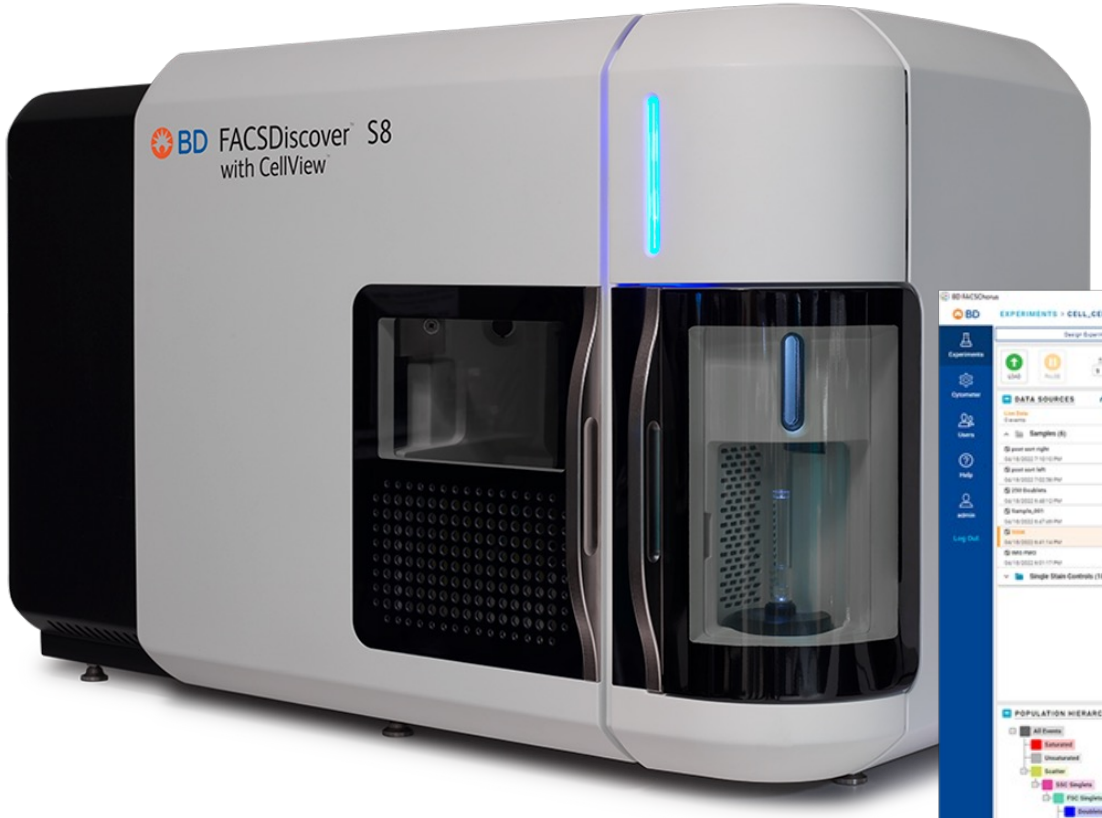
