SYM-AM-23-087

Excerpt from the Proceedings of the Twentieth Annual Acquisition Research Symposium

Acquisition Research:
Creating Synergy for Informed Change

May 10–11, 2023

Published: April 30, 2023

Software Acquisition and the Color of Money

Jeff Dunlap, CAPT USN (Ret.)—is a Faculty Member at the Department of Defense Management at the Naval Postgraduate School (NPS). Dunlap has over 25 years of experience in the Department of Defense (DoD) as an acquisition professional and has led several software-intensive programs.

As part of his service to NPS, Dunlap provides mentorship and thesis advice to military and civilian students researching software changes needed within the DoD to increase timeliness and value to the warfighter.

Dunlap has a BS from Virginia Tech at Blacksburg. He received an MS in engineering from NPS and Defense Acquisition University ACAT I PM certifications. [jeffrey.dunlap@nps.edu]

Abstract

The current Department of Defense (DoD) acquisition budgeting process provides funding visibility to Congress for hardware-intensive systems from requirement generation to ultimate disposal. Unfortunately, a square peg in a round hole quandary has occurred with a funding mismatch as modern software-intensive systems are required to comply with traditional funding appropriation breakout categories (aka colors of money). The 2019 Defense Innovation Board (DIB) Software and Acquisition Practices (SWAP) report identified the funding challenges of continuous software development and stated, “Colors of money doom software projects.”

In the fiscal year (FY) 2020 National Defense Authorization Act, Congress created a pilot program with a new appropriation category for software-intensive DoD programs (BA-8). The challenge to the DoD is to prove via quantifiable metrics that a single appropriation of funds enables speed-to-capability deliveries in the software pilots. Other contributing factors made it difficult to discern the effects of BA-8, as revealed by the pilot program metrics, which highlighted potential future study areas. Regulations and policies regarding funding that do not consider the continuous delivery of software capability to the user after the fielding event milestone can lead to confusion about the appropriate appropriations to use and their timing.

Executive Summary

Software acquisition, development, and support practices within the Department of Defense (DoD) had not fundamentally changed since the implementation of the “waterfall” model in the 1975 DoD Directive 5000.29. In 1987, The Defense Science Board Task Force on Military Software recommended a shift away from waterfall software practices (more common in hardware- intensive programs) to an iterative prototype (agile) lifecycle model. The seminal report of 2019 from the Defense Innovation Board (Software and Acquisition and Practices [SWAP]) was fundamental in describing a tailored, software-specific pathway that guided acquisition change. The end goal of the software change recommendations from the SWAP report was to empower acquisition professionals to deliver relevant and secure capabilities at the “speed to need” using modern software practices found in the commercial sector.

The acquisition and sustainment process for fielding and supporting software-intensive systems changed with the software pathway of the 2020 DoD Instruction (DoDI) 5000.02 (Operation of the Adaptive Acquisition Framework). Although simplifying acquisition policy has eliminated numerous obstacles, the budgeting system (PPBES) categories do not effectively apply to software development that follows iterative and continuous approaches. In contrast, acquisition strategies for incremental waterfall software development programs aligned closely to the budget appropriations spending categories of development, production, and operations & sustainment phases.

Included in the 2019 SWAP report was the recommendation to create a new appropriation category that would allow software-intensive programs to be funded as a single budget appropriation item since “software is never done.” The model for a modern continuous software acquisition is that there is no separation between development, production, and operations & sustainment. In 2021, the “bleached” or “colorless” appropriation pilot (Budget Activity 8 [BA-8] for software and digital technology pilot programs) began with nine DoD software-intensive programs. The BA-8 pilot provided a single appropriation that could be utilized for any legitimate expenditure.

Intuitively, it makes sense that programs developing continuous capability in an agile fashion would need money that has no restrictions (such as development, testing, or maintenance). Metrics to validate and assess the effectiveness of these pilots would allow the DoD to understand the impact of BA-8 on delivering capabilities “at the speed of relevance.” Since BA-8 is not limited to programs using the Software Pathway, understanding the secondary and tertiary impacts on capability delivery became important in determining what effect PPBES had on software-intensive programs. Data collection and metrics for the BA-8 pilots were required quarterly by Congress. An early assessment by the Under Secretary of Defense for Acquisition and Sustainment (USD A&S) for the effectiveness of BA-8 occurred in the 18th month of implementation. The results indicated that the pilot programs could not demonstrate the singular value of “colorless money.” Other factors were discovered to have as much influence on software acquisition as BA-8. The absence of measurable criteria to exhibit to Congress the worth of a solitary appropriation could hinder the establishment of a sustainable budget classification for software-intensive programs.

