Journal of Innovation
May 2025 | 27th Edition

Object Management Group

# Making the Case for Cybersecurity

## Mending the Digital Thread with OMG Standards for Risk-Centric DevSecOps

**Authors:**

Dr. Nikolai Mansourov
KDM Analytics
*nick@kdmanalytics.com*

Djenana Campara
KDM Analytics
*djenana@kdmanalytics.com*

Harrell Van Norman
USAF AFMC AFLMC/EZAS
*harrell.van_norman@us.af.mil*

## CONTENTS

## FIGURES

The digital transformation of critical infrastructure, services, and operations has outpaced the traditional models of cybersecurity. From cloud-native logistics to cyber-physical defense platforms, organizations now operate in highly dynamic, interconnected environments where failures cascade across systems, disrupt missions, business operations, and impact entire enterprises.

While cybersecurity has long been defined by the "CIA triad"—confidentiality, integrity, and availability—these principles no longer capture the full scope of today's threats or operational demands. Disruption, deception, and degradation of function can be more damaging than data loss alone. Cybersecurity must evolve from protecting isolated digital assets to ensuring the resilience of mission-critical operations across system-of-systems.

At the same time, threats have become more persistent, well-resourced, and adaptive. Nation-state actors, criminal networks, and AI-driven tools now exploit not only code-level vulnerabilities, but also architectural flaws, configuration gaps, and supply chains.

Conventional cybersecurity practices — static controls, periodic audits, patching, and especially manual risk assessments —are increasingly misaligned with this reality. Risk assessment remains disconnected from DevSecOps and lack the speed, precision, and scalability needed to influence system evolution. Vulnerability scanners, and compliance checklists are useful – but they cannot reason about evolving risk in a timely, mission-aware context.

Today's environment demands a new paradigm - one that treats cyber risk as a continuously evolving property – proactive, system-aware, and computable throughout the lifecycle.

This article introduces **Risk-Centric DevSecOps**, a framework that integrates cybersecurity risk into the digital engineering pipelines as a continuous, structured reasoning process. Security becomes a first-class engineering concern, inseparable from system modeling, mission analysis, and operational decision-making.

At the core of this approach is the *formal cybersecurity assurance case*—a machine-consumable structure of claims and evidence that connects design models, threat intelligence, attacks, vulnerabilities and mitigation controls. The assurance case becomes more central than conventionally assumed – it can orchestrate how tools interpret risks, assess controls, and reason about adversarial capabilities. The assurance argument becomes the logic of cybersecurity and the control loop for automation - allowing systems to adapt intelligently, while maintaining traceable real-time assurance aligned with mission outcomes.

Cybersecurity in this model is no longer reactive or manually curated. Knowledge items—about systems, attack techniques, defenses, vulnerabilities and assurance—are machine-consumable, continuously updated, and semantically integrated. The assurance case becomes a system-specific knowledge graph. As soon as the global community of cybersecurity researchers or engineers publish a change (e.g. a new CVE, a new attack technique, a new SBOM entry, or a new system function), it is pulled into the DevSecOps pipelines. Risk claims are recalculated, gaps in

the test coverage identified, controls reassessed, and affected stakeholders alerted—all automatically.

Our framework builds on model-based systems engineering (MBSE), DevSecOps culture, system assurance, and digital thread concepts. We identify where digital threads are currently broken and present a roadmap—aligned with the OMG Systems Assurance Task Force—to mend them through interoperable, knowledge-based standards.

What emerges is more than a smarter DevSecOps pipeline, but a new cybersecurity ecosystem: automated, proactive, explainable, and mission-aligned.

# 1 CYBERSECURITY AS A CONTINUOUS, KNOWLEDGE-BASED PROCESS

## 1.1 CYBERSECURITY IN THE SYSTEM LIFECYCLE

While vulnerability scanning, patching, and monitoring are now common operational practices, risk assessment—when done manually—remains a costly, static process, often disconnected from the fast-paced workflows of modern system development and deployment.

The Risk-Centric DevSecOps framework repositions risk assessment as a continuous activity, tightly integrated across the system lifecycle. Security is no longer an isolated function—it becomes a core engineering concern that evolves from requirements definition through architecture, implementation, integration, authorization, and sustained operation.

Making this shift requires seamless collaboration between diverse roles: engineers, system integrators, cybersecurity specialists, intelligence-base threat analysts, DevSecOps developers, cyber vulnerability assessors, test and evaluation personnel, operational staff, and governance authorities. When these handoffs break down, opportunities for attacks are not just overlooked – they're built in.

Importantly, the adversary is not a static entity. Attackers adapt quickly, and any viable framework must account for the temporal asymmetry between system builders, defenders, and threat actors. Proactively getting ahead of the attackers is the essence of digital, risk-centric cybersecurity.

In a risk-centric DevSecOps environment, cybersecurity is no longer just about enforcing coding practices and applying security patches. Instead, it becomes a process of delivering well-reasoned security mitigations in the right context, for the right threat environment—automatically, continuously, and at scale.

## 1.2 KNOWLEDGE AS THE BACKBONE OF CYBERSECURITY

The continuous collaboration in risk-centric DevSecOps is not just between teams—it is between tools. Automation replaces manual handoffs with machine-driven exchanges, where each stage in the pipeline produces and consumes structured, interpretable messages. These messages

encapsulate cybersecurity-relevant knowledge in a form that analytics tools can act upon. They form the semantic backbone of the pipelines, enabling seamless, automated handoffs. The most critical of these messages include:

- **System knowledge**: formal, structured facts about the system's architecture, behavior, data types, interfaces, and dependencies.
- **Attack knowledge**: items enumerating generic offensive techniques (e.g. from ATT&CK or CAPEC) and tailored, system-specific attacks paths.
- **Threat Intelligence**: items enumerating motivated and capable attackers of the specific deployment environment, their preferred techniques and the likelihood of attacks in this environment.
- **Vulnerability knowledge**: items enumerating generic weakness patterns (CWE) or known component vulnerabilities (CVE), and specific, contextualized vulnerability conditions identified in the system of interest.
- **Defense knowledge**: items enumerating generic defensive techniques and controls (e.g. from D3FEND or NIST 800-53) and tailored controls for the system of interest.
- **Assurance knowledge**: structured arguments and evidence that link system facts and mitigations to claims about mission security.

