Expanding DevSecOps to Embedded Systems; Is it possible?

Hasan Yasar

Technical Director, Adjunct Faculty Member

Software Engineering Institute | Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM21-0122

Outline

- Overview of DevSecOps
- HW/SW Development & Deployment

Overview of DevSecOps

What is DevOps?

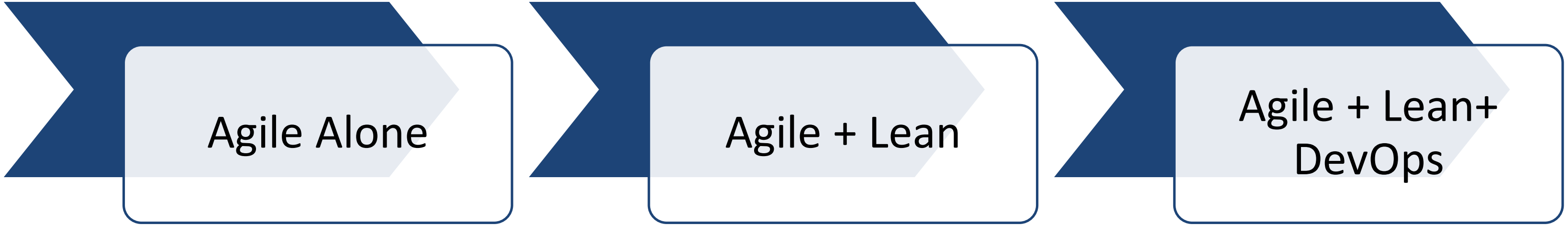
DevOps is a set of principles and practices which enable better communication and collaboration between relevant stakeholders for the purpose of specifying, developing, continuously improving, and operating software and systems products and services [1]

What isn't *DevOps*?

Systems Engineering, Tools, Waterfall

[1] IEEE 2675 DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment

Why Agile *AND* Lean *AND* DevOps?



Agile Alone

Agile + Lean

Agile + Lean+ DevOps

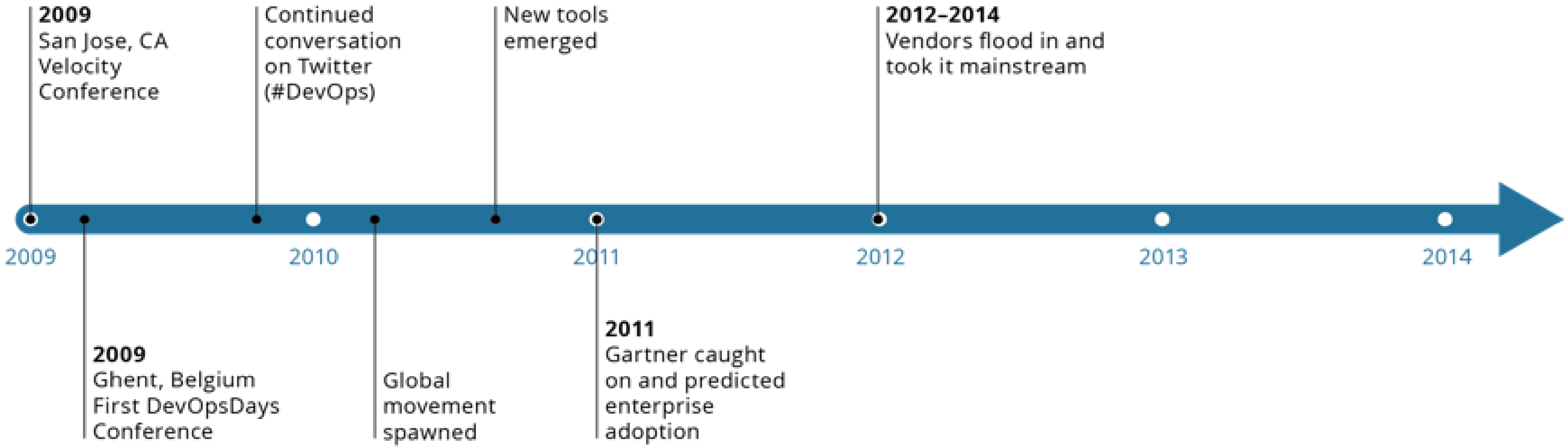
- Tends to focus just on small software teams
  - DOD context typically bigger, more complex
- Tends to assume that direct delivery to customers is feasible

- Adds typical hardware, system, and business/ management teams
- Adds principles and practices that reflect the larger complex system context
- Adds consideration of stakeholders like DT/OT (dev test and opn'l test) and certification

- Adds fast feedback technology infrastructure for continuous architecture, continuous integration, continuous deployment
- Particularly adds to Agile teams' efficiency and effectiveness in execution

DevOps is Newer

- The birth of DevOps was Patrick Debois's desire for "Agile Infrastructure."
- DevOps started as a grassroots movement—of practitioners, by practitioners.
- It caught on, went viral, not because of hype, but because of real results.
- —it's decentralized and open to all.



DevOps is an Extension of Agile Thinking

Agile

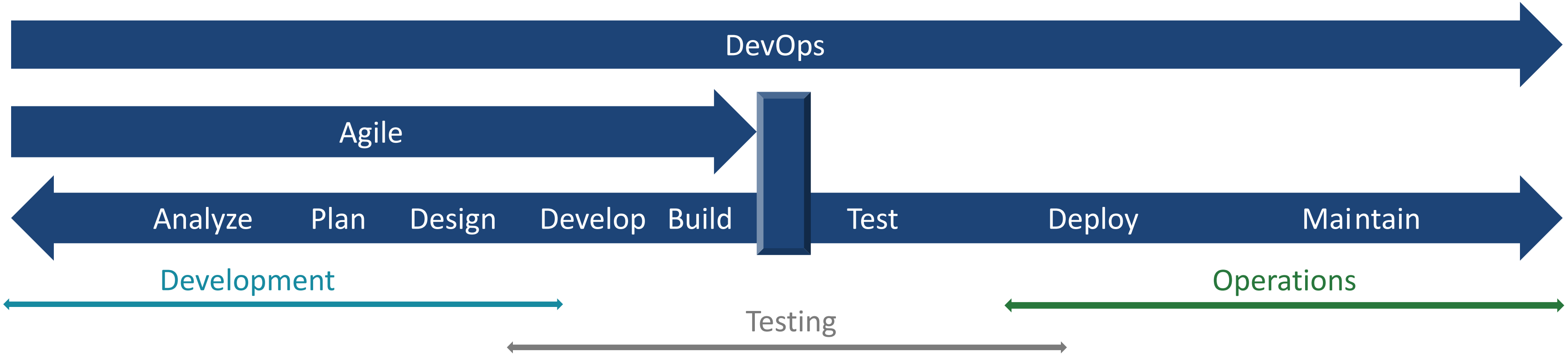
Embrace constant change

Embed customer in team to internalize expertise on requirements and domain

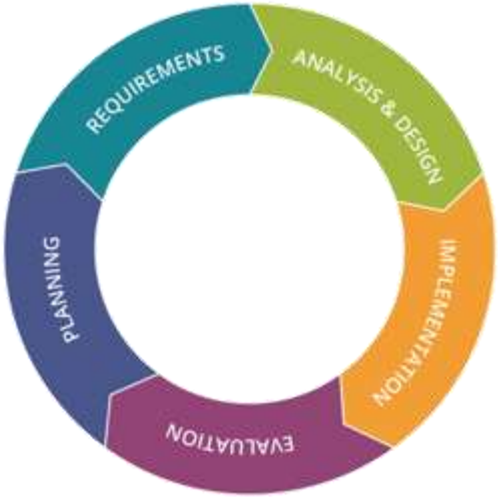
DevOps

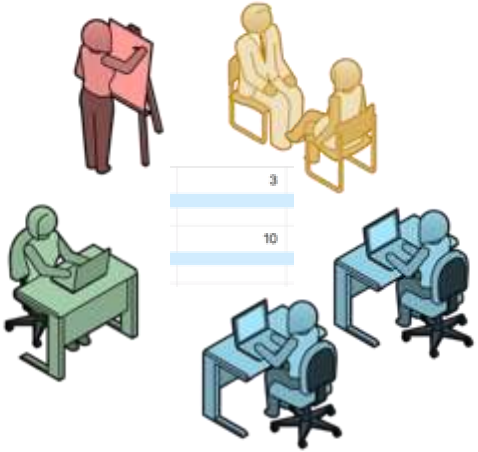
Embrace constant testing, delivery

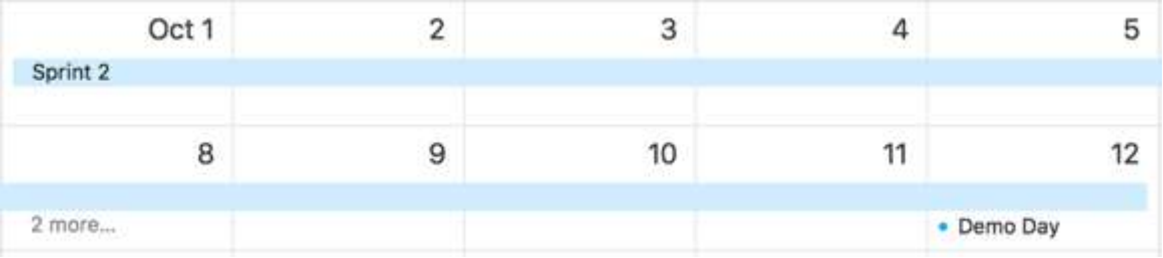
Embed operations in team to internalize expertise on deployment and maintenance



Agile Team Concept



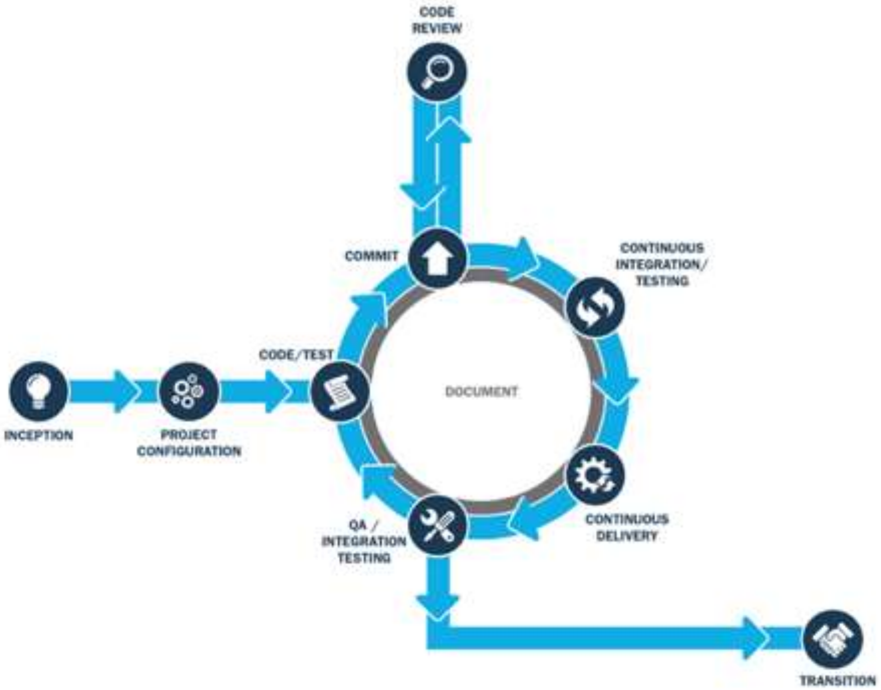




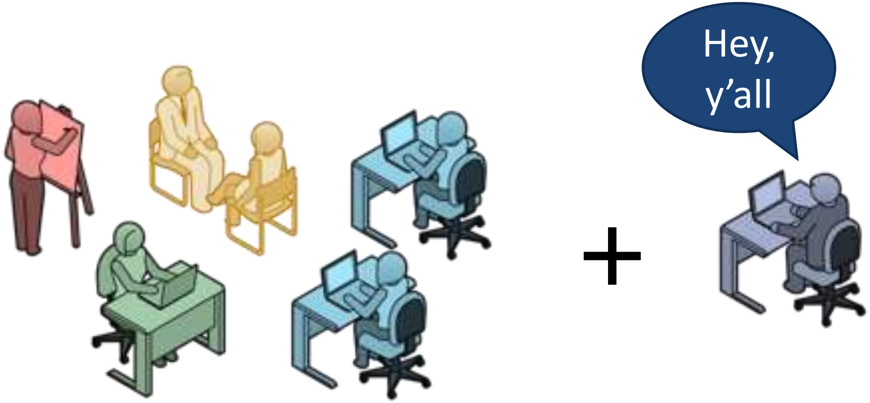
Oct 1	2	3	4	5
Sprint 2				
8	9	10	11	12
2 more...				• Demo Day

- Cross-functional team agrees to what can be accomplished in a sprint
- Teams are incentivized to help each other
- The team provides a demo to the customer at the end of the sprint
- Customer and leadership can correct course during next sprint

DevOps Team



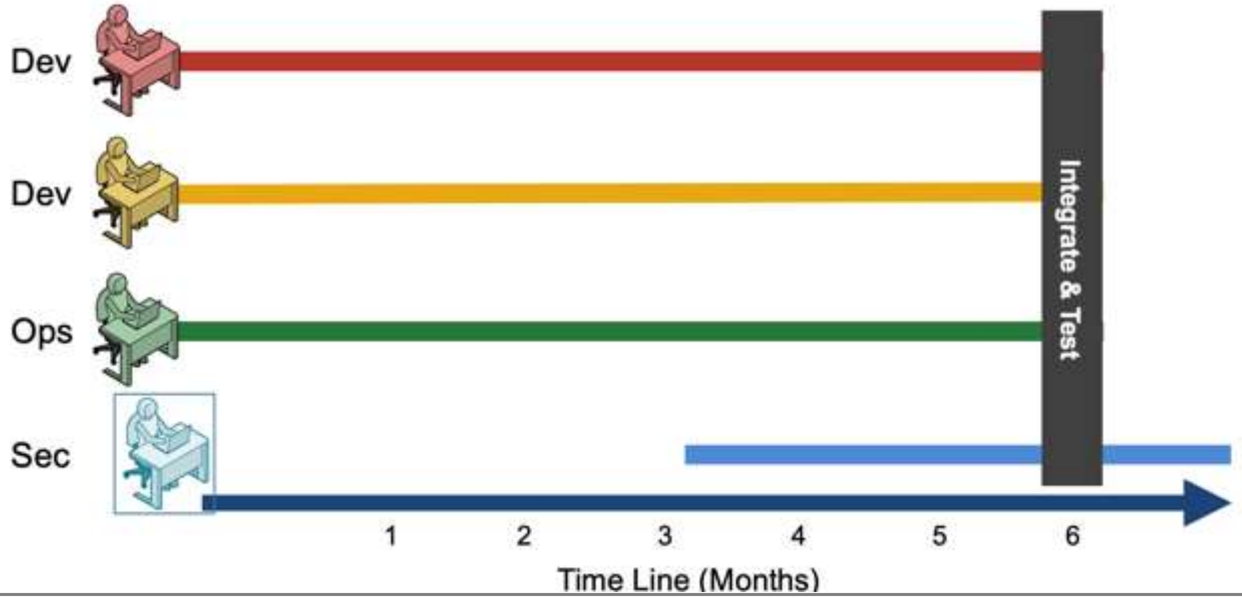
Cross-Functional Dev Team, Including IT Operations



