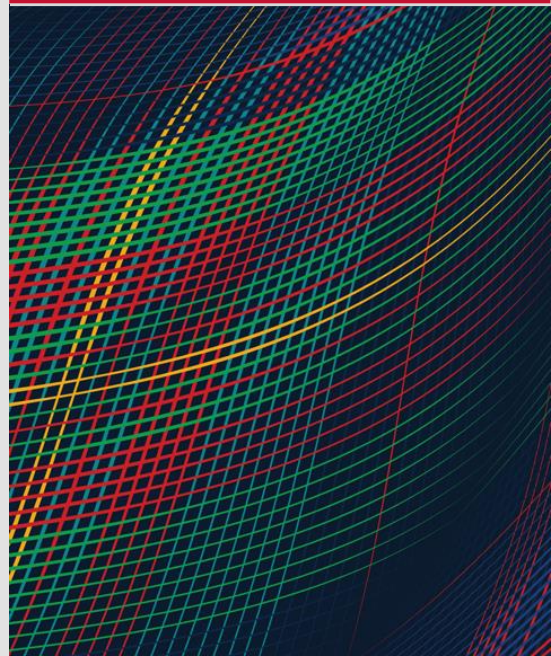
Carnegie Mellon University
Software Engineering Institute

Software Acquisition Pathway Primer for Senior Leaders



MARCH 1, 2025

Julie B. Cohen
William E. Novak
Patrick R. H. Place
Joseph Yankel

Document Markings

DM25-0282

Agenda

- **Software Acquisition Pathway (SWP) – Overview**
- Agile – Basics, Agile Culture Change
- SWP – Planning & Execution Phases
- DevSecOps (DSO) – Basic, Definition, Implementation
- SWP/Agile/DevSecOps Shared Concepts
- Summary

Objectives

- Ensure you are prepared to be good stewards of your programs that are utilizing the Software Acquisition Pathway (SWP).
- Provide enough information on Agile and DevSecOps to understand how SWP programs execute.
- Answer your questions regarding the SWP, Agile, and DevSecOps.

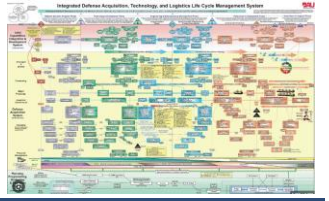
What Is the Software Acquisition Pathway Meant to Do?

- Improve software acquisition in the DoD.
- Take a risk-based approach and tailor the acquisition.
- Instantiate modern software development methods in DoD programs.
- Use Agile techniques (iterative and incremental).
- Use DevSecOps pipelines for software development and test.
- Enable faster delivery of software to the field.
  - Deliver at least once per year, but the goal is to deliver weekly or monthly.
- Ensure end users are involved in every stage of the process.

End-user involvement is vital to the success of programs on the SWP.

Software Is Different

Old Way

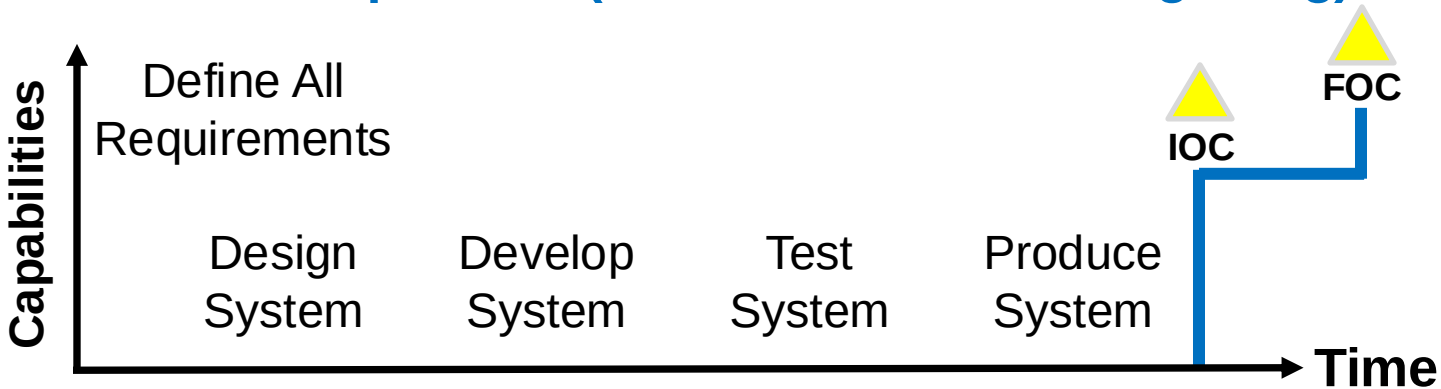


The threat of the same-old pace is real. The result is the U.S. is losing its technical edge to great-power competitors like

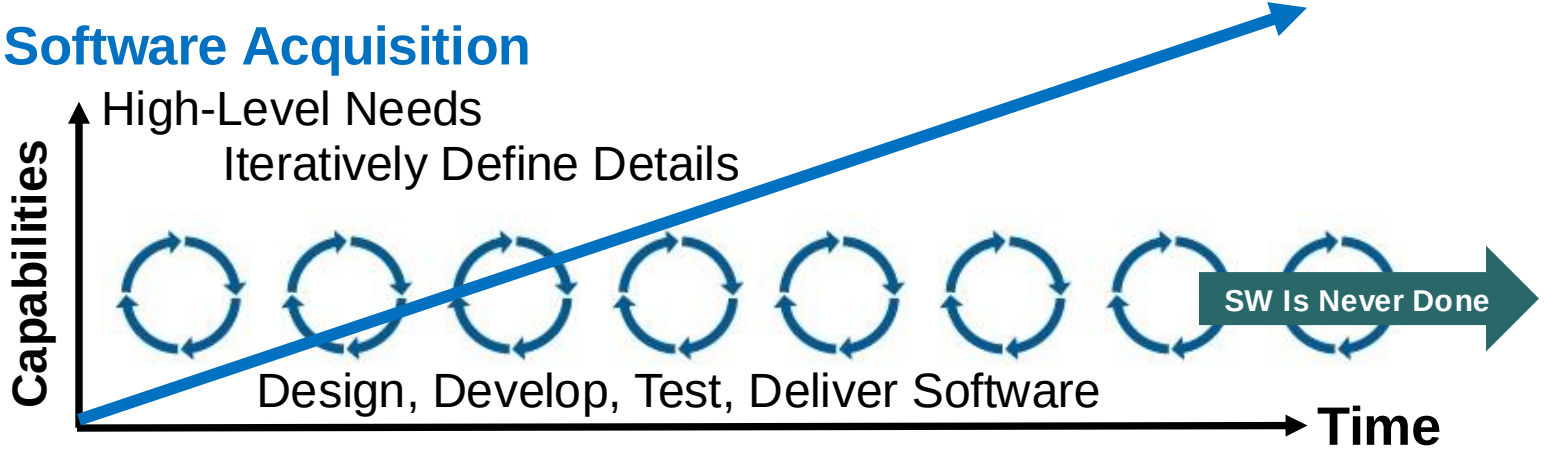
New Way

We may need to change software *every day* as a necessary factor for winning a fight.

Traditional Acquisition (Hardware Centric – Big Bang)

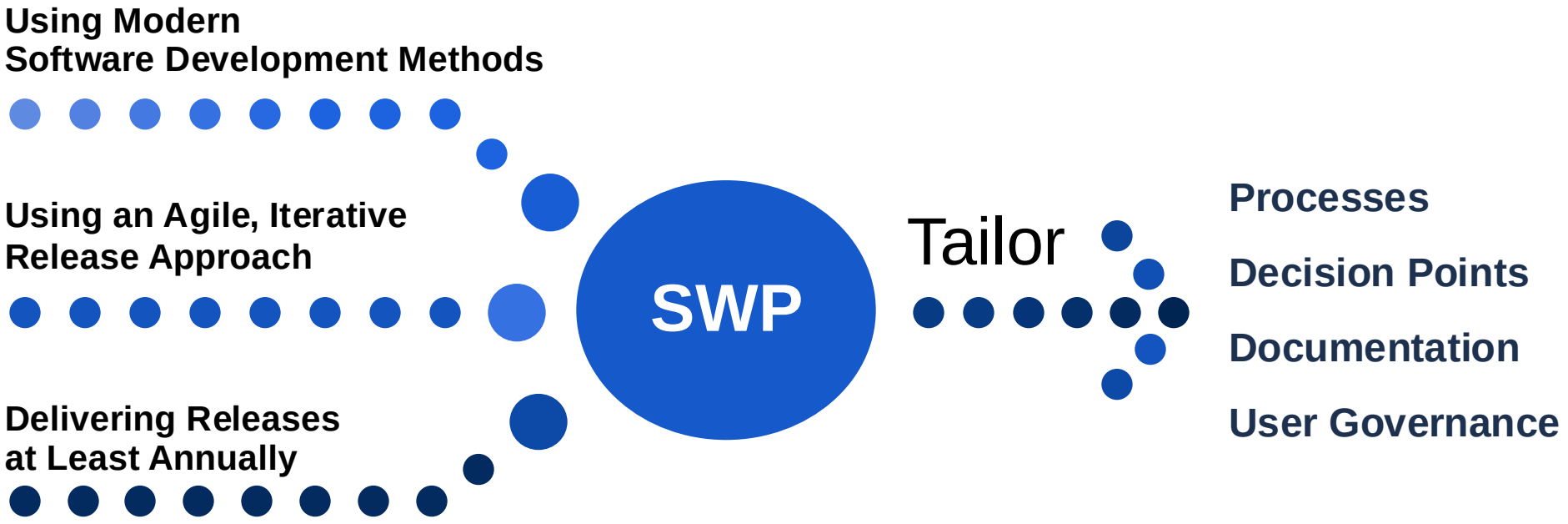


Software Acquisition

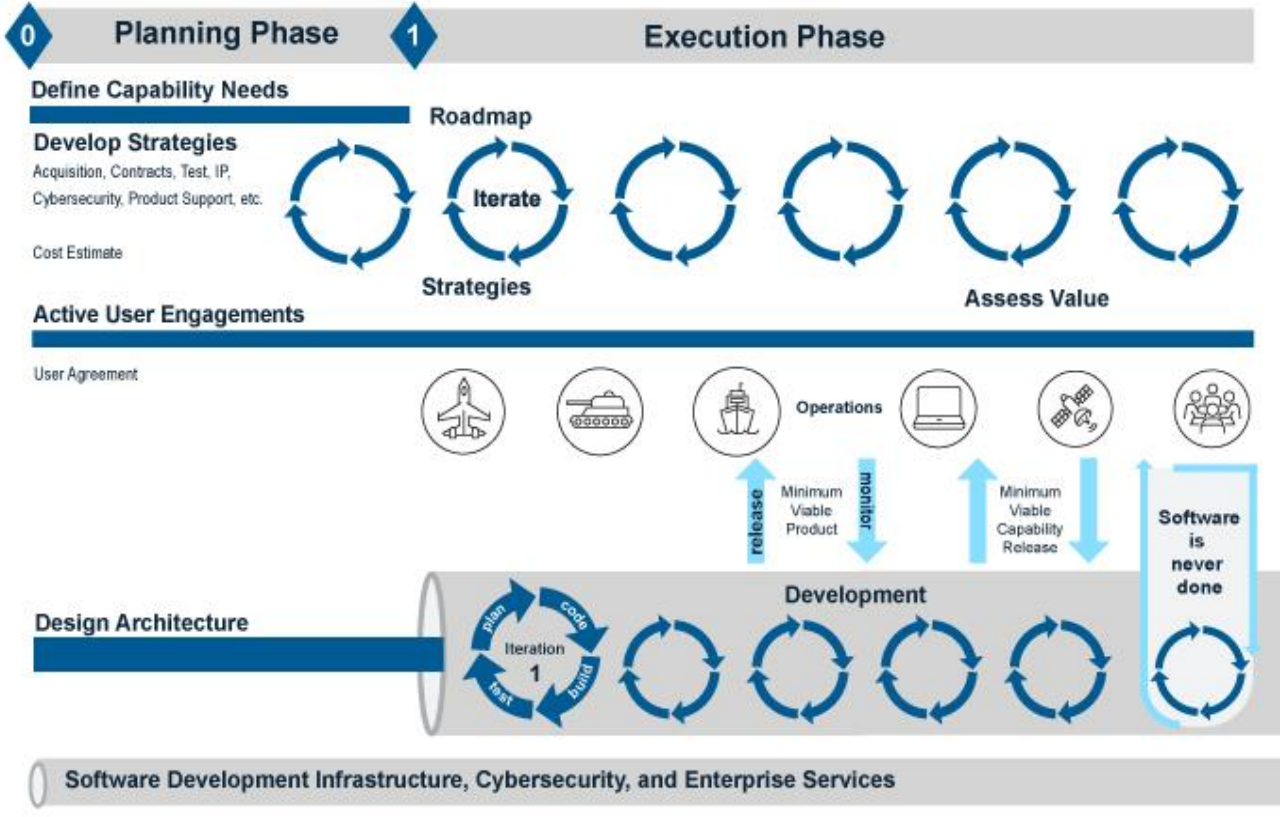


Decision Matrix for Using the SWP

If the software development program meets all three criteria, consider using the SWP.



Software Acquisition Pathway



https://aaf.dau.edu/aaf/software/ From DoD A&S

SWP Sub-Paths

Applications	Rapid development and deployment of custom software running on commercial hardware and cloud computing platforms
Embedded	Rapid development and insertion of upgrades and improvements for software embedded in weapon systems and other military-unique hardware systems
Business Systems	Rapid development and deployment of critical business system capabilities (All DBS systems must meet Clinger-Cohen Act and Title 10 USC 2222 requirements.)

An AI sub-path is being explored.

SWP Terminology

Carnegie Mellon University Software Engineering Institute

Capability Needs Statement (CNS): *A high-level capture of mission deficiencies, or enhancements to existing operational capabilities, features, interoperability needs, legacy interfaces, and other attributes that provide enough information to define various software solutions as they relate to the overall threat environment*

User Agreement: *A commitment between the sponsor and PM for continuous user involvement and assigned decision-making authority in the development and delivery of software capability releases*

Value Assessment: *An outcome-based assessment of mission improvements and efficiencies realized from the delivered software capabilities and a determination of whether the outcomes have been worth the investment (The sponsor and user community perform value assessments at least annually to inform DA and PM decisions*.)

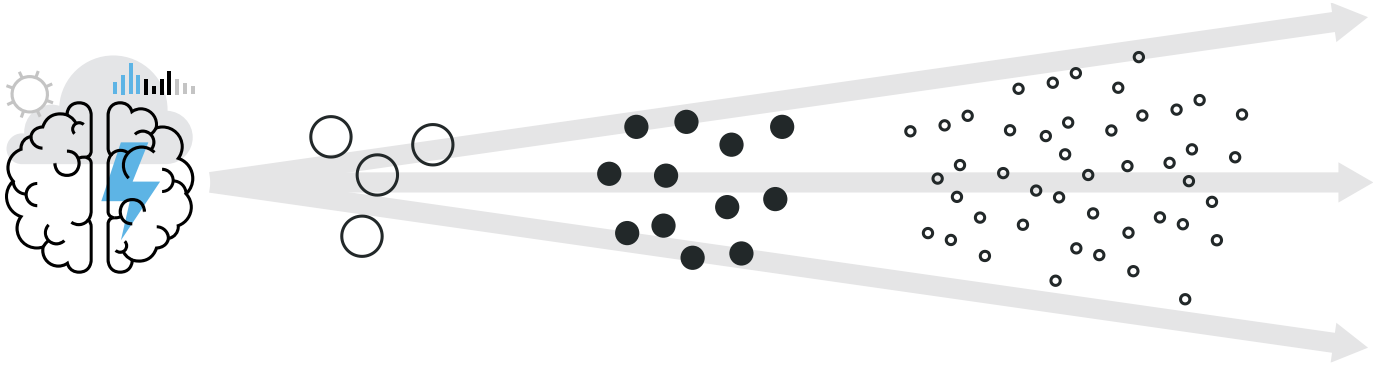
Program Backlogs: *Identified detailed user needs in prioritized lists that allow for dynamic reallocation of current and planned software releases; issues, errors, threats, and defects identified during development and operations, including software updates from third parties or suppliers, are captured in the program's backlogs to address in future iterations and releases*

Agenda

- Software Acquisition Pathway (SWP) – Overview
- **Agile – Basics, Agile Culture Change**
- SWP – Planning & Execution Phases
- DevSecOps (DSO) – Basic, Definition, Implementation
- SWP/Agile/DevSecOps Shared Concepts
- Summary

Enabling Mission Outcomes

Agile as a Mindset | **Described by 4 Values** | **Defined by 12 Principles** | **Manifested in Many Practices** | **Enabling Business Outcomes**

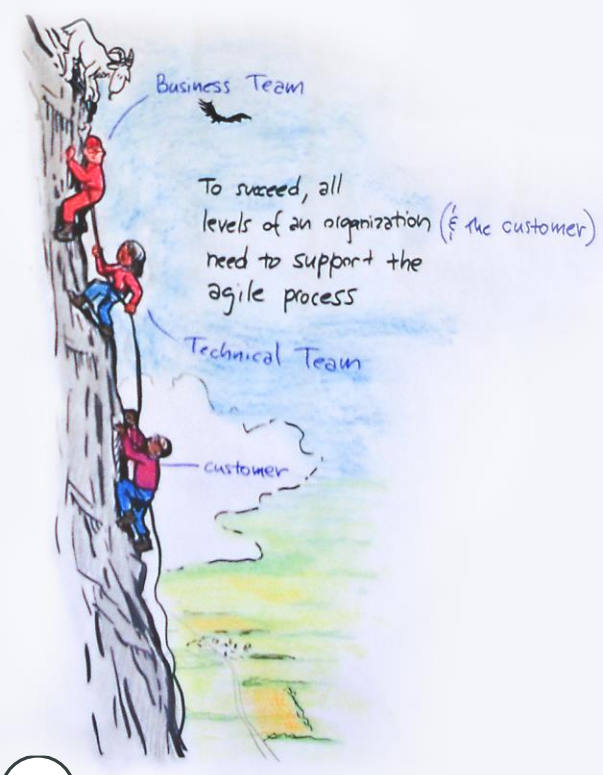


Time to Deployment
Warfighter Satisfaction
Contractor Satisfaction
Reliability Innovation
Responsiveness
Predictability

Implementing the practices, tools, and processes **without** the Agile mindset, values, and principles of the Agile Manifesto ***is NOT Agile!***

It isn't enough to adopt the practices of a successful team. You *must* adopt attitudes and a mindset for making decisions to adopt practices that will lead to your success.

Working Definition of Agile



Agile (*adj.*): An *iterative* and *incremental* (evolutionary) approach to **software** development which is performed in a *highly collaborative manner* by *self-organizing teams* within an *effective governance framework* with *"just enough" ceremony* that produces *high quality software* in a *cost effective and timely* manner which *meets the changing needs of its stakeholders*. [Ambler 2013]

[Ambler 2013] Ambler, Scott. *Disciplined Agile Software Development: Definition*. http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

Incremental delivery leads to the early realization of value, in terms of additional capability to the warfighter

Agile Terminology

Carnegie Mellon University Software Engineering Institute

Agile uses a lot of jargon (e.g., ceremonies)

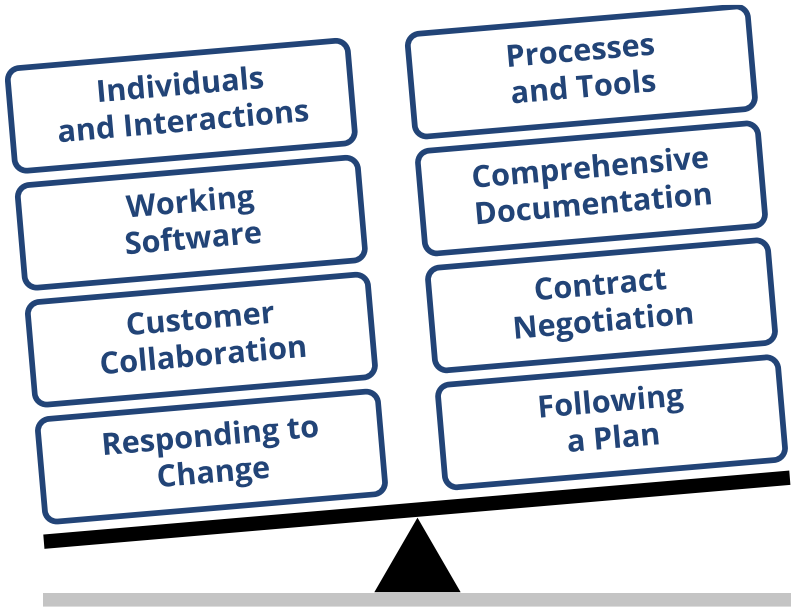
- Much of the terminology is specific to a practice.

