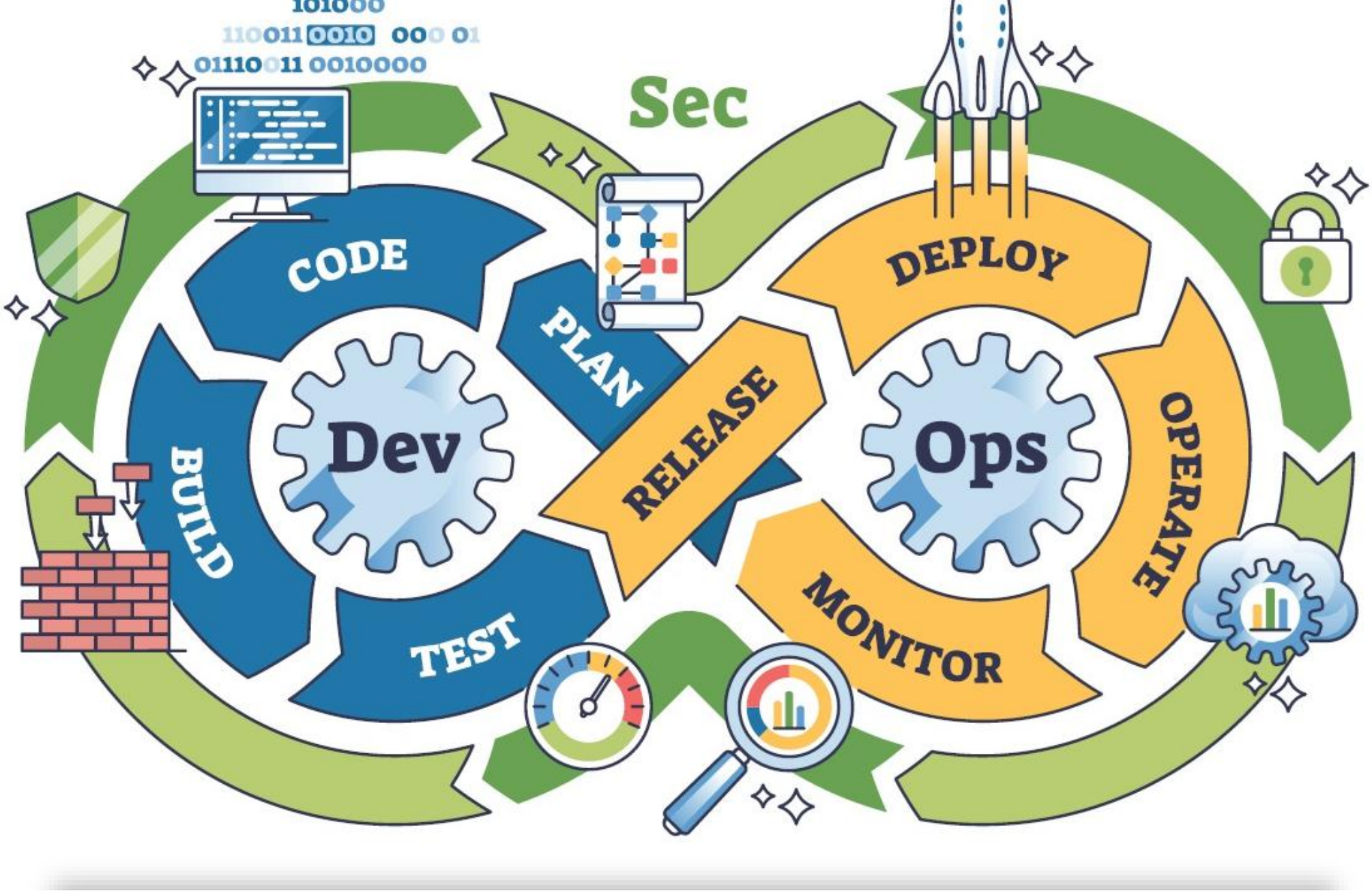


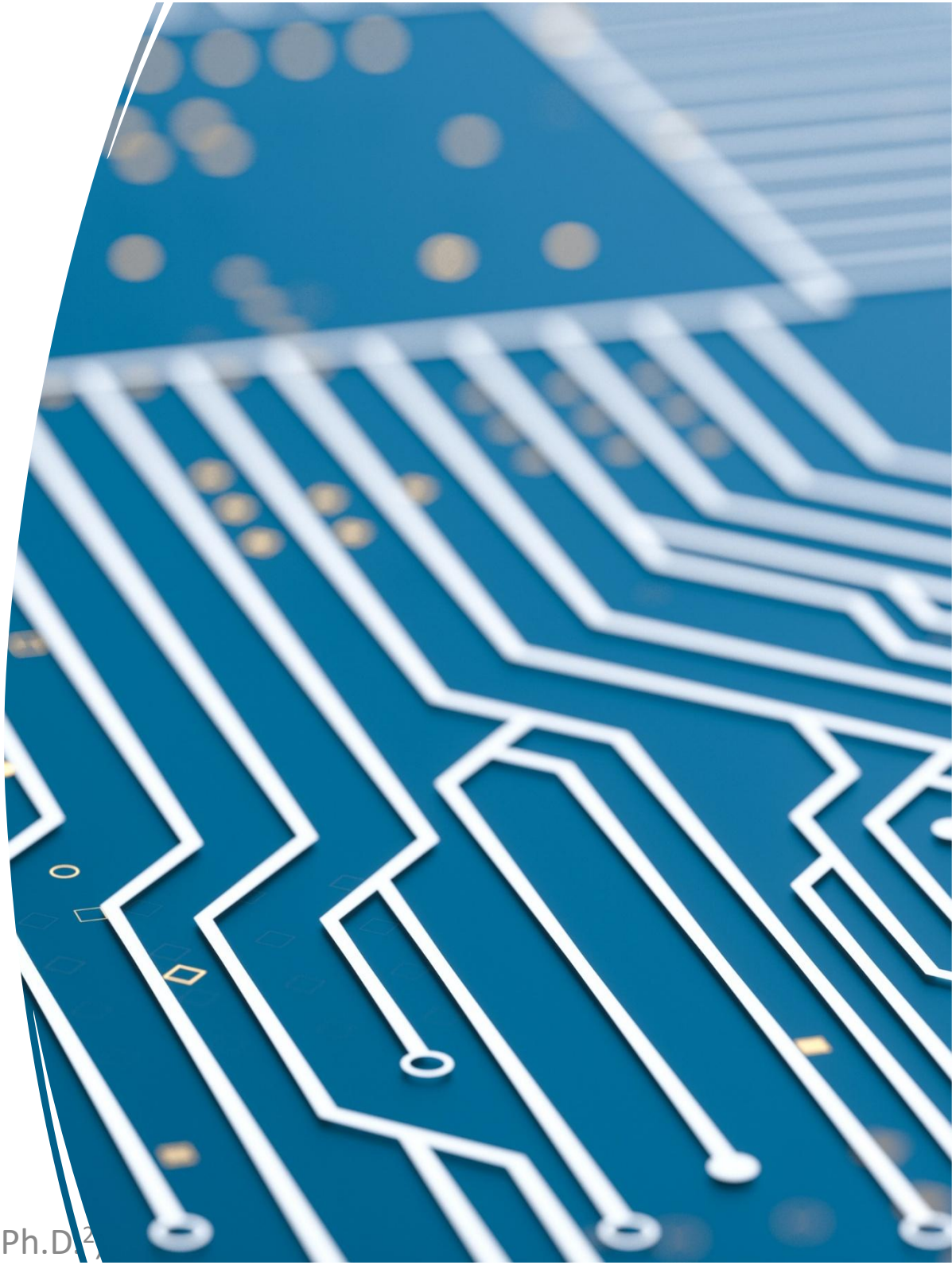
DevSecOps

Dr. Chuck Easttom

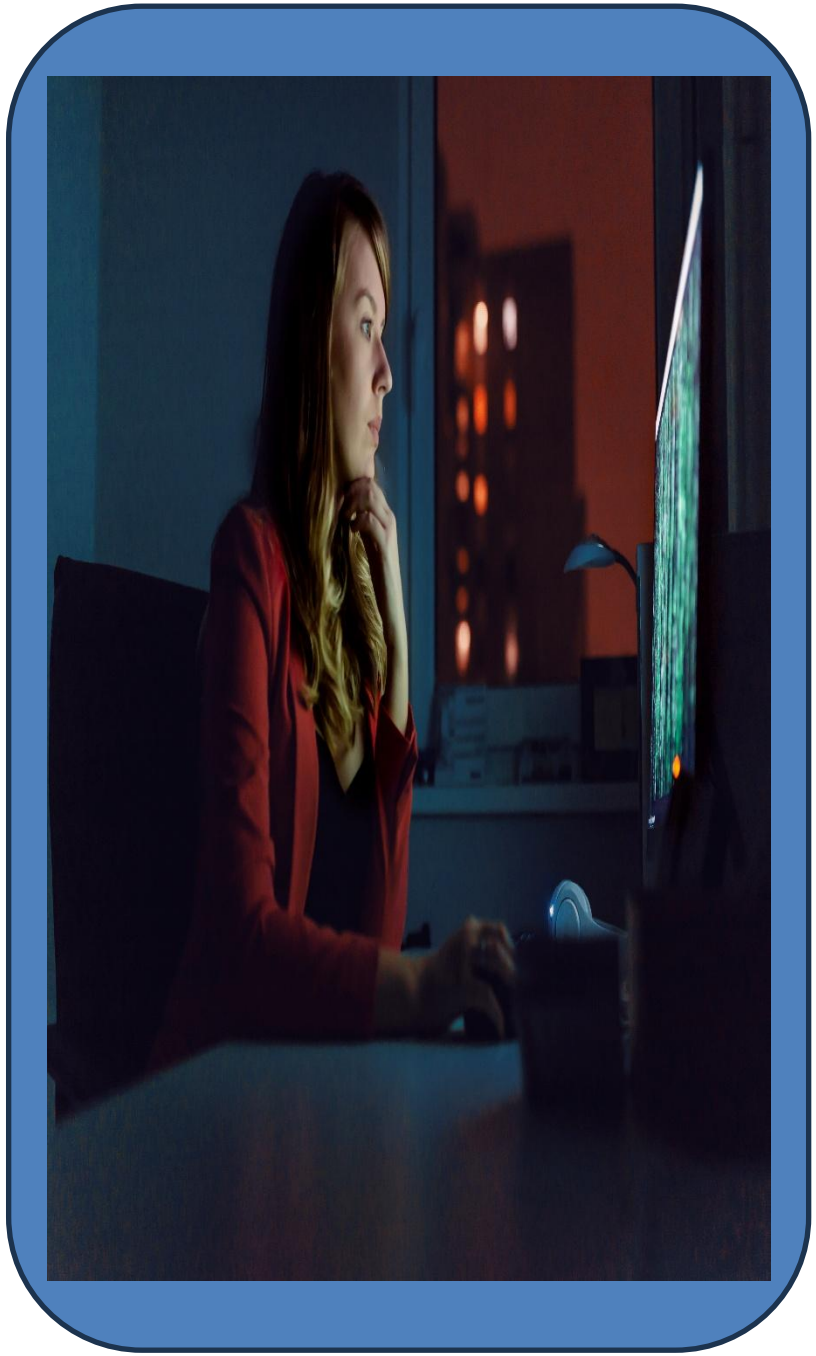
Who Am I

- Ph.D. Computer Science,
- Ph.D. Nanotechnology
- D.Sc. Cybersecurity
- Four masters (systems engineering, education, applied computer science, strategic and defense studies) 5th one in progress (Biology)
- 45 books
- 27 patents
- 80+ Industry certifications

Chuck Easttom, M.Ed., MBA, MSDS, MSSE, Ph.D.[2], D.Sc.
www.ChuckEasttom.com
chuck@chuckeasttom.com



Relevant to Current Course



- Over 30 years of professional software development experience
- Vice Chair of IEEE P23026 - Systems and Software Engineering -- Engineering and Management of Websites for Systems, Software, and Services Information from 2018-2021.
- Member of IEEE Software & Systems Engineering Standards Committee. Working on DevOps 2675 from 2017-2020
- Teach secure software engineering graduate course for Vanderbilt

Case 1

November 2012 The United States Air force had to cancel an Enterprise Resource Planning software project named "The Expeditionary Combat Support System" after it had cost approximately 1 Billion US Dollars but "failed to create any significant military capability". The costs are attributed to an "overwhelming" amount of additional custom coding and integration. It was determined that to complete the project would take another 1.1 billion dollars to get 1/4 of the original scope.

The U.S. Government Accountability Office released a report in March of 2012 that found many ongoing ERP projects by the nation's military are drastically behind schedule and over budget.

http://www.cio.com/article/721628/Air_Force_scraps_massive_ERP_project_after_racking_up_1_billion_in_costs

What is DevSecOps?

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

What is DevSecOps?

DevSecOps, and its predecessor DevOps, is a culture and philosophy. DevSecOps builds upon the value proposition of DevOps by expanding its culture and philosophy to recognize that maximizing cyber survivability requires integrating cybersecurity practices throughout the entire systems development lifecycle (SDLC). DevSecOps advances the growing philosophy and sentiment that reliance upon bolt-on or standalone monolithic cybersecurity platforms is incapable of providing adequate security in today's operational environments. Cybersecurity tooling that is fully isolated from the development and operational environments are *reactive* at best, whereas integrated automated tooling with the software factory is *proactive*.

- - DoD Enterprise DevSecOps Fundamentals

-DoD DevSecOps Fundamentals Playbook March 2021

DevSecOps Culture

Checklist

Learn what is involved in the DevSecOps culture.

Embrace automation for anything done repeatedly.

Read How to Build a Strong DevSecOps Culture by K. Casey, available online at: https://enterprisersproject.com/article/2018/6/how-build-strong-devsecops-culture-5-tips

Read The Phoenix Project: A Novel about IT, DevOps, and Helllarng Your Business Win by G. Kim, K. Behr, and G. Spafford, IT Revolution Press, Jan. 10, 2013

Fail fast, learn fast, fail small, and do not fail twice for the same reason!