In this framework, these knowledge items are not static—they are continuously ingested, transformed, and linked through inference engines. Cybersecurity becomes a data-driven process powered by a bespoke, contextualized data lake, continuously updated with both community-curated repositories and system-specific facts.

## 1.3 FROM TOOL-CENTRIC PIPELINES TO KNOWLEDGE PIPELINES

Traditional DevOps and DevSecOps pipelines are focused on coarse-grained artifacts—code repositories, containers, and static configuration files. Even when security scans are included, they remain reactive and detached from the evolving mission context.

In contrast, risk-centric DevSecOps introduces a fine-grained, semantically rich knowledge pipeline. Instead of moving only files, the pipeline moves structured knowledge items: model elements, claims, evidence, and risk inferences. It starts not with source code, but with a mission model. It continues through requirements definition, system modeling, software development, testing and evaluation, and into runtime monitoring—driven by a formal, continuously updated assurance argument.

The DevSecOps pipeline evolves into a distributed model-based reasoning system, contributing to a living cybersecurity assurance case. As models change, new vulnerabilities are discovered, or system configurations are updated, only the affected claims are re-evaluated. This allows teams to reason about cyber risk continuously, and at the speed of modern development.

### 1.4 Interoperability Standards and Automation

To realize this vision, interoperability standards are essential. For knowledge items to flow across tools, organizations, and lifecycle stages, they must be represented in a shared, semantically meaningful format—understood not just by people, but by machines.

Traditional pipelines rely on bespoke integrations and siloed toolchains. But to automate reasoning about risk and assurance, we need common models, shared vocabularies, and standardized structures that allow diverse tools to interpret and act on the same underlying knowledge.

Interoperability standards provide this foundation. They define how digital knowledge items are described, linked and exchanged across tools and organizations. Without a shared semantic foundation, digital threads break and automation stalls.

With these standards in place, pipeline stages can be triggered by fine-grained changes—not just to code, but to mission assumptions, system configurations, or updated intelligence. When a new attack technique appears, or a design element changes, the affected portions of the security argument can be automatically re-evaluated, generating updated risk posture and informing relevant stakeholders.

Ultimately, interoperability transforms the cybersecurity pipeline into a living, adaptive knowledge graph—where tools act on shared semantics, and each stakeholder contributes to, and benefits from, a continuously evolving understanding of system and mission risk.

## 2 System Facts as Foundational Knowledge for Cybersecurity Pipelines

What is the foundation for the assurance case? Cybersecurity, when approached as a continuous and fine-grained knowledge-driven process, is founded on a clear, machine-readable understanding of the system itself. These "system facts"—models of structure, behavior, interfaces, dependencies, and operational context—form the semantic foundation for tailoring other knowledge domains: threat intelligence, vulnerability data, assurance arguments, and security controls. Without accurate and interpretable system representations, automated analysis and adaptive defense are impossible.

At the heart of this system-level knowledge flow are two key modeling languages: SysML (Systems Modeling Language) [7] and UAF (Unified Architecture Framework) [8]. Together, they underpin Model-Based Systems Engineering (MBSE) and Mission Engineering, enabling stakeholders to build semantically rich, structured representations of systems-of-systems, mission objectives, and their interdependencies.

### 2.1 SysML: From Communication to Computation

SysML has revolutionized the way engineers specify and communicate designs. In contrast to the fragmented document-based approaches of the past—where PowerPoint, spreadsheets, and ad

hoc diagrams dominated—SysML offers a common, formal vocabulary for modeling systems in a tool-based environment. These tools do not merely draw diagrams; they store and interlink system information in a central model database, often referred to as the "authoritative source of truth."

These model elements - components, interfaces, functions, scenarios, constraints, and more - when semantically defined, allow for automated reasoning and analysis. Increasingly, SysML models are being reused throughout the lifecycle in cybersecurity applications such as attack surface evaluation, dependency analysis, and risk assessment.

However, SysML's flexibility—a strength in bespoke engineering workflows—poses a major challenge when models need to be interpreted by third-party cybersecurity tools. Without standardized semantics, the same concept (e.g. a data flow or computing node) might be represented differently across models, limiting tool interoperability and hindering automation.

In traditional MBSE environments, models are often consumed only by the engineering teams that created them. But cybersecurity is inherently inter-organizational: third-party tools, assessors, acquirers, engineers, intelligence analysts, T&E, and certification authorities must all interpret the same models to reason about risk.

This makes semantic interoperability essential. While SysML v2 improves internal model semantics and introduces better support for ontologies, it does not define what the models mean in terms of real-world cyber risk.

## 2.2 Tailoring Semantics for Cyber-Physical Systems

Cyber and cyber-physical systems introduce domain-specific needs beyond what generic SysML captures. These systems involve digital information processing, embedded hardware, software supply chains, and connections to physical environments. Elements are not merely abstract components—they represent processors, buses, protocols, BOMs and SBOMs, and mechanical or electrical subsystems. Connectors often carry data or control signals; exchanges have types, timing constraints, and trust implications.

Moreover, such systems include emergent properties—functions, capabilities, and mission outcomes—that arise from the collaboration between subsystems. Understanding these emergent elements is crucial for assessing mission assurance and cybersecurity posture.

Engineering teams address this by introducing proprietary stereotypes or model extensions, but without a shared standard, these adaptations remain opaque to external tools. As a result, even well-crafted models become inaccessible to cybersecurity analytics, and risk assessment remains disconnected from system design.

### 2.3 MENDING THE THREADS WITH OMG SPECTRA

This is where the OMG SPECTRA standard comes in. SPECTRA (Systems Profile for Effective Cyber Threat-based Risk Assessments) defines a standardized semantic profile for cyber and cyber-physical systems, expressed in SysML. Its purpose is to enable security-focused tools to extract meaningful knowledge from system models—accurately, consistently, and automatically.

SPECTRA provides machine-readable semantics for constructs relevant to cybersecurity, including replaceable parts, information channels, and domain-specific data types. It enables tools to identify the *technical digital surface* of a system—the parts and pathways exposed to potential attack—and reason about vulnerabilities and controls in that context.

