NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

CAPSTONE APPLIED PROJECT REPORT

SOFTWARE INTEGRATION OPTIONS FOR THE F-22 AND F-35 MAJOR DEFENSE ACQUISITION PROGRAMS

June 2023

By: **Pamela A. Kowal-Swartout**

Advisor: **Jeffrey R. Dunlap**
Co-Advisor: **Ceir Coral (DCMA-LMFW)**

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2023	**3. REPORT TYPE AND DATES COVERED** Capstone Applied Project Report

4. TITLE AND SUBTITLE SOFTWARE INTEGRATION OPTIONS FOR THE F-22 AND F-35 MAJOR DEFENSE ACQUISITION PROGRAMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Pamela A. Kowal-Swartout	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Major DOD programs, such as the F-22 Raptor and F-35 Lightning II programs, have software challenges. Given the recent top-down direction to improve the agility of critical software acquisition, development, and deployment, it is crucial to understand the remaining factors preventing improvement. This research included a broad review of government reports and recommendations, private-sector best practices and innovation, and military efforts to work with the private sector. These efforts revealed factors slowing progress, including the often-siloed structure of DOD programs, long development timelines, and rigid budget funding cycles. Acquisition processes for software ideally differ from those for hardware, but they are often undifferentiated in practice. Program leaders are frequently too invested in limited development approaches and resistant to recommendations from software experts. The research pinpointed several areas of the DOD's acquisition programs within which to incorporate changes to current practices. These changes should result in program improvements across areas of cost, schedule, and performance.

14. SUBJECT TERMS software development, software pathways, F-35 Lightning software challenges, F-22 Raptor software factories, updating software acquisition practices in MDAPs per SWAP recommendations	**15. NUMBER OF PAGES** 111
	16. PRICE CODE

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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SOFTWARE INTEGRATION OPTIONS FOR THE F-22 AND F-35 MAJOR DEFENSE ACQUISITION PROGRAMS

Pamela A. Kowal-Swartout, Civilian, Office of the Secretary of Defense

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CONTRACT MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 2023

Approved by: Jeffrey R. Dunlap
Advisor

Ceir Coral
Co-Advisor

Michael R. Schilling
Academic Associate
Department of Defense Management

THIS PAGE INTENTIONALLY LEFT BLANK

SOFTWARE INTEGRATION OPTIONS FOR THE F-22 AND F-35 MAJOR DEFENSE ACQUISITION PROGRAMS

ABSTRACT

Major DOD programs, such as the F-22 Raptor and F-35 Lightning II programs, have software challenges. Given the recent top-down direction to improve the agility of critical software acquisition, development, and deployment, it is crucial to understand the remaining factors preventing improvement. This research included a broad review of government reports and recommendations, private-sector best practices and innovation, and military efforts to work with the private sector. These efforts revealed factors slowing progress, including the often-siloed structure of DOD programs, long development timelines, and rigid budget funding cycles. Acquisition processes for software ideally differ from those for hardware, but they are often undifferentiated in practice. Program leaders are frequently too invested in limited development approaches and resistant to recommendations from software experts. The research pinpointed several areas of the DOD's acquisition programs within which to incorporate changes to current practices. These changes should result in program improvements across areas of cost, schedule, and performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAF	Adaptive Acquisition Framework
ALIS	Automated Logistics Information System
API	application programming interfaces
ATO	authority/authorization to operate
C2	command and control
C2D2	continuous capability development and delivery
CEO	chief executive officer
CIO	chief innovation officer
DAU	Defense Acquisition University
DCAR	DOD Centralized Artifact Repository
DevOps	development and operations
DevSecOps	development, security, and operations
DIB	Defense Innovation Board
DOD	Department of Defense
FoX	Fighter Optimization Experiment
GAO	Government Accountability Office
IP	intellectual property
IT	information technology
MDAP	major defense acquisition program
MVP	minimally viable product
ODIN	Operational Data Integrated Network
OODA	observe-orient-decide-act (loop)
OSA	open systems architecture
OSD	Office of the Secretary of Defense
P1	Platform One
PEO	program executive officer
ROTC	Reserve Officers Training Corps
SWAP	Software Acquisition and Practices (Study)

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge and warmly thank my advisors, Professor Jeffrey Dunlap and Colonel Ceir Coral, who made this work possible. The guidance and support of Chloe Woida from the Graduate Writing Center and her advice carried me through all the project writing stages. I would also like to thank my DCMA F-35 team members for letting my quest for knowledge be fed.

I would also like to give special thanks to my husband, Matthew Kowal, and my son, Tyler, for their patience, love, and unwavering guidance while taking this journey to satisfy this research and writing project.

Finally, without my faith in God, I would not have weathered all the challenges I faced working as an acquisition professional and master's student simultaneously.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This capstone report explores the challenges of software acquisition practices in the F-35 Lightning II and F-22 Raptor programs. Throughout defense acquisition history, programs have addressed the "golden triangle" of cost, schedule, and performance. High-profile defense programs, such the F-35 program, have significantly deviated from this initial acquisition baseline over the past 20 years due to the realization of risks (Ludwigson, 2021). The F-35's mission capabilities are controlled primarily by software, yet as adversaries' threats have evolved, the software has not. The F-35 was not considered a software-intensive program when it was conceived in 2000; however, in 2020, in satisfying designated acquisition criteria and blending technology and software upgrades, the program encountered more strife and struggled to meet the cost, schedule, and performance requirements (Walsh, 2021).

Despite former Secretary of Defense Ash Carter's concerns over threats from American adversaries China, Russia, Iran, and Korea and terrorism, the United States has been complacent in minimizing software breaches. The challenges facing the Pentagon involve developing, possessing, and integrating the latest and most current technology (Carter, 2019, p. 54). Nevertheless, American software solutions do not adequately resolve the highest-level threats, for one, because the mission, warfighter, and end-user experience a delay in the software released. Moreover, the software is often outdated before the initial release, so it represents a useless, wasteful application of taxpayer dollars.

The F-22 program was the precursor of the F-35 next-generation fighter jet. The F-22 is one of the first Air Force aircraft to encounter challenges involving new user threats in the field that require rapid software solutions. The Defense Innovation Board (DIB) in its 2019 Software Acquisition and Practices (SWAP) Study recommended that software-centric programs, such as those for the F-22 and F-35, incorporate the observe-orient-decide-act (OODA) loop into software development (McQuade et al., 2019). The DIB has emphasized improving development practices, planning, organization, and contracts given recommendations by Defense Acquisition University (DAU) South (Brady & Skertic, 2021). McQuade et al. (2019) stress the importance of modernizing the warfighter,

prioritizing delivery speed, supporting continuous adaption, and frequently upgrading software.

Furthermore, speed and cycle time matter in software development, as the F-22 and F-35 are combat-ready aircraft designed to react instantaneously. The lack of new, reliable software acquisition pathways stunts the programs' cost, performance, and schedule requirements (Brady & Skertic, 2021). The jets rely heavily on data-intensive software—at the speed of development—releasable to the warfighter. The Air Force Digital Service team, which oversees software development for the F-22 program, understands the importance of modernizing the technology more quickly while assuming additional risk (Ulsh & McCarty, 2019). The F-35 program is even slower to react to the same issues, according to the Government Accountability Office (GAO), which publishes annual reviews of the programs (Ludwigson, 2021, 2022).

A 2021 GAO report pinpoints where the F-35 program has failed to meet its software schedule. Ludwigson (2021) highlights "key practices that would enable [the GAO] to assess how the program uses agile software development data to manage cost and schedule concerns . . . identified in prior reports" and measure the results of new software practices (p. 3). The GAO recommends that the DOD "update its modernization schedule to reflect achievable time frames, identify and implement tools to enable automated data collection on software development performance, and set software quality performance targets" (Ludwigson, 2021, Highlights section). According to Ludwigson (2022), F-35 software delivery to the warfighter, particularly the timing and acceptance of software, is currently designed to benefit the fleet; however, the goal of the program is to improve software delivery to the user (Capaccio, 2022). Nevertheless, the F-35 program has been unsuccessful at incorporating best practices, and it has failed to deliver on schedule (Ludwigson, 2022).

A. PROBLEM STATEMENT

As major defense acquisition programs (MDAPs), the F-22 and F-35 programs must maintain air dominance. The programs strive to improve software delivery to the user but fail to meet all stakeholders' needs (Capaccio, 2022, para. 3). The latest annual GAO

report on the F-35 program documents shortcomings in software delivery to the warfighter (Ludwigson, 2022). One challenge for the F-22, and legacy programs no longer in production, is that it still needs to develop software methodologies for late in the life cycle. Software integration for the F-22 and F-35 at all stages of the programs is crucial for developing and releasing software that meets the users' immediate needs in the field.

The Air Force oversees the F-22 and F-35 programs to ensure best practices in software acquisition are applied, but the programs are not performing equally well. According to new guidance, the F-22 has experienced more success than the F-35 has in software practices. The F-35 program has struggled to adopt an agile mindset concerning software practices at the current production state, particularly because the program initially adopted a dual path, both at development and production. A recent interview with the F-22 program office lends insight into the simplification and speed of developing the software needed to address emerging threats. In sum, the divergent paths of both programs limited the F-35's software opportunities but benefited the F-22 in terms of agile development and acquisition methods.

When highly visible programs (such as the F-22 and F-35) fail to achieve their promised objectives, results-driven defense leaders notice these shortcomings. These Pentagon-level leaders have recently agreed that upgrades to software processes are a priority, with the goal of creating new acquisition capabilities (DIB, 2019). For example, Ellen Lord, former under secretary of defense acquisition and sustainment, focused on implementing the ownership of government-led software programs and resolving program risks (Brady & Skertic, 2021). Another prominent example is Nicholas Chaillan, former Air Force chief software officer, who championed the DOD's preeminence in all software areas under the tagline "mov [ing] at the pace of relevance" (Chaillan, 2023, p. 2). The DIB's 2019 recommendations address the ramifications of relying heavily on software for mission execution in U.S. national defense (McQuade et al., 2019). The changes these leaders have argued for will be essential in software acquisition, user agreements, directions, and general alterations in the operating and execution of software.

Without effective updates, the F-22 and F-35 programs cannot counteract threats unknown at the time of software development. The emerging threats facing the United

States are China, Russia, Iran, and North Korea (Carter, 2019). Software is the primary driver of the F-22 program and, even more so, the F-35 program; these programs require that the DOD develop, acquire, and operate them. The DOD has fallen behind by not understanding how to integrate new software acquisition and development processes effectively into the programs. Furthermore, new threats require the DOD to adopt a new phase that reacts outside the department's existing processes. Developing at an appropriate speed to transform these program processes to mitigate emerging threats is a challenge for the F-22's and F-35's software suite today.

Despite clear goals set by DOD leadership, lacking buy-in from decision-makers at the program level has effectively shifted delays in software deployment to the end-user (Walsh, 2021, p. 34). Program leaders' goals are misaligned in some cases with the Pentagon's goals—as mandated contract direction is absent—thereby delaying software integration into their programs (National Research Council, 2010) and degrading schedule, cost, and performance metrics (Oakley, 2022b). As a result, the programs cannot keep pace with users' demands, which are based on extreme, rapidly revolving conditions.

The benefits of developed software solutions include timely results, successful software builds at scale, real-time solutions, and decoupling of software and hardware; thus, rolling software releases can support built-in security features. Many program leaders have little experience or training in mission execution and best practices for software from the private sector. The absence of subject-matter experts and other savvy personnel in the DOD to explain and justify the integration and use of software solutions hinders the programs' risk management capabilities. However, some programs succeed while others falter. Understanding the factors causing this variation is necessary in guiding and educating program administrators to meet the top-down direction of improving software practices.

B. PURPOSE STATEMENT

The strategy of the research for this capstone report was to understand, describe, and develop options, stopgaps, and recommendations for agile software acquisition. This research focused on understanding the current state of software management in the F-22

and F-35 programs, including acquisition, performance, delivery, scope, cost-effectiveness, and efficiency. The aim of this capstone was to explain the continuing limitations of government capabilities in developing and administering a plan to own and manage millions of lines of code for the aircraft alone. The research setting explored the roles of several themes in shaping the software practices of these programs:

1. Ownership of intellectual property
2. Communication between roles that shape these outcomes
3. The lack of resources (e.g., money) to manage the programs.

The lack of an effective, standardized system management plan and integration model for software development in MDAPs means that while some leaders update their practices, others do not. One goal of this capstone was to find the root causes of (and explain) leaders' failing or refusing to incorporate agile software aspects, such as development, security, and operations (DevSecOps; see Figure 1) into the programs they manage. The research explored options for the F-22 and F-35 programs, both of which rely on software for mission execution to support and protect national defense. The ultimate research objective was to evaluate the program directives to encourage ownership of these government-led software programs, thereby resolving current and future program risks.

C. RESEARCH METHODS

The research first employed a variety of methods to understand the current state of software development in MDAPs and acquisition, as well existing government and military analyses of and recommendations for the problems within these programs. Then, a set of targeted approaches helped to identify which programs have performed better and why and to develop a knowledge base of best practices from the private sector.

In exploring the current state of software development and acquisitions of these MDAPs, the research surveyed a diverse set of documentation and literature. First, it was important to review statements of work and performance, as well as other documentation, related to the F-22 and F-35 programs. A review of related GAO reports revealed key details about the state of the programs, areas needing growth and improvement, and

potential risk. The review also involved analyzing reports from the DIB, particularly those detailing the 2019 SWAP Study, which identified shortcomings and reiterated the gaps in current practices. Also, to gain context and an understanding of the programs' status, the review considered books and articles from key leaders who defined the current and future state of the programs, as well as successful practices.

To assess the current level of responsiveness to the goals and recommendations for future development and to understand why some programs have failed, it was first necessary to review guidance that leaders have already received. This process involved a close reading of the National Defense Acquisition Act, publications by the Section 809 Panel, the SWAP Study and other publications by the DIB, DOD Instruction 5000.87, and other transaction agreements. Other helpful documents included SWAP Study publications referencing practices outside the military sphere and problems associated with MDAP software development.

Some military programs have been successful at integrating and adopting updated software guidance, so reviewing these successful programs could reveal what they have done right. Certain defense-sponsored software factories serve as powerful examples. For example, LevelUP, the Air Force's cyber software factory team, spearheaded Platform One (P1), and Kessel Run is a key driver of DOD innovation in software processes. The research process involved reviewing successful cases of "agile teaming" involving industry and government collaborations as examples of integrating DOD Instruction 5000.87. Two such successes are the Ground Based Strategic Deterrent and the F-22's SWAP vignette teaming with the Air Force Digital Service to update the capabilities of software factory pipelines. These key successes demonstrate how the services create software and purchase information technology (IT) capabilities.

The research process also explored a variety of sources to understand private-sector successes with software vis-à-vis military acquisitions. For example, symposium articles and presentations revealed commercial applications for off-the-shelf software. Such applications have the potential to forecast and mitigate software risks for the DOD. Illustrating the difference between the private sector and military in this process, Nicholas Chaillan is quoted as saying, "The military acts as an old Windows XT computer while the

private sector is today's Windows 10 or iPhone operating systems, 'which continuously roll out upgrades' and pivot to new environments" (Naegele, 2021, p. 1). Other helpful sources of information included articles from defense news outlets and military leaders analyzing private-sector software development and acquisition processes. Websites and news articles that disclosed private-sector software processes were also helpful. These sources from the private sector demonstrated the importance of timeliness, successful software builds at scale, real-time solutions, the decoupling of software and hardware, and the rolling out of software with built-in security features.

Finally, this author interviewed leaders and subject-matter experts to gather their perspectives and opinions based on questions derived from contracts and concerns of the programs' decision-makers and end-users regarding opportunities, risks, and issues. This author submitted a list of interview questions to the Naval Postgraduate School's Institutional Review Board, which determined that neither the information obtained for this capstone report nor the means of gathering it constituted human subject research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. COMPARATIVE ANALYSIS: HARDWARE AND SOFTWARE IN PROGRAM MANAGEMENT AND ACQUISITIONS

This chapter identifies and defines the differences between hardware and software to highlight potential risks in managing programs. These differences are distinct yet interdependent aspects. Still, program managers can alleviate unforeseen problems with hardware and software cost, schedule, and performance requirements by comprehending the variances between the two disciplines. The research, complemented by this author's professional experience, identified differences in the following areas: tangibility, ownership of intellectual property (IP), the development and maintenance life cycles, in-house personnel, authority rights, partial updates, performance measurements and standards, obsolescence and risk management, quality product development, security, critical pathways, and iteration of the design. These topics are vital and deserve consideration at all stages in any program where hardware and software are interdependent.

A. TANGIBILITY

Hardware is a tangible asset. It is an observable and known entity in procurement. Software is an intangible but critical element in operating the program's hardware. The differences between hardware and software are not limited to the costs of materials, labor, and production. The differences in cost estimation and auditability between the two commodities are vast. The software design budget is estimated around this uncertainty and dictates the path for adequately forecasting materials, labor, and resources.

Hardware is easier to estimate and audit because the design and required materials are capable of being captured on a bill of material (BOM), a list of all elements required to build the hardware and the costs quoted to deliver the quantity produced. The cost of materials and production, for example, is known, so there is no need to estimate and work with unknowns. The opposite is true of software.

The software cost estimate is based on source lines of code, so an iterative process is needed to estimate the cost when the number of embedded lines in the hardware interface

required to operate the program is unknown. In a MDAP, the lines of code, as well as changes to other software code, are not easily projected. In fact, Moore's law predicts that "software capacity doubles per unit expenditure every 18 months" (KK, 2009, para. 5).

The development or prototype process for software and hardware relies on extensive individual testing, yet in the end, software and hardware must work as a cohesive unit. Congress is currently reviewing possible reforms in the acquisition processes for software and hardware, as this critical topic is at the center of many defense programs, whose systems' reliance on software has shifted and required tangible changes (Williams, 2021b). The software requires a development process, and often, how the final product will look, how long it will take to produce, how many hours will go into that production, and how it will integrate with hardware are unknown.

Additional differences between hardware and software involve the resources required to write and develop software, which cannot be adequately forecasted. The iterative nature of software is complex, which leads to a lack of appropriate human resources.

B. INTELLECTUAL PROPERTY RIGHTS

Hardware and software are also different regarding ownership of design and IP rights. Hardware is visible, and design authority is retained by the material review board that retains ownership of the design. Software is invisible and depends on code to operate. Property rights for software are building blocks in a constant state of flux.

Who owns the software's design and possesses IP rights depends on various agreements, ownership development, and contract negotiations of rights and usages. The complexity and budget of programs are factors in the volume of software and the ability to retain design and ownership control by retaining ownership of both hardware and software. An example of the complexity of design is a commodity that has a design with multiple subassemblies with embedded operating hardware architectures. Each of these entities is subcontracted by the principal contractor. The budget and core competencies dictate at each level the ability to change the design by retaining control over software to operate the subassemblies required to operate the end unit.

The SWAP Study's recommendations identify that the most critical aspect of software acquisition is owning the IP of a program (McQuade et al., 2019). While there are numerous differences between hardware and software, IP ownership is the most critical aspect. Obstacles to the government's fully owning existing hardware and software IP are often attributed to unknown requirements related to software that emerge during use in the field, limited budgets, and the inability to secure human resources in the highly competitive nature of the software coding field.

Another impact of not having software IP ownership is that it limits the government's ability to regulate the time to execute updates and release the hardware. Program milestones at each stage of software and hardware development are unique but function interdependently. The cost of hardware, while higher than that of software, usually levels off after release unless repairs are needed. On the other hand, software, while operating, must release updates to threats identified by the user. The costs of these updates can be staggering if the government does not retain IP ownership at the beginning of development.

C. DEVELOPMENT AND MAINTENANCE LIFE CYCLE

The responsibility for developing and maintaining the program over its life cycle is another aspect that requires attention. A program's life cycle is defined as the time from conception to closure, with various phases and milestones interspersed. Hardware and software are substantially different in the development and maintenance of programs at each stage of the product life cycle. These differences primarily involve obsolescence. With hardware, the responsible party for changes and updates is more straightforward. Chiefly, the configuration manager has a clear role in updating all materials and managing updates to the revision levels released for a part's number rolls and engineering change break-ins, for example. While some of these changes may correspond with software, those that do not may be executed without considering software updates that ought to correspond with each hardware update.

The initial developer who owns the software IP rights is responsible for making changes, and updates depend on numerous variables, including who developed the software initially, how it integrates with hardware, what is driving the change, what is its commercial application, whether there are security concerns, and who has IP rights at each level. Crucially, the initial developer has dictated the language, code, architectures, design, and ability to change actions of the software. If there are multiple co-dependent languages and architectures of integrated software and multiple developers who own individual IP rights, it amplifies the complexity of the government's ability to incorporate the necessary changes.

The drivers for change derive from multiple sources. The private and public sectors want to go faster and innovate, but the DOD's software acquisition methods cannot support the current pace of innovation (Sutter, 2021). As demonstrated by the recent F-35 Fighter Optimization Experiment (FoX; see Figure 1), the designer and developer may require updates to realize software functionality, the end-user may need to adapt to new threats, the subcomponents may need updates, and new requirements may be contracted to drive the change. The program software estimations method includes the full life cycle cost, but for constant software changes, a strategic approach is preferable (Brady & Skertic, 2021). Notably, the drivers of such initiatives will define the costs and the owner's responsibility in executing these events.

Figure 1. Air Force Project FoX. Source: Sutter (2021).