Software Factory

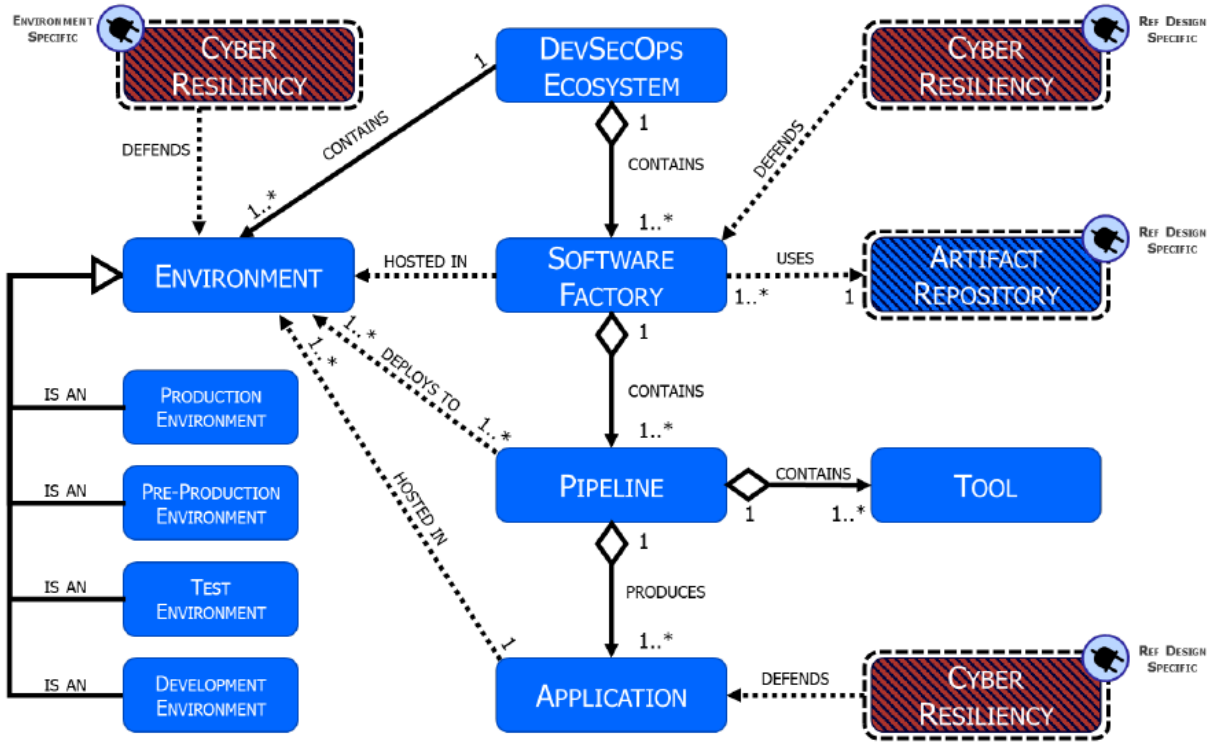
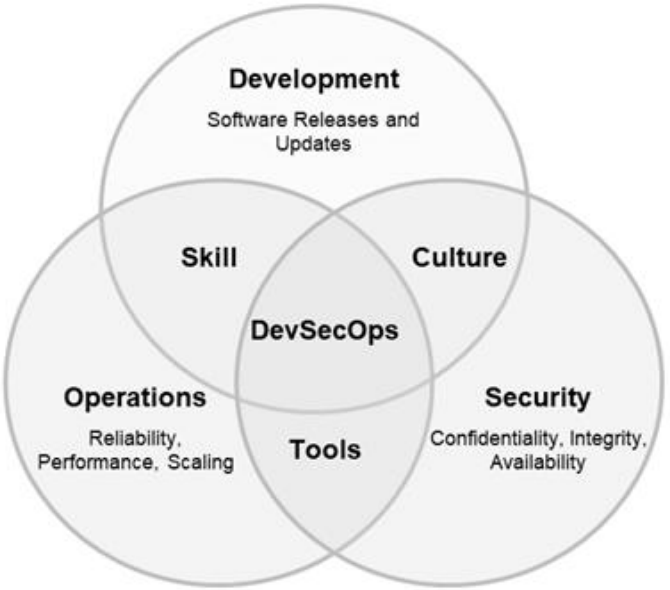


Figure 15 DevSecOps Conceptual Model with Cardinalities Defined

DoD Enterprise DevSecOps Fundamentals March 2021

Venn Diagram of DevSecOps

- DevSecOps is the integration of security practices in the software delivery model of DevOps
- In DevSecOps, everyone is responsible for security
- The goal of DevSecOps is to incorporate security in all the phases of the software development lifecycle by integrating security through tools, encouraging cross-skills, and establishing security as a code culture



Psychological Safety

New concepts inherently come with a degree of skepticism and uncertainty. Within the DoD, DevSecOps is a new concept, and the entire span of our workforce, from engineering talent, to acquisition professionals, through our leadership have many questions on this topic. The success of the commercial industry in using these practices has been widely documented.3 There are leaders who want DevSecOps, but cannot tell if they are already practicing DevSecOps, or how to effectively communicate their practices if they do. Acquisition professionals routinely struggle to understand how to effectively buy services predicated upon DevSecOps due to the perception that it is hard to put tangible frames around and a price tag on something seemingly conceptual. Skepticism and uncertainty can also drive undesirable actions and reactions across the DoD, such as bias and fear. It is human nature to instinctively fall back on life experiences in an attempt to bring experiential knowledge to an unfamiliar situation. When this happens, we unknowingly insert bias into decision making processes and understanding. When this happens, this must be recognized and corrected.

-DoD Enterprise DevSecOps Fundamentals March 2021

DevSecOps Goal

The goal of DevSecOps is to improve customer outcomes and mission value through the automation, monitoring, and application of security at every phase of the software lifecycle. *Figure 1 DevSecOps Phases and Continuous Feedback Loops* conveys the software lifecycle phases and continuous feedback loops.

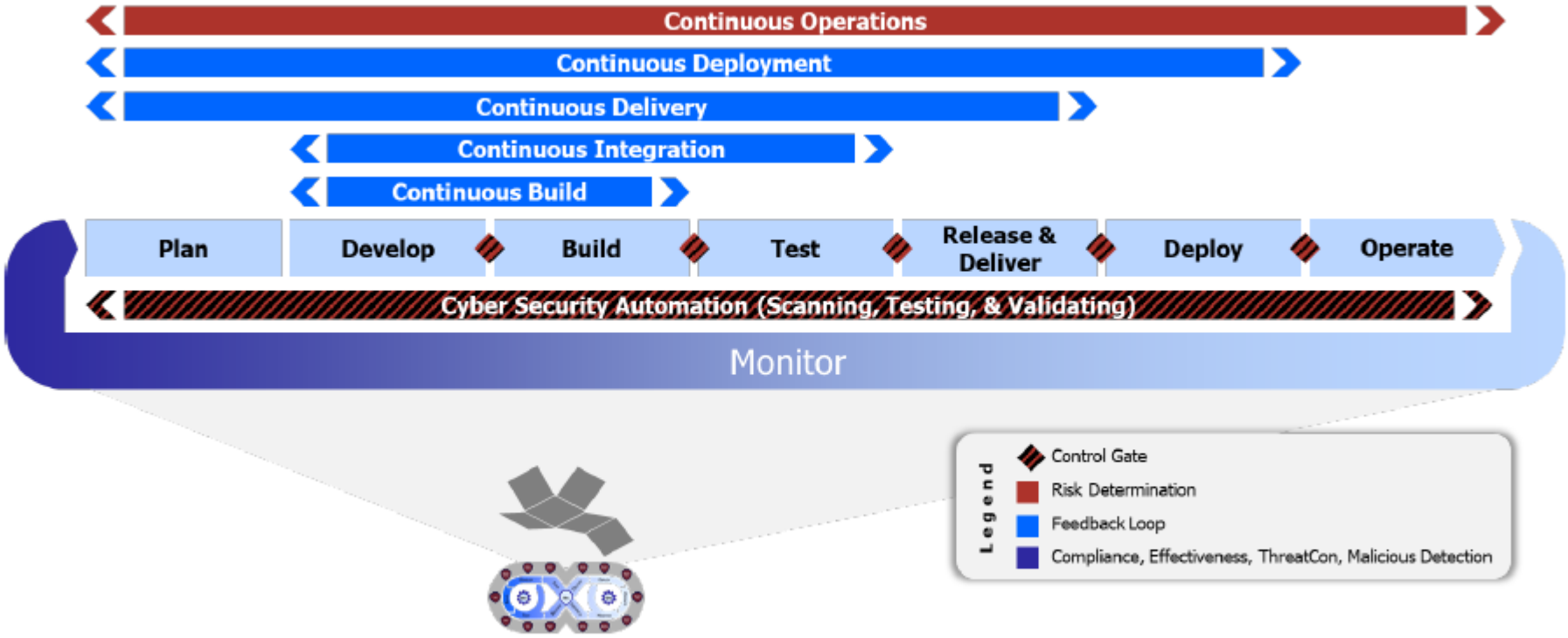


Figure 1 DevSecOps Phases and Continuous Feedback Loops

DevSecOps Pillars

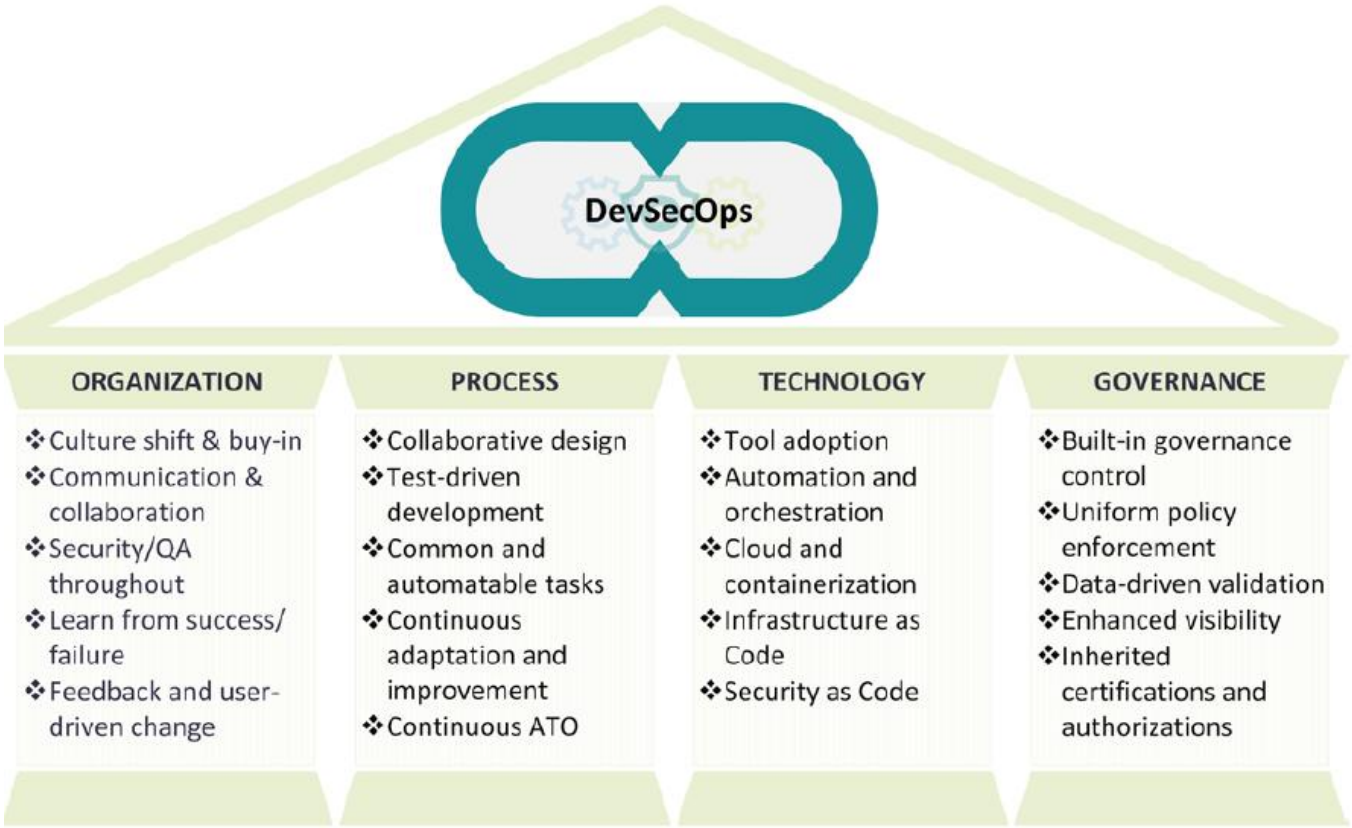


Figure 4: DevSecOps Pillars

DoD Enterprise DevSecOps Reference Design August 2019

DevSecOps Fundamentals Playbook

Shift Left

- Shift Left refers to the practice of moving tasks, responsibilities, or activities earlier (to the "left") in a process timeline.
- In traditional project timelines, time flows left to right — early stages like planning and design are on the left, while later stages like testing, deployment, and maintenance are on the right.
- By "shifting left", teams aim to:
- Catch problems earlier.
- Improve quality.
- Reduce costs and delays.

Shift Left

Meaning of Shift Left

Start testing during design and development, not

Introduce security checks early in development (e.

Start thinking about performance requirements at

Build and test deployment environments early in t

Code).

What DevSecOps Means

Traditional security management	New way
Security outsourced.	Security team is part of the team.
Security operation happens after each software release.	Security operation is integrated into each stage.
Manual and tool-based security scanning.	Least manual intervention. Scanning and data segregation are automated.
The development team only knows about the security issues after the release, and the security team provides feedback.	Each member of the software development and IT operation must be aware of the security aspect.

***Table 1.1**: Security operations, traditional versus DevSecOps*

Kumar Rath, Ashwini. Concepts and Practices of DevSecOps: Crack the DevSecOps interviews (English Edition) (p. 4). BPB Publications.

Shift Left

Shift Operational Test and Evaluation (OT&E) Left into the Pipeline
The Defense Innovation Board succinctly summed the goal of this play like this: “Speed and cycle time are the most important metrics for managing software. DoD must be able to deploy software faster without sacrificing its abilities to test and validate software

DevSecOps Fundamentals Playbook March 2021 Version 2.0

Shift Left Security

Shift-Left Security embodies the philosophy of moving security practices and considerations as early as possible in the development process—shifting them to the left on the development timeline. The traditional model often compartmentalizes security as a separate phase, typically occurring post-development. In contrast, DevSecOps integrates security seamlessly into every stage of the development lifecycle, with a focus on the earliest phases, such as planning and coding.

IFEROUDJENE, IZEM. DevSecOps: A complete implementation guide (Cyber Security Book 3)

Key Components of Shift-Left Security

Threat Modeling and Risk Assessment: Identifying potential threats and assessing risks at the planning and design phases. Automated Security Testing: Implementing automated tools for static analysis, dynamic analysis, and other security testing throughout the development process. Security Training and Awareness: Ensuring that development teams are equipped with the necessary security knowledge and best practices. Secure Coding Guidelines: Establishing and adhering to secure coding practices to prevent common vulnerabilities. Continuous Monitoring: Implementing continuous monitoring processes to detect and respond to security threats in real-time.

