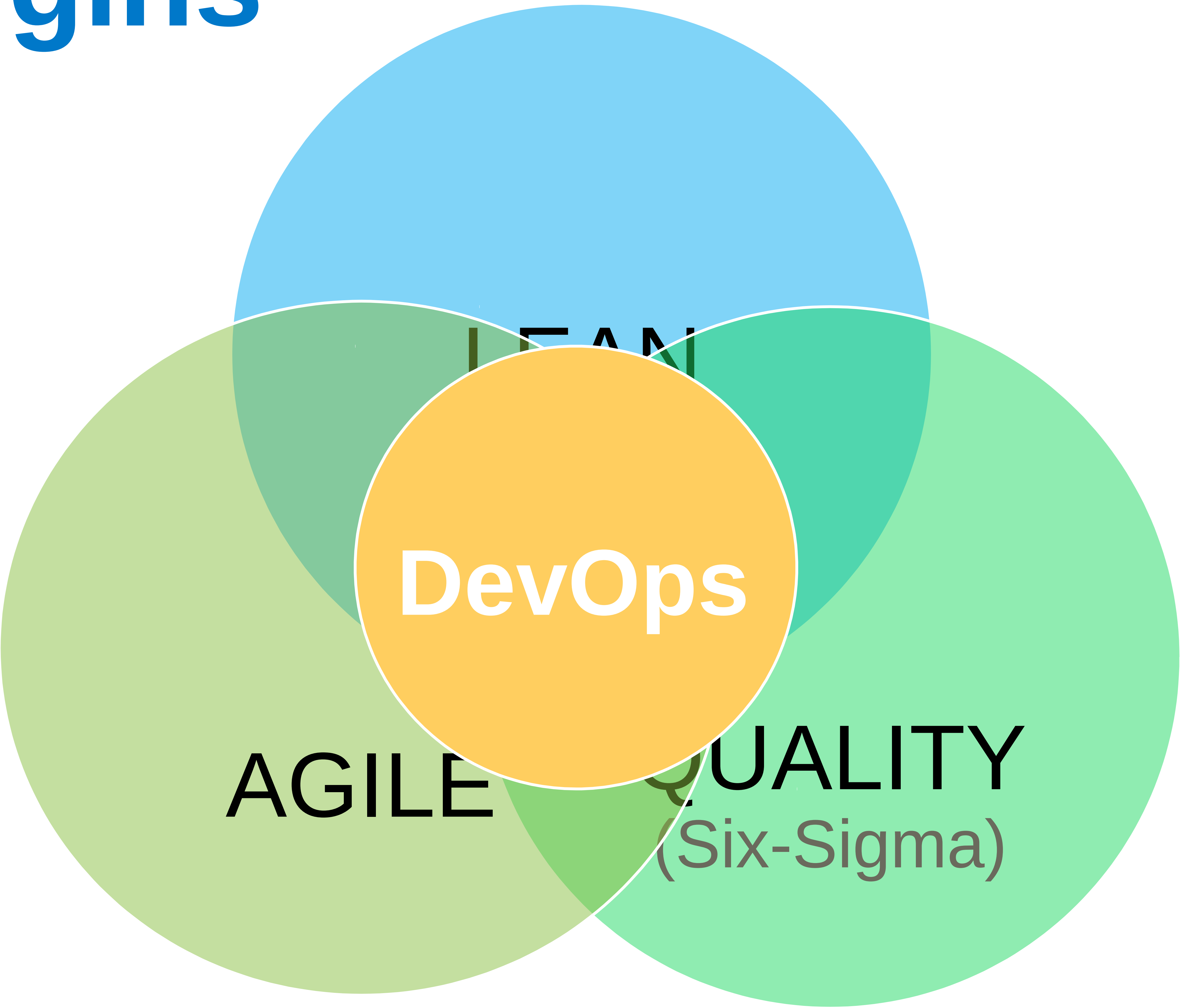
Hardware Inclusive DevOps

Applying DevOps Principals and Practices to Cyber-Physical Systems

Marguerite Bryan

Sr. Transformation Consultant
MBryan@321Gang.com
321 Gang, Inc

DevOps Origins



What is DevOps ?

Early DevOps brought IT and Operations functions together with a **shared goal** to improve the flow of code from desktops to productions systems.

Today, DevOps has expanded to encompass the **unification, optimization and automation** of the E2E functions, tools and processes which contribute to the creation and delivery of value.

The DevOps Catalyst

Agile & DevOps movements were catalyzed by the race for web / mobile app's to enable the shift from traditional [brick and mortar] business models to the **integrated, omni-channel, highly mobile and interactive customer engagement models** of our digital era.

This **software-centric digital revolution** drove the accelerated adoption and evolution of Agile, XP and DevOps practices, leaving HW (unintentionally) somewhat on the sidelines, as more of an outlier.

Hardware is different !

Software is extremely **malleable**

- new functionality is relatively easy to add, or change

Hardware is not as easily built, or repurposed

- given the cost of change is significantly higher, design has traditionally been driven by **upfront** architecture and engineering decisions

But, this distinction has led to a common misunderstanding that DevOps is not suited for hardware-inclusive systems

Debunking the MYTH !