The ownership of IP dictates the speed and release of the changes. The IP owners should encompass the resources and skillsets readily needed. Conversely, not owning the IP rights creates delays due to the inability to develop and release software. An example of commercial actions hampering the government is the recent Ukrainian request for U.S. assistance to intercede with a Russian interpreter using a communications satellite. The government did not own the IP, which would have allowed it to pivot rapidly, and SpaceX, a commercial entity, could not assist due to its not possessing the IP and being incapable of writing effective code to redirect the satellite. Four key software practices at leading commercial companies, including SpaceX, were identified in a GAO study (Oakley, 2022a). These best practices highlight new capabilities currently missing in the DOD yet recognized by various studies and reports.

D. IN-HOUSE PERSONNEL

Hardware and software differ on levels of availability of in-house personnel responsible for the design, development, repair, and maintenance of critical details.

Hardware design authorities and engineer support employ greater numbers of staff than software authorities, whose coders mitigate design and interface issues. In-house personnel develop, design, repair, and maintain software. In terms of personnel, there are more people available to fulfill these functions for hardware than for software. Moreover, competition for hardware and software differs based on opportunities, empowerment, creativity, and end items. Still, hardware design and maintenance might lag within the services, but these hardware outcomes are less risky than software performance, cost, and schedule, which are a significant concern for programs across the government.

The problem with resources designated to develop, design, repair, and maintain is twofold. Fewer software-centric positions are created in government and the military, and the structural alignment of these positions is often ineffective. For the most part, the government has been less effective than the private sector at creating enticing software-focused career paths. The lack of creative freedom combined with stringent governmental restrictions is not attractive to younger generations, which desire the creative empowerment offered by gaming, Silicon Valley, and other software endeavors. The private sector offers high competition for personnel, and fewer people join the government or military to focus on software.

Even though some agile software resources do exist within the DOD—Kessel Run and the Army Futures Command, to name two—they have a limited availability and capability to respond to all emerging needs. Individual software paths are still not cohesively managed as one system requirement, leading to unpredictable needs. The current DOD workforce struggles with inexperience, cutting-edge software methods, roles, enablers, and teaming hierarchy (Brady & Skertic, 2021). An overall strategy for developing and recruiting a competent digital workforce is not available for many programs, as research indicates. In those cases, managers of software programs cannot rely on in-house talent as do managers of similar hardware programs.

E. AUTHORITY RIGHTS

Software and hardware designs require collaboration among contractors whose expertise is in the product or service that the military service needs to procure. There are

key differences in the authority rights of the hardware and software purchasing, repair, update, and maintenance requirements. Hardware requires software to operate in most programs—and fundamental factors, if unknown, lead to failure when they do not follow the same logic and initial development of testing.

Hardware changes rely on material design authority. Any needed change over the product's life cycle will be reviewed by the material review board. When a hardware change is requested, associated software changes may be essential. The software changes lag the hardware updates due to limitations or restrictions related to the software architecture. Hardware engineering releases integrate changes whereas technical readiness reviews and changes are integrated into software production differently.

The DOD has struggled to purchase software that keeps pace with updates. Software must account for various subassembly architectures, and different code and trickle-down requirements must be tested before a full release. Additionally, software changes are iterative, so even once software is fully vetted, it might already be obsolete upon release. Emerging threats within user environments are problematic for software development in the current state of the DOD.

F. PARTIAL UPDATES

Hardware and software function differently concerning partial updates. Software may be acceptable via partial updates. The inability to integrate updates rapidly presents a new threat to military programs, so the ability to integrate agile updates means the aptitude to release partial updates as necessary. By contrast, hardware revisions must satisfy operational dimensions. Correlating these updates for software and hardware tends to be broken into programs with less testing required than for software alone. Software changes and updates in the current state require capabilities testing before the entire software block is released to the program.

With hardware, patches or partial updates do not make sense. A hardware component encompasses the entire end item and all changes that fulfill requirements; the first article inspection ensures that the product meets the mandatory quality for configuration management, which is satisfied when the updates are ready. Nevertheless,

software releases in most programs are not released as minimally viable products (MVPs), which require that only the functioning code be released. The quality parameters of software are different from those of hardware, yet both are operational in the same system.

With software, the goal of an MVP is making a difference with often temporary or partial updates. However, in most DOD programs, a full software release is the normal way of delivering the product. Moving to partial software releases, when they make sense to the end-user, is a sensible choice for the DOD as it would yield more rapid results.

G. PERFORMANCE MEASUREMENTS AND STANDARDS

The program requires performance measurements and standards to achieve the outcomes of the end items. The performance of both hardware and software is measured and standards used to report the overall progress of integration milestones in the program. One difference between hardware and software standards involves criteria for functionality. Hardware and software performance and measurements also differ in their need for coexistence and interdependency. Chris Lynch, co-founder and CEO of Rebellion Defense, supports both sentiments of hardware and software and the criticality of understanding this era of the DOD:

> When we think of the military, we think of an aircraft carrier, a tank, a jet, or a satellite floating around in outer space. Now those things aren't going to go away. That's very clear. But we're entering into what I think of as the software era of defense. And that's going to be driven by the flawless execution of software to do the mission of defense and national security. (Williams, 2021a, para. 11)

Hardware performance is based on hard specifications that can be visibly measured whereas software success is measured by the ability to execute the functions written in code. Military innovation has migrated to invisible and lethal lines of code embedded in the hardware. The change is momentous and gaining speed, as described by Lynch:

> It's cool to build big, heavy things. I get it. It's super awesome. We should think that those are great. But if you don't get just as excited about rolling out unlimited compute and unlimited storage, continuous integration and continuous deployment, and the ability to have people use Application Programming interfaces (APIs), I can tell you right now, none of it changes. It will not change. And you know what, it doesn't sound that exciting, but

> you got to get excited about it. You should be like, hell yeah. I love APIs. (Williams, 2021a, para. 12)

If hardware is within min–max tolerances, it is measured as working, and until it reaches a certain threshold of limited functioning, it makes little sense to change or update it. Hardware performance standards and measurements contain a matrix of parameters not limited to overall specifications, nonconformance, tolerances, configurations, testing limits, and contractual quality standards.

Software performance measurements are based on the software tested and deemed operationally ready when testing hardware. The DIB's SWAP Study identified the speed and cycle time for software as the most important metric (Bellairs, 2022) This measure allows the program to retain an advantage in supporting the mission need. Hardware that is late for delivery or suffers from nonconformance can affect software development and skew the measurement and standards of software design changes. Software performance can also be rendered useless as soon as it is released—the software either works or it does not.

The cost of negotiating between hardware and software performance standards may be high. Individually, the hardware may meet all required parameters and the software all standards, but when they interface, the question remains whether the overall performance will meet the standards. Once software and hardware are combined, new standards and measurements will require further development and updates.

H. OBSOLESCENCE AND RISK MANAGEMENT

The current supply chain environment of defense programs and contractors constantly weighs potential obsolescence and risk. Program obsolescence differs between hardware and software. Some programs actively include diminishing manufacturing sources in the contract for the hardware supply chain. However, software source requirements have no comparable supply chain for software procurement.

With hardware, diminishing sources, engineering changes, and obsolete components, for example, could lead to obsolescence, but the process takes time. Nevertheless, it is easy to predict and plan for when components will become obsolete and

put in place risk mitigators. With software, obsolescence is always right around the corner due to emerging security threats, changing architectures, and interfaces, and different parts of the software rely on more unknowns, such as languages, platforms, and interfaces. While hardware component obsolescence is contractually documented, with remedies to secure the required assets to support the program through agreement, software obsolescence or the failure to secure adequate resources to incorporate software via agile approaches is not structured in the contract.

The importance of software and its ability to moderate threats are disproportionate to hardware threats. For the hardware issues, contractors possess a defined understanding of the costs and requirements to mitigate potential obsolescence. Conversely, for software, unknown resources, code, architectures, interfaces, and languages, among other dynamic elements, create unknown costs and risks for most programs.

I. QUALITY PRODUCT DEVELOPMENT

The cost of a quality product and the time to design and develop it depend on both hardware and software. The measures and quality requirements are different for software and hardware. Hardware quality requirements have various standards based on different behaviors, constraints development, testing, and costs, vis-à-vis software, which meets varying guidelines developed by subject-matter experts.

Hardware is designed and developed differently. With hardware, good and effective design is built into the product from the beginning. If one has "done it right" since the start of the process, the output is something that will function well for a long period. In contrast, there is a built-in expectation for software that things will go wrong, and things will need to change and be updated. Fundamentally, the two products offer vastly different definitions of quality.

The quality software engineers completing the testing might not be the ones who developed and coded the software. Yet, with hardware testing, those who engineered the products also test them. Other quality differences between hardware and software include testing for proofs of concept, approved environments, and time requirements. Moreover, the tools required to build, test, and inspect the end product for a proof of concept and

release to move to the next path vary greatly. Also, while the standards and organizations establishing hardware and software quality differ, the interrelationships allowing successful interoperability of quality design, development, and productions are critical to meeting the desired executions.

The importance of these quality measures confirms that while the standards of measurement are different, the end design and upgrades meet quality standards for both software and hardware.

J. SECURITY

Program security affects both hardware and software. The applications for hardware and software, while co-dependent, are established and achieved to utilize different skillsets and competencies. Security in hardware programs is more straightforward than in software programs, as cybersecurity demands in software are in a constant state of flux. For example, while software security limits flexibility and communications in the F-22 Raptor program, hardware does not face this issue (Ulsh & McCarty, 2019). Security in software is an area of great concern for the government and commercial sector. Their desired results are the same, but the time to integrate agile changes is vastly different based on their IP ownership.

MDAPs face diverse challenges in integrating security in hardware and software. Hardware security is usually straightforward regarding the physical device, design rights, and configuration and process reviews. Software is much more complex and multilayered, as it may be corrupted and hacked and pose unknown threats.

Mounting cyberthreats have made cybersecurity a DOD imperative. Hackers are evolving rapidly; meanwhile, the DOD has mitigated the risk but not as effectively as ensuring complete protection. In a report titled *Resilient Military Systems and the Advanced Cyber Threat*, the Defense Science Board (2013) warns that the current information system might not ward off attacks by rivals. In identifying various known, unknown, and adversarial vulnerabilities, the board maintains that software systems and code must protect government programs against unknown access, bugs, viruses, operability challenges, and architectural defects, among other hazards. Software security also includes various

authentications, firewalls, and encryption to protect and detour attacks from internal and external system threats (Jeng, 2019).

Hardware security is straightforward and visible whereas software is multilayered and invisible. Hardware vulnerability alerts are another difference. Software and hardware are required to interoperate, but hardware is at risk if the software is not adequately protected against threats. Whereas the security of both hardware and software may become outdated, the ramifications are different for each. While hardware is not easy to modify, software may be changed more easily by modifying code. In defense programs, for example, old hardware must be removed and new builds of physical products coordinated, and software code must be updated, to accommodate evolving security threats (Madhurihammad, 2022).

K. CRITICAL PATHWAYS

Critical paths and processes in programs are prototypical in planning and forecasting both hardware and software, yet they have distinct and comingled lanes. Hardware and software are the two major elements in most defense programs. The software benefits from the hardware but can be degraded if the appropriately designed critical paths for each are not coordinated.

The critical paths for software and hardware within a program differ greatly, yet they overlap to successfully test and operate the desired end item. A critical path involves the items that require significant time in a program to complete. Critical paths for software depend on time development, manufacturing, and testing milestones, identifying and managing tasks of the project and completion timelines for the available hardware resources. The DOD's software assurance concept demonstrates what to include in critical paths for software, including critical program information and critical technology, which outlines important factors of system architecture, design, and development activities defined in the specification, statement of work, and testing of the following aspects:

1. evaluating teams,
2. appropriate security clearances,

3. secure work environments,
4. contractor policies and outsourcing,
5. commercial off-the-shelf systems lineages,
6. trusted development and life cycle scenarios,
7. software shortcomings, and
8. impact of changes to code and potential security hazards (DIB, 2018).

When software design is unknown, hardware design can continue, though it may create new and unplanned software demands. When hardware design is unknown, software design is delayed or incomplete or needs to be updated with new, unexpected elements. Critical paths for hardware include a bill of material and a manufacturing schedule hierarchy dependent on the longest estimate to obtain materials, testing, and qualifications to meet the blueprint.

The overall critical paths of hardware and software require optimizing and working to meet both the schedules and simulations of the complex program. Software techniques cannot identify hardware situations that overlap information and milestones. Where both may forecast isolated system functions, they must ultimately work cohesively. An example of such a hierarchy results in a manufactured piece of hardware operating with software. The software cannot be developed or tested until the hardware is tested to meet the design configurations. In the F-35, for example, the radar system cannot be tested unless the subcomponents are installed and the architectures for other systems built.

The critical paths for software and hardware, while co-dependent, also operate separately. Like a symphony conductor, the program manager coordinates the interactive critical path milestones. Designing and measuring software are different tasks altogether. The same metrics cannot be used to measure both. Unfortunately, some programs still try to use metric means or software to measure hardware (McQuade et al., 2019). The lack of a software-centric approach to technology acquisitions while continuing the previous hardware-centric "always done it this way" approach fails to address the current state

operating today (Williams, 2020). The ability to recognize the shift to a software environment highlights these changes between hardware and software pathways.

L. ITERATION OF THE DESIGN

The hardware and software designed have different iterations that may not always be aligned or forecasted within the programs. Software iteration of design is different from hardware, as depicted in Figure 2. Sean Brady, the DOD's senior lead for software acquisition, is one authority. Modernizing all aspects of software acquisition in most MDAPs and understanding the pathways are critical (Brady & Skertic, 2021). According to the DIB (2018), "Modern software methods make use of a much more iterative process, often referred to as 'DevOps,' in which development and deployment (operations) are a continuous process" (p. 3). The iterative nature of the design in hardware and software is fundamentally different. The hardware design is static and comprises modeled engineering platform software that is iterative, unnatural, and constantly developed, designed, and updated based on the environments in which the hardware must operate.

Software changes rapidly, so iteration is fundamental to ongoing functioning and maintenance. The continuous change in software development is different from the rigidity of the hardware life cycle under which it operates. Furthermore, the testing requirements for software and hardware evaluation diverge significantly. The tools and culture of software and hardware support also differ based on rapid transitions, monitoring, and end-user engagement.

Another discrepancy between software and hardware is the structure of the composition and repeatable design parameters. Software changes are multiple and require repeated testing cycles. Some iteration is expected but not extensive. The goal is to minimize iteration.

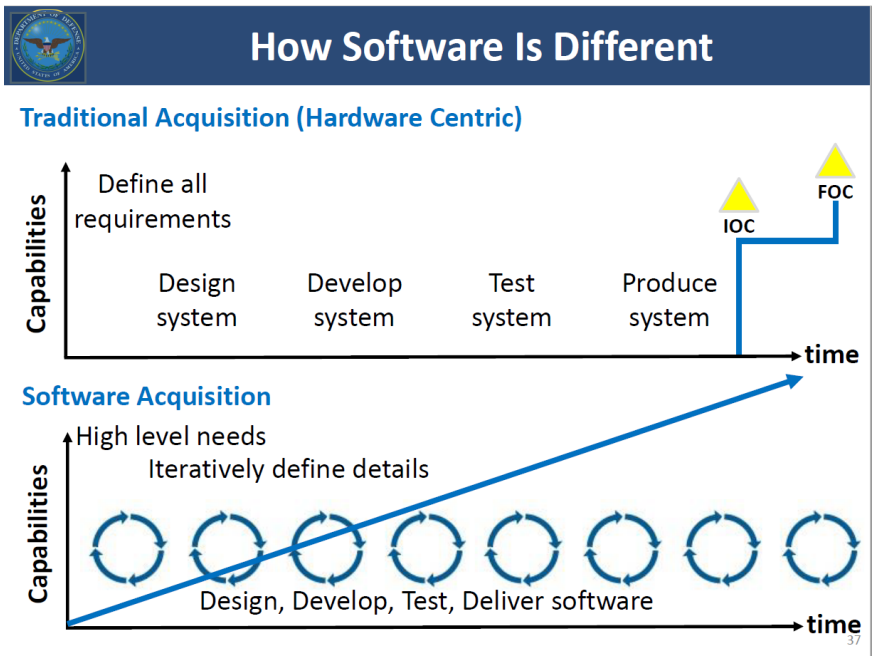

Figure 2. DOD's Software Acquisition Pathway. Source: Brady and Skertic (2022).

Hardware fluctuations are gauged through first article testing, blueprints, and configuration margins. Software constraints, on the other hand, require frequent feedback from the developers and users to test the viability of the efforts.

A critical variance in iterative design and development is that hardware is built and tested while software is repeatedly built, tested, and fixed in a continuous loop. Software updates and nature must be executable, or they are worthless. To illustrate, the scheduled Block 4 upgrade for the F-35 program will consist of 80 percent software and only 20 percent hardware change integrations. Thus, based on the status of this aspect, the percentage of hardware is far less important and desirable than the software iteration.

M. CONCLUSION

The DIB and subject-matter experts agree that the DOD program's defense software is vastly different from the hardware. The hardware development and maintenance are straightforward, but the ongoing software capabilities may require continuous development throughout the life of the program (Bellairs, 2022).

THIS PAGE INTENTIONALLY LEFT BLANK

III. CURRENT STATE OF MDAP SOFTWARE ACQUISITION

Over the past decades, the DOD has faced new threats. Leaders at the Pentagon are aware of the current state of weapons acquisition (Department of Defense [DOD], 2021b; McQuade et al., 2019; Oakley, 2021b; Sager, 2021). Specifically, software development and acquisition guidance are ineffective in the DOD's structure today. The software problem that programs face stems from the current acquisition and development system's failing to align the appropriate software type and pathway with the mission need. Each weapons system is designed to operate under different parameters and to withstand these different environments. Moreover, these weapons must perform at critical times regardless of the era in which they were developed, yet system-required updates to improve performance are often not easily executed because of unknown interoperability issues.

MDAPs are not created equal—each program requires specific results and outcomes, and each encompasses different IP, operations, testing, and threat responses. Former Secretary of Defense Ash Carter (2019) said, "I inherited a system that had not been designed to respond nimbly to quickly evolving threats" (p. 31). The DOD has not been consistently successful in incorporating the agile methods envisioned by leadership when deciding the appropriate guidance for all software acquisition (McQuade et al., 2019). The existing processes for software development and deployment take too long and are expensive, and warfighters are not getting access to the software they need. Additional risks delay access to the necessary resources and tools that software can provide to the end-user, according to a 2019 DevSecOps memo about modernizing software development, security, and operations (Deasy & Lord, 2019). The risks, issues, and opportunities in software acquisition, based on this capstone research, support the consideration of alternative software pathways.

The Pentagon acknowledges gaps in the DOD's understanding of software's influence on its programs (Deasy & Lord, 2019). DOD leadership's objective is to modernize the military by increasing the use of appropriate software pathways in legacy, current, or new programs (Deasy & Lord, 2019). While the most recent results have been

inadequate, division heads have yet to clearly define software improvement expectations for the services. Given these gaps in software, the evolving threats from adversaries of the United States, the world's greatest superpower, put America at a significant disadvantage. The software pathways have inherent potential if the DOD promotes updates in acquisition, such as the Adaptive Acquisition Framework (AAF) and DevSecOps, which support the use of software factories to speed up the development and processes. Leaders are taking notice of these promising software pathways (Brady & Skertic, 2021). The move of advanced software architecture to Kubernetes and cloud migration tools has proven an admirable yet slow integration across MDAPs (DAU, 2023). Adopting knowledge-based practices and interagency software sharing to identify potential roadblocks is one aspect supporting these efforts.

This chapter defines software techniques and architectures and discusses software pathways currently in use by MDAPs. Then, it explores innovative opportunities to expedite software development, testing, and release to the field.

A. SOFTWARE PATHWAYS

There are various tools for acquiring and developing software in the DOD toolbox. The challenge is to choose an adequate approach to software development with the desired result of meeting the cost, schedule, and performance of the defense programs. Recent program predicaments stem from the inability of older legacy software architectures to interoperate without several workarounds. Software pathways offer a choice of the methods: waterfall, agile, continuous capability development and delivery (C2D2), DevSecOps, and hybrids of these. However, recently suggested software paths for many of the mature defense programs might not be adaptable to these methods or, thus, capable of pivoting to meet the established warfighter program schedules. Furthermore, DOD Instruction 5000.87 requires a one-year software capabilities development and delivery schedule (DOD, 2020), but these instruction requirements might not be feasible with older systems. Some of these pathways reflect new policies and guidance from DOD leaders for implementing software at the level of leading commercial-sector companies (Deasy & Lord, 2019). The pathways show promise in expediting development cycles to deliver an

MVP, which provides essential capabilities to the user for evaluation and feedback to developers.

A program's software is never truly in a sustainment phase, as the OODA loop perpetuates its development and redesign in the field (Maurer, 2019). One notable challenge is that an MDAP's software requirements are often misaligned with the program stages of available software pathways (McQuade et al., 2019). For example, during the production stage of a program, the software requirements are estimated, and in the sustainment phase, the user provides feedback on how the software performs in the field. These differences in operational needs for software could be a benefit if new software pathways were made available to MDAPs (Walsh, 2021). Some legacy MDAPs contain different versions of software that require additional software interfaces, which may increase the lines of code required to operate (Department of the Air Force, 2008). These excessive software efforts do not always increase the opportunities for the program or user.

