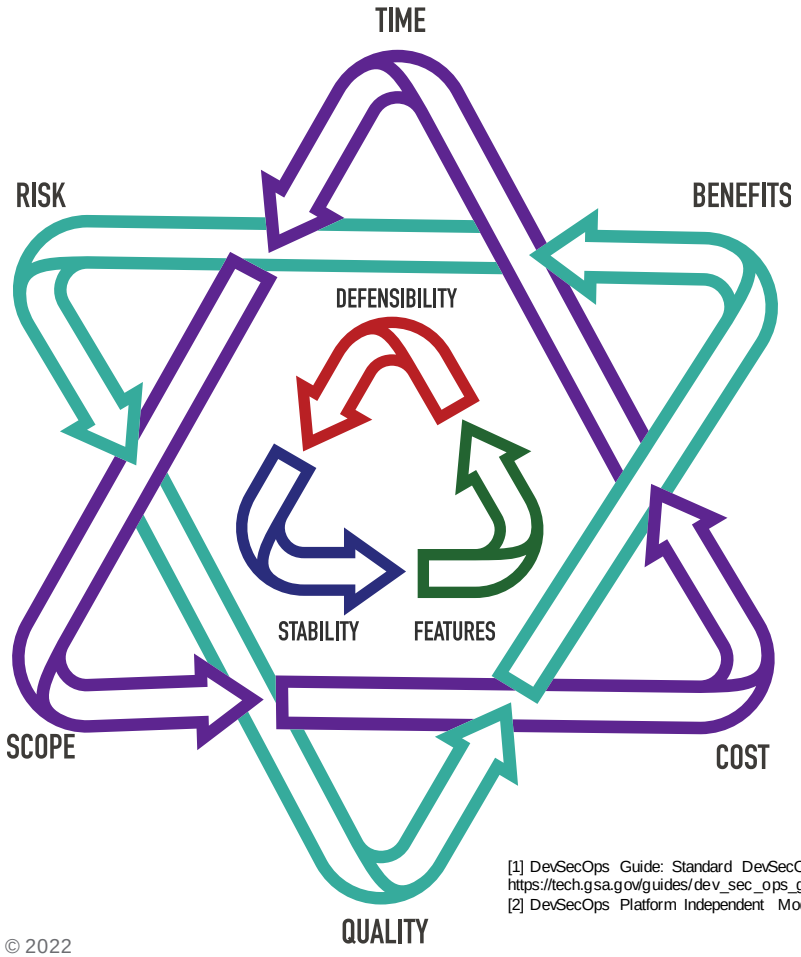
DevSecOps in Risk-Averse Environments

Timothy A. Chick
CERT Systems Technical Manager, CMU-Software Engineering Institute
Adjunct Faculty Member, CMU-Institute for Software Research

Carnegie Mellon University
Software Engineering Institute

Document Markings

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



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

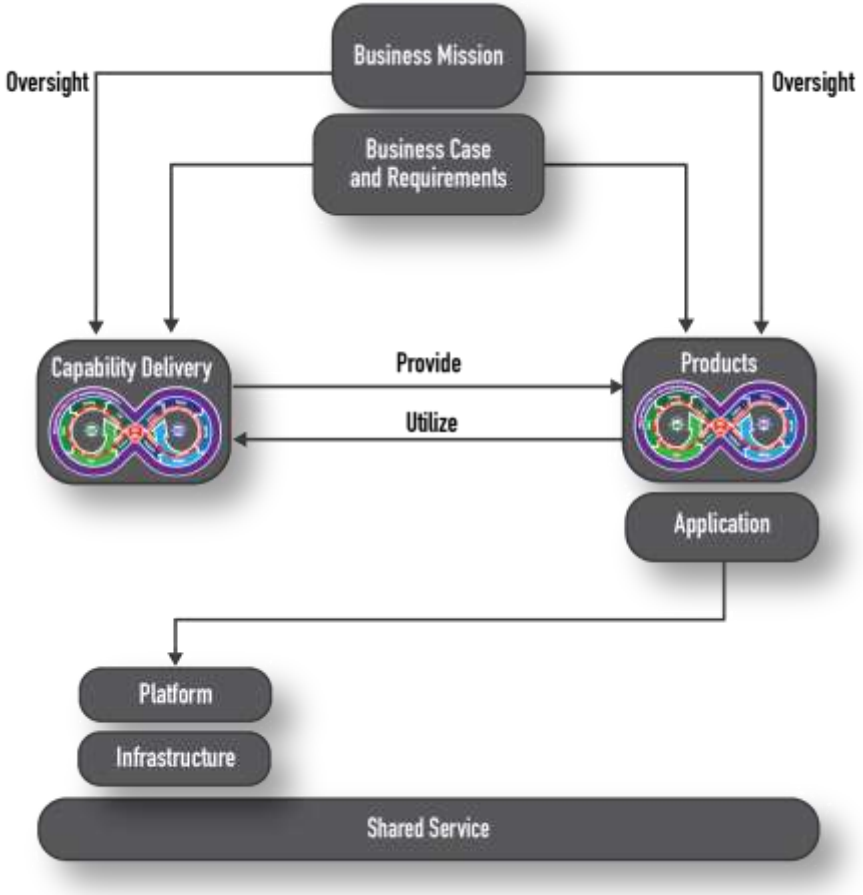
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

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

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

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

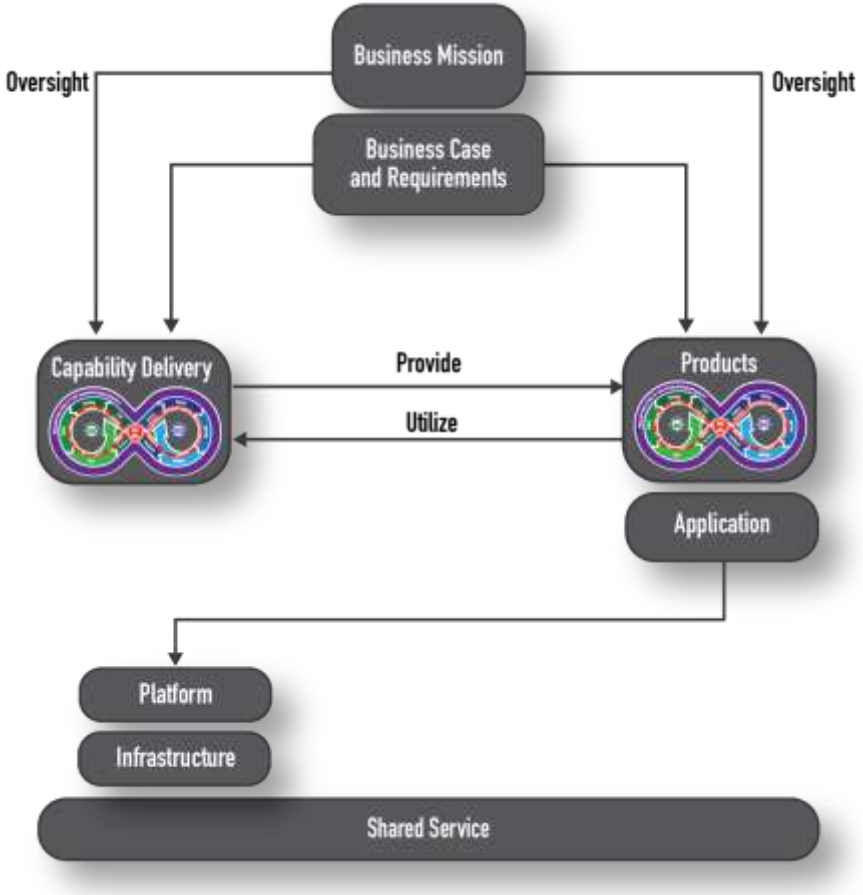
An Enterprise View



All DevSecOps-oriented enterprises are driven by three concerns:

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

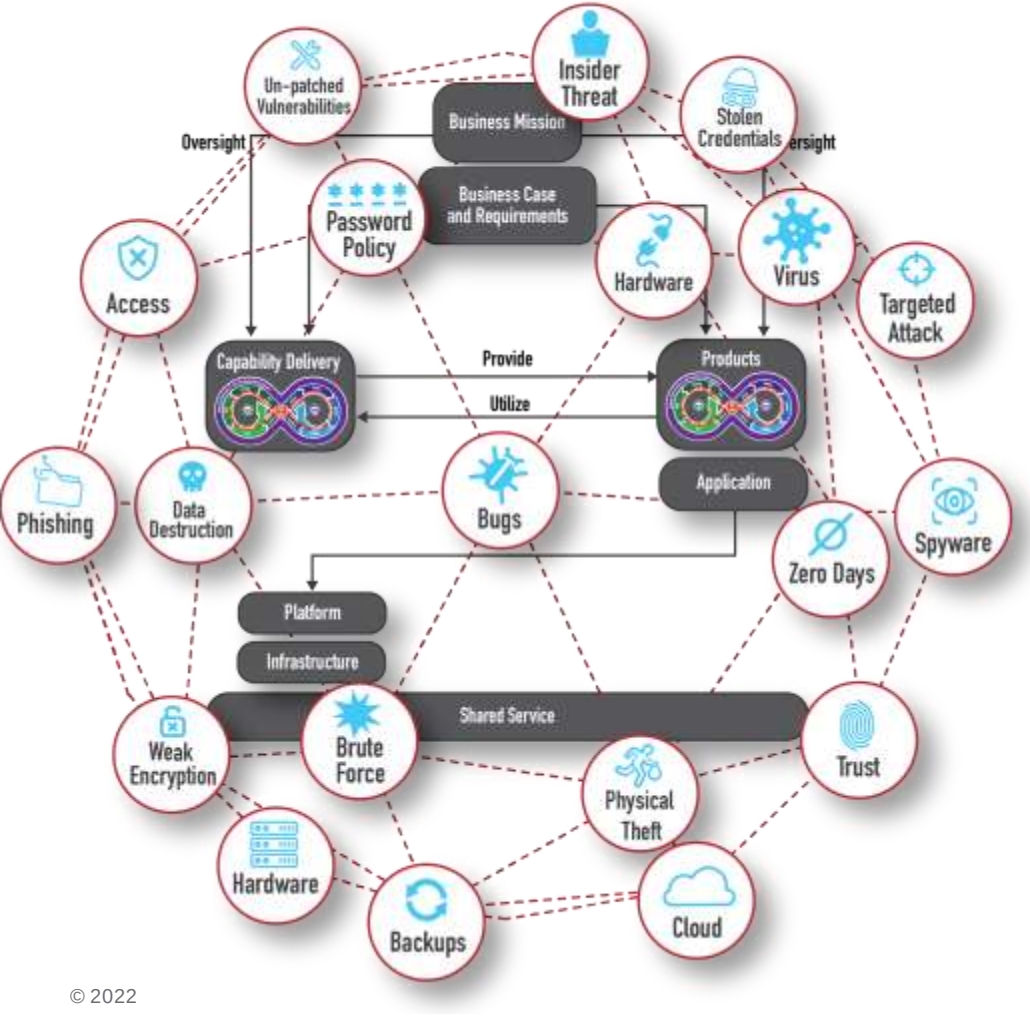
Challenge 1 for DevSecOps: connecting process, practice, & tools



