SYM-AM-24-073

Excerpt from the Proceedings of the Twenty-First Annual Acquisition Research Symposium

Acquisition Research: Creating Synergy for Informed Change

May 8–9, 2024

Published: April 30, 2024

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

Innovation in Software Acquisition: The Good, Bad, and Ugly

Jeffrey Dunlap, CAPT USN (Ret.)—joined the Department of Defense Management at Naval Postgraduate School in 2019. Before that, he was in the Defense Industrial Base for 8 years after retiring in 2011 from military service. While in the military, CAPT Dunlap was an Acquisition Professional and the Program Manager for several software-intensive systems in development. Operationally, he was a major department head on LHA-1 and CVN-76 during major software upgrades and installations. Dunlap holds master's and bachelor's degrees in engineering. [Jeffrey.dunlap@nps.edu]

Abstract

The Department of Defense (DoD) has recognized that interconnected warfighting systems are vulnerable due to their inability to swiftly adapt to new technologies and effectively combat advanced cyber threats. The commercial sector has developed methods to rapidly and securely implement new software capabilities without significant interference to current operations. Transitioning from entrenched practices, such as exhaustive requirement reviews and protracted capability deliveries, to a more iterative and continuous deployment model poses challenges. Value Stream Management has emerged as a means to identify and address inefficiencies, such as silos and bottlenecks, that hinder the prompt delivery of capabilities to the edge of friction. The initial step towards fostering a culture of innovation and enabling the successful flow of capabilities is identifying and eliminating unnecessary delays and legacy obstacles. A coordinated effort within the DoD is necessary to ensure successful innovation in software acquisition. This effort must include identifying and modifying counterproductive organizational behaviors, empowering lean practices, and employing adaptive change management to increase delivery velocity.

Executive Summary

The Department of Defense Instruction (DoDI) 5000.87 introduces a modern Software Acquisition Pathway (SWP) to simplify the acquisition of software-centric applications. However, the Department of Defense (DoD) still faces obstacles in fostering innovation and achieving the desired outcomes within the middle tier and major capability acquisition pathways due to entrenched business practices (slow and bureaucratic) and remnants of legacy certification processes. There can be resistance to change within any large organization, and the DoD is no exception. Cultural inertia can impede modernization initiatives and complicate the adoption of novel software processes and architecture. Many DoD software-intensive systems are decades old and cannot respond to rapidly changing threats. Legacy systems can be difficult to update or replace, leading to increased costs and reduced flexibility. These challenges restrict the swift delivery of capabilities to the point of friction in a relevant time frame.

The Challenger

In January 2023, the combat systems aboard the USS Gravely failed to intercept an incoming anti-ship cruise missile in the Red Sea. This incident was a stark reminder of the unnecessary risks our sailors face due to legacy software and outdated processes. These systems, designed to counter Cold War–era threats, are hampered by their monolithic architectures and industrial-age risk-averse mindsets, making it difficult to swiftly deploy software fixes and enhancements. This failure underscores the urgent need for substantial innovation in our software acquisition programs.

The Department of Defense (DoD) has traditionally treated software acquisition as a secondary concern to hardware-centric systems. However, the Defense Science Board (DSB) and the Defense Innovation Board (DIB) have consistently voiced their concerns over the DoD's outdated acquisition processes and delays in delivering software capabilities to the warfighter. Their recommendations, backed by their extensive knowledge and experience, have repeatedly

stressed the need for significant reforms to keep pace with the rapidly evolving threats and technological advancements.

Over the past 4 decades, numerous reports and studies have highlighted the DoD's reluctance to adopt new software development practices. It can be challenging to embrace change when past acquisition efforts have proven successful against threats from the Soviet era. The need for the DoD to quickly adapt to changing mission requirements is directly linked to China's influence and capabilities. Historically, the DoD has been the leader in innovative technologies, but investment by the private sector to field new capabilities with velocity has changed the playing field. The DoD can no longer control the narrative of which countries can access commercial technologies. Realizing that countries or non-state actors are accelerating their capabilities using commercial technologies and practices has jolted the DoD into action.

