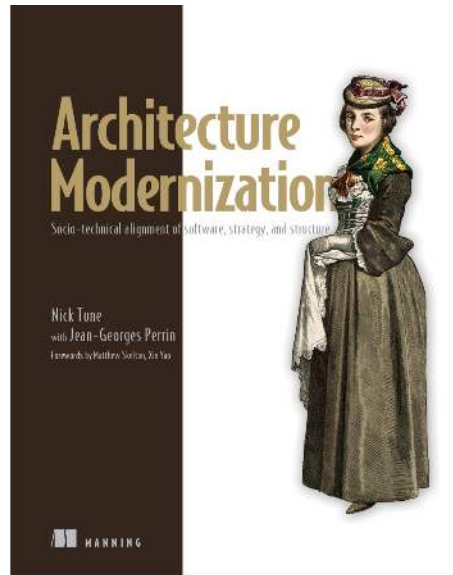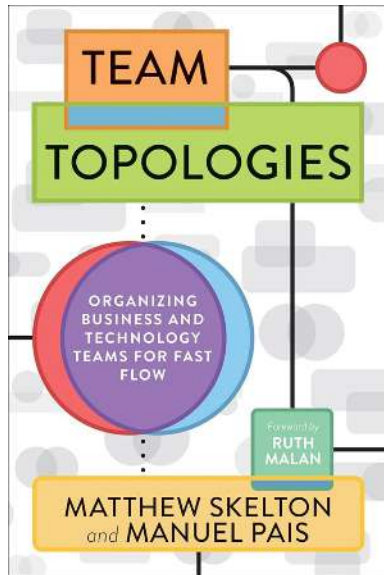- IaC and provision of new test environments via slackbot

- Weekly automated retraining of ML models on new data

But…

- Customer onboarding & data integration took 50 days 😱

⇒ Reduced to 25 days through a series of small refactorings 🚀

A holistic perspective on ST systems
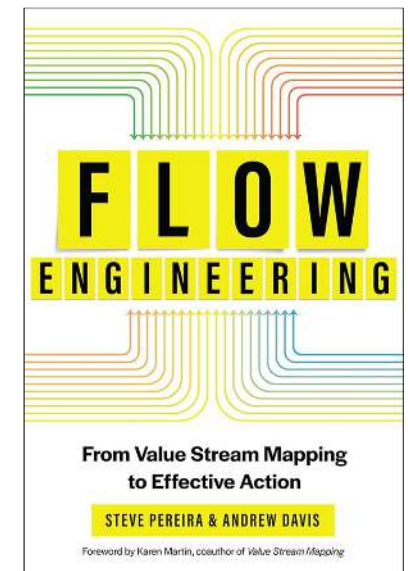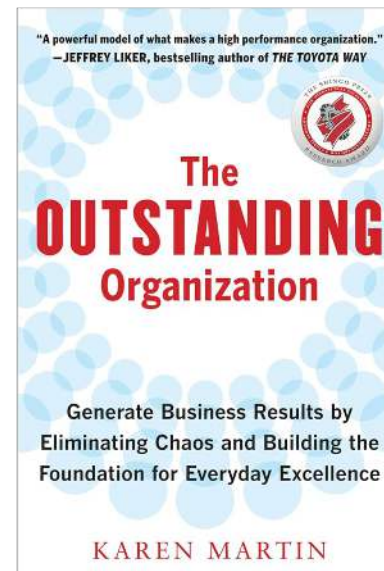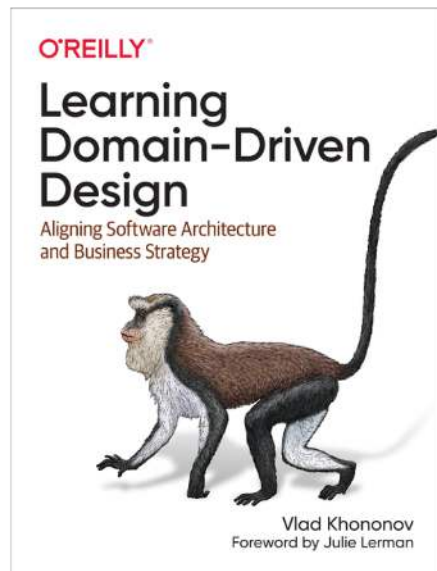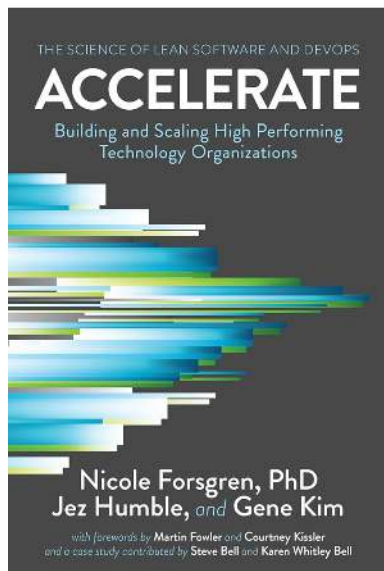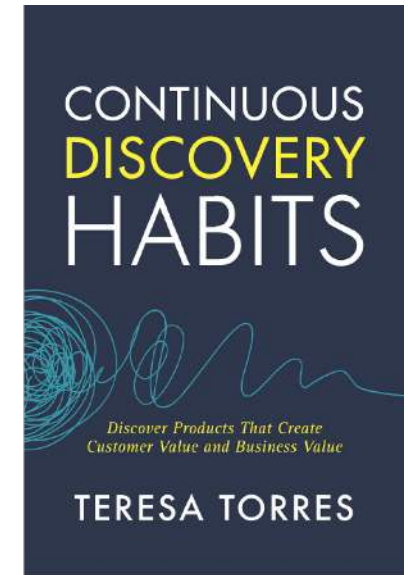
Your company is a ST system

