Build Secure Application with **DevSecOps!**

Dr. Tom Longstaff

Hasan Yasar

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.

DM19-1060

Build Secure Application with DevSecOps!

DevOps

DevOps and How it started

DevOps is a set of principles and practices emphasizing collaboration and communication between software development teams and IT operations staff along with acquirers, suppliers and other stakeholders in the life cycle of a software system [1]

- Patrick Debois “Agile infrastructure and operations: how infra-gile are you?”, Agile 2008 Conference
- John Allspaw “ 10+Deploys per Day: Dev and Ops Cooperation”, Velocity 2009
- DevOpsDays, October 30th 2009, #DevOps term born

[1] IEEE P2675 DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment

Who are Dev?

- Follow Agile methodologies
  - Using Scrum, Kanban and modern development approaches
  - Self directing, self managed, self organized
- Using any new technology
  - Each Dev has own development strategy
  - OpenSource,
- Allowed to have
  - Close relationships with the business
  - Software driven economy

Want to deliver software faster with new requirements…

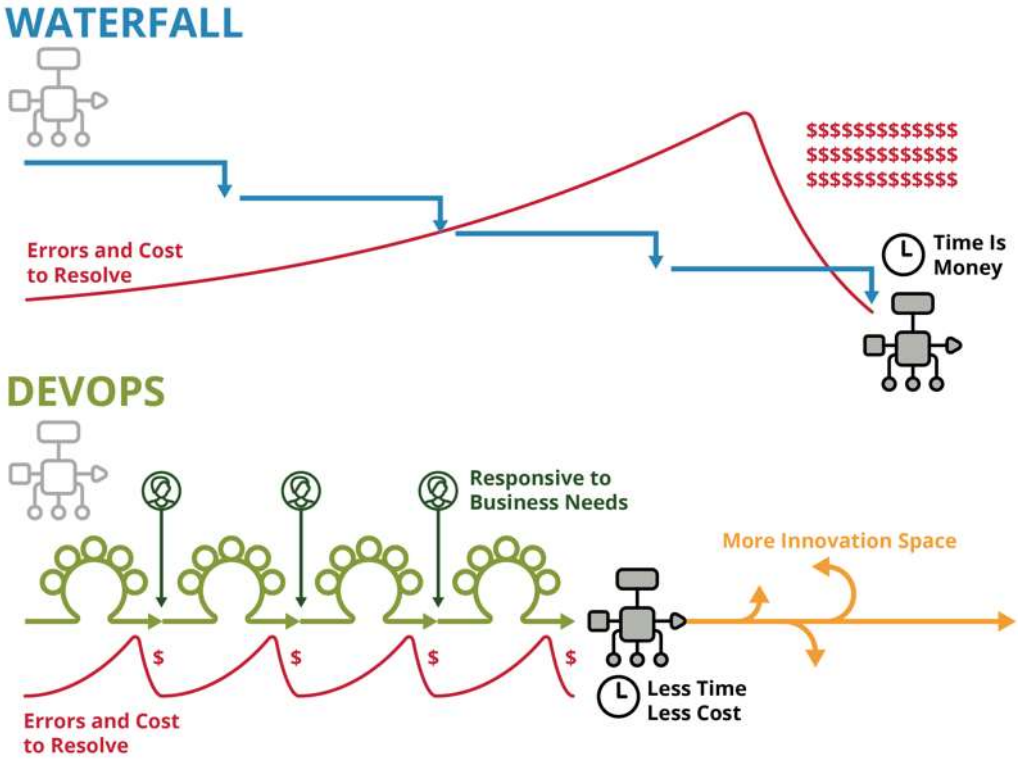
Who are Ops?

- Operations
  - Runs the application
  - Manages the infrastructure
  - Support the applications

- Operations provides
  - Service Strategy
  - Service Design
  - Service Transition
  - Service Operations
  - Secure systems

Want to maintain stability, reliability and security…

Key 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**

DevOps has four Fundamental Principles

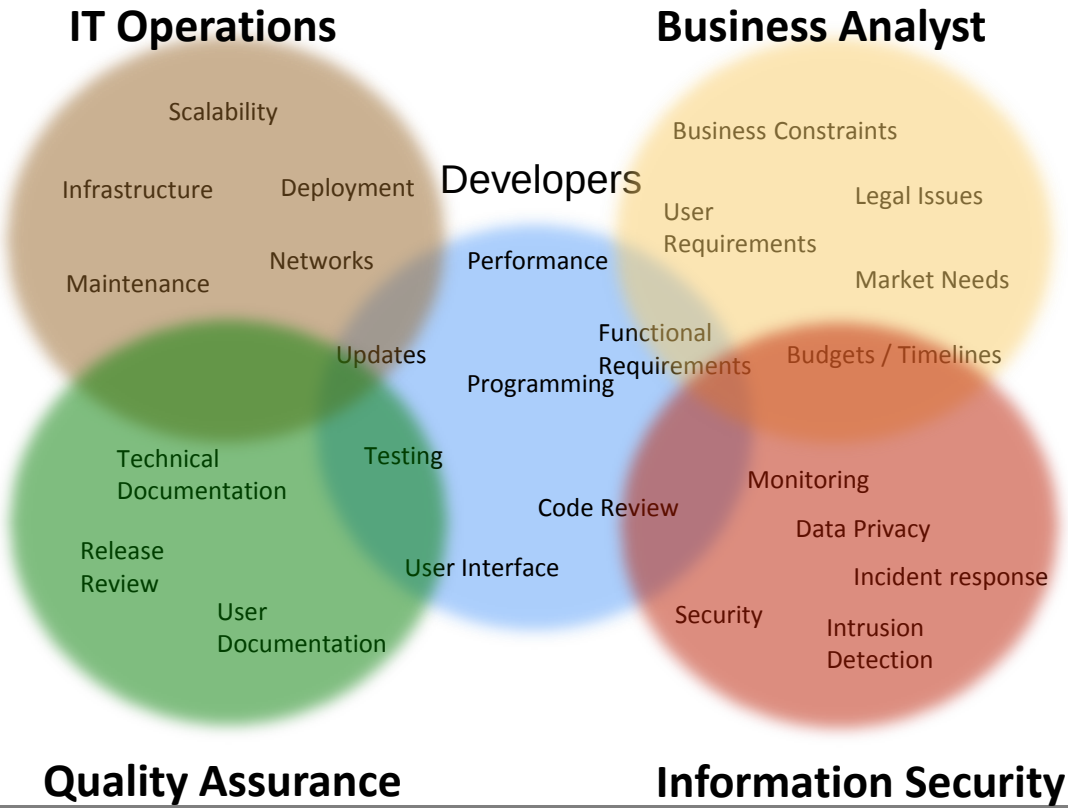
- **Collaboration:** between project team roles

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

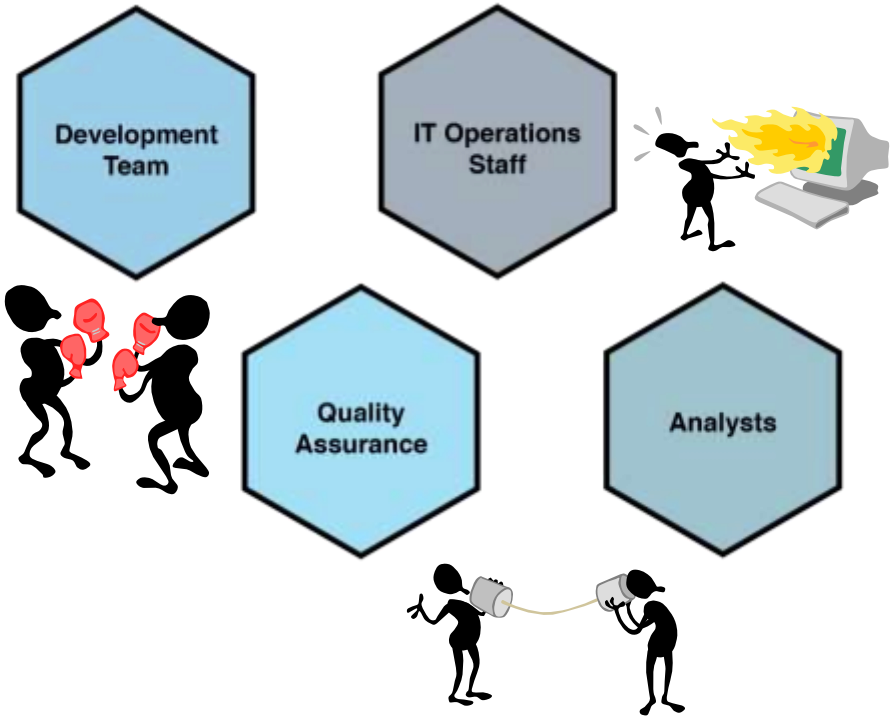
- **Automation:** deployment, testing, provisioning, any manual or human-error-prone process