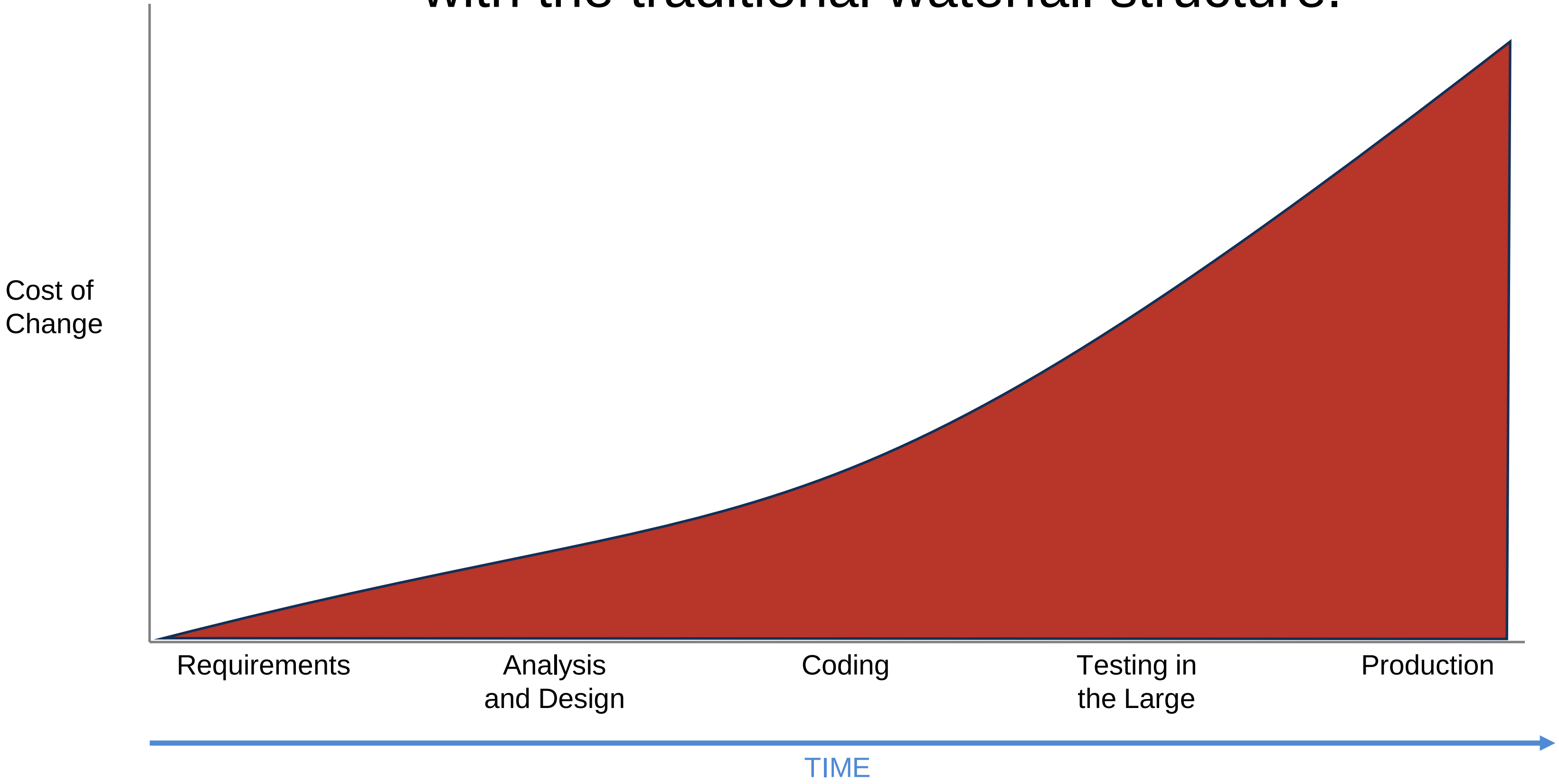
- IT Ops included early in development – deploy to ops-like environment EARLY
- Automation enables fast testing and deployment

Why This Matters: Waterfall Timeline Complications

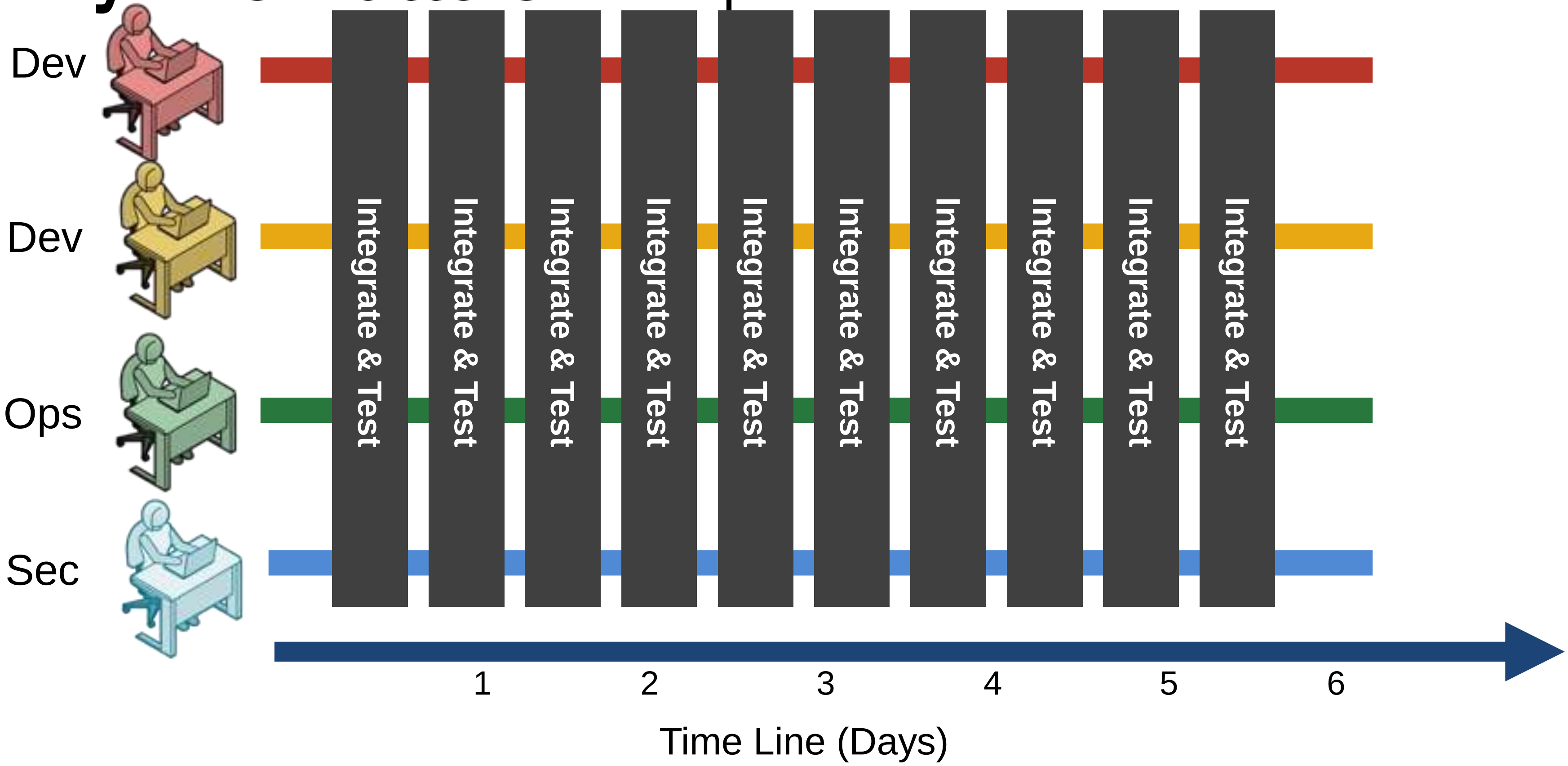
In a waterfall scenario, integration and testing only occurs at the end of the timeline after months of Development / Operations / Security work.



The cost of change increases **exponentially** over time with the traditional waterfall structure.



Why This Matters: DevOps Timeline



Dev

Dev

Ops

Sec

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

Integrate & Test

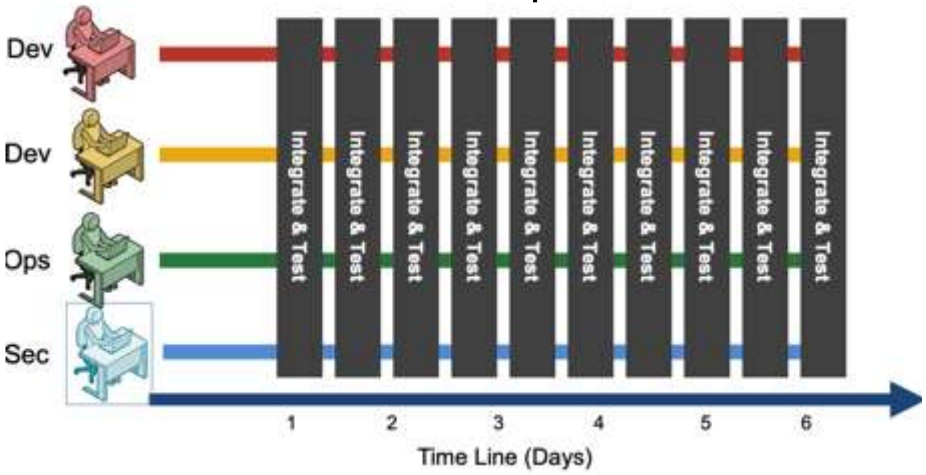
Integrate & Test

1 2 3 4 5 6

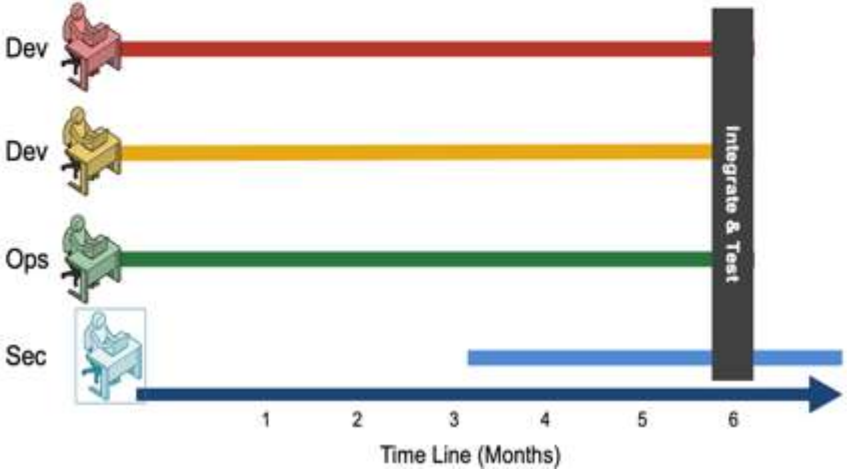
Time Line (Days)

Why This Matters:

DevOps



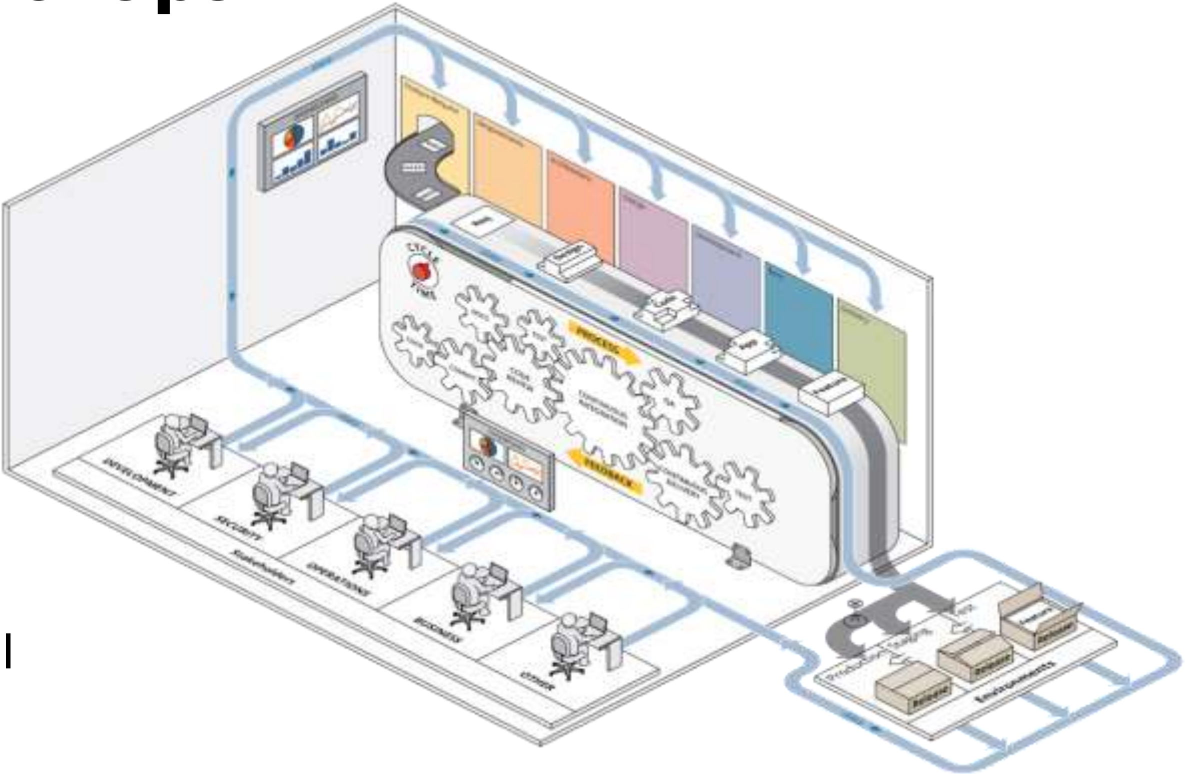
Waterfall Hierarchy



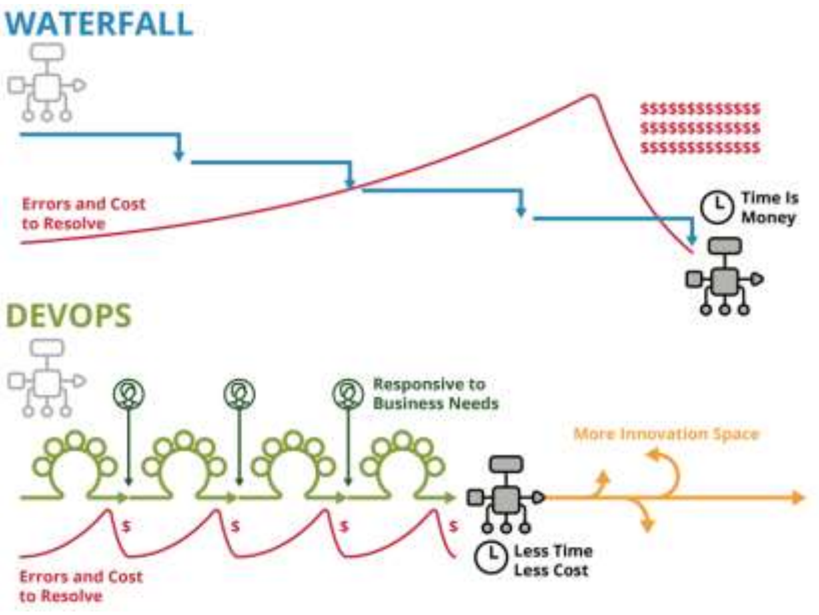
Switching to adopt DevOps practices can reduce the current required time to integration and testing from months to days.