Together, SysML and UAF enable digital threads—structured, traceable pathways linking mission objectives to design elements, test artifacts, controls, and runtime behaviors. These threads form the foundation for big data analytics and adaptive security reasoning, provided they are unbroken and semantically coherent.

Importantly, SPECTRA doesn't require discarding existing SysML or UAF practices. Instead, it acts as a **semantic overlay**—filtering generic system and mission models to highlight the elements necessary for cybersecurity analysis. This enables automated tailoring of threats, risk scenarios, and mitigations based on the actual architecture of the system of interest.

## 3 RISK CLAIMS AS INFERENCE IN THE RISK-CENTRIC DECSECOPS PIPELINES

Having extended the DevSecOps culture to the model-based systems engineering, and mission engineering, how can risks be identified in this continuous environment, to make it risk-centric?

### 3.1 MANUAL RISK ASSESSMENT IS DEAD

In agile, cloud-native environments, manual risk assessments are a bottleneck. They depend on human interpretation of static documents, are rarely aligned with system reality, and cannot react to fast-changing threat conditions. Worse, they are disconnected from DevSecOps pipelines—creating a critical gap between operational velocity and assurance posture.

In contrast, the proposed framework treats risk assessment as a continuous, automated process, embedded directly into the development and operations lifecycle. System facts, vulnerabilities, threat models, and control data are represented as a structured knowledge graph. As this graph evolves—due to design changes, new CVEs, updated threat intelligence, or even runtime anomalies—machine-readable claims are recalculated, and assurance arguments are updated automatically.

Each stakeholder, from engineers to mission owners, receives targeted, up-to-date insight into the cybersecurity implications of their work. Risk awareness becomes persistent, explainable, and actionable—at the speed of deployment.

## 3.2 Reimagining Risk as Structured Inference

This transformation begins with a new conceptual foundation: risk is not a score—it is a structured situation. It exists when a motivated attacker can exploit a system-specific vulnerability to harm a mission-relevant asset. This condition must be supported by traceable evidence about the system, its exposures, and its defenses.

In this framework, each risk involves a structured argument—a chain of logical subclaims about attackers, attack paths, system vulnerabilities, mitigation gaps, and potential impacts. These claims are orchestrated by an assurance argument that does not sit passively at the end of the pipeline, but instead drives the inference of knowledge, the propagation of evidence, and the triggering of analytic actions.

Automation becomes possible because each claim type has defined inputs, semantic meaning, and justifiable outputs. The pipeline is no longer a sequence of tools—it becomes a reasoning engine, producing machine-consumable narratives of risk.

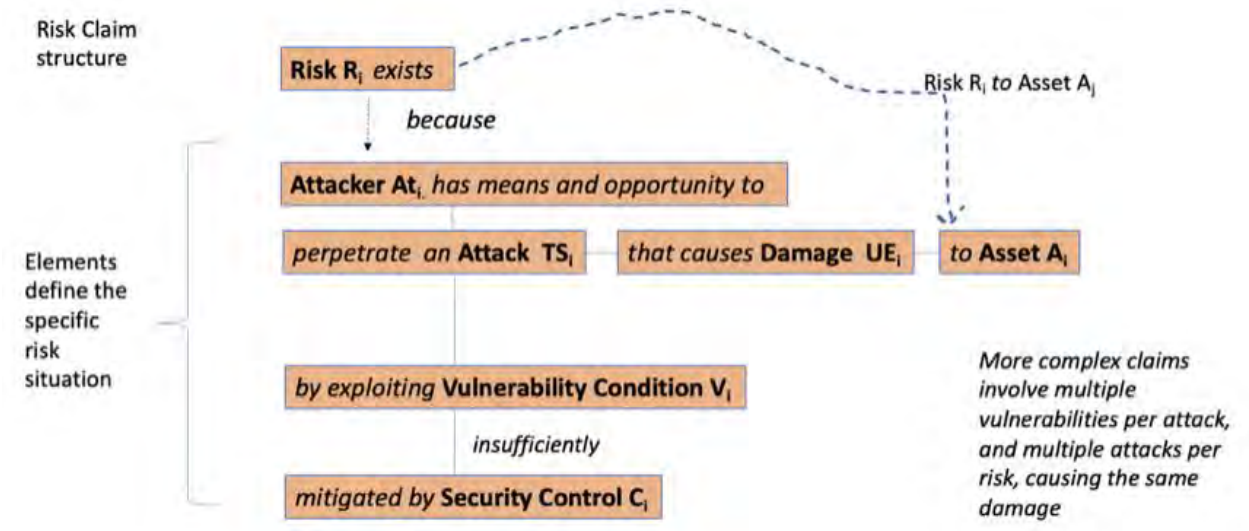


Figure 3-1: Risk claim structure.

## 3.3 The Attack-centric Risk Claim Structure that Supports Automation

At the heart of this reasoning structure lies a critical design decision: attack is the common denominator of all risk logic. Every risk ultimately resolves to the presence of one or more technically feasible, insufficiently mitigated attack paths.

This is not just a modeling convenience—it is what enables automated inference and integration with external cybersecurity knowledge. Repositories like MITRE ATT&CK (offensive techniques) and MITRE D3FEND (defensive techniques) organize knowledge around attacks. By aligning risk claims to this structure, we enable the flow of curated, community-maintained knowledge into bespoke security assessments.

Each subclaim depicted in Figure 3-1—risk **R**, asset **A**, attacker **At**, attack **Ts**, undesired effect **UE**, vulnerability condition **V**, and security control **C**—is also an *object* in the linked knowledge graph and is *formally characterized*, ensuring traceability and semantic clarity.

The individual risk subclaims/objects are derived from the normalized system facts, as defined by SPECTRA. This structured model enables reasoning systems to identify, enumerate, and link together relevant facts and evidence from disparate sources: system models, threat intelligence databases, code analysis tools, and even runtime telemetry.