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

Collaboration: *Many stakeholders*

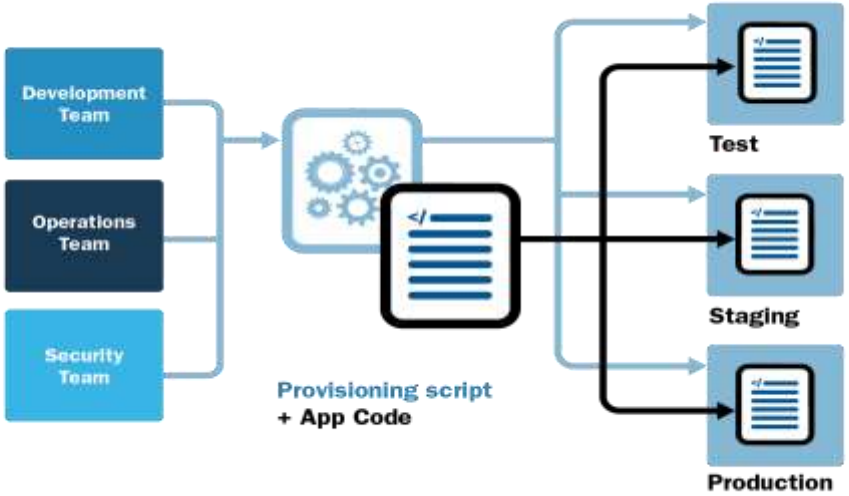


Collaboration: *Silos Inhibit Collaboration and poor communication*



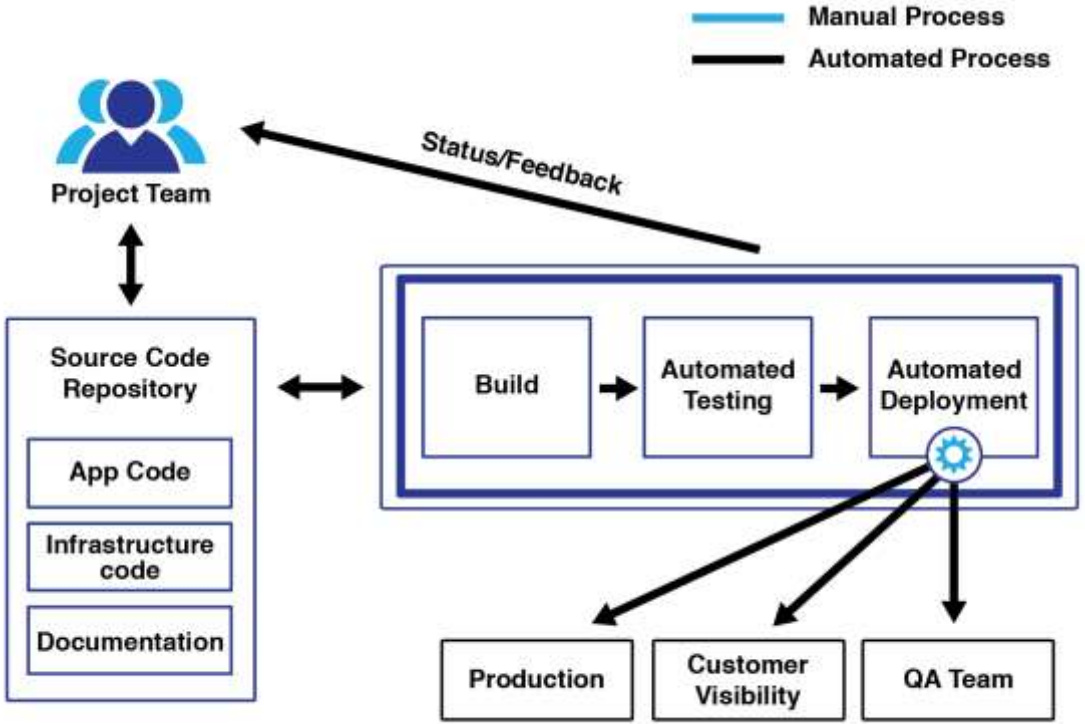
Infrastructure as Code (IaC)

A program that creates infrastructure,

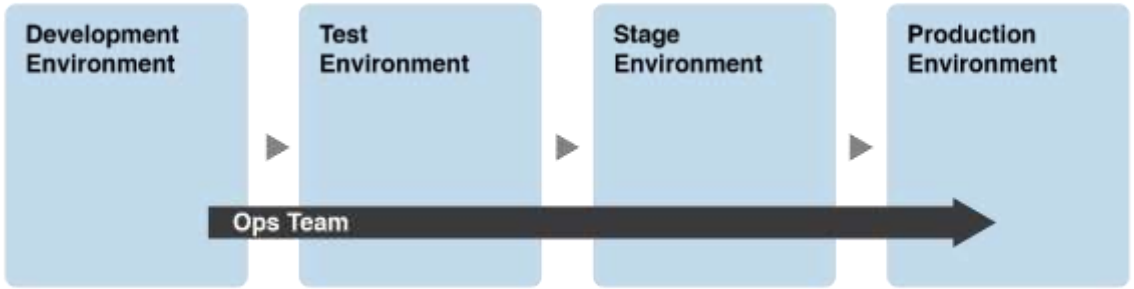


A concretely defined description of the environment is good material for conversation between team members.

Automation : *Continuous Integration (CI)*

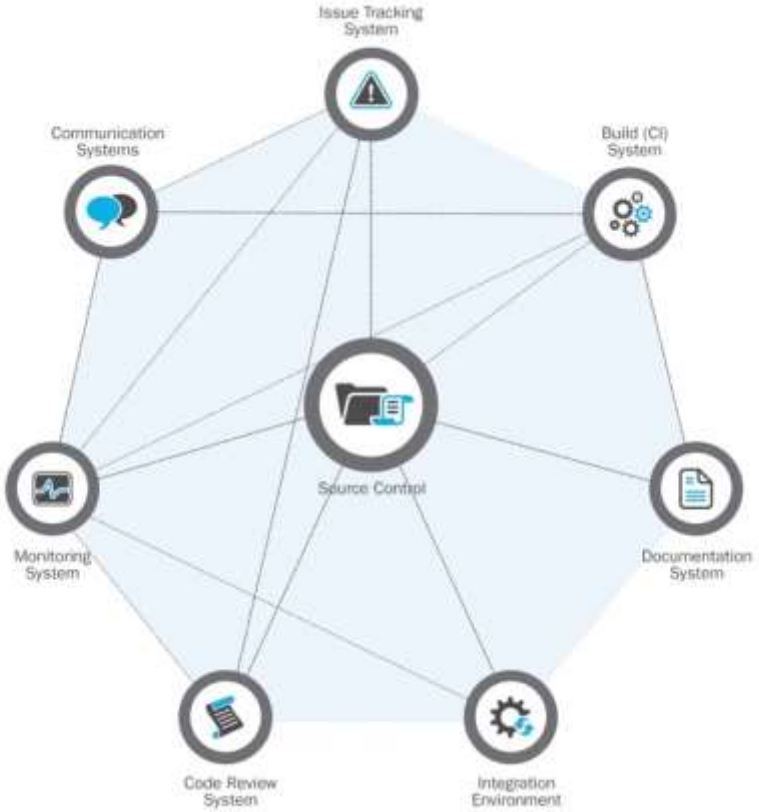


Automation : *Continuous Delivery / Deployment (CD)*

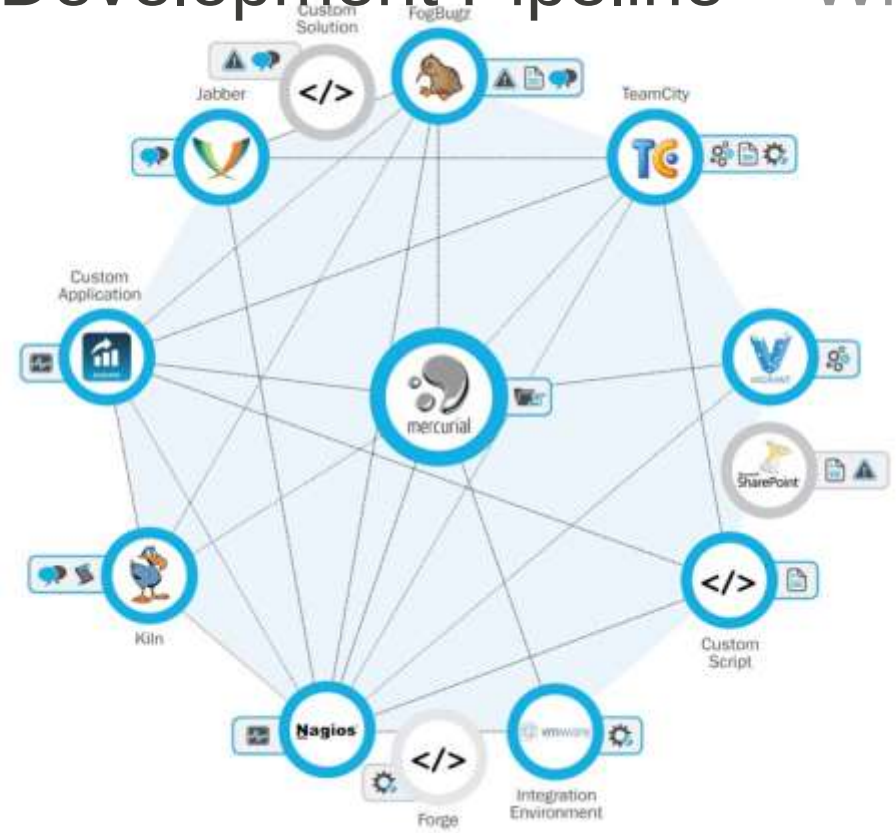


Shift Left Operational Concerns Enforced by Continuous Delivery with parity across various environment

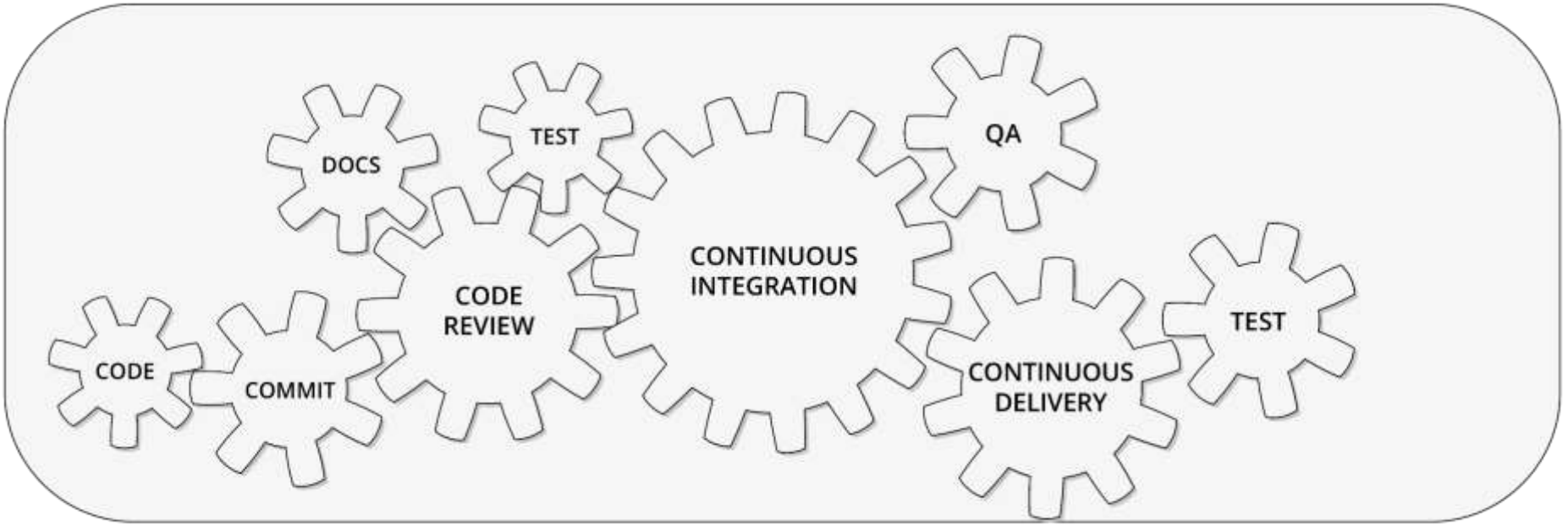
Integrated Development Pipeline - General



Integrated Development Pipeline – With Tooling



Automation with IaC, CI, CD

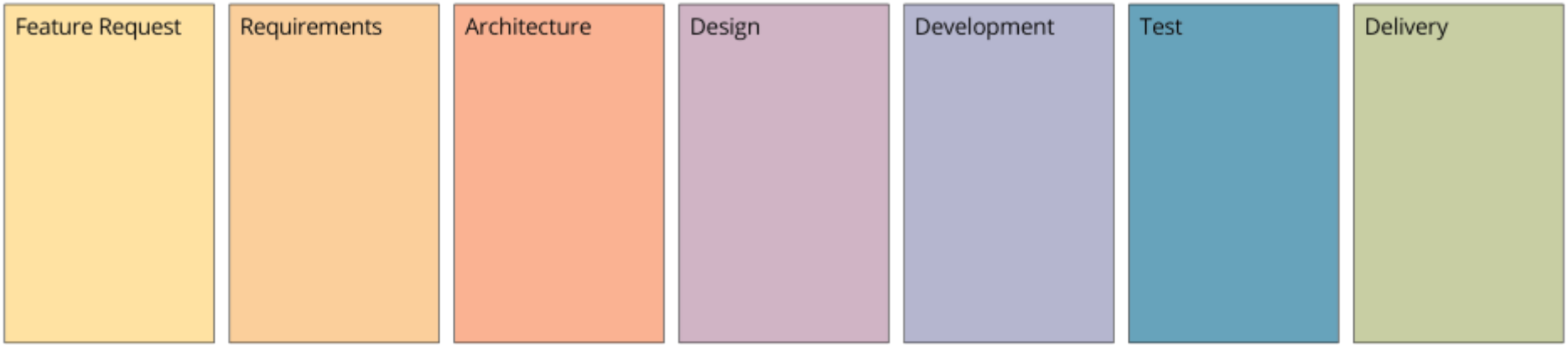


Build Secure Application with DevSecOps!

