Does Your DevSecOps Pipeline Only Function as Intended?

Timothy A. Chick

CERT Technical Manager, Applied Systems Group

Carnegie Mellon University – Software Engineering Institute

Document Markings

Agenda

About DevSecOps

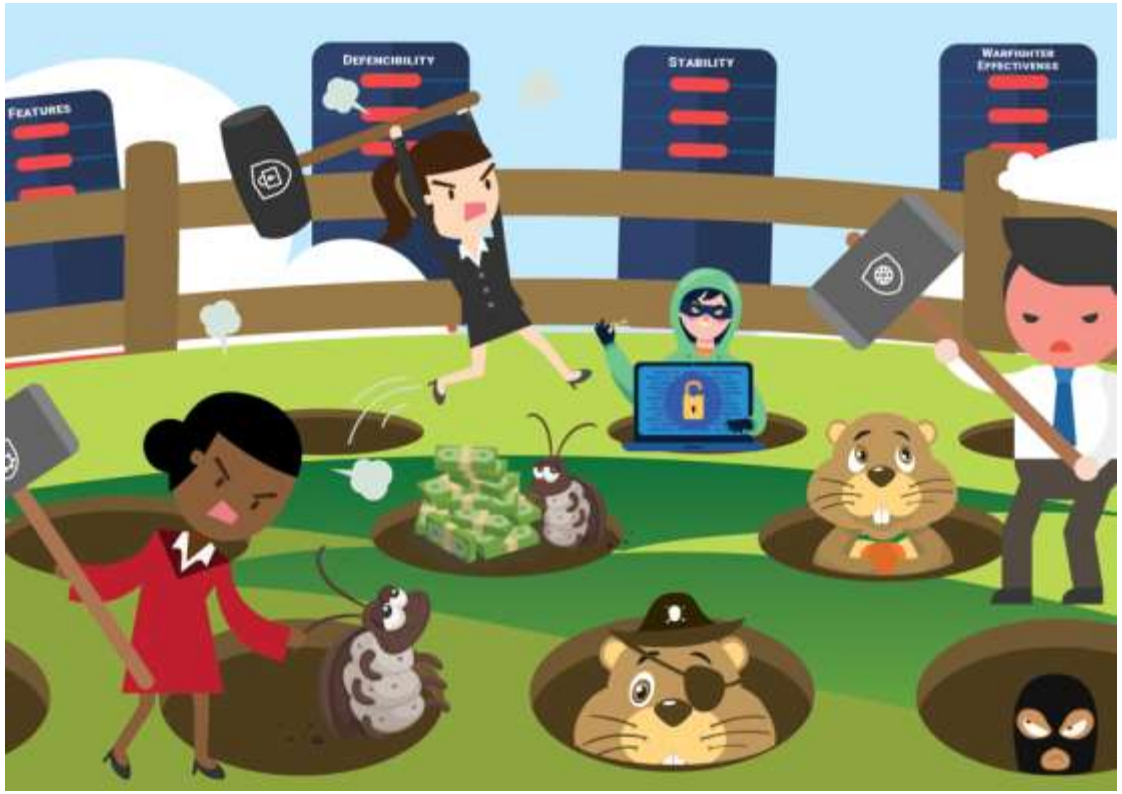
Challenges associated with DevSecOps

- Challenge 1: Connecting process, practice, and tools
- Challenge 2: Cybersecurity of pipeline and product

Addressing the Cybersecurity challenges with MBSE

About DevSecOps

Today's Whac-A-Mole Approach



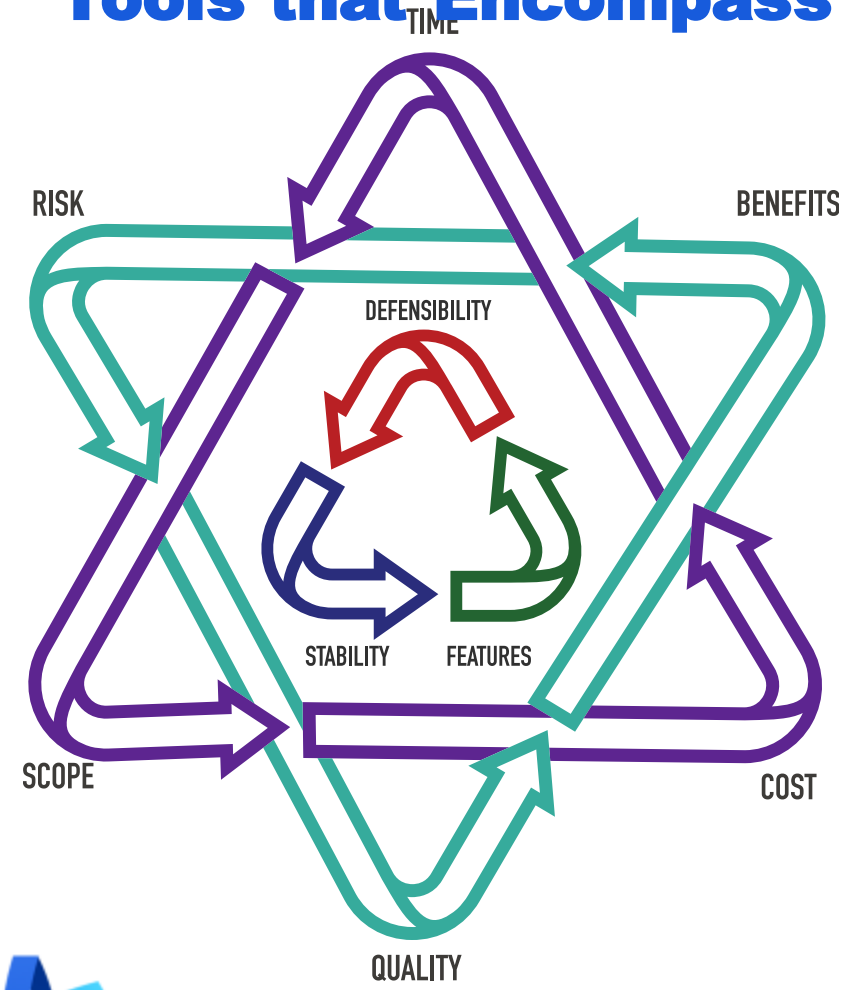
Winning in Features and Effectiveness, but Losing in Defensibility and Stability

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.

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].

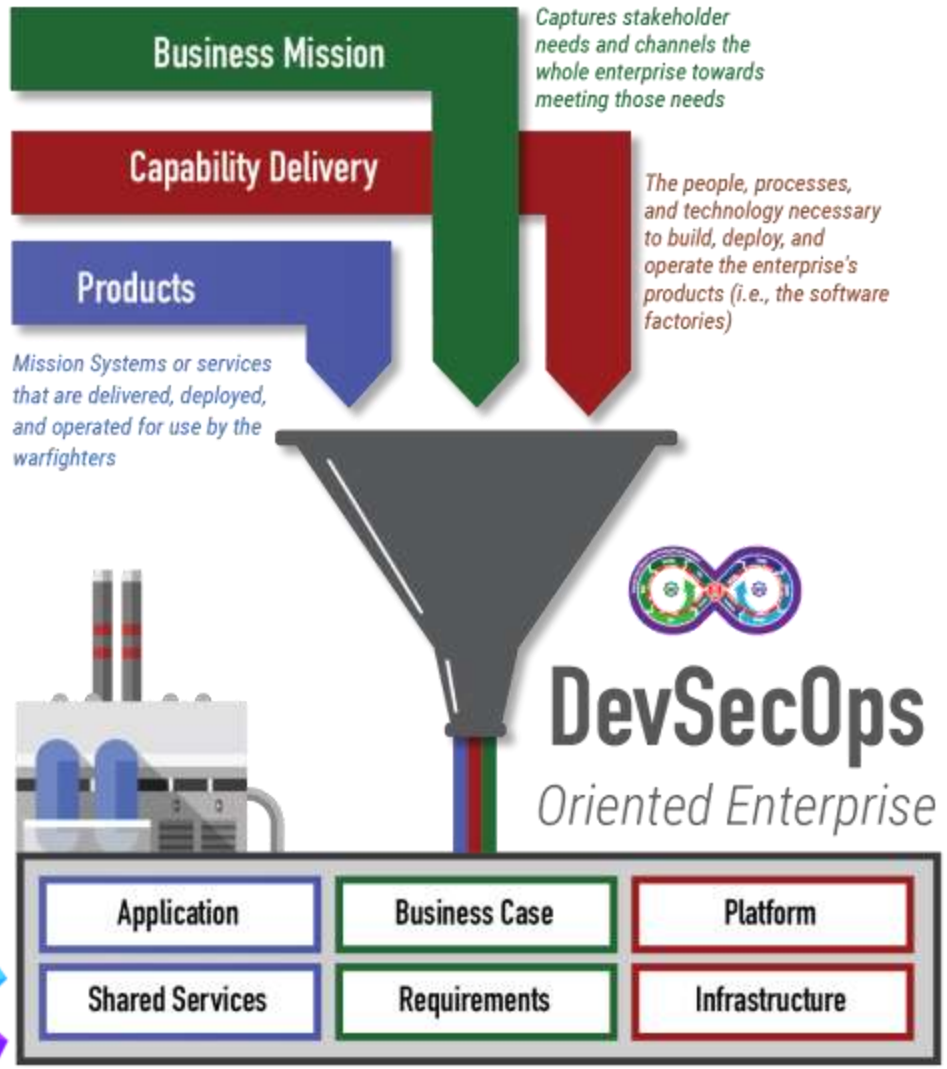
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