Resistance

Newton's laws of motion are the foundation of classical mechanics, studying how objects move and interact. As a thought exercise, with a comparison to Newton's laws, think of the DoD acquisition system, its regulations, and its behaviors as "the body or mass":

- **Inertia:** The 1st law states that a body moving at constant speed in a straight line will keep moving in a straight line unless a force acts upon it.
- **Change:** The 2nd law states that the time rate of change of a body's momentum is equal in magnitude and direction to the force imposed on it.
- **Resistance:** The 3rd law states that when two bodies interact, they apply forces to one another that are equal in magnitude and opposite in direction (action and reaction).

Using the 1st law as a general principle of inertia, individuals, teams, or organizations will operate in their current manner without an impetus to change. Assuming that change will happen within the DoD by simply willing it to happen or writing a policy without a push or a pull force is folly without clearing away obstacles, giving maneuver space and top cover.

The 2nd law of change (Force = mass × acceleration) is that the vector sum of forces (F) on an object is equal to the mass of that object multiplied by the acceleration of the object. In this comparison, mass is the DoD acquisition process bureaucracy, and a significant power influencer force (F) must be applied to effect the acceleration, to make change ($a = F/m$). Change is possible and timely when more influential forces drive the change (bigger F). A significant mistake is believing that change can come quickly from a small group or lower-echelon organization acting on the total bureaucracy (smaller F).

Newton's 3rd law could be a comparison for change resistance within a large organization, and it is also worth considering. Starting with the 1st law, we have the DoD acquisition process operating with significant built-up years of "inertia" (this is how it has always been done philosophically). As discussed in the 2nd law, a force must be applied to make change happen—and the larger, the better to have a significant impact. The 3rd law suggests that resistance to change will occur at all levels within the organization and push back on the force of change for various reasons: lack of understanding or urgency, career risk avoidance, entrenched culture or behaviors, lack of knowledge on what needs to change, or minimum training on how to make the change. The first reaction is to apply additional force to overcome the resistance, but throwing a ball hard against a wall only makes it come back faster and harder until the thrower can no longer catch the ball. Aggressively enacting change within large bureaucratic organizations has consequences. Effective change requires significant time investment to break down the causes of resistance and solve them individually. Leadership plays a crucial role as the force for change, but the time needed often exceeds political leadership appointment periods.

The Empire Force

The institutional way of developing software within DoD acquisition was primarily the waterfall method (1970s), which values completeness in requirements and design over the speed of capability delivery. Because there was no near-peer threat after the fall of the former Soviet Union in the mid-1980s, there seemed little interest in the DoD acquisition community in changing business practices to deliver quality software faster.

Meanwhile, personal computer prices became within reach of the general population, and software applications exploded with the dawn of the Internet in the 1990s, creating an "arms race" for competitive software advantage and market share. Software development methods began to evolve within the commercial software industries, allowing first-to-market strategies to emerge.

The waterfall method had significant disadvantages in delivering software quickly, and changing to a more responsive software development model was a live-or-die decision within this new commercial environment. The C suite mandated the need for change, and resistance was addressed through training and implementing different cultural values needed for the workforce. In 2001, the commercial sector described the agile software method and the behaviors required to succeed in the changing software environment.

The Rising

Iterative software methods like agile have emerged to deliver quicker initial value in smaller chunks instead of complete software packages. Potential software users now have a voice and freely give feedback to the developer, which ultimately reduces wasted effort and accelerated capability delivery. Commercial software companies recognize that having skilled software coders and developers who can innovate is a strategically important asset. They understand that fostering a workforce capable of continuously creating, improving, and delivering cutting-edge software solutions is crucial for maintaining a competitive edge in the rapidly evolving technology landscape.

As the Internet of Things exploded, so did communities of bad actors looking to exploit software and design vulnerabilities for personal or political objectives. Unfortunately, manual software testing approaches place a heavy reliance on the skills and diligence of individual testers to identify defects and issues, often towards the latter stages of the software delivery pipeline. The need to deliver quality software with speed gave rise to automated testing tools in development, which, in turn, started the philosophy of DevOps in 2007.