DevSecOps

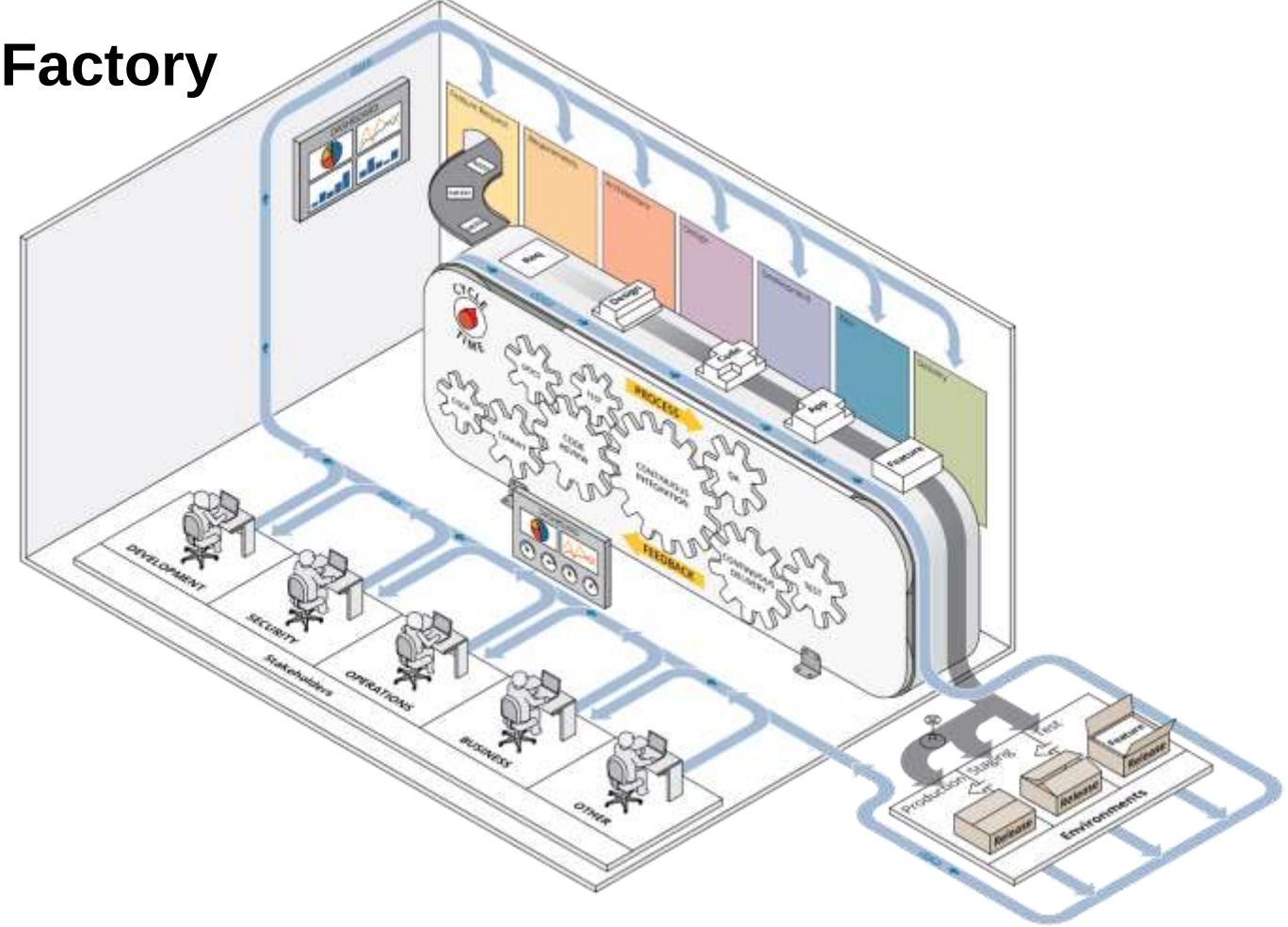
DevSecOps is a model on integrating the software development and operational process considering security activities: requirements, design, coding, testing, delivery, deployment and operations.