- **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

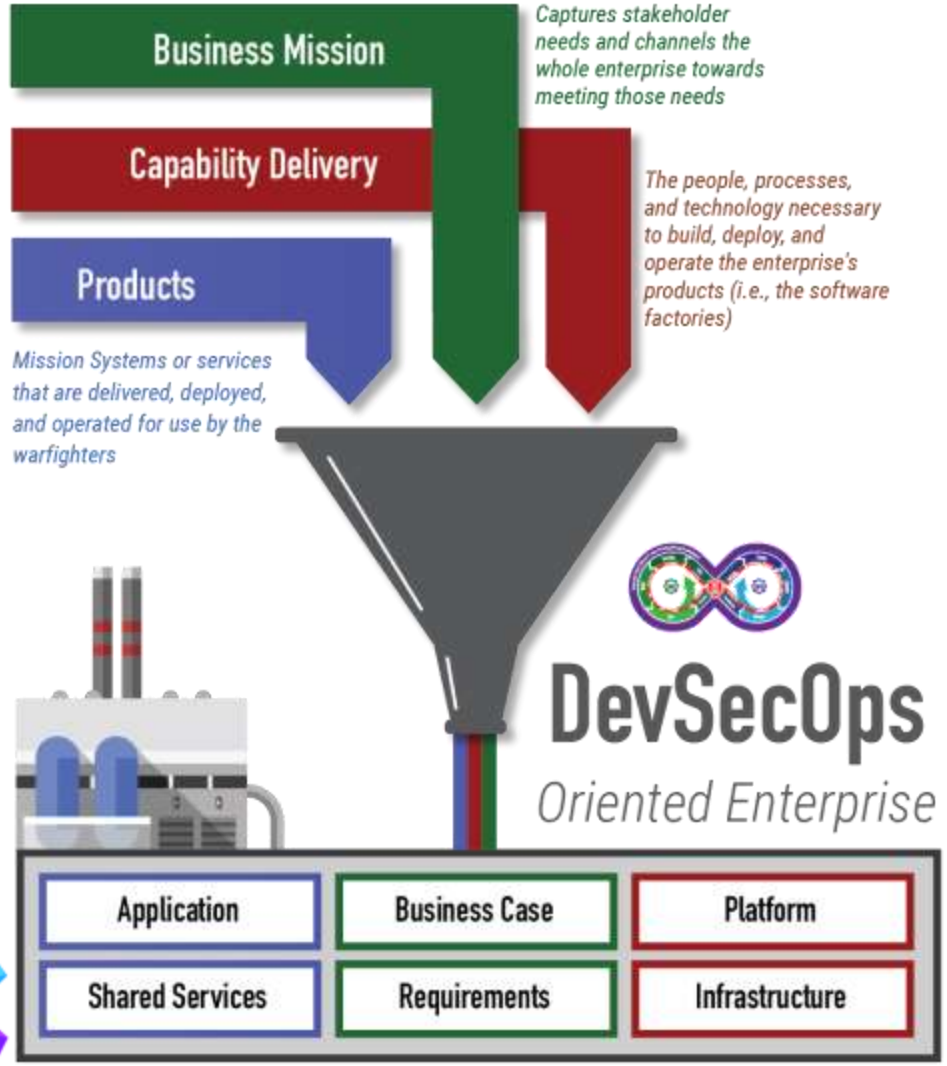


All DevSecOps-oriented enterprises are driven by three concerns:

- **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** – the units of value delivered by the organization. Products utilize the capabilities delivered by the software factory and operational environments.

Challenges Associated with DevSecOps

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

Many valid approaches to implementation[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…

Many Valid Approaches to Implementation[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.

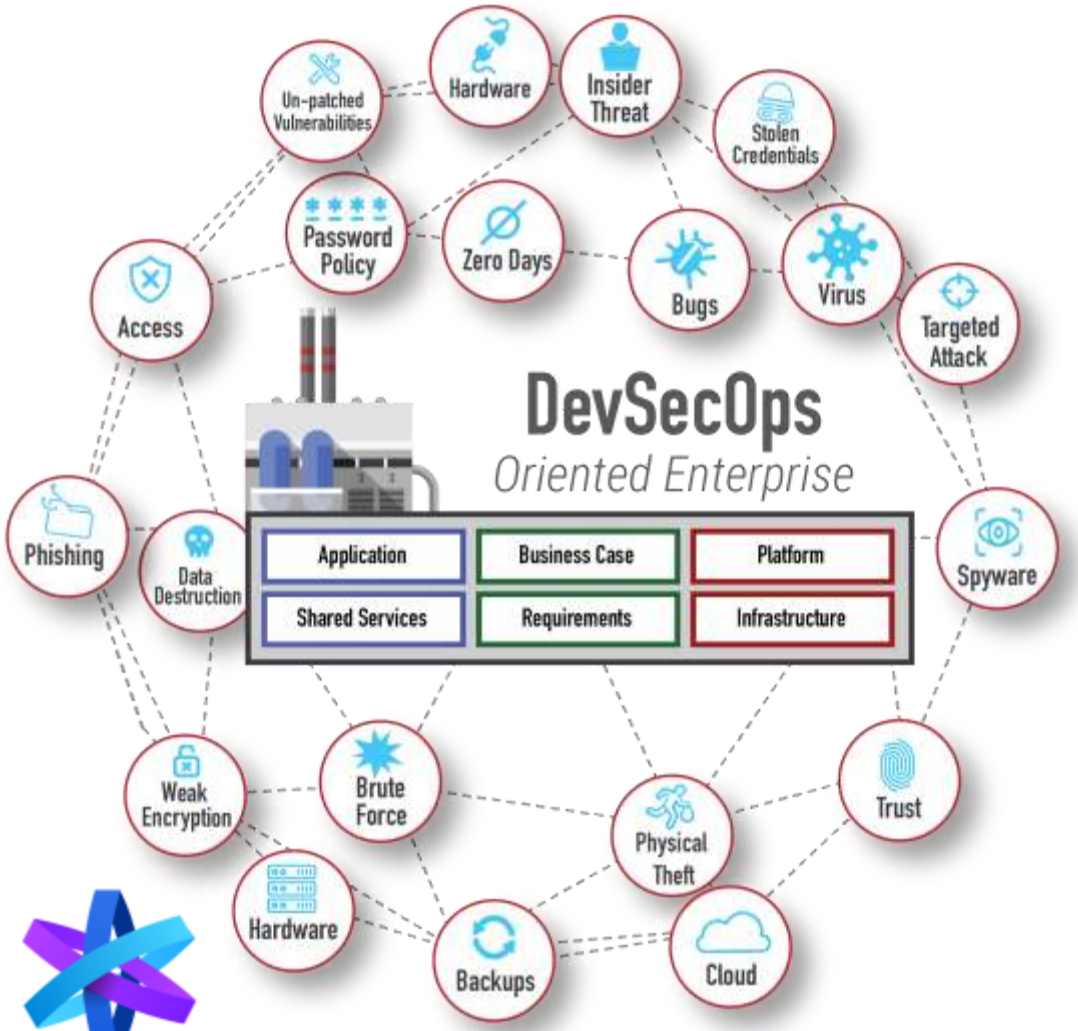
Selecting the Appropriate Techniques

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.

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?

Software Assurance (SwA)

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 perfect 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

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.

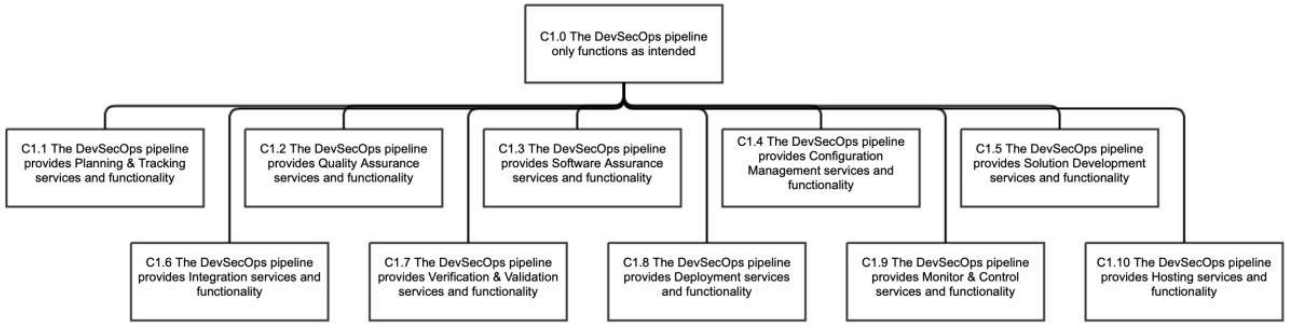
- It is a structured approach used to argue that available evidence supports a given claim
- It provides the organization with the basis for making risk-based choices tied to assuring that the pipeline only functions as intended.
- It provides requirements for automated systems testing, or other evidence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

How Is It Done Today, and What Are the Limits of Current Practice?

- Currently, guidance and policies focus on functionality and leave the major decision making to the programs:
  - "DoD organizations should define their own processes, choose proper activities, and then select tools suitable for their systems to build software factories and DevSecOps ecosystems" [1]
  - "The PM shall ensure that software teams use iterative and incremental software development methodologies (such as Agile or Lean), and use modern technologies (e.g. DevSecOps pipelines) … " [2]
- Program offices lack sufficient capability to design, build, and implement a DevSecOps continuous integration/continuous delivery (CI/CD) pipeline.
- Current guidance
  - fails to prepare the program office to address the full socio-technical aspects of DevSecOps
  - is not definitive and require a considerable amount of interpretation, resulting in:
    - DevSecOps perspectives not being fully integrated in DoD guidance and policy documents
    - Program offices being unable to perform an analysis of alternative (AoA) in regards to the DevSecOps pipeline tools and processes
    - Multiple programs using similar infrastructure and pipelines in different and incompatible ways, even within the same program
    - Suboptimal tools and security controls
- Large and complex DoD weapon system acquisition programs have already embraced model-based engineering, but have not applied the same techniques to their DevSecOps CI/CD pipelines. This limits a program office's ability to build a cyber-physical software factory that is fit for purpose.

