Secure by Design

Timothy A. Chick
CERT Applied Systems Group Technical Manager, CMU-Software Engineering Institute
Adjunct Faculty Member, CMU-Software and Societal Systems Department (S3D)

Carnegie Mellon University
Software Engineering Institute

Document Markings

Today: Program Office Whac-A-Mole

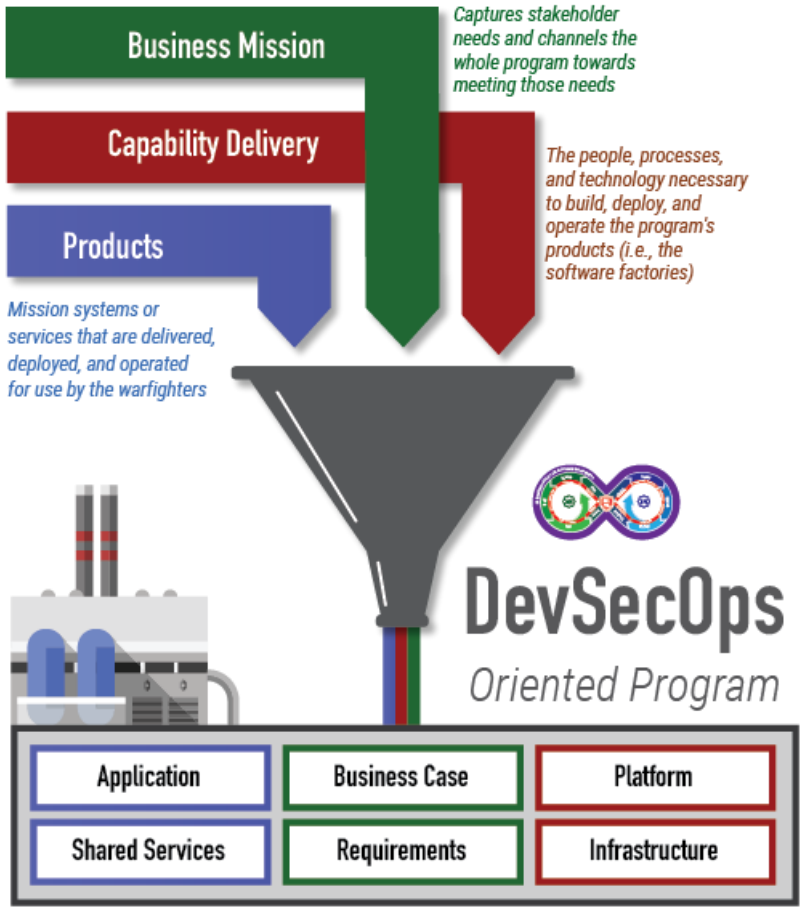
Winning in Features and Effectiveness, but Losing in Defensibility and Stability

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

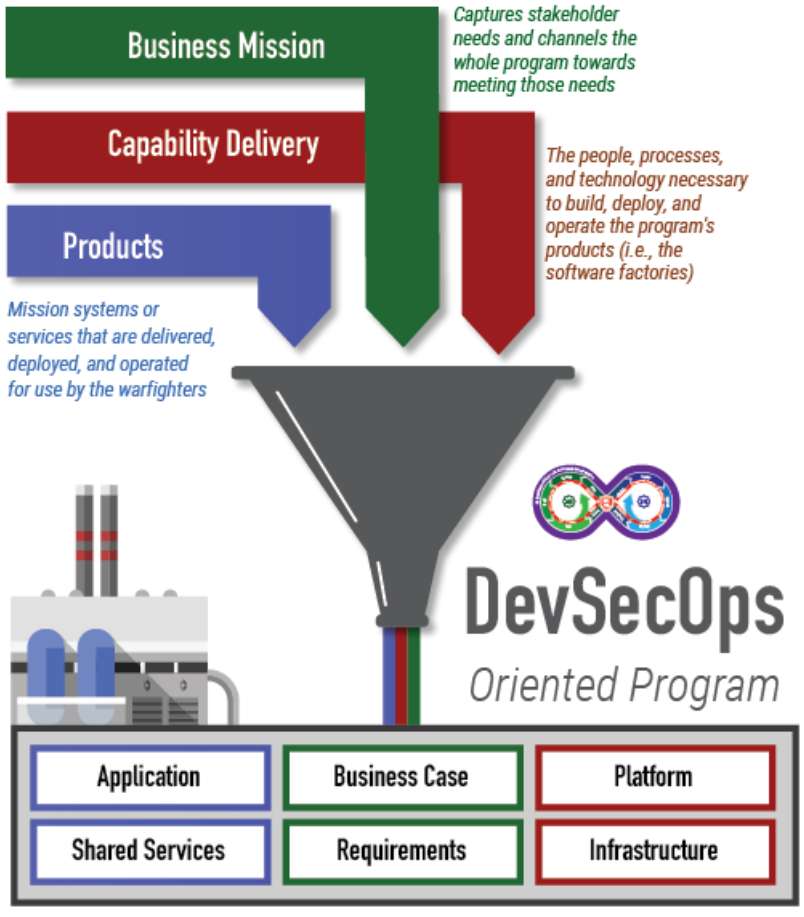
A Program View



All software oriented programs 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 program exists
- **Capability to Deliver Value** – covers the people, processes, and technology necessary to build, deploy, and operate the program's products
- **Products** – the units of value delivered by the program. Products utilize the capabilities delivered by the software factory and operational environments.

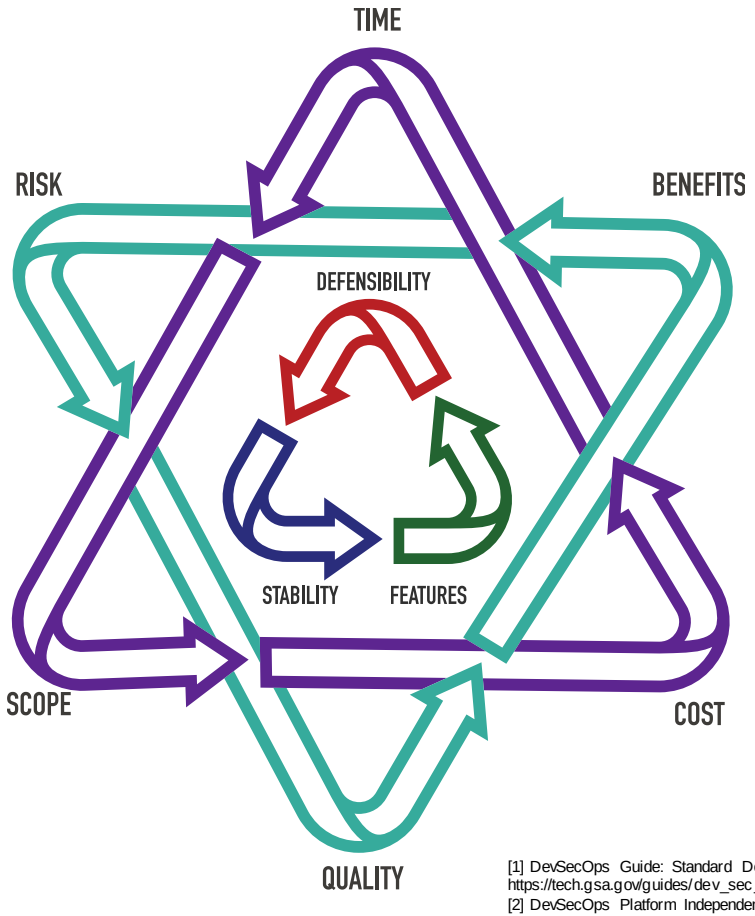
Challenge 1: connecting process, practice, and tools



Capabilities and Products are not static.

- Infrastructure and shared services are often maintained across multiple organizations
- Processes, practices, and tools must evolve to meet the needs of the products being built and operated
- Products must evolve to meet changing needs, defects found, and changes to other systems.

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].

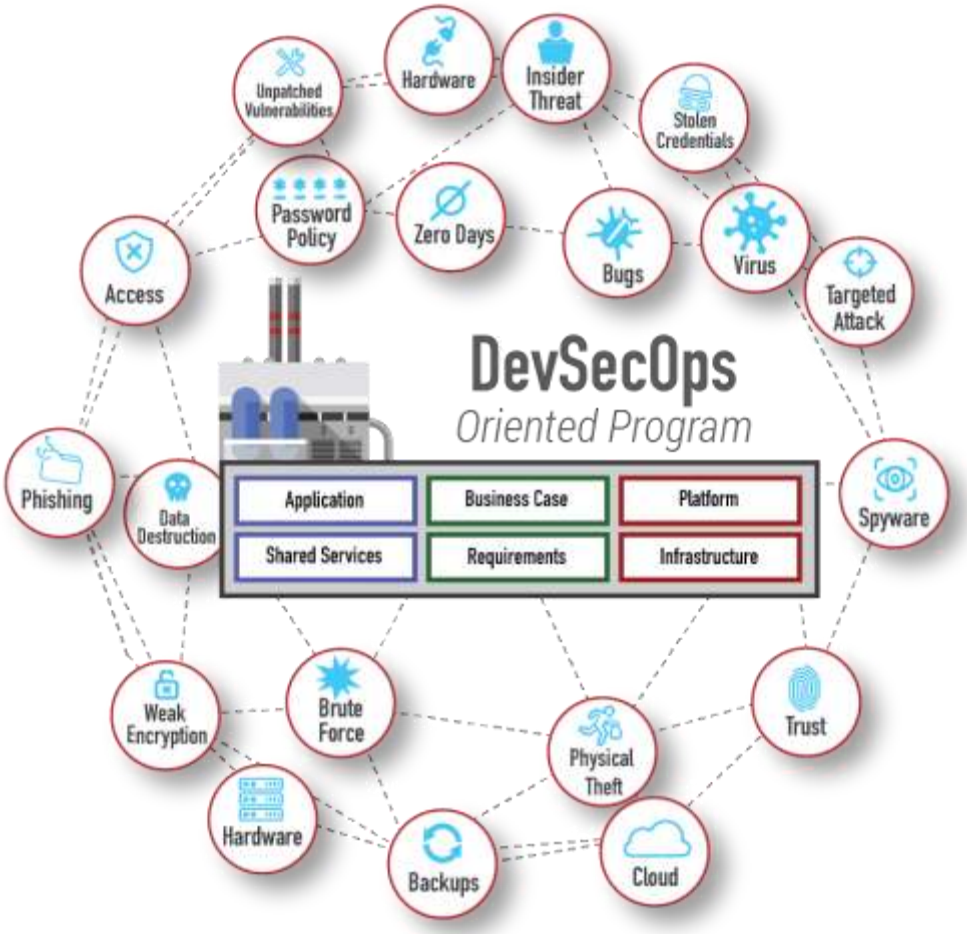
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

- **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/

Challenge 2: Addressing Threats to both 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?

