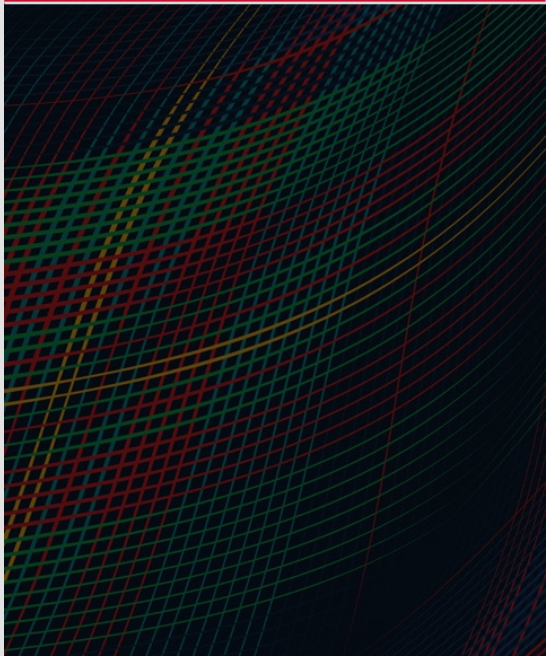
# Are your DevSecOps Capabilities Mature?

**OCTOBER 12, 2023**

Timothy A. Chick

Carnegie Mellon University
Software Engineering Institute



# Distribution Statement


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.

DM23-1084

# Agenda

Carnegie Mellon University Software Engineering Institute

**What is DevSecOps – It's complicated**

**Understanding the Complexity**

**Capability Maturity**

**DevSecOps Capability Maturity Visualization**

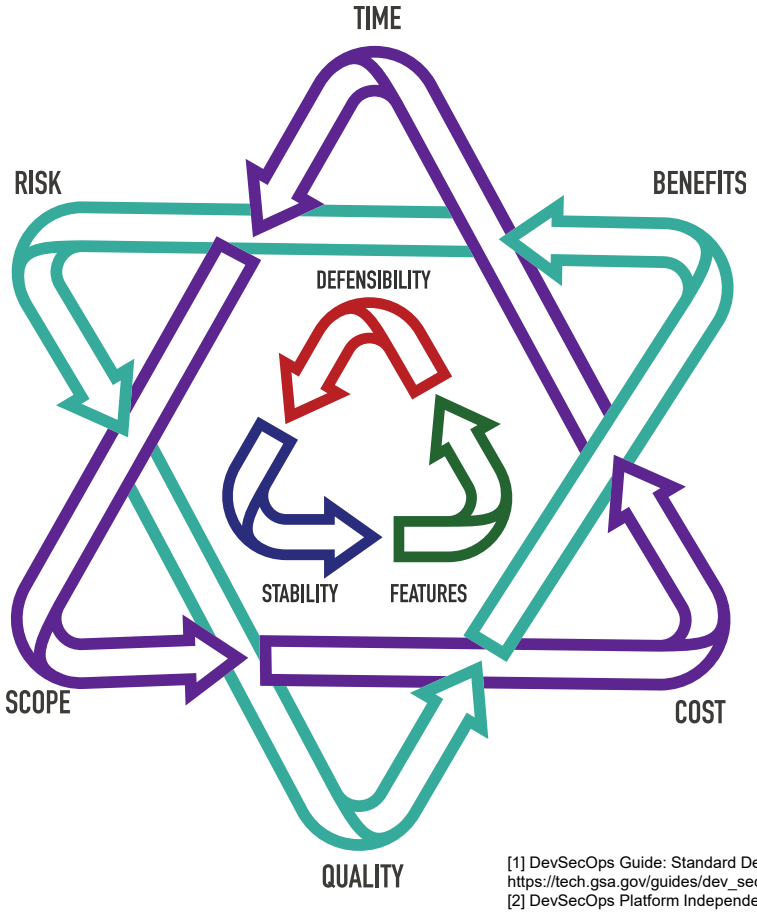
**Conclusion**

# What is DevSecOps – It’s complicated

Carnegie Mellon University
Software Engineering Institute

# DevSecOps: Modern Software Engineering Practices and Tools that Encompass 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].

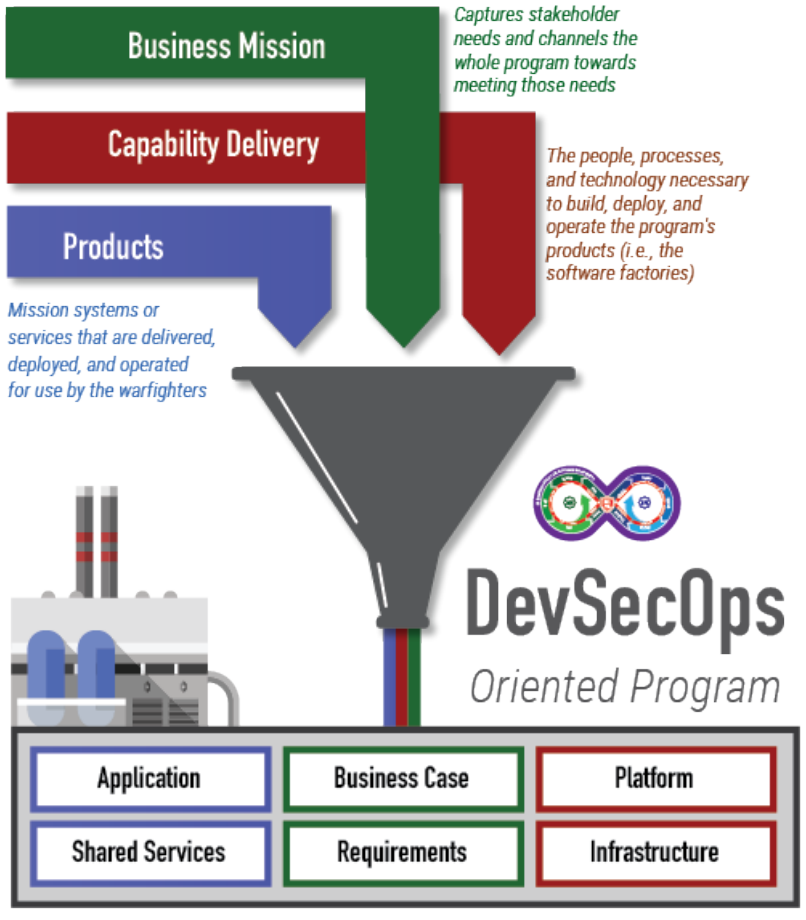
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

- **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

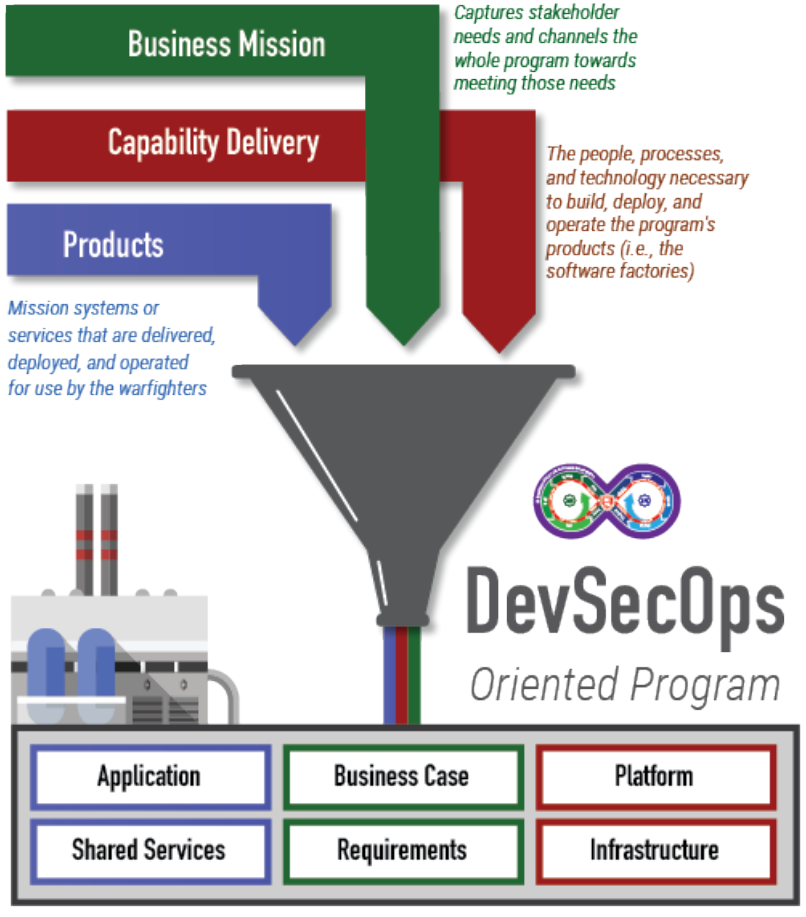



All software-oriented enterprises are driven by three concerns:

- **Business Mission** – captures stakeholder needs and channels the whole program in meeting those needs. It answer 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** – the units of value delivered by the program. Products utilize the capabilities delivered by the software factory and operational environments.

# Challenge 1: connecting process, practice, and tools

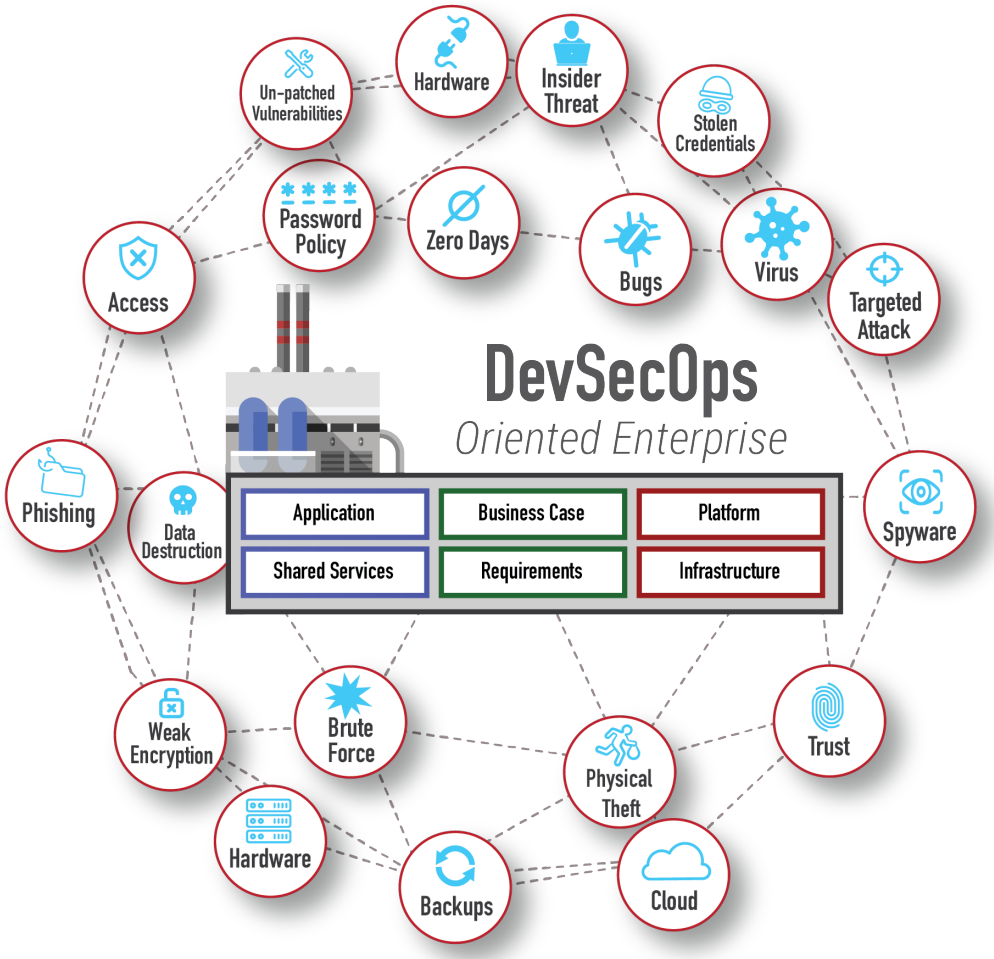





Creation of the DevSecOps (DSO) pipeline for building the product is not static.

- Tools for process automation must work together and connect to the planned infrastructure
- Infrastructure and shared services are often maintained across multiple organizations (Cloud for infrastructure, third parties for tools and services, etc.)
- Processes, practices, and tools must evolve to meet the needs of the products being built and operated

# Challenge 2: Cybersecurity of Pipeline and Product




The tight integration of Business Mission, Capability Delivery, and Products, using integrated processes, tools, and people, increases the attack surface of the product under development.

