Carnegie Mellon University
Software Engineering Institute

Using Model-Based Systems Engineering (MBSE) to Assure a DevSecOps Pipeline is Sufficiently Secure

Timothy A. Chick
Scott Pavetti
Natasha Shevchenko

May 2023

TECHNICAL REPORT
CMU/SEI-2023-TR-001
DOI: 10.1184/R1/22592884

CERT Division

http://www.sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

DM23-0444

Table of Contents

List of Figures

List of Tables

Acknowledgments

We are grateful for the comments from those who reviewed various drafts of this report: Bob Ellison, Tim Morrow, Mary Popeck, and Carol Woody.

Abstract

Many enterprises and government programs are concerned that adversaries may abuse weaknesses in a DevSecOps pipeline to inject exploitable vulnerabilities into their products and services. This report presents an approach using model-based systems engineering (MBSE) and the DevSecOps Platform Independent Model (PIM) to evaluate and mitigate the cybersecurity risks associated with a given enterprise's, or government program's, DevSecOps pipeline(s). The analysis approaches this paper describes focuses on ensuring that the DevSecOps pipeline and its associated products are implemented in a secure, safe, and sustainable way; are sufficiently free from vulnerabilities; and the capabilities only function as intended. Ultimately, the PIM provides analysts with a minimum set of MBSE tools to assist with threat identification, analysis, documentation, and subsequent mitigations.

1 Introduction

Organizations struggle in applying DevSecOps practices and principles in a cybersecurity-constrained environment because they lack a consistent basis for managing software-intensive development, cybersecurity, and operations in today's embedded and distributed systems-deployment scenarios. They typically focus on creating pipeline functionality to produce output quickly and efficiently and applying some controls to show that they addressed compliance mandates. Organizations are challenged in addressing basic security questions, such as "Are the right controls in place to meet the appropriate cybersecurity needs?" and "Are these controls applied appropriately in the pipeline?"

An authoritative reference model that is augmented to consider system assurance, such as the DevSecOps PIM [CMU SEI 2022], enables organizations to fully design and execute an integrated DevSecOps strategy that addresses stakeholder needs with cybersecurity in all aspects of the DevSecOps pipeline. An assurance case can demonstrate the adequacy for both the pipeline and the embedded or distributed system. While builders of embedded and distributed systems desire to reap the flexibility and speed expected when applying DevSecOps, they need reference material and a repeatable defensible process to confirm a given DevSecOps pipeline is implemented in a secure, safe, and sustainable way.

Modeling allows extensive verification through MBSE tools before effort is wasted to "burn" chips, select and install specific tools, and execute physical tests. The DevSecOps PIM provides embedded and distributed DevSecOps system builders the ability to select from the information provided by experts to

- specify the DevSecOps requirements to the lead system integrators who need to develop a platform-specific solution that includes the designed system, simulation or 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 establish a cyber assurance case for structuring evidence to demonstrate that a system and DevSecOps pipeline are sufficiently free from vulnerabilities and function only as intended
- confirm the selected platform-specific solution has sufficient cyber assurance

In this paper, we will focus on the use of the DevSecOps PIM to frame a cyber assurance case, showing how the evidence we gathered can be combined into an argument demonstrating that the risks associated with a given DevSecOps pipeline instance have been adequately addressed. Using the PIM as guidance, an organization, or project, can develop a platform-specific assurance case to demonstrate whether key cyber aspects are addressed, how they are addressed, and how well the corresponding solution handles known DevSecOps cybersecurity risk. This in turn provides the organization with the basis for making risk-based decisions tied to the adequacy of the security controls and processes selected and deployed. This approach structures pathways and guidance for automated systems testing or collecting other evidence, such as scenarios where hardware in loop is used for quality assurance. Actual testing provides the evidence needed to support the assurance claims, but the DevSecOps PIM defines the assurance case structure.

2 Approach and Validity

As depicted in Figure 1, DevSecOps is an approach that integrates development of features (Dev), defensibility or security (Sec), and stable delivery/operations (Ops) of software systems to reduce the time required to move from need to capability and provide CI/CD with high-quality software [Morales 2020]. The DevSecOps pipeline is a socio-technical system made up of a collection of both software tools and processes [Bass 2015]. It is not a computer-based system to be built or acquired; it is a personal, team, and organizational mindset that relies on defined processes for the rapid development, fielding, and operations of software and software-based systems utilizing automation where feasible to achieve the desired throughput of developing, fielding, and sustaining new product features and capabilities.

Figure 1: Development, Security, and Operations Integration

Since enterprise architecture and MBSE are the best practices for designing and formalizing a description of a complex information system in social context, we created the DevSecOps PIM. The PIM can now be used to effectively design, develop, and sustain a secure and stable DevSecOps pipeline. We define a DevSecOps pipeline as "a socio-technical system composed of both software tools and processes. As the capability matures, it seamlessly integrates the three traditional factions that sometimes have opposing interests: development, which values features; security, which values defensibility; and operations, which values stability. A DevSecOps pipeline emerges when continuous integration of these three factions is used to meet organizational, project, and team objectives and commitments" [CMU SEI 2022].

To begin a cybersecurity risk analysis of a specific DevSecOps pipeline, it is necessary to define a reference architecture for DevSecOps. The purpose of a reference architecture, such as the DevSecOps PIM, is to capture the organization, mission, people, processes, and systems (hardware and software) necessary to fully realize a mature DevSecOps-oriented enterprise or program. This provides a framework for identifying and mitigating security risks that should be considered in a specific pipeline instantiation.

The value of using enterprise architecture and MBSE approaches is based on an assertion that DevSecOps pipelines are complex systems. By definition, a system is "an assemblage or combination of things or parts forming a complex or unitary whole" [Dictionary.com 2023]. Thus, DevSecOps is a system. It also possesses the characteristics of a socio-technical system [SEBoK 2022] and a computer-information system, since DevSecOps pipelines are composed of people, processes, and computer technology that are "designed to collect, process, store, and distribute information" [Wikipedia 2023a]. 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 socio-technical computer information systems.

The idea of applying MBSE methods to socio-technical systems is not new [Asan 2013, Haskins 2007, Miller 2012, Oosthuizen 2018, Palmer 2016]. In addition to the cited publications, companies have adopted the use of MBSE and virtual modeling tools in their everyday practices [Dassault Systèmes 2023]. It is a standard practice [Object Management Group 2010] to use model-based approaches such as business-process modeling (BPM) to design or describe patterns of human activities as a context of the functioning of a computer information system (i.e., business process). As we identified that a DevSecOps pipeline combines characteristics of both socio-technical and computer-information systems, using BPM and MBSE approaches are the logical next step.

As articulated in Figure 2, all DevSecOps-oriented enterprises, or government programs, are driven by three concerns:

1. business mission
2. capability to deliver value
3. products

The *business mission* captures stakeholder needs and channels the whole enterprise, or program, in meeting those needs. The business mission is owned by the organization's core executive and is supported by various business functions depending on the domain in which the enterprise, or program, operates. This part of the organization can answer the questions "Why and for whom does the enterprise, or program, exist?"

The *capability to deliver value* in a DevSecOps organization covers the people, processes, and technology necessary to build, deploy, and operate the enterprise's, or program's, products. In general, this consists of the software factory and product operational environments; however, it does not consist of the products themselves. In the DevSecOps PIM, this is commonly referred to as the "system" and is synonymous to a DevSecOps pipeline.

Products generically are the units of value the organization delivers. In a DevSecOps-oriented organization, these products are the components, applications, services, and outputs that the organization delivers and deploys for customers to use. These products utilize the capabilities the software factory and operational environments deliver. In the DevSecOps PIM, this is commonly referred to as the "product under development."

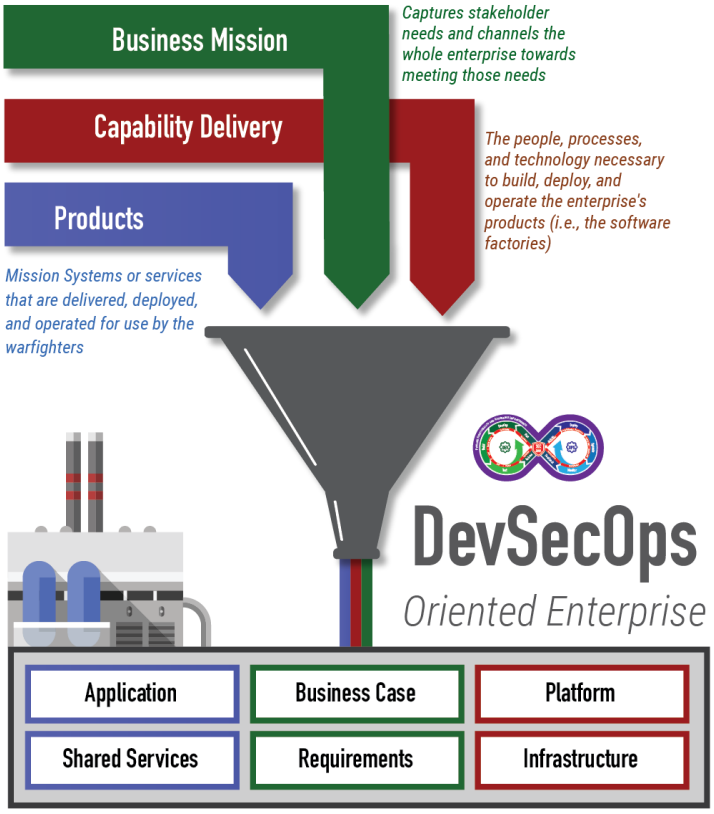

Figure 2: Business Mission, Capability Delivery, and Products

The enterprise, or government program, provides the business case and requirements to each of the other concerns that are responsible for providing the capability to deliver value and the value itself. Both capability delivery and product development execute a DevSecOps process using different process steps to achieve their planned outcomes. However, they need to synchronize with each other periodically to ensure that the software factory and operational environments remain capable of meeting the needs of the products under development. Security is improved when duties are separated, providing another reason for segregation.

As visualized in Figure 3, an attack surface is the accumulation of all possible attack vectors in which a threat actor can access a system and perform unauthorized actions. The smaller the attack surface, the easier it is to protect. 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. Traditional products are operated or deployed in environments segregated from the environment in which they were developed. Thus, traditional cybersecurity practices have focused on protecting the final delivered product. With the adoption of DevSecOps tools and techniques and the increased coupling between the product being built and the tools used to build them, the attack surface of the product continues to grow, incorporating segments of the development environment. Threat analysis helps to focus the builders' attention to areas of greatest concern for security risks and identify the attack opportunities that could require additional mitigations.

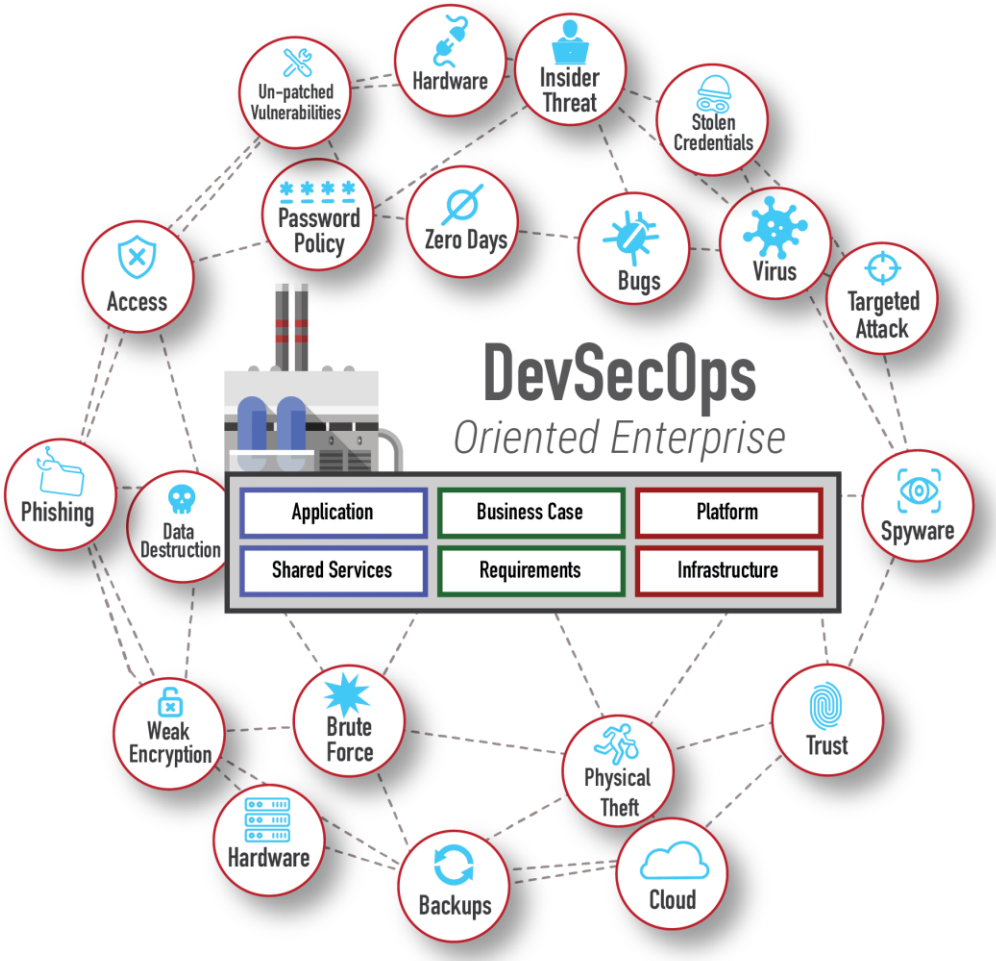

Figure 3: Business Mission, Capability Delivery, and Products Attack Surface

The DevSecOps PIM helps establish security requirements that builders can apply consistently to pipeline capabilities, which in turn can make the product more secure. This allows the DevSecOps pipeline to become a part of the enterprise architecture of the system being built, in contrast to current practices where the DevSecOps pipeline is not included in the overall system architecture and does not effectively integrate with the compliance and operational context of the products and services.

3 DevSecOps Platform Independent Model (PIM) Overview

Most literature discussion around DevSecOps depicts the concepts using some variation of the infinity diagram shown in Figure 4. This high-level conceptual diagram is generally used to visualize the cultural and engineering practices that break down barriers and open collaboration between the development, security, and operations organizations using automation to focus on rapid, frequent delivery of secure infrastructure and software to production.

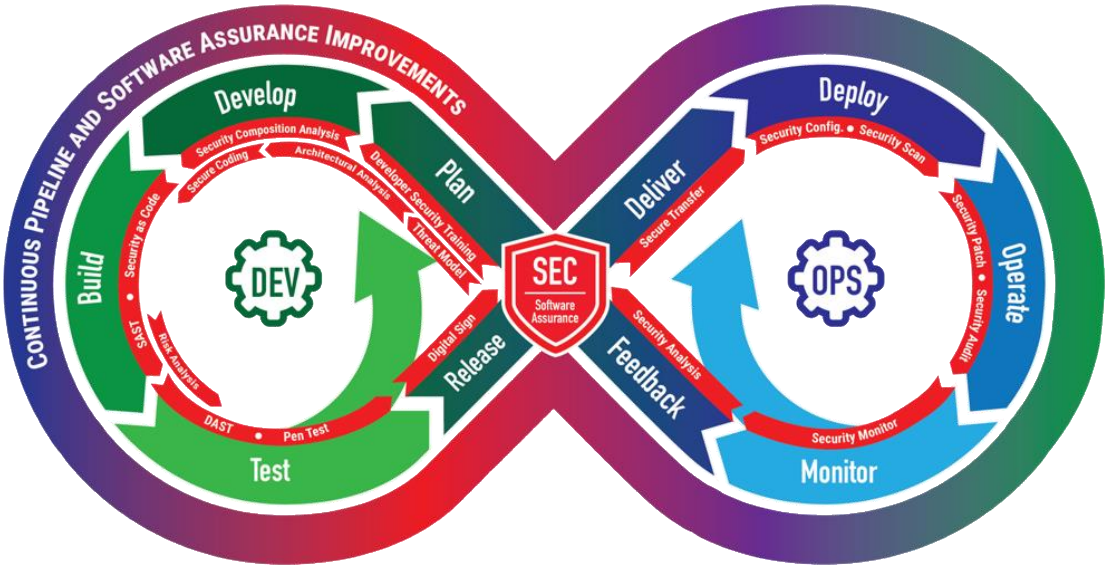

Figure 4: DevSecOps Infinity Diagram

The DevSecOps PIM takes the DevSecOps Infinity Diagram concepts and implied interaction and explicitly defines the people, tools, processes, and associated interactions needed to instantiate a DevSecOps pipeline (e.g., system) and a product under development. Figure 5 provides a high-level view of the model's content, which follows the Unified Architecture Framework (UAF). The DevSecOps PIM is broken down into six sections:

1. Dictionary – This defines key terms unique to the model and references to source material used in the creation of the PIM.
2. System requirements – These define the DevSecOps requirements in terms of shall statements. The requirements are broken down into seven categories: governance, requirements, architecture and design, development, test, deliver, and system infrastructure
3. Strategy – Given the system requirements, what are the capabilities a DevSecOps pipeline (or system) needs to provide? To answer this question, the model defines 10 capabilities needed to achieve the desired effect. Capabilities define the ways and means the system will use to implement the requirements.
4. Operational – This captures how the DevSecOps pipeline (e.g., system) and product under development works at operational and logical levels. It consists of both operational, structural, and connectivity viewpoints. The operational-process views capture the flow of major activities and the data and resources

needed to perform the given activity. The structure and connectivity views capture the logical organizations of the activities and performers.

