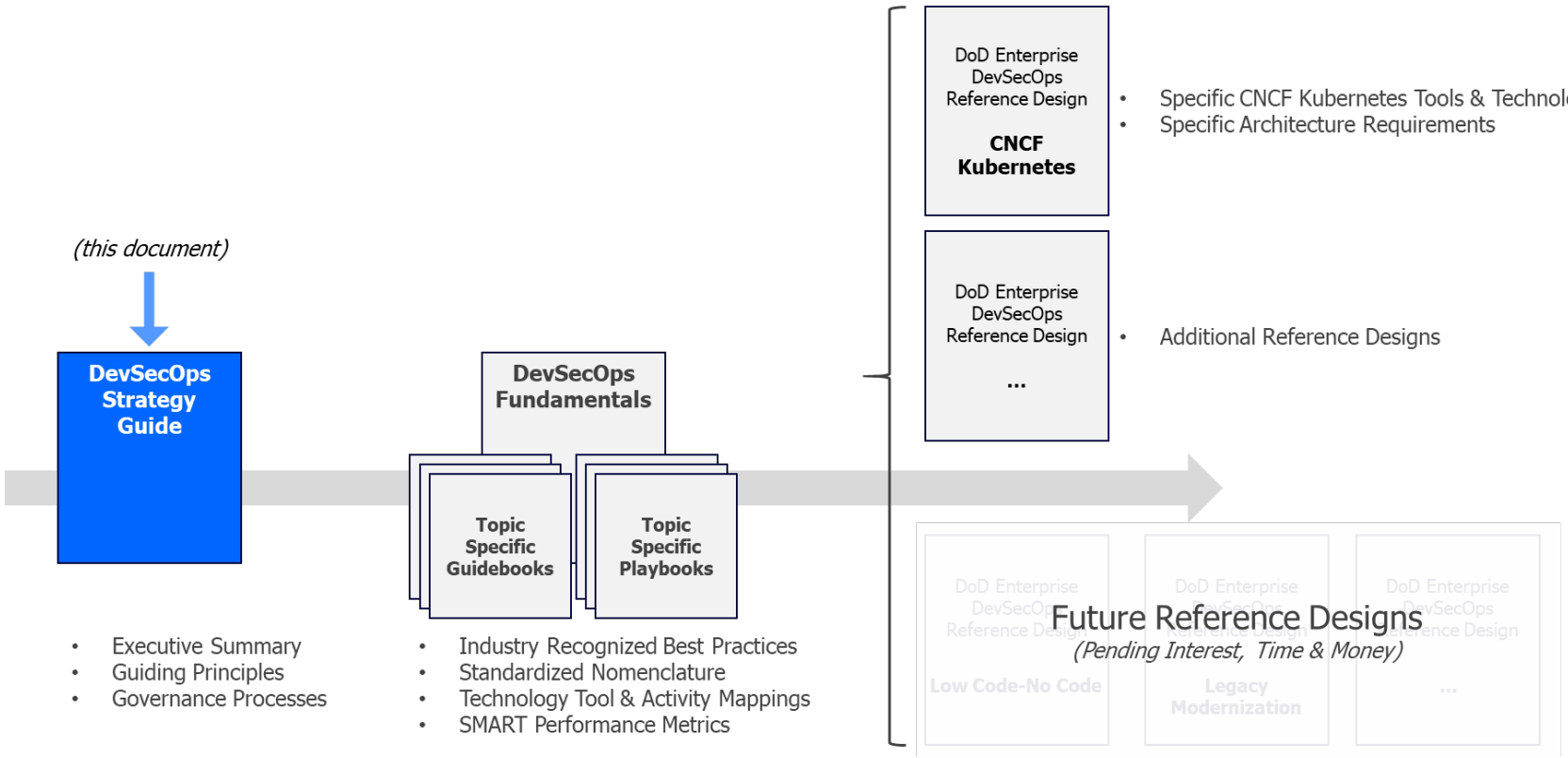
DoD Enterprise DevSecOps Strategy Guide

September 2021

Version 2.1

.

Document Set Reference:



Document Approvals

Approved by:

_____

John B. Sherman

Chief Information Officer of the Department of Defense (Acting)

Approved by:

_____

Stacy A. Cummings

Under Secretary of Defense for Acquisition and Sustainment (Acting)

Contents

Figures

1 Executive Summary

Many programs and missions across the Department of Defense (DoD) lack software development practices that meet industry standards for agility. The majority of current cybersecurity frameworks (NIST Cybersecurity Framework, ODNI Cyber Threat Framework, NSA/CSS Technical Cyber Threat Framework v2 (NTCTF), MITRE ATT&CK, etc.) focus predominately on post-production deployment attack surfaces. Furthermore, every release cycle is perceived as an uphill battle between development teams that attest to functionality, operational test and evaluation teams trying to confirm specific functionality, operations teams struggling to install and operate the product, and security teams bolting on protection mechanisms as an afterthought. To *deliver resilient software capability at the speed of relevance* the department needs to implement strategies that focus on cybersecurity and survivability across the development process. The DoD isn't alone in this journey; industry has already minimized deployment friction through a cultural shift to DevSecOps (development, security, and operations).

The DoD CIO and the Office of the Under Secretary of Defense for Acquisition and Sustainment (OUSD A&S) recognize the urgent need to rethink our software development practices and culture by leveraging the commercial sector for new approaches and best practices. DevSecOps is such a best practice as it enables the delivery of resilient software capability at the speed of relevance, a central theme of software modernization across the DoD. DevSecOps is a proven approach widely adopted by commercial industry and successfully implemented across multiple DoD pathfinders. DevSecOps is a core tenant of software modernization, technology transformation, and advancing an organization's software development ecosystem to be more resilient, while ensuring cybersecurity and metrics/feedback are paramount.

The DevSecOps software lifecycle approach creates cross-functional teams that unify historically disparate evolutions – development (Dev), cybersecurity (Sec), and operations (Ops). As a unified team they follow Agile principles and embrace a culture that recognizes resilient software is only possible at the intersection of quality, stability, and security, as depicted in *Figure 1*.