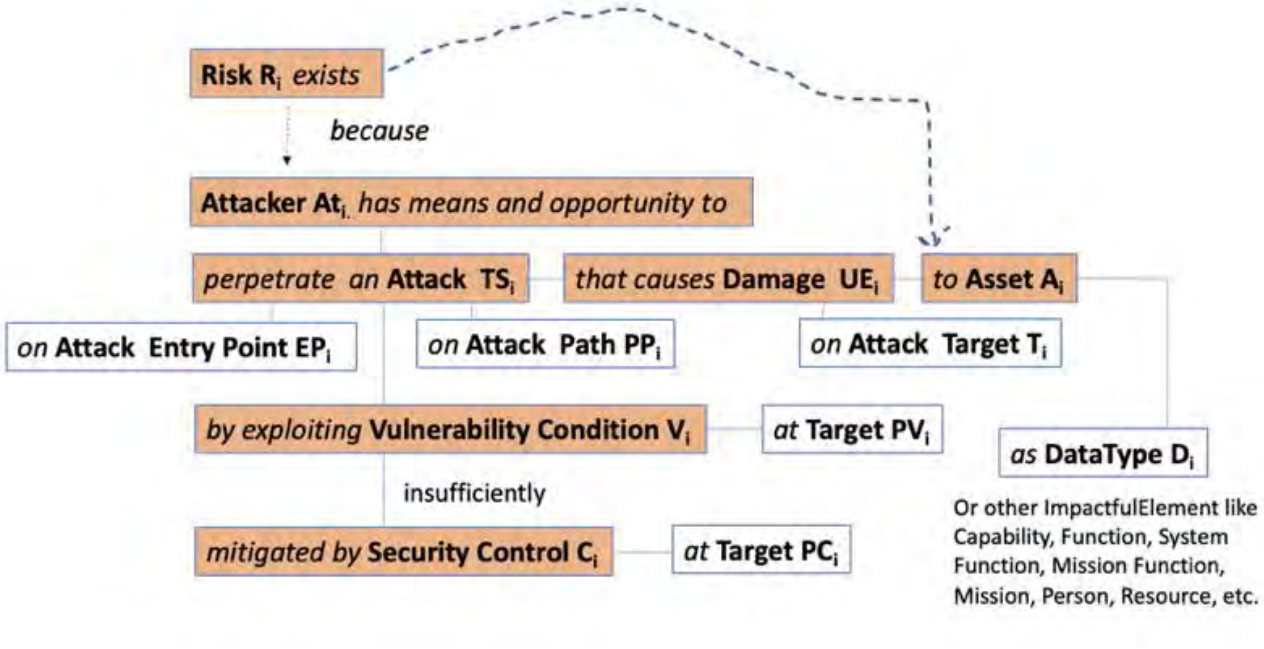


Figure 3-2: Risk caim with links to system facts.

## 3.4 Risk Clustering and Comprehensive Enumeration

Once risks are grounded in viable attacks, they can be comprehensively enumerated. The system builds *risk clusters*—interconnected graphs of logically attack claims and associated assets, impacts, and defenses. These clusters do not merely enumerate attack paths; they represent fully reasoned, evidence-backed narratives of how mission outcomes could be compromised, and under what circumstances. Figure 3-3 depicts the results of risk calculation.

Importantly, the framework separates:

- A technical *feasibility claim*, asserting that an attack is possible given current conditions.
- A *likelihood claim*, which incorporates threat environment and attacker modeling to estimate plausibility.
- A *residual risk claim*, which accounts for the presence and quality of mitigations.

This separation is what enables real-time integration of threat intelligence. The technical surface of the system does not change based on attacker capabilities—but the *likelihood of exploitation*

does. Threat likelihood reflects the capabilities, intentions, and opportunities of attackers, as well as the trust boundaries and operating environment of the system.

Risk measurements also incorporate the presence and quality of mitigations—controls and countermeasures modeled or planned. This supports "what-if" simulations where different defense strategies are applied to the same attack graph.

This modular architecture reflects zero-trust principles: risk is assessed based on what the system exposes, not what it assumes about its users or environment. Trust can be represented as a re-calibrated attack likelihood claim in a given threat environment, not a baked-in assumption.


Figure 3-3: Risk measurement claims.

By clearly separating the attack surface from the threat intelligence and mitigation, the framework supports:

- Continuous re-evaluation as threat intelligence evolves
- Mission-aware prioritization of risks
- Evidence-driven trade-offs between cost, control, and exposure.

# 4 Attack Path Characterization in Risk-Centric DevSecOps

As an attack path is central to making a risk claim, how can we orchestrate attack path enumeration to justify completeness of risk claims, and do so continuously?

In a risk-centric DevSecOps framework, an attack is not merely a theoretical threat—it is a system-specific scenario in which an adversary exploits architectural or operational conditions to compromise mission-relevant assets. To support automation and continuous reasoning, such attack paths must be formally characterized. This means linking offensive techniques to the specific system facts that make them technically feasible.

We define *attack knowledge item* as the structured, machine-consumable representation of how and where an attack can occur in a given system. Consider the two linked objects—Attacker and Threat Scenario—illustrated in Figure 4-1. Together these two objects characterize a single attack path, merging the knowledge that comes from separate sources – threat intelligence, offensive techniques repository and system facts. An attacker is represented as a tuple: *<attacker, attacker category>.*

A Threat Scenario is represented as a tuple *<impactful element, target, attack path, artifact category, artifact, technical impact category, attack type, attacker category>.* The first six components are system facts and describe the specific operational, architectural and supply chain features involved. The last three are *attack characteristics* that describe the nature of the attack. These tuples capture the full characterization of a *tailored* attack path and establish links to generic knowledge items.

These objects are linked directly to the rest of the risk claim structure, allowing them to serve as structured, inferable elements in risk reasoning. The **completeness claim** of attack enumeration depends on the selection of the attack characteristics and their alignment with the community's shared understanding of cyberattacks.

This is where knowledge repositories like CAPEC, STIX, MITRE ATT&CK and ontologies like MITRE D3FEND come into play.

Knowledge repositories provide critical building knowledge items related to offensive techniques. CAPEC provides a taxonomy of attack patterns based on intent and behavior. STIX defines a schema for sharing cyber threat intelligence across organizations. ATT&CK catalogs observed adversary tactics and techniques, providing a bridge from abstract models to operational realities.