Delivery Speed is not a Characteristic of the Deterministic Waterfall Approach

The waterfall development method’s primary utilization was in software engineering for decades. This approach follows a linear sequential path, where each phase of the software development process must be completed before moving to the next phase. The method begins with requirement gathering, followed by design, implementation, testing, deployment, and maintenance. Its rigid structure ensures that each phase must be finished before the next phase can begin, making it difficult to make changes later in the process. Despite its limitations, the waterfall method is still used in some projects where requirements are well-defined, and flexibility is unnecessary. Software acquisition, development, and support practices within the DoD have not fundamentally changed since implementing the “waterfall” model in the 1975 DoD Directive 5000.29. The waterfall process is not inherently good or bad, as its effectiveness depends on a clear understanding of the requirements in advance, which must be known and remain unchanged. Waterfall methodology is very deterministic, where all the software’s functions or features are understood in advance, and the entire software is either accepted or rejected at verification (Figure 1).

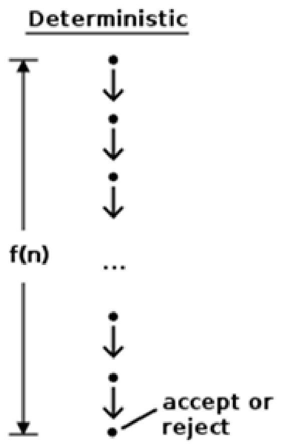

Figure 1. Deterministic Software Development

Deterministic programs often lack the flexibility and agility to deliver capabilities quickly. Software programs that follow the deterministic waterfall approach cannot deliver capabilities quickly for several reasons. Some of the most common causes include

1. Outdated technology: Software developers often build software using outdated technology. As a result, integrating new features and functionalities becomes difficult and time-consuming.
2. Complexity: Software programs themselves often exhibit a high level of complexity and are hard to modify. The reason for this is that numerous developers with their preferred programming languages, tools, and methodologies may have contributed to the development of the program over time.
3. Lack of documentation or code: DoD software programs may have little or no documentation or software code due to intellectual property or data rights missing in the contract deliverable. Even if the code is available, it is difficult for other developers to understand how the software works and how to modify it without the original developer's documentation.
4. Limited resources: software programs may not have the resources or budget to invest in modern software development practices. The delivery of new capabilities can be slowed down due to this.
5. Technical debt: Over time, software programs may accumulate technical debt, which refers to the cost of maintaining and updating software that was not properly designed or developed in the first place. Technical debt can slow development and make adding new functionality difficult without introducing new bugs or issues.

Before 2020, the DoD's acquisition framework did not encourage a modern software development methodology, whereby contracts with the Defense Industrial Base are awarded without complete visibility of the requirements. Currently, the DoD's budgeting process (PPBES) still mandates deterministic knowledge of the total acquisition requirements, as well as its timing and cost, regardless of its development approach.

Modern Software Development Enables Speed

In the early 2000s, the private sector shifted away from traditional heavyweight methods of developing software-intensive systems, such as the waterfall approach. The need to meet market demands for speed and capture customers by delivering working software drove this pivot. The rise of lightweight software development methods such as Agile and the automation software tools needed to build, integrate, test, and deploy continuously began the DevOps/software factory concept. The DoD began to question whether it could take a page from the private sector and refactor how software is developed and deployed to the customer (warfighter) to meet the "speed to capability" demand signals to maintain the warfighter advantage. The Defense Science Board has stated over the years that shifting to a modern software process is not a technology issue but a process and culture question.

A self-evaluation of the software acquisition process in 2019 by the Under Secretary of Defense for Acquisition and Sustainment (USD A&S) reinforced that the DoD is a performance-based bureaucracy that focuses on time, schedule, and budget to evaluate the performance of its programs. The DoD's acquisition strategy was guided by the capability-based assessment process, also known as JCIDS, to counter future threats to the national security mission. This requirements-based process provided justification inputs into the budgeting process (PPBES), which produces a current and future year budget forecast (5 years into the future). When comparing the commercial marketplace and decisions that the private sector often makes on software development capability investments to dominate competitors, it becomes evident that there is a great divide between the two processes.

DoD Acquisition Guidance underwent a significant process change in 2020: the Adaptive Acquisition Framework (AAF; DoDI 5000.02). The objective of this modification was to provide the end user with prompt and cost-effective solutions that are efficient, appropriate, durable, and environmentally sustainable. Following this release, the Software Acquisition Pathway (SWP; DoDI 5000.87) further defined the purpose to facilitate rapid and iterative delivery of software capability (e.g., software-intensive systems or software-intensive components or sub-systems) to the user (Figure 2). The SWP Characteristics were similar to the commercial marketplace where the user became the focus.

This pathway integrates modern software development practices such as Agile Software Development, Development, Security, Operations, and Lean Practices. Small cross-functional teams that include operational users, developmental and operational testers, software developers, and cybersecurity experts leverage enterprise services to deliver software rapidly and iteratively to meet the highest priority user needs. These mission-focused, government-industry teams leverage automated tools for iterative development, builds, integration, testing, production, certification, and deployment of capabilities to the operational environment. (Office of the Under Secretary of Defense for Acquisition and Sustainment [OUSD(A&S)], 2020a)

The DoD defined this shift in software acquisition procedures as a rapid, iterative approach to software development that reduces costs, technological obsolescence, and acquisition risk. However, because many software acquisition programs involve either applications or embedded software, there are differences in planning and execution timing. In addition, to expedite speed to capability execution and get to quick wins, several steps required by traditional capability acquisition programs were relaxed or eliminated. Unfortunately, the PPBE funding process was unaffected by this acquisition initiative and still follows the cold war era processes with little ability to flex based on emerging threats. The DoD's move towards a non-deterministic software architecture is rapidly growing in practice.