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

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

Challenge 2 for DevSecOps: cybersecurity of pipeline and product



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

Managing and monitoring all of the various parts to ensure the product is built with sufficient cybersecurity and the pipeline is maintained to operate with sufficient cybersecurity is complex.

How do you focus attention to areas of greatest concern for security risks and identify the attack opportunities that could require additional mitigations?

All Techniques are Wrong, but Some are Useful[1]

George Box is famously quoted as saying, “All models are wrong but some are useful.” The same can be said for the various Agile and DevSecOps methods, as much of the material around Agile and DevSecOps assumes a simplification or idealization of a model development team.

The key to successful Agile and DevSecOps implementation is understanding how you will instantiate the Agile manifesto, Agile principles and DevSecOps principles.

The principles have implications for the characteristics of the lifecycle that can be used.

But there’s still more than one valid way of implementing the principles….

All Techniques are Wrong, but Some are Useful$_2$

The family of Agile and DevSecOps methods has grown since 2000 to incorporate techniques that address team, project, and enterprise levels of scaling.

Hybrids of multiple methods and techniques are common practice in both industry and government.

This is one reason it's so difficult to say a program is "Agile" or "doing DevSecOps correctly," or not.

To succeed, you must select the correct techniques, regardless of chosen methods, to meet your organization's and customer's goals, objectives, and missions.

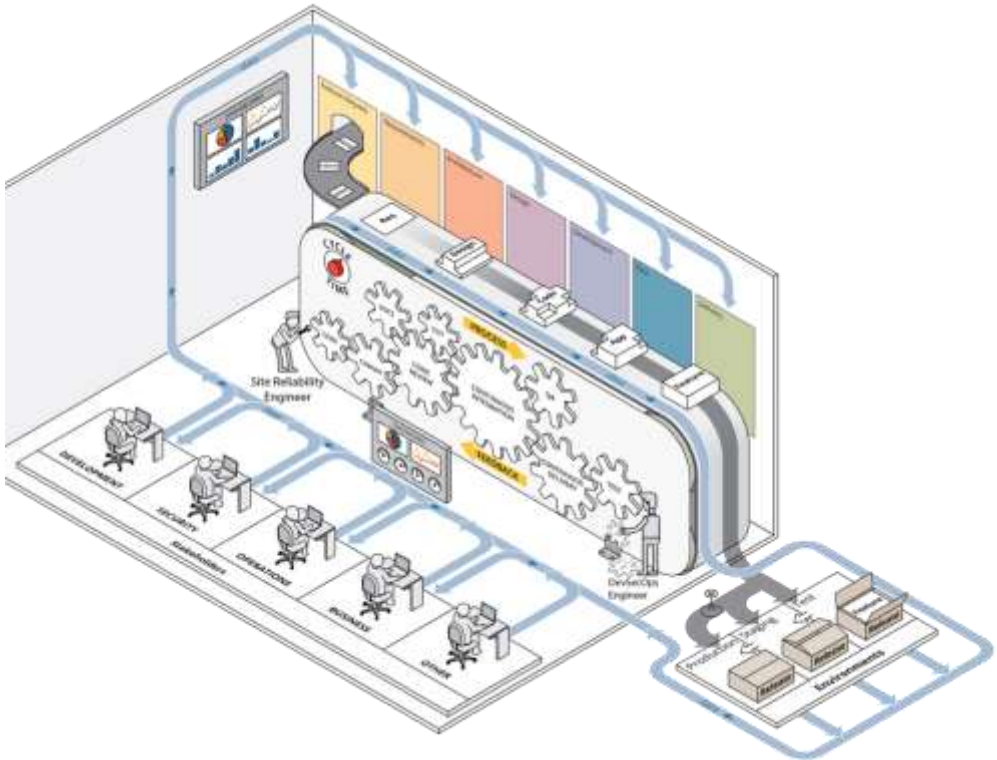
Selecting the Appropriate Techniques

Three Fundamental Factors

1. Identifying **the ability of the organization** to adopt new techniques
   - Successful adoption requires the absorption of associated costs, as well as expending the required time and effort.
2. Determining **the suitability of Agile and DevSecOps practices in the development** of a given product or system
   - Development and product characteristics play a large role in determining the suitability of a particular agile technique.
   - The desired product qualities also play a role in determining appropriate agile technique
3. Determining **the suitability of Agile and DevSecOps practices for the organization** developing the product or system.

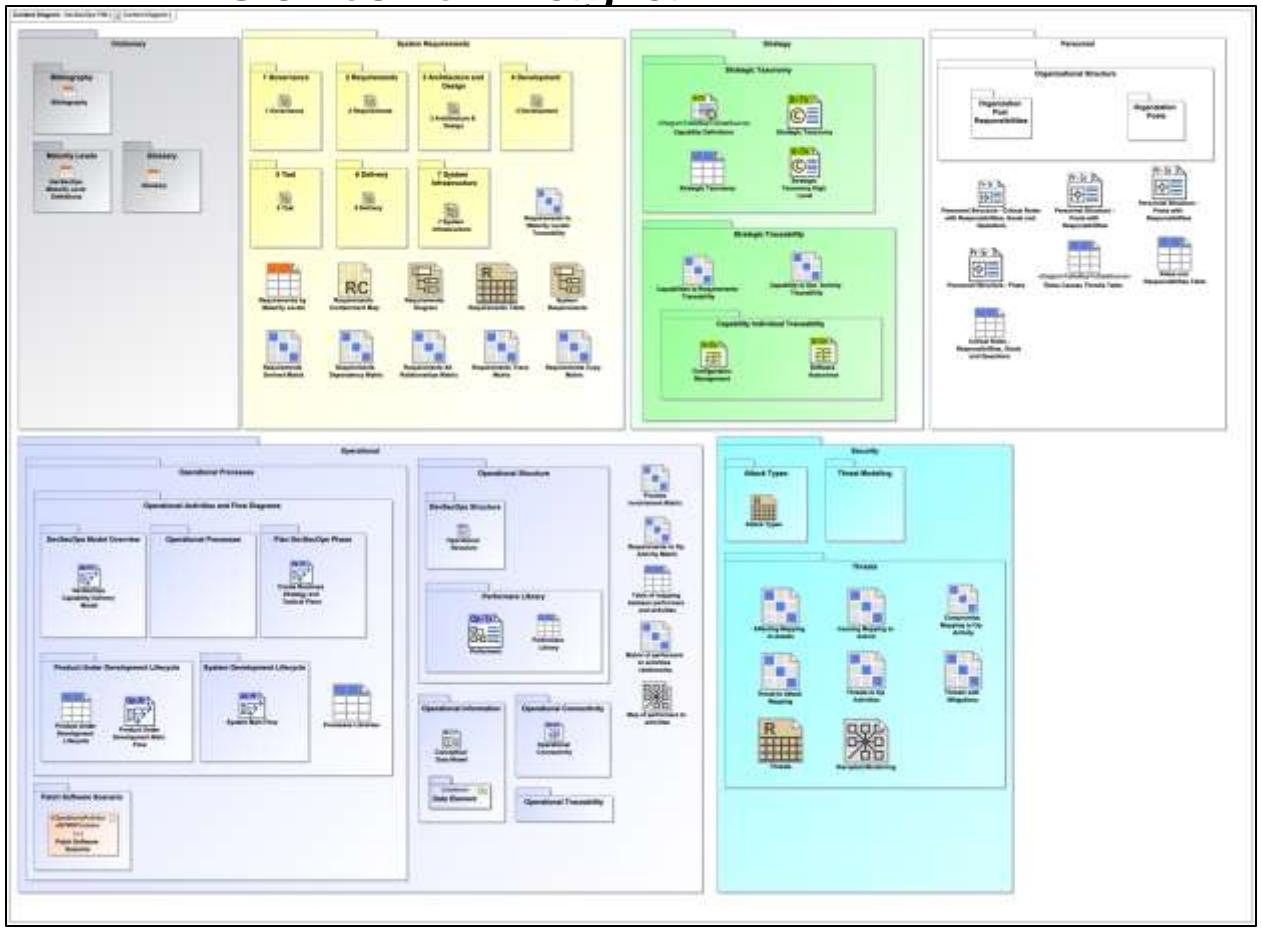
Adapted from Sidky, Ahmed; James Arther, *Determining the Applicability of Agile Practices to Mission and Life-critical Systems,* Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). pp 3-12.

