Software Is Never Done

Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board

May 3, 2019

Cover design by the Office of Secretary of Defense, Graphics and Presentations Division

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

DIB | DEFENSE INNOVATION BOARD

5 April 2019

I am pleased to forward the final report of the Software Acquisition and Practices (SWAP) study conducted by the Defense Innovation Board (DIB). The study, co-chaired by Dr. Richard Murray and Dr. Michael McQuade, was executed pursuant to Section 872 of the 2018 National Defense Authorization Act (NDAA). It makes 10 primary recommendations and 16 additional recommendations to address the most critical statutory, regulatory, and cultural hurdles DoD faces in modernizing its approach to software. In a year replete with reports on artificial intelligence, quantum computing, and blockchain, it may seem mundane to write about software, but software is the foundation of all things digital, and the Department does it exceedingly poorly. It is scarcely possible to imagine how the Department can achieve the modernization objectives of the National Defense Strategy without overhauling its approach to software, which is what motivated us to take on this task.

I have questioned the usefulness of the myriad studies and reports on technology authorized by Congress and DoD over the years. These mandates are often executed in a vacuum and the recommendations rarely seen through to implementation. Consequently, we resolved early on to conduct the SWAP study differently, with emphasis on pragmatism, accessibility, and candor. The iterative and inclusive approach the SWAP study team opted to take gave Congress, industry partners, and Department stakeholders at all levels unprecedented opportunity to participate in and contribute to the vision for the future of DoD software. Most importantly, each one of these communities provided critical feedback to ensure the recommendations contained in this report are timely, actionable, and contextualized by the ultimate mission -- swiftly delivering capability to the warfighter. Fittingly, it seems the recipe for delivering value in reports is the same as in modern software development: user feedback. Ultimately, we produced a report that changed my mind about the usefulness of such reports, and I hope you agree.

I hope the draft implementation plans at Appendix A are particularly valuable; we intended to provide sufficient detail to make them immediately implementable, but not so much as to be inflexible. They should provide the Department and Congress a starting point from which to act on the recommendations and instill critical accountability across the organization and in industry.

I recommend that this report be distributed to the offices within OSD and the Services that deal with software technology and acquisition directly, and to the appropriate industries and organizations that support them, as well as to numerous organizations that have seen their mission transformed by software, however reluctant they may appear to be to acknowledge it. With regard to industry and private sector partners, we deliberately wrote for the defense industrial base as well as for those companies in the national security innovation base that may not identify themselves as defense suppliers, but whose support will be essential for the Department's future technological relevance. Pay mind to the report's first theme: software *truly* is ubiquitous; and a modern acquisition and development approach should be architected at the enterprise-level with broad support and understanding from the "users."

Finally, I fully support the DIB's further involvement, at the discretion of relevant Department leadership, as advisors in the implementation of the recommendations contained in this report. We consider it our privilege to have engaged with such talented and devoted uniformed personnel, civilians, and contractors over the course of the SWAP study and we welcome a continued dialogue about how to make the ideas contained in this report a reality.

Sincerely,

[illegible]

Dr. Eric Schmidt
Chairman
Defense Innovation Board

DIB | DEFENSE INNOVATION BOARD

MEMORANDUM TO THE CHAIRMAN, DEFENSE INNOVATION BOARD

SUBJECT: Final Report of the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) Study

Attached is the final report from the DIB's SWAP study, documenting our efforts, analysis and conclusions working with Congress and DoD on how to develop, procure, assure, deploy, and continuously improve software for use in the Department. In developing this report over the past eighteen months, we have had substantial conversations with congressional staffers, DoD leadership at many levels, program offices, contractors, (government and private sector) software developers, and a variety of other representatives from government, industry, academia, and the public. There is broad consensus on the goal of delivering high quality software to DoD users in a manner that is timely, secure, and cost effective. Our study details the external and self-inflicted barriers DoD faces in implementing modern software practices and lays out steps to address current gaps. We hope that our recommendations will serve as a basis for the implementation phase of this work. We are happy and ready to support that effort.

We would like to thank the study members, Gilman Louie, Milo Medin, Jennifer Pahlka and Trae' Stephens for their contributions to the report. Bess Dopkeen served as the initial study director and established an outstanding structure for the study and the support staff. Jeff Boleng supported the study throughout its initial phase and stepped in as study director after Bess. We would also like to acknowledge the outstanding help provided by Courtney Barno, Kevin Garrison, Nick Guertin, Devon Hardy, Sandra O'Dea, Forrest Shull, and Craig Ulsh. A longer list of the many people who have helped on the study is included in the Acknowledgements section of the main report and in Appendix J.

J Michael McQuade

J. Michael McQuade

[illegible]

Richard M. Murray

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

Extended Abstract

U.S. national security increasingly relies on software to execute missions, integrate and collaborate with allies, and manage the defense enterprise. The ability to develop, procure, assure, deploy, and continuously improve software is thus central to national defense. At the same time, the threats that the United States faces are changing at an ever-increasing pace, and the Department of Defense's (DoD's) ability to adapt and respond is now determined by its ability to develop and deploy software to the field rapidly. The current approach to software development is broken and is a leading source of risk to DoD: it takes too long, is too expensive, and exposes warfighters to unacceptable risk by delaying their access to tools they need to ensure mission success. Instead, software should enable a more effective joint force, strengthen our ability to work with allies, and improve the business processes of the DoD enterprise.

Countless past studies have recognized the deficiencies in software acquisition and practices within DoD, but little seems to be changing. Rather than simply reprint the 1987 Defense Science Board (DSB) study on military software that pretty much said it all, the Defense Innovation Board's (DIB's) congressionally mandated study[1] on Software Acquisition and Practices (SWAP) has taken a different approach. By engaging Congress, DoD, Federally Funded Research and Development Centers (FFRDCs), contractors, and the public in an active and iterative conversation about how DoD can take advantage of the strength of the U.S. commercial software ecosystem, we hope to move past the myriad reports and recommendations that have so far resulted in little progress. Past experience suggests we should not anticipate that this report will miraculously result in solutions to every obstacle we have found, but we hope that the two-year conversation around it will provide the impetus for figuring out how to make the changes for which everyone is clamoring.

In this report, we emphasize three fundamental themes:

1. **Speed and cycle time are the most important metrics for managing software**. To maintain advantage, DoD needs to procure, deploy, and update software that works for its users at the speed of mission need, executing more quickly than our adversaries. Statutes, regulations, and cultural norms that get in the way of deploying software to the field quickly weaken our national security and expose our nation to risk.
2. **Software is made by people and for people, so digital talent matters**. DoD's current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly fast or deep enough to meet its mission needs. New mechanisms are needed for attracting, educating, retaining, and promoting digital talent and for supporting the workforce to follow modern practices, including developing software hand in hand with users.

[1] 2018 National Defense Authorization Act (NDAA), Sec. 872. Defense Innovation Board analysis of software acquisition regulations.

3. **Software is different than hardware (and not all software is the same)**. Hardware can be developed, procured, and maintained in a linear fashion. Software is an enduring capability that must be supported and continuously improved throughout its life cycle. DoD must streamline its acquisition process and transform its culture to enable effective delivery and oversight of multiple types of software-enabled systems, at scale, and at the speed of relevance.

To take advantage of the power of software, we advocate four main lines of effort:

A. **Congress and DoD should refactor statutes, regulations, and processes for software**, enabling rapid deployment and continuous improvement of software to the field and providing increased insight to reduce the risk of slow, costly, and overgrown programs.

B. **The Office of the Secretary of Defense (OSD) and the Services should create and maintain cross-program/cross-Service digital infrastructure** that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage them using modern development methods; and eliminate the existing hardware-centric regulations and other barriers.

C. **The Services and OSD will need to create new paths for digital talent (especially *internal* talent)** by establishing software development as a high-visibility, high-priority career track and increasing the level of understanding of modern software within the acquisition workforce.

D. **DoD and industry must change the practice of how software is procured and developed** by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cybersecurity is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

Report structure. The main report provides an assessment of the current and desired states for software acquisition and practices, as well as a review of previous reports and an assessment of why little has changed in the way DoD acquires software, with emphasis on three fundamental themes. The report's recommendations are broken into four lines of effort, with a set of primary recommendations provided for each (bold), along with additional recommendations that can provide further improvements. Each recommendation is accompanied by a draft implementation plan and potential legislative language.

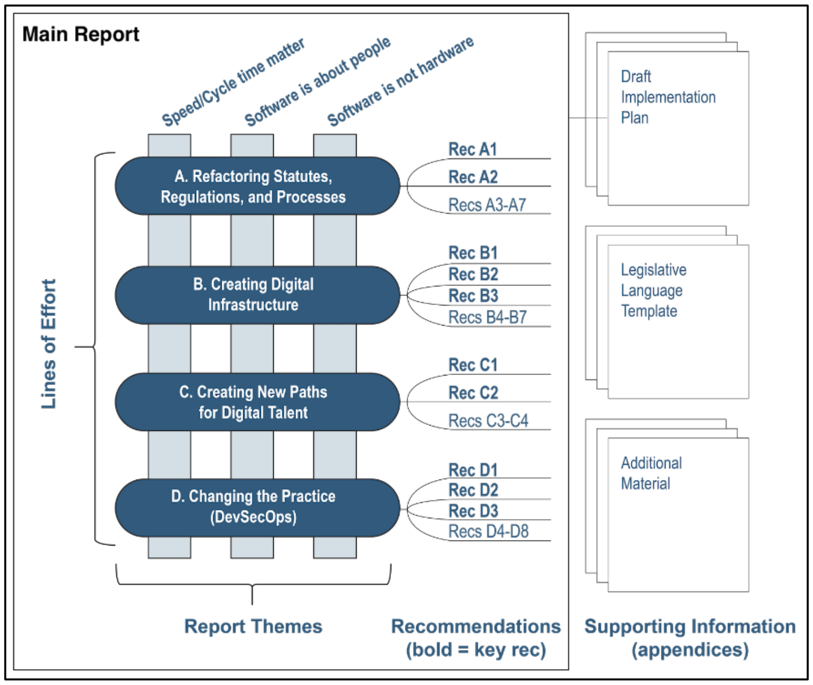

Main Report
Speed/Cycle time matter
Software is about people
Software is not hardware
A. Refactoring Statutes, Regulations, and Processes
Rec A1
Rec A2
Recs A3-A7
B. Creating Digital Infrastructure
Rec B1
Rec B2
Rec B3
Recs B4-B7
C. Creating New Paths for Digital Talent
Rec C1
Rec C2
Recs C3-C4
D. Changing the Practice (DevSecOps)
Rec D1
Rec D2
Rec D3
Recs D4-D8
Lines of Effort
Report Themes
Recommendations (bold = key rec)
Draft Implementation Plan
Legislative Language Template
Additional Material
Supporting Information (appendices)

Table of Contents

Supporting Information

Author's note: This is an abridged version of the Defense Innovation Board's (DIB's) full Software Acquisition and Practices (SWAP) report; some contents have been omitted or reordered to optimize readability and highlight the critical portions. For a complete copy of the report, please visit the DIB's website at https://innovation.defense.gov/software/ or contact the SWAP study team (contact information provided on the back cover of this report).

Chapter 0. README (Executive Summary)

In 2011, Marc Andreessen claimed in an op-ed for *The Wall Street Journal* that "Software Is Eating the World."[2] He argued that *every* industry (not just those considered to be "information technology") would be transformed by software—bytes rather than atoms. Eight years later, it is clear he was right.

This transformation is happening in defense, and we are not prepared for it. Software is leveling the playing field with our rivals, eroding the advantages we have spent many decades accruing. Software is the focal point of many important advances in national security technology, including data analytics, artificial intelligence (AI), machine learning (ML), and autonomy. Software is ubiquitous. It is part of everything the Department of Defense (DoD) does, from logistics to management to weapon systems. U.S. national security is critically dependent on the capabilities of DoD's software.

DoD must be able to develop, procure, assure, deploy, and continuously improve software faster than our adversaries. Unfortunately, DoD still treats software much like hardware, and often misunderstands the relationship between speed and security. As a result, a large amount of DoD's software takes too long, costs too much, and is too brittle to be competitive in the long run. If DoD does not take steps to modernize its software acquisition and development practices, we will no longer have the best military in the world, no matter how much we invest or how talented and dedicated our armed forces may be.

The good news is that there are organizations within DoD that have already acknowledged the risks of falling further behind in software and are leveraging more modern acquisition and development practices with notable success. The Defense Digital Service (DDS), the Defense Innovation Unit (DIU), the Joint Improvised-Threat Defeat Organization (JIDO), and the Air Force's Kessel Run are examples that demonstrate that DoD has the ability to ship world-class software. The challenge remains doing this at scale.

DoD needs to build on these foundations to create an ecosystem and standard operating procedures that enable the practices of great software without requiring employees to "hack the system." To do that, we must address the prioritization, planning, and acquisition processes and policies that create the worst bottlenecks for deploying capability to the field at the speed of relevance. Further, we must address all the practices that not only put the U.S. Armed Forces at risk and reduce the efficiency of DoD's operations, but also drive away the very people who are most needed to develop this critical capability.

Our adversaries are already doing this. China actively leverages its private industry to develop national security software (particularly in AI), recruits top students under the age of 18 to work on "intelligent weapons design,"[3] and poaches U.S. software talent directly from the United States. In Russia, Vladimir Putin has told students, that "artificial intelligence is the future, not only for Russia, but for all humankind.... Whoever becomes the leader in this sphere will become the ruler

[2] Marc Andreessen, "Why Software Is Eating the World," *The Wall Street Journal*, August 20, 2011, 1.
[3] Stephen Chen, "China's Brightest Children Are Being Recruited to Develop AI 'Killer Bots,'" *South China Morning Post*, November 8, 2018.

of the world."[4] We can and must outcompete with software and the people who make it, not only to maintain U.S. military superiority but also to ensure that the power that software represents is used in accordance with American values.

What this report is about. This report summarizes the assessment of the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study. Congress charged[5] the DIB to recommend changes to statutes, regulations, processes, and culture to enable the better use of software in DoD. We took an iterative approach, mirroring the way modern software is successfully done, releasing a sequence of concept papers describing our preliminary observations and insights. (The latest versions of these are included in Appendix E.) We used those papers to encourage dialogue with a wide variety of individuals and groups to gain insights into the current barriers to implementing modern software effectively and efficiently. This document captures key insights from these discussions in an easy-to-read format that highlights the elements that we consider critical for DoD's success and serves as a starting point for continued discussions required to implement the changes that we recommend here.

This report is organized as follows:

- **Extended Abstract:** A two-page summary of the key takeaways from the report.
- **README** (this document)**:** A more detailed executive summary of the report. (A README file is used by the open source software community to provide essential information about a software package.) If your boss heard about the report or read the extended abstract, thought it was intriguing, and asked you to read the entire report and provide a short summary, cut and paste this chapter into your reply and you should be good to go.
- **Recommendations Cheat Sheet:** A list of the main lines of effort and primary recommendations, so you can pretty much stop at that point—or better yet, stop after suggesting to your boss they adopt them all.
- **Chapters 1–4:** Short descriptions of key areas and topics. If you attach the extended abstract to any one of these as a preface, it should be comprehensible.
- **Chapter 5:** A more detailed description of the recommendations and our rationale.
- **Supporting Information:** To ensure that the executive summary and the main body of the report satisfy the takeoff test[6] and the staple test,[7] we put most of the additional information generated during the study into a set of appendices. These provide a wealth of examples and

[4] James Vincent, "Putin Says the Nation that Leads in AI 'will be the ruler of the world,'" *The Verge*, September 4, 2017: https://www.theverge.com/2017/9/4/16251226/russia-ai-putin-rule-the-world.
[5] Section 872 of the FY18 NDAA directed the Secretary of Defense to "direct the Defense Innovation Board to undertake a study on streamlining software development and acquisition regulations." The DIB-SWAP members were charged to "review the acquisitions regulations applicable to, and organizational structures within, the Department of Defense…; review ongoing software development and acquisition programs…; produce specific and detailed recommendations…; and produce such additional recommendations for legislation." See Section 872 of the FY18 NDAA at https://www.congress.gov/115/plaws/publ91/PLAW-115publ91.pdf or Appendix J of this report.
[6] Reports should be short enough to read during takeoff, before the movies start and drinks are served.
[7] Any report that is going to be read should be thin enough to be stapled with a regular office stapler.

evidence, but we took care to put our essential arguments up front for less wonky types. Some highlights:

- **Draft implementation** (Appendix A)**:** For each recommendation, a summary of the background, desired state, stakeholders, role of Congress, and actions to be taken.
- **Legislative language** (Appendix B)**:** In response to 2016 NDAA Section 805, template legislative language for a new acquisition pathway and appropriation category for software, aligned with our recommendations.
- **An alternative to P-Forms and R-Forms** (Appendix C)**:** A different mechanism for budget submissions for software programs.
- **FAQs** (frequently asked questions, Appendix D)**:** A list of the most common questions that we get about the study and our attempt to answer them. (Question 1: Hasn't all of this been recommended before? A: Yes…).

Note: If you are reading any portion of the report in paper form, a navigable version is available at http://innovation.defense.gov/software.

Overarching themes. The rise of electronics, computing, and networking has forever transformed the way we live: software is a part of almost everything with which we interact in our daily lives, either directly through embedded computation in the objects around us or indirectly through the use of information technology through all stages of design, development, deployment, and operations. Our military advantage, coordination with allies and partners, operational security, and many other aspects of DoD activities are all contingent upon our software edge, and any lack thereof presents serious consequences. Software drives our military advantage: what makes weapon systems sophisticated is the software, not (just) the hardware.

Commercial trends show what is possible with software, from the use of open source tools to agile development techniques to global-scale cloud computing. Because of these changes, software can be developed, deployed, and updated much more quickly, which means systems need to be in place to support this speed. But modern software development requires a new set of skills and methodologies (e.g., generalist software engineers, specialized product management, DevOps and DevSecOps, agile development). Hence, the policies and systems surrounding software must be transformed to support software, not Cold-War-era weapon manufacturing.

The incoming generation of military and civilian personnel began life digitally plugged-in, with an innate reliance on software-based systems. They will demand new concepts of operations, tactics, and strategies to maintain the edge they need. If DoD can refactor its acquisition processes and transform its culture and personnel policies before it is too late, this software-savvy generation can still set the Department on the right course.

As we studied the methods that the private sector has used to enable software to transform its operations and considered how to best apply those practices to the defense enterprise, three overarching themes emerged as the basis for our recommendations:

1. Speed and cycle time are the most important metrics for software.

2. Software is made by people and for people, so digital talent matters.
3. Software is different than hardware (and not all software is the same).

Speed and cycle time are the most important metrics for software. Most DoD software projects are currently managed using “waterfall” development processes, which involve spending years on developing requirements, taking bids and selecting contractors, and then executing programs that must meet the listed requirements before they are “done.” This results in software that takes so long to reach the field that it is often not well matched to the current needs of the user or tactics of our adversaries, which have often changed significantly while the software was being written, tested, and accepted. Being able to develop and deploy faster than our adversaries means that we can provide more advanced capabilities, respond to our adversaries’ moves, and be more responsive to our end users. Faster reduces risk because it demands focus on the critical functionality rather than over-specification or bloated requirements. It also means we can identify trouble earlier and take faster corrective action, which reduces cost, time, and risk. Faster leads to increased reliability: the more quickly software/code is in the hands of users, the more quickly feedback can focus on efforts to deploy greater capability. Faster gives us a tactical advantage on the battlefield by allowing operation and response inside our adversaries’ observe–orient–decide–act (OODA) loops. Faster is more secure. Faster is possible.

Software is made by people and for people, so digital talent matters. Current DoD human resource policies are not conducive to attracting, retaining, and promoting digital talent. Talented software developers and acquisition personnel with software experience are often put in jobs that do not allow them to make use of those talents, particularly in the military where rotating job assignments may not recognize and reward the importance of software development experience. As Steve Jobs observed,[8] one of the major differences between hardware and software is that for hardware the “dynamic range” (ratio between the best in class and average performance) is, at most, 2:1. But, the difference between the best software developer and an average software developer can be 50:1, or even 100:1, and putting great developers on a team with other great developers amplifies this effect. Today, in DoD and the industrial base that supports it, the people with the necessary skills exist, but instead of taking advantage of their skills we put them in environments where it is difficult for them to be effective. DoD does not take advantage of already existing military and civilian personnel expertise by offering pay bonuses, career paths that provide the ability to stay in their specialization, or access to early promotions. Skilled software engineers and the related specialties that are part of the overall software ecosystem need to be treated as a special force; the United States must harness their talent for the great benefits that it can provide.

Software is different than hardware (and not all software is the same). Over the years, Congress and DoD have established a sophisticated set of statutes, regulations, and instructions that govern the development, procurement, and sustainment of defense systems. This process evolved in the context of the Cold War, where major powers designed and built aircraft carriers, nuclear weapons, fighter jets, and submarines that were extremely expensive, lasted a very long time, and required tremendous access to capital and natural resources. Software, on the other hand,

[8] Steve Jobs, “Steve Jobs: The Lost Interview,” interview by Robert X. Cringely for the 1995 PBS documentary, *Triumph of the Nerds*, released to limited theaters in 2012, video.

is something that can be mastered by a ragtag bunch of teenagers with very little money—and can be used to quickly destabilize world powers. Currently most parts of DoD develop, procure, and manage software like hardware, assuming that it is developed based on a fixed set of specifications, procured after it has been shown to comply with those specifications, "maintained" by block upgrades, and upgraded by replaying this entire procurement process linearly. But software development is fundamentally different than hardware development, and software should be developed, deployed, and continuously improved using much different cycle times, support infrastructure, and maintenance strategies. Testing and validation of software is also much different than for hardware, both in terms of the ability to automate but also in the potential vulnerabilities found in software that is not kept up to date. Software is never "done" and must be managed as an enduring capability that is treated differently than hardware.

Main lines of effort. DoD's current approach to software is a major driver of cost and schedule overruns for Major Defense Acquisition Programs (MDAPs). Congress and DoD need to come together to fix the acquisition system for software because it is a primary source of its acquisition headaches.

Bringing about the type of change that is required to give DoD the software capabilities it needs is going to take a significant amount of work. While it is possible to use the current acquisition system and DoD processes to develop, procure, assure, deploy, and continuously improve DoD software, the statutes, regulations, processes, and culture are debilitating. The current approach to acquisition was defined in a different era, for different purposes, and only works for software projects through enormous effort and creativity. Congress, the Office of the Secretary of Defense (OSD), the Armed Services, defense contractors, and the myriad government and industry organizations involved in getting software out the door need to make major changes (together).

To better organize our specific recommendations, we identified broad lines of effort that bring together different parts of the defense ecosystem as stakeholders. Here are the four main lines of effort that we recommend they undertake:

A. **(Congress and DoD) Refactor statutes, regulations, and processes for software,** enabling rapid deployment and continuous improvement of software to the field and providing increased insight to reduce the risk of slow, costly, and overgrown programs. The management and oversight of software development and acquisition must focus on different measures and adopt a quicker cadence.

B. **(OSD and the Services) Create and maintain cross-program/cross-Service digital infrastructure** that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage it using modern development methods; and eliminate the existing hardware-centric regulations and other barriers.

C. **(The Services and OSD) Create new paths for digital talent (especially internal talent)** by establishing software development as a high-visibility, high-priority career track—with specialized recruiting, education, promotion, organization, incentives, and salary—and increasing the level of understanding of modern software within the acquisition workforce.

D. **(DoD and industry) Change the practice of how software is procured and developed** by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cyber protection is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

None of these can be done by a single organization within the government. They will require a bunch of hard-working, well-meaning people to work together to craft a set of statutes, regulations, processes, and (most importantly) a culture that recognizes the importance of software, the need for speed and agility (theme 1), the critical role that smart people have to play in the process (theme 2), and the impact of inefficiencies of the current approach (theme 3). In many ways this mission is as challenging as any combat mission: while participants' lives may not be directly at risk in defining, implementing, and communicating the needed changes to policy and culture, the lives of those who defend our nation ultimately depend on DoD's ability to redefine its approach to delivering combat-critical software to the field.

Refactor statutes, regulations, and processes, streamlined for software. Congress has created many workarounds to allow DoD to be agile in its development of new weapon systems, and DoD has used many of these to good effect. But the default statutes, regulations, and processes that are used for software too often rely on the traditional hardware mentality (repeat: software is different than hardware), and those practices do not take advantage of what is possible (or, frankly, necessary, given the threat environment) with modern software. We think that a combination of top-down and bottom-up pressure can break us out of the current state of affairs, and creating a new acquisition pathway that is tuned for software (of various types) will make a big difference. To this end, Congress and DoD should prototype and, after proving success, create mechanisms for ideation, appropriation, and deployment of software-driven solutions that take advantage of the unique features of software (versus hardware) development (start small, iterate quickly, terminate early) and provide purpose-fit methods of oversight. As an important aside, note that throughout this study our recommendations adhere to this guiding axiom—start small, iterate quickly—the same axiom that characterizes the best of modern software innovation cycles (see the "DIB Ten Commandments of Software" in Appendix E for more information about the DIB's guiding principles for software acquisition).

Create and maintain cross-program/cross-Service digital infrastructure. Current practice in DoD programs is that each individual program builds its own infrastructure for computing, development, testing, and deployment, and there is little ability to build richer development and testing capabilities that are possible by making use of common infrastructure. Instead, we need to create, scale, and optimize an enterprise-level architecture and supporting infrastructure that enables creation and initial fielding of software within six months and continuous delivery of improvements on a three-month cycle. This "digital infrastructure," common in commercial IT, is critical to enable rapid deployment at the speed (and scale) of relevance. In order to implement this recommendation, Congress and DoD leadership must figure out ways to incentivize the Services and defense contractors to build on a common set of tools (instead of inventing their own) *without* just requiring that everyone uses one DoD-wide (or even Service-wide) platform. Similarly, OSD will have to define non-exceptions-based alternatives to (or at least pathways through) Joint Capabilities Integration and Development System (JCIDS), Planning, Programing, Budget and Execution

(PPB&E), and Defense Federal Acquisition Regulation Supplement (DFARS)[9] that are optimized for software. The Director, Operational Test and Evaluation (DOT&E) will need new methods for OT&E that match the software's speed of relevance, and Cost Assessment and Program Evaluation (CAPE) will have to capture better data and leverage AI/ML as a tool for cost assessment and performance evaluation. Finally, the Services will need to identify, champion, and measure platform-based, software-intensive projects that increase software effectiveness, simplify interconnectivity among allies, and reform business practices. Subsequent chapters in our report provide specific recommendations on each of these areas.

Create new paths for digital talent (especially internal talent). The biggest enabler for great software is providing great people with the means to contribute to the national security mission. While the previous recommendations speak to providing the tools and infrastructure DoD technologists need to succeed, it is equally important that the Department's human capital strategies allow them to even do this work consistently in the first place. Driving the cultural transformation to support modern, cloud-based technology requires new types of skills and competencies, changing ratios of program managers to software engineers, moving from waterfall development to DevSecOps[10] development, and dealing with all of the change management that comes with it. This is not an easy task, but arguably one of the most important. While compensation is a major driver in attracting competitive talent, DoD must also make changes in the roles, methodologies, cultures, and other aspects of the transformation that industry is already undergoing and that the government must undergo as well.

Increasing developer talent is not the only workforce challenge. DoD must also change how the government manages its programs and contractors, which goes beyond just moving to DevSecOps development. The government must have experts well steeped in the software development process and architecture design to adequately manage both organic activities and contracted programs. They must have the skills to detect when contractors are going down the wrong path, choosing a bad implementation approach, or otherwise wasting government resources. This is perhaps the best argument for ensuring we have software development experience natively in the government, rather than relying primarily on external vendors; unless there are software-knowledgeable members on the core team, it is impossible to effectively monitor and manage outsourced projects. This is especially true with the movement to DevSecOps.

In implementing this change in the workforce, it is particularly important to provide new career paths for digital talent and enable the infrastructure and environment required to allow them to succeed. The current General Schedule (GS) system favors time in grade over talent. This simply will not work for software. The military promotion system has the same problem. As with sports, great teams make a huge difference and, in software, we need to make sure those teams have the tools they need to succeed and reward them appropriately—through recognition, opportunities for impact, career advancement, and pay. Advanced expertise in procurement, project management, evaluation and testing, and risk mitigation strategies will also be needed to create the types of elite teams that are necessary. A key element of success is finding ways to keep talented

[9] Common DoD acronyms are defined in Appendix I (Acronyms and Glossary).
[10] An iterative software development methodology that combines development, security, and operations as key elements in delivering useful capability to the user of the software. See Section 2.1 for details.

people in their roles (rather than transferring them out because it is the end of their assignment), and promoting people based on their abilities, not based on their years of service.

Change the practice of how software is procured and developed. The items above are where we think Congress and the Department should focus in terms of statutory, regulatory, and process changes. But a major element is also the need to change the *culture* around software within Congress, DoD, and the defense industrial base. We use the term “DevSecOps” as our label for the type of culture that is needed: iterative development that deploys secure applications and software into operations in a continuing (and continuous) fashion.