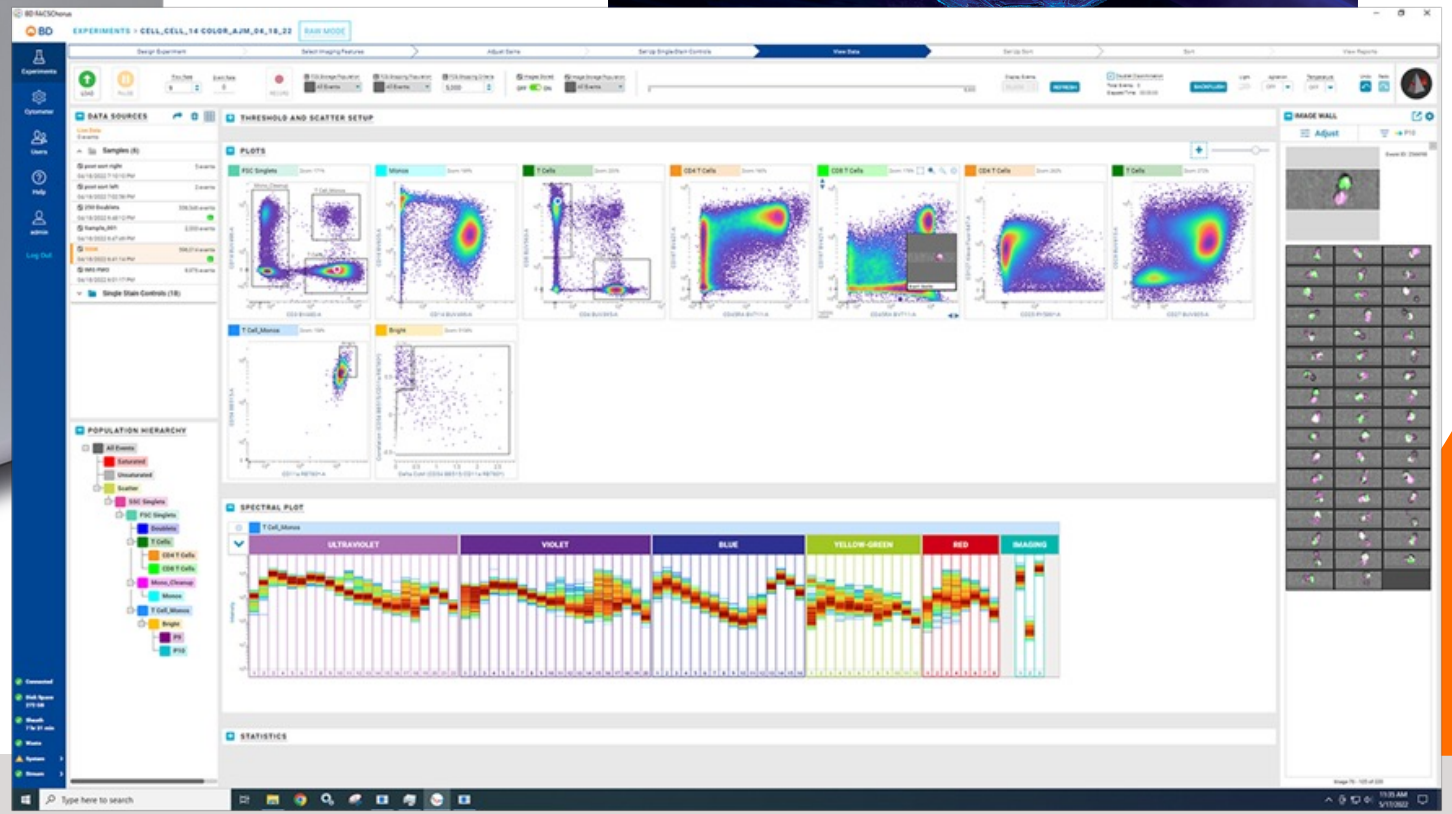
Context on BD Biosciences