Regarding program platforms, the interview responses from the F-22's program office suggest that the F-22 is in the sustainment phase. A common challenge for older fielded platforms is the integration of "brownfield" software—those with legacy languages and methods—with modern software languages, processes, and tools, along with limited code refactoring. In the end, the introduction of open systems architecture (OSA) is an important step for facilitating the shift in future development to a "greenfield" (John Adams IT, 2020). An OSA is a modular software architecture approach that allows interoperability in weapon systems (Brady & Skertic, 2021).

For many programs, software requirements are funded and budgeted primarily for the development phase, during which most of the software progress, testing, and release occurs. Delays in software development typically delay the production phase and carry over to sustainment, the point at which acquiring the software and hardware to iteratively develop new software for emerging threats offers limited responsiveness (B. Burton, personal communication, September 15, 2021). The mindset of some program administrators is that the production phase of the program involves only the hardware—yet the hardware cannot function without the software (Francis et al., 2001).

This faulty understanding of software and hardware categories in acquisitions means that each is funded under different categories of money and time, as defined by the planning, programming, budgeting, and execution process. These categories are known in the DOD as the colors of money: research, development, test, and evaluation; procurement; operations and maintenance; military personnel; and military construction (AcqNotes, 2021). Challenges arise when the software and hardware are authorized under different colors of money but need to operate as one system. For example, when the first two categories—research, development, test, and evaluation and procurement—are budgeted for simultaneously, the rules for appropriating and obligating the funds dictate different timelines for the completion of the contract.

The following subsections discuss the benefits and challenges of each software path. The key takeaway is understanding whether and when to incorporate these paths into the program.

1. Waterfall

The waterfall software development model, or the "simpler model," is the predominant architecture used in MDAPs today (Wheeler 2018). The waterfall method requires compliance at each testing milestone before release to the field, incorporating a linear path whereby each stage must be finished before the software can advance to the next stage of development, cascading like a waterfall, through software implementation (Oakley, 2021b; see Figure 3). The traditional waterfall method delivers software to the field at an estimated time, between three and 10 years, according to the DOD (Chaillan, 2023). The DOD's software development and acquisition process incorporates waterfall techniques that comply with the Joint Capabilities Integration and Development System and acquisitions processes (McCaney, 2020).

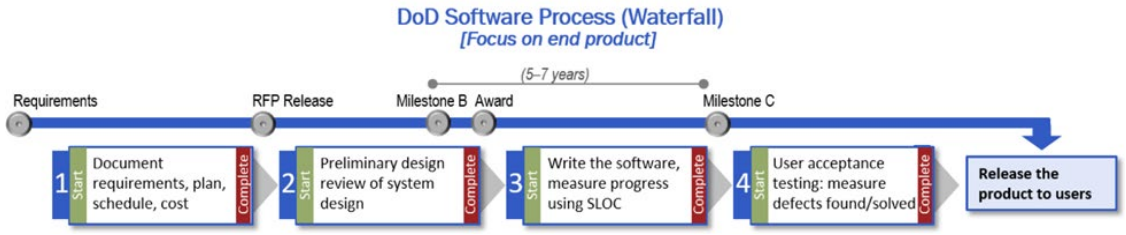

Figure 3. Waterfall Methodology.
Source: Defense Science Board (2018).

The waterfall process was introduced in the 1970s by computer scientist Winston Royce in an article titled" Managing the Development of Large Software Systems." Recognizing that this process involved risk, Royce (1987) recommended prototyping and iterative processes in software. Unfortunately, the DOD and defense contractors misinterpreted the intent of his linear work, as Royce's (1987) recommendations did not apply this path to software development because it exposed additional process risks. Given this model's reputation for simplicity, others have misinterpreted the waterfall model as reducing risk for large military programs, yet Royce advised against its use based on these increased risks (Wheeler, 2018).

The problem with waterfall development is that it delivers updates far more slowly than DevSecOps does (Chaillan, 2023). Using the waterfall method, the software application's life, development process, architecture, deployment, and infrastructure timelines are prolonged months or even years beyond those applying DevSecOps (Chaillan, 2021a). The length of time makes it difficult to keep up with evolving technologies. As Naegele (2021), writing for *Air & Space Forces Magazine*, explains, "DevSecOps breaks development down into manageable pieces and pushes iterative improvements out in periodic sprints, while waterfall development delivers updates far more slowly. The difference can be compared between an old Windows XT computer and today's Windows 10 or iPhone operating systems, which continuously roll out" (para. 3). The waterfall method fails to deliver software rapidly, particularly because release testing takes too much time to address threats in the field. Thus, the waterfall method cannot keep pace with the needs of the warfighter (McCaney, 2020).

GAO and SWAP Study reports, as well as government software reforms, have acknowledged that the current move to agile software development is necessary to address threats (McCaney, 2020). Nevertheless, many legacy programs have not migrated from the traditional waterfall method due to other accreditations required by the DOD. Program administrators have not considered the potential long-term benefits of the change process perhaps because they believe further delays are inevitable (McQuade et al., 2019). Responses to the interview questions from the F-22's system program office describe concerns over moves from older brownfield software platforms to newer greenfield platforms. The benefits of moving to modern software pathways from legacy platforms that use, for example, the waterfall process, pose a significant challenge in merging two diverse software system architectures (B. Burton, personal communication, September 15, 2021). However, brownfield applications can be used to address upgrades and redevelop existing applications. Nevertheless, distinguishing between the two types of software allows the program to innovate all the while sustaining its needs (B. Burton, personal communication, September 15, 2021).

2. Agile/Scrum/Agile-Like

According to a 2019 GAO report, the term agile is deemed as an all-encompassing term for a variety of software practices (Walsh, 2021). As described in the book *Head First Agile*, "Agile calls for the delivery of software requirements in small and manageable predetermined increments" (Ludwigson, 2022, p. 27). As revealed in multiple yearly GAO reports, "this model is based on an inspect-and-adapt approach" wherein "requirements change frequently and software is released in increments" (Ludwigson, 2022, p. 30; Walsh, 2021). GAO research has illustrated how the "Agile frameworks produce ongoing releases, each time adding small changes to the previous release" (Ludwigson, 2022, p. 27). According to DOD (2019a) guidance, the software is tested at each stage in the process, to ensure that the product can be delivered to the user. According to a 2021 GAO report on the F-35 program, in agile collaboration, the customers, developers, and testers work together throughout the project (Oakley, 2021b, p. 4).

Failing to consider agile aspects in older MDAPs represents missed opportunities to modify any subsequent software upgrades to the mission. An environment that fosters software program success must first have the right conditions for integrating these agile development techniques. Guiding choices made along this path are favorable or unfavorable condition of the market, the customer's involvement, innovation, the modularity of work, and the impact of interim mistakes. Agile and Scrum methods consider the assets of methods and optimized methodologies, assisting with various software problems while simplifying and streamlining the solutions. Adopting a sharing mindset is crucial to successfully implementing an "agile manifesto" (Stellman & Greene, 2017).

The goal of an agile manifesto is to decipher the best path to solve software development challenges. As outlined in the manifesto, the team needs an agile mindset, so it can produce the most effective software for the end-user. Among the manifesto's guiding principles and values are delivering software early, continuously reducing timescales or iterations, welcoming changing requirements, and discovering the most timely and vital working software release for the stakeholders (Stellman & Greene, 2017).

As the most common approach to agile software methods, Scrum also promotes an iterative software process (cPrime, n.d.). Scrum software development focuses on the project's development and framework (Stellman & Greene, 2017). The three team roles in Scrum are the Scrum master, the project owner, and the development team, all of whom guide and develop the team's scope (see Figure 4). The workload of the software team divides into sprints, otherwise known as cycles. According to Stellman and Greene (2017), these cycles are equal timeframes, typically lasting 30 days or two weeks. The sprint begins with planning and identifying what features to address from the backlog of products to construct. These new items comprise the sprint backlog, which the team works to complete (Stellman & Greene, 2017).

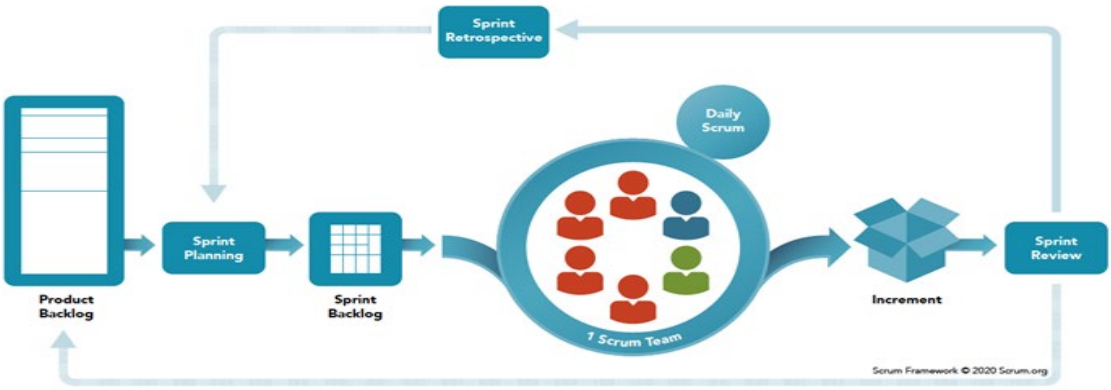

Figure 4. Scrum Framework. Source: Scrum.org (2020).

The benefits of using agile development involve collaborative processes that reduce software projects to smaller releases or sprints and automate the testing process (McCaney, 2020). Software functionality is tested against the operational needs of the weapons system to provide continuous delivery to the user (McCaney, 2020). This software testing path alleviates the requirement for authority to operate (ATO), a DOD manual testing procedure for software accreditation (McCaney, 2020). The DOD is slowly moving toward developing software through agile processes. As mentioned previously, these processes break down software features into sprints (McCaney, 2020). In each sprint, the software must pass the testing process and be released to upgrade the system for which it was designed. The results allow the DOD to respond rapidly to new threats with software upgrades (McCaney, 2020).

The limitations of these software methods may include bottlenecks in software capability training in some functional areas (Tate & Bailey, 2020). Furthermore, incorporating new software methods in some scenarios requires advanced and specialized architectures (Tate & Bailey, 2020). This aspect of incorporating agile software methods can add time to the delivery of the capability or system on contract. These additional requirements may initially slow the software development process until the updated training is satisfied.

3. C2D2

The C2D2 concept, developed by Vice Admiral Mat Winter, former F-35 program executive officer (PEO), applies an agile-like approach by delivering frequent incremental software and hardware improvements. C2D2 software was intended to address changing technology (Pant, 2019) and to release software incrementally to increase warfighting capabilities (Guertin, 2022). According to Pant (2019), the following goals of C2D2 illustrate the method's foundation in agile software development:

- Develop and test software changes;
- Identify deficiencies in testing;
- Shorten the design time;
- Complete design work on new data processors;
- "Establish laboratory and flight-test assets for modernization requirements verification" (para. 5); and
- "Address deficiencies from developmental testing, and conduct planning and systems engineering work for initial capabilities" (para. 5).

The benefits of C2D2 are similar to those of agile software development. For one, the C2D2 software process utilizes commercial best practices and develops software capabilities in smaller, more manageable increments while accelerating the capabilities to the fleet (Ludwigson, 2021). The C2D2 software method was designed to reduce delays in providing software capabilities needed by the warfighter. For example, with C2D2, system enhancements for the radar and collision avoidance systems of the F-35 program are delivered every six months (Ludwigson, 2021). However, the comptroller reported in FY2021 defense budget that C2D2 has failed to meet the desired improvements to the software capabilities (Guertin, 2022). While four incremental software releases—development, testing, identification of defects, and fixes for release—were originally planned, the C2D2 concept applied to the F-25 program produced multiple bugs in the

software and increased the increments to 10, thereby delaying the forecasted software drops (Pant, 2019; see Figure 5).

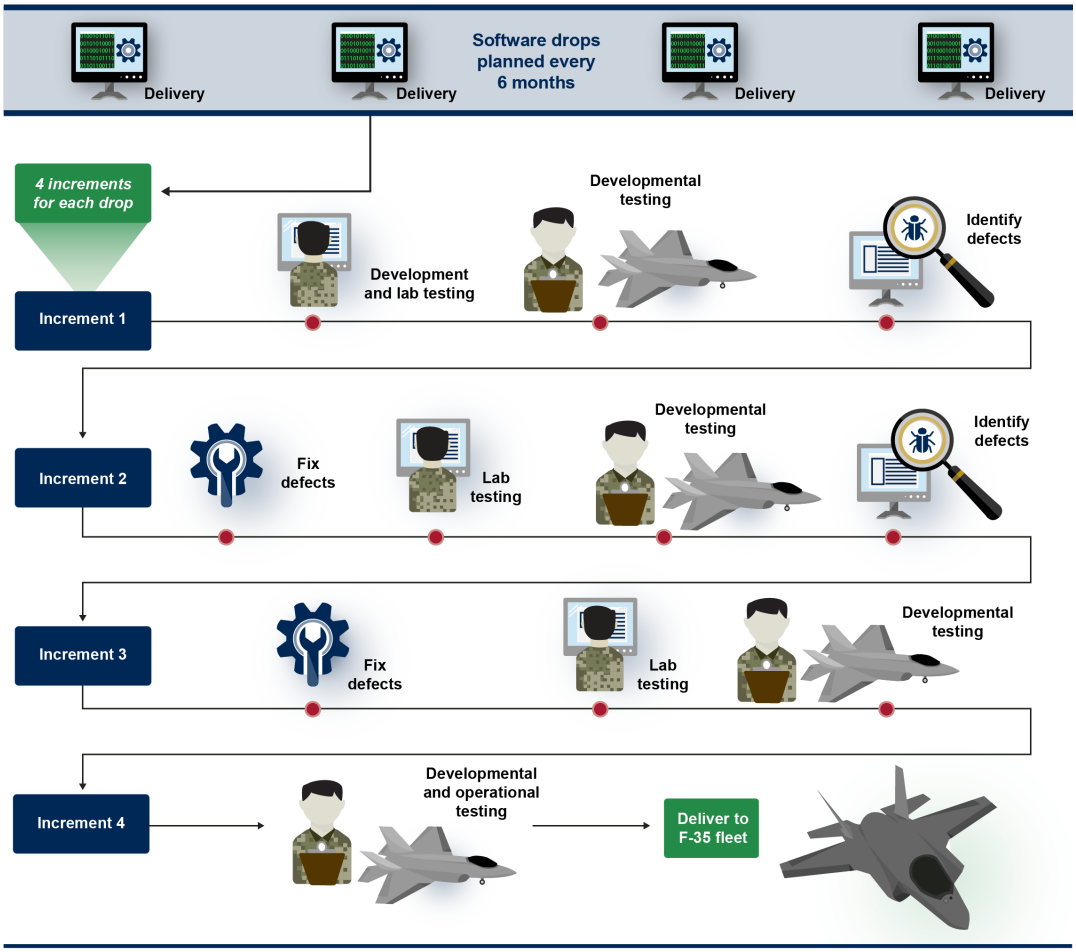

Figure 5. Notional C2D2 Iterative Development Testing and Delivery Schedule. Source: Ludwigson (2021, p. 30).

The expectation for C2D2 was to correct the current deficiencies by incorporating agile software methods and to test the capabilities, but the F-35 program saw only minimal benefits because the software was released as the technology matured (Sullivan, 2018). Robert Behler, the Pentagon's operational test and evaluation director, characterizes these limitations as follows: "The current Continuous Capability Development and Delivery (C2D2) process has not been able to keep pace with adding new increments of capability as planned" (Insinna, 2021, para. 18). Behler observes, "Software changes, intended to

introduce new capabilities or fix deficiencies, often introduced stability problems and adversely affected other functionality" (Insinna, 2021, para. 18). The C2D2 effort is also late to contract schedules. Congress has considered discontinuing current funding based on GAO reports that the delivery milestones are unachievable and offer minimal benefit to the F-35 program (Grazier, 2020). C2D2's rapid implementation and testing of software development code, vis-à-vis the waterfall approach, have been blamed for delays and unachievable capabilities for the warfighter (Ludwigson, 2021).

The F-35's program office and contractor Lockheed Martin are failing to deliver the software milestones established by C2D2 software targets, thus missing the program objectives. C2D2, the software development concept for the F-35 program's development, testing, and release requirements, is late to contract and in an overrun position of the contract costs (Ludwigson, 2021). This is a classic case of software needs misaligned with contract awards.

4. DevSecOps

DevSecOps is a collaborative software approach between developers and users that provides software continuously to adapt to rapid challenges by the customer (Walsh, 2021). As noted in a 2023 GAO report, DevSecOps "is an iterative software development methodology that combines development, security, and operations as key elements in delivering useful capability to the user of the software" (Oakley, 2023, p. 17). Furthermore, as described in a Red Hat (2023) article, "DevSecOps in software is a process which entails running multiple agile paths at the same time to develop the next state of an application." When applying this approach, stakeholders need to bear in mind that development, security, and operations are crucial aspects of the software user capability (Red Hat, 2023).

As documented in the SWAP Study, "DevSecOps is best described as the conventions and practices creating collaborative and communicative partnerships between development and operation groups" (McCaney, 2020). According to Chaillan, these practices incorporate automated development of software delivery and infrastructure changes within programs that adopt this method (Naegele, 2021). By incorporating the DevSecOps mindset into software integration, all stakeholders can engage in all phase

strategies and create interoperability. See Figures 6 and 7 for a comparison of the "old" and DevSecOps methods of software development.

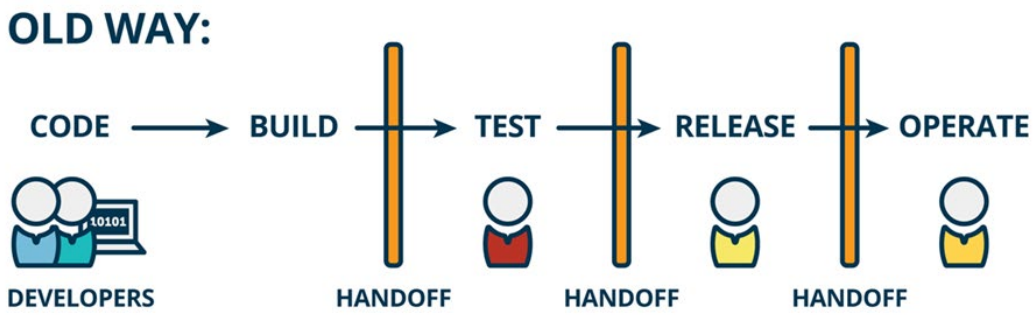

Figure 6. Current Software Development Process.
Source: General Services Administration (n.d.).

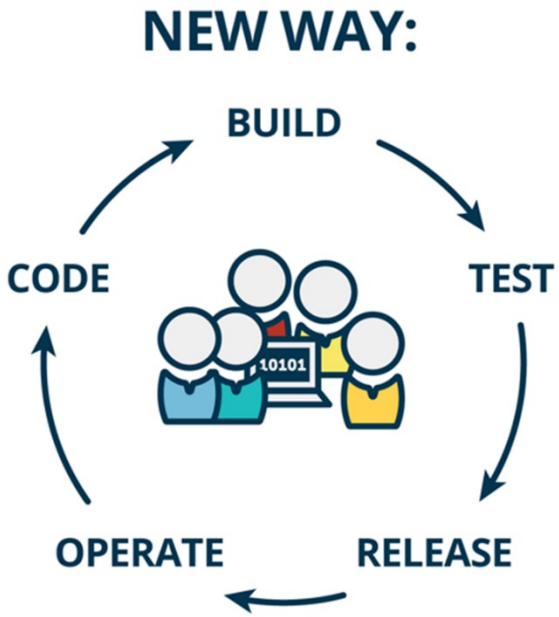

Figure 7. DevSecOps Software Development Process.
Source: General Services Administration (n.d.).

The following are benefits of using DevSecOps, as demonstrated by Chaillan's DOD enterprise initiatives:

- Collaboration among the Office of the Secretary of Defense, Acquisition and Sustainment; the Office of the Chief Innovation Officer (CIO); and the services;
- DOD program integration with agile software at a faster pace;
- Timeliness and centralization in using a DOD Centralized Artifact Repository (DCAR), Iron Bank, with containers (see Figure 8);
- Faster mission software development using Kubernetes methods (see Figure 9);
- Resilience (the ability to restart the process);
- Baked-in security (automatic checks for issues);
- Adaptability, which minimizes software downtime;
- Automation in designed infrastructure models;
- Auto-scaling based on the user's need;
- Abstraction layer to eliminate software locked into one platform (Chaillan, 2023);

Furthermore, DevSecOps addresses the following barriers in software development:

- Culture—Removes gaps from differing agendas and promotes a willingness to change;
- Technology—Adopts best practices developed to keep pace with warfighter needs;
- Training—Educates constantly on the best methods; and
- Contracting—Adopts the agile acquisition framework (Skertic, 2019).

The Iron Bank DCAR with containers provides a central software capabilities location to certify, secure, and develop software (Chaillan, 2023). This approach enables access to the DOD community authorized in the supply chain for all software engineers. This software initiative is important because it speeds up development by automating the process in a secure environment (Office of the Chief Software Officer, 2019; see Figure 8).