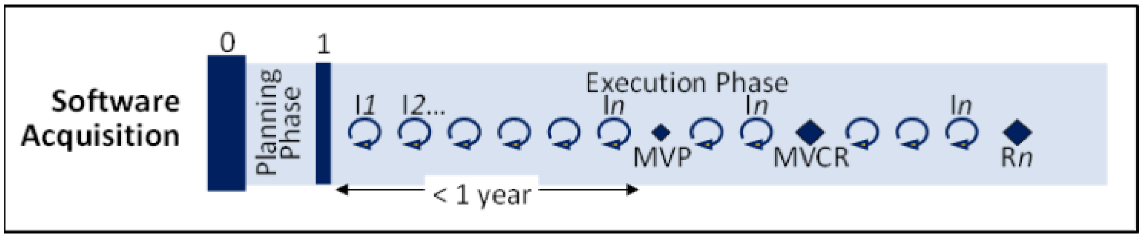

Figure 2. The Software Acquisition Pathway: Iterative Development of Application Software

Monolithic Architectures Have Inertia to Change

Most of our weapon systems' offensive and defensive capabilities are software-controlled, and the ability to respond to opportunities and threats requires software updates at the tactical edge. The DoD built most of its warfighting software using a Monolithic Architecture. Monolithic Architecture is a traditional software design approach that involves developing an application as a single, self-contained unit. This architecture is characterized by tightly coupled program components or functions, meaning they are highly dependent on each other and tightly integrated into the overall application. This design approach makes developing, deploying, and maintaining the software more manageable, as everything is contained within a single codebase. It's like building a large, complex building with all the rooms and floors interconnected and dependent on each other.

The downside of this approach is that it can limit scalability and flexibility, as changes to one component may affect the entire application, and it can be challenging to add new features or scale the system as it grows in complexity. Monolithic Architecture requires all associated components to be present for code execution or compilation and for the software to run. Moreover, modifying a single program component may necessitate modifying other software elements, leading to the entire application requiring recompilation and testing. Such a process can consume a significant amount of time and hinder the agility and swiftness of software development teams. An example of this problem is the delay in enhancements/updates to the shipboard combat systems software (AEGIS/SSDS), which typically exceeds 6 years to get to the warfighter and, by administrative procedure, never updated while deployed operationally (PEO IWS X).

Modern Software Architecture Brings Speed to Delivery

The DevSecOps process (software factory) is the big buzzword in DoD software acquisition. Looking at Figure 3, there is a continuous process of engagement with the development team (software coders), security experts (helping the coders learn and verify best practices), and the users (operators of the system). The development and operational environments are closely related and connected through telemetry, enabling health and status reporting with user feedback. Agile software coding principles and culture within the software factory remain fundamental to the team's success.

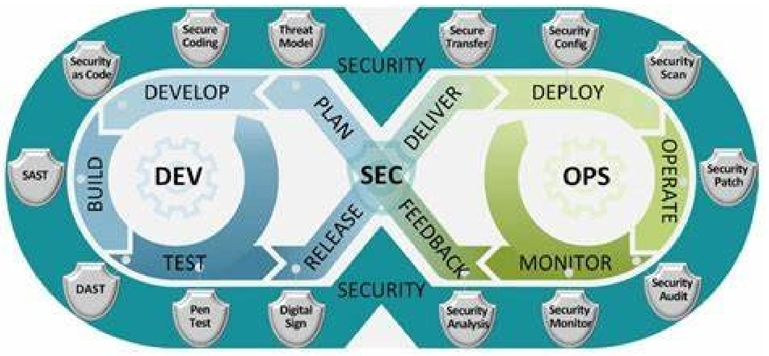

Figure 3. DevSecOps Distinct Lifecycle Phases and Philosophies

Most people can intuitively grasp the conceptual value of connecting software coders to the system's users, as this enables feedback and integrates security experts into the DevSecOps process. However, the concepts of modern software architecture go far beyond the figure of connecting rings. Understanding how a software factory can continuously deliver cyber-secure software to the tactical edge is also connected to the "colors of money" discussion. The DoD Chief Information Officer (CIO; 2021) has provided a DevSecOps Strategy Guide as a starting reference to what the advantages of a software factory may bring.

The software factory is the "Dev" component of DevSecOps (Figure 4). It pertains to the processes where software developers continuously integrate and test their code in a secure cloud environment. The "pipelines" produce software applications of self-contained functionality, also known as "containers." Container applications are lightweight (<10 megabytes) and bundled in a release package. Under the traditional acquisition approach, the software is compiled into machine language and provided as a single monolithic package containing multiple features, typically exceeding 10 gigabytes in size. Every time a change is made or added to the software, the entire monolithic package has to be recompiled and re-installed. The software factory differs substantially from the monolithic waterfall method because it does not compile functionality into a single software package. Instead, each container application executes specific functions upon receiving a request from an orchestrator and terminates afterward.