Experienced Symptoms and Root Causes

Symptoms

- Slow, incompatibilities, inconsistencies, duplication
- Long front and tail end for releases, most defects found just before release, late (customer) feedback
- High accidental complexity
- Software organization not scalable
- Complacency

Root Causes

- Siloed and hardware-centric program view of software
- Large batch processing for releases
- Highly coupled software/point solutions
- Focus on short-term features only

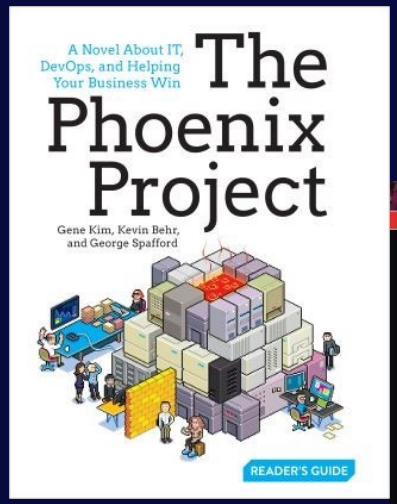
Strategic Partners to Accelerate Change:

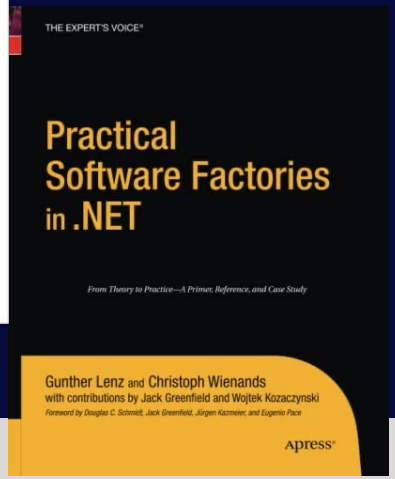
/thoughtworks

SOFTWARECPR®
CRISIS PREVENTION AND RECOVERY, LLC

Digital Transformation “Phoenix” style

- Need to innovate faster
- Do more with less
- Lean and Agile execution
- Comply with regulatory requirements





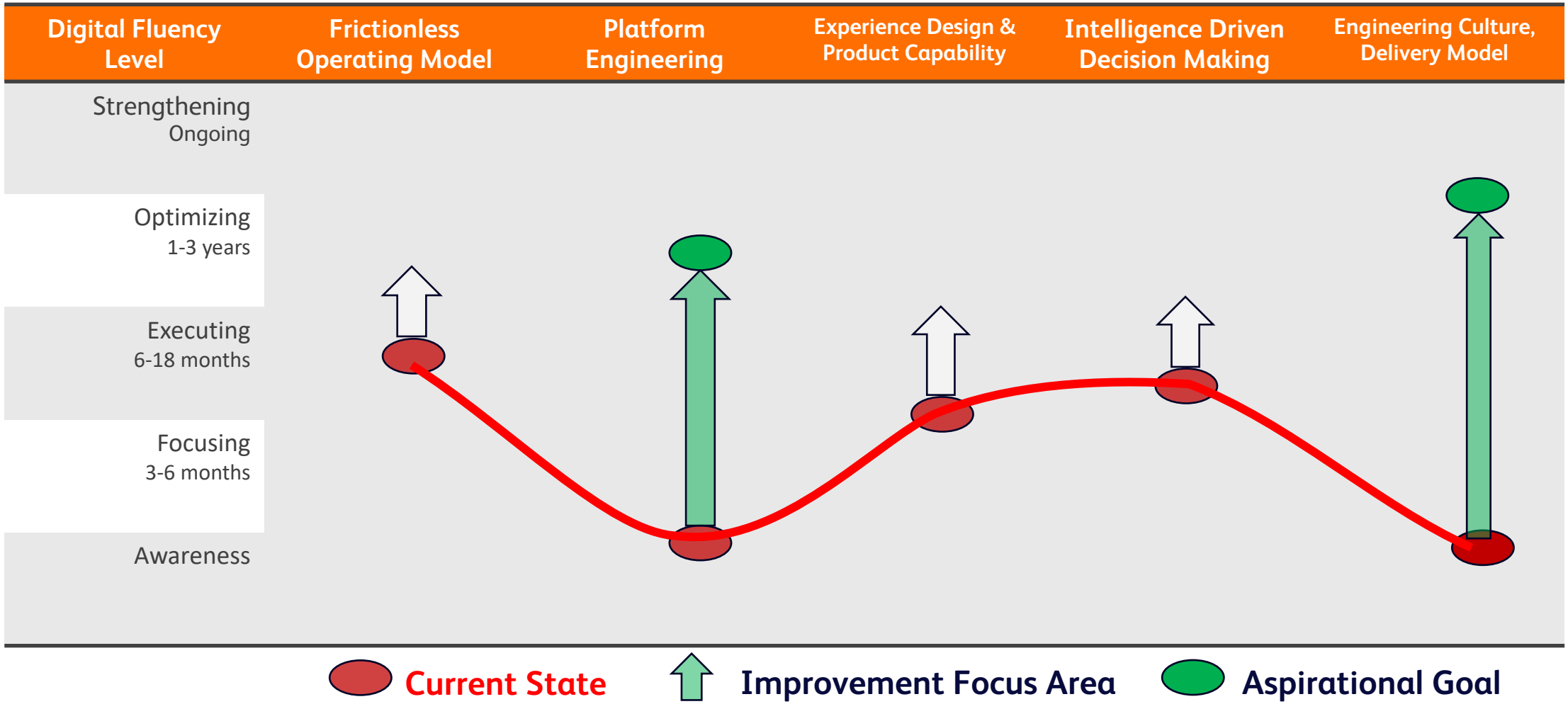
The First Way: Flow/Systems Thinking

- Define current status and areas of improvement
- Lightweight value stream mapping for new product development
- Add the software portfolio view