Common Terms

- **Scrum:** *An approach to managing small teams (typically 5-9 people)*
- **Sprint:** *Fixed duration (typically 1 week to 1 month) in which development goals are set and completed*
- **Increment:** *A fixed number of Sprints in which larger units of work are planned and completed*
- **User Story:** *A description of a piece of work (typically small enough to complete in a Sprint)*
- **Epic:** *A description of a piece of work to be broken down into User Stories*
- **Definition of Done:** *The set of activities deemed necessary to complete a piece of work*
- **Minimum Viable Capability Release (MVCR):** *A subset of the entire system that provides useful functionality and is appropriate for release*

Reorienting the Manifesto for Agile *Software Development* Toward *System Acquisition*

Through this work, we have come to value the following:



That is, while there is value in the items on the right, we value the items on the left more.

Common Myth

The manifesto is often **mis**interpreted to mean

- no documentation,
- no process, and
- no plan!

https://agilemanifesto.org/history.html

Agile Principles

Carnegie Mellon University Software Engineering Institute

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable **software**.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together** daily throughout the project.
5. **Build projects around motivated individuals**. Give them the environment and support they need and trust them to get the job done.
6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**
7. **Working software is the primary measure of progress.**
8. Agile processes **promote sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity—**the art of maximizing the amount of work not done—is essential.
11. The **best architectures, requirements, and designs emerge from self-organizing** teams.
12. At **regular intervals**, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Principles: http://agilemanifesto.org

Reminder: Agile Is a Mindset

Adopting the mindset requires change in culture at all levels:

- Culture change is hard and requires leadership commitment.
- Leadership also needs to focus on enabling Agile and removing barriers.

Examples of mindset change from

- Buying a product to buying an ongoing delivery stream
- Focusing on formal milestones to focusing on demonstrations of working systems
- “Hands off” oversight to active engagement
- Engineering the perfect solution at the outset to choosing “good enough” for now and adjusting as needed

Summary

Agile will not solve all complex problems of government software acquisition efforts.

- *But it has contributed to successful software acquisitions.*

Benefits from Agile show only when developer and acquisition efforts are aligned.

Government oversight obligations must change with Agile development.

- *There have been negative consequences in organizations that do not address these changes.*

Changing the oversight approach in Agile settings means asking different questions on a new cadence.

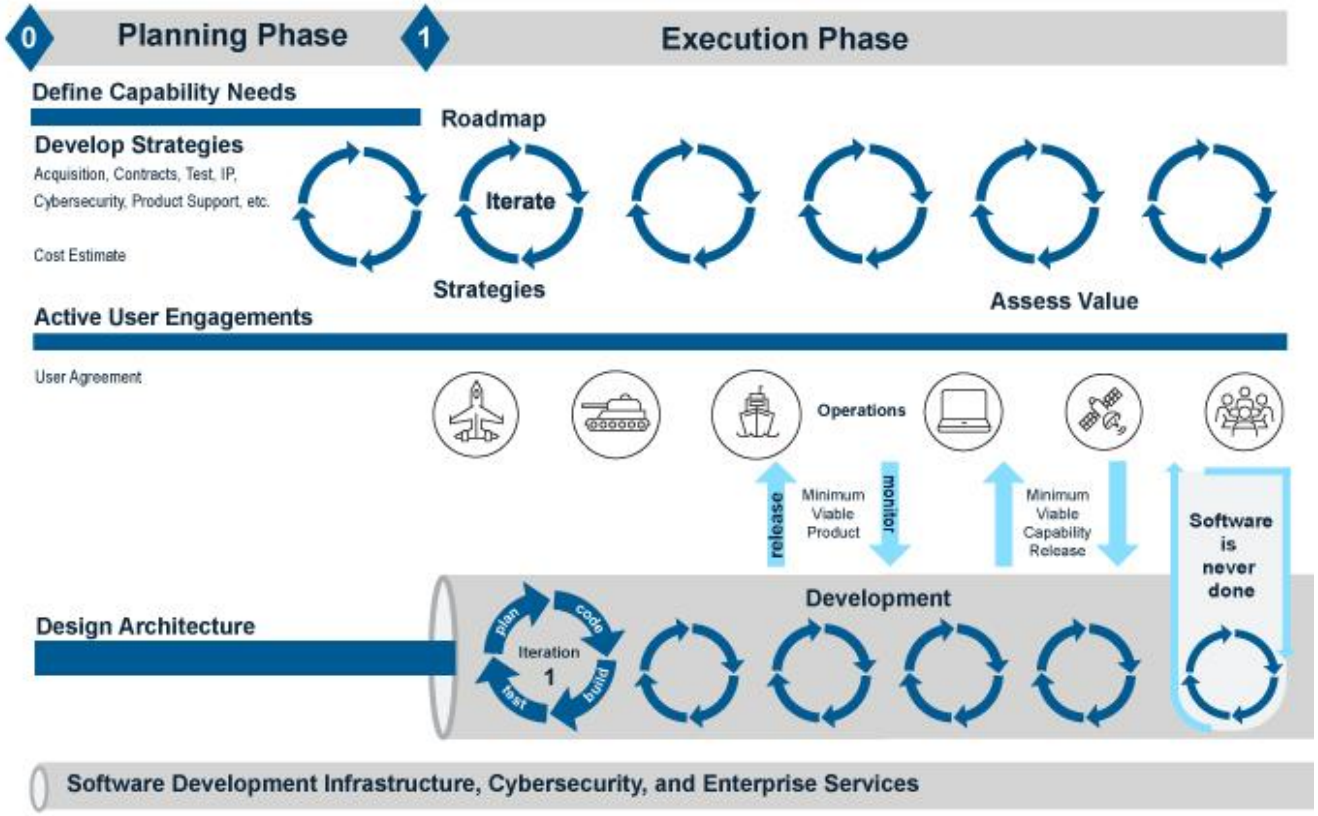
- *It leads to different measurement and reporting approaches as well.*

A focused government workforce development effort is key to enabling the knowledge, skills, and abilities needed for effective oversight and interaction in Agile settings.

Agenda

- Software Acquisition Pathway (SWP) – Overview
- Agile – Basics, Agile Culture Change
- **SWP – Planning & Execution Phases**
- DevSecOps (DSO) – Basic, Definition, Implementation
- SWP/Agile/DevSecOps Shared Concepts
- Summary

Software Acquisition Pathway



https://aaf.dau.edu/aaf/software/

Planning Phase 1/2

Purpose: Better understand the users' needs and prepare for execution

Needed to enter: Draft CNS approved by the sponsor and ADM

Activities:

- Plan the approach to deliver software capabilities that meet user needs.
- The DA will select a PM and may establish a new program office or assign an existing program office to shape and plan the software acquisition.
- The PM will develop a constrained, tailored set of strategies to acquire, develop, and deliver the software capabilities, and the PM will obtain the necessary resources (e.g., people, funding, technology) to effectively execute the strategies.
- Begin developing the software design and architecture, leveraging existing enterprise services as much as possible.

Planning Phase 2/2

Activities (continued):

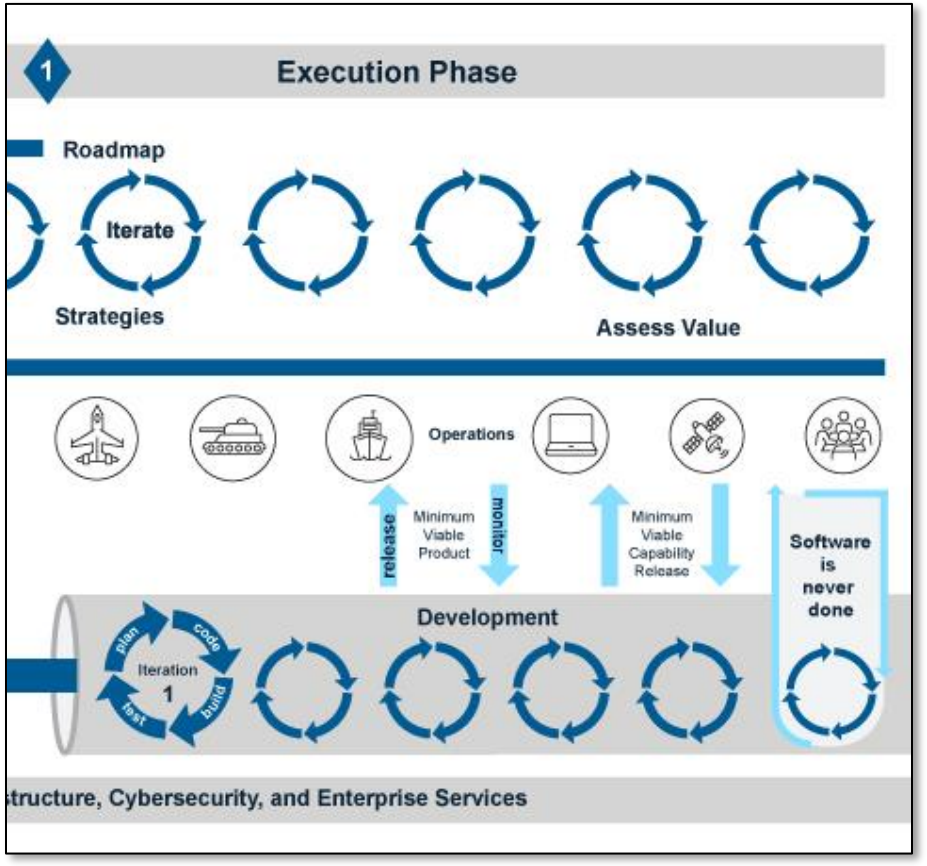
- Consider the development environment, processes, automated tools, designs, architectures, and implications across the broader portfolio, component, and joint force.
- Plan for continuous testing and evaluation and cybersecurity, with maximum automation.
- Develop a risk-based lifecycle management approach to address software vulnerabilities, supply chain and development environment risks, and intelligence threats throughout the entire lifecycle.
- The program may develop prototypes to explore possible solutions, consider architecture options, and solicit user feedback.
- The DA will validate that there are appropriate strategies, analysis, and resources in place to successfully transition from the planning phase to the execution phase.

Planning Phase – Major Artifacts

- CNS or SW-ICD (for Joint Equities) 
- User Agreement (UA)
- Acquisition Strategy (Includes Market Research, Acquisition Approach, Business Strategy, Contract Strategy, IP Strategy, Product Support Strategy, Metrics Plan, Risk Management Plan)
- Cybersecurity Plan
- Test Strategy
- Information Support Plan
- Cost Estimate 

Programs using the embedded software sub-path align their strategies with the programs they will be integrated into.

Execution Phase – Key Activities

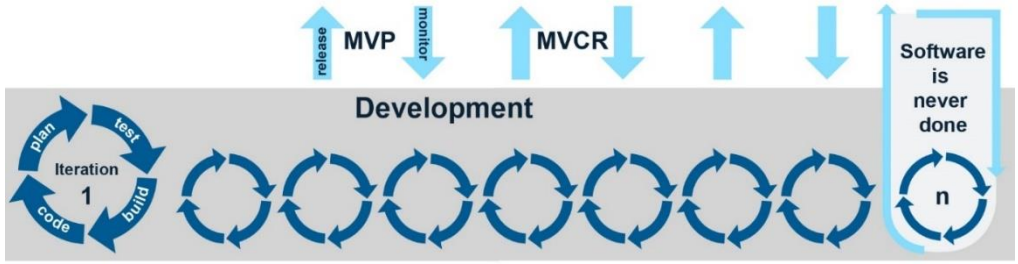


Continuous Improvement to Maximize Mission Impact

- Product Roadmap
- Program Backlogs
- Active User Engagements
- Develop and Deliver Software
- Track Metrics
- Value Assessments

From DoD A&S

Execution Phase – MVP, MVCR, and Iterative Deliveries



The statute requires software deliveries at least *annually*.

Minimum Viable Product (MVP)

Early version of software for users to evaluate and provide feedback

DEMO

ASAP

Minimum Viable Capability Release (MVCR)

Fully tested set of value-added features fielded to an operational environment

DELIVER

Within 12 months of obligating funds

https://aaf.dau.edu/aaf/software/mvp-mvcr/

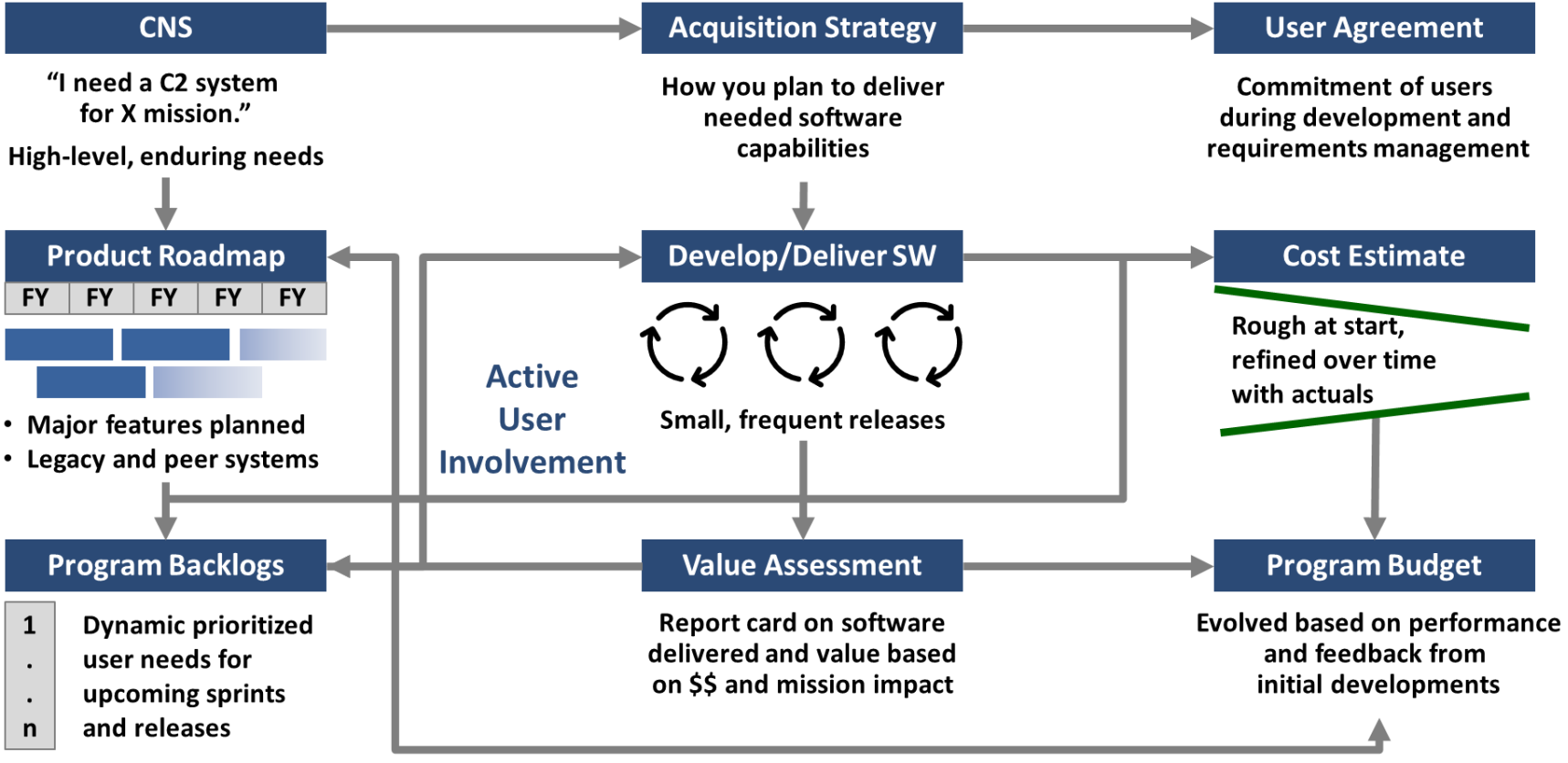
SWP Information Requirements

Entering the Planning Phase	
ADM Signed by DA	Draft Capability Needs Statement
Entering the Execution Phase	
Capability Needs Statement (Equiv.)	User Agreement
Acquisition Strategy	Cybersecurity Strategy
Test Strategy	IP Strategy
Product Support Strategy	Information Support Plan
Program Cost Estimate and ICE	CARD
ADM Signed by DA	
During the Execution Phase	
System Architecture	Product Roadmap
Program Backlogs	Strategy Updates
CARD/Cost Estimate Updates	Value Assessment
Program Metrics	Program Reporting

Balance speed with rigor; focus on delivering software, not documents.

See details at: https://aaf.dau.edu/aaf/software/develop-strategies/

SWP Interplay of Key Elements



Agenda

Software Acquisition Pathway (SWP) – Overview

Agile – Basics, Agile Culture Change

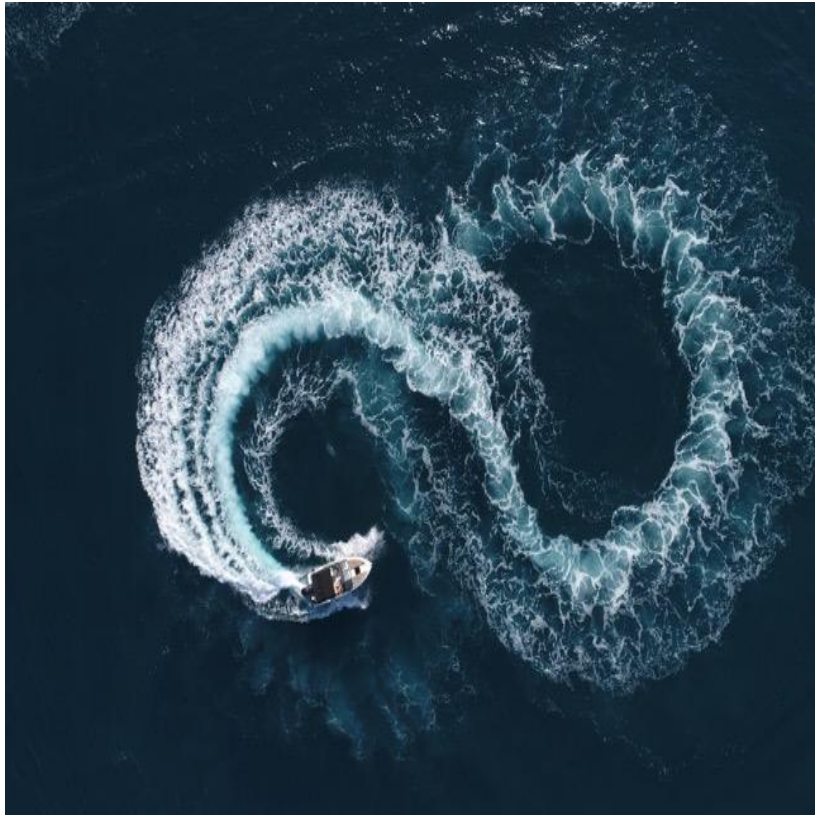
SWP – Planning & Execution Phases

DevSecOps (DSO) – Basic, Definition, Implementation

SWP/Agile/DevSecOps Shared Concepts

Summary

DevSecOps: Simply a term for modern software engineering practices and tools that encompass the full software lifecycle



DevSecOps is a "cultural and **engineering practice** that breaks down barriers and opens **collaboration between development, security, and operations** organizations **using automation** to focus on rapid, frequent delivery of secure infrastructure and software to production. It encompasses intake to release of software and manages those flows predictably, transparently, and with minimal human intervention/effort" [1].