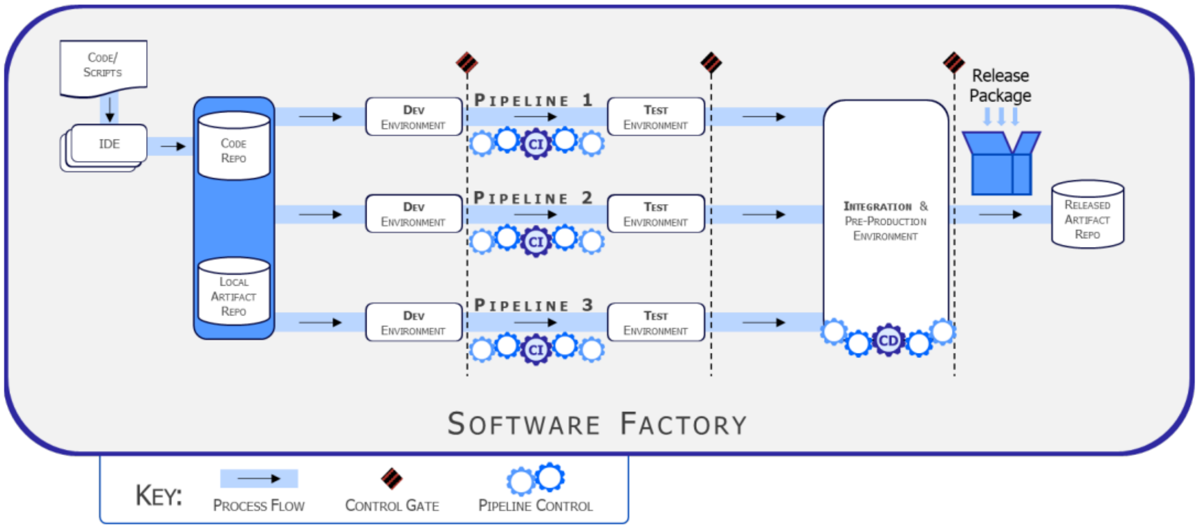

Figure 4. Software Factory Construct

The software factory achieves modern architecture by breaking the traditional monolithic into discrete domains and orchestrating containers to perform these domain services. Microservice architecture is an architectural style that structures an application as a collection of container services. In a rapidly changing environment where maintaining warfighting dominance is crucial, the microservice architecture enables organizations to deliver large, complex applications quickly, frequently, reliably, and sustainably. Figure 5 shows the conceptual difference between a monolithic and microservice architecture. The key advantage of the microservice architecture is the speed at which users can add/modify capability. Development teams can rapidly deploy individual software components without redeploying the entire application. The DevSecOps Software Factory follows a pipeline process to develop new containerized software, which involves testing, integration, and release into the repository. The size of the software matters, as bandwidth is often a limiting factor at the tactical edge or in a contested spectral environment.

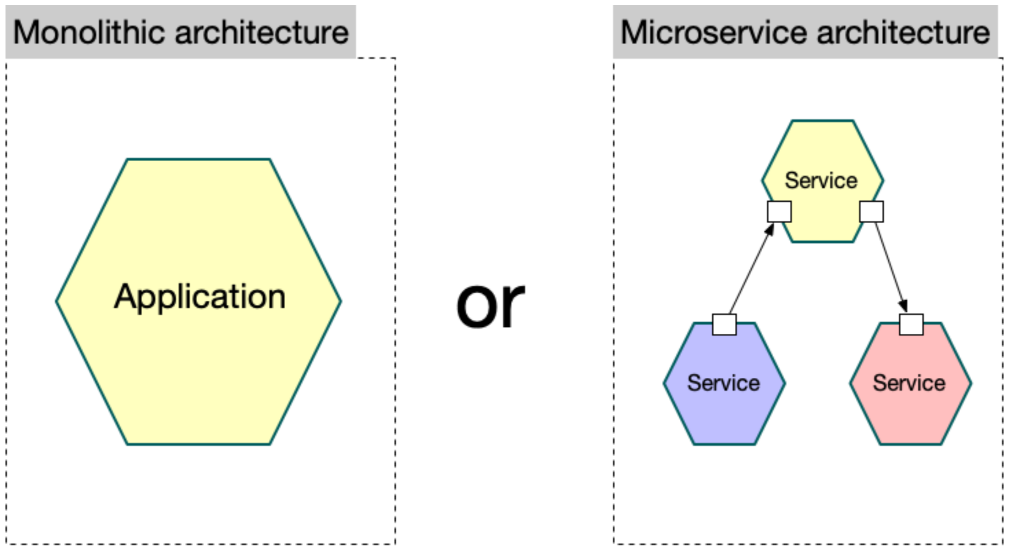

Figure 5. The Conceptual Difference Between Software Architectures

Acquisition Categories of Money Doom Software Factories?