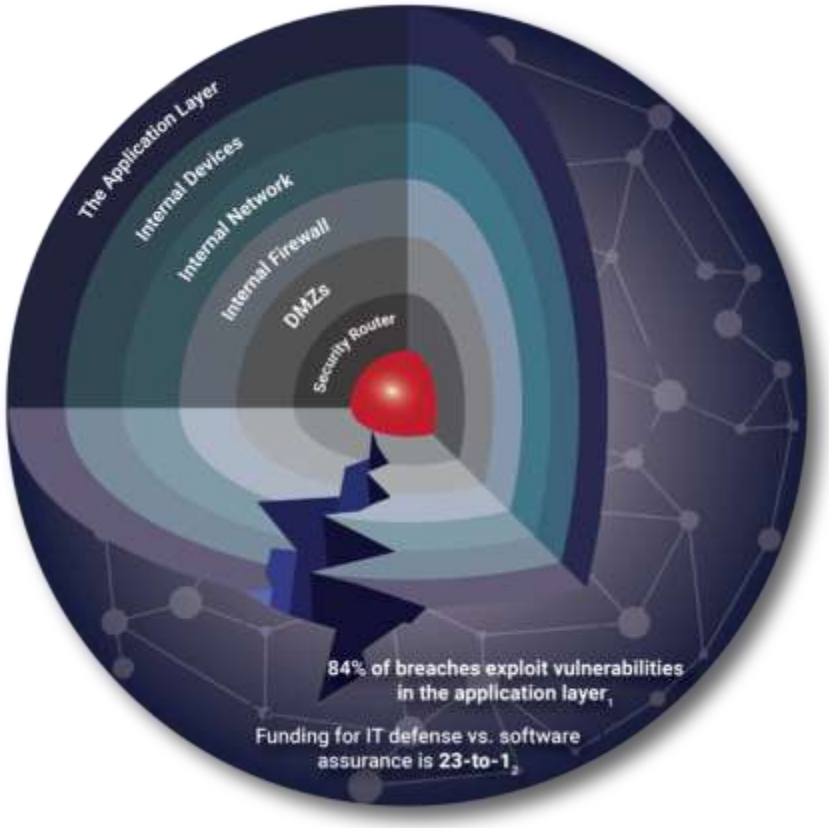
Using a capability service to attack a product isn't new

https://www.itworld.com/article/2861675/cyberattack-on-german-steel-factory-causes-massive-damage.html

"**Steelworks compromise causes massive damage to furnace.**

One of the most concerning was a targeted APT attack on a German steelworks which ended in the attackers gaining access to the business systems and through them to the production network (including SCADA). The effect was that the attackers gained control of a steel furnace and this caused massive damages to the plant."

One Opening is all an Adversary Needs



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!”

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.

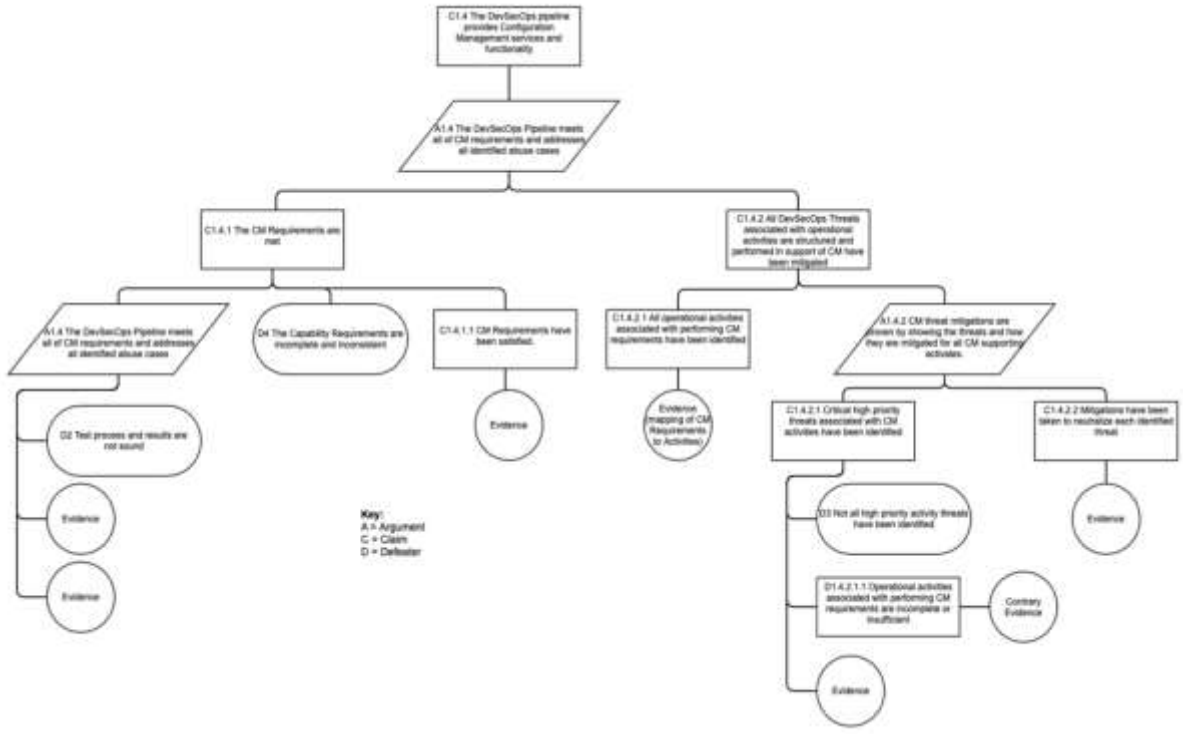
- It is a structured approach used to argue that available evidence supports a given claim
- It provides the organization with the basis for making risk-based choices tied to assuring that the pipeline only functions as intended.
- It provides requirements for automated systems testing, or other evidence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

Assuring that your Program only Functions as Intended

Carnegie Mellon University Software Engineering Institute

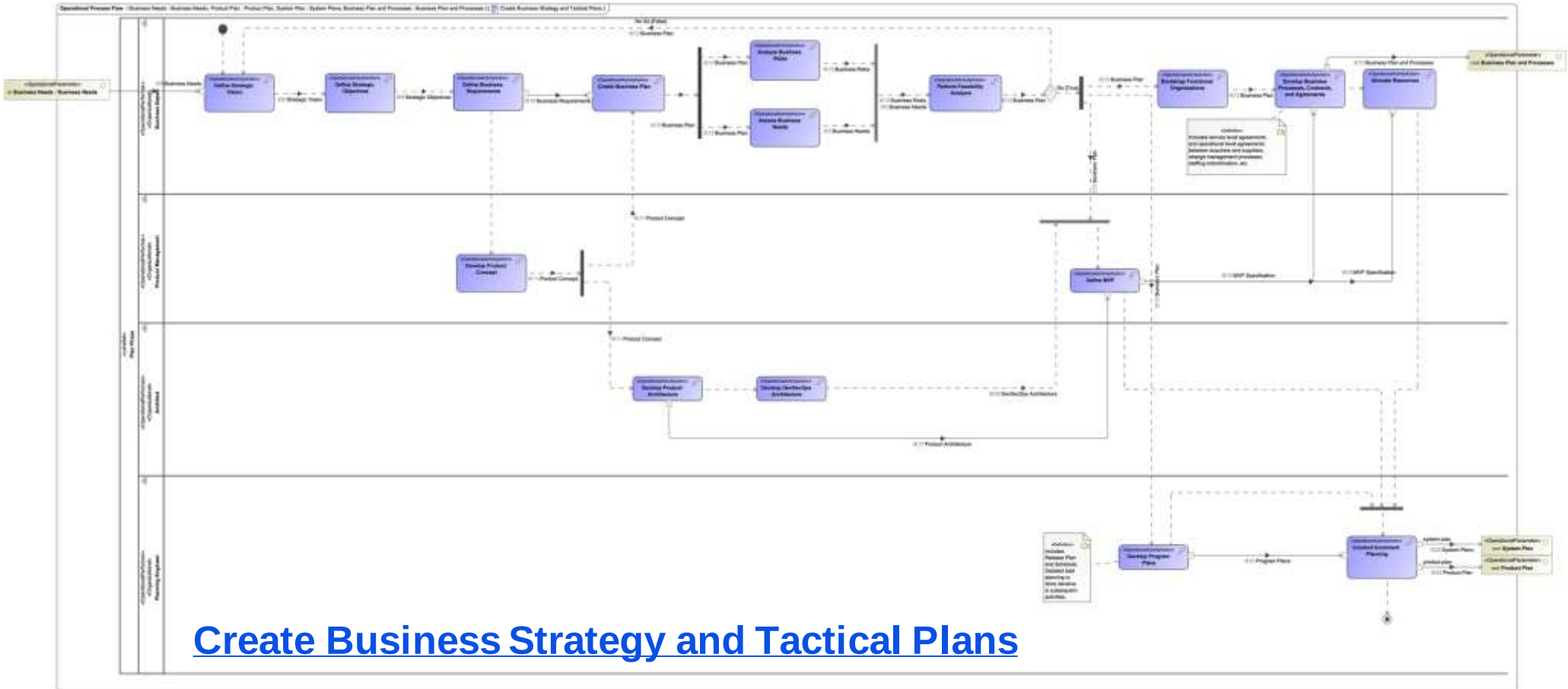
Assurance cases are composed of the following elements:

- 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.