[1] DoD Enterprise DevSecOps Reference Design, https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf?ver=2019-09-26-115824-583
[2] DoD Software Acquisition Pathway Interim Policy and Procedures, https://aaf.dau.edu/aaf/software/

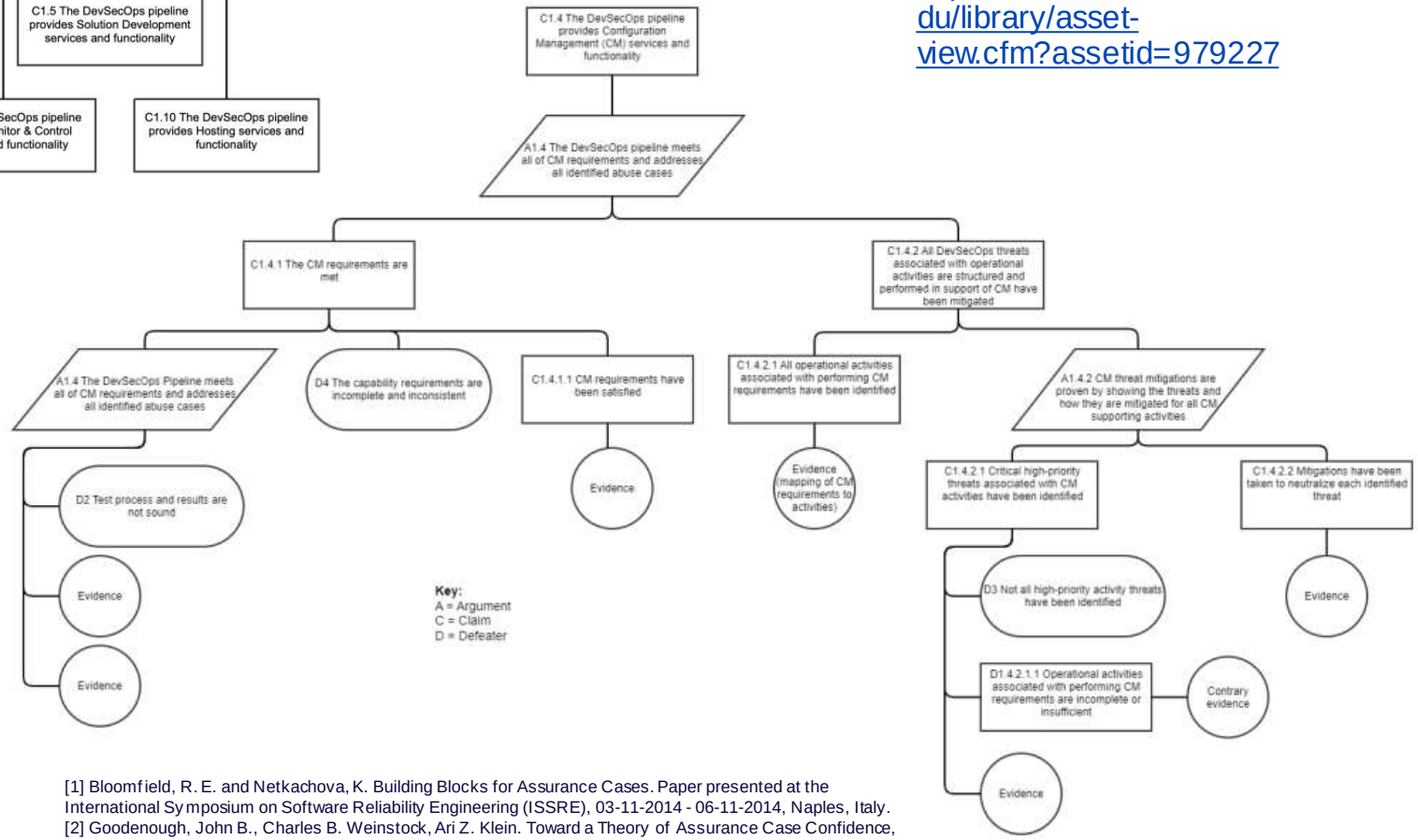
Structuring a DevSecOps Assurance Case



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]."

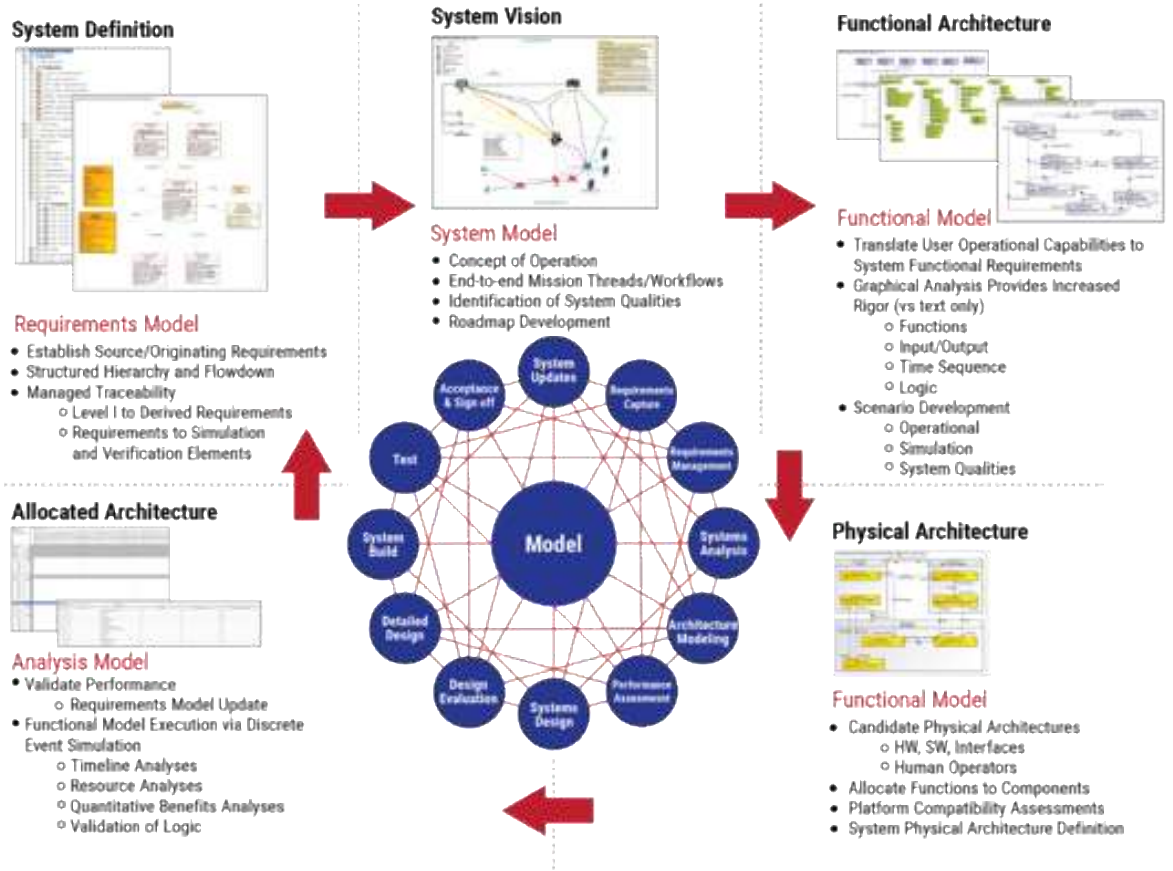
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227



[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.

Addressing the Cybersecurity Challenge with MBSE

Model Based Systems Engineering



- ***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

*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.

Why Apply Enterprise Architecture (EA) and Model-based Engineering to DevSecOps? - 1

The validity of using EA and model-based engineering approaches is based on an assertion that DevSecIos CI/CD pipelines are complex systems.

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

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

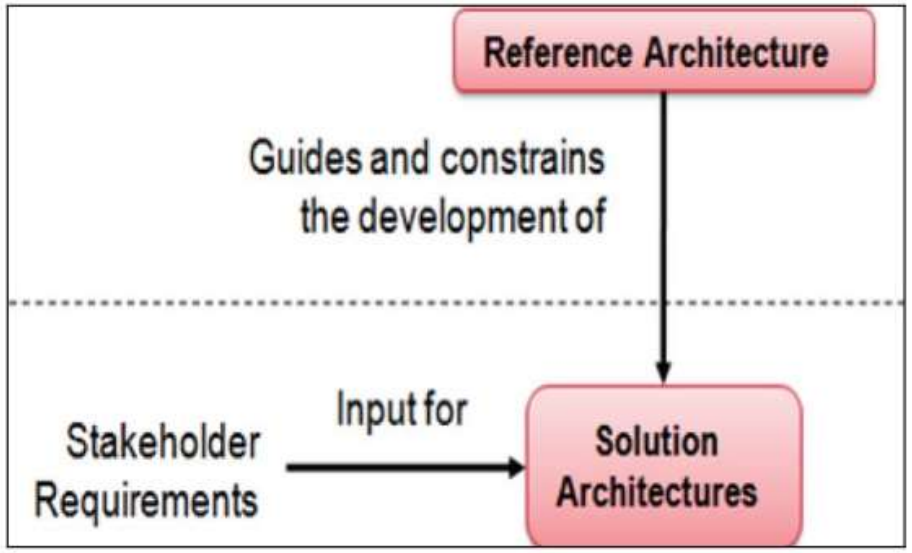
Why Apply Enterprise Architecture (EA) and Model-based Engineering to DevSecOps? - 2

- The idea of **applying model-based engineering methods to sociotechnical systems is not new**. Examples include; social systems [1] [2], complex command and control system [3], border security [4], sociotechnical systems (DoD's forward base camps and emergency response organizations) [5]
- The overall **adoption of model-based engineering and virtual modeling tools in everyday practices has grown**. Adopters include; Airbus, Boeing, Toyota, Lockheed Martin, Ford, P&G BAE Systems, Jet Propulsion Laboratory, MITRE, US Navy, US Army, Biotronik, Bernafon, Hospira, Philadelphia Insurance Companies, and many others [6].
- **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 [7].
- **EA and model-based engineering are the best practices** for designing and formalizing a description of a complex information system in a social context, **thus we propose to use them to create a DevSecOps reference architecture**.

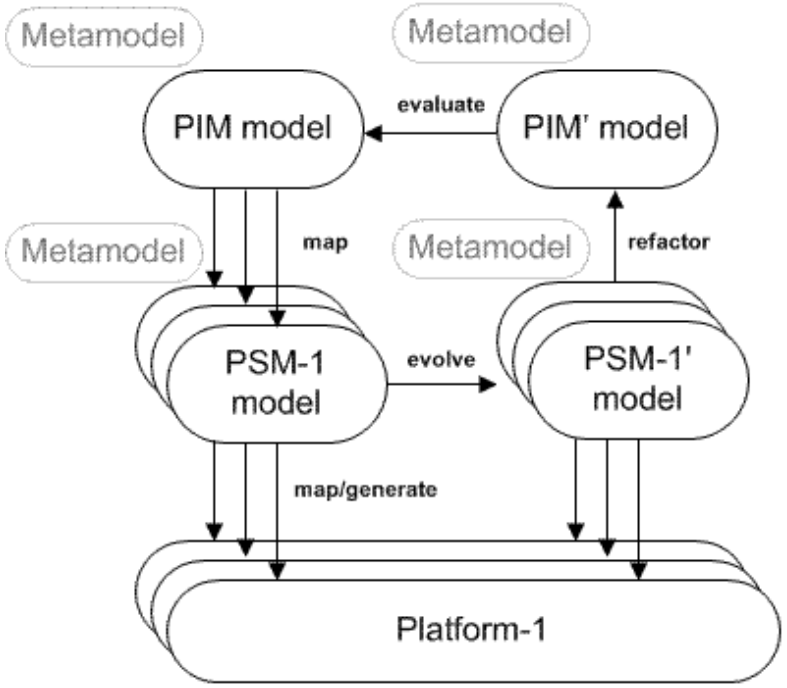
[1] Haskins, C. 2008a. "Using patterns to transition systems engineering from a technological to social context," Systems Engineering, 11: 147–155.
[2] Palmer, E. 2016. "Investigating structural gender inequality in the Norwegian pension system: An example of using MBSE in the evaluation of social systems," 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] Industry Expertise, https://www.nomagic.com/industry-expertise
[7] OMG Standards for BPM, https://www.omg.org/technology/readingroom/BPM.htm

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].