Yes, **DevOps** <u>can</u> be leveraged for hardware inclusive systems:

- some DevOps practices can be **applied directly** in development of cyber-physical systems *– we will review these*
- some DevOps practices can be **adapted** for hardware-inclusive systems *– we will review these*
- many of the contemporary / emerging engineering tools, techniques and practices are *accelerating the inclusion* of hardware in DevOps programs, workflows, and cultures *– we will review some of these*

Hardware Inclusive DevOps

O rganization
A rchitecture
S imulation
I teration
S ynchronization

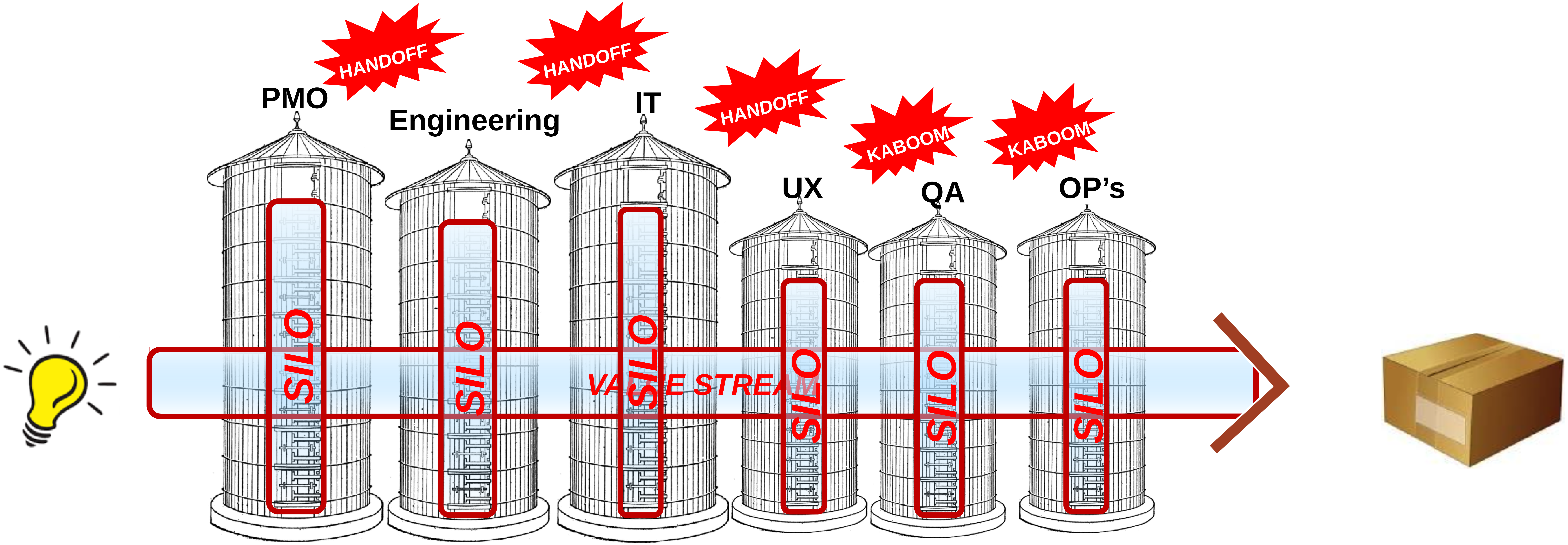
mnemonic

OASIS

Organization: Structures

Traditional Organizational Structures

Agile Organizational Structures



Organization: Culture

INDUSTRY BUZZ

"The Fifth Discipline"

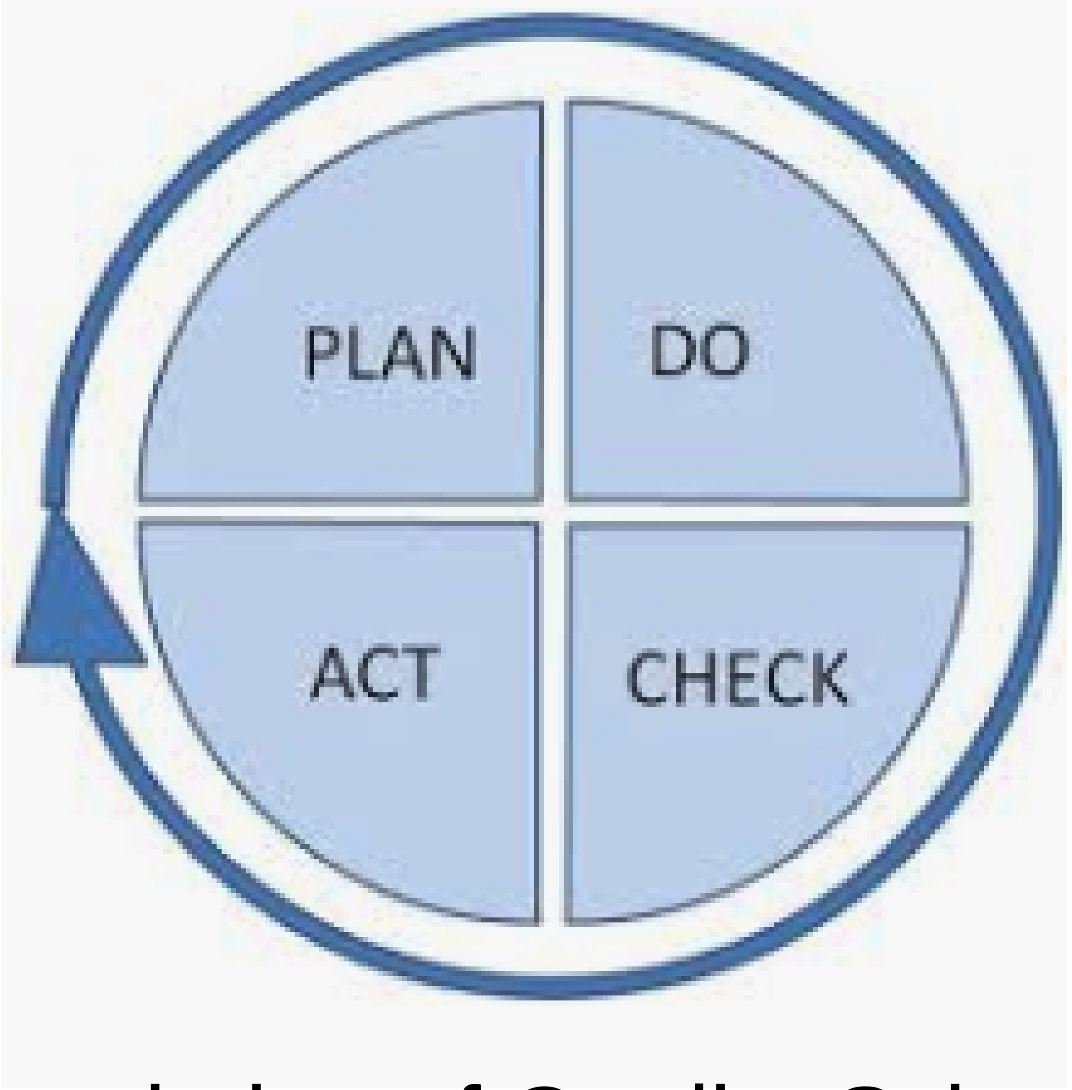
"Quality Culture"