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 in order 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/

The DevSecOps PIM Provides

Carnegie Mellon University Software Engineering Institute

- **Consistent guidance** and modeling capability that ensure all proper layers and development concerns relevant to the organization's, project's, and team's needs are captured
- The basis for creating a DevSecOps platform-specific model (PSM) that can be incorporated into the product's model-based engineering approach as the DevSecOps master model is included in the product's model. This allows **proper modeling of DevSecOps design trades within a project's Analysis of Alternatives (AoA) processes**, resulting in less costly and more secure products.
- The **basis for metrics and documentation of trade-offs** to be captured and analyzed through the model-based engineering approach. The model provides dynamic matrices of if those points were addressed, how they were addressed, and how well the corresponding (to the points) module is covered.
- **The basis for performing risk modeling against decisions and DevSecOps model-based engineering to ensure security controls and processes are properly selected and deployed**

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

- specify the DevSecOps requirements to the lead system integrators who need to develop a platform-specific solution that includes the designed system, simulation/testing platforms, and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the system evolves
- apply DevSecOps principles to complex systems that do not follow well-established software architectural patterns used in industry
- provide a basis for threat and attack surface analysis that can form the basis of a cyber assurance case for structuring evidence to demonstrate that a system and DevSecOps pipeline functions only as intended
- confirm the selected platform-specific solution has sufficient cyber assurance

Deeper Dive into DevSecOps Challenges

Carnegie Mellon University Software Engineering Institute

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

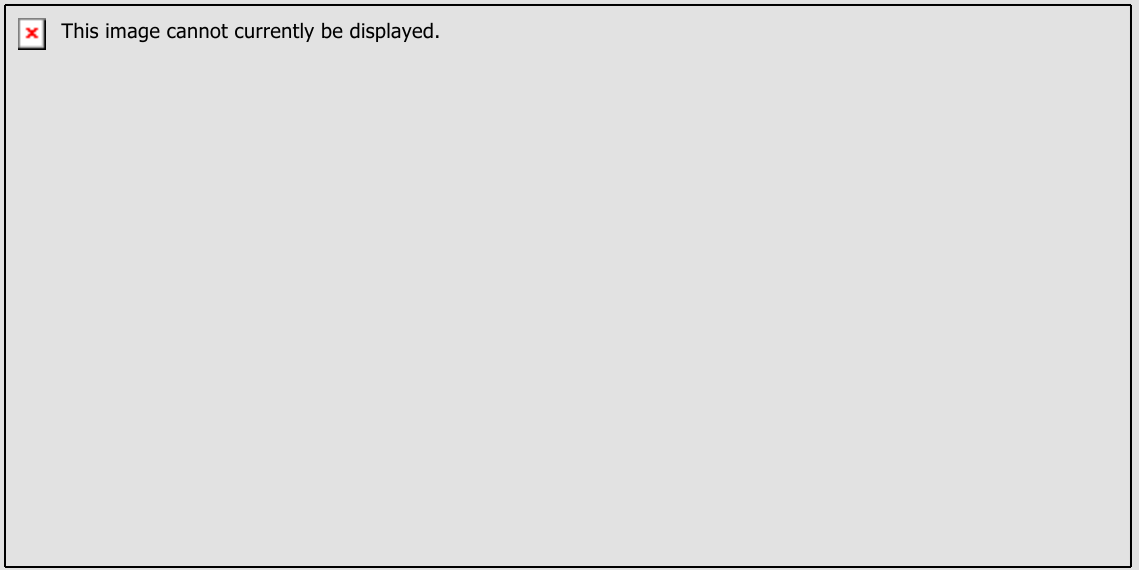
Challenge 2 for DevSecOps: cybersecurity of pipeline and product

Questions?

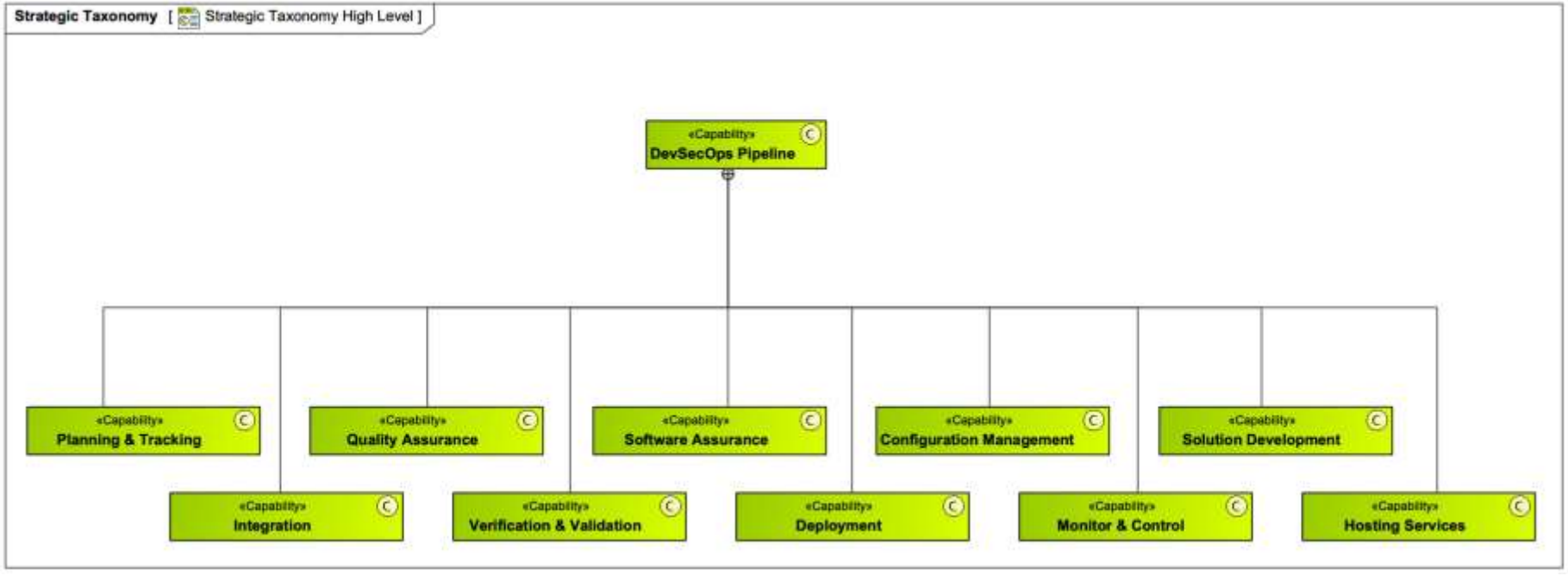
Through proper balance, programs should be able “to maintain a constant pace (i.e., play Topple) indefinitely” that results in a system that is:

- Trustworthy - No exploitable vulnerabilities exist, either maliciously or unintentionally inserted
- Predictable - When executed, software functions as intended and only as intended
- Timely – Features are delivered as the speed of relevance

Connecting Process, Practice, & Tools



As a DevSecOps system matures, so will its capabilities



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

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.

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

Areas of Focus

Carnegie Mellon University Software Engineering Institute

Lifecycle:

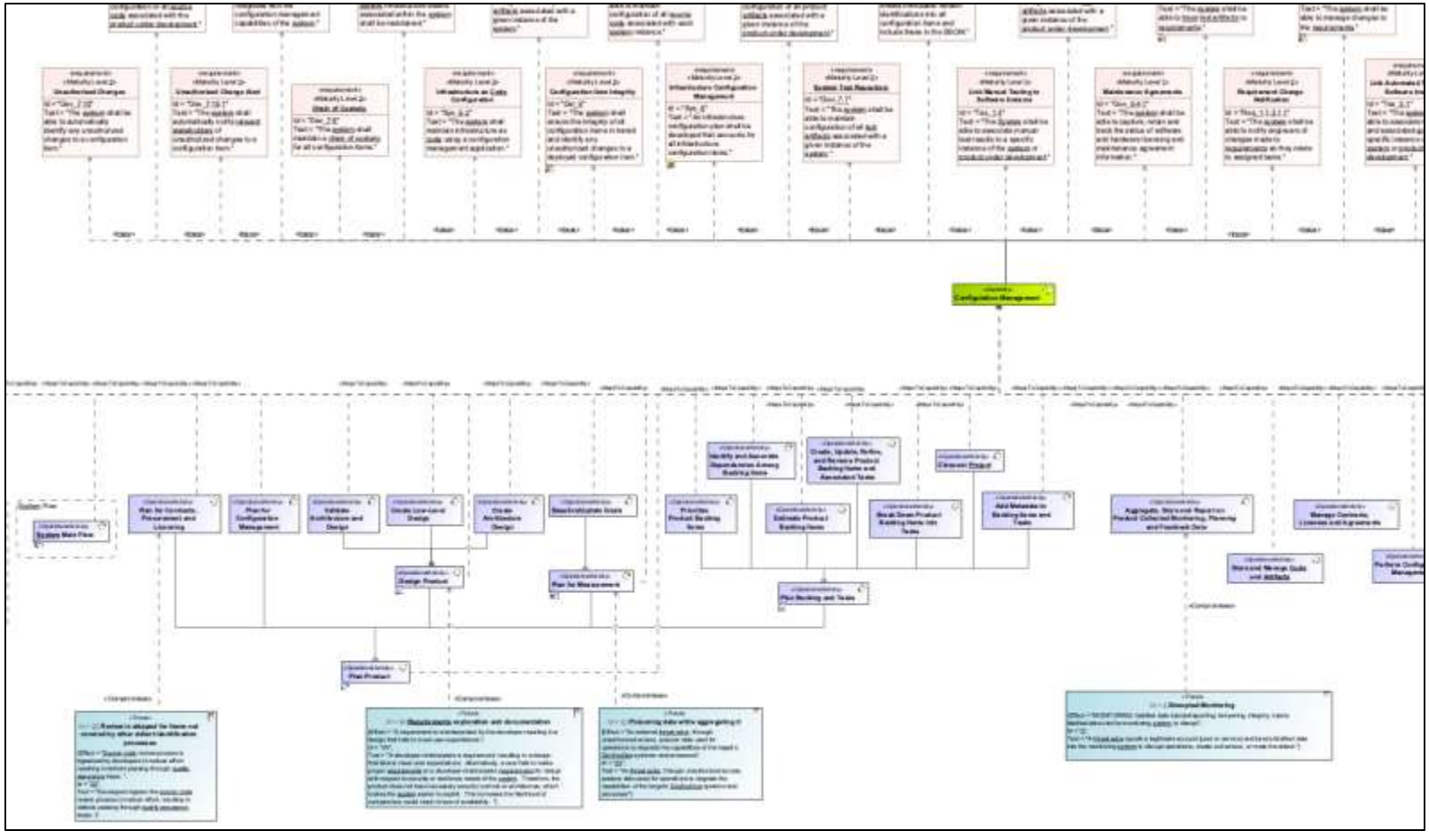
- Governance
- Requirements
- Architecture & Design
- Development
- Test
- Delivery
- System Infrastructure