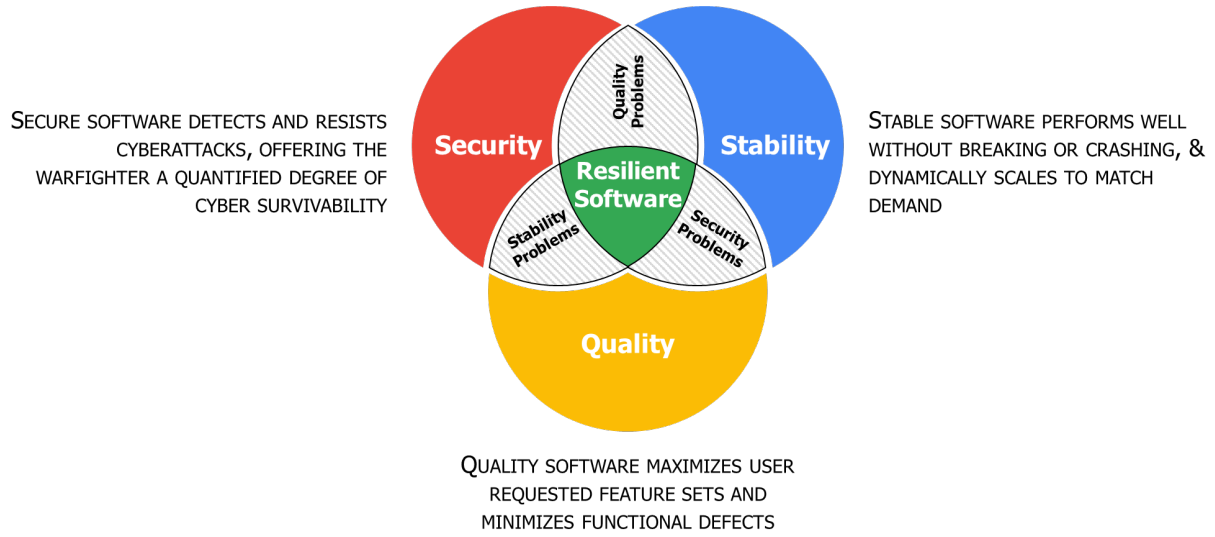


Figure 1 Pillars to Achieve Resilient Software Capabilities

The benefits of adopting DevSecOps include:

- **Reduced mean-time to production**: Reduces the average time it takes from when new software features are required until they are running in production;
- **Increased deployment frequency**: Increases how often a new release can be deployed into the production environment;
- **Decreased mean-time to recovery:** Decreases the average time it takes to identify and resolve an issue after a production deployment;
- **Decreased change-fail rate:** Decreases the probability that a new feature delivered in production will result in a failure in operations;
- **Fully automated risk management**: Well defined control gates perform risk characterization, monitoring, and mitigation as artifacts are released and promoted through every step, from ideation through production;
- **Baked-in Cybersecurity:** Software updates and patches delivered *at the speed of relevance*.

The DoD Enterprise DevSecOps Strategy, along with its supporting document set, provides education, best practices, and implementation and operational guidance to Information Technology (IT) capability providers, IT capability consumers, application teams, and Authorizing Officials.

2 Document Set Structure

The momentum and interest in DevSecOps continues to rapidly expand across the DoD and the Defense Industrial Base (DIB). Early adopters of DevSecOps at the DoD have matured to proven practitioners; the resulting wave of fast followers has created a set of practitioners operating at an intermediate skill set level, and as new programs explore adopting DevSecOps, these novice practitioners are looking for guidance, direction, terminology clarification, and best practices. This expanding ecosystem justifies the shift from a single document to a document set, as depicted below in *Figure 2*. A document set approach better supports novice, intermediate, and expert practitioners concurrently by enabling them to quickly find the material they seek to include: a primer on DevSecOps as a strategy, access to fundamental concepts and succinct explanations of the DevSecOps lifecycle, and/or specific reference design guidance with deep, technical content.

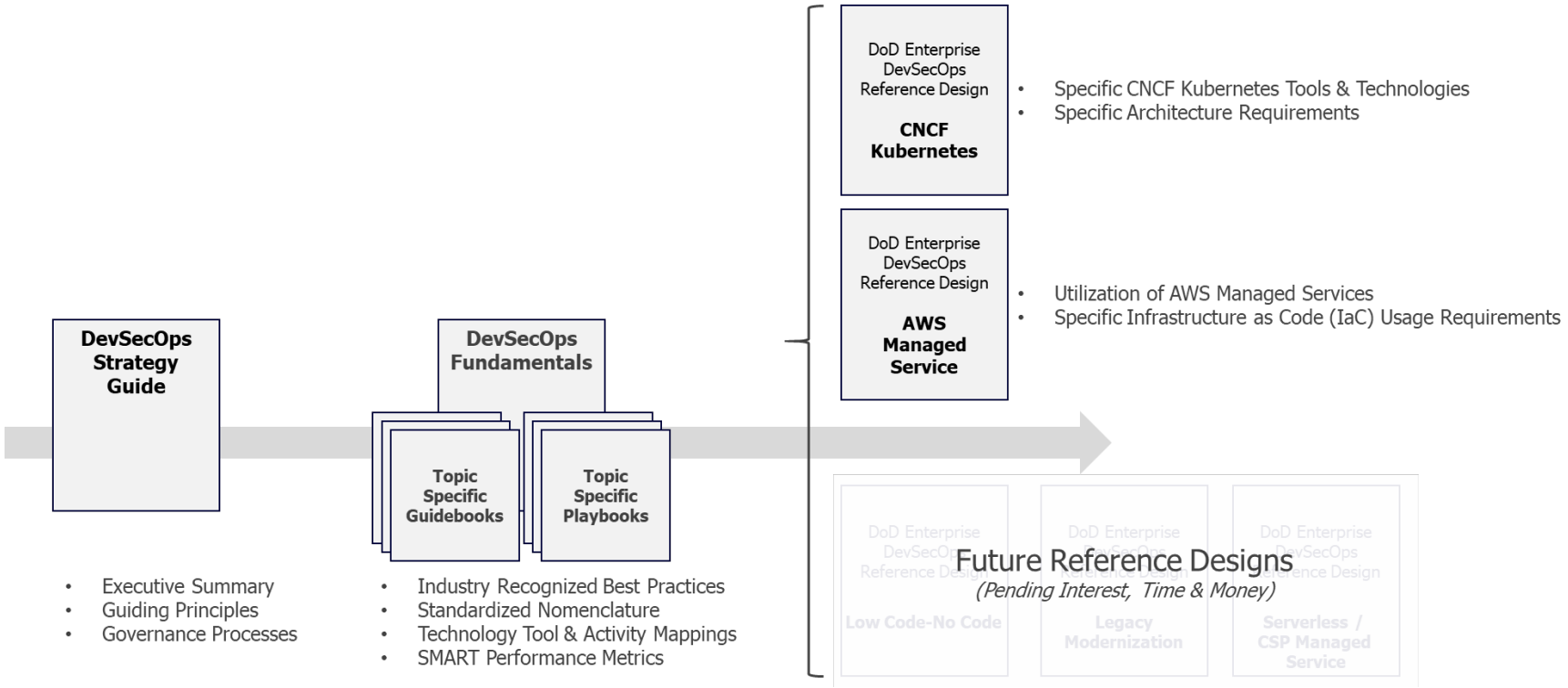


Figure 2 DevSecOps Document Set Overview

2.1 DevSecOps Strategy Guide Document

The **DevSecOps Strategy Guide** (this document) provides an executive summary of DevSecOps as a whole by establishing a set of strategic guiding principles that every approved DoD enterprise-wide DevSecOps reference design must support. This document is generally consumed by PEOs and anyone in non-technical leadership positions.