Key practices of DevOps

- Feature to deployment
- Iterative and incremental development
- Automation in every phase of the SDLC
- Continuous feedback
- Metrics and measurement
- Complete engagement with all stakeholders
- Transparency and traceability across the lifecycle



Benefits of DevOps

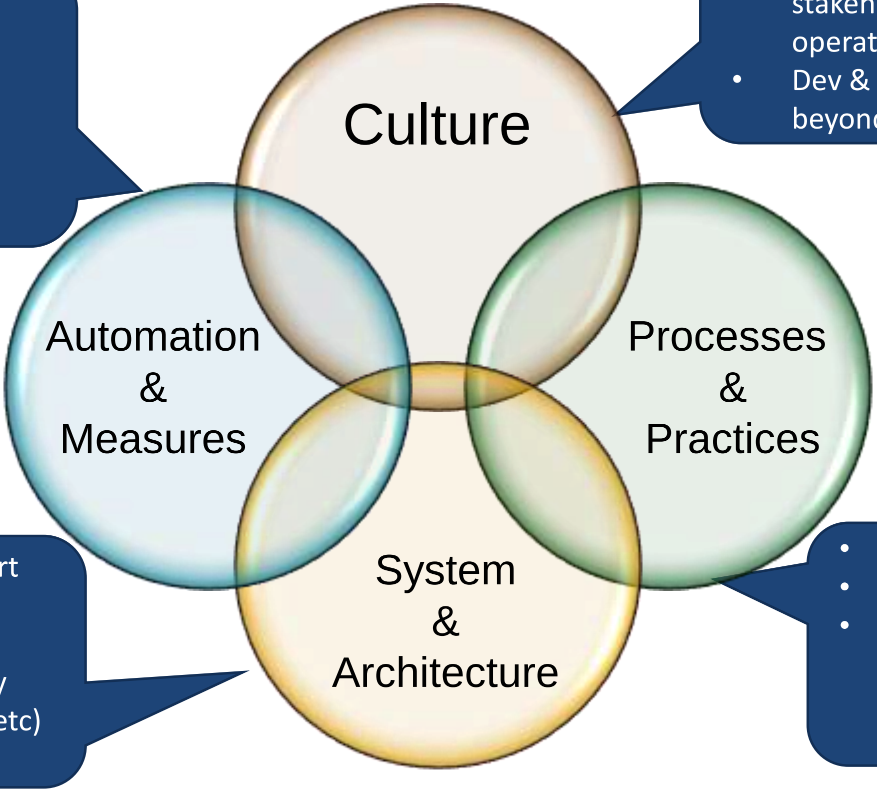


- Reduced errors during deployment
- Reduced time to deploy and resolve discovered errors
- **Repeatable** steps
- **Continuous availability** of pipeline and application
- Increased innovation time
- **Responsiveness** to business needs
- **Traceability** throughout the application lifecycle
- Increased stability and quality
- **Continuous feedback**

Might Seem Simple, but not EASY!

- What Some People Think Boundaries of DevSecOps is!
- Automate repetitive, error-prone tasks
- Static & Dynamic Systems Analysis
- Performance dashboards

- All roles collaborate
- Dev, Ops, Sustainment have stakeholders that understand operational drivers
- Dev & Ops support products beyond delivery



- System architected to support integration and automation goals
- Represents important quality attributes (scalable, secure, etc)

- Value stream understanding
- Whole pipeline accounted for
- Continuous integration, automated test, virtualization, self-serve, scripting, automated deployment…

DevOps Has Four Fundamental Principles

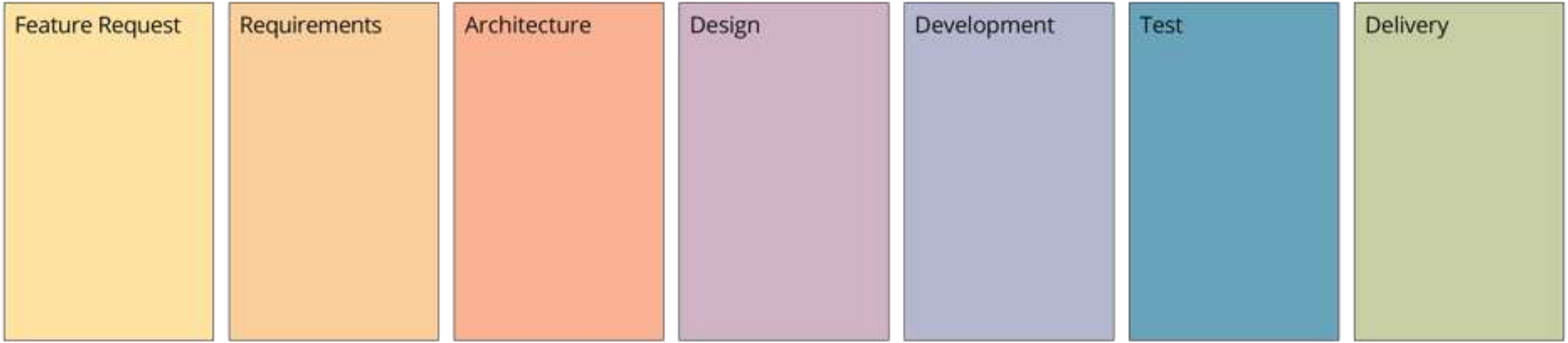
1. **Collaboration:** creating ‘cross-functional’ teams

2. **Infrastructure as Code:** all assets are versioned, scripted, and shared where possible

3. **Automation:** deployment, testing, provisioning, any manual or human-error-prone process

4. **Monitoring:** any metric in the development or operational spaces that can inform priorities, direction, and policy

Reminder: SW Development Phases



Feature Request

Requirements

Architecture

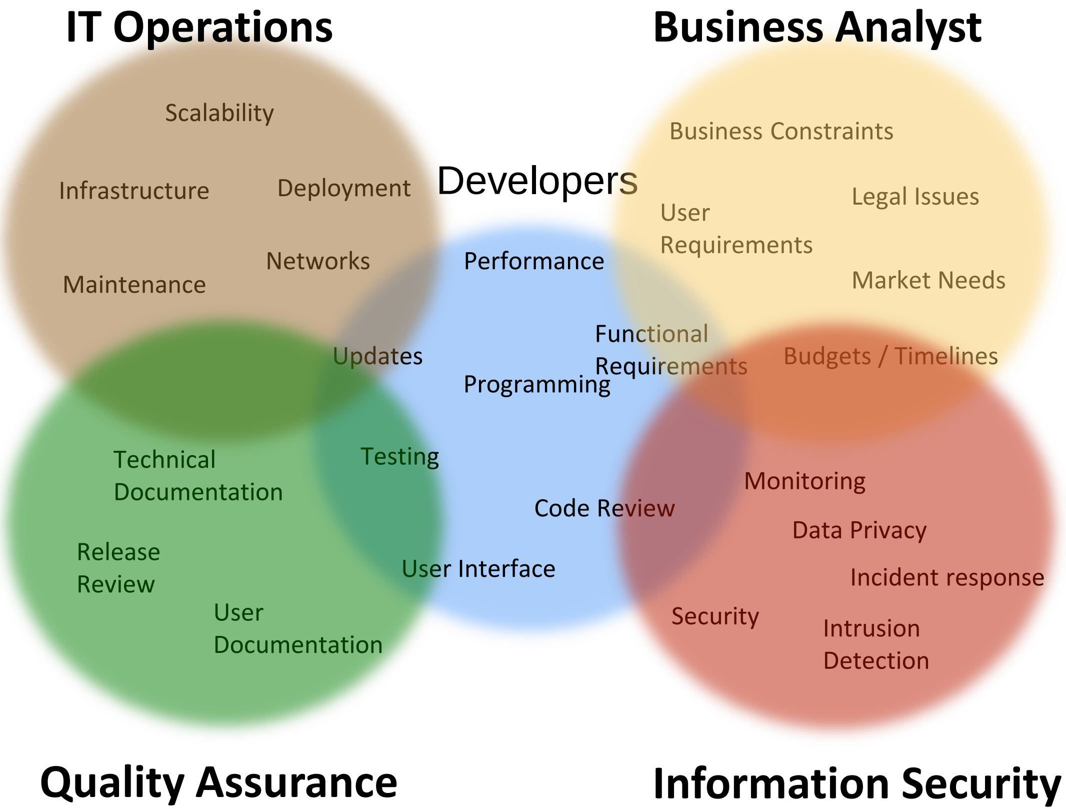
Design

Development

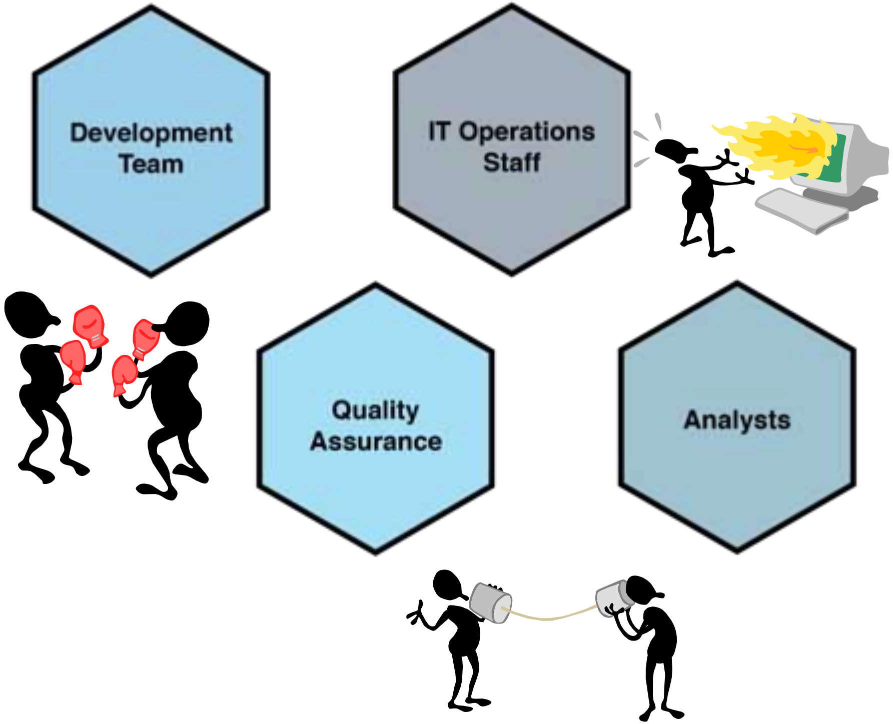
Test

Delivery

Collaboration: *Many stakeholders*

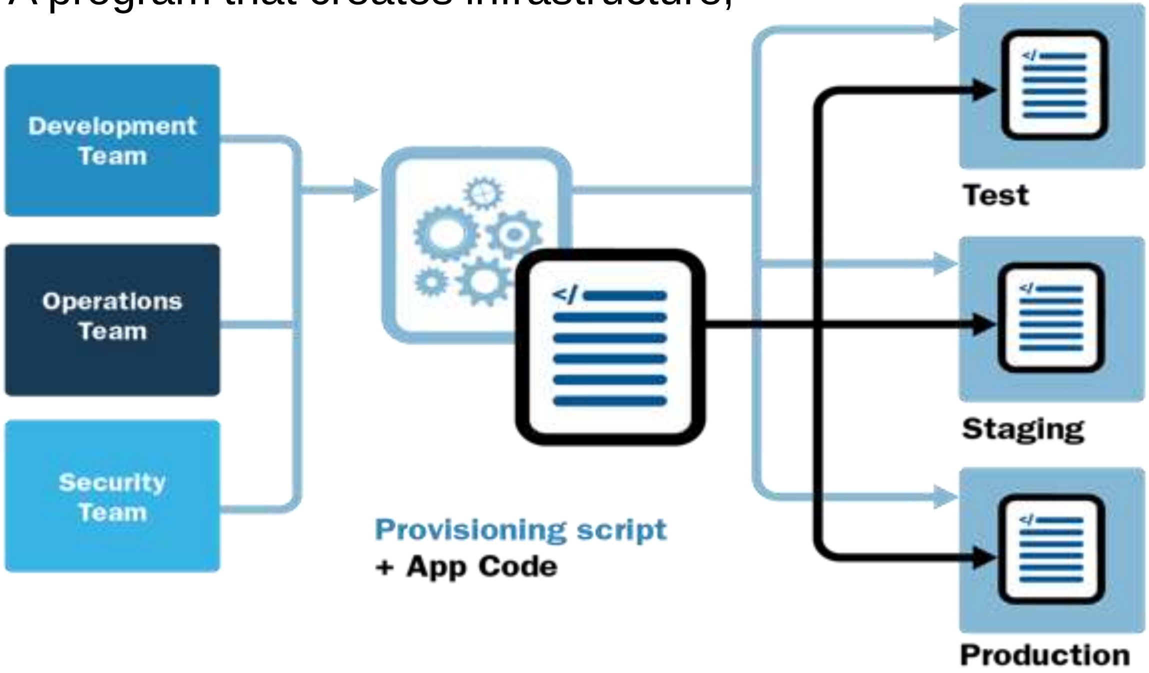


Collaboration: *Silos Inhibit Collaboration and poor communication*



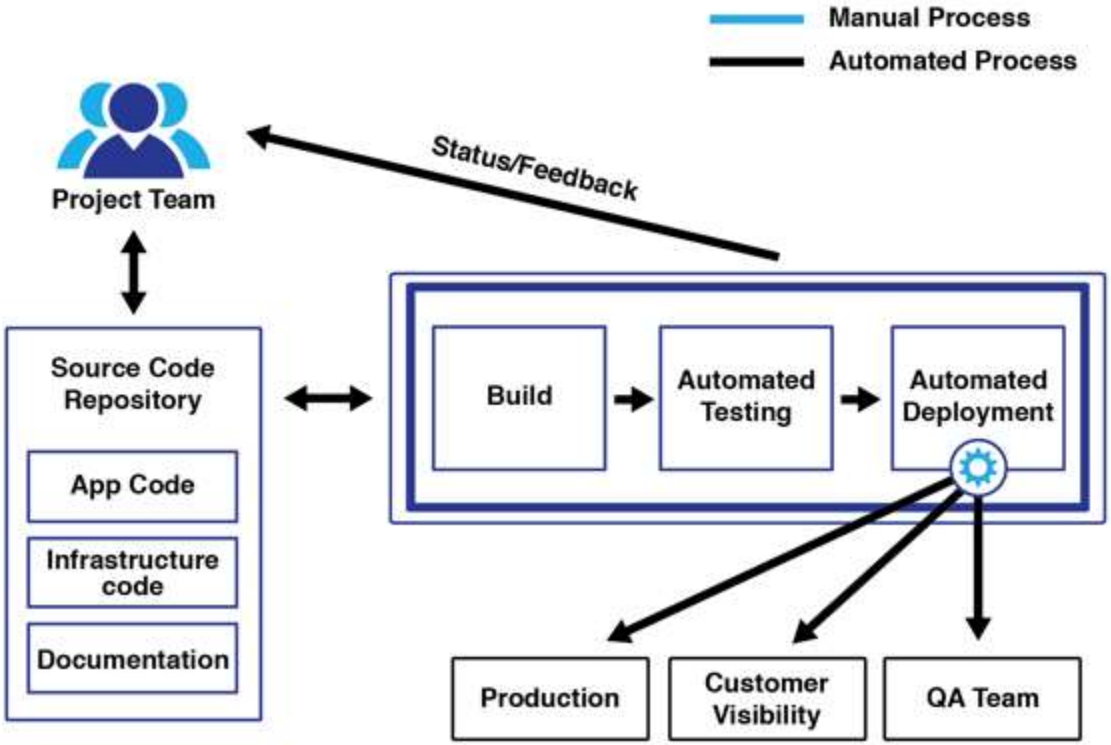
Infrastructure as Code (IaC)