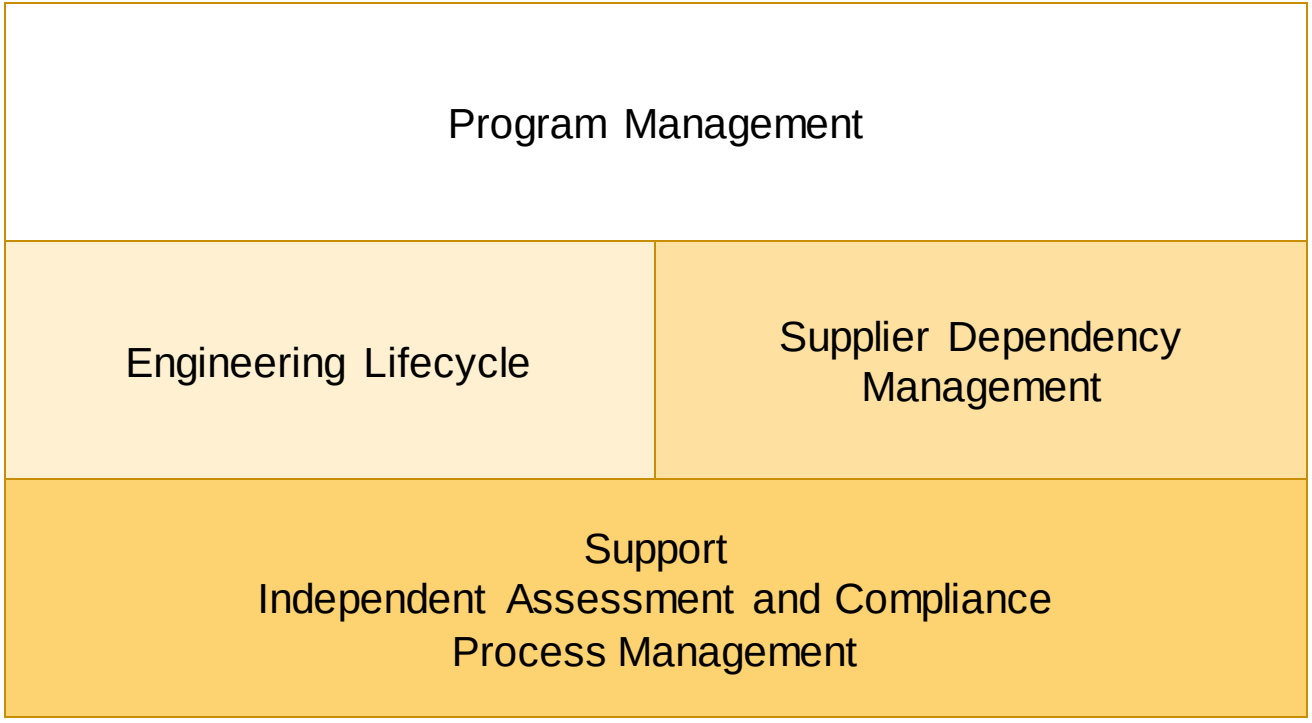
Security Starts at Inception



Create Business Strategy and Tactical Plans

Acquisition Security Framework (ASF)

Acquisition Security Framework (ASF)



Four of the six areas are ready for use: Program Management, Engineering Lifecycle, Supplier Dependency Management, and Support. The remaining areas have been drafted and will be completed this calendar year.

What is the ASF?

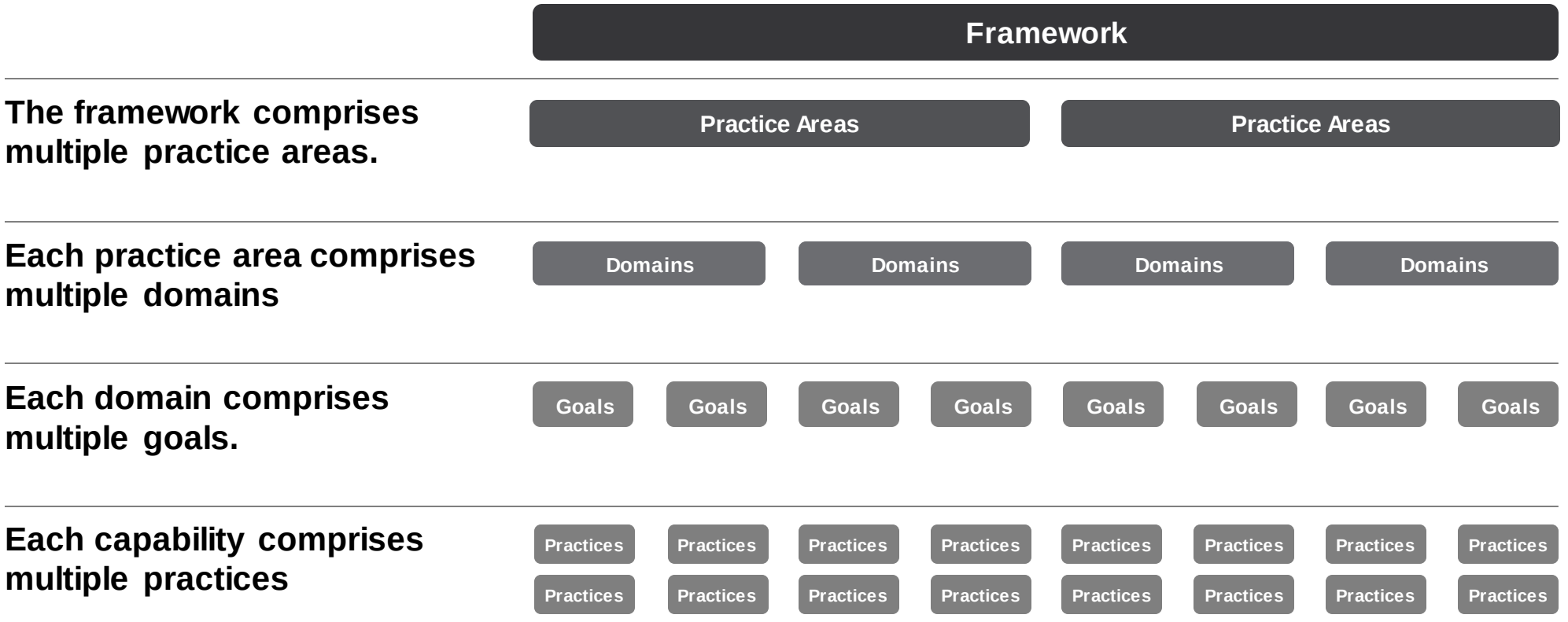
The Acquisition Security Framework (ASF) is a collection of leading practices for building and operating secure and resilient software-reliant systems.

The ASF is designed to proactively enable system security and resilience engineering across the lifecycle and supply chain.

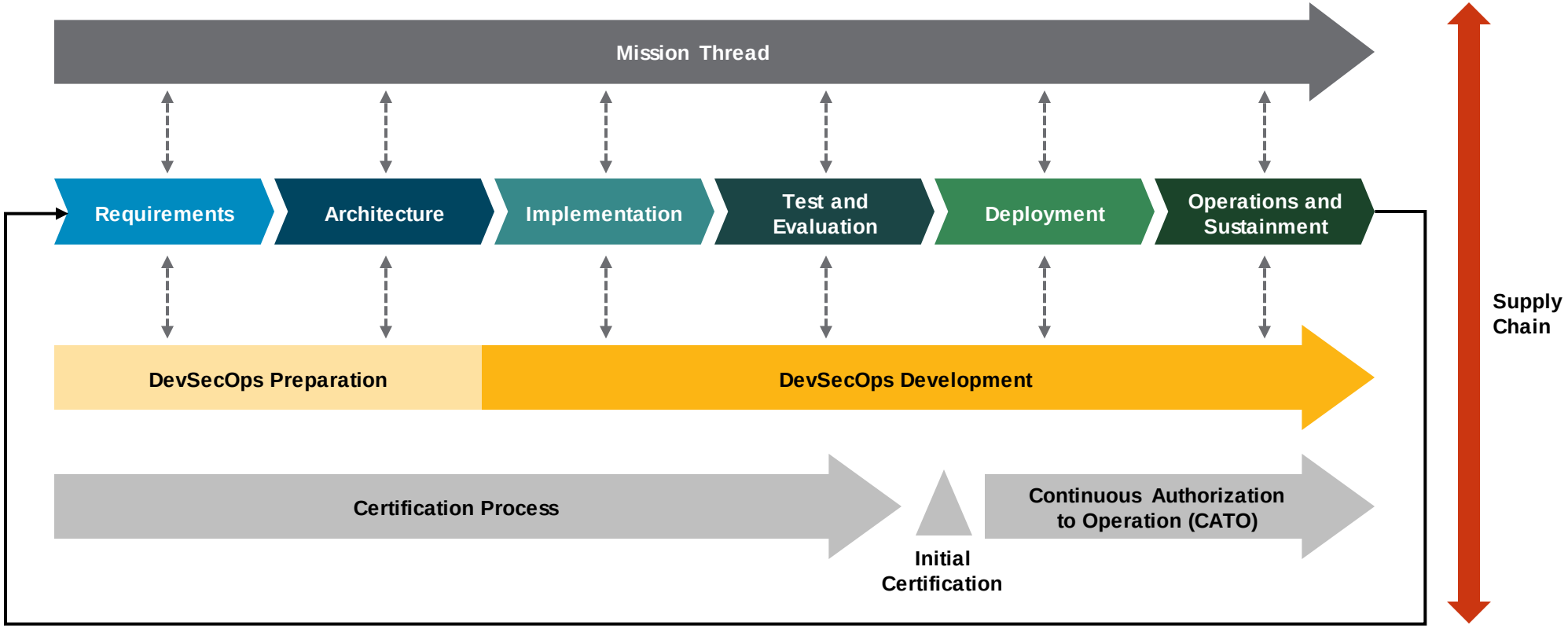
ASF provides a roadmap for building security and resilience into a system rather than attempting to "bolt it on" after deployment.

ASF facilitates efficient and predictable systems environments and more manageable delivery and risk outcomes.