The strategy guide advocates for a *versioned* DevSecOps governance process, including a more rigorous and evolving type of Authorization to Operate (ATO) known as **Continuous Authorization to Operate (cATO)**. cATO is predicated upon the cyber survivability posture across the *entire* software supply chain and is driven by real-time metrics gathered at every step, compared to the current method which conducts a snapshot in time view once every three years to authorize networks. The Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study emphasized the fact that *software is never done*.[1] An implied corollary to this statement is *cyberspace adversaries never quit*. The actions taken to achieve a level of cyber survivability today may be insufficient tomorrow, justifying the need for DevSecOps Reference Designs linked to a specific version of cATO in order to avoid stale processes that could result in exposure, or worse, compromise.

2.2 DevSecOps Fundamentals Document

The **DevSecOps Fundamentals,** including associated topic-specific guidebooks and playbooks, establishes consistent nomenclature, a curated *and versioned* technology map, and explores a series of Specific, Measurable, Achievable, Relevant, and Timely (SMART) performance metrics used to manage and monitor a DevSecOps CI/CD pipeline. **Guidebooks** are intended to provide deep knowledge and industry best practices with respect to a specific topic area. **Playbooks** consist of one-page *plays*, structured to consist of a best practice introduction, salient points, and finally a checklist or call-to-action. The Fundamentals document, and its associated guidebooks and playbooks, are generally consumed by DoD enterprise platform providers and specific DoD organization DevSecOps teams that manage (instantiate and maintain) a specific DevSecOps software factory implementation.

2.3 DevSecOps Reference Design Document(s)

DevSecOps Reference Design documents define *specific* tools, technologies, pipeline constructs, and architectures. A reference design is independently versioned, and must be built atop of the Fundamentals document. It should also build from and reference the various guidebooks and playbooks. The intention is to create a composable reference design capacity that eliminates redundant or inconsistent definitions. Reference designs are then empowered to drive continuous improvement and adopting an industry best practice of an “n – 1” support lifecycle, ensuring programs continuously modernize and avoid stagnant solutions. The DevSecOps Reference Design documents are generally consumed by DoD program application teams that develop, secure, and operate mission applications.

[1] Defense Innovation Board (DIB), “Software Acquisition and Practices (SWAP) Study.” May 03, 2019, [Online]. Available: https://innovation.defense.gov/software.

3 Assumptions

This document set makes the following assumptions:

- For organizations deploying new business solutions or modernizing existing software systems, deploying to an approved or provisionally authorized (PA) cloud environment will become their preferred solution technically and culturally. For weapons systems, the environment will likely continue to be on premise to facilitate hardware-in-to-loop (HWIL) testing with embedded systems.
- Rapidly changing technology dictates designing the DevSecOps pipelines and patterns for flexibility as new development capabilities enter/exit the commercial product market.
- Cybersecurity elements will leverage cloud service provider (CSP) managed service capabilities where practicable. Teams will aggressively seek to integrate automated feedback, patching, alerting and other authorized network security measures.
- The DoD Enterprise DevSecOps reference designs aspire to adopt and leverage industry best practices and standards. Each reference design must name the specific technologies as an addendum to the capabilities that power its software factory pipelines. Reference designs are expected to iterate more frequently than the DevSecOps Strategy Guide, encouraging rapid innovation at a pace closer to industry. They also must provide specifics around technical capabilities and specific technology products that power the design.
- Each DoD Enterprise DevSecOps reference design will clearly assert compliance to a *specific version* of the DevSecOps Tools and Activities Guide, and *approved* or *provisionally authorized* reference designs must assert compliance using the de facto software industry standard of *n-1* for the major version. This enables common best practices and the tooling map to evolve over time within the DevSecOps Fundamentals document and prevents a reference document from inadvertently becoming stale and at greater risk for exposure or compromise.
- The DoD must acknowledge a lock-in posture; recognizing vendor lock-in, *and* recognizing product, version, architecture, platform, skills, legal, and mental lock-in also exist.[2] Avoiding vendor lock-in without considering other types of lock-in is ill-advised. Finally, nothing is more dangerous than mental lock-in.
- The DevSecOps strategy must have the capability to scale to any type of operational requirement needing software capabilities, including:
    - Business systems
    - Command and Control systems
    - Embedded and Weapon systems
    - Intelligence Analysis systems

[2] M. Flower, “Don’t get locked up into avoiding lock-in,” [Online]. Available: https://martinfowler.com/articles/oss-lockin.html [Accessed 8 February 2021].

- Autonomous systems
- Human-Machine Collaboration systems
- Artificial Intelligence / Machine Learning systems
- Cybersecurity defensive and offensive systems

The cultural principles espoused by this strategy and within the DevSecOps Fundamentals document are universally and equally applicable to every DoD Enterprise DevSecOps reference design.

4 DevSecOps Defined

DevSecOps describes an organization's cultural and technical practices, aligning them in such a way to enable the organization to reduce the gaps between a software developer team, a security team, and an operations team. Adoption improves processes through daily collaboration, agile workflows, and a continuous series of feedback loops. *Figure 3* visually depicts DevSecOps distinct phases and philosophies, the specifics of which are elaborated upon in the *DevSecOps Fundamentals* document. **NOTE:** The cybersecurity activities in the outer rim are notional and do not represent a complete set of activities undertaken by a team due to spacing limitations.

