

Build Secure Applications and Systems with **DevSecOps**

Doug Reynolds

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2022 Carnegie Mellon University.

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.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

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.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM22-0843

Build Secure Applications and Systems 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?

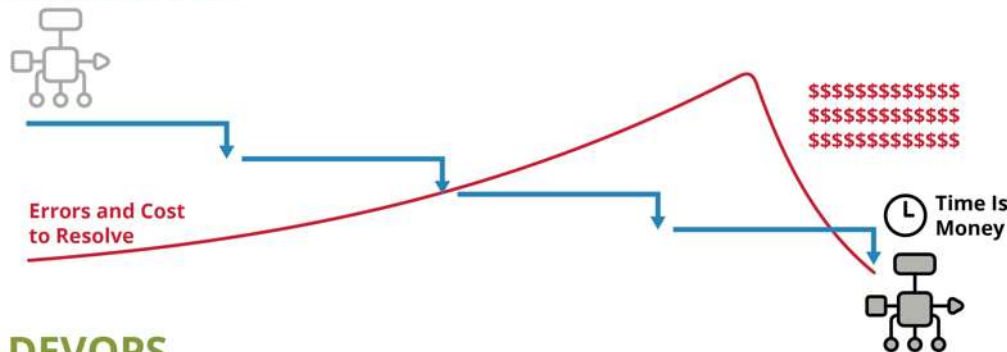


- Operations
 - Runs the application
 - Manages the infrastructure
 - Support the applications
- Operations provides
 - Service Strategy
 - Service Design
 - Service Transition
 - Service Operations
 - Secure systems

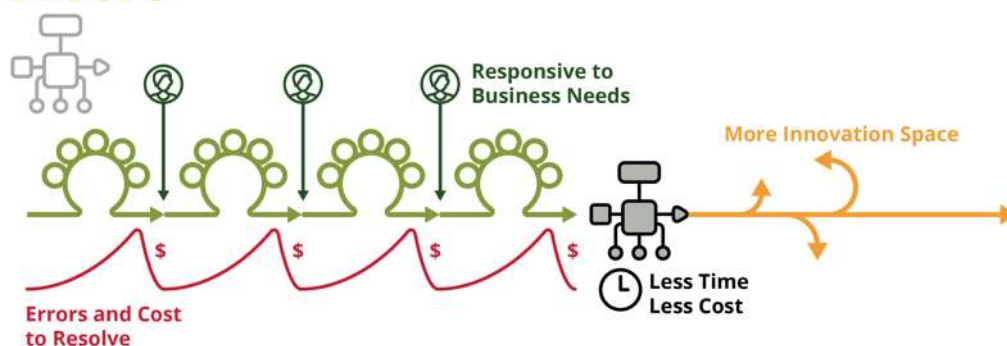
Want to maintain stability, reliability and security...

Key Benefits of DevOps

WATERFALL



DEVOPS



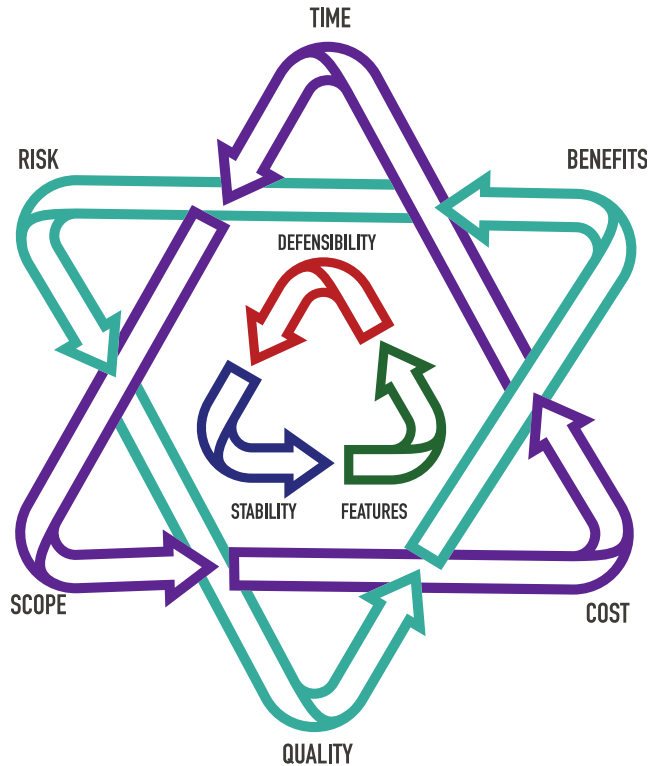
- Reduced errors during deployment
- Reduced time to deploy and resolve discovered errors
- **Repeatable** steps
- **Continuous availability** of pipeline and application
- Increased innovation time
- **Responsiveness** to business needs
- **Traceability** throughout the application lifecycle
- Increased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