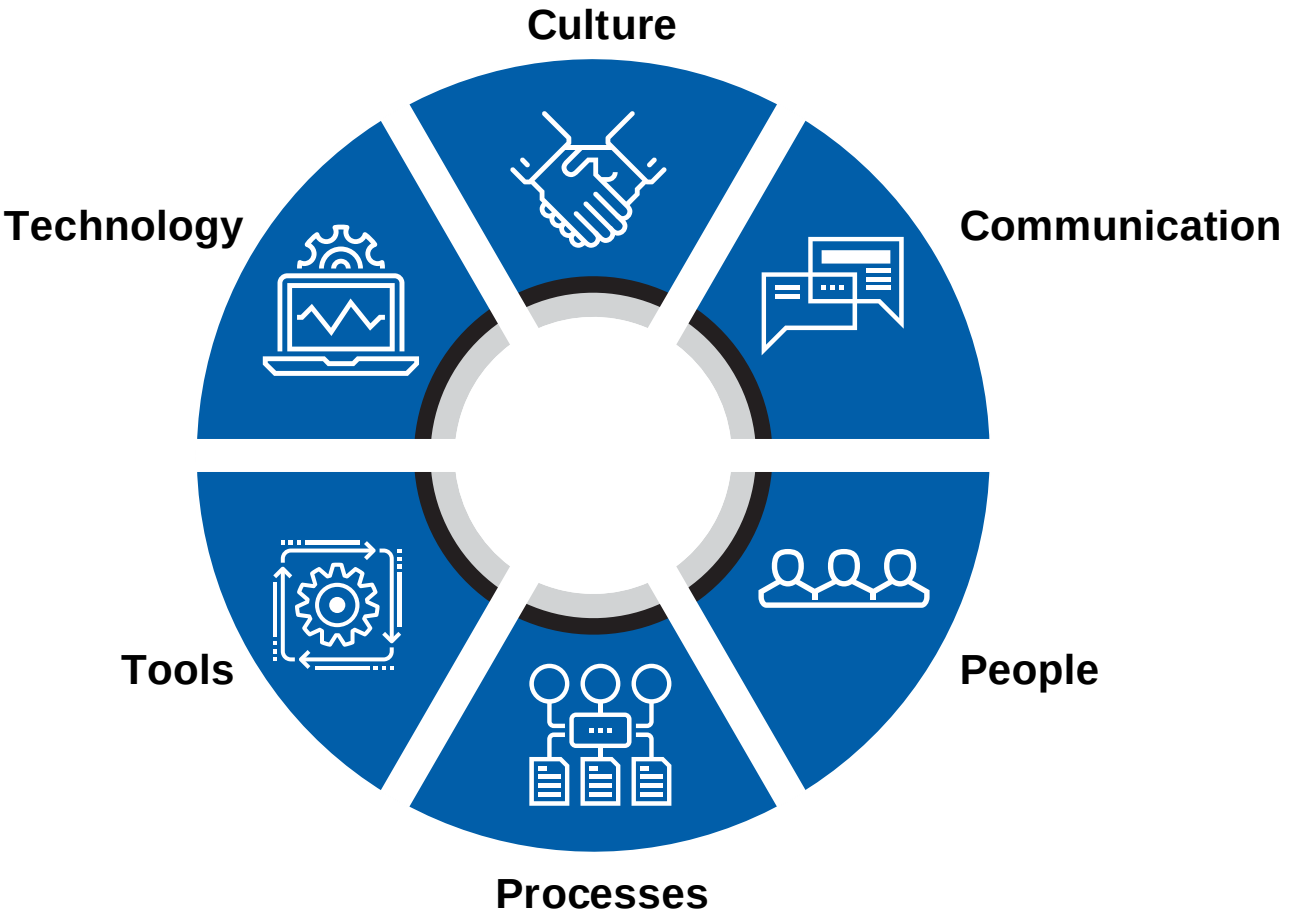
A **DevSecOps Pipeline** attempts to seamlessly integrate "three traditional factions that sometimes have opposing interests:

- **development**; which values features;
- **security**, which values defensibility; and
- **operations**, which values stability" [2].

Not only does one need to balance the factions, but one must also do so in a way that balances **risk**, **quality,** and **benefits** within **time**, **scope**, and **cost** constraints.

[1] DevSecOps Guide: Standard DevSecOps Platform Framework. U.S. General Services Administration. https://tech.gsa.gov/guides/dev_sec_ops_guide.
[2] DevSecOps Platform Independent Model, https://cmu-sei.github.io/DevSecOps-Model/

DevSecOps Key Drivers



DevOps Has Four Fundamental Principles

Collaboration: All stakeholders must be aligned on shared goals, breaking down silos, and delivering high-quality software faster.

Infrastructure as Code: Infrastructure is managed using versioned, machine-readable code to provide consistent environments and reduce human error.

Automation: Any manual or human-error-prone processes for testing, provisioning, or deployment implemented using CI/CD pipelines

Monitoring: Provide visibility into development or operational spaces that can inform priorities, direction, and policy.

DevSecOps Is an Extension of Agile Thinking

Agile

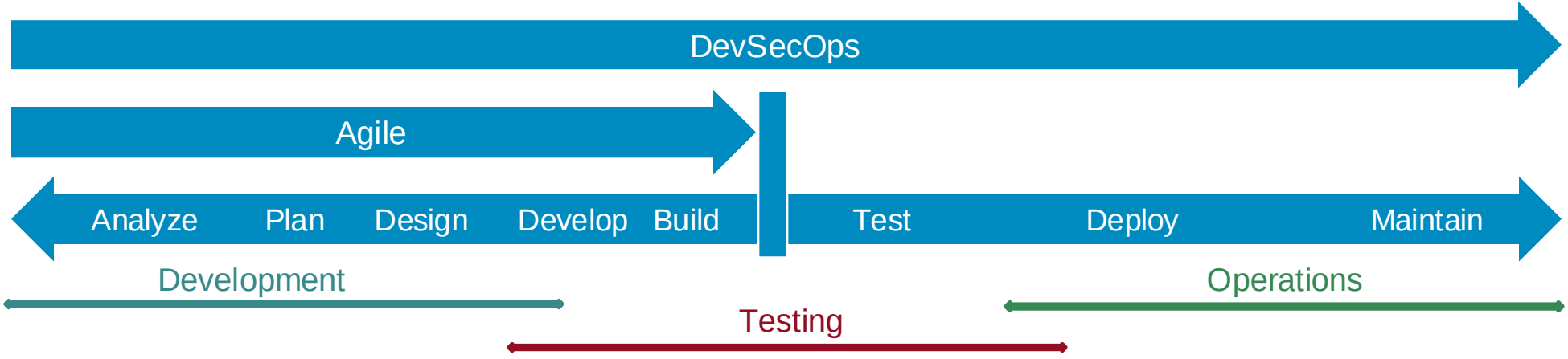
Embrace constant change.

Embed the customer in the team to internalize expertise on requirements and the domain.

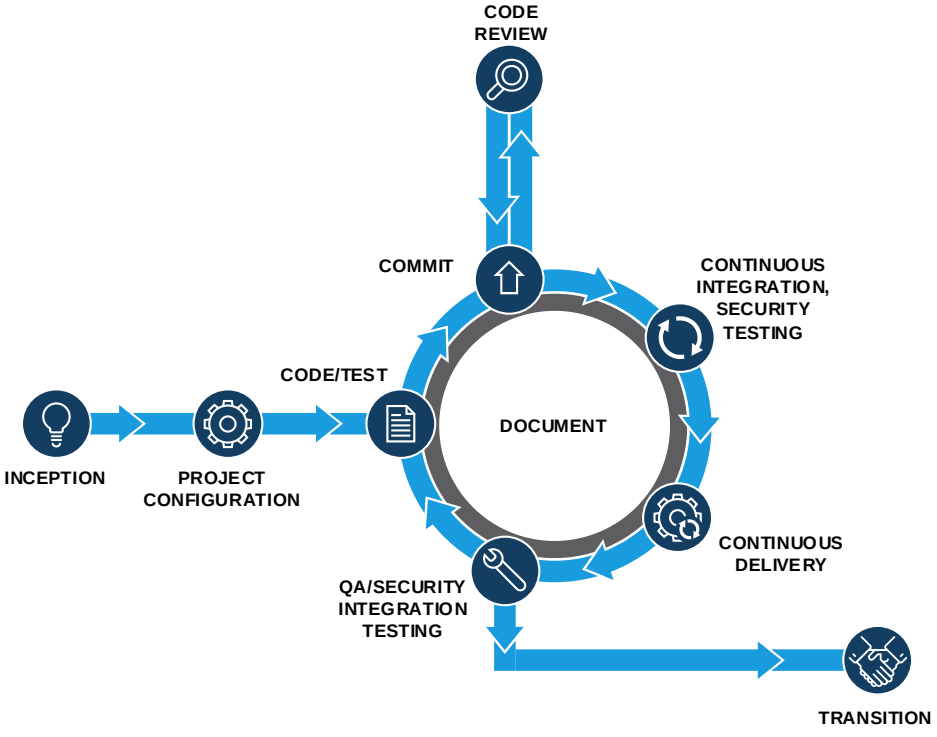
DevSecOps

Embrace constant testing and delivery.

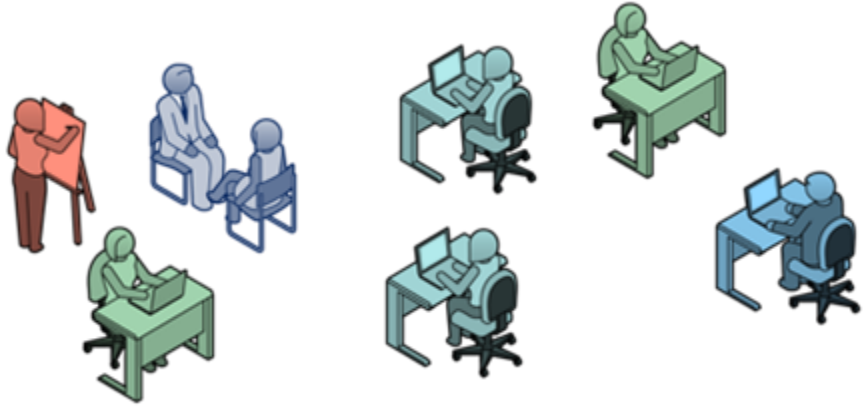
Embed operations in the team to internalize expertise on deployment and maintenance.



DevSecOps Team

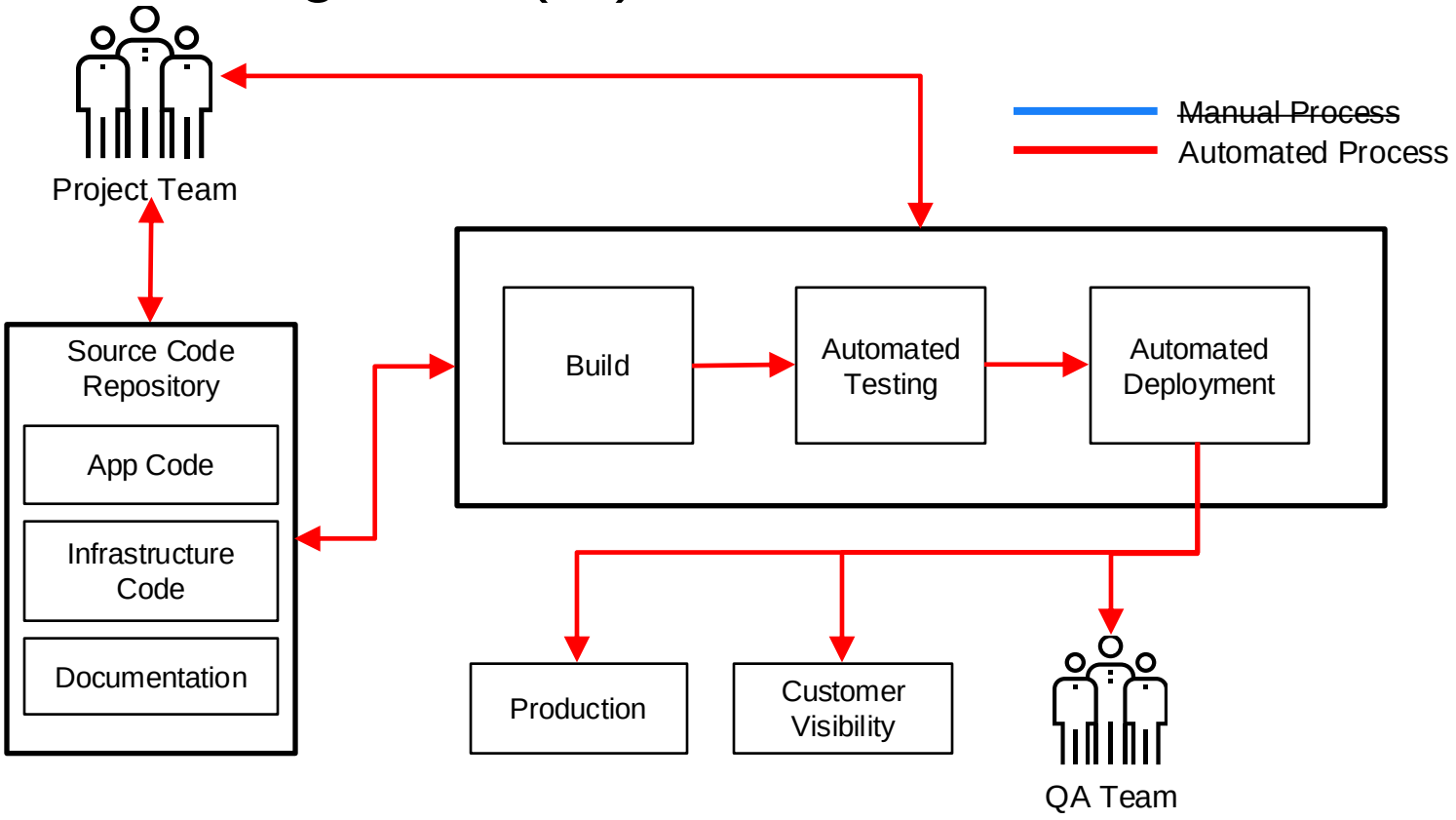


Cross-Functional Development Team, Including IT Operations and Cybersecurity



- Including IT operations and cybersecurity early in development enables deployment to an ops-like environment EARLY in the SDLC.
- Automation enables fast testing and deployment.
- Involve every stakeholder!

Continuous Integration (CI)



Continuous integration is a process that continually merges a system's artifacts, including source code updates and configuration items from all stakeholders on a team, into a shared mainline to build and test the developed system.

Agenda

Software Acquisition Pathway (SWP) – Overview

Agile – Basics, Agile Culture Change

SWP – Planning & Execution Phases

DevSecOps (DSO) – Basic, Definition, Implementation

SWP/Agile/DevSecOps Shared Concepts

Summary

Shared Themes Across SWP, Agile, and DevSecOps 1/4

Aspect	Software Acquisition Pathway (SWP) (*Vision & Policy*)	Agile (*Strategy*)	DevSecOps (*Mechanism*)
Requirements 	Agile Requirements – high-level CNS or SW-ICD. Existing documents can be used when transitioning from another pathway.	Accommodates changing requirements (customer changes, new threats, or based on new learning from early use). Demos should be provided at the end of each sprint or increment to get user feedback on implementation.	CI/CD pipelines are used to continually deliver software for further testing and demonstrating capability. Tooling promotes collaboration and transparency of processes and data.
Test 	Software development testing and operational test and evaluation will be integrated, streamlined, and automated to the maximum extent practicable.	Encourages use of cross-functional teams (e.g., developers and testers) with a test-first mentality. Testers as well as users and other stakeholders can participate in demonstrations. Differentiates between demonstration and test.	Applications and infrastructure are tested using automation via CI/CD. Security and quality are tested continuously, and test artifacts are made available via shared repositories.

Shared Themes Across SWP, Agile, and DevSecOps 2/4

Aspect	Software Acquisition Pathway (SWP) (*Vision & Policy*)	Agile (*Strategy*)	DevSecOps (*Mechanism*)
User Collaboration	Requires a user agreement. Requires value assessments by the users at least annually.	Expects the users to participate frequently (at least in backlog prioritization and demonstrations).	All stakeholders should have access to collaborative tooling and dashboards that are interconnected via automation pipelines.
Metrics	Specific biannual metrics are required to help monitor pathway health. The SWP website includes recommended metrics.	Uses metrics available through the tooling to help track development and testing progress and help to gain insight on project status.	The automated collection and visualization of data is gathered from the many tools interconnected to CI/CD pipelines.

Shared Themes Across SWP, Agile, and DevSecOps 3/4

Carnegie Mellon University Software Engineering Institute

Aspect	Software Acquisition Pathway (SWP) (*Vision & Policy*)	Agile (*Strategy*)	DevSecOps (*Mechanism*)
Contracting 	Recommends multi-award contracting.	Recommends contracting for development capacity versus a specific capability. This typically becomes contracting for a specific number of individuals/teams.	Consider a separate contract for building/maintaining the DSO environment (including the use of government sustainment organizations). DevSecOps pipelines aid in the integration of multiple/modular contracts and provide the proper environment to deliver incremental 'small-batch' capability.
Security	Requires a cybersecurity strategy.	Teams build security into their practices, processes, and pipelines. Incremental development provides opportunities for early cyber checks and correction.	CI/CD pipelines execute security testing whenever changes are made or on a regular schedule. A continuous ATO can be obtained when following best practices of DSO and by collaborating closely with all stakeholders to agree-upon implemented processes. Tools and components can be accessed from a secure source (e.g., Iron Bank).

Shared Themes Across SWP, Agile, and DevSecOps 4/4

Carnegie Mellon University Software Engineering Institute

Aspect	Software Acquisition Pathway (SWP) (*Vision & Policy*)	Agile (*Strategy*)	DevSecOps (*Mechanism*)
Organizational Culture	A culture of performance and a thoughtful, innovative, and disciplined approach to program management.	A people-centered culture operating in rapid learning and fast decision cycles. Face-to-face interaction within development teams. Self-organizing teams produce better results.	Collaborative culture and communication across development, security, and operations. DevSecOps provides the environment and tooling for collaboration by enabling the collection and visualization of metrics and providing common platforms and services for all stakeholders.

Agenda

Software Acquisition Pathway (SWP) – Overview

Agile – Basics, Agile Culture Change

SWP – Planning & Execution Phases

DevSecOps (DSO) – Basic, Definition, Implementation

SWP/Agile/DevSecOps Shared Concepts

Summary

Leadership Responsibilities for SWP Programs

Ensure the culture embraces Agile and DevSecOps processes, ideas, and collaborations.

Ensure the program office is tailoring program documentation and execution for program needs.

Questions You Should Be Asking 1/2

- Does the culture throughout the program (leadership, program office, users, testers, certifiers, contracting, etc.) support iterative development?
- Is the CNS at the right level for the project?
- Is the program office staffed appropriately?
- Are all users prepared to be actively involved in the program?
- Are leaders prepared to engage all stakeholders and embrace a collaborative culture?
- Is the program using modular contracting, and if not, is the reason sufficient?

Questions You Should Be Asking 2/2

- Do the long-term and short-term roadmaps lay out an executable map to the delivery of capabilities?
- Has the program determined which DevSecOps platform to use?
  - Who is building the platform if it is not a service? The government or a vendor?
- Have metrics been identified to allow leadership to assess progress?
  - Who is responsible for implementing data collection for metrics?

Contact Us

Software Acquisition Pathway Primer for Senior Leaders

Software Acquisition Pathway (SWP) Back-Up Slides

Carnegie Mellon University
Software Engineering Institute

What *YOU* Can Do To Enable Software Success

Train Executives, the Workforce	• Truly understand the new software development paradigm. • Teams and contractors go through training together.
Promote New Culture	• Value teaming, speed, and agility. • Small *empowered* teams continuously improving. • *Value-creation focused:* Fail fast; fail cheap.
Processes	• Continue lean processes for SW requirements, T&E, etc. • Transform for small, frequent software releases.
Platforms and Services	• Invest in enterprise/portfolio platforms and services. • Drive open standards, interoperability, and an API ecosystem.
Industry	• Seek out more software companies who “get it.” • Integrate portfolios of SW vendors under government leads.

Barriers to Modern Software Development

Culture	• Unlearn old practices—defining, controlling programs. • Contractor, acquirer, operator, and security in separate groups.
Processes	• Significant documentation and long timelines to acquire. • Linear: requirements, design, develop, test, and deliver.
Platforms and Services	• Limited enterprise platforms, APIs, and services to use. • Lack of investment ($, expertise) to shape and scale.
Workforce	• Workforce lacks experience and training in modern SW. • Struggle with new roles and teaming structures.
Industry and Contract Incentives	• Traditional primes limited in modern SW expertise. • Incentivized to remotely develop closed SW systems.

Key Elements of SW Acquisition Pathway

- Modern software development practices (Agile, DevSecOps, Lean)
- Capitalizing on active user engagement and enterprise services
- Software is rapidly and iteratively delivered to the operational environment to meet the highest priority user needs
- Tightly coupled mission-focused government-industry software teams
- Automated tools for development, integration, testing, certification

Source: DODI 5000.02 Section 4.2

Source: Brady, Sean. *How Software Acquisition & DevSecOps Increase the Lethality of the DoD*, DSO Days, Oct 2020.

Law Requires Timely Deliveries

FY20 NDAA Section 800

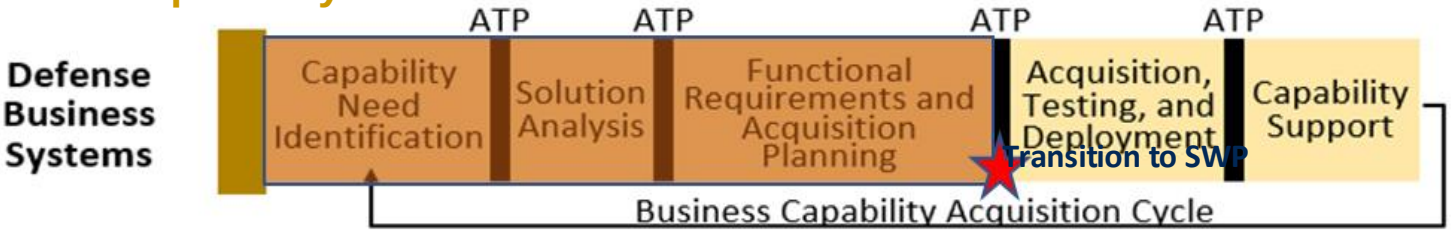
Demonstrate the viability and effectiveness of capabilities for operational use no later than **one year** after the date when **funds are first obligated** to develop the new software.