Numerous projects and groups have demonstrated the ability to implement DevSecOps within the existing acquisition system. But the organizations we previously mentioned—DDS, JIDO, DIU, and Kessel Run—are the exception rather than the rule, and the amount of effort required to initiate and sustain their activities is enormous. Instead, DoD should make legacy programs that use outdated techniques for developing software fight for existence (and in most cases replace them with new activities that embrace a DevSecOps approach).

Getting started now. The types of changes we are talking about will take years to bring to complete fruition. But it would be a mistake to spend two years figuring out what the answer should look like, spend another two years prototyping the solutions to make sure we are right, and then spend two to four more years implementing the changes in statutes, regulations, processes, and culture that are actually required. Let’s call that approach the “hardware” approach. Software is different than hardware, and therefore the approach to implementing change for software should be different as well.

Indeed, most (if not all) of the changes we are recommending are not new and not impossible to make. The 1987 DSB Task Force on Military Software,[11] chaired by legendary computer scientist Fred Brooks, wrote an outstanding report that already articulated much of what we are saying here. And the software industry has already implemented and demonstrated the utility of the types of changes we envision. The problem appears to be in getting the military enterprise to adopt a software mindset and implement a DevSecOps approach in a system that was intended to make sure that things would not move too quickly.

DoD could address many of our issues by adopting existing best practices of the private sector for agile development, including making use of software as a service; taking advantage of modern (cloud) infrastructure, tools, computing, and shared libraries; and employing modern software logistics and support delivery systems for software maintenance, development, and updating (patching). We do not need to study these; we need to get going and implement them. Here is a proposed timeline for implementing the primary recommendations of this report, starting *now*:

- (Immediately): Define, within 60 days after delivery of this report to Congress, a detailed implementation plan and assign owners to begin each of the top recommendations.

[11] Defense Science Board Task Force, *Military Software* (Washington, DC: Office of the Under Secretary of Defense for Acquisition, September 1987), https://apps.dtic.mil/dtic/tr/fulltext/u2/a188561.pdf.

- FY19 (create): High-level endorsement of the vision we articulate here, and support for activities that are consistent with the desired end state (i.e., DevSecOps and enterprise-level architecture and infrastructure). Identify and launch programs to move out on the priority recommendations (start small, iterate quickly). If you are reading this and are in a position of leadership in your organization, pass this on to others with your seal of approval and a request for your team to develop two or three plans of action for how it can be applied in your domain. If someone comes to you with a proposal that aligns with the objectives we have outlined here, find a way to be on the front line of changing DoD to a "culture of yes."
- FY20 (deploy): Initial deployment of authorities, budgets, and processes for software acquisition and practices reform. Execute representative programs according to the lines of effort and recommendations in this report, implement now, measure results, and modify approaches. Implement this report in the way we implement modern software.
- FY21 (scale): Streamlined authorities, budgets, and processes enabling software acquisition and practices reform at scale. In this time frame, we need a new methodology to estimate as well as determine the value of software capability delivered (and not based on lines of code).
- FY22 (optimize): Conditions established so that all DoD software development projects transition (by choice) to software- enabled processes, with the talent and ecosystem in place for effective management and insight.

In the remainder of this report, we provide a rationale for the approach that we are advocating. Chapter 1 makes the case for why software is important to DoD, including a taxonomy of the different types of software that need to be considered (not all software is the same). In Chapter 2, we describe how software is developed in the private sector and what is required in terms of workforce, infrastructure, and culture. Chapter 3 is an attempt to summarize what has already been said by other studies and groups, why the situation has not changed, and how we think this study can potentially lead to a different outcome. Chapters 4 and 5 contain our recommendations for how to move forward. In Chapter 4, we present three alternative paths to consider: doing the best we can with the current system; streamlining statutes, regulations, and processes so that they are optimized for software (instead of hardware); and making more radical changes that create entirely new appropriation categories and acquisition pathways. Finally, Chapter 5 describes the path that we recommend be taken, broken out along the lines of effort described above, and with a set of 10 primary recommendations followed by 16 additional recommendations (a detailed draft implementation plan for implementing each is included in Appendix A).

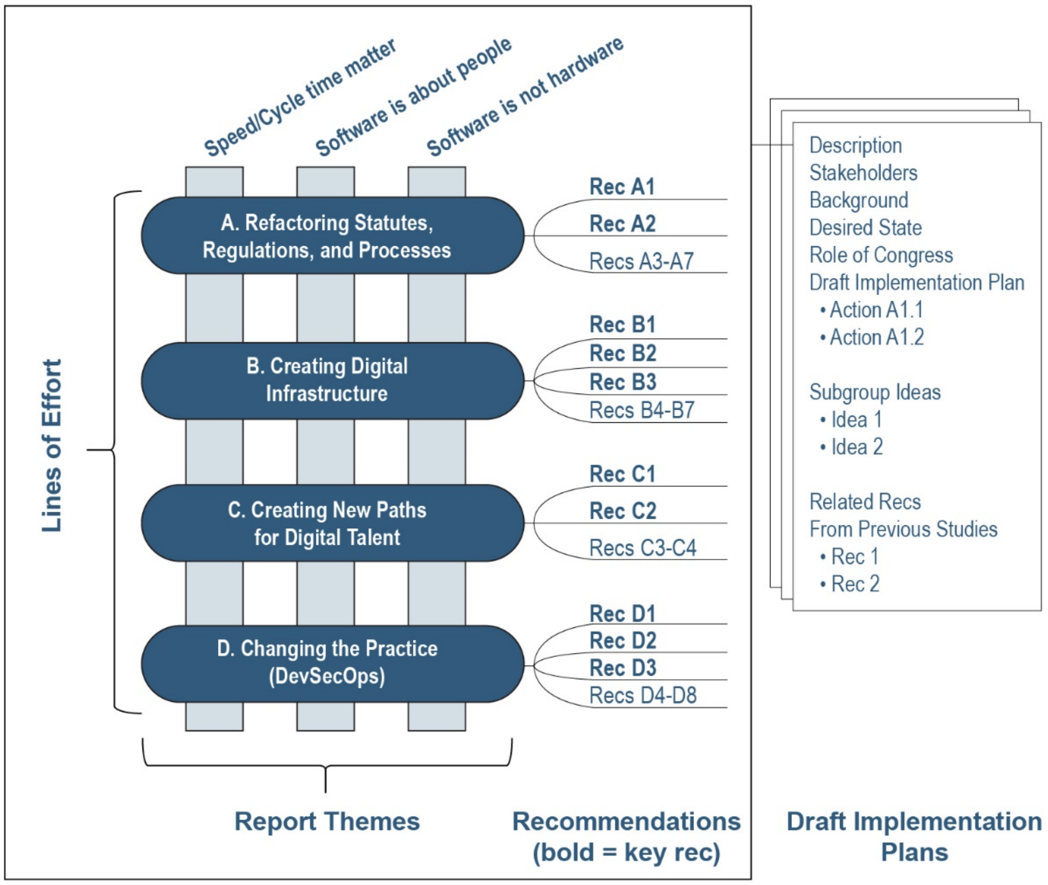
A two-page summary ("cheat sheet") of the lines of effort and recommendations follows.

DIB SWAP Study
Recommendations "Cheat Sheet"

This sheet contains a list of the recommendations from the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study. The recommendations below include input from the following sources:

- DIB Guides for Software (Appendix E)
- SWAP working group reports (Appendix F)
- Previous software acquisition reform studies (starting with the 1987 DSB study)

The recommendations are organized according to four major lines of effort and each recommendation contains background information, a proposed owner for implementing the recommendation, as well as a more detailed draft implementation plan, a list of other offices that are affected, and additional details. The following diagram documents this structure:



For each recommendation, a draft implementation plan can be found in Appendix A that gives more detail on the rationale, supporting information, similar recommendations, specific action items, and notes on implementation. Potential legislative language to implement selected recommendations is included in Appendix B.

The Ten Most Important Things to Do (Starting Now!)

Line of Effort A (Congress and OSD): Refactor statutes, regulations, and processes for software	
A1	Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics
A2	Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment
Line of Effort B (OSD and Services): Create and maintain cross-program/cross-Service digital infrastructure	
B1	Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors
B2	Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis
B3	Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services
Line of Effort C (Services and OSD): Create new paths for digital talent (especially *internal* talent)	
C1	Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices
C2	Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software
Line of Effort D (DoD and industry): Change the practice of how software is procured and developed	
D1	Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source
D2	Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough
D3	Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays

Chapter 5 provides additional context and Appendix A contains draft implementation plans.

Chapter 1. Who Cares: Why Does Software Matter for DoD?

The future battlespace is constructed of not only ships, tanks, missiles, and satellites, but also algorithms, networks, and sensor grids. Like no other time in history, future wars will be fought on civilian and military infrastructures of satellite systems, electric power grids, communications networks, and transportation systems, and within human networks. Both of these battlefields—electronic and human—are susceptible to manipulation by adversary algorithms.

— Cortney Weinbaum and Lt Gen John N.T. "Jack" Shanahan, "Intelligence in a Data-Driven Age," (Joint Force Quarterly 90, 2018), 5

This chapter provides a high-level vision of why software is critical for national security and the types of software we will have to build in the future. We also provide a description of different types of software, where they are used, and why a one-size-fits-all approach will not work.

1.1 Where Are We Coming From, Where Are We Going?

While software development has always been a challenge for the Department of Defense (DoD), today these challenges greatly affect our ability to deploy and maintain mission-critical systems to meet current and future threats. In the past, software simply served as an enabler of hardware systems and weapons platforms. Today, software defines our mission-critical capabilities and our ability to sense, share, integrate, coordinate, and act.

Software is everywhere and is in almost everything that the Department operates and uses. Software drives our weapon systems; command, control, and communications systems; intelligence systems; logistics; and infrastructure, and it drives much of the backroom enterprise processes that make the Department function. If cyber is the new domain in which we are fighting, then our ability to maintain situational awareness and our ability to fight, defend, and counter threats will be based on the capabilities of our software. In this new domain, software is both an enabler as well as a target of the fight.

As our military systems become increasingly networked and automated, as autonomy becomes more prevalent, and as we become more dependent on machine learning (ML) and artificial intelligence (AI), our ability to maintain superiority will be directly linked to our ability to field and maintain software that is better, smarter, and more capable than our adversaries' software. Even our ability to defend against new physical and kinetic threats such as hypersonics, energetics, and biological weapons will be based on software capabilities. We need to identify and respond to these new threats as they happen in near real time. Our ability to identify and respond to these new threats will be based on our ability to develop and push new software-defined capabilities to meet those threats on time scales that greatly outpace our adversaries' ability to do so.

The need to meet future threats requires us to rethink how we develop, procure, assure, deploy, and continuously improve software. DoD's current procurement processes treat software programs like hardware programs, but DoD can no longer take years to develop software for its major systems. Software cannot be an afterthought to hardware, and it cannot be acquired, developed, and managed like hardware. DoD's acquisition and development approaches are increasingly antiquated and do not meet the timely demands of its missions. Fixing the

Department's software approach involves more than just making sure that we get control over cost and budget; it concerns our ability to maintain our fighting readiness and our ability to win the fight and counter any threat regardless of domain and regardless of adversary.

1.2 Weapons and Software and Systems, Oh My! A Taxonomy for DoD

Not all software systems are the same, and therefore it is important to optimize development processes and oversight mechanisms to the different types of software DoD uses. We distinguish here between two different aspects of software: *operational function* (use) and *implementation platform*. To a large extent, a given operational function can be implemented on many different computational platforms depending on whether it is a mission support function (where high-bandwidth connectivity to the cloud is highly likely) or a field-forward software application (where connectivity many be compromised and/or undesirable).

We define three broad operational categories:

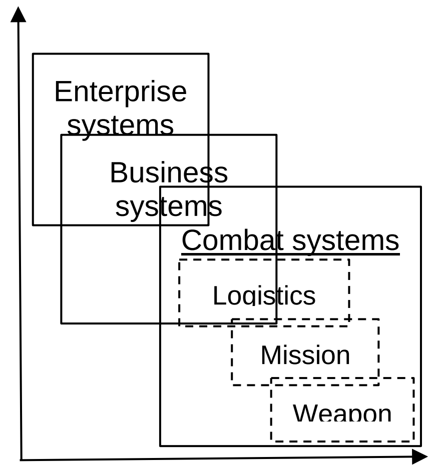


Figure 1.1. Different types of software.

- *Enterprise systems*: very large-scale software systems intended to manage a large collection of users, interface with many other systems, and generally used at the DoD level or equivalent. These systems should always run in the cloud and should use architectures that allow interoperability, expandability, and reliability. In most cases the software should be commercial software purchased (or licensed) without modification to the underlying code, but with DoD-specific configuration. Examples include e-mail systems, accounting systems, travel systems, and human resources (HR) databases.

- *Business systems*: essentially the same as enterprise systems, but operating at a slightly smaller scale (e.g., for one of the Services). Like enterprise systems, they are interoperable, expandable, reliable, and probably based on commercial offerings. Similar functions may be customized differently by individual Services, though they should all interoperate with DoD-wide enterprise systems. Depending on their use, these systems may run in the cloud, in local data centers, or on desktop computers. Examples include software development environments and Service-specific HR, financial, and logistics systems.

- *Combat systems*: software applications that are unique to the national security space and used as part of combat operations. Combat systems may require some level of customization that may be unique to DoD, not the least of which will be specialized cybersecurity considerations to enable them to continue to function during an adversarial attack. (Note that since modern DoD enterprise and business systems depend on software, cyber attacks to disrupt the operations of these systems have the potential to be just as crippling as those aimed at combat systems.)

We further break down combat systems into subcategories:

- *Logistics systems*: any system used to keep track of materials, supplies, and transport as part of operational use (versus Service-scale logistics systems, with which they should interoperate). While used actively during operations, logistics systems are likely to run on commercial hardware and operating systems, allowing them to build on commercial off-the-shelf (COTS) technologies. Platform-based architectures enable integration of new capabilities and functions over time (probably on a months-long or annual time scale). Operation in the cloud or based on servers is likely.

- *Mission systems*: any system used to plan and monitor ongoing operations. Similar to logistics systems, this software will typically use commercial hardware and operating systems and may be run in the cloud, on local services, or via a combination of the two (including fallback modes). Even if run locally (such as in an air operations center), they will heavily leverage cloud technologies, at least in terms of critical functions. These systems should be able to incorporate new functionality at a rate that is set by the speed at which the operational environment changes (days to months).

- *Weapon systems*: any system capable of delivering lethal force, as well as any direct support systems used as part of the operation of the weapon. Note that our definition differs from the standard DoD definition[1] of a weapon system, which also includes any related equipment, materials, services, personnel, and means of delivery and deployment (if applicable) required for self-sufficiency. The DoD definition would most likely include the mission and logistics functions, which we find useful to break out separately. Software on weapon systems is traditionally closely tied to hardware, but as we move toward greater reliability of software-defined systems and distributed intelligence, weapon systems software is becoming increasingly hardware independent (similar to operating systems for mobile devices, which run across many different hardware platforms).

We also define several different types of computing platforms on which the operational functions above might be implemented:

- *Cloud computing*: computing that is typically provided in a manner such that the specific location of the compute hardware is not relevant (and may change over time). These systems typically run on commercial hardware and use commercial operating systems, and the applications running on them run even as the underlying hardware changes. The important point here is that the hardware and operating systems are generally transparent to the application and its users (see figure 1.2).

[1] The Department of Defense, *DoD Dictionary of Military and Associated Terms* (Washington, DC: Department of Defense, as of February 2019), 252.

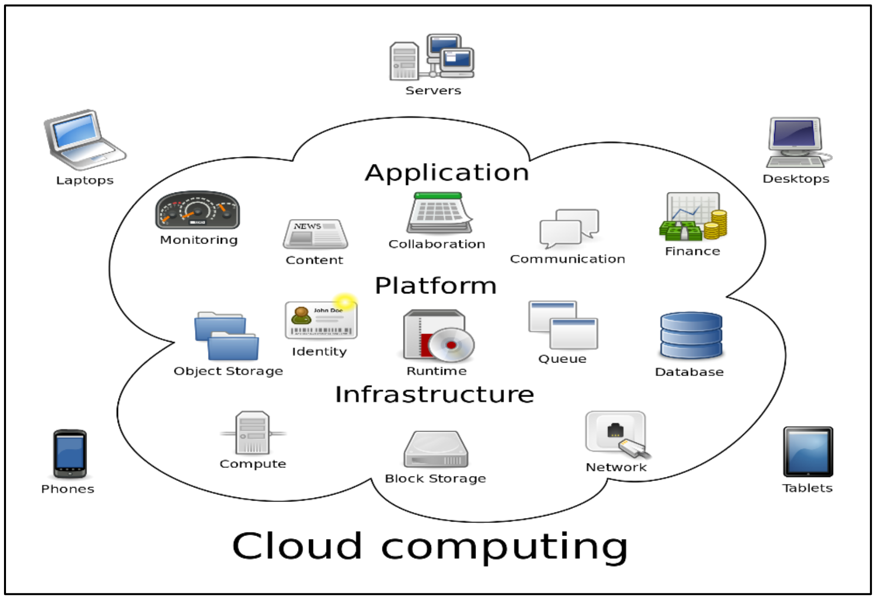


Figure 1.2. Cloud computing environment.
[Image by Sam Johnston is licensed under CC BY-SA 3.0]

- *Client/server computing*: computing provided by a combination of hardware resources available in a computing center (servers) as well as local computing (client). These systems usually run on commercial hardware and use commercial operating systems.
- *Desktop/laptop/computing*: computing that is carried out on a single system, often by interacting with data sources across a network. These systems usually run on commercial hardware and use commercial operating systems.
- *Mobile computing:* computing that is carried out on a mobile device, usually connected to the network via wireless communications. These systems usually run on commercial operating systems using commodity chipsets.
- *Embedded computing*: computing that is tied to a physical, often-customized hardware platform and that has special features that require careful integration between software and hardware (see figure 1.3).

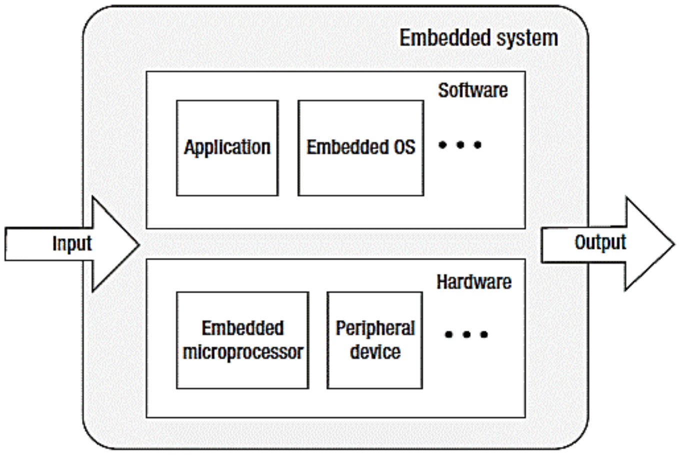


Figure 1.3. Embedded system architecture.
[Image from Ebrary.net]

A single software system may have multiple components or functions that span several of these definitions, and components of an integrated system likely have elements that do the same. The key point is that each type of software system has different requirements in terms of how quickly it can/should be updated, the level of information assurance required, and the organizations that will participate in development, testing, customization, and use of the software. Different statutes, regulations, and processes may be required for different types of software (and these would differ greatly from those used for hardware).

Having defined systems that deliver effects and the kinds of computing platforms on which software is hosted, we now distinguish between four primary types of software. We use these terms throughout the rest of the report to differentiate the acquisition and deployment approaches needed for different types of software:

- **Type A (Commercial Off-the-Shelf [COTS] applications):** The first class of software consists of applications that are available from commercial suppliers. Business processes, financial management, HR, software development, collaboration tools, accounting software, and other "enterprise" applications in DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Unmodified commercial software should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, the Department should modify its processes, not the software.

- **Type B (Customized Software):** The second class of software constitutes those applications that consist of commercially available software that is customized for DoD-specific usage. Customization can include the use of configuration files, parameter values, or scripted functions tailored for DoD missions. These applications generally require (ongoing) configuration by DoD personnel, contractors, or vendors.

- **Type C (COTS Hardware/Operating Systems):** The third class of software applications is those that are highly specialized for DoD operations but run on commercial hardware and standard operating systems (e.g., Linux or Windows). These applications will generally be able to take advantage of commercial processes for software development and deployment, including the use of open source code and tools. This class of software includes applications written by DoD personnel as well as those that are developed by contractors.

- **Type D (Custom Software/Hardware):** This class of software focuses on applications involving real-time, mission-critical, embedded software whose design is highly coupled to its customized hardware. Examples include primary avionics or engine control, or target tracking in shipboard radar systems. Requirements such as safety, target discrimination, and fundamental timing considerations demand that extensive formal analysis, test, validation, and verification activities be carried out in virtual and "iron bird" environments before deployment to active systems. These considerations also warrant care in the way application programming interfaces (APIs) are potentially presented to third parties.

We note that these classes of software are closely related to those described in the 1987 Defense Science Board (DSB) study on military software, which categorized software as "standard" (roughly capturing types A and B), "extended" (type C), "embedded" (type D), and "advanced" (which the study categorized as "advanced and exploratory systems," which are not so relevant here).

1.3 What Kind of Software Practices Will We Have to Enable?

The competitor that can realize software-defined military capability the fastest is at an advantage in future conflicts. We must shorten our development cycles from years to months so that we can react and respond within the observe–orient–decide–act (OODA) loop of the threats we face. Agile methodologies such as DevSecOps enable this rapid cycle approach (see "Detecting Agile BS" in Appendix E for more information about agile methodologies), and in addition to development we will need to test and validate software in real time as part of the integrated approach that DevSecOps demands. Quality assurance must be a continuous and fully integrated process throughout every phase of the software cycle. We need to build software pipelines that are able to develop and deploy software and provide updates as quickly as modern-day commercial companies so that we can respond to new threats (especially when the target will be our software). We must treat software as a continuous service rather than as block deliverables. It is important to have the agility in our procurement approach that will allow program managers to change priorities based on the needs and timing of the end users.

In the near future, DoD's acquisition and use of business systems should closely mirror industry and the private sector. DoD should modify its processes to mimic industry's best practices rather than try to contract for and maintain customized software. Figure 1.4 illustrates how this looks at Facebook (see also Section 2.1 for examples of best practices in industry).

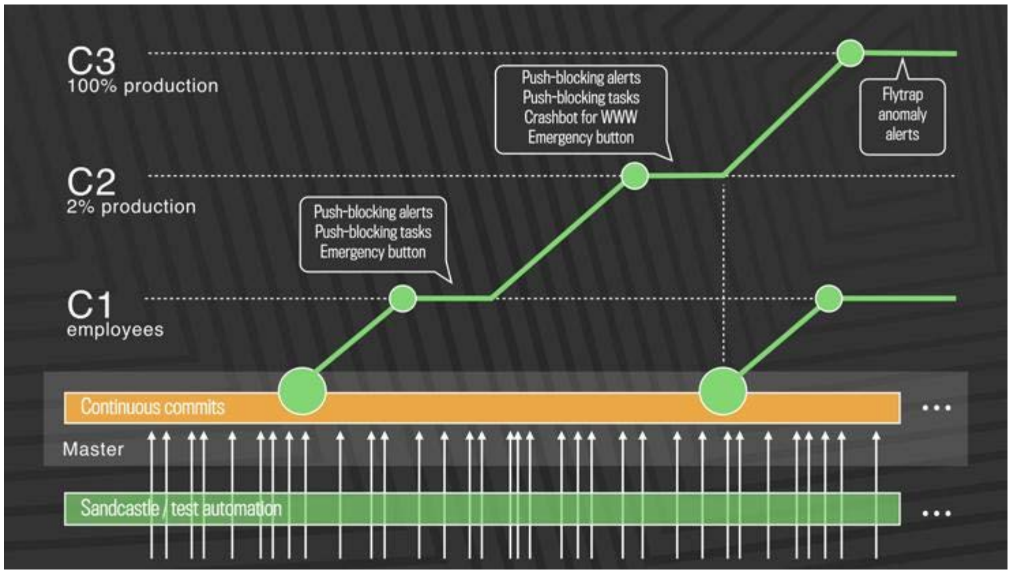


Figure 1.4. Facebook's continuous delivery process. Code updates that have passed a series of automated internal tests (bottom) land in the master development branch and are pushed out to employees (C1). In this stage, push-blocking alerts are generated if there are problems, and an emergency stop button keeps the release from going any further. If everything is OK, changes are pushed to 2 percent of production (C2), where signal and monitor alerts are again collected, especially for edge cases that testing or employee use may not have picked up. Finally, changes are rolled out to 100 percent of production (C3), where the "Flytrap" tool aggregates user reports and provides alerts on any anomalies. The cycle time between updates can be as short as a few hours. [Diagram and caption adapted from Facebook Engineering Blog, 31 Aug 2017 post on "Rapid release at massive scale"]

DoD should also adopt commercial logistics and mission planning software (COTS) wherever possible and reduce its reliance on government off-the-shelf (GOTS) solutions. Good logistics and mission software reduces process complexity, improves situational awareness, reduces costs, and simplifies planning while improving speed of delivery and streamlining performance.

For software that is closely tied to hardware, software-defined systems should be easier to develop, maintain, and upgrade than classic embedded systems. A well-designed system would allow new capabilities to be delivered directly to the edges of the network from the cloud in the same way new capabilities are delivered to consumer mobile devices.

DoD should manage software by measuring value delivered to the user rather than by monitoring compliance with requirements. Accountability should be based on delivering value to the user and solving user needs, not on complying with obsolete contracts or requirements documents.

Program managers must identify potential problems earlier (ideally, within months) and take corrective action quickly. Troubled programs must fail quickly, and the Department needs to learn from them. As we witnessed throughout our work on this study, many software programs are too big, are too complex, and take too long to deliver any value to users. Development must be staged and follow the best practice of smaller deliverables faster, with higher frequency of updates and new features. Initially, program development should focus on developing the “minimum viable product” (MVP) and getting it delivered to the customer more quickly than traditionally run programs. (The MVP for a software program represents the first point at which the code can start doing useful work and also at which feedback can be gathered that supports refinement of features.)

Software developers within the defense community need the same modern tools, systems, environments, and collaboration resources that commercial industry has adopted as standard. Without these, the Department undermines the effectiveness of its software developer base, and its ability to attract and retain our software human capital, both within DoD and among its suppliers. With the introduction of new technologies like ML and AI and the ever-increasing interdependence among networked heterogeneous systems, software complexity will continue to increase logarithmically. DoD needs to continuously invest in new development tools and environments including simulation environments, modeling, automated testing, and validation tools. DoD must invest in research and development (R&D) into new technologies and methodologies for software development to help the Department keep up with the ever-growing complexity of defense systems.

1.4 What Challenges Do We Face (and Consequences of Inaction)?

The world is changing. The United States used to be the dominant supplier of software and the world leader in software innovation. That is no longer the case. Due to the global digital revolution driven by the consumer and commercial markets, countries are building their own indigenous software capabilities and their own technology clusters. Countries like China are making huge investments in AI and cyber. China's 2030 plan envisions a $1 trillion AI industry in China.[2] China wants to become a cyber superpower and is investing in its capital markets, universities, research centers, defense industry, and commercial software companies to reach that goal.[3]

The potential long-term consequences of inaction are that our adversaries' software capabilities could catch and surpass those of the United States. If that happens, our adversaries would be able to develop new capabilities and potentially iterate faster than we can. They could respond to our defense systems faster than we can respond to theirs. If their algorithms and AI become superior to ours, they could hold a decisive advantage when any of our systems go up against any of theirs. And if their cyber capability becomes superior to ours, they could shut us down, cause chaos, continue to steal our secrets as they choose and without repercussions—especially if we could not attribute those attacks. Our adversaries' software capabilities are growing rapidly. If we do not keep pace, we could lose our defense technology advantage within a decade or much sooner.

[2] Vikram Barhat, "China Is Determined to Steal A.I. Crown from US and Nothing, Not Even a Trade War, Will Stop It," CNBC, May 4, 2018, https://www.cnbc.com/2018/05/04/china-aims-to-steal-us-a-i-crown-and-not-even-trade-war-will-stop-it.html.
[3] "China Is Seeking to Become a Cyber Superpower," *The Economist*, March 20, 2018, https://www.economist.com/graphic-detail/2018/03/20/china-is-seeking-to-become-a-cyber-superpower; and Rogier Creemers, Paul Triolo, and Graham Webster, "Translation: Xi Jinping's April 20 Speech at the National Cybersecurity and Informatization Work Conference," New America Blog Post, April 30, 2018, https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-xi-jinpings-april-20-speech-national-cybersecurity-and-informatization-work-conference/.

Chapter 2. What Does It Look Like to Do Software Right?

Deliver performance at the speed of relevance. Success no longer goes to the country that develops a new technology first, but rather to the one that better integrates it and adapts its way of fighting. Current processes are not responsive to need; the Department is over-optimized for exceptional performance at the expense of providing timely decisions, policies, and capabilities to the warfighter. Our response will be to prioritize speed of delivery, continuous adaptation, and frequent modular upgrades. We must not accept cumbersome approval chains, wasteful applications of resources in uncompetitive space, or overly risk-averse thinking that impedes change. Delivering performance means we will shed outdated management practices and structures while integrating insights from business innovation.