A program that creates infrastructure,



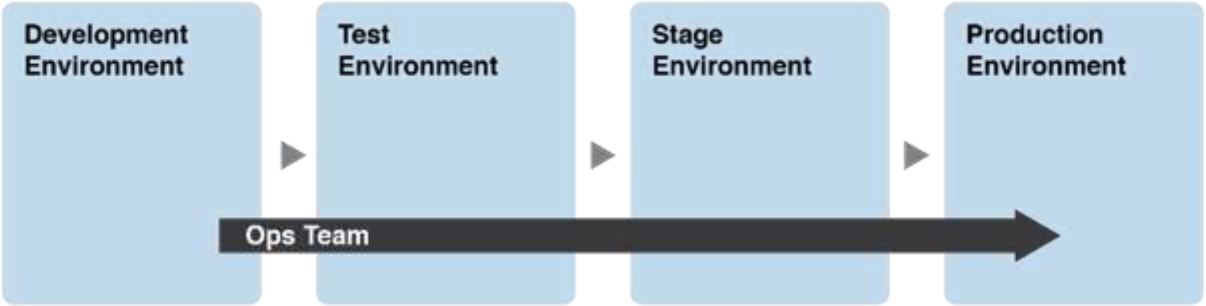
A concretely defined description of the environment is good material for conversation between team members.

Automation : *Continuous Integration (CI)*



Continuous integration is a process that continually merges a system's artifacts, including source code updates and configuration items from all stakeholders on a team, into a shared mainline to build and test the developed system.

Automation : *Continuous Delivery / Deployment (CD)*



Shift Left Operational Concerns Enforced by Continuous Delivery with parity across various environment

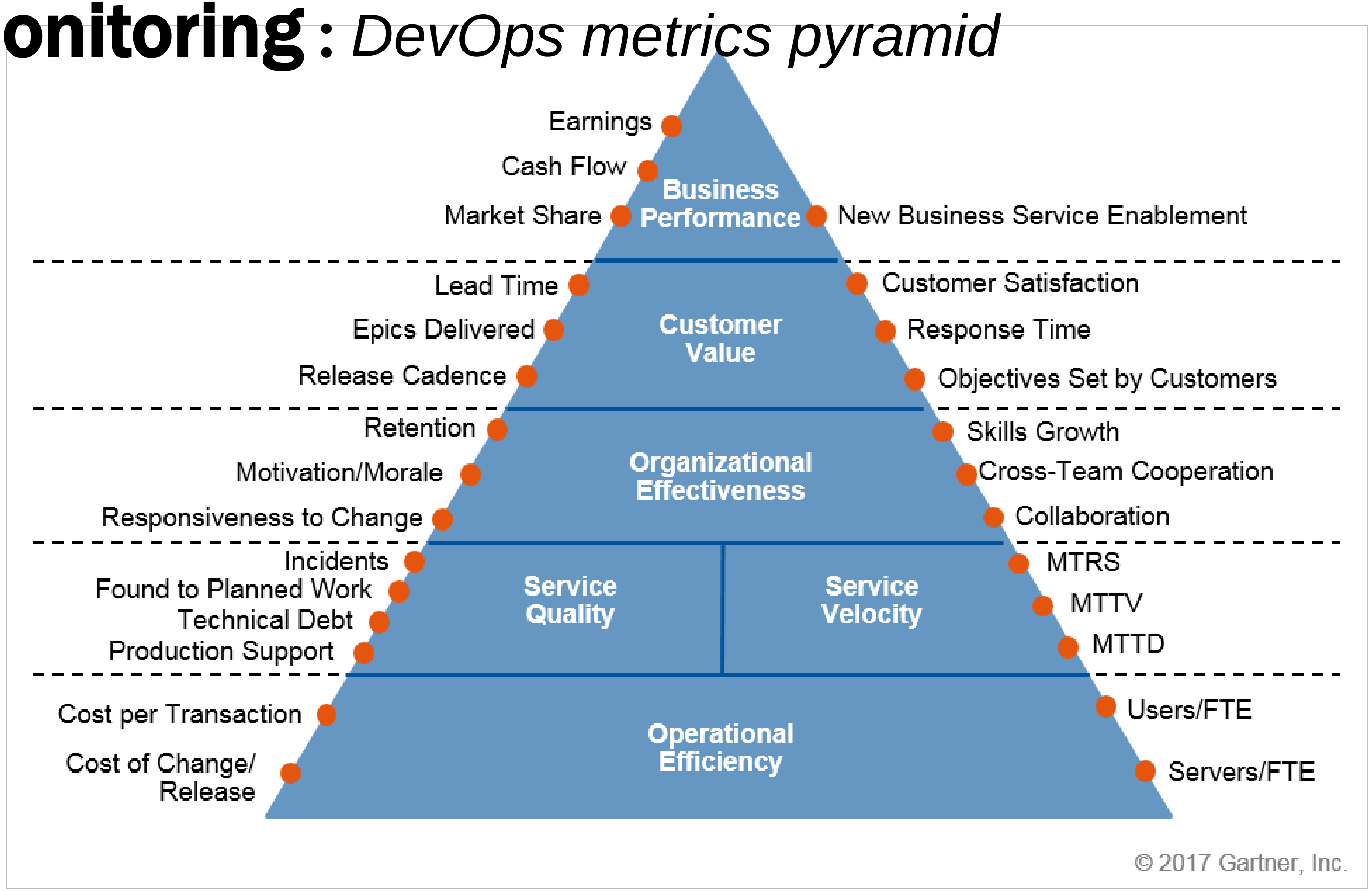
Continuous delivery is a software engineering practice that allows for frequent releases of new software to staging or various test environments through the use of automated testing.

Continuous deployment is the automated process of deploying changes to production by verifying intended features and validations to minimize risk.

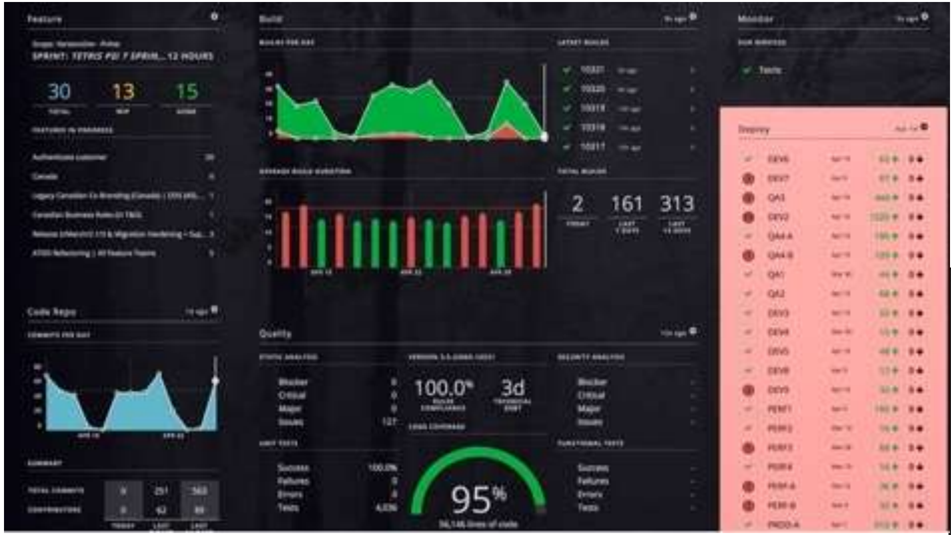
Monitoring : *DevOps metrics*

- Without metrics there is no way to know if you are improving in your performance of processes to answer :
  - Is the service delivering value to the users?
  - Is the service operating properly?
  - Are we achieving business goals?
  - Is the service secure?
  - Is the infrastructure adequate?
  - Is the service being attacked?
  - Can future needs be supported?
  - Are we able to plan new product? If so, how much?
  - Are we compliant?

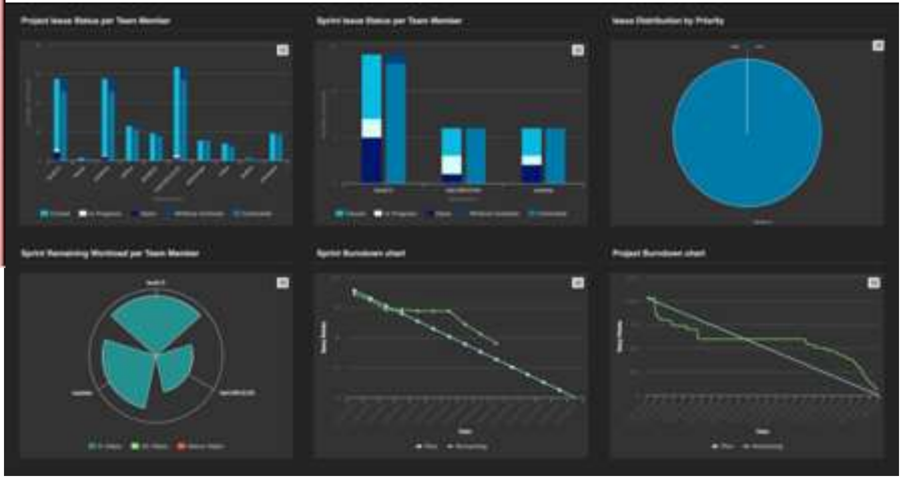
Monitoring: *DevOps metrics pyramid*



Monitoring : *Dashboard*

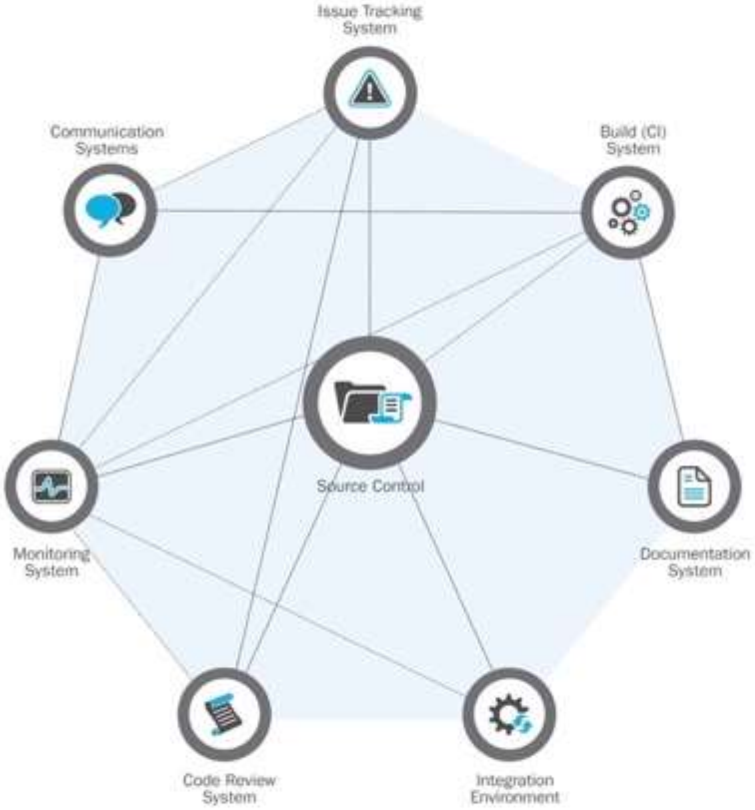


Dashboards can hold a large amount of information and are good in displaying outliers to expected behaviors.

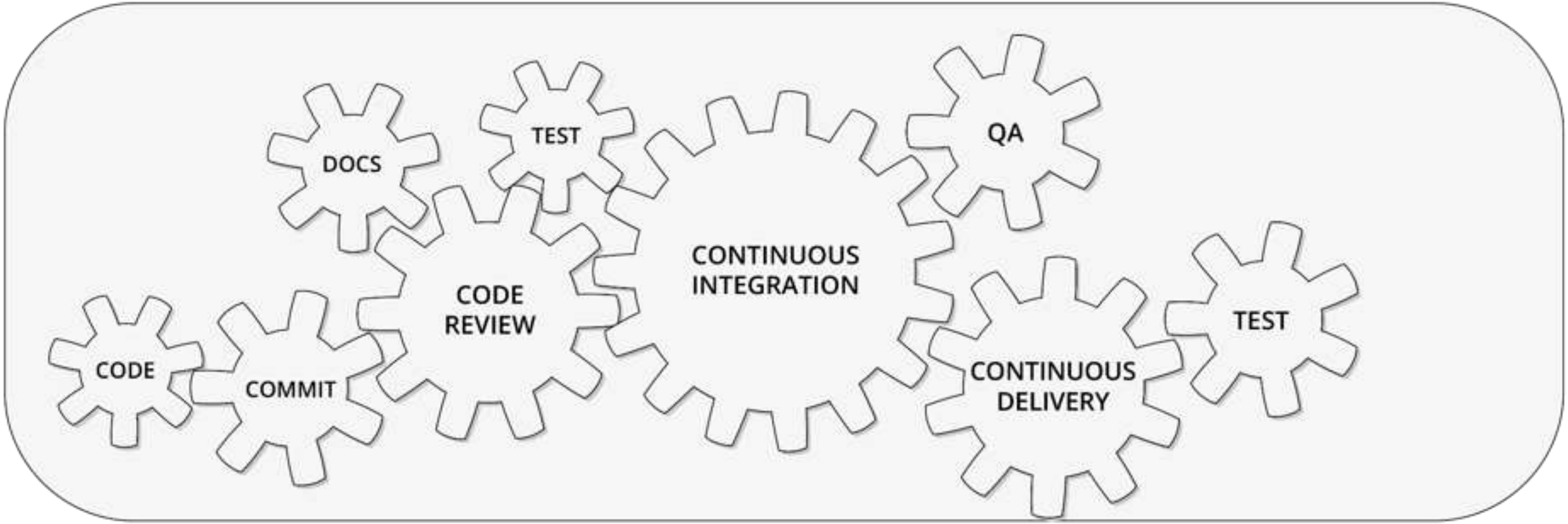


Acquisition, product development, and programs make many assumptions.