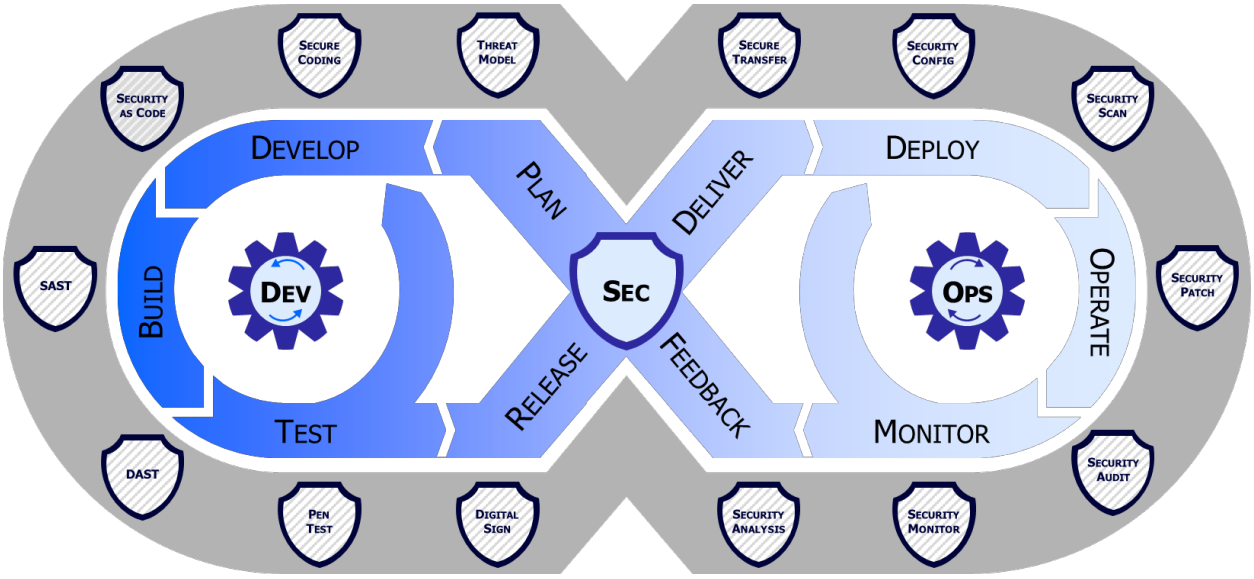


Figure 3 DevSecOps Distinct Lifecycle Phases and Philosophies

Pioneering programs using DevSecOps for several years have concretely demonstrated that its adoption can deliver resilient software capability at the speed of relevance; and by integrating cybersecurity at every step, as depicted in *Figure 3*, the cyber survivability of the artifacts and applications produced is enhanced. DevSecOps strives for faster and more secure software delivery while achieving consistent governance and control.

The document set construct acknowledges that there is no uniform set of DevSecOps practices or tooling. Each DoD organization is expected to tailor its culture and align its DevSecOps practices to its own unique processes, products, security requirements, and operational procedures. **DevSecOps platforms and their underlying software factories are expensive, and every DoD organization is encouraged to seek out an existing Reference Design platform and leverage the cATO that comes with it.** Embracing DevSecOps requires organizations to shift their culture, evolve existing processes, adopt new technologies, and strengthen governance.

5 Formal Recognition of the Software Supply Chain

The software supply chain is a logistical pathway that covers the entirety of **all** hardware, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), technology force multipliers, tools and practices that are brought together to deliver specific software capabilities. A notional software supply chain is depicted in *Figure 4*.

Software supply chains exist for business systems, weapon systems, and everywhere software is developed and deployed. It is easy, but naïve and incorrect, to dismiss an embedded system in a projectile as "isolated" and disconnected. The projectile includes embedded software that was compiled, including relying on 3rd party libraries and linking to hardware drivers, and relies upon features of embedded firmware. Additionally, the very performance of the software was likely tested using model and simulation software executing within a High Performance Computing (HPC) Cloud.

DevSecOps *philosophies* span multiple links of the Software Supply Chain. DevSecOps cannot exist without this logistical supply chain – Integrated Development Environments (IDEs), build tools, code repositories, artifact repositories, testing software suites, and many others pieces of software must work together in unison to effectively execute a DevSecOps powered software factory. The totality of these environments must be considered when evaluating the software supply chain.

The cybersecurity and risk postures of a specific artifact or application, illustratively, would be calculated using the product rule across the entirety of the software supply chain. If the compiler is 90% secure, the code repository is 90% secure, the artifact repository is 90% secure, and the container orchestrator is 90% secure – the overall system is **not** 90% secure. The cybersecurity level of the end-to-end ecosystem is actually .9 * .9 *.9 * .9, or roughly 65% secure. **This is why the creation of hardened DoD Enterprise DevSecOps Reference Designs are so critical**; DevSecOps aims to harness the collective expertise and knowledge across the *entire* software supply chain to mitigate risk at each step. Only then can the overall cyber survivability of the ecosystem significantly increase. To further illustrate this point, if a DevSecOps team only increases security 5%, raising each level from 90% to 95%, then overall cyber survivability jumps from 65% to 81%.