5. Personnel – This captures the human views associated with the DevSecOps pipeline (e.g., system) and product under development instantiations. This viewpoint was extended with an Involvement custom profile that implements a version of a responsible, accountable, consulted, and informed (RACI) matrix (see Appendices A and B).
6. Security – This captures the cybersecurity aspects, including results of threat modeling activates, such as threats and threat scenarios, attack types, and relationships with corresponding threat actors. This viewpoint was extended with the Threat-Modeling custom profile (see Appendices A and B).

For more information, the DevSecOps PIM and associated introduction material can be found at https://cmu-sei.github.io/DevSecOps-Model/ and https://www.sei.cmu.edu/go/DevSecOpsPIM.

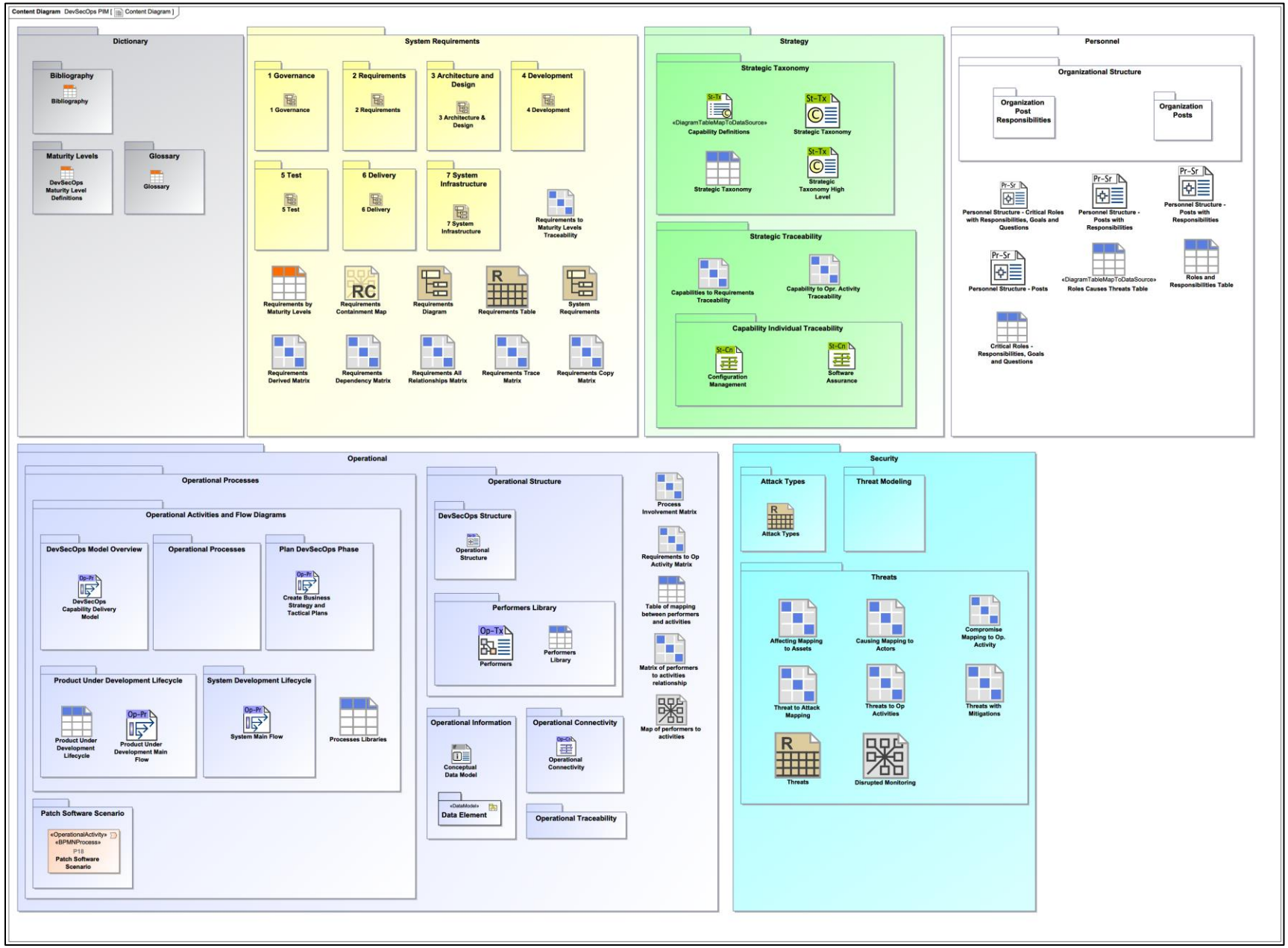

Figure 5: DevSecOps PIM Content Diagram

4 Managing Risk

Many business and government sectors have compliance, legal requirements, and processes and frameworks for managing risk. Examples include

- Federal Information Security Management (FISMA)
- General Data Protection Regulation (GDPR)
- Health Insurance Portability and Accountability (HIPAA)
- International Electrotechnical Commission (IEC)
- International Organization for Standardization (ISO)
- National Institute of Standards and Technology (NIST)
- Payment Card Industry Data Security Standard (PCI DSS)

While using such standards is a good starting point, most implementations are focused on the product's compliance, which doesn't explain why the requirements are in the standard to begin with. What risks are the standards trying to mitigate? Does a given implementation mitigate the risks better than another? What is the cost and return on investment between the various implementations? The questions continue. Additionally, the standards focus only on the final product and miss the fact that the pipeline used to build and maintain the product must also meet security criteria.

Threat modeling is an analysis technique frequently used by cybersecurity experts to provide the context needed to identify and reduce the cyber risks and assure the software and overall system function only as intended [Shevchenko 2018]. However, most threat modelers focus on the product and miss the pipeline in their analysis. Threat modelers need to focus on the entire attack surface. DevSecOps can be seen as one form of modern software engineering practices and tools that encompasses the full software development lifecycle. Given the tight coupling of development and operations, the product has become a continuation of the DevSecOps pipeline where security aspects carry over from the pipeline to the product. Threat modeling augments secure development practices and tools, along with security automation techniques and security operations for the full system lifecycle. When done well, the overall risks associated with the DevSecOps pipeline and associated products will be reduced and the compliance and legal requirements will naturally be addressed within the engineering lifecycle.

Understanding risk is hard. In cybersecurity alone, the Open Risk Manual [Open Risk Manual n.d.] has identified over 70 categories of risk, such as access control, data breach, denial of service, malware, situational awareness, vulnerability assessments, and so forth. Without being able to quantify or reason around the cybersecurity risks associated with a given product and DevSecOps pipeline, one will not be able to properly balance features, defensibility, and stability and make necessary tradeoffs so that those properties are optimally maintained to achieve a given organization's or project's mission and vision in a cost-effective way. As shown in Figure 6, one must consider the properties in a way that balances risk, quality, and benefits within their time, scope, and cost constraints. The DevSecOps PIM is designed to set the stage by defining what must be considered such that a platform-specific DevSecOps pipeline can balance the properties within reasonable constraints to meet a given mission and vision.

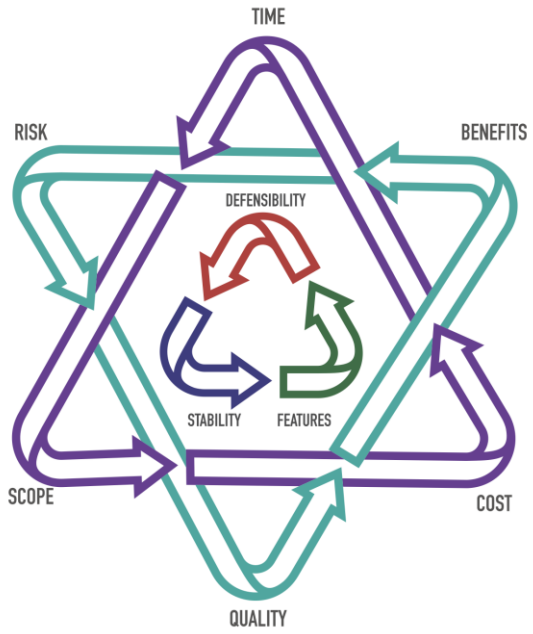

Figure 6: DevSecOps Equities

5 Assurance Cases and Defeaters

Builders and evaluators can use an assurance case to reason about the degree of security for both the pipeline and the product. It is a structured approach used to argue that available evidence supports a given claim, thus explaining why a claim about the system property holds true. The DevSecOps PIM can incorporate the elements needed to frame a software assurance case by showing how gathered evidence can be combined into an argument demonstrating that the risks associated with a given pipeline instance have been adequately addressed. This in turn provides the organization with the basis for making risk-based choices that assure the pipeline functions only as intended. "Confidence in the truth of a hypothesis or claim increases as reasons for doubting its truth are identified and eliminated. Possible reasons for doubting the truth of a claim arise from analyzing an assurance case using defeasible reasoning concepts" [Goodenough 2012]. This approach provides logical pathways and guidance for automated systems testing or other evidence-collection techniques used for quality assurance. Actual test results provide the evidence needed to support the assurance claims.

Assurance cases are composed of the following elements:

- Claims – Claims are "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" [Bloomfield 2014].
- Arguments – Arguments "link the evidence to the claim" [Bloomfield 2014] by stating the assumption(s) on which the claim and the evidence are built.
- Evidence – "Evidence 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" [Bloomfield 2014].
- Defeaters – Defeaters are "possible reasons for doubting the truth of a claim" [Goodenough 2012]. Table 1 below defines the three types of defeaters for which evidence would be used to counter or confirm.

Table 1: Three Kinds of Defeaters [Goodenough 2012]

Kind of Defeater	Definition
Rebutting	Defeaters that eliminate belief in a claim by providing information that contradicts the claim
Undercutting	Defeaters that specify conditions under which the claim is not necessarily true even if the premise is true
Undermining	Defeaters that invalidate one or more of the premises (in which case, even if the inference rule is valid and all rebutting defeaters have been eliminated, we still have a reduced basis for believing in the truth of the associated claim)

An assurance case is considered complete when no credible new information would change the degree of belief in the claim. There are three criteria one can use in evaluating an assurance case.

1. Positive – The soundness of the argument can be logically validated or checked using credible evidence and reasoning. This forces one to contemplate defeaters at the evidence level.
2. Negative – One must actively search for and resolve defeaters.
3. Residual doubts – One must assess the risk of consequences and the likelihood of potentially valid defeaters that cannot be fully resolved.

The concept of using assurance cases with software is not new. Since software is a core component of many safety-critical systems, one can reference several guidance references or standard documents in building software-safety assurance cases [IEC 62304, ISO 26262, MIL-STD-882D, SAE ARP4754/ARP4761, RTCA DO-178C]. There are even a few focused on system and information security, such as the RTCA's Airworthiness Security Certification Course (RTCA DO-326, DO-355, and DO-356). For safety, the software must mitigate all possible hazards that may compromise the safety attributes of the given system. These hazards are usually caused by a failure, which is a deviation from the intended behavior caused by errors in the functioning of one or more system components. Failures can result from a combination of many sources including human error, poor processes, defective software, and inadequate hardware maintenance. The system must be designed to handle the hazardous conditions appropriately in addition to delivering the intended functionality.

The same concepts used in safety assurance can be applied to cybersecurity assurance. Cybersecurity assurance can be defined as the "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 2010]. Traditionally cybersecurity assurance is addressed through process-based standards, such as the NIST Risk Management Framework (SP 800-53). As systems become more complicated and interconnected, process-based standards fail to assure system owners that the system functions only as intended under all operational circumstances. For example, they fail to answer how the system will

- behave outside of normal operating conditions, particularly for ad hoc and adverse conditions
- account for constant change in the people, technology, and software
- adapt to changing threat landscapes

Applying safety assurance case practices to cybersecurity takes a property-based approach focusing on intent, correctness, and risk. This allows the organization to establish and maintain the appropriate security boundaries and constraints needed to assure the system functions only as intended throughout the life of the product.

6 Structuring a DevSecOps Pipeline Assurance Claim Using MBSE

The DevSecOps PIM provides the foundational architecture model of the socio-technical components and interactions of a DevSecOps pipeline builders and evaluators need to structure critical claims and arguments regarding the pipeline's ability to function only as intended and mitigate cybersecurity risks. While the PIM provides a foundational structure to build an assurance case, the complete analysis and evidence needed to complete the assurance case is beyond the scope of the PIM and must be addressed within the instantiated system itself. The evidence used to support the claims and arguments should be specific to the people, process, and technology of the system or products. In addition to provided claims, sub-claims, and arguments, builders and evaluators can use analysis to identify defeaters for functionality and ways in which threat mitigations might be defeated, thereby reinforcing the confidence that the assurance case is sound.

As noted earlier, the DevSecOps pipeline is a socio-technical system made up of both a collection of software tools and processes [Bass 2015]. The DevSecOps PIM maps the complex relationships between the capabilities needed to fulfill the requirement, the operational processes and activities that demonstrate how the system exhibits the capabilities, and the people and roles performing the processes. Builders and evaluators must understand the complex socio-technical relations of the DevSecOps pipeline before they can begin to adequately derive arguments and evidence that rationalize an assurance case to the point in which no credible new information would change the degree of belief in the claim that the pipeline functions only as intended. While the PIM provides the framework for an assurance case, it is insufficient in deriving and providing the evidence needed to complete the assurance case. Builders and evaluators need a well-understood instantiated DevSecOps pipeline to complete the assurance case of a given pipeline, as it will include the exact configuration and specific details of the pipeline one needs to test to yield the necessary evidence.

The claim selected to initiate the assurance case is of critical importance in setting the context for establishing confidence. When building a safety-assurance case, the top claim is typically something related to the system, such as "The system is safe." This high-level claim would then be broken down into sub-claims with arguments and evidence in support of the top claim that the system is safe. When applying assurance case concepts to security, one could start with a top claim such as "The system is secure," but this cannot be specifically demonstrated to show that no insecure issues exist.

Using the definition of cybersecurity assurance we previously referenced, an appropriate top claim could be "The software systems and services function only in the intended manner." A top claim more specific to DevSecOps would be "The DevSecOps pipeline only functions as intended." This claim can be assured if one can prove that all the pipeline's key business services and functionality perform as intended. The key business services and functionality (or, in other words, capabilities) should have been identified during the requirements-analysis process (see Appendix A). Figure 7 depicts the capabilities identified during this process and Table 2 defines the capabilities developed as an output to the process.