The Practice

DevOps practice promotes better communications and collaboration between development and operations teams to address change challenges. The term "DevSecOps" emerged around 2012, emphasizing the integration of security practices and mindset as a focal point within the software development and operations lifecycle. This evolution to DevSecOps recognized security as everyone's responsibility and that addressing security considerations early and continuously throughout the process was crucial for delivering secure, high-quality software at scale.

The DoD's recognition of China as a rising threat to national security underscores the gravity of the situation and the need to reenergize innovation. The DoD's acquisition process was seen as inadequate in responding to emergent threats (1st law), highlighting the urgent need for change in the DoD's software acquisition practices. This recognition has sent shockwaves throughout the DoD, prompting a significant rebranding of the Defense Acquisition System (DoD Directive 5000.01) in 2020.

Major Force

A major force (2nd law) for change to the bureaucracy of DoD acquisition culminated in January 2020 with the *Operation of the Adaptive Acquisition Framework* (DoD Instruction 5000.02), signed by the Under Secretary of Defense for Acquisition and Sustainment. The Adaptive Acquisition Framework (AAF) defines the influencer forces needed to effect change and empower leadership to employ thoughtful, innovative, and disciplined approaches to deliver capability to the warfighter that is relevant and timely to the fight. As part of the change, the recognition occurred that software differs from hardware systems. The Software Pathway (SWP) of Acquisition aims to facilitate rapid and iterative delivery of software capability (e.g., software-intensive systems or software-intensive components or subsystems) to the user. Additional force multipliers added responsibilities in the *Operation of the Software Acquisition Pathway* (DoD Instruction 5000.87) to institutionalize recent changes. Policy within SWP aimed to reduce the resistance to change by removing institutional barriers and insisting on demonstrating viability within a short period (3rd law).

Whiplash Effect

In DoD acquisition, waterfall software development methodologies have almost overnight become a faux pas, as the AAF SWP requires iterative software methods. DoD acquisition also became aware of the DevOps movement in the commercial space, where capabilities could be developed and delivered within days or even hours. Constantly chasing after the latest technological trend or innovation to compensate for years of neglect and stagnation can have a detrimental "whiplash effect" within an organization. This approach often leads to a lack of focus, wasted resources, and disruption to existing processes and workflows.

Recognizing that not all software systems are created equal within the DoD acquisition landscape is crucial. The diverse range of mission requirements and operational contexts necessitates considering different software development and delivery models tailored to the specific mission model at hand. A one-size-fits-all approach to software acquisition and development needs to be revised to address the varying needs and constraints of different DoD programs and systems. Due diligence requires evaluating system criticality, operational environments, security requirements, and integration with existing infrastructure to determine the most appropriate software development methodology. For instance, mission-critical systems with stringent safety and security requirements, such as those used in weapons systems or command and control applications, may necessitate a more structured and rigorous development approach, such as waterfall.

Maybe Not So Evil?

Waterfall emphasizes extensive up-front planning, documentation, and thorough testing to mitigate risks and ensure compliance with strict standards. Waterfall software development is ideally suited when all mission requirements are known, documented, and quantified. Waterfall often results in a longer development cycle and delayed feedback by design since the entire capability is delivered in one shot. Any changes or course corrections required due to evolving requirements or unforeseen issues necessitate rework and can significantly impact the delivery timeline. DoD program managers need the flexibility to decide which software development method meets the program's mission needs. Delivering partial or iterative working software capabilities is not necessarily the best option in a Tomahawk cruise missile guidance system or the engine control software in an F-35.

Sweet Spot

Agile and iterative software development methods embrace an adaptive approach, breaking the project into smaller, manageable iterations or sprints to deliver working software of value to users sooner. Frequent feedback loops and opportunities exist for course correction based on user input and evolving needs. Retiring technical debt earlier and delivering working software sooner allows for the early identification and mitigation of issues. Delivering iterative software capability based on a Capability Needs Statement for a traditional Information Technology (IT) networked or Command, Control, Communications, Information (C4I) system is squarely in the sweet spot for agile software development. Major capability and middle tier acquisitions have all had challenges implementing modern software development methods within the hardware-dominated pathway.