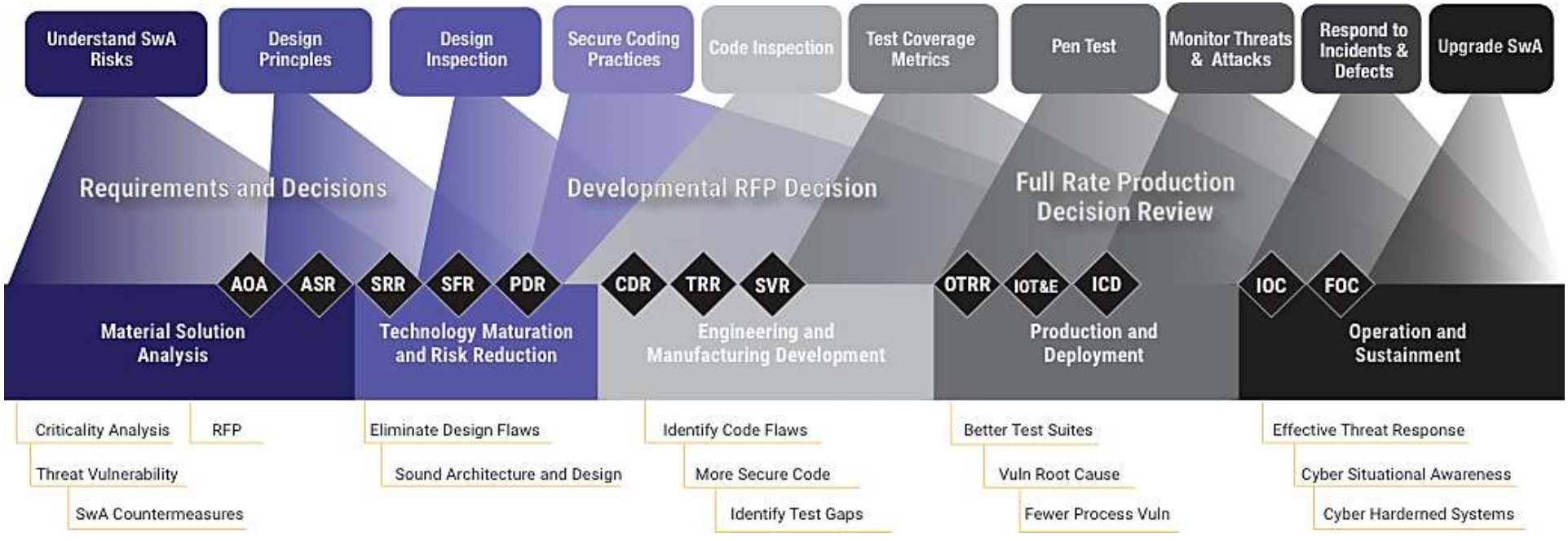
ASF Structure



Cybersecurity Problem Space



Just like Quality, Security is a lifecycle challenge



Security Requirements Challenges

Carnegie Mellon University Software Engineering Institute

Typical problems with security requirements

- Stated as specific security solutions (practices) and not real requirements
  - Ex: Only authorized users shall access personal healthcare information
- Too narrowly focused on security in a particular application
  - Ex: use SSL for Web communication
- Compliance mandates are substituted for security requirements
  - Ex: An audit log must be maintained of every access to the patient's healthcare information
- Focused on selection of controls after designs are complete
- Ignored in requirements elicitation because no stakeholders are knowledgeable enough about security impacts to state their security requirements

Merely Specifying Security Features is Insufficient

Carnegie Mellon University Software Engineering Institute

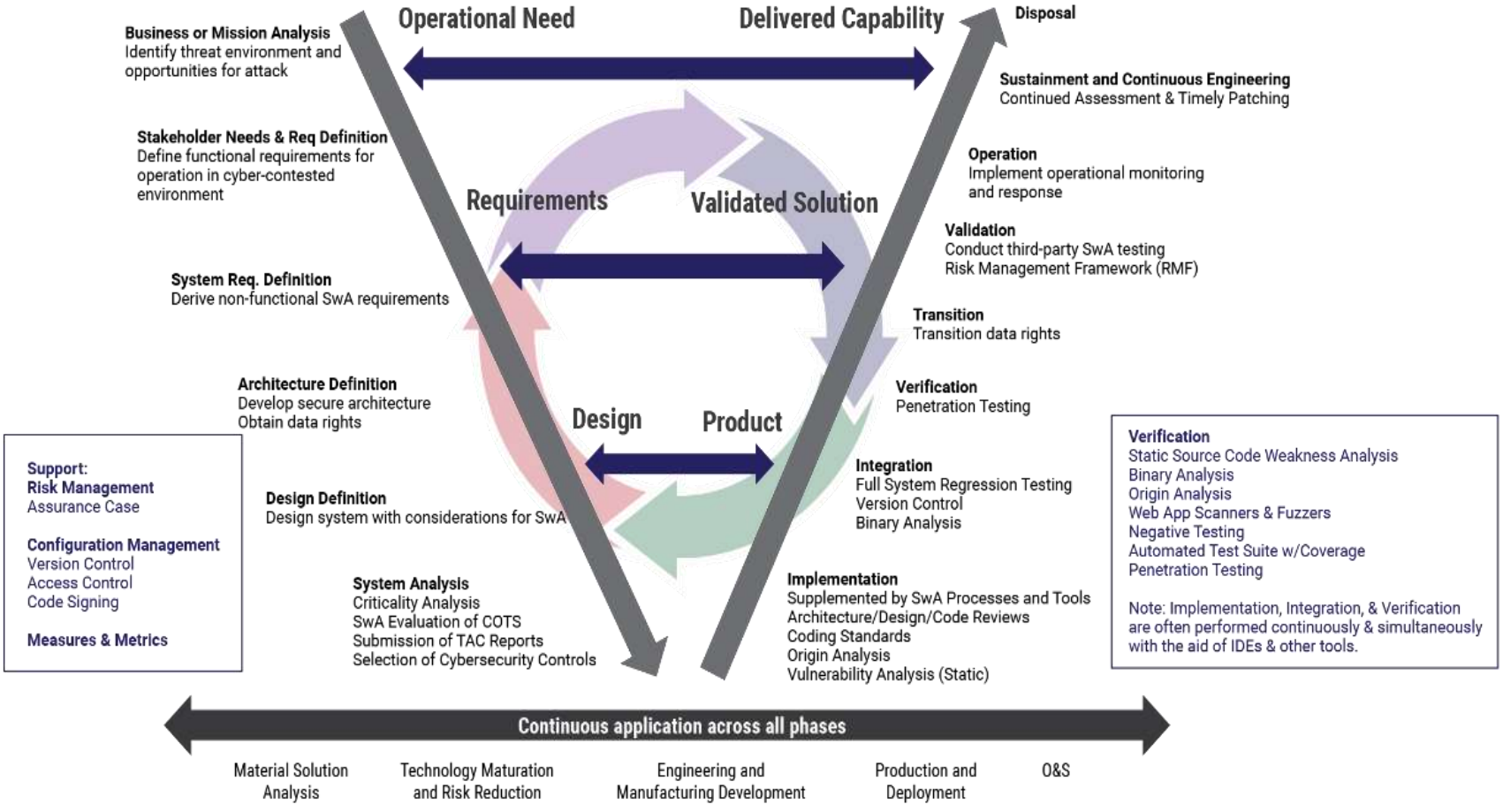
One needs to

- anticipate ways in which a system can be misused by adversaries
- perform systematic, rigorous, and customized threat analysis
- associate attack methods with the likely identified threats
- define and document mitigation strategies aimed at thwarting the attacks
- Write appropriately specific security requirements

"*Early specification of security requirements positively impacts fundamental architectural decisions that enable security concerns to be addressed from the ground up, rather than added as late-in-the-day patches in an attempt to remediate security vulnerabilities.*"

https://resources.sei.cmu.edu/asset_files/TechnicalNote/2018_004_001_516627.pdf

Software Assurance Activities Mapping



Threat Modeling

- **Threat Modeling** is the process of creating an abstraction of a system, aimed at identifying attackers' abilities and goals, and using that abstraction to generate and catalog possible threats that the system must mitigate.
- While security can be analyzed at the networking and code levels to prevent buffer overflows, SQL injection attacks, etc. there is value in **creating a mindset of defensive thinking** early in the requirements and architecture phases.
- **Defensive thinking** means that for every new feature, one must think about how it could be abused or defeated by adversaries.
- The defensive thinking mindset **underlies the approach to threat modeling**

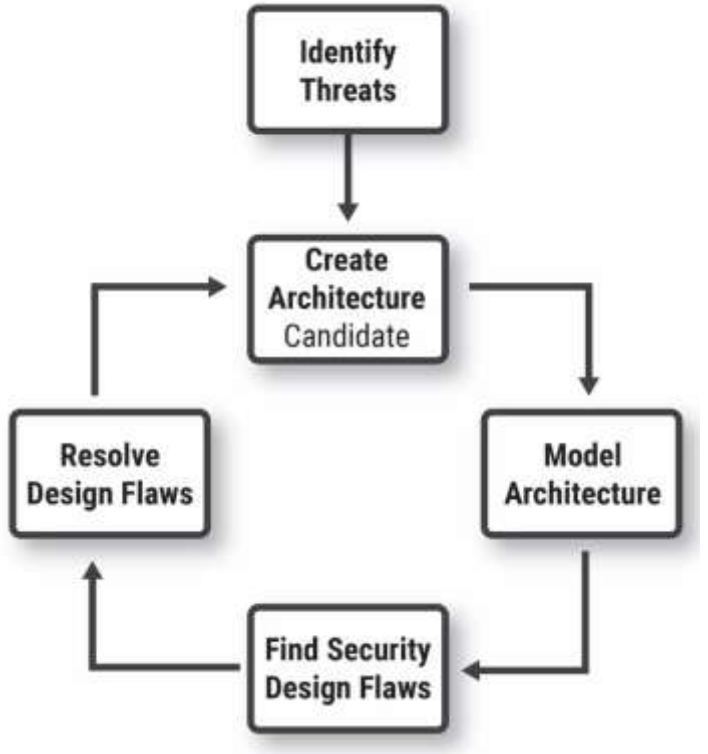
https://insights.sei.cmu.edu/blog/the-hybrid-threat-modeling-method/