— U.S. Department of Defense, "Summary of the 2018 National Defense Strategy of the United States of America: Sharpening the American Military's Competitive Edge," (Washington, DC: U.S. Department of Defense, 2018), 10

In many cases, the software acquisition approaches and practices in place within DoD today look strange and perplexing to those familiar with commercial software practices. While the mission-, security-, and safety-critical nature of DoD's software in the context of embedded weapons will have an impact on practices, the extreme degree of divergence from contemporary commercial practice has been an area of our focus. Our case studies, site visits, and other study activities allowed a closer look into the reasons for divergence and whether the absence of many commercial best practices is justified.

2.1 How It Works in Industry (and Can/Should Work in DoD): DevSecOps

Figure 2.1 A former U.S. Marine Corps sergeant, now a Microsoft field engineer, works with an IT support specialist with the Navy as part of his job to travel to commercial companies and military bases across the country and train IT staff about a systems management product. [Photo by Sgt. Shellie Hall]

Modern software companies must develop and deliver software quickly and efficiently in order to survive in a hyper-competitive environment. While it is difficult to characterize the entire software sector, in this section we outline a set of practices—based on documented approaches in industry[4]—that are representative of commercial environments where the delivery of software capability determines the success or failure of the company. These practices generally hold true in other industries where companies have unexpectedly found themselves in the software business due to an increasing reliance on software to provide their key offerings, such as automotive, banking, healthcare, and many others. In any

[4] Fergus Henderson, "Software Engineering at Google" (arXiv:1702.01715 [cs.SE], January 31, 2017).

environment, software engineering practices must be matched with the recruitment and retention of talented software expertise. These practices must be honed over time and adapted to lessons learned.

At a high level, DoD must move from waterfall and spiral development methods to more modern software development practices such as Agile, DevOps, and DevSecOps. "DevOps" represents the integration of software development and software operations, along with the tools and culture that support rapid prototyping and deployment, early engagement with the end user, automation and monitoring of software, and psychological safety (e.g., blameless reviews). "DevSecOps" (as depicted in figure 2.2) adds the integration of security at all stages of development and deployment, which is essential for DoD applications. DoD should adopt these techniques, with appropriate tuning of approaches used by the Agile/DevSecOps community for mission-critical, national security applications. DoD should use open source software when possible to speed development and deployment and leverage the work of others.

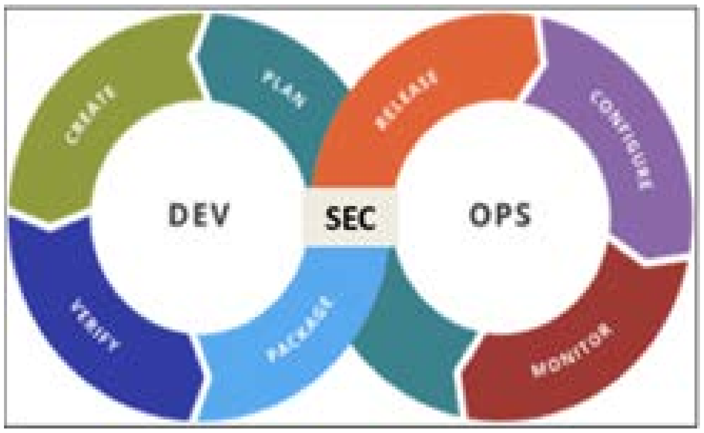


Figure 2.2. Continuous integration of development, security, and deployment (DevSecOps). [Adapted from an image by Kharnagy, licensed under CC BY-SA 4.0]

Generally, successful software companies have developed best practices in three categories:

Software development. These are software engineering practices that include source code management, software build, code review, testing, bug tracking, release, launch, and postmortems. Key best practices applicable to DoD software programs include the following:

- All source code is maintained in a single repository that is available to all software engineers. There are control mechanisms to manage additions to the repository, but in some cases all engineers are culturally encouraged to fix problems, independent of program boundaries.
- Developers are strongly encouraged to avoid "forking" source code (creating independent development branches) and focus work on the main branch of the software development.
- Code review tools are reliable and easy to use. Changes to the main source code typically require review by at least one other engineer, and code review discussions are open and collaborative.
- Unit test is ubiquitous, fully automated, and integrated into the software review process. Integration, regression, and load testing are also widely used, and these activities should be an integrated, automated part of daily workflow.
- Releases are frequent—often weekly. There is an incremental staging process over several days, particularly for high-traffic, high-reliability services.

- Postmortems are conducted after system outages. The focus of the postmortem is on how to avoid problems in the future and not on affixing blame.

Project management. Software projects must contribute to the overall aim of the business, and efforts must be aligned to that end goal.

- Individuals and teams set goals, usually quarterly and annually. Progress against those goals is tracked, reported, and shared across the organization. Goals are mechanisms to encourage high performance but can be decoupled from performance appraisal or compensation.
- The project approval process is organic. Significant latitude to initiate projects is given at all levels, with oversight responsibility given to managers and executives to allocate resources or cancel projects.

People management. Given the scarce number of skilled software engineers, successful software companies know how to encourage and reward good talent. Examples include the following:

- Engineering and management roles are clearly separated, with advancement paths for both. Technical career progression (e.g., for advanced and senior developers, fellows and senior fellows) parallels management career ladders; technical professionals receive similar compensation and accrue comparable respect within the organization. Similar distinctions are made between technical management and people management. The ratio of software engineers to product managers and program managers ranges from 4:1 to 30:1.
- Mobility throughout the organization is encouraged. This allows for the spread of technology, knowledge, and culture throughout the company.

In addition to these specific software development practices, another common approach to managing programs in industry is to move away from the specifications and requirements approach towards a feature management approach. This approach allows program managers to make agile decisions based on evolving needs and capabilities. Using a feature management approach, a program manager has a list of features and capabilities ranked by need, risk, cost, resources, and time. This list of capabilities is two to three times larger than what generally can be accomplished within a given time frame, a given budget, and a set of resources. Program managers make decisions about the feature mix, match investments to needs, and balance risk against performance. Capabilities are tested and delivered on a continuous basis, and maximum automation is leveraged for testing.

In industry, software programs initially start as an MVP. An MVP has just enough features to meet basic minimum functionality. It provides the foundational capabilities upon which improvements can be made. MVPs have significantly shorter development cycles than traditional waterfall approaches. The goal of MVPs is to get basic capabilities into users' hands for evaluation and feedback. Program managers use the evaluation and feedback results to rebalance and re-prioritize the software capability portfolio.

Portfolio success is measured based on performance of the *delivery* of capabilities as measured against user needs and strategic objectives within an investment cycle. Value is determined by output measurements rather than process measurements. Portfolio value is the aggregate of the total value of all of the capabilities delivered divided by total cost invested within a period of time. Blending higher risk/higher reward capabilities with lower risk/lower reward capabilities is the art of good portfolio management. Within a given period of time, program managers use diversification to spread risk and rewards. Good program managers identify troubled projects early and are encouraged either to quickly correct the problems or to quickly abandon failing efforts so that remaining resources can be husbanded and then reallocated to other priorities.

Software budgets are driven by time, talent, compute resources, development environment, and testing capabilities required to deliver capabilities. The capability and cost of talent vary greatly between software engineers, designers, programmers, and managers. The quality of engineering talent is the single largest variable that determines cost, risk, and duration of a software project. Good portfolio managers must take inventory of the range of software talent within a program and carefully allocate that talent across the portfolio of capabilities development.

2.2 Empowering the Workforce: Building Talent Inside and Out

One of the biggest barriers to realizing the software capabilities the Department so desperately needs is the way the Department manages the people necessary to build that capability. DoD cannot compete and dominate in defense software without a technical and design workforce within the Department that can both build software natively and effectively manage vendors to do the same, using the proven principles and practices described above. Some of the Department's human capital practices actively work against this critical goal.

If the Department wants to be good at software, it must be good at recruiting, retaining, leveraging, managing, and developing the people who make it. When we look at private-sector organizations and institutions that effectively use software to fulfill their mission, they

- understand the software professionals that they have, understand their workforce needs at a high level, and understand the gap between the two. (We say "at a high level" because we believe the gap is large enough that it is much more important to begin closing the gap than it is to measure the gap with too much precision.)
- have a strategy to recruit the people and skills they need to fulfill their mission, understanding what they uniquely have to offer in a competitive market.
- clearly understand the competencies required by software professionals in their organizations and the expectations of these professionals at each level in the organization.
- define career ladders for technical professionals that map software competencies and expectations from entry level to senior technical leadership and management.
- offer opportunities for learning and mentorship from more senior engineering and design leaders.
- count engineering and design leaders among their most senior leadership, with the ability to advocate across silos for the needs of the software and software acquisition workforce and support other senior leaders in understanding how to work with both.

- support a cadre of leadership able and empowered to create a culture of software management and promote common approaches, practices, platforms, and tools, while retaining the ability to use judgement about when to deviate from those common approaches and tools.
- reward software professionals based on merit and demonstrated contribution rather than time in grade.

Unfortunately, these are not the common descriptors for the software workforce practices in today's DoD.

DoD has long recognized that medicine and law require specialized skills, continuing education, and support and made it not only possible but desirable and rewarding to have a career as a doctor or lawyer in the armed forces. In contrast, software developers, designers, and managers in the Services must practice their skills intermittently and often without support as they endure frequent rotations into other roles. DoD does not expect a trained physician to constantly rotate into deployments focused on aviation maintenance, nor does it interrupt the training of a lawyer to teach him or her HR skills. Who would be comfortable being treated by a physician who worked in an institution that lacked common standards of care and provided no continuing education? And though software is often a matter of life and death, DoD's current human capital practices include all of these counterproductive features.

Figure 2.3. Airmen participate in Kessel Run's pair programming. [U.S. Air Force photo by Rick Berry]

The process to retool human capital practices to meet the challenge of software competency in DoD must start with the people the Department already has who have software skills or who are interested in acquiring them. Unlike medicine, software skills can be acquired through self-directed and even informal training resources such as on-demand, online webinars and coding boot camps, etc., and the Department has military and civilian individuals who have taken it upon themselves to gain technical skills outside of or in addition to formal DoD training. This kind of initiative and aptitude, especially when it results in real contribution to the mission, should be rewarded with appropriate opportunities for career advancement in this highly sought-after specialty. As we have witnessed during site visits for this study, there are also many individuals with more formally recognized software skills who are working with determination and even courage to try to deliver great software in service of the mission, but whose efforts to practice modern software techniques are poorly supported, and often actively blocked. Changes to policy that make clear the Department's support for these practices will help, but they must be married with support for the individuals to stay and grow within their chosen field. DoD could leverage several possible human capital pathways:

- Core military occupational series (MOS) and civilian occupational series for software development that include subcategories to address the various duties found in modern software development (e.g., developers/engineers, product owners, and designers).
- A secondary specialty series/designator for military members for software development. Experts come from various backgrounds, and a special secondary designator or occupational series for Service Members would be invaluable to tapping into their expertise even if they are not part of the core "Information Technology" profession.
- A Special Experience Identifier or other Endorsement for military and civilian acquisition professionals that indicates they have the necessary experience and training to serve on a software acquisition team. This Identifier or Endorsement should be a requirement to lead an acquisition team for a software procurement. Furthermore, this Identifier or Endorsement needs to be expanded to the broader team working the software procurement to include legal counsel, contract specialists, and financial analysts.

2.3 Getting It Right: Better Oversight AND Superior National Security

Getting software right in the Department requires more than changing development practices; oversight (and budgeting and finance) must also change. Those responsible for oversight of DoD software projects will need to learn to ask different questions and require different kinds of information on different tempos, but their reward will be more clarity, greater satisfaction with military software investments, and, ultimately, stronger national security.

Rules of thumb for those in appropriations and oversight roles over DevSecOps projects include the following:

Expect value to the user earlier. Oversight of monolithic, waterfall projects has generally focused on whether the team hit pre-determined milestones that may or may not represent actual value or even working code, and on figuring out what to do when they do not. When evaluating and appropriating funds to DevSecOps projects, it is more suitable to judge the project on the speed by which it delivers working code and actual value to users. In a waterfall project, changes to the plan generally reflect the team falling behind and are a cause for concern. In a project that is agile and takes advantage of the other approaches this study recommends (including software reuse), the plan is intended to be flexible because the team should be learning what works as they code and test.

Ask for meaningful metrics. Successful projects will develop metrics that measure value to the user, which involves close, ongoing communication with users. Source lines of code (SLOC) is not a measure of value and should not be used to evaluate projects in any case, as its use creates perverse incentives.

Assign a leader and hold him or her accountable. Part of the role of oversight is to ensure that there is a single leader who is qualified to lead in a DevSecOps framework and has the authority and responsibility to make the decisions necessary for the project to succeed. That person should have the authority to assign tasks and work elements; make business, product, and technical decisions; and manage the feature and bug backlogs. This person is ultimately responsible for how well the software meets the needs of its users, which is how the project should be evaluated.

Clarity and quality of leadership has long been tied to successful defense programs. Consider Kelly Johnson with the U-2, F-104, and SR-71. Paul Kaminski with stealth technology. Admiral Hyman Rickover with the nuclear Navy. Harry Hillaker with the F-16; and Bennie Schriever with the intercontinental ballistic missile. The list goes on. The United States Digital Service recognized this with Play 6 of the *Digital Services Playbook*—Assign One Leader and Hold That Person Accountable.[5] DoD would do well to remember this part of its history and work this practice into its oversight plan.

Speed increases security. Conventional wisdom in DoD says that programs must move slowly because moving quickly would threaten security. Often, the opposite is true. As we have learned from the cyber world, when we are facing active threats, our ability to achieve faster detection, response, and mitigation reduces the consequences of an attack or breach. In the digital domain, where attacks can be launched at machine speeds, where AI and ML can probe and exploit vulnerabilities in near real time, our current ability to detect, respond, and mitigate against digital threat leaves our systems completely vulnerable to our adversaries.

> The Department of Defense (DoD) faces mounting challenges in protecting its weapon systems from increasingly sophisticated cyber threats. This state is due to the computerized nature of weapon systems; DoD's late start in prioritizing weapon systems cybersecurity; and DoD's nascent understanding of how to develop more secure weapon systems. DoD weapon systems are more software dependent and more networked than ever before…. Potential adversaries have developed advanced cyber-espionage and cyber-attack capabilities that target DoD systems. (U.S. Government Accountability Office, Weapon Systems Cybersecurity: DoD Just Beginning to Grapple with Scale of Vulnerabilities [Washington, DC: U.S. Government Accountability Office, Oct 9, 2018], 2)

DoD must operate within its adversaries' digital OODA loop. Much like today's consumer electronic companies, the Department needs the ability to identify and mitigate evolving software and digital threats and to push continuous updates to fielded systems in near-real time.

DoD must be able to deploy software faster without sacrificing its abilities to test and validate software. To accomplish this, the Department needs to reimagine the software development cycle as a continuous flow rather than discrete software block upgrades. It should not only modernize to use a DevSecOps approach to software development but should also modernize its entire suite of development and testing tools and environments. DoD needs to be able to instrument its fielded systems so that we can build accurate synthetic models that can be used in development and test. The Department needs to be able to patch, update, enhance, and add new capabilities faster than our adversaries' abilities to exploit vulnerabilities.

Colors of money doom software projects. The foundational reasons for specific Congressional guidance on how money is to be spent make sense. But because software is in continuous

[5] "Digital Services Playbook," U.S. Digital Service, https://playbook.cio.gov/#plays_index_anchor.

development (it is never “done”—see Windows, for example), colors of money tend to doom programs. We need to create pathways for “bleaching” funds to smooth this process for long-term programs.

Do not pay for the factory every time you need a car. Appropriators must realize that DoD desperately needs common infrastructure if it is to increase the speed and quality of the software it produces. Today, it is as if the Department were buying cars but paying for the entire factory to build each car separately. Appropriators should fund the smart development of common infrastructure and reward its use in individual programs and projects. Evaluators should be wary of programs and projects that fail to articulate how they are taking advantage of common infrastructure and reusable components.

Standard is better than custom. In the same vein as the above, appropriators and evaluators should understand the benefits of using standards from the software development industry. Standards enable quality, speed, adoption, cost control, sustainability, and interoperability.

Technical debt is normal, and it is worth investing to pay it down. “Technical debt” refers to the cost incurred by implementing a software solution that is expedient rather than choosing a better approach that would take longer. Appropriators and evaluators should understandably expect to see progress in terms of features on a regular basis. The exceptions are when software teams must pay down technical debt or refactor code for greater performance. (This often results in fewer lines of code but higher performance, which is why it is a mistake to judge a software project based on the number of lines of code.) These periodic investments are to be expected on a DevSecOps project and are necessary to ensure the overall quality and stability of the project.

Use data as a compass, not a grade. Too often, evaluators and appropriators receive data about a program that suggests it is failing, but by the time they receive it, there is not much to be done about it. Data is collected manually, then processed and presented, and by the time it is being discussed, it is out of date. Mostly what happens at this point is that the project is given a poor grade, which makes the teams increasingly risk averse and demoralized. Instead, projects should be instrumented—equipped with built-in ways of seeing how and where they are going—so that the data is available both to the teams and to evaluators in time to make adjustments. In this model, the data is more like a compass, helping all parties make small corrections quickly to avoid the poor grade. An effective oversight function will help steer projects and hold them accountable, rather than punish poor performance.

2.4 Eye on the Prize: What Is the R&D Strategy for Our Investment?

The nature of software development may radically change in the near future. It is essential that the DoD adequately fund R&D programs to advance the fields of computer science, including computer programming, AI and ML, autonomy, quantum computing, networks and complex systems, man–machine interfaces, and cybersecurity.

Today, computers are controlled by programs that are comprised of sets of instructions and rules written by human programmers. AI and ML change how humans teach computers. Instead of providing computers with programmed instructions, humans will train or supervise the learning algorithm being executed on the computer. Training is inherently different than programming. Data becomes more important than code. Training errors are very different than programming errors. Hacking AI is very different than hacking code. The use of synthetic environments and “digital twins” (simulation-based emulators of physical components) may also become increasingly important tools to train a computer. The impact of AI and ML on software development will be profound and necessitates entirely new approaches and methods of developing software.

New computing technologies are also on the horizon. Experts may agree that we are many years away from developing a universal quantum computer (UQC), a generally programmable computer combining both classical and quantum computing elements. Nevertheless, the United States cannot afford to come in second in the race to develop the first UQC. The challenge is not only confined to development of the UQC hardware, but includes developing quantum computing programming languages and software. We also need to continue to invest in new quantum-resistant technologies such as cryptography and algorithms and apply those technologies as soon as possible to protect today’s data and information from tomorrow's UQC attacks.

The field of computer science continues to advance with the discovery and development of new computer architectures and designs. We have already seen the impact of new architectures such as cloud computing, GPUs (graphics processing units), low-power electronics, and Internet of Things (IoT) on computing. New architectures are being studied and developed by both industry and academia. DoD should not only continue to invest in the development of new architectures but also to invest in new methods for quicker adoption of these technologies.

Given today's challenge of cybersecurity and software assurance, R&D must continue developing more trusted computing to thwart future cyber attacks and creating abilities to execute software with assurance on untrusted networks and hardware.

DoD should invest in new approaches to software development (beyond Agile), including the use of computer-assisted programming and project management. While agile development is currently a best practice in industry, managing the software cycle is still more art form than science. New analytical approaches and next-generation management tools could significantly improve software performance and schedule predictability. The Department should fund ongoing research as well as support academic, commercial, and development community efforts to innovate the software process.

Chapter 3. Been There, ~~Done~~ Said That: Why Hasn't This Already Happened?

Probably the most dangerous phrase you could ever use in any computer installation is that dreadful one: "but we've always done it that way." That's a forbidden phrase in my office.

— Rear Admiral Grace Hopper (1906-1992), computer programmer, presentation at MIT Lincoln Laboratory on 25 April 1985, 23m41s

DoD and Congress have a rich history of asking experts to assess the state of DoD software capabilities and recommend how to improve them. A DoD joint task force chaired by Duffel in 1982 started its report by saying,

> Computer software has become an important component of modern weapon systems. It integrates and controls many of the hardware components and provides much of the functional capability of a weapon system. Software has been elevated to this prominent role because of its flexibility to change and relatively low replication cost when compared to hardware. It is the preferred means of adding capability to weapon systems and of reacting quickly to new enemy threats. (Report of the DoD Joint Service Task Force on Software Problems, 1982)

Indeed, this largely echoes our own views, although the scope of software has now moved well beyond weapon systems, the importance of software has increased even further, and the rate of change for software is many orders of magnitude faster, at least in the commercial world.

Five years later, a task force chaired by Fred Brooks began its executive summary as follows:

> Many previous studies have provided an abundance of valid conclusions and detailed recommendations. Most remain unimplemented. … [T]he Task Force is convinced that today's major problems with military software development are not technical problems, but management problems. (Report of the Task Force on Military Software, Defense Science Board, 1987)

This particular assessment, from over 30 years ago, referenced over 30 previous studies and is largely aligned with the assessments of more recent studies, including this one.

And finally, in its 2000 study on DoD software, Defense Science Board (DSB) Chair Craig Fields commented that,

> Numerous prior studies contain valid recommendations that could significantly and positively impact DoD software development programs. However the majority of these recommendations have not been implemented. Every effort should be made to understand the inhibitors that prevented previous recommendations. (Defense Science Board Task Force on Defense Software, 2000)

So to a large extent the problem is not that we do not know what to do, but that we simply are not doing it. In this chapter we briefly summarize some of the many reports that have come before ours and attempt to provide some understanding of why the current state of affairs in defense software is still so problematic. Using these insights, we attempt to provide some level of confidence that our recommendations might be handled differently (remembering that "hope is not a strategy").

3.1 37 Years of Prior Reports on DoD Software

The following table lists previous reports focused on improving software acquisition and practices within DoD.

Date	Org	Short title / Summary of contents
Jul'82	DoD	Joint Service Task Force on Software Problems 37 pp + 192 pp Supporting Information (SI); 4 major recommendations The opportunities and problems posed by computer software embedded in DoD weapon systems were investigated by a joint Service task force. The task force members with software experience combined existing studies with the observations of DoD project managers. The task force concluded that software represents an important opportunity in regard to the military mission. Further, it was concluded that technological excellence in software is an important factor in maintaining U.S. military superiority, but that many problems facing DoD in software endangers this superiority.
Sep'87	DSB	Task Force on Military Software 41 pp + 36 pp SI; 38 recommendations The task force reviewed current DoD initiatives in software technology and methodology, including the Ada effort, the STARS program, DARPA's Strategic Computing Initiative, the Software Engineering Institute (SEI), and a planned program in the Strategic Defense Initiative. The five initiatives were found to be uncoordinated, and the task force recommended that the Undersecretary of Defense (Acquisition) establish a formal program coordination mechanism for them. In spite of the substantial technical development needed in requirements setting, metrics and measures, tools, etc., the Task Force was convinced that the major problems with military software development were not technical problems, but management problems. The report called for no new initiatives in the development of the technology, some modest shift of focus in the technology efforts underway, but major re-examination and change of attitudes, policies, and practices concerning software acquisition.
Dec'00	DSB	Task Force on Defense Software 36 pp + 10 pp SI; 6 major recommendations The Task Force determined that the majority of problems associated with DoD software development programs are a result of undisciplined execution. Accordingly the Task Force's recommendations emphasized a back-to-the-basics approach. The Task Force also noted that numerous prior studies contain valid recommendations that could significantly and positively impact DoD software development programs. The fact that the majority of these recommendations have not been implemented should lead to efforts designed to understand the inhibitors preventing these recommendations from being enacted.
2004	RAND	Attracting the Best: How the Military Competes for Information Technology Personnel 149 pp; no explicit recommendations Burgeoning private-sector demand for IT workers, escalating private-sector pay in IT, growing military dependence on IT, and faltering military recruiting all led to a concern that military capability was vulnerable to a large shortfall in IT personnel. This report examined the supply of IT personnel compared to the military's projected future manpower requirements. It concluded that IT training and experience, augmented by enlistment bonuses and educational benefits as needed, seemed sufficient to ensure an adequate flow of new recruits into IT. However, sharp increases in military IT requirements had the potential to create difficulties.
Feb'08	NCMA	Generational Inertia: An Impediment to Innovation? 7 pp; no explicit recommendations This article cites data to the effect that approximately 50 percent of the acquisition workforce is within 5 years of retirement. Rather than being a problem, the article feels that retirement of senior contracting specialists could effectively lead to acquisition reform: "Senior contracting specialists' resistance to change and indifference to professional development is the elephant

		in the room that acquisition reformers are unwilling to acknowledge."
Mar'09	DSB	Task Force on Department of Defense Policies and Procedures for the Acquisition of Information Technology 68 pp + 2 pp dissent + 15 pp SI; 4 major recommendations with 13 subrecommendations The primary conclusion of the task force is that the conventional DoD acquisition process is too long and too cumbersome to fit the needs of the many IT systems that require continuous changes and upgrades. The task force recommended a unique acquisition system for information technology.
2010a	NRC	Achieving Effective Acquisition of Information Technology in the Department of Defense 164 pp + 16 major recommendations This study board was asked to assess the efficacy of DoD's acquisition and test and evaluation (T&E) processes as applied to IT. The study concluded that DoD is hampered by "a culture and acquisition-related practices that favor large programs, high-level oversight, and a very deliberate, serial approach to development and testing (the waterfall model)." This was contrasted with commercial firms, which have adopted agile approaches that focus on delivering smaller increments rapidly and aggregating them over time to meet capability objectives. Other approaches that run counter to commercial, agile acquisition practices include "the DoD's process-bound, high-level oversight [that] seems to make demands that cause developers to focus more on process than on product, and end-user participation often is too little and too late."
2010b	NRC	Critical Code: Software Producibility for Defense 148 pp + 15 major recommendations This study was charged to examine the nature of the national investment in software research and ways to revitalize the knowledge base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs. The study notes the continued reliance by DoD on software capabilities in achieving its mission and notes that there are important areas where DoD must push the envelope beyond mainstream capability. In other areas, however, DoD benefits by adjusting its practices to conform to government and industry conventions, enabling it to exploit a broader array of more mature market offerings.
Jul'16	CRS	The Department of Defense Acquisition Workforce: Background, Analysis, and Questions for Congress 14 pp; no explicit recommendations The increase in the size of the acquisition workforce has not kept pace with increased acquisition spending, which has signified an increase not only in the workload but also in the complexity of contracting work. This report summarized four Congressional efforts aimed at enhancing the training, recruitment, and retention of acquisition personnel.
Dec'16	CNA	Independent Study of Implementation of Defense Acquisition Workforce Improvement Efforts 147 pp + 30 pp SI; 21 major recommendations This report examines the strategic planning of the Department of Defense regarding the acquisition workforce (AWF). The study found significant improvements in several areas that "not only reversed the decline in AWF capacity from the 1990s, but also reshaped the AWF by increasing the number of early and mid-career personnel."
Feb'17	SEI	DoD's Software Sustainment Study Phase I: DoD's Software Sustainment Ecosystem 101 pp; 5 major recommendations Since the time in the early 1980s when software began to be recognized as important to DoD, software sustainment has been considered a maintenance function. After almost four decades, DoD is also at a tipping point where it needs to deal with the reality that software sustainment is not about maintenance, but rather it is about continuous systems and software engineering for the life cycle to evolve the software product baseline. This report recommends

		changing that paradigm to enable the innovation needed to address a rapidly changing technology environment, specifically through investments in human capital, better performance measurement of software sustainment, and better visibility for the software portfolio.
Mar’17	BPC	Building a F.A.S.T. Force: A Flexible Personnel System for a Modern Military 82 pp + 15 pp SI; 4 major themes with 39 recommendations This study describes today’s DoD personnel system as out of step with contemporary needs and issues: “the current system is typically poorly coordinated, lacks accountability, is unable to quickly obtain specialized talent, and fosters a groupthink mentality within the force.” It concludes that an effective personnel system has to build a force that is adaptable to new threats as they arise and technically proficient (among other characteristics).
Feb’18	DSB	Design and Acquisition of Software for Defense Systems 28 pp + 22 pp SI; 7 (high-level) recommendations + ~32 subrecommendations The Task Force assessed best practices from commercial industry as well as successes within DoD. Commercial embrace of iterative development has benefited bottom lines and cost, schedule, and testing performance, while the Department and its defense industrial base partners are hampered by bureaucratic practices and an existing government-imposed reward system. The Task Force concluded that the Department needs to change its internal practices to encourage and incentivize new practices in its contractor base. The assessment of the Task Force is that the Department can leverage best practices of iterative development even in its mission-critical software systems.
2018	2016 NDAA	Section 809 Panel - Streamlining and Codifying Acquisition 1,275 pp; 93 recommendations The Section 809 Panel was established by Congress in the FY 2016 NDAA to address issues with the way DoD buys what it needs to equip its warfighters. The panel published an Interim Report and a three-volume Final Report, containing a total of 93 recommendations aimed at changing the overall structure and operations of defense acquisition both strategically and tactically. Some changes hold potential for immediate effect, such as those that remove unnecessary layers of approval in the many steps contracting officers and program managers must take and those that remove unnecessary and redundant reporting requirements. Other changes require a large shift in how the system operates, such as buying readily available products and services in a manner similar to the private sector and managing capabilities from a portfolio, rather than program, perspective.
Apr’19	DIB	Software Is Never Done; Refactoring the Acquisition Code for Competitive Advantage (this document) 78 pp + 207 pp SI; 4 main lines of effort, 10 primary and 0x10 additional recommendations In this report, we focus on three overarching themes: (1) speed and cycle time are the most important metrics for managing software; (2) software is made by people and for people, so digital talent matters; and (3) software is different than hardware (and not all software is the same). We provide a set of major recommendations that focus on four main lines of effort: (A) refactoring statutes, regulations, and processes specifically for software—including acquisition, development, assurance, deployment, and maintenance—to remove hardware-centric bottlenecks while providing more insight and better oversight; (B) creating and maintaining interoperable (cross-program/cross-Service) digital infrastructure to enable continuous and rapid deployment, scaling, testing, and optimization of software as an enduring capability; (C) creating new paths for digital talent and increasing the level of understanding of modern software within the acquisition workforce; and (D) changing the practice of how software is procured and developed by adopting modern software development approaches.