Agile–Scrum–Fall (BS)

Several DoD acquisition programs have taken a conservative approach to migrate from waterfall to an iterative/agile-like software development environment ("agile–scrum–fall"). At the onset, this might seem like an excellent way to balance the strengths of both methodologies. However, it often leads to more challenges and inefficiencies than a pure agile or waterfall approach. Agile and waterfall are fundamentally different in their philosophies. Trying to combine these two can lead to confusion and conflict.

One of the core principles of agile is the continuous and incremental delivery of value. This approach allows for regular feedback and adjustments, ensuring the end product is closely aligned with user needs and expectations. The waterfall methodology is designed to deliver all value at the end of the project, following a strict sequence of phases. Combining these two methodologies into an "agile waterfall" approach can lead to inefficient use of resources. The continuous and adaptive nature of agile can be hindered by the sequential structure of waterfall, potentially delaying the delivery of value. This delay can result in longer lead times, increased costs, and a final product that may not fully meet user needs due to the lack of regular feedback and adjustments. Managing an "agile waterfall" project can be more complex than managing a purely agile or purely waterfall project, as it requires balancing both methodologies' conflicting demands and processes. There is a risk that instead of getting the best of both worlds, an "agile waterfall" approach might end up with a poor implementation of both methodologies, leading to suboptimal results. The DIB report of 2019 introduced the term "agile BS" to describe ineffective implementations of agile methodologies.

Poor implementation of any software method can lead to bugs, security vulnerabilities, system instability, and a poor user experience. It can also result in wasted resources, both in terms of time and money, as significant effort may be required to fix the issues caused by poor implementation. Enhanced risk of failure can negatively impact the program or product's reputation, leading to a loss of user trust and potential funding loss.

The Architect

Waterfall and agile methodologies primarily influence the software development process but can also indirectly impact the software architecture deployment method. The development methodology (waterfall or agile) can affect how the architecture evolves, especially regarding adaptability, scalability, and responsiveness to changing requirements.

- In a waterfall approach, the software architecture is typically defined upfront and remains unchanged throughout development and deployment. Monolithic software architectures are simpler to develop and easier to test, and match the change cycle of waterfall development.

- Agile methodologies, emphasizing iterative development and continuous feedback, can lead to evolving architectures like Microservices or segmented Monoliths. Microservices are more flexible in adding capabilities or making changes while deployed in the user environment but require careful management due to inherent complexities.

The Monolith

The DoD faces challenges in modernizing and delivering capabilities at a relevant speed. One of the factors contributing to this challenge is the dominant use of monolithic software architectures. This legacy approach builds software applications as a single, interconnected unit, resulting in a cohesive and unified system. However, this architecture can lead to slow and risky changes due to the tight coupling of components. Scaling and maintaining the application can also become challenging (often called “Spaghetti Code”). Additionally, the accumulation of technical debt occurs when software defects are discovered during testing or operations, with limited options for immediate correction until the next build cycle. Monolithic architecture might allow for faster initial development and deployment, but it can become increasingly complex and challenging to manage as the application grows.

The X Factor

Microservice software architectures allow modifying or updating capability without redeploying the entire software code, which is attractive to users at the edge of friction with limited bandwidth. The choice of software architecture can impact the speed of development and the level of user engagement. Microservices architecture, while potentially more complex to set up initially, can allow for faster, more independent development of different application parts and make it easier to adapt to changing user needs.

Killing the Monolith

Legacy DoD software systems based on a monolithic architecture have few options to modernize since they are typically large, complex, and tightly coupled, making them difficult to modify, scale, or update. The lack of access to the source code and intellectual property rights further complicates modernization. The vendor lock-in situation limits the DoD’s ability to adapt and evolve these systems in response to changing needs and technologies, and it stifles innovation by reducing competition. As a result, the DoD often pays premium prices for incremental improvements in a spiral development process. Modern practices can help to break down large, monolithic systems into smaller, more manageable components; enable continuous and iterative development; and foster a more competitive and innovative environment:

- A “brownfield” comes from building on a “brownfield” site, where existing structures or infrastructure must be worked around or incorporated into the new design. A brownfield project means modifying or upgrading an existing software application, system, or infrastructure. Developers must work with and consider the existing code, systems, and technologies, which can limit their options and flexibility compared to a greenfield project. Brownfield projects can leverage existing investments and resources and often have a more precise scope and predictable outcomes. One common brownfield project involves modernizing or upgrading legacy systems. These are frequently older software applications or systems that have been used for a long time and may have outdated technology, limited functionality, or maintenance challenges. The brownfield project aims to update the system, improve its performance, enhance its features, and ensure its compatibility with modern technologies. Brownfield projects can also involve integrating multiple existing systems into a cohesive solution. Some advantages of brownfield

efforts are making software improvements, fixing bugs, optimizing performance, and addressing security vulnerabilities.

- The Strangler Pattern is a software development approach that gradually transforms a legacy system into a new one. The name comes from the strangler fig, which grows on another plant, gradually enveloping and replacing it. The Strangler Pattern involves building a new system around the edges of the old system. The new system intercepts calls to the old system, handling those it can and passing on those it cannot to the old system. Over time, more and more functionality is moved from the old system to the new system until, eventually, the old system is "strangled"—that is, it becomes redundant and can be retired. This approach allows for incremental modernization, reducing the risks associated with big-bang replacements. It also provides continuous value delivery, as new features can be added to the new system even when the old system is retired.
- A "greenfield" event or project refers to developing code from scratch without constraints imposed by prior work. A greenfield project means creating a new software application, system, or infrastructure without considering prior work, existing systems, or legacy code. Developers can use the latest technologies, methodologies, and best practices without being constrained by compatibility with older systems or technologies. Greenfield projects can offer more freedom and creativity, but they can also come with challenges, such as the need to make many decisions from scratch and the potential for scope creep due to the lack of defined boundaries.

The Shining

The DoD is taking a significant step forward by embracing DevSecOps as the next major advancement. Programs like Kessel Run and PEO IWS X Integrated Combat Systems are early success stories, showcasing the effective implementation of DevSecOps practices using a microservice architecture. The DoD recognizes the benefits of adopting DevSecOps, including developing and deploying secure, high-quality software faster and lowering total ownership costs. The DevSecOps approach fosters a culture of collaboration and shared responsibility for security, further enhancing the overall effectiveness of software development within the DoD. DevSecOps leads to more secure, high-quality software delivered faster.

G-Forces

An illustrative comparison of g-force can describe the force exerted on DoD acquisition during rapid acceleration or deceleration within the organization. Higher g-forces, such as those experienced in high-speed maneuvers, can exert greater forces on the acquisition system, leading to increased risk and potential physiological effects of overpromising results. Adopting DevSecOps can significantly accelerate the software development process. DevSecOps (also known as Software Factory) delivers faster and more frequent releases into the production environment and becomes available to operations based on their need. The acceleration g-forces of continuous integration/continuous delivery (CI/CD) and automation principles of DevSecOps have exposed secondary challenges with the DoD's cut-and-paste adoption of commercial practices. DevSecOps integrates security and testing throughout the development lifecycle, enabling potential issues to be addressed earlier and more quickly via automation. DoD manual processes left unchecked negate the value proposition of DevSecOps. The DoD must adopt new processes, culture, and mindset rather than simply applying legacy processes to the new methodology. Shifting "left" authorizations and certifications to rapidly field new capabilities to the user often requires tremendous force (Newton's laws) beyond the program manager's capabilities.

Speed Bottlenecks

CI/CD accelerates DevSecOps, where automated builds and tests are run continuously and delivered to the production environment. Applying legacy software certification processes to DevSecOps creates a speed bottleneck. Legacy software development approaches often involve long development cycles, with integration and testing phases happening after development. Manual testing and risk-averse siloed teams can decelerate the advantages gained in the DevSecOps software development lifecycle. For example, if the Authorization to Operate is performed manually and only at the end of the development cycle (as is common in legacy processes), it can delay software delivery.

Everything Changes With Release Frequency (Be Careful What You Ask For)

The cadence of continuous software delivery (or deployment) into the operational environment dramatically impacts the DevSecOps architectural investments aligned with the need for agility. The operational "user burden" becomes the driving factor for the release frequency.