IFEROUDJENE, IZEM. DevSecOps: A complete implementation guide (Cyber Security Book 3)

Principles of Software Delivery

Continuous Integration usually refers to integrating, building, and testing code within the development environment. Continuous Delivery builds on this, dealing with the final stages required for production deployment.

Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment. In order to do Continuous Deployment, you must be doing Continuous Delivery.

Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

Continuous Delivery

There are several key principles for a successful transition to a DevSecOps culture:

- Continuous delivery of small incremental changes.
- Bolt-on security is weaker than security baked into the fabric of the software artifact.
- Value open source software.
- Engage users early and often.
- Prefer user centered & Warfighter focus and design.
- Value automating repeated manual processes to the maximum extent possible.
- Fail fast, learn fast, but don't fail twice for the same reason.
- Fail responsibly; fail forward.
- Treat every API as a first-class citizen.
- Good code always has documentation as close to the code as possible.
- Recognize the strategic value of data; ensure its potential is not unintentionally compromised.

Principles of Software Delivery

The aim of the deployment pipeline is threefold.

First, it makes every part of the process of building, deploying, testing, and releasing software visible to everybody involved, aiding collaboration.

Second, it improves feedback so that problems are identified, and so resolved, as early in the process as possible.

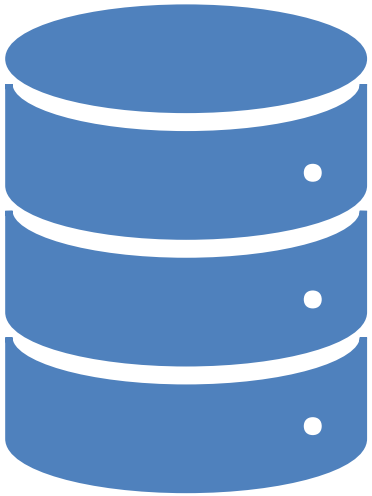
Finally, it enables teams to deploy and release any version of their software to any environment at will through a fully automated process.

Principles of Software Delivery

Frequent. If releases are frequent, the delta between releases will be small. This significantly reduces the risk associated with releasing and makes it much easier to roll back. Frequent releases also lead to faster feedback.

Feedback is essential to frequent, automated releases. There are three criteria for feedback to be useful.

1. Any change needs to trigger the feedback process.
2. The feedback must be delivered as soon as possible.
3. The delivery team must receive feedback and then act on it.



Principles of Software Delivery

Every Change Should Trigger the Feedback Process

A working software application can be usefully decomposed into four components: executable code, configuration, host environment, and data.

If any of them changes, it can lead to a change in the behavior of the application. Therefore, we need to keep all four of these components under control and ensure that a change in any one of them is verified.

The Feedback Must Be Received as Soon as Possible

The key to fast feedback is automation.

If you have manual processes, you are dependent on people to get the job done. People take longer, they introduce errors, and they are not auditable. Moreover, performing manual build, test, and deployment processes is boring and repetitive.

Principles of Software Delivery

The Delivery Team Must Receive Feedback and Then Act on It

It is essential that everybody (developers, testers, operations staff, database administrators, infrastructure specialists, and managers) involved in the process of delivering software is involved in the feedback process.

If people in these roles do not work together on a day-to-day basis, it is essential that they meet frequently and work to improve the process of delivering software.

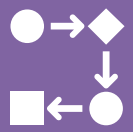
A process based on continuous improvement is essential to the rapid delivery of quality software.

Iterative processes help this kind of activity—at least once per iteration a retrospective meeting is held where everybody discusses how to improve the delivery process for the next iteration.

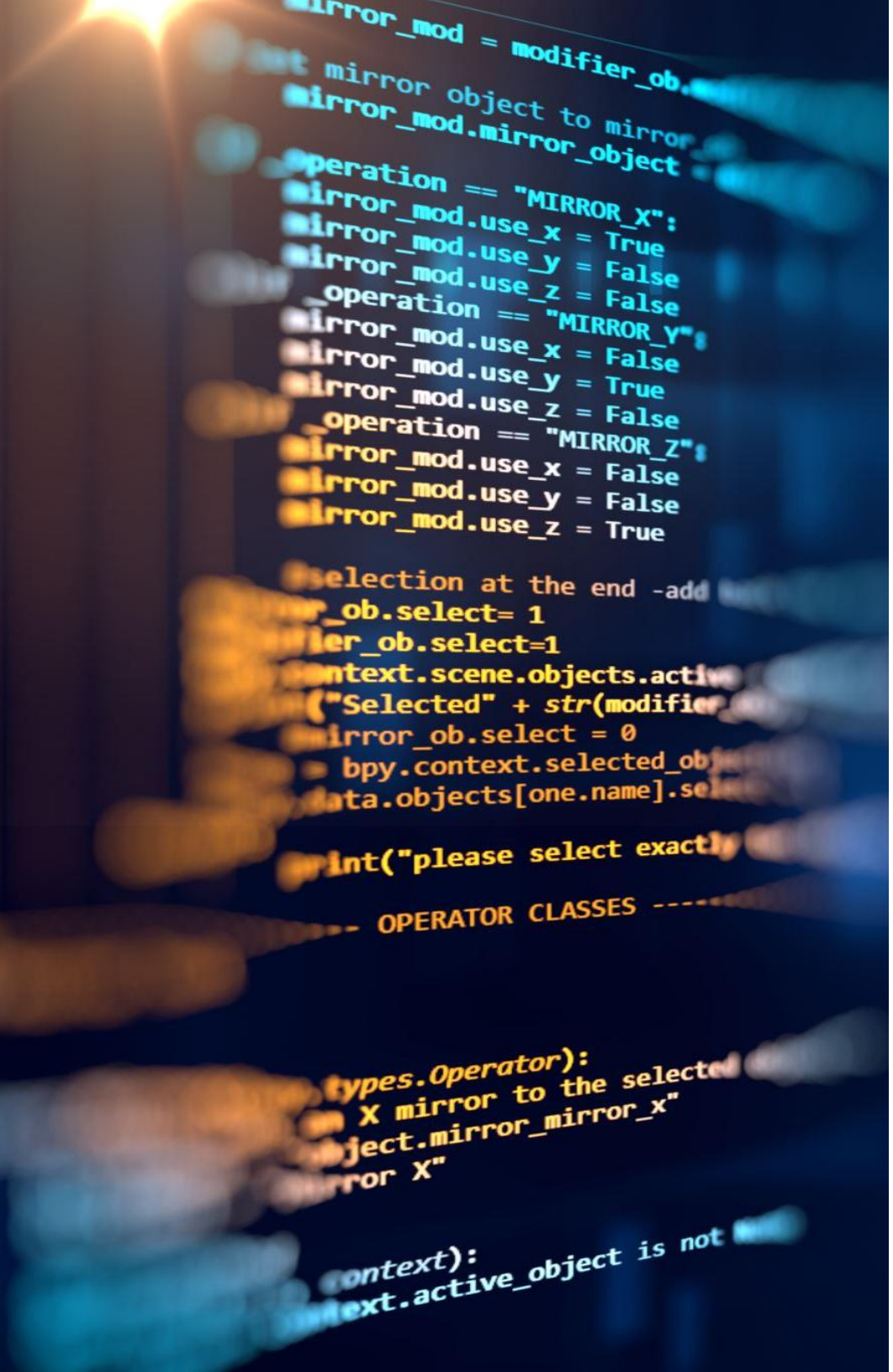
Principles of Software Delivery

Being able to react to feedback also means broadcasting information. Using big, visible dashboards (which need not be electronic) and other notification mechanisms is central to ensuring that feedback is fed-back and makes the final step into someone's head.

Finally, feedback is no good unless it is acted upon. This requires discipline and planning.



When something needs doing, it is the responsibility of the whole team to stop what they are doing and decide on a course of action. Only once this is done should the team carry on with their work.



Principles of Software Delivery

Create a Repeatable, Reliable Process for Releasing Software:-

Releasing software should be easy. It should be as simple as pressing a button.

It should be easy because you have tested every single part of the release process hundreds of times before.

The repeatability and reliability derive from two principles: automate almost everything, and keep everything you need to build, deploy, test, and release your application in version control.

Principles of Software Delivery

Deploying	Deploying software ultimately involves three things:
Provisioning and managing	Provisioning and managing the environment in which your application will run (hardware configuration, software, infrastructure, and external services).
Installing	Installing the correct version of your application into it.
Configuring	Configuring your application, including any data or state it requires.

Principles of Software Delivery

Automate what you can

Most development teams don't automate their release process because it seems such a daunting task.

It's easier just to do things manually. Perhaps that is true the first time they perform a step in the process, but it is certainly not true by the time they perform that step for the tenth time.

Automation is a prerequisite for the deployment pipeline, because it is only through automation that we can guarantee that people will get what they need at the push of a button.

You don't need to automate everything at once. You can, and should, automate gradually over time.

There are some things it is impossible to automate.(eg:- Demonstrations of working software to representatives of your user community cannot be performed by computers.) However, the list of things that cannot be automated is much smaller than many people think.

Principles of Software Delivery

Keep Everything in Version Control

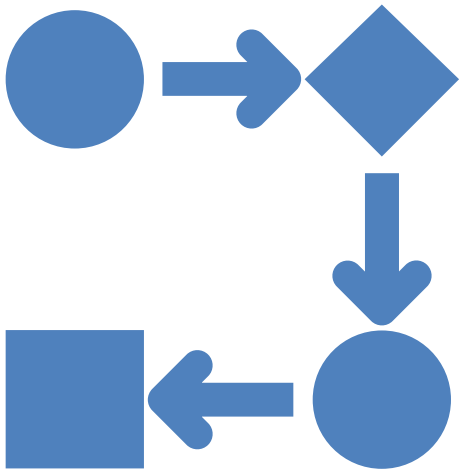
Everything you need to build, deploy, test, and release your application should be kept in some form of versioned storage.

This includes requirement documents, test scripts, automated test cases, network configuration scripts, deployment scripts, database creation, upgrade, downgrade, and initialization scripts, application stack configuration scripts, libraries, toolchains, technical documentation, and so on.

All of this stuff should be version-controlled, and the relevant version should be identifiable for any given build.

That is, these *change sets* should have a single identifier, such as a build number or a version control change set number, that references every piece.

Principles of Software Delivery



Continuous Improvement

It is worth emphasizing that the first release of an application is just the first stage in its life.

All applications evolve, and more releases will follow. It is important that your delivery process also evolves with it.

The whole team should regularly gather together and hold a retrospective on the delivery process.

This means that the team should reflect on what has gone well and what has gone badly, and discuss ideas on how to improve things.

Somebody should be nominated to own each idea and ensure that it is acted upon. Then, the next time that the team gathers, they should report back on what happened.

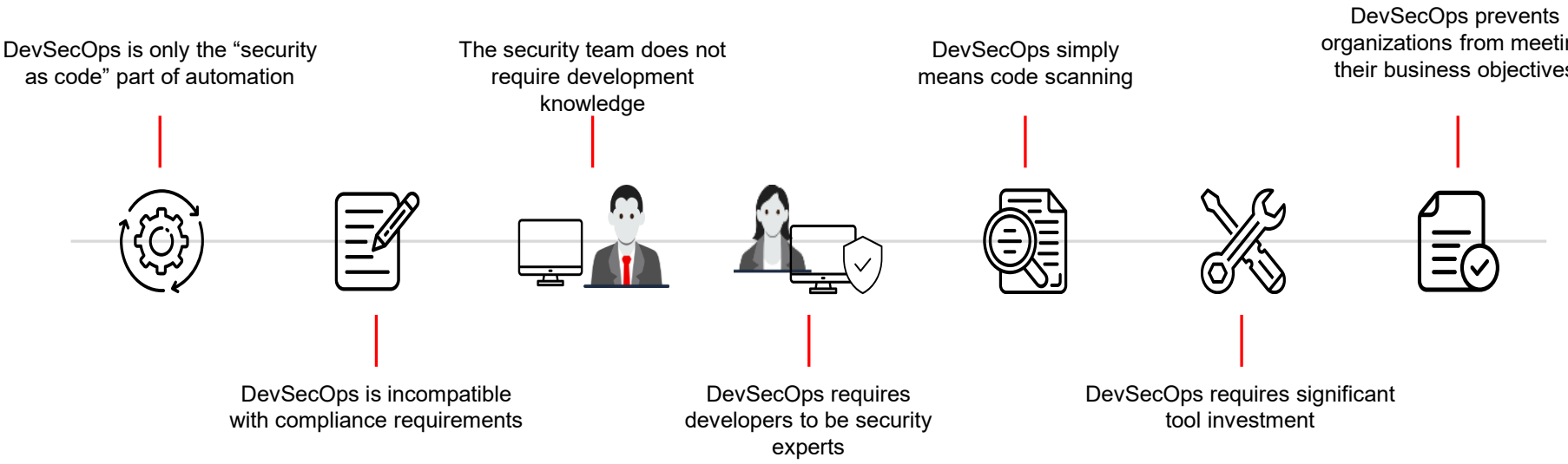
This is known as the *Deming cycle*: plan, do, study, act.

DevSecOps State of Mind

DevSecOps is a mindset, rather than just a collection of rules and tools. To the organizations that have been practicing DevOps, it is different way of thinking about security.

- **Distinguished Features of DevSecOps**
- Open collaboration on shared objectives
- Security at the source
- Automation of repetitive tasks
- Risk-oriented operations and actionable insights
- Holistic approach to security objectives
- Proactive monitoring and recursive feedback
- Operations engineering

Common Misconceptions and Misinterpretations Regarding DevSecOps



Model Definitions

Model:
A representation of the most essential features of a physical object or process used as a pattern for reasoning about, analyzing, or predicting behavior

Abstract Model:
A model with no concrete instances, e.g., a meta-model, or model of a model, used to reason about models.

Software Development Model:
A model of the phases, activities, products, and roles of people involved with the development of software.

Software Computation Model (Component Model):
A model of the structure, behavior, and dependencies of a software component.