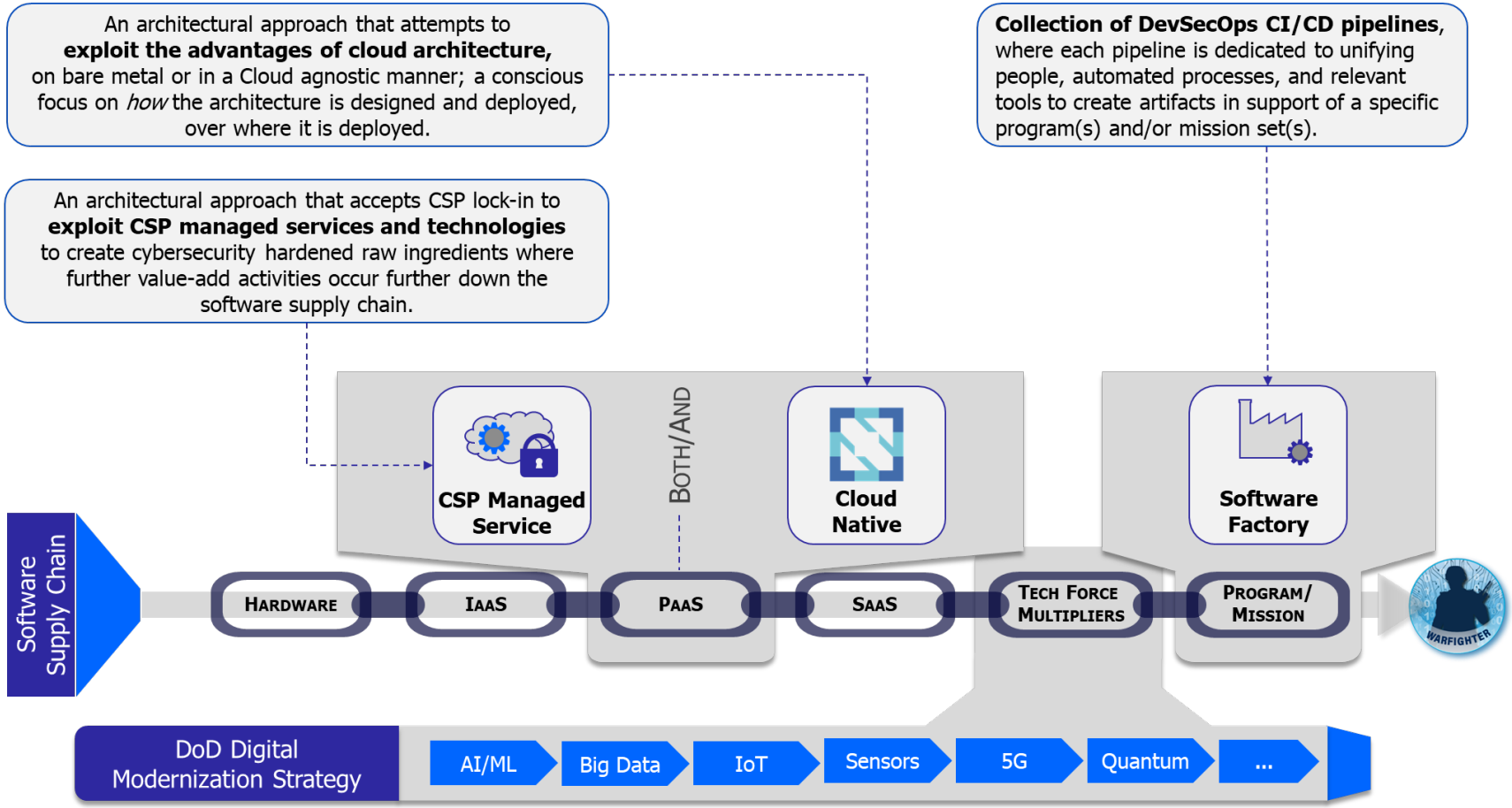


Figure 4 Notional Software Supply Chain

6 Construction of Software Factories

Software factories, like the normative example illustrated in *Figure 5* Normative Software Factory Construct, are strongly linked to a specific software supply chain. As in physical factories, software factories may contain multiple assembly lines, or in software parlance, *pipelines*. Each pipeline contains and defines a complete set of tools, process workflows, scripts, and environments that co-exist to produce a set of production quality software artifacts. A software supply chain "has a" software factory, but the software factory itself is not an entire software supply chain.

As in physical factories, automation is a central theme. Software factories employ automation at multiple levels and across multiple activities in the *develop, build, test,* and *release and deliver* phases of the DevSecOps lifecycle. Each environment in the process is automated to the maximum extent that is safely and securely possible, rehydrated using **Infrastructure as Code (IaC)** and **Configuration as Code (CaC)** that run on various tools. A software factory is inherently designed for multi-tenancy and can automate software production for multiple projects.

DoD needs multiple software factories tuned for specific types of software systems, such as web applications or embedded systems that may include significant amounts of hardware in the loop (HWIL) for automated testing. It also requires software factories operating at different information Impact Levels (IL), from IL-2 through IL-6, and above.[3] Under the shift to a cATO, each software factory will have its processes, teams, and storage reviewed and certified to feed into a continuous monitoring system. Additionally, this shift greatly lessens the initial burden of achieving an ATO for each piece of software, as the process and roll out is certified and feeds into a continuous monitoring architecture.

The ingestion of artifacts ("*raw ingredients*") from external systems and the subsequent promotion of value-add artifacts created within the software factory to downstream consumers requires additional coordination, much of it automated. Software factories and their operations are covered in-depth in the DevSecOps Fundamentals document.

[3] DISA, "Department of Defense Cloud Computing Security Requirements Guide, v1r3," Mar 6, 2017

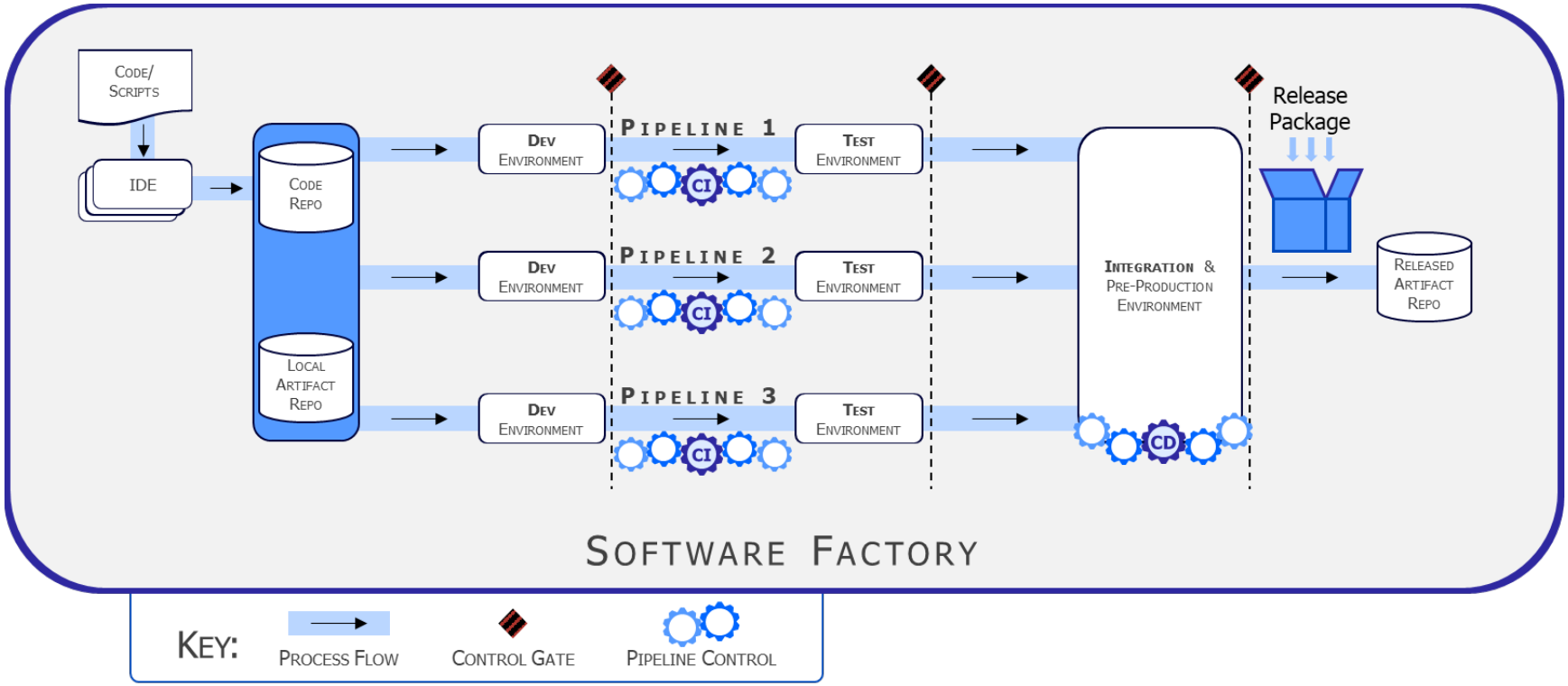


Figure 5 Normative Software Factory Construct

7 DevSecOps Guiding Principles

The adoption of DevSecOps, the Software Factories that drive these ecosystems, and cATO combine to represent a seismic strategic shift in the way DoD procures and delivers resilient software capability at the speed of relevance. This strategy is captured in the following set of DevSecOps guiding principles, which are abstract when considered in isolation, but create guardrails for DevSecOps teams creating and/or working on top of a specific DoD Enterprise DevSecOps Reference Design. Each of these guiding principles will be unpacked in the sections that follow to unequivocally explain both the intention and the expectation. The DevSecOps Guiding Principles are depicted in Table 1.