Value of Modeling Security

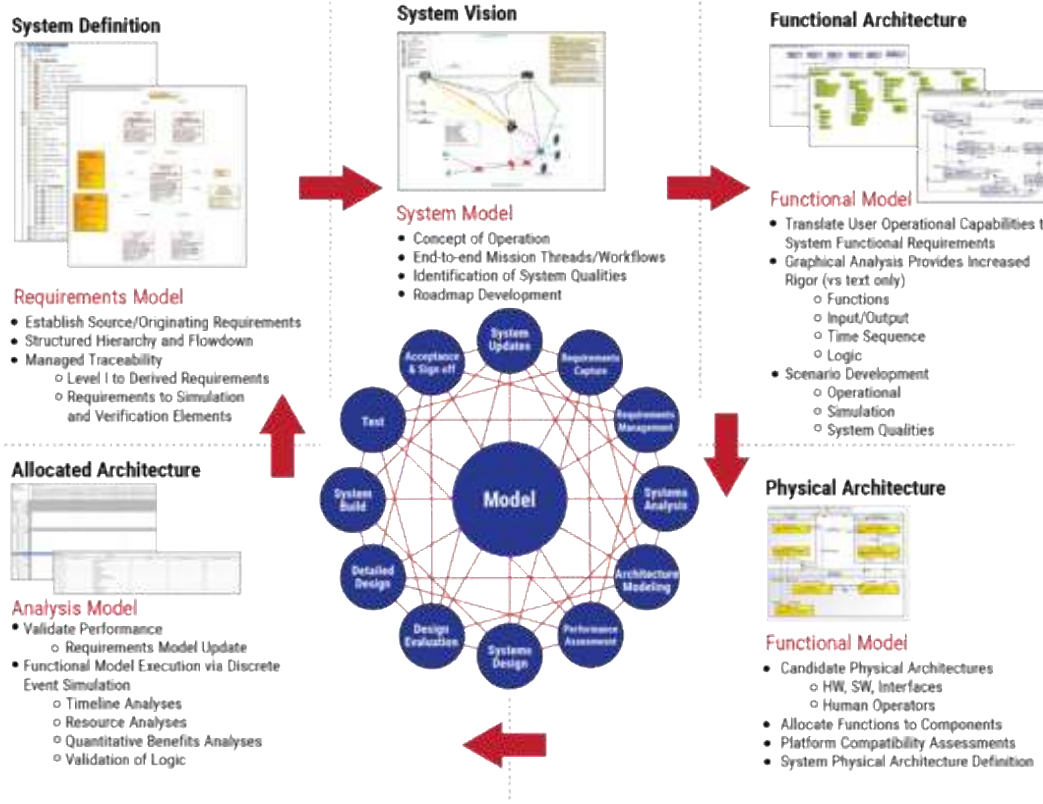
Crucial security decisions to address threats are made in the architecture.

Analyzing an architecture is a huge opportunity for improving security.

Threat Modeling methods can be combined with MBSE to create a more robust and well-rounded view of potential threats.



Model Based Systems Engineering



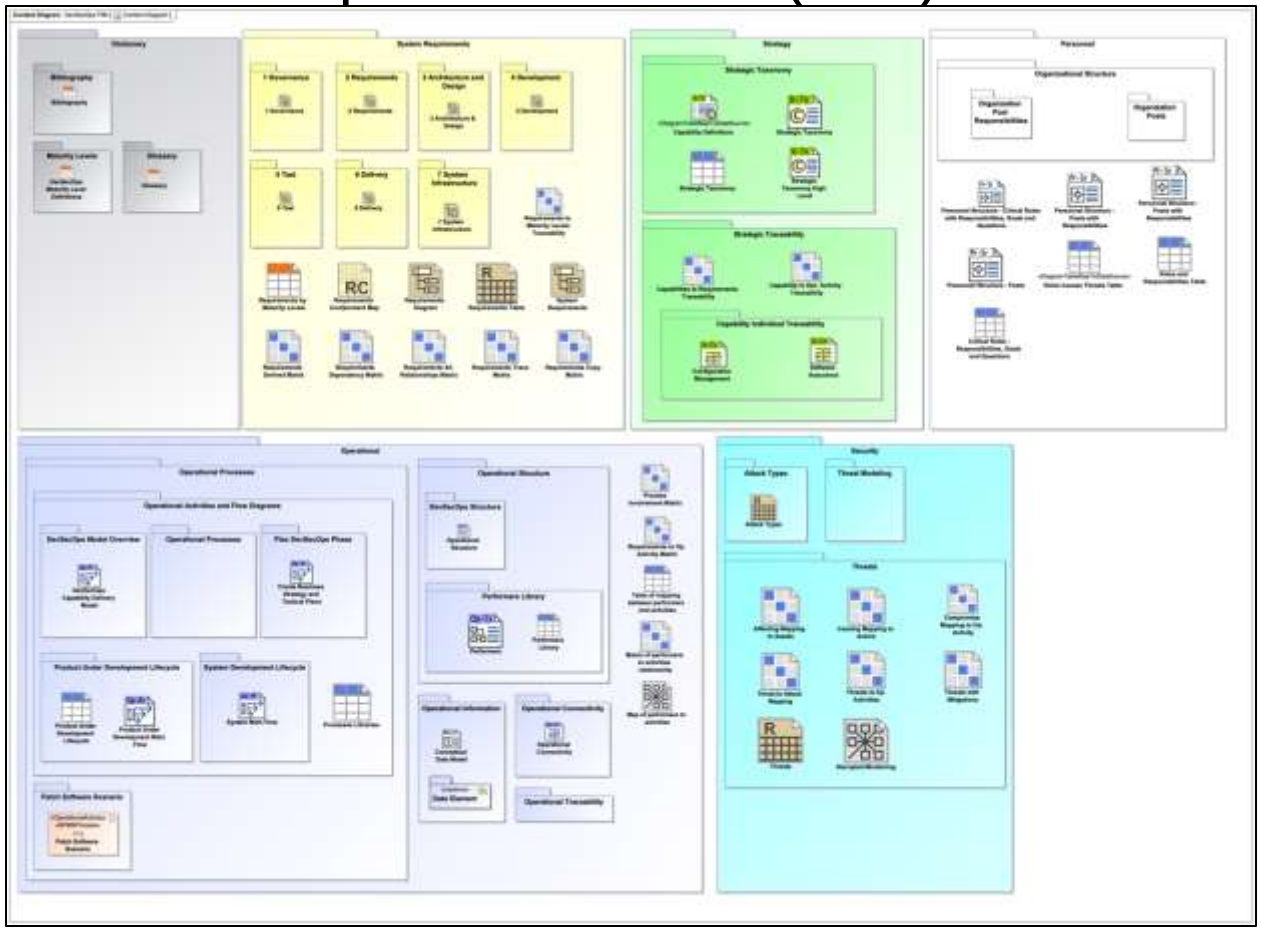
*The Digital System Model contains the most current requirements, key mission/business operations, architecture, design details, implementation details, test and evaluation details, and supporting documentation.

- ***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

A DevSecOps Example

Carnegie Mellon University
Software Engineering Institute

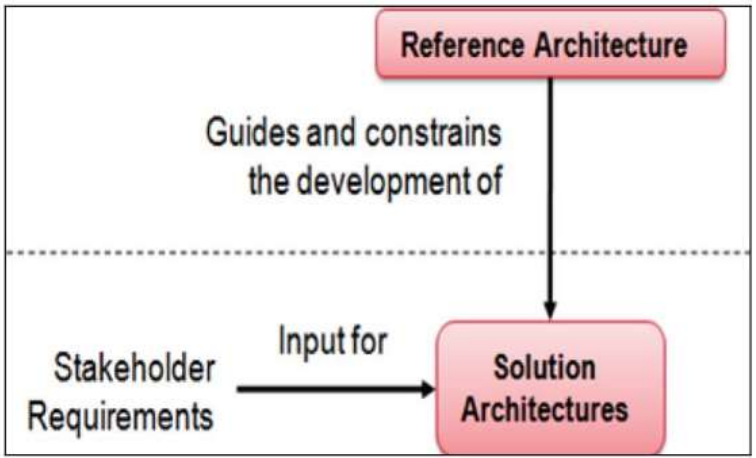
DevSecOps Platform Independent Model (PIM) - Content Diagram



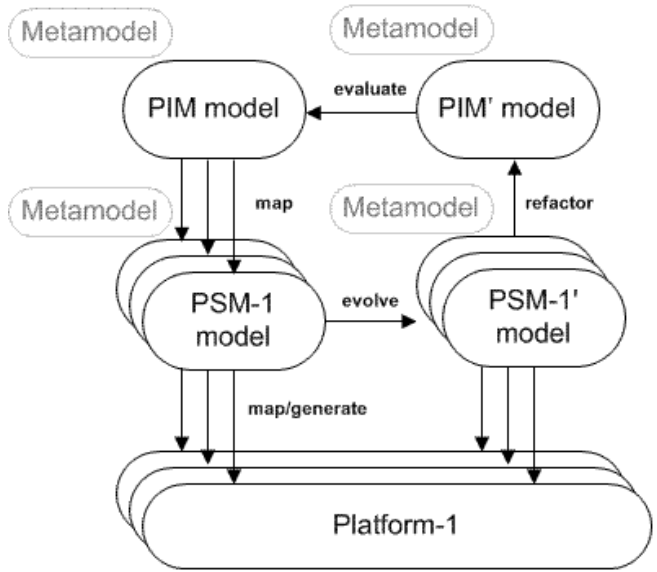
https://cmu-sei.github.io/DevSecOps-Model/

Reference Architecture/Platform Independent Model (PIM)

A **Reference Architecture** is an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions [1].



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.