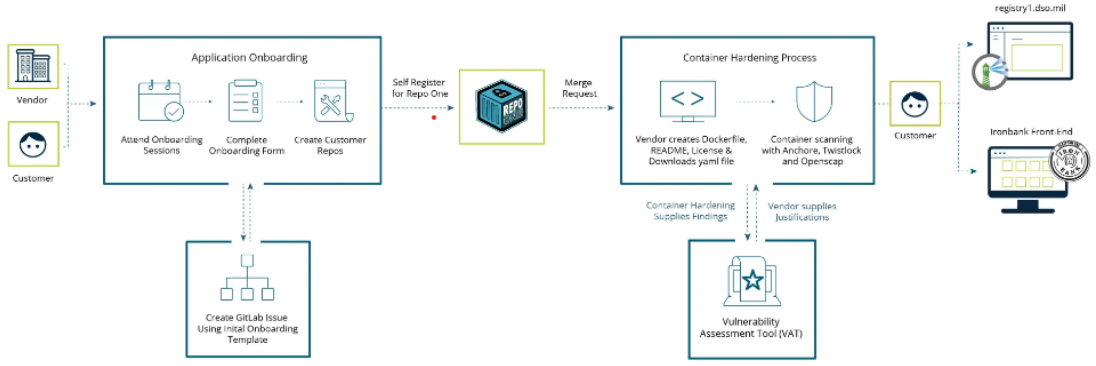

Figure 8. Iron Bank. Source: P1 (n.d.-b).

Kubernetes is a software development infrastructure that uses development-like pods that act as small containers of data. This platform, which eliminates the program's reliance on one software supplier, is authorized using a centrally located repository with security designed into the software (Chaillan, 2021a; see Figure 9).

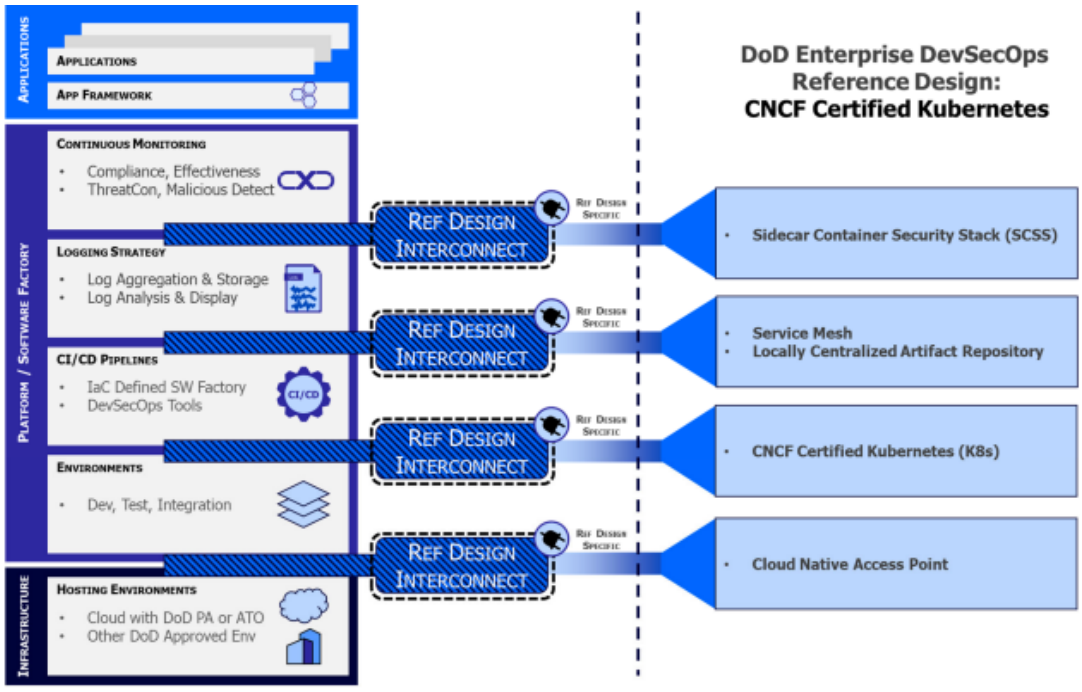

Figure 9. Kubernetes Reference Design Interconnections. Source: DOD (2021a, p. 11).

When DevSecOps integrates DCAR/Iron Bank attributes using Kubernetes, every stage of software development is automated through a commercial-like cloud method (McQuade et al., 2019). As shown in Figure 10, a DevSecOps technology stack merges IT capabilities with Cloud One commercially designed methods and P1 services (Chaillan, 2019). P1 is a software factor and service that allows access to Cloud One, a government software hosting service (Chaillan, 2023). Cloud One supports the Air Force and provides software testing, production, and computing applications (Chaillan, 2023).

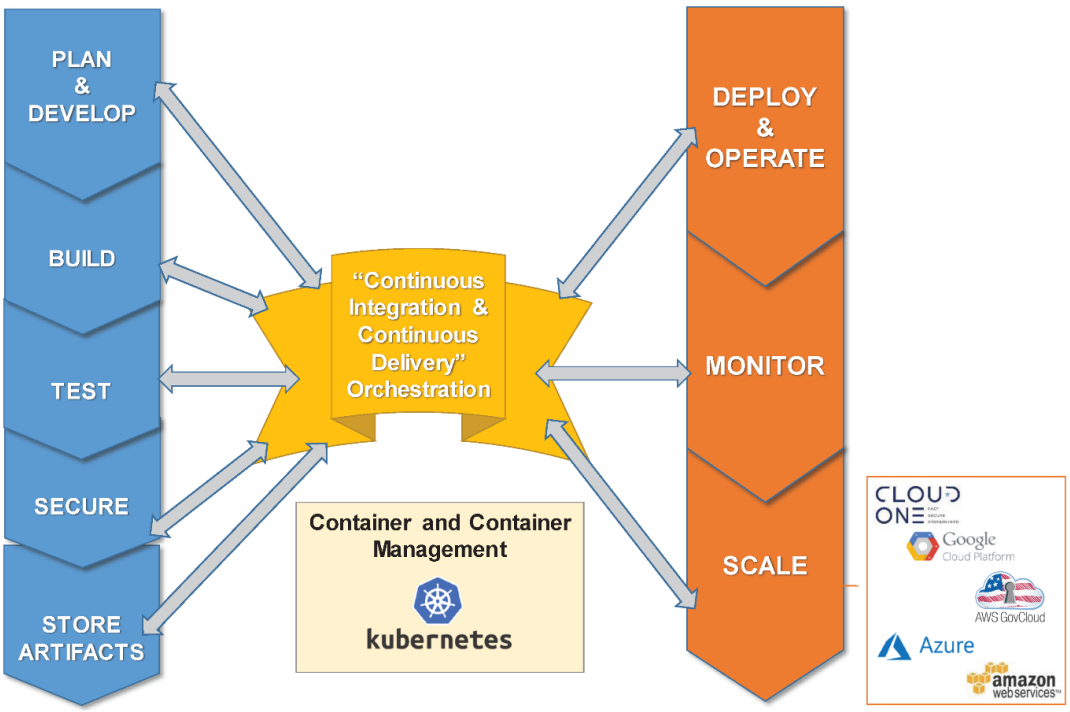

Figure 10. DOD Enterprise DevSecOps Technology Stack. Source: Chaillan (2019, p. 16).

Additional benefits of DevSecOps include the increased security of software over the outdated waterfall method (see Figure 11). When using DevSecOps, software security concerns emerge earlier in the development phase than when using the waterfall method. According to the DIB, security in software is an essential element (Oakley, 2021b). The DOD benefits from the DevSecOps process because it provides better reporting metrics, testing thresholds, and automation. Automation is an advantage to programs because it removes the manual testing process, which requires ATO (McCaney, 2020). In DevSecOps, ATO forms a continuous process of approving and certifying that the software is ready to use (McQuade et al., 2019). This repeatable process is yet another key advantage of DevSecOps (McLaughlin, 2019).

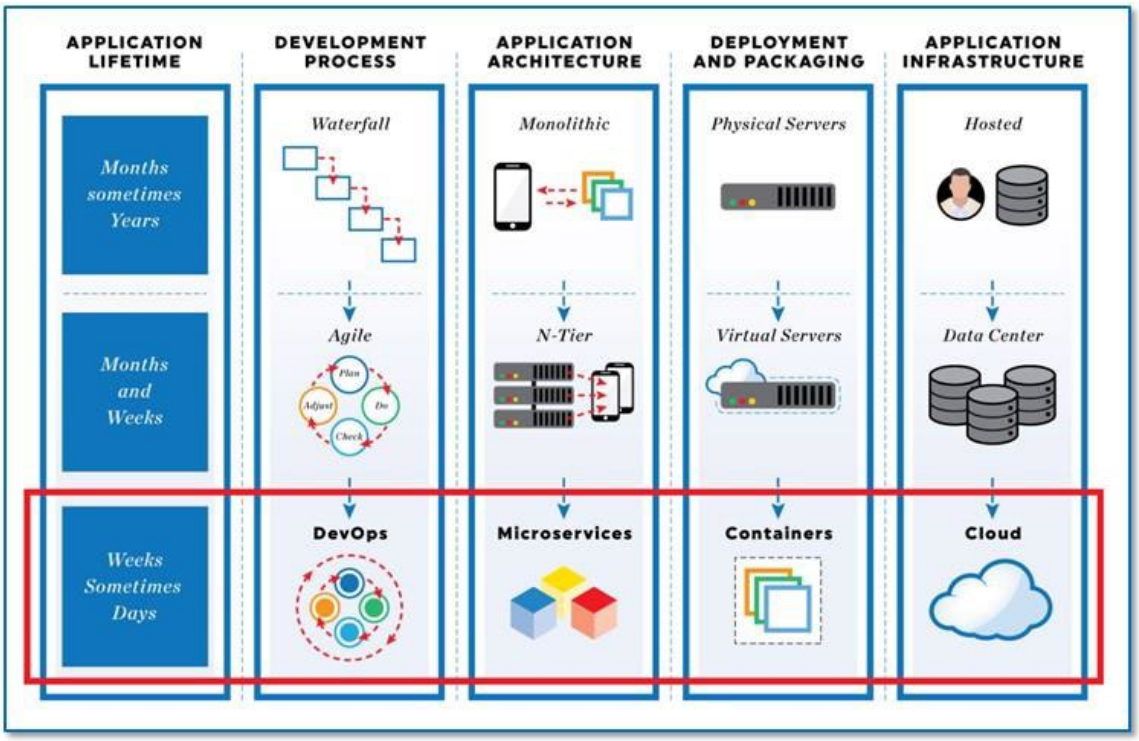

Figure 11. From Waterfall to DevSecOps. Source: Chaillan (2019).

A 2019 DOD memo documents the limitations of using DevSecOps (Deasy & Lord, 2019). Due to its unique structuring, timing, and challenges, DevSecOps could not be applied in its existing form. For example, defense regulations require annual operational releases, which are infeasible, based on an interview with a representative from the F-22 program office (B. Burton, personal communication, September 15, 2021). Furthermore, the Navy's F-22 release process cannot support anything more frequent than an annual release. Moreover, the retrofit process, which includes only operational flight program modifications, can be lengthy and must be balanced with F-22 combat-coded fleet readiness. Thus, the F-22 program must weigh the capabilities included in each recurring software release with its current inability to meet the annual release requirement highlighted in DOD instruction. The F-22 program office expects that the guidance will continue to evolve and be reevaluated as it does (B. Burton, personal communication, September 15, 2021).

5. Hybrid/Mixed

A 2021 GAO report describes this hybrid approach as a "combination of two or more different methodologies" or "systems used to create a new model" (Oakley, 2021b, p. 4). According to the F-22 program's General David Basset, the Defense Contract Management Agency commander, aspects of existing programs make it impractical or impossible to use things or transition to start using updated agile software methods (Basset, personal communication, July 15, 2021; B. Burton, personal communication, September 15, 2021).

B. SOFTWARE FACTORIES

A software factory is an organized software development approach associated with a supply chain for producing software solutions (DOD, 2021b). This approach can be assimilated from the manufacturing process used for hardware items. Software factories comprise assembly lines or pipelines consisting of different environments that rely on hardware-like software (DOD, 2021b; see Figure 12).

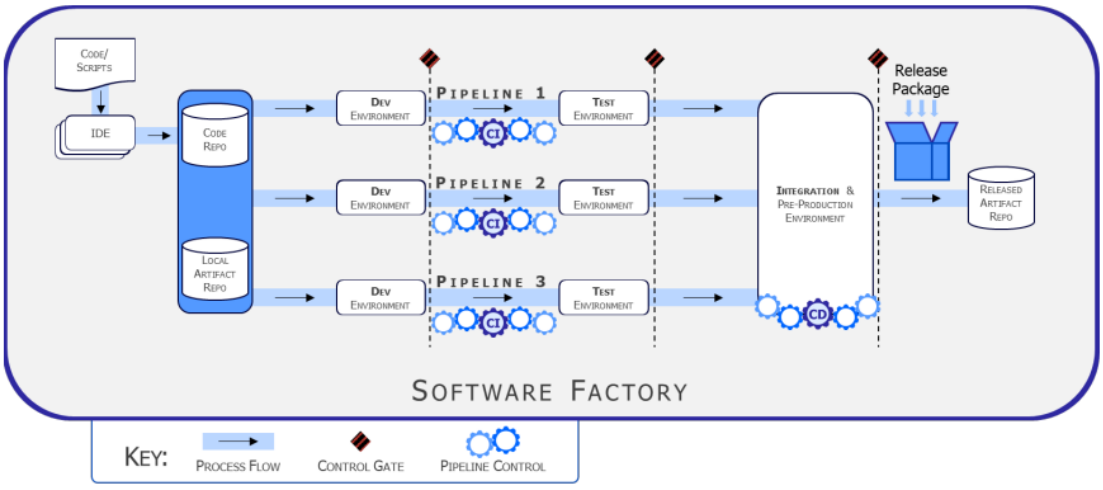

Figure 12. Software Factory Construct. Source: DOD (2021b).

Two of the DOD's software factories create repeatable paths for developing and delivering software applications (VMware, n.d.). One software factory addresses

cybersecurity using a toolchain that identifies rules constituting vulnerabilities in the software at every level. Through a style sheet, the software factory checks lines of code for rule violations and communicates them to the programmers (Eckstein, 2021). The National Institute of Standards and Technology supplies guidelines as a starting point for the cyber rules established in a software factory environment. This repeatable process disseminates the new code and identifies new exposures. The logic errors produced from testing user-created capabilities automatically relay new issues to the team.

Software factories contribute options to the DOD (Chaillan, 2021a). Other important elements that software factories supply include quick, consistent code releases and refined development using innovative techniques (VMware, n.d.). Their approaches include DevSecOps, agile software methods, and cloud technologies, which eliminate excess infrastructure (VMware, n.d.). Software factories support innovation by combining both software tools and teams, and their development practices standardize and reuse code (VMware, n.d.). In sum, software factory teams use knowledge-based approaches to produce more effective software.

The software services and data needed for the rapid acquisition of critical software systems that support the warfighter require a collaborative effort among all stakeholders, including members of industry and the military. Today, limited software personnel exist in the military (McQuade et al., 2019). Indeed, most software coders opt to work in the commercial sector rather than enlist in the services (Oakley, 2020). According to Beachkofski and Helfrich (2021), it is vital to integrate military service software professionals into collaborative efforts to guide technological decisions.

Software factories incorporate the needs of the entire defense enterprise, altering the software frameworks and team collaboration efforts across many programs. There are several identified benefits of software factories, such as P1 and Kessel Run:

- Platform services that enable rapid cross-platform application and cross-environment portability;
- Common security authentication services;

- Common policy enforcement services;
- Consistent security-monitoring services;
- Continuous consumption of the industry's leading platforms; and
- Collaborative cost savings by weighing sound buys against leases against built business cases and decisions (Chaillan, 2023; Office of the Chief Software Officer, 2019).

Figure 13 depicts the Air Force's software factory hubs, which represent a single-platform ecosystem for DOD weapons systems. The various innovative hubs allow interaction with software factories across the country.

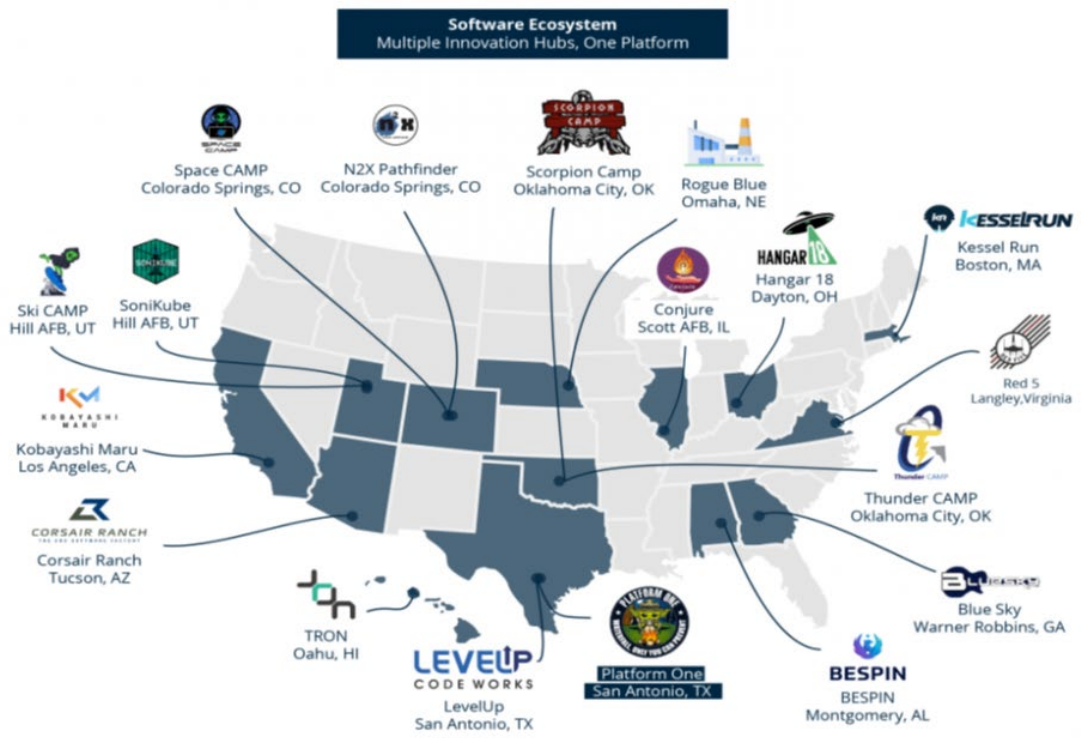

Figure 13. Software Factory Hubs. Source: Office of the Chief Software Officer (2019); P1 (n.d.).

1. Kessel Run

Kessel Run is an Air Force-owned software developer in Boston, Massachusetts, that supports all capabilities of the warfighter (Hitchens, 2020). It scales software to design, manufacture, and operate weapons systems. Kessel Run practices have adopted lean development, extreme programming, and user-centered design (Office of the Chief Software Officer, 2019). Lean development allows simple software development of the most critical items by the quickest method possible. Extreme programming integrates the speed of software development. Software testing and programming are linked and capable of pivoting to any changes in code that arise. User-centered design is an approach that highlights the value of the user to the team. The iterative nature of software is the main driver that validates software conventions based on new testing or research (Office of the Chief Software Officer, 2019).

In addition to its development practices, Kessel Run values continuous design through delivery, feedback, and learning. Its embrace of these standards allows responsiveness to unknown operational threats and enables a continuous feedback loop. Kessel Run has been designed to recognize and mitigate risk. Its seven programs work to transform, develop, and provide software solutions, focusing on command and control (C2) capabilities intended to support its principles and values (Office of the Chief Software Officer, 2019).

Kessel Run's motto, "the speed of need," promotes the integration of next-generation software technology into weapons systems (Kessel Run, 2021, para. 3). The goals alter the software paths to build and deliver capabilities (Office of the Chief Software Officer, 2019). Kessel Run uses proven software methods developed by the commercial sector (Office of the Chief Software Officer, 2019). This software factory has proven successful in collaborating with military service members in major weapons systems, resulting in innovative, agile software methods that are responsive to users in any programming domain at any point in time (Office of the Chief Software Officer, 2019).

The benefits of using Kessel Run depend on the type of program, operational need, platform, and applications identified by the program and end-user. One platform in Kessel

Run's arsenal, the All Domain Common Platform, manages C2 globally with a vision of an all-operational domain (Hitchens, 2020). C2 is how operational decisions are made (Hoehn et al., 2022; see Figure 14). The goal of this platform is to provide network communications in an end-to-end system. This platform is designed to simplify software development, scaling software features to integrate new applications and services more rapidly for the end-user from any location around the globe (Hitchens, 2020). The resulting data help to improve program performance, as the mission's applications are produced quickly (Office of the Chief Software Officer, 2019).

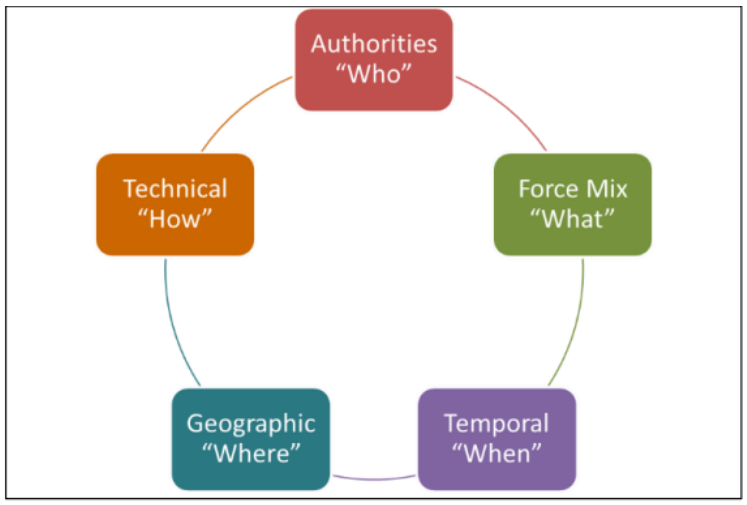

Figure 14. C2 Model. Source: Hoehn et al. (2022).