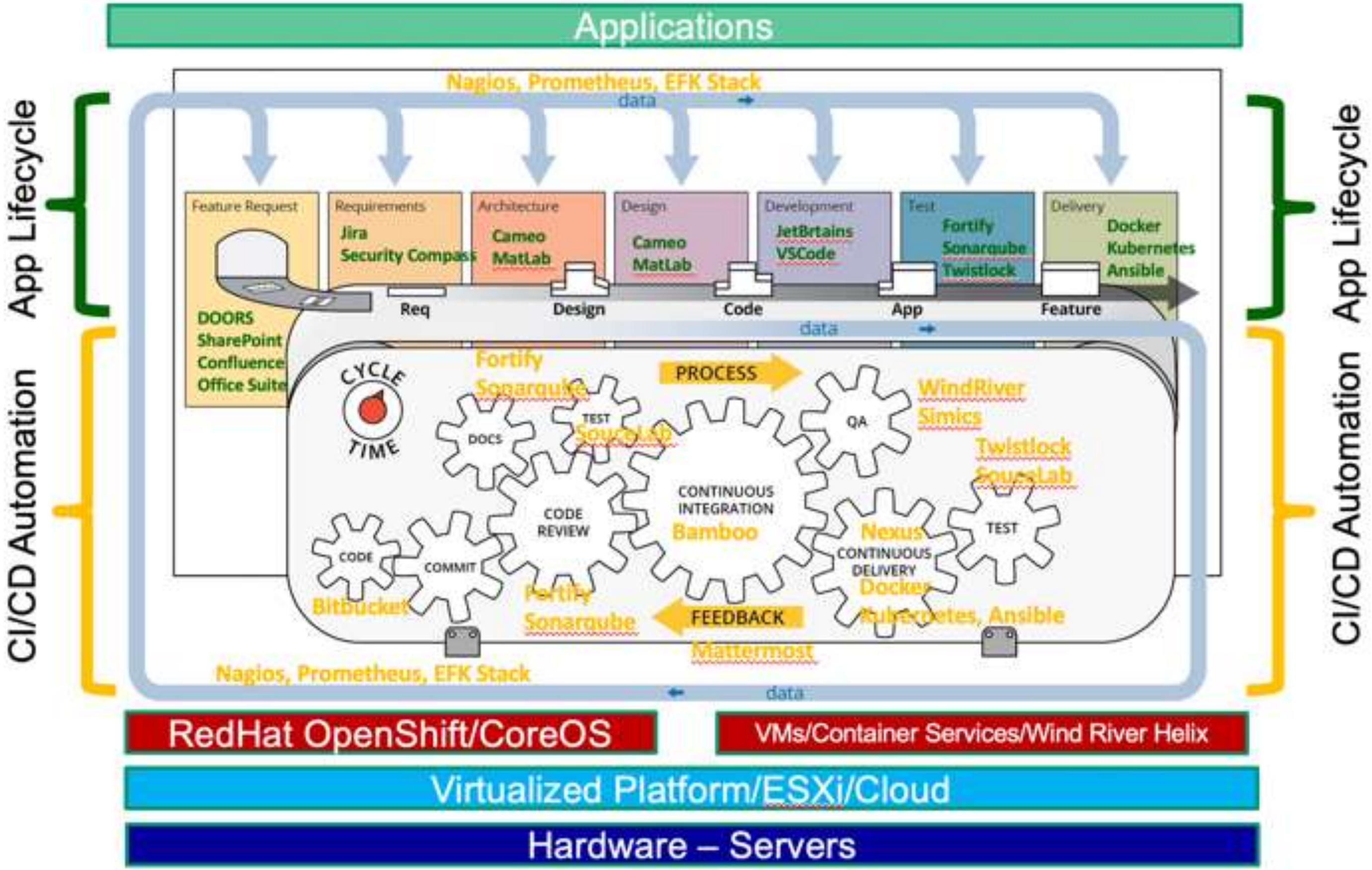
Integrated Development Pipeline - General



Automation with IaC, CI, CD



DevOps Stack: Exemplary DoD tool stack



DevSecOps?

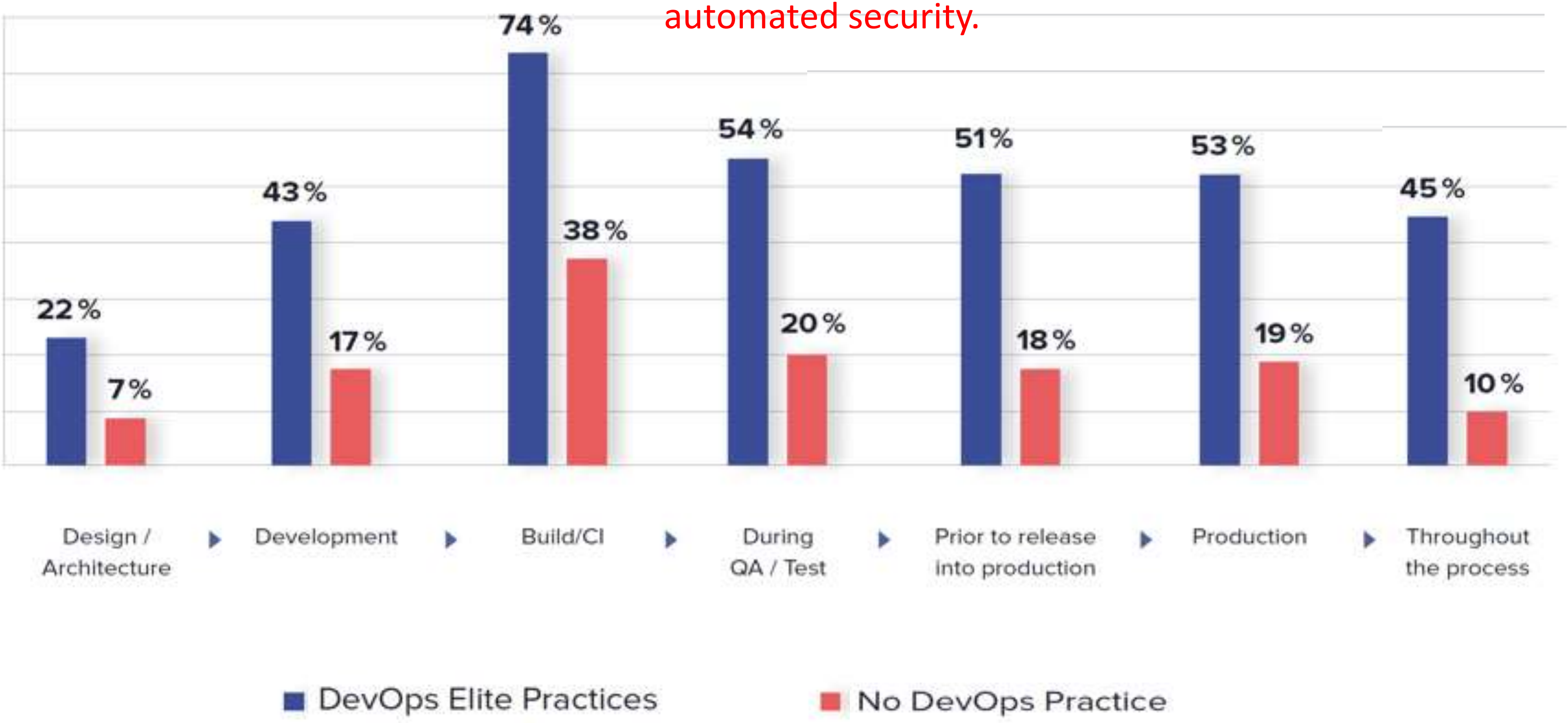
DevSecOps is a model on integrating the software development and operational process considering security activities: requirements, design, coding, testing , delivery , deployment and incident response.

Mature DevOps practices are constantly testing, deploying and validating that software meets every requirement and allows for fast recovery in the event of a problem. As a result we can easily say,

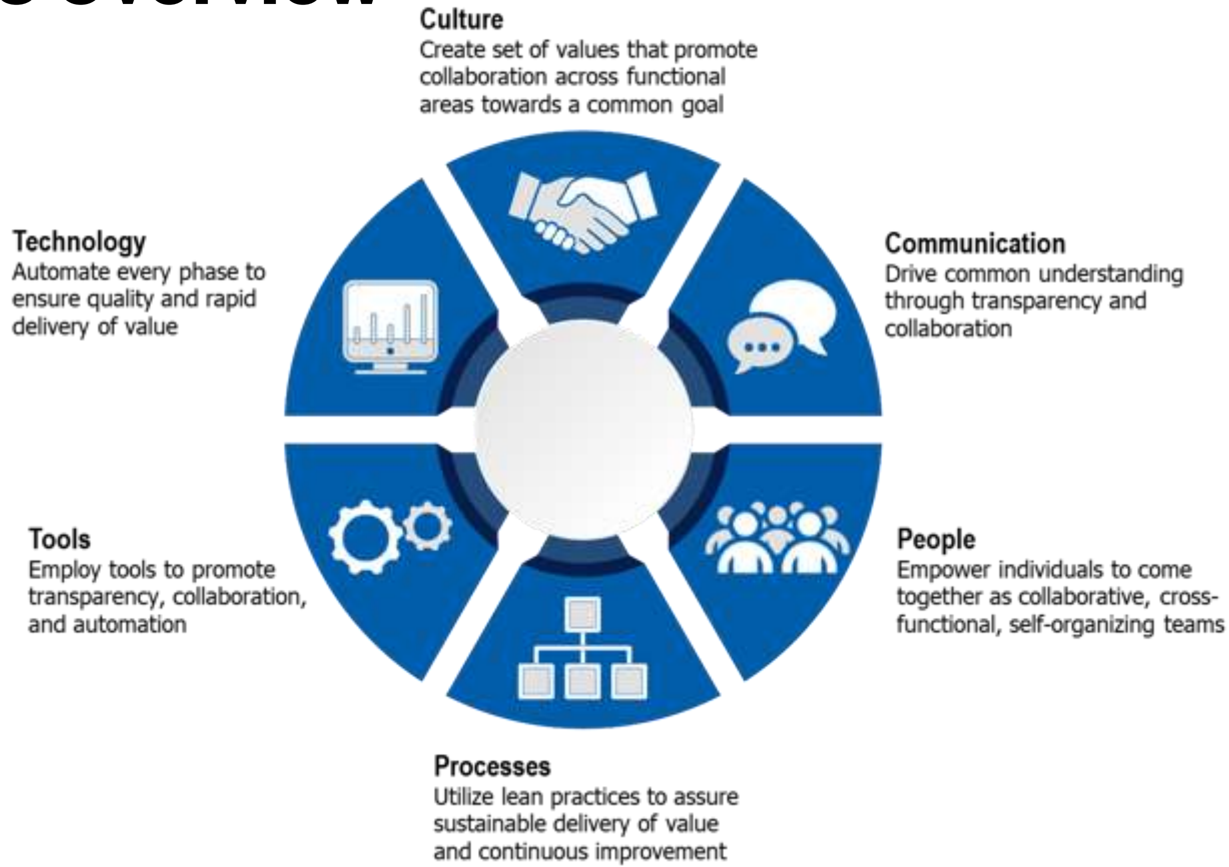
“DevSecOps is DevOps done right”

Automated Security Analysis

Security automation across SDLC: Mature DevOps practices are 350 % more likely to integrate automated security.



DevSecOps Overview



Culture
Create set of values that promote collaboration across functional areas towards a common goal

Communication
Drive common understanding through transparency and collaboration

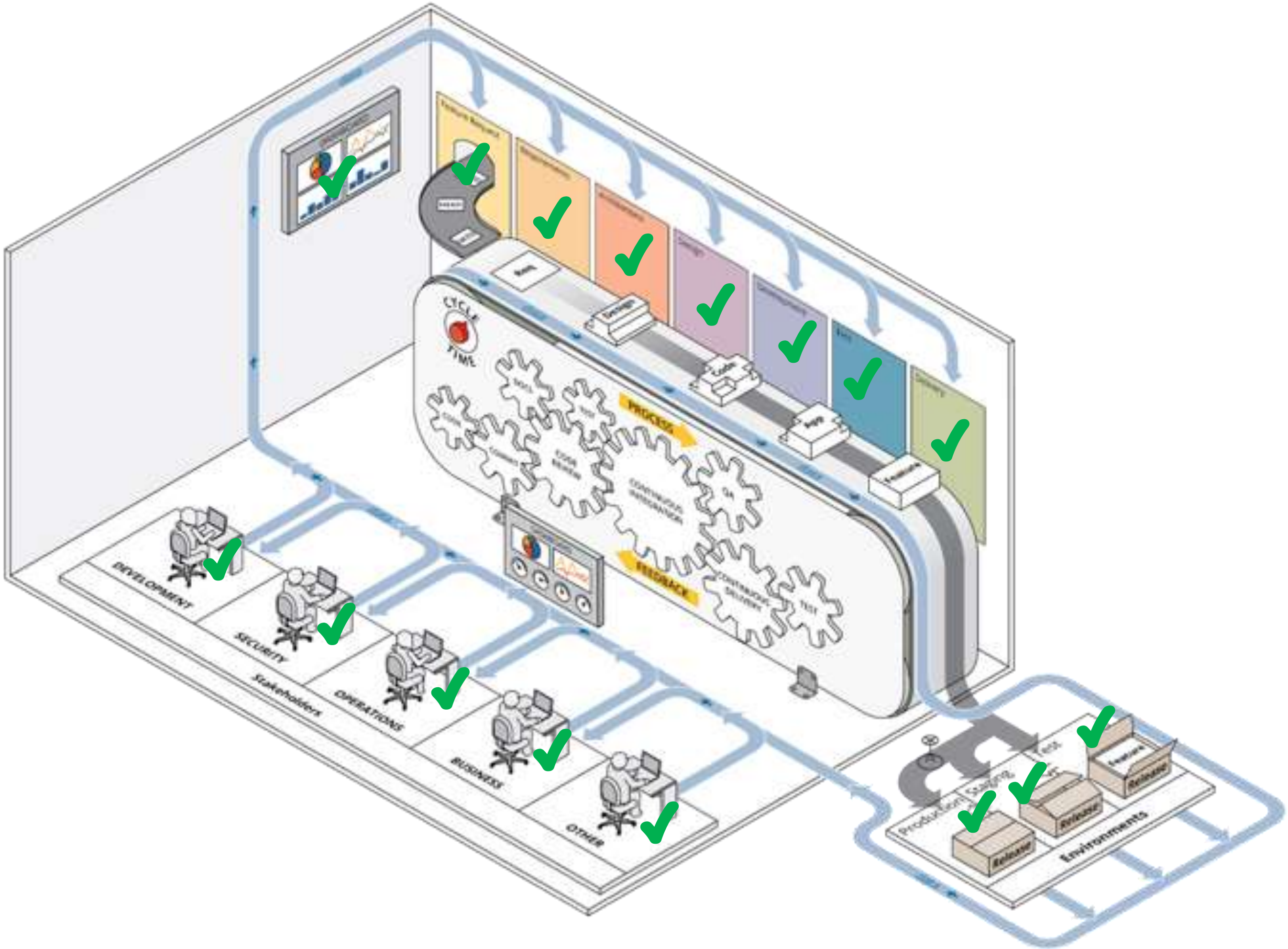
People
Empower individuals to come together as collaborative, cross-functional, self-organizing teams