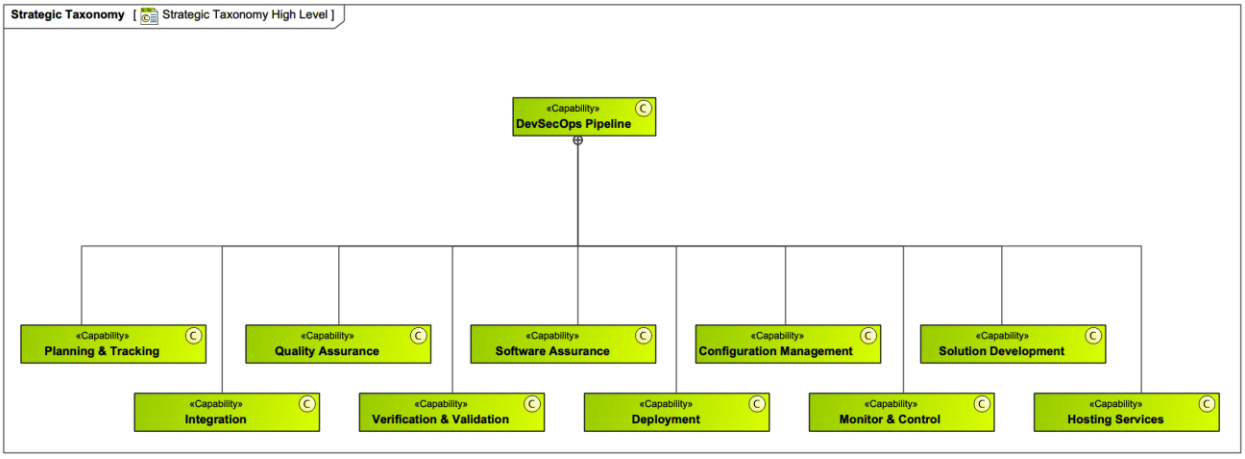

Figure 7: DevSecOps Capabilities as Stated in PIM

Table 2: DevSecOps Capability Definitions as Stated in PIM

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

Capability	Definition
Monitor & Control	Monitor and control involves continuously monitoring activities, communicating status, and taking corrective action to proactively address issues and consistently improve performance. More mature projects automate as much of this as possible. Appropriate visibility enables timely corrective action to be taken when performance deviates significantly from what was expected. A deviation is significant if it precludes the project from meeting its objectives when left unresolved. Items that should be monitored include cost, schedule, effort, commitments, risks, data, stakeholder involvement, corrective action progress, and task and work-product attributes like size, complexity, weight, form, fit, or function.
Planning & Tracking	Planning and tracking is the set of practices one uses to define tasks and activities. It also includes the resources one needs to perform those tasks and activities, achieve an objective or commitment, and track progress (or lack thereof) towards achieving the given objective. It provides the mechanisms required to inform relevant stakeholders where an effort currently is within the process and whether it is on track to provide the expected outcomes. These mechanisms allow relevant stakeholders to determine what has been accomplished and what adjustments or corrective actions need to occur to account for impediments and other unforeseen issues. Ideally, impediments and issues are proactively identified and addressed. Practices include documenting activities and breaking them down into actionable work to which one can assign resources, capturing dependence, forecasting, mapping work to requirements, collecting data, tracking progress to commitments, and reporting status. The planning and tracking capability of a system matures as the automation and integration of associated practices increases.
Quality Assurance	Quality assurance is a set of independent activities (i.e., free from technical, managerial, and financial influences, intentional or unintentional) designed to provide confidence to relevant stakeholders that the DevSecOps processes and tools are appropriate for and produce products and services of suitable quality for their intended purposes. It assumes that the organization's, team's, and project's policies and procedures have been defined based on all relevant stakeholder needs, which will result in a value stream that consistently produces products and services that meet all relevant stakeholder expectations. The quality assurance capability of a system matures as its ability to assess adherence to and the adequacy of the defined policies and procedures improves.
Software Assurance	Software assurance is the level of confidence that software functions only as intended and is free from vulnerabilities either intentionally or unintentionally designed or inserted as part of the software throughout the full software lifecycle. It consists of two independent but interrelated assertions: 1. The software functions only as intended. It exhibits only functionality intended by its design and does not exhibit functionality not intended. 2. The software is free from vulnerabilities, whether intentionally or unintentionally present in the software, including software incorporated into the final system. It is the responsibility of the DevSecOps system to ensure that software that meets the organization's threshold for software assurance is allowed to be deployed and operated.
Solution Development	Solutions development determines the best way of satisfying the requirements to achieve an outcome. Its goals are to evaluate baseline requirements and alternative solutions to achieve them, select the optimum solution, and create a specification for the solution. Each development value stream develops one or more solutions, which are products, services, or systems delivered to the customer, whether internal or external to the enterprise.
Verification & Validation	Verification and validation is the set of activities that provides evidence that the system or application under development has met the requirements and criteria that are expected. The scope includes the general realm of testing, verifying, and validating activities and matures as automation, feedback, and integration with other elements increase.

If we are saying that the DevSecOps pipeline cybersecurity assurance claim is "The DevSecOps pipeline only functions as intended," and the above capabilities (Figure 7) represent the functionality of the pipeline, we can make a next step and break the top-level cybersecurity assurance claim into sub-claims based on the 10 DevSecOps capabilities as a starting point for the analysis (Figure 8). See the mapping between the capabilities and the sub-claims in Table 3.

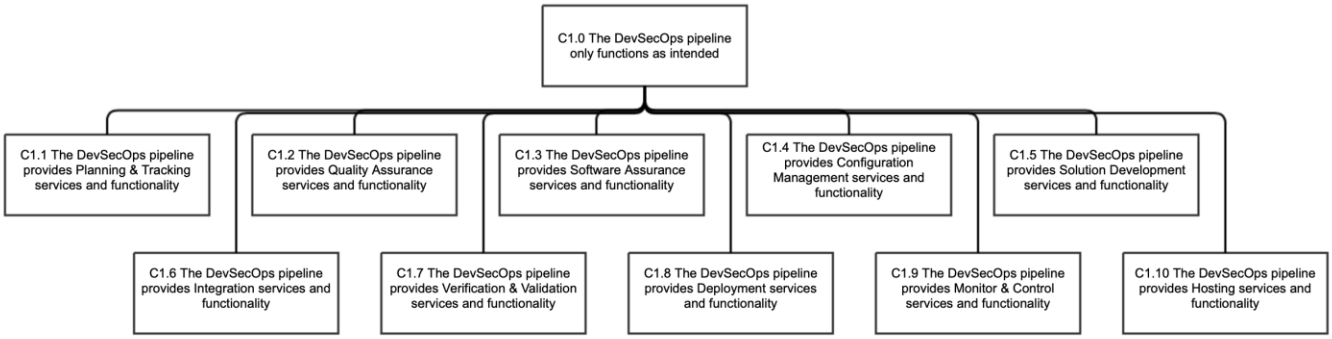

Figure 8: DevSecOps Pipeline Top-Level Assurance Claims

Table 3: Mapping DevSecOps Capabilities to DevSecOps Pipeline Top-Level Assurance Claims

DevSecOps Pipeline	**C1.0 The DevSecOps pipeline functions only as intended.**
Planning & Tracking	C1.1 The DevSecOps pipeline provides Planning & Tracking services and functionality.
Quality Assurance	C1.2 The DevSecOps pipeline provides Quality Assurance services and functionality.
Software Assurance	C1.3 The DevSecOps pipeline provides Software Assurance services and functionality.
Configuration Management	C1.4 The DevSecOps pipeline provides Configuration Management services and functionality.
Solution Development	C1.5 The DevSecOps pipeline provides Solution Development services and functionality.
Integration	C1.6 The DevSecOps pipeline provides Integration services and functionality.
Verification & Validation	C1.7 The DevSecOps pipeline provides Verification & Validation services and functionality.
Deployment	C1.8 The DevSecOps pipeline provides Deployment services and functionality.
Monitor & Control	C1.9 The DevSecOps pipeline provides Monitor & Control services and functionality.
Hosting Services	C1.10 The DevSecOps pipeline provides Hosting services and functionality.

The sub-claims in Figure 8 can be further subdivided until one reaches a level where there is sufficient evidence to support the claims. Evidence is considered sufficient when no credible new information would change the degree of belief in the associated claim. Only an instantiated DevSecOps pipeline can provide evidence to the lower level of sub-claims that requires details on the pipeline's implementation. The PIM can support higher-level claims, assist with building the case, and provide a structure for collecting the evidence. To control scope, the rest of this paper will use the configuration management capability as an exemplar of how to do an assurance case using the PIM—specifically the corresponding assurance case for C1.4. The DevSecOps pipeline provides configuration management services and functionality (see Figure 8 and Table 3). This claim is further broken down in Figures 9 through 12. Please note that, in order to complete the DevSecOps pipeline assurance case C1.0, all 10 elements in Figure 8 would need to be broken down into sub-claims with sufficient arguments and evidence

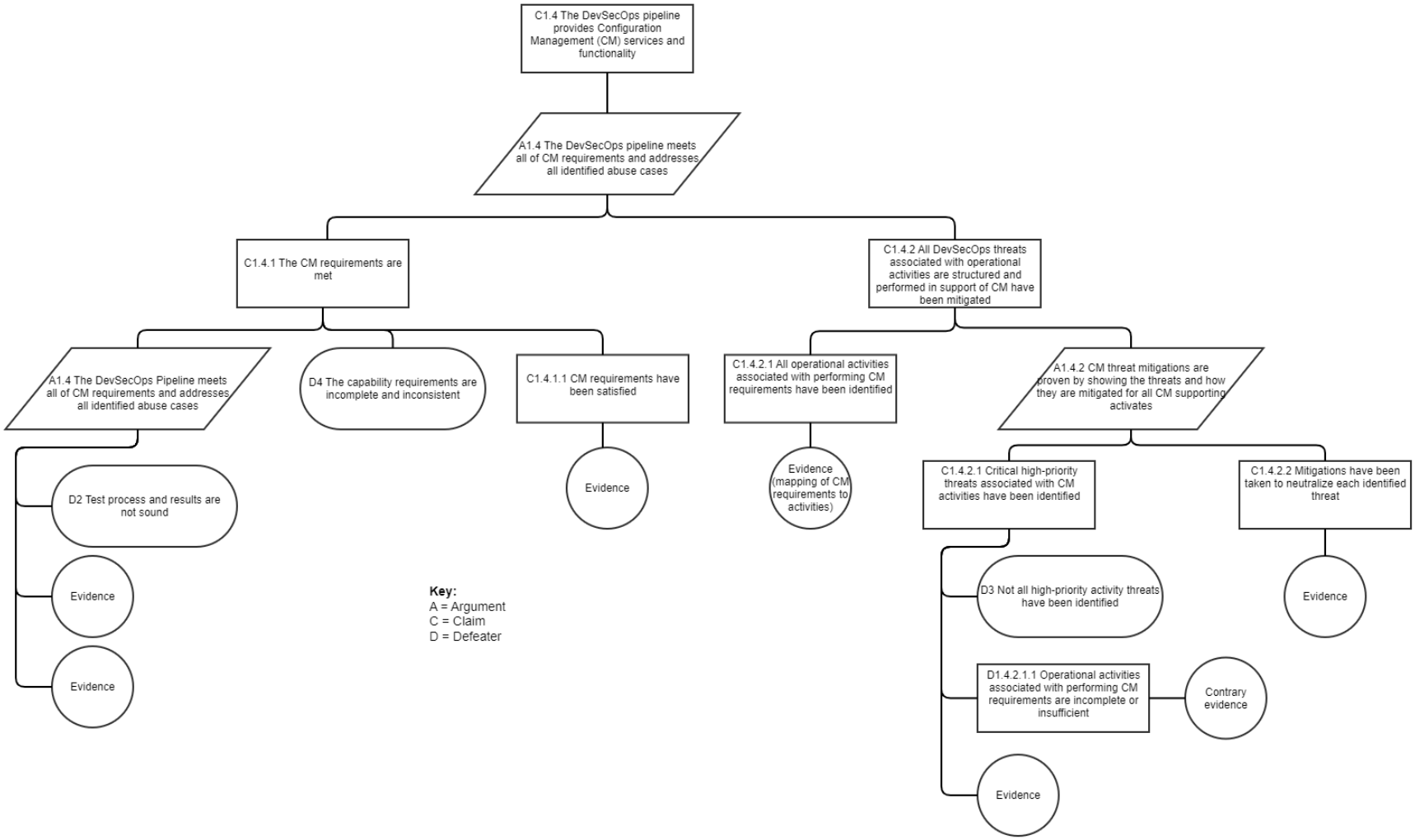

Figure 9: DevSecOps Configuration Management Assurance Case

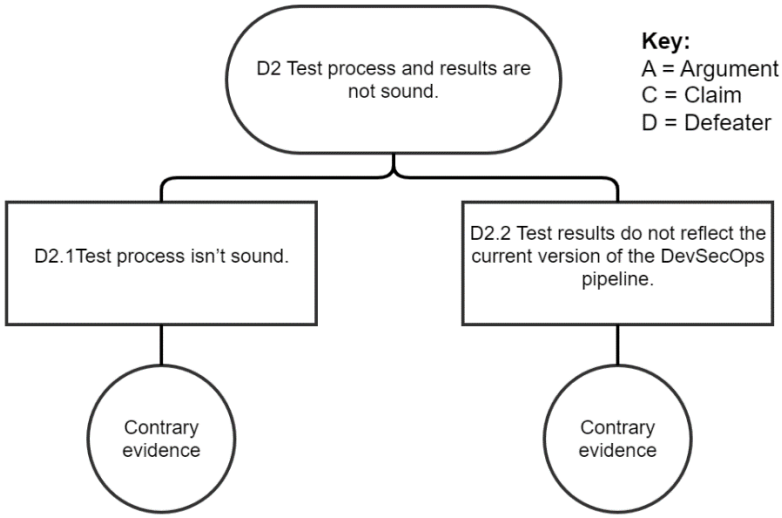

Figure 10: Test Process and Results Defeater

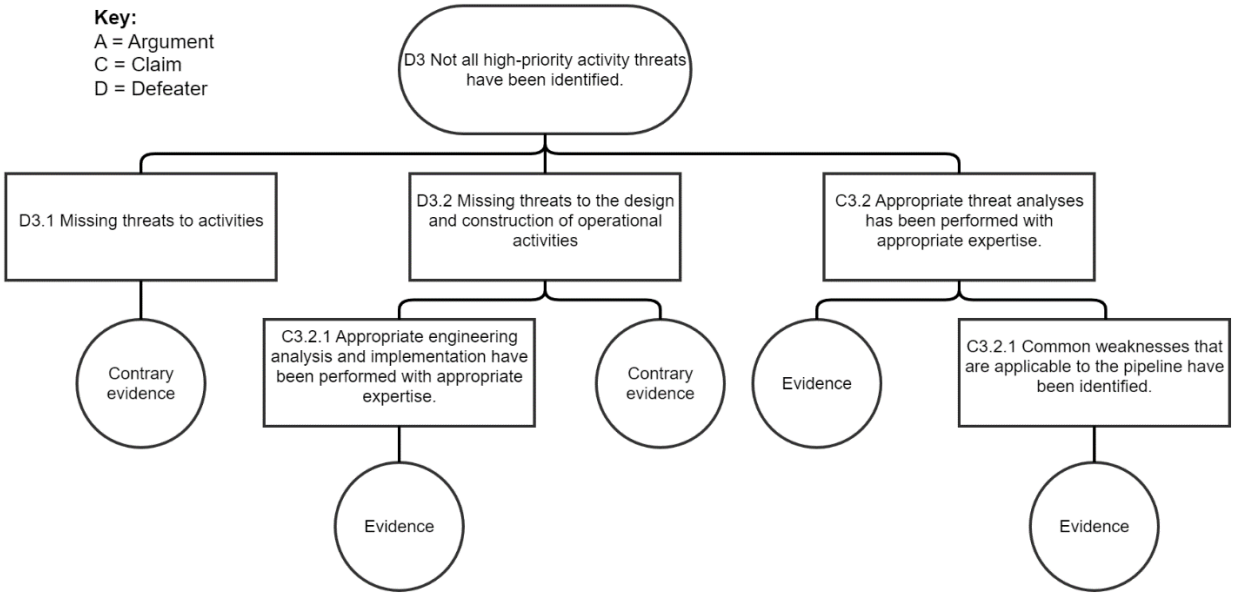

Figure 11: High-Priority Activity Threats Defeater

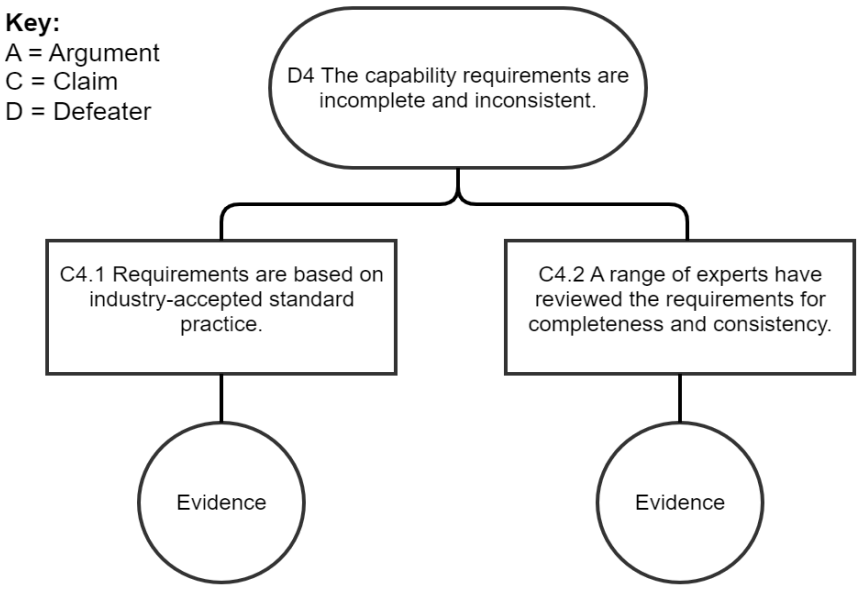

Figure 12: Incomplete and Inconsistent Capability Requirements Defeater

The first argument on Figure 9 stated, "A1.4 The DevSecOps pipeline meets all of configuration management (CM) requirements and addresses all identified abuse cases." This argument requires an architect to make sure that all requirements and abuse cases related to CM have been accounted for. Views within the PIM can help address this argument; however, the system requirements and abuse cases are represented by different elements and reside in different viewpoints. Thus, the analysis must be split in half.