Software Computational Models

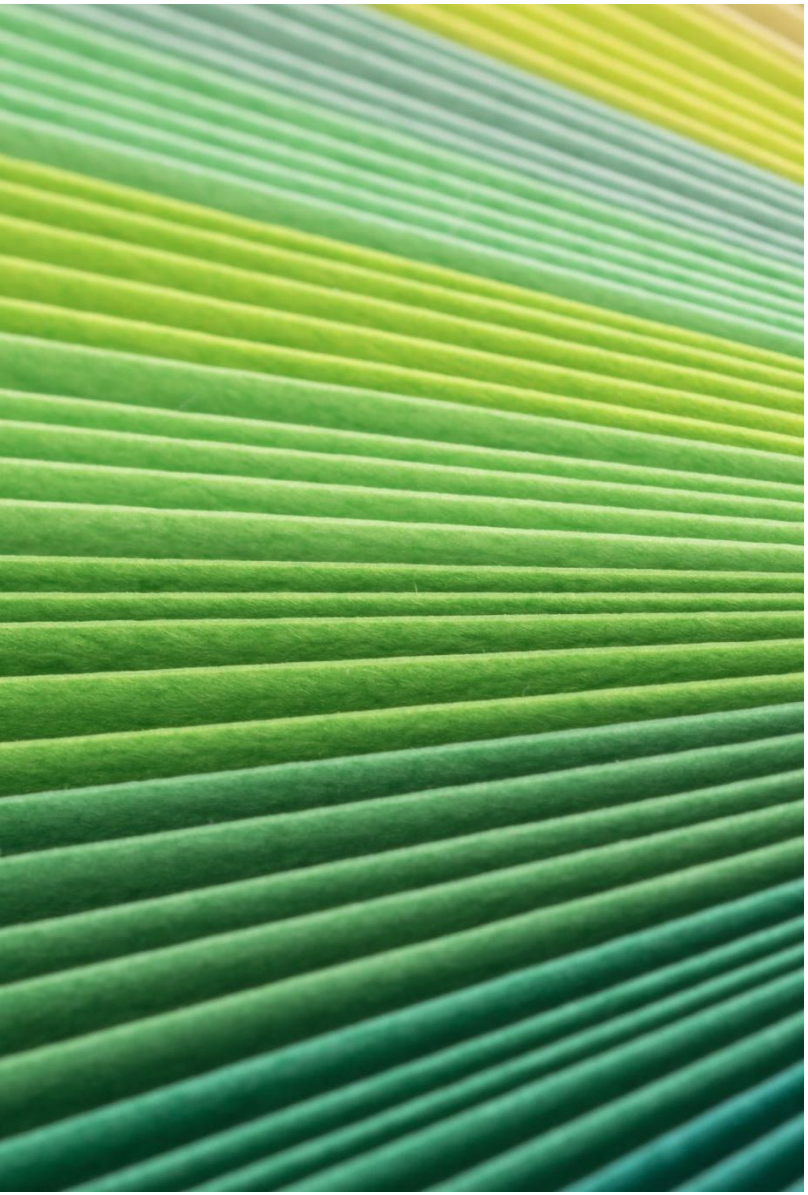
Computational models describe structure, behavior, and dependencies of software components. They enable us to reason about and predict the behavior of software systems and their components.

Without computational models, large-scale software development would be virtually impossible. We need the abstractions provided by computational models to understand how these large systems are supposed to work.

We use these models when we write requirements specifications and conduct design reviews.

Waterfall Model

- **Requirements** – defines needed information, function, behavior, performance and interfaces.
- **Design** – data structures, software architecture, interface representations, algorithmic details.
- **Implementation** – source code, database, user documentation, testing.



Waterfall Strengths

- Easy to understand, easy to use
- Provides structure to inexperienced staff
- Milestones are well understood
- Sets requirements stability
- Good for management control (plan, staff, track)
- Works well when quality is more important than cost or schedule

Waterfall Deficiencies

- All requirements must be known upfront
- Deliverables created for each phase are considered frozen – inhibits flexibility
- Can give a false impression of progress
- Does not reflect problem-solving nature of software development – iterations of phases
- Integration is one big bang at the end
- Little opportunity for customer to preview the system (until it may be too late)

Agile Process Models

Agile software engineering combines a philosophy and a set of development guidelines

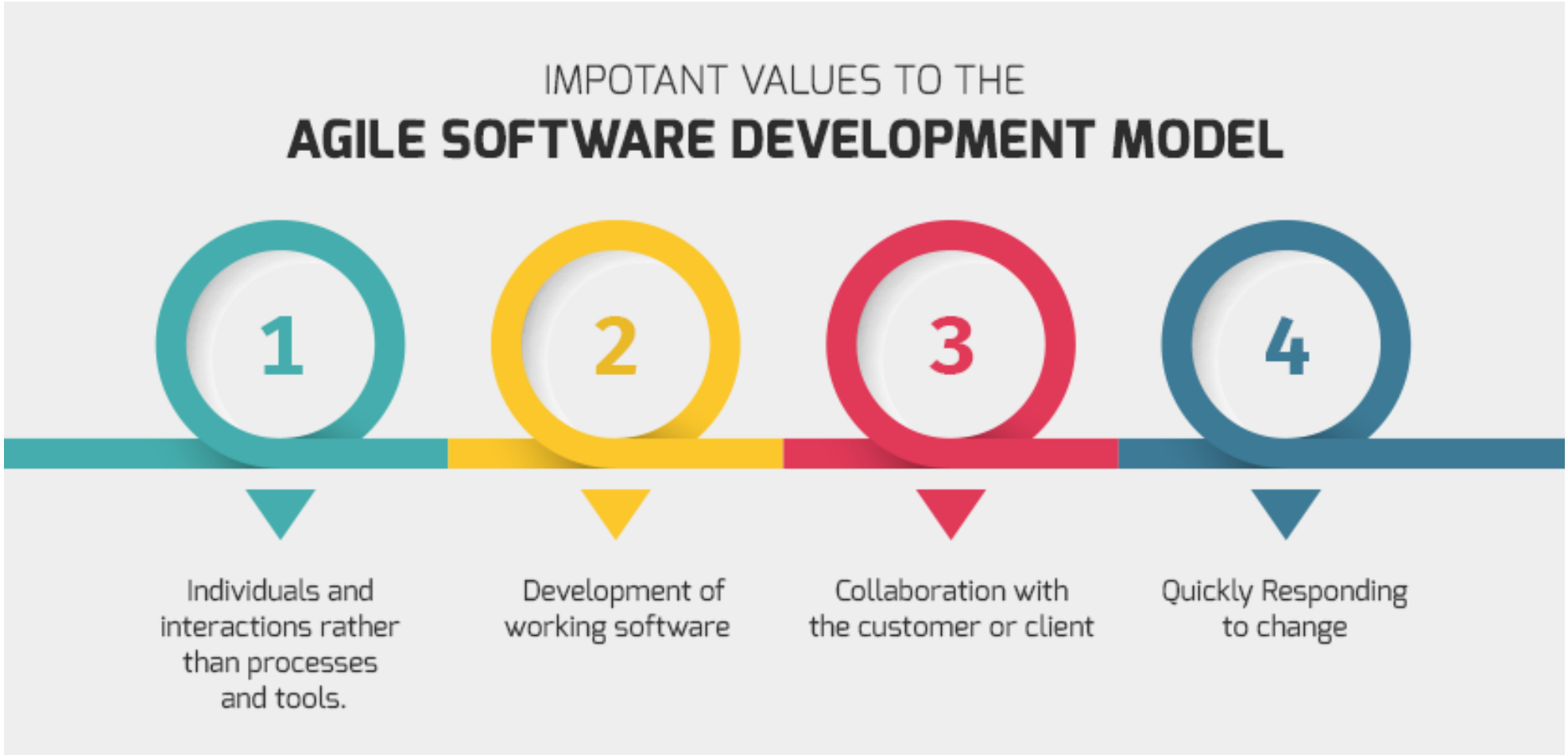
Philosophy

- Encourages customer satisfaction and early incremental delivery of the software
- Small highly motivated project teams
- Informal methods
- Minimal software engineering work products
- Overall development simplicity

Development guidelines

- Stress delivery over analysis and design
- Active and continuous communication between developers and customers

Agile Process Models



Agile Manifesto

1.Customer satisfaction by early and continuous delivery of valuable software.

2.Welcome changing requirements, even in late development.

3.Deliver working software frequently (weeks rather than months).

4.Close, daily cooperation between business people and developers.

5.Projects are built around motivated individuals, who should be trusted.

6.Face-to-face conversation is the best form of communication (co-location).

7.Working software is the primary measure of progress.

8.Sustainable development, able to maintain a constant pace.

9.Continuous attention to technical excellence and good design.

10.Simplicity—the art of maximizing the amount of work not done—is essential.

11.Best architectures, requirements, and designs emerge from self-organizing teams.

12.Regularly, the team reflects on how to become more effective, and adjusts accordingly.

Standards for Agile

There are standards specifically for Agile:

1. ISO/IEC/IEEE International Standard 26515 - Systems and software engineering - Developing information for users in an Agile environment
2. Integrating Software Testing Standard ISO/IEC/IEEE 29119 to Agile Development

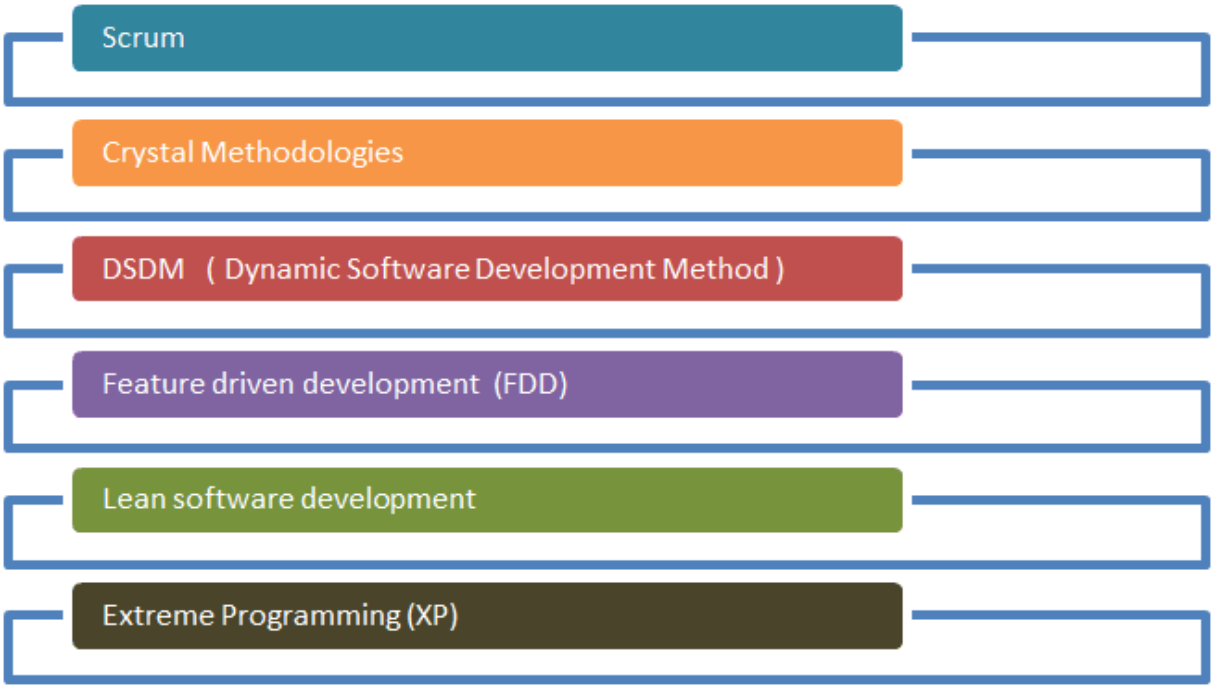
a. "Information developers may contribute to the design and test of the software as well as to **project artifacts and life cycle documents, such as user stories, use cases,** and personas." (Emphasis added). There have been no user stories or use cases produced in this matter.

b. "The user requirements should be the basis for the creation of design documents such as personas, user stories, or scenarios". There have been no actual design documents produced in this matter.

c. "The Agile development team shall use a controlled source location for published designs and high-level design documents. Comments and discussions around the design should be included with the published designs." Again, there have been no actual design documents produced in this current matter. Furthermore, the design documents and discussions about the design should be captured in a repository. This did not occur.

d. "Design documents such as user stories and scenarios can help the Agile development team understand the value of these tasks for the users and provide a mechanism for developing testing to validate the tasks as well as provide guidance on how to complete the tasks." Again, there have been no design documents produced in this case. Furthermore, user stories which are fundamental to Agile are not even mentioned anywhere.

Agile Process Models



The Agile Subway Map

The colored "subway" lines represent practices from the various Agile approaches or areas of concern.
Click on the "station" circles to read the full definition for each term.

- Extreme Programming
- Teams
- Lean
- Scrum
- Product Management
- DevOps
- Design
- Testing
- Fundamentals

Principles of agile methods

Principle	Description
Customer involvement	The customer should be closely involved throughout the development process. Their role is provide and prioritise new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognised and exploited. The team should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and design the system so that it can accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process used. Wherever possible, actively work to eliminate complexity from the system.

Waterfall vs. Agile vs. DevOps Methodology

Area of Differentiation	Waterfall	Agile	DevOps
Approach	• Fully predictable systems • Systems can be specified in advance • Assumes that the business needs remain constant throughout a project • Limited to software development	• Rapid delivery of software through the integration of business, development, and quality assurance • Repeated "sprint" cycles • Assumes that the priority of business needs may change • Limited to software development	• Implements automation by cross-functional teams to deploy changes continuously • Provides continuous feedback throughout a loop • Limited to development and operations
Goal	Gathering and clarifying all the requirements upfront	Continuous improvement and maturity in application development	Increased frequency and enhanced quality of application releases
Iteration	Sequential development process	Iterative development process	Incremental correction of problems and release of code as it becomes available
Teams Collaboration	Low because the teams work in functional silos	• Improved due to short development cycles and highly engaged businesses • Small teams of designers, developers, and testers	High because all stakeholders/multiple teams are involved in the project development from beginning
Quality	Low quality because issues are not identified until the testing phase	Improved because issues are identified for each sprint	High because of automated unit testing during the project development
Risk	Increases as the project development progresses	Decreases as the project development progresses	Decreases as the project development progresses
Business Ownership of Project	No	Yes	Yes
Delivery of Value/Scheduling	Slow (monthly)	Rapid (daily/weekly)	Continuous
Level of Communication and Documentation	Comprehensive	Light	Light
Level of Automation	Low	Varied	High
Response to New Business Needs	Limited because the specifications for a project are given in detail	Responsive; prioritization is enabled by repeated delivery	Highly responsive; business requirements are defined precisely by teams
Customer Feedback	Infrequent at project completion	Frequent at project completion	Continuous

Scrum Meetings

There are different types of scrum meetings:

•Daily standup meetings: This is a very short meeting that is generally no longer than 15-20 minutes. In this meeting, all the product owners, architects, and project managers meet to check the status of the sprints.

•Sprint planning meetings: In this meeting, everyone comes together to decide the duration of a sprint and the number of sprints needed to complete the task. Sprints are generally no longer than 30 days.