2. Platform One

P1 is an Air Force–based modern cloud platform software factory. Its software solutions begin with a 90% day-one mindset instead of starting from nothing (P1, n.d.-a). The mission of P1 is to transform how software is delivered to the warfighter. P1 incorporates innovative software frameworks, collaborations, and security measures to develop software solutions. The vision for P1 is for the platform to accelerate the capabilities of the DOD, the goal being to retain DOD dominance over adversaries by speeding up software deployability (P1, n.d.-a).

P1 (n.d.-a) incorporates the following core principles into its mission: software to scale, accountability, simplicity, a different culture, human resources, and training. The P1 concept of software to scale identifies the features that are not satisfied until all requirements are met, including software testing, documentation, automation, and training for all stakeholders (Office of the Chief Software Officer, n.d.). The next principle is member accountability to one's own work, words, and access to the team, which succeeds or fails in every detail. A misunderstanding is not an excuse to ignore individual responsibility; thus, a team member is successful only when the goals outlined by the team are met (Office of the Chief Software Officer, n.d.). The team's obsession is writing baseline code for the DOD weapons systems at every level (Office of the Chief Software Officer, n.d.).

A belief in accountability sets up the unique culture adopted by the P1 team mission. The status of the team over the individual and the idea that mistakes are learning opportunities depart from the normal credence of many MDAPs. Admitting imperfections and recognizing the value of every team member are encouraged in P1. Moreover, the team is encouraged to think creatively and foster changes that might deviate from normal protocol (Office of the Chief Software Officer, n.d.).

Another benefit of P1 is having the right human resources. Everyone in P1 writes code—there is no exception. This edict provides the quintessential cross-training for staff and allows coverage for anyone unavailable for any reason. The code must go on. The P1 team promotes decision-making by members, and the individual with most of the data needs to execute the choice. For example, for technical decisions, the expert in this field makes the decision regardless of rank (Office of the Chief Software Officer, n.d.).

The training principle promotes continuous learning. The P1 culture strives always to improve the brain inputs for each member, placing the greatest emphasis on investment in people. The priority is to foster individual self-improvement. This momentum facilitates paying it forward to the entire team (Office of the Chief Software Officer, n.d.).

3. Limitations of Software Factories

Software factories are not appropriate for every program. For one, their inability to synchronize structure contracts with hardware and software architectures is not commonly communicated to the program's decision-makers. Also, they lack transparency between the government's planning, programming, and budget execution and DOD stakeholders. Moreover, such a software path misaligns with the colors of money typically used for defense acquisition programs (Francis et al., 2001). In some circumstances, limitations lie in the DOD's software acquisition choices, which follow a linear path for software development (DIB, 2019). This route incorporates the legacy model of estimating software lines of code for each phase. The current software environment limits the ability to match, detect, and defer U.S. adversaries with most legacy weapons.

As an Air Force CIO, Chaillan did not approve of the DOD's bureaucratic procedures. The absence of an acquisition team to award relevant software contracts is a missed opportunity for DevSecOps across the DOD. The acquisition team works with the program's stakeholders and exercises the purchasing path to pursue. Normalizing the integration of software factories into the DOD was the desired path of Chaillan. The capability of "pushing over the air new software updates to weapons systems," including rapid software designs, was another goal (Chaillan, 2023, p. 7). Its misapplication of software factories is a symptom of the DOD's ailing acquisition culture (Chaillan, 2021a). Another limiting factor is that the DOD continues to invest in parallel software architectures. These software system stacks do not provide the desired benefit. Not all DOD systems are adaptable to iterative development. Many legacy DOD platforms consist of controls that no longer require development. Yet legacy systems that surpass their predicted life cycle may benefit from some type of software automation.

The limitations of the waterfall method are exposed by other software pathways (Walsh, 2021). In 2019, the DIB reported that the traditional waterfall approach was inadequate for some programs, particularly because it could not compete with the potential cost savings of iterative software development (Walsh, 2021). For example, Agile/Scrum and software factories allow an incremental approach to smaller software blocks evaluated by the user community (McQuade et al., 2019). Indeed, numerous commercial entities

migrated from the waterfall method long ago and established better ways for validating software.

The limitations of software factories are related to the marketability and value of their services to programs. One factor is a program's capacity to identify which software factories are at scale (Bailey & Tate, 2021). The program must decide whether its coding conditions—the environment, customer, innovation, modularity, and impact of mistakes—warrant moving to such an iterative development software pathway (Bailey & Tate, 2021). The value that software factories offer to legacy programs includes that of the current software's development and operational effectiveness. Furthermore, when the weapons system encompasses foreign allies and foreign partner nations, IP ownership may prove problematic for a software factory. Likewise, complex software language conversions may be limited in their operational aspects when different codes and architectures are not considered or known by the developer.

Kessel Run's limitations reflect whether the program's culture accepts changes to its existing software pathways. Kessel Run's real-world applications integrate security features using the DevSecOps process. Its ability to limit software weaknesses and deliver code quickly to the field depends on the authority of use (Office of the Chief Software Officer, n.d.). For example, the F-35 program's vast allies and partner nations—Australia, the United Kingdom, Belgium, the Czech Republic, Denmark, Poland, Switzerland, Greece, Norway, Italy, Finland, Canada, Germany, Japan, the Republic of Korea, and Israel—may have different rights and partner or foreign military service agreements that prohibit integrating certain software changes for security reasons (DOD, 2017). Air Force Major Rachel Mamroth, the deputy chief of acquisitions for Kessel Run, recognizes both the opportunities and challenges of migrating all software responsibilities to software factories (personal communication, February 25, 2022).

Another limiting factor of software factories is their lack of ownership and IP management in developing new software. IP is intangible property that includes information, products, or services protected by law (DOD, 2019b), but software and technical data are intangible property. The government does not own adequate IP rights, according to Ellen Lord, former undersecretary of defense acquisition and sustainment,

who created guidance for retaining IP ownership in MDAPs (DOD, 2019b). Experts in the field raise valid concerns about IP ownership issues given the government's resources for managing millions of lines of code. Major defense contractors and their subcontractors benefit from the government's developing and working on software code (McQuade et al., 2019). A program might decide to retain IP through the contractors who possess the code and incorporate the new methods. In addition to new capabilities that address threats encountered by the warfighter, a faster response within the acquisition community requires that resources be supported by the government and converted into a new tool for the program's arsenal.

When software code is not developed by the overseeing office, the office fails to fully own all the IP rights. The changes implemented in DOD Instruction 5010.44 involve educating the program teams to seek evolving software resources capable of delivering rapid software architectures (DOD, 2019b). The new software pathways do not work for everything. The DOD has only a handful of software factories, and the factories do not have the capacity for every program. Besides, defense personnel are not accustomed to working with these software factories. Nevertheless, security concerns minimize the capability of leveraging software factories in some applications (Bailey & Tate, 2021).

C. FACTORS LIMITING INNOVATIVE MDAP SOFTWARE PRACTICES

Many factors limit innovative software practices in MDAPs and prevent the programs from considering any innovative software pathways. These factors include culture, communication, knowledge-based practices, resources, life cycle, stovepipe methods, and hierarchy levels. These seven factors—combined with the linear structure of DOD acquisition, development, and operational testing and evaluation—limit the full scope.

1. Culture

Change is difficult for people and organizations to accept and integrate due to the rigidity of the structure, bureaucracy, egotism, education, or the leadership's agenda (Thornberry, 2023). Creating an innovative environment and fostering an innovation mindset offer advantages for service members.

One potential reason for the lack of integration of new pathways lies in the military's chain of command. The military's organizational norms involve leading and not questioning authority. Additionally, it is standard practice not to encourage mistakes in order to learn from them. This type of mindset is outdated (Thornberry, 2023). Fear exists within the ranks when the services champion innovative methods (Coram, 2020). Studies show the higher the rank of a military officer, the less likely she will initiate change (Gleason, 2022). The military promotion process includes silent yet understood decision protocols. To illustrate, once an officer becomes a colonel, the next promotion the board considers is her eligibility for general. This critical timeframe signals the likelihood of this promotion, so the impact of fostering innovation at this level weighs heavily on the individual who embraces both but must decide her path: either her career or change within the military. It is often not both (Coram, 2020).

The DOD desires to develop high-tech weapons, but it is risky and expensive (Carter, 2019). The lessons not learned from failures were manifest in the hypersonic missile program, which the DOD decided to halt (Thornberry, 2023). Leadership was perplexed when China and Russia ramped up missile development efforts (Thornberry, 2023). The need to keep pace with U.S. adversaries requires developing innovative arms. In generating technological advancements that align new requirements with the DOD's mission, the DOD must recognize that an initial failure often precedes the success of a program (Carter, 2019). The pause in this effort now places the United States at a disadvantage due to the cultural norms of the DOD mindset (Thornberry, 2023).

While some segments of the DOD are using updated systems to drive modernization, there are still prevalent cultural barriers limiting the agility and private–public sector collaboration needed (Thornberry, 2023). The club that champions admittance is a type of fraternity that incorporates other generals' inputs into the decision (Coram, 2020). Nonconformist leaders are viewed as outliers of the current behavioral leadership norms (Coram, 2020). Moreover, according to the officers' colleagues, the rank they achieve denotes the belief in the type of leaders they are and the extent of their capabilities (Coram, 2020). The rules of engagement in the military create a potentially toxic atmosphere when officers buck the cultural climate of decision-makers (Coram,

2020). This environment is based on the type of decisions executed by up-and-coming innovative thinkers. Working outside the leadership's unspoken cultural norms might be a career killer if the decisions do not align with a leader's agenda (Coram, 2020). Innovative original thinkers who challenge the purpose and scope of generals' and admirals' programs tend to be ostracized regardless of their overall contributions. For example, John Boyd, who revolutionized aircraft fighter tactics and maneuverability in aeronautics, was an outspoken innovator who created enemies and was treated as an outcast (Coram, 2020). The services ultimately bypassed his promotion to general.

The benefits of establishing a new culture include a collaborative environment among all stakeholders. Some aspects of agile methods are scaled into organizations, and initiating communications with the team to listen and validate changes is a critical aspect of culture shifts.

2. Communications

Communication problems result when MDAP requirements are ill defined. The miscommunication of requirements may be unknown at the time of contract award. When there is a lack of open communications, there can be new challenges incorporating software changes (Miller et al., 2022). An inflexible organizational hierarchy, cultural inertia, and security requirements might pose challenges for open communications (Miller et al., 2022). These challenges highlight the importance of communicating realistic program requirements and the opportunity for adequately forecasting the costs, schedules, and resources needed. The program depends on both software and hardware for successful operational performance. This dependency remains throughout the program's life cycle and requires the communication of any upgrades, problems, or corrections to mitigate deficiencies disclosed in the system (Ludwigson, 2021). The entirety of needs is not adequately revealed to or by all stakeholders in the planning and development stages (Mortlock et al., 2022). The ramifications of poorly defined requirements involve delays in meeting the weapons functions and testing milestones. These delays, whether not communicated or filtered through messages, create additional problems that affect all

stakeholders' obligations, including multiple levels of cost, delivery, performance, and mission milestones in the entire program (Insinna, 2021).

The ineffective communication of program requirements contributes to the misalignment of hardware and software program performance, costs, and overall capabilities (Mortlock et al., 2022). Accurate communications enable insight into the forecasted timeline and performance to adjust the plan for the next step of the acquisition (Mortlock et al., 2022). Various program actors benefit from budgets allocated to new and existing programs when only the upside is communicated. Yet, comments from anonymous DOD software engineers on the fallout from inaccurate communications in software pathways facilitate software change (Miller et al., 2022). These examples highlight another factor in software delays—the trickle-down effect on the hardware from software capabilities (Walsh, 2021). This problem has not been resolved nor has the time been estimated to address it. The milestones of MDAPs require unfiltered communications to report the timelines to meet each percentage of the driver (see Figure 15).

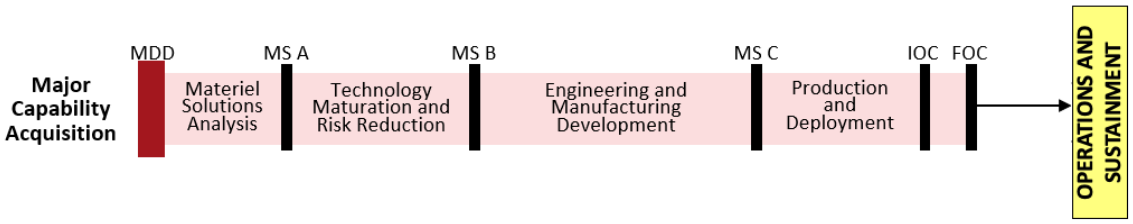

Figure 15. Acquisition Milestone Process. Source: DAU (n.d.-c).

The trend of varying sizes and complexity of DOD programs may contribute to the amount and quality of communications among stakeholders. Missed opportunities for communication are the result of both individuals and agendas. The individuals and agendas are not aligned with the intent of creating an open environment to collaborate. This setting is ineffective in supporting the desired program outcomes. Underlying personal biases and hidden agendas stifle open, honest communication (Gleason, 2022). Sharing expectations across disciplines fosters a realistic measure of achievement, which is one goal of the new Adaptive Acquisition Framework (see Figure 16). This framework's tenets are to simplify,

tailor, empower, use data analytics, manage risk, and highlight sustainment drivers (DAU, 2019).

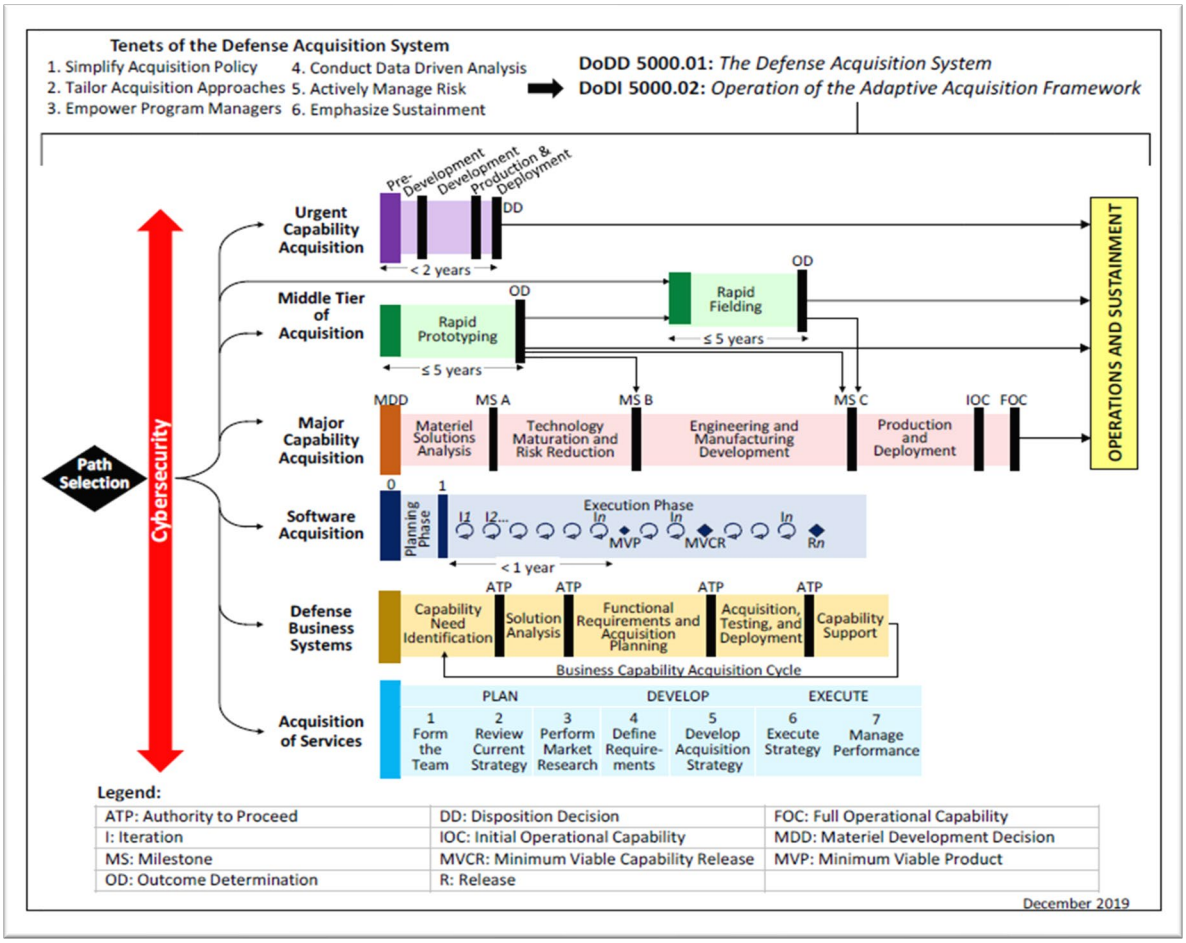

Figure 16. Adaptive Acquisition Framework.
Source: DAU (n.d.-b)

These tenets allow program managers to exercise the option of using either a single pathway or a combination of acquisition pathways (DAU, 2019). The communication importance connected by the Adaptive Acquisition Framework is driven by milestone achievement (Mortlock et al., 2022). If they are not adequately communicated, the critical software requirements will fail to meet the milestone criteria (Mortlock et al., 2022). These options facilitate the program's ability to exercise an analysis of alternatives that considers which pathway best suits the program's needs. Each member is accountable for

determining the performance criteria that add to the system's achievement of outcomes. The stakeholders may benefit both in the short and long term from improving reliability, maintenance, costs, and resource allocation and eliminating obsolescence.

3. Knowledge-Based Practices

A DOD acquisition approach to improving the outcomes in programs applies knowledge-based practices (Oakley, 2019). This approach incorporates lessons learned for programs that have insufficient knowledge in the development phase (Oakley, 2019). It comprises three acquisition phases—technology development, system development, and production (Oakley, 2019). The three phases align with three points of knowledge, whose criteria differ (Oakley, 2019). Under a knowledge-based approach, technical readiness is required before a program advances to the next phase. Other areas of focus include stable design, testing, and integration capabilities (Oakley, 2019). These practices and oversight can provide "rapid fielding of warfighter capabilities" (Oakley, 2019, p. 5).

The problem with incorporating the knowledge-based approach in software acquisitions and development is the lack of skills across DOD personnel to address maturing technology. The current program structure does not align with the design of performance capabilities; neither does the DOD understand how to integrate emerging technologies within a knowledge-based concept (Oakley, 2020). Defense programs have attempted to restructure themselves to align with knowledge-based approaches, but the result is that some programs are canceled when the restructuring fails to align with updates to new milestones (Oakley, 2019).

With the knowledge-based concept, a successful program gains insight into and confirms the maturity of the technology intended to stabilize the design and control of the production processes (Oakley, 2019). The knowledge-based approach integrates three points to provide information to the program to facilitate the decision-making process. The first decision point is to identify all program needs and resources to invest in product development (Oakley, 2019). This point includes identifying all resources that can achieve all the customer's requirements (Oakley, 2020). The second point identifies whether the design is stable and can meet the performance criteria (Oakley, 2019). If the criteria are

met, the program may invest in building a product prototype and testing it. The third and final decision point forecasts the elements of the program—the cost, schedule, and performance. According to the GAO, if this criterion is met, the program may begin manufacturing the first components (Oakley, 2020).

According to the GAO, some of the DOD's historically poor-performing programs serve as opportunities for using knowledge-based approaches (Sullivan, 2008). The GAO recommends aligning the budgets of major programs to fit with the resources available to develop a strategic priority with aspects of fiscal year appropriations that predict realistic program outcomes (Oakley, 2020). According to one GAO report, the programs that fail to adopt the knowledge-based approach will be reduced or terminated (Oakley, 2020). Such an outcome helps to align dollars and resources with DOD objectives (Oakley, 2020). Those programs that ignore the GAO's recommendations toward a knowledge-based approach might experience limited performance, higher costs, and schedule delays.

Secretary of Defense Lloyd Austin is quoted in the fiscal year 2023 *Defense Budget Overview* as saying, "Strategic readiness is improving the Department's understanding of the comprehensive and cumulative impacts of the decisions we make today on our future readiness" (Office of the Under Secretary of Defense (Comptroller)/Chief Financial Officer [OUSD], 2022, p. 3-8). According to the OUSD (2022), the Strategic Readiness Framework keeps the department's eyes "on the horizon, ensuring the urgent competing demands of the present are carefully balanced with the importance of preparing for the future" (p. 3-8). Furthermore, the *Defense Budget Overview* "identifies the lapses within the DOD where there are misaligned resources to strategy, strategy misalignment to policy, and policy misalignments to the will of the American people" to recommit efforts to learn and implement corrections into business processes (Austin, 2022).

A knowledge-based approach to software-centric programs would support the program at each knowledge point (Oakley, 2019). Each year, the GAO reports whether certain MDAPs are over budget and which ones have failed to learn from their mistakes (Sullivan, 2008). The undesirable results include cost increases, schedule fluctuations, and minimal software development practices (Francis et al., 2001; Ludwigson, 2021; Oakley, 2019, 2020; Sager, 2021; Walsh, 2021).