We will start our analysis with the requirements. As described in Appendix A, the PIM contains a dedicated viewpoint for system requirements, which makes it easier to locate them and perform requirements engineering and analysis, including traceability of the requirements to the rest of architecture. Since the CM aspect of the DevSecOps system is represented in the PIM with the CM capability, CM-related requirements should be linked with CM capability. Looking at Figure 13, which depicts a collapsed view of the traceability matrix, one can see that CM capability mapped to 28 requirements from the system-requirements package. Figure 14 shows the same matrix expanded, exposing the linkage between CM capability and specific requirements.

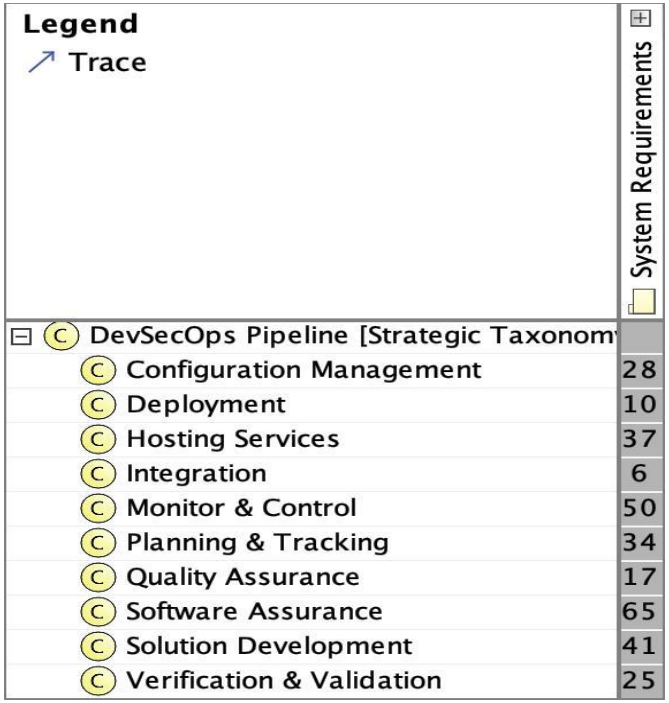

Figure 13: Capability to Requirement Mapping

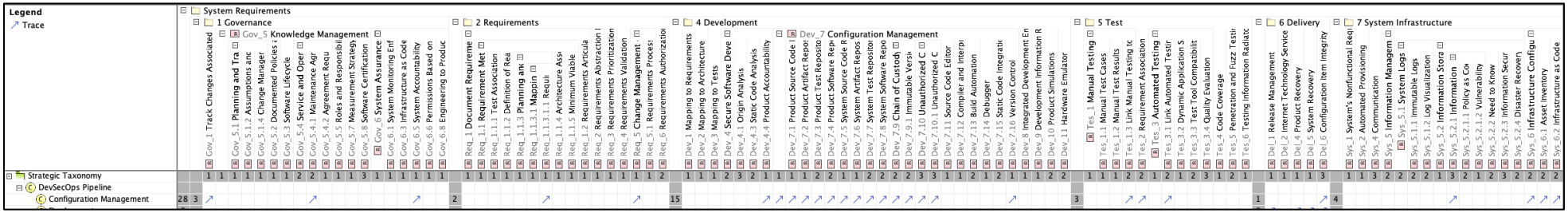

Figure 14: Capability to Requirements Traceability Matrix

The abuse cases in the PIM are represented by threat elements within the "security" viewpoint (see Appendices A and B). Threats do not directly link to a capability in the PIM. To see the association, one needs to look at the Capability-Operational Activity Threat chain.

In Figure 15, the CM capability traces to several requirements and to operational activities connected to threats (below the operational activities).

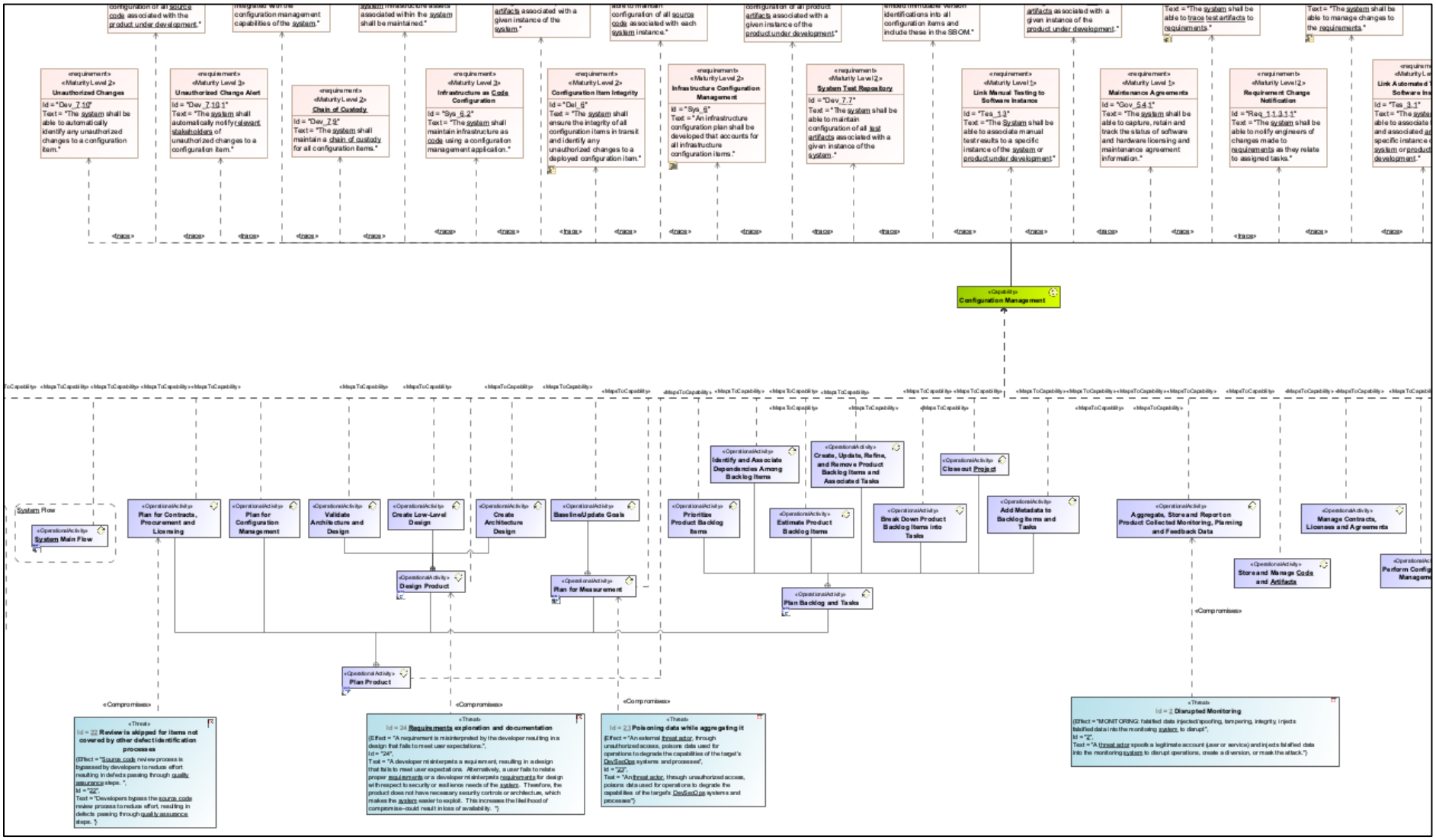

Figure 15: Threats Traced to Capabilities via Operational Activities

The PIM can present this information in several different views. For example, Figure 16 is a behavioral map that shows the linkage from CM capability to the threats via connection to the operational activities. On Figure 19 (which illustrates matrix threats to operational activities) and Figure 20 (which illustrates matrix operational activities to CM capability), an analyst can see the traceability from the other direction—from threats to operational activities to capability.

Figure 16: Configuration-Management Capability Behavioral Map

Analysis of these views will produce evidence to support argument A1.4 that all CM requirements and abuse cases have been accounted for.

To support sub-claim C1.4.1, which states, "The CM requirements are met" (Figure 9), the traceability matrix (Figure 13) shows 28 specific requirements traced to CM capability. The PIM cannot produce final evidence that the requirements are met. This is the job of verification & validation activities within the given DevSecOps pipeline instance. Although an analyst can use this matrix to analyze capability-to-requirements mapping, the analyst must make sure corresponding requirements are complete and consistent.

Also, the PIM can provide an analyst with additional tools to help with the analysis. For example, an expert can examine each requirement in detail using a requirements diagram (Figure 17). These types of diagrams depict all elements and connections directedly related to a requirement, such as requirements decomposition, derivation, and copy and trace as well as traceability to capabilities and documentation. This view can help one analyze the requirements for completeness and consistency to provide evidence to a sub-claim.

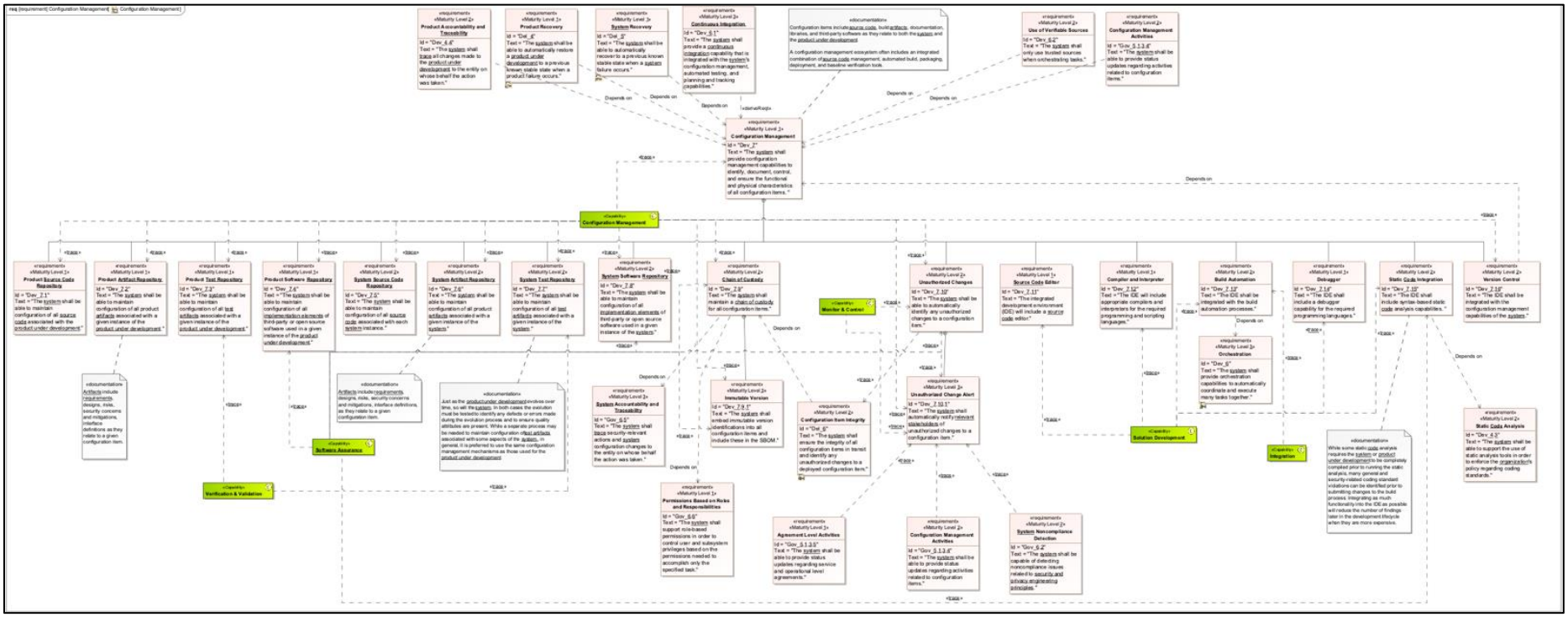

Figure 17: Requirement Diagram

Even though the PIM cannot produce final evidence that the requirements are met, it can provide the intermediate evidence that the presented architecture will satisfy CM requirements. The PIM is process-centric and uses operational activities to satisfy DevSecOps requirements. To find evidence for sub-claim C1.4.1.1, which states, "CM requirements have been satisfied" (Figure 9), an analyst can use requirements to the operational-activities mapping shown on Figure 18. This requirements-satisfy matrix depicts mapping specifically for CM.

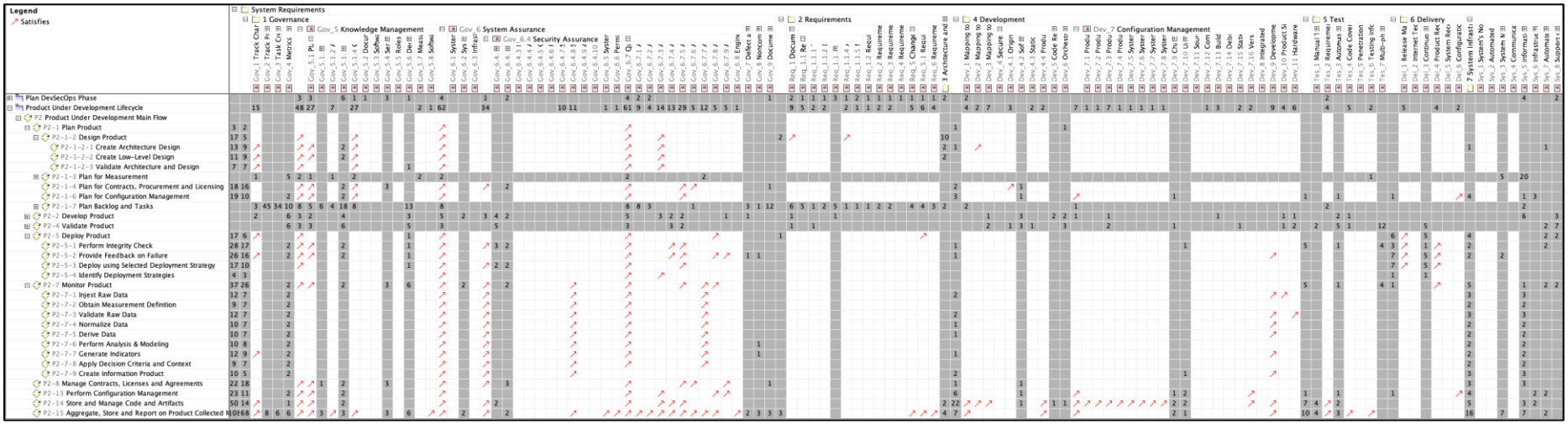

Figure 18: Requirements-Satisfy Matrix

Argument 1.4.1, "Test results showing all 28 CM requirements are verified," and defeater D2, "Test process and results are not sound," can only be fully addressed by a well-understood instantiated DevSecOps pipeline. This can be accomplished by demonstrating how the requirements in the PIM have been satisfied with supporting evidence.

In support of analyzing claim C1.4.2, "All DevSecOps threats associated with how operational activities are structured and performed in support of CM have been mitigated" (Figure 9), the threat-to-operational-activity matrix (Figure 19) can be used to identify applicable threats that should be mitigated.