As the table shows, studies dating back to at least 1982 have identified software as a particular area of growing importance to DoD—and software acquisition as requiring improvement—and the frequency and urgency of such studies identifying software acquisition as a major issue requiring reform has increased markedly since 2010. Notable recent examples include the 2010 studies by the National Research Council on *Achieving Effective Acquisition of Information Technology in the Department of Defense* and *Critical Code: Software Producibility for Defense*, the 2017 study conducted by the Carnegie Mellon University Software Engineering Institute (SEI) on DoD's Software Sustainment Ecosystem, and the 2018 DSB study on *Design and Acquisition of Software for Defense Systems*.

The properties of software that contribute to its unique and growing importance to DoD are summarized in this quote from the 2010 *Critical Code* study:

> Software is uniquely unbounded and flexible, having relatively few intrinsic limits on the degree to which it can be scaled in complexity and capability. Software is an abstract and purely synthetic medium that, for the most part, lacks fundamental physical limits and natural constraints. For example, unlike physical hardware, software can be delivered and up-graded electronically and remotely, greatly facilitating rapid adaptation to changes in adversary threats, mission priorities, technology, and other aspects of the operating environment. The principal constraint is the human intellectual capacity to understand systems, to build tools to manage them, and to provide assurance—all at ever-greater levels of complexity. (*Critical Code: Software Producibility for Defense*, NRC, 2010)

Prior studies have observed that much of DoD software acquisition policy is systems- and hardware-oriented and largely does not take these unique properties into account.[6]

The lack of action on most of the software recommendations from these studies has also been a subject of perennial comment. The DSB's 2000 study noted this phenomenon:

> [Prior] studies contained 134 recommendations, of which only a very few have been implemented. Most all of the recommendations remain valid today and many could significantly and positively impact DoD software development capability. The DoD's failure to implement these recommendations is most disturbing and is perhaps the most relevant finding of the Task Force. Clearly, there are inhibitors within the DoD to adopting the recommended changes. (Task Force on Defense Software, Defense Science Board, 2000)

The situation has not changed significantly since then despite additional studies and significant numbers of new recommendations. There is little to suggest that the inhibitors to good software practice have changed since 2000, and it is likely that the pace of technological change and addition of new capabilities provided by software have only increased since then.

Major categories of prior recommendations. The SWAP study team conducted a literature review of prior work on DoD software acquisition and extracted the specific recommendations that had been made, binning them according to major topics. The focus of the effort was on recent studies, with the bulk of the work since 2010, resulting in 139 recommendations that were extracted and categorized.

[6] For example, "DoD's Software Sustainment Study Phase I: DoD's Software Sustainment Ecosystem," SEI, 2017.

A few prevailing themes stood out from this body of work, representing issues that were commented upon in multiple studies:

- Contracts: contracts should be modular and flexible.
- Test and evaluation: test and evaluation (T&E) should be incorporated throughout the software process with close user engagement.
- Workforce: software acquisition requires specific skills and knowledge along with user interaction and senior leadership support.
- Requirements: requirements should be reasonable and prioritized; X (the focus of each report) should advocate for the need to move from compliance-based, overly prescriptive requirements to more iterative approaches.
- Acquisition strategy/oversight: DoD should encourage agencies to pursue business process innovations.
- Software process: the Department should adopt spiral/agile development approaches to reduce cost, risk, and time.

The three areas that were dealt with most often in the prior studies were acquisition oversight, contracting, and workforce. These three topics alone accounted for 60 percent of all of the recommendations we compiled. We summarize the major recurring prior recommendations in each of those areas as follows:

Recommendations from recent work in acquisition oversight:

- Ensure non-interruption of funding of programs that are successfully executing to objective (rather than budget), while insulating programs from unfunded mandates.
- Ensure that durations be reasonably short and meaningful and allow for discrete progress measurement.
- Design the overall technology maturity assessment strategy for the program or project.
- Encourage program managers to share bad news, and encourage collaboration and communication.
- Require program managers to stay with a project to its end.
- Empower program managers to make decisions on the direction of the program and to resolve problems and implement solutions.
- Follow an evolutionary path toward meeting mission needs rather than attempting to satisfy all needs in a single step.

Recommendations from recent work in contracting:

- Requests for proposals (RFPs) for acquisition programs entering risk reduction and full development should specify the basic elements of the software framework supporting the software factory, including code and document repositories, test infrastructure, software tools, check-in notes, code provenance, and reference and working documents informing development, test, and deployment.

- Establish a common list of source selection criteria for evaluating software factories for use throughout the Department.
- Contracting Officers (KOs) must function as strategic partners tightly integrated into the program office, rather than operate as a separate organization that simply processes the contract paperwork.
- Develop and maintain core competencies in diverse acquisition approaches and increase the use of venture capital–type acquisitions such as Small Business Innovative Research (SBIR), Advanced Concept Technology Development (ACTD), and Other Transaction Authority (OTA) as mechanisms to draw in nontraditional companies.

Recommendations from recent work on workforce issues:

- Service acquisition commands need to develop workforce competency and a deep familiarity with current software development techniques.
- The different acquisition phases require different types of leaders. The early phases call for visionary innovators who can explore the full opportunity space and engage in intuitive decision making. The development and production phases demand a more pragmatic orchestrator to execute the designs and strategies via collaboration and consensus decisions.
- U.S. Special Operations Command (USSOCOM) must develop a unique organizational culture that possesses the attributes of responsiveness, innovation, and problem solving necessary to convert strategic disadvantage into strategic advantage.
- Encourage employees to study statutes and regulations and explore innovative and alternative approaches that meet the statutory and regulatory intent.
- Rapid acquisition succeeds when senior leaders are involved in ensuring that programs are able to overcome the inevitable hurdles that arise during acquisition, and empower those responsible with achieving the right outcome with the authority to get the job done while minimizing the layers in between.

To help illustrate the continuity of the history of these issues and the lack of progress despite consistent, repeated similar findings, we consider the case of recommendations related to software capabilities of the acquisition workforce (areas where we are also recommending change).

Calls to improve DoD’s ability to include software expertise in its workforce have a long history. DoD studies dating back to 1982 have raised concerns about the technical competencies and size of DoD’s software workforce [DSB’82, DSB’87]. In 1993, the DoD Acquisition Management Board identified a need to review the DoD’s software acquisition management education and training curricula. This study concluded that no existing DoD workforce functional management group was responsible for the software competencies needed in the workforce and that software acquisition competencies were needed in many different acquisition career fields. However, the Board asserted that no new career field was needed for Software Acquisition Managers.

In 2001, the same concerns regarding the software competencies of the DoD acquisition workforce once again surfaced. The DoD Software Intensive Systems Group conducted a software education and training survey of the acquisition workforce.[7] This survey demonstrated that less than 20 percent of the ACAT program staff had taken the basic Software Acquisition Management course (SAM 101) and that less than 20 percent of the ACAT program staff had degrees in computer science, software engineering, or information technology. The specific recommendations from this analysis included (1) instituting mandatory software-intensive systems training for the workforce; (2) developing a graduate-level program for software systems development and acquisition; and (3) requiring ACAT 1 programs to identify a chief software/ systems architect.

A year later, Congress mandated that the Secretary of each military department establish a program to improve the software acquisition processes of that military department.[8] Subsequently each Service established a strategic software improvement program (Army 2002, Air Force 2004, and Navy 2006). These Service initiatives have continued at some level. However, with the sunsetting of the Software Intensive Systems Group at the Office of the Secretary of Defense (OSD) level, the enterprise focus on software waned. During this same period, the Navy started the Software Process Improvement Initiative (SPII), which identified issues preventing software-intensive projects from meeting schedule, cost, and performance goals. This initiative highlighted the lack of adequately educated and trained software acquisition professionals and systems engineers.

In 2007, OSD issued guidance to create the Software Acquisition Training and Education Working Group (SATEWG) with a charter to affirm required software competencies, identify gaps in Defense Acquisition Workforce Improvement Act (DAWIA) career fields, and develop a plan to address those gaps. This group was composed of representatives from the Services, OSD, and other organizations, including the SEI. The group developed a software competency framework that identified four key knowledge areas and 29 competencies that could inform the different acquisition workforce managers about the software competencies to be integrated into their existing career field competency models. There has been no follow-on effort to evaluate the progress of the SATEWG or its outcomes.

Today, in the absence of a DoD-wide approach to describing, managing, and setting goals against a common understanding of needed software skills, each Service (as well as each software sustainment organization) has evolved its own approach or model for identifying software competencies for its workforce.

This historical context highlights two key points. First, DoD has long recognized the challenges of addressing the technical competencies and size of the software workforce across the life cycle. However, there is limited evidence of the outcomes from these different efforts. Second, this history clearly indicates that acquiring software human capital and equipping that workforce with the necessary competencies are persistent and dynamic challenges that demand a continuous enterprise strategy.

[7] Dennis Goldenson, & Matthew Fisher, *Improving the Acquisition of Software Intensive Systems* (CMU/SEI-2000-TR-003), (Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5171.
8 Public Law 107-314, Section 804, 2 December 2002, https://www.govinfo.gov/content/pkg/PLAW-107publ314/html/PLAW-107publ314.htm.

3.2 Breaking the Spell: Why Nothing Happened Before, but Why This Time Could Be Different

Given the long and profound history of inaction on past studies, we have attempted to create our own "Theory of (Non)Change." Why does the Department struggle to step up to rational, generally agreed-upon change? We offer the following three drivers:

The (Patriotic and Dutifully) Frozen Middle. Our process in executing this study has been to talk to anyone and everyone we could within various departments of DoD and the Services, to gather as many different perspectives as possible on what is needed, and to find out what is working and what needs to be stomped upon. As with many change management opportunities, we find significant top-down support for what we are trying to do, especially from those who see the immediate need for more, better, faster mission capability and those at the command level who are directly frustrated by the current processes that are just not working. At the other end, we see digital natives demanding change but with limited power to make it happen—people who are fully enmeshed in how the tech world works, people who have all the expectations that have been created by their private-sector lifestyle and economy. And then we have *the middle,* who are dutifully following the rules and have been trained and had success defined for a different world. For *the middle,* new methodologies and approaches introduce unknown risks, while the old acquisition and development approaches built the world's best military. We question neither the integrity nor the patriotism of this group. They are simply not incentivized to the way we believe modern software should be acquired and implemented, and the enormous inertia they represent is a profound barrier to change.

Unrequited Congress. Congress is responsible for approving and overseeing DoD's development programs. While it is clear that Congress takes its oversight role seriously, it does so knowing that to have oversight requires something to oversee, and it understands its fundamental responsibility is to enable the Department to execute its mission. But oversight matters, and recommendations for change that do not also provide insight into how new ways of doing things will allow Congress to perform its role are a very tough sell. In addition, there is a sense of unrequited return from past changes and legislation such as Other Transaction Authorities (OTAs), pilot programs, and special hiring authorities. In many cases, Congress believes it has already provided the tools and flexibilities for which DoD has asked. It is perhaps unreasonable to expect a positive response to ask for more when current opportunities have not been fully exploited.

Optimized Acquisition (for Something Else!).

> *Knowing was a barrier which prevented learning. — Frank Herbert*

While some may (justifiably) argue that the current acquisition system is not optimized for anything, it is the product of decades of rules upon rules, designed to speak to each and every edge case that might crop up in the delivery of decades-long hardware systems, holds risk elimination at a premium, and has a vast cadre of dedicated practitioners exquisitely trained to prosper within that system. This is a massive barrier to change and informs our recommendations that argue for major new ways of acquiring software and not just attempt to re-optimize to a different local maximum.

What we are trying to do that we think is different. Given the long history of DoD and Congressional reports that make recommendations that are not implemented, why do we think that this report will be any different? Our approach has been to focus not on the report and its recommendations *per se*, but rather on the series of discussions around the ideas in this report and the people we have interacted with inside the Pentagon and at program site visits. The recommendations in this report thus serve primarily as documentation of a sequence of iterative conversations, and the real work of the study is the engagements before and after the report is released.

We also believe that there are some ideas in the report that, while articulated in many places in different ways, are emphasized differently here. In particular, a key point of focus in this report is the use of speed and cycle time as the key drivers for must change and the need to optimize statutes, regulations, and processes to allow management and oversight of software. We believe that optimizing for the speed at which software can be utilized for competitive advantage will create an acquisition system that is much better able to provide security, insight, and scale.

Finally, we have tried to make this report shorter and pithier than previous reports, so we hope people will read it. It also is staged so that each reader, with his or her specific levels of authority and responsibility, can navigate an efficient path to reaching his or her own conclusions on how best to support what is contained here.

3.3 Consequences of Inaction: Increasing Our Attack Surface and Shifting Risk to the Warfighter

So what happens if history does, in fact, repeat itself and we again fail to step up to the changes that have been so clearly articulated for so long? Certainly by continuing to follow acquisition processes designed to limit risk for the hardware age, we will not reduce risk but instead will simply transfer that risk to the worst possible place—the warfighter who most needs the tools in her arsenal to deliver the missions we ask her to perform. But in addition, as we have continually stressed throughout this study, there are several real differences in today’s world compared to the environment in which past efforts were made.

First, and most important, weapon systems, and the bulk of the operational structure on which DoD executes its mission, are now fundamentally software (or software-defined) systems, and as such, delays in implementing change amplify the capability gaps that slow, poor, or unsupportable software creates. Second, the astonishing growth of the tech sector has created a very different competitive environment for the talent most needed to meet DoD’s needs. Decades ago, DoD was the leading edge of the world’s coolest technology, and passionate, skilled software specialists jumped at the chance to be at that edge. That is simply not the case today, and while a commitment to national security is a strong motivator, if the changes recommended in this study are not implemented, the competitive war for talent, *within our country,* will be lost.

The modern software methodologies enumerated in this report—and the recommendations concerning culture, regulation and statute, and career trajectories that enable those methodologies—are the best path to providing secure, effective, and efficient software to users.

Cyber assurance, resilience, and relevance are all delivered much more effectively when done quickly and incrementally, using the tools and methods recommended in this study.

Finally we call attention back to Section 1.4 (What are the challenges that we face [and consequences of inaction]?). To summarize: “The long-term consequence of inaction is that our adversaries’ software capabilities can catch and surpass ours. … Our adversaries’ software capabilities are growing as ours are stagnating.”

Chapter 4. How Do We Get There from Here: Three Paths for Moving Forward

The history of technology is the story of man and tool-hand and mind-working together. If the hardware is faulty or if the software is deficient, the sounds that emerge will be discordant; but when man and machine work together, they can make some beautiful music.

— Melvin Kranzberg, Technology and History: Kranzberg's Laws, (Technology and Culture, 27[3]:1986), 558

The previous three chapters provided the rationale for why we need to *do* (not just say) something different about how DoD develops, procures, assures, deploys, and continuously improves software in support of defense systems. The private sector has figured out ways to use software to accelerate their businesses and DoD should accelerate its incorporation of those techniques to its own benefit, especially in ensuring that its warfighters have the tools they need in a timely fashion to execute their missions in today's hardware-enabled, software-defined environment. In this chapter, we lay out three different paths for moving forward, each under a different set of assumptions and objectives. A list of some representative, high-level steps is provided for each path, along with a short analysis of advantages and weaknesses.

4.1 Path 1: Make the Best of What We've Got

Congress has provided DoD with substantial authority and flexibility to implement the mission of the Department. Although difficult and often inefficient, it is possible to implement the recommendations outlined in this report making use of the existing authorities and, indeed, there are already examples of the types of activities that we envision taking place across OSD and the Services. In this section, we attempt to articulate a path that builds on these successes and does not require any change in the law nor major changes in regulatory structure. The primary steps required to implement this path should focus on changing the practices by which software is developed, procured, assured, and deployed as well as updating some of the regulations and processes to facilitate cultural and operational changes.

To embark on this first path, DoD should streamline its processes, allowing more rapid procurement, deployment, and updating of software. OSD and the Services should also work together to allow better cross-service and pre-certified Authorization to Operate (ATO), easier access to large-scale cloud computing, and use of modern toolchains that will benefit the entire software ecosystem. The acquisition workforce, both within OSD and the Services, should be provided with better training and insight on modern software development (one of the more frequent recommendations over the past 37 years) so that they can take advantage of the approaches that software allows that are different than hardware. Most importantly, government and industry must come together to implement a DevSecOps culture and approach to software, building on practices that are already known and used in industry.

The following list provides a summary of high-level steps that require changes to DoD culture and processes, but could be taken with no change in current law and relatively minor changes to existing regulations:

- Make use of existing authorities such as OTAs and mid-tier acquisition (Sec 804) to implement a DevSecOps approach to acquisition to the greatest extent possible under existing statutes, regulations, and processes.
- Require cost assessment and performance estimates for software programs (and software components of larger programs) to be based on metrics that track speed and cycle time, security, code quality, and useful capability delivered to end users.
- Create a mechanism for ATO reciprocity between Services and industrial base companies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapons) systems, and Services.
- Remove obstacles to DoD usage of cloud computing on commercial platforms, including Defense Information System Agency (DISA) cloud access point (CAP) limits, lack of ATO reciprocity, and access to modern software development tools.
- Expand the use of (specialized) training programs for chief information officers (CIOs), Service acquisition executives (SAEs), program executive officers (PEOs), and program managers (PMs) that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software.
- Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition.
- Require access to source code, software frameworks, and development toolchains, with appropriate intellectual property (IP) rights, for all DoD-specific code, enabling full security testing and rebuilding of binaries from source.
- Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code quality to assess, manage, and terminate software programs (and software components of hardware programs).
- Shift the approach for acquisition (and development) of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success—or be terminated quickly.
- Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough.
- Shift from a list of requirements for software to a list of desired features and required interfaces/characteristics to avoid requirements creep or overly ambitious requirements.
- Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas.
- Invest in transition of emerging approaches from academia and industry to creating, analysis, verification, and testing of software into DoD practice (via pilots, field tests, and other mechanisms).

- Automatically collect all data from DoD weapon systems and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).
- Mandate a full program review within the first 6–12 months of development to determine if a program is on track, requires corrective action, or deserves cancellation.

This path has the advantage that the authorities required to undertake it are already in place and the expertise exists within the Department to begin moving forward. We believe that the there is strong support for these activities at the top and bottom of the system, and several groups (e.g., the Defense Digital Service [DDS], the Joint Improvised Threat Defeat Organization [JIDO], and Kessel Run) have demonstrated that the flexibilities exist within the current system to develop, procure, assure, deploy, and update software more quickly. The difficulty in this path is that it requires individuals to figure out how to go beyond the default approaches that are built into the current acquisition system. Current statutes, regulations, and processes are very complicated; there is a "culture of no" that must be overcome; and hence using the authorities that are available requires substantial time, effort, and risk (to one's career, if not successful). The risk in pursuing this path is that change occurs too slowly or not at scale, and we are left with old software that is vulnerable and cannot serve our needs. Our adversaries have the same opportunities that we do for taking advantage of software and may be able to move more quickly if the current system is left in place.

4.2 Path 2: Tune the Defense Acquisition System to Optimize for Software

While the first steps to refactoring the defense acquisition system can be taken without necessarily having to change regulations, the reality of the current situation is that Congress and DoD have created a massive "spaghetti code" of laws and regulations that are simply slowing things down. This might be OK for some types of long-development, long-duration hardware, but as we have articulated in the previous three chapters it is definitely not OK for (most types of) software.

This path takes a more active approach to modifying the acquisition system for software by identifying those statutes, regulations, and processes that are creating the worst bottlenecks and modifying them to allow for faster delivery of software to the field. We see this path as one of removing old pieces of code (statutory, regulatory, or process) that are no longer needed or that should not be applied to software, as well as increasing the expertise in how modern software development works so that software programs (and software-centric elements of larger programs) can be optimized for speed and cycle time.

The following list provides a set of high-level steps that require some additional changes to DoD culture and process, but also modest changes in current law and existing regulations. These steps build on the steps listed in path 1 above, although in some cases they can solve the problems that the previous actions were trying to work around.

- Refactor and simplify Title 10 and the defense acquisition system to remove all statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule.

- Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification.
- Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers that force categorization into hardware-oriented regulations and processes.
- Replace the Joint Capabilities Integration and Development System (JCIDS), the Planning, Programming, Budgeting and Execution (PPB&E) process, and the Defense Federal Acquisition Regulation Supplement (DFARS) with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to decide on allocation priorities.
- Create, implement, support, and require a fully automatable approach to T&E, including security, that allows high-confidence distribution of software to the field on an iterative basis (with frequency dependent on type of software, but targeting cycle times measured in weeks).
- Prioritize secure, iterative, collaborative development for selection and execution of all new software programs (and software components of hardware programs) (see DIB's Detecting Agile BS as an initial view of how to evaluate capability).
- For any software developed for DoD, require that software development be separated from hardware in a manner that allows new entrants to bid for software elements of the program on the basis of demonstrated capability.
- Shift from certification of executables, to certification of code, to certification of the development, integration, and deployment toolchain, with the goal of enabling rapid fielding of mission-critical code at high levels of information assurance.
- Require CIOs, SAEs, PEOs, PMs, and any other acquisition roles involving software development as part of the program to have prior experience in software development.
- Restructure the approach to recruiting software developers to assume that the average tenure of a talented engineer will be 2–4 years, and make better use of highly qualified experts (HQEs), intergovernmental personnel act employees (IPAs), reservists, and enlisted personnel to provide organic software development capability.
- Establish a Combat Digital Service (CDS) unit within each Combatant Command (COCOM) consisting of software development talent that can be used to manage Command-specific IT assets, at the discretion of the combatant commander. DDS, operating at the OSD level, is a good model for what a CDS can do for each COCOM.

Pursuing this path will allow faster updates to software and will improve security and oversight (via increased insight). In many cases, the Department is already executing some of the actions required to enable this path. The weakness in this path is that software would generally use the same basic approach to acquisition as hardware, with various carve-outs and exceptions. This approach runs the risk that software programs still move too slowly due to the large number of people who have to say yes and the need to train a very large acquisition force to understand how software is different than hardware (and not all software is the same).

4.3 Path 3: A New Acquisition Pathway and Appropriations Category for Software to Force Change in the Middle

The final path is the most difficult and will require dozens of independent groups to agree on a common direction, approach, and set of actions. At the end of this path lies a new defense acquisition system that is optimized for software-centric systems instead of hardware-centric systems and that prioritizes security, speed, and cycle time over cost, schedule, and (rigid) requirements.

To undertake this path, Congress and OSD must write new statutes and regulations for software, providing increased (and automation-enabled) insight to reduce the risk of slow, costly, and overgrown programs and enabling rapid deployment and continuous improvement of software to the field. Laws will have to be changed, and management and oversight will have to be reinvented, focusing on different measures and a quicker cadence. OSD and the Services will need to create and maintain interoperable (cross-program/cross-Service) digital infrastructure that enables rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage it using modern development methods; and eliminate the existing hardware-centric regulations and other barriers for software (and software-intensive) programs. Finally, the Services will need to establish software development as a high-visibility, high-priority career track with specialized recruiting, education, promotion, organization, incentives, and salary.

The following list of high-level steps are required to pursue this path, builds on the steps listed in the previous paths:

- Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics.
- Create a new appropriations category that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment.
- Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors.
- Plan and fund computing hardware (of all types) as consumable resources, with continuous refresh and upgrades to the most recent, most secure operating system and platform components.
- Create software development groups in each Service consisting of military and/or civilian personnel who write code that is used in the field, and track individuals who serve in these groups for future DoD leadership roles.

This path attempts to solve the longstanding issues with software by creating an acquisition pathway and an appropriations category that are fine-tuned for software. It will require a very large effort to get the regulations, processes, and people in place that are required to execute it effectively, and there will be missteps along the way that generate controversy and unwanted publicity. In addition, it will likely be opposed by those currently in control of selling or making software for DoD, since it will require that they retool their business to a very new approach that is not well defined at the outset. But if successful, this path has the potential to enable DoD to develop, procure, assure, deploy, and continuously improve software at a pace that is relevant for modern missions and builds on the substantial success of the U.S. private sector.

Chapter 5. What Would the DIB Do: Recommendations for Congress and DoD

It takes a lot of hard work to make something simple, to truly understand the underlying challenges and come up with elegant solutions.

— Steve Jobs as quoted by Walter Isaacson, "How Steve Jobs' Love of Simplicity Fueled a Design Revolution," (Smithsonian Magazine, September 2012)

In this final chapter we lay out our recommendations for what Congress and DoD should do to implement the type of software acquisition and practices reform that we believe is needed for the future. Our recommendations are organized according to four lines of effort, each of which bring together different parts of the defense ecosystem as stakeholders:

A. Congress and OSD should refactor statutes, regulations, and processes for software
B. OSD and the Services should create and maintain cross-program/cross-Service digital infrastructure
C. The Services and OSD should create new paths for digital talent (especially *internal* talent)
D. DoD and industry must change the practice of how software is procured and developed

For each of these lines of effort, we have identified the 2–3 most important recommendations that we believe Congress and DoD should undertake. These "Top Ten" primary recommendations were chosen not because they solve the entire problem but because they will make the biggest difference; without them, substantial change is not likely. In addition, we have identified 16 additional recommendations for consideration once the execution of the first 10 recommendations is successfully underway. For each recommendation, a draft implementation plan is provided in Appendix A that gives a list of actions that can be used to implement the recommendation, as well as more detail on the rationale, supporting information, and similar recommendations from other studies. Potential legislative and regulatory language to implement selected recommendations is included in Appendix B. While we have tried hard to provide specific actions, owners, and target dates that will drive an implementation plan for each recommendation, we recognize that in the end, owners will be decided by the Department's response to our study and owners will use our actions as a starting point to their own implementation plans.

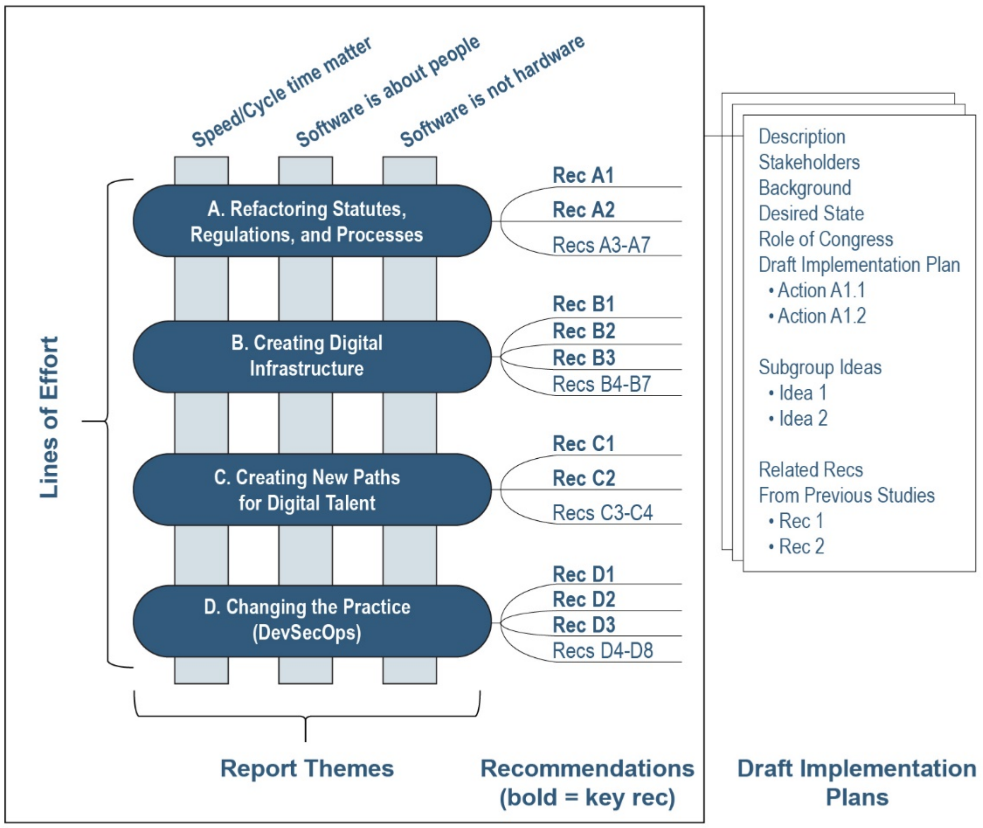


Figure 5.1 Recommendation structure. For each line of effort, a set of primary recommendations (bold) is provided, along with a set of additional recommendations for consideration. Each recommendation contains a draft implementation plan that includes background information on the rationale, vision, and stakeholders.