Managing and monitoring all 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?

# 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.

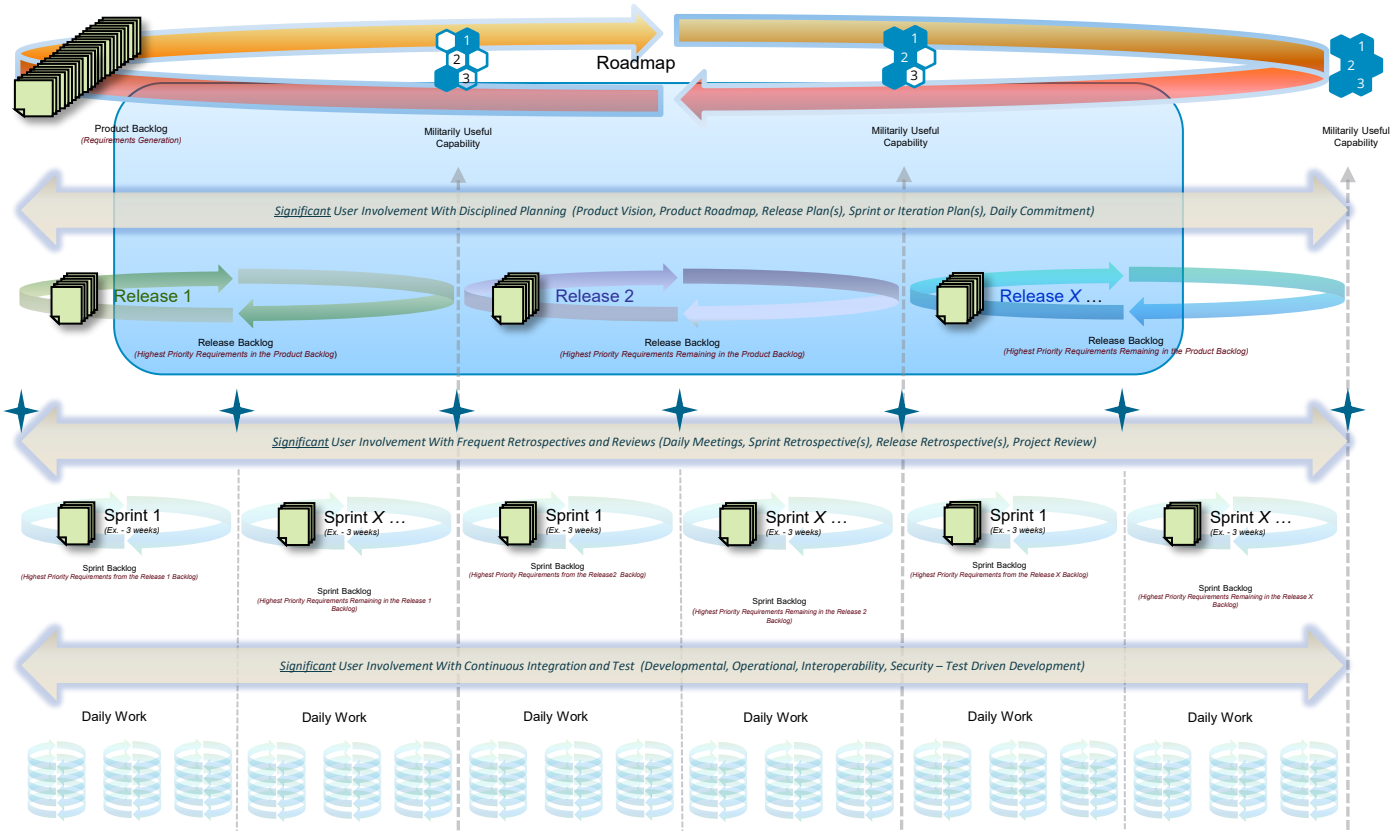
NOTE: this is in addition to the Agile Principles. You **CANNOT** do DevOps without first doing Agile.

# Not a Myth: Agile *is* likely to fail if it's "only the development team" that adopts the new practices



***Frequent failure mode:*** Business and/or operations doesn't keep up with what the development teams can deliver.

***Why?***
Shift to Agile-based requirements definition and management is more of a change than practices of development alone.




Graphic Source: Figure 4, *Parallel Worlds: Differences in Agile and Waterfall Differences & Similarities,* S. Palmquist et al, SEI-2013-TN-021, October 2013.

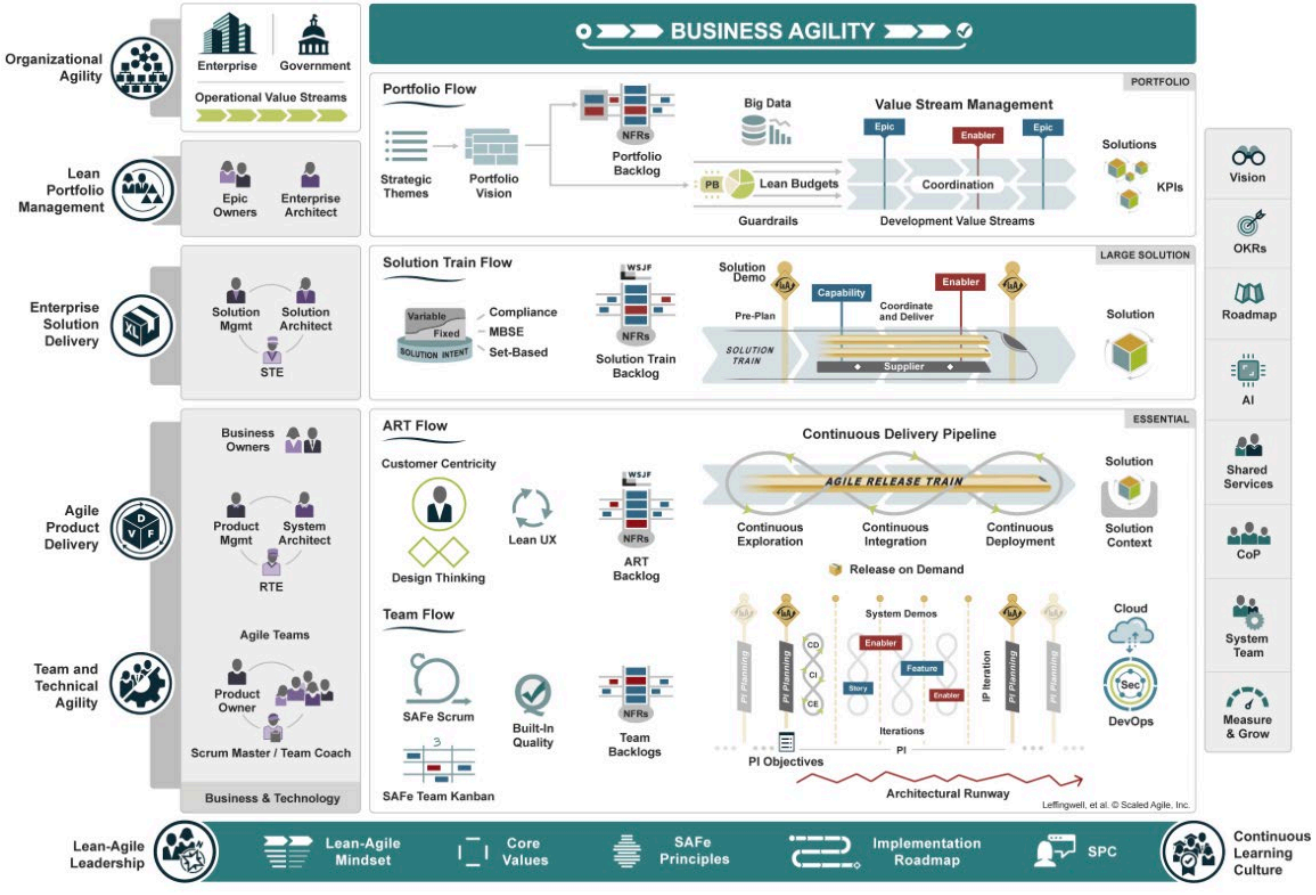
# BUT, the Agile Principles Were Designed and Focused on Small Teams



There are lots of things beyond supporting a small team that you must focus on when scaling above a few small teams:

- managing the interfaces among the many products/system components that multiple teams are working on…
- figuring out how to synchronize releases and events across multiple teams…
- figuring out how to get the inventory (backlog) of requirements organized productively to support the development pace of multiple small teams….
- dealing with specialty disciplines (UX, security, etc.) that have significant inputs to the evolving product but aren't needed as full-time team members….
- Etc.

# Scaled Agile Framework (SAFe)—
# Scaling Framework that is Most Frequently Seen in Large Organizations

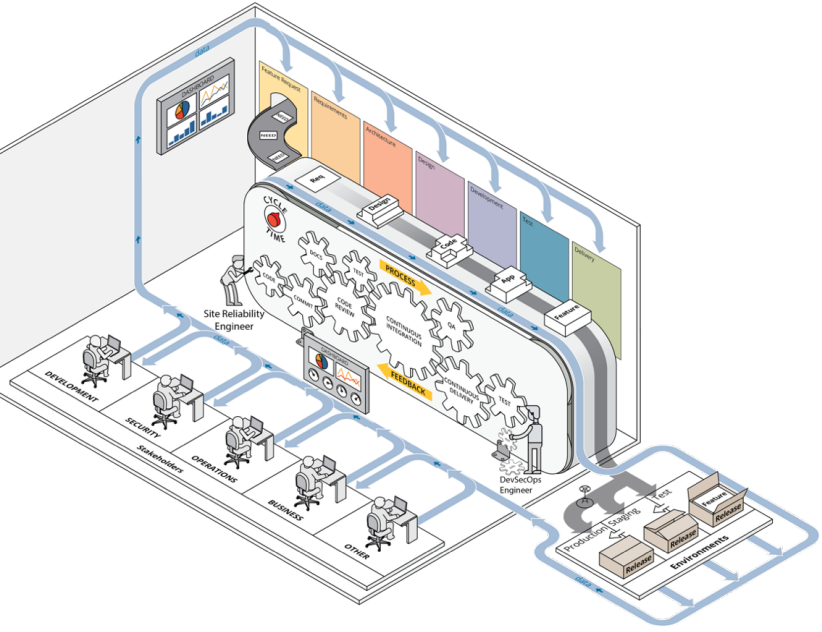



SAFe leverages lean engineering practices at Portfolio, Value Stream, and Program levels.

SAFe leverages Scrum, XP, and Kanban practices at the Team level.

NOTE: Frameworks are often misapplied because of the failure to understand that they are only “frameworks” and not “process specifications”

# DevSecOps is a Complex System




- **System** is “an assemblage or combination of things or parts forming a complex or unitary whole” [1]. Thus, DSO is a system.
- DSO also possesses the **characteristics of a socio-technical system** [2] and a computer information system, since DSO is composed of **people, processes, and computer technology** that are “designed to collect, process, store, and distribute information” [3].
- If we add to this definition that **DSO pipelines are composed of independently developed, independently maintained, likely physically and logically distributed, task-dedicated, interoperable components**, then we can affirm that DSO pipelines are **complex sociotechnical** computer information systems.

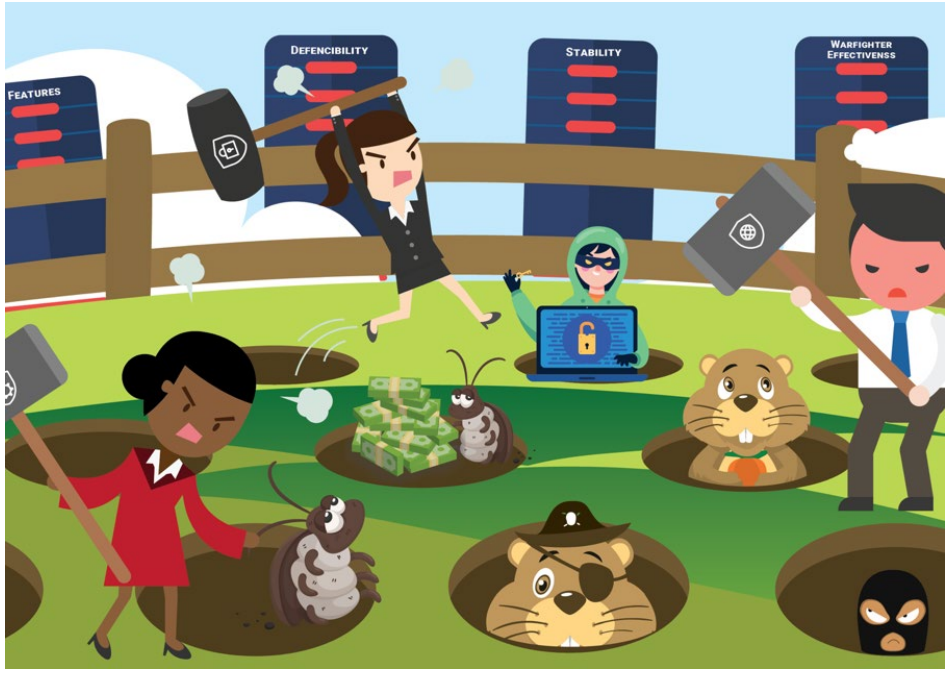
[1] system, https://www.dictionary.com/browse/system
[2] SEBoK, https://www.sebokwiki.org/wiki/Sociotechnical_System_(glossary)
[3] Information system, https://en.wikipedia.org/wiki/Information_system

# Understanding the Complexity

Carnegie Mellon University
Software Engineering Institute

# Avoiding Pipeline Whac-A-Mole






The usage of a pipeline is the cornerstone of a mature and robust DevSecOps environment.