The First Way: Flow/Systems Thinking (1/4)

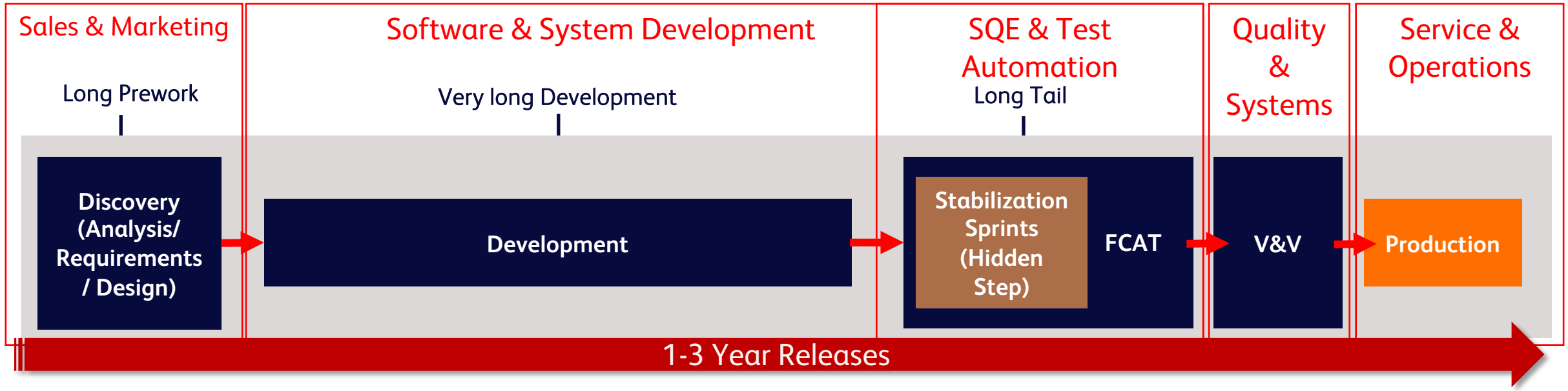
Digital Fluency Model of the Organization*

* https://www.thoughtworks.com/en-us/digital-fluency



The First Way: Flow/Systems Thinking (2/4)

Lightweight Value Stream Mapping **for New Product Development**



Root Causes:

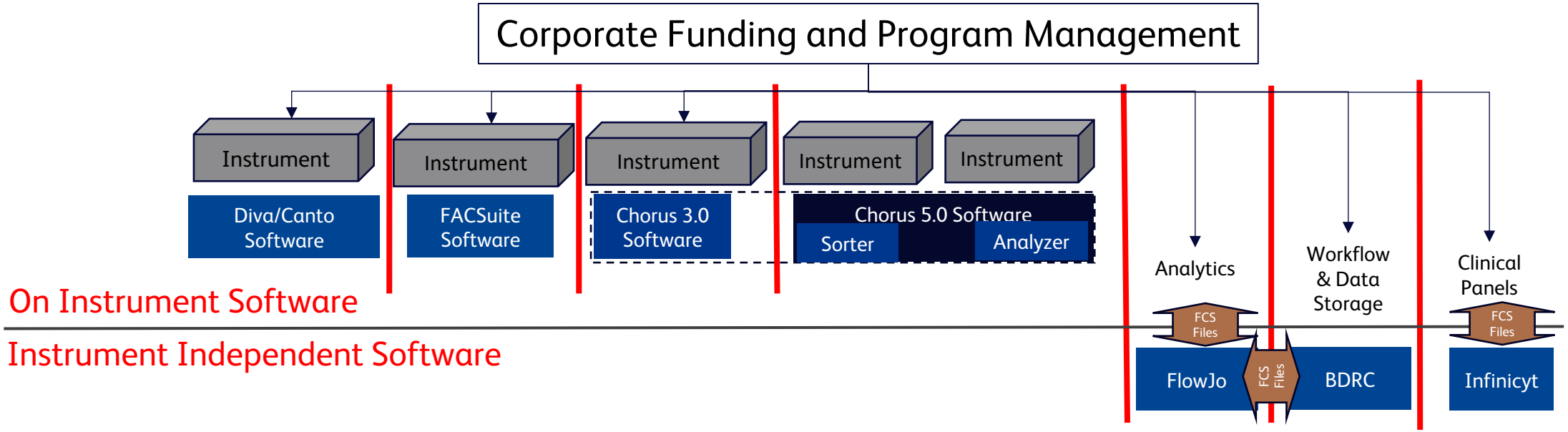
- Large batch processing and long Scrumerfall release cycles
- Many hand-offs
- Attributes of “Dark Agile” https://bit.ly/3F5bDAE