Processes
Utilize lean practices to assure sustainable delivery of value and continuous improvement

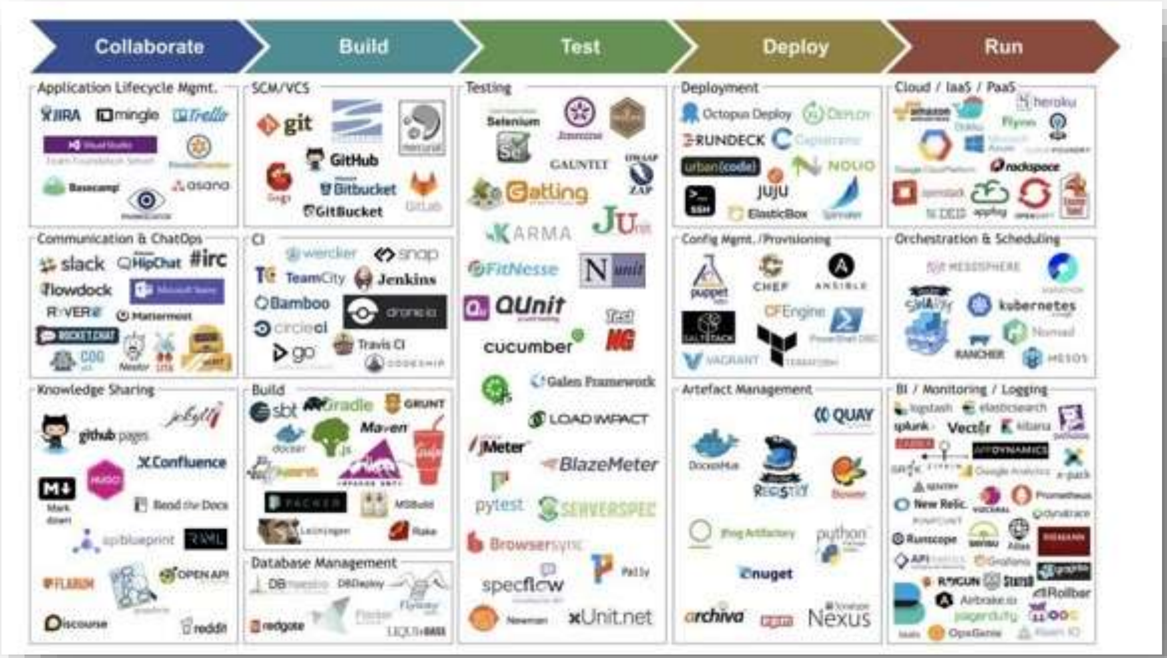
Tools
Employ tools to promote transparency, collaboration, and automation

Technology
Automate every phase to ensure quality and rapid delivery of value

Think Security from Inception to Deploy and improve every delivery



Agile practices+ *Architecture* + *Process*+ *Culture* = **Requirements** for an *Automated* DevSecOps Pipeline



The DevSecOps technology stack is *NOT* where we start

- Automating processes that don’t add to or lead to value is more waste

HW/SW Development & Deployment

Problem statement for large organizations

System Context	Cultural Context
System Structure	• Organizational Structure
Quantitative Information Flow	• Qualitative Information Flows
Heterogeneous Elements	• Heterogenous Subculture
Emergent behavior	• Mental Models
Interfaces	• Relationships
Nomenclature	• Language

(Some of the) Problems We hear about on Large, Complex Programs

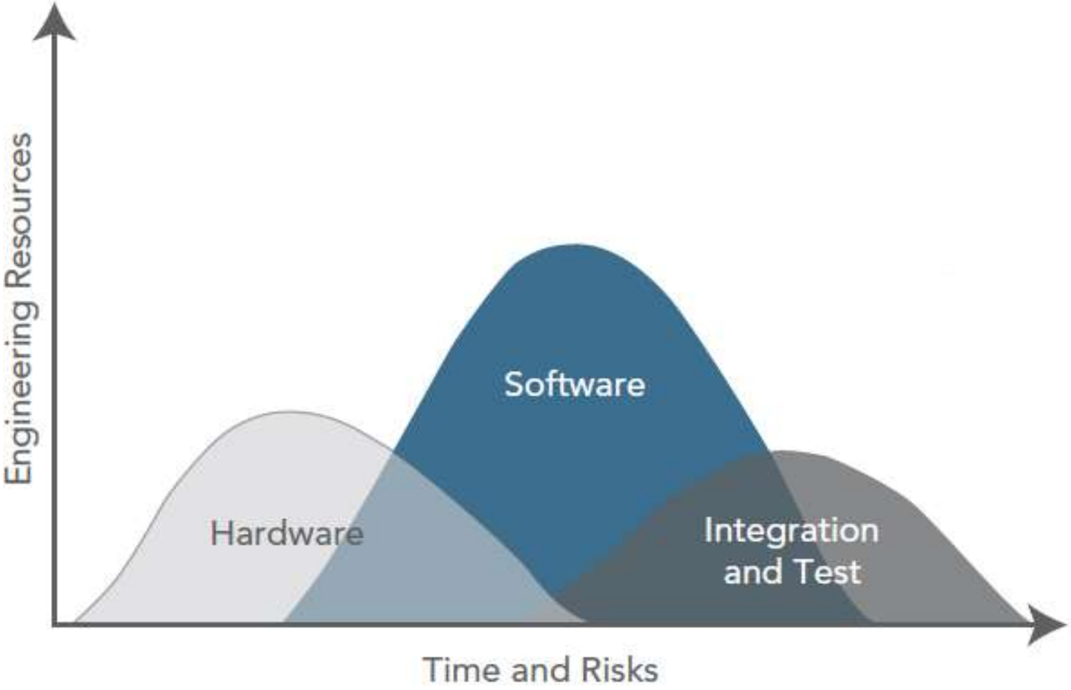
- Lack of alignment among stakeholders on practices used to engineer, develop, integrate, test, certify
- Lack of alignment among stakeholders on tools used to engineer, develop, integrate test, certify
- Lack of transparency – data, measures, decisions – among stakeholders
- “Nothing is done until everything is done”—large batch processes and mindset
- Delays due to governance cadence are routine

DevSecOps (DSO) Contribution to Solving Above Problems

Lack of alignment among stakeholders on practices used to engineer, develop, integrate, test, certify	DSO makes practices explicit for moving through value stream to delivery
Lack of alignment among stakeholders on tools used to engineer, develop, integrate test, certify	DSO uses a defined and agreed upon (by all stakeholders) set of tools to automate various aspects of value stream processes
Lack of transparency – data, measures, decisions – among stakeholders	DSO tools have the capability of enabling transparency, where participants choose
"Nothing is done until everything is done"—large batch processes and mindset	DSO automation enables small batches to flow through the value stream efficiently
Delays due to governance cadence are routine	DSO allows defined governance decisions to be automated based on explicit criteria

The Problem - HW /SW integration

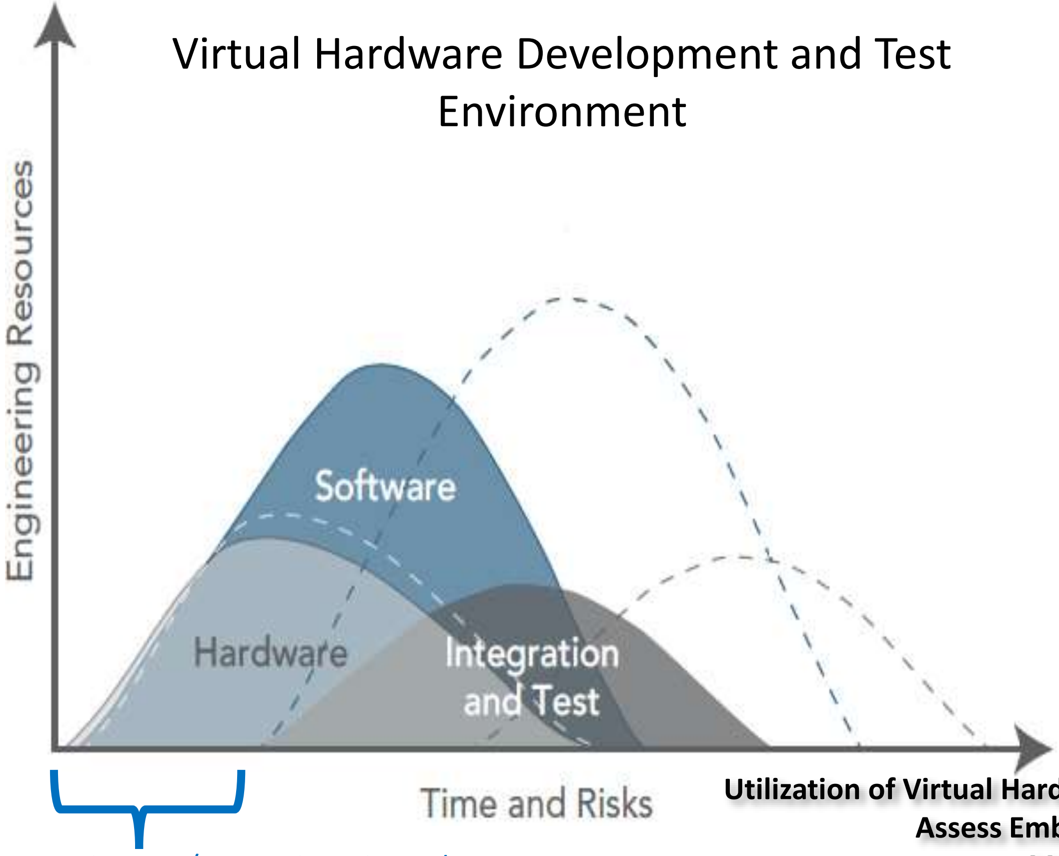
Traditional Software and Hardware Development and Test



- *Embedded HW Availability **Delays** Final Integration and Test*
- Software and hardware **issues identified late** the development life cycle costing schedule and cost impact.
- HW/SW **defects released** into fielded system
- HW design spec verification **Delay**
- Software architecture **risks will not be identified and mitigated** until much later in the software life-cycle
- **Requires expensive hardware** and association maintenance
- ***Minimum support for PDR, CDR*** *milestone with working virtual system*
- *M&S support **Delay***

HW & SW Design flaws identification delay resulting in cost and schedule overrun

The Solution - HW /SW integration



First HW/SW Engineering Release.

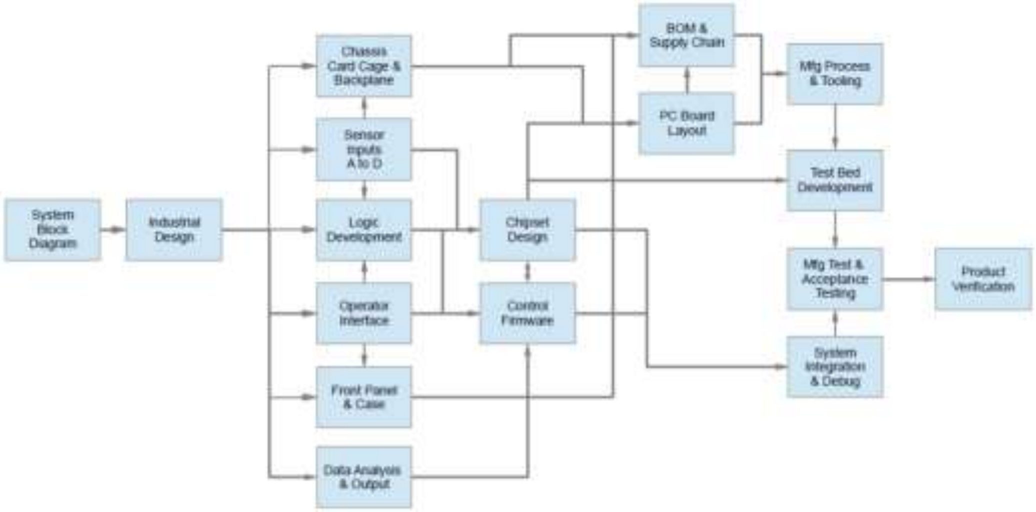
***Early Virtual HW Solves the Problem**:*

- ✓ *Embedded HW available **Early** for SW (including firmware) & HW **Integration and Test***
- ✓ *SW & HW **defect identification early and Minimizes Rework**: Cost avoidance using virtualization design verify design meets requirements and design specification.*
- ✓ HW Support **Design Specs verification**
- ✓ *SIV&V analyst can **perform dynamic analysis***
- ✓ ***Less expensive** then hardware*
- ✓ ***Support for PDR and CDR** milestone with working virtual system*
- ✓ ***Architecture** Risk Mitigation*
- ✓ ***Higher Fidelity** capability for M&S and Training environment **early***

Utilization of Virtual Hardware Environments will Accelerate Government's Ability to Assess Embedded SW and Provide Detailed SW Analysis Much Earlier in the Development Cycle