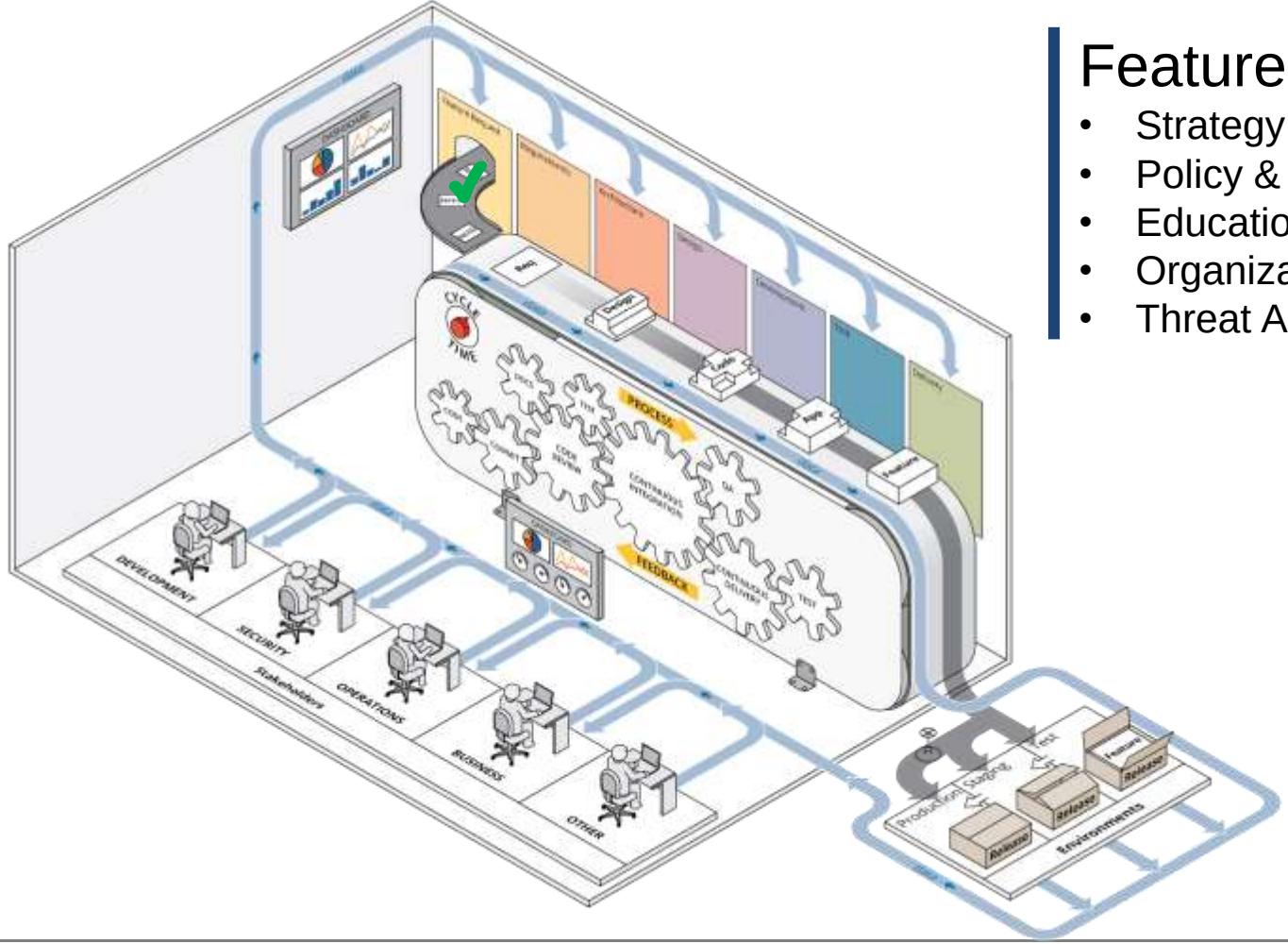
Phases



The DevSecOps Factory

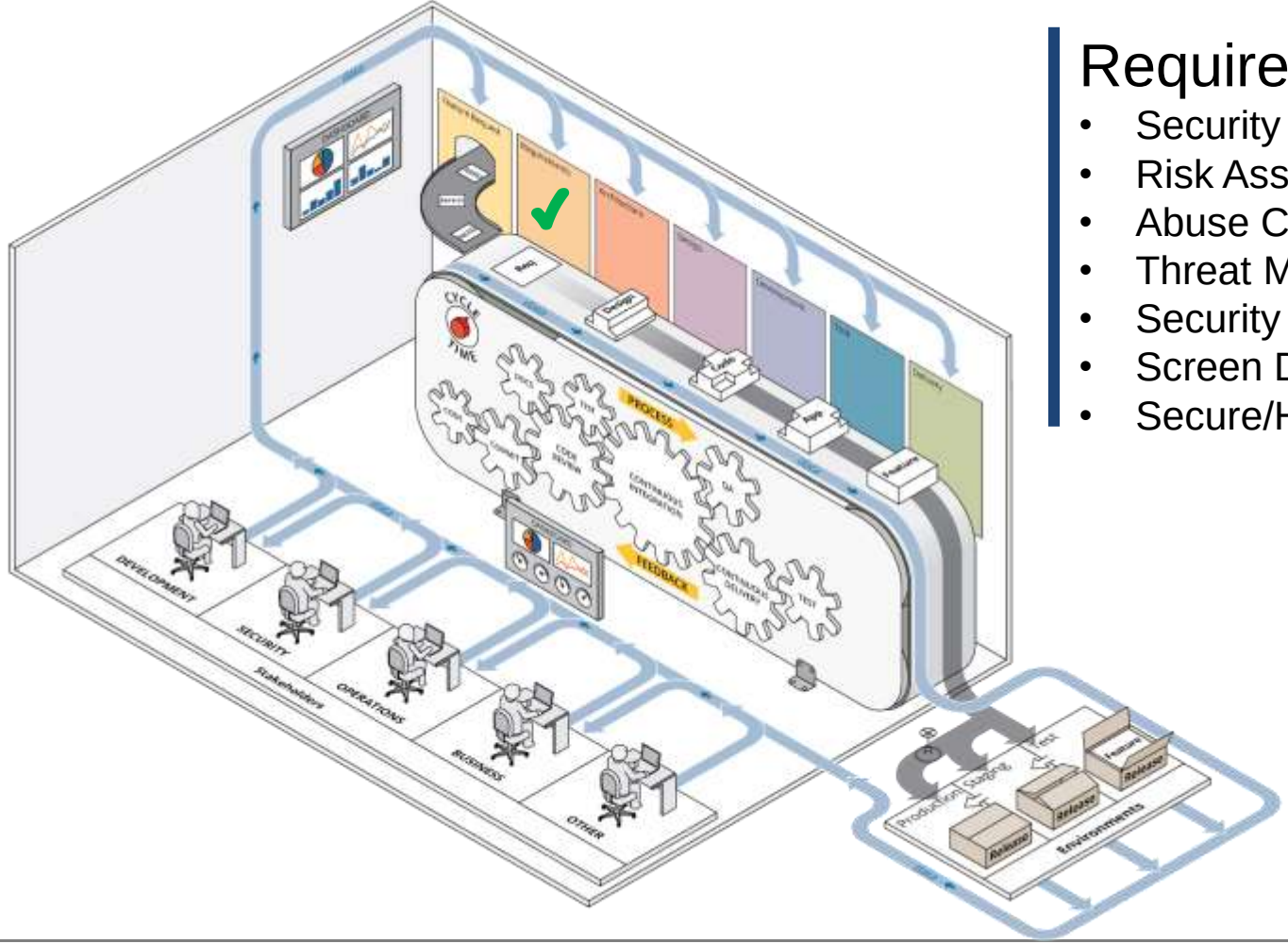
- Feature to deployment
- Iterative and incremental development
- Automation in every phase of the SDLC
- Continuous feedback
- Metrics and measurement
- Complete engagement with all stakeholders
- Transparency and traceability across the lifecycle





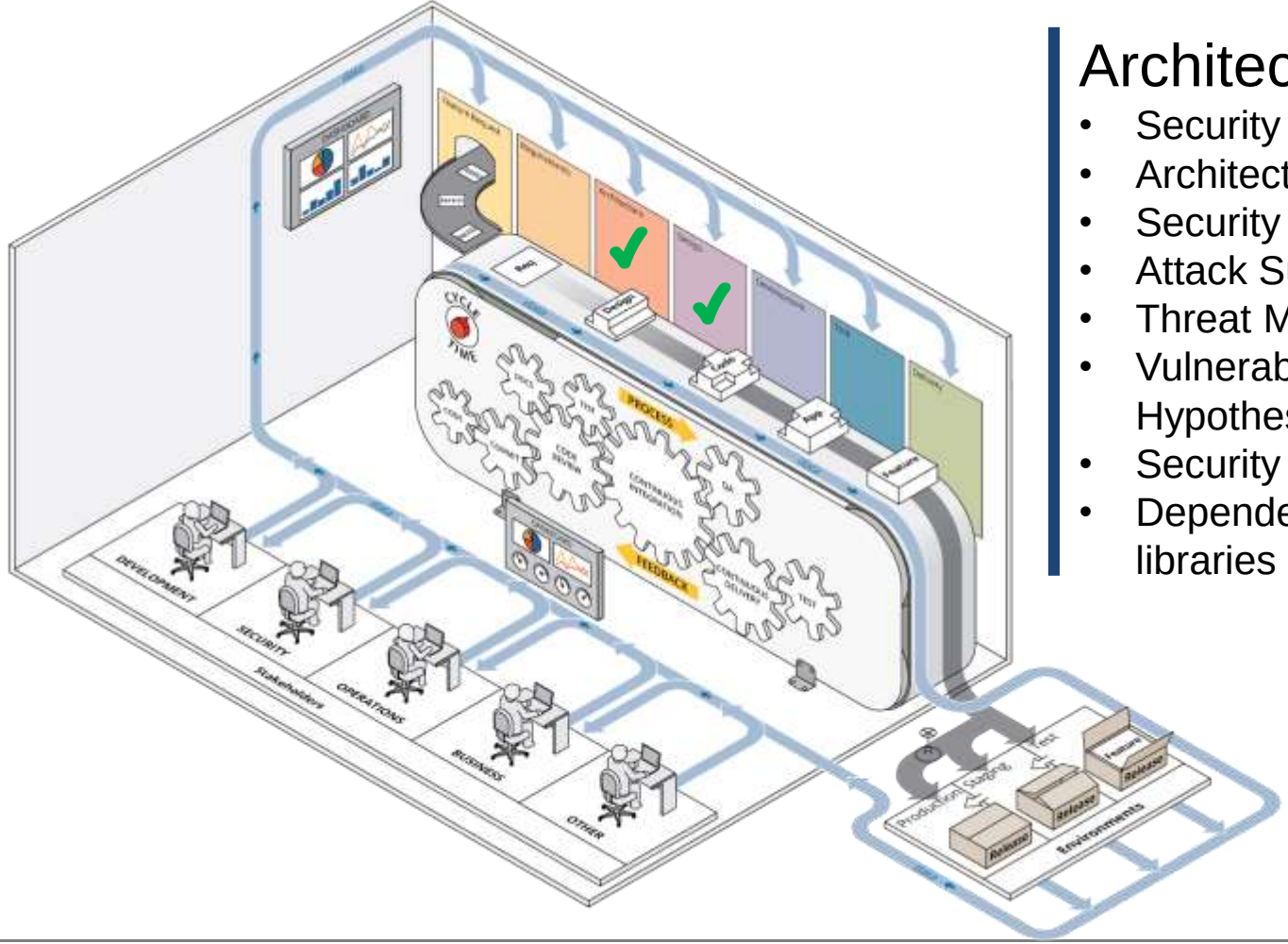
Feature Request

- Strategy & Metrics
- Policy & Governance
- Education & Security Guidance
- Organizational Risk Factors
- Threat Assessment



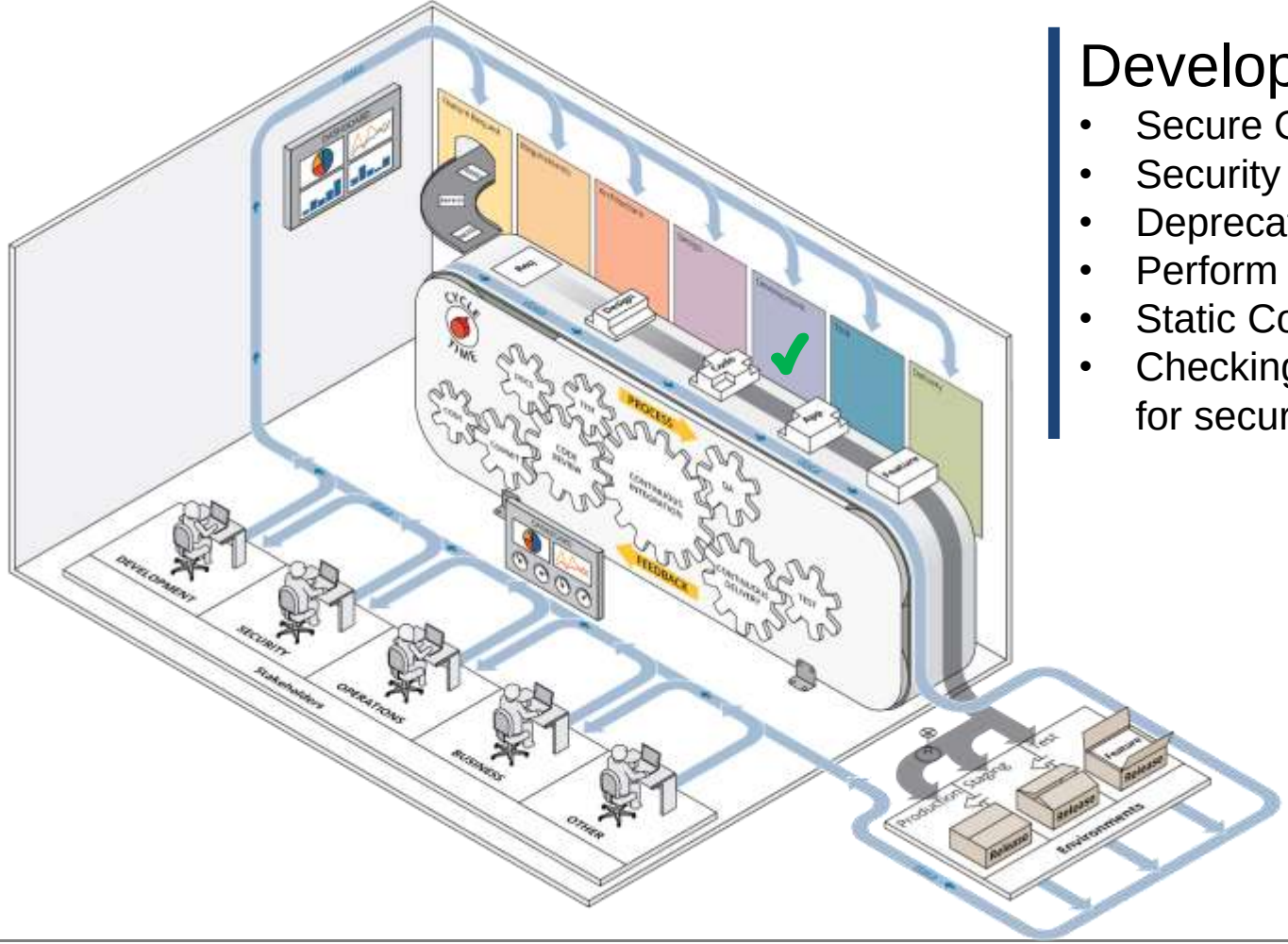
Requirements

- Security Requirements (SFR/SAR)
- Risk Assessment
- Abuse Case Development
- Threat Modelling
- Security Stories
- Screen Development Tools
- Secure/Hardened Environments