A CI/CD pipeline provides for the vetting, building, testing, packaging, and delivery of a change to the desired destination.

No standard implementation of DevSecOps or a pipeline exists that works for everyone.

Each organization must analyze its

- Existing and desired culture
- Budget constraints
- Defined business cases
- Solution space

So how do you:

- go about building it?
- decide what to implement first, second, …?

It is extremely important to have a well-defined end state in mind before implementation begins to avoid costly mistakes.

The end state must consider the perspective of all stakeholders, including software architects, developers, system administrators, test and quality-assurance engineers, security officials, management, end users, etc.

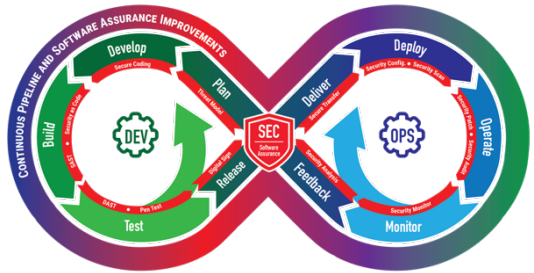
# Insufficient details for a complicated socio-technical system



While one can search “DevSecOps” or “DevOps” and find a lot of literature that paints a picture of what it could be or should be, they are not definitive and require a considerable amount of interpretation

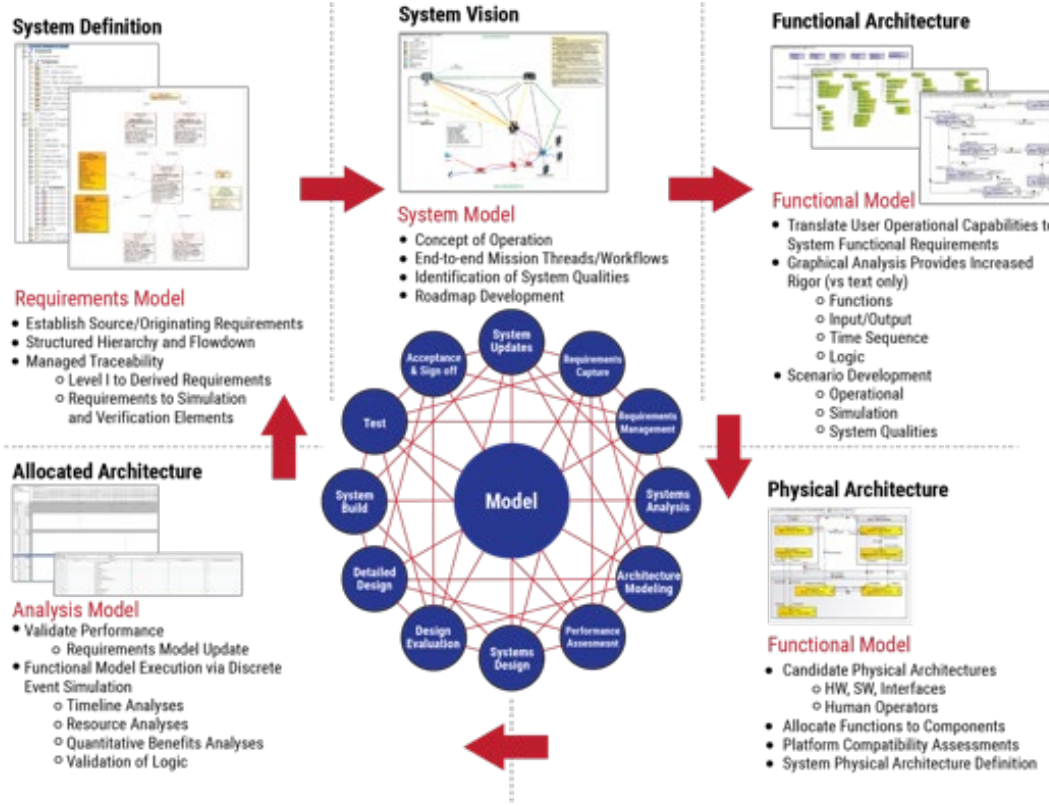
Resulting in:

- Perspectives not being fully integrated in organizational guidance and policy
- projects being unable to perform an analysis of alternatives (AoA) in regard to the pipeline tools and processes
- multiple projects using similar infrastructure and pipelines in different and incompatible ways, even within the same organization
- suboptimal tools and security controls
- Etc.



# Socio-technical Systems Require Model Based Systems Engineering






*The Digital System Model contains the most current requirements, key mission/business operations, architecture, design details, implementation details, test and evaluation details, and supporting documentation.

- ***Not* yesterday's Document-Centric Systems Engineering!**
- MBSE uses a Digital System Model* to facilitate common system understanding and decision-making.
- The Digital System Model* is the single authoritative source of truth
- System and Components can be integrated at various levels of abstraction and fidelity
- Model Views are chosen to best communicate information to a variety of stakeholders via the dynamic creation of multiple, consistent, accurate views
- Impacts of changes are more easily analyzed and evaluated

# Why Apply Model-based Engineering to DevSecOps?



The idea of **applying model-based engineering methods to socio-technical systems is not new**. Examples include; social systems [1] [2], complex command and control system [3], border security [4], sociotechnical systems [5]

The overall **adoption of model-based engineering and virtual modeling tools in everyday practices has grown**.

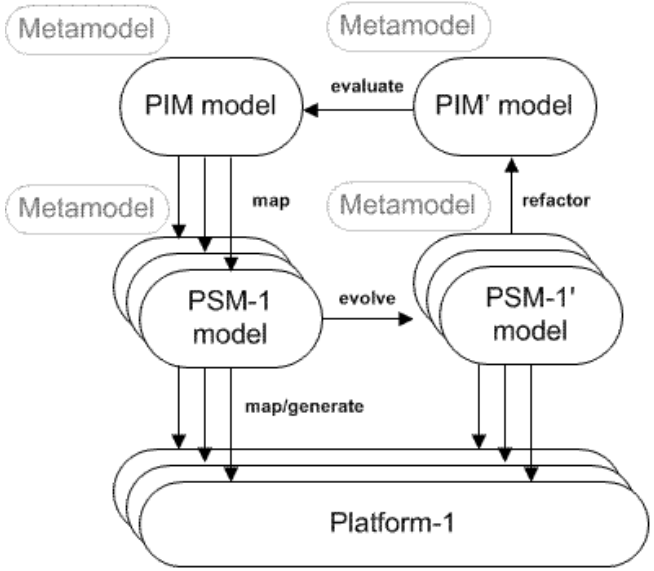
**Using a model-based approach** such as BPM (Business Process Modeling) **to design or describe patterns of human activities as a context of the functioning of a computer information system**, aka business process, is a **standard practice in industry** now [6].

[1] Haskins, C. 2008a. “Using patterns to transition systems engineering from a technological to social context,” Systems Engineering, 11: 147–155.
[2] Palmer, E. 2016. “Investigating structural gender inequality in the Norwegian pension system: An example of using MBSE in the evaluation of social systems,” 26th Annual INCOSE International Symposium (IS 2016), Edinburgh, Scotland, UK, July 18-21, 2016
[3] Oosthuizen, R., Venter, J.P., Serfontian, C. 2018 “Model Based Systems Engineering Process for Complex Command and Control Systems,” 23rd International Command and Control Research and Technology Symposium (ICCRTS 2018), Pensacola, USA, 6-9 November 2018
[4] Asan, E., Albrecht, O., Bilgen, S. 2013 “Handling Complexity in System of Systems Projects – Lessons Learned from MBSE Efforts in Border Security Projects,” 4th International Conference on Complex System Design & Management, pp. 281-299.
[5] Miller, Lori Ann, "Modeling forward base camps as complex adaptive sociotechnical systems" (2012). Masters Theses. 6943.
[6] OMG Standards for BPM, https://www.omg.org/technology/readingroom/BPM.htm

# Bridging the gap between theory and implementation



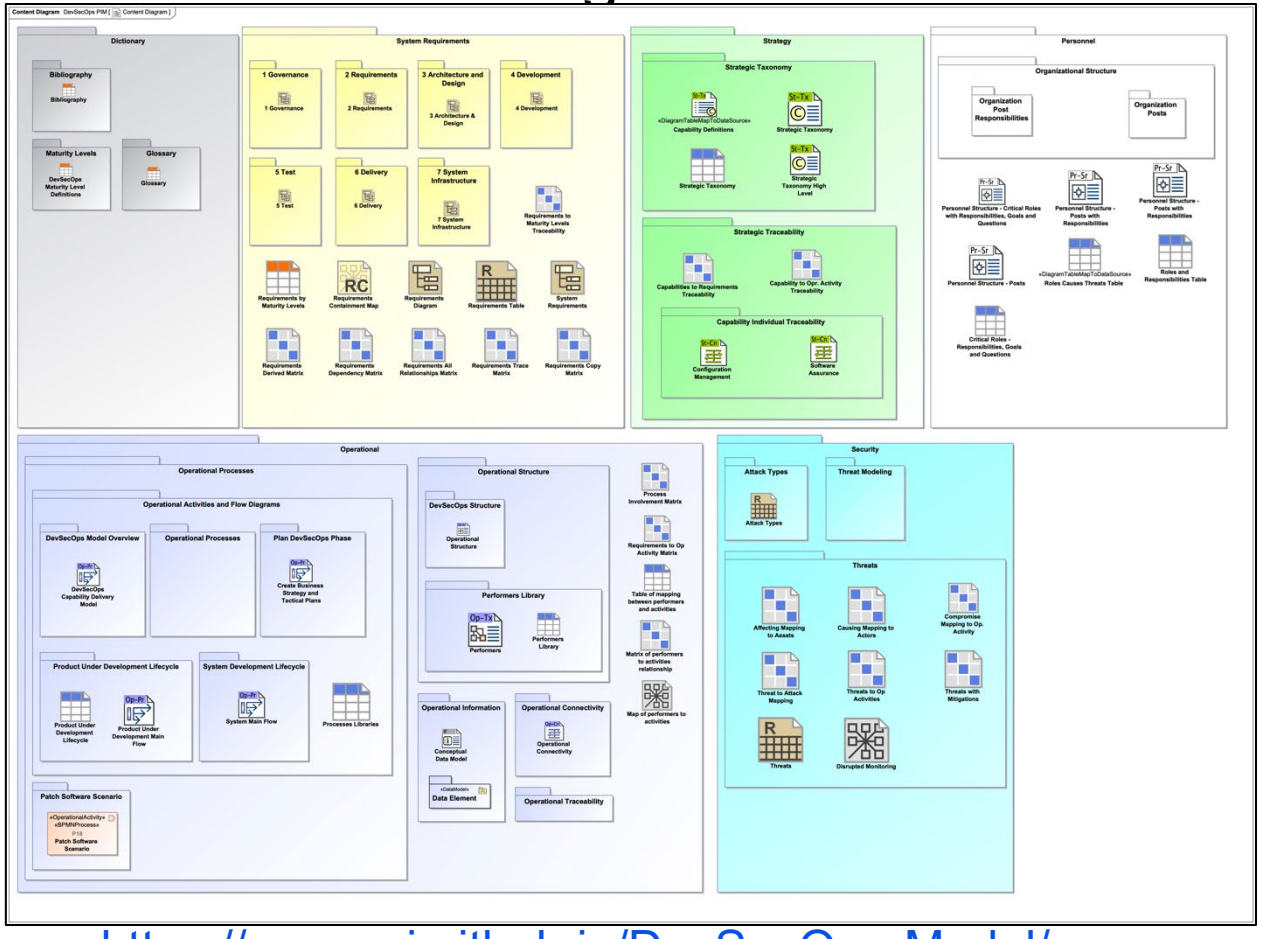
A PIM is a general and reusable model of a solution to a commonly occurring problem in software engineering within a given context and is independent of the specific technological platform used to implement it.