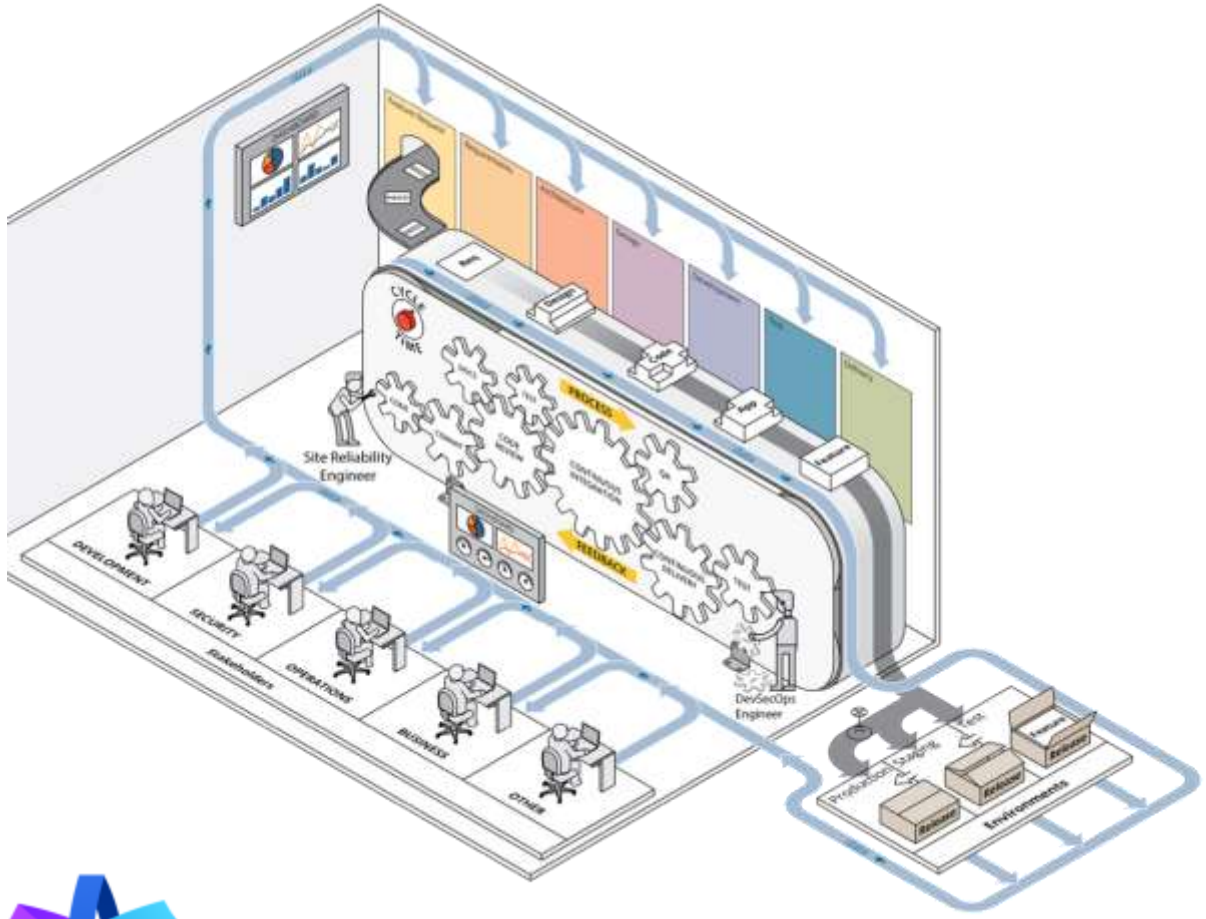
A PIM is a general and reusable model of a solution to a commonly occurring problem in software engineering within a given context and is independent of the specific technological platform used to implement it.



NOTE: PSM = Platform Specific Model

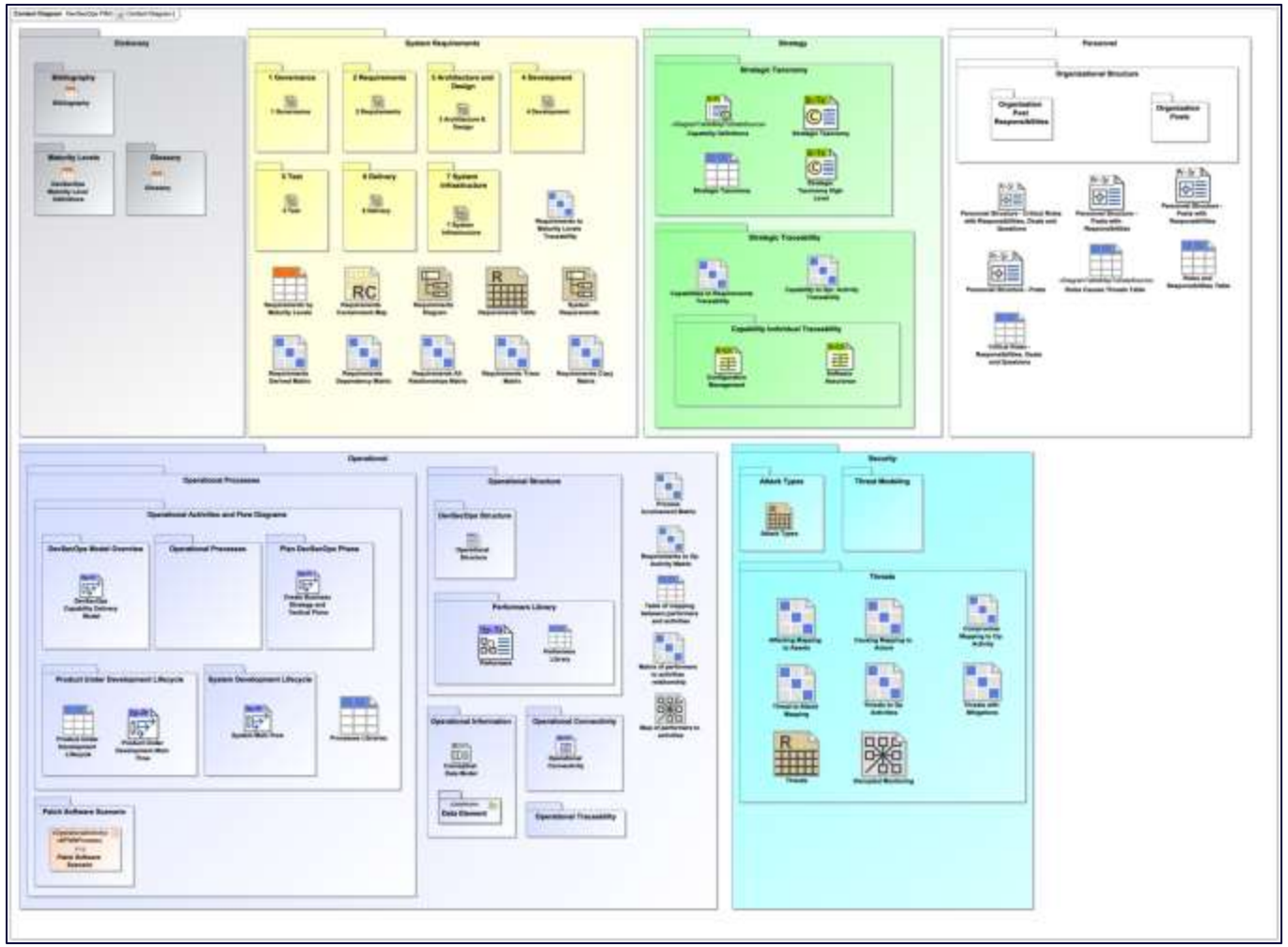
[1] DoD Reference Architecture Description, https://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf

DevSecOps Platform Independent Model (PIM)



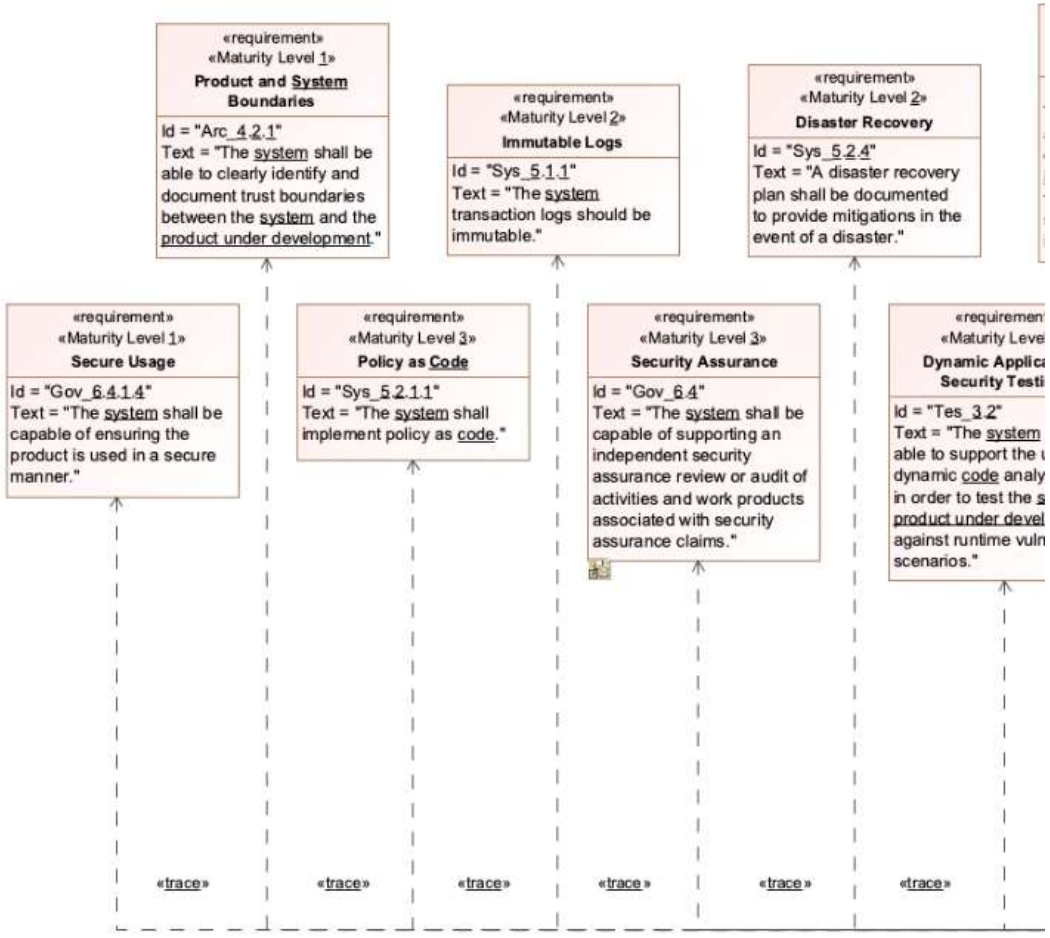
- is an authoritative reference to fully design and execute an integrated Agile and DevSecOps strategy in which all stakeholder needs are addressed
- enables organizations to implement DevSecOps in a secure, safe, and sustainable way in order to fully reap the benefits of flexibility and speed available 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/