"Continuous Improvement"

"Learning Organization"

Deliberate adoption of disciplined scientific method,
rather than assumptions and guesswork
in continually testing hypotheses
and diagnosing impediments

Deming Cycle



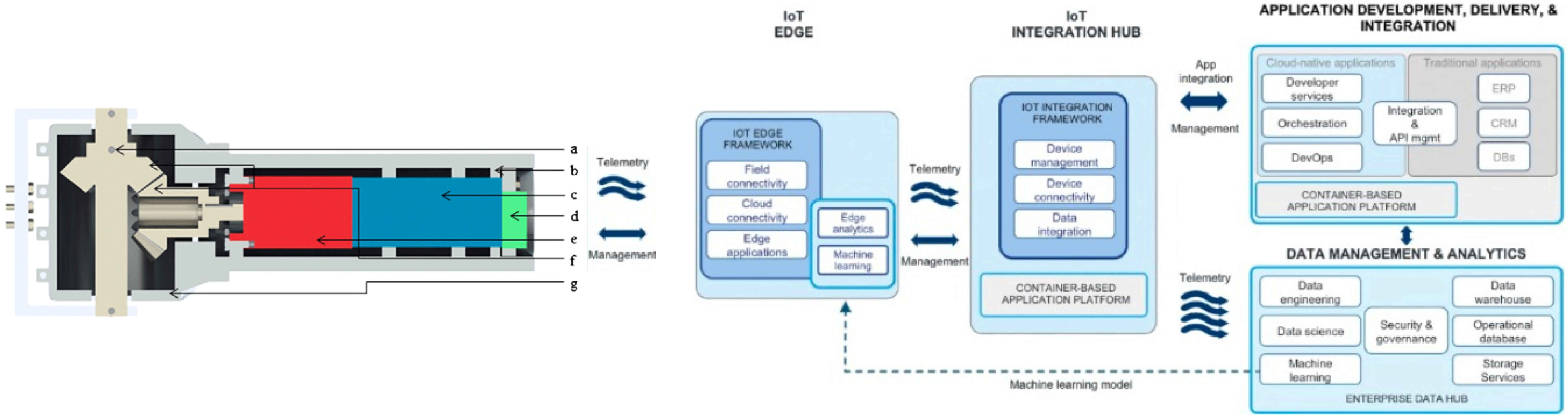
The foundation of Quality Science, and Agile Methods including DevOps

3 things you can do today ?

1. Learn more about Value Streams and Mapping
2. Build bridges to other functional areas of your organization
3. Translate assumptions into hypotheses to be tested,
   - *Scientific Method / Deming Cycle -*

Architecture

- emergent (set-based) design: allow for early '*uncertainty*' with disciplined experimentation and continuous learning cycles
- modularity: component decoupling and layers of abstraction, enabling component-level replacement, upgrades and fixes
- API's and interfaces well-defined to highest standards
- serviceability: enabling ability to alter functionality post deployment
- operability: logging/telemetry enabling faster problem isolation



Simulation

"Simulation" in the **broader context** of low-cost *modeling* of a component's look, feel and, behavior so we can test assumptions early and often – when the cost of change is lowest

- vastly reduce lead time for hardware components needed in the validation of solution design, development and integration
- reduce cost of experiments and learning, enabling set-based design
- learn early and often, across teams of teams
- support an iterative development approach with simulation models which can be continuously expanded from lowest to highest fidelity

Let's look briefly at Prototyping, Simulators and Emulators

- Prototyping

Any **accurate-scale**, partially or completely functional version of the designed component, created using simpler, **less expensive production processes** than required for final product.

- useful in early experiments when detailed data for more complex simulation is not available
- can be useful in testing physical form and function with end users
- generally in early phases of deign, develop, test, with simulation of broader functionality and integrations later

- Prototyping Examples

Looks-like Prototypes:

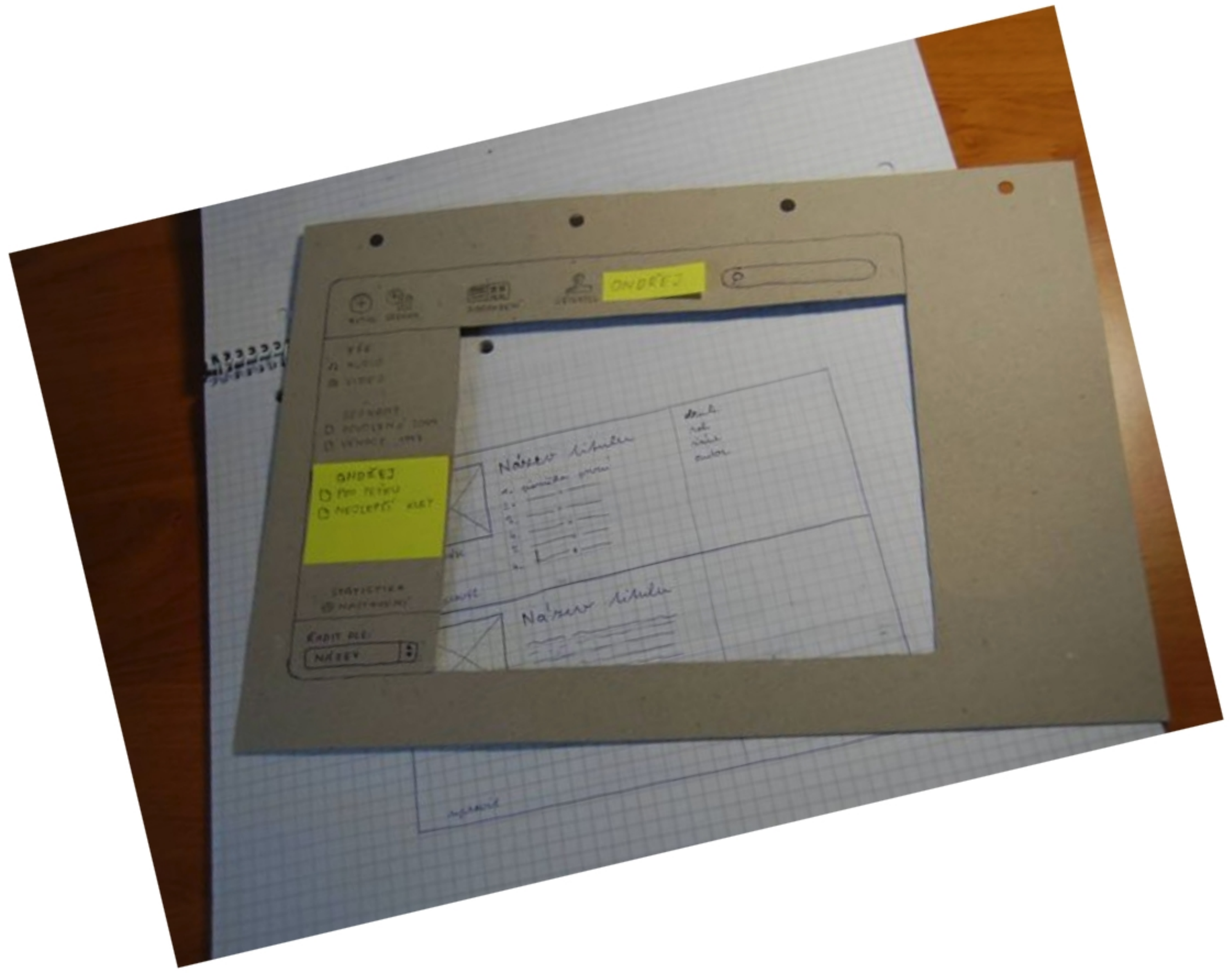
- Foam, Clay, Cardboard
- 3D printing technology - additive: limited to 3D printer resins
- Computer Numerical Control (CNC) – subtractive: can model with desired production plastic

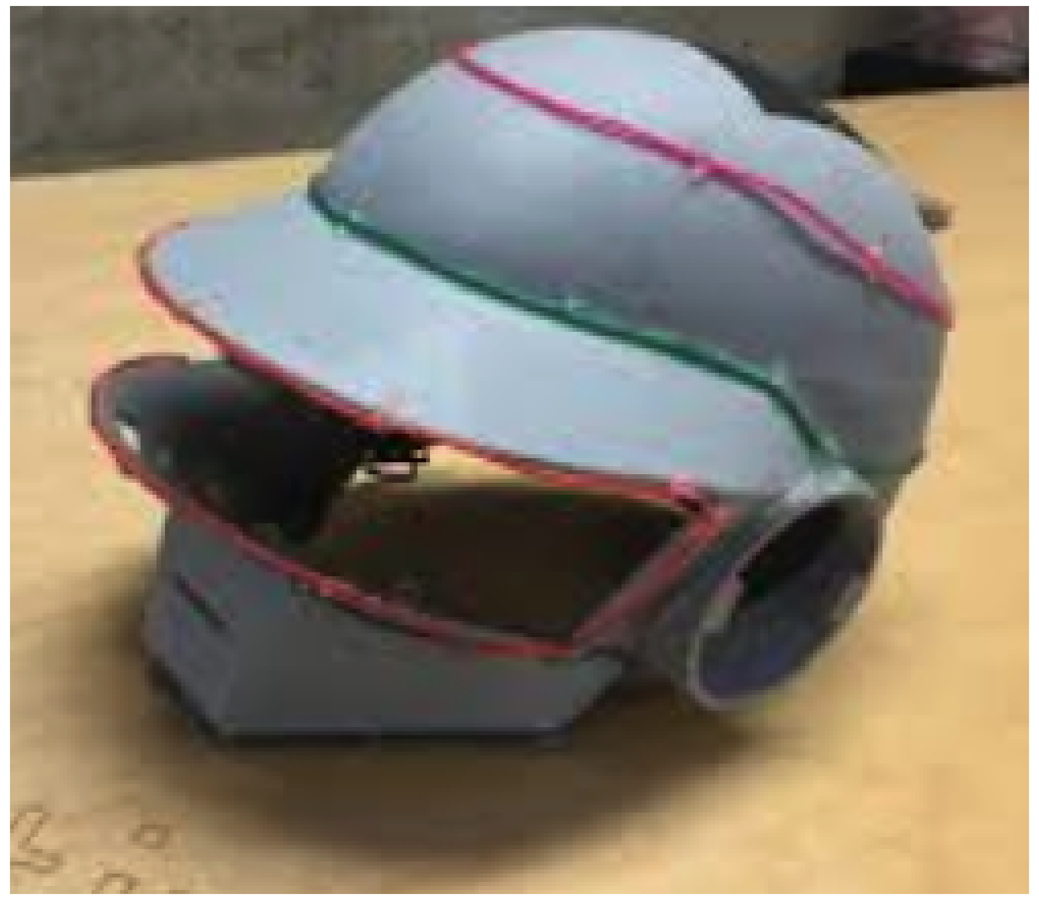
Acts-like Prototypes:

- adding functional behaviors, usually internal electronics
- leverage of off-the-shelf components & development kits *(e.g. Raspberry Pi)*
- expanding then to custom Printed Circuit Board (PCB)

Engineering Prototype: appearance and functionality come together

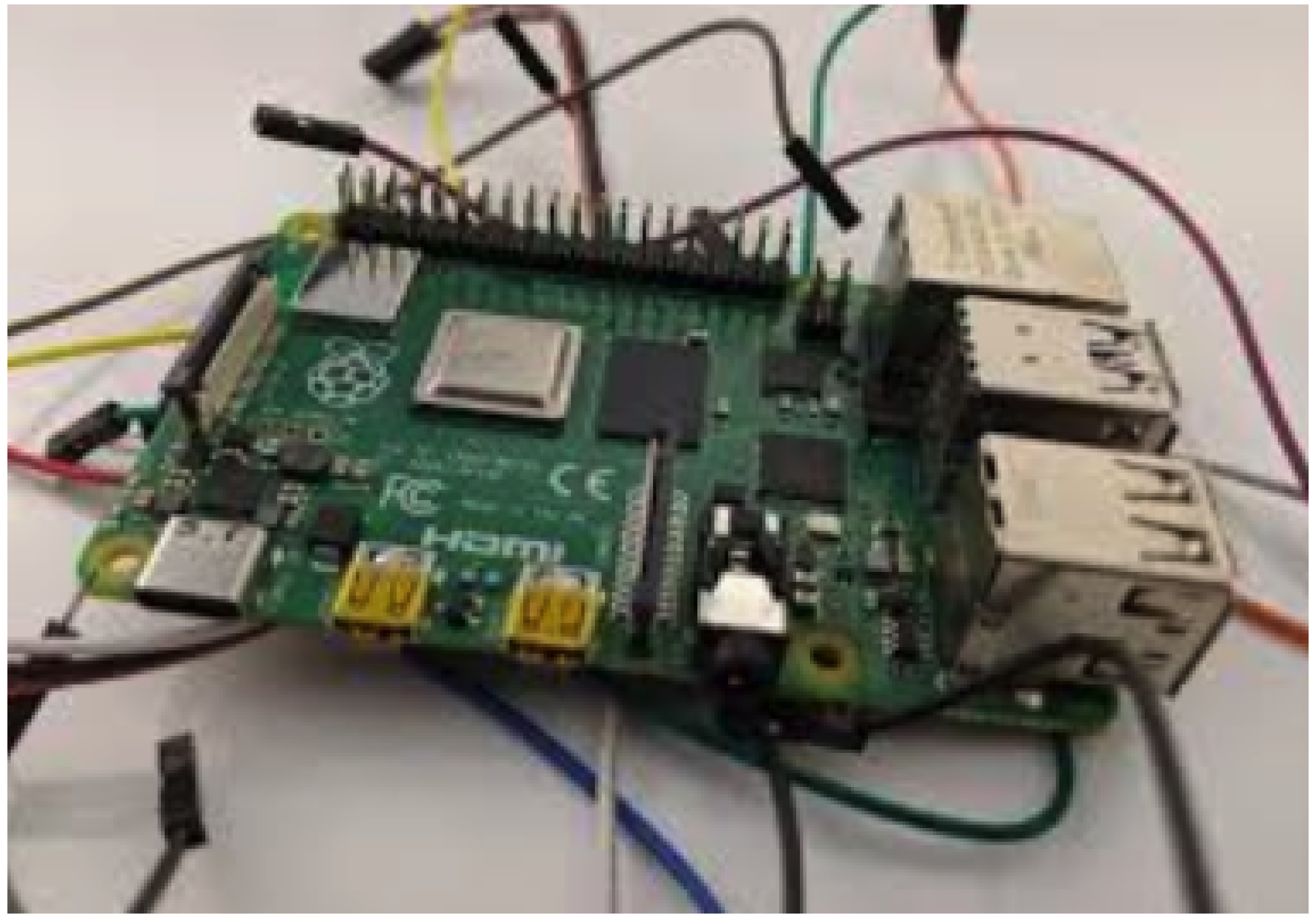
Low Fidelity Prototyping: Paper, Foam, Clay





Higher Fidelity Prototyping: 3D Printing

Functionality Prototyping



Emulation vs. Simulation

Simulators: ***model***

- mimics **outwardly** observable behavior of the target device
- internal state of emulator does not have to accurately reflect the ‘target’
- written in higher level languages