Build Secure Applications and Systems with DevSecOps

DevSecOps



DevSecOps: Simply a term for modern software engineering practices and tools that encompasses the full software lifecycle.



DevSecOps is a cultural and **engineering practice** that breaks down barriers and opens **collaboration between development, security, and operations** organizations **using automation** to focus on rapid, frequent delivery of secure infrastructure and software to production. It encompasses intake to release of software and manages those flows predictably, transparently, and with minimal human intervention/effort [1].

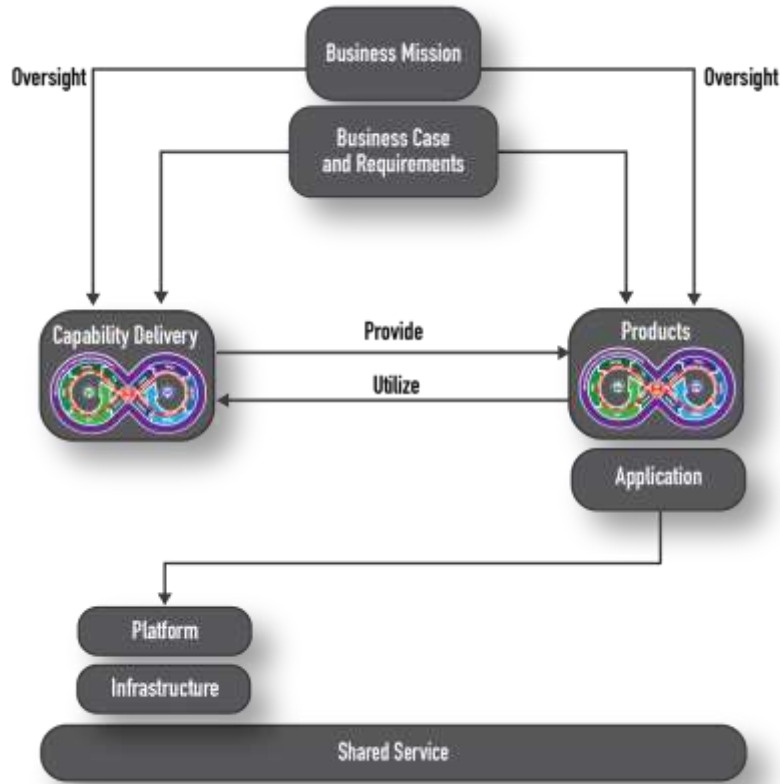
A **DevSecOps Pipeline** attempts to seamlessly integrate “three traditional factions that sometimes have opposing interests:

- **development**; which values features;
- **security**, which values defensibility; and
- **operations**, which values stability [2].”

Not only does one need to balance the factions. They must do so in a way that balances **risk**, **quality** and **benefits** within their **time**, **scope**, and **cost** constraints.