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

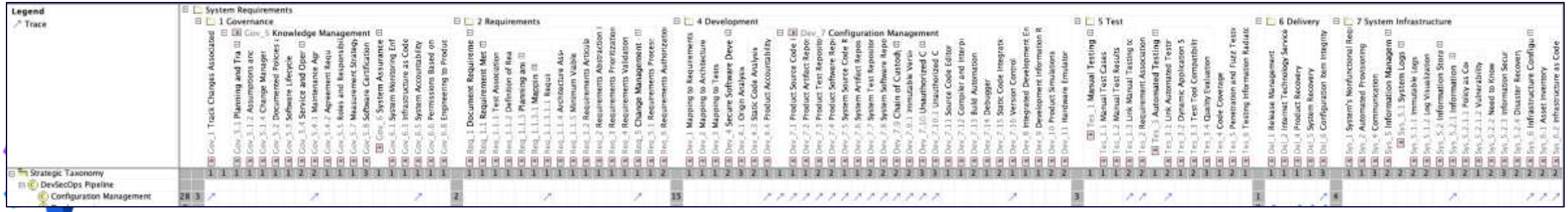
Requirements Table Link

DevSecOps Capability/Strategic Viewpoint

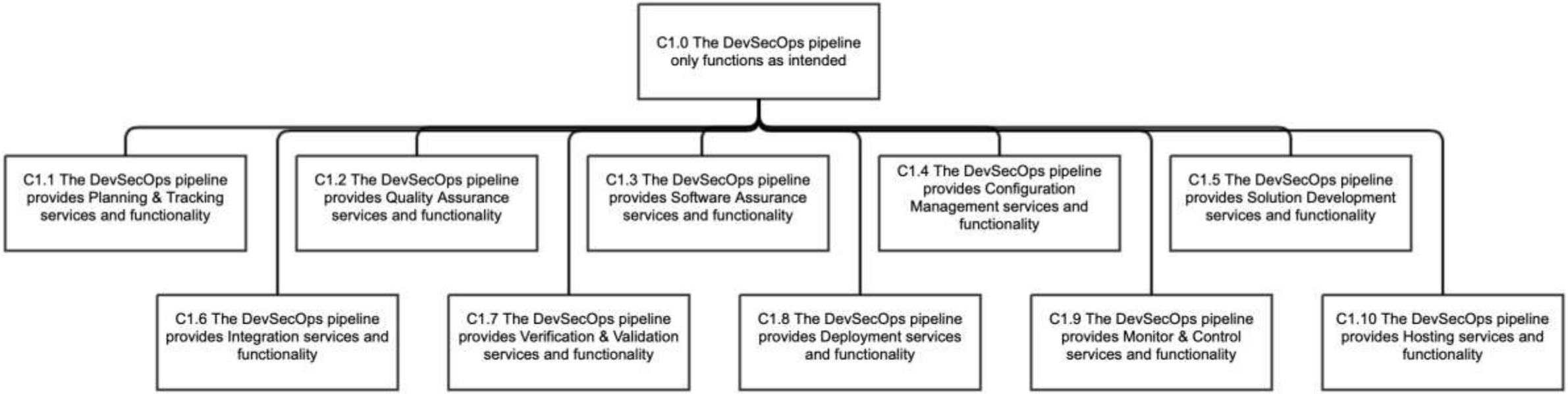
A capability is a high-level concept that describes the ability of a system to achieve or perform a task or a mission.

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

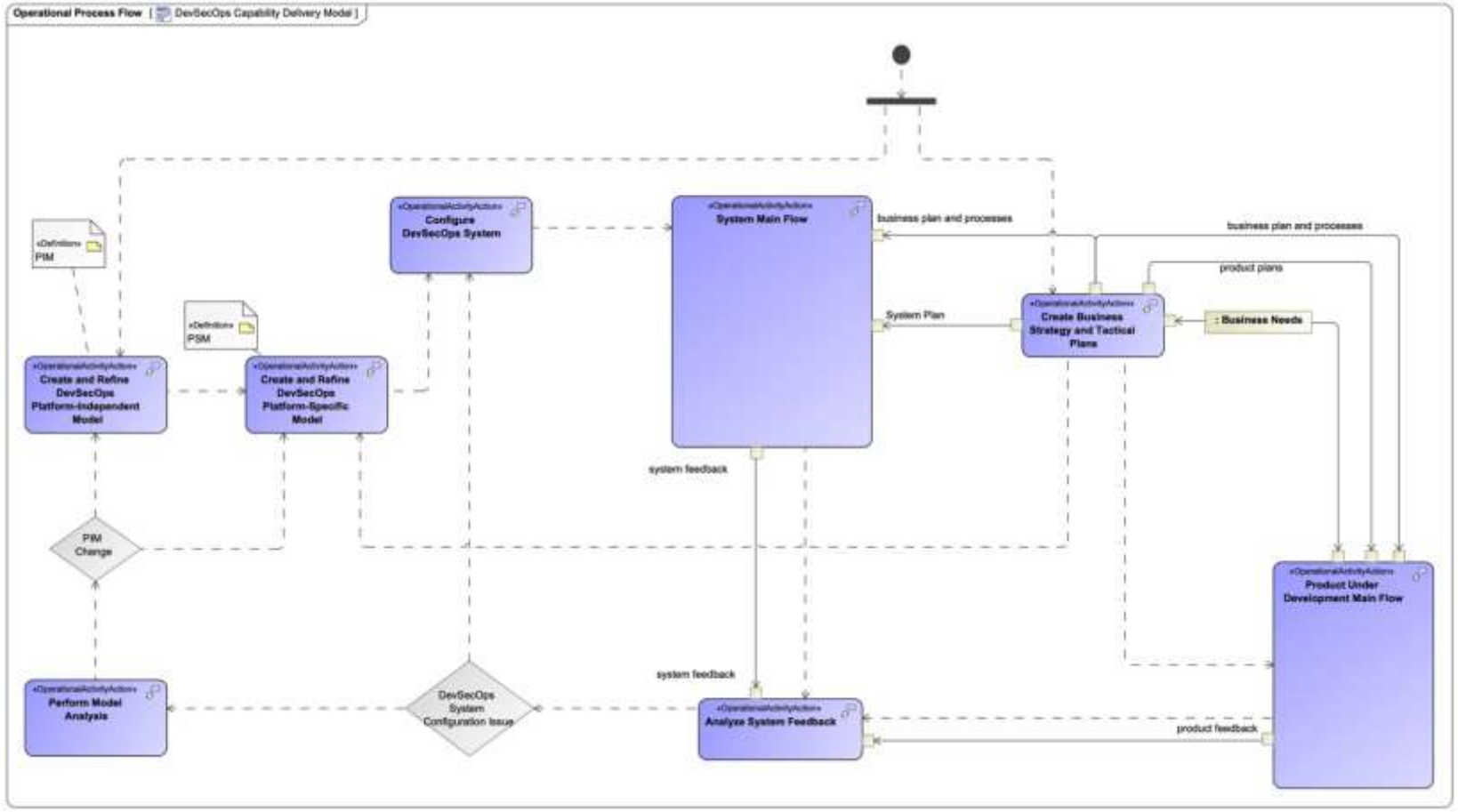


Structuring a DevSecOps Assurance Case Around Capability



https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227

DevSecOps Operational Viewpoints



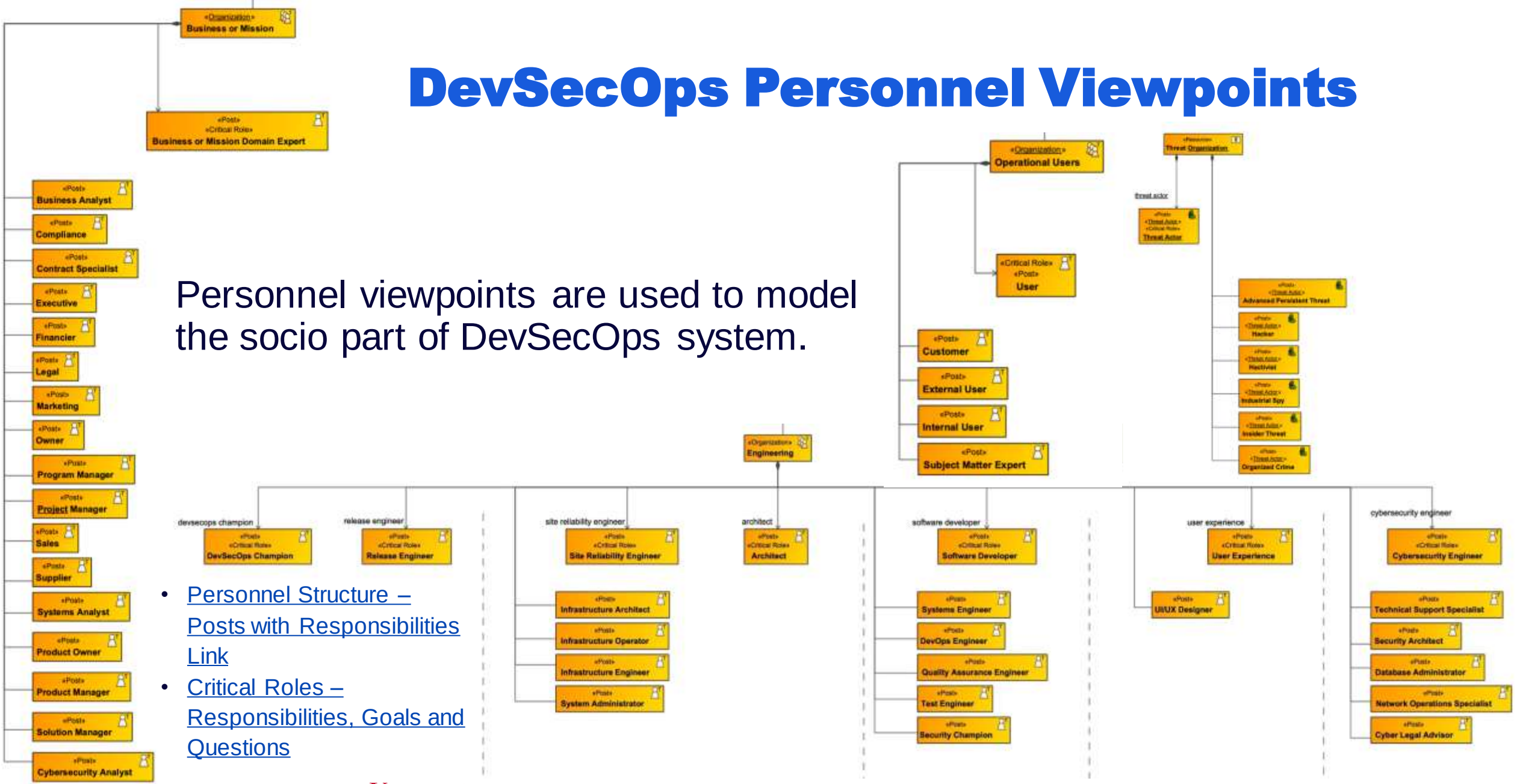
[DevSecOps Capability Delivery Model Link]