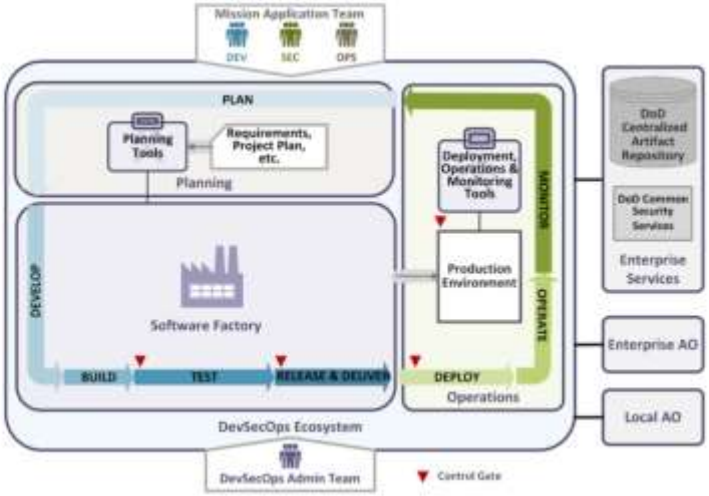
DevOps Helps, *But There Are Barriers*

Hardware Development

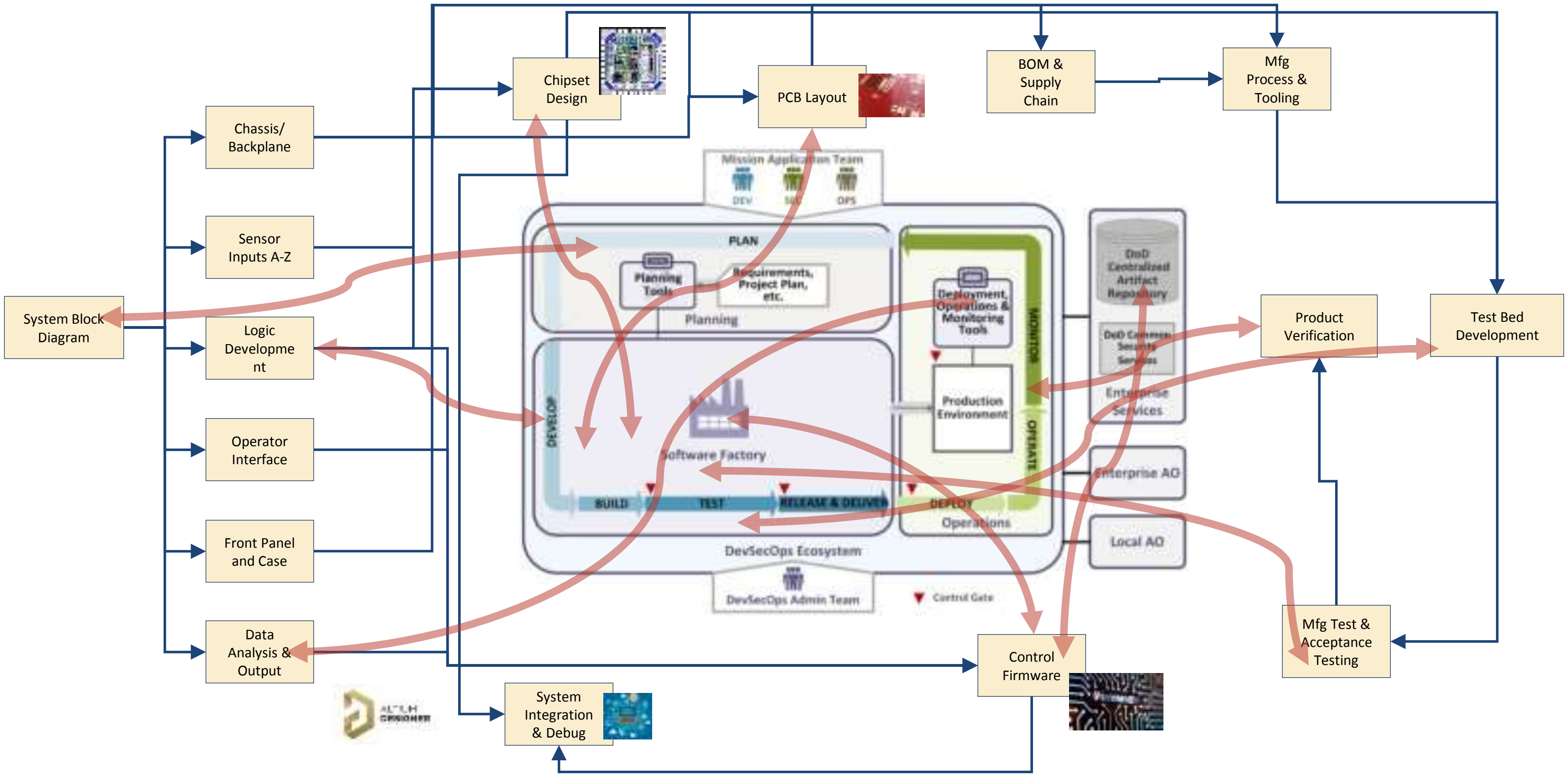


Hardware adoption of DevOps?

Software DevOps



Breaking Barriers



Chipset Design

PCB Layout

BOM & Supply Chain

Mfg Process & Tooling

Chassis/ Backplane

Sensor Inputs A-Z

System Block Diagram

Logic Development

Operator Interface

Front Panel and Case

Data Analysis & Output

System Integration & Debug

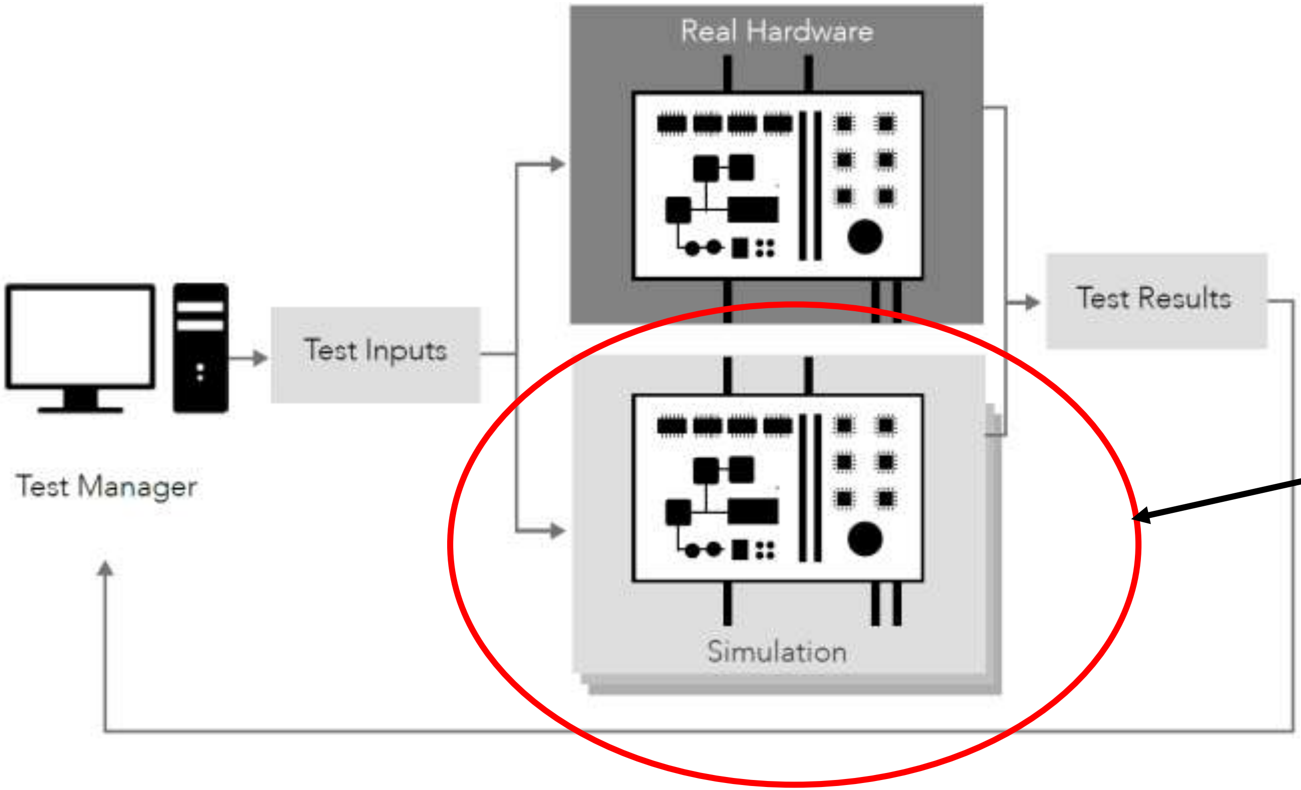
Control Firmware

Product Verification

Test Bed Development

Mfg Test & Acceptance Testing

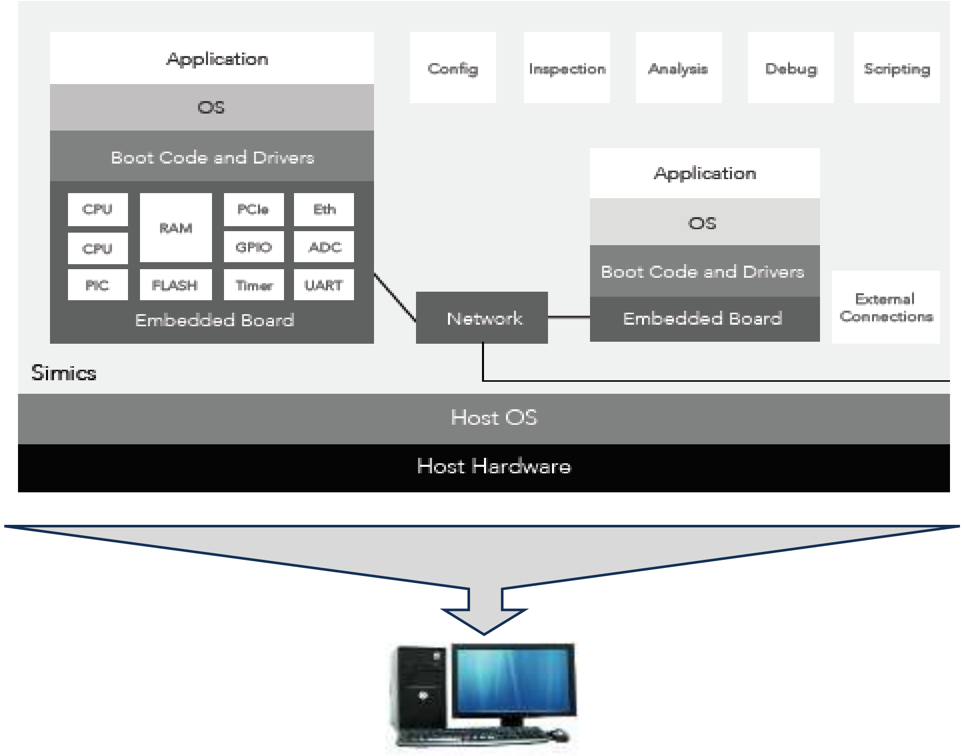
What is a Virtual HW Environment



Utilizes Virtual Hardware When Real Hardware is Unavailable

How Does it Work?

Virtual Hardware Architecture



Utilizes a Scalable Hardware Architecture to act as a simulated Hardware Platform for the Real Hardware.

- Hosts the **actual embedded software**
- Expandable, **powerful platform** contains multiple processors, I/O cards, memory modules and network interfaces to perform complex real-time computations
- **Utilizes real-time operating system** and time synchronization to maintain accurate system timing

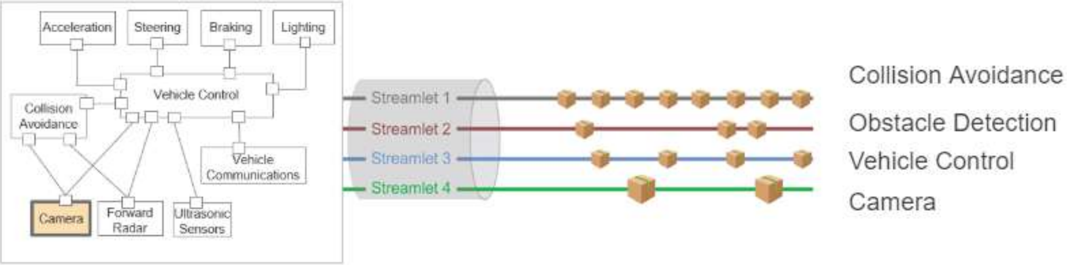
Industrial DevOps Principles * – HW/SW delivery

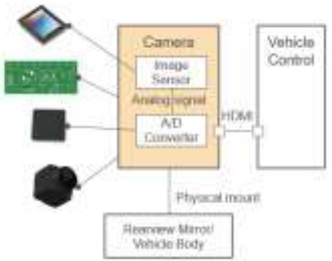
1. Visualize and organize around the value stream
2. Multiple Horizons of Planning
3. Base decisions on objective evidence of system state and performance
4. Architect for Scale, Modularity, and Serviceability
5. Iterate / Reduce batch size / Get fast feedback
6. Cadence and Synchronization
7. Continuish Integration
8. Test Driven Development

* IT Revolution Industrial DevOps & Applied Industrial DevOps Paper

Challenges – Example : Autonomous Vehicle

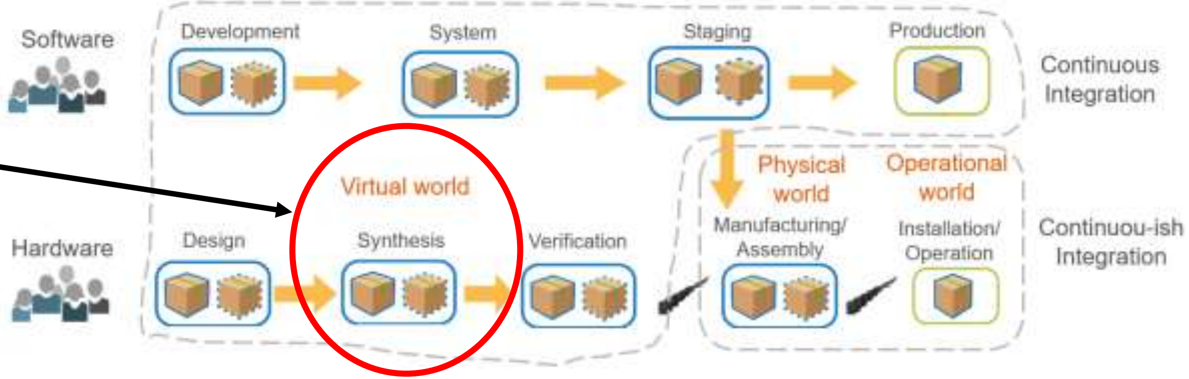
Modularity enables continuous flow in software and hardware





DevSecops delivery pipelines for software and hardware

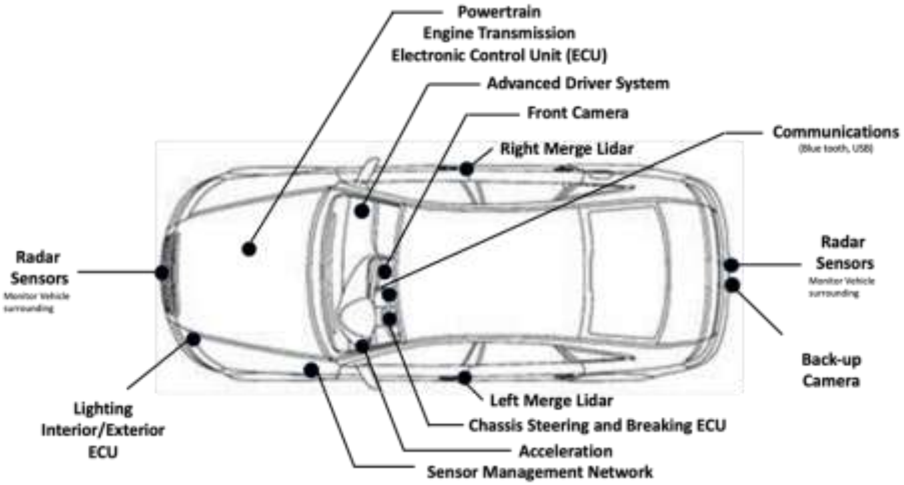
Virtual Hardware



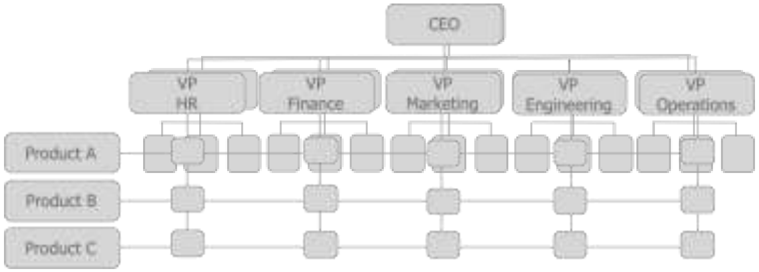
Example Solution: Autonomous Vehicle

Autonomous vehicles have similar complexity and human safety details as many of the products that DoD currently do.