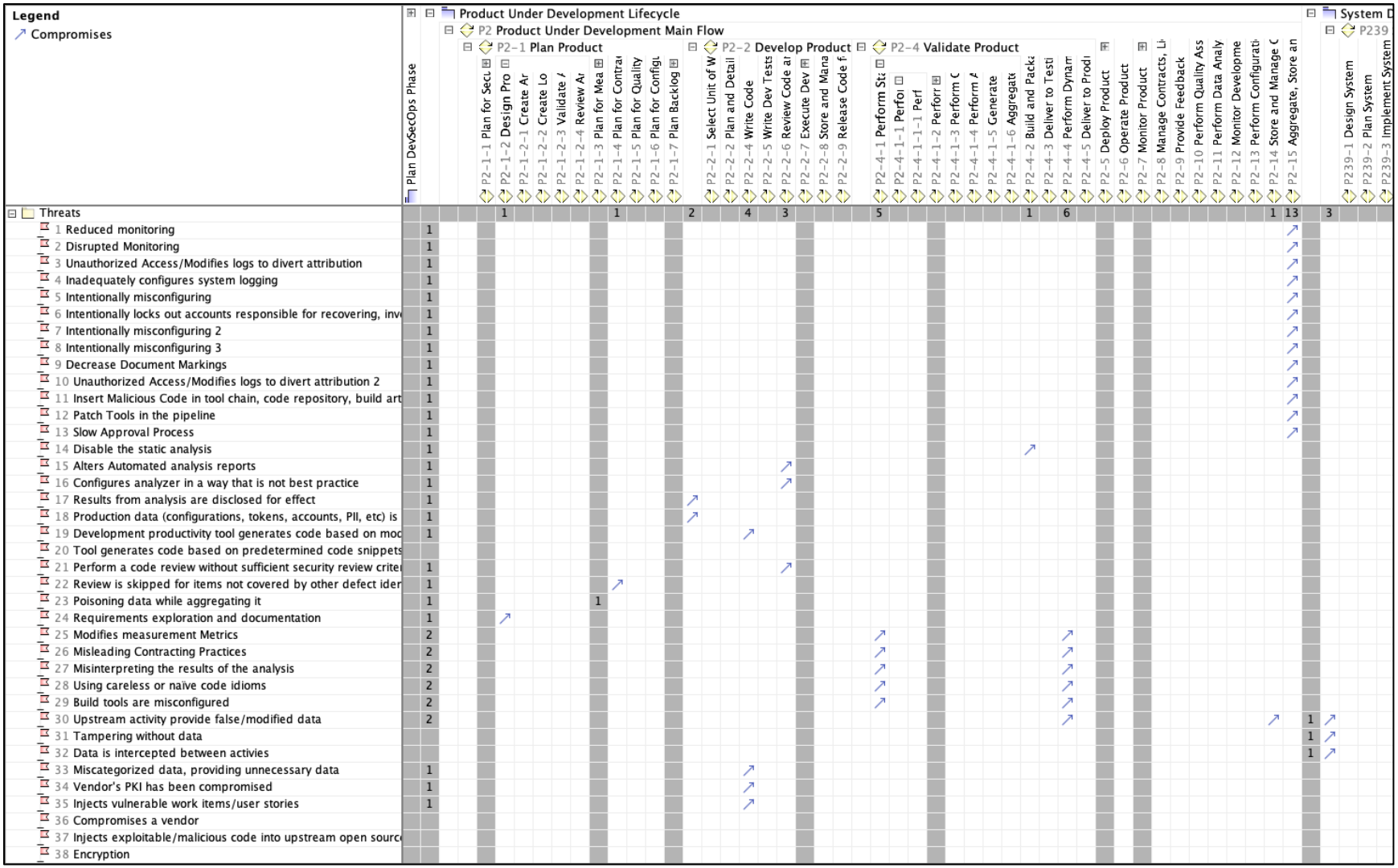

Figure 19: Threat-to-Operational-Activity Matrix

In support of sub-claim C1.4.2.1, "All operational activities associated with performing CM requirements have been identified" (Figure 9), the threats traced to CM capability via operational activities in Figure 15 can be used as evidence that all operational activities associated with performing CM have

been identified. Alternatively, the capability-to-operational-activity matrix (Figure 20) illustrates the linkage between CM capability and corresponding operational activities and can provide the missing information. Analysts can also use it as a tool to effectively identify and correct inconsistences in the model.

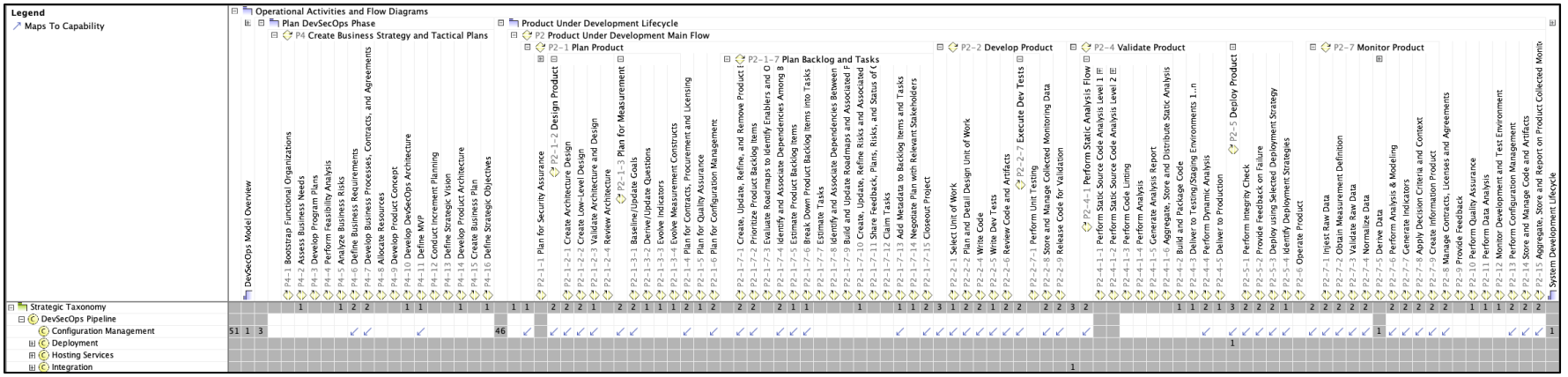

Figure 20: Capability-to-Operational-Activity Matrix

In support of argument A1.4.2, "CM threat mitigations are proven by showing the threats and how they are mitigated for all CM-supporting activates," (Figure 9), analysts can use several views from the PIM depending on which point of view they believe best supports the argument. Figure 21 provides a view of threats in the model with their attributes and relationships, including mitigations (see the "Mitigated By" column).

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 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		

Figure 21: Threats with Attributes

Only a well-understood instantiated DevSecOps pipeline can fully address sub-claim C1.4.2.2, "Mitigations have been taken to neutralize each identified threat" (Figure 9). An analyst can accomplish this by demonstrating how threats have been identified and mitigated with supporting evidence.

The requirements-satisfy matrix (Figure 18) can assist an analyst with defeater D1.4.2.1.1, "Operational activities associated with performing CM requirements are incomplete or insufficient" (Figure 9), as well as with defeater D3.1, "Missing threats to activities" (Figure 11). Additionally, an analyst can find a combined analysis of the threat-modeling diagram (like Figure 22) and operational-activity flow diagram (like Figure 23) helpful for finding evidence for or against defeater D3.1.

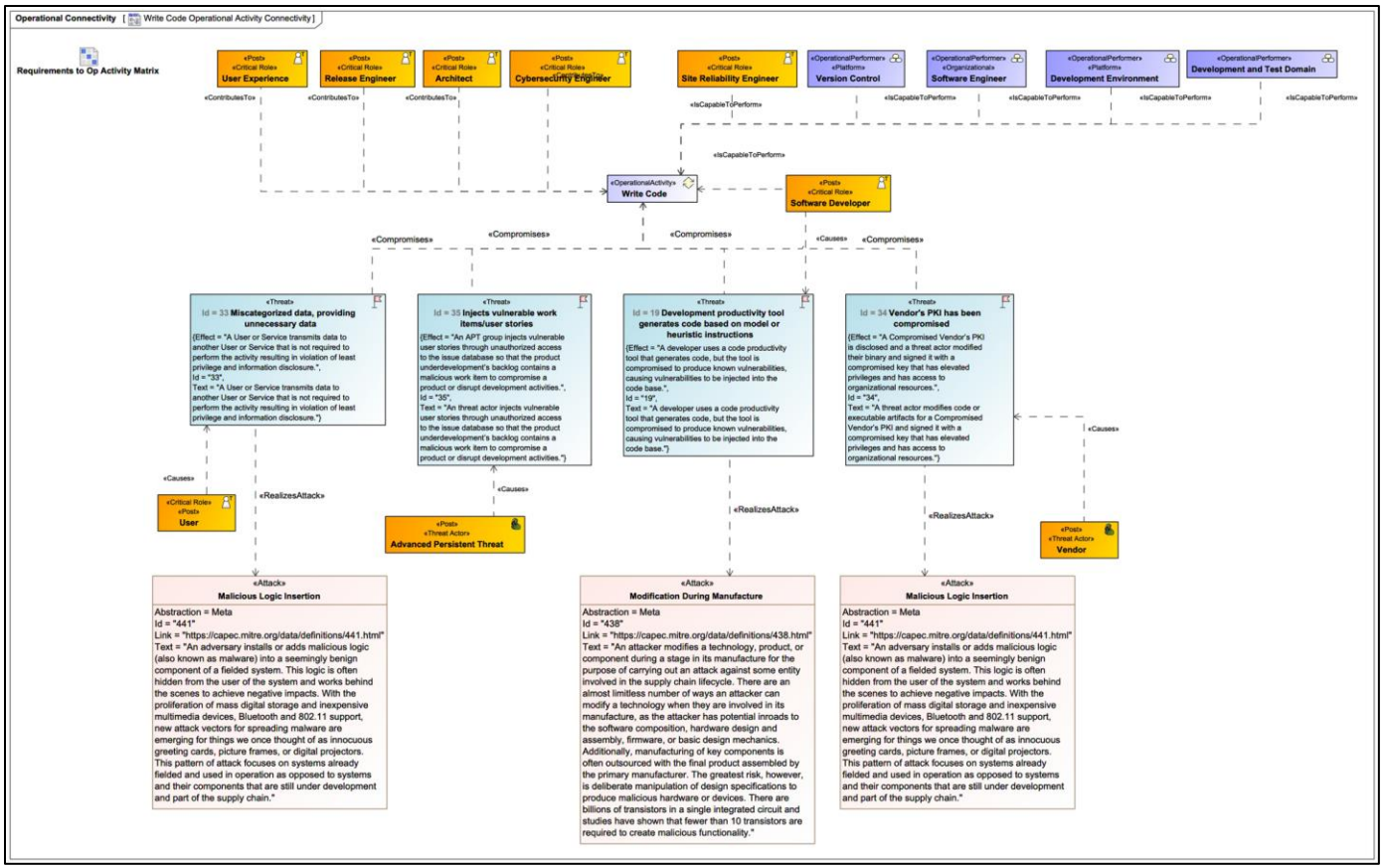

Figure 22: Threat-Modeling Diagram for Write Code Operational Activity (Example)

In support of sub-claim C3.2.1, "Appropriate engineering analysis and implementation have been performed with appropriate expertise," and against defeater D3.2, "Appropriate threat analyses have been performed with appropriate expertise" (Figure 11), an analyst will need to analyze the threat-modeling diagram for an operational activity and its flow, as demonstrated in Figure 23. They will also need to analyze the sub-activity with their corresponding threat-modeling diagram, such as the one shown in Figure 22. This is like the analysis performed for defeater D3.1.

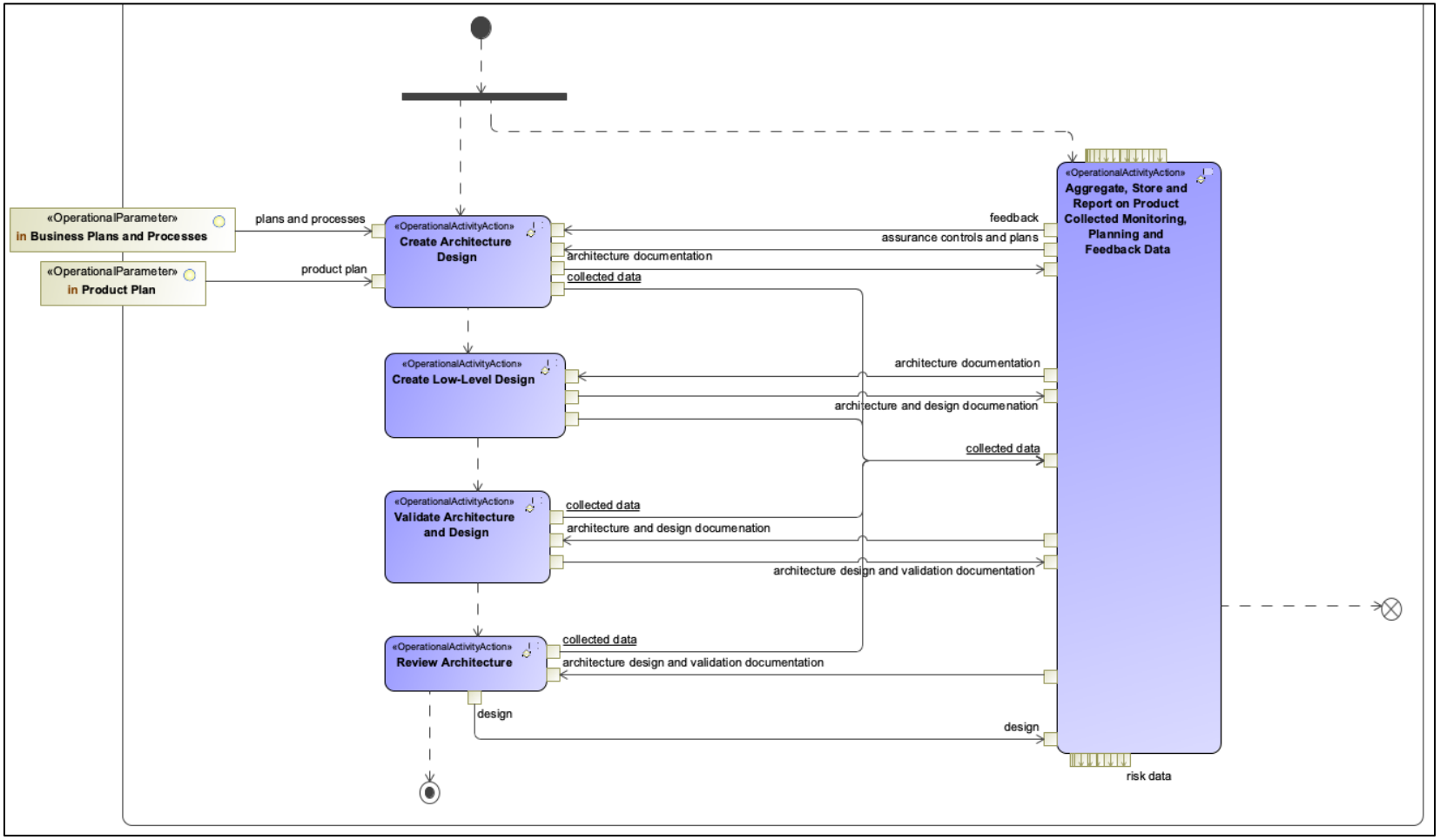

Figure 23: Flow Diagram for Design Product Operational Activity (Example)

The views of threat relationships in the PIM that the threat-to-attack matrix (Figure 24) and the threat-to-role matrix (Figure 25) provide can help with sub-claims C3.2, "Appropriate threat analysis have been performed with appropriate expertise," and C3.2.1, "Common weaknesses that are applicable to the pipeline have been identified" (Figure 11). The threat-to-attack matrix (Figure 24) can be used to identify which threat scenario or abuse case used which attack. Additionally, this matrix can be used to identify complex scenarios, such as ones that utilize more than one technique.