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

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

Legend
- Approves
- ContributesTo
- Is Capable To Perform
- Observes
- *Multiple (one-way)*

Organization Posts	Architect	Business Analyst	Business or Mission Domain I	Compliance	Contract Specialist	Customer	Cyber Legal Advisor	Cybersecurity Analyst	Cybersecurity Engineer	Database Administrator	DevOps Engineer	DevSecOps Champion	Executive	External User	Financier	Infrastructure Architect	Infrastructure Engineer	Infrastructure Operator	Internal User	Legal	Marketing	Network Operations Specialis	Owner	Product Manager	Product Owner	Program Manager	Project Manager	Quality Assurance Engineer	Release Engineer	Relevant Stakeholders	Sales	Security Architect	Security Champion	Site Reliability Engineer	Software Developer	Solution Manager	Subject Matter Expert	Supplier	System Administrator	Systems Analyst	Systems Engineer	Technical Support Specialist	Test Engineer	UI/UX Designer	User	User Experience
Operational Activities and Flow Diagrams	93		69	1	3				99		3	58								2				4	14			101					100	102					14					29	92
DevSecOps Model Overview	6		5						5			5																5					5	5										2	5
Plan DevSecOps Phase	17		16						15			15																15					15	16										7	15
Product Under Development Lifecycle	70		47	1	3				79		3	38								2				4	14			81					80	81					14					20	72
P2 Product Under Development Main Flow																																													
P2-1 Plan Product	40		23		2				40		1	18								2				4	11			41					41	41					14					15	40
P2-2 Develop Product	10		3						10			4																11					9	10											10
P2-4 Validate Product	2		1						4															1				4					5	5											3
P2-5 Deploy Product									6		2													2				6					6	5										2	
P2-6 Operate Product	7		↙						↙																			↙					↙	↙										↙	↙
P2-7 Monitor Product	11		11						11			11																11					11	11											11
P2-8 Manage Contracts, Licenses and Agreements	8	↙	↙		↙				↙																			↙					↙	↙											↙
P2-9 Provide Feedback	9	↙	↙						↙			↙																↙					↙	↙										↙	↙
P2-10 Perform Quality Assurance	9	↙	↙	↙					↙			↙																↙					↙	↙											↙
P2-11 Perform Data Analysis	8	↙	↙						↙			↙																↙					↙	↙											↙
P2-12 Monitor Development and Test Environment	7	↙	↙						↙																			↙					↙	↙											↙
P2-13 Perform Configuration Management	2		↙																														↙											
P2-14 Store and Manage Code and Artifacts	8	↙	↙						↙			↙																↙					↙	↙											↙
P2-15 Aggregate, Store and Report on Product Collected Monitoring, Pl	9	↙	↙						↙			↙																↙					↙	↙										↙	↙

Critical Roles are mapped to Operational Activities.

Process Involvement Matrix Link

Threat Scenarios

Template:

Part	Description
Activity	The activity diagrammed in the PIM or of a well-understood instantiated DevSecOps pipeline. There can be more than one activity applied to the threat scenario.
Actor	The person or group that is behind the threat scenario. Threat actors can be malicious or unintentional, and they may be a person or group internal to an organization structure. Developing a standard set of actors is beneficial for this step. Persona non grata could be useful in determining malicious actors.
Action	A potential occurrence of an event that might damage an asset, a mission, or a 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.
Statement	Structured prose summarizing the six-part security scenario.

Example:

Part	Description
Activity	Develop product, static and dynamic analysis
Actor	Insider threat
Action	Results from analysis are disclosed for effect
Attack	Information disclosure
Asset	Analysis results
Effect	Damaged organization, vulnerabilities publicly enumerated for a product under development
Objective	Develop a targeted exploit for the product under development, financial attack
Statement	An insider threat publicly releases the results of static and dynamic analysis to the public to damage the organization's reputation.

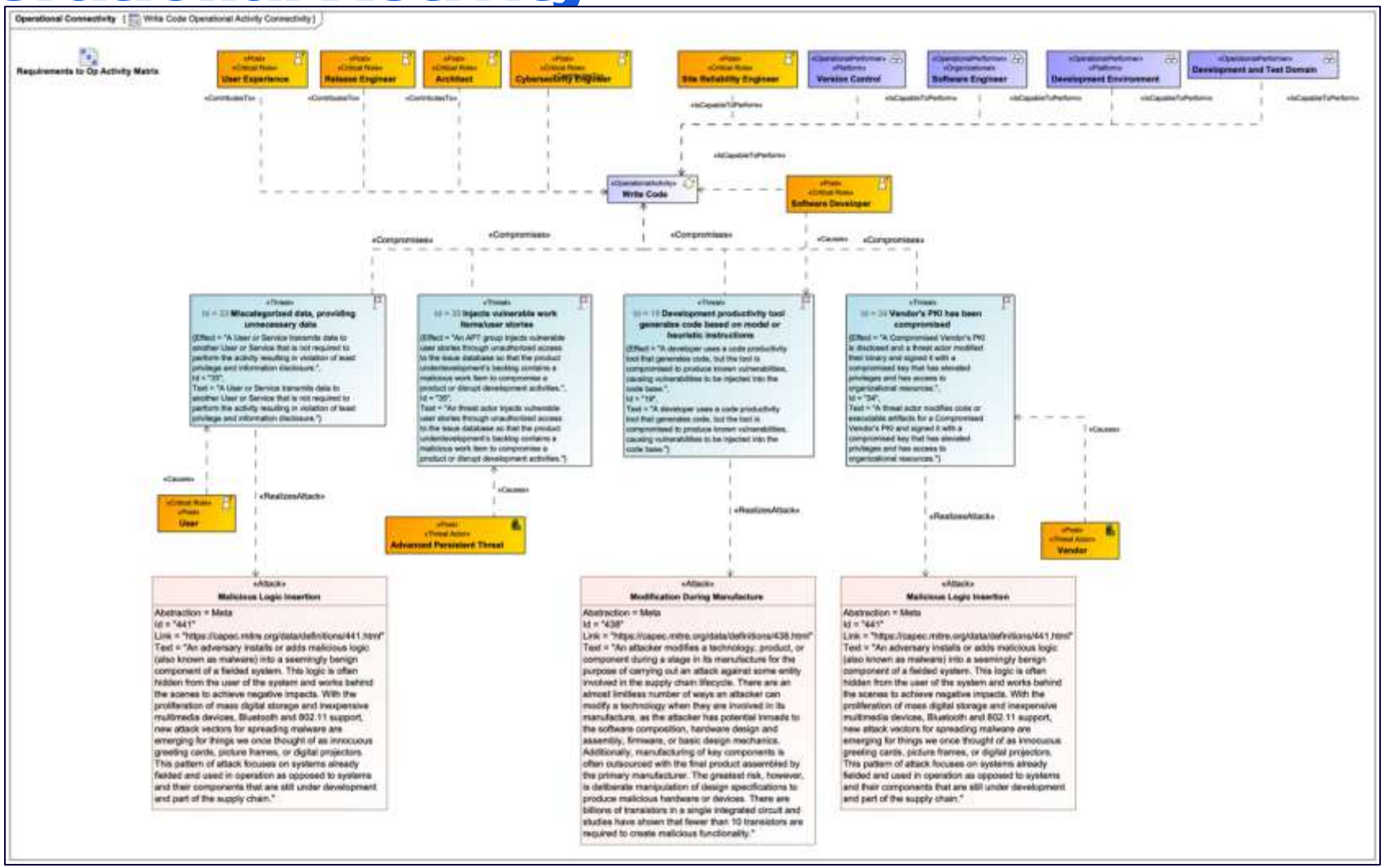
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227