Handoff

FCAT Feature Complete & Acceptance Test

The First Way: Flow/Systems Thinking (3/4)

Software Portfolio Analysis



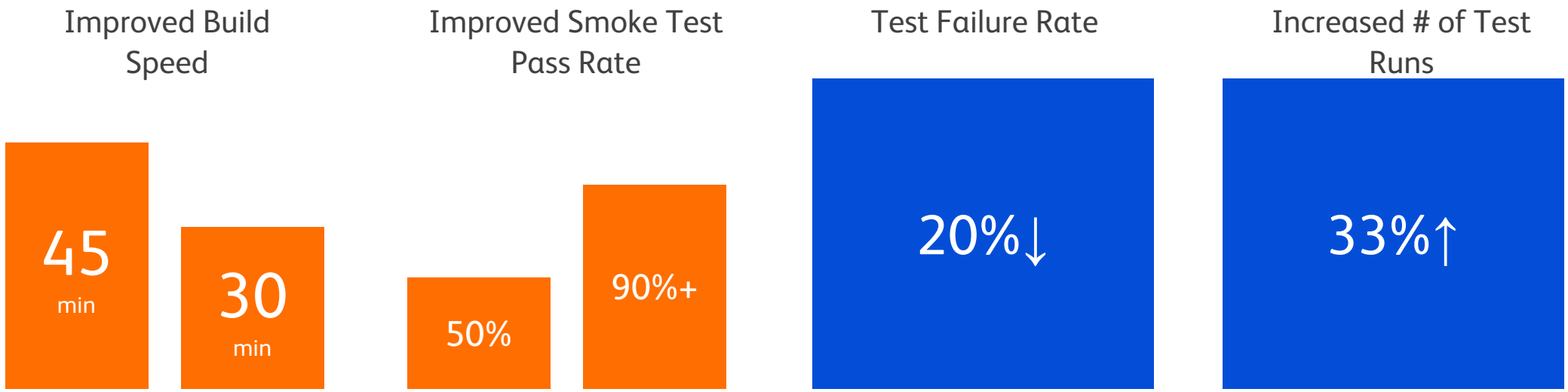
Root Causes:

- Siloed and hardware-centric program view of software
- Conway's law is real! https://martinfowler.com/bliki/ConwaysLaw.html
- Lack of Platform Engineering https://bit.ly/3Vrg2mE

The First Way: Flow/Systems Thinking (4/4)

Improvement Examples to Reduce Defect Leakage

- Revamped test strategy and roadmap https://martinfowler.com/articles/practical-test-pyramid.html
- Focus on test automation https://github.com/robotframework/
- More stringent Definition of Done and hard gates for check-ins