• **Relentless pursuit of Agile principles and culture within a software factory construct**
• **Software factories mandate baked-in security via integral and comprehensive security practices across the entirety of the software supply chain leveraging Zero Trust (ZT) and behavior detection principles.**
• **Integrated, automated & continuous end-to-end testing and monitoring, from ideation through production, with clearly defined control gates for release candidate promotion**
• **Immutability of infrastructure achieved via "*x as Code*" design patterns**
• **Adoption of Cloud-smart and data-smart architectural motifs throughout**

Table 1 DevSecOps Guiding Principles

These guiding principles represent the starting point for establishing common nomenclature and a curated and versioned approach to DevSecOps adoption. The DevSecOps Fundamentals document builds on these guiding principles by formalizing each phase of the DevSecOps lifecycle. The DevSecOps Tools and Activities Guidebook defines the activities individuals perform on a daily basis when part of a DevSecOps team, and the required and preferred types of tools required to be considered a DevSecOps team. Further, each DevSecOps Reference Design builds upon the principles and practices though a layer of specificity covering tool and processes, addressing technology specific interconnects, and adding additional required and preferred tools and activities that a team must adopt. When principles, practices, and tools combine properly, the result is an efficient, transparent, and harmonized software factory that is capable of delivering new features at the speed of operational relevance, while maintaining the level of security required to operate in national security environments.

7.1 Relentless pursuit of Agile

The Agile Manifesto captures core competencies that define functional relationships that every DevSecOps team should value:[4]

- Individuals and Interactions over Processes and Tools

[4] Beck, K. *et. al.*, 2001. Manifesto for Agile Software Development. [Online]. Available at: https://agilemanifesto.org.

- Working Software over Comprehensive Documentation
- Customer Collaboration over Contract Negotiations
- Responding to Change over Following a Plan

The first core competency emphasizes the value and importance that individuals work together, but it should not be interpreted that processes and tools are irrelevant. The same holds true for the other core competencies; documentation is still needed, but not at the cost of working software; Agile teams still create sprint plans, etc.

The manifesto further defines 12 Principles of Agile Software, offering an additional reinforcement that teams must prioritize customer engagement.[5]

The software factory construct is wide-reaching, spanning across acquisition, engineering, testing, cybersecurity, operations, and even leadership. The adoption of Agile principles and the shift to DevSecOps can be challenging for many organizations. Leaders want to embrace DevSecOps but are ill prepared to teach it across their organization and even less prepared to measure its implementation. Acquisition professionals struggle to understand how to contract DevSecOps because it is hard to put tangible metrics and a price tag on something largely viewed as a set of conceptual principles. Uncertainty inherently creates fear and generates bias. Adoption is always easier when experiential knowledge from similar situations can be identified and applied. Without this experiential knowledge, bias is unconsciously applied in both comprehension and decision making. Recognizing these biases is the first step of cultural change.

The DoD has statutory obligations to evaluate if a given investment produced value based on the combination of time, resources, and money spent. A study entitled “The Psychology of Sunk Cost” was conducted and published in the 1985 issue of *Organization Behavior and Human Decisions Processes*.[6] This study reveals a bias where we tend to commit to something based on a perception that the reward *must* be great enough given the investment made. This is why we chase after a good sale despite the fact that it may be a 3-hour drive and the cost of gas alone outstrips the savings. Recognition of the sunk cost fallacy is vitally important in adopting and in executing within a DevSecOps culture.

7.2 Software factories mandate baked-in security

In the DoD, security accreditation alone has proven a long and red-tape laden process. Software factories mandate staunch cyber survivability postures via integral and comprehensive security practices across the entirety of the software supply chain. DevSecOps weaves cybersecurity practices throughout each lifecycle phase, shifting cybersecurity practices to the left, advancing ZT architectures, recognizing the value gained from highly-automated unit, functional, integration, and security testing. This is a key DevSecOps differentiator; functional *and* security capabilities are tested and built simultaneously, with a series of recognized control gates that aim to prevent defect escapes and enhance the cyber survivability of the software artifact before release into the next environment. This approach also bakes in metrics that can

[5] Beck, K. *et. al.*, 2001. Manifesto for Agile Software Development. [Online]. Available at: https://agilemanifesto.org/principles.html.
[6] Arkes, Hal R. & Blumer, Catherine, 1985. "The psychology of sunk cost," Organizational Behavior and Human Decision Processes, Elsevier, vol. 35(1), pages 124-140, February.

be passed on to cyber defenders, enabling for better monitoring and more targeted feedback returning to engineers for future updates and patches.

7.3 Integrated, automated & continuous end-to-end testing and monitoring

The shift towards Continuous Authorization to Operate (cATO) stipulates continuous testing and monitoring that shifts the risk assessment further left to evaluate the *people*, *platform*, and *processes* using real-time data-driven metrics. Each program must build and implement a unique integrated testing and control gate decisional process in partnership with their AO. As a principle, each of the phases of DevSecOps contribute in their own unique way to the real-time performance metrics that form the cornerstone of cATO. Continuous monitoring is also required for ZT effectiveness, as defined in NIST SP 800-27.[7]

7.4 Immutability of infrastructure achieved via "*x* as Code" design patterns

The shift to immutable infrastructure using Infrastructure as Code, Policy as Code, and Everything as Code techniques provides security and value in a number of ways. First, it obviates lethargic and error-prone step-by-step deployment and configuration guides performed manually. In a legacy, manual driven approach, it is too easy to inadvertently skip a step or mistype a configuration command. Second, it confirms that the command will execute as expected, mitigating the risk of a change without any type of peer review before execution. Third, by providing a standard deployment model, a standard set of outputs can be auto-ingested into Defensive Cyber Operations (DCO) platforms and data collection/visualization mediums. This allows DCO to begin instantaneously and provides data analytics to identify the necessary next innovations.

This guiding principle establishes a clear mandate for automated infrastructure configuration driven by code. Code can be version controlled, tested, peer reviewed, and its execution (logs) tracked. The precise practices and tooling approaches are defined further within the DevSecOps Fundamentals volume and specific software factory platforms, respectively.

7.5 Adoption of Cloud-smart and data-smart architectural motifs throughout

There is an optimistic vision that portrays the Cloud as offering endless computing capacity, guaranteed availability, and lower operational costs. The reality is that an improperly architected application remains as brittle in a Cloud environment as it did operating in a Regional Data Center. If not re-architected, it may actually be more unreliable and more expensive to operate. The shift to Cloud must be accompanied by the adoption of new architectural design patterns and an overpowering preference to build atop existing enterprise services instead of reinventing duplicative capabilities.

Further, data generation, transport, and consumption shows no signs of abating. Software architectures must consciously acknowledge this with smarter application programing interfaces (API) designs, caching strategies, and data tagging/labeling. Development teams must understand that even the data produced *within* the software factory is consequential, especial in terms of providing an AO with trustworthy cybersecurity metrics that enable and support cATO.