SEI DevSecOps Platform Independent Model (PIM)

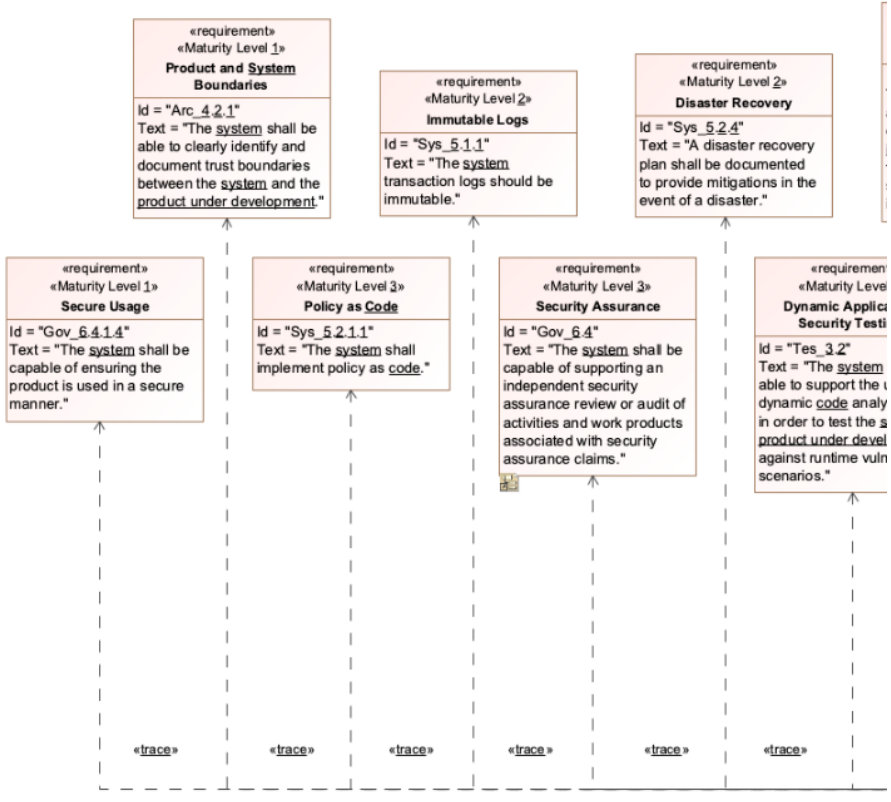
- is an authoritative reference to fully design and execute an integrated Agile and DevSecOps strategy in which all stakeholder needs are addressed
- enables organizations to implement DevSecOps in a secure, safe, and sustainable way to fully reap the benefits of flexibility and speed available from implementing DevSecOps principles, practices, and tools
- was developed to outline the activities necessary to consciously and predictably evolve the pipeline, while providing a formal approach and methodology to building a secure pipeline tailored to an organization's specific requirements

# DevSecOps PIM - Content Diagram




https://cmu-sei.github.io/DevSecOps-Model/

# DevSecOps Requirements




Example of Requirements Representation in Diagrams from PIM

All requirements are organized into categories based on logical and functional groupings:

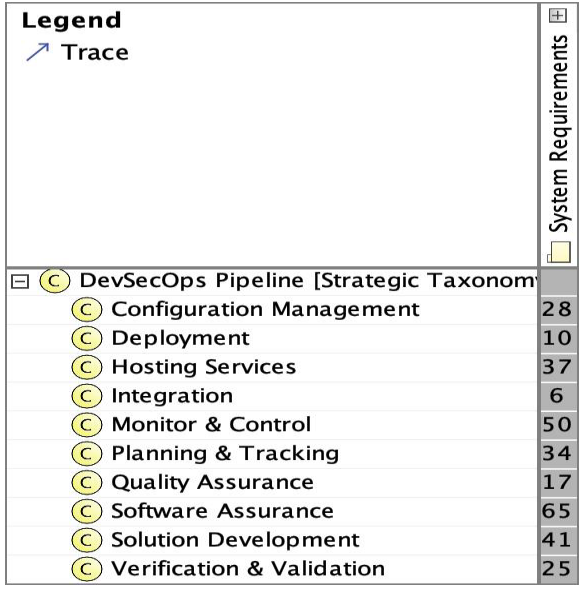
- Governance
- Requirements
- Architecture and Design
- Development
- Test
- Delivery
- System Infrastructure

Requirements Table Link

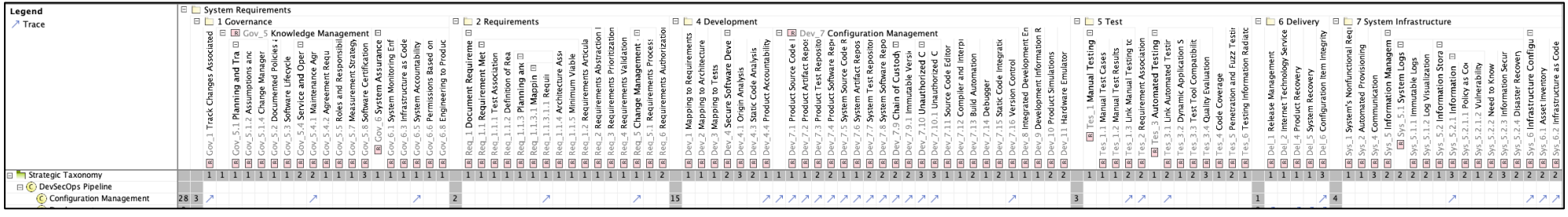
# DevSecOps Capability/Strategic Viewpoint

A capability is a high-level concept that describes the ability of a system to achieve or perform a task or a mission.

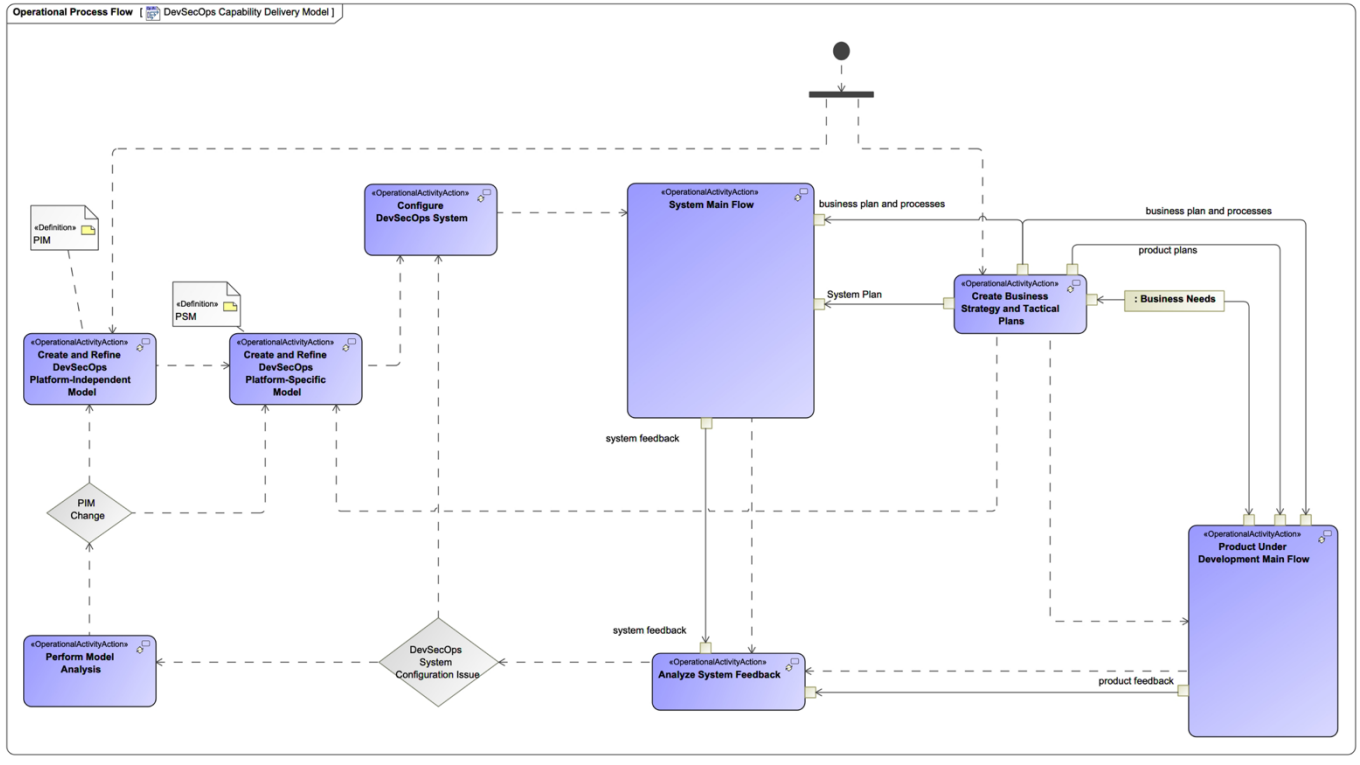
All requirements in the DevSecOps PIM were allocated to corresponding capabilities.




- Capability to Requirements Traceability Link
- Capability to Operational Activity Traceability Link
- Capability Definitions Link
- Strategic Taxonomy High Level



# DevSecOps Operational Viewpoints




- [DevSecOps Capability Delivery Model Link]

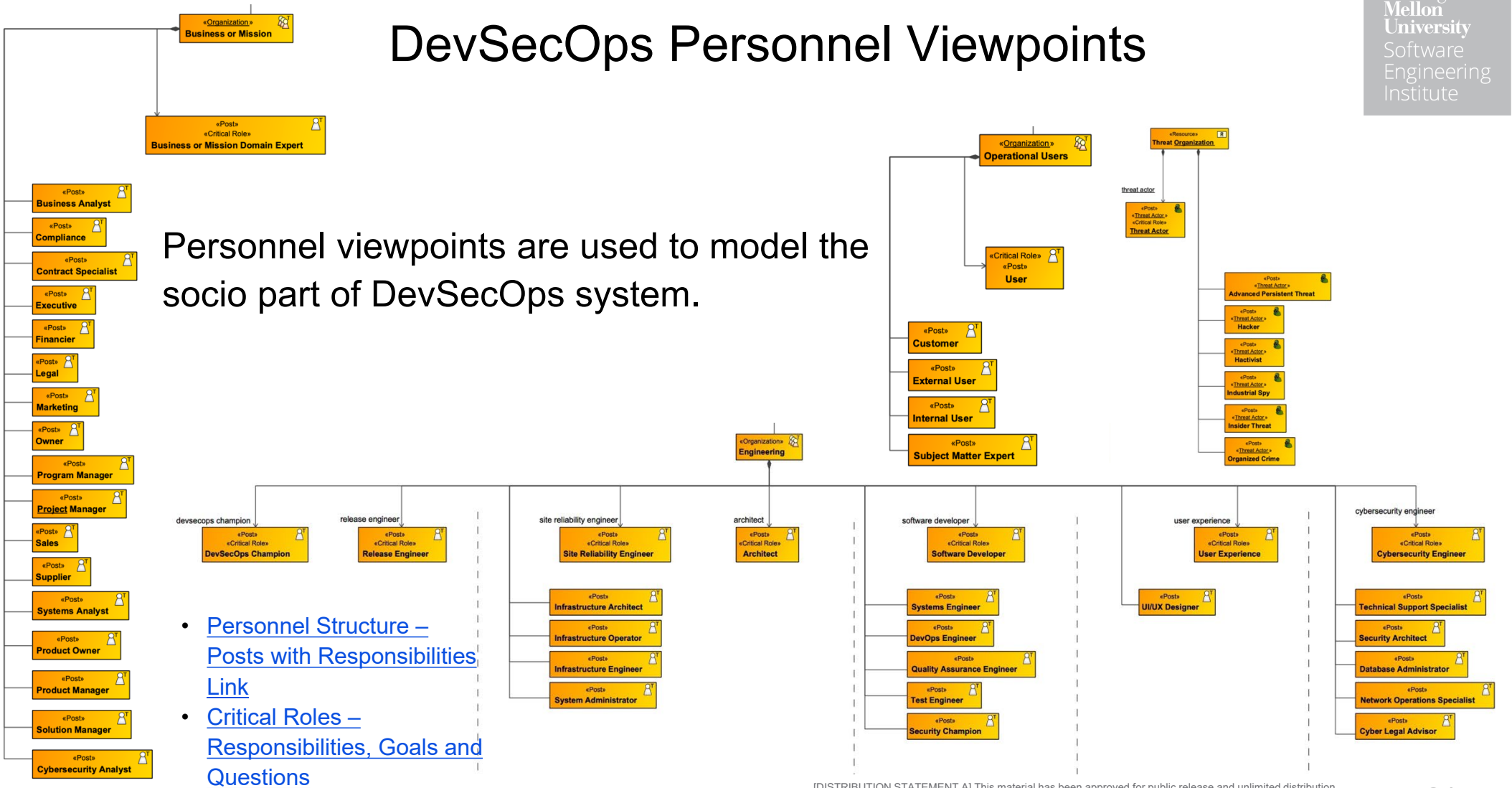
An operational model for a system describes behavior of the system to conduct enterprise operations.