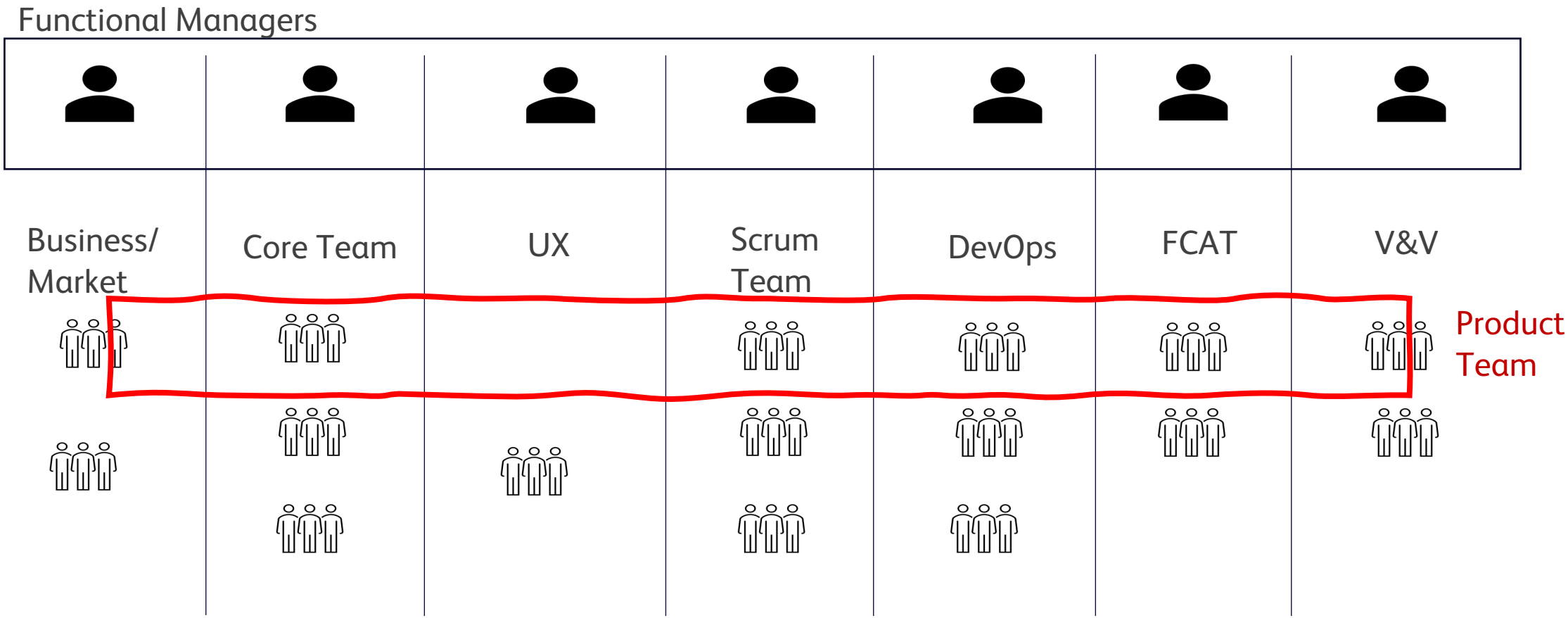


The Second Way: Amplify Feedback Loops

- Inverse Conway Maneuver
- Reduce Batch Sizes and Increase Feedback
- Shift Left

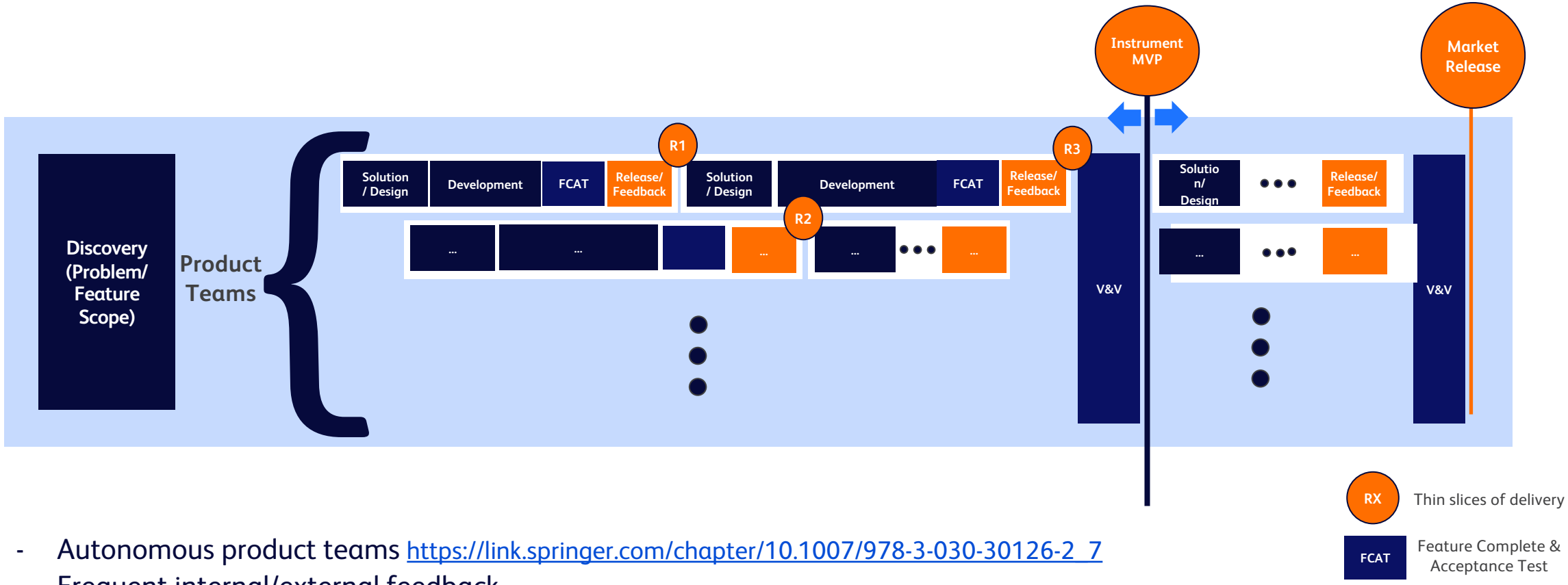
The Second Way: Amplify Feedback Loops (1/3)