Threat Scenario Generation Workshop

Purpose	Capture threats and mitigations in UAF views.	
Entry criteria	The following UAF-defined views have been created for the system under evaluation: • requirements diagrams • operational-process flows • personnel structure for the operational organization • relationships between operational activities and system requirements • relationships between operational activities and posts (i.e., critical roles), including the involvement relationships	
General	As the system architecture and associated system instantiation evolves, so will the threats and corresponding mitigations. Threats should be identified, evaluated, and captured continuously outside this process.	
Step	**Activities**	**Description**
1	Identify threats	Complete threat-scenario templates for the operational activities within the UAF model using the threat-scenario generation workshop.
2	Model threat scenarios	Create threat elements in the model for each threat scenario using the threat-modeling profile: • Use "action" field as the threat's name. • Use the "statement" field as the threat's text. • "Effect" and "objective" should be mapped to the corresponding threat's attributes. • "Threat's ID" may be autogenerated by the modeling digital tool or have a preset custom structure.
3	Identify attacks for the threats	Formalize the "attack" statement for each threat scenario. Use one of the industry standards, like MITRE CAPEC, as a guide. One threat scenario can be realized by more than one attack pattern.
4	Model attacks	• For each identified attack, create an "attack" element in the model using the threat-modeling profile. • Connect attacks with corresponding threats with the "RealizesAttack" relationship (from threat to attack).
5	Identify post associated with causing and mitigating threat	• By analyzing the "actor" field in each threat scenario, identify if the actor is internal or external. • If internal, determine which posts from the personnel viewpoint's organizational structure, preferably marked as a "critical role," will correspond with the actor. Identified posts should have one of the involvement relationships with compromised operational activity; most likely, it is a performer. If there are no posts in the operational organization structure that can be the actor in the scenario, then a new post needs to be created in the model. • If the actor is external to the system's organization, then it needs to be created in the model in a separate package from the personnel viewpoint, and the "threat actor" stereotype from the threat-modeling profile needs to be applied to it. • Identify which post from the organization structure should be responsible for creating a mitigation strategy for the given threat. Identified posts may have one of the involvement relationships with the compromised operational activity; most likely, it is an observer.
6	Model post-to-threat associations	• In the personnel viewpoint, create new posts that were identified during step 5. • For each threat element, create a "causes" relationship (from the threat-modeling profile) with each post that represents an actor. • For each threat element, create an "OwnsRisk" (UAF) relationship with each post that was identified as responsible for the mitigation strategy for the threat in step 5.
7	Create operational connectivity diagrams	• Using an operational connectivity diagram, create a 360 degree view for each operational activity, and display on it – corresponding posts that have the involvement relationships with the activity – all threats that compromise the activity – attacks that realize those threats – threat actors/posts that cause the threat – posts that own (responsible for mitigation strategy) each threat – if identified, elements of security viewpoint or systems architecture that mitigate each threat – if identified, elements of systems architecture that perform the activity, implement it, or have other relationships with the activity
Exit Criteria		Create operational connectivity views for each operational activity with identified threats and associated metadata.

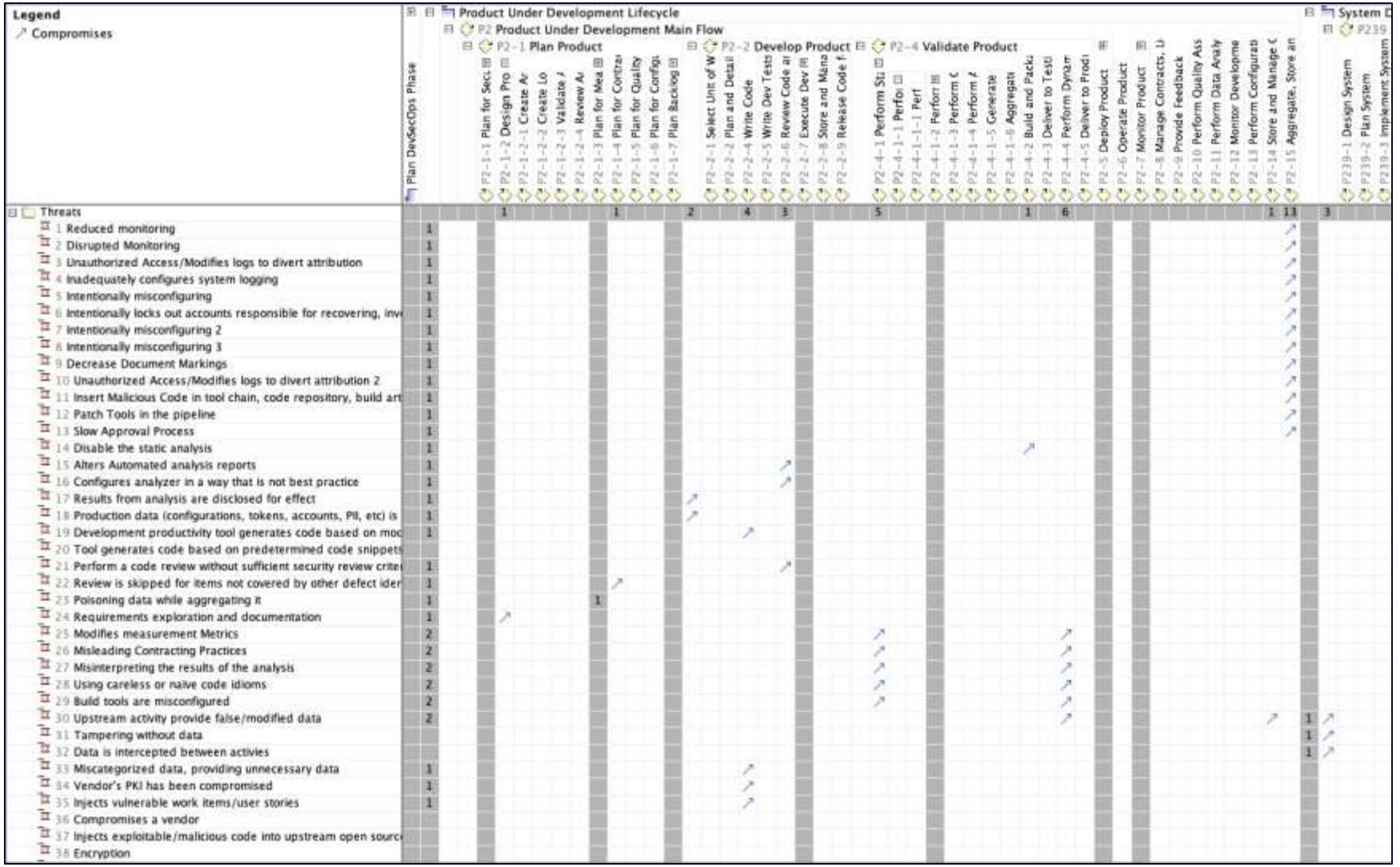
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227

Example Threat Modeling Diagram for Write Code Operational Activity



[Write Code Operational Activity Connectivity Link]

DevSecOps Threat to Operational Activity Matrix



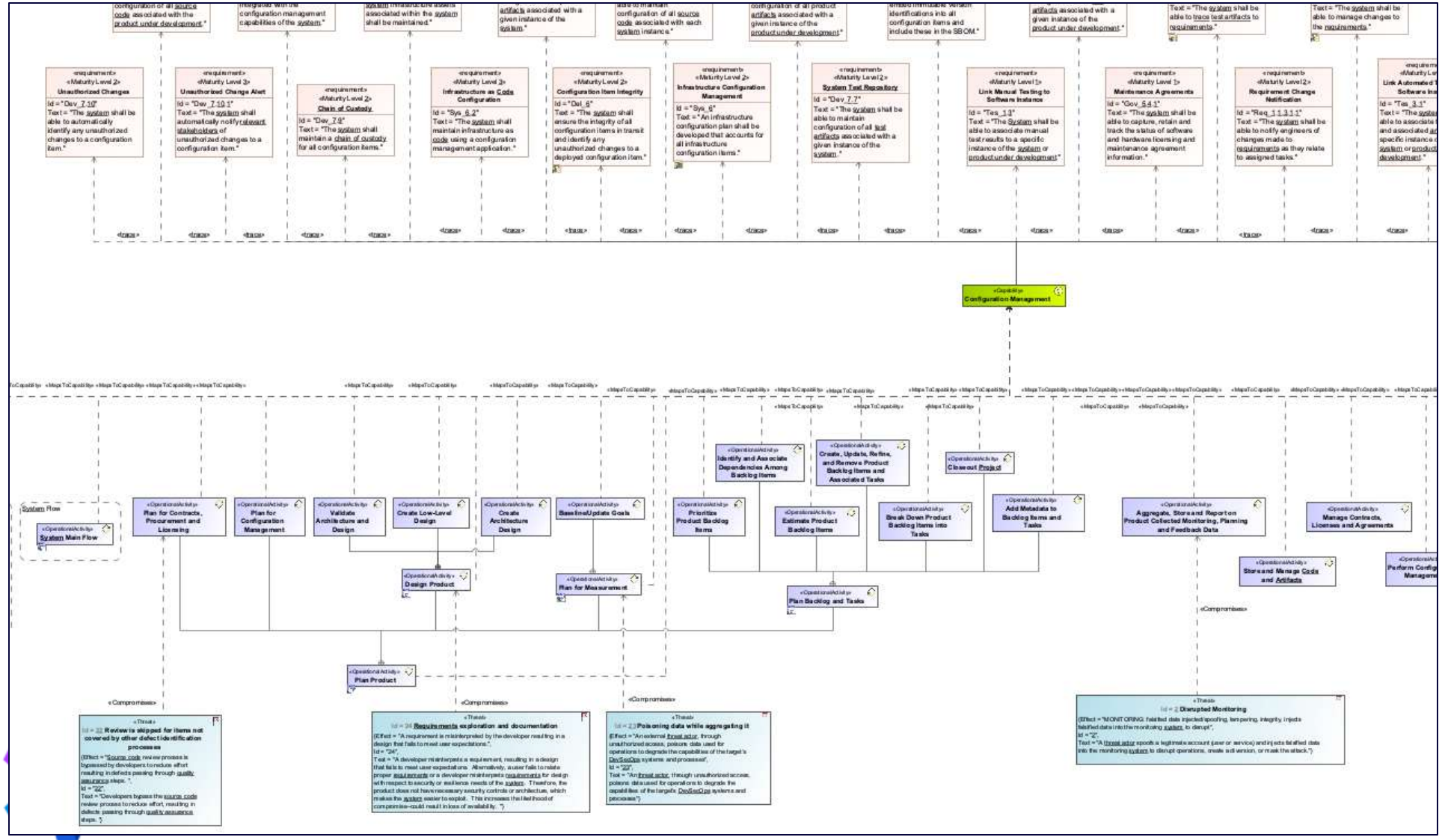
[Threats to Operational Activities Link]

DevSecOps Threats with Attributes

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.m org/data/definitions/1000.h
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 161? Most signific 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]