Purpose ⭐
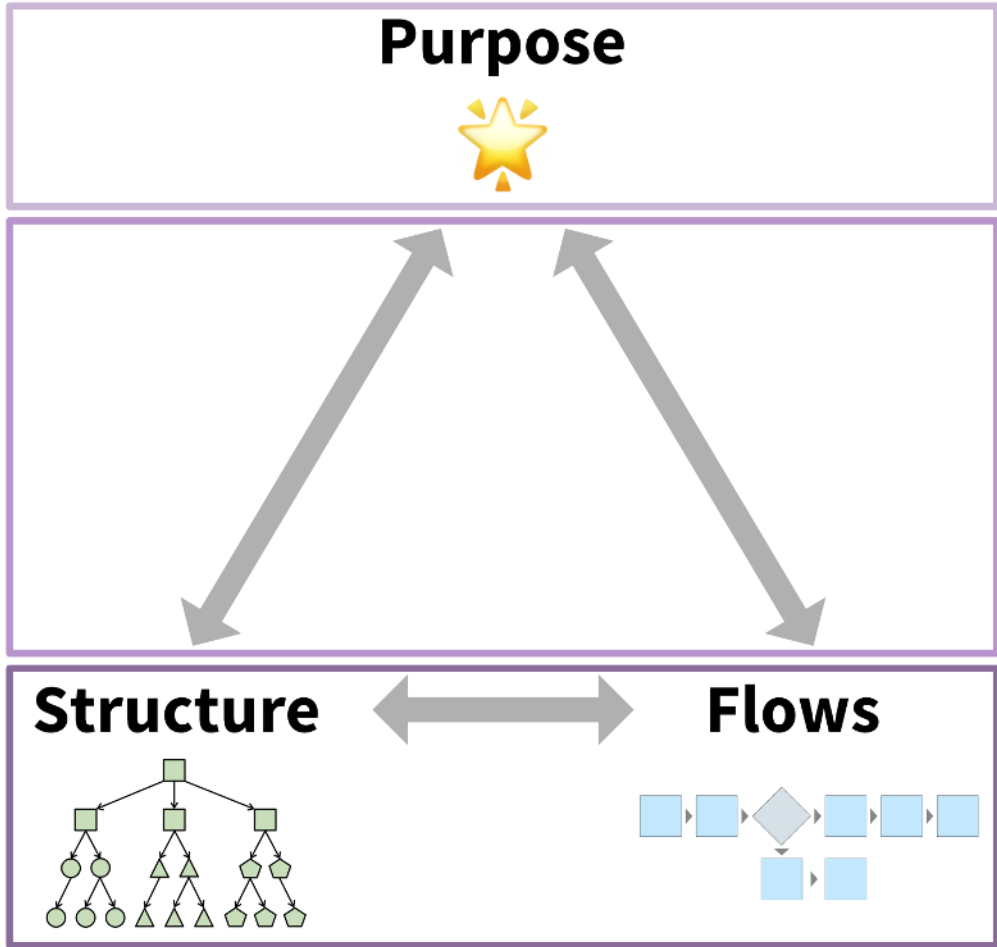
Structure ⟷ **Flows**

Why — Outcome
Value for customers
Getting from A to B

What — Output
Product / Service
Car

How — Process
Manufacturing
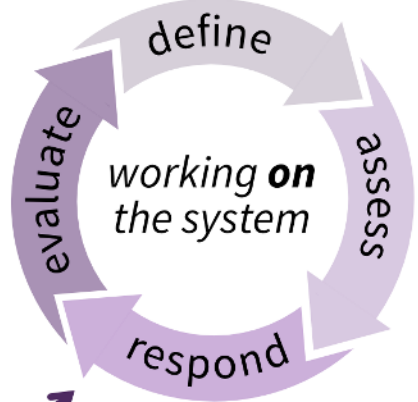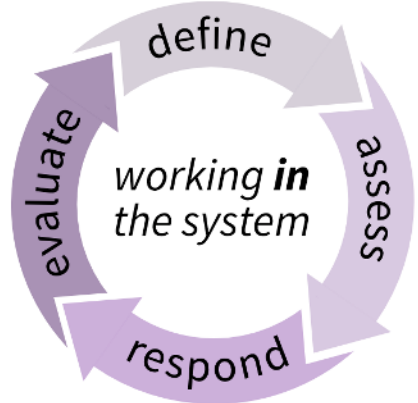
define · assess · respond · evaluate
*working **in** the system*

define · assess · respond · evaluate
*working **on** the system*

Refactoring

Refactoring a ST system – Part 1
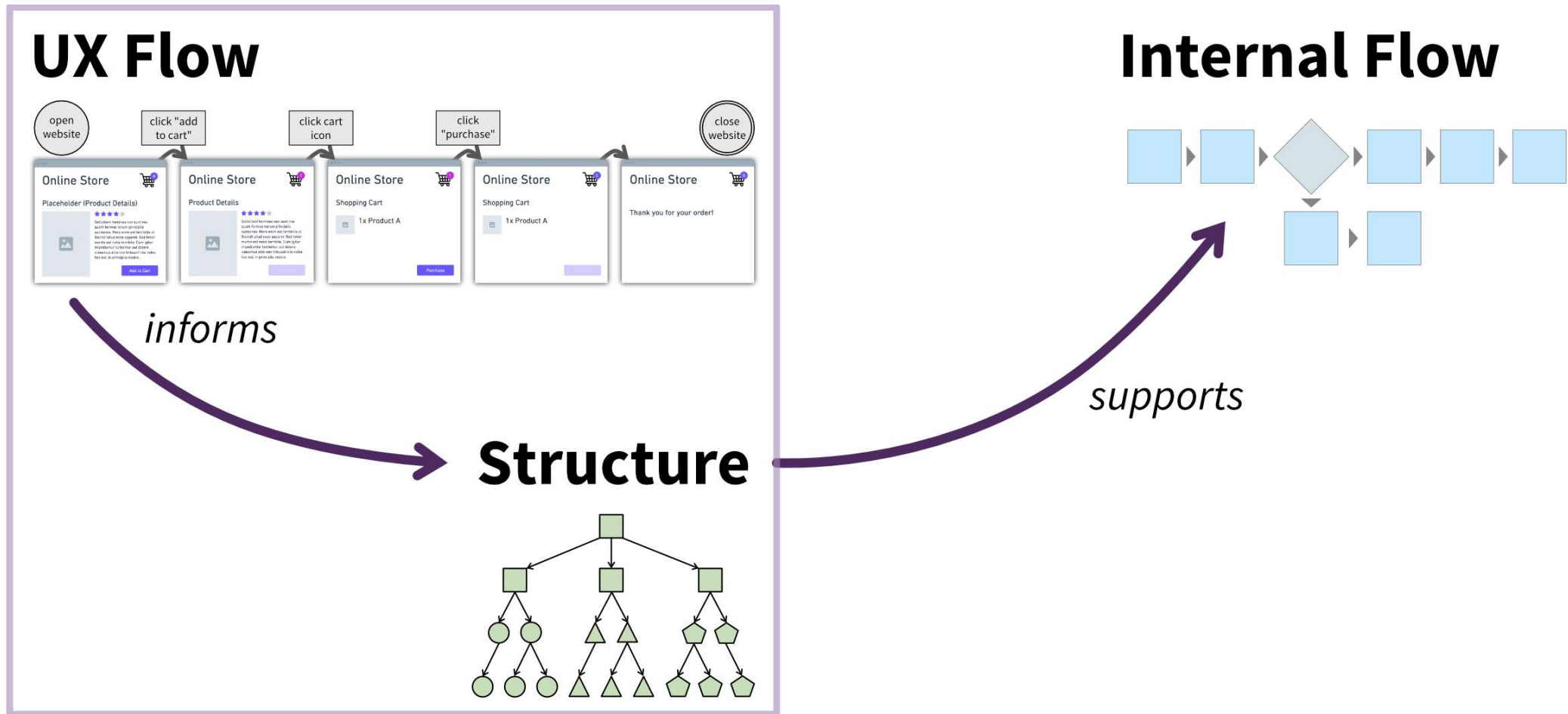
UX Flow

informs

Structure

Internal Flow

supports

Aligning Structures with UX Flow

User
Use Cases / Jobs-to-be-done

Product
Information Architecture

Teams
Organizational Structure

Code
Software Architecture & Repos

Product UX: Information Architecture

Online Learning Platform

Information Architecture (IA) aligned with Use Cases:

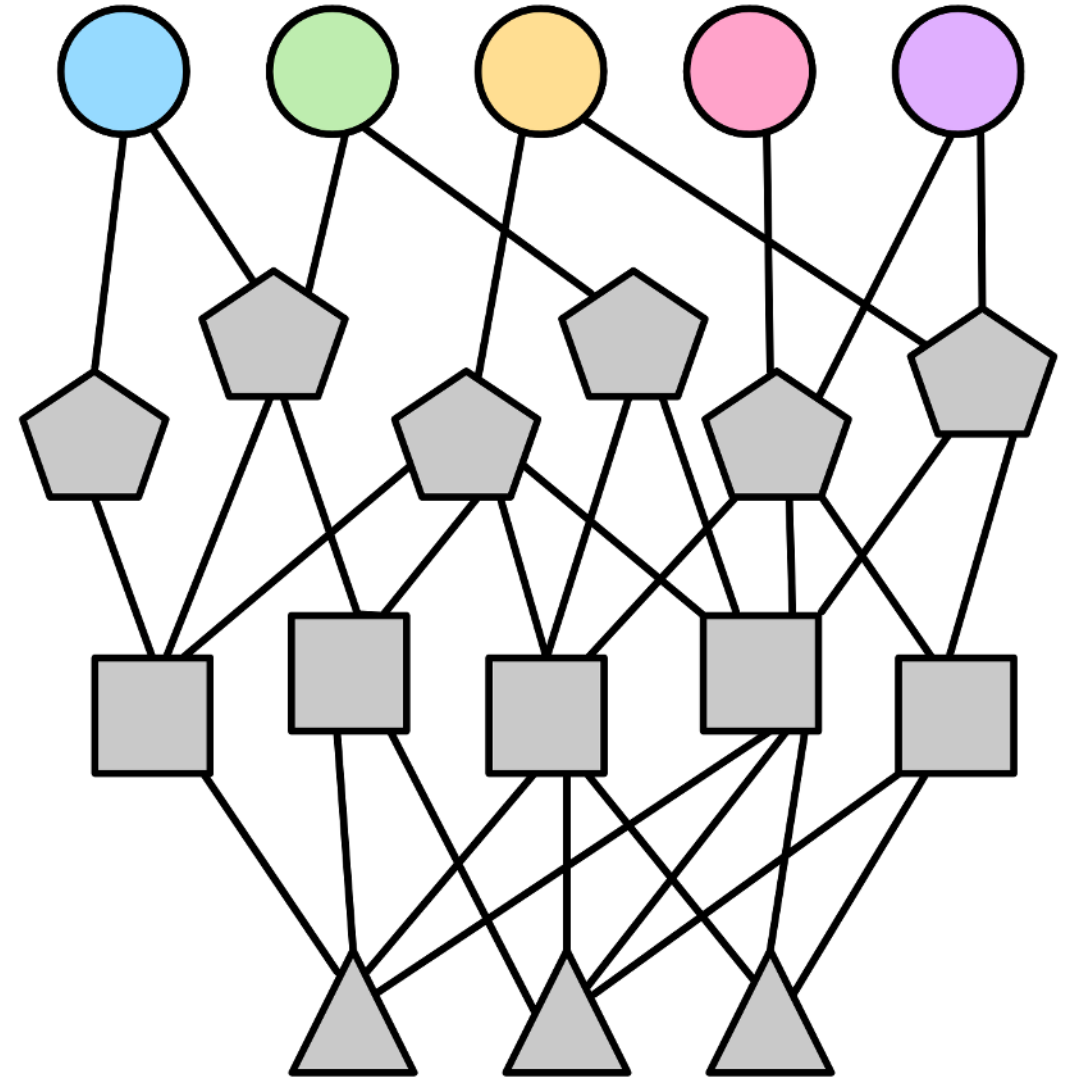| Topic A	Topic B	Topic C	Topic D	Topic E