These programs miss opportunities when their leadership chooses not to use knowledge-based approaches. They have the potential to adopt best practices developed through the proof of concept to improve an underperforming program. For example, the first missed opportunity is in identifying the program requirements and matching the available resources before deciding whether to invest in product development. The program creates risks, issues, and opportunities for each critical pathway, including realizing the desired outcomes identified through the first approach. The second missed opportunity occurs when stabilizing the design and meeting performance criteria during aircraft manufacturing, when missed milestones can be identified. The third missed opportunity is manifest in the elements of the program—cost, schedule, and performance. If this criterion is not satisfied, program administrators may decide not to begin manufacturing the first components. When aircraft programs do not apply a knowledge-based approach, their administrators do not understand the connections between technology maturity, stability of the design, and producibility, thus delaying the schedule, decreasing performance, and increasing costs (Oakley, 2020).

MDAPs that adopt the GAO's recommendations relating to knowledge-based acquisition concepts benefit from understanding the critical technology maturity requirements before moving on to the next milestone (Oakley, 2019, 2020). The program offices and contractors will identify the risks before they become an issue for the program, thus forcing the DOD to revisit program expenditures and expectations in complex programs to detect problems and leverage opportunities identified early in the entire process. The final decision is whether the return on investment outweighs the savings from canceling the entire program.

4. Resources

One contributing factor is the problem of securing, retaining, and attracting the appropriate human capital. In their report, Francis et al. (2001) warn that human capital does not align with the needs of programs to improve cost, schedule, and technological maturity. The GAO identifies three key problems: resource gaps addressed before the next phase of the program, inflexible stakeholders, and ill-defined roles and responsibilities

among the customers and developers (Francis et al., 2001). Programs cannot respond to emerging threats if the appropriate technologies cannot be developed because of the lack of adequate software development resources (Francis et al., 2001). A prime example of this problem is the continual misalignment in aircraft systems, which is especially true of the DOD's F-35 program. The program's missed milestones can be attributed to all three issues (Ludwigson, 2022). Prioritizing the program schedules is essential for meeting milestones but misapplied in meeting contract requirements (Capaccio, 2022). The next area is the lack of flexibility in a schedule when the program is already experiencing delays and contract realignment is not an option. Unavailable software resources due to milestone delays constrain program execution, especially for software (Tate & Bailey, 2020). Moving from the F-35 Automated Logistics Information System (ALIS) to the Operational Data Integrated Network (ODIN) has hindered the F-35's capabilities because the resources and development paths are inconsistent with the program's milestones (Hoehn & Gertler, 2022). Finally, the disconnect between the who, what, when, and where is not forecasted to meet realistic timelines.

Lockheed Martin, the primary contractor for the F-35 program, is constantly under scrutiny because it manages the software aspect as an area of development, not production. The F-35 aircraft is a flying computer that operates with more than eight million lines of software code (Capaccio, 2022). However, the DOD does not adequately control all the IP rights for the 24 million lines of software code that the F-35 requires (Hoehn et al., 2022). Without the required software code, the F-35 cannot communicate successfully with all the support systems (Hitchens, 2020). This issue limits the alignment of resources between the contractor and the government.

There are missed opportunities for both the DOD and industry when adequate resources are not available, including the ability to protect profits and deliver goods and services. These requirements misaligned with the skillsets of resources create a greater risk for all stakeholders. If all program phases accurately align resources with timelines, they allow time for growth and completion of a task. The two forces are compulsory, not to be oppose one another but to accept opportunities to integrate new prospects for future growth and profits.

5. Life Cycle

The program's life cycle is defined as multiple stages consisting of development, production, and sustainment (McQuade et al., 2019). The life cycle of a program, as defined by defense acquisition projects, is the basic structure broken into phases (Rendon & Snider, 2019). The sustainment phase is also known as operations and maintenance of the program (McQuade et al., 2019). As noted on the DAU's website, "The life cycle process takes the program through research, development, production, deployment, support, upgrade, and finally, demilitarization and disposal" (DAU, n.d.-a, para. 3). In the organizational environment, the life cycle activities require the capability of supporting the use of innovative software practices. The DOD has reported that out of 18 programs implementing agile software methods, 12 concluded that the current program life cycle is not supportable (Walsh, 2021). The various elements related to executing new software development practices in the program's life cycle have both challenges and opportunities. Balancing legacy software with future software execution requires justifying all aspects of cost, schedule, and performance (Department of the Air Force, 2008).

Adequate cost estimators are missing for the software requirement's life cycle to move from a traditional to agile approach (Brady & Skertic, 2021). The cost-estimating tools that forecast the software's life cycle are not evolving (McQuade et al., 2019). DOD programs, such as MDAPs, are required by law to report all expenditures and obligations, including budget performance, constraints, resources, funding colors of money, quality, and stages of the operational software (Department of the Air Force, 2008). The cost drivers associated with software innovation are often inexplicable. For example, some programs might experience the opposite effect of savings but produce inefficiencies and have to rework costs. The cost limitations of some programs may defer the availability of innovative DOD software based on the return on investment to the government.

Software is a critical path that constantly evolves to solve complex glitches, communicate, and interface with problems at all phases of the MDAP life cycle (Department of the Air Force, 2008). The entire process requires decision support to align with the program's life cycle software needs. At each stage of the program's life cycle,

various innovative software options might be justified to support the changes in operational scope (Chaillan, 2023).

During the development and production stages of the program, there are cost, schedule, and performance thresholds. During the sustainment phase of the program's life cycle, these same thresholds are missing. The key is to adequately capture the progress reported to the milestones correlated with the performance, cost, and schedule (McQuade et al., 2019). This account will confirm whether the software is the driving factor in weapons readiness if after production and during the sustainment phase there is a missed opportunity to incorporate new software methods (McQuade et al., 2019).

The program's performance over its life cycle is inadequate. The inability to forecast the system's "cradle to grave" requirements creates unknown gaps in the program. These risks may emerge for numerous unforeseen reasons not captured in the development or production phases. Software should be iterative, yet performance updates from the development and production stage to the sustainment phase miss crucial opportunities to refine the software. Once the weapon is released to the field, performance is monitored by a multitude of organizations. Each team provides various product support elements, and a lack of coordination means failing to analyze software readiness of the weapon in the field (McQuade et al., 2019).

6. Hierarchy Levels

Government agencies are vastly different from commercial businesses. The levels of hierarchy report to different individuals. Each of these individual stakeholders has different needs. The levels of hierarchy stifle the speed and integration of changes but ensure public accountability. The Pentagon has unusual demands. The programs it oversees must comply with thousands of guidelines (Carter, 2019). The DOD's stifling hierarchy limits some programs' innovation efforts but supplies transparency to the taxpayer and Congress (Carter, 2019).

As revealed in a 2022 report published by the RAND Corporation, Voss and Ryseff (2022) map a "series of hierarchical reviews of the Military Service for each of the systems . . . initiated by inputs at the working level" (p. 12). According to McQuade et al. (2019),

the "senior enterprise-level decision-making body for requirements" is involved in the acquisition decision-making process (p. 25). The structure includes a multitude of parameters of oversight and compliance reports to justify actions. The changes in administration and favored policies differ from a commercial corporation's hierarchy (Carter, 2019). The government's hierarchy makes for a dysfunctional system response. In addition to Congress, each individual may have one's own agenda (Carter, 2019). These individual projects change the rules of the federal government, leading to overly burdensome levels of hierarchy.

7. Stovepipes

McQuade et al. (2019) argue that "the current DOD organizational structure includes many separate stovepipes—each with its bureaucracy and staff—that are empowered to say no, rather than work toward solutions to warfighter problems" (p. 3). A stovepipe is "an isolated and narrow channel of communication" (Merriam-Webster, n.d.). The DOD works like a program-centric model with stovepipe-driven requirements. This operational structure feeds into the missing budget and acquisition processes. Stovepiped methods yield isolated working levels. As revealed in the SWAP Study, "The budget and acquisition officials exert control when they stay within their stovepipes" (McQuade et al., 2019, p. S130). The SWAP Study illustrates that innovative software "efforts to coordinate the processes have been less successful than hoped, and decision-making has remained largely an ongoing process" (McQuade et al., 2019, p. S58). As described by McQuade et al. (2019), "The result is a system in which senior decision-makers and their supporting staffs devote too much attention to process, procedure, and paperwork, rather than focusing on the major strategy and risk decisions that should be made at the working level" (p. S27). They continue: "Too often, innovative solutions are bogged down by a micromanaged process in which to support a decision" (McQuade et al., 2019, p. 125).

Changes to the DOD's software pathways require, according to McQuade et al. (2019), a "rapid, iterative approach to capability development" with swiftly evolving software platforms (p. 83). Instead, the status quo comprises several programs using "static configurations that [have lasted] more than a decade" (p. S79). Weapons system agencies

may have valid reasons for not moving away from the stovepipe decision mindset. The report suggests some of these reasons include "reduc [ing] costs, technological obsolescence, and acquisition risk" (McQuade et al., 2019, p. 1). However, this siloed decision construct deters not only emerging "small, innovative programs, but also large programs designed to meet future threats" (McQuade et al., 2019, S134). The authors conclude that "the ability to respond to new threats and new technology developments" is a limiting factor when software decisions do not include all stakeholders (p. S134). Notably, product support strategies are different for each system and at each stage based on the program's organizational structure (Lee, 2020). Thus, the decision-making systems embedded into military and intelligence communities are often disjointed and "stovepiped" (Doubleday, 2023).

D. STATE OF THE F-22 AND F-35 SOFTWARE EFFORTS

Two major DOD aviation programs, the F-35 and F-22, illustrate the inadequate alignment of decision-making with software methods within the entire acquisition development cycle. The software acquisition programs for the F-22 and F-35 differ primarily in the development and production phases.

1. F-22

The F-22 Raptor program was one of the first Air Force programs to identify software gaps in program modernization. The F-22 program had to overcome cultural changes before implementing new software pathways to bridge these gaps (Miller et al., 2022). Among the SWAP Study's suite of documentation is a vignette detailing viable solutions through collaboration to achieve the desired results. In it, Ulsh and McCarty (2019) identify an opportunity to incorporate modernization techniques that culminate in "greater speed and agility while addressing new threats faster by delivering user-requested software" (p. 1). The F-22 vignette also explores solutions not customarily considered for the program but essential for rapidly mitigating current and future risks.

Addressing the program's shortcomings has not been easy because it means the program administrators' admitting to them, including the fact that vendors have not been meeting performance conditions. Symptoms of the F-22's software underperformance have

included static, rigidly defined requirements, monolithic capability delivery, burdensome documentation, contracting inefficiencies and duplications of efforts, and automation and incremental testing challenges (Ulsh & McCarty, 2019). Harnessing the words of Mike Tyson, researchers for the SWAP Study explain the pain of these admissions: "Everyone has a plan until they get punched in the face" (Ulsh & McCarty, 2019, p. 54). The F-22 program's ability to take the punch would mean incorporating changes while reducing software releases and categorizing the current shortfalls of the business plan.

2. F-35

The list of challenges for the F-35's software development is lengthy. Current C2D2 and Block 3 upgrades are far behind the curve, and the subcontractors who support most of the upgrades to the Electro-Optical Distributed Aperture System and radar are failing to meet the agreed performance specifications. The current Block 4 upgrade involves full combat capabilities, including a weapons suite with nuclear abilities. According to Tirpak (2019), writing for *Air & Space Forces Magazine*, "Block 4 comprises 53 improvements to counter both air and ground-based threats emerging from China" (para. 3). As Tirpak (2019) explains, "They are primarily new or enhanced features executed in software; the original rollout plan was in stages, with updates every April and October starting in 2019 and continuing through at least 2024" (para. 17). However, the plan to deliver these upgrades in 2023 is behind schedule and at risk of not meeting the milestones. These delays are the result of other numerous delays in subcontractor-reliant software changes before final integration into the F35 software architecture and infrastructure. General Bogdan, former F-35 PEO, has noted that the complexity of Block 4 software is a risk but that sharing similar capabilities across the battlefield for coordinated attacks is difficult (Hoehn et al., 2022).

Hence, one risk is now a major problem. The plan was to create software drops in four increments of code, thus allowing the contractor to develop, test, and fix the software, and test it again, before moving to the jets in the field. The current software is behind schedule, and the contractor faces difficulties achieving even simple efforts in real time. If

the current requirements are not achievable, the complex software development required for Block 4 cannot realistically meet the contract.

E. CONCLUSION

The evolution of software pathways has shown both their benefits and limitations. The benefits include a more rapid release of new software and upgrades to weapons performance standards. The limitations mean that the systems may not have every software change in the software release. Rapid acquisition offers significant benefits; however, the software certification process is not constructed congruently, as demonstrated in timelines that extend beyond one year. Therefore, development under DevSecOps could be quite challenging and limited. The F-22 program took a proactive role in helping transform software processes and procedures through the introduction of cloud-based solutions through modern software metrics (B. Burton, personal communication, September 15, 2021). According to Chaillan, an agile path improves software acquisition and ensures the development teams can "groom their backlog and move at the pace of relevance" (Chaillan, 2021a). He maintains that "only Platform One and teams like Kessel Run are truly end-to-end agile" (Chaillan, 2021a, para. 2).

IV. CONCLUSIONS AND RECOMMENDATIONS

The role that software plays in MDAPs today is critical. The DOD has identified software concerns as a major threat to the execution of the mission. The DOD has made developing, acquiring, certifying, testing, and deploying software the priority for future missions (McQuade et al., 2019). Major DOD agencies require new ways of conducting business to find and fix software-related capabilities (Eckstein, 2021). For a myriad reasons, the department has adopted some new software development approaches but not at the speed required to address emerging threats. All the findings and recommendations in this chapter stem from the conclusion that software acquisition and development in MDAPs need to be revamped across the DOD. There is no one reason, but rather a combination of reasons for the current disjointed actions in MDAPs that creates inertia and resistance to the alignment of effective software activities.

A. FINDINGS

The research has found several areas within the DOD's MDAPs that should incorporate changes to current practices. The changes should result in program improvements across cost, schedule, and performance and disciplines, including acquisition development, deployment, agility, and integration. Areas of concern include resistance to change, inertia, inadequate program knowledge of software pathways, acquisition structure, misaligned software strategies, civilian frustration, and software/hardware misalignment of program requirements. Government reports dating from the early 2000s have documented ongoing software issues that have plagued most MDAPs, including the F-22 and F-35 programs, with little progress toward resolving software deficiencies. The software-centric nature of the aircraft and their sustainment means that no one remedy is an option.

1. Resistance to Agile Software Pathways

The research shows that in the DOD's culture, there is resistance and barriers to entry for technological changes in its programs. One example is the Air Force's F-22 Raptor program, which had to overcome resistance to change the software acquisition and

development processes (Miller et al., 2022). According to a 2019 GAO report, of the 22 military programs reviewed that claimed to incorporate agile software methods, only six confirmed the software pathways adequately met the intent of agile (Oakley, 2019).

The resistance to risk paralyzes program leadership, so it does not explore agile software pathways. This paralysis slows the program's momentum toward innovation, especially when it concerns software challenges. GAO and 2019 SWAP Study reports challenge the blending in of the software technology and upgrades in the F-22 and F-35 programs. These challenges also affect the program's momentum when the opportunities to improve are wasted, thus creating disconnects in the program. Other factors contributing to this resistance point include leadership beliefs and government hierarchy. Leaders with software-centric programs fail to understand the critical importance of software in the wars they are fighting today. They need to grasp that no one software pathway can adequately solve all complexities of a software system, but understanding smaller sections that can benefit from changes is not commonplace in the DOD.

One example of a program's failing to explore agile software methods is the F-35 next-generation aircraft, which requires over 24 million lines of code to operate. The F-35 aircraft operates with various other logistics and communication systems. The reliance on continual software updates is inadequate for the program since it interferes with other maintenance actions. This inferior software performance has been reported by the GAO year after year; meanwhile, the software pathway has not changed when leaders have had the opportunity.

However, for the F-35 logistics system, an Air Force software factory was used to improve its software development. The program office has made minor adaptations to P1's capabilities in the program, such as working with MDAP software transitions. Its plan was to transition from ALIS, a software platform based on 1990s architecture, to the newer ODIN technology (Hoehn & Gertler, 2022). The proposed updates to the ALIS system would allow tracking and control of aircraft components and life cycle upgrades supporting sustainment functions. The F-35 program office contracted the software updates with Air Force software developers at Kessel Run (R. S. Mamroth, personal communication, February 25, 2022). The requested work for the contract was completed, yet a follow-on

contract was not issued. The follow-on work went back to the contractor that had originally developed ALIS, and the name changed to ODIN (R. S. Mamroth, personal communication, February 25, 2022). The assumption is that MDAPs such as the F-35 program are transitioning to new software acquisition. However, these options are often not supported because the software development and release schedule are chronically late.

2. Software Challenges—Development and Acquisition Comprehension

Within the DOD's MDAPs, there are disconnects in understanding software between the acquisition community and the development team. This capstone research, supported by interviews, GAO reports, and SWAP Study reports, has offered examples of comprehending and using software pathways successfully. There is no one-size-fits-all development process, yet best practices may provide the opportunity to improve software timelines from development to release—if the program's leadership is aware of these alternatives. Results will vary based on the program's age, complexity, and amount of required software. Nevertheless, the available options for securing software do not align with the DOD's contract strategies. These misaligned paths create challenges with software development and acquisition practices, which ought to adequately address the software methods that best fit the MDAP.

A major issue lies in the lack of ownership of IP, which extends the control of updating software code. This lack of adequate IP management and ownership limits the DOD's ability to incorporate its software innovation directives. These directives intend to drive truly agile DevSecOps aspects into legacy programs that were not designed for the new operating environments they encounter today. Throughout this research, statements by Ashton Carter, former secretary of defense, and Nicolas Chaillan, former CIO of the Air Force, highlight the DOD's need to update software across the programs (Carter, 2019, 2021a; Hitchens, 2020). These reactions show the DOD is falling behind its adversaries and the commercial sector's ability to respond.

The DOD has led the way for agile software development pathways within some organizations, but the results are not what was expected or desired. The F-22 and F-35 are combat-ready aircraft, but they cannot respond quickly to emerging threats due to their

software inadequacies. GAO reports indicate that as of 2022, the F-35's software and technology refresh has been delayed several times (Ludwigson, 2022). It is unacceptable to take years when the opportunity exists to develop the same software solutions in a shorter time. It is now possible for the DOD to adopt new platforms that yield results in weeks or months. MDAPs must be willing to change the way they are developing software and to produce better results.

The research found a lack of collaboration in individually managed programs. Though overseen by the same military service, there was a failure to disclose lessons learned across programs, resulting in missed opportunities. The advantages of sharing knowledge-based and best practices concerning agile software development are overlooked at the top levels of leadership. The F-22 has had software development success using P1, as confirmed by the SWAP Study's vignettes and Lieutenant Colonel Burton's written responses to the interview questions for this study (B. Burton, personal communication, September 15, 2021). Benefits of the F-22 software were clear once the cultural hurdles were solved (Miller et al., 2022). This resulted in the program's promoting agile development and acquisition methods.

The F-35 program, however, is failing to see the same results. New threats to security are constantly fluctuating and, in many cases, are unknown. DOD leaders at each level must embrace the capacity to adapt to emerging threats, which software can alleviate at a much faster rate. The research has revealed that while many individuals in the DOD understand the new software requirements, others do not embrace these new directions. This uneven understanding creates gaps in re-forecasting software performance for the entire program's life cycle.

3. Misaligned Software Acquisition Pathways

The research has shown that the DOD acquisition structure and paths to procure software are misaligned. This misalignment skews the goals of program leaders. In some cases, the Pentagon's end goals should be included in the contract awards when new software acquisition pathways are absent. Otherwise, software integration into their programs will be a lengthy, onerous process (National Research Council, 2010). Constant

delays result in the degradation of schedule, cost, and performance metrics (Ludwigson, 2022). The consequences are further delays in meeting users' demands, which are based on rapidly developing and critical conditions.

The important role that software plays in programs, especially the F-35 program, has expanded exponentially over the past decades. According to a 2019 GAO report, the F-35's software acquisition has not explored all agile software (Oakley, 2019). The software underperforms and cannot address new threats in the required timeframe. The funding structure for authorization, appropriations, and obligations in the directions allowed by the colors of money hinders fund transfers for software when they are needed.

While DOD instructions and other transaction agreements apply to new endeavors, they do not apply to legacy DOD programs whose software needs to be refreshed, according to the SWAP Study (McQuade et al., 2019). This can become a problem when potential updates to the software are needed. When the decision to update the software pathways is justifiable—for instance, a move toward innovative solutions such as software factories—the plan must identify the ramifications. Moving away from the current sole-source software structure can improve the government's agility. One benefit of this requirement is that the contractor and the government can interact and share best practices and knowledge-based processes. Furthermore, the F-22 and F-35 programs incorporate additional agile DevSecOps software practices in a few elements when opportunities arise.

The research has revealed examples of MDAPs whose administrators freely admit they do not understand the role of software nor how best to develop and acquire it, while other programs refuse to change paths (Hall, 2018). This mindset in program structure enforces the existing inertia and gathers little momentum to change, which is part of a larger cultural issue that plays a part in the research context (Hall, 2018).