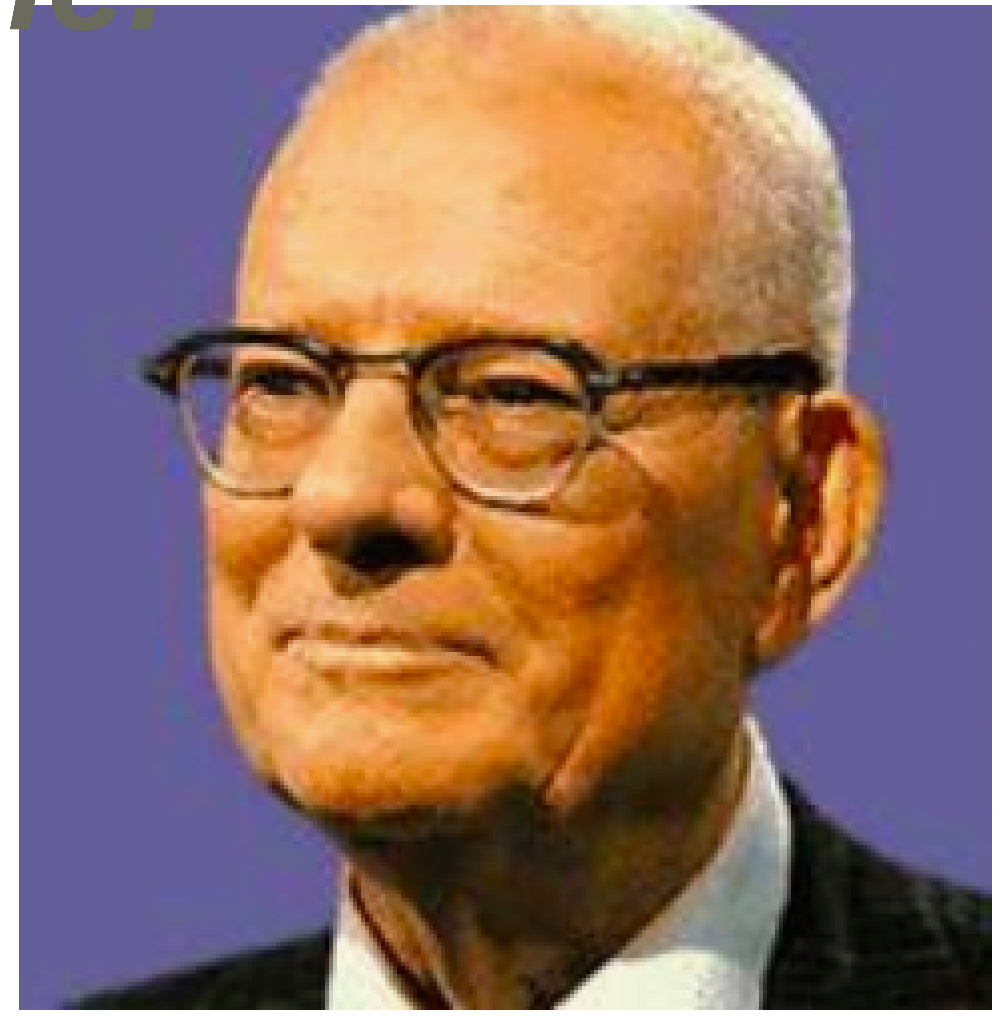
Emulators: ***replicate***

- modeling the **internal, underlying state** of target
- useful in early design and **debugging** of components which are integrated into the system
- written in HDL

Systems Thinking

All of these simulation techniques enable earlier, more frequent

integration and learning,
optimizing the system as a whole.



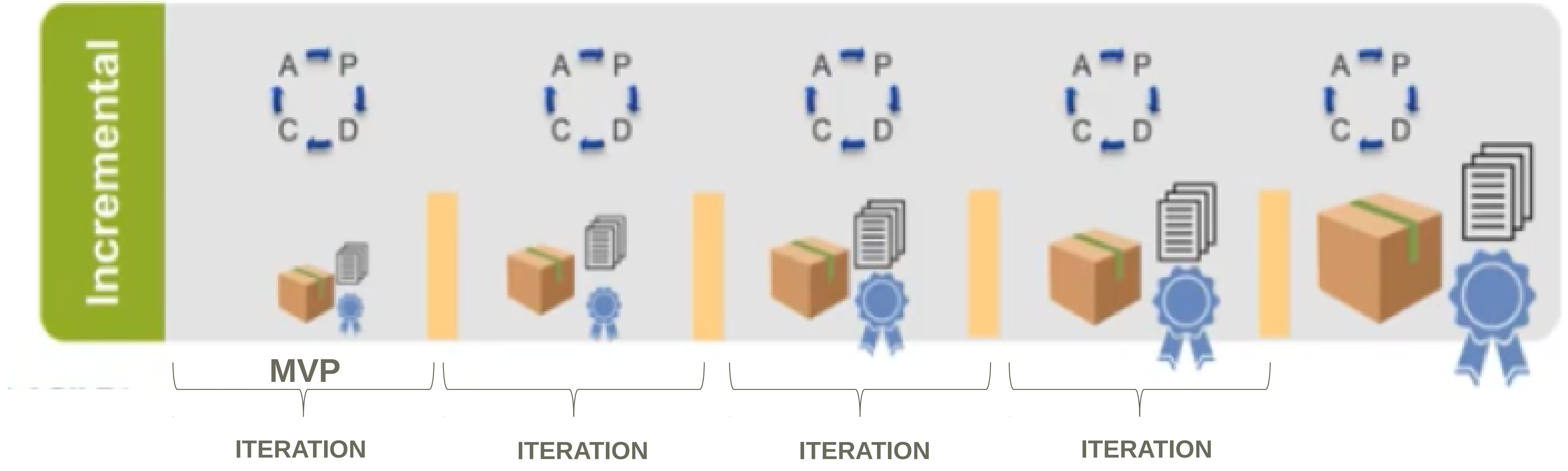
‘Left to themselves,
component becomes selfish…
and starve the whole’

W. Edward Deming

Iterate

What:

- standard, fixed-length timebox, where Agile Teams deliver some incremental value in the form of working, tested systems
- the basic building block of Quality, Agile and DevOps practices
- enabled through the use of various hardware simulation techniques