The main operational processes for DevSecOps includes development process for the product, as well as the DevSecOps process itself.

# DevSecOps Personnel Viewpoints

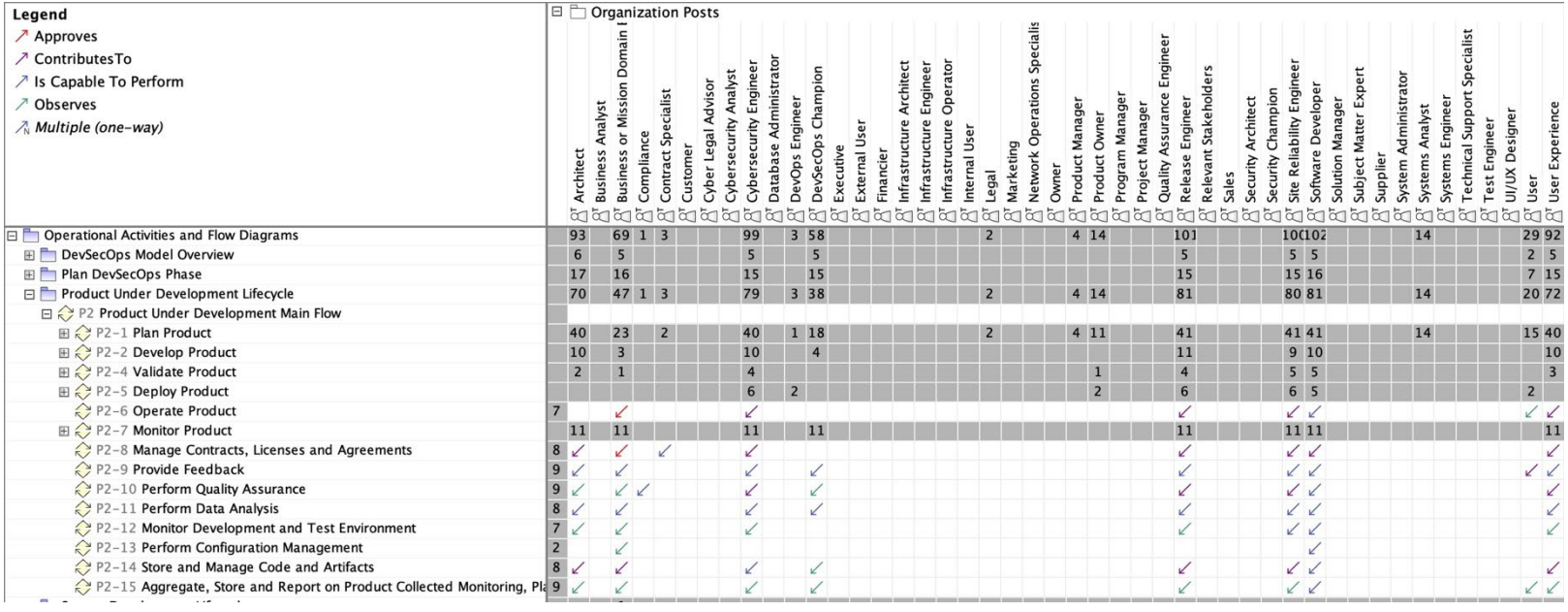
Carnegie Mellon University Software Engineering Institute

Personnel viewpoints are used to model the socio part of DevSecOps system.




- Personnel Structure – Posts with Responsibilities Link
- Critical Roles – Responsibilities, Goals and Questions

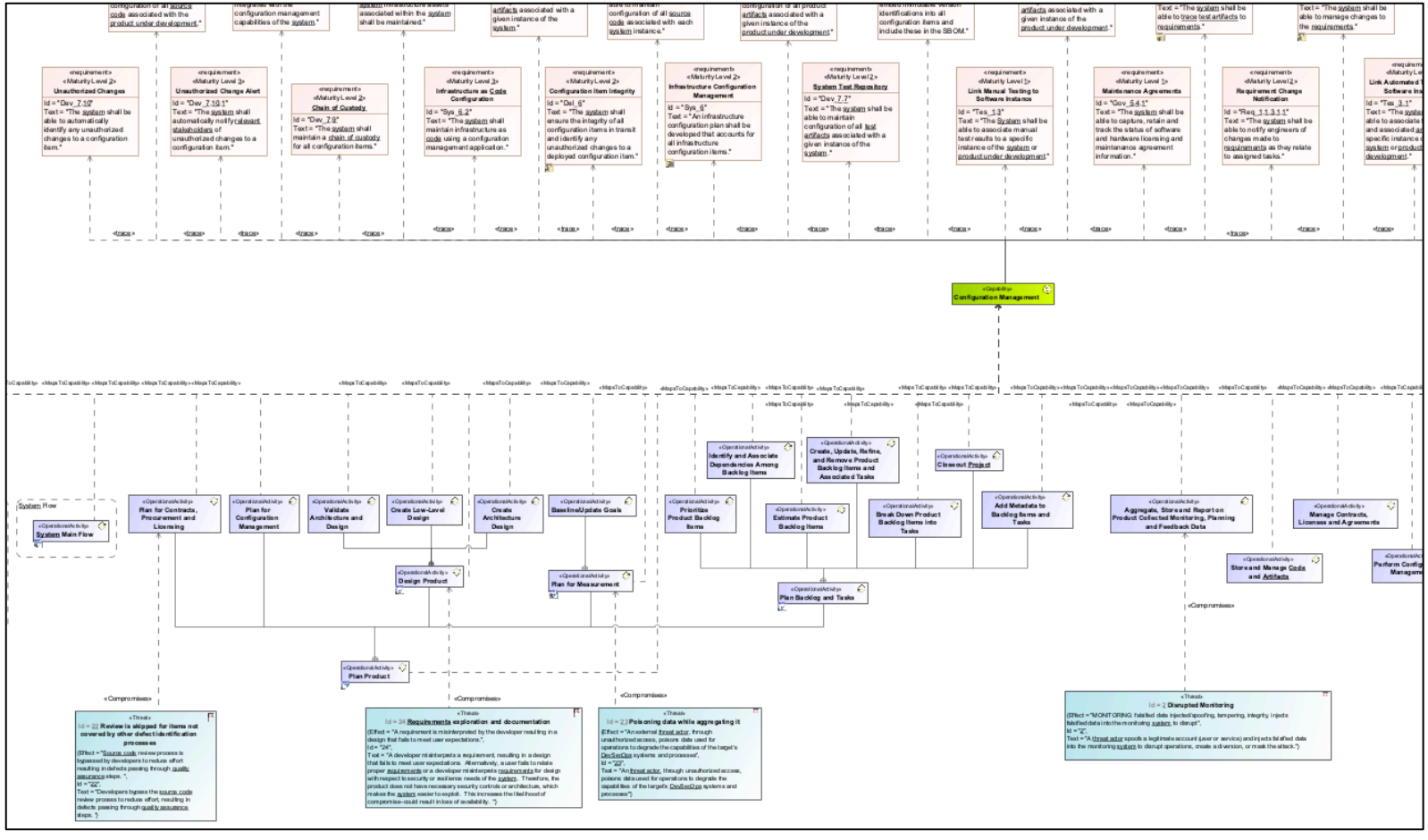
# Everyone Plays a Role in DevSecOps




- [Process Involvement Matrix Link]

Critical Roles are mapped to Operational Activities.

# Capturing the Theats of the DevSecOps System




Example of Threats Traced to Capabilities via Operational Activities

[Configuration Management Complexity Link]

# The DevSecOps PIM enables Organizations, Projects, Teams, and Acquirers to

- specify the DevSecOps requirements to the lead system integrators tasked with developing a platform-specific solution that includes the designed system and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the system evolves
- apply DevSecOps methods to complex products that do not follow well-established software architectural patterns used in industry
- provide a basis for threat and attack surface analysis to build a cyber assurance case to demonstrate that the product and DevSecOps pipeline are sufficiently free from vulnerabilities and that they function only as intended
- evaluate the capabilities of software factories

# Capability Maturity

Carnegie Mellon University
Software Engineering Institute

# Capability Maturity



- A maturity model is a set of characteristics, attributes, indicators, and patters that represent progression and achievement in a particular domain or discipline
- A maturity model allows an organization, or software factory, to have its practices, processes, and methods evaluated against a clear set of artifacts that establish a benchmark
- Capability maturity levels are arranged in an evolutionary scale that defines measurable transitions from one level of capability to another.
- Maturity models can be used to
  - Determine an organization's current level of capability and then apply these methods over time to drive improvements
  - Determine how well a program is performing by examining the capabilities of its sister programs.
- The SEI has been defining such models and associated appraisal methods for over 30 years.

# DevSecOps Capability Maturity



As a DevSecOps system matures, so will its capabilities

DevSecOps can be broken down into 10 capabilities

- These capabilities are groupings of requirements that, when combined, define a collective competency in performing a set of functional activities across the product lifecycle

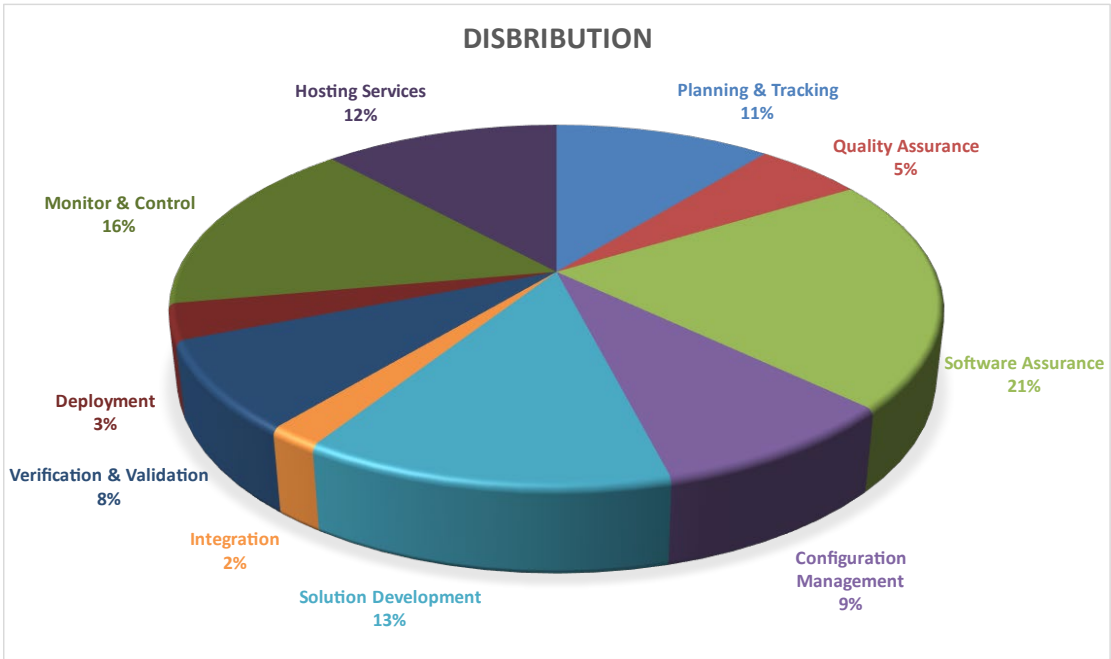
The capability levels represent the measure of consistency and completeness

- This is usually achieved through increased automation, in which functional activities are performed.