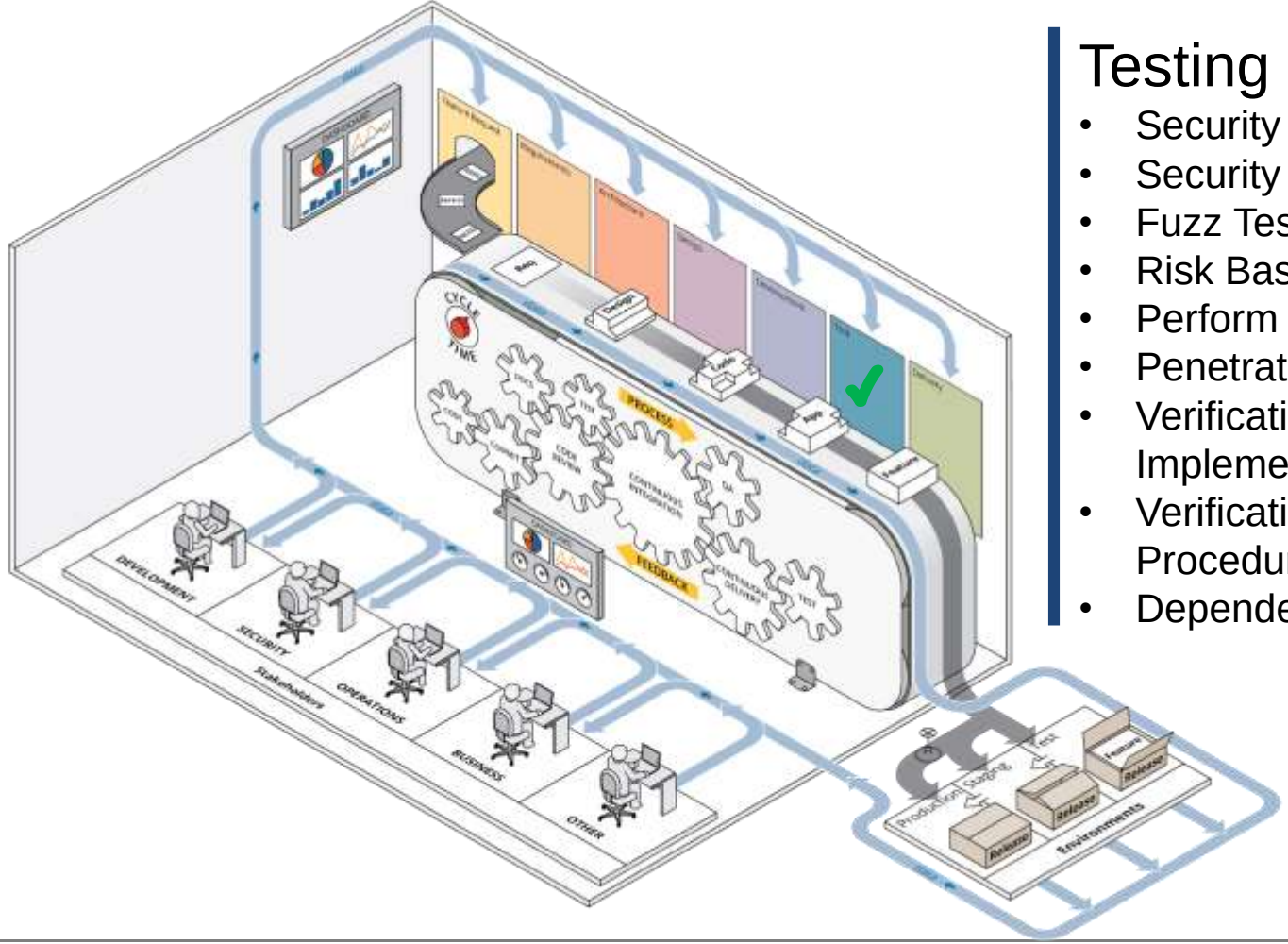
Architecture & Design

- Security Architecture
- Architectural Risk Analysis
- Security Design Requirements
- Attack Surface Analysis
- Threat Modelling
- Vulnerability Analysis and Flow Hypothesis
- Security Design Review
- Dependencies List, Open-source libraries



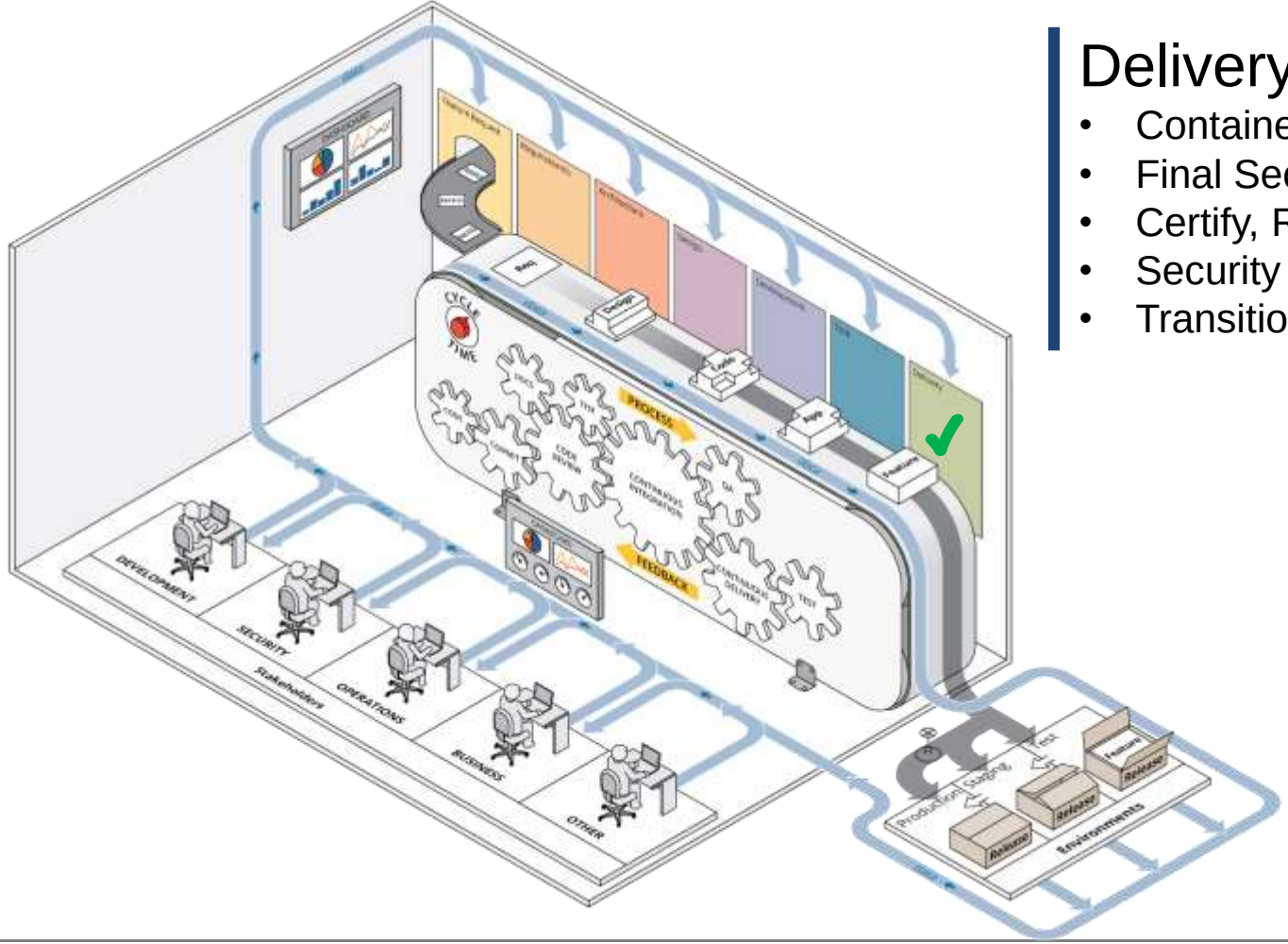
Development

- Secure Coding Practices
- Security Focused Code Review
- Deprecate Unsafe Functions
- Perform Security Unit Testing
- Static Code Analysis
- Checking of process and procedures for secure coding & traceability



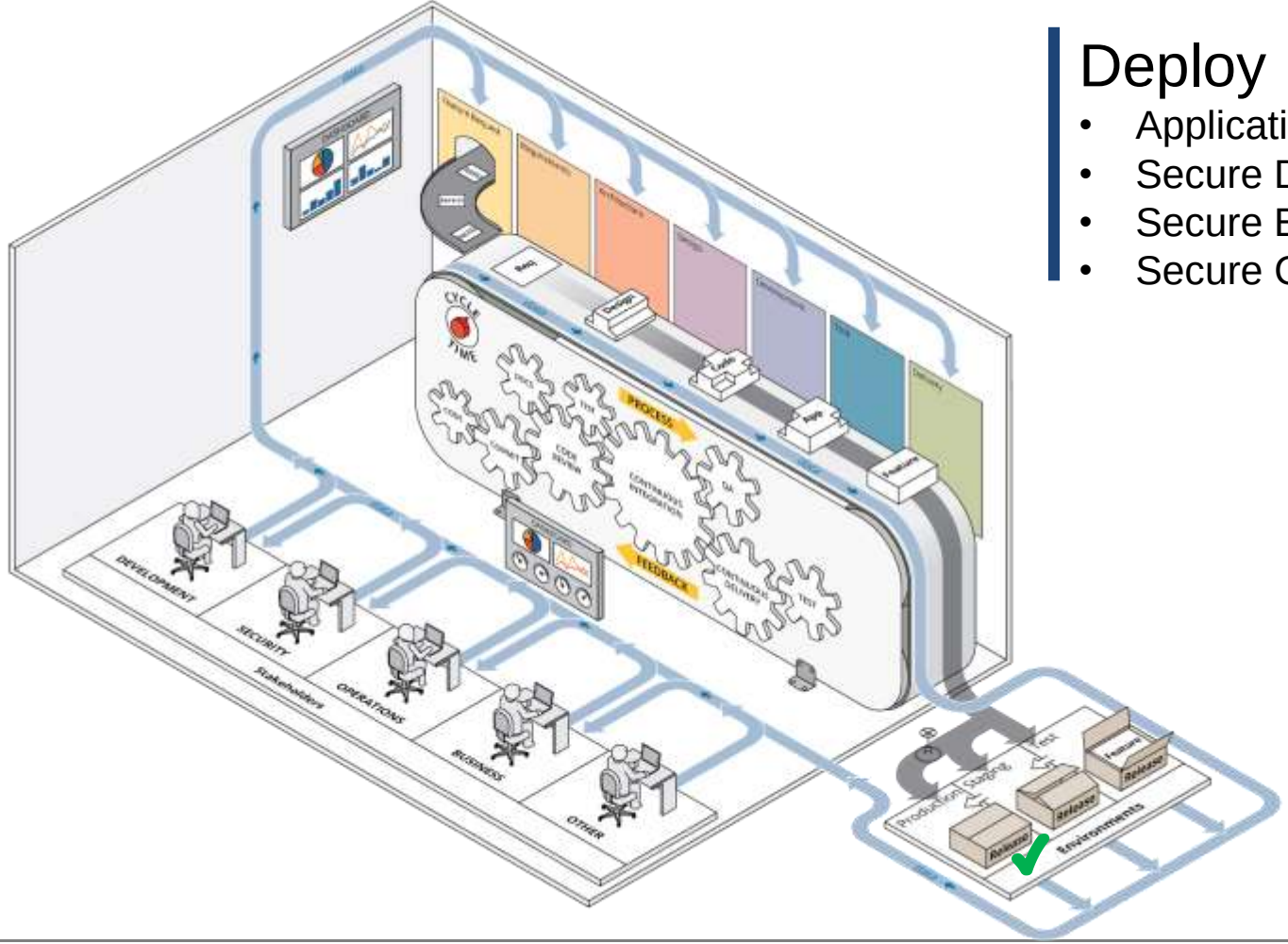
Testing

- Security Test Planning
- Security Testing
- Fuzz Testing
- Risk Based Security Testing
- Perform Dynamic Analysis
- Penetration Testing
- Verification of Security Implementation
- Verification of Process and Procedures
- Dependency Monitoring