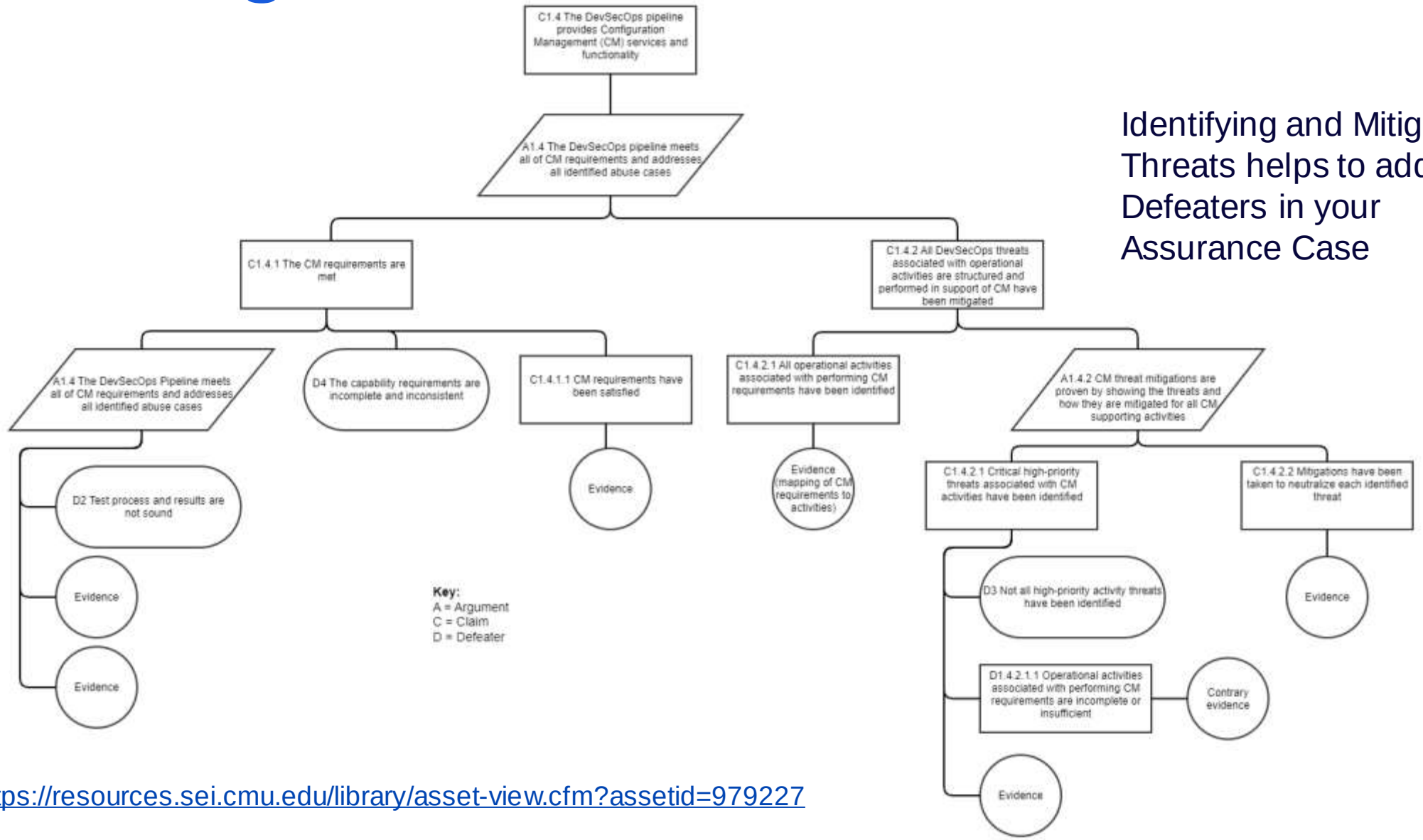
Capturing the Complexity of the DevSecOps System



Example of Threats Traced to Capabilities via Operational Activities

[Configuration Management Complexity Link]

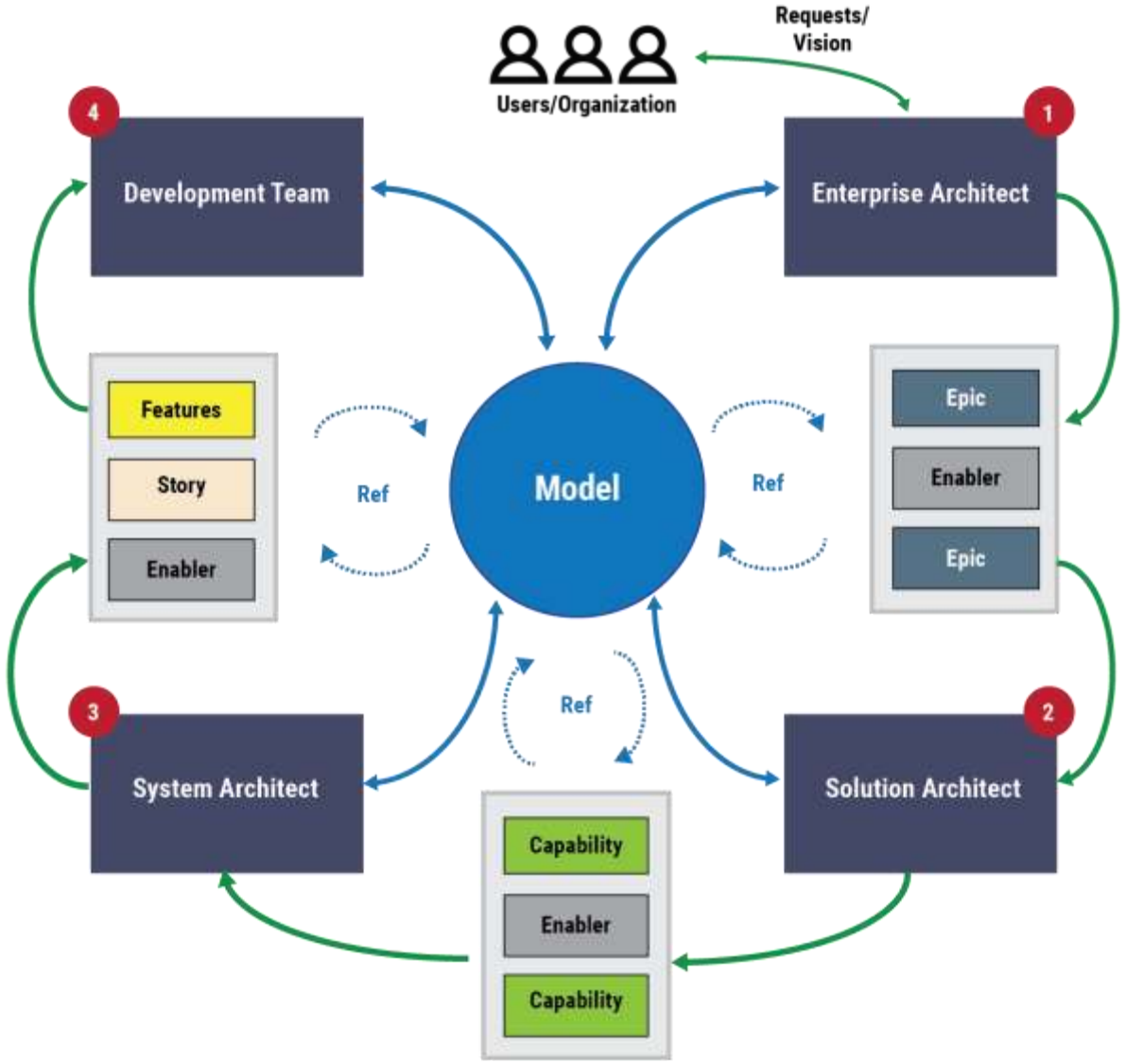
Addressing Assurance Case Defeaters



Identifying and Mitigating Threats helps to address Defeaters in your Assurance Case

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227

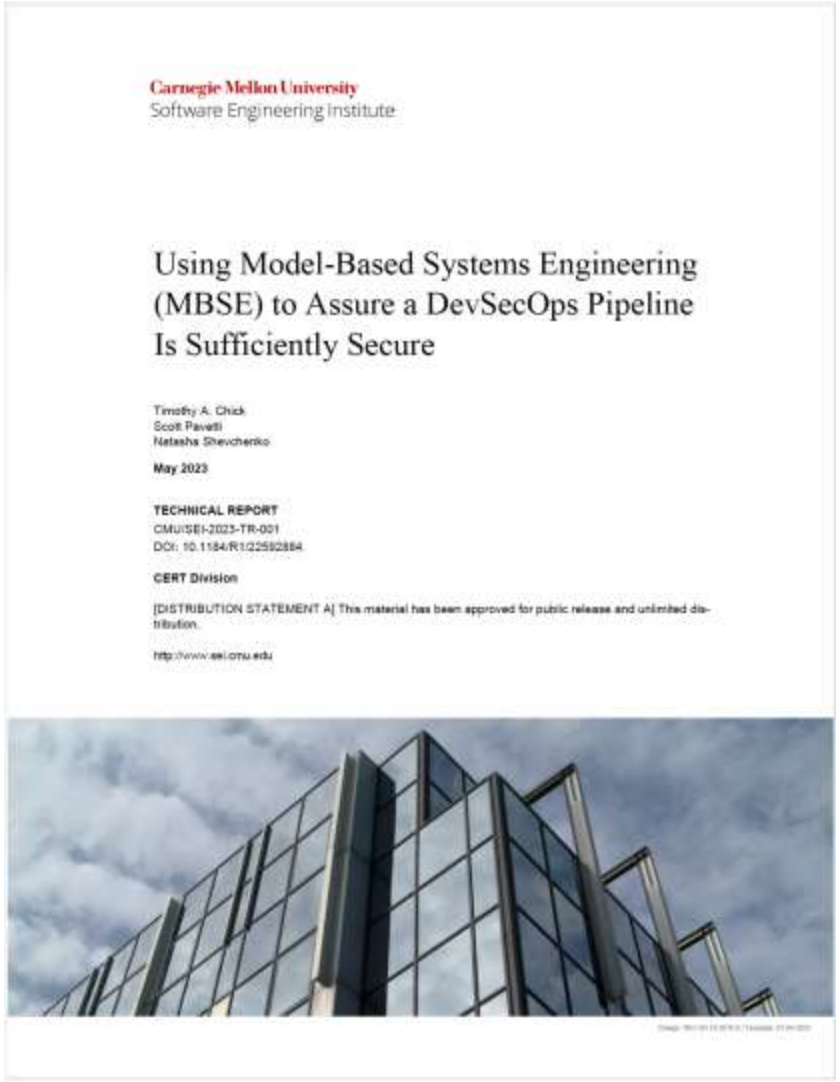
Marrying Architecture Practices with Agile Practices is Possible



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
- evaluate the capabilities of software factories
- 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

Summary



The use of model based systems engineering in the design, implementation, and sustainment of your DevSecOps socio-technical system will assist you in building 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.

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=979227

Contact Information

Timothy A. Chick

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

Adjunct Faculty Member, CMU-Software and Societal Systems Department, School of Computer Science

tchick@sei.cmu.edu

https://www.sei.cmu.edu

https://s3d.cmu.edu

https://www.cylab.cmu.edu