- **Major new features** that require system installation downtime and training will follow a more deliberate release and internal testing process. The AAF SWP calls for new capabilities cadence to be delivered to operations at least annually.
- **Minor updates**, including small feature tweaks and bug fixes, can also require system installation downtime. The delivery cadence depends on the software factory's architectural setup and the user windows that bring down the system. With minor process modifications, quarterly or monthly updates are possible.
- **Plugins**—new features or functionality added to existing applications without changing their code—can be updated without downtime and are the least intrusive to the user. Several weekly releases are possible but require significant process modifications within the software factory.

Pit of Despair

A world-class DevSecOps Software Factory architecture performing at one cadence is terrible at another. A yearly release cadence does not require the degree of automation, tooling, and lean processes as one on a quarterly or monthly release cadence. Shifting to a weekly or daily release cadence requires an entirely different Software Factor architecture and automated processes. Figure 1 shows how changing the cadence from monthly to weekly leaves significant gaps in the software factory operations. Everything must change to achieve a weekly or faster delivery cadence, including a new architecture, tooling, and processes.

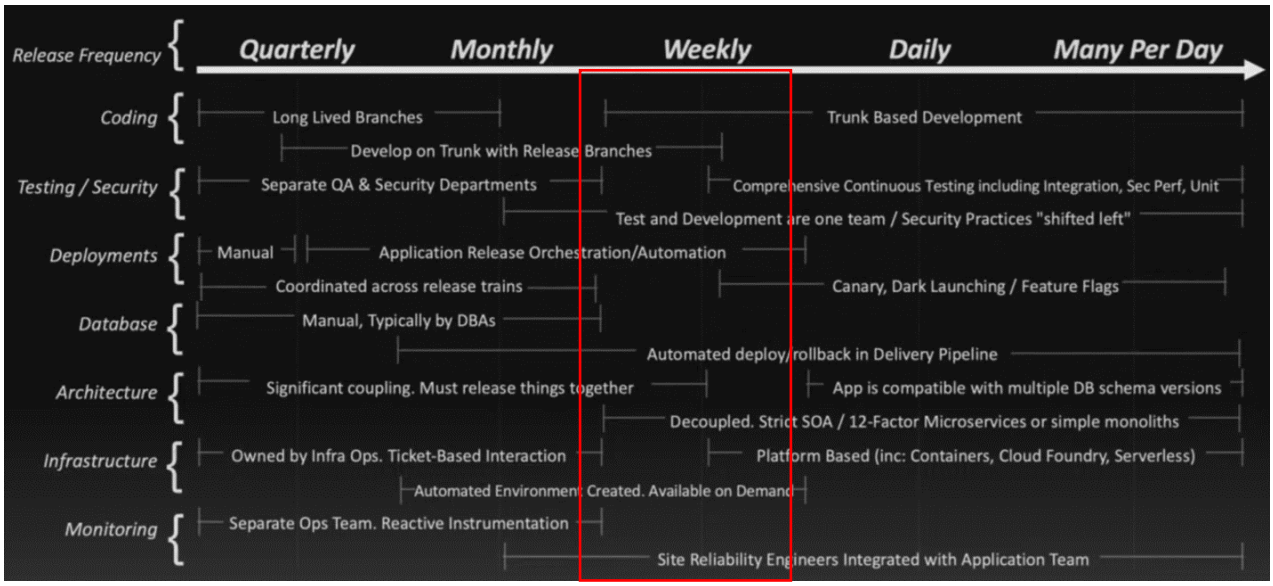

Figure 1. The Original Version Is From Lean Enterprise by Jez Humble et al.