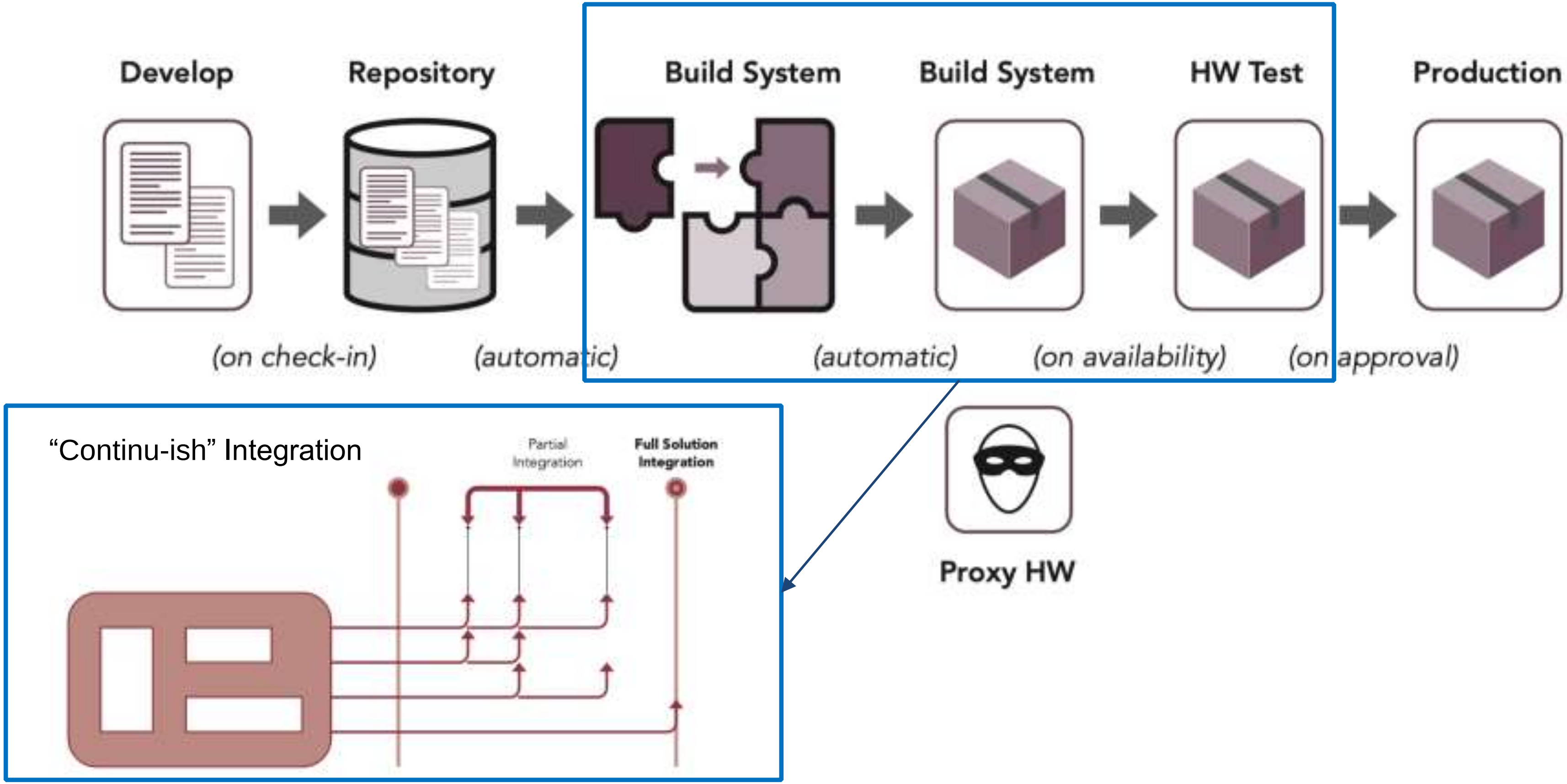
System Context



Cultural Context



Software/System development pipelines workflow



Develop

Repository

Build System

Build System

HW Test

Production

(on check-in)

(automatic)

(automatic)

(on availability)

(on approval)

Proxy HW

“Continu-ish” Integration

Partial Integration

Full Solution Integration

Takeaways

DevSecOps: *People*

DevOps requires heavy collaboration between all stakeholders

- Continuous secure design / architecture decisions
- Agreed-on environment / network configuration
- Continuous secure deployment planning
- Continuous secure code review

DevOps requires constantly available, open communication channels:

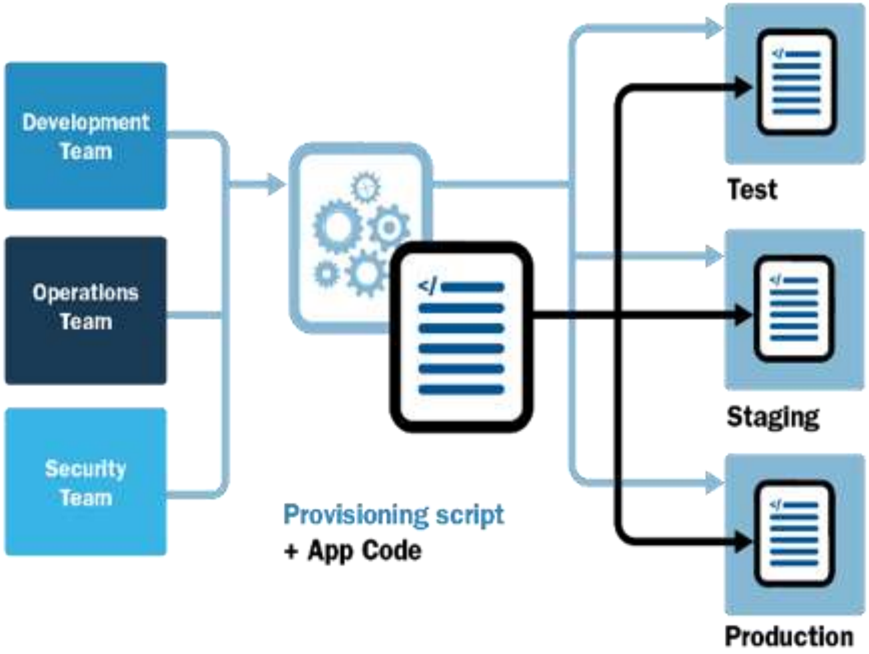
- Dev, Ops and Security together in all project decision meetings, virtually or physical but sharing a common collaboration environment
- Chat/email/Wiki services available to all team members

DevSecOps: *Process*

Establish a *process* to enable *people* to succeed using the *platform* to develop secure applications such that:

- communication is constant and visible to all
- tasks are testable and repeatable
- human experts are free to do challenging, creative work
- tasks can be performed with minimal effort or cost
- teams have confidence in task success after past repetitions
- deployment is faster, and quality releases are more frequent

DevSecOps: *Platform*

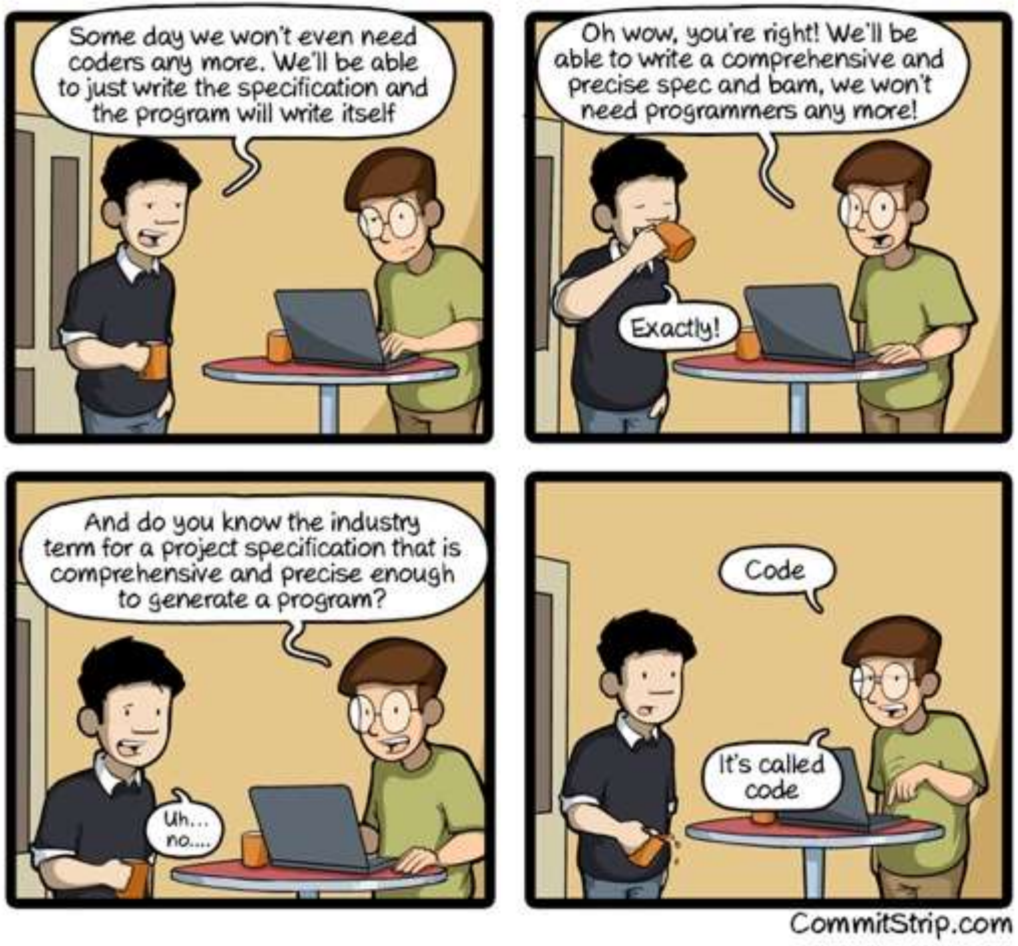


Where **people** use **process** to build software:

- Automated secure environment creation and provisioning
- Automated secure infrastructure testing
- Parity between development, QA, staging, and production environments
- Sharing and versioning of environmental configurations
- Collaborative environment between all stakeholders

Summary

Leveraging the power of HW/SW DevSecOps pipeline for large complex systems is an industry step change and the companies that solution this problem first will increase transparency, reduce cycle time, early HW/SW integration, test automation, increase value for money, and innovate faster.


Some day we won't even need coders any more. We'll be able to just write the specification and the program will write itself
Oh wow, you're right! We'll be able to write a comprehensive and precise spec and bam, we won't need programmers any more!
Exactly!
And do you know the industry term for a project specification that is comprehensive and precise enough to generate a program?
Uh... no....
Code
It's called code
CommitStrip.com

For more information…

DevOps: https://www.sei.cmu.edu/go/devops

DevOps Blog: https://insights.sei.cmu.edu/devops

Webinar : https://www.sei.cmu.edu/publications/webinars/index.cfm

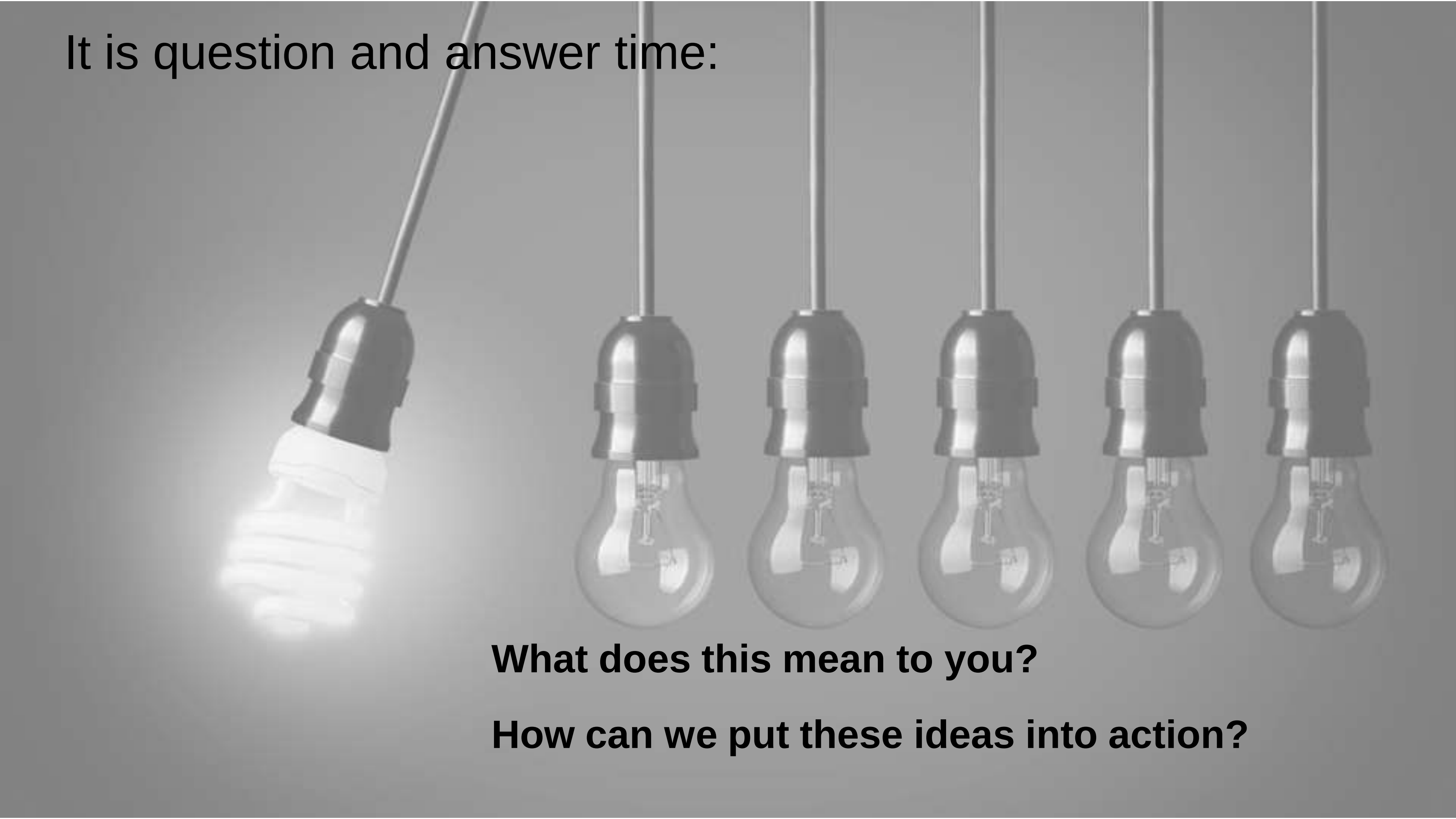
Podcast : https://www.sei.cmu.edu/publications/podcasts/index.cfm

Thank You

Hasan Yasar

Technical Director, Adjunct Faculty Member
Continuous Deployment of Capability
hyasar@sei.cmu.edu
@securelifecycle

It is question and answer time:



What does this mean to you?

How can we put these ideas into action?