4. Effects of Software Incentives

Another finding from the research is that the structure of contracts does not always incentivize the desired results in software development. According to the GAO, despite clear goals set by DOD leadership, limited buy-in from the decision-makers at the program level has effectively shifted today's current delays in software deployment to the end-user

(Sager, 2021). Over the past decade, the GAO has repeatedly reported on the software challenges of the F-35 program. Since the software schedule is not a priority or aligned with the current F-35 incentive structure, it is overlooked.

If the program office cannot fill vacant software positions, it is likely missing software elements as part of the incentive requirement. Without personnel with software subject-matter expertise in place, experience cannot play a role in shaping contract incentive measures in the contract type (McQuade et al., 2019). Furthermore, a lack of software development oversight results not only in missing key opportunities to improve the contractor's performance but also in neglecting to monitor the software performance risks before testing, when it is too late to change. This monitoring practice should guide each program's understanding of what the software development options should be in the award and how they flow down in the contract with the suppliers. Nevertheless, incentives included in the contract file to produce agility in the software might end up incentivizing the opposite behavior in development. This problem results in costs continuing to increase as the quality diminishes and the schedule is chronically postponed.

The research has examined a different aspect of the F-35's proposed ALIS system that allows tracking and control of aircraft components throughout the program's life cycle. The upgrades support sustainment functions. The ALIS system is a logistics tool intended to assist ground crews with tracking maintenance schedules and spare parts and producing aircraft readiness reports (Hitchens, 2020). The problems with ALIS required a complete system overhaul (Hitchens, 2020). The contractor failed to adequately develop, deliver, and deploy to the government what it promised. The F35 program contracted software development and improvement for the ALIS-to-ODIN transition to the team at Kessel Run, an Air Force software factory (Hitchens, 2020).

The F-35 program pivoted to develop an improved logistics system. The ODIN software development portion using P1 innovation was not funded in the following budget. The work returned to Lockheed Martin in the next year's contract award. The concerns missed by the program leaders when incentives are included in the contract may result in unintended consequences. These incentives drive bad behavior in some cases if not clearly defined in the contract.

5. Human Resources—Government and Civilian Sectors

The research found that the DOD struggles to retain personnel skilled in software because opportunities in the private sector are often more attractive. The former first CIO of the Air Force criticized the DOD's inability to create, retain, and adopt rapid software pathways. Talented individuals leave government service because they become frustrated with being incapable of facilitating change and bringing innovation to the DOD as they had envisioned (Chaillan, 2021a). They are not accustomed to the rigidity of the government's hierarchy, financing, and oversight. The decision-making process in the government is slow. Congress is the watchdog over the taxpayer's dollars. The structure of the government is designed not to act rapidly, unlike the responsiveness of the commercial sector. The challenge of retaining expert personnel creates flux in the department and vacancies in a highly competitive software market.

The DOD's limited ability to retain and attract these highly sought-after human resources is a huge concern, as discovered during the research. The resignation of former CIO Nicholas Chaillan was a result of the DOD's challenges in retaining software experts. In his resignation letter, Chaillan (2021a) said, "One of the main reasons for my decision was the failure of OSD [Office of the Secretary of Defense] and the Joint Staff to deliver on their own alleged top 'priority'" (para. 36). According to Chaillan (2021a), "Saying what was wanted was not backed up with the appropriate actions. They could not 'walk the walk.' I put my reputation on the line" (para. 36).

The DOD's failure to facilitate the desired changes in software to Agile and DevSecOps was the result of being largely unempowered to fix basic issues (Chaillan, 2021a). The desired changes in the DOD have been interpreted differently by various individuals. The DOD hires technologically talented individuals for their skill sets. According to Chaillan (2021a), "The DOD, overall, needs to stop staffing Enterprise IT teams as if IT is not a highly technical skill and expertise" (para. 36). The goal is to employ technological innovations and software pathways that will bridge the current gaps in MDAPs, but the rigidity of the government's bureaucratic processes limits the achievement of this goal.

6. MDAP Lessons Learned—Too Big to Fail

MDAPs, such as the F-35 program, which has reported multiple underperforming technological elements throughout its life cycle, are not adequately scrutinized by leadership. The research findings show a need to adopt a knowledge-based approach to weapons systems. This movement allows a clearer measure of the performance of the system. The government's ability to justify whether a program should continue to receive funding, regardless of political pushback, has improved. Resistance to counseling programs that fail to perform needs to be met with result-based decision-making without pressure from policymakers. Findings from testimony before the Senate Subcommittee on Readiness and Management Support, Committee on Armed Services, on DOD acquisition reform do not support MDAPs' proceeding without confirmation of the resources (Oakley, 2021a). The requirements should match the funding, technology, and schedule, unless the three factors deviate.

F-22 and F-35 program challenges also diverge when accurately estimating and purchasing adequate software designs. Certain aspects of the directives are more easily and rapidly applied. Lockheed Martin, the primary manufacturer of both the F-22 and F-35, controls the software segregating capability in the Navy's system. When a major program tries to execute any agile software solutions, the factors to consider include an inadequate understanding of the software acquisition process, disjointed and overlapping software requirements, limited software incentives, issues related to ownership and managing of IP rights, and the lack of suitably trained human resources in software.

The Pentagon's testing office reports that the current contractual requirements of the F-35 continue to be delayed. Once the software is fully vetted, it is "immature, deficient, and insufficiently tested" (Capaccio, 2022, para. 1). At the same time, the program office continues to believe that product improvements are achievable without changing the software development pathway. As detailed in a 2022 GAO report, the DOD testing officer contends that the F-35 is unsuccessful in incorporating best practices and fails to deliver to schedule (Ludwigson, 2022). The current DOD software contracts fail to achieve these improvements. The methods are inefficient and fail to leverage the software resources developed by other agencies. DOD leadership should identify any political pressures that

prevent programs from applying realistic evaluations of the budget to performance criteria. The "zero defect mentality" is defined as trying something and scrubbing the program when it fails (Hall, 2018, p. 12). The overall lessons learned from failing should justify the decision to continue with less visible programs, such as the hypersonic missile. Being too big to fail, along with the added political pressure, should not set the special criteria for underperforming and cost-overrunning programs to continue (Grazier, 2020).

7. Differences between Hardware and Software in Programs

The research into the F-22 and F-35 programs suggests the difficulties of holistically managing software and hardware in required paths to fit the programmatic milestones. This research concludes that the Air Force's legacy programs no longer in production, such as those for the F-22, still need to develop software methodologies late in the programs' life cycle. Regarding software integration, at all stages of the F-22 and F-35 programs, it is important to develop and release software to meet the user's immediate needs in the field.

One crucial difference between hardware and software involves the effect of security requirements. For example, the F-22 program has revealed the ways that security requirements for software limit flexibility and communications but do not affect the hardware (Ulsh & McCarty, 2019). Security in software is an area of significant concern within the government as well as the commercial sector.

The estimation of costs also differs between hardware and software. It is extremely complicated to estimate program software development efforts, for one, because the type and amount of labor are unknown. Also, software lines of code are complicated and not accurately forecasted in DOD programs. In contrast, hardware has a finite number of materials and labor resources, so they are easier to estimate. The results show resistance from the DOD toward aligning the requirements of software with a different color of money, which would allow solid estimations and auditability to forecast and report performance.

Hardware needs software to function. The critical paths and metrics for identifying potential risks related to hardware are concrete since the hardware comprises tangible

assets. The critical paths for software are different and cannot be developed or tested against the performance certifications until the hardware is available. Moreover, the variables that affect hardware and software pathways and the funding of each in the programs are vastly different.

The research found divergence in the hardware and software available to in-house. Such differences result in additional risk to the programs in meeting cost, schedule, and performance. Resistance to changing this problem is based on the timelines and resources required for software development. These limitations are made manifest in the support provided to the government's digital workforce vis-à-vis the personnel of hardware manufacturing programs.

B. RECOMMENDATIONS

The assessments derived from this capstone research encourage any applicable DOD opportunity to meet the security requirements in software integration into military programs that are responsible for long-term weapons system management. The strategies recommended here can support a program's software integrators, who are currently struggling to facilitate change. MDAPs, such as the F-35 program, should not be exempt or protected from SWAP Study directives, as long they make sense. The DOD's program hierarchy needs to change and be brought down to a level playing field when software is concerned. The understanding that classified or sensitive data are exempt should not exclude portions of programs that will benefit from deviation from the current methods.

1. Government Lead

The government, not the contractor, needs to spearhead the program's directives. The contractor is often currently in the lead, but the program's leaders should take back the decision-making. The DOD should deviate from the traditional defense contractor sole-source mentality and assume the role of an integrator that responds to emerging weapons software needs. This shift in responsibilities could help develop and integrate software updates and field releases rapidly. This concept is essential to enticing the contractor to this shift from its current roles. The benefit for industry partners, government stakeholders, and the program office would be a new partnership to mitigate risks and reshape accountability.

Partnership with industry could provide faster software development and releases to the field. The PEO should have the insight to provide industry partners with what the program needs to achieve changes in the software. Existing limitations hinder access to adequate software pathways in both the acquisition and program life cycle. A shift to this new type of oversight responsibility, however, allows the programs to address some of the current software constraints. The government's taking this initiative should incentivize sole-source defense contractors to connect with non-traditional suppliers to provide competencies for the best value for the government as well as their bottom line. The government could also increase competition for the integration contract as it would not be limited by contractor-developed software IP rights.

Through these changes, MDAPs could integrate innovative software factories' knowledge-based and best practices into meeting software needs. Additionally, selecting the best contract type to protect all software development in the acquisition structure is needed. The structure should adopt the AAF to drive the industry to field new software developments and testing based on these adaptive milestones. The results should also pivot from the standard program management hierarchy to one of a portfolio mindset. It will take all stakeholders to come to a meeting of the minds, and the services will have to persuade the large defense contractors that it is in their best interest to invest in this path.

Another out-of-the-box recommendation is for the DOD to change its rigid culture by rewarding new software perspectives at the Pentagon. These new practices should allow employee movement from private industry into the government. The benefit to MDAPs is that it will provide the DOD with highly developed individuals who have industry experience and skill sets that the DOD is lacking. The DOD must shift to becoming a competitive option for tech-savvy resources whom it is currently missing among its personnel.

2. Software Prioritization

The software directives should be prioritized whenever feasible. Understanding the limitations of managing IP and code not developed by the overseeing office should drive the program development teams to seek evolving resources that can deliver rapid software

architecture. Legacy systems that surpass their predicted life cycle may benefit from software automation associated with software factors at some scale. Systems not adaptable to iterative development consist of platform controls that no longer require development.

MDAPs should apply favorable/unfavorable conditions in prioritizing software development methods. If the user's environment will not benefit from agile software, then the potential impact of use should serve as a checklist for whether to move toward iterative development. These methods cannot quickly adapt to realize the potential benefits due to their inability to change the overall Defense Acquisition System, thus leaving few other options.

3. Software Acquisition Methods

Acquisition reviews should always include agile software practices in some capacity within the programs. DOD Instruction 5000.85 deviates from the standard acquisition life cycle for acquiring software, yet the software instructions deviate from the current DOD processes concerning software options (DOD, 2021c). The AAF streamlines software acquisition. Contracting officers should be trained and empowered to incorporate aspects of this framework to be responsive to emerging software development requirements. The program terms are authorized to incorporate these pathways for different needs in the same program. This flexible approach should provide the contracting officers with the authority to act at the lowest level to incorporate new software pathways to satisfy software procurement.

The software acquisition directives explored in this research reveal that several program variables are often misunderstood. These aspects are not adequately considered in the entire life cycle of the program. Software decisions should reflect the system, operating structure, policies, stakeholders, acquisition options, new directives, agile pathways, and budget colors of money, and the overall effort across development, production, and sustainment needs. The research includes all of these factors, so it offers an inclusive source selection method that justifies the path to support the entire program.

The recommendation of this research is to allow acquisition professionals to execute this path when acquiring software. The complexity of software with the underlying

IP rights limits the integration. Program offices' contracting officers struggle to adequately capture the entire cost of the software. The ability to adequately estimate an unknown effort or requirement is impossible. Government regulations fail to align with emerging software pathways as the planning, programming, budgeting, and execution process misses the intent to adequately meet all aspects to protect software acquisition. The recommendation to justify purchasing program software under the AAF needs to be examined during the source selection review. Its inclusion in the yearly program objective memorandum created by each MDAP should be reviewed by software specialists to align the program's cost, schedule, and performance parameters.

4. Congressional Mandate for Consulting Service CIOs

The DOD should establish a comprehensive software program for all leaders, officers, and the acquisition community to use for bidding and procurement strategies. All program managers should approach CIOs for software development and support options. The DOD's chief software officer must expect more robust engagement from program officers to fully vet the best available software ecosystem for the multiple innovation hubs developed for MDAPs.

Congressional leaders control the regulatory and statutory requirements under which the DOD conducts business. Congress also has the power to require the implementation of modern software practices by the government. The SWAP Study provided a list to the legislative branch outlining the "do's and don'ts" of software (DIB, 2018). The recommendations are difficult to accomplish but worth implementing. Their value lies in removing roadblocks and facilitating service collaboration. The establishment of the yearly budget process requires CIOs to justify the software pathways the programs are choosing.

5. Centralized Life Cycle

Programs should transition to a centralized life cycle management mindset. Introducing new software capabilities requires updating the overall software process from the cradle to the grave of the code. Flaws in software programs should be segregated at the baseline development process and treated differently from software in the field that needs

updating. Outlining the type of software architecture requirements while applying the correct software approach to legacy software, as opposed to new program development software, may create diverse situations for software operators. The programs should first propose what may be developed as another system that best meets the user's needs.

Software guides program operations at all life cycle stages. Software is an adaptive tool to supply rapid responses to emergent threats. Software code at each stage of the life cycle has risks. The DOD should identify the appropriate leadership guidance that incorporates Panel 809, the National Defense Acquisition Act, SWAP, DIB, DOD Instruction 5000.87, other transaction agreements, and whatever is justifiable in the decision-making process. The recommendations should be adopted holistically across the life of the program. The result is transparency in that the true costs of the software reside in the program. The alignment of each life cycle stage—development, production, and sustainment—can reveal potential requirement shifts for both tangible and intangible assets. The forecasting requirements change in production and then in the sustainment stage. The estimated resources needed to execute software updates and testing have different critical paths for software than for hardware. These paths should be constantly reviewed for potential misalignment risks that would affect the program's milestones.

6. DOD-Wide Software Office

The future command of a congressionally mandated DOD-wide software office would avoid wasted resources and duplicated effort. Tackling software problems with a team from each of the five services—the Air Force, Navy, Army, Marines, and Space Force—would help to integrate lessons learned. This shift from stovepiping to collaboration should incorporate knowledge-based practices that identify performance, schedule, and cost risks to the government. It is important to challenge the team to develop and release what once took months to years to accomplish. This adaptation ensures that once the software is thoroughly tested and loaded on any system to support the warfighter, it is not outdated and a waste of funds (McQuade et al., 2019). The DOD should migrate away from the usual way the government conducts software deployment to a better option that produces the desired results.

When the DOD encompasses service-wide software capabilities, it should be capable of rapid deployment, its optimal goal. Collaborative teams, including in the software community, need to gain complete insights into all available methods of making well-informed options and decisions for future program capabilities (Sherman, 2022). Moreover, a new creative software-centric office should connect all systems to communicate with one another. The goal should be one integral "force-wide" combat system (Eckstein, 2021, para. 19).

7. Inertia/Momentum

The government's failure to act on emerging software threats is not due to a lack of acknowledgment of the problem but rather a question of how to correct the software gaps. The different services all encounter a level of software concerns in their systems. Some of the services have developed software commands, such as the Army Futures Command, the Air Force's P1 and Kessel Run, and the Navy's Forge. MDAP leadership has not incorporated these resources for various reasons, some of which are by design or self-imposed, for self-preservation, or from government construct.

Legacy MDAPs contain different software architectures that may not be fully understood by the program offices. The contractor owns the IP development rights by design, so the decision to integrate new software changes is often not executed. This fact is also a condition of the program's self-imposed inertia. The control of IP is solely within the defense contractors' wheelhouse. Thus, the MDAP manages potential software delays reactively instead of proactively. This reactive approach limits the opportunities to realign software development and requirements. Furthermore, the competitive software professional career field leads to software vacancies in the programs. These gaps leave software progress and monitoring neglected in the program. The result is that many software risks are unknown until they become issues. The contractor currently has no monetary incentive to relinquish software development to the government. Still, the government does not possess the IP rights or employ adequate software professionals to manage the entirety of software management.

8. Government—Contractor Business Structure

Government contracting personnel within the acquisition community should be encouraged to think creatively about what is sensible in purchasing software. Software is iterative and fluctuates wildly in some applications—the expectation that the acquisition community exercises all available options for procurement should be a priority for the DOD. Executable actions that provide agile and compliant results require clear objectives to meet the program milestones while achieving the appropriate software strategy. It is vital to have stakeholder transparency and buy-in. This approval empowers the team to analyze software alternatives with senior leadership support. Future cultural shifts must identify what software pathways will best fit each stage of the entire software life cycle to mitigate new threats facing the DOD.

The model of the Navy's Forge team could inspire a shift of the contractor's role as an integrator within the services that develop and manage MDAPs. The contractor would be incentivized in a revised acquisition source selection to incorporate agreements with the team. This team would include newly created, unbiased government software communities. The benefit would be in utilizing best practices and knowledge-based approaches to meet the warfighter's requirements earlier. The MDAP, not the contractor, would decide the level of software oversight in defense projects and the point at which to pivot to another pathway.

Future software-intensive MDAP contracts could include a rapid bidding process that involves all stakeholders. The government's software commands, defense contractors, and industry could collaborate to develop the solution to the software problem. The source selection structure should incentivize this process as a requirement. The metrics would report and measure the cost, schedule, and performance of these software practices. This architecture would incentivize both the government and industry teams as a cost savings for the software. The process would require streamlining and encouraging teams with technology leaders in industry and academia.

9. Program Management Shift to Portfolio Management

The SWAP Study and other DOD experts support changing the MDAP management structure. Programs would transition to a centralized team responsible for similar classes of weapons systems to exercise decisions based on updates in guidance, regulations, statutes, and threats. According to Chaillan, "During the global COVID pandemic, we witnessed the benefits of DevSecOps thanks to the fantastic Platform One team and its incredible delivery of capabilities" (In the Nic of Time, 2021, para. 32). Chaillan continues: "There is no valid reason not to use and mandate DevSecOps now for custom software" (In the Nic of Time, 2021, para. 38). According to Chaillan (2021a), "It is effectively guaranteeing a tremendous waste of taxpayer money and creates massive cybersecurity threats as well as prevents us from delivering capabilities at the pace of relevance" (In the Nic of Time, 2021, para. 39). Furthermore, the imminent risk loss of life and potentially prevents capabilities from being made available when needed—often overnight whenever world events demand.

10. Government–Industry Software Pilot Program

The DOD continues to raise concern over the absence of software personnel and training. These concerns include the ability to compete with the private sector, career growth, compensation, creative ability, and rigidity. The solution may be for the government to create a pilot program that allows a cooperative agreement with industry that allows movement for experts to work for both stakeholders. This option might involve teaming with academia to have the flexibility to exercise actions that benefit the program but may be restricted by the government's stovepiped structure. Potential benefits might include knowledge-based sharing across industry and the DOD of lessons learned and cutting-edge technology. Congress could create this new service policy and issue actions that could be updated annually through the National Defense Acquisition Act to promote this type of innovative behavior.

11. Software Career Path Redesign

Congress controls decisions concerning DOD directives. The SWAP Study recommends generating new software career fields that could attract software experts.

These new fields might attract individuals who previously saw no promotional opportunities in the military. Potentially, ROTC programs at universities that specialize in software development would entice new potential candidates. Additionally, the MDAP's PEO could engage the services of CIOs to recruit students by sharing new opportunities available in the DOD. Examples of new software factories and their ability to solve emerging issues within this government discipline would be critical selling points for any potential candidates. This opportunity might incorporate the recommendation to work in tandem with any pilot program in industry to incorporate new commercial best practices.

The required skill set of software personnel will depend on the complexity of change, the age of the code, the processes used to develop it, the amount of risk introduced if pursuing new software processes, and the number of budgetary constraints to implement the software fully. Quick error identification, continual and immediate user feedback, and the mitigation of security risks and unknown daily threats are among the most crucial skill sets.

12. Summary

Can new and agile pathways decrease costs, integrate automation, and increase security checks for leaks and hacking? The options, authority, and capability, while available, require scrutiny, knowledge, and cultural change and, in many cases, need momentum to keep progressing to meet the user in the field of software struggles. This research substantiates the current software challenges of many MDAPs, and one of the goals in moving toward the right software pathway would be identifying new processes to lessen the burdens on the programs. The conclusions gathered here reinstate past recommendations and identify stagnant software processes in the DOD. These methods cannot quickly adapt to realize the potential benefits due to one's inability to change the overall Defense Acquisition System.

C. FUTURE RESEARCH

In order to most effectively achieve the recommendations from the previous section, more understanding is needed in the following areas.

1. Create Pilot Programs

How could a pilot program be designed to retain software experience in the DOD? The allegiance of personnel historically lies in which party funds their paychecks. The creation of a pilot program could enable personnel to move freely between private industry and the government when justified. When a problem arises that can be solved with software, the private sector can maneuver with less red tape and obstacles than the government can in similar problem-solving. This flexibility in funding, from both the government and contractor, could attract more personnel to the government service who have the technological expertise to be competent software developers. Commercial enterprise, such as SpaceEx software solutions and other Silicon Valley initiatives, might benefit the programs. This research could explore existing pilot programs and investigate how such programs could be applied industry-wide. The residual problem that would face this type of program involves overly cautious individuals with security concerns. The personnel who would participate in these programs have equal allegiance to the government and industry.