Capabilities:

- Configuration Management
- Deployment
- Hosting Services
- Integration
- Monitor & Control
- Planning & Tracking
- Quality Assurance
- Software Assurance
- Solution Development
- Verification & Validation

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

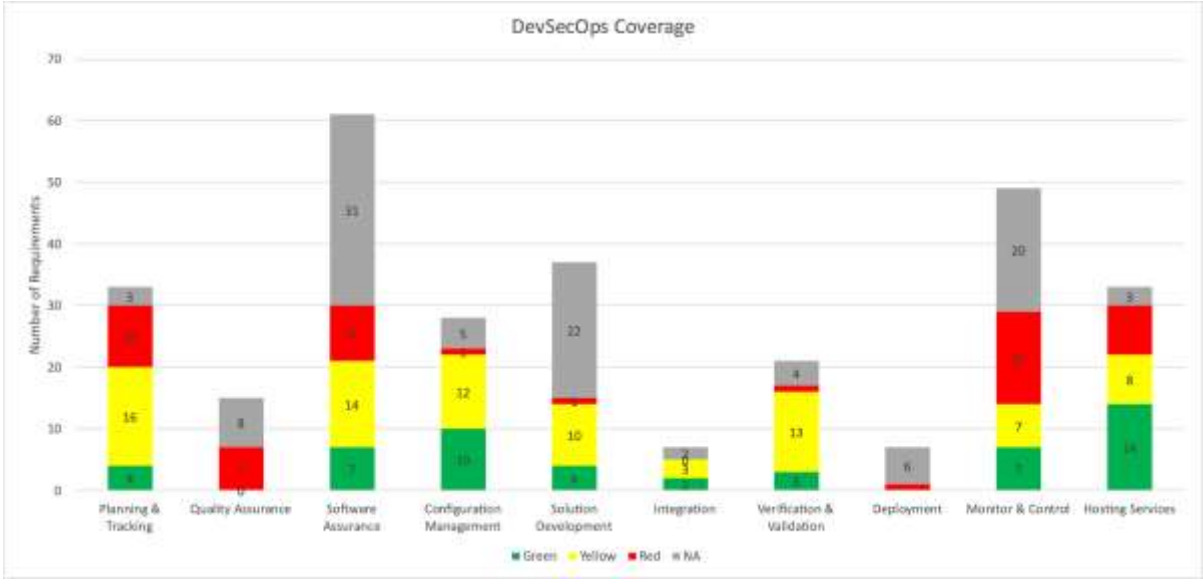
DevSecOps is a complex interconnected system

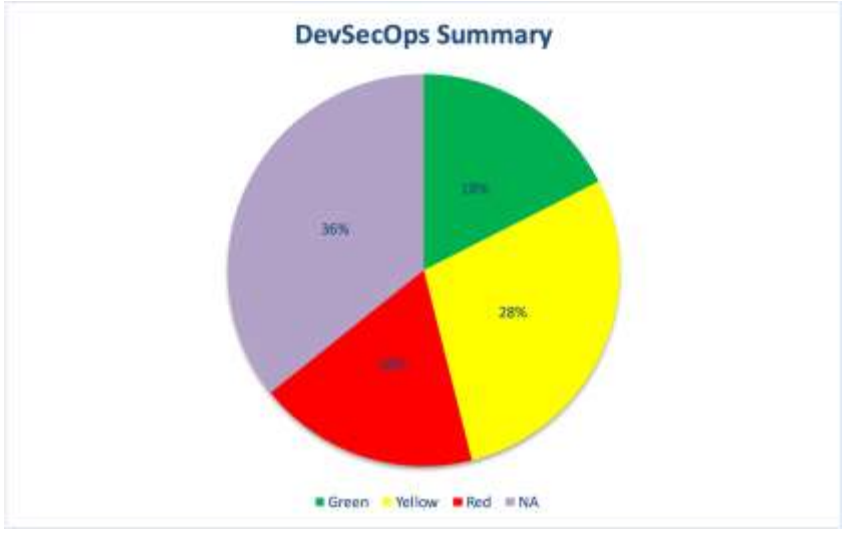


Example of Threats Traced to Capabilities via Operational Activities

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

Capability Summary

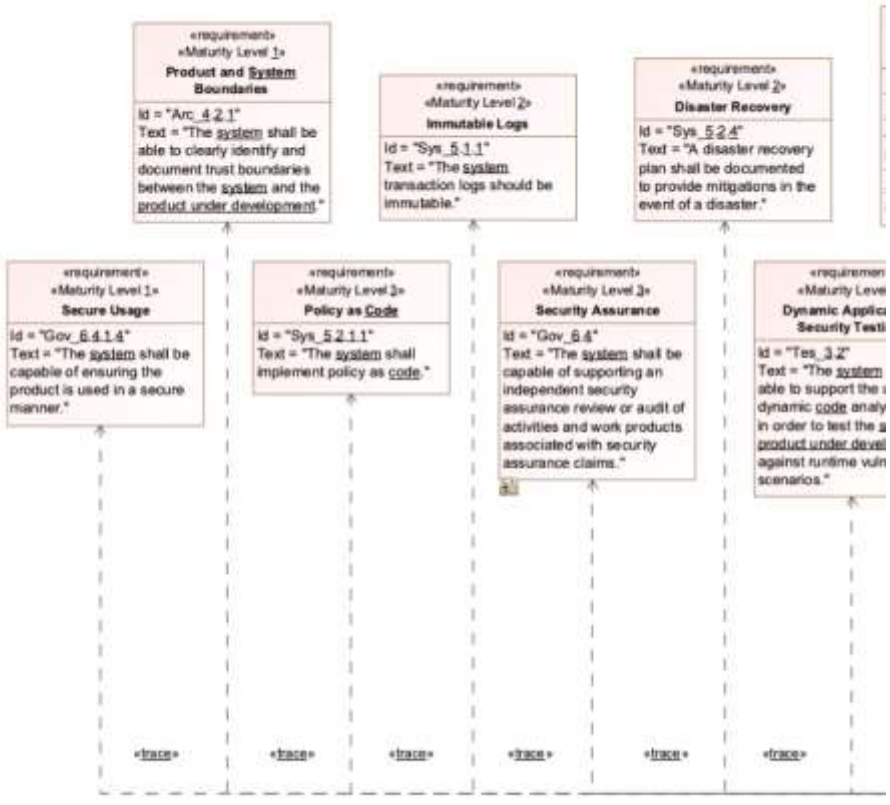




Over 200 Agile/DevSecOps requirements broken into 10 Capabilities and 4 Maturity Levels

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

DevSecOps Requirements



Example of Requirements Representation in Diagrams from PIM

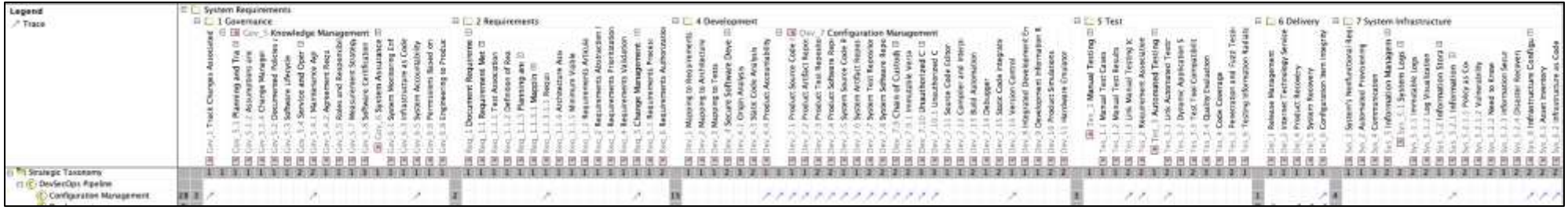
All requirements are organized into categories based on logical and functional groupings:

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

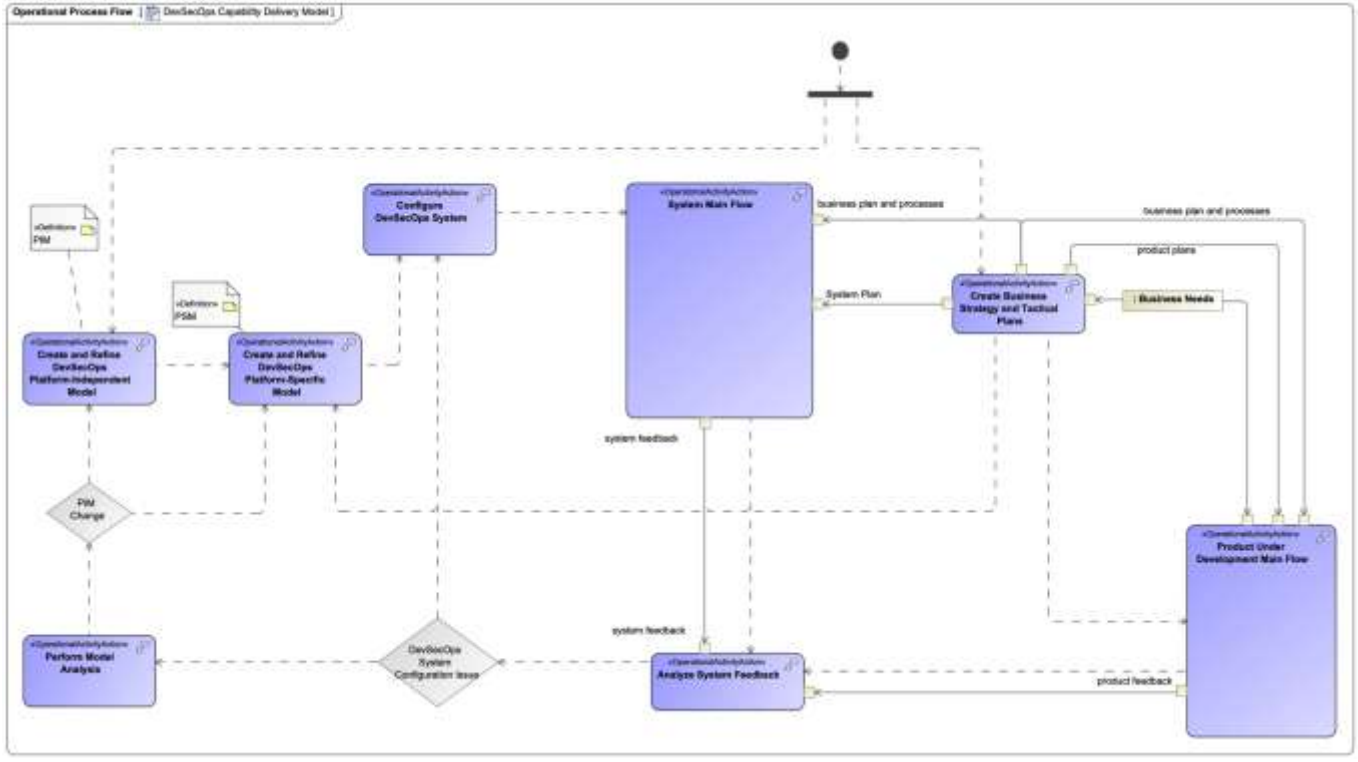
DevSecOps Capability/Strategic Viewpoint

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



DevSecOps Operational Viewpoints

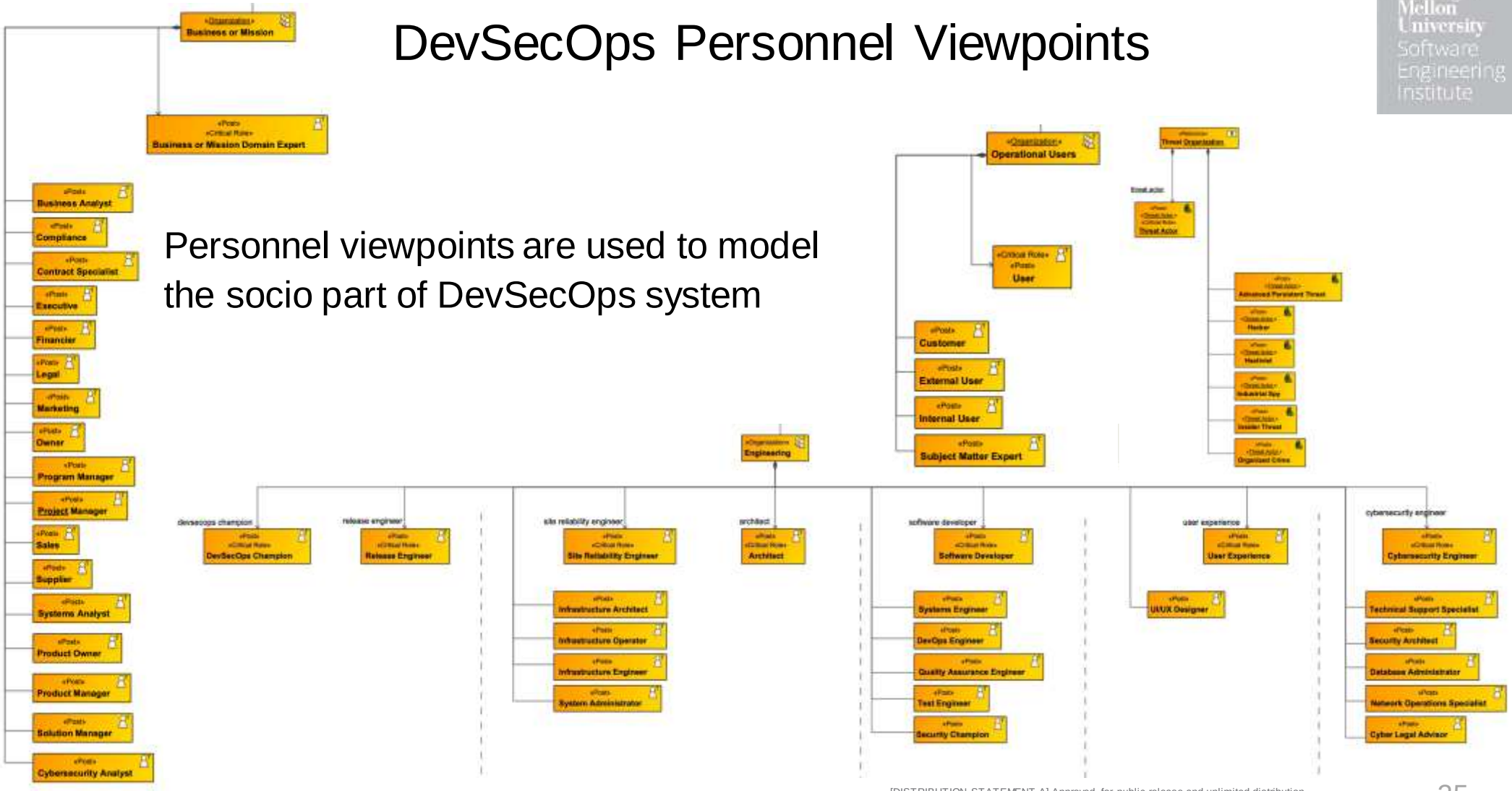


An operational model for a system describes behavior of the system to conduct enterprise operations

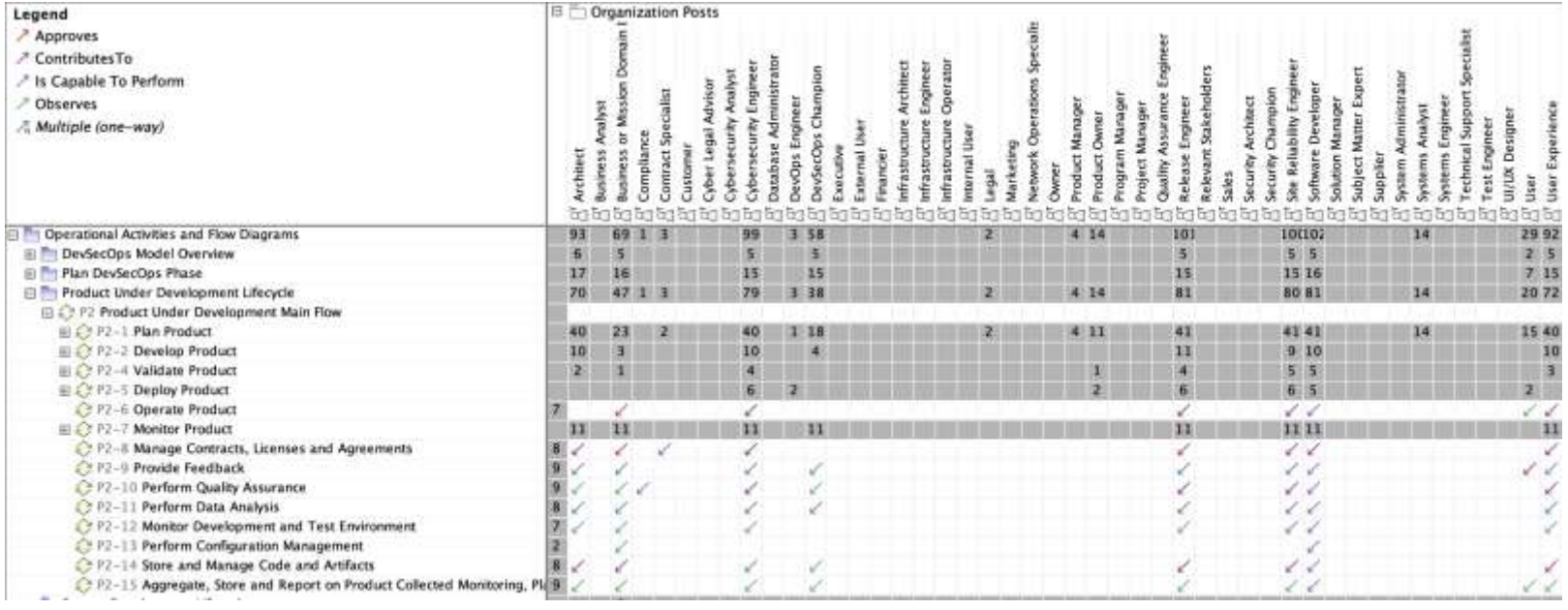
The main operational processes for DevSecOps includes development process for the product, as well as DevSecOps process itself.