Texts   Texts
Videos   Videos
Quizzes   Quizzes

IA aligned with Functionality:

Texts	Videos	Quizzes

Topic A   Topic A
Topic B   Topic B
Topic C   Topic C
Topic D
Topic E

Aligning Structures with UX Flow

User
Use Cases / Jobs-to-be-done

Product
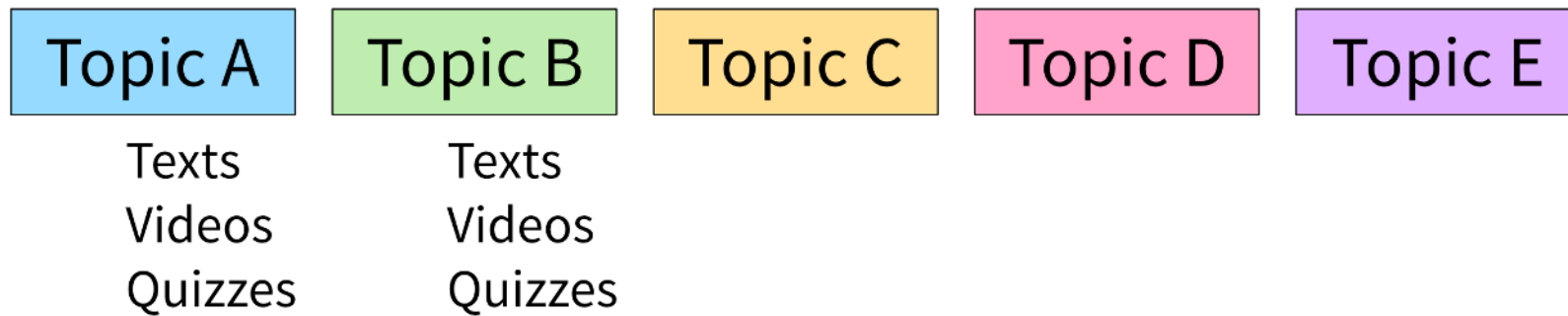Information Architecture
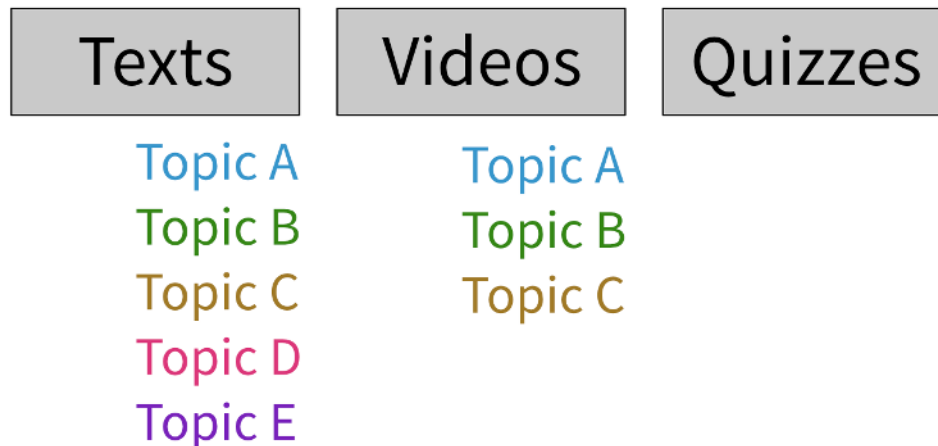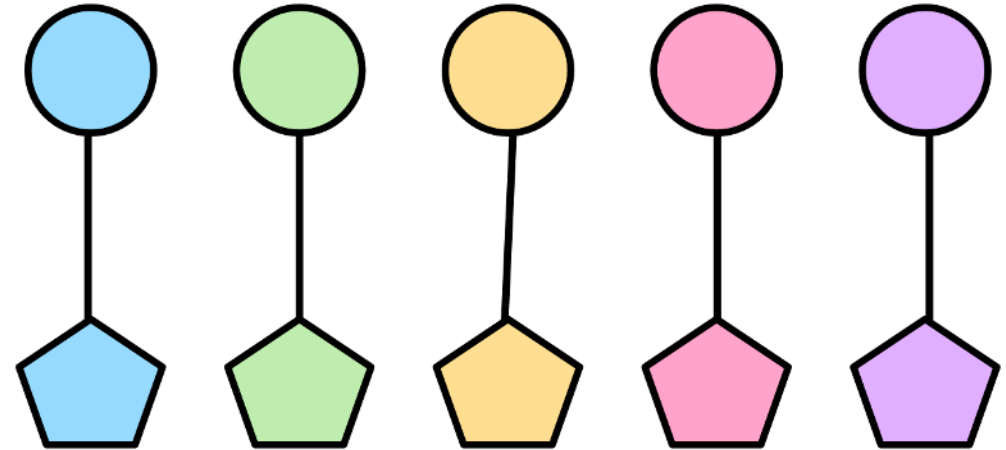
Teams
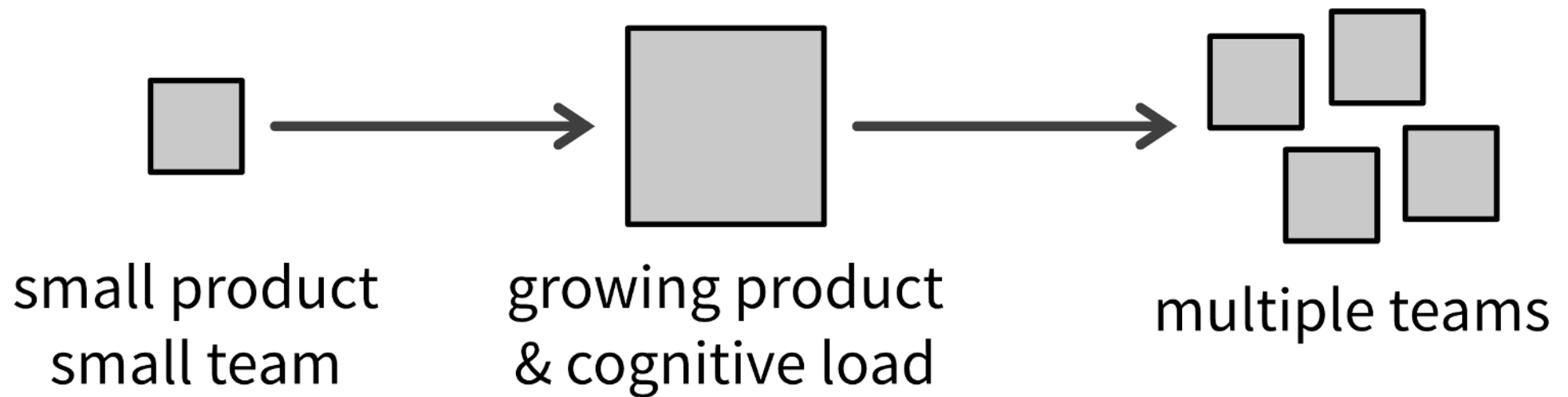Organizational Structure

Code
Software Architecture & Repos

From Silos to Product Teams

Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**Waterfall**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**"Agile" with release silo**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**"Agile" able to release**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**Design as team member**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**DevOps design/dev collab**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**Feature Factory**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**Product Team w/ "mini CEO"**
Opportunity Selection	Reqs Planning	Design	Build	Test	Release	Run	**Product Team**

Team Boundary   External handoff   Internal handoff   Cross-functional collaboration Start | Work | Finish together