However, these sources are often descriptive and not directly usable for inference. *Tailoring* is required: mapping abstract techniques to concrete system facts and the attack characteristics that drive attack enumeration.

This gap is partially addressed by MITRE D3FEND, which introduces a formal ontology of defense techniques and the digital artifacts they operate on—such as files, memory blocks, log entries, or network packets. Crucially, it also enables alignment with MITRE ATT&CK, forming a bidirectional mapping between offense and defense.

However, tailoring of digital artifacts remains a challenge without a strong foundation of system facts. This is where standards like OMG SPECTRA become crucial.

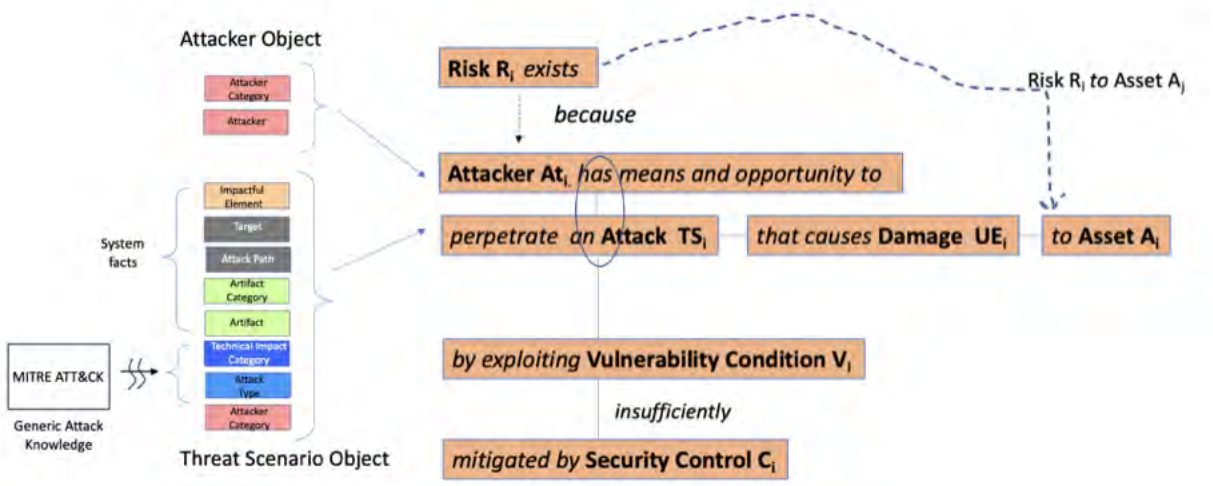


Figure 4-1: Contextualized attack objects.

For instance, consider the ATT&CK technique *Adversary-in-the-Middle*. On its own, this technique is abstract. But when mapped against a SPECTRA model that defines a system's information pathways, analysts can pinpoint which channels lack encryption or inadequate monitoring. That's where this technique becomes a tailored attack path, directly feeding into a risk claim for the system of interest.

To achieve a living, adaptive assurance case, Risk-Centric DevSecOps pipelines must be connected to the knowledge repositories, so that the knowledge items can flow from community-curated repositories all the way to system-specific data lake, triggering risk re-evaluation when a new attack technique is published.

This closes the loop between engineering intent, deployed artifacts, and evolving threat landscapes. Importantly, this also allows threat intelligence databases—like those encoded in D3FEND—to be leveraged not just for awareness but for assurance, processed by the automated pipelines of interoperable tools.

Looking forward, the interoperability between system models and attack ontologies must be strengthened. The current "digital thread" between the system of interest and the attack databases is still partially broken—due to inconsistent semantics, tool fragmentation, and lack of tailoring workflows at the mission and design stages. To mend this, the OMG roadmap for SPECTRA includes a formal ontology, developed in collaboration with MITRE D3FEND, to harmonize attack representations with system design data.

This SPECTRA ontology will define mappings between SPECTRA modeling constructs and the Digital Artifact types in D3FEND, enabling seamless, bidirectional reasoning. It will support early-stage modeling—at the mission and system engineering phases—allowing organizations to predict and mitigate risks before implementation begins.

# 5 VULNERABILITY CHARACTERIZATION IN RICK-CENTRIC DEVSECOPS

As the risk claim and risk measurement depends on systematically identifying vulnerabilities, how can we avoid being reactive and falling behind the attackers?

In contemporary cybersecurity practice, vulnerability knowledge is ubiquitous, but often superficial. Tools and databases offer long lists of known issues but rarely provide the system-specific context needed to prioritize or understand them as part of a broader assurance strategy. Within a risk-centric DevSecOps pipeline vulnerabilities must be treated not as isolated flaws, but as contextual relationships between contextualized attacks, defenses, and system-specific design choices.

At its core, a vulnerability is a situational condition—an *imbalance* when a tailored attack intersects with an insufficient or absent defense.

To support automation and continuous reasoning, vulnerability conditions must be formally characterized. Consider the objects depicted in Figure 5-1. Vulnerability Condition object is represented as a tuple *<target, vulnerability category>.* Allocated control object that is associated with the mitigation claim, is represented as a tuple *<target, control type, vulnerability category>.* Finally, the vulnerability finding object is represented as a tuple *<target, vulnerability category, vulnerability finding category, vulnerability finding>*.

These objects are linked directly to the rest of the risk claim structure, allowing them to serve as structured, inferable evidence within the assurance case. A vulnerability category is aligned with the attack characteristics *<technical impact category, attack type, attacker category>* and supply chain artifact characteristics *<artifact category, artifact>.* Thus, a vulnerability condition is a *profile of an attack* – representing the necessary conditions of attack's success.

The **completeness claim** of vulnerability condition enumeration depends heavily on the selection of *vulnerability category* and their aligned with the community's understanding of vulnerabilities and mitigating controls.