•Sprint review meetings: These are meetings where a review is done once sprints end. These meetings showcase what has been done around the product.

•Retrospective meetings: These meetings are for checking what has been done right and what has gone wrong. Checking the backlog meetings: In this meeting, the product backlog is tracked and checked to see how soon the product backlog can be worked upon.

-Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST (p. 48). Packt Publishing. Kindle Edition.

Problems with agile methods

It can be difficult to keep the interest of customers who are involved in the process.

Team members may be unsuited to the intense involvement that characterizes agile methods.

Prioritizing changes can be difficult where there are multiple stakeholders.

Maintaining simplicity requires extra work.

Contracts may be a problem as with other approaches to iterative development.

Problems With Agile

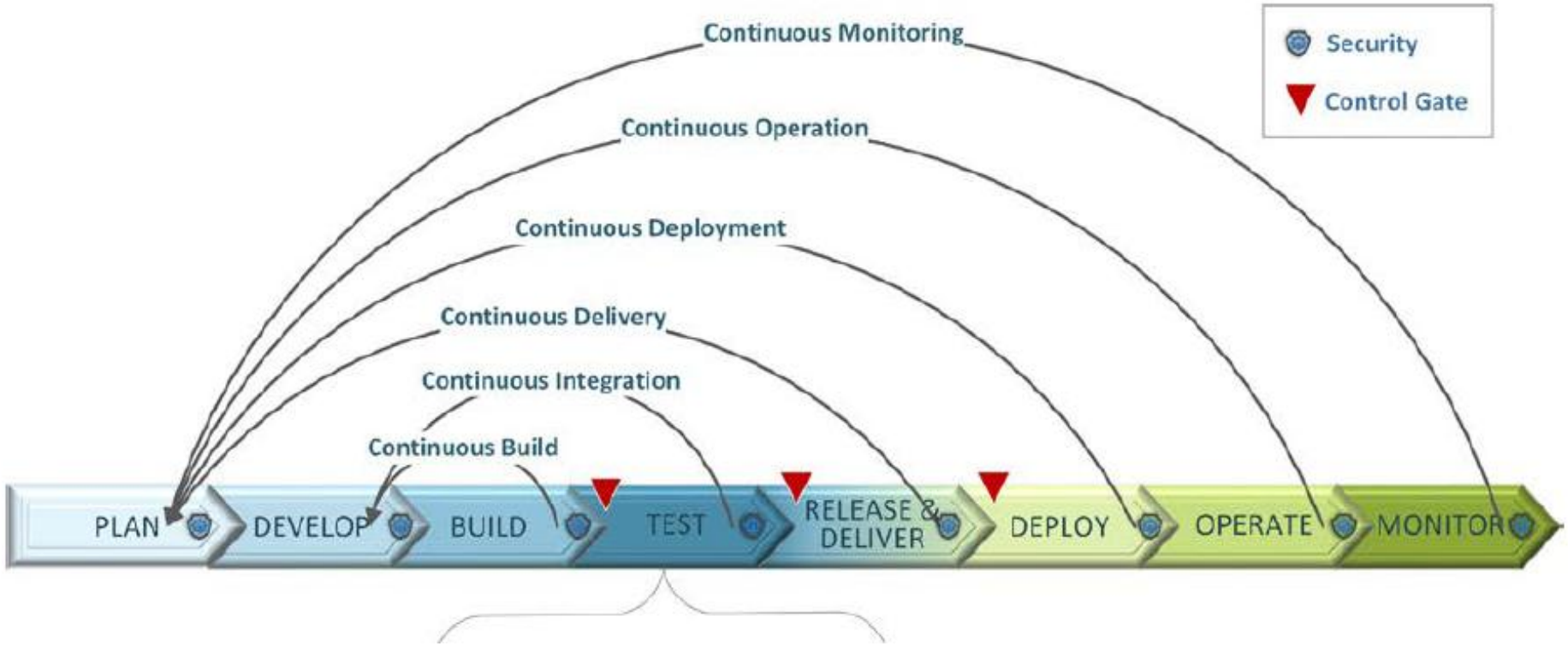
- Study consisting of 600 UK and US software engineers finds projects adopting Agile Manifesto practices are 268% more likely to fail than those which do the opposite.
- https://www.theregister.com/2024/06/05/agile_failure_rates/

- https://drj.com/industry_news/268-higher-failure-rates-for-agile-software-projects-study-finds/

DevOps

Software Engineering Institute—suggested defining DevOps as "a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality"

- IEEE 2675

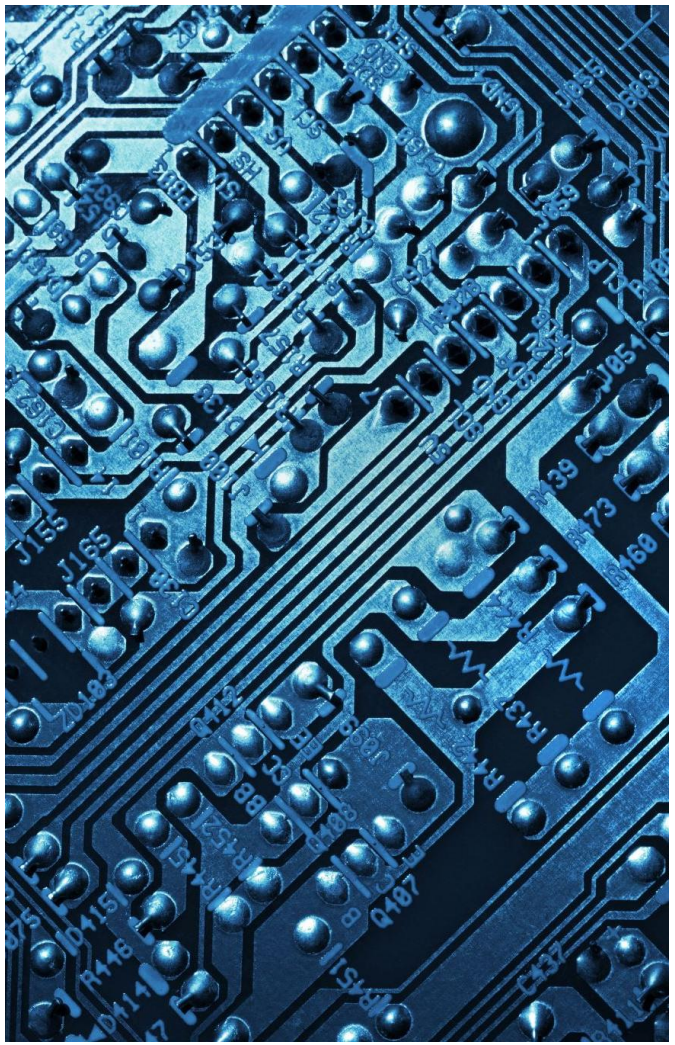


DoD Enterprise DevSecOps Reference Design August 2019

DevSecOps

NIST Special Publications

- SP 800-12: An Introduction to computer Security. A handbook. Chapter 4 is common threats
- SP 800-14: Generally Accepted Principles and Practices for Securing Information Technology Systems
- SP 800-18: Guide for Developing Security Plans for Federal Information Systems
- SP 800-27:Engineering Principles for Information Technology Security (A Baseline for Achieving Security)
- SP 800-30:Risk Management Guide for Information Technology Systems (this has been superseded)
- SP 800-61: Computer Security Incident Handling Guide
- SP 800-64:Security Considerations in the System Development Life Cycle
- SP 800-100: Information Security Handbook: A Guide for Managers



The Shift from DevOps to DevSecOps

DevSecOps is an extension of DevOps. DevSecOps was introduced to increase the speed of DevOps. By integrating security into DevOps processes, operations teams were motivated and measured to stabilize production to meet service-level agreements (SLAs). It was about making new changes, but they needed to be made quickly. This made it look like a lot of things were being left behind.

-Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST (p. 53). Packt Publishing. Kindle Edition.

Addressing DevOps Security Challenges

- **Strengthen the DevOps Security by Implementing the Following:**
  - Leverage Automation
  - Manage vulnerabilities effectively and efficiently
  - Enforce robust DevOps secrets management solutions such as Azure Key Vault, AWS Secrets Manager, and Cyberark
  - Implement effective privileged-access management solutions to minimize the risks from internal and external threats
  - Centralized auditing and logging system
  - Design secure infrastructure
  - Perform continuous assessment to identify and remediate security issues
  - Integrate the security tools in the DevOps pipeline
  - Use configuration management (IaC) for implementing security as code

Metric	Description
Deployment frequency	It tracks the frequency of deployments to production in a specific time frame
Change lead time (for application)	It measures the time between a code commit and production
Change volume (for applications)	It measures the number of user stories that are deployed in a given period
Change failure rate	It is the percentage of failed production deployments
MTTR (for applications)	Mean Time To Repair/ Mean Time to Recover
Availability	It measures the uptime or downtime in a particular time frame as per the service-level agreement
Customer issue volume	It measures the number of issues reported by customers in a given period
Customer issue resolution time	It is the time taken for resolving an issue reported by a customer
Time to value	It measures the time between a request for a feature and the recognition of business value from that feature
Time to ATO	It is the time from the start of Sprint 0 to achieving an Authorization to Operate
Time to patch vulnerabilities	It is the time taken from the detection of a vulnerability in the application or platform to the successful deployment of a patch in production

In DevSecOps, it is not sufficient to measure software development and delivery performance on tempo and stability metrics, as they cannot provide sufficient security and can result in the accumulation of security issues

To measure success in the implementation of DevSecOps, the U.S. General Service Administrator (GSA) has introduced the following high-value DevSecOps metrics

Metrics for Measuring DevSecOps Success

DevSecOps Culture: Technology

- Technology enables teams to effectively implement development, security, and operation processes in the DevSecOps pipeline
- **The Following Technologies can be Used by an Organization for Effectively Implementing DevSecOps Culture:**
  - **Source code repository:** It is the main technology in which other technologies used in the DevSecOps pipeline can be integrated. It can be used to store project files, web pages, and patches and can be accessed either publicly or privately
  - **Configuration management:** Automating configuration management and integrating it in the development cycle ensures **proper tracking and easy reporting of changes**
  - **Orchestration:** There are various solutions for orchestrating the deployment of infrastructure. Container orchestration such as Kubernetes can be used to implement and manage repeatable security best practices
  - **Host hardening:** Implementing host hardening in DevSecOps automation reduces attacks on applications as it removes unnecessary applications, disables irrelevant accounts on the system, etc
  - **Securing docker images:** It is important to follow best practices to **secure docker images** such as avoiding unnecessary user privileges, signing images digitally, and using the docker secret feature
  - **CI/CD for patching:** The metadata linked with each asset can be used to **implement patches for security vulnerabilities** at the CI/CD level

DevSecOps Culture: Technology

- **Secure coding standards:** Coding standards continually change, and code should be checked with updated security recommendations. Code changes must be validated and tested against new recommendations
- **Application-level assessment:** In DevSecOps, it is important to assess the application-level security to determine the risk posture and to mitigate vulnerabilities before an attack occurs. The following can be used to improve the security posture:
  - **Source Code Scanning** can be performed using static application security testing (SAST) tools to identify security vulnerabilities in the code
  - **Dynamic Application Security Testing (DAST)** scans a website at runtime and analyzes inputs, forms to identify vulnerabilities
  - **Integration of SAST with IDE** allows developers to receive a notification when insecure coding practices are followed. This helps mitigate vulnerabilities quickly
  - **Binary Scanning** must be performed to find security issues from a coding checklist, which is then signed digitally to ensure security
  - **Pre-deployment Auditing:** To ensure security in the DevSecOps pipeline, use a **predefined template** to deploy resources
  - **Post-deployment Auditing:** After instantiating pre-defined templates, verify them to check for any difference from the **pre-deployment scans** that may cause security threats
  - **Automated Vulnerability Management** should be integrated into the infrastructure scanning platforms to track all the identified vulnerabilities
    - **Automated compliance scan** or automated security configuration assessments can be used to reduce the compliance cost. It also allows sharing of the compliance information with the GRC tools
- **Secrets Management:** Secrets can be the source of a breach; therefore, they should be secured. With the use of an **auto-generated secrets store,** the entities for temporary usage should have less time-to-live (TTL)

DevSecOps Best Practices

The Best Practices for DevSecOps Include the Following:

- **Train Developers on Secure Coding:** Train development teams on secure coding practices as they may not be aware of insecure coding
- **Automation:** Automation plays a key role in the success of DevSecOps. Auditing becomes easier with automation using metadata
- **Proper Tool Selection:** The tools used in the DevSecOps pipeline should make the software delivery process fast and produce accurate and actionable results
- **Threat Modeling:** It helps understand the existing and emerging threats to resources as well as the security controls required to safeguard the resources
- **Compliance Implementation:** Create metadata to describe the compliance requirements, which can be integrated into the assets
- **Incident Management:** To respond to incidents in a measurable and consistent manner, action plans and runbooks should be created in advance
- **Secrets Management:** Secrets such as credentials and keys should be generated for temporary use, and the authentication mechanism should be different for each environment
- **Pre-Deployment and Post-Deployment Auditing:** To ensure that the required security level is achieved, pre-deployment auditing should use pre-defined templates for creating resources. Post-deployment auditing ensures that the security level gained through pre-deployment auditing is valid

Continuous Security in DevSecOps

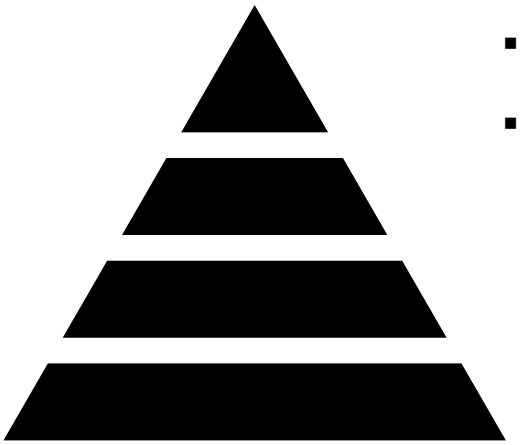
- In continuous security, security is an integral part of the CI/CD pipeline to develop secure applications
- Continuous security works along with other continuous development processes and makes the product and infrastructure more secure
- Continuous security should be a core implementation for the DevOps pipeline as more frequent application changes and releases occur
- It helps in mitigating most of the security issues before the product release
- It can be implemented into the existing DevOps pipeline by making changes while keeping application security in mind
- Continuous security facilitates the proper management of security alerts for security incidents by setting metrics on applications and infrastructure
- With continuous security, potential attack vectors can be identified and fixed by automating the remediation process

Continuous Security in DevSecOps

Implementing continuous security provides the confidentiality, integrity, and availability (CIA) traits for the pipeline