Traditionally, acquisition programs have followed the same development pattern over the last 40 years. Although there has been encouragement to tailor the acquisition pattern to reduce wasted effort, pathways did not exhibit variances based on the product being developed until the implementation of DoDI 5000.02 in 2020. The Major Capability Acquisition pathway is typical of how appropriation categories of money are programmed into the budget. As the product progresses through the milestones (MS A/B/C), funding is primarily for Research, Development, Test, and Evaluation (RDT&E). Procurement funds are the dominating category spent after MS C, and once fielded, the category shifts to Operations and Sustainment (Figure 6). Unfortunately, the budgeting process (PPBES) sees all acquisition pathways progressing through these funding categories.

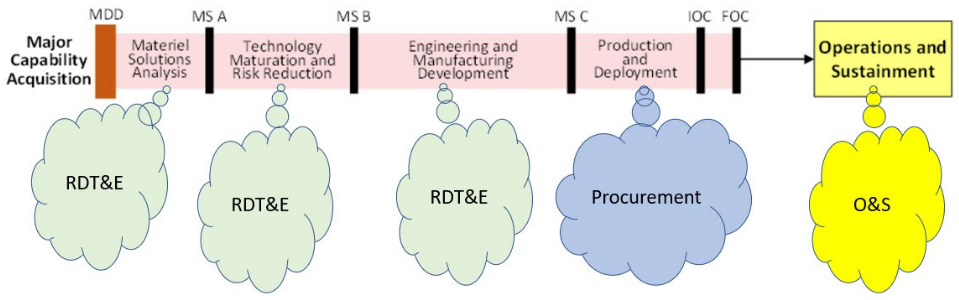
MDD
MS A
MS B
MS C
IOC
FOC
Major Capability Acquisition
Materiel Solutions Analysis
Technology Maturation and Risk Reduction
Engineering and Manufacturing Development
Production and Deployment
Operations and Sustainment
RDT&E
RDT&E
RDT&E
Procurement
O&S

Figure 6. Principle Acquisition Categories of Funding for MCA

The Software Acquisition Pathway is detailed in DoDI 5000.87 and illustrates what is occurring during the two phases: planning and execution (Figure 7).

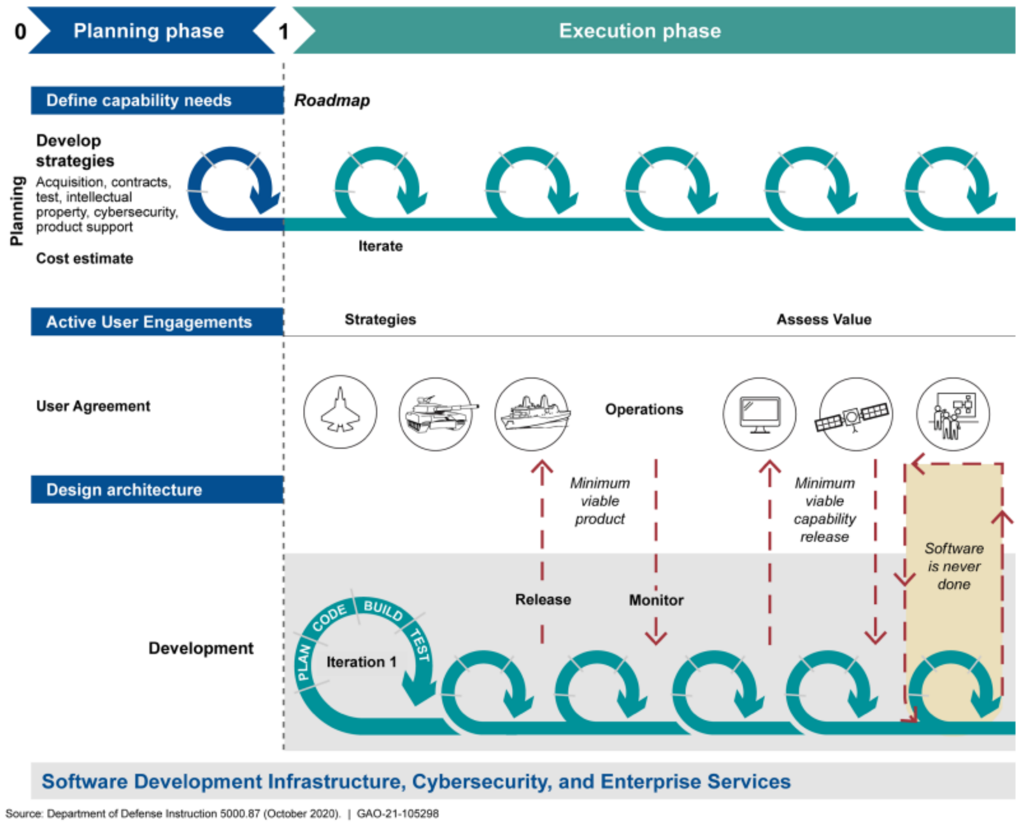

Figure 7. The Software Acquisition Pathway

Gone are the funding triggers for product procurement, operations and sustainment from the software acquisition pathway, and in their place, an iterative capability development that is never done. Executing a program following this pathway presents a PPBES dilemma on what category of money is needed and when. The operational funds in the MCA pathway are triggered upon capability deployment to users with specific task guidelines:

> Types of expenses funded by O&M appropriations generally include DoD civilian salaries, supplies and materials, maintenance of equipment, certain

equipment items, real property maintenance, rental of equipment and facilities, food, clothing, and fuel. (Defense Acquisition University, 2023)

Funding category and timing decisions are usually based on performance improvement or testing requirements, as illustrated in Figure 9. This flowchart is valuable for the MCA pathway but has little relevance when executing continuous delivery in the Software Pathway. If a new software container is developed that increases the fielded software system's performance, RDT&E is required using this flowchart. Procurement dollars are needed if a software correction is made to a container and the system is in production. The challenge with "colors of money" in software development lies in determining whether the release of the minimum viable product marks the start of production or the point at which the system is in service. Trying to fit the concept of "colors of money" into software projects is akin to fitting a square peg in a round hole, which can lead to project delays. The reasons for specific congressional guidance on how money is spent make sense only from an accountability perspective. But because software is in continuous development (it is never "done"), colors of money (besides RDT&E) tend to reduce the agility to obligate funds when reprogramming is required. We need to create pathways for "bleaching" funds to smooth this process for long-term programs (DIB, 2019).

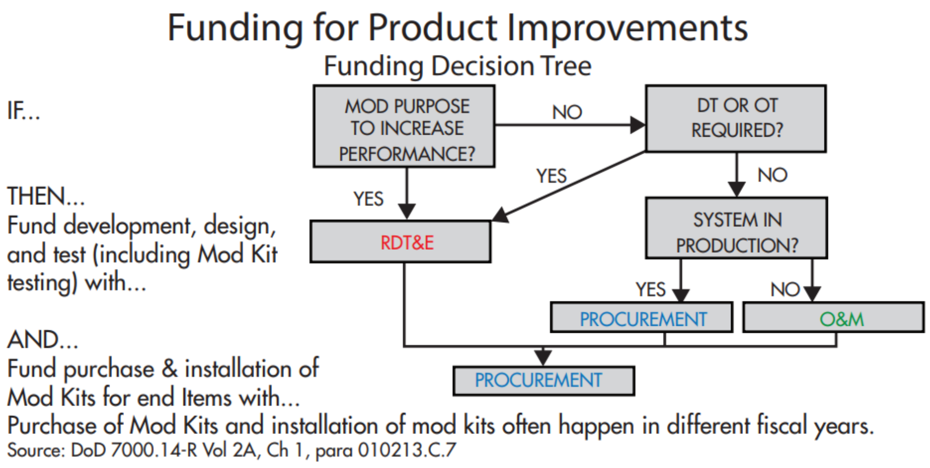
Funding for Product Improvements
Funding Decision Tree
IF...
MOD PURPOSE TO INCREASE PERFORMANCE?
NO
DT OR OT REQUIRED?
YES
NO
THEN...
Fund development, design, and test (including Mod Kit testing) with...
YES
RDT&E
SYSTEM IN PRODUCTION?
YES
NO
PROCUREMENT
O&M
AND...
Fund purchase & installation of Mod Kits for end Items with...
PROCUREMENT
Purchase of Mod Kits and installation of mod kits often happen in different fiscal years.
Source: DoD 7000.14-R Vol 2A, Ch 1, para 010213.C.7

Figure 9. Funds Management Platinum Card Decision Tree

Metrics on BA-8 Show Other Influencers

Congress established a pilot program in FY2021 to provide a single appropriation BA-8 (colorless money) to several software-intensive programs. This new appropriation category for software capability delivery has no separation between RDT&E, production, and sustainment. This initiative was a multi-year pilot to collect and analyze metrics to inform a final recommendation to make this an enduring appropriation.

Congress required quarterly BA-8 metrics from the USD A&S. Their FY2021 report to Congress stated that they did not consider BA-8 a silver bullet. Although BA-8 is expected to address some critical challenges faced by programs adopting commercial software development practices, it is not a comprehensive solution to all their problems. Metrics assist leadership in comprehending the effectiveness of pilot programs implementing BA-8. The metrics that OSD picked for the pilot programs in FY2021 are shown in Table 1. These metrics are adapted from the Google DevOps Research and Assessment (DORA) team and are used by DevOps teams to measure their performance and find out whether they are "low performers" to "elite performers."

Table 1. BA-8 Pilot Metrics Collected

Factor Influence on Measure	Product Delivery Lead Time	Release Frequency to Operational Environment	Deployment Frequency to Production	Mean Time to Restore	Change Fail Percentage
BA-8 Single Appropriation	High	Medium	Medium	Medium	Low

The USD A&S acknowledged in the fourth quarterly report to Congress for FY2022 that while there is compelling evidence of improvements provided by BA-8 for the pilot program, it is primarily qualitative. Quantitative measures were utilized to measure the influence of BA-8 based on traditional commercial software factory metrics.