[7] National Institute of Standards and Technology, "NIST Special Publication 800-207, Zero Trust Architecture." August, 2020.

8 DevSecOps Process Overview

The overall DevSecOps lifecycle phases are covered in-depth in the DevSecOps Fundamentals document. *Figure 6* visually depicts the DevSecOps phases, feedback loops, and control gates. The lifecycle is built around a series of *sprints*, with each sprint covering the Plan, Develop, Build, Test, Release & Deliver, Deploy, Operate, and Monitor phases. This graphic contains the identical set of steps depicted previously in *Figure 3* as an infinite loop, but it has been "unfolded" to effectively illustrate the multiplicity of continuous feedback loops. Visually, the cybersecurity automation is depicted as the foundational core underpinning all lifecycle phases, permeating each phase with multiple touch points, and directing actions that are taken based on real-time metrics derived from actual product usage and performance.

The other feedback loop covered below is the Continuous Monitoring loop. This loop must bring together a deep, rich set of real-time performance metrics and supporting data to continuously evaluate the totality of the software environment. This loop serves two main functions; cybersecurity monitoring to ensure events and incidents are handled in accordance with DoD mandates and policies and live data feedback and interaction between network defenders and developers. In doing so, the antiquated snapshot view of network security is replaced with real time feeds, allowing security actions to be taken by local defenders, monitoring teams (Cybersecurity Service Providers, or CSSPs), incident response teams (Cyber Protection Teams, or CPTs) and Command and Control (C2) elements of U.S. Cyber Command/Joint Force Headquarters – DoD Information Network (JFHQ-DoDIN).

Feedback loops are critical mechanisms that overlap with specific DevSecOps lifecycle phases. Each feedback loop is built upon transparency and speed. As an illustration, when a software developer commits code to a branch, a build is automatically triggered to confirm the code still builds correctly, and if it doesn't, the developer is immediately notified of the problem. The DevSecOps Fundamentals document covers each feedback loop and the value it adds to the software supply chain's software factory.

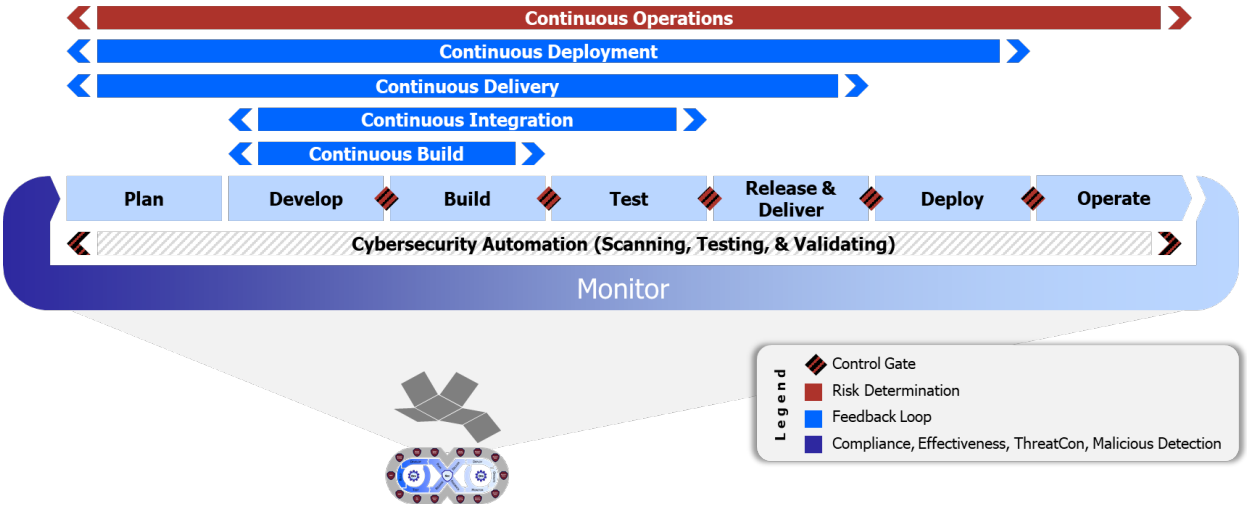


Figure 6 DevSecOps Lifecycle Phases, Continuous Feedback Loops, & Control Gates

9 DevSecOps Management and Governance

Governance in the context of DevSecOps consists of the processes used to actively assess and manage the risks associated with the mission program throughout the lifecycle. Governance activities never end; but are now integrated at each step of the process reducing lag time from inception to production to operations.

Paramount to the DevSecOps Strategy is that cybersecurity automation must permeate the entirety of the software supply chain, never being bolted on as an afterthought. DevSecOps underlying software factory concept is one part of the software supply chain, but it merits additional scrutiny because this is where sensitive, mission-specific tactics, techniques, and procedures are converted into sensitive software algorithms. DevSecOps Management and Governance of the software factory stipulates that a series of cybersecurity control gates execute deep, meaningful, repeatable, and mission-relevant automated cybersecurity metrics. In *Figure 7*, we see one of the pipelines. The black and red diamonds represent the automated cybersecurity control gates that must be cleared before any artifacts can be promoted between the disparate development environments (*dev, test, pre-prod, etc.*).

This automation exemplifies what it means to ensure cybersecurity permeates the entirety of every phase of DevSecOps. Further, it visually depicts explicit gates where Developmental and Operational Test and Evaluation (DT&E and OT&E) must *shift left*. This shift allows the team to rapidly identify quality or stability that should be addressed prior to the promotion of any artifact to the next level. Finally, SMART performance metrics related to both team performance and cyber survivability are collected at each control gate, every time. **These metrics form one of the bedrock principles behind cATO, producing a certified software factory.**

9.1 Management Structure

The management objective of DevSecOps must be both “top-down” and “bottom-up” to balance the larger strategic goals of software modernization across the DoD. Senior leader buy-in is crucial for success, though buy-in at the staff level is equally important. This engenders a sense of ownership, which encourages the appropriate implementation of processes related to governance and enables team members to support continuous process improvement. Continuous process improvement – seeking opportunities to simplify and automate whenever and wherever possible – is essential for governance to keep pace with a rapidly changing world while implementing a continuous feedback loop to ensure that automation is not done at the cost of security.

9.2 Recommended Governance

Early DevSecOps efforts in the DoD, such as Defense Threat Reduction Agency (DTRA) have leveraged and adopted commercial best practices with great success. The DTRA Governance document identifies Five Fundamental Principles of Next Generation Governance (NGG):[8]

1. *Run IT with Mission Discipline*: Tie requirements back to your organization’s mission. Every action should be aligned to the mission. If they are not, then an evaluation should be performed with continuous process improvement to address how to tie actions to missions.

[8] Defense Threat Reduction Agency (DTRA), "Next-Generation Technology Governance," 2018.