5.1 The Ten Most Important Things to Do (Starting Now!)

In this section we lay out what we believe are the most important steps for Congress and DoD to take to fully leverage the opportunities presented by software and the private sector's strength in modern development practices. Our commitment to these steps will directly impact the Department's ability to achieve the 2018 National Defense Strategy[9] goals of increased lethality, stronger alliances while positioning for new partnerships, and reformed business practices for better performance and affordability.

[9] U.S. Department of Defense, *Summary of the 2018 National Defense Strategy: Strengthening the American Military's Competitive Edge*, (Washington, DC: U.S. Department of Defense), https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf.

Line of Effort A. Congress and OSD should refactor statutes, regulations, and processes for software, providing increased insight to reduce the risk of slow, costly, and overgrown programs and enabling rapid deployment and continuous improvement of software to the field. Reinvent management and oversight, focusing on different measures and a quicker cadence.

Figure 5.2. The West Front of the U.S. Capitol. [Photo by Architect of the Capitol]

Recommendation A1. Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics

Current law, regulation, policy, and internal DoD processes make DevSecOps-based software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale DevSecOps software development practices to meet mission needs. The desired state is that programs have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilize IA-accredited commercial development tools.

Implementation of this recommendation could be accomplished by having USD(A&S), in coordination with USD(C) and Cost Assessment and Program Evaluation (CAPE), submit a legislative proposal using Sec 805 to propose new acquisition pathways for two or more classes of software (e.g., application, embedded), optimized for DevSecOps, for approval by the House and Senate Armed Services Committees. A draft of such language, in response to 2016 NDAA Section 805, is included in Appendix B. If approved, USD(A&S) could develop and issue a Directive-Type Memorandum (DTM) for new software acquisition pathways, and the SAEs could issue Service-level guidance for new acquisition pathways. USD(A&S), with SAEs, should select an initial set of programs that are using DevSecOps to convert to or utilize the new software acquisition pathways at the same time as developing and implementing training at Defense Acquisition University (DAU) on new software acquisition pathways for all acquisition communities (FM, Costing, PM, IT, SE, etc.). As the pathways become better understood, the DTM can be converted to a DoD Instruction (5000.SW?), incorporating lessons learned during initial program implementation.

This recommendation is supported by the ideas for change listed by the Acquisition & Strategy subgroup and is aligned with the recommendations of the 1987 and 2009 DSB studies.

Recommendation A2. Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment

Current law, regulation, and policy treat software acquisition as a series of discrete sequential steps; accounting guidance treats software as a depreciating asset. These processes are at odds with software being continuously updated to add new functionality and create significant delays in fielding user-needed capability. The desired state is the establishment of a new appropriation (major force program category) so that programs are better able to prioritize how effort is spent on new capabilities versus fixing bugs/vulnerabilities, improving existing capabilities, etc. Such prioritization can be made based on warfighter/user needs, changing mission profiles, and other external drivers, not constrained by available sources of funding.

Implementation of this recommendation could be accomplished by having USD(A&S) submit a legislative proposal to create a new appropriations category for software and software-intensive programs for approval by the House and Senate Armed Services Committees and funding by the House and Senate Appropriations Committees. A draft of such language, linked to the acquisition pathway described in Recommendation A1, is included in Appendix B. The DoD Comptroller, working with CAPE, would need to make necessary modifications in supporting PPB&E systems to allow use and tracking of the new software appropriation. USD(A&S), in coordination with the SAEs, should select the initial programs that will use the new software appropriation from among those that are currently using DevSecOps-compatible development approaches. Budget exhibits for the new software appropriation, replacing the current P-Forms and R-Forms, should be prepared by USD(A&S) working with USD(C), CAPE, and the Appropriations Committees, and those programs selected to use the new appropriation category should begin using the exhibits upon selection into the category (see Appendix C). Finally, the Federal Accounting Standards Advisory Board in coordination with USD(A&S) and USD(C) will need to change the audit treatment of software for this category to : (1) create a separate category for software instead of characterizing software as property, plant, and equipment; (2) establish a default setting that software is an expense, not an investment; and (3) ensure that “sustainment” is an integrated part of the software life cycle.

This recommendation builds on the recommendations in the DIB’s Ten Commandments of Software (at Appendix E) and our Visit Observations and Recommendations that budgets for software (and software-intensive) programs should support the full, iterative life cycle of the software. In addition, the Acquisition & Strategy, Appropriations, Contracting, and Sustainment & Modernization subgroups all had recommendations that support this approach. The basic approach advocated here was also articulated in the 1987 DSB task force on military software and Government Accountability Office (GAO) studies in 2015 and 2017, and is consistent with the Portfolio Management Framework Recommendations 41 and 42 of the Section 809 Panel.

Line of Effort B. OSD and the Services should create and maintain cross-program/ cross-Service digital infrastructure that enables rapid deployment, scaling, and optimization of software as an enduring capability, managed using modern development methods in place of existing (hardware-centric) regulations and providing more insight (and hence better oversight) for software-intensive programs.

Figure 5.3. Soldiers review the Army's Command Post Computing Environment, a software system that consolidates tools, programs, and tasks into an integrated, interoperable, and cybersecure computing infrastructure framework. [U.S. Army photo by Dan Lafontaine, PEO C3T]

Recommendation B1. Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors

Currently, each DoD program develops its own development and test environments, which requires redundant definition and provisioning, replicated assurance (including cyber), and extended lead times to deploy capability. Small companies have difficulties providing software solutions to DoD because those software and development test environments are not available outside the incumbent contractor or they have to build (and certify) unique infrastructure from scratch. The desired state is that defense programs will have access to, and be stakeholders in, a cross-program, modern digital infrastructure that can benefit from centralized support and provisioning to lower overall costs and the burden for each program. Development infrastructure supporting continuous integration/continuous delivery (CI/CD) and DevSecOps is available as best-of-breed, and government off-the-shelf (GOTS) is provided so that contractors want to use it, though DoD programs or organizations that want or need to go outside that existing infrastructure can still do so.

Recommendation B2. Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis

To deliver software at speed, rigorous, automated testing processes and workflows are essential. Current DoD practices and procedures often see operational test and evaluation (OT&E) as a tailgate process, sequentially after development has been completed, slowing down delivery of useful software to the field and leaving existing (potentially poorly performing and/or vulnerable) software in place. The desired state is that development systems, infrastructure, and practices are focused on continuous, automated testing by developers (with users). To the maximum extent possible, system operational testing is integrated (and automated) as part of the development cycle using data, information, and test protocols delivered as part of the development environment. Testing and evaluation/certification of COTS components occurs once (if justified), and then ATO reciprocity (Rec B3) is applied to enable use in other programs, as appropriate.

Recommendation B3. Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services

Current software acquisition practice emphasizes the differences among programs: perceptions around different missions, different threats, and different levels of risk tolerance mean that components, tools, and infrastructure that have been given permission to be used in one context are rarely accepted for use in another. The lack of ATO reciprocity drives each program to create its own infrastructure, repeating time- and effort-intensive activities needed to certify elements as secure for their own specific context. The desired state is that modern software components, tools, and infrastructure, once accredited as secure within the DoD, can be used appropriately and cost-effectively by multiple programs. Programs can then spend a greater percentage of their budgets on developing software that adds value to the mission rather than spending time and effort on basic software infrastructure. COTS components are accredited once and then made available for use in other programs, as appropriate.

Line of Effort C. The Services and OSD should create new paths for digital talent (especially internal talent) by establishing software development as a high-visibility, high-priority career track and increasing the level of understanding of modern software within the acquisition workforce. Increased internal capability is necessary both to allow organic (internal) development and to enable the Department to best serve as a knowledgeable partner for software acquired from commercial sources.

Figure 5.4. Airmen assigned to the 707th Communications Squadron, which supports more than 5,700 personnel around the world, update software for Air Force networks. [U.S. Navy photo by Rick Naystatt/Released]

Recommendation C1. Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices

DoD's capacity to apply modern technology and software practices to meet its mission is required to remain relevant in increasingly technical fighting domains, especially against peer adversaries. While DoD has both military and civilian software engineers (often associated with maintenance activities), the IT career field suffers from a lack of visibility and support. The Department has not prioritized a viable recruiting strategy for technical positions, and has no comprehensive training or development program that prepares the technical and acquisition workforce to adequately deploy modern software development tools and methodologies. The desired state is that DoD recruits, trains, and retains internal capability for software development, including by Service Members, and maintains this as a separate career track (like DoD doctors, lawyers, and musicians). Each Service has organic development units that are able to create software for specific needs and that serve as an entry point for software development capability in military and civilian roles (complementing work done by contractors). The Department's workforce embraces commercial best practices for the rapid recruitment of talented professionals, including the ability to onboard quickly and provide modern tools and training in state-of-the-art training environments. Individuals in software development career paths are able to maintain their technical skills and take on DoD leadership roles.

Recommendation C2. Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software

Acquisition professionals have been trained and had success in the current model, which has produced the world's best military, but this model does not serve well for software. New methodologies and approaches introduce unknown risks, and acquisition professionals are often not incentivized to make use of the authorities available to implement modern software methods. At the same time, senior leaders in DoD need to be more knowledgeable about modern software development practices so they can recognize, encourage, and champion efforts to implement modern approaches to software program management. The desired state is that senior leaders, middle management, and organic and contractor-based software developers are aligned in their view of how modern software is procured and developed. Acquisition professionals are aware of all of the authorities available for software programs and use them to provide flexibility and rapid delivery of capability to the field. Program leaders are able to assess the status of software (and software-intensive) programs and spot problems early in the development process, as well as provide continuous insight to senior leadership and Congress. Highly specialized requirements are scrutinized to avoid developing custom software when commercial offerings are available that are less expensive and more capable.

Line of Effort D. DoD and industry must change the practice of how software is procured and developed by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cybersecurity is an integrated element of the entire software life cycle, and purchasing existing commercial software whenever possible.

Figure 5.5. Connected battle command suites. [U.S. Army photo]

Recommendation D1. Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for all DoD-specific code, enabling full security testing and rebuilding of binaries from source

Source code for many DoD systems is not available to DoD for inspection and testing, and DoD relies on suppliers to write code for new compute environments. As code ages, suppliers are not required to maintain codebases without an active development contract, and "legacy" code is not continuously migrated to the latest hardware and operating systems. The desired state is that DoD has access to source code for DoD-specific software systems that it operates and uses to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most important) upgrades (patches and enhancements). DoD is able to rebuild executables from scratch for all of its systems and has the rights and ability to modify (DoD-specific) code when new conditions and features arise. Code is routinely migrated to the latest computing hardware and operating systems, and routinely scanned against currently known vulnerabilities. Modern IP language is used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors are able to use licensing agreements that protect any IP that they have developed with their own resources. Industry trusts DoD with its code and has appropriate IP rights for internally developed code.

Recommendation D2. Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough

Current DoD systems often rely on security-at-the-perimeter as a means of protecting code from unauthorized access. If this perimeter is breached, then a large array of systems can be compromised. Multiple reports by the GAO, the Department of Defense Office of Inspector General (DoDIG), and other agencies have identified cybersecurity as a major issue in acquisition programs. The desired future state is that DoD systems use a zero-trust security model in which it is not assumed that anyone who can gain access to a given network or system should have access to anything within that system. DoD uses regular and automated penetration testing to track down vulnerabilities, and engages red teams to attempt to breach our systems before our adversaries do.

Recommendation D3. Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays

Current DoD requirements processes significantly impede its ability to implement modern software development practices by forcing programs to spend years establishing requirements and insisting on satisfaction of requirements before a project is considered "done." This impedes rapid implementation of features that are of greatest value to the user. The desired state is that rather than a list of requirements for every feature, programs should establish a minimum set of requirements required for initial operation, security, and interoperability, and place all other desired features on a list that will be implemented in priority order, with the ability for DoD to redefine priorities on a regular basis.

5.2 The Next Most Important Things to Tackle

DoD must make a large number of changes to fully realize the vision that 37 years of studies have articulated. This study solicited input from a wide range of stakeholders in the defense software enterprise, including OSD and Service leaders, industry participants in our visits and roundtables, and FFRDC personnel who helped put together our report and identify the recommendations that we should make. The list of recommendations below are the next 0x10 (16) recommendations that we believe can be implemented after actions on the 10 above are solidly underway (like software, implementing recommendations is never "done"). We list these second not because they are dependent on the primary recommendations but simply to emphasize the urgency of the Top Ten.

ID	Recommendation
A3	Require cost assessment and performance estimates for software programs (and software components of larger programs) of appropriate type be based on metrics that track speed and cycle time, security, code quality, and functionality
A4	Refactor and simplify Title 10, DFARS, and DoDI 5000.02/5000.75 to remove statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software; while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule
A5	Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification
A6	Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers associated with hardware-focused regulations and processes
A7	Replace JCIDS, PPB&E, and DFARS with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to determine allocation priorities for software capabilities

B4	Prioritize secure, iterative, collaborative development for selection and execution of new software development programs (and software components of hardware programs), especially those using commodity hardware and operating systems
B5	Remove obstacles to DoD usage of cloud computing on commercial platforms, including DISA CAP limits, lack of ATO reciprocity, and access to modern software development tools
B6	Shift from certification of executables for low- and medium-risk deployments to certification of code/architectures and certification of the development, integration, and deployment toolchain
B7	Plan and fund computing hardware (of all appropriate types) as consumable resources, with continuous refresh and upgrades to current, secure operating systems and platform components
C3	Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition
C4	Restructure the approach to recruiting digital talent to assume that the average tenure of a talented engineer will be 2–4 years, and make better use of HQEs, IPAs, special hiring authorities, reservists, and enlisted personnel to provide organic software development capability, while at the same time incentivizing and rewarding internal talent
D4	Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, user value, and code quality to assess, manage, and terminate software programs (and software components of hardware programs)
D5	Shift the approach for acquisition and development of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success––or be terminated quickly
D6	Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas
D7	Invest in transition of emerging tools and methods from academia and industry for creating, analyzing, verifying, and testing of software into DoD practice (via pilots, field tests, and other mechanisms)
D8	Automatically collect all data from DoD national security systems, networks, and sensor systems, and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).

5.3 Monitoring and Oversight of the Implementation Plan

It would be naive to believe that just listing the recommendations above will somehow ensure they are quickly and easily implemented after 37 years of previous, largely consistent recommendations have had relatively minor impact. We believe that DoD should use these recommendations (and the ones that preceded them) to create an implementation plan for review by stakeholders (including the DIB, if there is interest). This implementation plan might use as its starting point the proposed implementation plans that we have articulated in Appendix A, with agreement by the Secretary of Defense, the Undersecretaries of Defense, the Service Chiefs, CAPE, and DOT&E to support the creation and execution of the next iteration of the implementation plan.

We propose the following timeline for implementing the recommendations proposed here:

- (Immediately): Define, within 60 days after delivery of this report to Congress, a detailed implementation plan and assign owners to begin each of the top recommendations.

- FY19 (create): High-level endorsement of the vision of this report, and support for activities that are consistent with the desired end state (i.e., DevSecOps and enterprise-level architecture and infrastructure). Identify and launch programs to move out on the priority recommendations (start small, iterate quickly).
- FY20 (deploy): Initial deployment of authorities, budgets, and processes for reform of software acquisition and practices. Execute representative programs according to the main lines of effort and primary recommendations in this report. Implement these recommendations in the way we implement modern software: implement now, measure results, and modify approaches.
- FY21 (scale): Streamlined authorities, budgets, and processes enabling reform of software acquisition and practices at scale. In this time frame, adopt a new methodology to estimate as well as determine the value of software capability delivered (and not based on lines of code).
- FY22 (optimize): Conditions established so that all DoD software development projects transition (by choice) to software-enabled processes, with the talent and ecosystem in place for effective management and insight.

5.4 Kicking the Can Down the Road: Things That We Could Not Figure Out How to Fix

Despite the fairly comprehensive view that we have attempted to take in this study regarding how to improve the defense software enterprise, there are a number of challenges remaining that we were not able to address. We summarize these here for the next study (or perhaps one 37 years from now) to consider as DoD continues this path forward.

Over-oversight. DoD's sprawling software enterprise has many oversight actors, spanning Congress, OSD, Service or Component leadership, and other executive branch actors like the GAO. These actors each take frequent oversight action in attempts to improve the software in specific programs and also make well-intentioned efforts to improve the health of the overall system. However, these oversight actions focus primarily on addressing the behavior of the people developing and maintaining the software, overlooking the fact that the oversight itself is equally part of DoD's software problem. Ultimately, we cannot fix software without fixing oversight.

There are at least two categories of problems when it comes to software oversight: structural and substantive.

From a structural perspective, there are too many actors involved in oversight. A program manager, tasked with leading a software development effort, may have as many as 17 other actors who can take some form of oversight action on the program. Most of these individuals do not possess the authority to cancel a program unilaterally, but all have the ability to delay progress or create uncertainty while seeking corrective action for their concerns. These oversight actors often have overlapping or unclear roles and authorities, as well as competing interests and incentives. This means that in addition to the necessary checks and balances required between organizations, there is debate and active competition inside each of the organizations with, for example, various offices in OSD arguing among themselves in addition to arguing with Congress and the Services. Further, there is significant personnel turnover within these positions, meaning that any consensus tends to be short lived.

Substantively, the various oversight actors often do not possess a shared understanding of what constitutes good practice for software or its oversight. Further, these actors may not share a common vision for what DoD’s software enterprise should look like today or in the future. The majority of oversight attention and action is placed on individual programs than on considering portfolios in the aggregate or the performance of the system as a whole. This program oversight is highly subjective in nature, relying on reports and PowerPoint slides presenting narratives and custom-created data. Worse, this oversight operates primarily according to conventional wisdom associated with the oversight of hardware programs, using decades-old heuristics when considering cost, schedule, and performance.

Without understanding what good looks like, or the right questions to ask, oversight actors risk enacting poor fixes. These actions can also be at odds with stated policy. Oversight actions are always more powerful than written policy, meaning that disparities between the two create the risk of cognitive dissonance or a shadow policy environment. Disparities also put program leadership in the unfair position of having to resolve the competing priorities of others, with the knowledge that failure to do so will lead to more blame and action from above.

Structural and substantive problems lead to oversight that is inconsistent and confusing, making it essentially impossible to systematically identify symptoms, determine root causes, or implement scalable fixes. This, in turn, allows everyone involved in DoD software development and maintenance to feel aggrieved, blame everyone other than themselves for systemic issues, and continue their behavior without reflection or change, thus perpetuating the cycle.

The approach by oversight organizations both on the Hill and in DoD should be that policy is treated as the current hypothesis for how best to ship code that DoD’s users need. Through the use of data-driven governance, each program should then be tested against that policy while also being a test of the policy. The hypothesis, and policy, must be continually updated based on standard data that is recognized by, and accessible to, all oversight actors. Implementing such an approach is within the power of the oversight community but would be challenging and appears unlikely given current culture and practices. Regardless, those involved in the oversight of DoD software should not expect meaningfully improved outcomes for that software until the oversight practices used to improve that software are themselves improved.

Promotion practices. Software is disproportionately talent driven. Access to strong engineering talent is one of the most important factors that determine the success or failure of software projects. All that our rivals have to do to surpass us in national security applications of software such as AI, autonomy, or data analytics is to leverage their most talented software engineers to work on those applications. And yet in DoD, as much as we struggle to attract those with technical talent, we also struggle to elevate the talent we have.

The companies and institutions that are winning the software game recognize the importance of identifying and cultivating talented software leaders (whether they are engineers, managers, or strategists working closely with contractors) and actively promote and reward employees based on merit and demonstrated contributions. In contrast, human capital practices in DoD, sometimes by design and sometimes by habit and culture, narrowly limit how technical talent can be evaluated and often prioritize time in grade. The Department needs to figure out how to recognize when civilians and Service Members show an aptitude for software and software management and be able to promote, reward, and retain these individuals outside of the current constraints.

Using commercial software whenever possible. DoD should not build something that it can buy. If there is an 80 percent commercial solution, it is better to buy it and adjust—either the requirements or the product—rather than build it from scratch. It is generally not a good idea to over-optimize for what we view as “exceptional performance,” because counter-intuitively this may be the wrong thing to optimize for as the threat environment evolves over time. Similarly, DoD should take actions to ensure that both the letter and spirit of commercial preference laws (e.g., 10 USC 2377, which requires defense agencies to give strong preference to commercial and non-developmental products) are being followed.

There is a myth that the U.S. private sector—where much of the world’s software talent is concentrated—is unwilling to work on national security software. The reality is that DoD has failed to award meaningful government contracts to commercial software companies, which has generally led to companies making a *business decision* to avoid it. DoD’s existing efforts to target the commercial software sector are governed by a “spray and pray” strategy, rather than by making concentrated investments.[10] DoD seems to love the idea of innovation, but does not love taking sizeable bets on new entrants or capabilities. It is interesting that Palantir and SpaceX are the only two examples since the end of the Cold War of venture-backed, DoD-focused businesses reaching multibillion dollar valuations. By contrast, China has minted around a dozen new multibillion dollar defense technology companies over the same time period. Some of these problems are purely cultural in nature and require no statutory/regulatory changes to address. Others likely will require the changes detailed in our recommendations.

That said, in many cases, there will not be an obvious “buy” option on the table. DoD and the Services should also work together to prioritize interoperable approaches to software and systems that enable rapid deployment, scaling, testing, and optimization of software as an enduring capability; manage them using modern development methods; and eliminate selected hardware-centric regulations and other particularly problematic barriers. The Services should find ways to better recognize software as a key area of expertise and provide specialized education and organizational structures that are better tuned for rapid insertion and continuous updates of software in the field and in the (back) office.

[10] While the overall funding commitments are large—$2 billion from DARPA for AI, for example—those commitments have resulted in few, if any, contracts for private companies other than traditional defense contractors. They have therefore failed to create significant incentives for the commercial tech sector to invest in government applications of AI.

Acknowledgments

The SWAP study members are indebted to a large number of individuals who helped provide valuable input, guidance, and support for the study and for the creation of this report.

We would first like to thank the SWAP study team, who coordinated the many activities associated with the study, including arranging for visits, briefings, and meetings; running the SWAP working group activities; and assisting with the production of the final report. Our initial study director, Bess Dopkeen, was detailed to the study from CAPE and provided outstanding leadership to the overall study. Her vision, energy, and knowledge of the Department were essential in establishing the interactive nature of this activity and helping us obtain insight into the many previously unknown aspects of DoD. She was succeeded by Jeff Boleng, the USD(A&S) Special Assistant for Software, who initially served as our liaison to A&S and took over as study director when Bess departed the Pentagon. Bess and Jeff were assisted by three outstanding members of the core SWAP team: Courtney Barno, Devon Hardy, and Sandra O'Dea. The study and the report could not have come together without the tireless (and patient!) efforts of Bess, Courtney, Devon, Sandy, and Jeff, who participated in every aspect of the report and helped us shape its content, style, and tone.

The SWAP study was also assisted by individuals from the Institute for Defense Analyses (IDA), SEI, and MITRE who served as our experts on the acquisition process and were invaluable in working through the detailed recommendations. Their knowledge of past studies, the acquisition regulations, the many novel approaches to acquisition reform, and the language of the acquisition community helped us better understand the challenges and opportunities for software acquisition and reform. We would particularly like to thank Kevin Garrison (IDA), Nick Guertin (SEI), Tamara Marshall-Keim (SEI), Forrest Shull (SEI), and Craig Ulsh (MITRE) for their help, encouragement, and constant advice.

A major element of the study was the participation of a large SWAP working group consisting of DoD employees who worked with Bess and the SWAP team to provide input to the study and to articulate pain points, ideas for changes, and proposed updates to legislation and regulations. A full list of individuals who participated in the working groups is listed in Appendix J, but we would particularly like to thank John Bergin, Ben FitzGerald, Bill Greenwalt, Amy Henninger, Paul Hullinger, Peter Levine, Melissa Naroski Merker, Jane Rathbun, Ed Wolski, and Philomena Zimmerman.

The Defense Innovation Board (DIB) staff were tightly linked to the SWAP study, which took place under the auspices of the Science and Technology (S&T) Committee. Josh Marcuse was instrumental in initiating the study (including identifying and hiring Bess) and providing keen insights into the report contents and recommendations. Mike Gable, Janet Boehnlein, and Christopher "Bruno" Brunett served as our designated federal officers (DFOs), accompanying us on trips, visits, and meetings and helping us uphold the Federal Advisory Committee Act (FACA) guidelines in a manner that enabled us to interact in a transparent and interactive way with members of the public, the Department, and Congress.

Many high-ranking officials within the Pentagon took the time to meet with us and provide their input, views, and encouragement for our efforts. Chief among these was Ellen Lord, Under Secretary of Defense for Acquisition & Sustainment, who provided input to our study and support for our meetings, while always being careful to help protect the independence of the study team in support of the charge from Congress. We would also like to thank Bob Daigle (CAPE), Dana Deasy (CIO), Bob Behler (DOT&E), Hondo Guerts (USN), and Will Roper (USAF) for their willingness to meet with us on multiple occasions.

Finally, we are indebted to the many individuals working on DoD programs with whom we met, both in industry and in government. On our many visits and in countless briefings, individuals who were working within the current system, and often pushing the boundaries of what is possible, gave us their honest insights and feedback. We are particularly grateful for the help we received from Tory Cuff, Leo Garciga, and CAPT Bryan Kroger, for their willingness to speak with us and help us understand what the future could look like.

SWAP Vignettes

To help illustrate some of the issues facing the Department in the area of software acquisition and practices, the SWAP study solicited a set of “vignettes” on different topics of relevance to the study. These vignettes represent “user stories” contributed by study team members and collaborators; the views expressed here do not necessarily reflect the views of the SWAP study (though they are consistent with the overarching themes contained in the report). The intent of these vignettes is to provide some additional points of view and insights that are more specific and, in some cases, more personal.

List of vignettes:

Vignette 1 – Implementing Continuous Delivery: The JIDO Approach

Forrest Shull

One theme that emerges from the work in this study is that DoD certainly does have successes in terms of modern, continuous delivery of software capability; however, in too many cases, these successes are driven by heroic personalities and not supported by the surrounding acquisition ecosystem. In fact, in several cases the demands of the rest of the ecosystem cause friction that, at best, adds unnecessary overhead to the process and slows the delivery of capability. The Joint Improvised-Threat Defeat Organization (JIDO), within the Defense Threat Reduction Agency, is a compelling example.

JIDO describes itself as "the DoD's agile response mechanism, a Quick Reaction Capability (QRC) as a Service providing timely near-term solutions to the improvised threats endangering U.S. military personnel around the world."[11] As such, the speed of delivery is a key success criterion, and JIDO has made important improvements in this domain. Central to accomplishing these successes has been the adoption of a DevSecOps solution along with a continuous ATO process, which exploits the automation provided by DevSecOps to quickly assess security issues.

At least as important as the tooling are the tight connections that JIDO has enabled among the stakeholder groups that have to work together with speed to deliver capability. JIDO has personnel embedded in the user communities associated with different COCOMs, referred to as Capability Data Integrators (CDIs). These personnel are required to be familiar with the domain, familiar with the technology, and forward-leaning in terms of envisioning technical solutions to help warfighter operations. Almost all CDIs have prior military experience and are deployed in the field, moving from one group of users to another, helping to train them on the tools that are available, and at the same time understanding what they still need. CDIs have tight reachback to JIDO and are able to identify important available data that can be leveraged by software functionality and can be developed with speed through the DevSecOps pipeline.

JIDO has also focused on knocking down barriers among contractors and government personnel. JIDO finds value in relying on contractor labor that can flex and adapt as needed to the technical work, with effort spent on making sure that the mix of government personnel and multiple contractor organizations can work together as a truly integrated team. To accomplish this, JIDO has created an environment with a great deal of trust between government and contractors. There are responsibilities that are inherently governmental and tasks that can be delegated to the contractor. Finding the right mix requires experimentation, especially since finding the personnel with the right skillset on the government side is difficult.