[1] DevSecOps Guide: Standard DevSecOps Platform Framework. U.S. General Services Administration. https://tech.gsa.gov/guides/dev_sec_ops_guide. Accessed 17 May 2021.
[2] DevSecOps Platform Independent Model, <https://cmu-sei.github.io/DevSecOps-Model/>

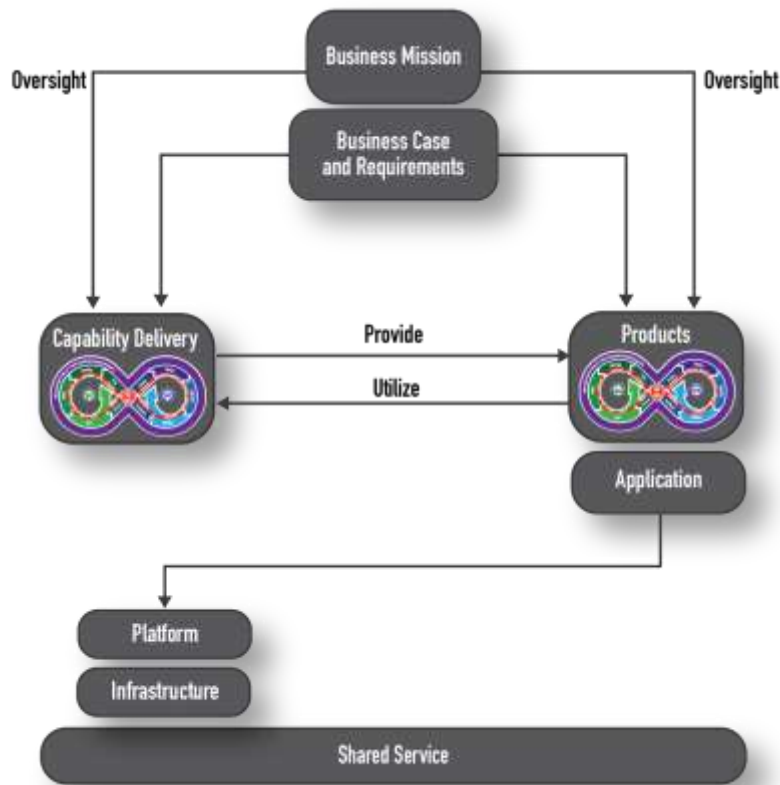
An Enterprise View



All DevSecOps-oriented enterprises are driven by three concerns:

- *Business Mission* - captures stakeholder needs and channels the whole enterprise in meeting those needs. It answers the questions Why and For Whom the enterprise exists
- *Capability to Deliver Value* - covers the people, processes, and technology necessary to build, deploy, and operate the enterprise's products
- *Products* - are the units of value delivered by the organization. Products utilize the capabilities delivered by the software factory and operational environments.

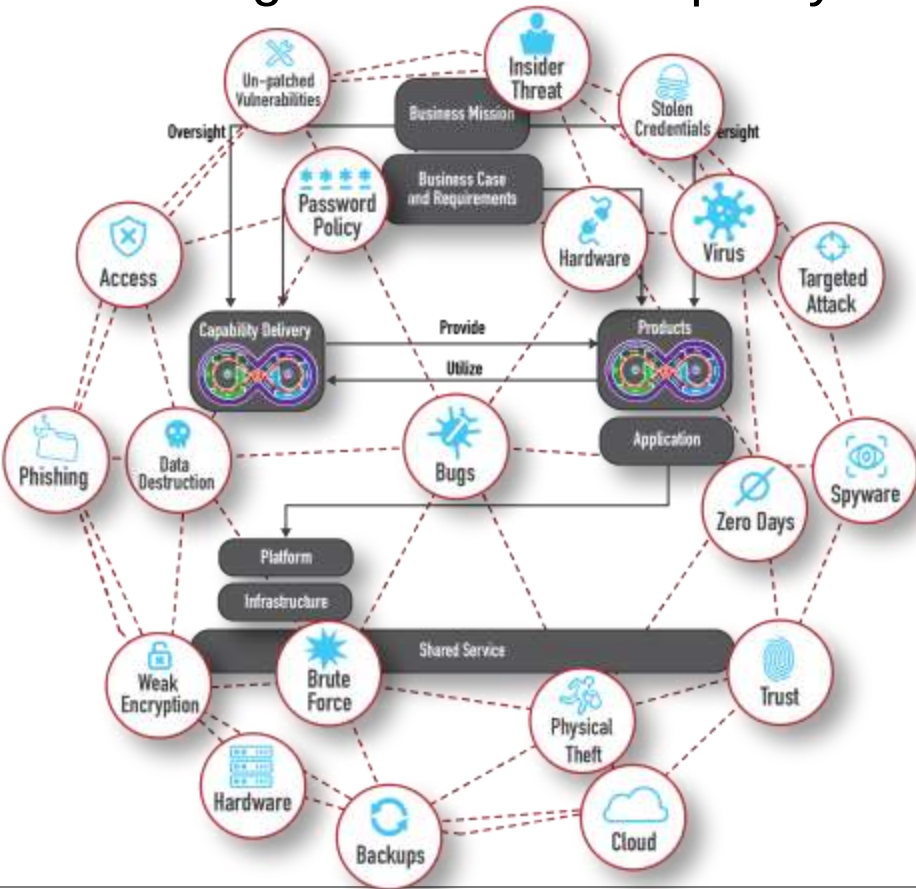
Challenge 1 for DevSecOps: connecting process, practice, & tools