Supporting Roles

Few software acquisition programs have the same constraints, and the AAF recommends that program managers consider the program's unique characteristics and tailor it within boundaries. The following ideals are presented as DoD acquisition pivots to employ a modern software process:

- **Top Cover (Navy Example):** Support for the software acquisition program has to span from top to bottom of the organization. The team should include supporting members to achieve synergy, including the Resource Sponsor (OPNAV), the User (Fleet), The Decision Authority (PEO), the Installation Activity and Shipbuilding Acquisition Program Management (SHAPM), Office of the Secretary of Defense, Congress, and Industry.
- **Clear and Well-Defined Requirements:** Clearly define the requirements and objectives of the software program. The first step is understanding the end users' and stakeholders' needs and any regulatory or compliance requirements. A clear understanding of the desired outcomes will help guide the development process and ensure the software meets the intended purpose.
- **Effective Project Management:** Implement effective project management practices to ensure the software program is delivered on time, within budget, and according to the defined scope. Basics like establishing a project plan, setting realistic timelines, allocating resources appropriately, and regularly monitoring progress are essential.
- **Agile Development Methodology:** Consider adopting an agile development methodology like scrum or kanban. Agile methods promote iterative development, collaboration, and flexibility, allowing continuous improvement and adaptation to changing requirements. This approach can help mitigate risks and ensure the software program meets evolving needs.
- **Skilled Development Team:** Assemble a skilled and experienced development team with the technical expertise to execute the project successfully. The government team should include software engineers, designers, testers, and other relevant roles. A competent team will be able to effectively translate requirements into functional software and address any technical challenges that may arise.
- **User-Centric Design:** Prioritize user experience and usability in the software's design. Engage with end users to gather feedback and incorporate their needs and preferences

throughout development. A user-friendly interface and intuitive functionality will enhance user adoption and satisfaction.

- **Modular Contracting:** Traditional DoD contracts often aim to deliver a complete, fully functional system at the end of the contract period. This approach can be problematic for software development, where delivering functionality in smaller increments and iterating based on feedback is often more effective. Modular contracting, which breaks down the project into smaller, more manageable pieces, has been recommended as a best practice but has faced resistance. Each capability module is contracted separately, allowing for more flexibility, risk management, and opportunities for innovation.
- **Other Transaction Authority Contracts:** Other Transaction Authority (OTA) is a vehicle federal agencies use to obtain or advance research and development (R&D) or prototypes. OTAs are not subject to the Federal Acquisition Regulation (FAR), which gives agencies more flexibility in their contracting processes. However, there are still rules and guidelines that must be followed. OTAs can be used for basic, applied, and advanced research and prototype projects. If a prototype project is successful, the agency can award a follow-on production contract or transaction without competition. At least one nontraditional defense contractor or nonprofit research institution must significantly participate in the project to use an OTA. Competitive procedures are generally used to award OTAs. Still, an agency may award a noncompetitive OTA in certain circumstances, such as when a particularly innovative concept or capability is involved or when time is of the essence.
- **Defending the Budget Using DevSecOps:** It is essential to highlight the value and benefits that DevSecOps brings to the DoD. Potential cost savings can be achieved by implementing DevSecOps practices. By integrating security and quality assurance throughout the software development lifecycle, you can identify and address issues earlier, reducing the cost of fixing them later in the development process or during production. Additionally, automation and CI/CD pipelines can streamline processes, reduce manual effort, and improve efficiency, resulting in cost savings over time. By integrating security practices from the beginning of the development process, vulnerabilities and risks can be identified and addressed early on. This proactive approach helps to minimize the potential for security breaches and reduces the associated costs and operational damage that can result from security incidents. DevSecOps promotes collaboration, communication, and shared responsibility among development, security, and operations teams. This cultural shift can improve teamwork, increase innovation, and foster a more efficient and productive work environment.
- **Ongoing Support and Maintenance:** Plan to continue software program support and maintenance after its initial deployment. The focus should include addressing bug fixes, implementing updates and enhancements, and providing technical support to users. Regular maintenance and updates will ensure that the software remains secure, reliable, and aligned with evolving needs.

Nuggets (From PEO IWS X PM Integrated Combat Systems, CAPT Phillips)

- The DoD is its own unique culture, but it is not that different.
- Deliberately build your culture.
- Know your stakeholders and constantly communicate.
- Have a robust communications plan and talking points.
- Know your market and compete where you can. (Forward progress is better than none.)
- Big bang almost always fails.
- Under-promise and over-deliver.
- Hope that the competition underestimates you because that gives you an opening.