Amplitude

Source: https://amplitude.com/blog/journey-to-product-teams-infographic

Growth & increased cognitive load
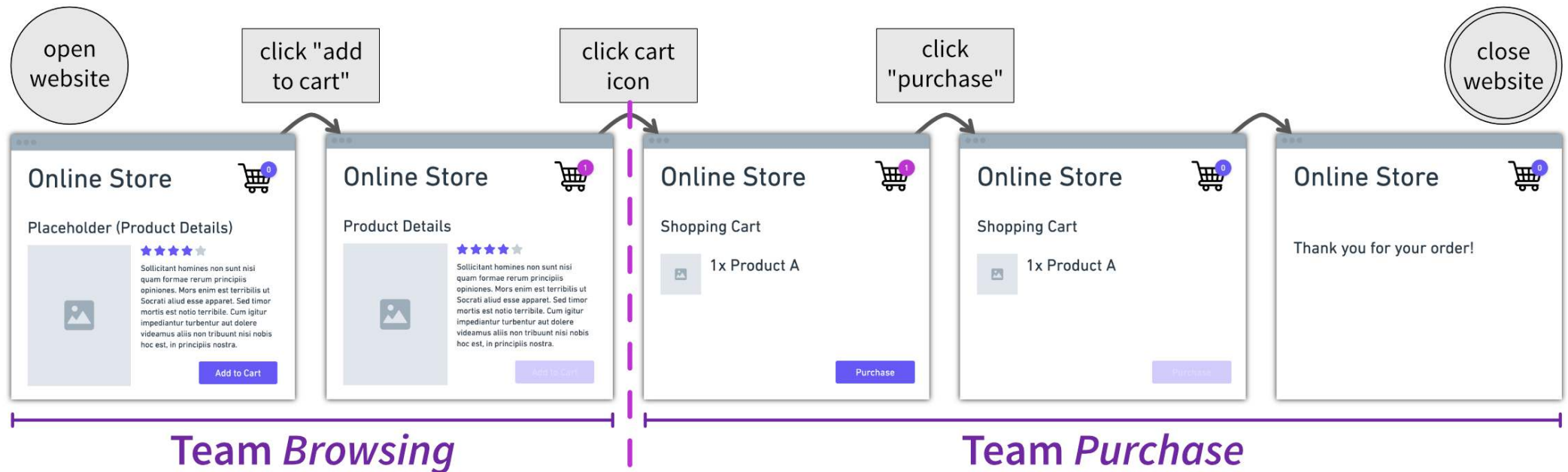

small product
small team
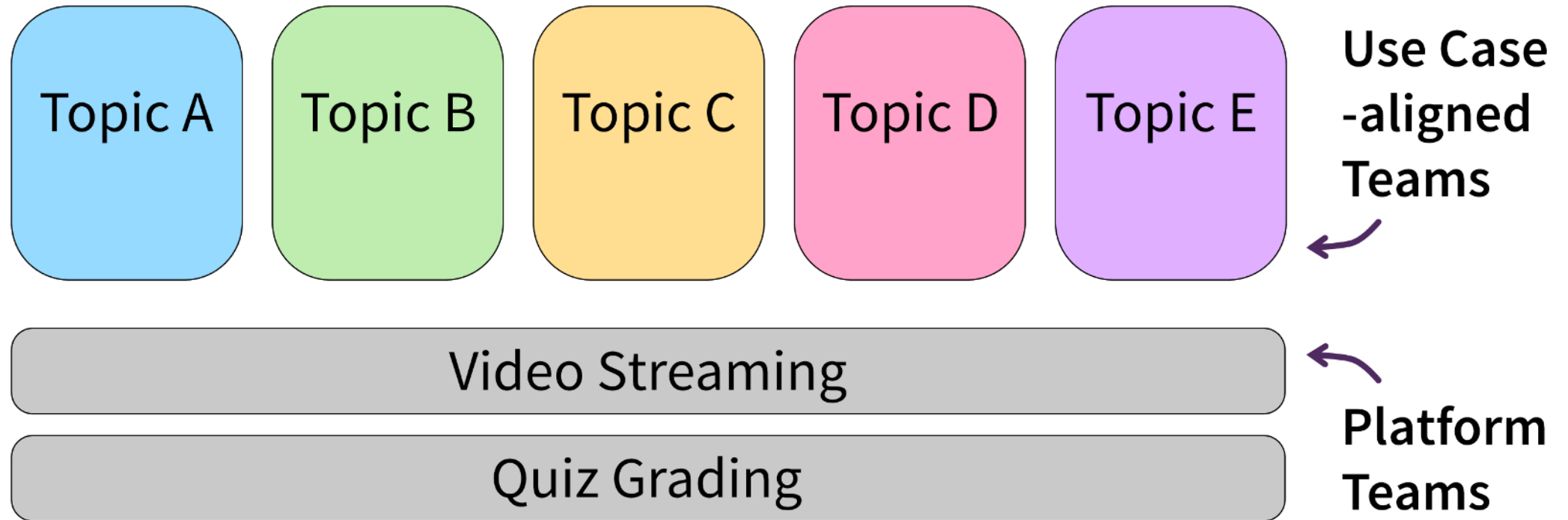
growing product
& cognitive load

multiple teams

⇒ **Goal:** minimize hand-offs & dependencies between teams

Teams own use cases

- Identify distinct user groups (e.g., buyer & seller)
- Split by use case

Don't forget platform teams!

| Topic A | Topic B | Topic C | Topic D | Topic E |

Use Case -aligned Teams

| Video Streaming |
| Quiz Grading |

Platform Teams

- Define clear responsibilities and interfaces between teams!

- Additionally: matrix organization & communities of practice

Inverse Conway Maneuver

Conway's Law:

> Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.
>
> — Melvin E. Conway, How Do Committees Invent? (1968)
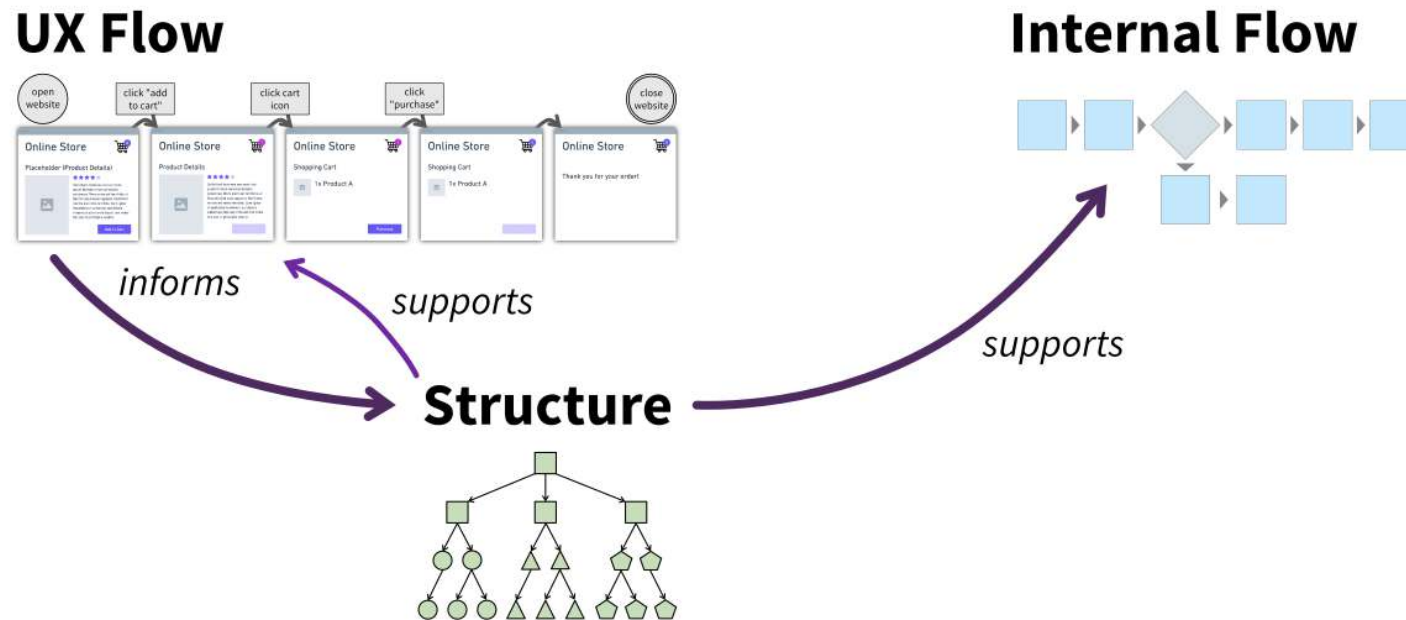
Aligning Structures with UX Flow