- Three elements are important in establishing continuous security in the pipeline:
  - **Culture:** Security should be the central focus of an organization's culture
  - **Infrastructure:** Infrastructure should be implemented with the recommended security best practices
  - **Software Delivery:** The process of software delivery should focus on security
- Three principles of continuous security that should be considered:
  - **Shift Left:** By implementing the shift-left principle, bugs can be fixed at an early stage of SDLC, which saves time and cost in the SDLC
  - **Automation:** The chance of errors in repetitive tasks can be reduced by implementing automation. It can also reduce time and improve productivity
  - **Continuous Improvement:** The continuous improvement of security is required for better overall security. It can be achieved by analyzing the current behaviors and searching for possibilities of improvement

Continuous Testing for CI/CD Pipeline Security



Testing Pyramid

- **Testing automation** follows a model called a testing pyramid
- It provides an overview of the complete testing process starting from small unit testing to complete connected processes
  - **Level 1:** It is the base of the pyramid and involves unit testing, which uses the **white-box method**. Here, unit tests are performed on small pieces or individual units of code in the software
  - **Level 2:** In integration testing, code units are tested in terms of how they interact with each other, which involves a chain of components
  - **Level 3: Black-box testing** is used in end-to-end testing. The scope of testing depends on the process. It is performed to test the workflow from the user's perspective

Continuous testing can be successfully adopted into the CI/CD pipeline by performing automated testing

Continuous Testing for CI/CD Pipeline Security

- The following are testing terms other than those mentioned in the pyramid:
- **Functional Testing:** The function of the application is tested to determine whether it **satisfies the specified requirement** from the end-user perspective
- **Regression Testing:** Ensures that **old features are working as intended** even after adding new features and frequent updates. Regression testing increases as changes to an application increase
- **Load Testing:** Tests the behavior of the software under high input or stressful conditions
- **Performance Testing:** Tests the individual parts of application performance in terms of speed and effectiveness
- **API Testing:** APIs are exposed to third-party developers and should be tested in terms of expectations for functionality, performance, reliability, and security. This is typically performed as black-box testing
- **User Interface (UI) Testing:** UI testing is performed using automated tools that mimic the interactions performed by the users

DevSecOps Maturity Model

Domain	Description
Culture & Collaboration	Security collaboration between developers, ops, and security teams.
Governance & Compliance	Policies, standards, and regulatory alignment.
Security Testing & Automation	Integration of security checks in CI/CD pipelines.
Secure Coding Practices	Adoption of secure coding standards and peer reviews.
Vulnerability Management	Identification and remediation of vulnerabilities across the lifecycle.
Monitoring & Logging	Security visibility and observability in production.
Infrastructure & Configuration Security	Use of IaC, secure configurations, and cloud security controls.

DevSecOps Maturity Model

Level	Description
0 - Absent	No DevSecOps practices in place. Security is reactive.
1 - Initial	Ad hoc security checks, basic tooling, no formal integration.
2 - Defined	Security policies and tooling are established but manual.
3 - Managed	Security is integrated into CI/CD with some automation.
4 - Optimized	Fully automated, scalable, and adaptive security posture. Security as code.

DevSecOps Maturity Model

Key Features at Higher Maturity Levels

Security-as-Code: All security policies and controls are codified and version-controlled.

Automated Scanning: Static (SAST), dynamic (DAST), and dependency scanning integrated in CI/CD pipelines.

Shift Left: Security testing starts early in development, including in IDEs and pre-commit hooks.

Threat Modeling & Risk Assessment: Continuous and automated threat models using tools and context-aware analysis.

Immutable Infrastructure: All deployments are automated, consistent, and auditable.

Continuous Monitoring: Real-time alerting, anomaly detection, and behavioral analysis

DevOps Cycle

It is famously illustrated with an infinity loop (*Figure 1.1*) broadly running over six stages:

- **Design:** You design new software, an improvement to existing software, or even a modification to the software to align with requirements.
- **Code:** The phase includes programming activities of coding, compiling (when needed) and testing the software units.
- **Integrate:** While different software developers work on different units, integration does the process of merging the changes into a codebase that will function as the designated software system.
- **Deploy:** After integration, the software system needs to be deployed at one or multiple server locations as per the deployment architecture.
- **Operate:** Users start using the system, they can be a selected user group (for doing a pilot) or the actual end users for whom the software is designed.
- **Monitor:** While the software is in use, its accuracy, usability, and performance must be monitored, and adequate feedback is gathered for the next cycle in the loop.

- Kumar Rath, Ashwini. Concepts and Practices of DevSecOps: Crack the DevSecOps interviews (English Edition) (p. 3). BPB Publications.

Embed Security into DevOps

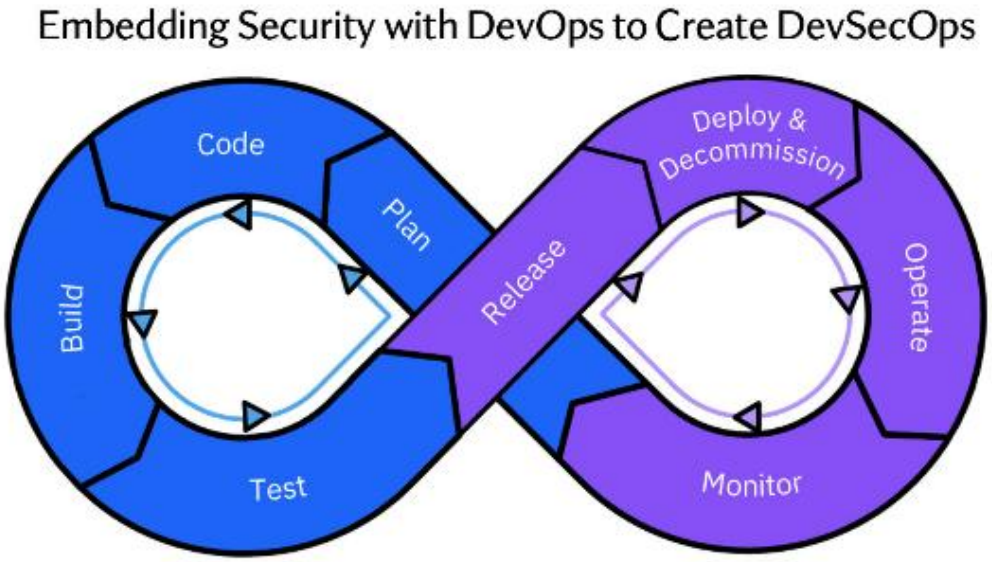


Figure 1.5: DevSecOps in action

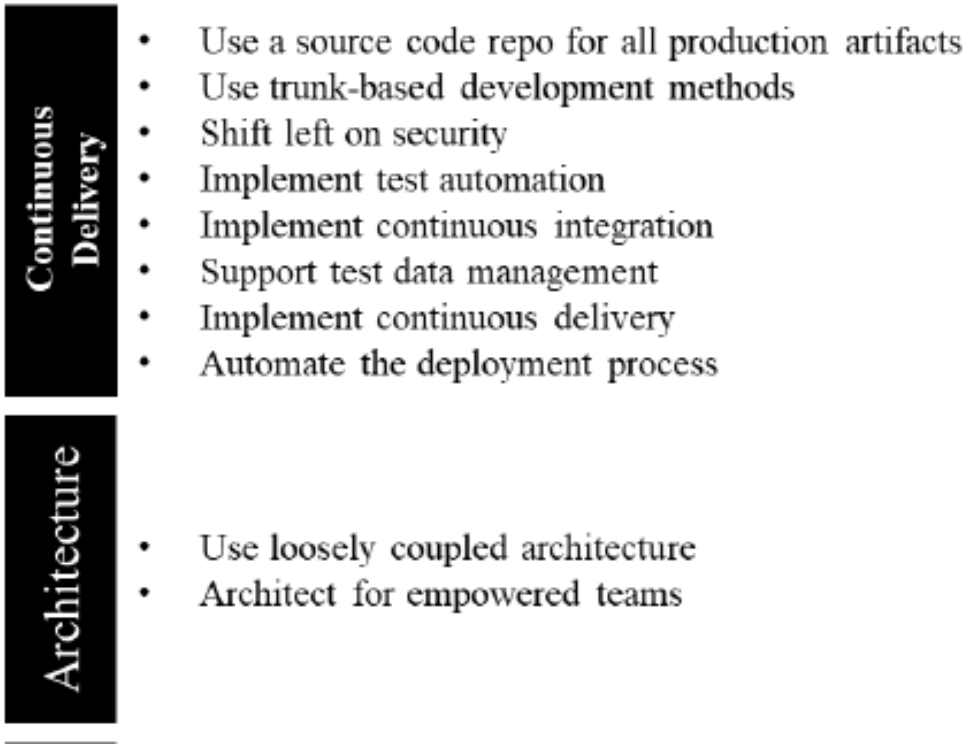
Sehgal, Vandana Verma. Implementing DevSecOps Practices: Understand application security testing and secure coding by integrating SAST and DAST. Packt Publishing.

DevOps Basics

- Interactions may be dependent or independent
  - Dependency – heating a pot of water raises the temperature
  - Independent – heating water never changes refrigerator temp
- DevSecOps practices are *interdependent*
  - Resilient with complex interactions
  - Non-linear, less predictable outcomes
  - Each element linked to outcomes

Drive Continuous Improvement through Key Capabilities

There are 24 key capabilities that drive improvements across both the DevSecOps team and its organization.[8] The capabilities are organized into five broad categories: Continuous Delivery, Architecture, Product and Process, Lean Management & Monitoring, and Cultural. Cultural change is often the hardest thing to address. The 24 key capabilities are:



- DoD DevSecOps Fundamentals Playbook March 2021

Drive Continuous Improvement through Key Capabilities

- DoD DevSecOps Fundamentals Playbook March 2021

Cultural

- Adopt a Likert scale survey to measure cultural change progress
- Encourage and support continuous learning initiatives
- Support and facilitate collaboration among and between teams
- Provide resources and tools that make work meaningful
- Support or embody transformational leadership

Product & Process

- Gather and implement customer feedback
- Make the flow of work visible through the value stream
- Work in small batches
- Foster and enable team experimentation

Lean Management & Monitoring

- Have a lightweight change approval process
- Monitor across application and infrastructure to inform business decisions
- Check system health proactively
- Improve processes and manage work with work-in-process (WIP) limits
- Visualize work to monitor quality and communicate throughout the team

Establish a Software Factory

- All custom software development should be driven through the software factory construct using DevSecOps. There are several ways to instantiate a DoD DevSecOps Software Factory / Platform. At this time, the option with the least friction is to use the DoD-approved DevSecOps Managed Service Provider (MSP), Platform One. Platform One is operated as an authorized-to-use Platform with integrated continuous authorization to operate (cATO) practices. It leverages several enterprise-class services, including Iron Bank as a recognized DoD hardened artifact repository and Repo One for source code management.
- Another option is to establish a software factory using a Cloud Service Provider (CSP) with a DoD ATO or Provisional Authorization (PA). Leverage the CSP's managed services, ideally through IaC practices, to establish a DevSecOps Software Factory.

- -DoD DevSecOps Fundamentals Playbook March 2021

Establish a Software Factory

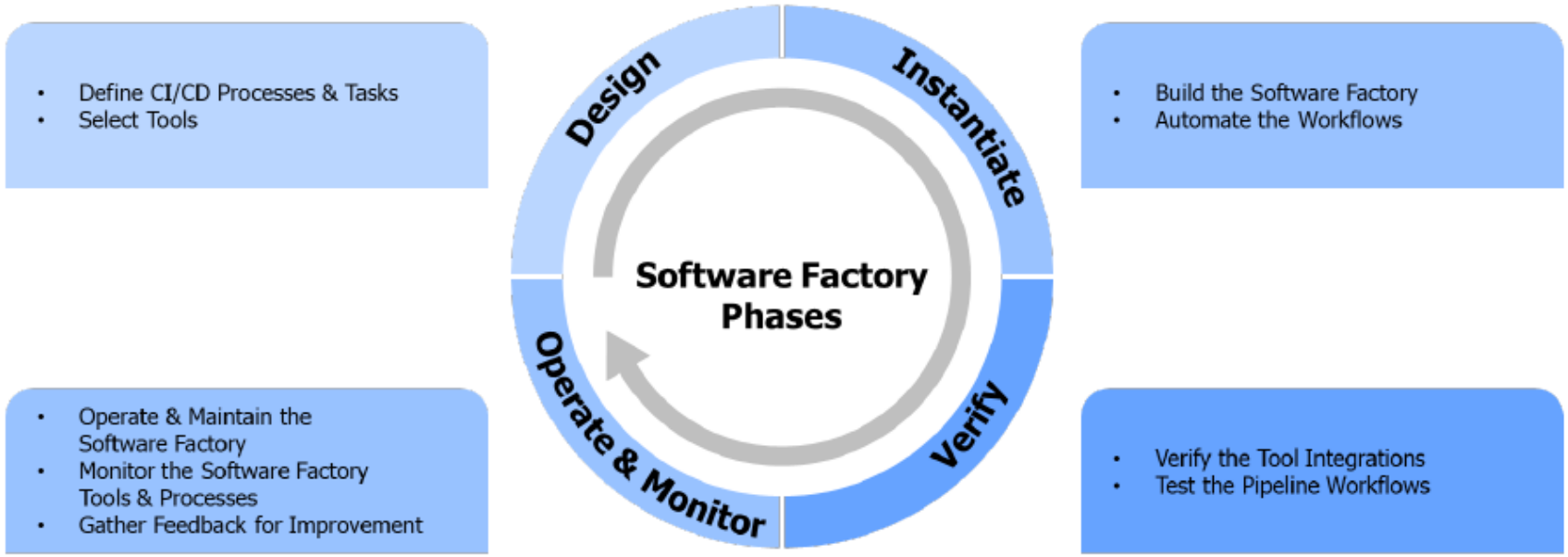
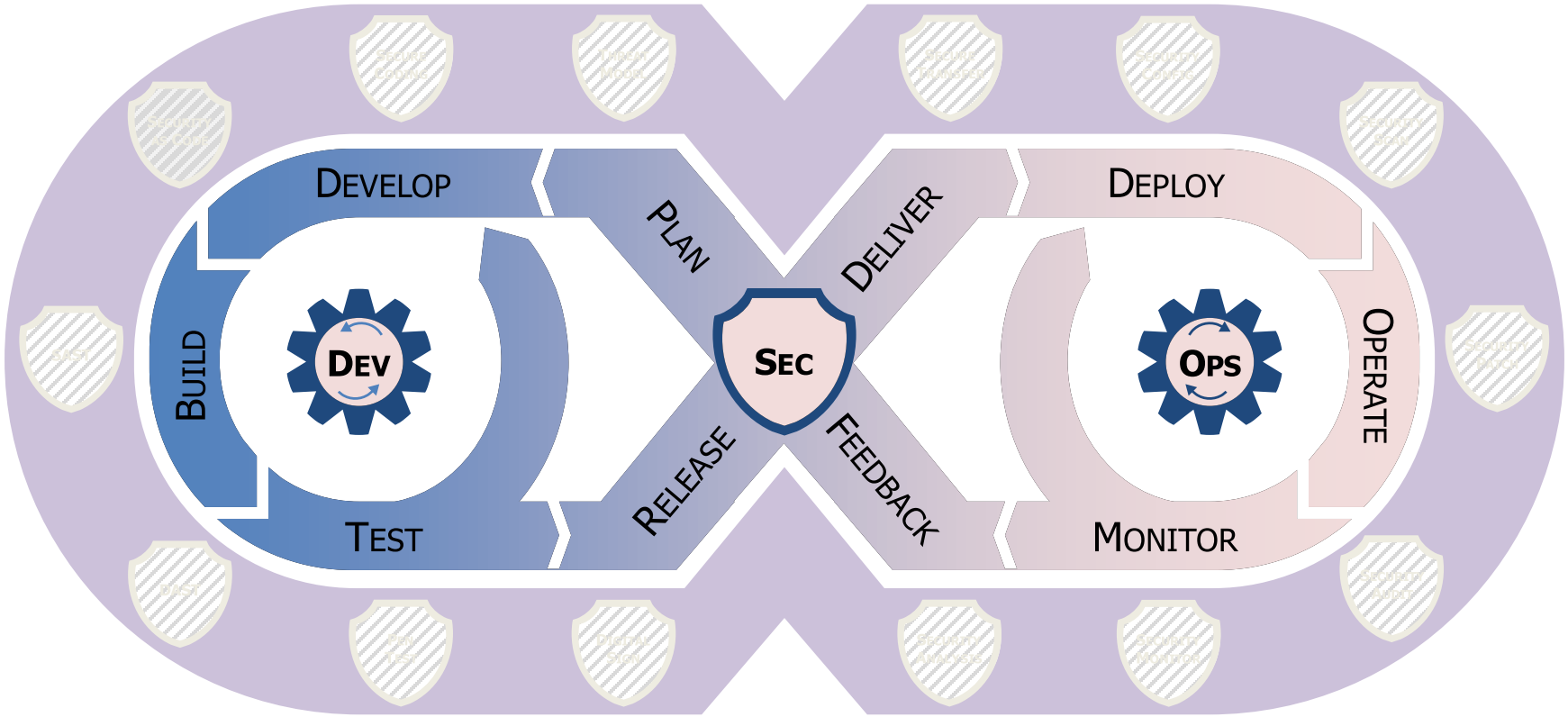