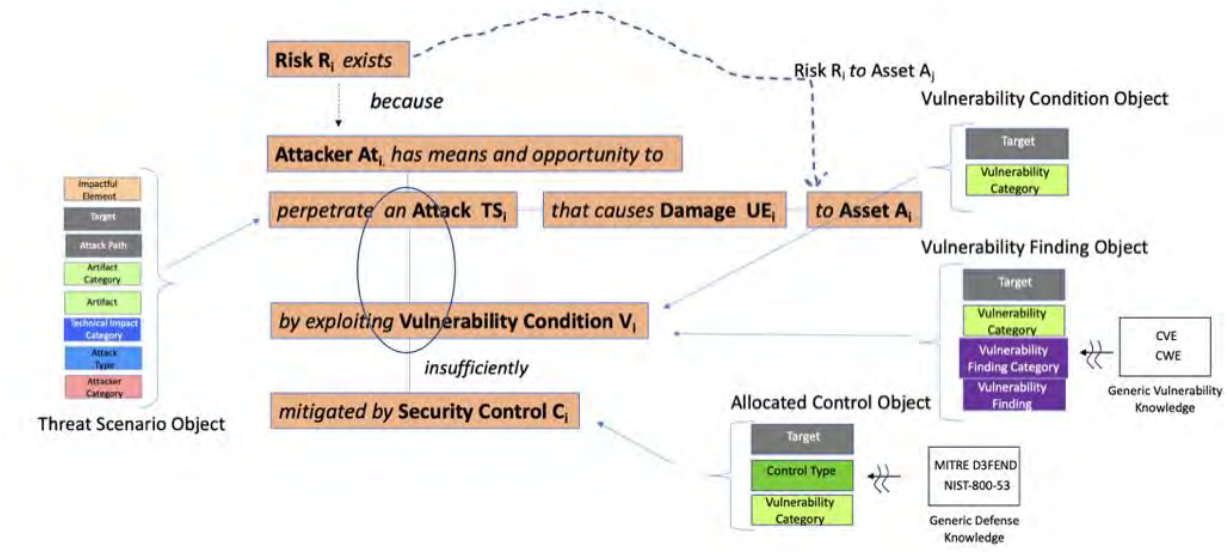


Figure 5-1: Contextualized vulnerability objects.

This is where generic vulnerability repositories play a key role:

- CVE (Common Vulnerabilities and Exposures): Cataloged flaws in known software components, useful for inventory-based scanning.
- CWE (Common Weakness Enumeration): A taxonomy of generalized coding flaws and design errors, used primarily by static analysis tools.

While foundational, these sources are limited in their ability to support tailoring. They are detection-oriented—not inference-ready—and lack direct alignment with system-specific models or mission goals. Furthermore, they are typically invoked late in the lifecycle, when options for architectural change are limited and remediations become costly. As knowledge items, generic vulnerabilities are unacceptably reactive.

The proposed framework is proactive, as vulnerability is characterized as a necessary condition of an attack, directly linking vulnerability enumeration to attack enumeration and control selection. To support a living, proactive assurance case, community-curated vulnerability knowledge must also flow seamlessly through the DevSecOps pipeline. For example, the publication of a new CVE should automatically trigger re-evaluation of associated vulnerability conditions, updating the risk argument and notifying stakeholders. This depends on a mended digital thread between vulnerability databases, system models, and the argument structure—a thread that is not yet fully realized.

In the meantime, findings from vulnerability scans—while reactive—can still be valuable. They can be linked as *evidence* to the appropriate Vulnerability Condition objects, strengthening mitigation claims and anchoring risk assertions in verifiable data.

As with attack characterization, the key to scalable automation is semantic structure. By aligning system-specific vulnerabilities with shared vocabularies and modeling them as part of a formal risk argument, organizations can move from patchwork detection to predictive, contextualized defense, at the speed and scale demanded by modern cyber environments.

# 6 SECURITY ARGUMENT AS THE ORCHESTRATOR IN RISK-CENTRIC DEVSECOPS

Having extended the DevSecOps pipelines to continuously inference risk subclaims, how can we guarantee that a change in the mission model or a threat intelligence update will result in delivering the right risk information to each stakeholder?

## 6.1 ARGUMENT-DRIVEN AUTOMATION OF CYBERSECURITY PIPELINES

The orchestration follows the structure of risk claim, outlined in Figure 6-1. It begins with collecting foundational mission, system facts—defined in SysML and enriched through the SPECTRA profile. From there, attacks are enumerated and tailored to the system model. Independently, assets are enumerated and tailored to the system, then undesired events are enumerated. Then risk clusters are assembled.

Security controls (e.g. from NIST 800-53) are added as subclaims and evaluated in terms of their ability to mitigate relevant attack paths. Vulnerability conditions are enumerated as profiles of attacks to proactively identify imbalances between feasible attacks and available defenses. The "wave" of inferences, triggered by an incremental change in the system model or the threat environment of the deployed system, culminates in updated risk measurements and re-calculating risk distributions.

What distinguishes this framework is that each step is driven by the argument itself. When the argument asserts that a particular interface exposes an asset, it triggers inference logic to build data objects, tailor generic knowledge, if necessary, tailor the associated subclaims and update the argument's state. The assurance case becomes an active reasoning loop—continuously acquiring, interpreting, and integrating knowledge.

## 6.2 EXAMPLE: ASSET CLAIMS AND PIPELINE ORCHESTRATION

In a structured risk argument, assets are not just data objects—they are subclaims at the base of the risk claim structure. As depicted in Figure 6-1, assets are represented as tuples of the form ⟨*asset category*, *impactful element*⟩—for example, *⟨information asset, datatype("07", "current position")⟩.*

Each identified asset $A_i$ is first represented by a basic **existence claim**: *"Asset $A_i$ exists within the system of interest."* This claim is further supported by a **traceability claim**: *"Asset $A_i$ is traceably derived from the system or mission model."* This ensures that the asset is not arbitrarily declared but is anchored in engineering artifacts, providing defensibility and model consistency. However, for this to support a valid risk model, a second layer of justification is needed as a **viability**

**claim**: *"Asset $A_i$ is viable"—that is, it performs a mission-relevant function or contributes to a critical operation, making it worth protecting.*

Beyond individual assets, the argument must justify **completeness**: that the union of all identified assets represents **all viable assets** within the system's operational and mission context. This claim is essential, as risk reasoning hinges on understanding what is at stake. If critical assets are omitted, derived risks may be incomplete or misleading. The justification for this completeness claim typically draws on system and mission models (e.g. via SPECTRA), mapping system units and data types to operational goals and verifying their inclusion through traceability.