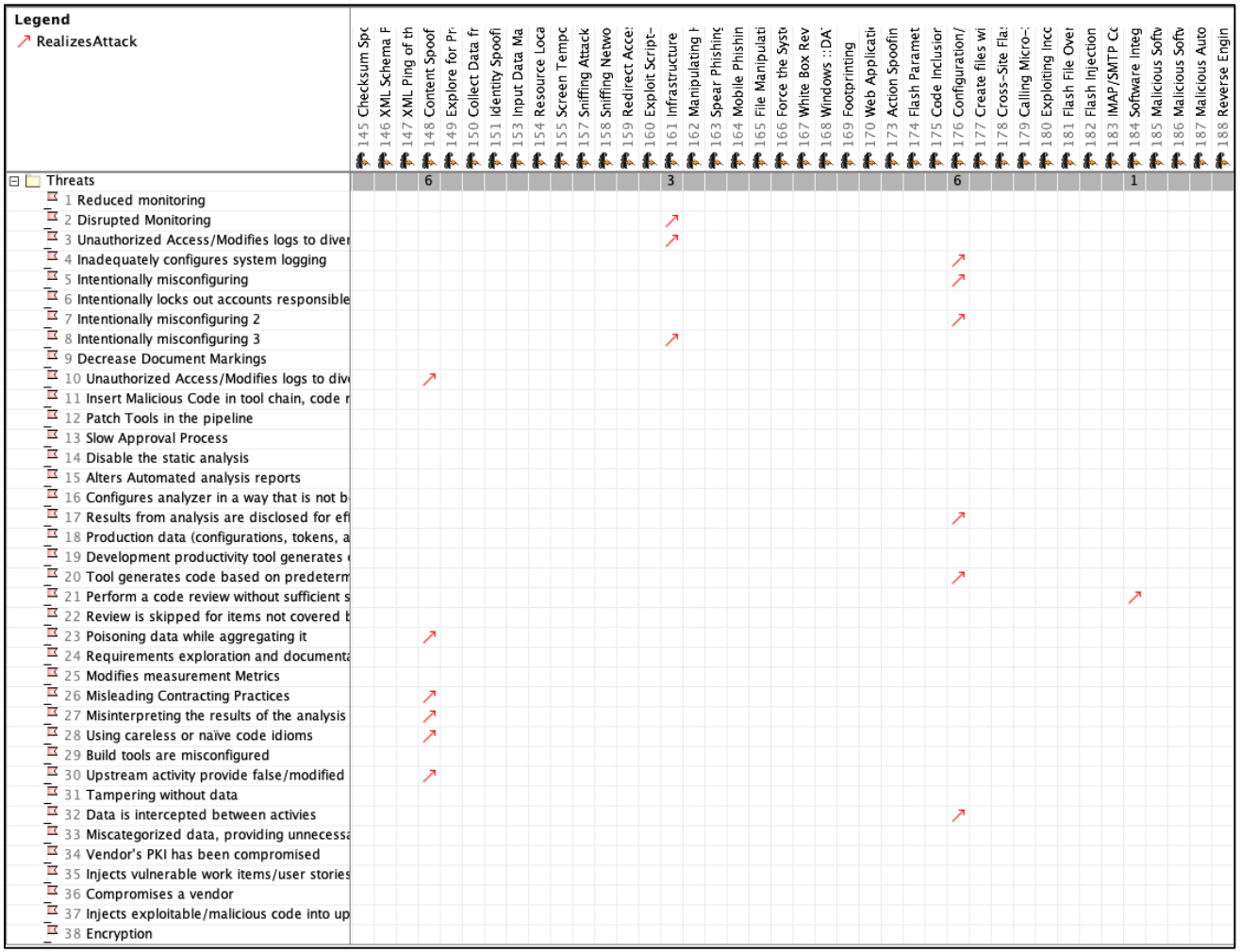

Figure 24: Threat-to-Attack Matrix

Analyzing internal and external threat actors with the threat-to-role matrix (Figure 25) and types of attack in the threat-to-attack matrix (Figure 24) can help an analyst identify the types of weaknesses related to a specific scenario, operational process, or technology.

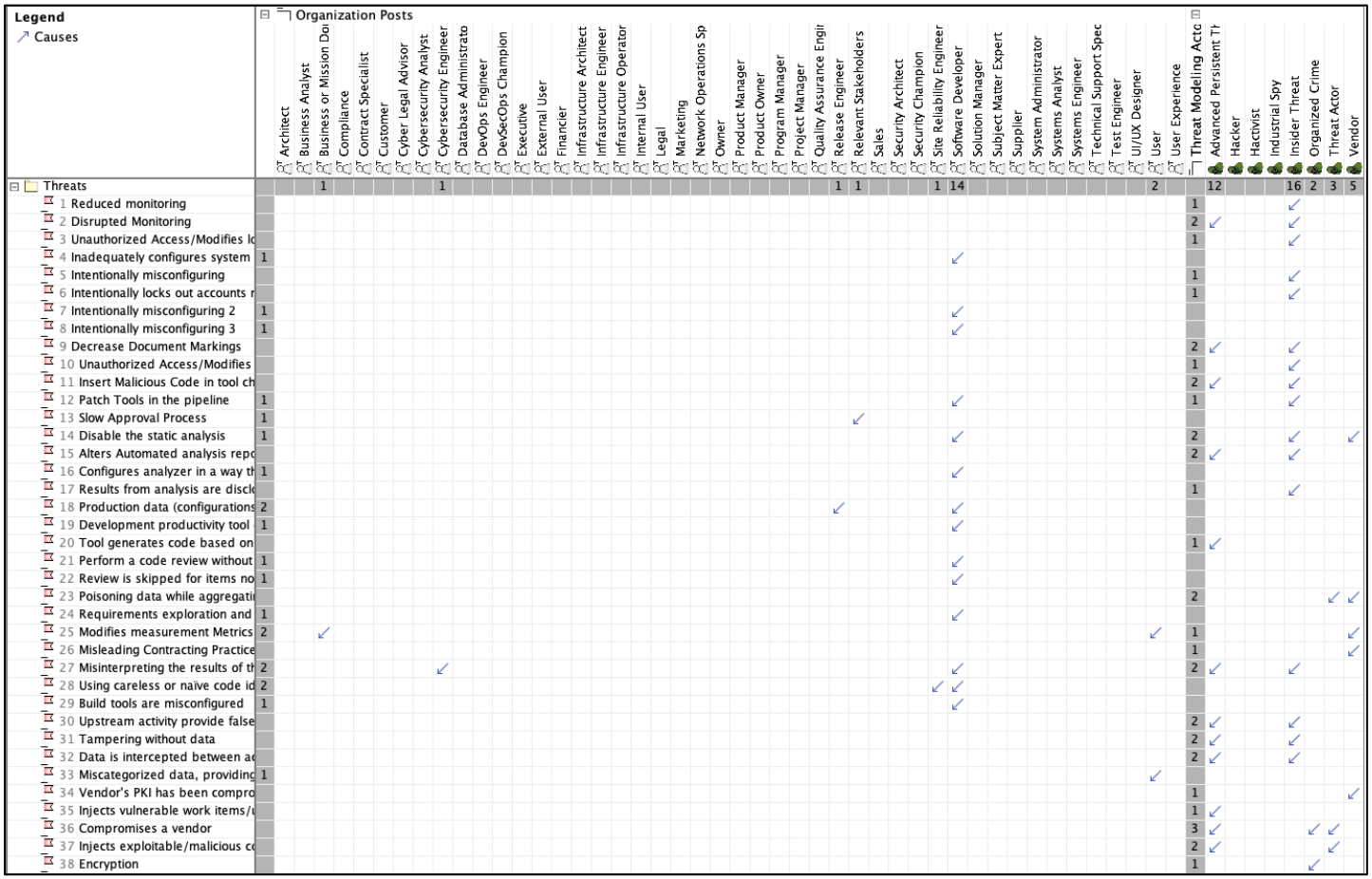

Figure 25: Threat-to-Role Matrix

Sub-claim C1.4.2.1 (Figure 9) and defeater D3 (Figure 11) are related to high-priority threats. The first round of prioritization can be done after an analyst initially analyzes the threats. Other rounds should occur with the help of a well-understood instantiated DevSecOps pipeline where one can create similar diagrams and views as those shown in Figures 13–25 with implementation-specific information.

7 Summary

The DevSecOps PIM provides reference material in support of a repeatable defensible process so that organizations can be confident that a given DevSecOps pipeline and its associated products are implemented in a secure, safe, and sustainable way. The PIM provides a basis for threat and attack-surface analysis forming the basis of a cyber-assurance case for structuring evidence to demonstrate that a product and DevSecOps pipeline are sufficiently free from vulnerabilities and function only as intended. The PIM also confirms that the selected platform-specific solution has sufficient cyber assurance. The PIM identifies specific threats to the DevSecOps pipeline, but it is not an all-exhaustive list. The PIM helps analysts identify potential threats to the system that should be addressed by the processes or technology in the instantiated DevSecOps pipeline. It provides a minimum set of MBSE tools to assist with threat identification, analysis, and documentation of a given DevSecOps pipeline (or system).

Appendix A MBSE Model with Cybersecurity Extension

This section introduces the techniques that we used to build the DevSecOps PIM. The DevSecOps PIM was built as an MBSE digital model. We constructed several views from the UAF to model DevSecOps as a socio-technical and information system.

- Requirements – We defined the DevSecOps requirements in terms of "shall" statements and broke them down into seven categories: governance, requirements, architecture and design, development, test, deliver, and system infrastructure.
- Capability/Strategic – Given the system requirements, what are the capabilities a DevSecOps pipeline or system needs to provide? To answer this question, the model defines 10 capabilities needed to achieve the desired effect. Capabilities define the ways and means the system will use to implement the requirements.
- Operational – This captures how the DevSecOps pipeline (e.g., system) and product under development work at an operational and logical level. It consists of operational, structural, and connectivity viewpoints. The operational process views capture the flow of major activities and the data and resources needed to perform the given activity. The structure and connectivity views capture the logical organizations of the activities and performers.
- Personnel – This captures the human views associated with the DevSecOps pipeline (e.g., system) and product under development instantiations. This viewpoint was extended with an custom involvement profile that implements a version of the RACI matrix [Prince 2022].
- Security – This captures the cybersecurity aspects, including the results of threat-modeling activates such as threats or threat scenarios, attack types, and relationships with corresponding threat actors. We extended this viewpoint with the threat-modeling custom profile.

Requirements Viewpoint

We organized all requirements into categories based on logical and functional groupings:

- governance
- requirements
- architecture and design
- development
- test
- delivery
- system infrastructure

We engineered and analyzed requirements using MBSE methodology and different relationships (derive, depends, copy, trace).

In diagrams, such as Figure 26, we present the requirements as a light pink box or as a light pink rectangle icon with the letter "R" in it.

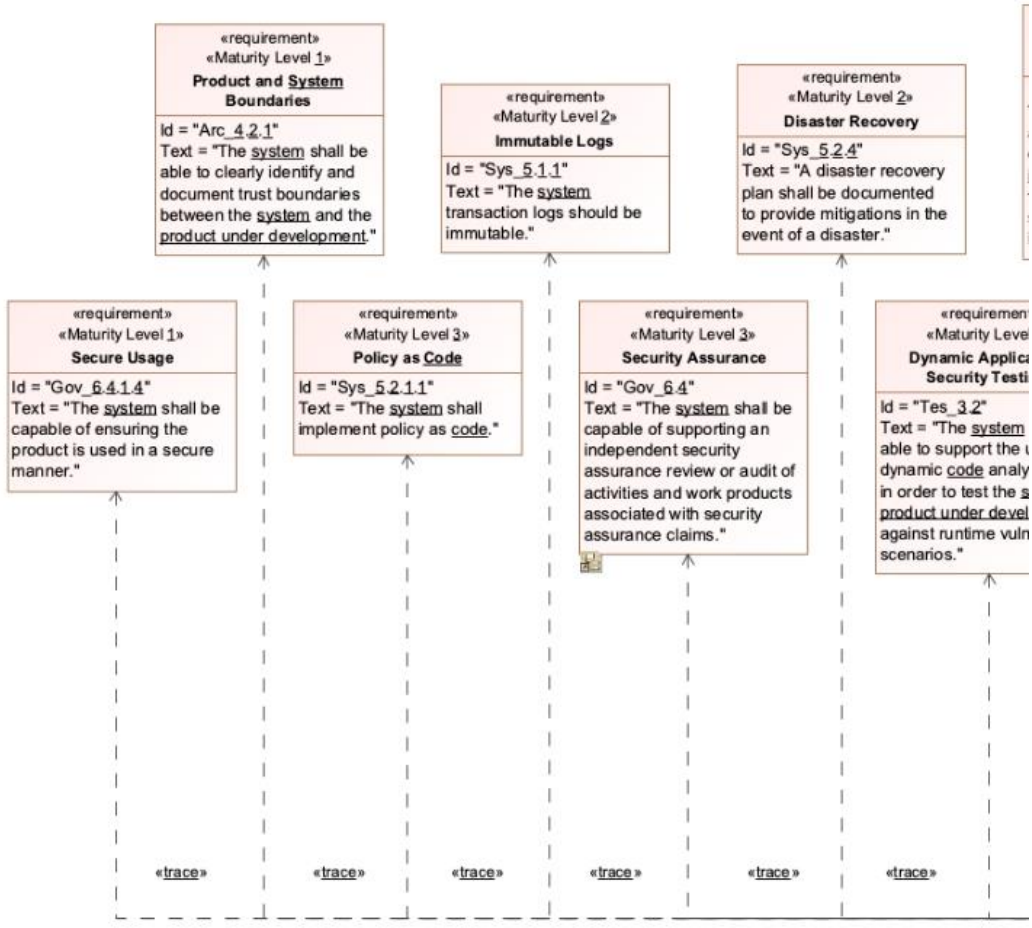

Figure 26: Example of Requirements Representation in Diagrams from PIM

Capability/Strategic Viewpoint

To architect or model a complex system on an enterprise level, we used capabilities to represent a strategic view of the system's functionality. A capability is a high-level concept that describes the ability of a system to achieve or perform a task or a mission.

The following DevSecOps capabilities were identified:

- planning and tracking
- quality assurance
- software assurance
- configuration management
- solution development
- integration
- verification and validation
- deployment
- monitor and control
- hosting services

We allocated all requirements in the DevSecOPs PIM to corresponding capabilities. This allocation allowed modelers to perform a gap analysis on the requirements. Capabilities on diagrams will be presented as a green box (as seen in Figure 27) or with a yellow circle icon with the letter "C" in it.

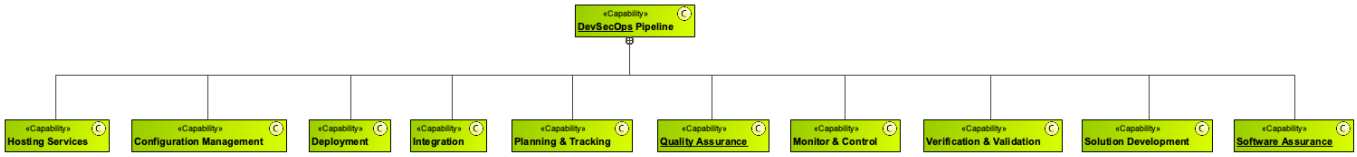

Figure 27: DevSecOps Capabilities Representation in Diagrams from PIM

Operational Viewpoint

An operational model for a system describes the behavior of the system to conduct enterprise operations. It should be "material independent" [DoD Deputy Chief Information Officer 2010]. The main operational processes for DevSecOps includes development processes for the product as well as the DevSecOps process itself. These processes are modeled by the "operational activity" element and are represented by a light lilac box or by a light-yellow rhombus icon with an arrow-like outline (see Figure 28).

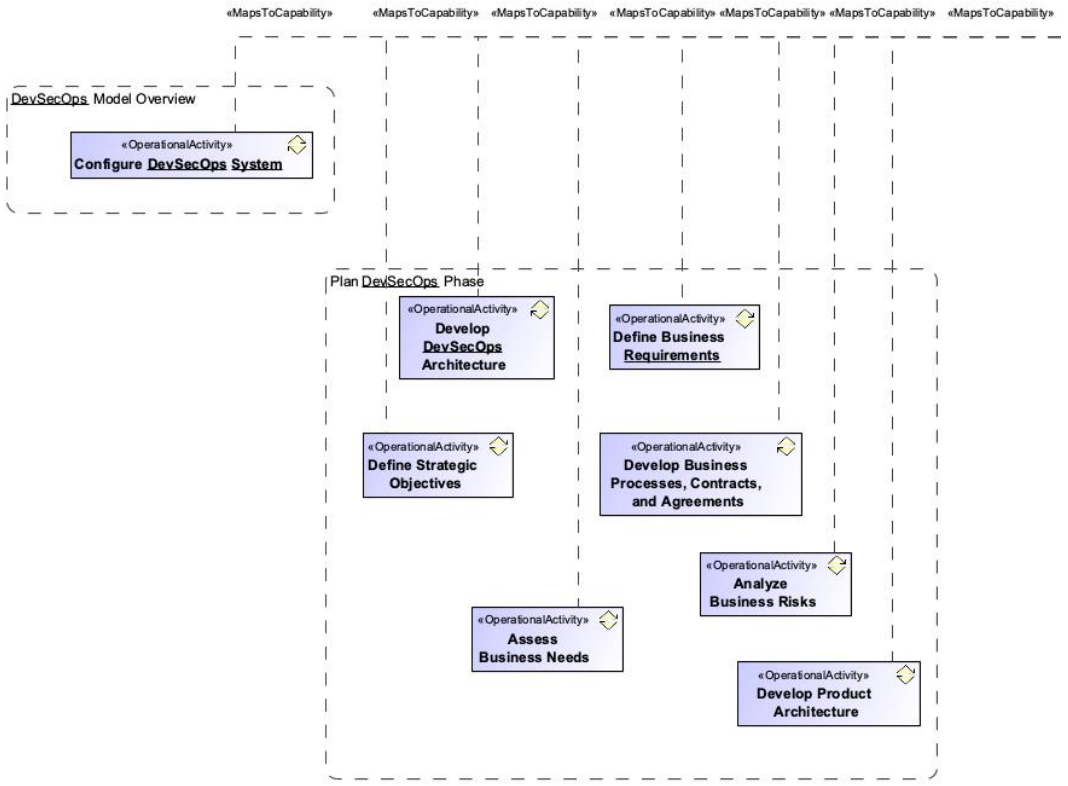

Figure 28: Example of Operational Activities Representation in Diagrams from PIM

Personnel Viewpoint

We used UAF's personnel viewpoint to model an organization structure that supports the socio part of a DevSecOps system. We implemented a RACI matrix in the model where we identified critical and non-critical roles and responsibilities for the operational processes that may be involved in or impact an operational behavior of the system. Unfortunately, this viewpoint is not standard within UAF, thus a custom profile was created to introduce additional new elements and relationships that enable modeling more aspects of cybersecurity and the overall socio-technical relationships within DevSecOps.