New capabilities shall be continuously **updated and delivered at least annually** to iteratively meet requirements.

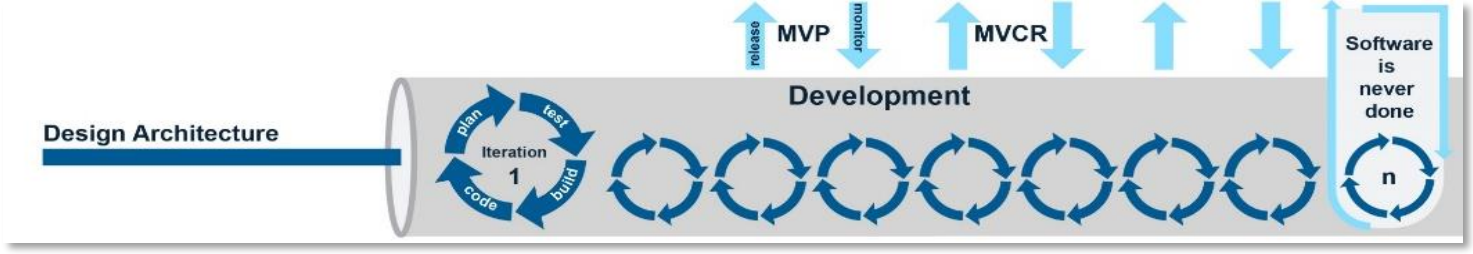
For embedded software, this annual update timeline applies after initial operational acceptance.

The Hybrid SWP-DBS Program

The program uses both the SWP pathway and the DBS pathway to achieve the best outcome.



- **Initiate a business system on the DBS pathway and conduct initial planning using the deliberate BCAC process steps.**
- **Concurrently plan for transition to the SWP (Execution Phase) at the Acquisition ATP.**



This program is best suited for large ERP-type systems where planning will take considerable time given the large functional community.

SWP Cost Estimating Process

1. **Form the team.**
2. **Review the program information.**
3. **Confirm the work breakdown structure.**
4. **Establish a process.**
5. **Articulate the assumptions.**
6. **Understand the potential costs.**
7. **Craft the initial cost estimate.**
8. **Conduct the analysis.**
9. **Integrate the team.**
10. **Iteratively update.**

Capabilities Needs Statement (CNS)

CNS - A high-level capture of need with enough information to define the software solution space and consider the threat environment

The sponsor and requirements manager identifies the operational software capabilities needed.

Draft a CNS to start the software pathway.

Refine it during planning phase and approve it prior to entry into the execution phase

A&S acquisition enablers shop collaborating with components to encourage the adoption of flexible and streamlined requirement processes for the SWP.

Clear Understanding of What Is Needed

https://aaf.dau.edu/aaf/software/cns/

Software Acquisition Pathway (SWP)

The DoD Runs on Software; the SWP Enables Modern Software Development in the DoD.

The acquisition pathway is designed for modern software development by integrating Agile and DevSecOps for rapid, iterative deliveries.

FY20 NDAA Section 800 directed the DoD to develop SWP

- Implements key DIB and DSB software acquisition recommendations
- Exempt from JCIDS, MDAP, accelerated timelines for deliveries (<annual)
- Paths for applications and embedded software (and now business systems)

Collaboratively iterated, developed DODI 5000.87 policy

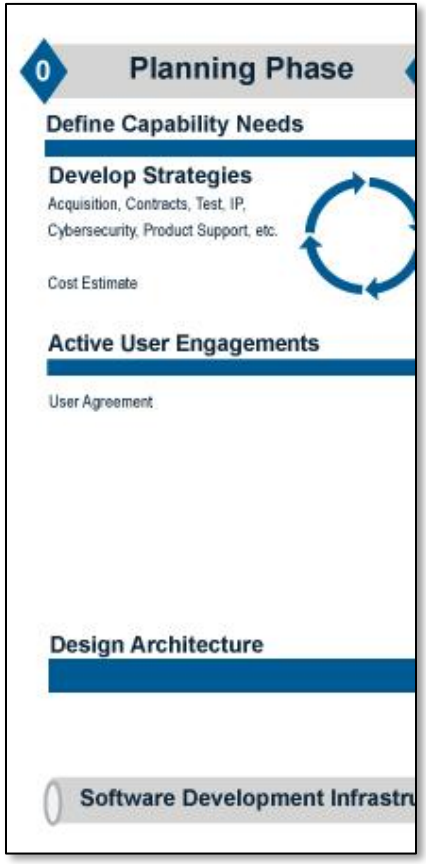
- Streamlined, tailored processes, documents, and reviews for rapid software
- Robust guidance, templates, and workshops to enable program success

Leverages SW Factories, Enterprise Services, and DevSecOps pipelines

- Bake in interoperability, cybersecurity—enterprise efficiencies

https://aaf.dau.edu/aaf/software/

Planning Phase



This phase focuses on understanding the users' and systems' needs and planning the approach to deliver capabilities to meet those needs.

Key Artifacts

- Capability Needs Statement
- User Agreement
- Program Strategies
  - Acquisition Strategy
  - Contracting Strategy + IP Strategy
  - Test Strategy + Cybersecurity Strategy
  - Product Support Strategy
- Cost Estimate

From DoD A&S
https://aaf.dau.edu/aaf/software/planning-phase/

Modern Software Management
Baselines and Progress

Traditional Acquisition Programs

Software Acquisition Programs

- **Dynamic Requirements**
- **Dynamic Costs and Budgets**
- **Regular Cadence of Deliveries**
- **Active User Engagements**
- **Continuous Improvement**

APBs are NOT required for SWP programs; they are discouraged.

- Cannot effectively define and lock down requirements
- Cannot effectively estimate costs on flexible scope

SWP programs are effectively build to budget.

- Plan, iterate, and refine as more is known
- Scale, pivot based on operations, priorities, budget, threats, and performance

Other Potential Metrics

Potential metrics to use with the SWP are discussed on the DAU site.

AREA	SAMPLE METRICS
Process Metrics	Release Burnup, Lead Time
Quality Metrics	Change Fail Rate, Automated Test Coverage
SW Development Progress	Deployment Frequency, Progress Against Roadmap
Cybersecurity	Mean Time to Detect, Mean Time to Remediate
Cost	Total Cost Estimate, Burn Rate
Value Metrics	Value Assessment Ratings, Delivered Features

Note: Earned Value Management is not used on the SWP. Instead use Agile progress measures to gauge program performance.

https://aaf.dau.edu/aaf/software/metrics-and-reporting/

Contracting Back-Up Slides

How Contracting for Software Is Different

Instead of a single monolithic contract for the entire program…

Modular Contracting: A portfolio of contracts enabling a program to scale and evolve

Modular Contracting Benefits

- Contract for different types of requirements to get the "right" skills & expertise.
- Mitigate the risk of changing requirements/user needs.
- Enable continuous competition: performance incentives for future dev work (SW dev teams).
- Create opportunities for innovative solution providers to contribute a piece of a larger solution.

Preferred approach for acquiring major IT systems in accordance with 41 U.S.C. §2308; implemented at FAR 39.103

Modular Contracting Considerations

Infrastructure
(Cloud/On-Prem Solutions)

Agile Software Dev Teams
(Coding, App dev)

Dev Environment
(CI/CD pipeline, SWF)

SME Support
(e.g., S/W & DSO Engineers)

https://aaf.dau.edu/aaf/software/contracting-strategy/

Contracting for SW Development Environment

CI/CD Pipeline/Software Factory

- DevSecOps environment for Agile software development
- DoD CIO Enterprise DSO Reference Design documents define specific tools, technologies, constructs, and architectures

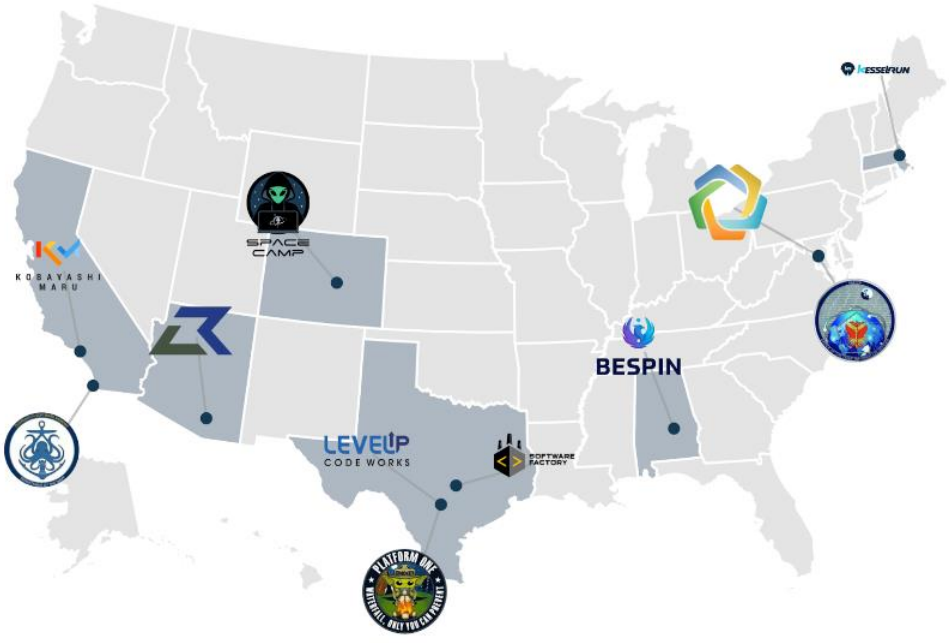
Key Elements

- Automated tools/processes tailored for the program/mission
- Continuous authorization to operate (cATO) (Requires active cyber defense)
- Open API Architecture

Notional Contracting Strategies

- Existing software factories (e.g., PlatformOne, CReATE, Overmatch Software Armory)
- Commercial pipeline solutions (AWS, Azure, Google) through Joint Warfighting Cloud Capability (JWCC) IDIQ
- Commercial Solutions Opening (CSO) to award FAR contracts or other transactions for innovative commercial solutions
- SBIR Phase III awards (sole source to SBIR Phase I/II vendors)
- IDIQ awards

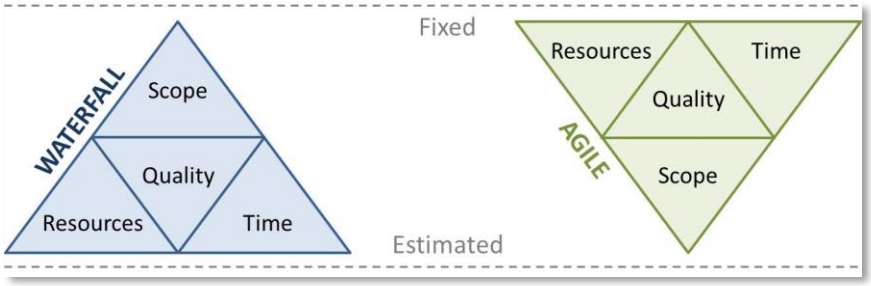
Existing DoD Software Factories



https://coalition.dso.mil/

Contracting for Agile Software Dev Teams 1/3

Traditional (waterfall) development contracts are plan driven; **scope is a fixed** and defined set of requirements that does not change without an engineering change proposal and subsequent contract modification.



Agile software development is characterized by continuous change and a dynamic reprioritization of requirements, a value-driven approach where **cost and schedule are fixed** to allow scope to change.

Traditional (Waterfall) Development	Agile Software Development
Solicitation/Contract Award	
Detailed technical and system requirements are provided in the solicitation. Contractors **propose a technical solution** to meet the required capability. The contract is awarded based on the strength of the proposed **technical solution**	A product vision (**high level vision** of system functionality) is provided in the solicitation. Contractors **propose development teams** to iteratively deliver the capability. The contract is awarded based on the strength of the proposed **software development expertise**.
Contract Execution	
Software is developed in accordance with **fixed requirements** provided in the solicitation and awarded contract. Software is **tested and delivered at completion** of long and linear development phases. Contractor performance is measured using **criteria established at contract award**.	Software is planned and developed in short release cycles in accordance with **dynamic government-managed product backlogs**. Software is **tested as part of the dev process**, demonstrated with users during sprints, and **incrementally delivered** as releases. Contractor performance is measured against **criteria established for each sprint/release cycle** and SW metrics.

Contracting for Agile Software Dev Teams 2/3

Multiple contracts for software development can be executed to develop discrete capabilities of the overall solution managed by the program.

Software Dev Teams

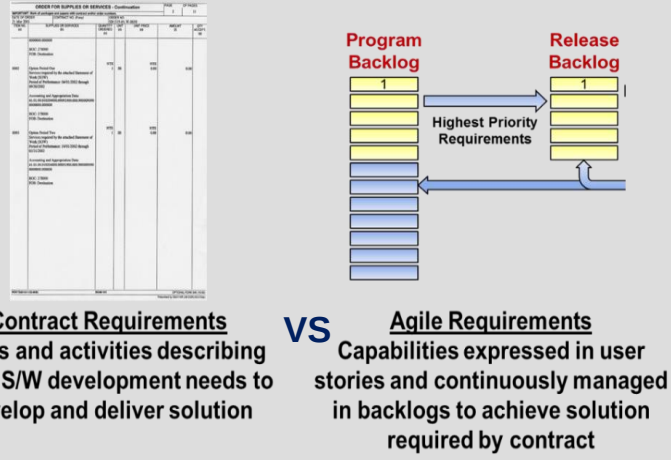
- The govt. has an overall integrator role (or leverage separate integrator contractor); it manages overall DSO and Agile development processes; it also prioritizes backlogs.
- Contractor development teams provide Agile software development expertise.

Key Elements

- Align individual contracts with the architecture, vision, and dev. environment.
- A variety of labor categoriess & functions + sample hours combine to establish ceiling; LCATs can adjust within ceiling on a dev sprint basis without contract modification.
- Collaborative Agile dev process; mutually agreed-upon *definition of done* for each sprint/release; SW metrics for quality of capability delivered.

Notional Contracting Strategies

- GSA Schedule/BPA awards
- Commercial Solutions Opening (CSO) to award FAR-contracts or OTs
- SBIR Phase III awards (sole source to SBIR Phase I/II vendors)
- Small business set asides
- Existing IDIQ awards or establish new IDIQ



Contract for Software Development Expertise vs. Specific Requirements

Contracting for Agile Software Dev Teams 3/3

Contract Types

Contracts for SW development teams have been executed using all contract types.

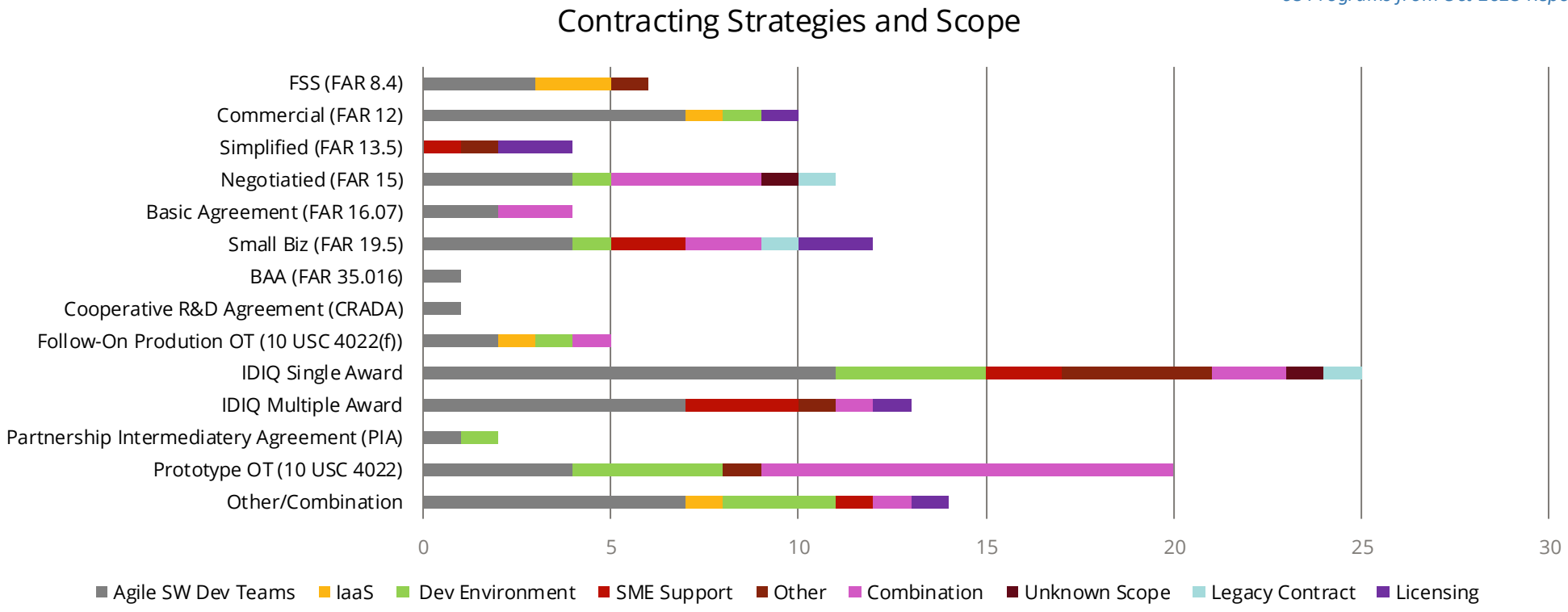
Contract type selection will depend on a variety of factors including the following:

- Selected contract strategy (no CR for FAR 8.4/12; IDIQ contract type limitations)
- Level of government involvement in development activities

	Fixed-Price	Time & Materials	Cost-Reimbursement
Use When	• Estimates can be reasonably identified and estimated, usually by historical costs.	• Development team velocity is unknown. • Government is the integrator.	• The necessary labor hours/labor mix is not well understood. • Not concerned about excluding small businesses without certified systems.
How to Use	• Fixed price level of effort are provided over a specified time period enabling product backlog requirements to continuously evolve.	• Scope increments into task requirements aligned to the product backlog. • Labor rates and ceiling price must be established.	• Establish fixed work cycles and software release cycles to enable government to prioritize work.

SWP Program Contracting Strategies

63 Programs from Oct 2023 Reporting



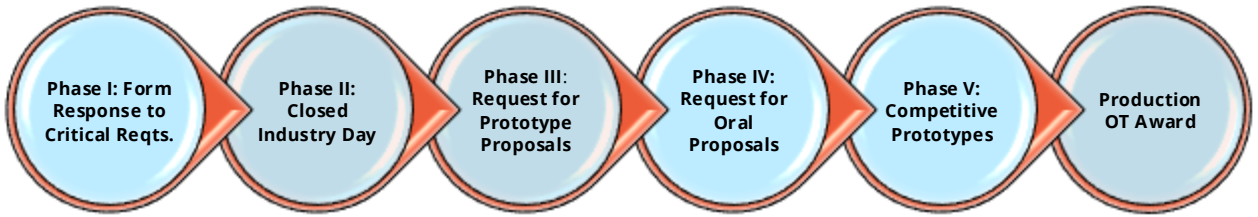
Total = 128 contracts

Leveraging OTs for S/W Dev: An Example

Prototype OT to Follow-On Production OT

Phased competitive process to downselect and award Prototype OT; winner received follow-on Production OT award to develop program solution

6 months



4 questions regarding security clearances *(200+ to 31 vendors)*

Attendance required to advance in the process *(31 vendors)*

10-page tech volume, $1m ceiling *(12 to 11 vendors)*

90-minute briefings *(11 to 2 vendors)*

8-week competitive prototype for a software development process and product *(2 vendors)*

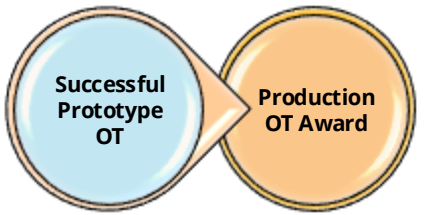
Custom PaaS dev environment, CI/CD pipelines; user app dev; 10-year PoP; $100-500m threshold *(1 vendor)*

1 year

Subsequent Production OT

Competitive OT Prototype process leveraged by a separate program to award subsequent Prod OT award to a vendor not previously selected

2 months



Proposal requested to implement successfully prototyped solution *(1 vendor)*

PaaS + DataOps solution for digital apps; 10-year PoP; $100m threshold *(1 vendor)*

Contracting for Software Solutions Best Practices

Award Timelines

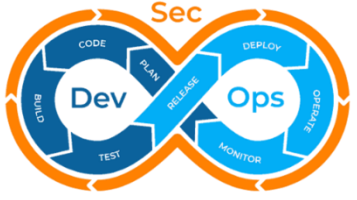
- Select contract strategies that offer streamlined procedures (i.e., existing GSA/BPAs, BOAs, IDIQs, CSOs, OTs, FAR 12, FAR 13.5, SBIR Phase III).