Product Delivery Lead time in a software factory (DevSecOps) measures how much time has elapsed between committing code and deploying it to production, tracking the time spent on implementing, testing, and delivering changes to the codebase. BA-8 positively influenced product delivery lead time, indicating the ability to move quickly through the process. Product delivery lead time, for example, has other high-influence items: total funding, developer staffing, developer skill, development environment, test facilities, developmental and operational test support, and system complexity, as seen in Table 2.

The USD A&S report for FY2022 states that numerous factors, not just BA-8, have an equivalent or more significant impact on metric outcomes. Therefore, it was difficult to quantify the effect of BA-8 in isolation precisely. Table 2 provides a comprehensive analysis of the factors influencing software program activities by considering these additional variables.

Table 2. Software Factory "Other" Quantitative Factors Having Influence

Factor Influence on Measure	Product Delivery Lead Time	Release Frequency to Operational Environment	Deployment Frequency to Production	Mean Time to Restore	Change Fail Percentage
Total Funding	High	Medium	Medium	Medium	Medium
Developer Staffing	High	Medium	High	High	High
Developer Skill	High	Low	Low	Medium	High
Development Environment	High	Medium	Medium	High	Low
Test Facilities	High	High	High	Medium	High
Developmental & Operational Test Support	High	Low	High	Medium	Low\\
Time to get Authority to Operate	Low	Medium	High	Low	N/A
Capability Complexity	High	High	High	High	High
User Ability to Accept Releases	N/A	N/A	High	N/A	N/A
Contracting Methods	Medium	Low	Low	N/A	N/A

Metrics, such as product development lead time and deployment frequency into production, help teams understand their overall engineering performance. In addition, they provide the software program with an objective way to measure and improve software delivery. Metrics help DevSecOps teams quickly identify bottlenecks and inefficient processes in their development pipeline and create a plan to improve their daily work (Software.com, 2023). Several quantitative factors identified in the BA-8 Pilot play a crucial role in delivering high-quality software to the user within the deadline and are of significant value beyond the BA-8 efforts.

Software Metrics That Add Value

Commercial Software teams use modern iterative software methods to emphasize development using fixed cost and time, with flexible requirement estimates. Defining all of the software requirements at the program's start is impractical, as this is counter to agile software non-deterministic development methodology. Current software cost estimation and reporting processes and procedures in the DoD have proven to be time-consuming, highly inaccurate, and time late. Metrics of Earned Value Management for software development cannot match the continuous capability delivery and maintenance velocity of DevSecOps. Metrics that align with the DevSecOps approach and offer continuous visibility into program progress are necessary.

The SWAP report recommends that projects develop metrics that measure value to the user (or customer satisfaction), which involves close, ongoing communication with users. How this metric of "user value" is calculated is undefined in the BA-8 Pilot. In the commercial sector, many agile software teams use broader business indicators to gauge overall performance and product quality. The software factory doesn't directly own or collect data for these metrics since they represent customer satisfaction, value delivery, and flexibility.

The measurement of cost and performance for software factories are automated within the infrastructure tools and report continuous speed and cycle time, cybersecurity vulnerabilities, code quality, and functionality to assess, manage, and justify terminating a software program (if needed). In addition, software code performance metrics address issues such as deployment rate and speed of delivery, response, and recovery from outages, and can be automatically generated continuously.

Future Funding of Software Programs Uncertain

Congress did not authorize additional BA-8 pilots in FY2023 due to the perceived lack of quantitative metrics from the USD A&S. All of the Senior Department Software Acquisition Executives provided qualitative inputs for the BA-8 Pilot, Fourth FY2022 Quarterly Report to Congress, and the Army's comment on the value of BA-8 funding (OUSD A&S, 2022) was particularly relevant to this paper.

> Given the modern and ever-changing software environment, the legacy funding model of RDT&E, procurement, and O&M makes it difficult to effectively and efficiently acquire and develop software. With the Army's need to remain competitive and defeat near-peer adversaries, the Army must be able to rapidly secure, enhance, and maintain software. Legacy software development practices cannot keep up with the pace of change required to address the ever-changing threat landscape. They also establish clear lines between software development (new capabilities) and software maintenance (cyber and software fixes) activities. This division of activities aligned well with current funding models; development = RDT&E and maintenance = O&M.
>
> With the advancements of cloud computing, Agile software development, and Development, Security, and Operations (DevSecOps), everything is

> integrated and must operate at a rapid pace. These modern software practices do not distinguish between software development and software maintenance. The software is viewed as a product that is continuously evolving. These practices involve adding capability, fixing software problems, and cyber-securing software with a single team as part of a single software delivery. This cultural and technological change removes the line between software development and software sustainment, making it challenging to fund those activities separately with different appropriations. Without the use of BA-8, it will require a very cumbersome and difficult process to identify exactly the number of hours each team member spends on adding capability (RDT&E) and fixing problems (sustainment – O&M).

From a qualitative perspective, the services agreed with the SWAP report view of the value of colorless money. The understanding that software is no longer a monolithic delivery and that capability can be delivered to the warfighter at the tactical edge in lightweight application containers is a quantum jump forward. However, the genuine concern ultimately resides with the funding needed for the software factory itself.