On the diagrams and other views from the model, the role elements are depicted as an orange box (Figure 29) or as a white or green human-like icon.

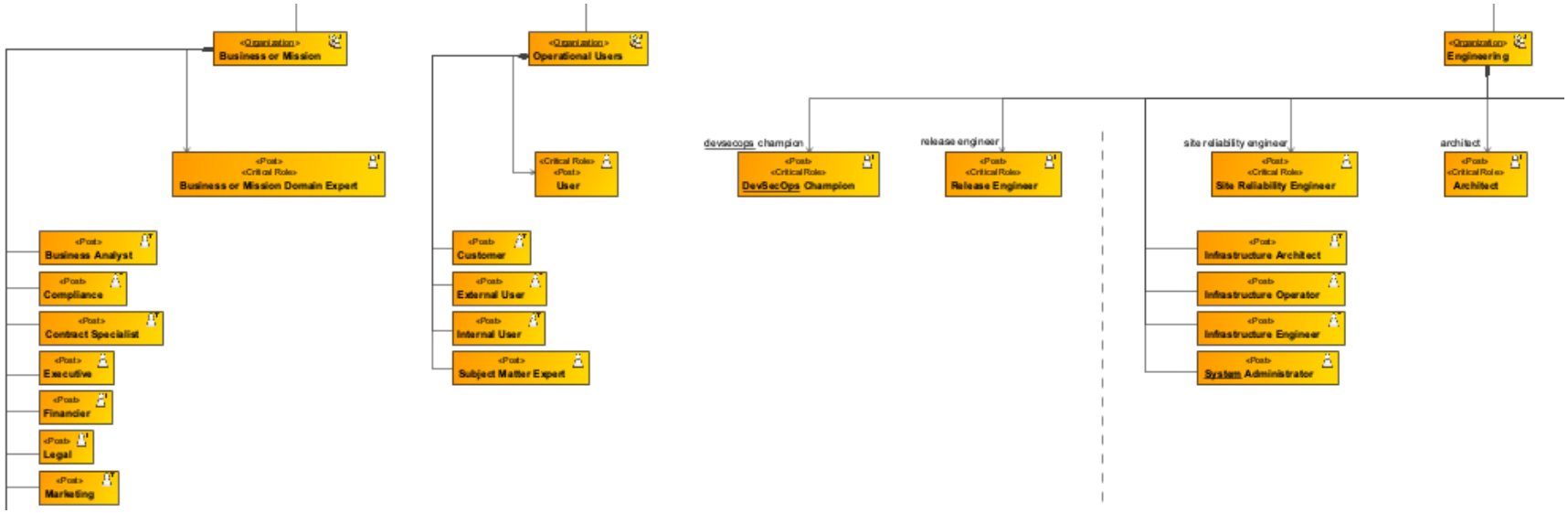

Figure 29: Example of Roles Representation in Diagrams from PIM

Security Viewpoint

The security viewpoint serves to address "the security constraints and information assurance attributes" [OMG Unified Architecture Framework 2020]. It provides architects with a way to define security aspects (including risks, security processes, and mitigations) and trace their implementation throughout the architecture. Unfortunately, this viewpoint does not support more detailed security practices to analyze the system's operational and solution architecture, such as threat modeling. To fill this gap, this viewpoint was extended with the threat-modeling custom profile (see Appendix B), which introduced additional new elements and relationships that enable modeling more aspects of cybersecurity.

To gather the information needed to populate the DevSecOps security viewpoint, the team performed a threat-modeling workshop that identified threats and threat actors that may compromise the DevSecOps system's processes. See Appendix B: Building and Modeling Threat Scenarios for more details regarding the workshop. The workshop concentrated on the operational aspects of the system where several threat scenarios and their effects were identified. In addition, workshop participants determined compromised operational processes and threat actors. Further analysis resulted in participants identifying one or more cyber attack types for each threat scenario. We used the Common Attack Pattern Enumeration and Classification (CAPEC) catalog [Mitre 2023] as a library of attack types. After the workshop, we modeled the results for each threat scenario with standard UAF security viewpoint elements and a threat-modeling custom profile (see Appendix B for details).

In the model, and as a result in this paper, threats and other security-related elements are depicted as blue boxes with red flag or blue shield icons, and attack types are depicted as pink boxes with lightning icons or just lightning icons (Figure 30).

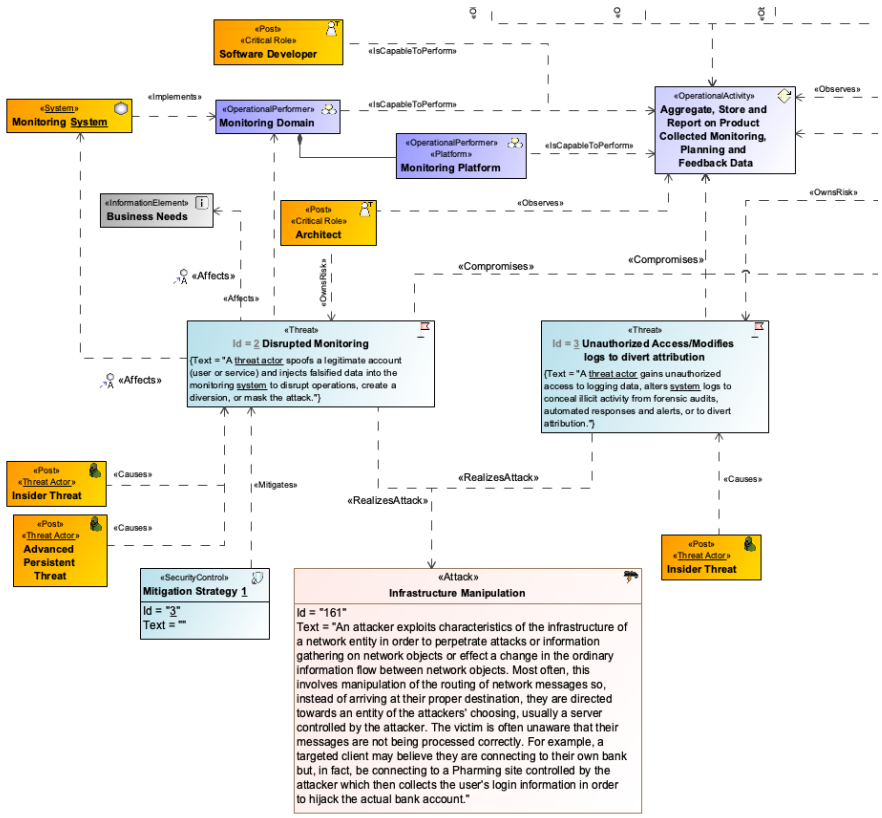

Figure 30: Example of Security Elements Representation in Diagrams from PIM

Appendix B: Building and Modeling Threat Scenarios

Table 4 details a structured approach to building threat scenarios using a piecewise construction technique. The method is to first identify the components of the threat scenario, then construct the statement from the components. Threat statements can be used to help create software assurance cases and building-resilient systems. Table 5 illustrates an example of how we used the different elements of the threat scenario to create the statement. Construction of threat scenarios will require a PIM or a well-understood instantiated DevSecOps pipeline as inputs to the method. Without an architecture to reference, any threat scenario created as an outcome of this process may not be specifically applicable to a real implementation.

Threat scenarios have been defined to consist of six parts using the following structure:

[Actor] [Action] [Attack] [Asset] ([Effect] | [Objective])

The terms of the threat scenario structure are defined in Table 4, below. Using this structure along with a few adjectives and prepositions, one can form a threat-scenario statement as follows:

An [actor] performs an [action] to [attack] an [asset] to achieve an [effect] and/or an [objective].

An example of a completed threat scenario using this structure can be seen in Table 5, below.

Table 4: Threat-Scenario Template Definitions

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.

Table 5: Threat-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	Damage organization, vulnerabilities are 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.

Table 6 outlines the process developed for creating threat scenarios in a team setting. This process is presented in the format of a workshop involving 10 steps and includes specific entry and exit criteria. The model used as input can be either a PIM or a well-defined instantiated DevSecOps pipeline. PIM-generated threat scenarios may apply to the instantiated DevSecOps pipeline but may be more generic and lack the expressive power and specificity of threat scenarios created using documentation associated with an instantiated DevSecOps pipeline.

Table 6: Threat-Scenario-Generation Workshop

Purpose	Identify threat scenarios for a given system	
Entry criteria	The following UAF-defined views have been created for the system under evaluation: • requirements diagrams • operational-process flows • relationships between operational activities and system requirements • operational resource structure, posts (i.e., roles), and corresponding responsibilities, including the involvement relationships	
General	As the system architecture and associated system instantiation evolves, so will the threats and corresponding mitigations. While this process defines an approach to systematically define applicable threat scenarios for the given system, threats should be identified, evaluated, and captured continuously outside this process. During the structured and unstructured brainstorming activities, there are no right or wrong ideas. The goal is to identify any reasonable action that can be taken to exploit the various activities within the system to ultimately impact the final product. The ideas will be evaluated later in the process.	
Step	**Activities**	**Description**
1	Planning	• Identify relevant stakeholders. Participants must contain a mix of engineering, operational, user, business, and cybersecurity experience. • Schedule a date and time or series of events in which all relevant stakeholders can actively participate.
2	Kick-off Event	• Review the workshop process and introduce participants. • Discuss the goals and objectives of the workshop. • Introduce participants to the concept of system threats and review a few example threat scenarios that follow the format of the threat-scenario template.
3	System and Architectural Overview	• Outline the system's purpose and constraints. • Review the system's architectural views and relationships. – Requirements

		– Strategy – Personnel – Operational
4	Operational Process Flow Focus Area	• Select an operational-process flow to focus the threat scenario generation. • Review the selected operational-process flow to gain an 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.
5	Unstructured Brainstorming	• Select an operational activity within the operational-process flow. • Either working individually or in pairs, brainstorm threats for the selected operational activity and write them down. Threats can bridge multiple operational activities. The brainstormed ideas should be captured in the individual's natural language. • Using an affinity diagram, organize the threats identified by the whole group and remove duplicates. • Create a list of potential threats to the system.
6	Structured Brainstorming	• Use the same operational activity as in step 5. • Break into groups of two–three people. • In small groups, identify ways that the operational activity may be exploited to interrupt the confidentiality, integrity, and/or availability of the system. Utilize the process-specific Spoofing, Tampering, Repudiation, Information Disclosure, Denial of service, and Elevation of privilege (STRIDE) threat-modeling taxonomy to reduce individual bias and to holistically identify threats to the given activity. • Using an affinity diagram, organize the threats identified by the whole group and remove duplicates. • Add new threats to the list of potential threats to the system created in step five.
7	Threat-Scenario Definition	• If this is the first time any of the participants have written threat scenarios, select a threat from the list and complete the threat-scenario template as a group. Repeat until everyone understands how to complete the threat-scenario template. • Break into small groups of three–four people. • Divide the list of potential threats to the system between the small groups. Alternatively, create a pull system in which the small groups claim a potential threat from a centralized list as needed. • In small groups, complete the threat-scenario template for each assigned or pulled potential threat. • Review and update all completed threat scenarios as a whole group, removing or consolidating duplicates.
8	Operational Activity Threat Identification	• Select the next operational activity within the selected operational process flow. • Repeat steps 5–7. • Repeat step 8 until threats have been identified for all operational activates within the selected operational process flow.
9	Operational-Process Flow Threat Identification	• Repeat steps 4–8 until threats have been identified for all operational process flows for the given system.
10	Consolidation and Review	• Consolidate all threat scenarios into a central list. • Review and accept the threat scenarios.
Exit Criteria		Create a list of structured threat scenarios that cover the operational activities in the given system.

STRIDE is a model used for identifying computer security threats. Table 7 represents a STRIDE taxonomy developed in the context of UAF, which is the enterprise architecture representation used by the DevSecOps PIM. The DevSecOps PIM specifically uses UAF's operational viewpoint that addresses operational processes. The scarce (missing or insufficient) process was added to the traditional STRIDE mnemonic; thus, STRIDES is represented in the Table 7 columns. This adaptation was derived from [Shostack 2014].

Table 7: Process-Specific STRIDES Threat-Modeling Taxonomy

Element	Interaction	Spoofing	Tampering (modifying, corrupting, losing, destroying)	Repudiation	Information disclosure	Denial of service	Elevation of privilege	Scarce (missing or insufficient) process
Process	Process has outbound data flow to data managing process	Downstream-data-managing activity is spoofed, and main process writes to the wrong place	Tampering with outgoing data		Current activity provides to data-managing-activity data of wrong classification (e.g., unauthorized distribution of controlled data)			An activity is skipped entirely
	Process sends output to external process	Downstream external activity is spoofed, and the wrong activity is communicated with		Downstream external activity claims not to have been called by current activity	An external activity receives data that it should not have access to	Current activity is not available due to corrupted state	An external activity can impersonate an internal activity and use its privilege	
	Process sends output to external interactor (human)	A role that is performer, approver, contributor, or observer for the activity is spoofed		Role disclaims seeing the output	Unauthorized user/role gets access to an activity	Current activity is not available due to responsible role unavailability		
	Process has inbound data flow from data-managing process	Upstream data managing activity is spoofed	Activity is corrupted by data read from a data-managing activity			Current activity is not available due to data flow interruption	Current activity internal state is corrupted based on data read from upstream activity	

Element	Interaction	Spoofing	Tampering (modifying, corrupting, losing, destroying)	Repudiation	Information disclosure	Denial of service	Elevation of privilege	Scarce (missing or insufficient) process
	Process has inbound data flow from a trusted process	Current activity believes it is getting data from an upstream activity	Tampering with data incoming into activity	Downstream external activity denies receiving data from current activity		Current activity is not available due to control flow disruption	An activity passes data that allows it to change the internal flow of the current activity	
	Process has inbound data flow from external process	Current activity believes it is getting data from an upstream activity						

Element	Interaction	Spoofing	Tampering (modifying, corrupting, losing, destroying)	Repudiation	Information disclosure	Denial of service	Elevation of privilege	Scarce (missing or insufficient) process
Data flow (commands/responses)	Crosses machine boundary		Tampering with data incoming into activity (human-in-the-middle attack)		Data that passes from activity to activity is sniffed "on the wire" (intercepted by an unauthorized actor)	The data flow between activities is interrupted by an external entity		
Data store (database/backend)	Process has outbound data flow to data store		Tampering with incoming data (corrupted in a data store)	Current activity claims not to have provided data to data-managing activity	Data was disclosed by data-managing activity, such as mishandling of data or unauthorized access to the activity			
	Process has inbound data flow to data store			Current activity claims not to have received data from data-managing activity	Data was disclosed by data managing activity, such as mishandling of data or unauthorized access to the activity	Data managing activity fails to store information		
External activity (an activity that is part of a separate process/flow)	External interactor passes input to process	Main activity is confused about the identity of the performer/contributor role		Current activity claims not to have received data	Current activity receives unnecessary data	Data managing activity fails to provide authorized data access		
	External interactor gets input to process	Performer/contributor role is confused about the identity of the current activity						

The following table (Table 8) outlines a process for capturing threats and mitigations within the UAF standard. While UAF includes a security viewpoint, it was extended with a threat-modeling custom profile that introduced additional new elements and relationships that allow modeling of the various nuances of cybersecurity.