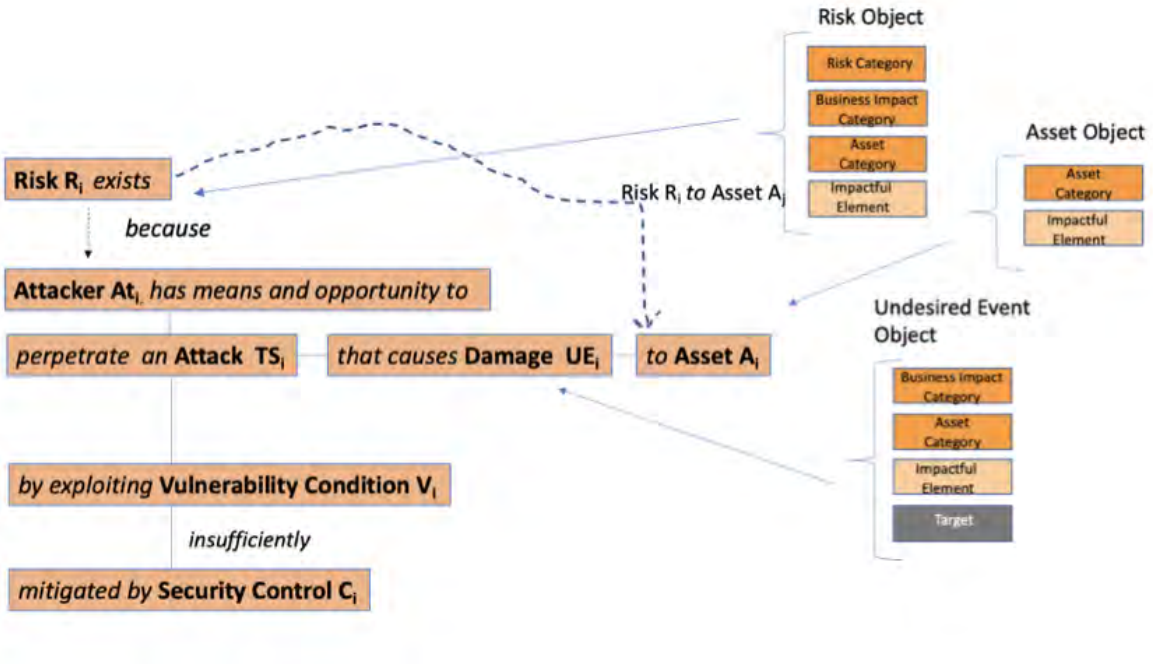


Figure 6-1: Contextualized risk and asset objects.

In a risk-centric DevSecOps framework, the assurance case does more than structure justification—it orchestrates the construction of the risk model itself. While the top-level claim is that *all viable assets have been identified* (i.e. the completeness claim), this cannot be evaluated in isolation. It must be operationalized through a **synergy** of claim templates, a risk metamodel, and inference rules. As the controller executes, it dynamically instantiates claims such as existence and viability, generating asset objects and their corresponding justifications based on foundational system facts.

The argument for completeness involves two subclaims: first, that the set of asset categories is *exhaustive* (e.g. all information, capability, and mission assets are covered); and second, that the input system facts are themselves *complete* with respect to the system of interest. Because inference rules are aligned with asset categories, executing the controller ensures that all category-relevant elements are explored. In this way, the assurance case becomes *a self-tailoring*

*mechanism*, merging structural reasoning with dynamic knowledge inference to justify the very model it constructs.

### 6.3 Assurance Case: From Documentation to Intelligent Control

Ultimately, this framework repositions the assurance argument as a computational structure that orchestrates cybersecurity automation. Each component of the pipeline—from model ingestion to threat tailoring, from vulnerability identification to control validation—is no longer an isolated task but a logically determined operation within the unfolding argument. The argument itself ensures that each step is justified, traceable, and mission-relevant.

By elevating the assurance case to this role, the framework not only bridges the gaps in today's fragmented cybersecurity practice but offers a vision for a future in which cybersecurity becomes computable—where reasoning about attackers, systems, and mitigations is no longer manual, reactive, or siloed, but continuous, integrated, semantically precise, and happening at the speed of need.

### 6.4 A Standards-Enabled Assurance Ecosystem

The success of this orchestration depends on the tight integration of interoperability standards across the domains of software, systems engineering, and cybersecurity. SPECTRA ensures that system and mission models are semantically accessible. The emerging OMG System Assurance Task Force standards Automated Risk Claims for Cyber (ARC-C) and Automated Risk Measurements for Cyber (ARM-C) define the risk metamodel. OMG Structure Assurance Case Metamodel defines assurance cases. D3FEND, in collaboration with SPECTRA's emerging ontology, completes the picture by enabling defense techniques to be mapped precisely to digital artifacts and attacker behaviors.

Together, these standards support a fully integrated reasoning infrastructure, where the assurance argument is not simply a record of confidence but the *semantic controller* of a living, adaptive security ecosystem.

## 7 Existing Approaches to Security Assurance Cases

While cybersecurity assurance is often framed as vulnerability detection or compliance certification, there is growing recognition of the need for more structured, semantically grounded approaches to expressing and reasoning about system security. Security assurance cases—structured arguments supported by evidence—are increasingly seen as essential for demonstrating confidence in the security of complex, evolving systems. Several standards, methods, and frameworks offer partial guidance on how to construct and maintain such arguments, though none yet support the fully automated, orchestrated assurance envisioned in risk-centric DevSecOps.

Assurance cases that focus on risk modeling were introduced in Nikolai Mansourov, Djenana Campara's work [9]. Several technical approaches to assurance cases have emerged across different communities as described in: Mazen Mohamed, et. al. [11], Sarker, et.al. [12], and Ardebili, et.al. [13].

Formal methods have contributed rigorous verification techniques (e.g. theorem proving, model checking) to assure correctness in critical systems, but these approaches remain difficult to scale and integrate with broader lifecycle assurance, see Kulik, et.al. [14]. System-level assurance frameworks similarly provide structured methodologies for evaluating system security but often lack the adaptability to handle modern threats and continuously evolving systems, see Akmir Shulka, et.al. [10].