Source Selection Techniques

- Leverage tech demos/challenges and oral proposals to assess vendor solutions, reduce B&P cost/cycle time, and streamline source selection processes.
- Leverage phased evaluations and advisory downselects to reduce the number of proposals evaluated in full.

Deliverables

- Consider CDRL alternatives when appropriate:
  - Leverage DevSecOps environment/processes to deliver data.
  - Data Accession List (contractor data generated during work effort).



https://aaf.dau.edu/aaf/software/contracting-strategy/

Legacy Contracting Constraints

Apply modern software development practices while operating on traditional legacy contracts.

Pursue modular contracting for different elements of SW and HW development vs. looking to the legacy contractor to do everything; what can be contracted separately to enable SW dev (i.e., dev environments, dev teams) and that doesn't conflict with the legacy contract scope?

Active discussion with contractor management team on intent for modern SW practices.

Shift to collaborative govt./contractor processes vs. contractor processes does their work and briefs the program at quarterly PMRs.

Actively engage with the contractor (daily/weekly) to lead prioritization of development work and be in the loop on progress/challenges.

Explore contract mods where it makes sense to pivot to or add CLIN types specific for software development (assuming the contractor has skilled modern SW developers).

Contracting for Innovative SW Solutions

Increase opportunities for non-traditional contractors (NDCs) to Provide critical capabilities.

Embrace modular contracting (in conjunction with open architectures & APIs) to create opportunities for innovators to contribute individual capabilities to larger program objectives.

Actively find & engage NDCs & shape strategies to enable broader interest and participation:

- DoD Innovation Ecosystem organizations
- OT Consortia
- SOCOM Vulcan Platform
- LinkedIn

Address barriers to entry for NDCs:

- Prioritize contract strategies enabling expedient source selection procedures to address NDC cash flow challenges.
- Prioritize contract types not requiring a Cost Accounting System and other compliance barriers for NDCs.
- Leverage demos/challenges to reduce NDC bid & proposal investments.
- Leverage CSO procedures and OT awards for flexibility to leverage commercial terms & conditions.

Software Acquisition Pathway Primer for Senior Leaders

Agile Back-Up Slides

Carnegie Mellon University
Software Engineering Institute

Key Agile Terminology 1/3

Cadence: *The fixed duration of the Agile team's development cycle (i.e., the length of a Sprint)*

- *Multiple teams should all adopt the same cycles so that the entire system can move quickly.*

Daily Scrum/Daily Stand-Up: *A short daily meeting to assess progress in the current sprint*

- *Each member of the team describes what they did yesterday, what they will do today, and any blockers.*

Sprint Planning: *A short meeting where the team determines which backlog items will be worked this Sprint*

- *This meeting is held at the start of the Sprint. The "rule of thumb" is that it should be no more than 1 hour per week in the Sprint.*

Sprint Demonstration: *Held at the end of the Sprint where the team demonstrates what has been accomplished*

Sprint Retrospective: *Held at the end of the Sprint where the team reflects on what went well, what can be improved, and the behaviors to change for the next Sprint*

Program Increment Planning: *A large event where an overall plan for the increment is developed and socialized across all teams and stakeholders*

Key Agile Terminology 2/3

Feature: *A larger unit of work, though desirably completed within a single increment*

Acceptance Criteria: *A collection of criteria by which you can determine if the work item (e.g., user story, epic, or feature) has the correct functionality*

Definition of Done: *The additional set of activities deemed necessary to complete to produce a piece of work (e.g., code scanning or document review)*

Minimum Viable Product (MVP): *Often confused with MVCR, but should be enough working software to prove out an idea (e.g., enough code to test the architecture)*

Backlog: *A prioritized list of tasks to complete to support a strategy*

Story Points: *A measure of the difficulty and effort needed to implement a User Story*

Sprint Velocity: *A measure of how much work a team can complete during a Sprint*

Burndown Chart: *Shows the amount of work remaining and the time needed to complete it*

Cumulative Flow Diagram (CFD): *Sometimes called a "sand chart"; representation of all work and its current state; developments should show work arriving at the same rate as it is completed*

Key Agile Terminology 3/3

Product Owner: *The customer representative who defines units of work (e.g., user stories, epics, and features) and prioritizes the Backlog*

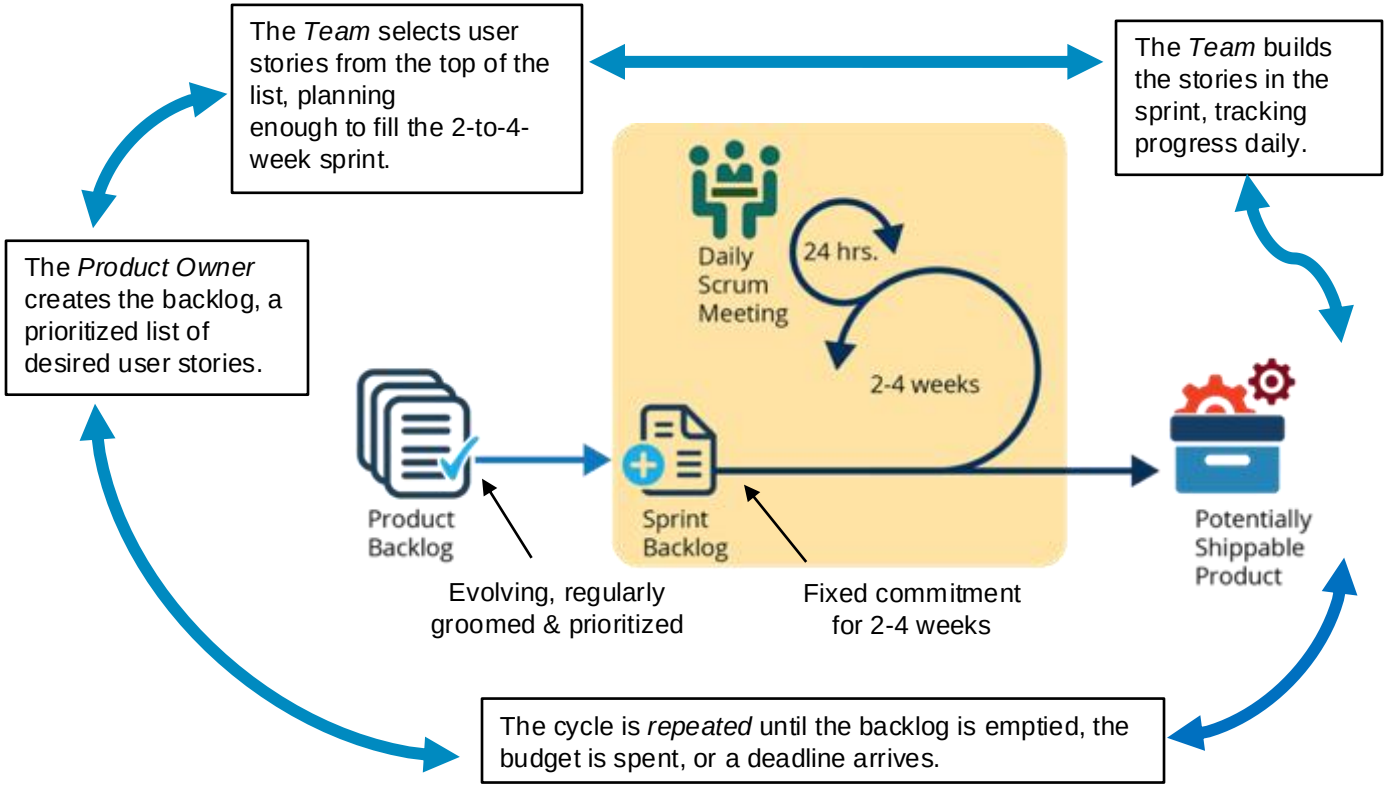
Scrum Master: *A team member who facilitates the team and is the person tasked with finding ways to remove blockers to work (by discussion with other teams or escalation as needed)*

Continuous Integration: *Where new code changes are frequently integrated into the code base and tested*

Single Baseline: *A strategy in which all artifacts (especially code) are maintained in a single baseline and all work is, when committed, merged into that baseline*

Agile at the Team Level: Scrum

Scrum is an iterative, incremental methodology for managing Agile software projects.



Interpretation of Agile Principles in Government 1/3

Agile Principle	Useful Interpretations in Government Settings
The highest priority is to satisfy the customer through early and continuous delivery of valuable software.	In government, the “customer” is not always the end user. The customer includes people who pay for, people who use, people who maintain, as well as others. These stakeholders often have conflicting needs that must be reconciled
Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.	Rather than saying “competitive” advantage, we usually say “operational” advantage. This principle causes culture clash with the “all requirements up front” perspective of many large, traditional approaches.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	What it means to “deliver” an increment of software may well depend on context. With large embedded systems, we are sometimes looking at a release into a testing lab. Also, for some systems, the operational users are not able to accept all “deliveries” on the development cadence because there are accompanying changes in the workflow supported by the software that require updates.
Business people and developers must work together daily throughout the project.	In government settings, we interpret “business” people to be end users and operators as well as the other types of stakeholders since, in many government settings, the business people are interpreted as the contracts and finance group.

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Interpretation of Agile Principles in Government 2/3

Agile Principle	Useful Interpretations in Government Settings
Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.	A frequent challenge in government is to provide a suitable technical and management environment to foster the trust that is inherent in Agile settings. Allowing teams to stay intact and focused on a single work stream is another challenge.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	In today's world, even in commercial settings, this is often interpreted as "high bandwidth" rather than only face-to-face conversation. Telepresence via video or screen sharing allows more distributed work groups than in the past.
Working software is the primary measure of progress.	Our typical government system development approaches use *surrogates* for software—documents that project the needed requirements and design—*rather than the software itself*, as measures of progress. Going through small batches in short increments allows this principle to be enacted, even in a government setting, although delivery may well be a test environment or some internal group other than the users themselves.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	This principle is a caution against seeing agility just as "do it faster." Note that this principle includes stakeholders outside of the development team as part of pacing.

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Interpretation of Agile Principles in Government 3/3

Agile Principle	Useful Interpretations in Government Settings
Continuous attention to technical excellence and good design enhances agility.	This is a principle that often is cited as already being compatible with traditional government development.
Simplicity—the art of maximizing the amount of work not done—is essential.	One issue with this principle in a government setting is that our contracts are often written to penalize the development organization if they don't produce a product that reflects 100% of the requirements. This principle recognizes that not all requirements we think are needed at the onset of a project will necessarily turn out to be things that should be included in the product.
The best architectures, requirements, and designs emerge from self-organizing teams.	Note that the principle does not suggest that the development team is necessarily the correct team for requirements and architecture. It is, however, encouraging teams focused in these areas to allow some autonomy to organize their work. Another complication in many government settings is that we are often re-architecting and re-designing existing systems.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	This principle is an attempt to ensure that “lessons learned” are actually learned and applied rather than just being “lessons written.”

Source: SEI Congressional testimony July 14, 2016 to House Ways and Means Committee.

Key Enablers for Agile Adoption

Carnegie Mellon University Software Engineering Institute

Acquisition Processes

- Collaborate: industry, acquirers, and users
- Enabling changes
- Rapid contract action
- Acquiring developer services versus product

Culture and Policies

- Small teams
- Fail fast/learn fast
- Delegated decisions
- Review SW, not docs.
- Continuously improve
- More execution rigor

User Involvement

- Active users involved
- High bandwidth comm.
- Demo interim sprints
- Provide ops. insights
- Prioritize requirements

Program Structure

- 6-12 month releases
- Tailor acq. processes
- Stakeholder buy-in
- Empowered teams
- Small iterative releases

Aligning Priorities

- Align program docs., processes, contracts
- Leverage loosely coupled architecture
- Rethink reviews

Agile Training

- Requires experienced gov't. and contractors
- Invest in training team
- Coaches working with PMO to implement when to use Agile

Source: Adapted from 2016 briefing to General E. Pawlikowski on USAF Agile Adoption, SEI & MITRE

Barriers to Agile Adoption

Acquisition Processes

- Long timelines
- Fully defined requirements up front
- Contract mods. costly

Culture and Policies

- PMOs struggle to tailor acquisition processes
- Change = risk
- Significant oversight

User Involvement

- Limited engagements
- Few end users available
- Serial requirements process (ops.→tech.)
- Limited demos late

Program Structure

- Up-front fixed scope
- Locked requirements
- Too detailed cost est.
- APB, EVM management
- Changes discouraged

Aligning Priorities

- Many stakeholders with competing priorities
- Conflicting developer direction, interpretation
- Disrupts team progress

Agile Training

- Limited insight and experience in Agile in gov't., defense industry
- False claims of Agile
- Need for leadership, culture, process, staff

Source: Adapted from 2016 briefing to General E. Pawlikowski on USAF Agile Adoption, SEI & MITRE

Key Aspects of Agile and DevSecOps Culture

Here are the key elements of a technical culture that support Agile & DevSecOps:

- Emphasize **speed of decision making** and communications.
- Make a **more transparent culture** that proactively raises problems and keeps aspirations separate from reality.
- Promote an **environment of openness and access** between employees and senior leadership.
- **Empower technical staff** and consistently consult them in decision making .
- Invest in a culture of leadership/management where **autonomy and accountability are balanced**.
- **Push decision-making down the command chain** and track/optimize decision-making velocity.
- Encourage employees (and protect them) when **raising controversial ideas or questions**.
- An “ombudsman” mechanism assures anonymity to receive, collate, and **act on grievances**.
- A **“badgeless” culture** in programs minimizes distinctions between government and contractors.

Help institute and support a culture that promotes hiring and retention, and makes DevSecOps work even better.

Similar Engineering Tasks with Different Perspectives

Traditional Approach		Agile Approach
Attempts to accurately predict the execution plan	➡	Believes that predictability is impossible
Fixed scope, time, cost prior to execution	➡	Fixed time & cost, but scope adapts for new learning
Measures success by adherence to the plan	➡	Measures success by delivery of working product
Values methodology and processes over people	➡	Values and empowers people over all else
Resists change in requirements and process	➡	Expects requirement changes and process improvement
Value and progress is measured by documentation (*there's no software until the end*)	➡	Progress is measured by delivery of the working product and value by customer feedback after use
Customer engagement is limited to the beginning (*requirements capture*) and end (*testing/QA*)	➡	Customer is regularly engaged to evolve the product (what the customer values guides investment/effort)
"Lessons learned" captured at end of project	➡	Frequent reviews/retrospectives to continuously improve

Leadership for Agile and DevSecOps

Carnegie Mellon University Software Engineering Institute

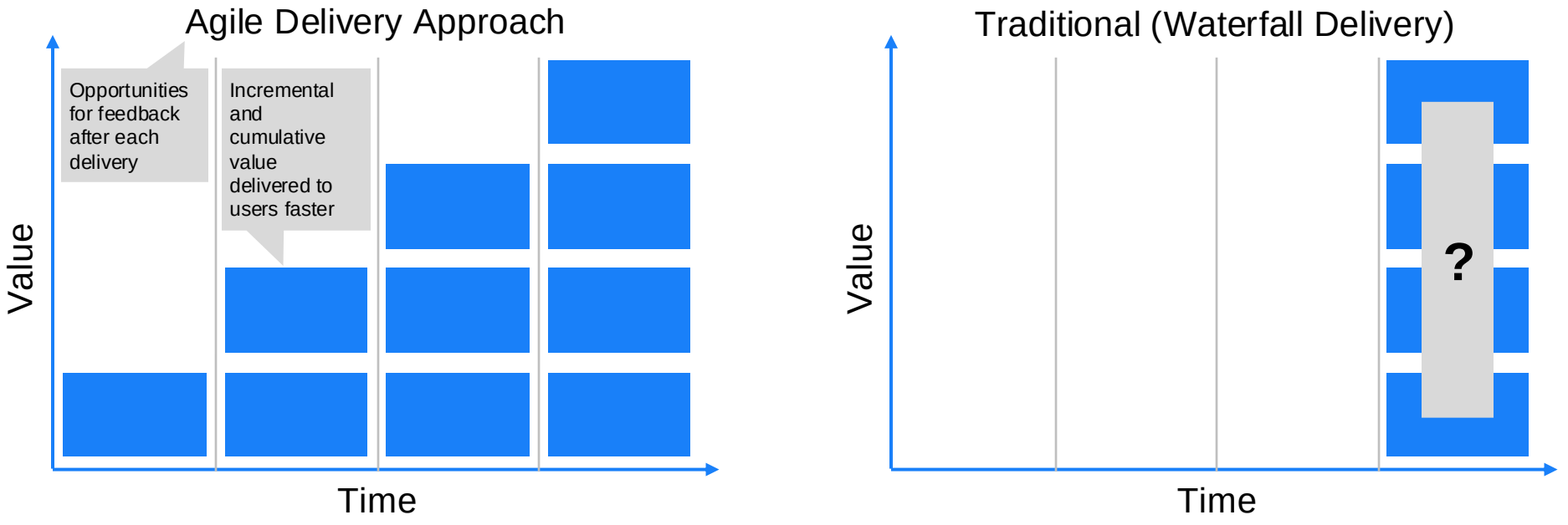
Understand that leading DevSecOps experts needs a different leadership style:

- Exhibit a **culture of meritocracy** with "*strong opinions loosely held.*"
- **Collect information asynchronously** using synch tools & ChatOps versus weekly reviews.
- **Only hold to roadmaps for months**—then change course as needed based on data.
- Encourage no more than **one meeting per day** and encourage (*polite*) dissension.
- Model behaviors of **transparency, risk-tolerance, humbleness, and inclusion**.
- Model a visible desire to show they are **learning new technical and acquisition skills**.
- **Empower the team to make decisions**, from novices to experts.
- "**Lead from the back of the room**," allowing experts to debate and lead, and showing support for the debate process.

Embracing the Agile and DevSecOps cultures is challenging, but it will produce significant benefits in the speed of mission d elivery.

Agile Value Accumulates Over Time

Agile delivers value incrementally, allows us to incorporate new learning and customer feedback that maximizes future value and gives us multiple near-term reflection points to make data-driven micro-investment decisions.



This is in contrast to predictive/waterfall where requirements are locked in early (when we know the least), there is no value until "big bang" value is delivered at the very end (after all investment is spent), and investment decisions are large and long-term.

Leadership of Agile Must Ask Different Questions

Selected Key Questions

- Are we learning as fast as we need to so that we will mitigate downstream software-related risks?
- Are we avoiding unnecessary technical debt?
- Is the early software showing positive signs of meeting the user's needs?
- Do we have sufficient and appropriate end user (surrogate) involvement?
- How are our SPO staff adapting to small-batch interactions? What, if any, changes in battle rhythm are necessary in the SPO to make Agile more effective/useful for us?
- Is decision making decentralized in a productive way?

Cheat Sheet of Agile/Lean “Facts”

Agile is….a philosophy, a set of tenets and principles that supports multiple implementation methods/approaches.

Agile is not …a single set of practices applied in a “one size fits all” way.

The most adopted set of Agile practices for small (mostly software) teams is…. **Scrum**.

The most adopted set of Agile/lean practices for larger system developments is…. **SAFe (Scaled Agile Framework)**.

The most adopted scaling framework by DoD contractors is…. **SAFe**.

SAFe is… **lean engineering philosophy and approach** coming from the top of the enterprise **meets Agile approaches** from the team level of the enterprise.

The most cited factor in failure of Agile adoption (regardless of government/commercial) is … **culture mismatch**.

Technology adoption approaches **in common with adoption of other types of practices** apply well in Agile adoption settings.

Agile engineering practices inside a waterfall acquisition environment cause multiple, serious challenges.

Key SEI Messages Related to Agile (Your “Cheat Sheet”) 1/2

Carnegie Mellon University Software Engineering Institute

Key SEI AIG Messages

- Agile is not a Silver Bullet, but it can be useful in Government development settings.
- Agile is principles based, which means there isn't only "one way" to do it correctly; look to the principles before evaluating an implementation.
- Lean engineering is closely related to Agile at the principles level, and scaling frameworks for Agile can benefit from lean principles as their basis.
- Acquiring contracted systems with contractors who use Agile implies a different set of relationships and approaches to achieving oversight and insight.
- OSD acquisition guidance (i.e., 5000.02) doesn't prohibit Agile, although it never mentions Agile specifically.
    - There are three model diagrams in 5000.02 that can be readily tailored to an Agile IT environment or an Agile weapon systems environment.
- Any contracting approach can be used with Agile if done with awareness of how Agile would work in that setting; however, some key elements of contracting (regardless of contract type) must be attended to.