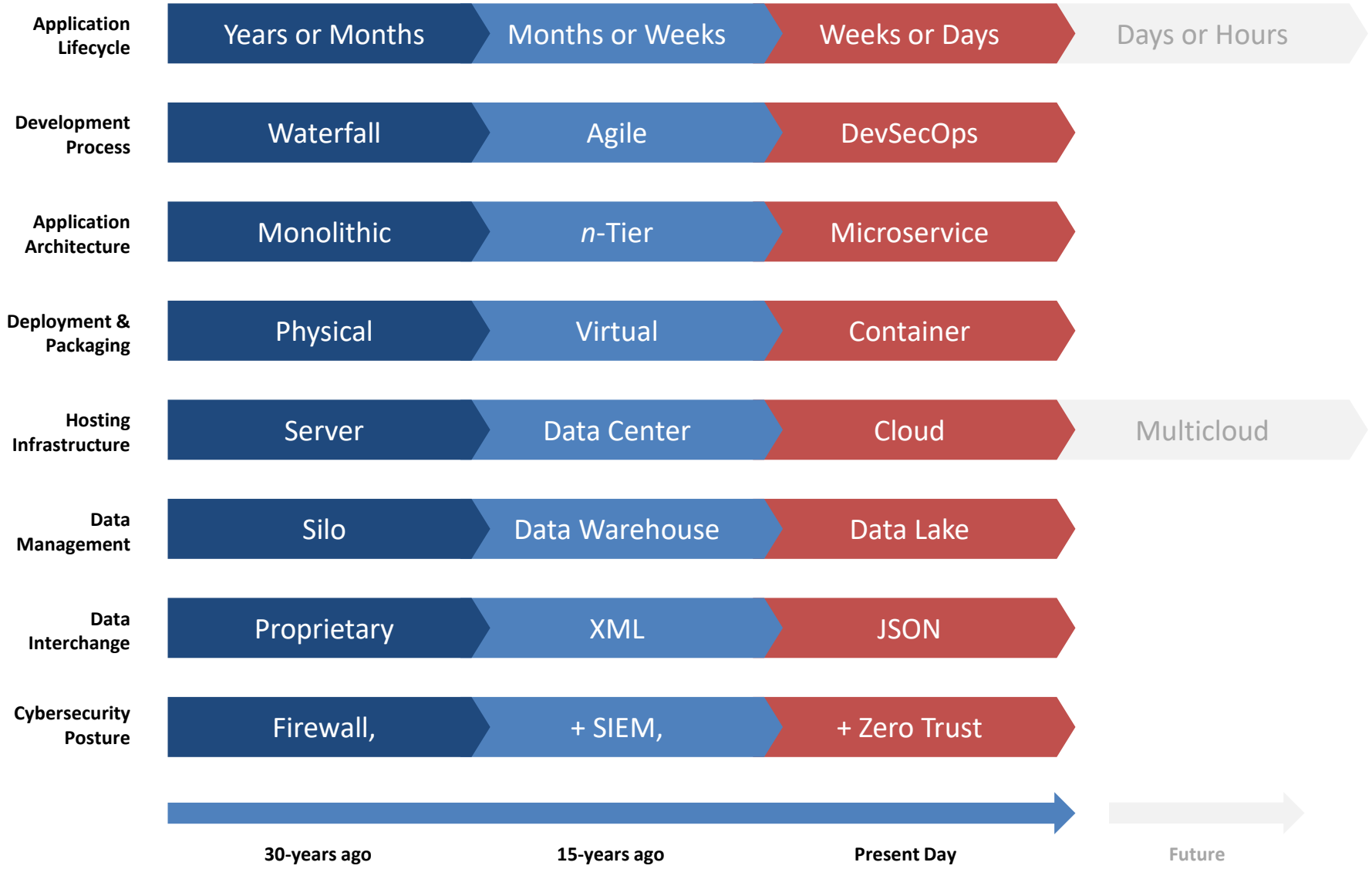
Figure 1 Software Factory Lifecycle Phases

-DoD DevSecOps Fundamentals Playbook March 2021

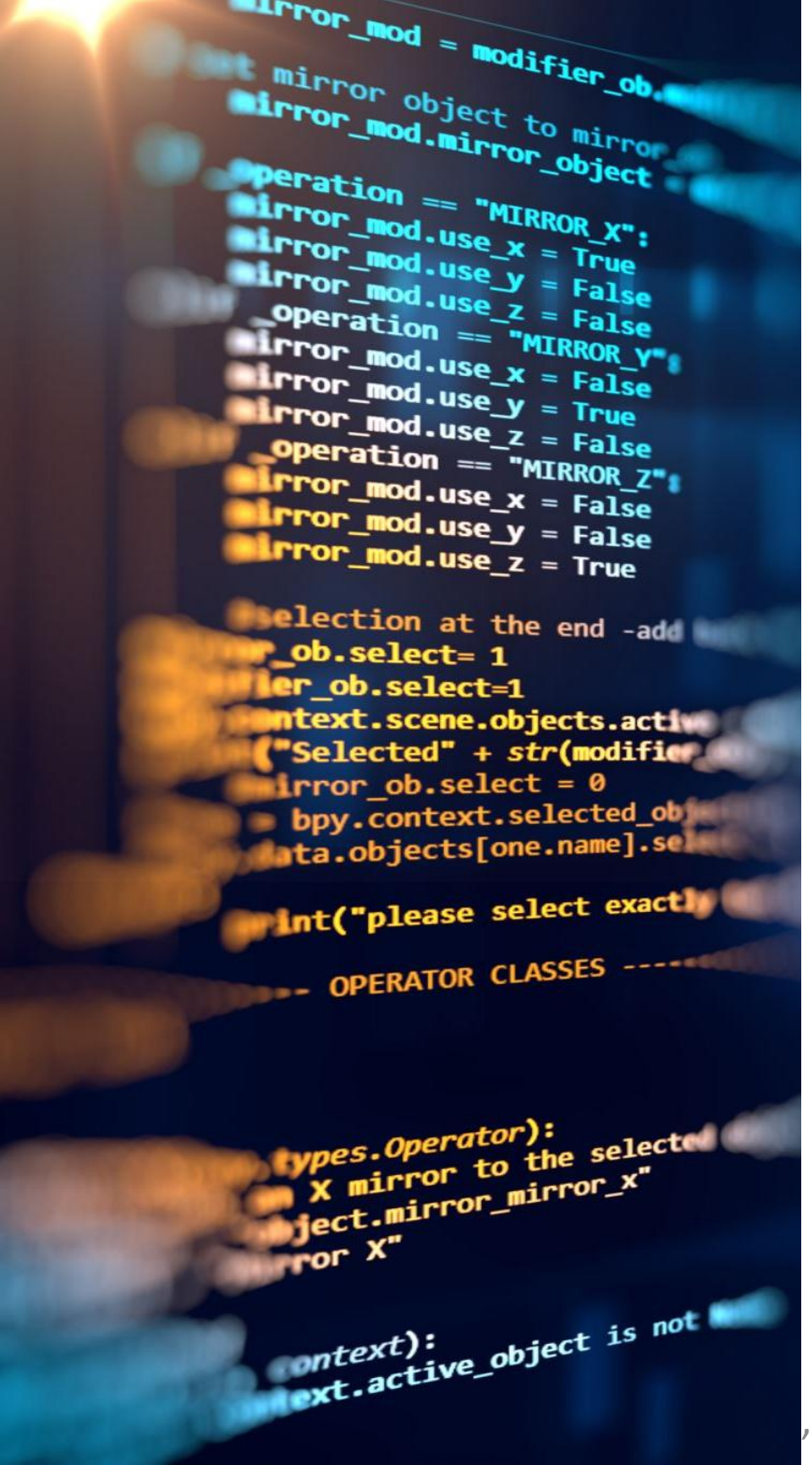
DevSecOps Loop



- DoD Enterprise DevSecOps-Source Diagrams



- DoD Enterprise DevSecOps-Source Diagrams



Version Control System

Version control is important when you're working on a software project over time.

- Many source and documentation files.
- The files change during development.
- You need to keep track of the different versions of each file, and possibly roll back to an earlier version.

A version control system (VCS):

- Records changes to your files over time.
- You can recall specific versions later.

Version Control System

A VCS is critical for a multi-person project.

How do you keep developers on a team from overwriting each other's changes?

How do you roll back to an earlier version of a file after deciding that the latest changes were bogus?

Who has the latest and greatest version of the source files?

Implementing an Effective DevSecOps Strategy

- A DevSecOps strategy focuses on security from the beginning and continues to focus on it throughout the software development lifecycle
- It is a complex process that uses the following for effective implementation
  - **Educate the Team:** Educate the team about the DevSecOps security **best practices** and explain their roles and responsibilities
  - **Allow Developers to Learn:** Developers are creative, and by training developers, they can implement the DevSecOps into the code. Provide the required tools to developers so that they can succeed in their tasks and contribute to the successful delivery of the secure software
  - **Conduct Audits and Security Assessments:** Audits and security assessments are a great way to **identify the existing vulnerabilities or flaws** and **remediate** them
  - **Enable Code-based Security:** From the beginning of the DevSecOps pipeline, developers should develop code using best practices
  - **Automation:** It is central to DevSecOps. Implement automation to perform repetitive and time-consuming tasks. This could enable teams to focus more on the quality of the software

Need for a Change Management Strategy

- In DevSecOps, different people work collaboratively on systems to perform different operations such as writing new code, changing networking configurations, and running tests to ensure security in all phases of software development
- For a successful implementation of change management, it is crucial to determine **how changes can be managed**, **what techniques** and **methodologies** can be used
- A change management strategy is important for DevSecOps as it ensures that the developers, security team, and operations team can handle system configurations and can make code changes faster and easier
- It also helps in **fixing security incidents faster**

Change Management Strategies

For the Successful Implementation of Change Management, Use the following Strategies:

- Use Agile and DevOps principles such as **automation** using rules and **thresholds** that can reduce human errors
- Use a **change-advisory board** (CAB) that provides support for change management by approving requests and by assessing and prioritizing changes
- Integrate with existing tools and processes such as the use of JIRA tickets to send emails for requesting change approvals along with summaries
- While using automation, **implement audit trails** to determine the time taken for making a change live, who approved it, etc
- Plan the release of changes early and avoid adding additional processes. For example, plan in advance for the mechanisms to be used for implementing changes effectively
- To reduce the negative impact of a change, **prepare for failures** and incidents using monitoring tools that can send an alert when a failure occurs
- Perform continuous improvement analyzing results, mitigating internal issues as well as the issues reported by customers

Summary of DevSecOps Tools Used in On-Premises Environments

Category of Conventional DevSecOps Tools	Tools
Threat Modeling Tools	OWASP Threat Dragon, ThreatModeler, Threatspec, RainDance, PyTM, MAL, Threagile, SD Elements, Tutamen Threat Model Automator, itemis SECURE, ThreatPlaybook, DREAD
Pre-Commit Hooks	Talisman, Crass, Git Secrets, pre-commit, detect-secrets, git-hound, TruffleHog
Software Composition Analysis	RubySec, Retire JS, requires.io
Static Analysis Security Testing	Bandit, Brakeman, Codesake Dawn, FindBugs, PMD, graudit, RIPS, Puma Scan, Reshift, INSIDER CLI, Spectralops, Klocwork, HCL Appscan, Fortify, Coverity, CodeWarrior
IDE Plug-ins	DevSkim, JFrog Eclipse, Snyk, CAT.net, SpotBugs, FindBugs, Find Security Bugs
Secrets Management	Hashicorp Vault, Torus, Keywhiz, EnvKey, Confidant, Doppler, Berglas
Vulnerability Management	ArcherySec, Defect Dojo, Jackhammer, ThreadFix, Qualys, Flexera, Rapid7 InsightVM, Falcon Spotlight, Vulnerability Manager Plus, IP360, Kenna Security, F-Secure Elements VM, GFI Languard, Greenbone's VM, beSECURE
Dynamic Application Security Testing (DAST)	Codename SCNR, Nikto, Acunetix, Fortify, WebInspect, Veracode Dynamic Analysis, w3af, Wapiti, Sentinel Dynamic, Rapid7, Mister Scanner, HCL AppScan, GitLab Ultimate
Security in Infrastructure as Code	Clair, Anchore Engine, Dagda, OpenScap, Dockscan, Snyk IaC Security, Infrastructure as Code (IaC) Security, Open VAS, CloudSploit, Accurics, Checkov, TFLint
Compliance as Code	InSpec, Serverspec, DevSec Hardening Framework, Kitchen CI, Docker Bench for Security
Web Application Firewall	ModSecurity WAF, NAXSI, WebKnight, Shadow Daemon, Imperva WAF