Maybe Treat Software Factories as an Enduring Service?

The DoD's issues with various appropriation categories could be addressed by adopting existing best practices in the private sector by establishing software factories as an enduring service. Software Factories established per the DoD software modernization strategy of 2021 should combine Cloud-based computing and use an assembled set of software tools enabling developers, users, and management to work together on a daily tempo to achieve delivery of a minimum viable product. The software development continues until a minimum viable capability is released into the user community. Funding for the software factory becomes either a time of material or a level of effort contract expense for labor that ebbs and flows as the software factory continues to add user-desired capabilities during the execution phase. In addition, the software factory itself has expenses such as software tool licenses and government salaries. As the number of features to be coded and delivered decreases over time, the software factory can either start new tasking from another pillar program or reduce the workforce to keep a core capability while the software is in user operation. Whether it is a new capability, fixing a deficiency, or cyber vulnerability, it is colorless money.

Software factories provide significant value as an enduring service for software development. By providing a consistent, standardized approach to software development, software factories can help to increase productivity, improve collaboration and quality, reduce risk, and provide ongoing support and maintenance for software applications. Additionally, software factories can benefit individual developers, enabling them to work with new technologies and tools and improve their skills over time. As software development becomes increasingly complex and demanding, software factories may play an essential role in enabling teams to work more efficiently and effectively and deliver high-quality software applications.

Concluding Thoughts

Pilot results are essential in confirming study assertions and making necessary adjustments to achieve desired results. Congress and the DoD have been aware since the Defense Science Board study of 1987 that monolithic waterfall acquisition of software takes too long, is too expensive, and exposes warfighters to an unacceptable risk by delaying their access to the software needed to ensure mission success. Software is responsible for most of the capabilities in our weapon systems and applications that provide command, control, and communications. The DoD realized a different “adaptive” acquisition process was needed to speed technology and capability delivered to the warfighter. The USD A&S in 2020 provided the framework leadership needs to drive change in software acquisition. By leveraging commercial practices, experience, and tools, the DoD implemented the DevSecOps (Software Factory) initiative to support software as an enduring and evolving capability that is continuously improved throughout its lifecycle. Modifying statutes, regulations, and processes will not accomplish the enduring result needed to prioritize speed and continuous capability delivery to the warfighter. The color of money pilot (BA-8) and the measures of effectiveness have revealed that other software factory influences, if addressed proactively, may have a positive synergistic effect on delivery velocity.

The BA-8 Pilot Software-intensive programs picked all had one thing in common; none were in receiving funding other than RDT&E appropriations. The ability of a software program to estimate years in advance in PPBES exactly how much RDT&E, Procurement, and O&M dollars are needed to support the software is an inexact art. Reprograming funding between appropriations is non-trivial and highly time-consuming. A new threat that emerges to a fielded system or an opportunity to exploit an enemy’s weakness may be lost in the bureaucracy of money exchange. Bleached or colorless money in the hands of the software factory would allow enhanced containerized applications to be developed, tested, and sent to the tactical edge in days, if not hours.

> Software is an enduring and evolving capability that must be supported and continuously improved throughout its lifecycle. DoD’s acquisition process and culture need to be streamlined for effective delivery and oversight of multiple types of software-enabled systems, at scale, and at the speed of relevance. Optimizing for software is the path forward. (DIB, 2019)

References

Defense Acquisition University. (2023, March). *Operations and maintenance (O&M) funds*. https://www.dau.edu/acquipedia/pages/ArticleContent.aspx?itemid=339

Defense Innovation Board. (2019, May). *Software is never done: Refactoring the acquisition code for competitive advantage.* Department of Defense. https://media.defense.gov/2019/Mar/14/2002101480/-1/-1/0/DIB-SWAP_STUDY_REPORT.PDF

Department of Defense Chief Information Officer. (2021, March). *Department of Defense enterprise DevSecOps strategy guide*. https://dl.dod.cyber.mil/wp-content/uploads/devsecops/pdf/DoDEnterpriseDevSecOpsStrategyGuide.pdf

Office of the Under Secretary of Defense for Acquisition and Sustainment. (2020a, January 23). *Operation of the adaptive acquisition framework* (DoD Instruction 5000.02). Department of Defense. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500002p.pdf?ver=2020-01-23-144114-093

Office of the Under Secretary of Defense for Acquisition and Sustainment. (2020b, October 2). *Operation of the software adaptive acquisition pathway* (DoD Instruction 5000.87). Department of Defense. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF?ver=virAfQj4v_LgN1JxpB_dpA%3D%3D

Office of the Under Secretary of Defense for Acquisition and Sustainment. (2022, November). *BA-8 pilot, fourth FY22 quarterly report to Congress*.

Program Executive Office Integrated Warfare Systems X. (2022, July 28). *ICS overview brief*.

Software.com. (2023, March). *Key engineering metrics in software delivery*. https://www.software.com/devops-guides/engineering-metrics