Key SEI Messages Related to Agile (Your "Cheat Sheet") 2/2

- Bringing Agile into an acquisition office is a classic technology adoption problem, and we use classic technology adoption solutions to help organizations through it.
- SAFe (Scaled Agile Framework) is the most frequently adopted scaling framework by defense contractors; that's why we teach it, not because it is always the best for every problem.
- You may have "hard" or "soft" adoptions of SAFe; it is meant to be tailored. With soft adoptions (only using some of the principles, not calling it SAFe, combining it with other approaches) you may not get all the benefits of the framework, but you may still get enough to make the shift worthwhile to the program.
- Most acquisition organizations don't run a software development shop themselves. (There are exceptions, of course.) But understanding the basics of Agile team practices and lean system development practices is needed in the acquisition office so that appropriate oversight decisions can be made.
- Scrum is the most frequently adopted Agile team practices framework; however, much of the work done in the SPO environment may be more amenable to a Kanban approach, if the SPO wants to "do" Agile, not just oversee it.
- Measurement in an Agile context has to be defined and implemented carefully to avoid incentivizing unproductive team behaviors.
- SPO leadership who are engaged with Agile developments need to ask some different questions when evaluating the progress/success of Agile for a team or at scale.

Software Acquisition Pathway Primer for Senior Leaders

DSO Back-Up Slides

Carnegie Mellon University
Software Engineering Institute

Continuous Delivery

Continuous delivery is the process of preparing software artifacts that are built and tested using continuous integration pipelines and made available for release to environments for further test or production. The release could still be a manual decision, but the software is in a deployable state.

- Deliver software as often as possible.
- Deploy software when there is a need.
- Containerization or similar technologies are a great mechanism to deliver software.
  - It contains dependencies that other software artifacts may require from external sources.
  - It is simpler to scan and analyze with security tools.

Continuous Deployment

Continuous deployment is a step up from Continuous Delivery, where software is deployed to production without manual intervention.

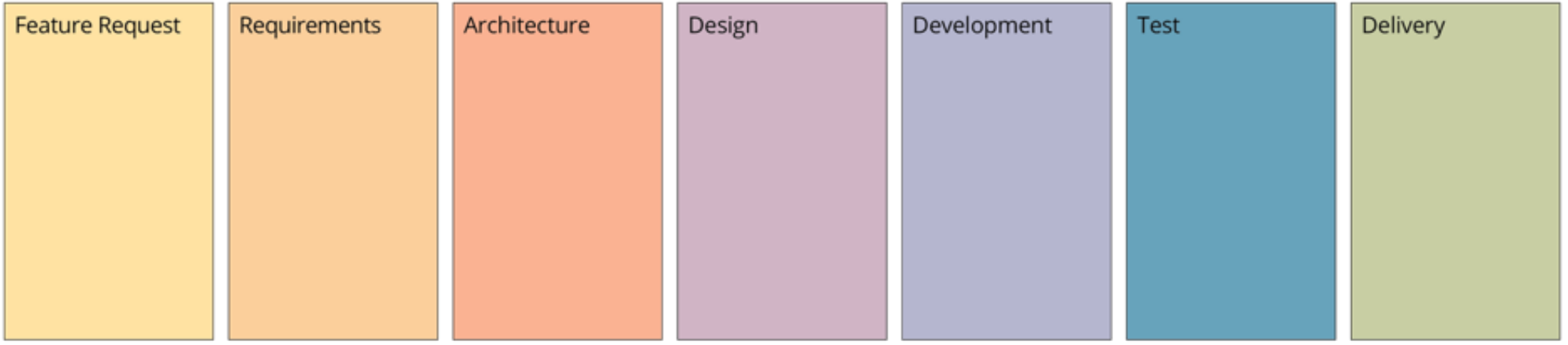
- This requires a high level of automation, testing, and monitoring to have high confidence in the stability and reliability of the software.
- Not every type of software system should be continuously deployed; rather, it is based on user requirements and business impact.

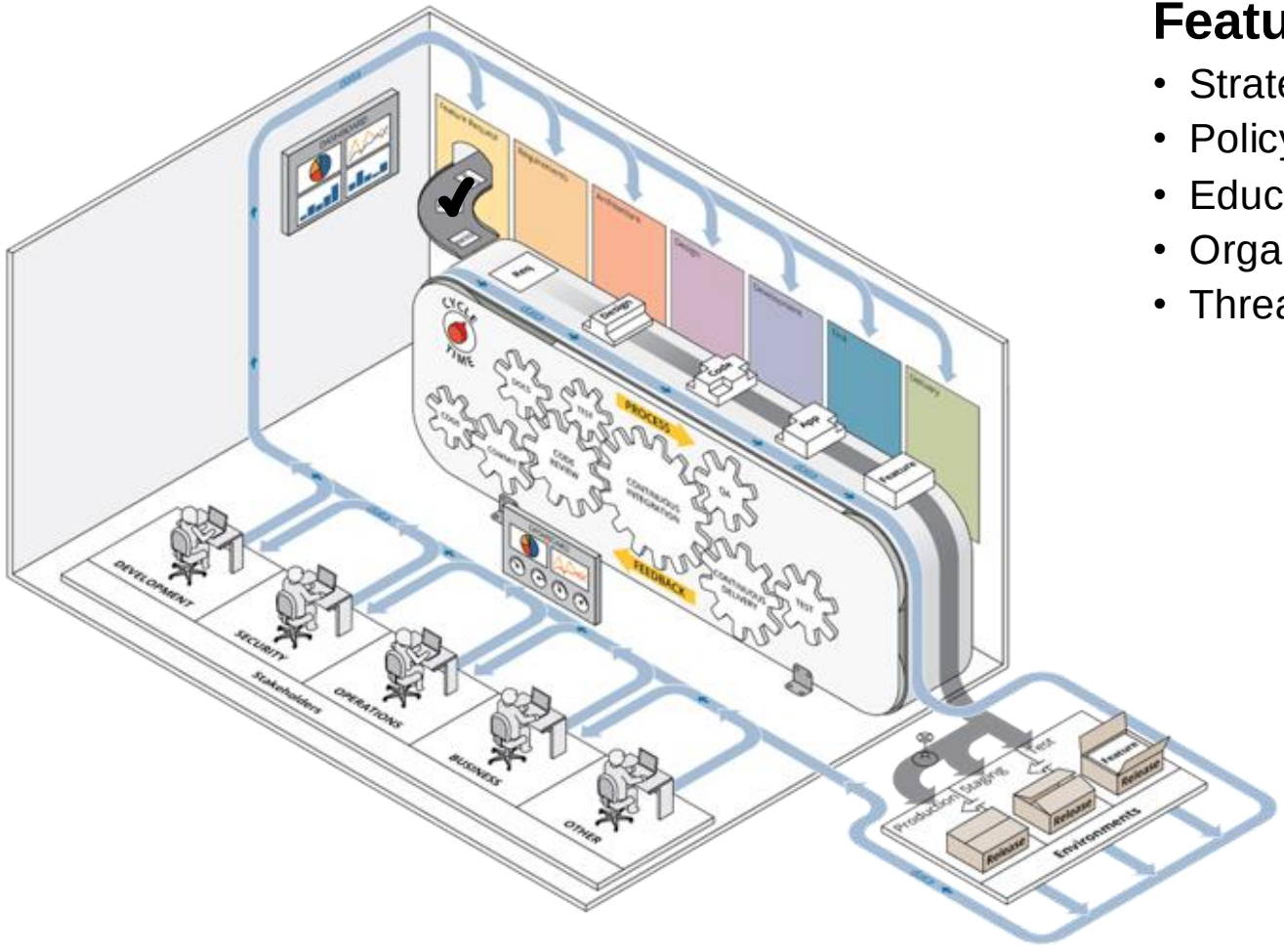
Continuous Monitoring

In DevSecOps, Continuous Monitoring is the practice of observing and analyzing a system's security, performance, and compliance in real time.

- Collect data from every tool in the DSO pipeline.
- Monitor data based on metrics that are meaningful to your organization.
- Use alerts or notifications to make continuous improvements to the system.
- It often requires custom tool creation and integration with existing sources of data (Infrastructure/pipeline engineers).

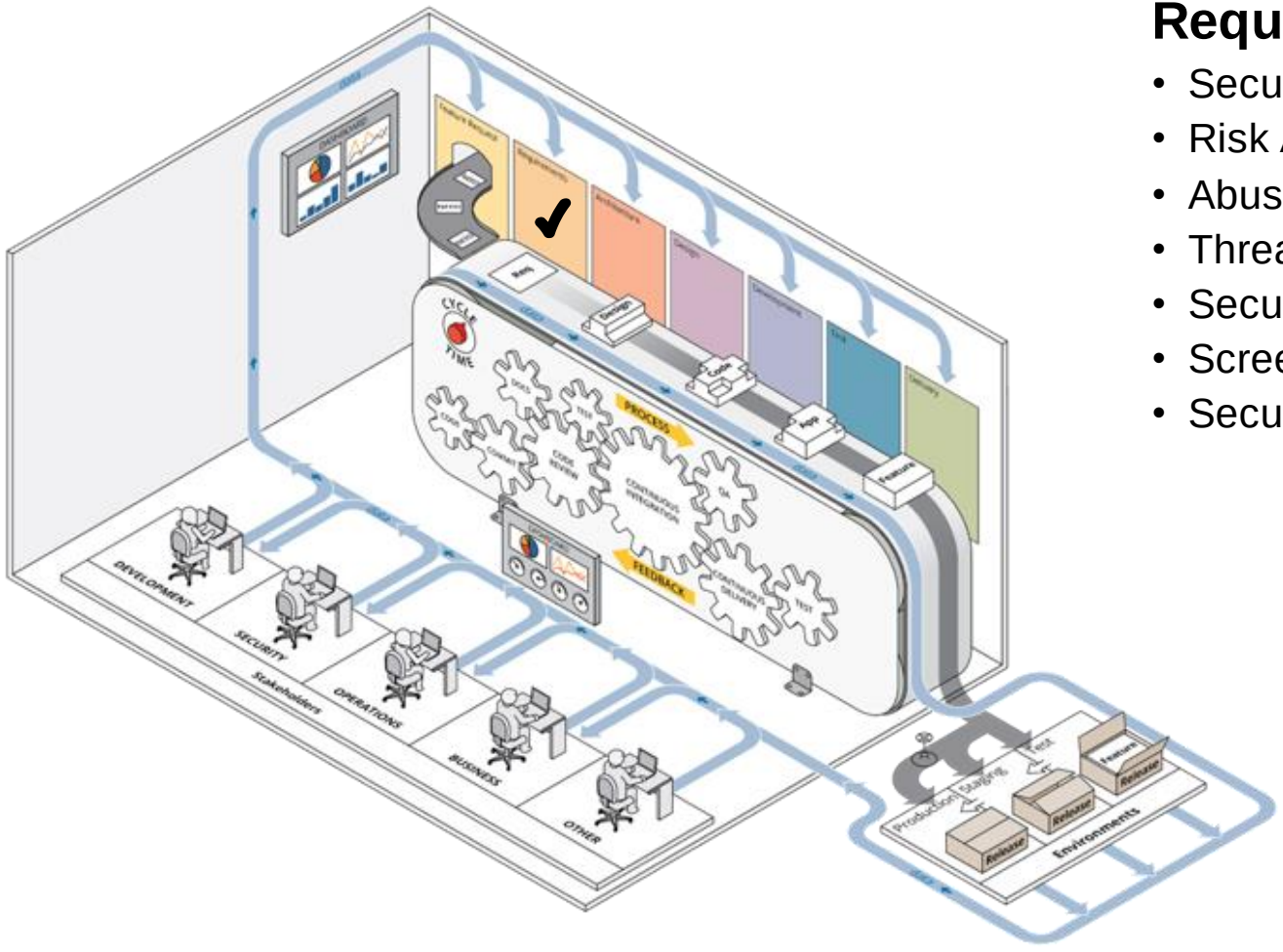
System Development Phases





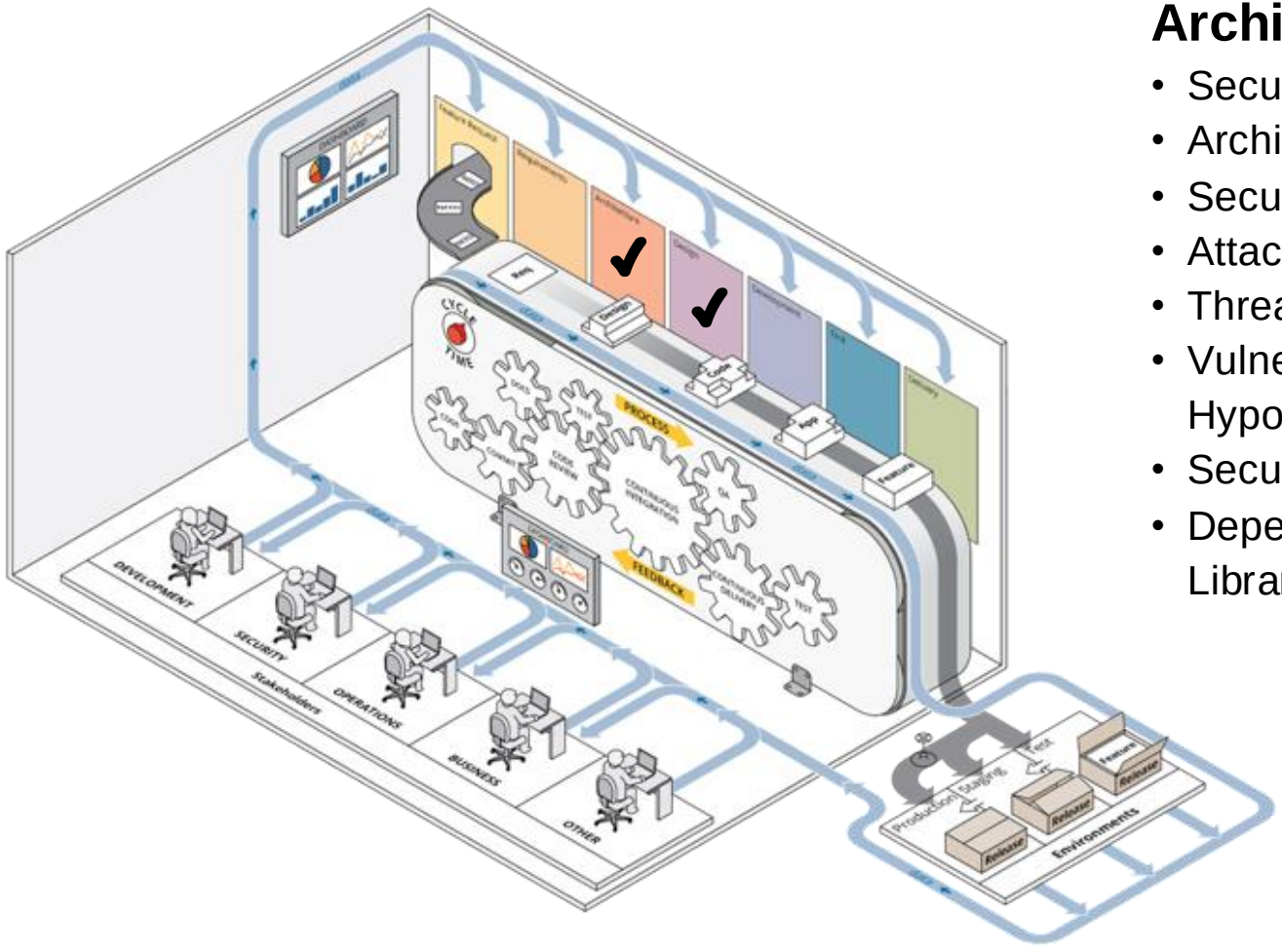
Feature Request

- Strategy & Metrics
- Policy & Governance
- Education & Security Guidance
- Organizational Risk Factors
- Threat Assessment



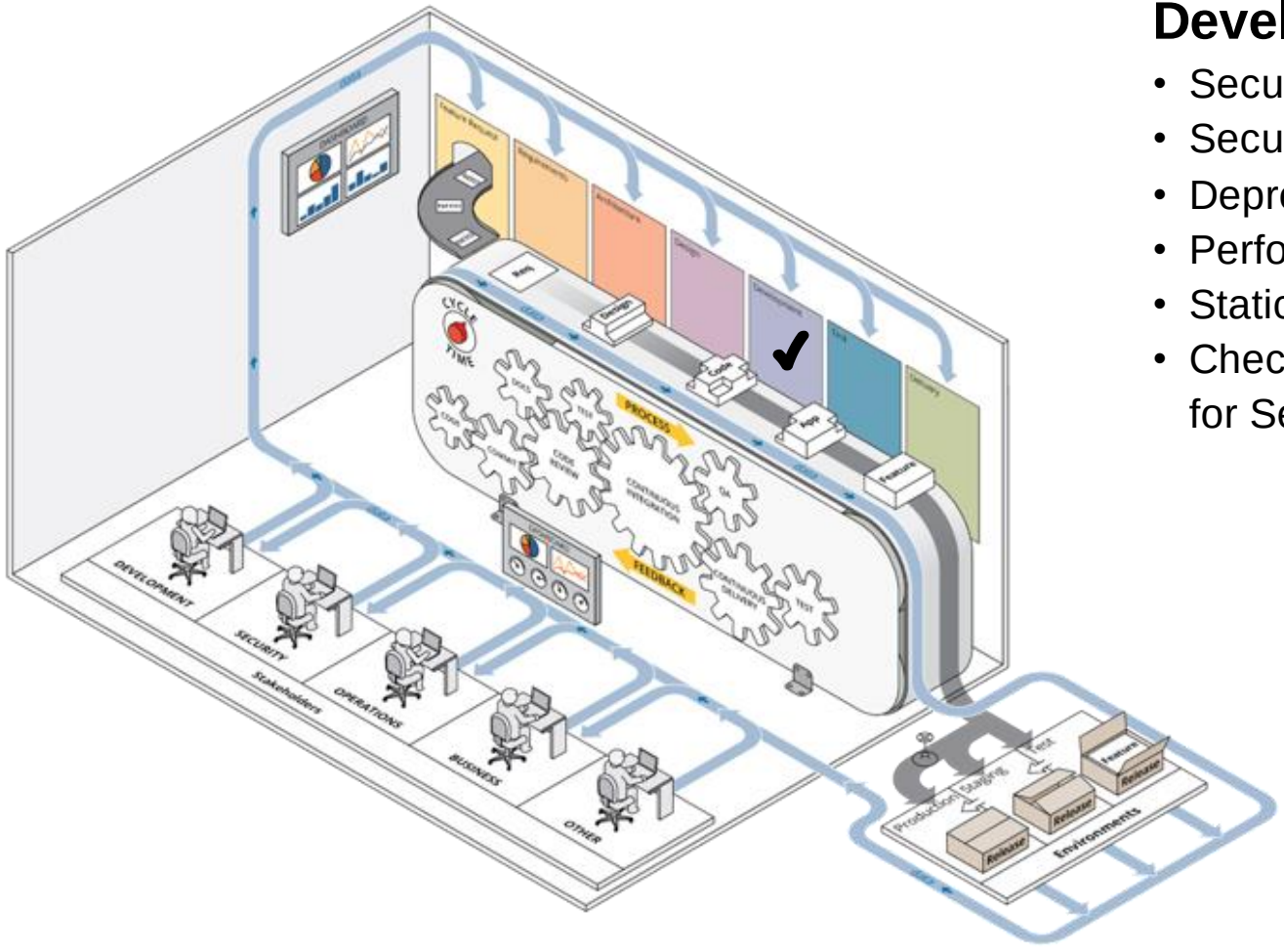
Requirements

- Security Requirements (SFR/SAR)
- Risk Assessment
- Abuse Case Development
- Threat Modelling
- Security Stories
- Screen Development Tools
- Secure/Hardened Environments