Creation of the DevSecOps (DSO) pipeline for building the product is not static.

- Tools for process automation must work together and connect to the planned infrastructure
- Everything is software and all pieces must be maintained but responsibility will be shared across multiple organizations (Cloud for infrastructure, 3rd parties for tools and services, etc.)

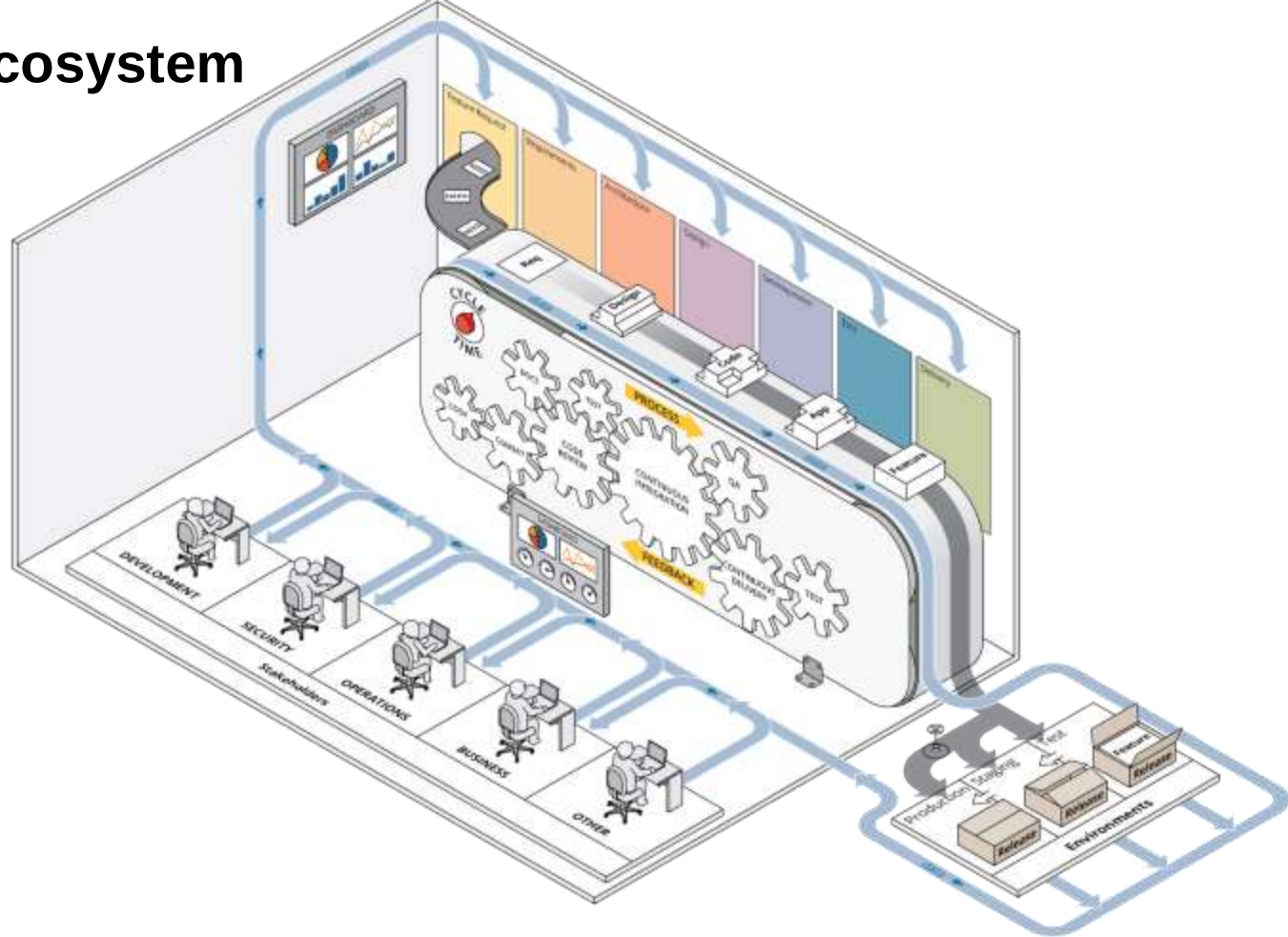
Challenge 2 for DevSecOps: cybersecurity of pipeline and product