Software Factory

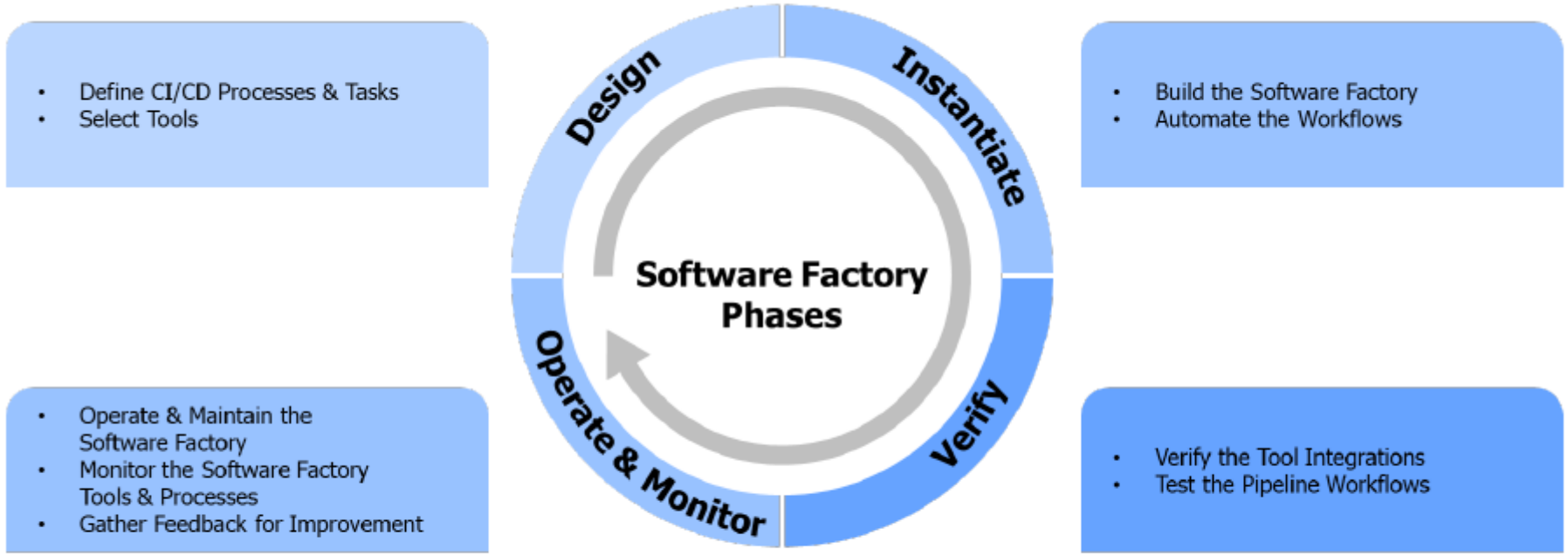


Figure 1 Software Factory Lifecycle Phases

DevSecOps Fundamentals Guidebook:DevSecOps Tools & Activities March 2021 Version 2.0

Software Factory

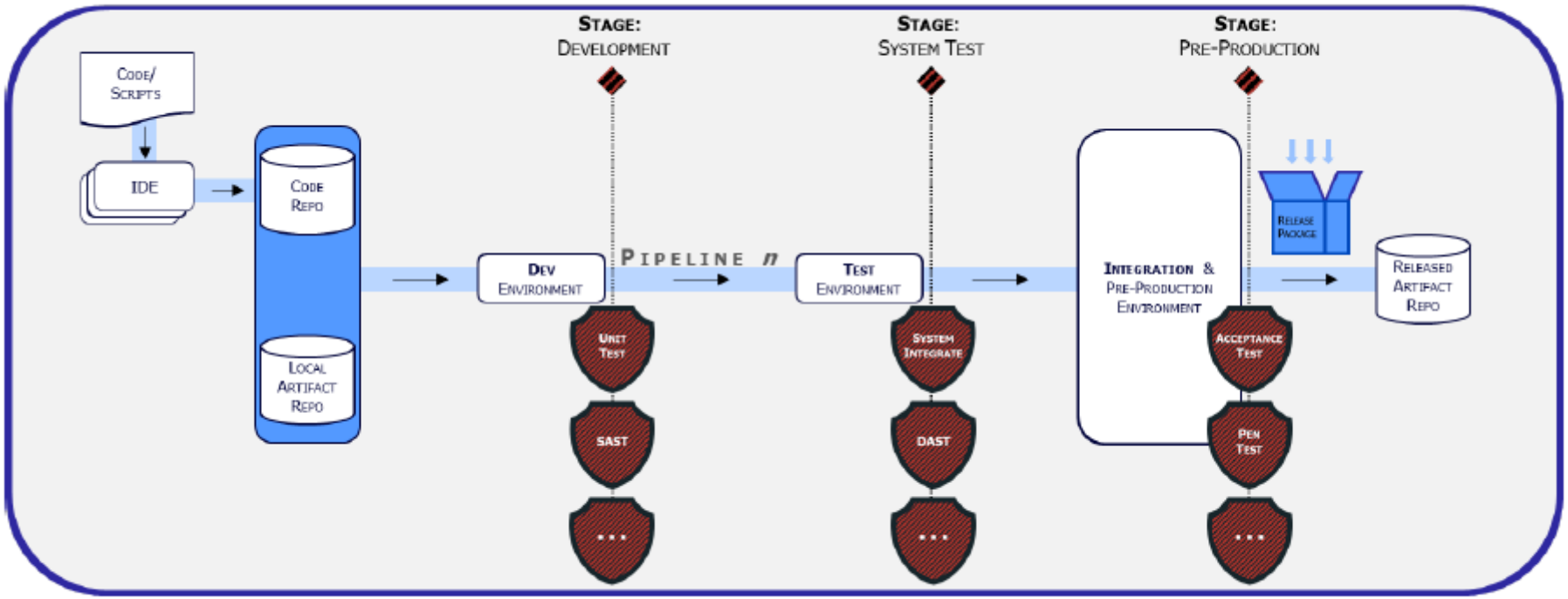


Figure 6 Notional expansion of a single DevSecOps software factory pipeline

DoD Enterprise DevSecOps Fundamentals March 2021

Software Factory

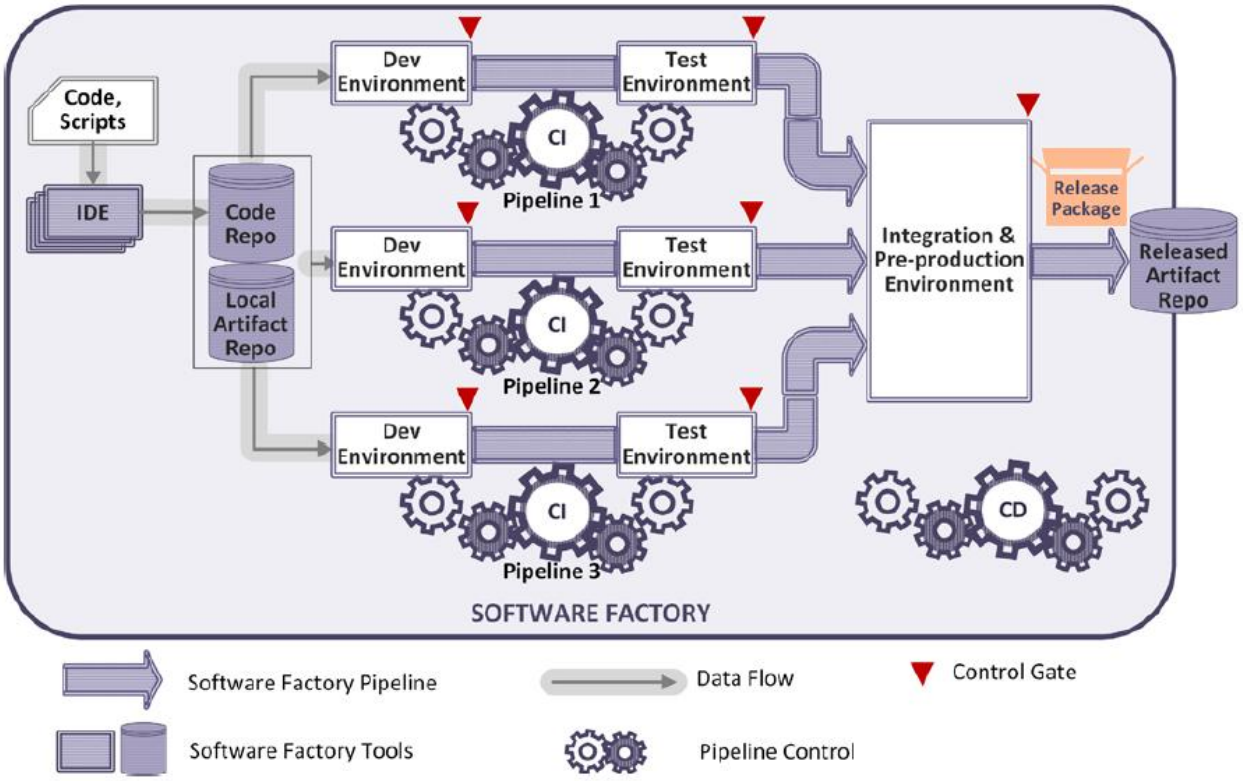


Figure 9: DevSecOps Software Factory

DoD Enterprise DevSecOps Fundamentals March 2021

Center for Internet Security (CIS)

The Center for Internet Security (CIS) Controls represent a prioritized sequence of actions that offer a defense-in-depth set of best practices to thwart the most common attacks on systems and networks. These controls are crafted by a consortium of IT experts utilizing their first-hand experience as cybersecurity defenders to produce these globally recognized security best practices. The CIS Controls' principles dovetail with DevSecOps, enhancing security throughout the software development lifecycle.

Kumar Rath, Ashwini. Concepts and Practices of DevSecOps: Crack the DevSecOps interviews (English Edition) (pp. 167-168). BPB Publications. Kindle Edition.

Center for Internet Security (CIS)

Control 1: Inventory and Control of Enterprise Assets

Control 2: Inventory and Control of Software Assets

Control 3: Data Protection

Control 4: Secure Configuration of Enterprise Assets and Software

Control 5: Account Management

Control 6: Access Control Management

Control 7: Continuous Vulnerability Management

Control 8: Audit Log Management

Control 9: Email and Web Browser Protections

Control 10: Malware Defenses

Control 11: Data Recovery

Control 12: Network Infrastructure Management

Control 13: Network Monitoring and Defense

Control 14: Security Awareness and Skills Training

Control 15: Service Provider Management

Control 16: Application Software Security

Control 17: Incident Response Management

Control 18: Penetration Testing

-https://learn.cisecurity.org

The 5 Ideals of DevSecOps

First Phase *Adopt Agile and Risk Management Framework (RMF)*

Introduce on Kanban&Scrum-based agile development teams

•Bing Agile coaches and streamlinethe development process

•Introduce new development practices,

•Continuous integration and delivery pipeline

•Centralized code repository

•Remove extraneous documentation : Focus on application delivery

•Establish a “Path-to-Production” to begin process improvements that better meet the

customer demand signal

Second Phase *Mature the enterprise, People, Policy and Agile Process*

- •Build a team culture
- •Transform program mindset to mission need as *Business Value*
- •Learn and adopt Agile process
- •RMF accreditation on systems of system level
- •NIST 800-37, 800-53
- •Establish MVP software delivery pipeline
- •Source Code repo, Build Environments, Collaboration platforms
- •Focus on Agility and Security

Third Phase *Modernization (Architecture and Tooling)*

- Commoditization of components and introduction of containerization
- Address technical dept in legacy application
- Migration to modular architecture (Microservices, MOSA)
- Improve development and deployment tools
- Integrated deployment pipeline : from inception to operation)
- Operationalize SREon infrastructure team
- Develop contracts SLAs and SLOs
- Introduce Audit vs Gate keeper

Fourth Phase *Operationalize DevSecOps)*

- Codify CI/CD tools, creating DevSecOpsPipeline for Continuous Authorization
- Automation and Immutable Environments:
- Source controlling Infrastructure-as-Code(IaC)
- Pipeline release supporting a live DoD system
- DevSecOpsenabled Data Science workflows and deployments
- Introduce New Governance approach based on DevSecOps
- Select small discreet parts of the enterprise as projects for DevSecOpsenablement

Fifth Phase *Mature DevSecOps and Scale*

- Scheduled a Pipeline to GO-LIVE event with Analytic
- Platform and tech refresh
- Orchestrate and extend the on-prem enterprise into the cloud
- Collect Lessons Learned
- Incentivize transparency between contractors and government
- Incentivize smaller releases
- Begin tracking development metrics
- Implement Deployment and Monitoring strategies

DevSecOps Maturity Levels

Term	Documentation
Maturity Level 1	Performed Basic Practices: This represents the minimum set of engineering, security, and operational practices that is required to begin supporting a product under development, even if only performed in an ad-hoc manner with minimal automation, documentation, or process maturity. This level is focused on minimal development, security, and operational hygiene.
Maturity Level 2	Documented/Automated Intermediate Practices: Practices are completed in addition to meeting the level 1 practices. This level represents the transition from manual, ad-hoc practices to the automated and consistent execution of defined processes. This set of practices represents the next evolution of the maturity of the product under development's pipeline by providing the capability needed to automate the practices that are most often executed or produce the most unpredictable results. These practices include defining processes that enable individuals to perform activities in a repeatable manner.
Maturity Level 3	Managed Pipeline Execution: Practices are completed in addition to meeting the level 1 and 2 practices. This level focuses on consistently meeting the information needs of all relevant stakeholders associated with the product under development so that they can make informed decisions as work items progress through a defined process.
Maturity Level 4	Proactive Reviewing and Optimizing DevSecOps: Practices are completed in addition to meeting the level 1-3 practices. This level is focused on reviewing the effectiveness of the system so that corrective actions are taken when necessary, as well as quantitively improving the system's performance as it relates to the consistent development and operation of the product under development.