DevSecOps Personnel Viewpoints

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



Everyone Plays a Role in DevSecOps

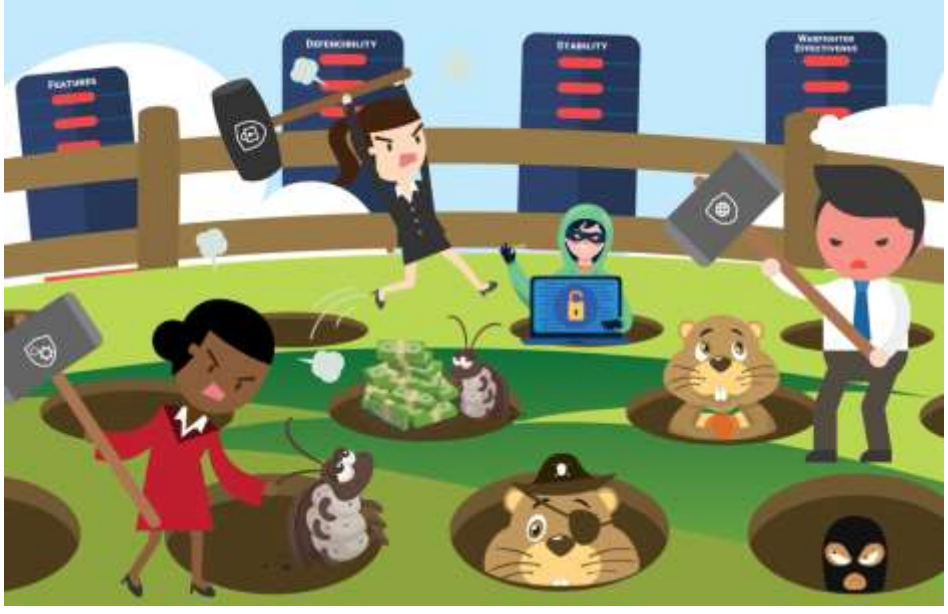


Critical Roles mapped to Operational Activities

Cybersecurity of Pipeline and Product

Carnegie Mellon University
Software Engineering Institute

Security Whack-A-Mole



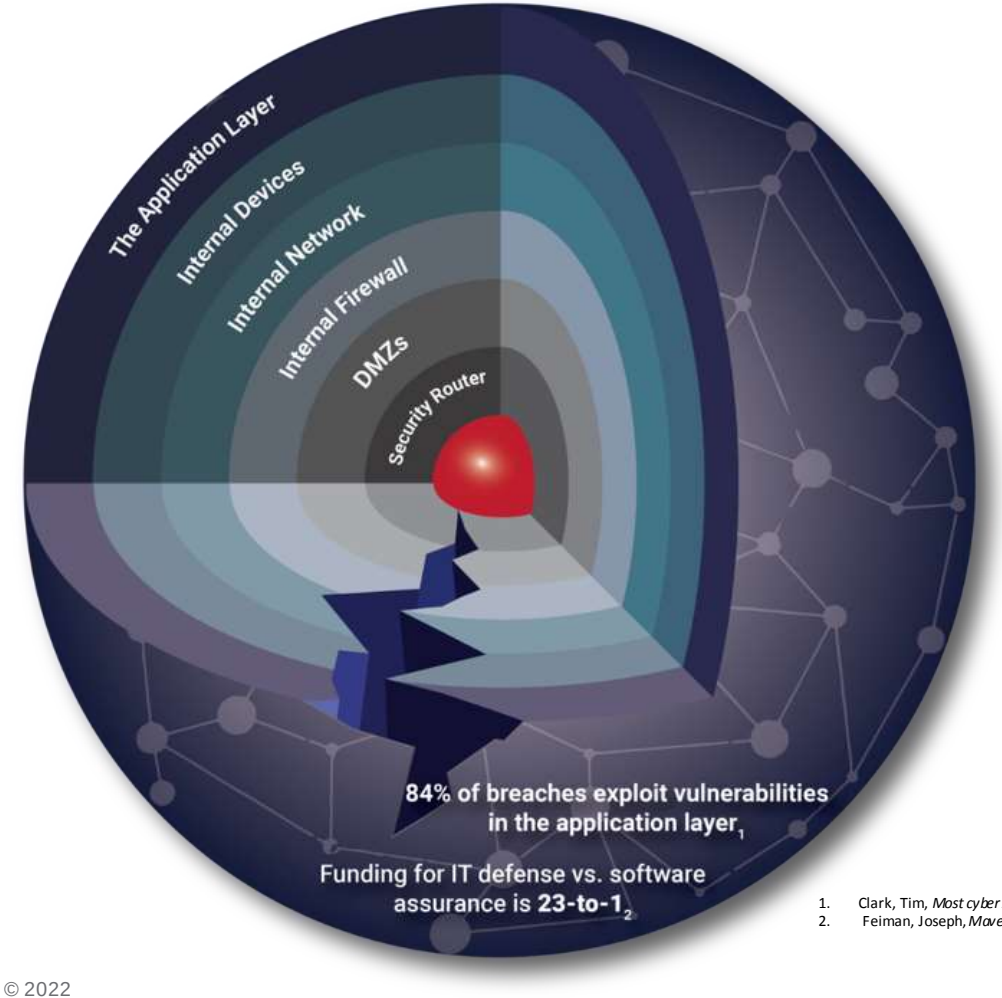
Winning in Features and Effectiveness, but Losing in Defensibility and Stability

In June of 2020 a generally successful DoD program completed an **8 week "Hardening the Software Factory" effort** in order to address **accumulated technical debt** and to address **insufficient security and operations** practices **due to the narrow focus on speed of delivery**.

These things occur, even in small relatively successful programs, when technical debt and insufficient security and operational practices are in place **due to lack of knowledge, experience, and reference material to fully design and execute an integrated DevSecOps strategy in which all stakeholder needs, including cybersecurity, are addressed.**

While playing Whack-A-Mole is inevitable, instead of missing the holes, or constantly hitting the same hole, the key is to fill in the holes.

Effective Security Requires a Holistic Approach



The Application Layer is the new perimeter exploited by 84% of breaches

Security must be engineered into the Lifecycle of Applications changing the way we build and buy technology

1. Clark, Tim, *Most cyber Attacks Occur from this Common Vulnerability,* Forbes. 03-10-2015
2. Feiman, Joseph, *Maverick Research: Stop Protecting Your Apps; It's Time for Apps to Protect Themselves,* Gartner. 09-25-2014. G00269825

Software Assurance (SwA)

Carnegie Mellon University Software Engineering Institute

DoD definition:

"the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the **software functions in the intended manner**."

[CNSS Instruction No. 4009; DoDi 5200.44 p.12]

SwA Curriculum Model definition:

Application of technologies and processes to achieve a required level of confidence that **software systems and services function in the intended manner**, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.

[Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum.* CMU/SEI-2010-TR-005. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415]

Risk

The perception of risk drives assurance decisions

- Assurance implementation choices (policies, practices, tools, restrictions) are based on the perception of threat and the expected impact should that threat be realized
- Perceptions are primarily based on knowledge about successful attacks
  - the current state of assurance is largely reactive
  - successful organizations learn from attacks and figure out how to react and recover faster and be vigilant in anticipating and detecting attacks
- Misperceptions are failures to recognize threats and impacts – “how could it happen to us?” or “it could not happen here!”

Interactions

Highly connected systems require alignment of risk across all stakeholders and systems otherwise critical threats will be unaddressed (missed, ignored) at different points in the interactions

- There are costs to addressing assurance which must be balanced against the impact of the risk
- Risk must also be balanced with other opportunities/needs (performance, reliability, usability, etc.)
- Interactions occur at many technology levels (network, security appliances, architecture, applications, data storage, etc.) and are supported by a wide range of roles

Trusted Dependencies

Your assurance depends on other people's decisions and the level of trust you place on these dependencies:

- Each dependency represents a risk
- Dependency decisions should be based on a realistic assessment of the threats, impacts, and opportunities represented by an interaction
- Dependencies are not static and trust relationships should be reviewed to identify changes that warrant reconsideration
- Using many standardized pieces to build technology applications and infrastructure increases the dependency on other's assurance decisions

Attacker

There are no prefect protections against attacks.

There exists a broad community of attackers with growing technology capabilities able to compromise the confidentiality, integrity, and availability of any and all of your technology assets and the attacker profile is constantly changing.

- The attacker uses technology, processes, standards, and practices to craft a compromise (socio-technical responses).
- Attacks are crafted to take advantage of the ways we normally use technology or designed to contrive exceptional situations where defenses are circumvented.

Mitigating Risk with Assurance Cases

Carnegie Mellon University Software Engineering Institute

Understanding risk is hard!

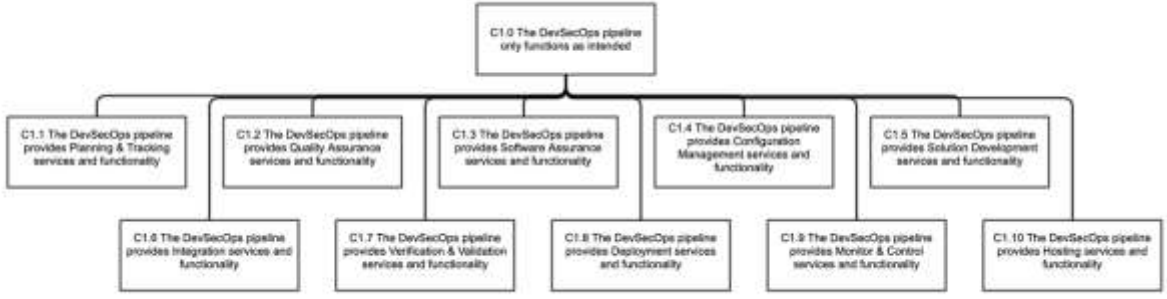
Without being able to quantify, or reason around, the cybersecurity risks associated with your product and DevSecOps pipeline, you will not be able to:

- properly balance between features, defensibility, and stability
- make necessary trade-off choices to achieve your organization's mission and vision in a cost-effective way

An assurance case can be used to reason about the adequacy for both the pipeline and the product