From a regulatory perspective, ISO/SAE 21434 [4] requires structured cybersecurity assurance cases across the automotive lifecycle. This marks a broader trend: organizations must increasingly demonstrate—not just declare—the adequacy of their defenses using structured, justifiable arguments. Though ISO/SAE 21434 is domain-specific, its principles are widely applicable.

# 8 USE CASES AND APPLICATIONS OF THE FRAMEWORK

A major application of this framework lies in transforming cybersecurity into a continuous process of test, evaluation, and assurance. Traditional approaches to cybersecurity—whether focused on vulnerability scans, checklist compliance, or point-in-time risk assessments—are increasingly misaligned with the pace and complexity of modern system delivery.

## 8.1 ENABLING CONTINUOUS TEST AND EVALUATION

In agile and DevSecOps environments, systems are developed, updated, and deployed on rapid timelines. Yet test and evaluation (T&E)—including cybersecurity assessments—remains one of the slowest and most manual phases of the lifecycle. Risk-centric DevSecOps changes this by integrating automated, model-based reasoning into every pipeline stage.

System models, risk claims, threat intelligence, and runtime telemetry are all structured into a living cybersecurity argument, where each subclaim (e.g. "*this control mitigates this attack on this node*") is traceable to test artifacts and verification outcomes. When a change occurs—such as a software update or a newly disclosed vulnerability—the argument identifies which claims are affected and automatically triggers the relevant tests. This enables incremental re-evaluation rather than full regression testing, reducing cost and time while maintaining assurance.

In this way, T&E becomes an ongoing, explainable, and mission-aligned process. Test results don't just pass or fail requirements—they resolve formal claims in a continuously evolving security assurance case.

### 8.2 SUPPORTING CONTINUOUS AUTHORIZATION

This same infrastructure enables *continuous authorization*, where operational security posture is evaluated in real time, not through static milestones. Traditional Authorization to Operate (ATO) processes (e.g. NIST SP 800-37) [6] are designed for relatively stable systems and rely on slow, manually curated risk documentation. As a result, they often lag behind system evolution or miss emergent threats.

With risk-centric DevSecOps, security becomes part of the pipeline. Each deployment carries with it an up-to-date assurance argument, grounded in real system data and validated controls. Only the affected claims are re-evaluated when updates occur, dramatically reducing overhead while improving precision and responsiveness.

This model aligns with ongoing efforts in the U.S. Department of Defense—such as the Continuous ATO (cATO) initiative—and reflects a growing need for security governance that evolves with the system itself.

## 9 CONCLUSION

As modern digital and cyber-physical systems grow more interconnected, complex, and mission-critical, traditional cybersecurity approaches—static assessments, reactive scans, and manual risk assessments—no longer suffice. What's needed is a shift: from fragmented controls to an integrated, continuously evolving cybersecurity practice grounded in formal reasoning, automation, and semantic traceability.

This paper introduced a framework for Risk-Centric DevSecOps, in which cybersecurity is reconceived as a continuous, knowledge-driven process. At its core is the formal cybersecurity argument—not a compliance artifact, but an intelligent controller that orchestrates tools, data, and decisions. Through structured claims and traceable evidence, the framework connects system models, threat intelligence, vulnerabilities, and mitigations into a cohesive risk picture—adapted to both system context and mission priorities.

The framework enables automation across asynchronous cybersecurity cycles: engineering, defense, governance, and threat modeling. System facts, attack characterizations, vulnerability descriptions, and controls flow through semantically aligned pipelines. Risk claims become the organizing principle, driving automated risk enumeration, prioritization, and remediation.

A central innovation is the use of the **assurance case as a controller**. Tools are no longer invoked in isolation—they are triggered to resolve specific claims. This enables real-time assurance: as systems evolve, only the impacted argument branches are re-evaluated, supporting continuous authorization and agile governance.

This vision is made possible by a family of interoperability standards developed by the OMG. Together, these standards enable a future where commercial tools can interoperate across the

lifecycle—engineering, risk analysis, assurance, and compliance—supporting scalable, explainable cybersecurity.

In short, this framework elevates cybersecurity to a computable, reasoning-based discipline. It empowers stakeholders to align system evolution with mission assurance, creating resilient, transparent, and adaptable digital infrastructure in the face of ever-changing threats.

# 10 REFERENCES

[1] NIST 800-30 Guide for Conducting Risk Assessments

[2] ETSI CYBER; Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA).

[3] ISO/IEC 27005:2022 Information security, cybersecurity and privacy protection — Guidance on managing information security risks

[4] ISO/SAE 21434 Road Vehicles Cybersecurity Engineering

[5] RTCA DO-356A Airworthiness Security Method and Considerations

[6] NIST 800-37 Risk Management Framework

[7] Sanford Friedenthal, Alan Moore, Rick Steiner, A Practical Guide to SysML, Morgan-Kaufmann, OMG Press

[8] Unified Architecture Framework (UAF), ISO/IEC 19540-1:2022 and ISO/IEC 19540-2:2022

[9] Nikolai Mansourov, Djenana Campara, Systems assurance: Beyond detecting vulnerabilities, 2010, Morgan Kaufmann, Elsevier, OMG Press

[10] Aknur Shukla, et. al. System Security Assurance: A Systematic Literature Review, Computer Science Review, 2022

[11] Mazen Mohamad, et. al. CASCADE: An Asset-driven Approach to Build Security Assurance Cases for Automotive Systems

[12] Sarker, et.al, Cybersecurity data science: an overview from machine learning perspective, Journal of Big Data, 2020

[13] Ardebili, et. al, Risk Assessment for Cyber Resilience of Critical Infrastructures: Methods, Governance, and Standards, Applied Sciences, Special Issue New Advances in Computer Security and Cybersecurity, 2024

[14] Kulik, et. al, A Survey of Practical Formal Method for Security, Formal Aspect of Computing, vol 34,1, 2022, ISSN 0934-5043

## 11 ACKNOWLEDGEMENTS

The views expressed in the *OMG Journal of Innovation* are the author's views and do not necessarily represent the views of their respective employers nor those of the Object Management Group® (OMG®).



- Return to *OMG Journal of Innovation landing page* for more articles and past editions.