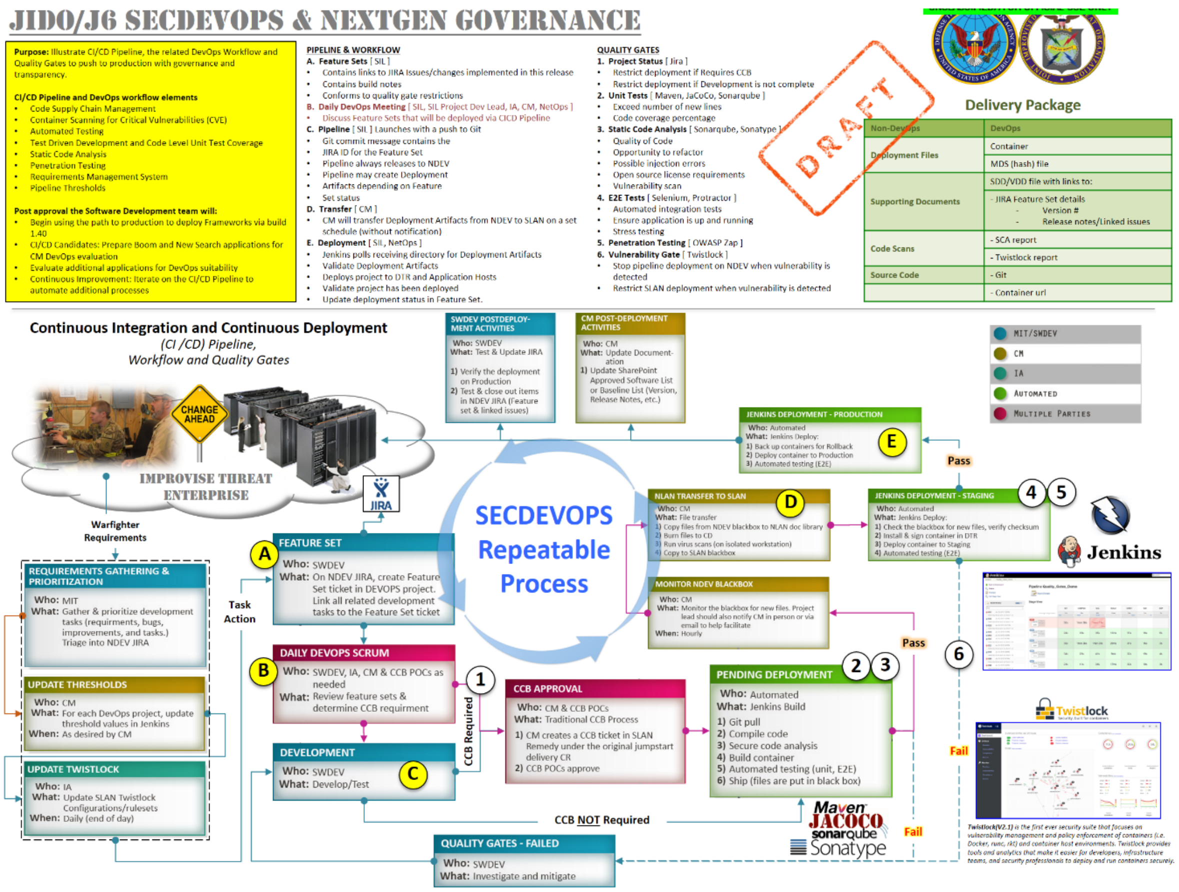
Despite these successes at bringing together stakeholders within the JIDO team, stakeholders in the program management office (PMO) sometimes describe substantial difficulties in working with the rest of the acquisition ecosystem, since on many dimensions the Agile/DevSecOps approach does not work well with business as usual. For example, they describe instances where the Services or the Joint Chiefs push back on solutions that were created to address requirements from the field. Thanks to the CDIs, JIDO can create a technical solution that answers identified

[11] JIDO SecDevOps Concept of Operations, v1.

requirements from warfighters in the field, but that does not mean it will get approval for deployment. There is a mismatch and potential for miscommunication when the organizations that control deployment don't own the requirements themselves.

Also, because JIDO operates in an agile paradigm in which requirements can emerge and get re-prioritized, it is difficult for the organization to justify budget requests upfront in the way that their command chain requires. JIDO addresses this today by creating notional, detailed mappings of functionality to release milestones. Since a basic principle of the approach is that capabilities being developed can be modified or re-prioritized with input from the warfighter, this predictive approach provides little or no value to the JIDO teams themselves. Even though JIDO refuses to map functionality in this way more than 2 years out, given that user needs can change significantly in that time, the program has had to add headcount just to pull these reports together.

JIDO has no problem showing value for the money spent. It is able to show numbers of users and, because it has personnel embedded with user communities, can discuss operational impact. As mentioned above, JIDO's primary performance metric is "response from the theater." Currently, JIDO faces a backlog of tasks representing additional demand for more of its services, as well as a demand for more CDIs. Despite these impactful successes, the surrounding ecosystem unfortunately provides little in the way of support and much that hinders the core mission. It is difficult to see how these practices can be replicated in other environments where they can provide positive impact, until these organizational mismatches can be resolved.



JIDO/J6 SECDEVOPS & NEXTGEN GOVERNANCE

Purpose: Illustrate CI/CD Pipeline, the related DevOps Workflow and Quality Gates to push to production with governance and transparency.

CI/CD Pipeline and DevOps workflow elements
- Code Supply Chain Management
- Container Scanning for Critical Vulnerabilities (CVE)
- Automated Testing
- Test Driven Development and Code Level Unit Test Coverage
- Static Code Analysis
- Penetration Testing
- Requirements Management System
- Pipeline Thresholds

Post approval the Software Development team will:
- Begin using the path to production to deploy Frameworks via build 1.40
- CI/CD Candidates: Prepare Boom and New Search applications for CM DevOps evaluation
- Evaluate additional applications for DevOps suitability
- Continuous Improvement: Iterate on the CI/CD Pipeline to automate additional processes

PIPELINE & WORKFLOW

A. **Feature Sets** [SIL]
- Contains links to JIRA Issues/changes implemented in this release
- Contains build notes
- Conforms to quality gate restrictions

B. **Daily DevOps Meeting** [SIL, SIL Project Dev Lead, IA, CM, NetOps]
- Discuss Feature Sets that will be deployed via CICD Pipeline

C. **Pipeline** [SIL] Launches with a push to Git
- Git commit message contains the
- JIRA ID for the Feature Set
- Pipeline always releases to NDEV
- Pipeline may create Deployment
- Artifacts depending on Feature
- Set status

D. **Transfer** [CM]
- CM will transfer Deployment Artifacts from NDEV to SLAN on a set schedule (without notification)

E. **Deployment** [SIL, NetOps]
- Jenkins polls receiving directory for Deployment Artifacts
- Validate Deployment Artifacts
- Deploys project to DTR and Application Hosts
- Validate project has been deployed
- Update deployment status in Feature Set.

QUALITY GATES

1. **Project Status** [Jira]
- Restrict deployment if Requires CCB
- Restrict deployment if Development is not complete

2. **Unit Tests** [Maven, JaCoCo, Sonarqube]
- Exceed number of new lines
- Code coverage percentage

3. **Static Code Analysis** [Sonarqube, Sonatype]
- Quality of Code
- Opportunity to refactor
- Possible injection errors
- Open source license requirements
- Vulnerability scan

4. **E2E Tests** [Selenium, Protractor]
- Automated integration tests
- Ensure application is up and running
- Stress testing

5. **Penetration Testing** [OWASP Zap]

6. **Vulnerability Gate** [Twistlock]
- Stop pipeline deployment on NDEV when vulnerability is detected
- Restrict SLAN deployment when vulnerability is detected

DRAFT

Delivery Package

Non-DevOps	DevOps
Deployment Files	Container
	MDS (hash) file
Supporting Documents	SDD/VDD file with links to:
	- JIRA Feature Set details - Version # - Release notes/Linked issues
Code Scans	- SCA report
	- Twistlock report
Source Code	- Git
	- Container url

Continuous Integration and Continuous Deployment
(CI /CD) Pipeline,
Workflow and Quality Gates

● MIT/SWDEV
● CM
● IA
● AUTOMATED
● MULTIPLE PARTIES
CHANGE AHEAD
IMPROVISE THREAT ENTERPRISE
JIRA
Warfighter Requirements
SWDEV POSTDEPLOYMENT ACTIVITIES
Who: SWDEV
What: Test & Update JIRA
1) Verify the deployment on Production
2) Test & close out items in NDEV JIRA (Feature set & linked issues)
CM POST-DEPLOYMENT ACTIVITIES
Who: CM
What: Update Documentation
1) Update SharePoint Approved Software List or Baseline List (Version, Release Notes, etc.)
JENKINS DEPLOYMENT - PRODUCTION
E
Who: Automated
What: Jenkins Deploy:
1) Back up containers for Rollback
2) Deploy container to Production
3) Automated testing (E2E)
Pass
SECDEVOPS Repeatable Process
NLAN TRANSFER TO SLAN
D
Who: CM
What: File transfer
1) Copy files from NDEV blackbox to NLAN doc library
2) Burn files to CD
3) Run virus scans (on isolated workstation)
4) Copy to SLAN blackbox
JENKINS DEPLOYMENT - STAGING
4
5
Who: Automated
What: Jenkins Deploy:
1) Check the blackbox for new files, verify checksum
2) Install & sign container in DTR
3) Deploy container to Staging
4) Automated testing (E2E)
Jenkins
MONITOR NDEV BLACKBOX
Who: CM
What: Monitor the blackbox for new files. Project lead should also notify CM in person or via email to help facilitate
When: Hourly
Pass
6
REQUIREMENTS GATHERING & PRIORITIZATION
Who: MIT
What: Gather & prioritize development tasks (requirments, bugs, improvements, and tasks.) Triage into NDEV JIRA
Task Action
FEATURE SET
A
Who: SWDEV
What: On NDEV JIRA, create Feature Set ticket in DEVOPS project. Link all related development tasks to the Feature Set ticket
UPDATE THRESHOLDS
Who: CM
What: For each DevOps project, update threshold values in Jenkins
When: As desired by CM
DAILY DEVOPS SCRUM
B
Who: SWDEV, IA, CM & CCB POCs as needed
What: Review feature sets & determine CCB requirment
1
CCB Required
CCB APPROVAL
Who: CM & CCB POCs
What: Traditional CCB Process
1) CM creates a CCB ticket in SLAN Remedy under the original jumpstart delivery CR
2) CCB POCs approve
PENDING DEPLOYMENT
2
3
Who: Automated
What: Jenkins Build
1) Git pull
2) Compile code
3) Secure code analysis
4) Build container
5) Automated testing (unit, E2E)
6) Ship (files are put in black box)
UPDATE TWISTLOCK
Who: IA
What: Update SLAN Twistlock Configurations/rulesets
When: Daily (end of day)
DEVELOPMENT
C
Who: SWDEV
What: Develop/Test
CCB NOT Required
Maven
JACOCO
sonarqube
Sonatype
Fail
Fail
Twistlock
QUALITY GATES - FAILED
Who: SWDEV
What: Investigate and mitigate

***Twistlock(V2.1)** is the first ever security suite that focuses on vulnerability management and policy enforcement of containers (i.e. Docker, runc, rkt) and container host environments. Twistlock provides tools and analytics that make it easier for developers, infrastructure teams, and security professionals to deploy and run containers securely.*

Slide image received from former DTRA-JIDO chief technology officer.

Vignette 2 – F22: DevOps on a Hardware Platform

Craig Ulsh and Maj Zachary McCarty

The F-22A Raptor program recognized a need for greater speed and agility and took action. In mid-2017, the F-22 Program Office realized the F-22A Raptor modernization efforts were not delivering at a speed that would keep pace with emerging threats. Program leadership secured the expertise of the Air Force Digital Service (AFDS). A joint team assessed the program and captured a series of observations and recommendations. The overarching assessment was:

> The Air Force must move faster, accept a greater amount of risk, and commit to radical change with how the F-22A modernization effort is managed and technology is implemented. Competitors are moving faster, and blaming poor vendor performance will not help the F-22A Raptor remain the dominant air superiority platform.

The F-22A Program Office realized that change was needed. The F-22 acquisition process, steeped in the traditional DoDI 5000 model, was slow and cumbersome, with initial retrofits taking at least 6 years to deliver. The program recognized the following symptoms:

- Requirements were static and rigidly defined.
- Capability was delivered in large, monolithic releases.
- Change was avoided and treated as a deviation from well-guarded baselines.
- The development team placed too much focus on intensive documentation.
- Separate programs with separate contracts drove inefficiencies and conflicting interests.
- Insufficient automation for incremental testing resulted in marathon test events. More specifically, the team identified a number of issues that are common among weapon systems:

Development practices. Development processes were matched to the traditional acquisition process. Large feature sets, multiple baselines, highly manual developer testing tools, and limited focus on continuous software infrastructure upgrades contributed to the slow capability delivery cycle. The team made several specific recommendations under the overarching recommendation for the software development teams to adopt modern software practices.

Planning. Several inefficiencies were identified in the planning process including lack of metrics for estimation of effort, inability to prioritize, and inefficient use of developer time. Again, the team proposed that the program adopt modern agile software processes.

Organization. Organizational gaps included poor collaboration across teams, lack of incentives for engineering talent, and competing priorities across multiple vendors.

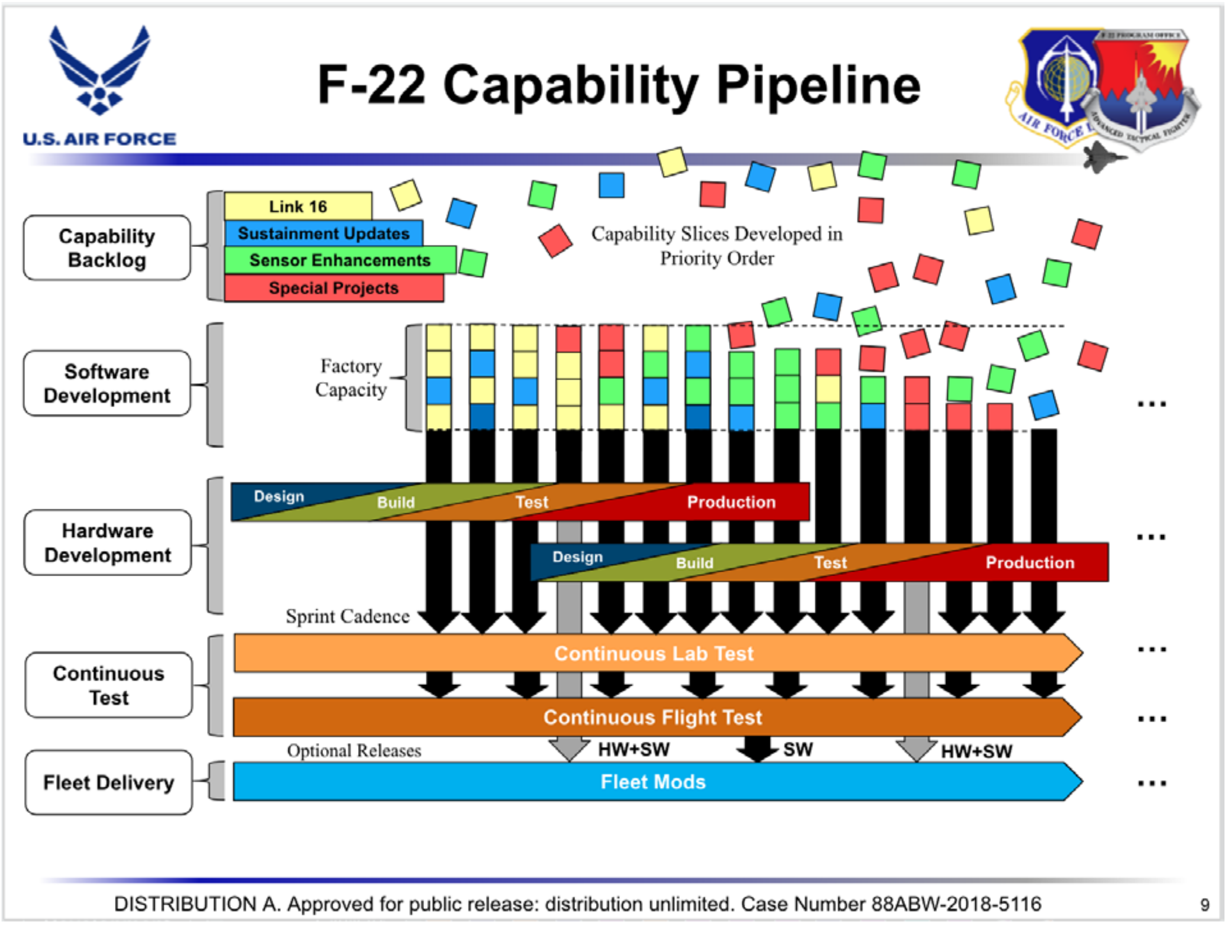
Contracts. The single most significant observation is the failure to prioritize.

In November 2017, the F-22 Program Office took several steps to accelerate the F-22A modernization efforts. In response to outdated development practices, the program office restructured TACLink 16 and TACMAN programs into a single agile development stream. To properly match the contractor effort with a new development approach, a "level of effort" for prime development labor was adopted. To address some of the planning concerns, steps were taken to adjust program alignments and authorities.

The F-22A Raptor program has made positive steps in adopting a more modern approach to both hardware and software acquisition. Perhaps the best example is a new contract structure that allows for quick reaction to emerging requirements and changing user priorities while incentivizing a long-time incumbent contractor for continuous improvement. The Program Office has learned lessons during the transition to more agile approaches, including:

- Culture change has been the biggest hurdle.
- The program must recognize and accept that things will go wrong.
- Security controls limit flexibility and communication.

The program is on the right track with a sound plan to accelerate delivery. But the program office also noted, in the immortal words of Mike Tyson, "Everyone has a plan until they get punched in the face."



DISTRIBUTION A. Approved for public release: distribution unlimited. Case Number 88ABW-2018-5116

Slide image received for briefing from F22A Raptor Program Office.

Vignette 3 – Making It Hard to Help: A Self-Denial of Service Attack for the SWAP Study

Richard Murray

DoD makes use of advisory committees consisting of a mixture of government, industry, and academic experts, all trying to help. However, the Department can make it extremely difficult for these groups to function, an example of what we refer to on the Defense Innovation Board (DIB) as a "self-denial of service attack."[12] The DIB SWAP study is itself a case in point.

<rant>

The DIB Software Acquisition and Practices (SWAP) study clock started ticking when the 2018 NDAA was signed on 12 December 2017. We had our first SWAP discussion at the Pentagon on 16 January 2018, before we had officially been requested by the Under Secretary for Defense (Acquisition and Sustainment) to start, but knowing this was coming (and using the DIB Science & Technology [S&T] committee to ramp up quickly). We identified potential subcommittee members by 12 February, and we were officially charged to carry out the study on 5 April 2018. The one-year Congressionally-mandated end date was thus set as 5 April 2019. The DIB S&T subcommittee submitted the list of suggested subcommittee members. Then we started waiting…

On 24 May, after a DIB meeting, one of the SWAP co-chairs found out that there had been no movement on these positions. He sent a note to the DIB's Executive Director, expressing disappointment and reiterating the importance of getting these people on board early in the study. The Executive Director tried to use this note to push things along. More waiting…

The first activity in which any new member of the SWAP subgroup participated took place on 1 November 2018— a full 30 weeks after our 52-week countdown started and 9 months after we had identified the people whom we wanted to enlist in to help in our study. Even this took repeated interventions by the DIB staff and, in the end, only two of the four people who we hoped could help were able to participate in the study. The timing was such that we had already visited five of the six programs with which we met, written seven of the eight concept papers that we generated, and held three of the four public meetings that provided input for our report.

Why did things take so long? These people were ready to help, had served in government advisory roles in the past, and provided incredibly valuable input in the end (but only in the end). Maybe we need some sort of "FACA Pre ✓" that allows DoD to make use of people who are willing to help and all we need to do is ask.

Another example: the SWAP study decided to use Google's G Suite as the means for writing our report. It had some nice features for collaboration and several of us were familiar with using it. Setting up a G Suite site is fast and easy, and a member of the study had previously created a site in a matter of minutes and had a fully operational, two-factor authenticated set of accounts

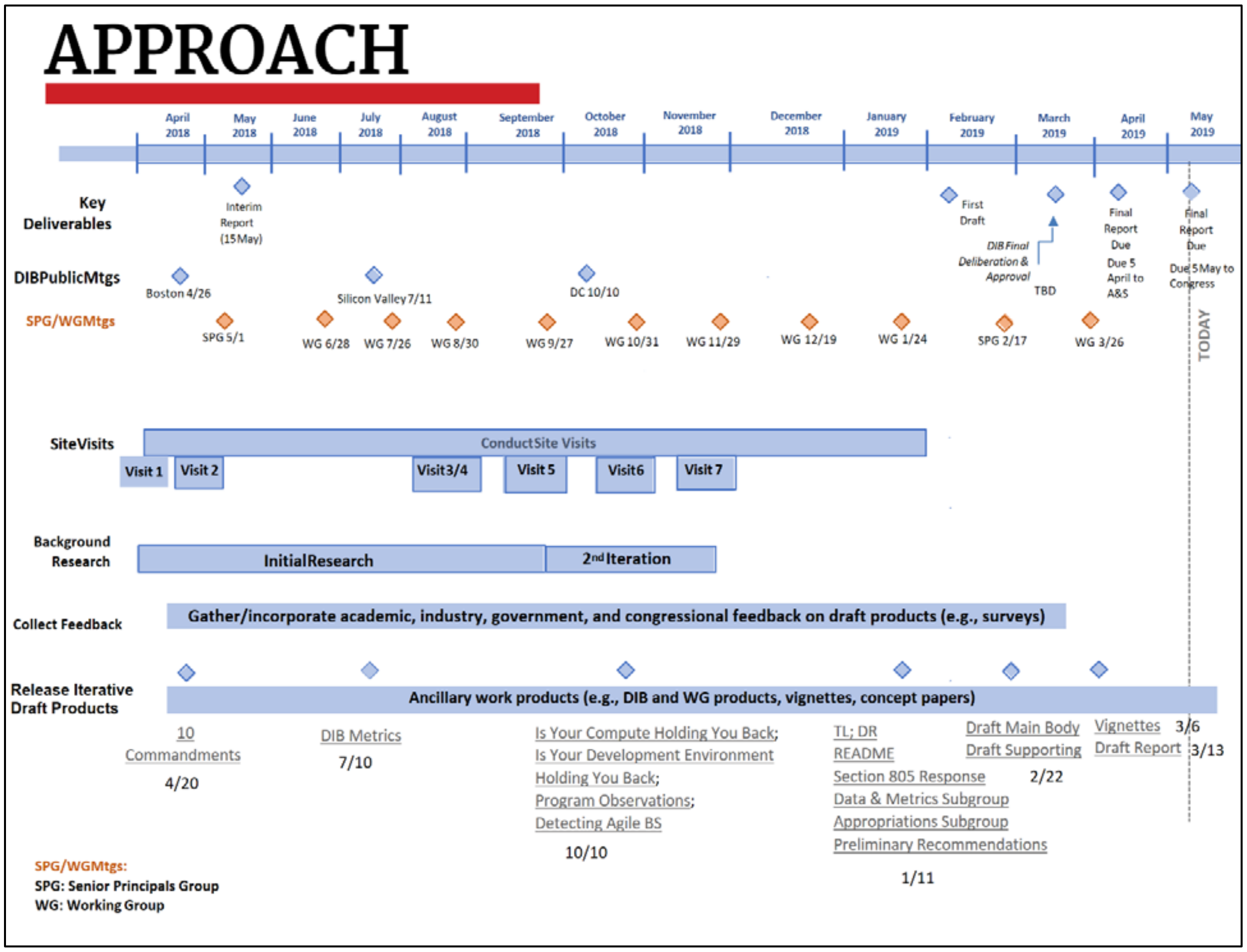
[12] The DIB first heard this term from one of the military instructors at the Air Force Academy and we now use it all the time.

up and running in less than a week. It turns out that the Department has the authority to create official G Suite sites and so we just needed to get permission to use it.

Our request went in ~10 April 2018. The site was created on 8 August 2018, 17 weeks after our request. As near as we can tell, the only thing that happened during the 4 months that it took to get the site working was that people said “no” and then other people had to spend time figuring out why they said no and either convincing them that this really was useful and a good solution for the study’s needs and/or going above their heads.

A major theme from the beginning of the SWAP study, and more generally in the DIB’s overall work, has been that DoD technology must move at the speed of (mission) need, faster than our adversaries and, certainly, not that much slower than what has proven possible and effective in the private sector. If the Department wants to take advantage of people who can help it be more effective in development and delivery of technology for improving national security, it should figure out how to *quickly* put together groups of people from inside and outside government, provide them with modern collaboration environments, and let them spend their time providing service to the Department instead of struggling with the bureaucracy.

</rant>



SWAP study schedule (used for briefings).

Vignette 4 – DDS: Fighting the Hiring Process Instead of Our Adversaries

Sean Brady, Kevin Carter, Justin Ellsworth

In novelist James Patterson and former President Bill Clinton's political thriller, *The President Is Missing*, a terrorist group threatens to unleash cyber-warfare on the Western World, bringing about the "Dark Ages." The President (in the story) must sneak away from the White House incognito, engage in shootouts, survive an ambush on Memorial Bridge, and assemble the best computer scientists from our government and military to take out the impending computer virus before it strikes.

At this point, the novel introduces a top "white hat hacker" who joins the President's team. She impresses the FBI with her hacking abilities and the Bureau hires her on the spot. In a sensational thriller that constantly demands suspended disbelief, this was by far the most unbelievable.

There's no way government hiring works that effectively or efficiently.

We know because we tried.

The Defense Digital Service (DDS) is an organization within the Pentagon tasked with driving a giant leap forward in the way DoD builds and deploys technology and digital services. One of DDS's most visible programs is Hack the Pentagon, the first bug bounty program in the history of the federal government. Bug bounties (also known as crowd-sourced hacking challenges) allow private citizens to harness their diverse range of talents to contribute and strengthen our nation's security posture in exchange for a monetary reward for finding security issues. Bug bounties are an integral part of private-sector security strategies at companies including Microsoft, Google, Twitter, and Facebook.

The winner of one of these Hack the Pentagon challenges was a 17-year-old high school student, who beat out 600 other invited hackers by reporting 30 unique vulnerabilities to the Department. After the challenge, he expressed interest in interning so he could help contribute to our nation's security outside of the challenges.

DDS staff spent the next 8 months and approximately 200 man hours trying to navigate the hiring process to bring the hacker onboard. DDS engaged with the Washington Headquarters Service, the Air Force internship program, and U.S. Army Cyber HR organizations to identify applicable hiring authorities and, more important, the HR specialists who could help drive the hiring actions for a non-traditional, but obviously qualified, candidate.

Unfortunately, what we found was a system ill-equipped to evaluate technical expertise (especially when demonstrated through experience or skill rather than certifications or education) and resistant to leveraging the full flexibilities and authorities provided.

Twice the hacker's resume was rejected as insufficient to qualify him at the necessary grade level for using direct hire authority. Ultimately, the candidate lengthened his resume to a total of five pages, which a classifier reviewed and determined would qualify him for the General Schedule (GS)-4 level, which equates to less than $16 per hour. (For what it's worth, the GS-5 only requires "experience that provided a knowledge of data processing ... gained in work such as a computer operator or assistant, [or] computer sales representative…" according to the OPM GS-2210: Information Technology Management Series General Schedule Qualification Standards). We like to point out that he would have qualified if he had worked a year at Best Buy.

Oh, and did we mention he landed on *TIME*'s List of the 25 Most Influential Teenagers of 2018? He is currently studying computer science at Stanford University.

We recognize that it is unreasonable to expect a classification specialist to understand and translate the experience listed in a resume into the education, demonstrated knowledge, and specialized experience requirements that must be met for each grade level in each job series.

The classification specialist may not have known how this particular candidate's listed experience developing *"mobile applications in IonicJS, mobile applications using Angular, and APIs using Node.js, MongoDB, npm, Express gulp, and Babel,"* met or did not meet the classification requirements of *"experience that demonstrated accomplishment of computer-project assignments that required a wide range of knowledge of computer requirements and techniques pertinent to the position to be filled.*"

This is why DDS provided a supporting memo to the classifier that identified where the candidate's resume and classification guide matched. However, the HR office refused to accept the supporting document despite OPM guidance that *"It is entirely appropriate (and encouraged!) to use Subject Matter Experts (SMEs) outside of HR to rate and rank applicants and determine the most highly qualified candidates for a position."*

Thankfully, our story, like *The President Is Missing*, has a happy ending. When it became clear that we would lose the hacker to a competing offer from the private sector, leaders at some of the highest levels of the Pentagon intervened and ordered their HR office to make the hire. With sufficient visibility and the right people assigned, the hacker's original (one-page) resume was reviewed and used to hire him at a reasonable but still below-market rate. We were ultimately able to hire him, but the process required escalation and is not scalable for more than a small number of hires.

The hacker, now 18, joined DDS as an employee during the summer of 2018 and during that time identified numerous vulnerabilities that threatened the security of information and potentially the safety of our nation.

His story was not isolated to one HR specialist or one service. As a Department, we made it as hard as possible for him to join (all while the private sector offered higher salaries and housing stipends). Hiring him did not require a new law or regulation; it required an understanding of his technical abilities, trust in those who evaluated him, and leadership that prioritizes people over process.

Vignette 5 – Kessel Run: The Future of Defense Acquisitions Is #AgileAF

Dan Ward

I've seen the future, and it's #agileAF.

Kessel Run's lab director welcomes new engineers. [U.S. Air Force photo by Todd Maki]

That's the hashtag used by an Air Force software company known as Kessel Run—the "AF" stands for Air Force, by the way. And I did say "software company," which is how members of this military unit describe their organization. Kessel Run does not look like any other program office the Air Force has ever seen. That is its great strength. That is its great peril. And that is why it is the future.

What's so great about Kessel Run? For starters, it delivers. As one example from many, in less than 130 days Kessel Run fielded an accredited Secret Internet Protocol Router (SIPR) cloud-native DevOps platform at Al Udeid Air Base, then replicated the instance at Shaw Air Force Base and fielded another DevOps platform at Osan Air Base in Japan. Don't worry if that last sentence sounded like technobabble—the point is they put stuff into the field quickly. In contrast, the previous program charged with addressing this need (which went by the catchy name "AOC 10.2") spent $430 million over 10 years before being terminated "without delivering any meaningful capability," to quote Senator John McCain. But while Kessel Run's ability to field operational software is noteworthy, its organizational achievement and the culture the team has built just might be the real breakthrough.