Table 8: Modeling Threats in UAF

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 view point'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 in the personnel view point, 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° 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.

Creating a Threat-Modeling Profile within UAF

As mentioned above, UAF security viewpoint was extended to accommodate threat-modeling aspects of cybersecurity. The extension includes the creation of a threat-modeling profile (Figure 31). It should be noted that this profile is not a stand-alone profile that can be recreated in any SysML model, like SysML-Sec [Roudier 2015]. This profile will work well as an extension to UAF security viewpoint and with other UAF viewpoints.

New element and relationships implemented by the threat-modeling profile include

- threat element with main attributes
  - ID
  - name
  - text
  - effect
  - objective
  - riskOwner
- attack element with main attributes
  - ID
  - name
  - text
  - abstraction
  - link
- threat-actor stereotype (to apply to post element representing external threat actors)
- security-requirement stereotype (to apply to requirement element)
- compromises relationship (from threat element to operational-activity element)
- RealizesAttack relationship (from threat element to attack element)
- causes relationship (from post element to threat element)

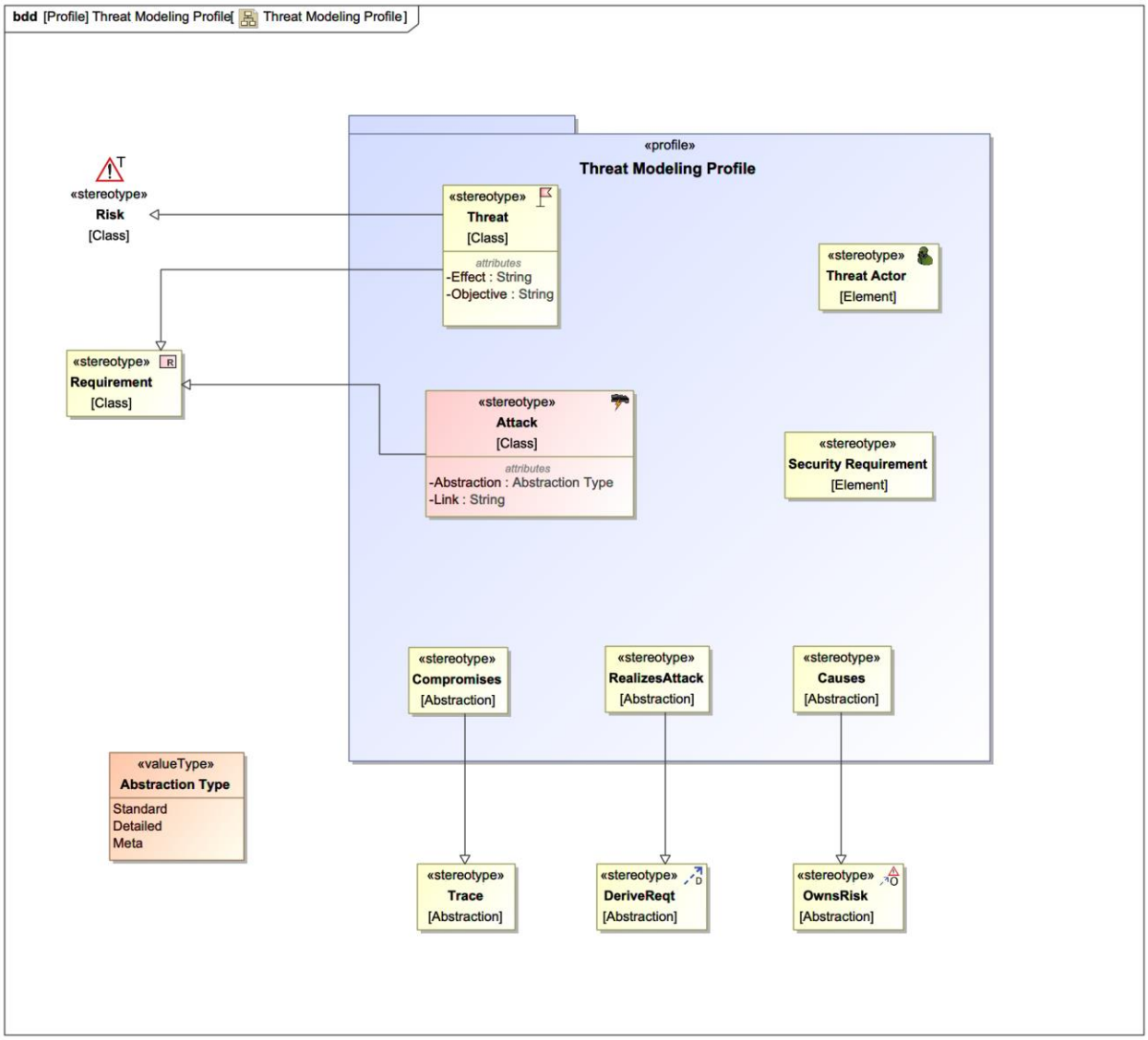

Figure 31: Threat-Modeling Custom Profile Diagram

Creating an Involvement Profile Within UAF

Another custom profile was created to accommodate the complexity presented by different stakeholders getting involved with the system's processes. The profile is called the "involvement profile." There are several approaches to identify the type of relationship between a relevant stakeholder and an operational process, such as a relevant stakeholder-involvement matrix (RSIM), stakeholder-role-assignment matrix (SRAM), or RACI matrix [Wikipedia 2023b]. The involvement profile implements a version of the RACI matrix by adopting terms and definitions to the PIM needs. Thus, the profile was designed using the following definitions:

- Producer – a role responsible for performing the activity or producing the deliverable. This role's action is to perform.
- Approver – a role accountable for approving the activity or deliverable. This role's action is to approve.
- Contributor – a role that needs to be given an opportunity to provide input on the activity or deliverable before it is completed. This role's action is to contribute.
- Observer – a role that needs to be informed of the activity or deliverable after it is completed. This role's action is to observe.

The involvement profile is an extension of UAF. All roles can be modeled as a performer/operational performer or any type of organizational resource (post, organization) from UAF. Additionally, UAF's relationship "IsCapableToPerform" can be used to model "to perform" action for the producer. However, the other three actions cannot be easily mapped to any of UAF's relationships. Thus, additional actions required that we create a custom profile. We created the involvement profile (Figure 32) to implement three roles (approver, contributor, and observer) and corresponding actions. Even though one can model approver, contributor, and observer roles with UAF standard elements, we included new elements in the profile to model cases when these roles needed to be modeled explicitly and exclusively from UAF standard elements. Thus, the involvement profile contains the following new elements and relationships:

- approver element
- observer element
- contributor element
- approves relationship (from a role element to operational activity)
- observes relationship (from a role element to operational activity)
- contributesTo relationship (from role element to operational activity)

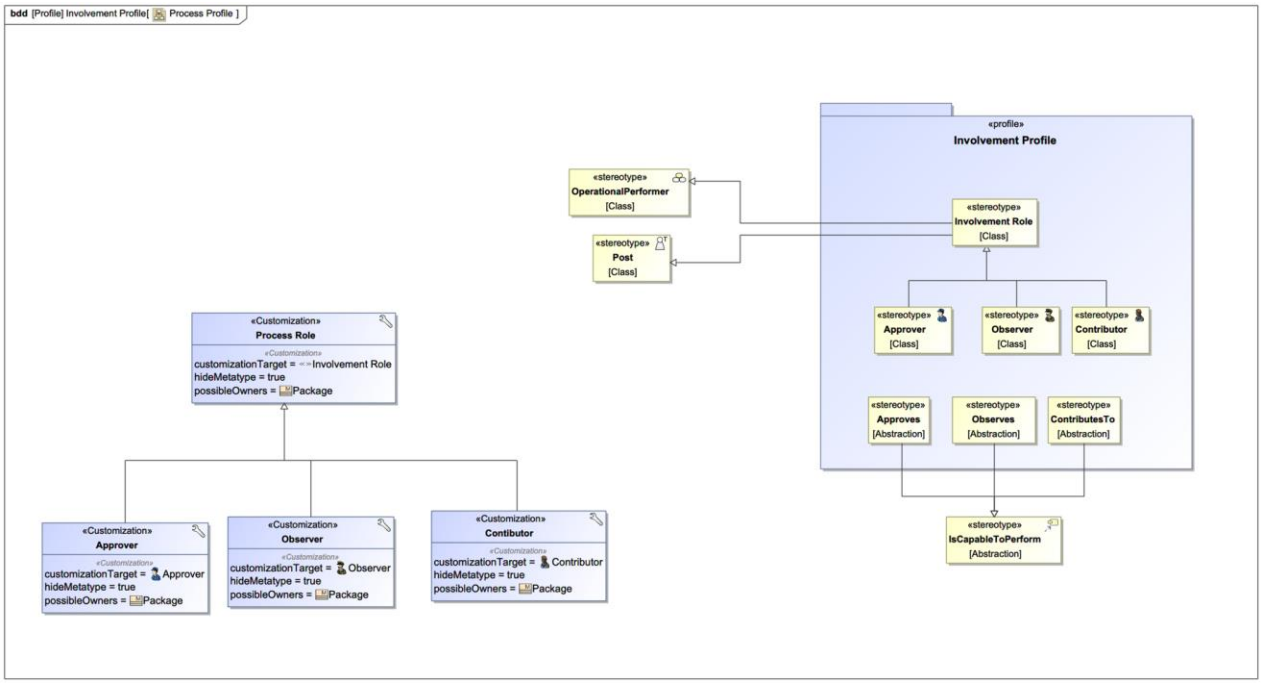

Figure 32: Involvement Profile Custom Profile Diagram

Abbreviations and Acronyms

BPM	Business-process modeling
CI/CD	Continuous integration/continuous deployment
CM	Configuration management
FISMA	Federal Information Security Management
GDPR	General Data Protection Regulation
HIL	Hardware in loop
HIPAA	Health Insurance Portability and Accountability Act
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
MBSE	Model-based systems engineering
NIST	National Institute of Standards and Technology
PCI DSS	Payment Card Industry Data Security Standard
PIM	Platform Independent Model
RACI	Responsible, accountable, consulted, and informed
RSIM	Relevant-stakeholder-involvement matrix
SaaS	Software as a service
SRAM	Stakeholder-role-assignment matrix
STRIDE(S)	Spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege, (scarce process)
UAF	Unified Architecture Framework

Bibliography

[Asan 2013]
Asan, E.; Albrecht, O.; & Bilgen, S. Handling Complexity in System of Systems Projects – Lessons Learned from MBSE Efforts in Border Security Projects. Pages 281–299. 4th International Conference on Complex System Design & Management. April 2021.

[Bass 2015]
Bass, Len; Weber, Ingo; & Zhu, Liming. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional. 2015. ISBN 978-0-13-404984-7.

[Bloomfield 2014]
Bloomfield, R. E. & Netkachova, K. Building Blocks for Assurance Cases. *International Symposium on Software Reliability Engineering (ISSRE)*. November 2014.

[CMU SEI 2022]
DevSecOps Platform Independent Model Portal. *Carnegie Mellon University Software Engineering Institute GitHub.* 2022. https://cmu-sei.github.io/DevSecOps-Model/

[Dassault Systèmes 2023]
Dassault Systèmes. Trasnforming Industries, Markets and Customer Experiences. *3Ds.com*. 2023. https://www.3ds.com/industries

[Dictionary.com 2023]
System Definition and Meanings. *Dictionary.com.* 2023. https://www.dictionary.com/browse/system

[DoD Deputy Chief Information Officer 2010]
DoD Deputy Chief Information Officer. DoDAF Viewpoints and Models. *Chief Information Officer, U.S. Department of Defense.* August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_operational/

[Goodenough 2012]
Goodenough, John B.; Weinstock, Charles B.; & Klein, Ari Z. *Toward a Theory of Assurance Case Confidence*. CMU/SEI-2012-TR-002. Software Engineering Institute, Carnegie Mellon University. 2012. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=28067

[Haskins 2007]
Haskins, Cecilia. Using Patterns to Transition Systems Engineering from a Technological to Social Context. *Systems Engineering*. Volume 2. Issue 11. 2007. Pages 147–155.

[Mead 2010]
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

[Miller 2012]
Miller, Lori Ann. Modeling Forward Base Camps as Complex Adaptive Sociotechnical Systems. *Masters Theses*. 6943. 2012.

[Mitre 2023]
Mitre. *Common Attack Pattern Enumeration and Classification*. April 2023. https://capec.mitre.org

[Morales 2020]
Morales, Jose A.; Turner, Richard; Miller, Suzanne; Capell, Peter; Place, Patrick R.; & Shepard, David James. *Guide to Implementing DevSecOps for a System of Systems in Highly Regulated Environments*. CMU/SEI-2020-TR-002. Software Engineering Institute, Carnegie Mellon University. 2020. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=638576

[Object Management Group 2010]
Object Management Group. About the Business Process Model and Notation Specification Version 2.0. *OMG.org*. December 2010. https://www.omg.org/spec/BPMN/2.0/About-BPMN

[OMG Unified Architecture Framework 2020]
OMG Unified Architecture Framework. *Unified Architecture Framework (UAF) Domain Metamodel*. OMG Document Number: formal/19-11-05. Object Management Group, Inc. 2020. https://www.omg.org/spec/UAF/1.1/DMM/PDF

[Oosthuizen 2018]
Oosthuizen, R.; Venter, J. P.; & Serfontian, C. Model Based Systems Engineering Process for Complex Command and Control Systems. *23rd International Command and Control Research and Technology Symposium*. November 2018.

[Open Risk Manual n.d.]
Cyber Risk Category. *Open Risk Manual*. n.d. https://www.openriskmanual.org/wiki/Category:Cyber_Risk

[Palmer 2016]
Palmer, Erika. 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*. July 2016. https://www.researchgate.net/publication/308092426_Investigating_structural_gender_inequality_in_the_Norwegian_pension_system_An_example_of_using_MBSE_in_the_evaluation_of_social_systems

[Prince 2022]
Prince, Claudia. RACI. *Project Management.com*. September 2022. https://www.projectmanagement.com/wikis/234008/RACI

[Roudier 2015]
Roudier, Yves & Apvrille, Ludovic. SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems. *3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. February 2015. https://ieeexplore.ieee.org/document/7323182

[SEBoK 2022]
Sociotechnical System (Glossary). SEBoK. October 2022. https://www.sebokwiki.org/wiki/Sociotechnical_System_(glossary)

[Shevchenko 2018]
Shevchenko, Nataliya. Threat Modeling: 12 Available Methods [blog post]. *SEI Blog*. December 3, 2018. https://insights.sei.cmu.edu/blog/threat-modeling-12-available-methods/

[Shostack 2014]
Shostack, Adam. *Threat Modeling: Designing for Security*. Wiley. 2014. ISBN 978-1-118-80999-0. https://www.wiley.com/en-br/Threat+Modeling%3A+Designing+for+Security-p-9781118809990

[Wikipedia.com 2023a]
Information System. *Wikipedia.com*. https://en.wikipedia.org/wiki/Information_system

[Wikipedia.com 2023b]
Responsibility Assignment Matrix. *Wikipedia.com*. https://en.wikipedia.org/wiki/Responsibility_assignment_matrix

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	April 2023	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Using Model-Based Systems Engineering (MBSE) to Assure a DevSecOps Pipeline is Sufficiently Secure	FA8702-15-D-0002

6. AUTHOR(S)

Timothy A. Chick, Scott Pavetti, Natasha Shevchenko

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2023-TR-001001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Enterprises are concerned that adversaries may abuse weaknesses in a DevSecOps pipeline to inject exploitable vulnerabilities into the enterprise's products and services. This report presents the DevSecOps Platform Independent Model (PIM), which offers enterprises an approach to use model-based systems engineering (MBSE) to evaluate and mitigate the risk that adversaries can exploit weaknesses and vulnerabilities in DevSecOps pipelines. The analysis approach this paper describes focuses on ensuring that the DevSecOps pipeline and its associated products are implemented in a secure, safe, and sustainable way; are sufficiently free from vulnerabilities; and the capabilities only function as intended. Ultimately, the PIM provides analysts with a minimum set of MBSE tools to assist with threat identification, analysis, and documentation.

14. SUBJECT TERMS	15. NUMBER OF PAGES
DevSecOps, Software Assurance, Threat Modeling	56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102