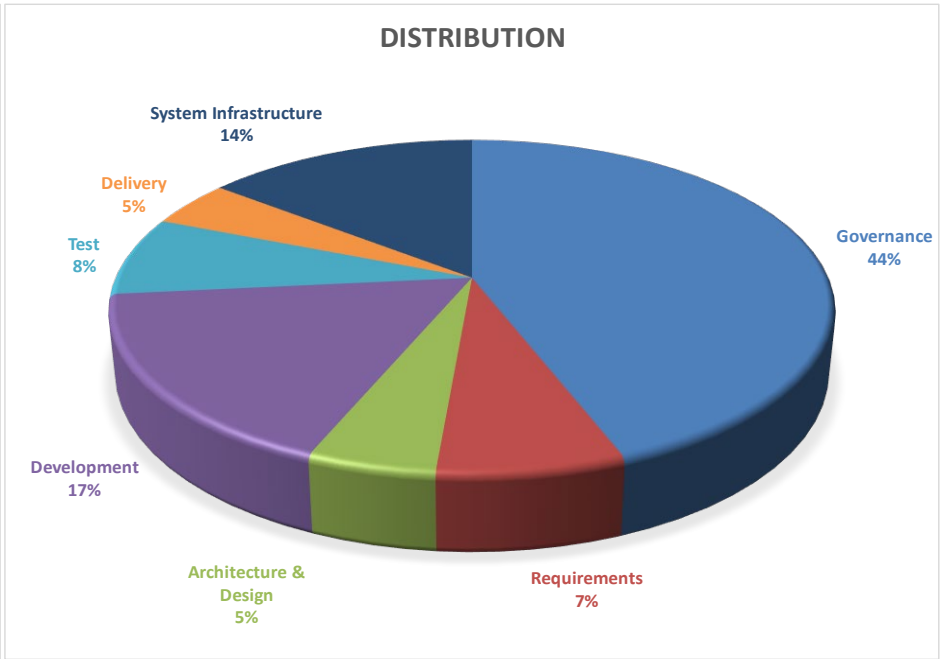
# DevSecOps Requirements

Carnegie Mellon University Software Engineering Institute

## 10 Capabilities




## Lifecycle View

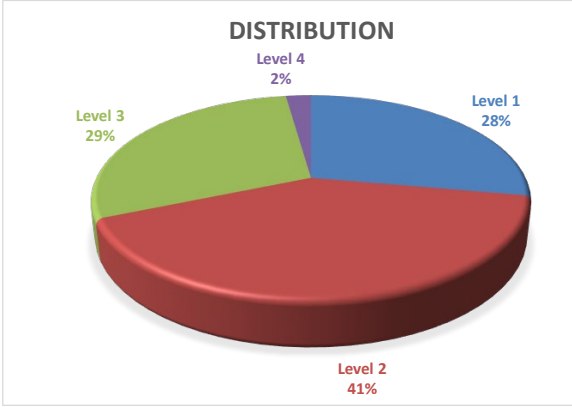



https://cmu-sei.github.io/DevSecOps-Model/

Distribution of over 200 defined DevSecOps Requirements

# DevSecOps Capabilities Evolve

| 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. |




Distribution of over 200 defined DevSecOps Requirements

https://cmu-sei.github.io/DevSecOps-Model/

# DevSecOps Maturity Levels



| Maturity Level | Title | Description |
|---|---|---|
| 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. |
| 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. |
| 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. |
| 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. |

Link to DevSecOps PIM

# DevSecOps Core Capabilities (1 of 3)

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Capability | Definition |
|---|---|
| Configuration Management | Configuration management is the set of activities used to establish and maintain the integrity of the system and product under development, and associated supporting artifacts throughout their useful lives. Different levels of control are appropriate for different supporting artifacts and implementation elements and for different points in time. For some supporting artifacts and implementation elements, it may be sufficient to maintain version control of the artifact or element that is traced to a specific instance of the system or product under development in use at a given time, past or present, so that all information related to a given instance, or version, is known. In that case, all other variations of the artifacts and elements can be discarded as subsequent iterations are generated or updated. Other supporting artifacts and implementation elements may require formal configuration, in which case baselines are defined and established at predetermined points in the lifecycle. Baselines, and subsequent changes, are formally reviewed and approved which will serve as the basis for future efforts. The configuration management capability of a system matures as the consistency and completeness of the integrity controls are put in place to capture all supporting artifacts and implementation elements associated with the system and product under development while keeping pace with the DevSecOps pipeline through automation and integration with all aspects of the lifecycle. This includes (1) monitoring the relationship between artifacts and elements for a given instance, or version, of the system or product under development, (2) capturing sufficient information to identify and maintain configuration items, even if those who created them are no longer available, (3) defining the level of control each artifact and element requires based on technical and business needs, (4) systematically controlling and monitoring changes to configuration items, and (5) enforcing and logging of all required relevant stakeholder reviews and approvals, based on the organization, project, and team policies and procedures. |
| Deployment | Deployment is the set of processes related to the delivery or release of the product under development into the environment in which users of the product interact with it. The deployment capabilities of the system mature with increased levels of automation and advanced rollback and release functionality. |
| Hosting Services | Hosting services are made up of the underlying infrastructure and platforms that both the system and product under development operate upon. This includes the various cloud providers, on premises bare-metal and virtualization, networks, and other software as a service (SaaS) that is utilized along with the management, configuration, access control, ownership, and personnel involved. |

# DevSecOps Core Capabilities (2 of 3)

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Capability | Definition |
|---|---|
| Integration | Integration is the process of merging changes from multiple developers made to a single code base. Integration can be made manually on a periodic basis, typically by a senior or lead engineer, or it can be made continuously by automated processes as individual changes are made to the code base. In either case, the purpose of integration is to assemble a series of changes, merge and deconflict them, build the product, and ensure that it functions as intended and that no change broke the whole product, even if those changes worked in isolation. |
| Monitor & Control | Monitor and control involves continuously monitoring activities, communicating status, and taking corrective action to proactively address issues and consistently improve performance. More mature projects automate as much of this as possible. Appropriate visibility enables timely corrective action to be taken when performance deviates significantly from what was expected. A deviation is significant if it precludes the project from meeting its objectives when left unresolved. Items that should be monitored include cost, schedule, effort, commitments, risks, data, stakeholder involvement, corrective action progress, and task and work product attributes like size, complexity, weight, form, fit, or function. |
| Planning & Tracking | Planning and tracking is the set of practices one uses to define tasks and activities. It also includes the resources one needs to perform those tasks and activities, achieve an objective or commitment, and track progress (or lack thereof) towards achieving the given objective. It provides the mechanisms required to inform relevant stakeholders where an effort currently is within the process and whether it is on track to provide the expected outcomes. These mechanisms allow relevant stakeholders to determine what has been accomplished and what adjustments or corrective actions need to occur to account for impediments and other unforeseen issues. Ideally, impediments and issues are proactively identified and addressed. Practices include documenting activities and breaking them down into actionable work to which one can assign resources, capturing dependence, forecasting, mapping work to requirements, collecting data, tracking progress to commitments, and reporting status. The planning and tracking capability of a system matures as the automation and integration of associated practices increases. |
| Quality Assurance | Quality assurance is a set of independent activities (i.e., free from technical, managerial, and financial influences, intentional or unintentional) designed to provide confidence to relevant stakeholders that the DevSecOps processes and tools are appropriate for and produce products and services of suitable quality for their intended purposes. It assumes that the organization's, team's, and project's policies and procedures have been defined based on all relevant stakeholder needs, which will result in a value stream that consistently produces products and services that meet all relevant stakeholder expectations. The quality assurance capability of a system matures as its ability to assess adherence to and the adequacy of the defined policies and procedures improves. |

# DevSecOps Core Capabilities (3 of 3)

Link to DevSecOps PIM

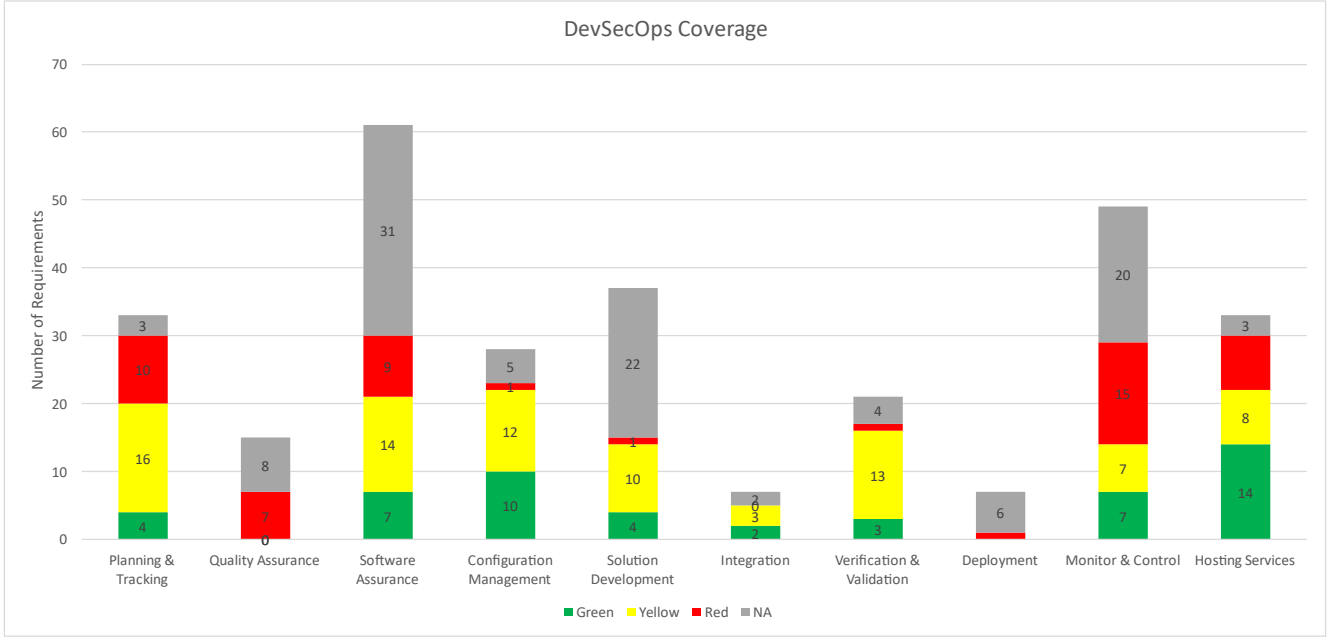
Carnegie Mellon University Software Engineering Institute

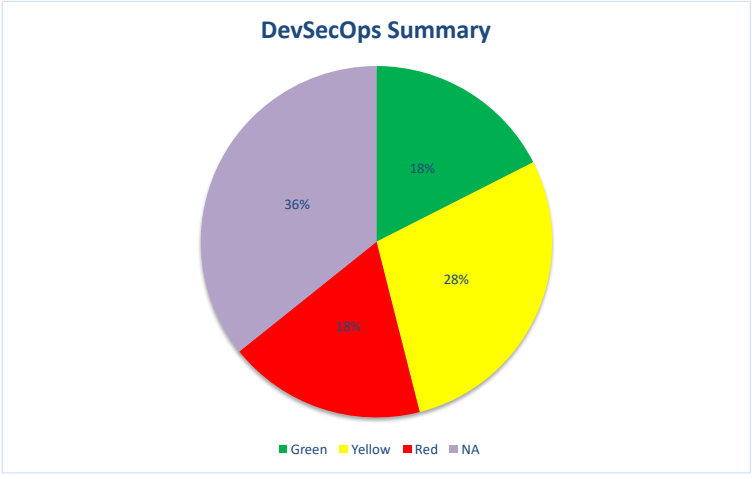
| Capability | Definition |
|---|---|
| Software Assurance | Software assurance is the level of confidence that software functions only as intended and is free from vulnerabilities either intentionally or unintentionally designed or inserted as part of the software throughout the full software lifecycle. It consists of two independent but interrelated assertions:<br><br>1. The software functions only as intended. It exhibits only functionality intended by its design and does not exhibit functionality not intended.<br>2. The software is free from vulnerabilities, whether intentionally or unintentionally present in the software, including software incorporated into the final system.<br><br>It is the responsibility of the DevSecOps system to ensure that software that meets the organization's threshold for software assurance is allowed to be deployed and operated. |
| Solution Development | Solutions development determines the best way of satisfying the requirements to achieve an outcome. Its goals are to evaluate baseline requirements and alternative solutions to achieve them, select the optimum solution, and create a specification for the solution. Each development value stream develops one or more solutions, which are products, services, or systems delivered to the customer, whether internal or external to the enterprise. |
| Verification & Validation | Verification and validation is the set of activities that provides evidence that the system or application under development has met the requirements and criteria that are expected. The scope includes the general realm of testing, verifying, and validating activities and matures as automation, feedback, and integration with other elements increase. |