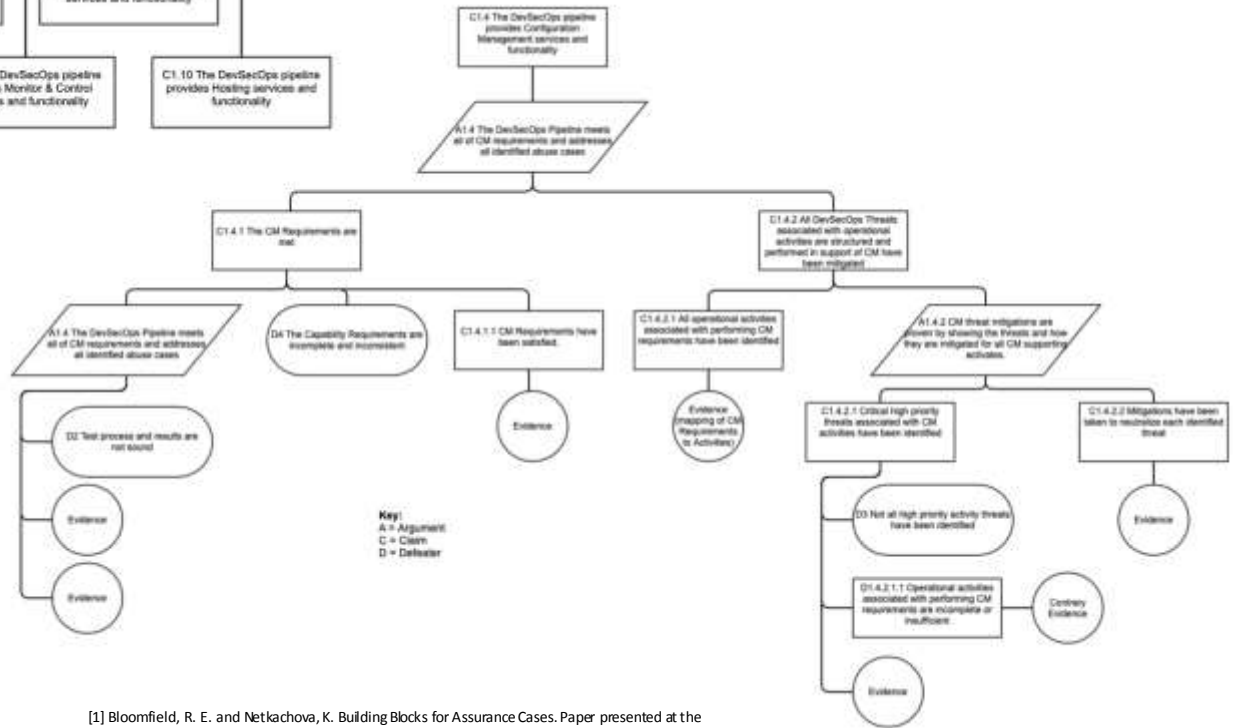
- It is a structured approach used to argue that available evidence supports a given claim
- It provides the organization with the basis for making risk-based choices tied to assuring that the pipeline only functions as intended
- It provides requirements for automated systems testing, or other evidence collection techniques
- Actual test results provides the evidence needed to support the assurance claims

Structuring a DevSecOps Assurance Case



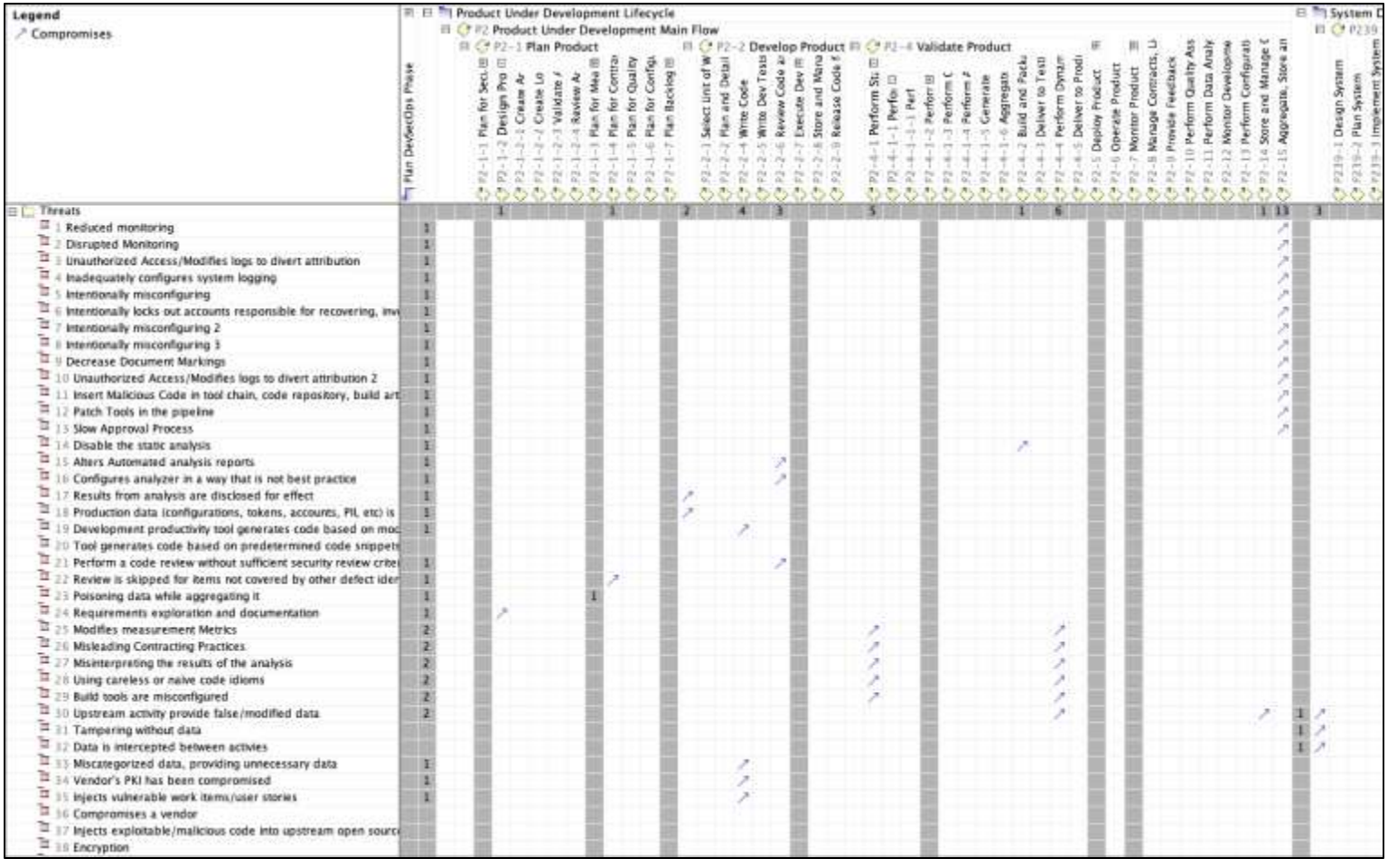
Assurance cases are composed of the following elements:

- Claims– "assertions put forward for general acceptance. They are typically statements about a property of the system or some subsystem. Claims that are asserted as true without justification become assumptions and claims supporting an argument are called subclaims [1]."
- Arguments – "link the evidence to the claim [1]" by stating the assumption(s) on which the claim and the evidence are built upon.
- Evidence – "Evidence that is used as the basis of the justification of the claim. Sources of evidence may include the design, the development process, prior field experience, testing, source code analysis or formal analysis [1]."
- Defeaters – "possible reasons for doubting the truth of a claim [2]."



[1] Bloomfield, R. E. and Netkachova, K. Building Blocks for Assurance Cases. Paper presented at the International Symposium on Software Reliability Engineering (ISSRE), 03-11-2014 - 06-11-2014, Naples, Italy.
[2] Goodenough, John B., Charles B. Weinstock, Ari Z. Klein. Toward a Theory of Assurance Case Confidence, CMU/SEI-2012-TR-002 September 2012.