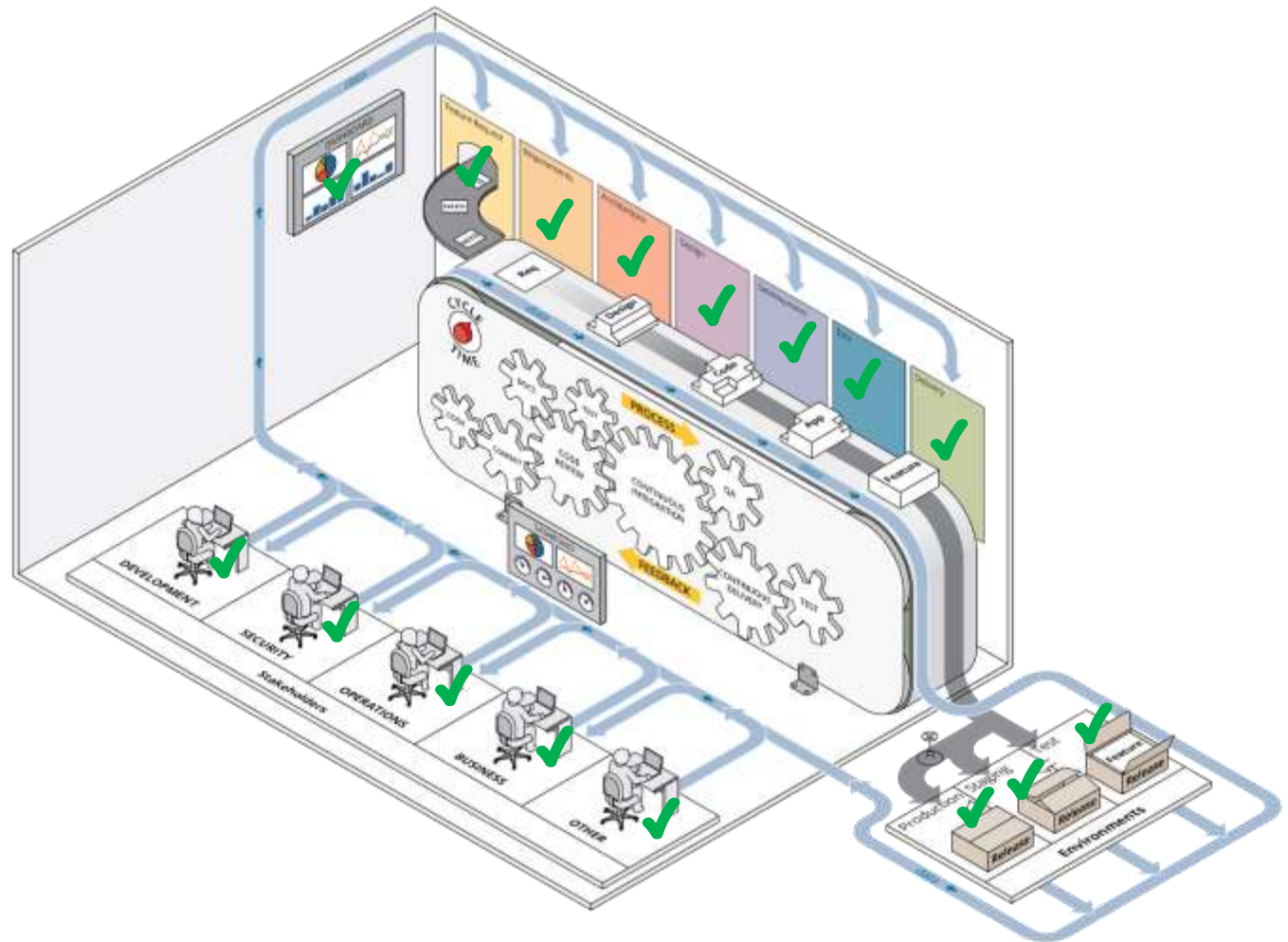
- The tight integration of Business Mission, Capability Deliver, and Products, using integrated processes, tools, and people, increases the attack surface of the product under development.
- Managing and monitoring all of the various parts to ensure the product is built with sufficient cybersecurity and the pipeline is maintained to operate with sufficient cybersecurity is complex.
- How do you focus attention to areas of greatest concern for security risks and identify the attack opportunities that could require additional mitigations?