# DevSecOps Capability Maturity Visualization

Carnegie Mellon University
Software Engineering Institute

# Capability Summary



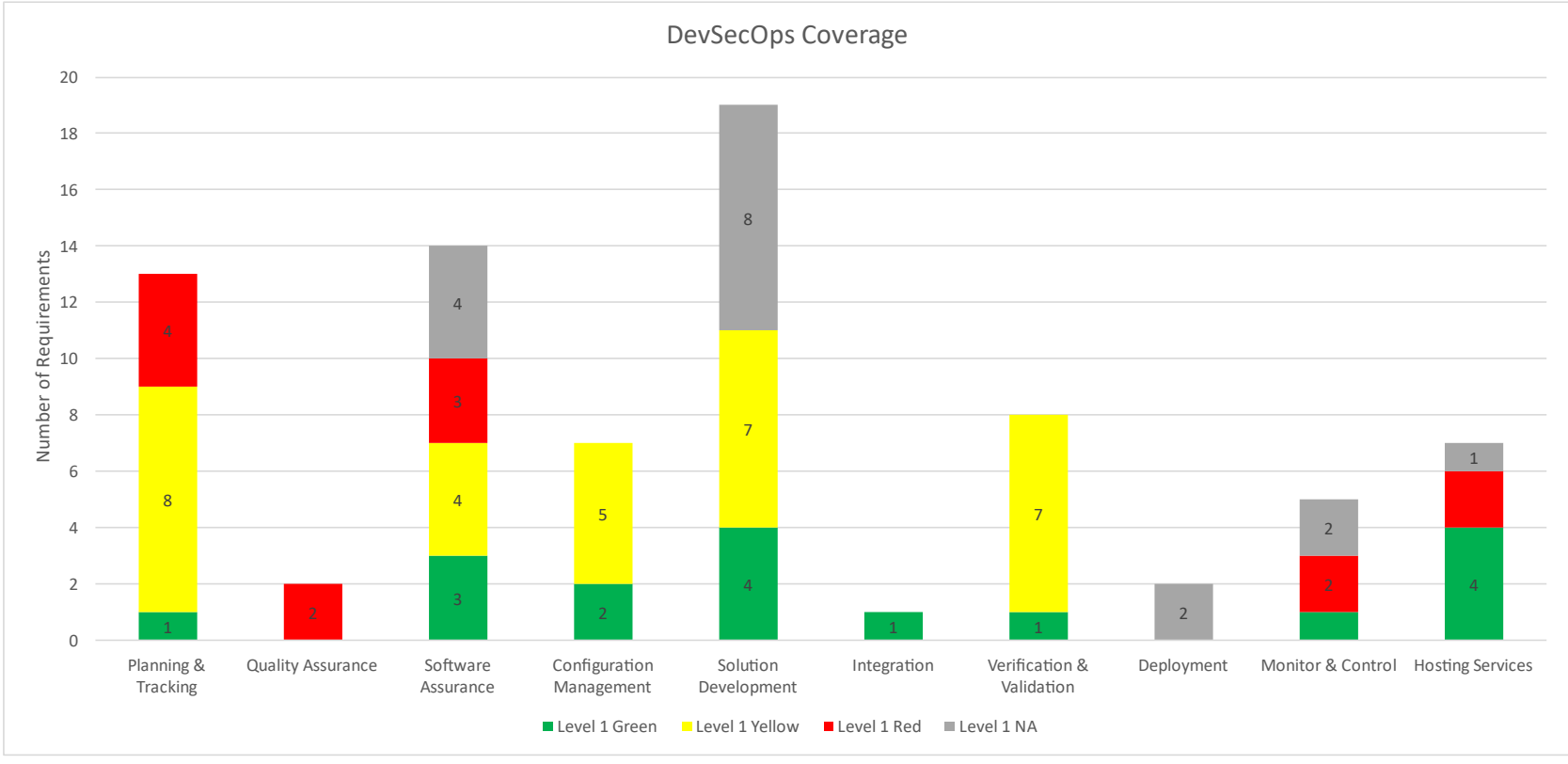




Over 200 Agile/DevSecOps requirements broken into 10 Capabilities and 4 Maturity Levels

| Key | Description |
|---|---|
| Green | Consistently Demonstrated |
| Yellow | Occasionally Demonstrated |
| Red | Insufficient Evidence Found |
| NA | Not Applicable to Program |

# Level 1

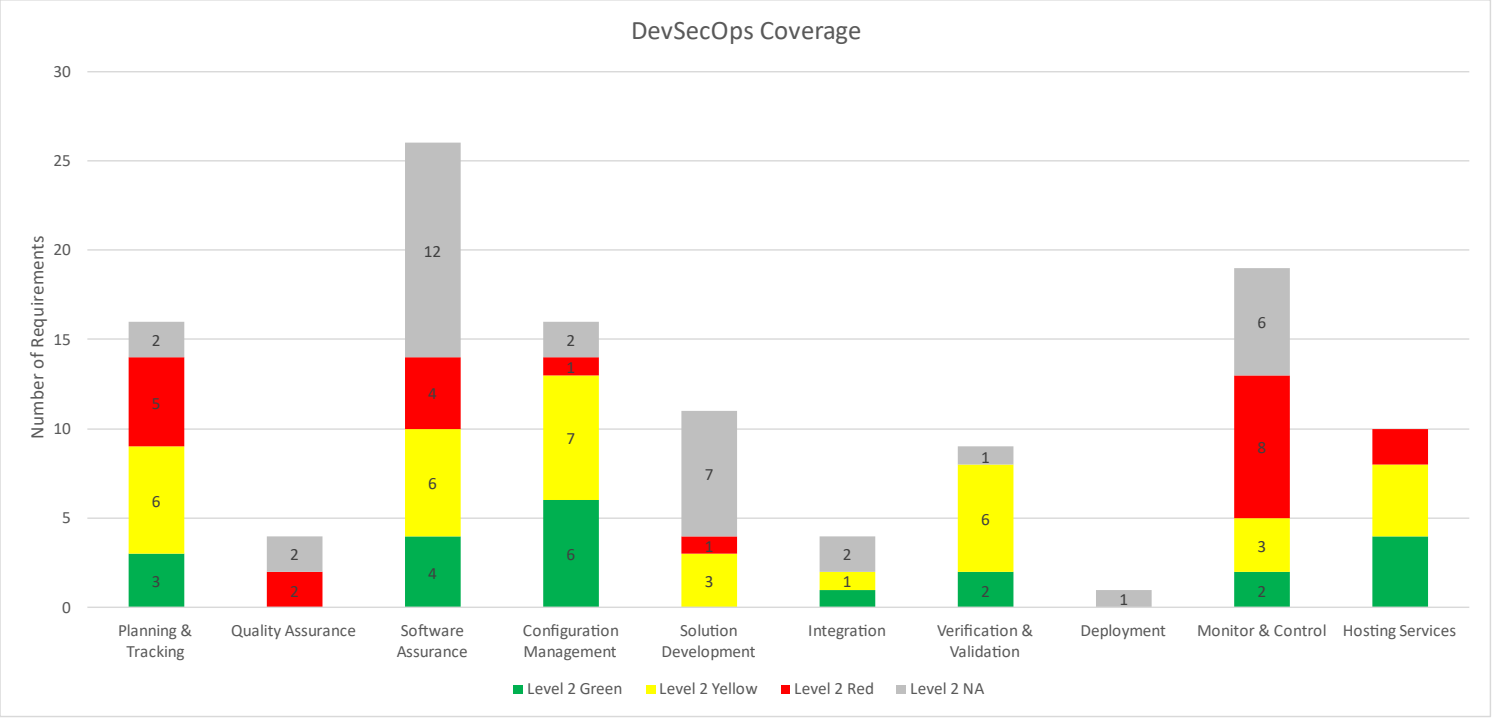
Performed Basic Practices: This represents the minimum set of engineering, security, and operational practices that is required to begin supporting a product under development, even if only performed in an ad-hoc manner with minimal automation, documentation, or process maturity. This level is focused on minimal development, security, and operational hygiene.




Over 60 Maturity Level 1 Agile/DevSecOps requirements broken into 10 Capabilities

# 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.




Over 80 Maturity Level 2 Agile/DevSecOps requirements broken into 10 Capabilities

# Summary

Enterprises need to be able to evaluate and articulate their capability to deliver value and the value of the Product/Service being being provided. These are 2 distinct, but integrated areas of concern.

The SEI DevSecOps PIM

- Bridges the gap between high-level DevSecOps concepts and actual instantiations
- Provide a single source of truth in which multiple perspectives and views can be analyzed
- Provides a basis for driving alignment across distinct pipelines within an enterprise
- Provides a roadmap to implementing DevSecOps
- Provides a repeatable approach to quantifying a pipeline's current capabilities

Through proper balance you should be able to play Topple indefinitely.

**Stop playing Whac-A-Mole!**

# Contact Information


**Timothy A. Chick**

CERT Applied Systems Group Technical Manager, CMU-Software Engineering Institute

Adjunct Faculty Member, CMU-Software and Societal Systems Department

tchick@sei.cmu.edu


https://www.sei.cmu.edu

https://s3d.cmu.edu

# Defining Each Capability's Maturity Levels

Carnegie Mellon University
Software Engineering Institute

# Configuration Management Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • All supporting artifacts and implementation elements that require configuration control are identified and documented.<br>• The level of configuration control for each supporting artifact and implementation element is defined.<br>• While the configuration management of supporting artifacts may be a fully manual process, an automated version control system, or set of systems, must be in place to track current and historical versions of files used to create implementation elements. |
| 2 | • Automated configuration management system(s) are in place for all identified supporting artifacts and implementation elements.<br>• Immutable logging is in place for all changes to configuration items and associated metadata, such as who made the change, when the change occurred, and what was changed.<br>• Changes to the system and product under development are associated with an approved requirement or change request.<br>• All relevant stakeholders are notified when changes to configuration items are requested.<br>• Some integration between the automated version control system used for file tracking and other aspects of the DevSecOps pipeline has occurred in order to enable the automatic triggering of other activities.<br>• The automated version control system traces relationships between test artifacts and requirements, and test results and associated artifacts, to a specific instance of the system or product under development in use at a given time, past or present. |
| 3 | • Manage and control the volatility of change. Be able to identify impacted supporting artifacts and implementation elements a given change request will impact.<br>• Use automatic discovery tools to scan current instance of system and product under development, and associated configurations, to identify mismatches between current instance and approved versions under configuration management in order to ensure integrity of the instantiated instances. Automatically report all mismatches to relevant stakeholders.<br>• The system shall automatically maintain an audit trail of all system configuration changes to include what was changed, who/what changed it, and when the change occurred.<br>• System only allows authorized individuals, or entities, to make specific types of changes to the product under development based on the individual's role, or entity's purpose, and where they are in the DevSecOps pipeline. |
| 4 | • Automatically correct any misconfiguration of the currently instantiated system and product under development based on approved supporting artifacts and implementation elements under configuration control.<br>• The system shall monitor user activities and actively identify security-related actions and system configuration changes that are uncharacteristic of the given user and notify relevant stakeholders of the uncharacteristic behavior to validate the change was appropriate and to avoid insider threats.<br>• A fully automated change proposal process is in place, where changes are proposed and automatically routed to relevant stakeholders for approval and implemented by the system. |

# Deployment Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • The system can manually recover if a failure occurs in a deployed product, deploying the product at the last known acceptable state. |
| 2 | • A quality criterion for the deployment of the system and product under development is defined.<br>• While monitoring for failures can be a combination of manual and automated detection processes:<br>- the system can automatically recover if a failure occurs in a deployed product, deploying the product at the last known acceptable state.<br>- the system can automatically recover the product to a previously working state in the event of system failure.<br>- the system can track the changes between deployed products and the personnel and reasoning involved in the change. |
| 3 | • Both the system and product under development are fully automated in terms of orchestration and deployment into target environments<br>• Various release strategies are supported to include canary, Blue-Green, multiple service, batch, rolling, and A/B testing.<br>• The product under development is deployed continuously, supported by sufficient automation in which no human intervention is required to release the product to its users.<br>• The system shall automatically collect the necessary data to monitor the system and product under development for failures and quality issues and alert relevant stakeholders when corrective actions are required.<br>• In the event that a failure or cancellation occurs during deployment of the product or system, the system will automatically restore a the most recent working version.<br>• Automated updating or patching of software used by the system. Patches are rolled out automatically to the various parts of the system. |
| 4 | • Continuous improvement of the testing procedures is performed based on the data collected from the system and product under development tests.<br>• The system shall automatically identify and track when the defined quality criteria have not been met and the automated quality controls have been bypassed. All relevant stakeholders will be automatically notified, and the noncompliance issue will be tracked to closure. |