- Change takes investment, but you will have to earn it.
- Be humble. Your first idea or version is almost always bad.
- The faster you learn, the better.
- You need to educate the entire industry on what you are doing.
- Slowly force change.
- Both an industrial and digital mindset is required to be a successful change agent in the future.

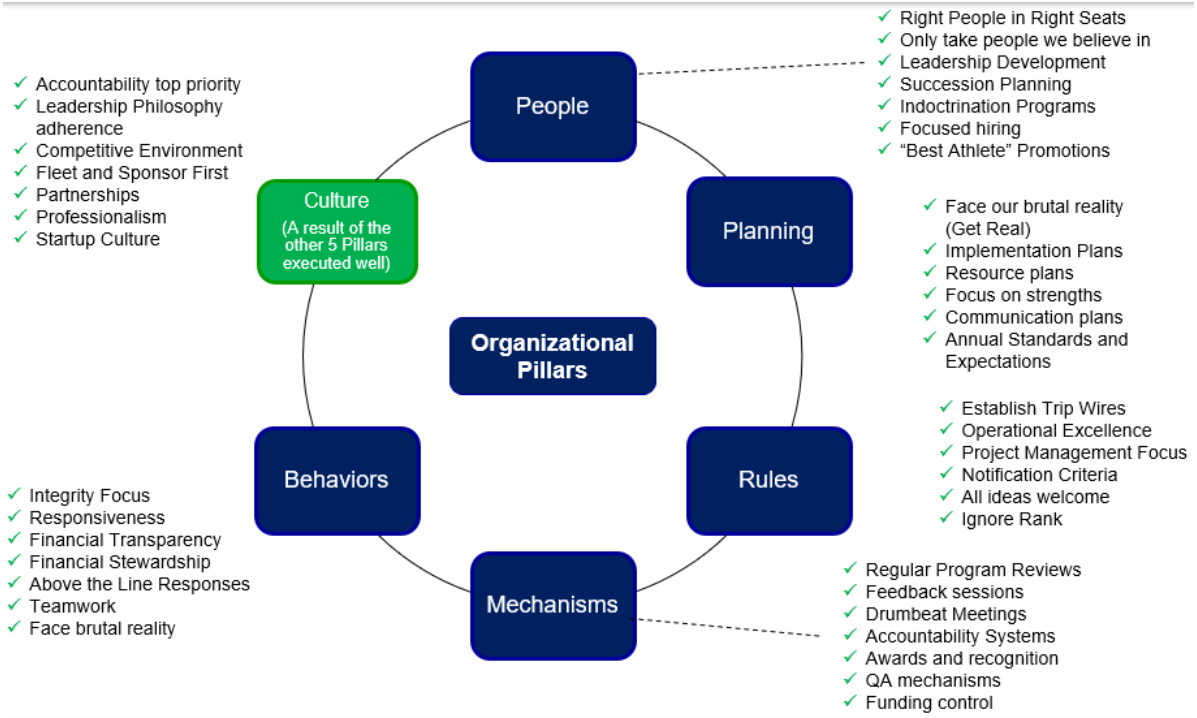

Figure 2. From PEO IWS X PM Integrated Combat Systems by CAPT Phillips

The End Game

Innovation in software acquisition within the DoD has experienced the good, the bad, and the ugly of a system accelerating without a complete understanding of the bureaucratic resistance and business practices necessary to achieve velocity.

Major considerations are needed for modular and flexible contracts, incorporating testing and evaluation throughout the software process and shifting left certifications and approvals to deploy at the speed of relevance. The importance of a trained and skilled workforce with user interaction and senior leadership support cannot be understated.

Focus is needed in understanding the significance of reasonable and prioritized requirements, advocating for a shift from compliance-based, overly prescriptive requirements to more iterative approaches like iterative/agile development approaches to reduce cost, risk, and time.

Addressing these innovation challenges requires a comprehensive approach that includes effective project management, stakeholder engagement, risk management, and a focus on iterative development and continuous improvement. Culture and behaviors take time to adjust to the applied force; they must be constant and consistent to ensure that the capability delivered is responsive to a changing threat.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net