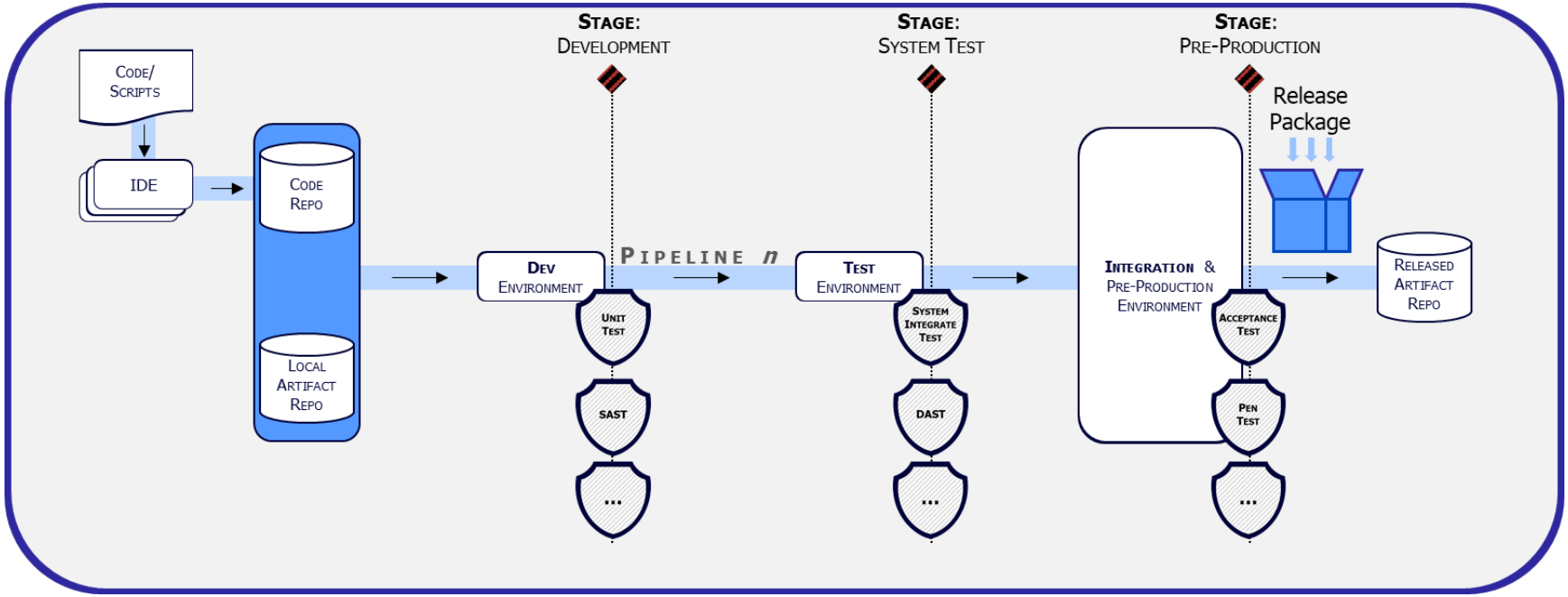


Figure 7 Notional expansion of a DevSecOps software factory with illustrative list of tests

2. *Invest in Automation*: Automate everything possible, including actions, business processes, decisions, approvals, documentation, and more. Automated testing should validate the design, public interface behaviors, and perform a wide array of functional and security tests. The documentation generated from these automated activities create the body of evidence required by the Risk Management Framework (RMF) and for historical audits when needed.
3. *Embrace Adaptability*: Accept that change can be required at any time, and all options are available to achieve it. Fail fast, fail small, and fail forward. The typical example of *failing forward* is exemplified like this: Consider a production release that fails to operate as expected. Instead of restoring the software to its pre-deployment state, the developer's change should be discrete enough that they can rapidly fix it and address the issue through a newer release in a comparable period of time.
4. *Promote Transparency*: Offer open access across the organization to view the activities occurring within the automated process and to view the auto-generated Artifacts of Record. Transparency generates an environment for sharing ideas and developing solutions comprised of Subject Matter Experts (SMEs) or leads from across the enterprise in the form of cross-functional teams to avoid the "silo effect." When composed of all representative stakeholders, the team possesses the skills needed to build a mission system and the collective ingenuity necessary to overcome all encountered challenges.
5. *Inherent Accountability*: Push down or delegate responsibility to the lowest level:
    - Strategic: This is related to the Change Control Board (CCB) or Technical Review Board (TRB); it involves "Big Change" unstructured decisions. These infrequent and high-risk decisions have the potential to shape the strategy and mission of an organization.
    - Operational: (Various Scrum) Cross-cutting, semi-structured decisions. In these frequent and high-risk decisions, a series of small, interconnected decisions are made by different groups as part of a collaborative, end-to-end decision process.
    - Tactical: (Global Enterprise Partners (GEP)/Product Owner/Developers Activities) Delegated, structured decisions. These frequent and low-risk decisions are effectively handled by an individual or working team, with limited input from others.

9.3 Assessment and Authorization

DoD Instruction (DoDI) 8510.01 is the existing governance policy and defines the processes that all DoD information system and platform information technology system must follow. CJCSM 6510.01B defines overall cyber incident handling.[9,10] Some of the concepts in modern software architecture and in DevSecOps present challenges against the current version of that policy. Some of these include:

- Microservices architecture, where the "system boundary" is in flux;
- Distributed systems, where the is no traditional "system boundary"; and

[9] DoDI 8510.01, "Risk Management Framework (RMF) for DoD Information Technology," March 12, 2014.
[10] CJCSM 6510.01B, "Cyber Incident Handling Program," July 10, 2012.

- Continuous integration / continuous deployment, where the “system” evolves in a controlled but rapid fashion.

As the DoD works to finalize its Software Modernization Strategy there is a recognition that the Department must continuously evaluate and update policies, regulations, and DoD standards (collectively, “compliance”) as appropriate. Nothing within the four-corners of the DoD Enterprise DevSecOps Strategy Guide, the DevSecOps Fundamentals document, or any specific DoD Enterprise DevSecOps reference design can be deemed as overruling existing governance policies of the department. However, more engaged collaboration, a recognition of the shortcomings of the current procedures, and a documented appetite to tackle outdated compliance approaches should be viewed in a positive light.

10 Conclusion

The adoption and assertion of DevSecOps cultural and philosophical practices are a central theme of DoD software modernization that will drive the *delivery of software capabilities at the speed of relevance*. This document establishes a unified set of DevSecOps guiding principles for the entirety of the DoD. These principles are weaved throughout the fabric of the other documents within the DevSecOps document set, visually depicted in *Figure 2*. In recognizing the depth and breadth of software development activities across the entirety of the department, specific DevSecOps Reference Designs empower specificity, demonstrating that neither a *one-size-fits-all* nor a *one-size-fits most* approach is sufficient. The future success and global relevance of the DoD demands an accelerated adoption of software industry best practices.