Inverse Conway Maneuver



The Second Way: Amplify Feedback Loops (2/3)

Accelerate Feedback Cycle

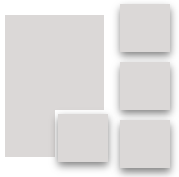


- Autonomous product teams https://link.springer.com/chapter/10.1007/978-3-030-30126-2_7
- Frequent internal/external feedback
- Increase software delivery and process automation

The Second Way: Amplify Feedback Loops (3/3)

Shift Left

Modularize



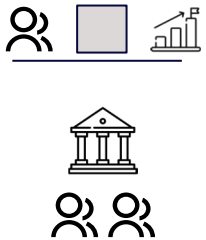
Create scalable organization

~

Modularized, evolutionary architecture

https://evolutionaryarchitecture.com/

Empower



Loosely coupled highly aligned product teams

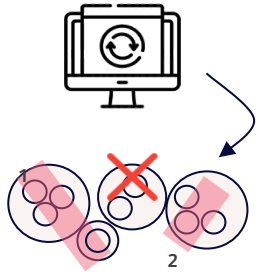
https://hbr.org/2018/05/agile-at-scale

~

Alignment via Lean Value Tree

https://thght.works/3F85raM

Enable



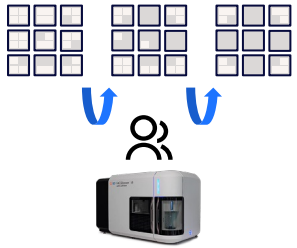
Decoupling software from instrument release cycle to enable small MVP