It turns out disruptive new technologies do not merely require cutting-edge tech. They also require new *organizational architectures*, to use Professor Rebecca Henderson's term, and very specific cultural features.

Easier said than done, of course. Building and sustaining these innovative structures inside a large legacy organization like the U.S. military requires replacing existing standards and norms. That's even harder than it sounds and is why so many large companies fail to make the switch.

Despite the difficulty, the Kessel Run team seems to have cracked the code and built a unique organization that operates at warp speed. The most visible difference between Kessel Run and business-as-usual military program offices is their location. Rather than spending all their time on the military base they are *technically* assigned to, Kessel Run personnel operate from a brightly lit We Work office in downtown Cambridge, MA. The conference rooms have Star Wars–themed names instead of Mil-Standard room numbers. The walls are covered in multi-colored sticky notes. The view of Boston is spectacular. You get the picture.

Only slightly less visible is Kessel Run's approach to contracting. Instead of handing the work over to a major defense contractor, team members built a collaborative partnership with a small-ish software company named Pivotal. Together they use DevOps methods like pair programming, where Air Force coders work side-by-side with Pivotal coders to produce software that runs on classified military systems and supports real-world military operations.

Where people sit and how they collaborate are just the tip of the iceberg. The Kessel Run culture is the product of hundreds of thoughtful design decisions that continually reinforce principles of learning, collaboration, critical thinking, and agility. The details of these decisions are beyond the scope of this short vignette, but the fact that Kessel Run continues to do the hard work of deliberately crafting and maintaining its culture is absolutely foundational to its success story.

That story is happening right now, so saying “the future is #agileAF” is actually an observation about the present. Kessel Run’s approach is what right looks like *today*. Kessel Run is the new standard of military acquisition excellence, and already the other Services are starting to follow suit. Just last month the U.S. Naval Institute’s blog had a post titled The Navy’s Kessel Run. When your program office’s name gets used in a headline like that, it’s a sure sign you’re doing something right.

Some skeptical commentators have expressed concern about the risks inherent in a high-speed operation like Kessel Run. In response, let’s hear from the four-star commander of U.S. Strategic Command, General John Hyten. He’s responsible for the nation’s nuclear arsenal and is precisely the type of serious, thoughtful, risk-averse leader we want in charge of nuclear weapons. If anyone has a definitive professional opinion on Kessel Run’s risk profile, it’s General Hyten.

On several occasions General Hyten has stated that what keeps him up at night is the thought that the U.S. military’s technology community has “lost the ability to go fast.” This inability to move quickly increases the likelihood of operational shortfalls and degrades our nation’s overall defense posture. In General Hyten’s assessment, going too slow is far riskier than going too fast. He sounds quite comfortable with Kessel Run’s pace.

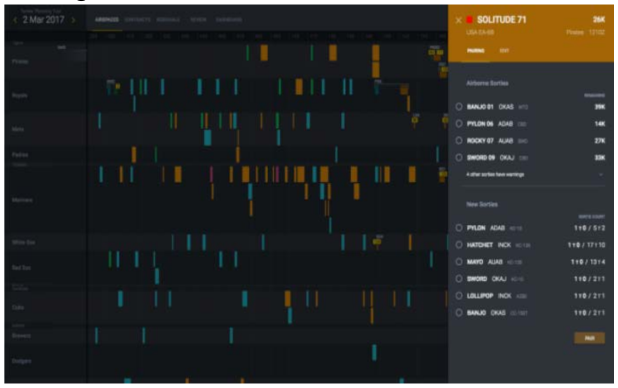
In a similar vein, Secretary of the Air Force Heather Wilson submitted a report to Congress in October 2018 that described Kessel Run’s achievements to date. She wrote “The use of Agile DevOps methodologies … is proving successful and we are able to rapidly deliver cloud native applications that increase operational utility. … *We believe we have demonstrated the ability to continuously deliver software that adds value to the warfighter.* ” (emphasis added.)

So the question is not whether the Kessel Run team delivers good results or addresses the needs of the operational community. It clearly does. Instead, the question is how long it will take the Department of Defense to adopt this organizational innovation on a larger scale. How long will DoD wait before making Kessel Run-style organizations and culture the default rather than the exception?

Replicating the Kessel Run culture requires more than giving all your conference rooms Star Wars-themed names and putting military personnel into civilian clothes. In fact, the best way to replicate the Kessel Run culture is to *not* replicate it exactly. The wisest imitators will use Kessel Run’s example for illumination, not imitation. They will learn from Kessel Run’s practices, not simply cut and paste them onto existing organizational structures. The wisest imitators will commit to having the difficult, ongoing conversations about values, attitudes, and beliefs that lead to genuine culture shifts. They will do the hard work of establishing and maintaining a healthy culture that unleashes people's talent and enables them to do their best work.

Kessel Run is not perfect, of course. It has collected a number of critics and skeptics alongside its fans and supporters. Interestingly, no critics see the project's shortcomings more clearly and pointedly than the Kessel Run members themselves. The team members are very aware they are still learning, still experimenting, still making mistakes and identifying opportunities for improvement. They are the first to tell you that Kessel Run has problems and struggles. They are quick to agree with some of their critics about ways the program can and should improve. That is the thing I admire most about this team. That just might be the most important practice for the rest of us to follow. And that is precisely why the future is #agileAF.

Whiteboard on which tanker refueling operations were planned. [Photo by U.S. Air Force]



The tanker refueling planning app that replaced the AOC's whiteboard. [Photo by U.S. Air Force]

Air Force Kessel Run Headquarters in Boston, MA. [U.S. Air Force photo by J.M. Eddins Jr.]

Vignette 6 – JMS: Seven Signs That Your Software (Program) Is in Trouble

Richard Murray

The DIB SWAP study visited the JMS (JSpOC [Joint Space Operations Center] Mission System) program in August 2018. The JMS team was open and cooperative, and the people working on the project were highly capable and well-intentioned. At the same time, our assessment of the program was that it was doomed to failure. Because the JMS program was restructured after our visit, we felt it was OK to spell out the problems as examples of what can go wrong.

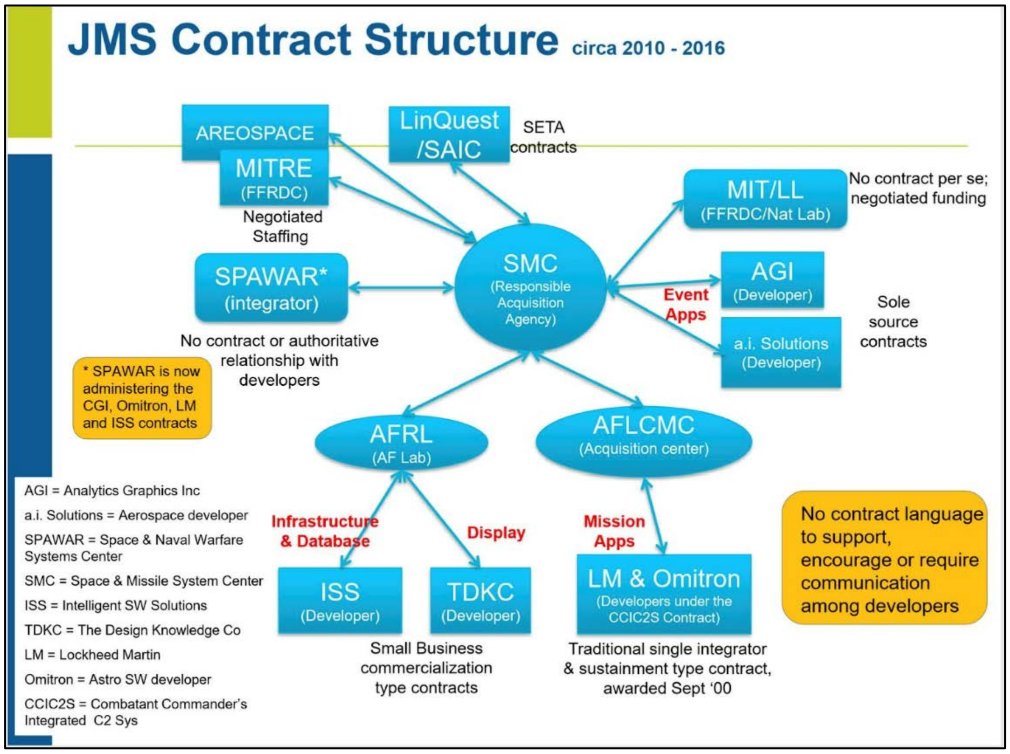
While there were many issues that led to the failure of the JMS program, the following seven are ones that are not a function of that program *per se*, but rather of the process that created it. We thus call these out as general things to look for as indications that your software (program) may be in trouble.

1. The problem is being made harder than it needs to be. JMS increment 2 had a budget of just under $1B. The basic function of the JMS system was to track objects in space. While there are engineering challenges to doing this with the proper precision, the basic problem is *not that hard*. Our sense was that the project could be converted to an "app" within AOC Pathfinder, or something equivalent. Assign 20–30 [50? 100?] programmers (+ 20% program management, administration) to work on it for 3 years at $10–20M/year, with first capability due in 6 months and increments every 2 weeks (based on user feedback). Interface to existing data sources (via software interfaces), run in the cloud, and use a scalable architecture that can get to 1M objects in the next year or two. Make sure that the app architecture can accept a commercial product if one is available that meets the needs of the user (there were some indications this might have already been happening). Target budget: $10–20M/year for first 5 years, $5–15M/year in perpetuity after that.

2. The requirements are outdated. Many of the requirements for JMS increment 2 appeared to trace back to its original inception circa 2000 and/or its restart in 2010. Any software program in which a set of software requirements was established more than 5 years ago should be shut down and restarted with a description of the desired end state (list of features with specifications) and a prioritization of features that should be targeted for simplest usable functionality.

3. The program organizational structure is designed to slow things down. Any software program with more than one layer of indirection between the prime contractor/integrator and the companies doing the engineering work should be shut down and restarted with a set of level-of-effort–style contracts that go directly from the system integrator to the companies delivering code. The system integrator should own the architecture, including the design specifications for the components that plug into that architecture.

4. The program contract structure is designed to slow things down even more. The program had at least a dozen contracts with all sorts of small companies and National Labs. It was apparently treated as a COTS integration problem with lots of pieces, but it was implemented in a way that seemed designed to ensure that nobody could make any progress.


JMS Contract Structure circa 2010 - 2016
AREOSPACE
LinQuest /SAIC
SETA contracts
MITRE (FFRDC)
Negotiated Staffing
MIT/LL (FFRDC/Nat Lab)
No contract per se; negotiated funding
SPAWAR* (integrator)
SMC (Responsible Acquisition Agency)
Event Apps
AGI (Developer)
a.i. Solutions (Developer)
Sole source contracts
No contract or authoritative relationship with developers
* SPAWAR is now administering the CGI, Omitron, LM and ISS contracts
AFRL (AF Lab)
AFLCMC (Acquisition center)
AGI = Analytics Graphics Inc
a.i. Solutions = Aerospace developer
SPAWAR = Space & Naval Warfare Systems Center
SMC = Space & Missile System Center
ISS = Intelligent SW Solutions
TDKC = The Design Knowledge Co
LM = Lockheed Martin
Omitron = Astro SW developer
CCIC2S = Combatant Commander's Integrated C2 Sys
Infrastructure & Database
Display
Mission Apps
ISS (Developer)
TDKC (Developer)
LM & Omitron (Developers under the CCIC2S Contract)
No contract language to support, encourage or require communication among developers
Small Business commercialization type contracts
Traditional single integrator & sustainment type contract, awarded Sept '00

JMS contract structure. [Photo courtesy of former JMS program office]

5. The program is implementing "waterfall with sprints" (otherwise known as Agile BS). The program was implementing "sprints" of ~6–9 months (Agile BS detector alert!). Sprints had hundreds of tasks spread across six development teams. Just coordinating was taking weeks. For a while the program had used 4-week sprints, but infrastructure was not available to support that cadence. Test happened after delivery of software, with very little automation.

6. The program management office is too big and does not know enough about software. We were told there were 200–260 FTEs in the program office. The overall program management should be limited to 10–20% of the size of the program so that resources are focused on the development team (including system architects, user interface designers, programmers, etc.), where the main work gets done. The program office must have expertise in software programs so that it is able to utilize contract and oversight structures that are designed for software (not hardware).

7. OT&E is done as a tailgate process. As an ACAT1 program, JMS was mandated to conduct operational test, a process that nominally required the program to freeze its baseline, do the tests, and then wait 120 days for report. The Operational User Evaluation conducted in early 2018 was terminated early by the Air Force due to poor performance of the system. The OT&E process being used by the program added information to support the termination decision, but it is important to note that had the program not been terminated the tailgate nature of the evaluation was one that would have added further delays.

The JMS program has since undergone major changes to address the issues above, so the criticisms here should be taken as an example of some of the signs that a program is in trouble.

Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board, 3 May April 2019

J. Michael McQuade and Richard M. Murray (co-chairs)
Gilman Louie, Milo Medin, Jennifer Pahlka, Trae' Stephens

Supporting Information

This document contains the supporting information for the Defense Innovation Board's (DIB's) Software Acquisition and Practices (SWAP) study.

Contents

Appendix A: Draft Implementation Plan

The following pages contain summaries for each recommendation that give more detail on the rationale, supporting information, similar recommendations, specific action items, and notes on implementation. The beginning of each recommendation summary includes the recommendation statement, proposed owner, background information, description of the desired state, proposed role for Congress, and a short list of actions describing how the recommendation might be implemented. The remainder of the summary contains a list of recommendations from the DIB Guides (contained in Appendix E of the supporting information), a list of recommendations from the working group reports (Appendix F of the supporting information), and some related recommendations from previous reports.

The recommendations listed here are relatively decoupled, but there are some dependencies between them, as shown to the right. In figure A.1, an arrow leading from one recommendation toward a second recommendation means that the first implementation depends at least somewhat on the implementation of the second. Hence by choosing one recommendation and following the arrows, the list of all recommendations that should also be implemented can be obtained.

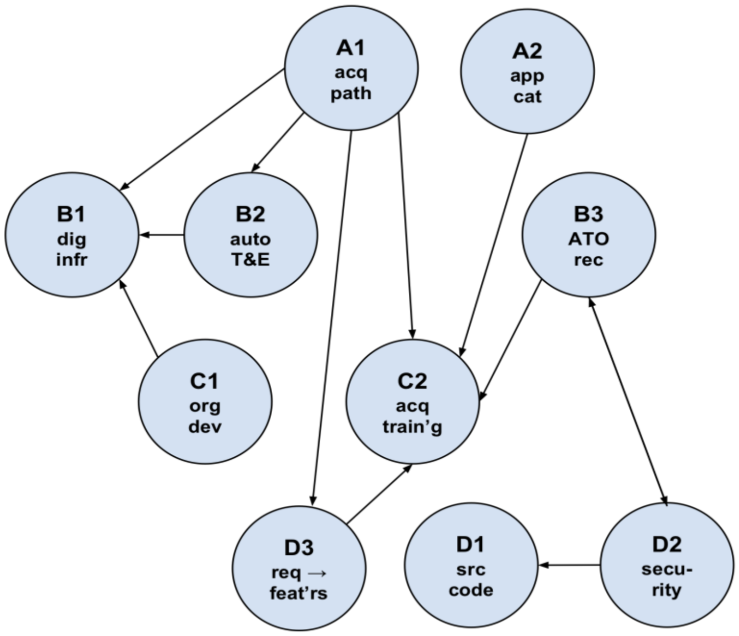


Figure A.1. Interdependency of recommendations.

The recommendations of the report are broken up into four primary lines of effort:

A. Refactor statutes, regulations, and processes for software

B. Create and maintain cross-program/cross-service digital infrastructure

C. Create new paths for digital talent (especially internal talent)

D. Change the practice of how software is procured and developed

For each of the lines of effort, we give a set of two or three primary recommendations (bold) and two to four additional recommendations (see Chapter 5 for insights).

Primary Recommendation A1
New Acquisition Pathway

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Establish one or more new acquisition pathways for software that prioritize continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics.**
Stakeholders	A&S, HASC/SASC, USD(C), CAPE, DOT&E, R&E/DT, SAE, Service FM & PA&E, Joint Staff
Background	Current law, regulation, policy, and internal DoD processes make DevSecOps software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale DevSecOps software development practices to meet mission needs.
Desired State	Tailored, software-specific pathways that provide guidance to acquisition professionals for navigating the acquisition and requirements life cycle to rapidly deliver capabilities. Each pathway streamlines the processes, reviews, and documents based on the type of IT/SW capability. Programs choosing these pathways have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilizing IA-accredited commercial development tools. Rapid acquisition authority should be available for software already in use and accredited, especially when purchased as a capability delivery (as a service). Over time, this becomes the default choice for software and software-intensive programs/program elements.
Role of Congress	This acquisition pathway should become the primary pathway that DoD chooses to use for software and software-intensive programs and should provide Congress with the insight required to oversee software projects that move at a much faster pace than traditional HW programs, with traditional metrics and milestones replaced by more software-compatible measures of progress.

Draft Implementation Plan		**Lead Stakeholder**	**Target Date**
A1.1	(optional) Submit legislative proposal using Sec 805 to propose new acquisition pathways for two or more classes of software (e.g., application, embedded), optimized for DevSecOps.	USD(A&S), in coordination with USD(C) and CAPE	Q3 FY19
A1.2	Create new acquisition pathway(s) for two or more classes of software, optimized for DevSecOps (based on A2c.1 or Appendix B.1).	HASC, SASC	FY20 NDAA
A1.3	Develop and issue a Directive-Type Memorandum (DTM) for the new software acquisition pathway.	USD(A&S)	Q1 FY20
A1.4	Issue Service-level guidance for new acquisition pathway.	SAEs	Q2 FY20

A1.5	Select 5 initial programs using modern software development (DevSecOps) to convert to or use new software acquisition pathway.	USD(A&S), with SAEs	Q2 FY20
A1.6	Develop and implement training at Defense Acquisition University on new software acquisition pathway for all acquisition communities (FM, Costing, PM, IT, SE, etc.).	USD(A&S)	Q3 FY20
A1.7	Convert DTM to DoD Instruction (perhaps 5000.SW), incorporating lessons learned during initial program implementation.	USD(A&S)	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Define software as a critical national security capability under Section 805 of FY16 NDAA "Use of Alternative Acquisition Paths to Acquire Critical National Security Capabilities."
Acq	Create an acquisition policy framework that recognizes that software is ubiquitous and will be part of all acquisition policy models.
Acq	Create a clear, efficient acquisition path for acquiring non-embedded software capability. Deconflict supplemental policies.
Acq	Develop an Enterprise-level Strategic Technology Plan that reinforces the concept of software as a national security capability and recognizes how disruptive technologies will be introduced into the environment on an ongoing basis.
Acq	Additionally, take all actions associated with Rec A2a to refactor and simplify those parts of Title 10, DoD 5000 and other regulations and processes that are still in force for software-intensive programs.

Related recommendations from previous studies

DSB87	Rec 13: The Undersecretary of Defense (Acquisition) should adopt a four-category classification as the basis of acquisition policy [standard (COTS), extended (extensions of current systems, both DoD and commercial), embedded, and advanced (advanced and exploratory systems)].
DSB87	Rec 14: USD(A) should develop acquisition policy, procedures, and guidance for each category.
DSB09	The USD(AT&L) should lead an effort, in conjunction with the Vice Chairman, Joint Chiefs of Staff, to develop new, streamlined, and agile capabilities (requirements) development and acquisition processes and associated policies for information technology programs.

Primary Recommendation A2
New Appropriation Category

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Create a new appropriation category for software capability delivery that allows (relevant types of) software to be funded as a single budget item, with no separation between RDT&E, production, and sustainment.**
Stakeholders	A&S, HAC-D/SAC-D, HASC/SASC, USD(C), CAPE, SAE, Service FM & PA&E, FASAB, OMB
Background	Current law, regulation, and policy treat software acquisition as a series of discrete, sequential steps; accounting guidance treats software as a depreciating asset. These processes are at odds with software being continuously updated to add new functionality, and they create significant delays in fielding user-needed capability.
Desired State	Appropriations for software and software-intensive programs use a Major Force Program (MFP) category that provides a single budget to support full life cycle costs of software, including development, procurement, assurance, deployment, and continuous improvement. Programs are better able to prioritize how effort is spent on new capabilities versus fixing bugs/vulnerabilities, improving existing capabilities, etc. Such prioritization can be made based on warfighter/user needs, changing mission profiles, and other external drivers, not constrained by available sources of funding.
Role of Congress	This should become the primary pathway that Congress uses to fund software and software-intensive programs and should provide Congress with the insight required to oversee software projects that move at a much faster pace than traditional HW programs, with traditional metrics and milestones replaced by more software-compatible measures of progress.

Draft Implementation Plan		**Lead Stakeholder**	**Target Date**
A2.1	(optional) Submit legislative proposal using Sec 805 to create a new appropriations category for software and software-intensive programs.	USD(A&S), with USD(C) and CAPE	Q3 FY19 for FY20 NDAA
A2.2	Create new appropriation category for software-intensive programs, with appropriate reporting and oversight for software (based on Action A2.1 or Appendix B.1).	HAC-D, SAC-D, with OSD, HASC, SASC	FY20 NDAA, FY20 budget
A2.3	Select initial programs using DevSecOps to convert to or use new SW Appropriation in FY20.	USD(A&S), with Service Acquisition Executives	Q4 FY19
A2.4	Define budget exhibits for new SW appropriation (replacement for P- and R-Forms; see Appendix C).	USD(A&S), with USD(C), CAPE, HAC-D, SAC-D	Q4 FY19

A2.5	Change audit treatment of software with these goals: (1) separate category for software instead of being characterized as property, plant, and equipment; (2) default setting that software is an expense, not an investment; and (3) “sustainment” is an integrated part of the software life cycle.	FASAB, with USD(A&S) and USD(C)	End FY20
A2.6	Make necessary modifications in supporting PPB&E systems to allow use and tracking of new software appropriation.	USD(C) and CAPE	Q1 FY21
A2.7	Ensure programs using new software appropriation submit budget exhibits in the approved format.	SAE with USD(C), CAPE	FY 22 POM

SWAP concept paper recommendations related to this recommendation

10C	Budgets should be constructed to support the full, iterative life cycle of the software being procured with amount proportional to the criticality and utility of the software.
Visits	Construct budget to support the full, iterative life cycle of the software.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Revise 10 USC 2214 to allow funding approved by Congress for acquisition of a specific software solution to be used for research and development, production, or sustainment of that software solution, under appropriate conditions.
App	A new multi-year appropriation for Digital Technology needs to be established for each Military Defense Department and the Fourth Estate.
App	Components will program, budget, and execute for information and technology capabilities from one appropriation throughout life cycle rather than using RDT&E, procurement, or O&M appropriations—often applied inconsistently and inaccurately—allowing for continuous engineering.
Con	Congress establishes new authority for contracting for SW development and IT modernization.
M&S	Revise 10 USC 2460 to replace the “software maintenance” with “software sustainment” and use a definition that is consistent with a continuous engineering approach across the life cycle.
M&S	A DoD Working Group should be established to leverage ongoing individual Service efforts and create a DoD contracting and acquisition guide for software and software sustainment patterned after the approach that led to creation of the DoD Open Systems Architecture Contracting Guide.
M&S	Acquisition Strategy, RFP/Evaluation Criteria, and Systems Engineering Plan should address software sustainability and transition to sustainment as an acquisition priority.
Con	Manage programs at budget levels, allow programs to allocate funds at project investment level.
Con	Work with appropriators to establish working capital funds so that there is not pressure to spend funds sooner than when you’re ready (iterative contracts may produce more value with less money).

Related recommendations from previous studies

GAO15	When assigning resources to all activities, the schedule should reflect the resources (labor, materials, travel, facilities, equipment, and the like) needed to do the work, whether they will be available when needed, and any constraints on funding or time.
GAO17	Hold suppliers accountable for delivering high-quality parts for their products through activities including regular supplier audits and performance evaluations of quality and delivery.

GAO17	Prioritize investments so that projects can be fully funded and it is clear where projects stand in relation to the overall portfolio.
CSIS18	Performance Based Logistics (PBL) contracts should have a duration that allows for tuning and re-baselining with triggered options and rolling extensions.
Sec809	Rec. 41: Establish a sustainment program baseline, implement key enablers of sustainment, elevate sustainment to equal standing with development and procurement, and improve the defense materiel enterprise focus on weapon system readiness.
Sec809	Rec. 42: Reduce budgetary uncertainty, increase funding flexibility, and enhance the ability to effectively execute sustainment plans and address emergent sustainment requirements.

Additional Recommendation A3
Metrics for Cost Assessment and Performance Estimates

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Require cost assessment and performance estimates for software programs (and software components of larger programs) of appropriate type be based on metrics that track speed and cycle time, security, code quality, and functionality.**
Stakeholders	CAPE, CMO, USD(A&S), Service CMOs and SAEs
Background	Current software cost estimation and reporting processes and procedures in DoD have proven to be highly inaccurate and time consuming. New metrics are required that match the DevSecOps approach of continuous capability delivery and maintenance and provide continuous insight into program progress.
Desired State	Program oversight will re-focus on the value provided by the software as it is deployed to the warfighter/user and will rely more heavily on metrics that can be collected in a (semi-)automated fashion from instrumentation on the DevSecOps pipeline and other parts of the infrastructure. Specific metrics will depend on the type of software rather than a one-size-fits-all approach.
Role of Congress	Congress needs to emphasize the need for new software acquisition reporting that focuses on value provided for the investment in software and frequency of deployments to the warfighter/user. Congress needs to work with CAPE and USD(A&S) to provide feedback on meaningful content and level of detail in reporting.

Draft Implementation Plan		**Lead Stakeholders**	**Target date**
A3.1	Identify (or hire) a small team (3-4) programmers to implement software for automated collection and analysis of metrics and provide them with a modern development environment.	CAPE, DDS	Q4 FY19
A3.2	Identify low-level metrics that are already part of standard commercial development environments (see Appendix C for reporting approach and Appendix E.2 (DIB's "Metrics for Software") for initial lists).	CAPE, SAO	MVP[1] Q4 FY19, then quarterly
A3.2a	Speed and cycle time: launch → initial use, cycle time	Dev team, users	
A3.2b	Code quality: unit test coverage, bug burn-rate, bugs-in-test:bugs-in-field	Dev team, users	
A3.2c	Security: patch → field, OS upgrade → field, HW/OS age	Dev team, users	
A3.2d	Functionality: user satisfaction, number/type of features/cycle	Dev team, users	
A3.2e	Cost: head count, software license cost, compute costs	Dev team, users	
A3.3	Identify 3-5 ongoing programs that are collecting relevant metrics and that partner with CAPE to collect and use data.	CAPE, A&S, CMO, SAEs	In parallel with A6.2

[1] Minimum viable product (first useful iteration)

A3.4	Create a mechanism to transfer and process low-level metrics from development team to PMO on a continuous basis with selectable levels of resolution across the program.	CAPE, SAEs, PMO	MVP Q4 FY19, then quarterly
A3.5	Begin reporting metrics to Congress as part of annual reporting; iterate on content, level, format.	CAPE, Comp, A&S	FY2020
A3.6	Use initial results to establish expectations for new proposed software or software-intensive projects and integrate use of new cost and performance estimates into contract selection.	A&S, SAEs, CAPE	FY2020
A3.7	Establish ongoing capability within CAPE to update metrics on continuous basis, with input from users (of the data).	CAPE	FY2021
A3.8	Identify and eliminate remaining uses of ESLOC as metric for cost and schedule estimation of software/software-intensive programs.	CAPE, SAEs	FY2022

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Revise estimation models - source lines of code are irrelevant to future development efforts, estimations should be based on the team size and investment focused (Cultural).

Related recommendations from previous studies

SEI01	Effort Estimation: • Utilize most likely effort estimates in proposals and status reports; • Find ways to promote the use of accurate effort estimation and productivity evaluation; • Lowest cost is not equivalent to best value. Question outliers.
OSD06	Adjust program estimates to reflect “high confidence”—defined as a program with an 80 percent chance of completing development at or below estimated cost—when programs are baselined in the Stable Program Funding Account.
SEI10	Don’t require PMO to adopt contractors’ estimate for the program—or else use the difference as PM “reserve.”
SEI10	Change from traditional 50% estimation confidence level to 80% level.
SEI10	DoD should consider use of Vickrey “second price” auction mechanism for acquisition proposal bidding.
SEI15	Use the government’s cost estimates (using perhaps an 80% confidence level) rather than contractors’ estimates as the basis for program budgets and place the difference (if the government’s estimate is larger) in a reserve fund available to program managers with sufficient justification. Contractors’ estimates should be acquired using mechanisms that promote accurate estimates, e.g., using Vickrey auctions, the Truth-Revealing Incentive Mechanism (TRIM), or more standard methods of review and acceptance by independent third parties.
DSB18	Rec 3b: The MDA with the Cost Assessment and Program Evaluation office (CAPE), the USD(R&E), the Service Cost Estimators, and others should modernize cost and schedule estimates and measurements.
DSB18	Rec 3b.1: [DoD] should evolve from a pure SLOC approach to historical comparables as a measurement, and should adopt the National Reconnaissance Office (NRO) approach (demonstrated in Box 5) of contracting with the defense industrial base for work breakdown schedule data to include, among others, staff, cost, and productivity.
DSB18	Rec 3c: The MDA should immediately require the PM to build a program-appropriate framework for status estimation.