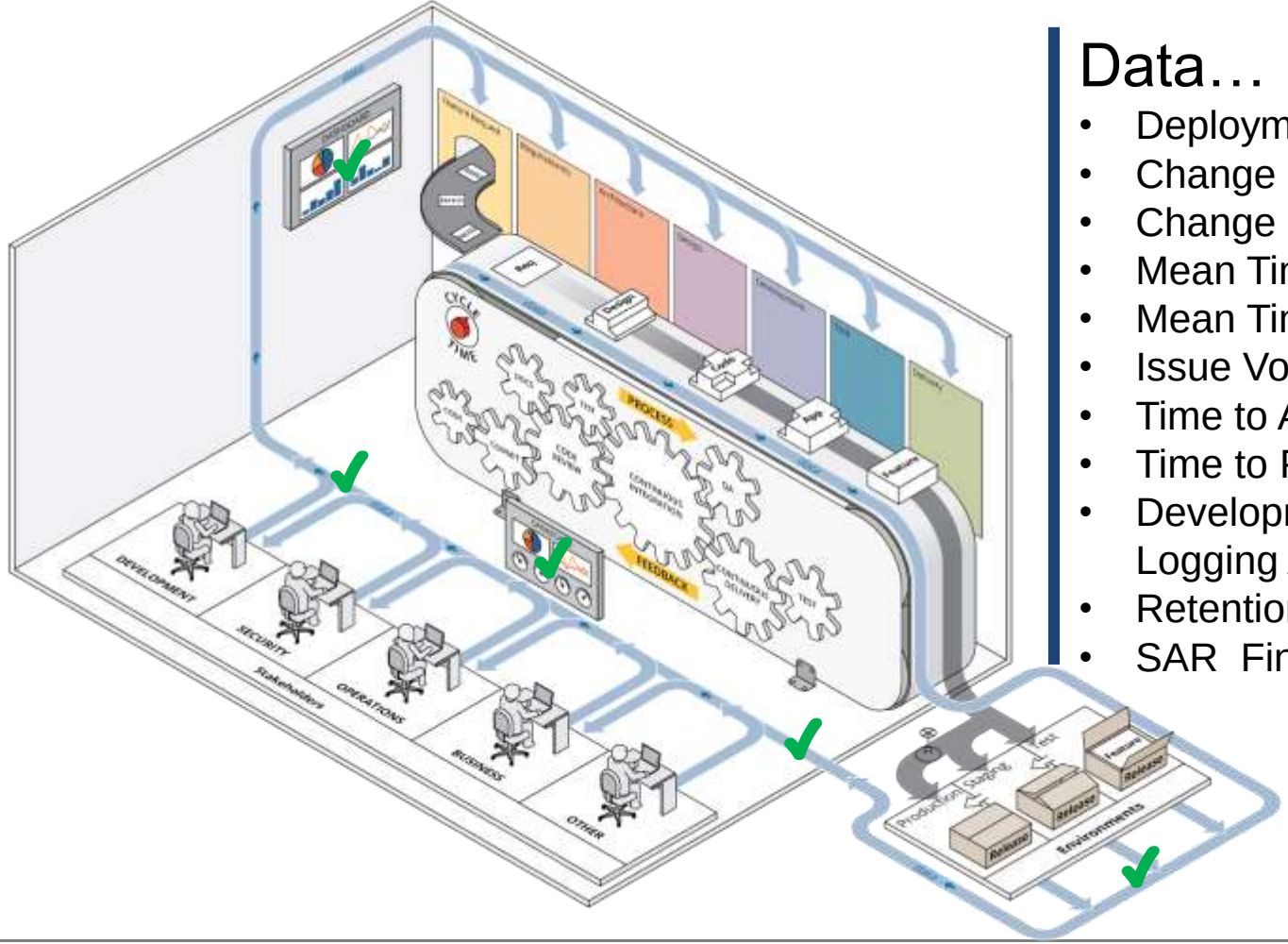
Delivery

- Container Security
- Final Security Review
- Certify, Release and Archive
- Security Acceptance Testing
- Transition Incident Response Plan



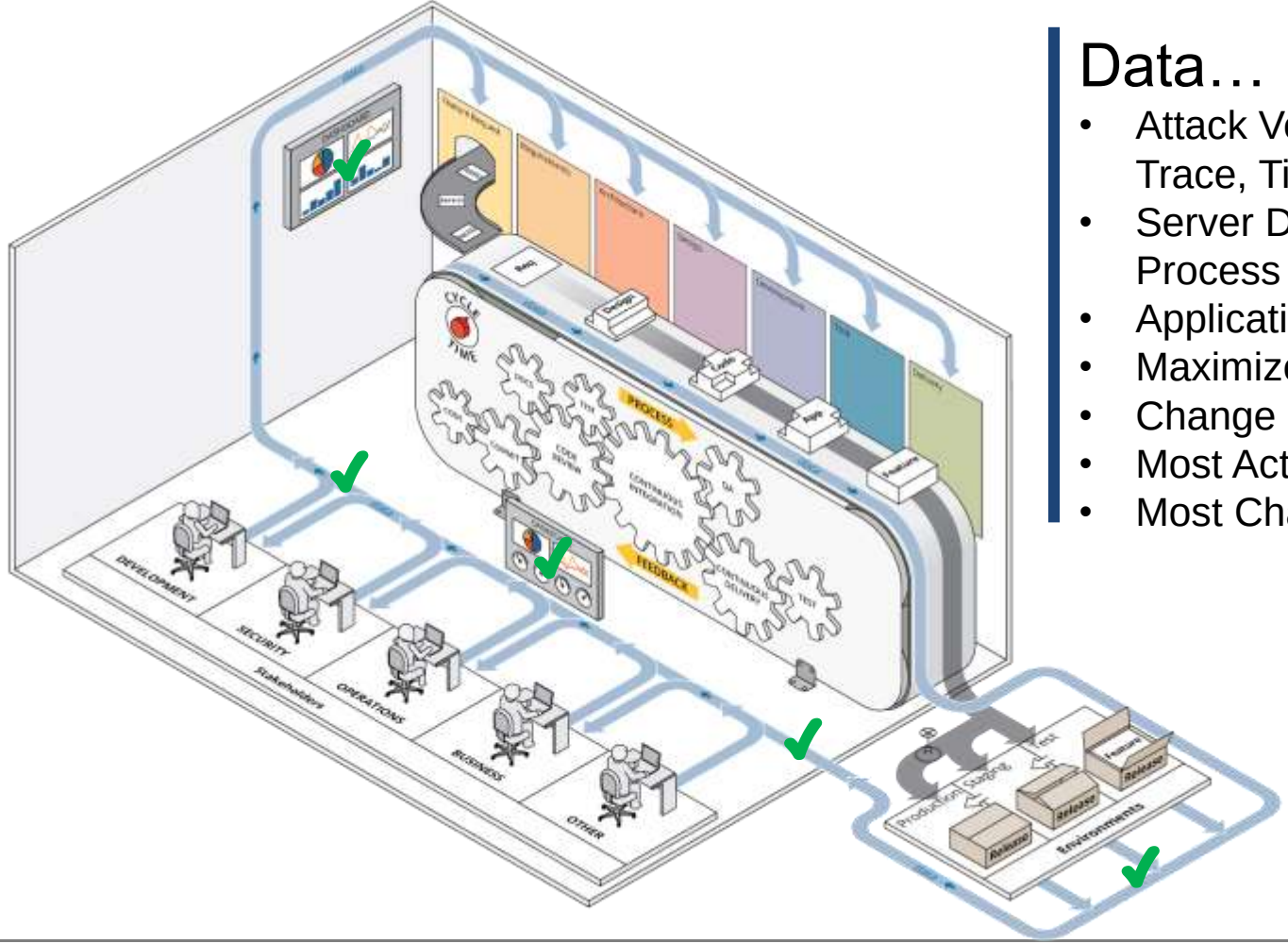
Deploy

- Application Security Monitoring
- Secure Deployment Process
- Secure Environment
- Secure Operational Enablement