~

DevOps Improvements

https://bit.ly/3VvS21W

Iterate



Frequent software delivery release cycle

~

Increase quality by reducing batch size

~

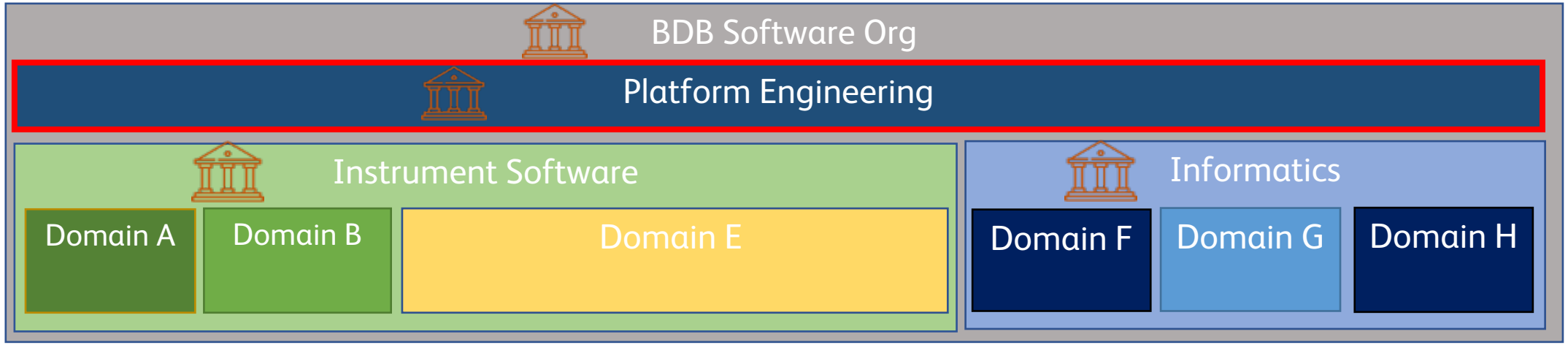
Faster customer feedback

The Third Way: Experimentation and Learning

- Enable Software Developers and Improve Customer Experience
- Create Scalable Software Organization
- Define Governance Model

The Third Way: Experimentation and Learning (1/3)

Enable Software Developers and Improve Customer Experience



The creation of a Platform Engineering Team with two main goals:

1. Explore and deliver on the value of additional Customer Experience across domains and products
2. Create efficiencies in Software Delivery to decrease Time to Market by
   a) Emergent reuse http://engineering-principles.onejl.uk/architecture/Design_for_Emergent_Reuse.html
   b) Additional process automation
   c) Developer enablement

New Governance

The Third Way: Experimentation and Learning (2/3)

Create Scalable Software Organization

Defined by Leadership Team	Defined by Autonomous Teams

• North Star
• Organizational Goals
• Organizational Structure
  • S/W Core Teams
  • Scrum Team(s)
• Governance Principles
• Stakeholder map/ Communication plan
• Reporting internal/external
	• Goals/Objectives that align with North Star https://openpracticelibrary.com/practice/lean-value-tree/
	• Measures of Success
	• Roadmap/ Timeline
	• Stakeholder map/ Communication plan
	• Dependencies on other teams
	• Customer Success (Product)
	• Architecture (S/W, DevOps, SQE)
	• Tech Debt and quality goals

S/W Team

Sponsor:

Engineering Lead :

Architect :

PO/UX Lead:

PM Lead:

Quality Lead :

Charter

- Planning and Coordinating dependencies
- Plan scrum team distribution
- Ensure and report on measurable progress and blockers toward organizational Goals
- Alignment with organizational goals/ objectives

Activities

- **Every 6 Months:**
  - Align on goals/ objectives with Software LT team
  - Align on the roadmap/ timeline with the LT team and publish at (TBD)
- **Every 3 Months:**
  - Release deliverables
- **Every Month:**
  - Monthly check-in with Software LT team (Periodic review cycle)
    - Roadmap/timeline review
    - Goal/ objective review
    - Risks / Blockers

2022 Cumulative Results

2022 H1 Achievements

80%↑ Improve smoke test pass rate

20%↓ test failure rate

33%↑ Improved build speed

100% pre-prod environment availability

2022 H2 Achievements (Pilot Team)

70%↓ in cycle time

First internal in 3 month release down from 24 months

30%↑ test coverage increase

From 50% to 80%

Successfully onboarded Pilot team for Software Dev methodology

Thank You

Links for Reference

- The First Way
  - Digital Fluency Model: https://www.thoughtworks.com/en-us/digital-fluency
  - The Practical Test Pyramid: https://martinfowler.com/articles/practical-test-pyramid.html
  - Robot framework: https://github.com/robotframework/
  - Conway's Law: https://martinfowler.com/bliki/ConwaysLaw.html
  - Dark Agile: https://bit.ly/3F5bDAE
  - DevOps vs. SRE vs. Platform Engineering: https://bit.ly/3VvS21W
- The Second Way
  - Team Autonomy: https://link.springer.com/chapter/10.1007/978-3-030-30126-2_7
  - Autonomous Product Teams https://link.springer.com/chapter/10.1007/978-3-030-30126-2_7
  - Evolutionary Architecture Architecture: https://evolutionaryarchitecture.com/
  - Architecture Domain Decomposition: https://bit.ly/3AMvUII
- The Third Way
  - Loosely coupled highly aligned product teams: https://hbr.org/2018/05/agile-at-scale
  - Lean Value Tree: https://openpracticelibrary.com/practice/lean-value-tree/
  - Alignment via Lean Value Tree: https://thght.works/3F85raM
  - Emergent reuse http://engineering-principles.onejl.uk/architecture/Design_for_Emergent_Reuse.html