The DevSecOps Ecosystem

- 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



Think Security
from Inception to
Deploy and
improve every
delivery



Build Secure Applications and Systems with DevSecOps

DevSecOps with Hardware in the Loop

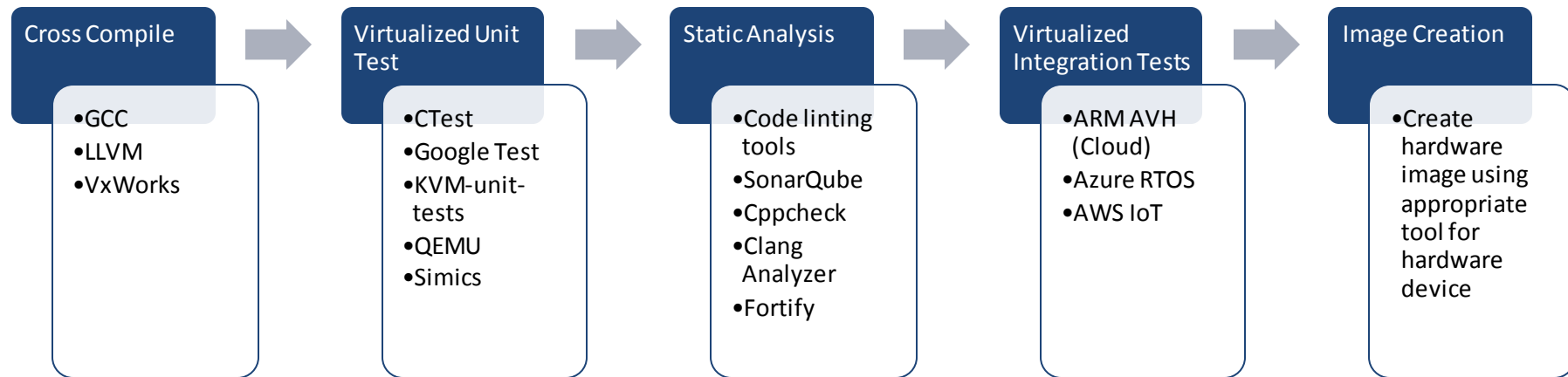


DevSecOps Hardware: Identify target hardware and toolchain

- Embedded real-time operating system
 - PowerPC
 - ARM
 - RISC-V
- Field-programmable gate array (FPGA)
 - VHDL
 - Verilog
 - System Verilog
- System on a Chip (SoC)
 - Hybrid embedded operating system running along with an FPGA design