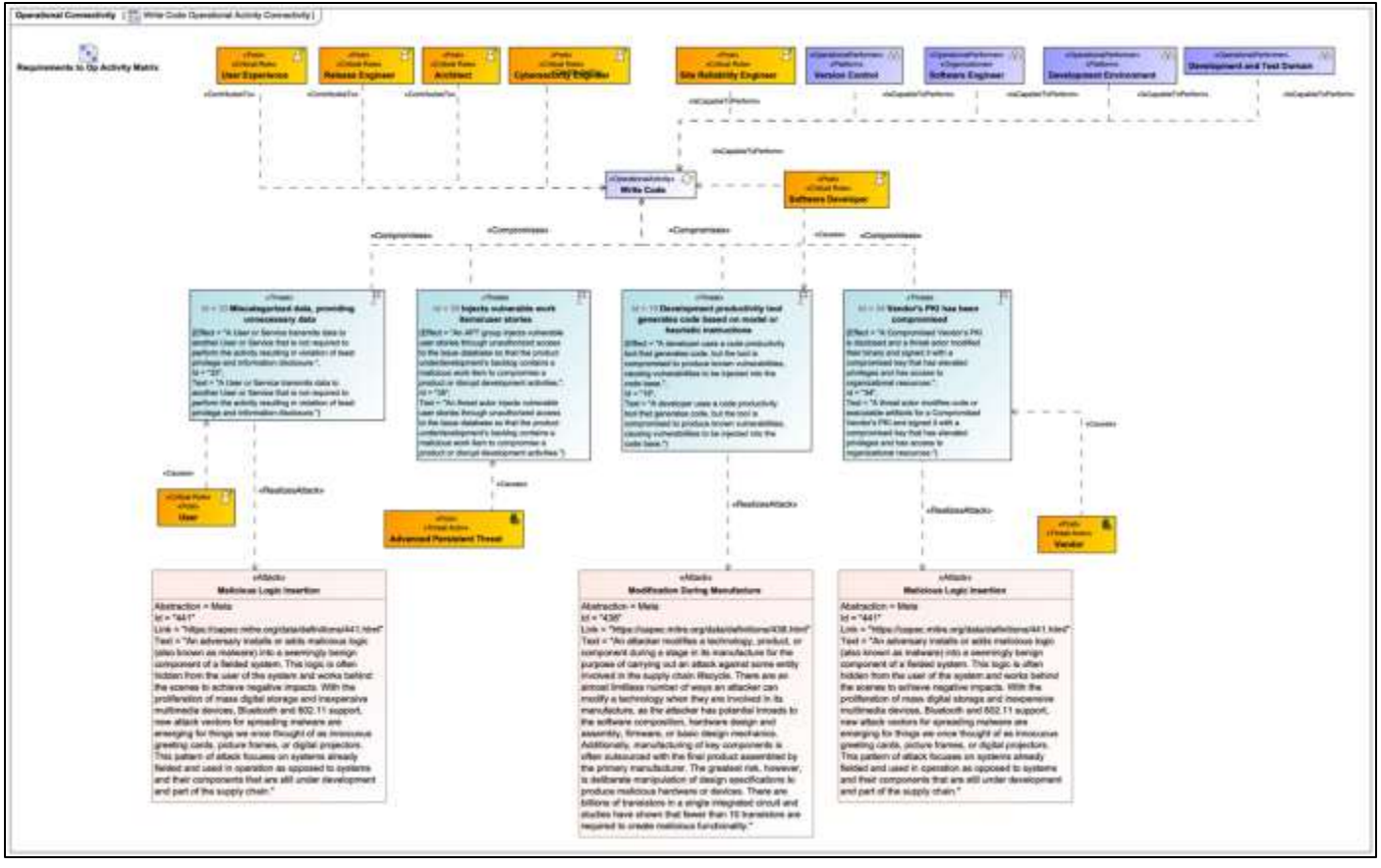
NOTE: PSM = Platform Specific Model

[1] DoD Reference Architecture Description, https://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

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

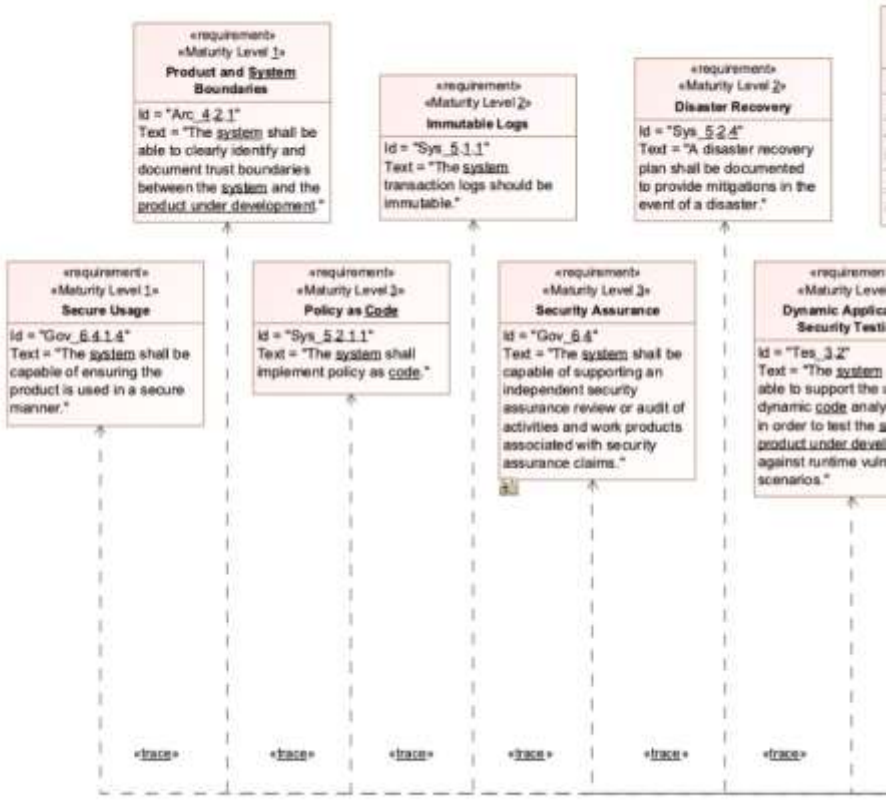
- specify the DevSecOps requirements to the lead system integrators tasked with developing a platform-specific solution that includes the designed system and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the 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

Example Threat Modeling Diagram for Write Code Operational Activity



[Write Code Operational Activity Connectivity Link]

Requirements



Example of Requirements Representation in Diagrams from PIM

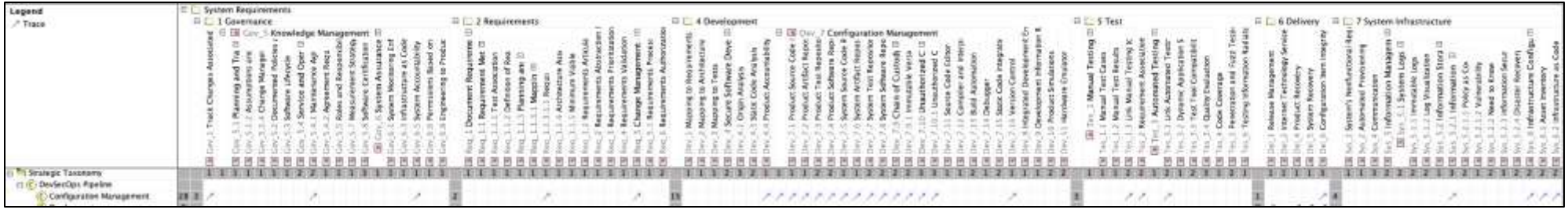
Requirements are organized into categories based on logical and functional groupings

Requirements Table Link

Capability/Strategic Viewpoint

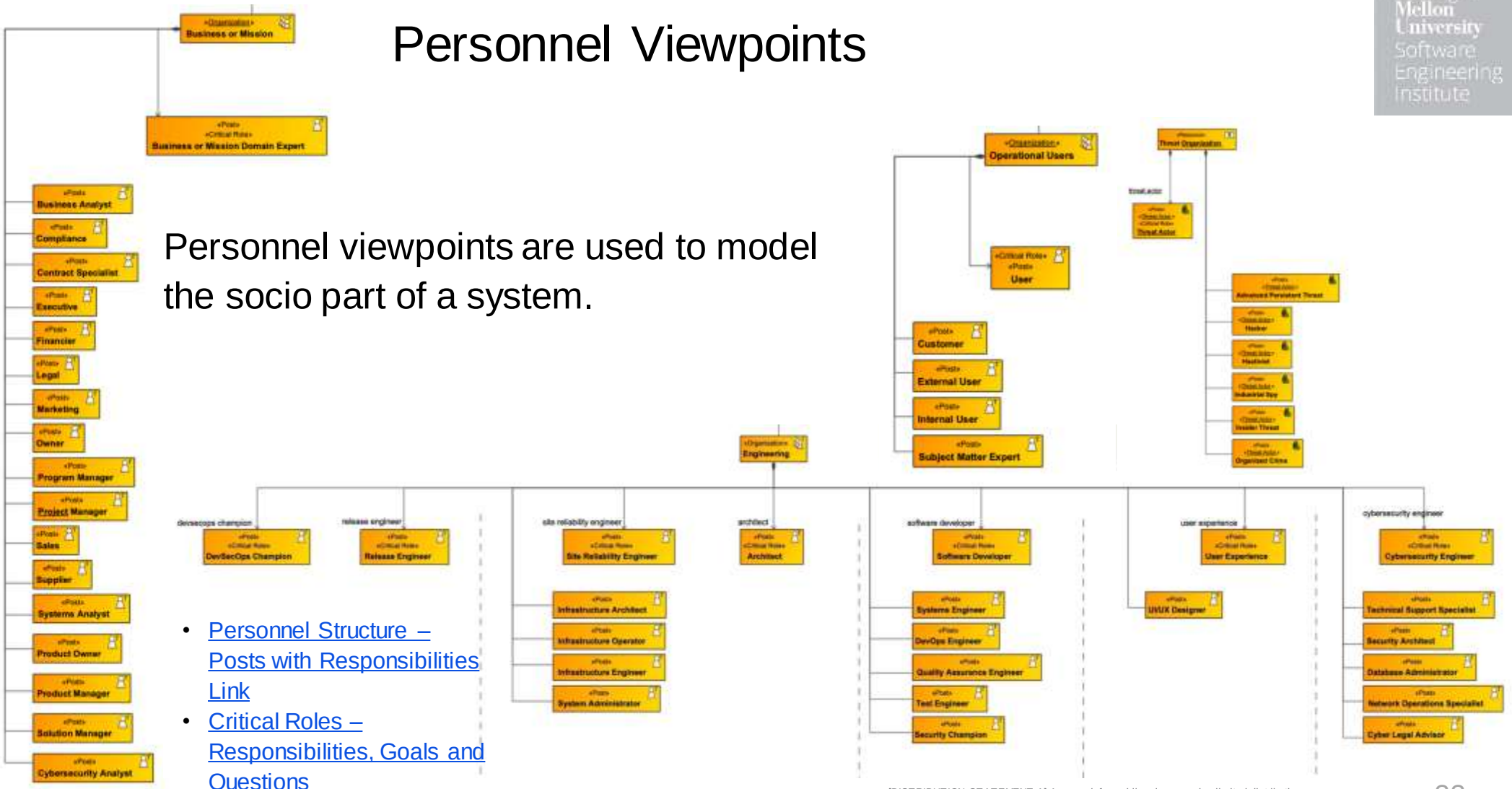
A capability is a high-level concept that describes the ability of a system to achieve or perform a task or a mission.