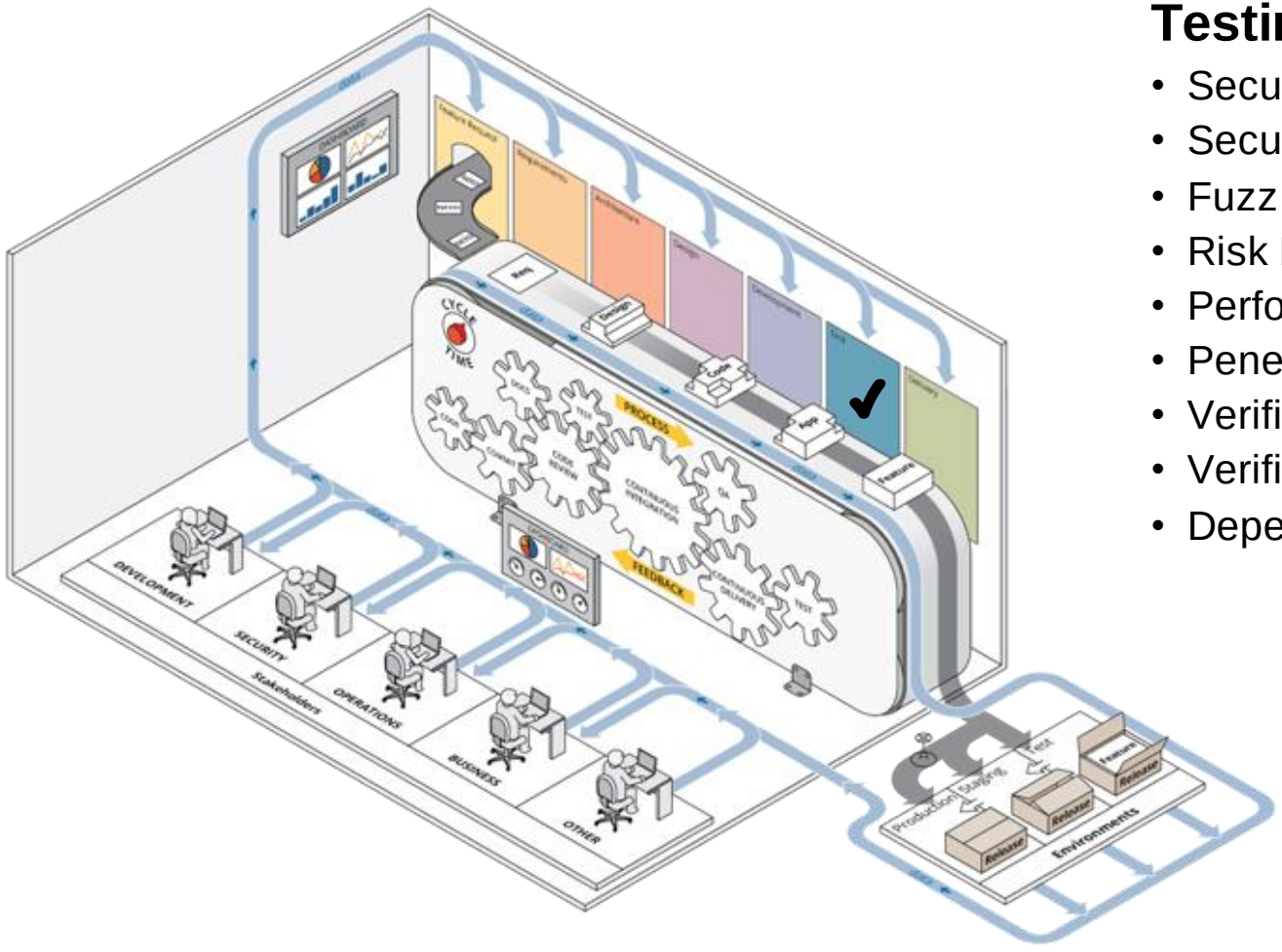
Architecture & Design

- Security Architecture
- Architectural Risk Analysis
- Security Design Requirements
- Attack Surface Analysis
- Threat Modelling
- Vulnerability Analysis and Flow Hypothesis
- Security Design Review
- Dependencies List, Open Source Libraries



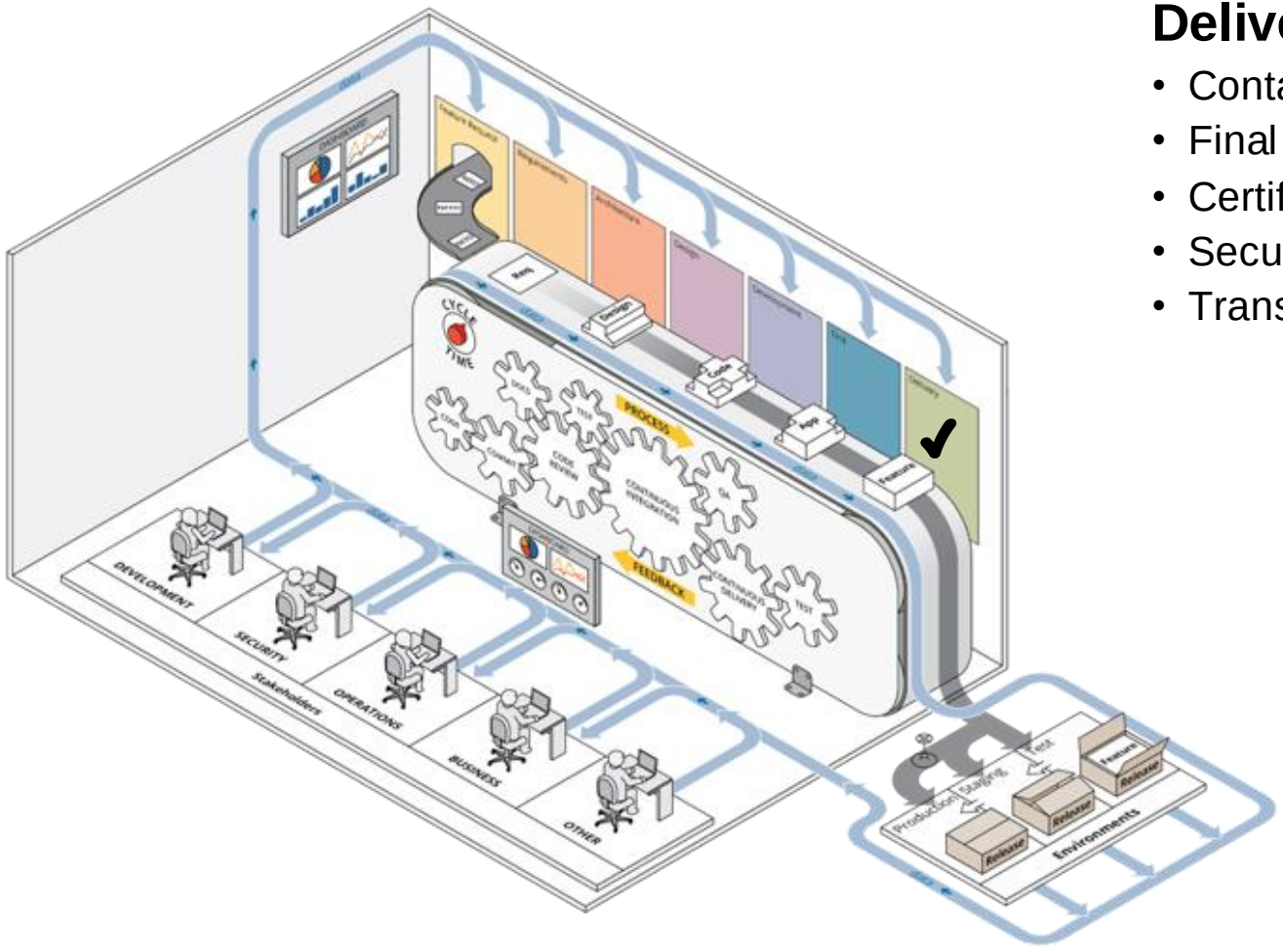
Development

- Secure Coding Practices
- Security Focused Code Review
- Deprecate Unsafe Functions
- Perform Security Unit Testing
- Static Code Analysis
- Checking of Process and Procedures for Secure Coding & Traceability



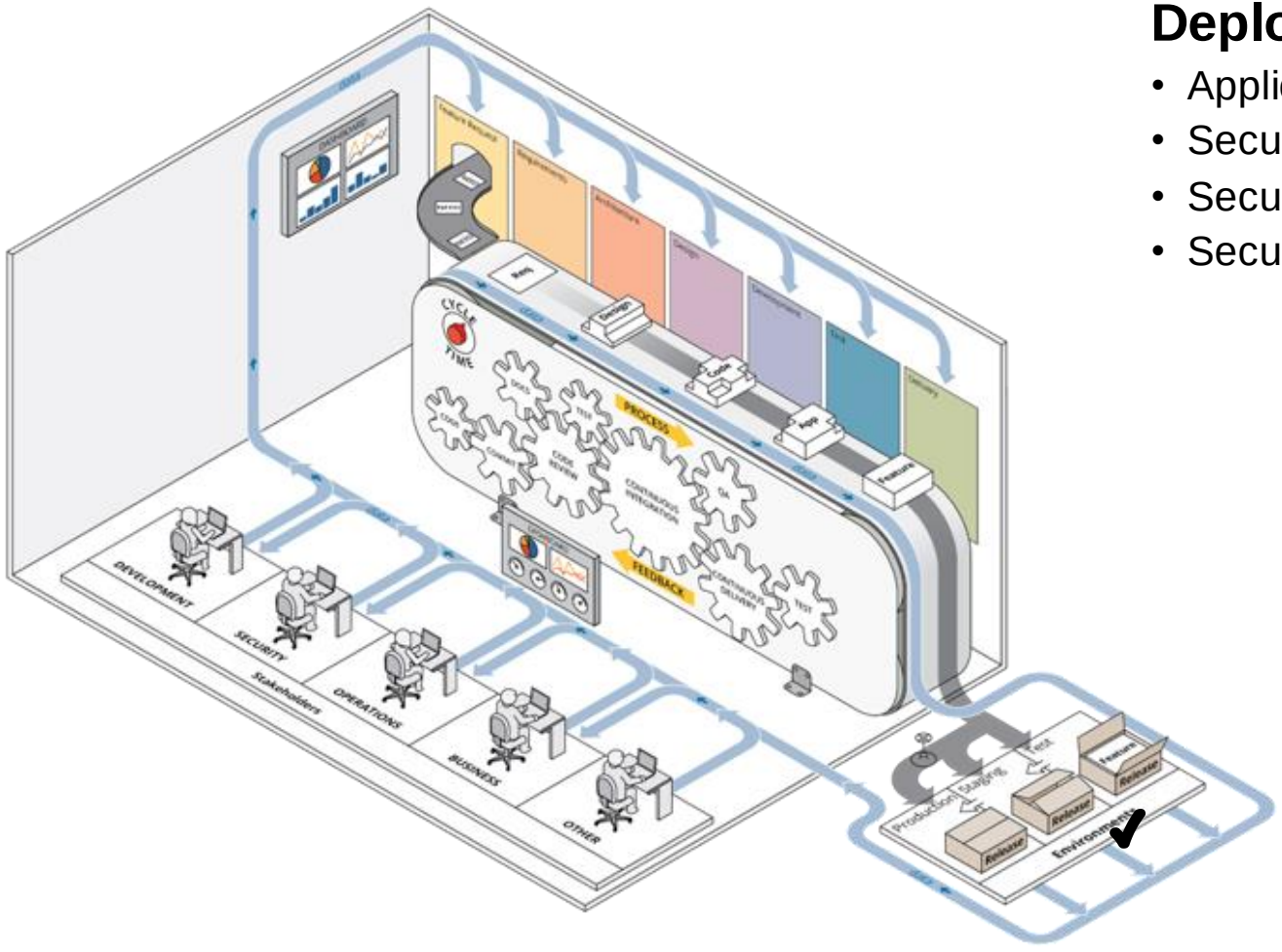
Testing

- Security Test Planning
- Security Testing
- Fuzz Testing
- Risk Based Security Testing
- Perform Dynamic Analysis
- Penetration Testing
- Verification of Security Implementation
- Verification of Process and Procedures
- Dependency Monitoring



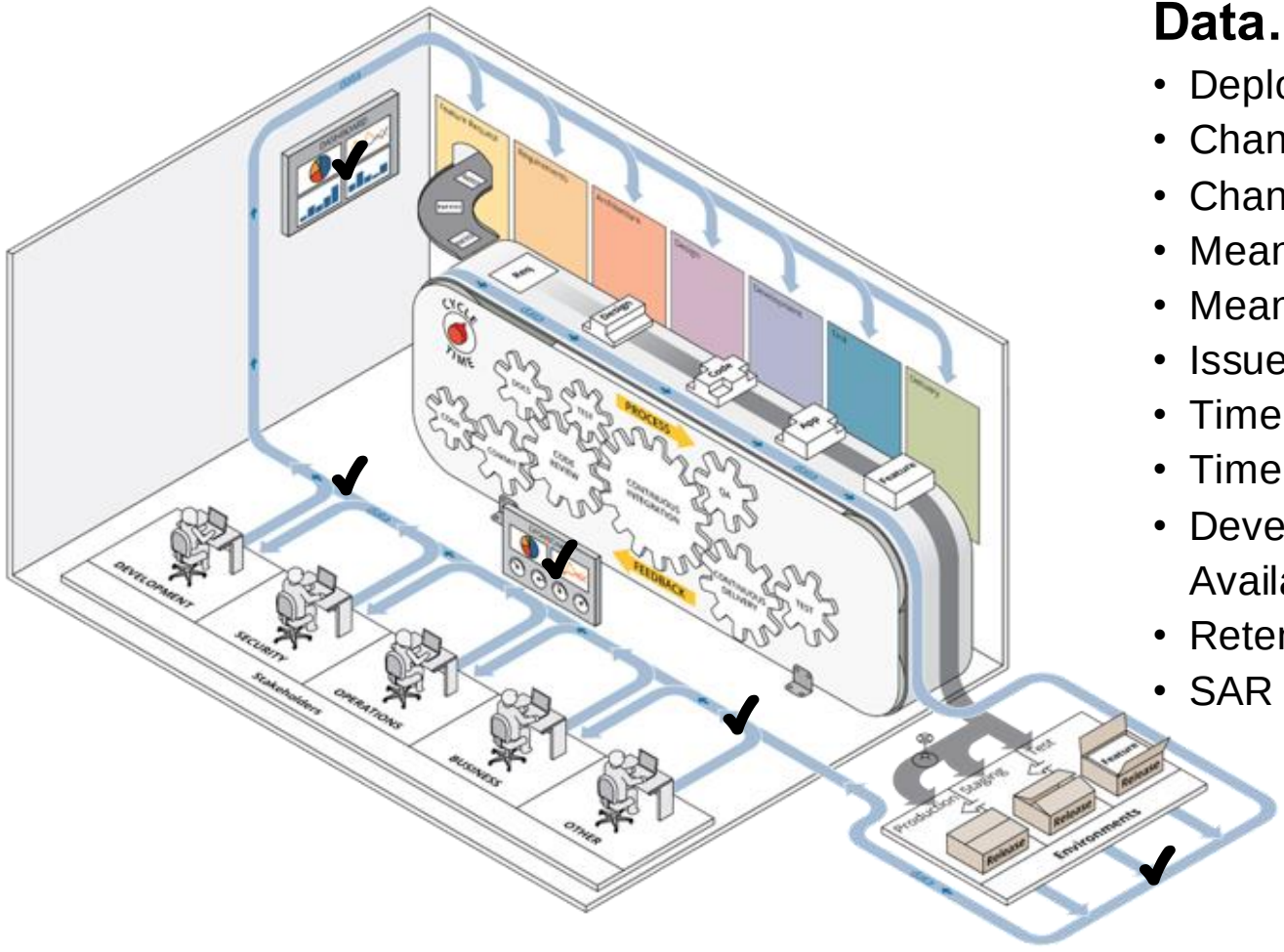
Delivery

- Container Security
- Final Security Review
- Certify, Release, and Archive
- Security Acceptance Testing
- Transition Incident Response Plan



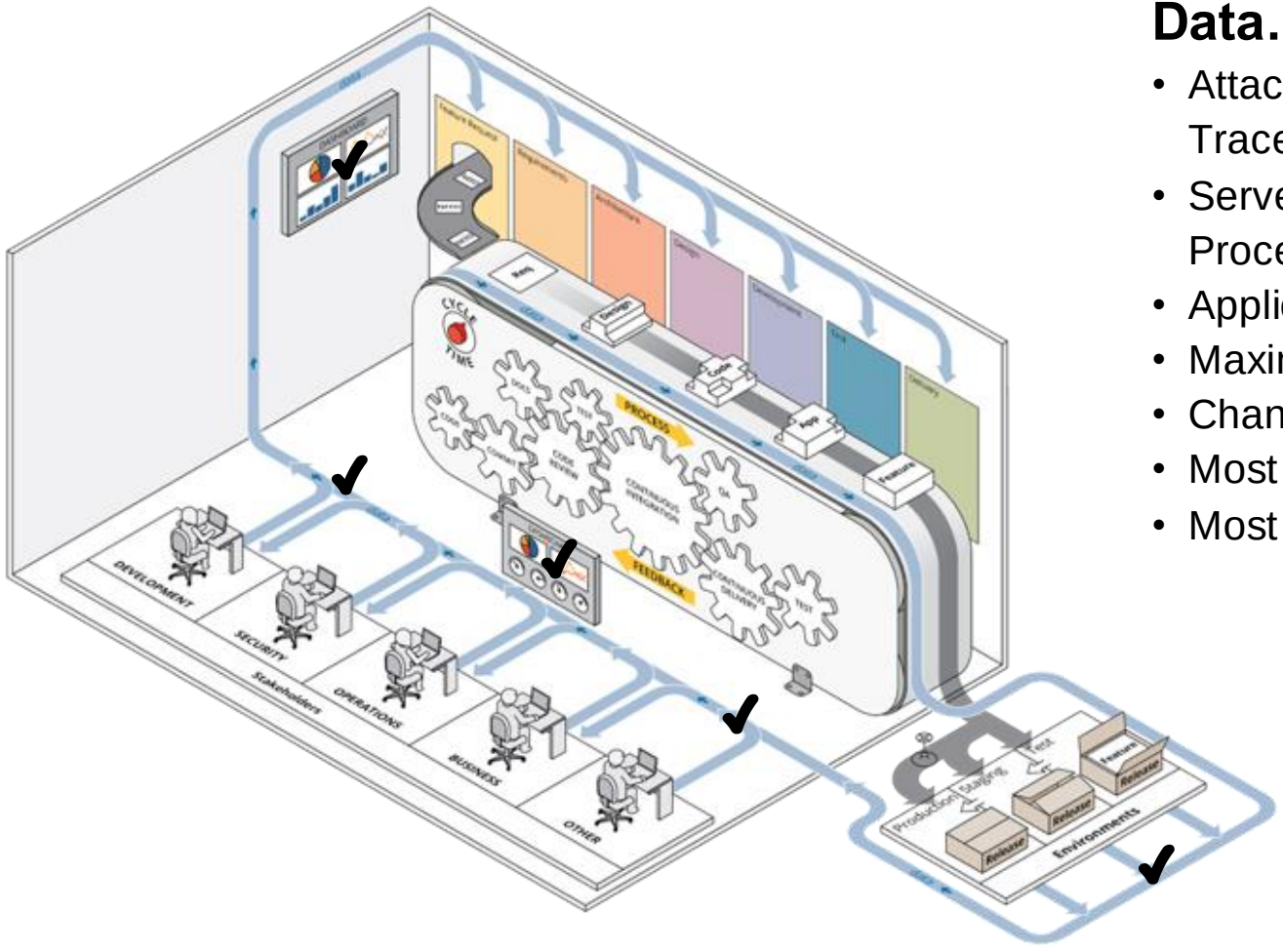
Deploy

- Application Security Monitoring
- Secure Deployment Process
- Secure Environment
- Secure Operational Enablement



Data…

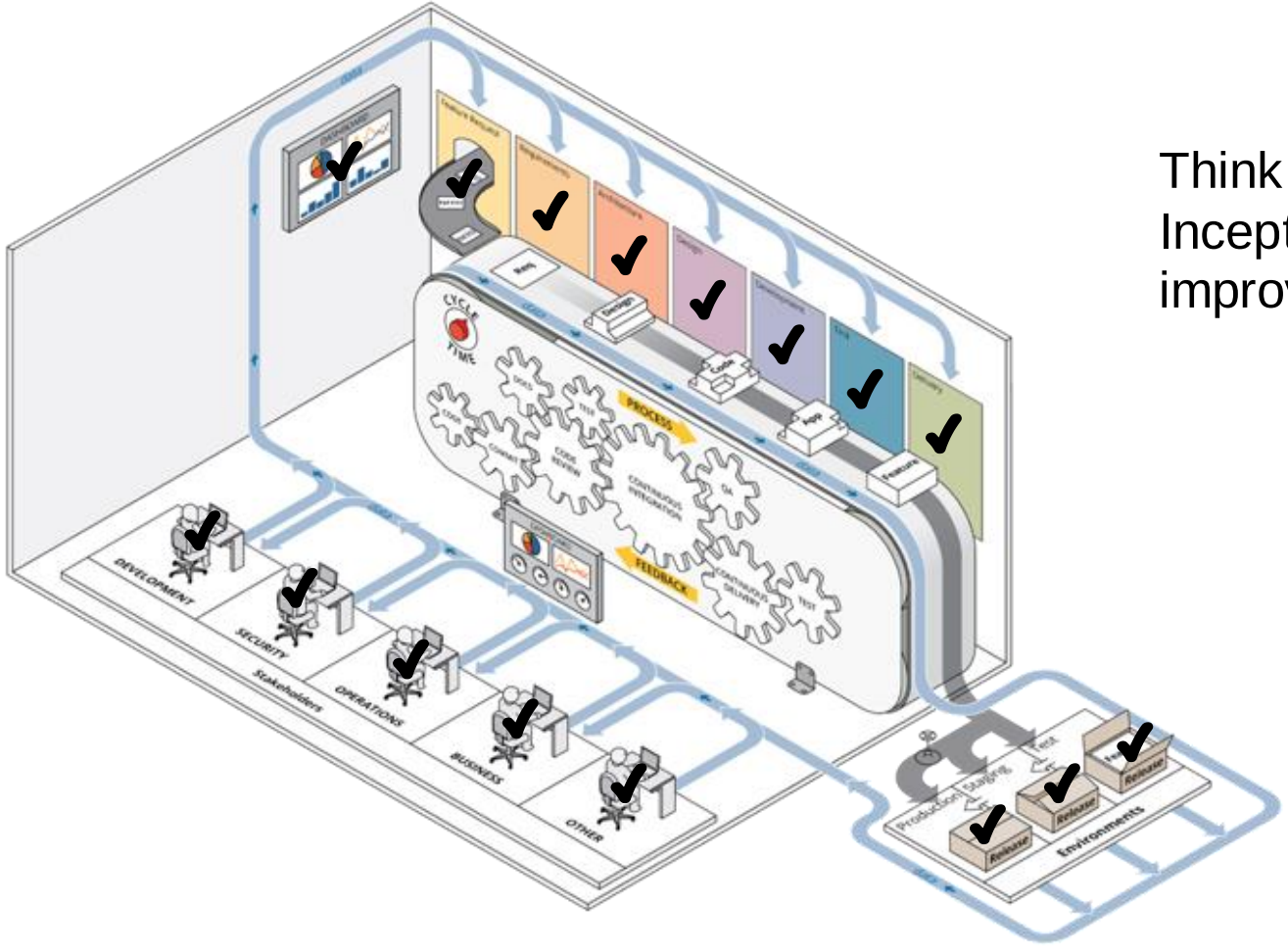
- Deployment Frequency
- Change Lead Time and Volume
- Change Failure Rate
- Mean Time to Recovery (MTTR)
- Mean Time to Detection (MTTD)
- Issue Volume and Resolution Time
- Time to Approval
- Time to Patch Vulnerabilities
- Development and Application Logging Availability
- Retention Control Compliance
- SAR Findings



Data…

- Attack Vector Details (IP, Stack Trace, Time, Rate of Attack, etc.)
- Server Disk Space, Load, and Process Monitoring
- Application Performance
- Maximize Monitoring
- Change in Size to Code Base
- Most Active Code Contributors
- Most Changed Code Areas

Security Considerations at Every Phase



Think Security from Inception to Deploy and improve every delivery.