# Hosting Services Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • The hosting services adequately support the scalability, reliability, regulatory, and security requirements to operate, maintain, and build an organization's product.<br>• The hosting services provide compatibility with the testing frameworks and tools utilized throughout system and product development lifecycles. |
| 2 | • Logs from hosting services are aggregated, auditable, and analyzable.<br>• System transaction logs are available and immutable.<br>• Performance metrics can be visualized and analyzed for hardware, software, database, and network components.<br>• Role-based access control is utilized throughout.<br>• All information collected uses proper techniques to mitigate privacy and sensitivity concerns, and can be properly disposed of when necessary.<br>• All configuration items are identified and resources are planned and executed in order to maintain configuration integrity of the given item.<br>• Disaster recovery processes are documented and supported. |
| 3 | • The system infrastructure is provisioned using IaC and is automated.<br>• Captured metrics can generate alerts based off of defined values.<br>• The system is able to automatically alert and communicate metrics associated with security risks of the underlying infrastructure to stakeholders so they can manage risk and make decisions regarding risk and impact to software applications.<br>• Automatic upgrading of operating system software and supporting services is in place. |
| 4 | • Qualities such as performance, capacity, security, compliance, and risk tolerance are continuously being monitored using automated tools. Results from the automated tools are automatically reported to all relevant stakeholders to ensure the quality of the automated process and to identify and track improvements to quality attributes.<br>• System configuration and performance are continuously being monitored using automated tools to identify and report all anomalies. Results from the automated tools are automatically reported to all relevant stakeholders so they can manage risk and make decisions.<br>• Infrastructure is immutable and can be automatically replaced versus update in place. |

# Integration Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • Documented, repeatable, processes exist that may be manual, automated, or some combination of the two.<br>• Some individual processes (e.g., merging changes) may require expert subjective judgement.<br>• Processes may require manual intervention between phases and/or to coordinate steps between disparate systems<br>• Some human-human and human-process contact occurs outside the orchestration pipeline.<br>• Process initiation is manual and irregular. |
| 2 | • Most individual processes are scripted and repeatable.<br>• Expert subjectivity has been removed from all processes by adopting processes with objective criteria for success.<br>• An orchestrated integration pipeline exists; however, it may not be fully automated.<br>• Some human-human and human-process contact occurs outside the orchestration pipeline.<br>• Integration process initiation is regular whether manual or automated. |
| 3 | • All individual processes are scripted and fully automated.<br>• An orchestrated integration pipeline controls all processes from start to finish.<br>• All human-process contact occurs from within the context of the orchestration pipeline (e.g., approvals captured in ticketing system, SCM, etc., and orchestration continues). |
| 4 | • The entire integration pipeline is fully automated, requiring no manual intervention.<br>• The entire integration pipeline runs in near real time as changes are committed to the code base.<br>• Alerts, notifications and results of integration are sent to relevant engineers automatically.<br>• A successfully integrated product is ready for delivery with no additional manual processes required. |

# Monitor & Control Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • All supporting artifacts and implementation elements that require monitoring and control are identified and documented.<br>• The level of monitoring and control for each supporting artifact and implementation element is defined.<br>• A policy and plan for planning and performing the monitor and control capability is established and maintained.<br>• The work products of the monitor and control capability are placed under appropriate levels of control. |
| 2 | • The people performing or supporting the monitor and control capability are trained as needed.<br>• Automated monitor and control system(s) are in place for all identified supporting artifacts and implementation elements.<br>• Automated collection of work products, measures, and measurement results are in place.<br>• Automated comparison of actual measurements to expected measurements is performed, and deviations are quantified.<br>• Automated alerting when significant deviations occur. |
| 3 | • The relevant stakeholders of the monitor and control capability are identified, involved, and are obtaining the information they need to make decisions.<br>• Sharing of monitor and control information to relevant stakeholders is automated.<br>• Stakeholders can tailor the visualizations of the information provided to meet their needs. |
| 4 | • The monitor and control capability is itself subject to being monitored and controlled and corrective action is taken when necessary.<br>• Automated collection of monitor and control capability work products, measures, measurement results, and improvement information, including records of significant deviation, criteria for significant deviation, and corrective action results, are in place.<br>• Root causes of defects and other problems in the monitor and control capability are identified and corrected.<br>• Monitor and control capability is itself subject to continuous improvement. |

# Planning & Tracking Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | Manual practices are used, with possible use of some rudimentary tools, that collect and store information used to track and report status and outputs from planning and tracking activities. |
| 2 | • Planning and tracking tools are used to define tasks and activities, along with the resources needed to perform them and achieve an objective or commitment, and track progress, or lack thereof, towards achieving the given objective.<br>• The tools provide the ability to capture and associate planning and tracking metadata, such as estimates, assumptions, prioritization, assignment, status, commitments, assets, association to implementation elements and supporting artifacts, and agreements. Metadata may consist of mostly manually collected information, with minimal automation.<br>• Automated visualization techniques are used to organize activities, understand dependencies, coordinate multiteam efforts, and road map future commitments. The automated system is used to share project plans and status of current activities with relevant stakeholders. |
| 3 | • The planning and tracking tools are able to coordinate multiple value streams at the organizational level. Planning and tracking activities are integrated to include both technical and non-technical activities, such as quality assurance, documentation, testing, and configuration management. Dependencies between technical and non-technical activities can be visualized in order to coordinate efforts and identify issues.<br>• Metadata is used to support estimation, projections and what-if scenarios simulations. Organizations, projects, and teams are able to customize metadata, and associated use, in order to meet relevant stakeholder needs.<br>• The planning and tracking tools are integrated with other tools in order to automatically collect metadata associated with various value stream activities. This includes defect, issues, and noncompliance efforts as they are automatically discovered and subsequently addressed and tracked to closure and asset management.<br>• Automated stakeholder notification and status reporting, and associated visualizations, are used to notify relevant stakeholders of changes to plans or commitments, status of current activities, deviations from defined thresholds, and asset renewals and maintenance. |
| 4 | Data is used to<br>• apply statistical analytical methods to planning and tracking practices in order to improve and optimize the team's, project's, and organization's ability to meet objectives and commitments<br>• provide objective quantitative status to relevant stakeholders<br>• automatically generate tasking and execute processes based on plan |

# Quality Assurance Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering

| Level | Description |
|---|---|
| 1 | • All relevant stakeholders associated with the products and services associated with the product under development and the system that supports it have been identified.<br>• All relevant stakeholder expectations and regulatory requirements are documented.<br>• Policies and procedures are developed and documented to describe how the DevSecOps processes and tools are required to be used in order to meet all relevant stakeholder requirements.<br>• Documented policies and procedures may use a traditional document-centered approach, and dissemination may be a manual process.<br>• All current policies and procedures are readily available to all personnel. |
| 2 | • Automated tools are used to maintain configuration control of policies and procedures.<br>• All relevant stakeholders are automatically notified of changes to policies and procedures.<br>• Independent resources have been identified and a plan exists to review or audit activities that have been defined within the documented policies and procedures.<br>• DevSecOps processes and tools are periodically audited based on the plan to identify noncompliance with policies and procedures and inadequacies regarding the value stream's ability to consistently produce products and services that meet all relevant stakeholders' expectations and regulatory requirements. The audits may be conducted manually, use automation, or a combination of both.<br>• All identified noncompliance and inadequacies are independently documented, reported to relevant stakeholders, and tracked to closure. |
| 3 | • DevSecOps tools are configured to automatically enforce policies and procedures as a product under development progresses through the system.<br>• Automated processes are monitored by an independent resource in order to detect and report noncompliance issues to all relevant stakeholders.<br>• Noncompliance and inadequacy issues identified through automated or manual auditing are documented and tracked to closure using an automated issue tracking system that is consistent with the tools used for all other planning and tracking purposes, in order to integrate all efforts that must be planned and tracked to completion.<br>• All quality assurance tools, such as origin and static analysis tools, are fully integrated into the system's pipeline, and associated policies are automatically enforced as the product under development progresses through the system.<br>• The system automatically monitors and enforces compliance to defined quality criteria as defined for both the product under development and the system regarding the implementation of enhancements and modifications. |
| 4 | • All automated activities are continuously being audit for noncompliance issues through the use of automated tools, with regards to both the system and product under development.<br>• Results from the automated auditing tools are automatically reported to all relevant stakeholders to ensure the quality of the automated auditing process, in addition to tracking noncompliance issues to resolution.<br>• The system automatically identifies and tracks when the defined quality criteria have not been met or the automated quality controls have been bypassed. All relevant stakeholders will be automatically notified and the noncompliance issue will be tracked to closure. |

# Software Assurance Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • All relevant stakeholders and expectations with regards to the products and services associated with the product under development and the system that supports it have been identified.<br>• System functional and nonfunctional requirements are documented.<br>• A comprehensive software bill of materials (SBOM) is compiled detailing all components that make up the DevSecOps system.<br>• All relevant system constraints and regulatory requirements are documented.<br>• Software assurance processes and tools are inventoried, and policies and procedures are written setting out how they are to be used to meet assurance requirements.<br>• Documented policies and procedures may use a traditional document-centered approach, and dissemination may be a manual process. |
| 2 | • Software assurance related to DevSecOps metrics are defined and collected.<br>• Baseline and threshold levels for software assurance are established.<br>• Metrics are tracked over time and made available to all stakeholders as needed.<br>• Results of system functional testing are collected and periodically analyzed.<br>• Known vulnerabilities in all components that make up the DevSecOps system are periodically collected and analyzed.<br>• Processes and policies are in place to periodically compare present metrics to past and make adjustments as necessary.<br>• Processes and policies are in place and reviewed periodically.<br>• Reports are reviewed from all software assurance products.<br>• Processes and policies are in place to identify when the level of software assurance implied by captured metrics and reports exceeds the organization's threshold and to make adjustments as necessary. |
| 3 | • The organization has established a comprehensive risk analysis and management program.<br>• Software assurance metrics, reporting, and analysis are incorporated into the risk management process.<br>• Results of the risk management process are incorporated into software assurance policies and procedures.<br>• Software assurance metrics and thresholds are periodically updated as a result of risk management activities.<br>• The organization prioritizes software assurance tasks based on the level of risk to the organization. |
| 4 | • All software assurance tools, or as many as are feasible, are run continuously and reports are disseminated automatically to all relevant stakeholders.<br>• Software that fails to meet the organization's software assurance thresholds is automatically prevented from being delivered or deployed.<br>• Automated procedures are in place to remediate software assurance issues found within the operating DevSecOps system. |

# Solution Development Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • All development activities and tools have been identified and documented.<br>• Tools are provided to enable users to edit, compile, and review source code.<br>• The ability for developers to trace links between requirements, architectural elements, and implementation elements is provided.<br>• A repository for all requirements and associated metadata is provided.<br>• Manual processes for assuring security and privacy compliance are defined and used.<br>• Processes for transitioning between development components are defined and used.<br>• Individual processes are scripted and repeatable.<br>• Processes may require manual intervention between phases and/or to coordinate steps.<br>• Process initiation may be manual or automated. |
| 2 | • Transitions between implementation elements, and supporting artifacts, are automated, possibly manually triggered.<br>• Secure coding practices and development coding standards are identified and documented.<br>• Traceability of software code origins is provided to provide a SBOM and verify use of most recent third-party components. |
| 3 | • Transitions between development components are fully automated, either triggered on a periodic schedule or automatically triggered based upon completion of another component's activity.<br>• Determination of requirement feasibility and validation analysis is supported.<br>• Model-based software engineering in order to provide continuous, iterative, and traceable requirements model is supported.<br>• Policy as code (e.g., Security Technical Implementation Guide (STIG) enforcement) is supported. |
| 4 | • Transitions between development components is performed continuously without human intervention.<br>• Code commits are continuously audited, with alerts to relevant stakeholders.<br>• "Digital twin" modeling of the production system is enabled.<br>• Advanced analysis to ensure compliance is supported. |

# Verification & Validation Capability Levels

Link to DevSecOps PIM

Carnegie Mellon University Software Engineering Institute

| Level | Description |
|---|---|
| 1 | • All relevant stakeholders, with regards to the products and services associated with the product under development and the system that supports it, have been identified.<br>• All testing cases, procedures, and their artifacts are configured, stored, and maintained for a given instance of a product under development.<br>• The system and product under development support the necessary technologies to execute tests. |
| 2 | • Automated tools are used to trace tests to requirements.<br>• Automated tools are used to trace tests cases and artifacts to specific versions of a product under development.<br>• Automated tools are used to configure, store, and execute tests.<br>• Test coverage reports are generated and captured for a specific instance of the system or product under development.<br>• Tests are performed across multiple phases of the software lifecycle, such as development, test, and operations, providing feedback continuously.<br>• Security patching is automatically tested, resulting in automated report generation and delivery.<br>• Both functional and nonfunction tests are manually or automatically executed. |
| 3 | • Tests are executed automatically using a continuous integration technique.<br>• An MBSE approach is used to plan and execute testing of the system and product under development.<br>• The system and product under development automatically execute quality tests that either passes or fails the appropriate component under test based on quality metrics for any change being made. Appropriate monitoring of the system and product under development enforces the quality metrics.<br>• The system provides the necessary environment to perform advanced security testing such as fuzz and penetration testing activities. |