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



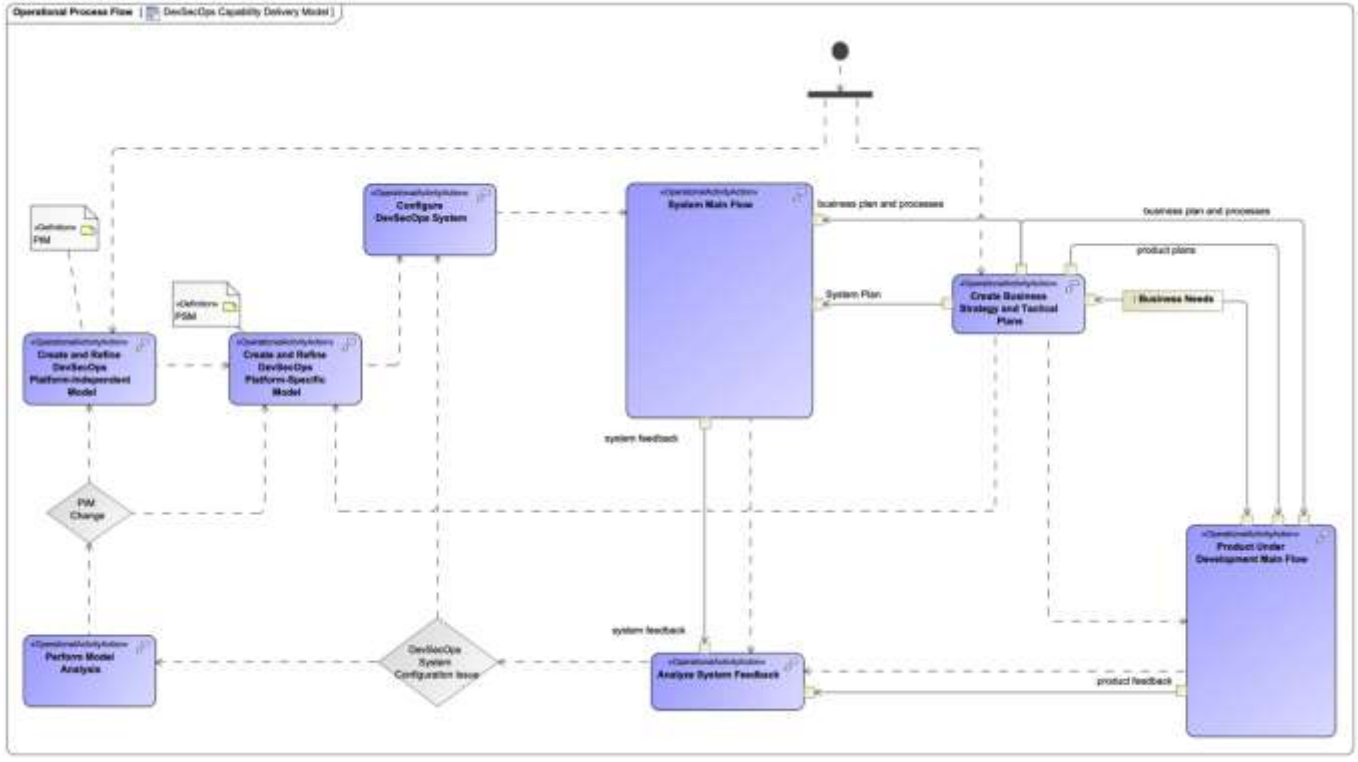
Personnel Viewpoints

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



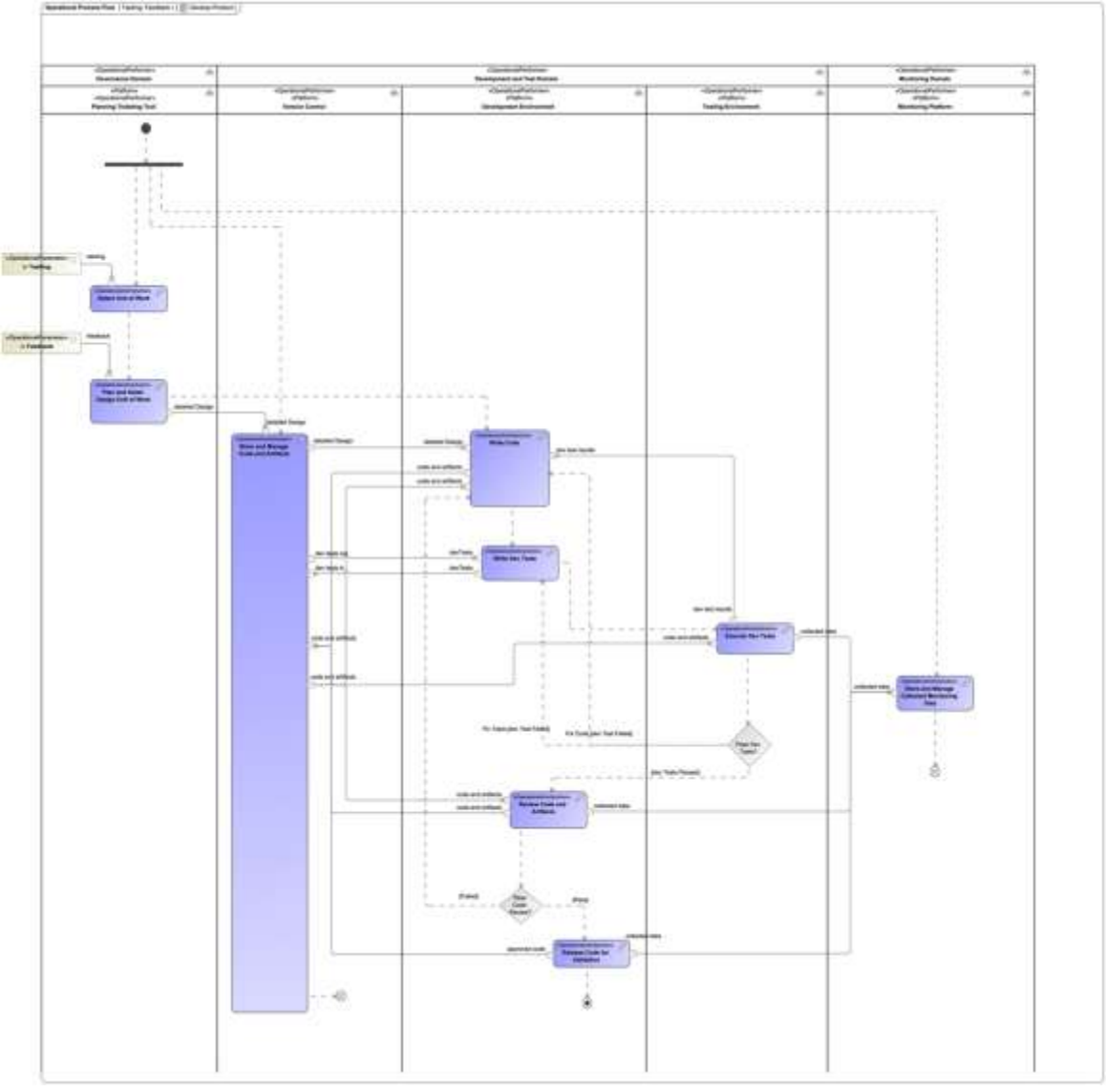
- Personnel Structure – Posts with Responsibilities Link
- Critical Roles – Responsibilities, Goals and Questions

Operational Viewpoints



- DevSecOps Capability Delivery Model Link

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



Operational Process Flow Focus Area

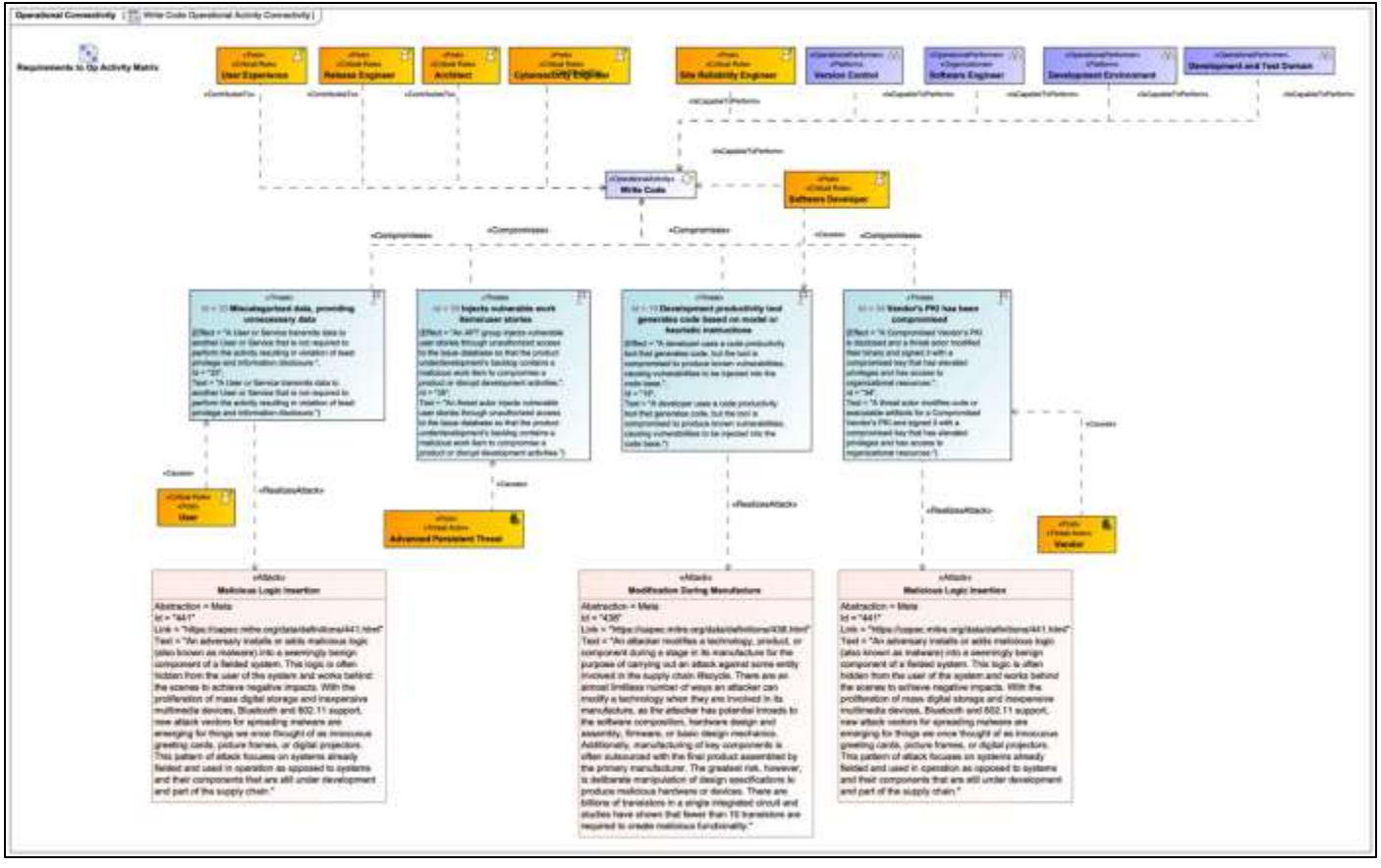
- Select an operational process flow to focus the threat scenario generation
- Review the selected operational process flow to gain understanding of the process, data flow between operational activities, and performers involved
- This may include reviewing associated requirements to understand the scope and context of the various operational activities

Six part Threat Scenario

STATEMENT TEMPLATE: An [ACTOR] performs an [ACTION] to [ATTACK] an [ASSET] to achieve an [EFFECT] and/or [OBJECTIVE].