2. Update the Source Selection Process

The updates Congress should create in the entire source selection process will require software pathway requirements when justified. Updates in the source selection process would assign new opportunities whereby the MDAP's PEOs meet with the services' CIOs to explore the applicable software paths for the entire program's life cycle. The DOD would need to adapt new acquisition processes to achieve the purchasing goal at every stage in the program life cycle. The result could be optimal software performance and development throughout the program. The derived metrics will measure the software's performance and time to release and pivot when necessary. A future research question could explore how to streamline and identify the levels to achieve this new mindset. The process to answer these questions could be to build a business case that includes all proposed directives and instructions and outlines the benefits and limitations.

3. Redesign the Contractor's Role in MDAPs

How can the government incentivize the defense contractors' relinquishing control to the government in MDAPs? As discussed in the previous section, the defense contractor's role needs to shift from design authority to integrator. Understandably, such a proposition would be difficult to sell to Congress, whose lifeblood comes from lobbyists who represent special interests. Indeed, the largest defense contractors have offices in Washington, DC, to keep their fingers on the pulse of policymakers. The Navy is piloting a program that will reinstate its ship-building control. If the change in roles is attractive for the contractor, this role might gain momentum. Another option would be to create more competition among smaller parties.

A future researcher could ask the question of how the military might retain this new role of design authority. The unseen benefits of this role include less oversight when the contractor seeks the best practices created within the larger DOD software enterprise and fewer contractors to expend future funding exploring potential software solutions when a fix already exists. This type of collaboration in all disciplines could improve cost, performance, and delivery.

LIST OF REFERENCES

AcqNotes. (2021, August 26). *PPBE process: Color of money.* https://acqnotes.com/acqnote/acquisitions/color-of-money

Austin, L. J., III. (2022, March 28). *The Department of Defense releases the president's fiscal year 2023 defense budget*. Department of Defense. https://www.defense.gov/News/Releases/Release/Article/2980014/the-department-of-defense-releases-the-presidents-fiscal-year-2023-defense-budg/

Beachkofski, B. K., & Helfrich, T. M. (2021). *Platform One, Kessel Run, we "believe" software principles* [Memorandum]. Department of the Air Force. https://kesselrun.af.mil/resources/images/news/We-Believe-Memo-P1-and-KR.pdf

Bellairs, R. (2022, October 15). *SWAP report for defense software overview.* Perforce. https://www.perforce.com/blog/qac/what-is-swap-report.html

Brady, S., & Skertic, B. (2021). *DOD's software acquisition pathway: Digital delivery at the speed of relevance* [Presentation slides]. DAU South. https://www.dau.edu/Lists/Events/Attachments/305/D1%20S5%20-%20Software%20Pathway.pdf

Capaccio, A. (2022, January 26). *Lockheed's F-35s get a flawed $14 billion software upgrade.* Bloomberg. https://www.bloomberg.com/news/articles/2022-01-26/f-35-fighter-jet-s--14-billion-software-upgrade-is-deployed-despite-flaws#xj4y7vzkg

Carter, A. (2019). *Inside the five-sided box: Lessons from a lifetime of leadership in the Pentagon*. Dutton.

Chaillan, N. (2019). *DOD Enterprise DevSecOps Initiative (Software Factory)* [Presentation slides]. Office of the Chief Software Officer. https://software.af.mil/wp-content/uploads/2019/12/DOD-Enterprise-DevSecOps-Initiative-Keynote-v1.7.pdf

Chaillan, N. M. (2021a). *It is time to say goodbye!* LinkedIn. https://www.linkedin.com/pulse/time-say-goodbye-nicolas-m-chaillan/

Chaillan, N. M. (2021b). *Let's catch-up with China within 6 months*. LinkedIn. https://www.linkedin.com/pulse/lets-catch-up-china-within-6-months-nicolas-m-chaillan/

Chaillan, N. (2023, February 28). DevSecOps: Modern cyber posture. *In the Nic of Time*. https://www.inthenicoftime.us/

Coram, R. (2020). *Boyd: The fighter pilot who changed the art of war*. Back Bay Books.

cPrime. (n.d.). *What is Agile? What is Scrum?* https://www.cprime.com/resources/what-is-agile-what-is-scrum/

Deasy, D., & Lord, E. (2019). *Software development, security, and operations for software agility* [Memorandum]. Office of the Secretary of Defense. https://software.af.mil/wp-content/uploads/2020/05/DevSecOps-Memo_Final_20191024.pdf

Defense Acquisition University. (n.d.-a). *Acquisition life cycle*. Retrieved June 2, 2023, from https://www.dau.edu/acquipedia/pages/ArticleContent.aspx?itemid=516

Defense Acquisition University. (n.d.-b). *Adaptive acquisition framework*. Retrieved May 31, 2023, from https://aaf.dau.edu

Defense Acquisition University. (n.d.-c). *Major capability acquisition*. Retrieved June 2, 2023, from https://aaf.dau.edu/aaf/mca/

Defense Acquisition University. (2019). Acquisition of services. In *Defense acquisition guidebook*. Defense Acquisition University. https://www.dau.edu/pdfviewer/Source/Guidebooks/DAG/DAG-CH-10-Acquisition-of-Services.pdf

Defense Acquisition University. (2023, March 22). *2023 DAU acquisition update—2 day event*. https://www.dau.edu/event/2023-Acquisition-Update

Defense Innovation Board. (2018). *Do's and don'ts for software*. https://innovation.defense.gov/Portals/63/DIB_DOS_DONTS_SOFTWARE_2018_10_05.pdf

Defense Innovation Board. (2019). *How to justify your budget when doing DevSecOps*. https://media.defense.gov/2019/May/02/2002127287/-1/-1/0/HOWTOJUSTIFYYOURBUDGETWHENDOINGDEVSECOPS.PDF

Defense Science Board. (2013). *Resilient military systems and the advanced cyber threat*. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. https://apps.dtic.mil/sti/pdfs/ADA569975.pdf

Defense Science Board. (2018). *Design and acquisition of software for defense systems*. Office of the Under Secretary of Defense for Research and Engineering. https://dsb.cto.mil/reports/2010s/DSB_SWA_Report_FINALdelivered2-21-2018.pdf

Department of the Air Force. (2008). *Weapon systems software management guidebook*. https://www.acqnotes.com/Attachments/USAF%20Weapon%20System%20Software%20Management%20Guide.pdf

Department of Defense. (2017, April 28). *Contracts for April 28, 2017*. https://www.defense.gov/News/Contracts/Contract/Article/1167080/

Department of Defense. (2019a). *Contracting considerations for Agile solutions: Key Agile concepts and sample work statement language*. https://www.dau.edu/cop/it/DAU%20Sponsored%20Documents/Contracting%20Considerations%20for%20Agile%20Solutions%20v1.0.pdf

Department of Defense. (2019b). *Intellectual property (IP) acquisition and licensing* (DOD Instruction 5010.44). https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/501044p.pdf

Department of Defense. (2020, October 2). *Operation of the software acquisition pathway* (DOD Instruction 5000.87). https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF

Department of Defense. (2021a). *DOD enterprise DevSecOps reference design: CNCF Kubernetes*. https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsReferenceDesign.pdf

Department of Defense. (2021b). *DOD enterprise DevSecOps strategy guide*. https://dodcio.defense.gov/Portals/0/Documents/Library/DoDEnterpriseDevSecOpsStrategyGuide.pdf

Department of Defense. (2021c). *Major capability acquisition* (DOD Instruction 5000.85). https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500085p.pdf

Doubleday, J. (2023, January 18). *DIA CIO sees intel community moving beyond "stovepipe" IT model.* Federal News Network. https://federalnewsnetwork.com/inside-ic/2023/01/dia-cio-sees-intel-community-moving-beyond-stovepipe-it-model/

Eckstein, M. (2021, April 12). *Navy software factory, the Forge, wants to reshape how ships get upgraded.* USNI News. https://news.usni.org/2021/04/12/navy-software-factory-the-forge-wants-to-reshape-how-ships-get-upgraded

Francis, P., Lea, M., Sullivan, M., & Taylor, K. (2001). *Best practices: Better matching of needs and resources will lead to better weapon system outcomes* (GAO-01-288). Government Accountability Office. https://www.gao.gov/assets/gao-01-288.pdf

General Services Administration. (n.d.). *What is DevOps?* Tech at GSA. https://tech.gsa.gov/guides/what_is_devops/

Gleason, J. P. (2022, April 15). *Optimal ignorance: A filter for intent-based leadership above the tactical level.* From the Green Notebook. https://fromthegreennotebook.com/2022/04/15/optimal-ignorance-a-filter-for-intent-based-leadership-above-the-tactical-level/

Grazier, D. (2020). *Spare us the F-35 parts mismanagement*. Project on Government Oversight. https://www.pogo.org/investigation/2020/07/spare-us-the-f-35-parts-mismanagement

Guertin, N. H. (2022, January). *Director, operational test & evaluation, FY 2021 annual report*. Office of the Operational Test and Evaluation Director. https://www.dote.osd.mil/Portals/97/pub/reports/FY2021/2021DOTEAnnualReport-opt.pdf?ver=_3wA1DAoalkLSLJEmAQARg%3D%3D

Hall, J. S., & O'Connor, J. M. (2018, March*). Learning technology adoption: Navy barriers and resistance* [Master's thesis, Naval Postgraduate School]. Defense Technical Information Center. https://apps.dtic.mil/sti/citations/AD1052658

Hitchens, T. (2020). *From "Mad Hatter" to "Torque": Kessel Run makes software for F-22, CV-22.* Breaking Defense. https://breakingdefense.com/2020/07/from-mad-hatter-to-torque-kessel-run-makes-software-for-f-22-cv-22/

Hoehn, J. R., Campbell, C., & Bowen, A. S. (2022). *Defense primer: What is command and control?* (CRS Report No. IF11805). Congressional Research Service. https://crsreports.congress.gov.pdf

Hoehn, J. R., & Gertler, J. (2022). *F-35 Joint Strike Fighter (JSF) Program* (CRS Report No. RL30563). Congressional Research Service. https://crsreports.congress.gov

Insinna, V. (2021, March 23*). F-35 program moves too slowly in deploying software, says government watchdog*. Defense News. https://www.defensenews.com/air/2021/03/23/f-35-program-not-moving-quick-enough-to-get-software-out-on-time-congressional-watchdog-finds/

In the Nic of Time. (2021, September 2). *It is time to say goodbye!* https://www.inthenicoftime.us/it-is-time-to-say-goodbye/

Jeng, A. (2019, September 11). The good, the bad, and the ugly of hardware security. *PUF Security*. https://blog.pufsecurity.com/2019/09/11/the-good-the-bad-and-the-ugly-of-hardware-security/

John Adams IT. (2020, December 3). *Greenfield vs. brownfield: Understanding the software development differences*. https://www.johnadamsit.com/software-development-greenfield-vs-brownfield/

Kessel Run. (2021, October 20). *ACC & Kessel Run reach historic agreement*. https://kesselrun.af.mil/news/ACC-KR-agreement.html

KK. (2009, July 17). Was Moore's law inevitable? *The Technium*. https://kk.org/thetechnium/was-moores-law/

Lee, C. (2020, December 3). *I/ITSEC news: Defense Department wants to expand acquisition framework*. National Defense. https://www.nationaldefensemagazine.org/articles/2020/12/3/defense-department-wants-to-expand-acquisition-framework

Ludwigson, J. (2021). *F-35 Joint Strike Fighter: DOD needs to update modernization schedule and improve data on software development* (GAO-21-226). Government Accountability Office. https://www.gao.gov/assets/gao-21-226.pdf

Ludwigson, J. (2022). *F-35 Joint Strike Fighter—Cost growth and schedule delays continue* (GAO-22-105128). Government Accountability Office. https://www.gao.gov/assets/gao-22-105128.pdf

Madhurihammad. (2022 June 16). *Difference between hardware security and software security*. Geeks for Geeks. https://www.geeksforgeeks.org/difference-between-hardware-security-and-software-security/

Maurer, D. (2019). *F-35 aircraft sustainment: DOD needs to address substantial supply chain challenges* (GAO-19-321). Government Accountability Office. https://www.gao.gov/assets/gao-19-321.pdf

McCaney, K. (2020, June 17). *DOD software behind the times? DevSecOps to the rescue.* GovLoop. https://www.govloop.com/dod-software-behind-the-times-devsecops-to-the-rescue/

McLaughlin, M. (2019). *How DevSecOps and Kubernetes can help transform the Pentagon*. C4ISRNET. http://hub.c4isrnet.com/whitepapers/

McQuade, J. M., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software is never done: Refactoring the acquisition code for competitive advantage*. Defense Innovation Board. https://media.defense.gov/2019/May/01/2002126688/-1/-1/0/SWAP%20ABRIDGED%20REPORT.PDF

Merriam-Webster. (n.d.). Stovepipe. In *Merriam-Webster dictionary*. Retrieved June 2, 2023, from https://www.merriam-webster.com/dictionary/stovepipe

Miller, A. W., Giachetti, R. E., & Van Bossuyt, D. L. (2022). Challenges of adopting DevOps for the combat systems development environment. *Defense Acquisition Research Journal*, *29*(1), 22–48. https://doi.org/10.22594/dau.21-870.29.01

Mortlock, R., Jones, R. D., Stewart, C. W., Deitrich, A. T., & Reid, J. M. (2022). Program management versus portfolio management in defense acquisition. In *Proceedings of the Nineteenth Annual Acquisition Research Symposium* (pp. 163–187). Naval Postgraduate School. https://dair.nps.edu/bitstream/123456789/4553/1/SYM-AM-22-040.pdf

Naegele, T. (2021, September 2*).* What drove Air Force Software Chief Chaillan to quit*. Air & Space Forces Magazine*. https://www.airandspaceforces.com/what-drove-air-force-chief-software-officer-to-quit/

National Research Council. (2010). *Achieving effective acquisition of information technology in the department of defense*. National Academies Press. https://doi.org/10.17226/12823

Oakley, S. S. (2019). *Weapon systems annual assessment: Limited use of knowledge-based practices continues to undercut DOD's investments* (GAO-19-336SP). Government Accountability Office. https://www.gao.gov/assets/gao-19-336sp.pdf

Oakley, S. S. (2020). *Defense acquisitions annual assessment: Drive to deliver capabilities faster increases the importance of program knowledge and consistent data for oversight* (GAO-20-439). Government Accountability Office. https://www.gao.gov/assets/gao-20-439.pdf

Oakley, S. S. (2021a). *DOD acquisition reform: Increased focus on knowledge needed to achieve intended performance and innovation outcomes* (GAO-21-511T). Government Accountability Office. https://www.gao.gov/assets/gao-21-511t.pdf

Oakley, S. S. (2021b). *DOD software acquisition: Status of and challenges related to reform efforts* (GAO-21-105298). Government Accountability Office. https://www.gao.gov/assets/gao-21-105298.pdf

Oakley, S. S. (2022a). *Leading practices: Agency acquisition policies could better implement key product development principles* (GAO-22-104513). Government Accountability Office. https://www.gao.gov/assets/gao-22-104513.pdf

Oakley, S. S. (2022b). *Weapons systems annual assessment: Challenges to fielding capabilities faster persist* (GAO-22-105230). Government Accountability Office. https://www.gao.gov/assets/gao-22-105230.pdf

Oakley, S. S. (2023). *Software acquisition: Additional actions needed to help DOD implement future modernization efforts* (GAO-23-105611). Government Accountability Office. https://www.gao.gov/assets/gao-23-105611.pdf

Office of the Chief Software Officer. (n.d.). *Platform One products and services: Customer DevSecOps Platform*. Retrieved June 4, 2023, from https://software.af.mil/dsop/services/

Office of the Chief Software Officer. (2019, November 15). *Kessel Run*. https://software.af.mil/softwarefactory/kessel-run/

Office of the Under Secretary of Defense (Comptroller)/Chief Financial Officer. (2022, April). *Defense budget overview: United States Department of Defense fiscal year 2023 budget request*. Department of Defense. https://comptroller.defense.gov/Portals/45/Documents/defbudget/FY2023/FY2023_Budget_Request_Overview_Book.pdf

Pant, A. (2019, August 8). *C2D2: An Agile defense acquisition model.* Indian Defense Review. http://www.indiandefencereview.com/spotlights/c2d2-an- agile- defence-acquisition-model/

Platform One. (n.d.-a). *Home page*. Retrieved May 31, 2023, from https://p1.dso.mil/

Platform One. (n.d.-b). *Iron Bank*. Retrieved June 2, 2023, https://p1.dso.mil/services/iron-bank

Red Hat (2023, March 10). *What is DevSecOps?* https://www.redhat.com/en/topics/devops/what-is-devsecops

Rendon, R. G., & Snider, K. F. (2019). *Management of defense acquisition projects.* American Institute of Aeronautics and Astronautics.

Royce, W. (1987). Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, 328–338. https://doi.org/5555/41765.41801

Sager, M. (2021). *High-risk series: Dedicated leadership needed to address limited progress in most high-risk areas* (GAO-21-119SP). Government Accountability Office. https://www.gao.gov/assets/gao-21-119sp.pdf

Scrum.org. (2020). *Scrum framework*. https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf

Sherman, J. B. (2022, January 24). *Software development and open source software* [Memorandum]. Department of Defense. https://dodcio.defense.gov/portals/0/documents/library/softwaredev-opensource.pdf

Skertic, R. (2019, December 12). *DCAR registry* [Video]. Defense Acquisition University. https://media.dau.edu/playlist/dedicated/62956591/1_fpz21i6j/1_m7dxgldf

Stellman, A., &. Greene, J. (2017). *Head first Agile: A brain-friendly guide to Agile principles ideas, and real-world practices*. O'Reilly.

Sullivan, M. J. (2008). *Defense acquisitions: A knowledge-based funding approach could improve major weapon system program outcomes* (GAO-08-619). Government Accountability Office. https://www.gao.gov/assets/gao-08-619.pdf

Sullivan, M. J. (2018). *F-35 Joint Strike Fighter: Development is nearly complete, but deficiencies found in testing need to be resolved* (GAO-18-321). Government Accountability Office. https://www.gao.gov/assets/gao-18-321.pdf

Sutter, J. (2021, April 19). *Reserve airman makes history with innovative Project FoX/F-35 development*. U.S. Air Force. https://www.af.mil/News/Article-Display/Article/2577421/reserve-airman-makes-history-with-innovative-project-foxf-35-development.html

Tate, D., & Bailey, J. (2020). *Software speed limits: What controls how fast we can modernize?* [Master's thesis, Naval Postgraduate School]. Defense Acquisition Innovation Repository. https://dair.nps.edu/handle/123456789/4227

Tate, D., & Bailey, J. (2021). Factors limiting the speed of software acquisition. In *Proceedings of the Eighteenth Annual Acquisition Research Symposium* (pp. 152–161). Naval Postgraduate School. https://dair.nps.edu/bitstream/123456789/4383/1/SYM-AM-21-076.pdf

Thornberry, M. (2023, February 2). *The Pentagon must make a culture shift to embrace innovation.* Defense News https://www.defensenews.com/opinion/commentary/2023/02/02/the-pentagon-hasnt-made-the-culture-shift-key-to-embracing-innovation/

Tirpak, J. A. (2019, February 25). Keeping the F-35 ahead of the bad guys. *Air & Space Forces Magazine*. https://www.airandspaceforces.com/article/keeping-the-f-35-ahead-of-the-bad-guys/

Ulsh, C., & McCarty, Z. (2019). *Vignette 2—F22: DevOps on a hardware platform.* Defense Innovation Board. https://media.defense.gov/2019/May/01/2002126695/-1/-1/0/VIGNETTE%202%20-%20F22%20DEVOPS%20ON%20A%20HARDWARE%20PLATFORM.PDF

VMware. (n.d.). *Software factory: Modern software development*. Retrieved June 1, 2023, from https://tanzu.vmware.com/software-factory

Voss, N., & Ryseff, J. (2022). *Comparing the organizational cultures of the Department of Defense and Silicon Valley*. RAND Corporation. https://www.rand.org/pubs/research_reports/RRA1498-2.html

Wallace, M. (2017, July 5). How software is eating the military and what that means for the future of war. *Fast Company*. https://www.fastcompany.com/40436077/how-software-is-eating-the-military-and-what-that-means-for-the-future-of-war

Walsh, K. (2021). *Software development: DOD faces risks and challenges in implementing modern approaches and addressing cybersecurity practices* (GAO-21-351). Government Accountability Office. https://www.gao.gov/assets/gao-21-351.pdf

Wheeler, D. A. (2018, April 28). The waterfall model. *David A. Wheeler* [blog]. https://dwheeler.com/essays/waterfall.html

Williams, L. C. (2020, December 21). *Software factories are the new "crown jewels," Air Force official says.* FCW. https://fcw.com/security/2020/12/software-factories-are-new-crown-jewels-air-force-official-says/258550/

Williams, L. C. (2021a). *DIU director: Look beyond reform to keep the technological edge*. Washington Technology. https://washingtontechnology.com/2021/11/diu-director-look-beyond-reform-to-keep-the-technological-edge/355510/

Williams, L. C. (2021b). *Why DOD is so bad at buying software*. FCW. https://fcw.com/acquisition/2021/11/why-dod-is-so-bad-at-buying-software/259180/

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California