User
Use Cases / Jobs-to-be-done

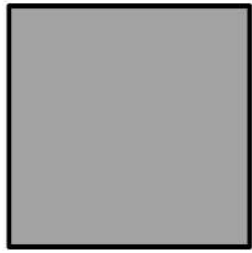**Product**
Information Architecture

Teams
Organizational Structure
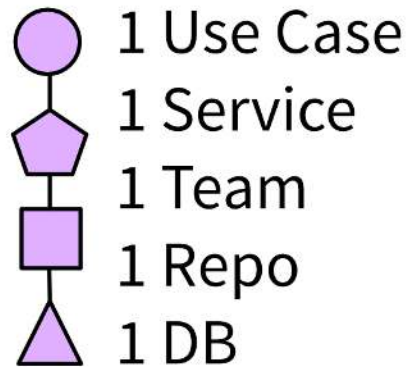
Code
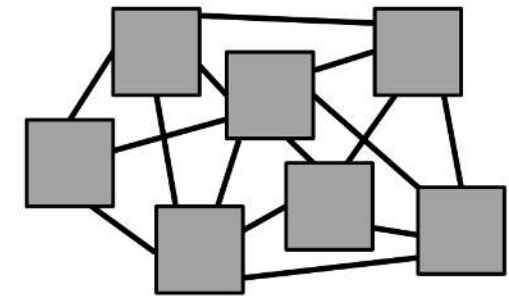Software Architecture & Repos
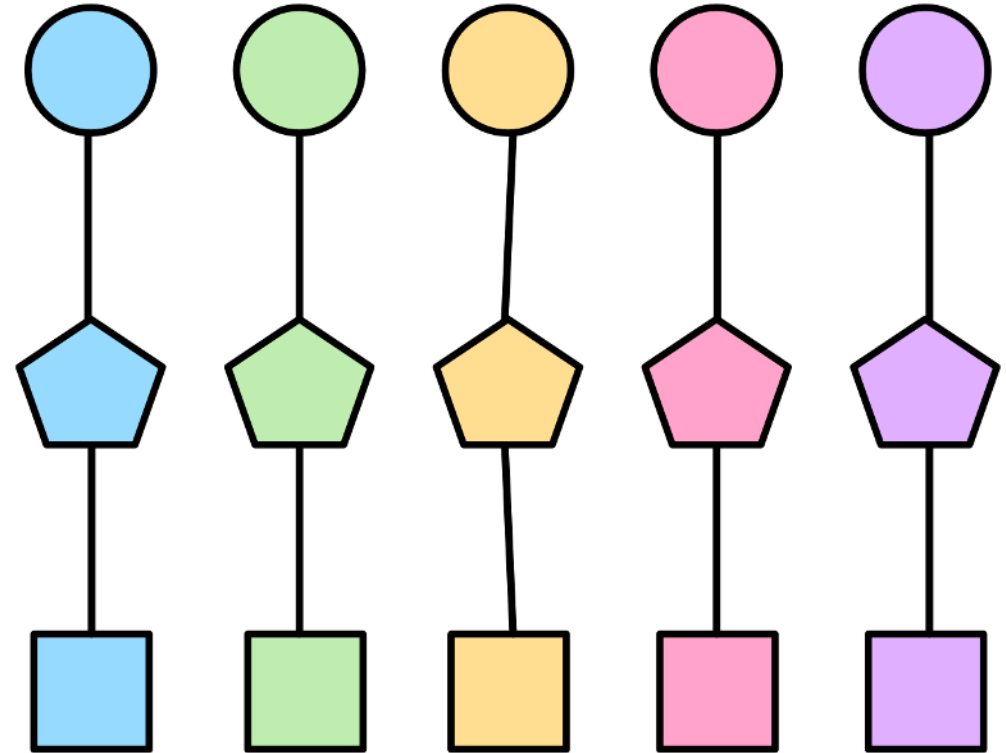
Balance Coupling

High Local Complexity

Monolith

😬

keep it modular & needs good CI/CD

Just right

○ 1 Use Case
⬠ 1 Service
▢ 1 Team
⬛ 1 Repo
△ 1 DB

🤩

manageable cognitive load

High Global Complexity

Coupled Microservices

🥴

beware of coupling at DB

Aligning Structures with UX Flow ✅

User
Use Cases / Jobs-to-be-done
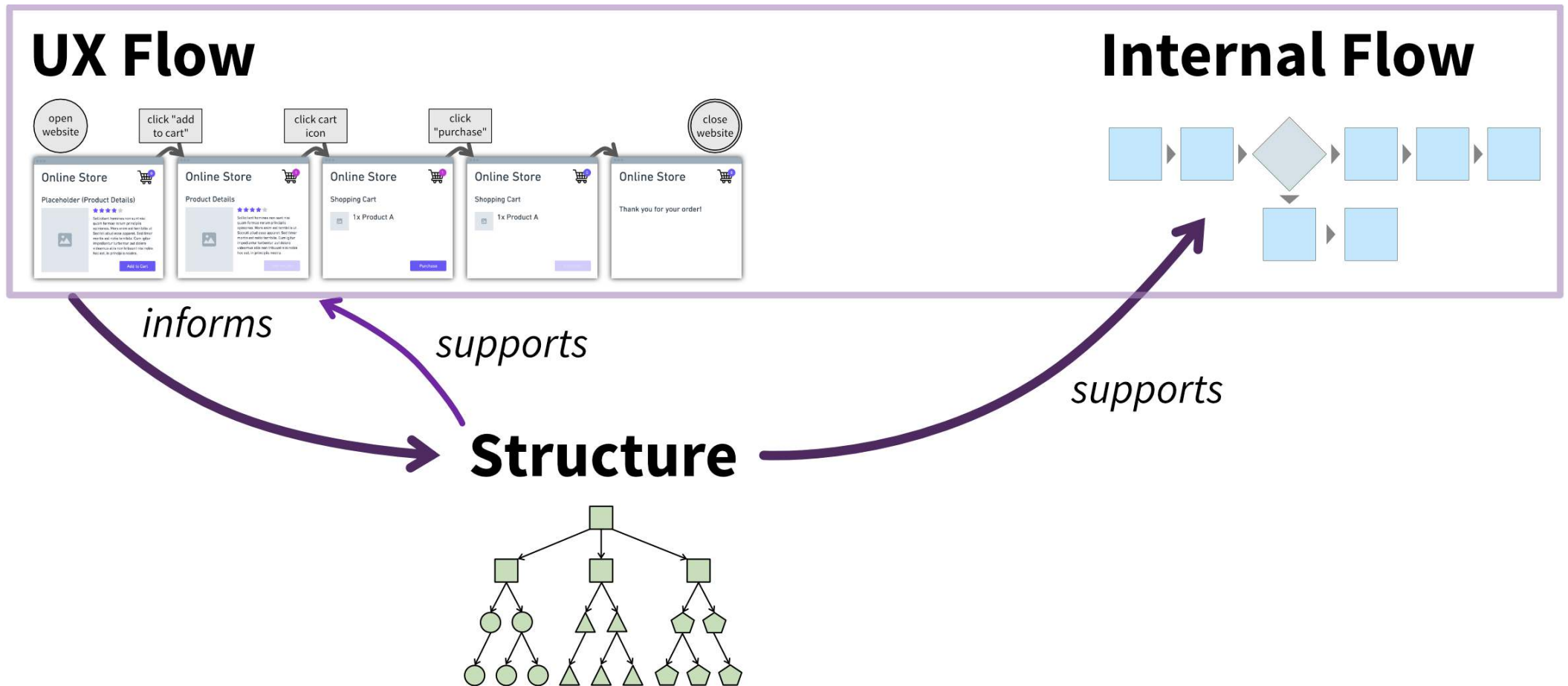
Product
Information Architecture

Teams
Organizational Structure

Code
Software Architecture & Repos
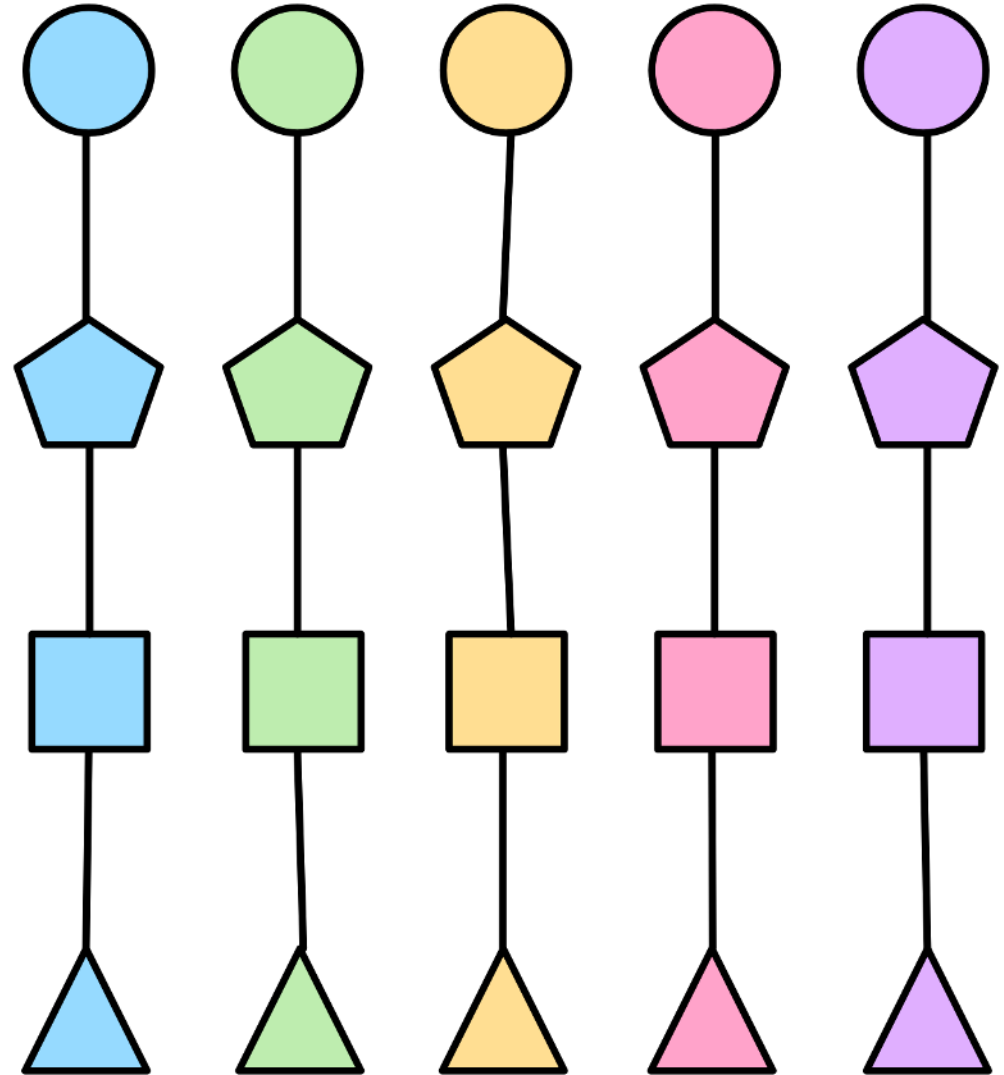
Refactoring a ST system – Pt. 2: Flows