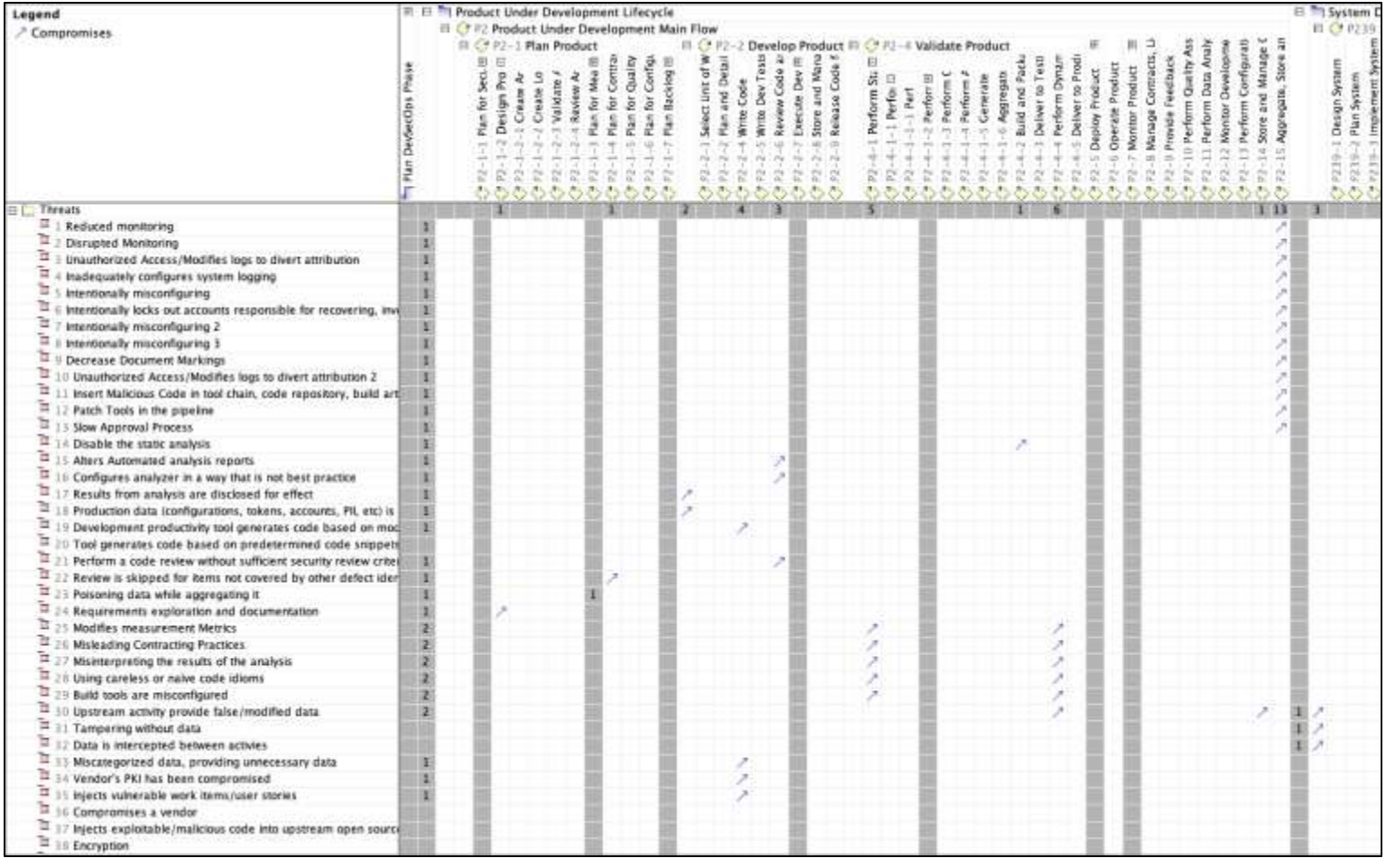
Part	Description
Actor	The person, or group, that is behind the threat scenario. Threat actors can be malicious or unintentional. Developing a standard set of actors is beneficial for this step. Persona non grata could be useful in determining malicious actors. Threat actor may be a person, or group, internal to an organization structure.
Action	A potential occurrence of an event that might damage an asset, a mission, or goal of a strategic vision.
Attack	An action taken that utilizes one of more vulnerabilities to realize a threat to compromise or damage an asset, a mission, or goal of a strategic vision.
Asset	A resource, person, or process that has value.
Effect	The desired or undesired consequence resulting from the attack.
Objective	The threat actor's motivation or objective for conducting the attack

Example Threat Modeling Diagram for Write Code Operational Activity



[Write Code Operational Activity Connectivity Link]

Threat to Operational Activity Matrix



[Threats to Operational Activities Link]

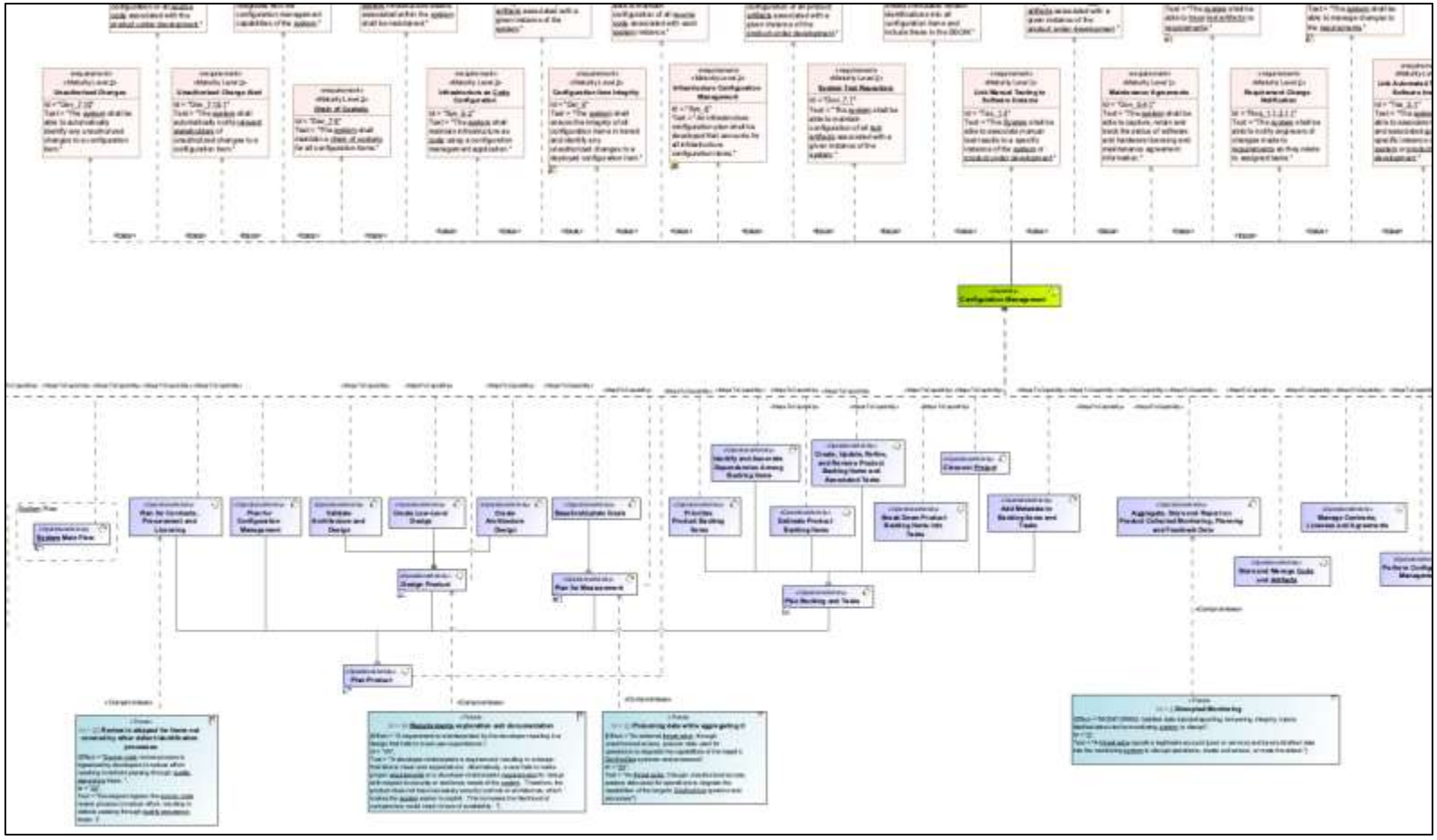
Threats with Attributes

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		

[Threats Link]

Capturing the Complexity of the System



Example of Threats Traced to Capabilities via Operational Activities

[Configuration Management Complexity Link]

Summary

The goal of every program is to deliver a solution 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

Security by design is achieved through integrating defensive thinking throughout the entire lifecycle.

Contact Information

Timothy A. Chick
Applied Systems Group Technical Manager, CMU-Software Engineering Institute
Adjunct Faculty Member, CMU-Software and Societal Systems Department

tchick@sei.cmu.edu

https://s3d.cmu.edu
https://www.sei.cmu.edu