DevSecOps Threat to Operational Activity Matrix

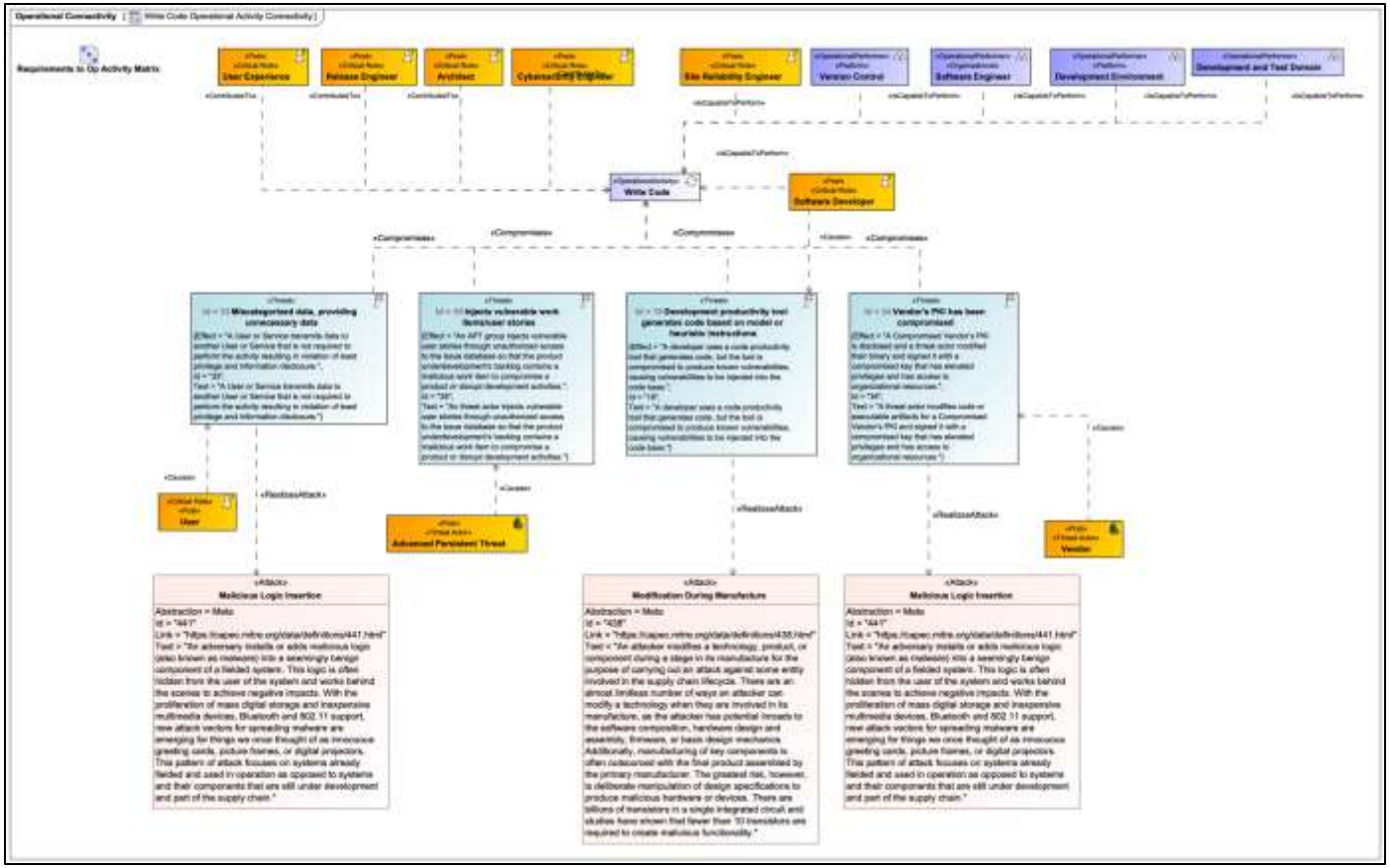


DevSecOps Threats with Attributes

Carnegie Mellon University Software Engineering Institute

Id	Name	Text	Effect	Compromises	Realized By Attack	Caused By	Mitigated By	Document
1	Reduced monitoring	A threat actor is made aware of a monitoring system's reduced capacity resulting in regular service outages leaving an open window of opportunity for an unobservable attack.	Reduced or misconfigured monitoring allows for nefarious activity to occur	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	607 Obstruction	Insider Threat		Much of this was pulled from CAPEC info https://capec.mitre.org/data/definitions/1000.html
2	Disrupted Monitoring	A threat actor spoofs a legitimate account (user or service) and injects falsified data into the monitoring system to disrupt operations, create a diversion, or mask the attack.	MONITORING: falsified data injected/spoofing, tampering, integrity, injects falsified data into the monitoring system to disrupt	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	161 Infrastructure Manipulation	Advanced Persistent Threat Insider Threat Architect Cybersecurity Engineer	SC1 Mitigation Strategy 1	Keep at the Meta Level and better explained in the "star
3	Unauthorized Access/Modifies logs to divert attribution	A threat actor gains unauthorized access to logging data, alters system logs to conceal illicit activity from forensic audits, automated responses and alerts, or to divert attribution.	Logs: insider threat modifies the logs to conceal activity	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	161 Infrastructure Manipulation	Insider Threat Site Reliability Engineer Cybersecurity Engineer		
4	Inadequately configures system logging	A threat actor has configured the collection of system logs in a way that limits the effectiveness of forensic audit activities.	Accidentally misconfiguring Logging - can't perform forensics work against what is captured	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	176 Configuration/Environment Manipulation	Software Developer		Could be 1617 Most significa improper configuration
5	Intentionally misconfiguring	A threat actor has configured the collection of system logs in a way that limits the effectiveness of forensic audit activities in order to conceal subsequent activities.	Intentionally misconfiguring the system	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	176 Configuration/Environment Manipulation	Insider Threat		
6	Intentionally locks out accounts responsible for recovering, investigating, or repairing the system	A threat actor spoofs an individual's account in order to create user action logs with the objective of making a targeted user in violation of security policy and reducing the targeted individual's organizational effectiveness.	Targeting individual with the intent that their login is denied, locking out individuals who should have access	P2-15 Aggregate, Store and Report on Product Collected Monitoring, Planning and Feedback Data	212 Functionality Misuse	Insider Threat		Could be a CAPEC - 184 So Attack
		Unit testing is insufficient to cover the requirements and abuse cases. A software or site reliability engineer doesn't		P2-15 Aggregate, Store and Report on Product Collected	176 Configuration/Environment	Software Developer		

Example Threat Modeling Diagram for Write Code Operational Activity



BACKUP Slides

Agile and DevSecOps Principles

Carnegie Mellon University
Software Engineering Institute

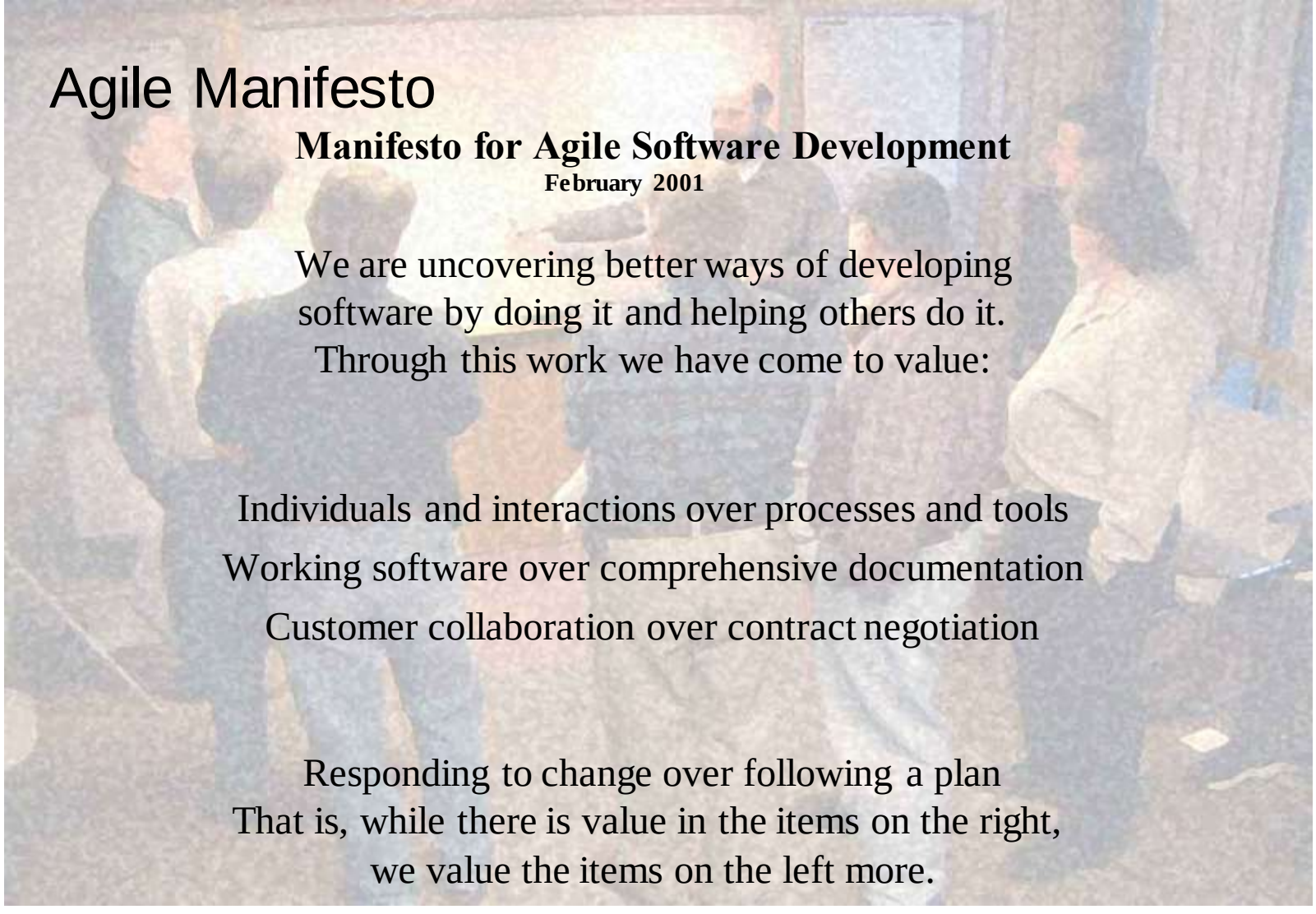
Working Definition of Agile

Agile

An *iterative* and *incremental* (evolutionary) approach to software development which is performed in a *highly collaborative manner* by *self-organizing teams* within an *effective governance framework* with *"just enough" ceremony* that produces *high quality software* in a *cost effective and timely* manner which *meets the changing needs of its stakeholders*. [Ambler 2013]

[Ambler 2013] Ambler, Scott. *Disciplined Agile Software Development: Definition*. http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

Agile Manifesto



Manifesto for Agile Software Development

February 2001

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan
That is, while there is value in the items on the right,
we value the items on the left more.

The Twelve Agile Principles[1]

1. Our highest priority is to **satisfy the customer through early and continuous delivery of valuable software**.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together daily throughout the project.**
5. **Build projects around motivated individuals.** Give them the environment and support they need, **and trust them to get the job done**.
6. The most efficient and effective method of **conveying information** to and within a development team is **face-to-face conversation**.

The Twelve Agile Principles$_2$

7. **Working software is the primary measure of progress.**
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to **maintain a constant pace indefinitely**.
9. **Continuous attention to technical excellence and good design enhances agility.**
10. Simplicity—**the art of maximizing the amount of work not done**—is essential.
11. **The best architectures, requirements, and designs emerge from self-organizing teams**.
12. **At regular intervals, the team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly.

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.