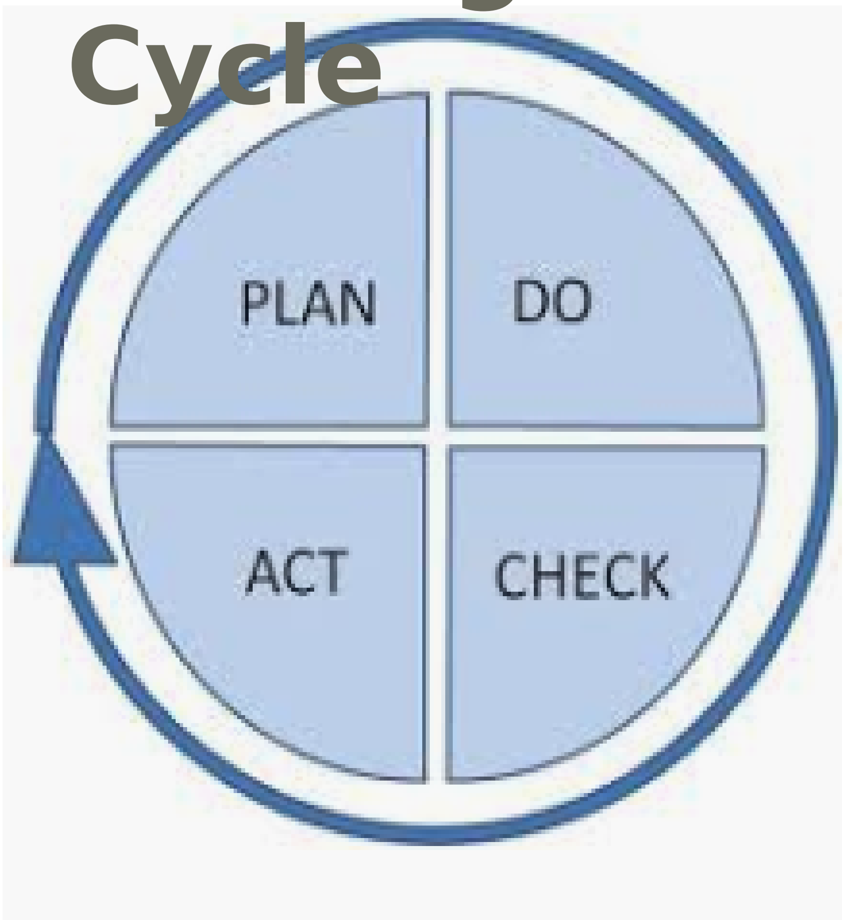


Iterate

Why:

- drives smaller batches of work, enabling fast and frequent feedback cycles for continuous learning
- enables regular integration points for teams of teams
- enables lower cost adjustments and defect resolution
- provides a regular, predictable cadence for testing of technical and business hypotheses

Deming Cycle



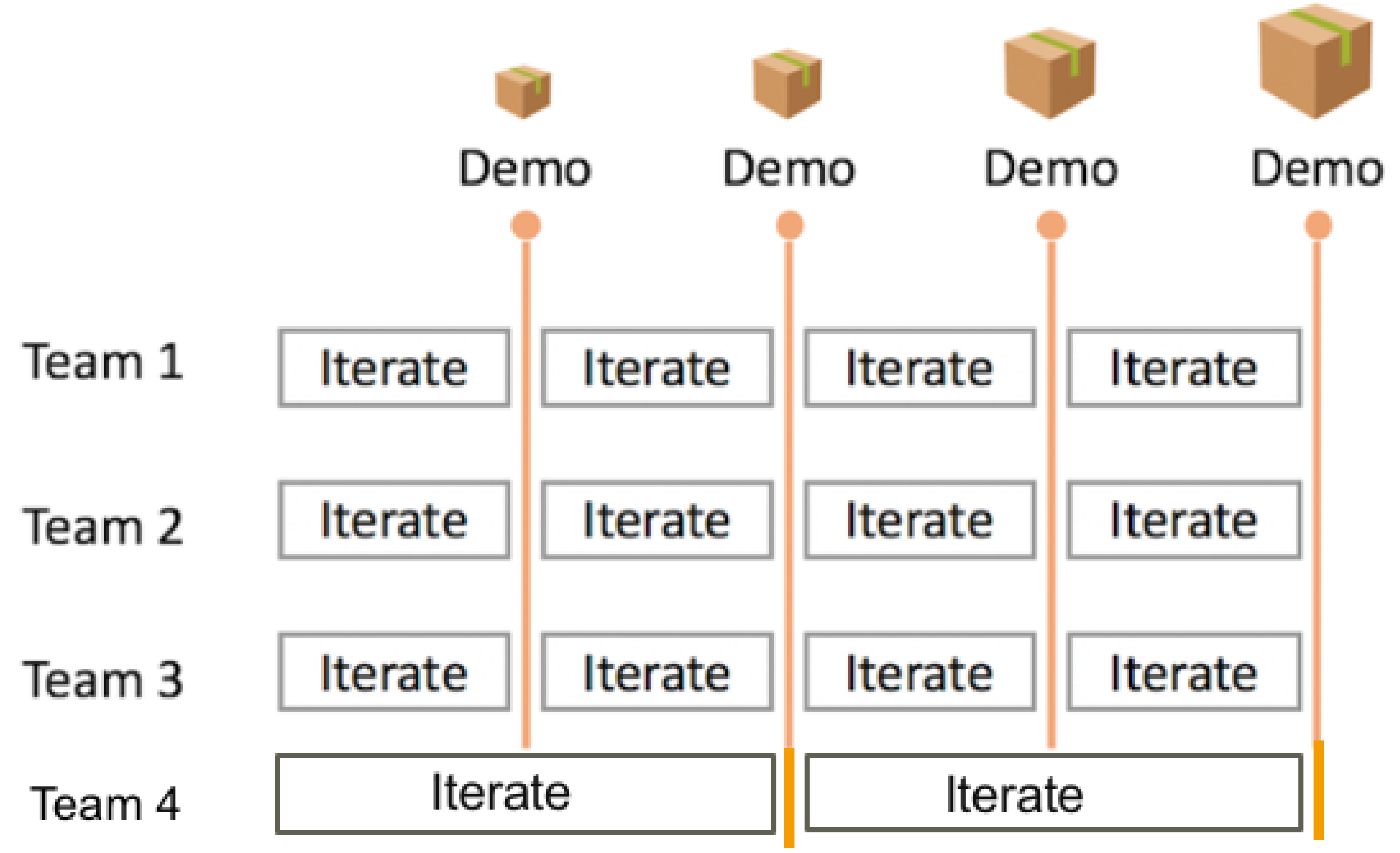
Synchronize

What:

- multiple streams of work (Teams of Teams) coordinated through regular integration, testing and learning events
- predictable cadence, enabled though defined iterations

Note: Synchronization does not require all teams to be on same cadence, as long as regular synchronization (integration) points are identified

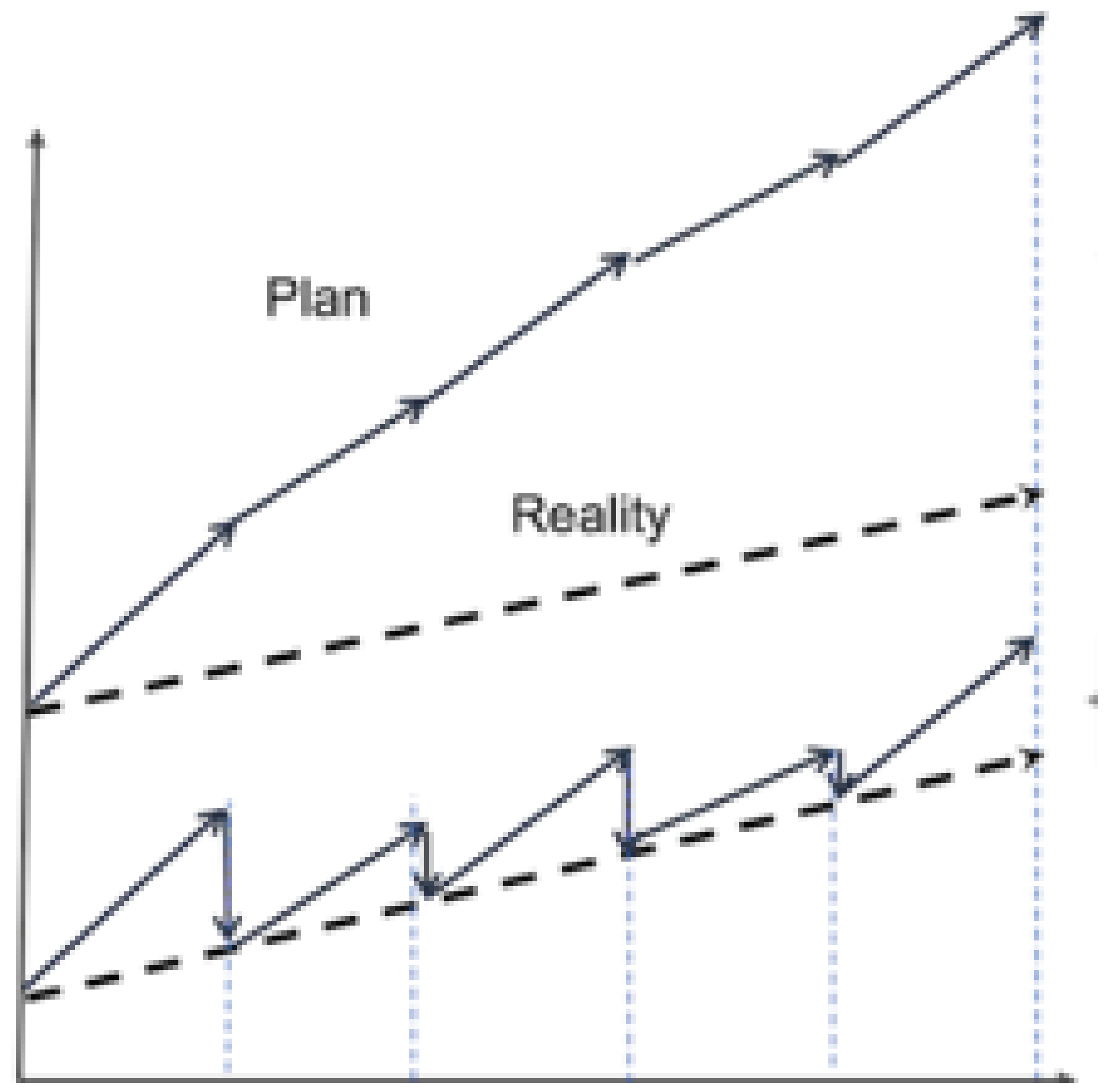
Adjustment for HW



Synchronize

Why:

- earlier, more robust management of dependencies, risk, compliance, security and quality – as we ***shift left*** to incorporate these requirements into earlier phases of the Value Stream



Integration Points

Better together…

Each element we reviewed today will support you in building and realizing early benefits from you hardware-inclusive DevOps programs.

However, it is when these practices are brought together that organizations can achieve the **most important outcomes**.

Thank You !

Useful Resources:

- Recorded DevOps Webcast: Expand Your DevOps Practice and Your Value to the Organization
- MIT White Paper: Overcome Modern Engineering Challenges: https://321gang.com/OptimizingtheEngineeringLifeCycle
- eBook: Agile Product Development: https://321gang.com/eBook-Agile-Product-Developmenthttps://321gang.com/eBook-Agile-Product-Development
- Recorded Webcast: https://321gang.com/works/the-economics-of-value-delivery-do-the-math

Marguerite Bryan
MBryan@321Gang.com

Agile & DevOps History

- 2001 Agile is Codified in the Agile Manifesto
- 2008 DevOps is Born: *“Birds of a Feather” in Toronto*
- 2009 First DevOpsDays in Belgium
- 2010 Agile inflection point
- 2013 “Phoenix Project”
- 2016 “DevOps Handbook”
- 2018 *DevOps Inflection Point, Forrester*