Flows worth looking at

- UX Flows:
  - Account creation & onboarding … and deletion
  - Purchase / subscription renewal & billing
- Internal Flows:
  - Onboarding of new developers
  - Deployment process (incl. feature flags, A/B tests)
  - Updating content for marketing campaigns
  - Database migrations
  - Rollback & disaster recovery
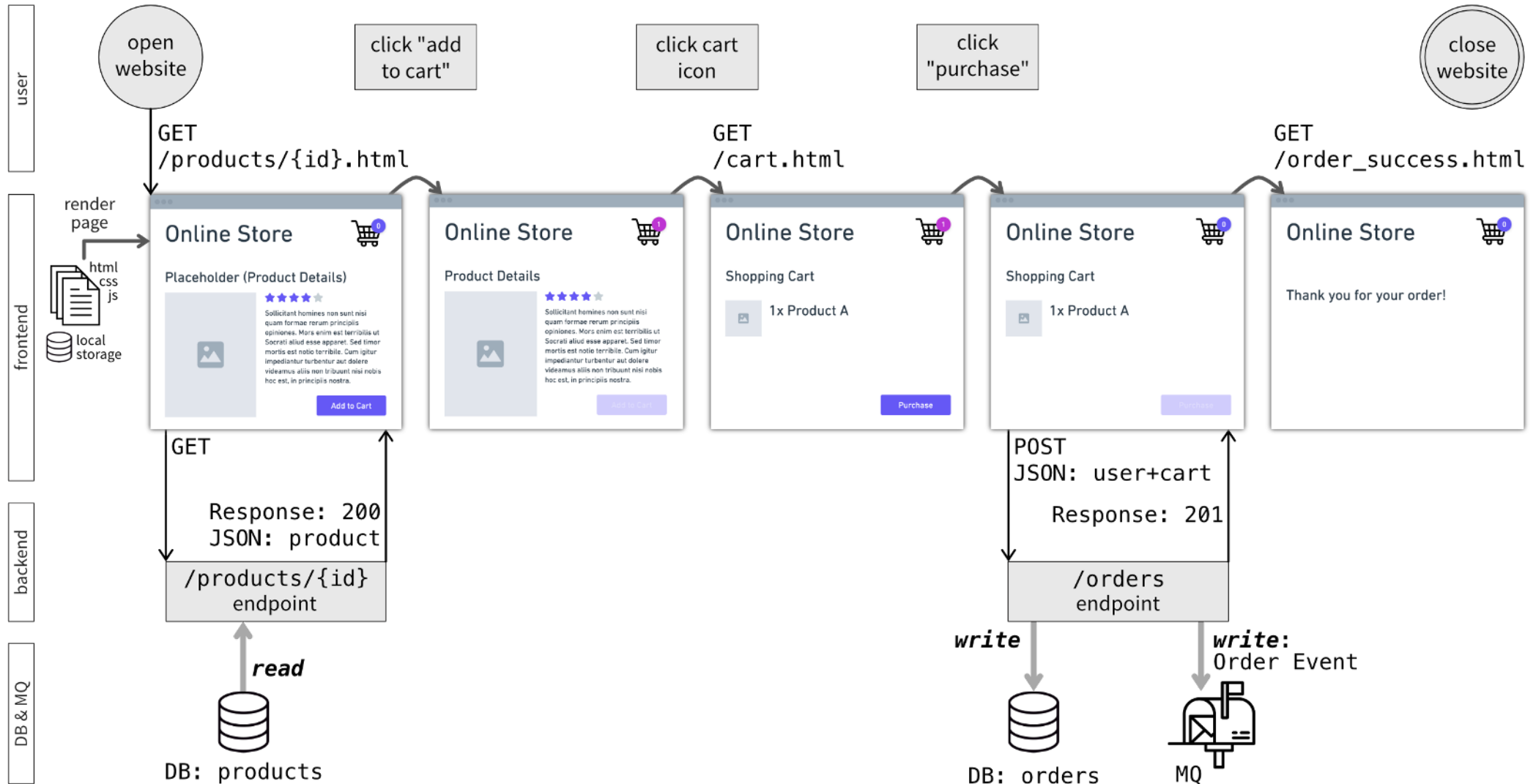
"Fix the Flow" Workshop

Preparation: Sketch the flow

- no need for formalities (like BPMN or UML)
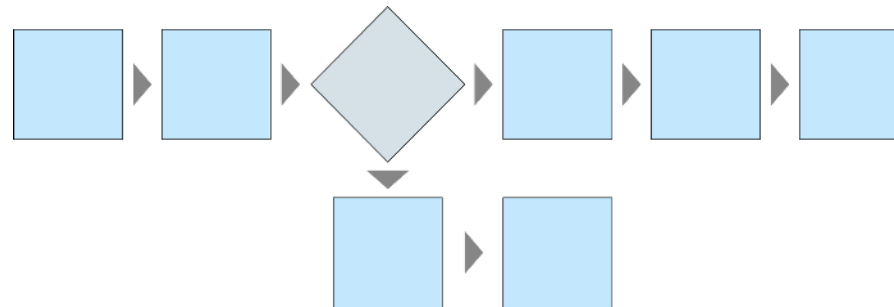
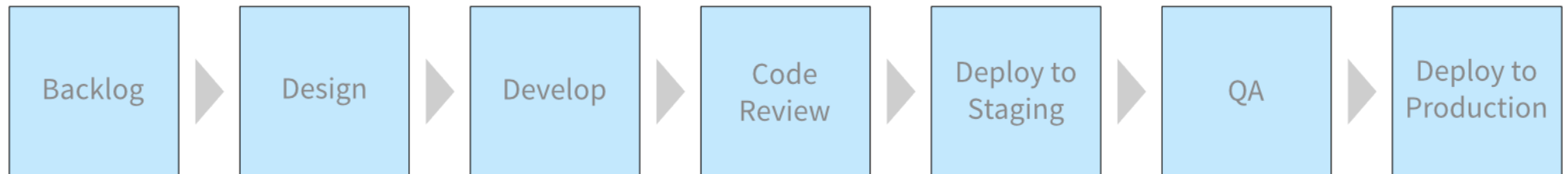- talk to different people to get diverse perspectives

Flow sketch with swimlanes

"Fix the Flow" Workshop – Step 1/6
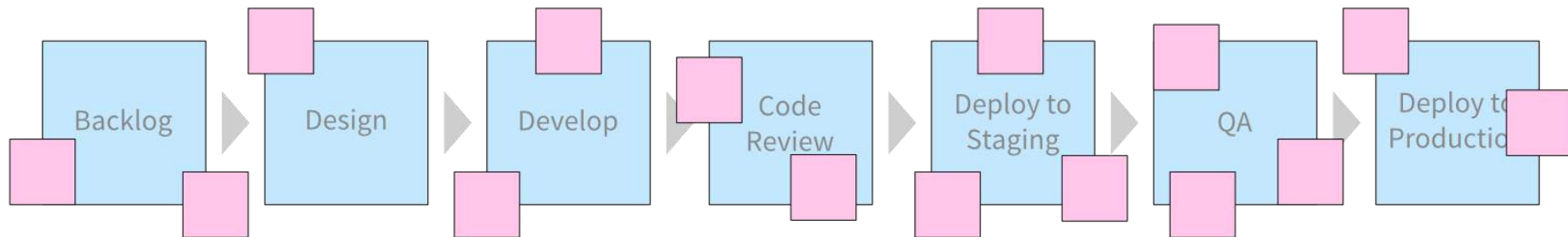
Align on the flow & target outcome:

- *How much time does this process take now? What's the goal?*

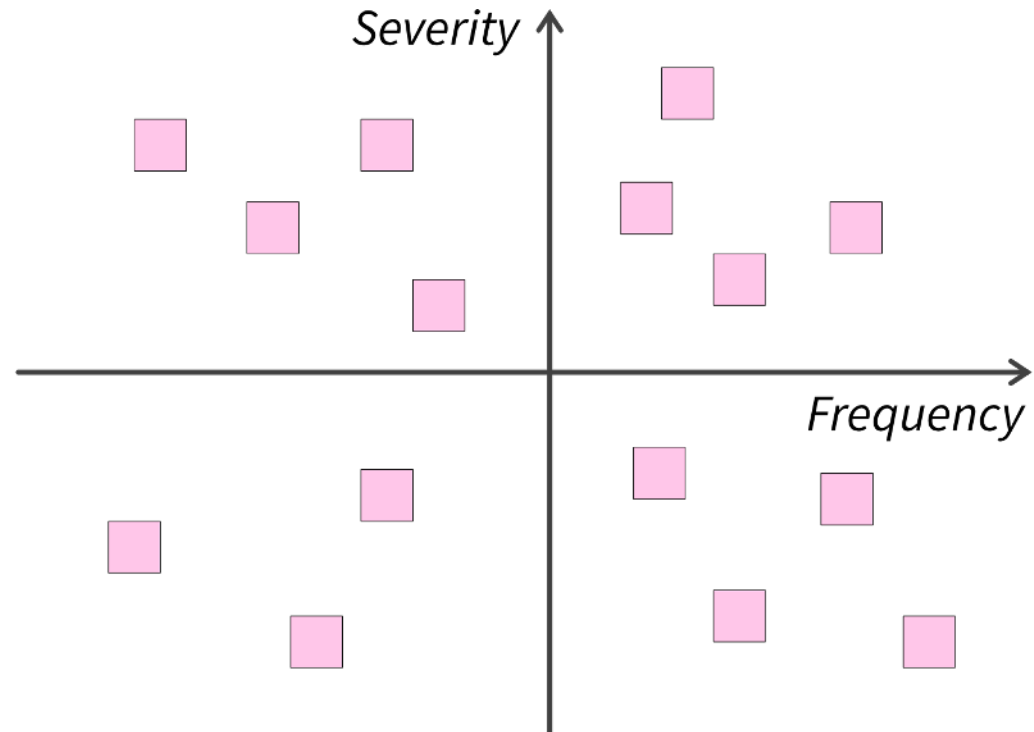| Backlog | | Design | | Develop | | Code Review | | Deploy to Staging | | QA | | Deploy to Production |

"Fix the Flow" Workshop – Step 2/6

Brainstorm pain points:

- *What is annoying?*
  (e.g., manual steps, dependencies, wait times, complexity)

"Fix the Flow" Workshop – Step 3/6

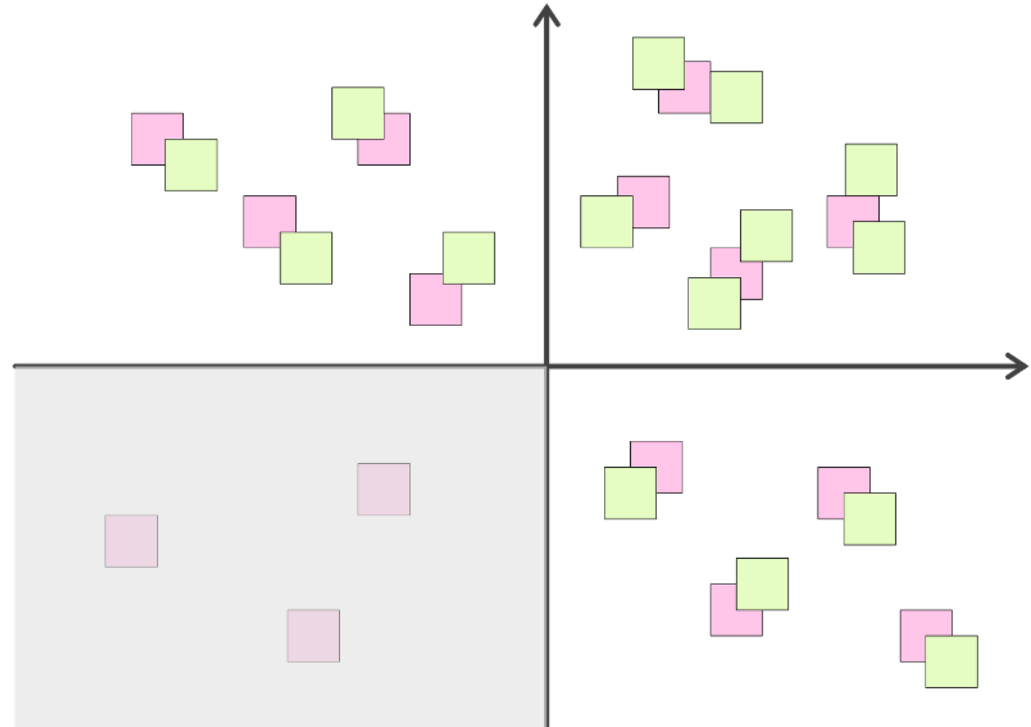**Prioritize pain points:**

- *How long does it take?*

- *How often does it happen?*

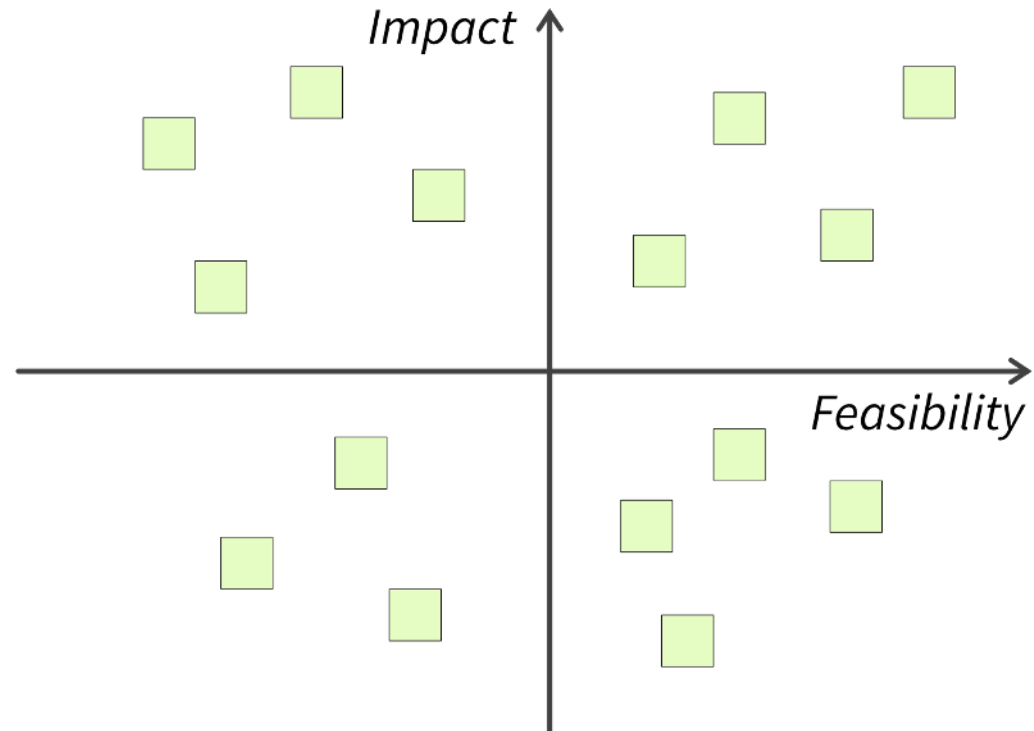# "Fix the Flow" Workshop – Step 4/6

Brainstorm solutions:

- *What can be improved?*
  - *simplify*
  - *automate*
  - *create self-service options*

"Fix the Flow" Workshop – Step 5/6

Prioritize solutions:

- *What's the impact on our target outcome (e.g., reduced time)?*

- *How easy would it be to implement?*
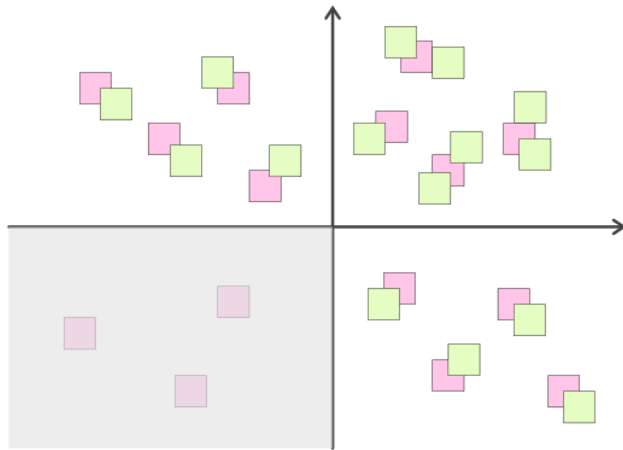
"Fix the Flow" Workshop – Step 6/6

Make a plan:

Assign owners & deadlines (e.g., in Jira)

1	Owner (by Date)
2	Owner (by Date)
3	Owner (by Date)
4	Owner (by Date)

"Fix the Flow" Workshop – Summary
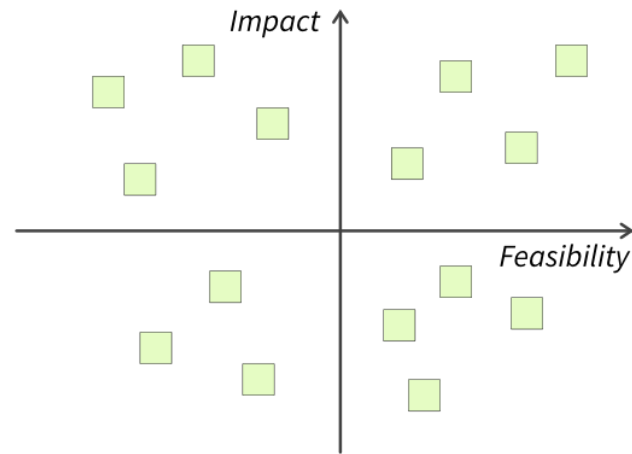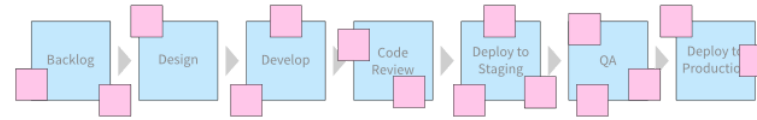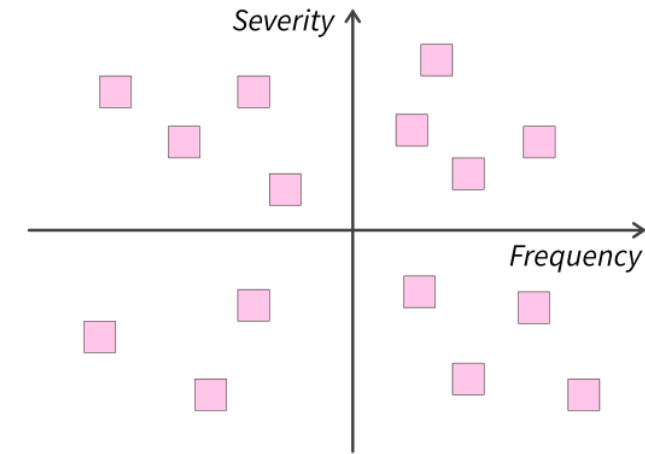
Step 1: Align on Flow & Target Outcome

Step 2: Brainstorm Pain Points
(e.g., manual steps, dependencies, wait times, complexity)

Step 3: Prioritize Pain Points

Step 4: Brainstorm Solutions
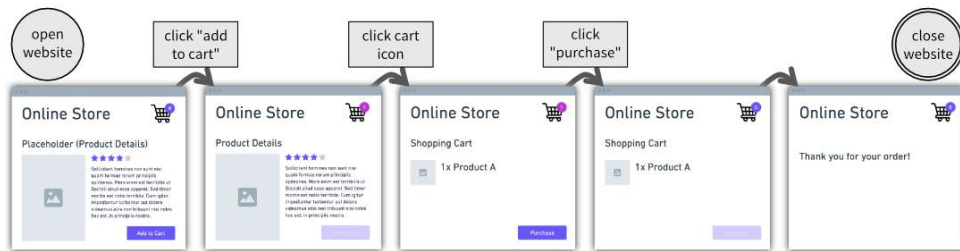*(e.g., simplify, automate, create self-service options)*

Step 5: Prioritize Solutions

Step 6: Make a Plan

Refactoring a ST system – closed loop

UX Flow

Internal Flow
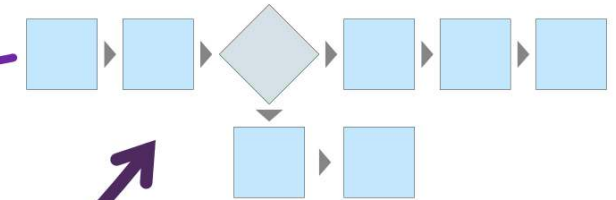
informs

supports

informs

supports

Structure
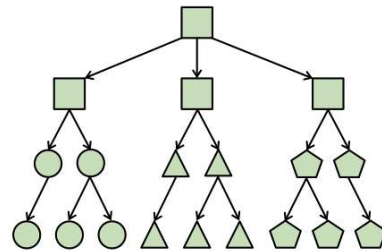
Need help? Hire me 🤓
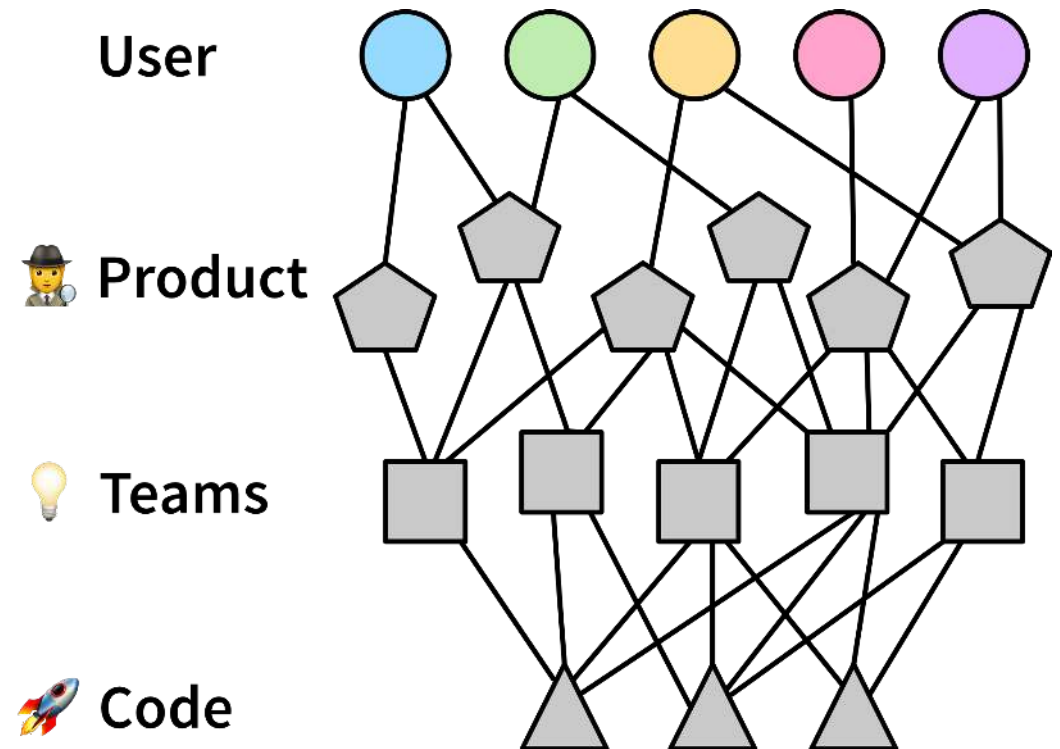
Freelance Consulting Services:

1. **System Audit**
   Identify opportunities

2. **Help with refactoring**
   Hands-on, part of your team

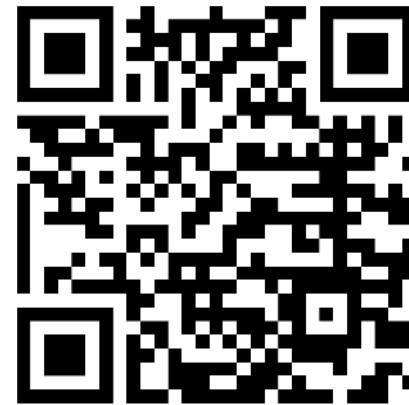User
🕵️ Product
💡 Teams
🚀 Code

Thank you for your attention!

Slides:

franziskahorn.de/resources.html

Connect:

linkedin.com/in/franziska-horn

Any questions?

www.franziskahorn.de