Data…

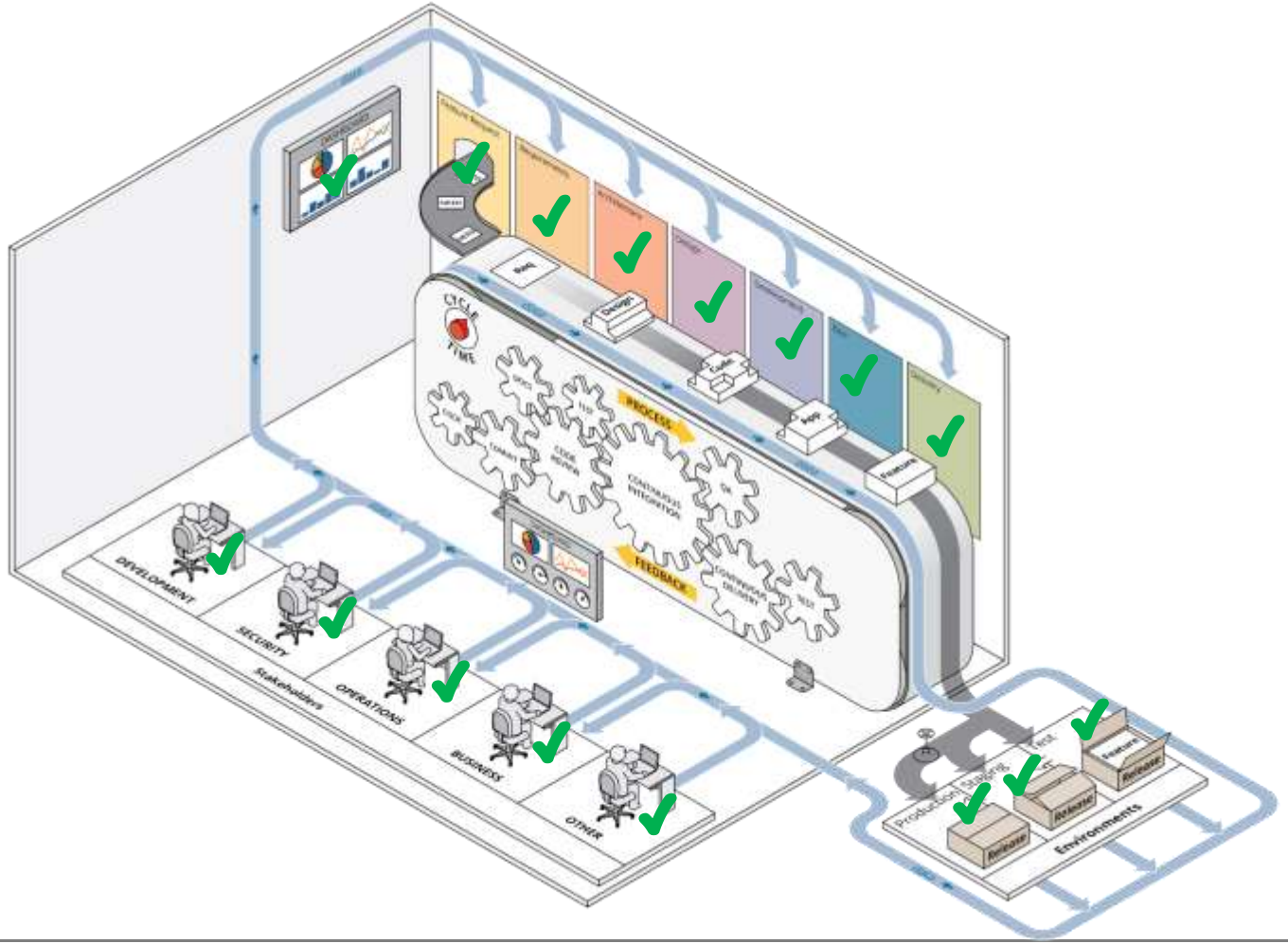
- Deployment Frequency
- Change Lead Time and Volume
- Change Failure Rate
- Mean Time To Recovery (MTTR)
- Mean Time to Detection (MTTD)
- Issue Volume and Resolution Time
- Time to Approval
- Time to Patch Vulnerabilities
- Development and Application Logging Availability
- Retention Control Compliance
- SAR Findings



Data…

- Attack Vector Details (IP, Stack Trace, Time, Rate of Attack, etc)
- Server Disk Space, Load and Process Monitoring
- Application Performance
- Maximize Monitoring
- Change in Size to Code Base
- Most Active Code Contributors
- Most Changed Code Areas

Think Security from Inception to Deploy and improve every delivery



Build Secure Application with DevSecOps!

Lesson Learned

DevOps Assessment: INSCOM 782nd MI BAT Dev Team

Analyzed

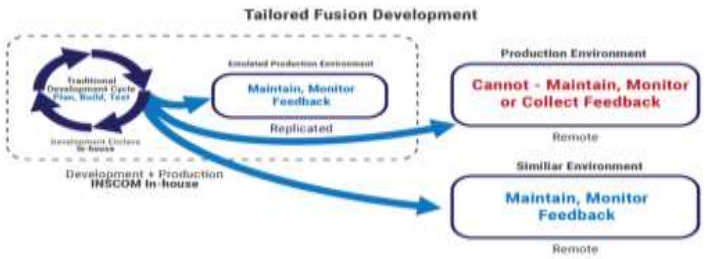
- Software projects: requests, approvals
- SDLC: development, test, deployment

Interviewed:

- Management, Engineering
- Quality Assurance, Operations

Discovered DevSecOps Obstacles

1. Air-gapped deployment, inability to make changes after production deploy
2. Lack of environment parity, unclear target environment descriptions
3. Lack of release versioning
4. Inefficient accreditation process, organizational silos
5. Disruptive involvement of many stakeholders, inconsistent communication channels, project time constraints
6. Lack of a standard documentation source



Takeaways

- Administrative approvals consumes development time
- Onsite approvals needed for fast turn around projects
- Environmental security eases and hardens DevOps implementation

Recommendations

- Utilize containerization or virtualization providing deterministic operating environments, lower risk of mishandling deliverables, and raise awareness of activity from the development, testing, and operations teams.

- Establish an emergency pathway for the development team to assume approval and continue at-risk until the official status makes its way through the proper channels.

- Disparate networks exist housing various documentation. Best to have a custom software solution to coordinate changes between networks or establish a permissible single source repository

DevOps Assessment: NGA GEOINT Services

Analyzed:

- Department wide operational model
- SDLC: development, test, deployment
- Purpose of each model component
- Interactions between components

Interviewed:

- Management, Engineering
- Finance, Operations

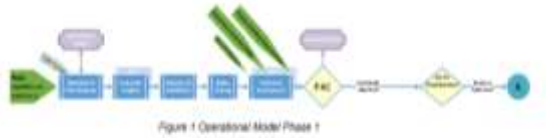


Figure 1 Operational Model Phase 1

Figure 2 Operational Model Phase 1 - Product Development and Testing

Takeaways

- DevOps is mostly cultural and less technical
- Uniform environments facilitates continuous development
- Project planning should include financials, management and security team

Discovered DevSecOps Obstacles

1. Suboptimal communication and collaboration between personnel of various operational model components and between GS and external entities
2. Lack of a GS internal software development environment for government employed software developers.
3. Tasking overlap and vaguely identified purpose of various components in the operational model.
4. Unclear policy of project entry, selection, and management

Recommendations

- Establish policies of enduring communication between various operational model component personnel for all projects.
- Creation of a GS internal DevOps-based development environment for government employed software.
- Modifications to the current model's layout removing and combining vaguely defined components.
- Create a single entry point with technical requirements, financials, and a project manager for software projects.

DevOps Assessment: NCWDG 2017

Analyzed:

- SDLC: development, test, operations
- Current approval processes
- Team composition, and interactions of CIV, GOV, CTR, MIL

Interviewed:

- Developers, Leadership, Operators
- Finance, Operations

Discovered DevSecOps Obstacles

1. Organizational Silos
2. Unclear target environment descriptions
3. Inconsistent communication channels
4. Project time constraints
5. Lack of collaboration space
6. Inability to make changes after code release to production

Takeaways

- Hardware acquisition process posed significant risk
- Hard to provide adequate development time to sailors due to military obligations
- HRE creates challenging DevSecOps processes
- Multiple technologies used, should be consolidated

Recommendations

- Facilities must be built to provide communication, collaboration, and environment parity for development.
- Use interoperable development tools to enable CI/CD, collaboration.
- Provide training on Agile, and DevOps.
- Pilot a project to demonstrate ideal DevSecOps interactions, processes, and workflows.
- Define standard SW development practices

Summary

- This is as much a culture shift as it is a technology shift
  - Blame-Free Culture, Cross-Silo Goals, Optimize Ease-of-Use
- Organizational Structure
  - Siloed on Acq, O&M and Security departments, Org structure based on system architecture
- Legacy Systems
  - Lack of modular architecture, old tools/language
  - DevSecOps is independent and agnostic of the domain
- DevSecOps is not just tooling
  - DevSecOps requires policy, skilled stakeholders, mindset changes
- Lack of Metrics and Measurements
  - Team should decide what to measure and continuously monitor
- DevSecOps is not barrier to ATO/Compliances
  - Design DevOps pipeline to comply with governance security (Built in)
- Inconsistent Environment
  - Environment Parity , shared infrastructure, utilize IaC
- Sustainment of DevSecOps pipeline
  - Tool updates, project configuration, tool license cost, effective usage of the platform, combability with the program needs

SEI/DSOI team GitHub Projects

- Once Click DevOps deployment
  https://github.com/DSOI-ALL/devops-microcosm
- Sample app with DevOps Process
  https://github.com/SLS-ALL/flask_api_sample
  - Tagged checkpoints
    - v0.1.0: base Flask project
    - v0.2.0: Vagrant development configuration
    - v0.3.0: Test environment and Fabric deployment
    - v0.4.0: Upstart services, external configuration files
    - v0.5.0: Production environment
- On YouTube:
  https://www.youtube.com/watch?v=5nQlJ-FWA5A

For more information…

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

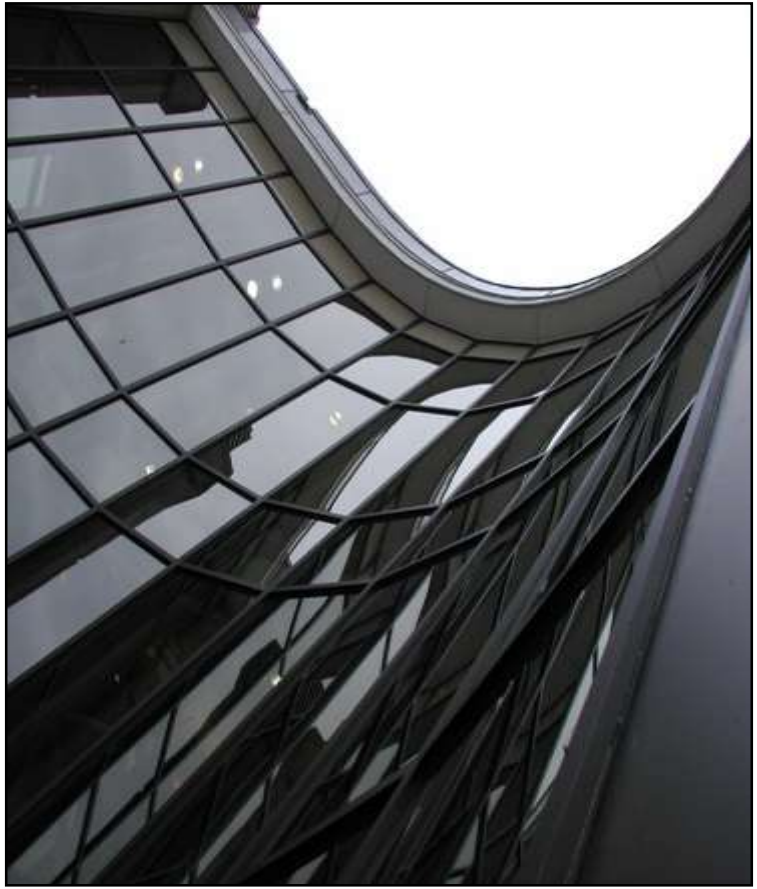
Webinar : https://www.sei.cmu.edu/publications/webinars/index.cfm