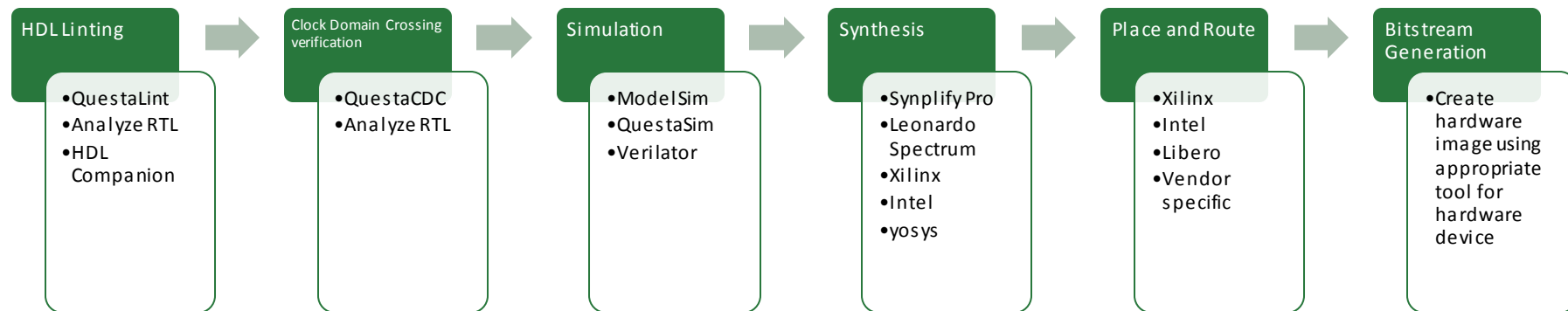
DevSecOps Hardware: Embedded toolchain for target hardware

Embedded software DevSecOps follows a similar path as standard software

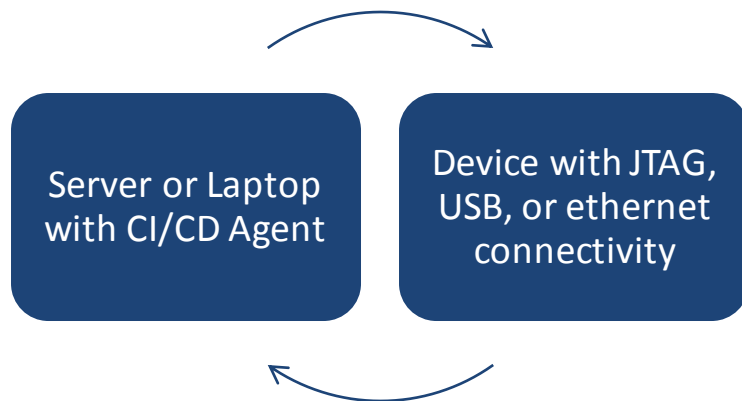


DevSecOps Hardware: FPGA toolchain for target hardware

FPGA DevSecOps requires specialized tools for RTL designs



DevSecOps Hardware: Device Connectivity



- Programming is triggered via CI/CD pipeline and device is programmed via its JTAG connection
- Various integration or smoke tests can be performed, utilizing a serial or network connection and common test frameworks, such as Pytest, JUnit, or RSpec, depending on the function of the device

DevSecOps Hardware: Device Connectivity Challenges

Device connectivity varies depending on the device and the environment in which it operates:

- Can block continuous delivery in a “production” environment since the device has to be pulled out of service for an update
- Can block device feedback outside of an integration laboratory and requires onboard diagnostics logging or monitoring
- Hardware device may have limited bandwidth for programming which will slow down the design and test cycle
- User feedback and acceptance testing is likely a manual process

Any Questions?

Doug Reynolds

DevOps Engineer

Software Solutions Division

Continuous Deployment of Capability

DevSecOps Innovations Group

djreynolds@sei.cmu.edu