ATO: Authority to Operate

What is an ATO?

- An ATO is **Authority to Operate**:
  - Authorizes the system to be placed on a production network
  - Interfaces with other components within the DoD
  - Authorizes access by end users to leverage these resources to execute the mission
- Key staff in the ATO process include the following:
  - AO (Authorizing Official)
  - ISSO (Information System Security Officer)
  - Security Assessor
- An AO makes a risk-based decision (**RMF NIST 800-37**) to grant an ATO for use of the system.
- The decision has to be formalized in an ATO letter.
  - An ATO letter must explicitly state the AO's acceptance of the following:
    - Use of the system at the Agency at the determined FIPS 199 impact level
    - All leveraged external services supporting the system
    - Any exceptions or exclusions of the Chief Security Officer (CSO) for use at the Agency

Software Acquisition Pathway Primer for Senior Leaders

Concepts Back-Up Slides

Carnegie Mellon University
Software Engineering Institute

Requirements

SWP

- Requires Agile requirements—high-level CNS or SW-ICD only—should be living documents.
- Can use existing documents if transitioning from another pathway and they meet the need.

Agile

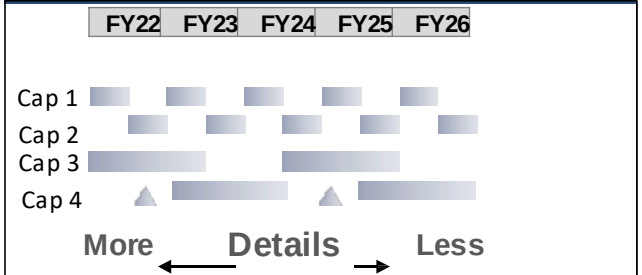
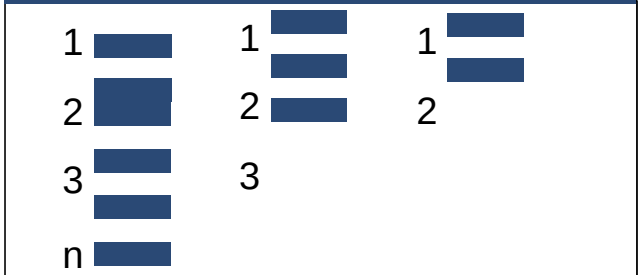
- Accommodates changing requirements (customer changes, new threats, or based on new learning).
- Demos should be provided at the end of each sprint or increment to get user feedback.

DSO

- CI/CD pipelines are used to continually deliver software for further testing and demonstration of capability.
- Tooling promotes collaboration and transparency of processes and data.

Requirements & Software Planning

Build, Measure, Learn: Evolving Mission, Adoption, Performance, Threats, Priorities, Tech

Overarching Requirements	Product/Program Roadmaps	Product/Program Backlogs
Capability Needs Statement (CNS) OR Software Initial Capabilities Document (SW-ICD)		

Overarching Document

Scope of users' operational needs, threats, environments

Flexibility in 2–5-year outlook

Visual Plans for Next 18 Months to 5 Years (FYDP)

Foster stakeholder collaboration, plans, investments, priorities, interdependencies, etc.

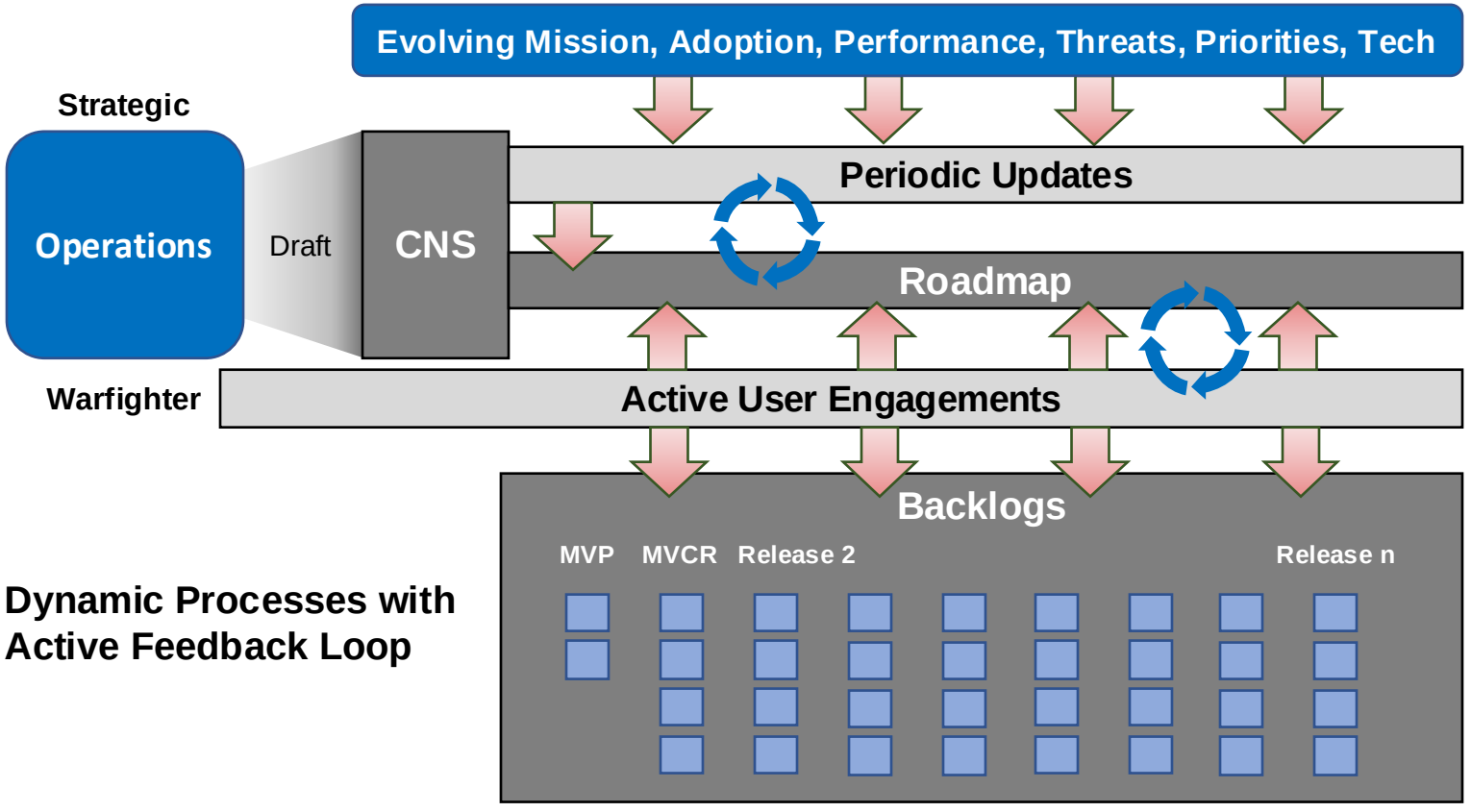
Dynamic, Prioritized Lists of Tactical User Needs

Each release focuses on highest priority needs.

Performance & user feedback shape future iterations.

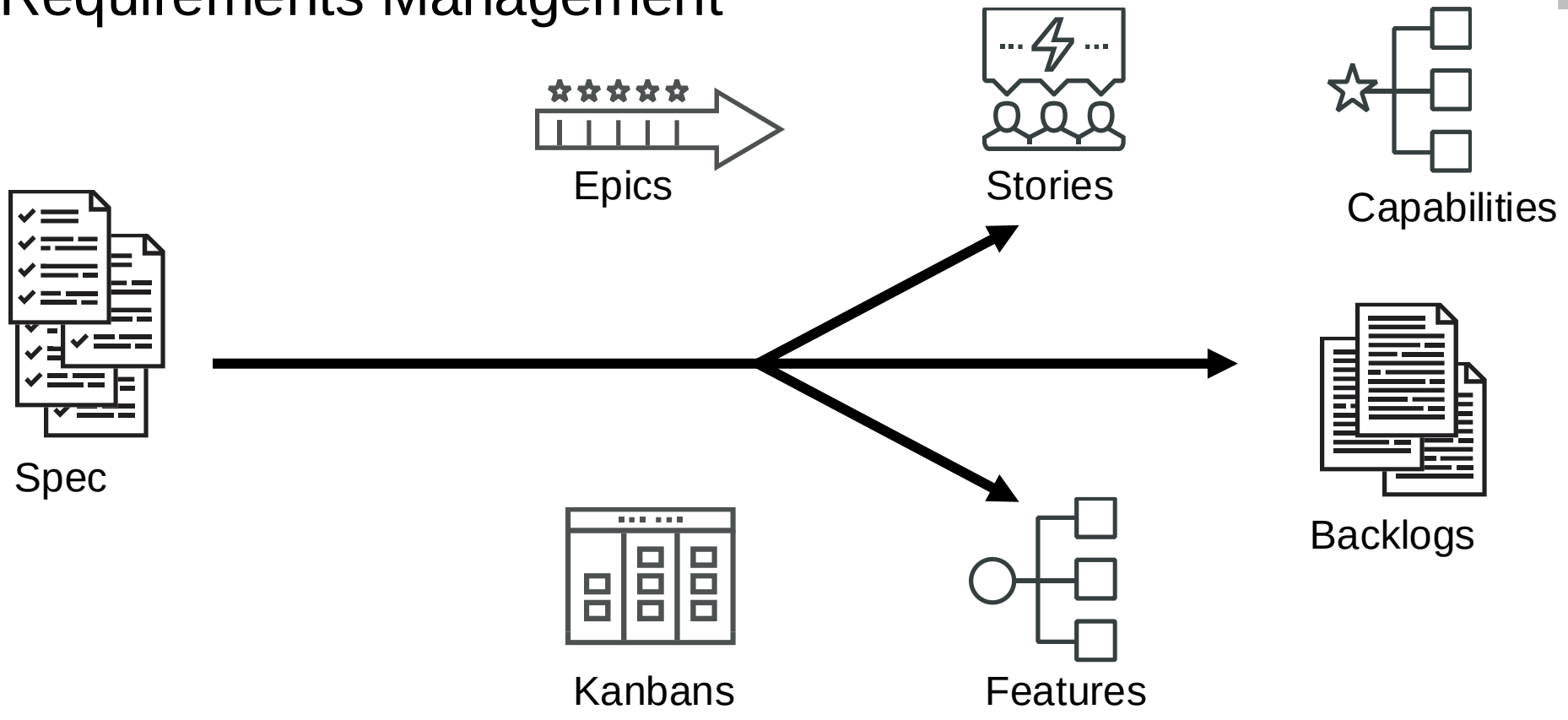
Active Sponsor/User Engagement Early and Throughout Development

Evolving Software “Requirements”



Dynamic Processes with Active Feedback Loop

Agile Represents a Major Change in Requirements Management



DevSecOps Mechanism for Requirements

DevSecOps environments should support tooling for requirements management and traceability.

- Traceability from requirements to a code commit is a goal to strive towards
- Requirements management systems should interconnect with other tooling such as:
  - Issue Tracking
  - CI/CD pipelines
  - Dashboards for visualization of metrics

T&E Key Challenges

Challenge	Way Ahead
TEMP Process	Move from TEMP to Software T&E Strategy with just-in-time test planning: ➢Test and Evaluation Strategy is developed and approved in months. ➢T&E procedures use a standardized automated test framework. ➢Capability T&E is developed concurrent with capability. ➢Approval is as low as possible (documented in User Agreement).
CDD/IS-CDD -> Capability Needs Statement	Deploy incremental capabilities that enhance operations: ➢ Move from testing a KPP to demonstrating capability.
Shift T&E Left	Move DT and OT stakeholders into the development process and the pipeline: ➢Use automated testing to demonstrate functionality and stability. ➢Use Chaos Engineering and other fuzz techniques to establish usability/stability in reduced timeframes. ➢Ability to release on the completion of the pipeline.
Shift OT Right	Establish continuous monitoring of operations to allow better operational assessment of capabilities as well as shape future investments and enhancements: ➢Use operational monitoring to supplement (shorten) OT. ➢Instrument the applications to learn the operational environment and use it. ➢Allow field experience to inform development.

User Agreement

Agreement Between the Operational and Acquisition Communities

Active User Involvement

Ensure users from across the ops community actively support development.

Provide ops insights and user feedback on interim and fielded software.

Resource co-location, travel, or virtual ceremonies, events, and meetings.

Informed Decision Making

Clear understanding of roles and responsibilities

Capturing and prioritizing requirements

Shaping roadmaps and backlogs

Integrating user inputs for value assessments

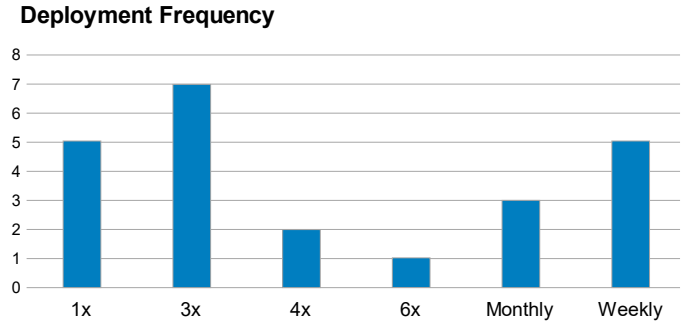
https://aaf.dau.edu/aaf/software/user-agreement/

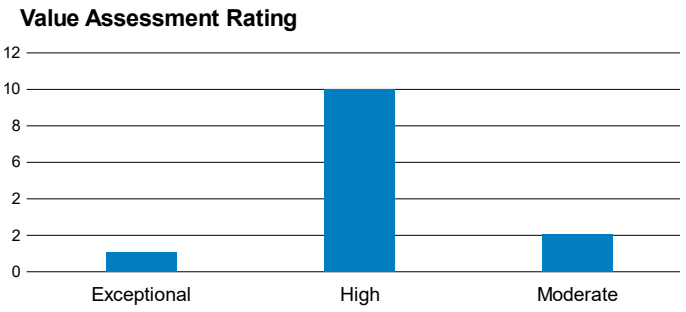
SWP Value Assessments

- Outcome-based assessment of mission improvements, performance, and efficiencies realized from delivered software
- The sponsor and users shall perform a value assessment at least annually
- Formal report card to complement active user engagements
- Validate software deliveries worth the investment
- Use to shape strategies, designs, requirements, and investments
- Focus on **maximizing value** to the Warfighters

https://aaf.dau.edu/aaf/software/value-assessment/

Metrics





- SWP programs tailor a set of metrics to capture progress, quality, efficiencies, cyber, value—maximize timely, automated metrics.
- SWP programs provide insight metrics to OUSD(A&S) semi-annually to assess health and trends of the SW pathway.
- Value Assessment by sponsors/users is a priority measure.

https://aaf.dau.edu/aaf/software/metrics-and-reporting/

"Speed and cycle time are the most important metrics for managing software." – Defense Innovation Board

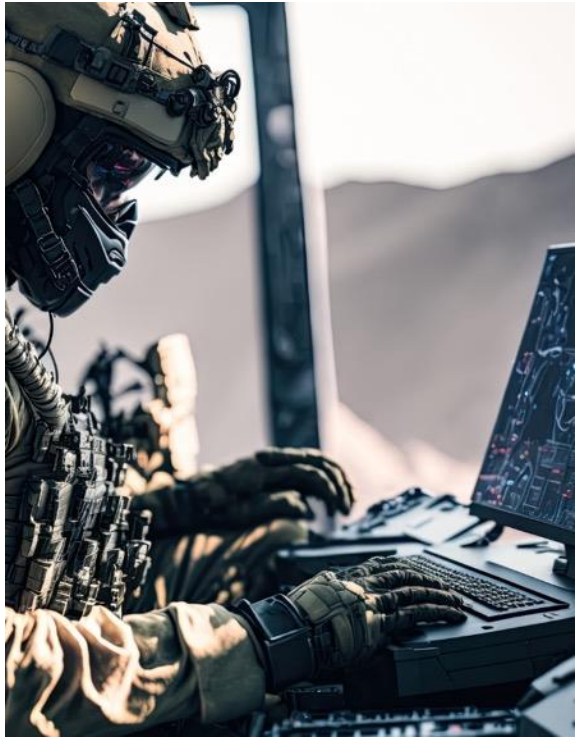
Software Pathway Reporting Metrics

Cyber Resilience Metrics
Avg. Lead Time for Authority to Operate
Continuous Authority to Operate in Place
Mean Time to Resolve Experienced Cyber Event
Mean Time to Experience Cyber Event

Performance Metrics
Avg. Deployment Frequency
Avg. Lead Time
Minimum Lead Time
Maximum Lead Time
Avg. Cycle Time
Change Fail Rate
Mean Time to Restore
Value Assessment Rating
Executive Summary from Last Value Assessment

Additional program information and qualitative questions are also included in semi-annual reporting.

DoD Depends on Software But Does Not Control Development



Software and system complexity is increasing software cost and vulnerability and jeopardizing military capability.

- The DoD does not produce most of the software it uses, but it must maintain that software.
- There are more and more capability results from software, and it will evolve for the lifetime of a system.
- Latent cyber vulnerabilities, those exposed during operations, and those due to underlying dependencies are putting the DoD at risk.
- Finding and fixing problems late causes rework and drives up costs.
- Software cost overruns are overwhelming program delivery and sustainment.

Modern software development and automated tools are critical.

SWP and Contracting

- Smaller, shorter-term acquisition and contracting
- Modular, adaptable contracting
- Contracting for development capacity not product
- Role of the integrator
- Ownership of data, IP, and the means of production (e.g., software factories) to avoid vendor lock
- Competitive collaboration approaches
- Metrics and data drive frequent investment and contractor decisions

Government Insight into Progress

EVENT	TIMING	PARTICIPANTS	COMMENTS
Backlog Prioritization	Prior to every increment	PMO, Sponsor, Users, Contractor	• Must get user and sponsor input. • Balance user requirements with infrastructure & tech debt needs.
Increment Planning	Prior to every increment	PMO, Sponsor, Users, Contractor, Developers, Testers	• Balance user requirements with infrastructure & tech debt needs. • Take developer & test availability into account.
Sprint Demos	End of each sprint	Contractor, PMO	• Review intermediate progress.
Increment Demos	End of each increment	Contractor, PMO, Users, Testers	• Review increment delivery • Planned versus delivered • User Input
Increment Retrospective	End of each increment	Contractor, PMO, Users, Testers	• What went well • What to improve