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

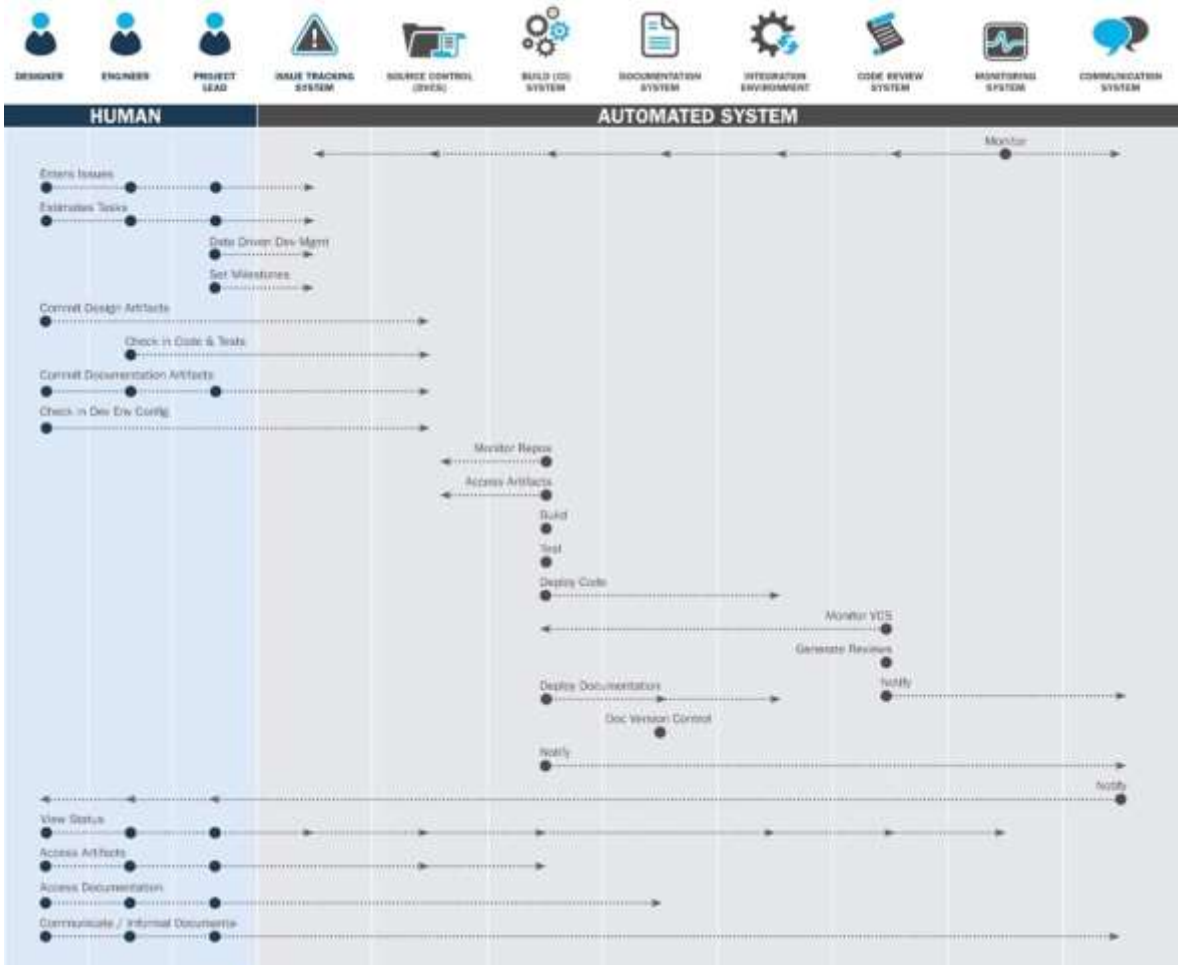
Any Questions?

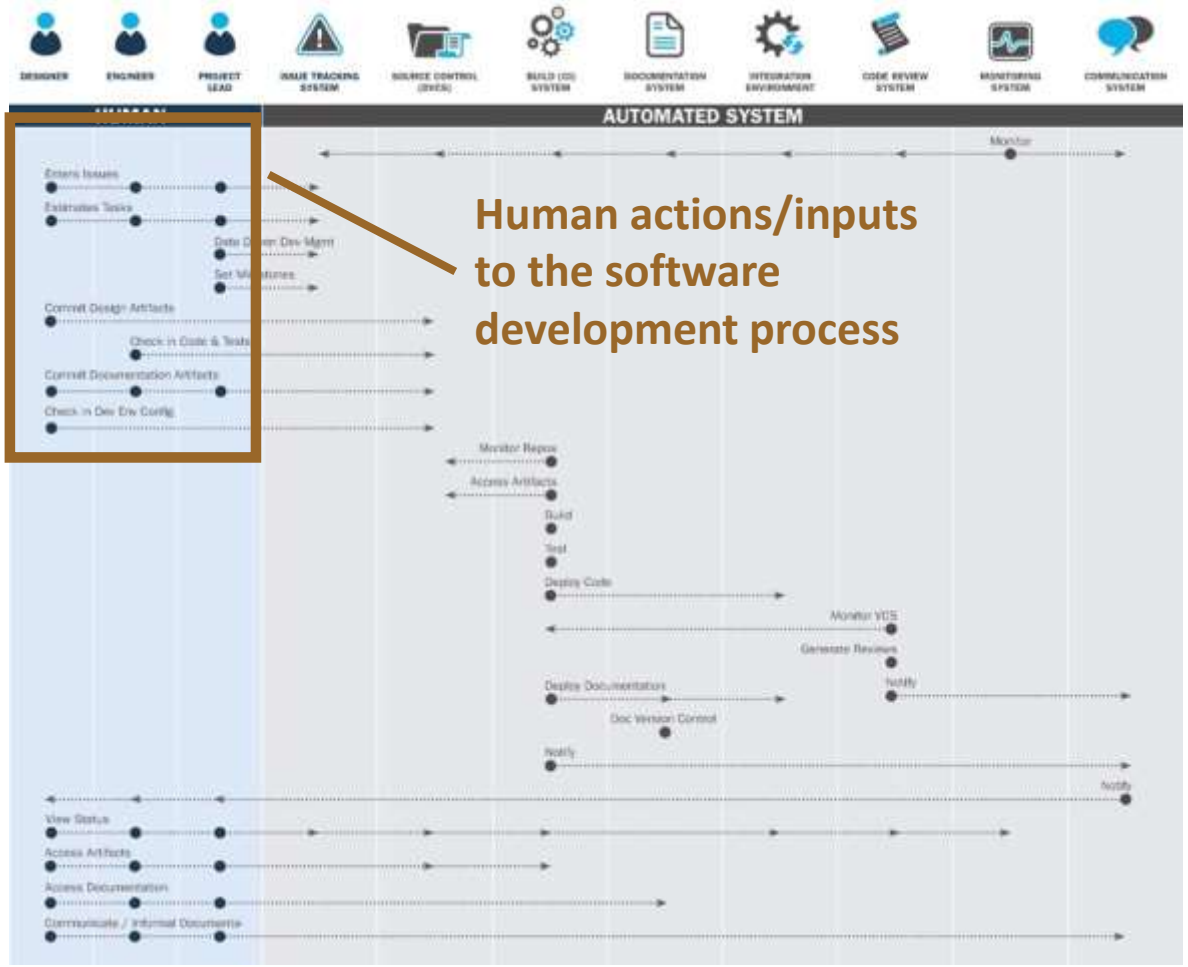
Thomas Longstaff
Chief Technology Officer
tal@sei.cmu.edu

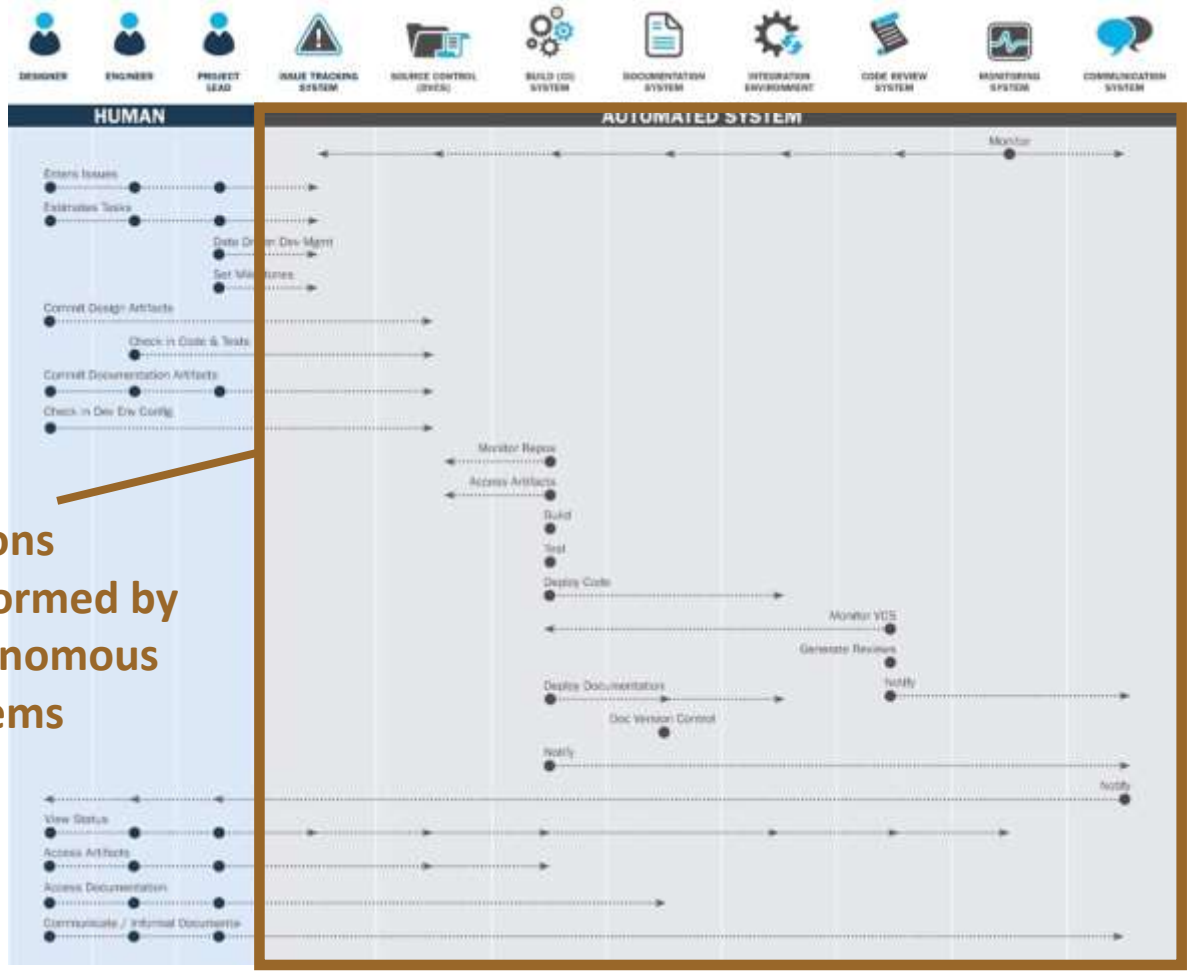
Hasan Yasar
Technical Director,
Continuous Deployment of Capability
hyasar@sei.cmu.edu



Backup







Actions performed by autonomous systems