Additional Recommendation A4
Simplify Laws and Policies

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Refactor and simplify Title 10, DFARS, and DoDI 5000.02/5000.75 to remove statutory, regulatory, and procedural requirements that generate delays for acquisition, development, and fielding of software while adding requirements for continuous (automated) reporting of cost, performance (against updated metrics), and schedule.**
Stakeholders	USD(C), CAPE, SAE, Service FM & PA&E, Joint Staff
Background	Current law, regulation, policy, and internal DoD processes make modern software development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale modern software development practices to meet mission needs. Recommendation A1 (new acquisition pathway) provides a pathway that is optimized for software, but it is also possible to modify existing statutes, regulations, and processes to remove barriers for software.
Desired State	Programs have the ability to rapidly field and iterate new functionality in a secure manner, with continuous oversight based on automated reporting and analytics, and utilizing IA-accredited commercial development tools. Congress has better insight into the status of software programs through improved reporting of relevant metrics (see also Recommendations A3 and D4 on metrics).
Role of Congress	Work with DoD to review current statutes and evaluate their effectiveness for different types of software, removing barriers that add time and interfere with the continuous nature of modern software development. See Appendix F for a list of issues to consider.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
A4.1	Submit legislative proposal(s) to simplify Title 10 for software (see also: Sec 809 Panel report).	USD(A&S)	Q3 FY19
A4.2	Convene working group with stakeholders and develop and issue a Directive-Type Memorandum (DTM) for the new simplified software acquisition process.	USD(A&S)	Q1 FY20
A4.3	Issue Service-level guidance for new simplified software acquisition process.	SAE	Q1 FY20
A4.4	Identify initial set of programs using modern software development methods to convert to or utilize new, simplified software acquisition process.	USD(A&S), with SAEs	Q1 FY20
A4.5	Convert DTM to DoD Instruction, incorporating lessons learned during initial program implementation.	USD(A&S)	Q1 FY20
A4.6	Develop and implement training at Defense Acquisition University on new, simplified software acquisition process for all acquisition communities (FM, Costing, PM, IT, SE, etc.).	USD(A&S)	Q1 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Ensure appropriate integration of a data strategy and the Department's Cloud Strategy. Examine a Steering Committee approach for management.
Acq	Examine the organizational structure with the intent of achieving a more responsive and flat organizational model that de-conflicts roles and responsibilities between the DoD CIO, the USD(A&S), and the CMO regarding software.
Acq	Re-focus the software acquisition workforce on teaming and collaboration, agility, improved role definition, career path advancement methods, continuing education and training opportunities, incentivization, and empowerment.
Acq	Increase flexibility and agility for software programs by eliminating mandated content for acquisition strategies and authorities in Section 821 of the FY16 NDAA, except for MDAPs.
Acq	Eliminate hardware-centric cost, fielding, and performance goals in 10 USC 2488 (established by Sec 807 of the FY17 NDAA) for software-intensive programs.
Acq	Eliminate Nunn-McCurdy breaches (10 USC 2433) for software-intensive programs and replace with continuous evaluation of software performance metrics.
Acq	Remove statutory definition of "major system" for software-intensive programs in 10 USC 2302 and 2302d to remove confusion, since most software in weapons systems inherently functions together to fulfill a mission need.
Acq	Develop language for 10 USC 2366 that allows exemption for software-intensive programs, where DOT&E must justify adding the program for oversight with the MDA and must streamline the process.
Acq	Only require DOT&E oversight for software-intensive programs when requested by the SAE, USD(A&S), or Congress, or if the program is an MDAP.
Acq	For the Fourth Estate, combine all three authorities for DBS under the DoD CMO. After one year, conduct assessment and make a determination if this should be applied to the Services as well.
Acq	Eliminate the separate annual funding certification process for defense business system from 10 USC 2222 or require that funding certification be merged in to the PPBE process.
Acq	Replace annual configuration steering board (CSBs) for software-intensive programs with board (or equivalent entities) established by the CAE, PEO, or PM [FY09 NDAA Sec 814; DoDI 5000.02].
Acq	Expand the FAR 39 (Acquisition of IT) to allow for one area to drive technology purchases. Unless otherwise stated, no other FAR rules would apply.
Acq	Rewrite FMR Volume 2A, Chapter 1, Section 010212(B) to [1] acknowledge that, for the purpose of modifying or enhancing software, there is no technically meaningful distinction between RDT&E, Procurement, and O&M; [2] eliminate the $250,000 barrier between expenses and investments (i.e., stop explicitly tying to a dollar threshold, the determination of whether software is an expense or an investment).
Acq	Revise or eliminate DoDI 8330.01 to eliminate the following elements for software-intensive programs: [1] NR KPP required; [2] DoD-specific architecture products in the DoDAF format that are labor intensive and of questionable value; [e] Interoperability Support Plans (ISPs) required, where DoD CIO can declare any ISP of "special interest"; [2] requirement of DT authority to provide assessments at MS C; [5] mandates JITC to do interoperability assessments for IT with "joint, multinational, and interagency interoperability requirements."

Acq	Revise PfM policy (DoDD 7045.20) to consider the role of data and metrics, as well as additional portfolios (like NC3), and determine authority for the policy.
Con	Separate Contract requirements (scope, PoP, and price) from technical requirements (backlog, roadmap, and stories).
Con	Use SOO vs. SOW to allow the vendor to solve the objectives how they are best suited.
Con	Establish clear and intuitive guidelines on how and when to apply existing clauses.
Con	Have standard clause applications for each of the above that must be excepted vs. accepted.
D&M	Congress could establish, via an NDAA provision, new data-driven methods for governance of software development, maintenance, and performance. The new approach should require on-demand access to standard (and perhaps real-time) data with reviews occurring on a standard calendar, rather than the current approach of manually developed, periodic reports.
M&S	Title 10 USC 2460 should be revised to replace the term “software maintenance” with the term “software sustainment” and use a definition that is consistent with a continuous engineering approach across the life cycle.
Req	The Joint Staff should consider revising JCIDS guidance to focus on user needs, bypassing the JCIDS process as needed to facilitate rapid software development. Guidance should specifically account for user communities (e.g., Tactical Action Officer (TAO), Maritime Operations Center (MOC) director) that do not have one specific PoR assigned to them, but use multiple systems and data from those systems to be effective.
Req	The Joint Staff should consider revising JCIDS guidance to separate functionality that needs high variability from the functionality that is deemed “more stable” (e.g., types of signals to analyze vs. allowable space for the antenna). Then implement a “software box” approach for each one in which the contours of the box are shaped by the functionality variability.
Req	The Joint Staff should consider revising JCIDS guidance to document stable concepts, not speculative ideas. Acknowledge that software requirement documents will iterate, iterate, iterate. JCIDS must change from a “one-pass” mentality to a “first of many” model that is inherently agile, delegating approval to the lowest possible level.

Related recommendations from previous studies

DSB87	Rec 21: DoD should examine and revise regulations to approach modern commercial practice insofar as practicable and appropriate.
NPS16a	Program offices spend far too much time generating paperwork and navigating the bureaucracy rather than thinking creatively about program risks, opportunities, and key elements of their strategies.
NDU17	Develop and maintain core competencies in diverse acquisition approaches and increase the use of venture-capital-type acquisitions, such as Small Business Innovative Research (SBIR), Advanced Concept Technology Development (ACTD), and Other Transaction Authority (OTA), as mechanisms to draw in non-traditional companies.
NDU17	Encourage employees to study statutes and regulations and explore innovative and alternative approaches that meet the statutory and regulatory intent.
Sec809	Rec. 62: Update the FAR and DFARS to reduce burdens on DoD’s commercial supply chain to decrease cost, prevent delays, remove barriers, and encourage innovation available to the

	Military Services.
Sec809	Rec. 74: Eliminate redundant documentation requirements or superfluous approvals when appropriate consideration is given and documented as part of acquisition planning.
Sec809	Rec. 75: Revise regulations, instructions, or directives to eliminate non-value-added documentation or approvals.
Sec809	Rec. 90: Reorganize Title 10 of the U.S. Code to place all of the acquisition provisions in a single part, and update and move acquisition-related note sections into the reorganized acquisition part of Title 10.

Additional Recommendation A5
Streamlined Processes for Business Systems

Line of Effort	Refactor statutes and regulations for software.
Recommendation	**Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially available products with minimal (source code) modification.**
Stakeholders	CMO, USD(A&S), Service CMOs, SAEs, DoD CIO
Background	Current DoD business processes are minimally standardized due to a high number of legacy systems that inhibit business process reengineering. In addition, solicitation for new business systems often insists on customization because DoD is “different,” resulting in hard-to-maintain systems that become obsolete (and possibly insecure) quickly.
Desired State	DoD uses standard commercial packages for enterprise and business services, changing its processes to match those of large industries, allowing its systems to be updated and modified on a much faster cadence. The only specialized defense business systems should be those for which there is no commercial equivalent (to include cases in which minor modifications would be required) and there is a funded internal capability to maintain and update the software at a near-commercial cadence.
Role of Congress	Congressional approval for new software development programs should be based on a clear assessment of the current state of commercial software and the need for DoD-specific customization. In many cases it should be possible to make use of commercial systems and modify the DoD process to be consistent with commercial practice rather than attempting to build and maintain specialized business systems. Support legislative change of 10 USC §2222, as needed.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
A5.1	Use a Net Promoter Score (NPS) assessment to identify 10 programs whose customers (soldiers, civilians, or others) believe the functionality could be better executed with commercial software.	CMO, with USD(A&S), Service counterparts	Q4 FY19
A5.2	Using the results of A5.1, select four projects for a more detailed assessment of possible savings and/or efficiency improvements.	CMO, with Service CMOs and business process owners	Q1 FY20
A5.3	Implement COTS opportunities, with contracts in place.	Services, with CMO oversight	Q1 FY21
A5.4	Submit legislative change proposal to modify Title 10 §2222 to reflect the lessons learned through process re-engineering to utilize commercially available system over DoD-specific solutions.	CMO, with USD(A&S) and Service counterparts	FY21

SWAP concept paper recommendations related to this recommendation

10C	Use commercial process and software to adopt and implement standard business practices within the Services.
D&D	For common functions, purchase existing software and change DoD processes to use existing apps.

Related recommendations from previous studies

DSB87	Rec 15: The USD(A) and the ASD(Comptroller) should direct Program Managers to assume that system software requirements can be met with off-the-shelf subsystem and components until it is proved that they are unique.
Sec809	Rec 16: Combine authority for requirements, resources, and acquisition in a single, empowered entity to govern DBS portfolios separate from the existing acquisition chain of command.
Sec809	Rec 18: Fund DBSs [defense business systems] in a way that allows for commonly accepted software development approaches.

Additional Recommendation A6
Enduring Capability

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Plan, budget, fund, and manage software development as an enduring capability that crosses program elements and funding categories, removing cost and schedule triggers associated with hardware-focused regulations and processes.**
Stakeholders	USD(A&S), USD(C), SAE, Service FM, HASC, SASC
Background	The current approach to acquiring software is based on projects that have a beginning and end. However, many missions are “enduring capabilities” and need software program and portfolio management that continually and perpetually deliver across the spectrum of new capability, incremental enhancements, and life cycle sustainment. The Department should pilot and then scale methods for appropriating software budgets for these enduring capability programs as an ongoing, regularly evaluated expense, with continuous oversight, rather than large, multi-year development contracts.
Desired State	The Department can manage software acquisition as an activity requiring continuous development, deployment, and sustainment, recognizing that software systems are long-lived and have a continuous need for a level of activity to evolve capabilities and address vulnerabilities. Assessment of progress will be maintained throughout the software lifespan by means of continual user engagement with working software, rather than at large-scale milestone gates that do not map well to the underlying technical activities.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholder	Target Date
A6.1	Modify FMR to implement this continuous funding approach.	USD(C)	Q4 FY19
A6.2	Select and launch five programs to be managed as enduring capability, two-year pilot projects.	USD(A&S) with SAE	Q4 FY19
A6.3	Work with FASAB to create an audit treatment of enduring capability software that has a category distinct from Property, Plant, and Equipment; defaults to treating software as an expense, not an investment; and does not distinguish between development and sustainment.	USD(A&S) with USD(C)	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Budgets should be constructed to support the full, iterative life cycle of the software being procured with amount proportional to the criticality and utility of the software.
D&D	Treat software development as a continuous activity, adding functionality continuously.

Additional Recommendation A7
Portfolio Management

Line of Effort	Refactor statutes, regulations, and processes for software.
Recommendation	**Replace JCIDS, PPB&E, and DFARS with a portfolio management approach to software programs, assigned to "PEO Digital" or an equivalent office in each Service that uses direct identification of warfighter needs to determine allocation priorities for software capabilities.**
Stakeholders	USD(A&S), CAPE, JCS, USD(C), SAE, Service FM & PAE
Background	The current requirements process often drives the development of exquisite requirements that tend to be overly rigid and specific and attempt to describe the properties of systems in dynamic environments years in advance. The speed of requirements development and analysis is out of sync with the pace of technology and mission changes. Most importantly, requirement documents that are developed are often disconnected with the end-user requirements.
Desired State	Software programs are managed using a portfolio approach, in which resources are available for reallocation across programs and funding categories based on the importance and opportunities of given elements of the portfolio. Relevant portfolios are defined based on the linkages between programs of similar function, as defined by OSD and/or Services.
Role of Congress	Congress should approve and monitor metrics of success defined within different portfolios and measure the progress against those metrics in determining allocations of funding to different portfolios (with the decisions within a portfolio made by the portfolio office, which would be held accountable for those decisions).

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
A7.2	Select initial capability areas in each Service to place under portfolio management by PEO Digital (or equivalent).	SAEs	Q3 FY19
A7.1	Issue guidance for management of software portfolios with a "PEO Digital" or similar office with OSD and/or the Services.	USD(A&S) SAE	Q4 FY19
A7.3	Stand up PEO Digital or equivalent office with necessary resources allocated and aligned.	SAE	Q1 FY20
A7.4	Implement new portfolio management methods for initial program capability areas.	PEO Digital	Q3 FY20
A7.5	Determine intermediate successes of, or required modifications to, portfolio management approach.	PEO Digital	Q1 FY21
A7.6	Establish portfolio management approach as standard work for software.	PEO Digital, SAE	FY22

SWAP working group inputs (reflected in Appendix F) related to this recommendation

App	Within each Component-unique Budget Activity (BA), Budget Line Items (BLINs) align by functional or operational portfolios. The BLINs may be further broken into specific projects to provide an even greater level of fidelity. These projects would represent key systems and supporting activities, such as mission engineering.
App	By taking a portfolio approach for obtaining software-intensive capabilities, the Components can better manage the range of requirements, balance priorities, and develop portfolio approaches to enable the transition of data to information in their own portfolios and data integration across portfolios to achieve mission effects, optimize the value of cloud technology, and leverage and transition to the concept of acquisition of whole data services versus individual systems.
App	This fund will be apportioned to each of the Military Departments and OSD for Fourth Estate execution.
App	Governance: management execution, performance assessment, and reporting would be aligned to the portfolio framework—BA, BLI, project.
Req	OSD and the Joint Staff should consider creating “umbrella” software programs around “roles” (e.g., USAF Kessel Run).

Related recommendations from previous studies

OSD06	Transform the Planning, Programming, and Budgeting, and Execution process and stabilize funding for major weapons systems development programs.
DSB09	The USD(AT&L) aggressively delegate milestone decision authority commensurate with program risk.
DSB09	The USD(AT&L) consider a more effective management and oversight mechanism to ensure joint program stability and improved program outcomes.
DSB09	Consolidate all acquisition oversight of information technology under the USD(AT&L) by moving into that organization those elements of the OASD (NII)/DOD CIO and Business Transformation Agency responsible for IT acquisition oversight. The remainder of OASD (NII)/DOD CIO is retained as it exists today, but should be strengthened as indicated in the previous recommendation.
Sec809	Rec 36: Transition from a program-centric execution model to a portfolio execution model.
Sec809	Rec 37: Implement a defense-wide capability portfolio framework that provides an enterprise view of existing and planned capability, to ensure delivery of integrated and innovative solutions to meet strategic objectives.
Sec809	Rec. 38: Implement best practices for portfolio management.
Sec809	Rec. 39: Leverage a portfolio structure for requirements.

Primary Recommendation B1
Digital Infrastructure

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field, and incentivize its use by contractors.**
Stakeholders	A&S, CIO, SAE, USD(C)
Background	Currently, DoD programs each develop their own development and test environments, which requires redundant definition and provisioning, replicated assurance (including cyber), and extended lead times to deploy capability. Small companies and other new entrants have difficulties providing software solutions to DoD because those environments are not available outside the incumbent contractor or because they have to build (and certify) unique infrastructure from scratch.
Desired State	Programs will have access to, and be stakeholders in, a cross-program, modern digital infrastructure that can benefit from centralized support and provisioning to lower overall costs and the burden for each program. Development infrastructure supporting CI/CD and DevSecOps is available as best of breed and GOTS provided so that contractors want to use it, though DoD programs or organizations that want or need to go outside of that existing infrastructure can still do so.
Role of Congress	Congress should track the availability, scale, use, and cost effectiveness of digital infrastructure, with the expectation that overall capacity will expand while unit costs decrease over time. Sufficient funding should be provided on an ongoing basis to maintain and upgrade digital infrastructure and to maintain best-of-breed capability that accelerates software development.

Draft Implementation Plan		**Lead Stakeholder**	**Target Date**
B1.1	Designate organization(s) responsible for creating and maintaining the digital infrastructure for each Service's digital infrastructure. Explore the use of tiered approaches with infrastructure at Service or Program level, as appropriate.	DoD CIO, USD(C) and Services (SAE and Service CIO)	Q3 FY19
B1.2	Designate organization(s) responsible for creating and maintaining digital infrastructure(s) for DoD agencies and organizations, including joint digital infrastructure available to the Services.	USD(A&S), with CIO, CMO	Q3 FY19
B1.3	Provide resources for digital infrastructure, including cloud solutions, pre-approved "drop-ship" local compute capability, approved development environments (see DIB Compute Environment concept paper, Appendix I [Glossary]).	USD(A&S), SAE with CAPE, USD(C)	FY20 budget
B1.4	Define baseline digital infrastructure systems and implement procurement and deployment processes and capability.	Responsible organizations from B1.1, B1.2	Q2 FY20

B1.5	Implement digital infrastructure and provide access to ongoing and new programs.	Responsible organizations from B1.1, B1.2	Q3 FY20
B1.6	Identify acquisition programs to transition to digital infrastructure.	SAE	Q2 FY20
B1.7	Transition programs to digital infrastructure.	SAE, CIO, PEO, PM	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Make computing, storage, and bandwidth, and programmers abundant to DoD developers and users.
D&D	Use validated software development platforms that permit continuous integration & delivery evaluation (DevSecOps platform).
Visits	Separate development of mission-level software from development of IA-accredited platforms.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

T&E	Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio.

Related recommendations from previous studies

DSB87	Rec 16: All methodological efforts, especially STARS, should look to see how commercially available software tools can be selected and standardized for DoD needs.
SEI01	Infrastructure: In distributed development activities, get high-quality, secure broadband communications between sites. It is an enabler, not a cost.

Primary Recommendation B2
Automated Testing and Evaluation

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis.**
Stakeholders	DOT&E, USD(A&S), DDR&E(AC), SAE, Service Test Agencies
Background	To deliver SW at speed, rigorous, automated testing processes and workflows are essential. Current DoD practices and procedures often see OT&E as a tailgate process, sequentially after development has completed, slowing down delivery of useful software to the field and leaving existing (potentially poorly performing and/or vulnerable) software in place.
Desired State	Development systems, infrastructure, and practices are focused on continuous, automated testing by developers (with users) with frequency dependent on type of software, but targets cycle times measured in weeks. To the maximum extent possible, system operational testing is integrated (and automated) as part of the development cycle using data, information, and test protocols delivered as part of the development environment. Embedded software in safety-critical systems is tested with high confidence in representative (physical and simulated) environments. Testing and evaluation/certification of COTS components is done once (if justified), and then ATO reciprocity (Rec B3) is applied to enable use in other programs, as appropriate. System-level testing using modeling and simulation ("digital twin") is routinely used.
Role of Congress	DOT&E should provide annual reports to Congress that describe the availability, scale, use, and effectiveness of automated T&E, with the expectation that level/depth of testing will increase at the same time as speed and cycle time are being improved.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
B2.1	Establish procedures for fully automated testing on digital infrastructure (Rec B1), updating DoDI 5129.47 and Service equivalents as needed.	USD(A&S), DOT&E, with Service Testers	Q1 FY20
B2.2	Establish processes for automated and red-team-based security testing, including zero-trust assumptions, penetration testing, and vulnerability scanning.	USD(A&S), DOT&E, with Service Testers	Q1 FY20
B2.3	Identify initial programs to use tools and workflows.	SAE	Q1 FY20
B2.4	Implement minimum viable product (MVP) tools and workflows on digital infrastructure (Rec B1).	SAE, DOT&E, with PMOs	Q2 FY20
B2.5	Migrate initial programs to digital infrastructure using automated T&E.	PEO, with Responsible Organizations	Q3 FY20
B2.6	Use tools and workflows, identify lessons learned and improvements (using DevSecOps iterative approach).	Service Testers, with PEO/PM	Q4 FY20

B2.7	Modify tools and workflows; document procedures.	Responsible Organizations, Service Testers	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years.
D&D	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing and security left.
Visits	Automate testing of software to enable critical updates to be deployed in days to weeks, not months or years (also requires changes in testing organization).
Visits	Add testing as a service.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	DOT&E should use test data collected through existing test methodologies present in software-intensive programs and not recommend or prescribe additional independent, one-time test events.
Acq	One-time IOT&Es or cybersecurity test events should not be recommended for software-intensive systems except in specific circumstances if warranted.
T&E	Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio.
T&E	DoD should expand DOT&E’s current capability to obtain state-of-the-art cyber capabilities on a fee-for-service basis.

Related recommendations from previous studies

DSB87	Rec 27: Each Service should provide its software Using Commands with facilities to do comprehensive operational testing and life-cycle evaluation of extensions and changes.
SEI12	Merge agile and security best practices (e.g., integrate vulnerability scans into continuous integration process, leverage automated test cases for accreditation validation, adhere to secure coding standards).
SEI16	Employ concurrent testing and continuous integration.
USDS	When issuing a solicitation, it should explain the agile software development process. The solicitation should also describe the required testing of functional requirements and make it clear that testing should be integrated into each sprint cycle.
IDA18a	Analysis of planned operational test lengths indicates that the test scope is generally not long enough, demonstrate operational reliability with statistical confidence.

Primary Recommendation B3
ATO Reciprocity

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Create a mechanism for Authorization to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components, and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapon) systems, and Services.**
Stakeholders	DoD CIO, A&S, Service CIOs, DISA
Background	Current software acquisition practice emphasizes the differences among programs: perceptions around different missions, different threats, and different levels of risk tolerance mean that components, tools, and infrastructure that have been given permission to be used in one context are rarely accepted for use in another. The lack of ATO reciprocity drives each program to create their own infrastructure, repeating time- and effort-intensive activities needed to certify elements as secure for their own specific context.
Desired State	Modern software components, tools, and infrastructure, once accredited as secure within DoD, can be used appropriately and cost-effectively by multiple programs. Programs can spend a greater percentage of their budgets on developing software that adds value to the mission rather than spending time and effort on basic software infrastructure. Accreditation of COTS components is done once and then made available for use in other programs, as appropriate.
Role of Congress	N/A

Draft Implementation Plan		**Lead Stakeholder**	**Target Date**
B3.1	Issue guidance making reciprocity the default practice in DoD with limited exceptions and update DoDI 8510.01 to reflect updated risk management framework. Exceptions should require signoff by the DoD CIO to discourage their use.	DoD CIO, with Service CIOs	Q3 FY19
B3.2	Establish DoD-wide repository for ATO artifacts with tools and access rules that enable Services to identify existing ATOs and utilize them when possible.	DoD CIO, with Service CIOs, DISA	Q4 FY19
B3.3	Implement procedures and access controls so that Authorizing Officials have visibility over other programs that are using compatible ATOs.	DoD CIO, with Service CIOs, DISA	Q2 FY20
B3.4	Implement mechanisms to allow FedRAMP and other non-DoD security certifications to be used for DoD ATO when appropriate based on intended use and environment.	DoD CIO, with FedRAMP	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Sec	As security is “baked in” to software during the development process, people must be educated about what that means as different tools look at different security aspects.
Sec	People must learn to appreciate that speed helps increase security. Security is improved when changes and updates can be made quickly to an application. Using automation, software can be reviewed quickly.
Sec	The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product.

Related recommendations from previous studies

SEI12	Define criteria for reaccreditation early in the project.
SEI12	Leverage long accreditation approval wait time with frequent community previews.
SEI12	Don’t apply all the information assurance controls blindly.

Additional Recommendation B4
Prioritize Modern Software Development Methods

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Prioritize secure, iterative, collaborative development for selection and execution of new software development programs (and software components of hardware programs), especially those using commodity hardware and operating systems.**
Stakeholders	USD(A&S), USD(C) DOT&E, SAE, Service Test Agencies
Background	Despite 37+ years of recommendations to stop using waterfall development for software programs, DoD continues to make use of hardware-centric approaches to development for software and software-intensive programs. While portions of the DoD 5000.02 Instructions apply to "Defense Unique Software Intensive" programs and "Incrementally Deployed Software Intensive" programs, these are still waterfall processes with years between the cycles of deployments (instead of weeks). These processes may be appropriate for some (though not all) embedded systems, but they are not the right approach for DoD-specific software running on commercial hardware and operating systems.
Desired State	DoD makes use of commercial software (without customization) whenever possible. When DoD-specific software development is required, contractors with demonstrated ability in the implementation of modern software development processes (e.g., Agile, DevOps, DevSecOps) are prioritized in the selection process and a contract structure is used that enables those methods to be successfully applied. For those applications for which hardware and software development are closely coupled, modern methods are still used as appropriate, especially in terms of information assurance testing.
Role of Congress	Congress should review metrics for performance on software (and software-intensive) programs with the expectation that modern methods of software are able to deliver software to the field quickly, provide rapid and continuous updates of capability, perform extensive automated testing, and track metrics for speed and cycle time, security, code quality, and useful capability.

Draft Implementation Plan		Lead Stakeholders	Target Date
B4.1	Establish metrics for evaluation of software development environments, following DSB 2018 recommendations on software factors and the DIB's "Development Environment" and "Agile BS Detector" concept papers.	USD(A&S)	Q3 FY19
B4.2	Issue Directive-Type Memorandum (DTM) to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q3 FY19
B4.3	Create new DoD Instruction (DoDI) 5000.SW (or update DoDI 5000.02 and 5000.75) to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q1 FY20

B4.4	Update courseware at Defense Acquisition University to specify DoD's default software development approach is secure, iterative, modular, and collaborative.	USD(A&S)	Q2 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
D&D	Require developers to meet with end users, then start small and iterate to quickly deliver useful code.
Visits	Adopt a DevOps culture: design, implement, test, deploy, evaluate, repeat.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Use collaborative tools and libraries so that all content is available to all parties at all times.
Con	Use an agile process to manage structure and technical requirements.
Sec	As security is "baked in" to software during the development process, people must be educated about what that means as different tools look at different security aspects.
Wkf	Incentivize defense contractors to demonstrate their ability to leverage modern software methodologies.
Wkf	Contractor Reform. Adjust future NDAA's to add incentives for defense contractors to use modern development practices. (See FY18NDAA / §§873 & 874)

Related recommendations from previous studies

DSB87	Rec 12: Use evolutionary acquisition, including simulation and prototyping, as discussed elsewhere in this report, to reduce risk.
DSB87	Rec 17: DoD should devise increased productivity incentives for custom-built software contracts and make such incentivized contracts the standard practice.
DSB87	Rec 18: DoD should devise increased profit incentives on software quality.
DSB87	Rec 23: The USD(A) should update DoD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," so that it mandates the iterative setting of specifications, the rapid prototyping of specified systems, and incremental development.
DSB87	Rec 24: DoD STD 2167 should be further revised to remove any remaining dependency on the assumptions of the "waterfall" model and to institutionalize rapid prototyping and incremental development.
DSB87	Rec 29: The USD(A) should develop economic incentives, to be incorporated into standard contracts, to allow contractors to profit from offering modules for reuse, even though built with DoD funds.
DSB87	Rec 30: The USD(A) should develop economic incentives, to be incorporated into all cost-plus standard contracts, to encourage contractors to buy modules and use them rather than build new ones.
DSB87	Rec 31: The USD(A) and ASD(Comptroller) should direct Program Managers to identify in their programs those systems, components, and perhaps even modules that may be expected to be acquired rather than built, and to reward such acquisition in the RFPs.
SEI12	Make sure Agile project teams understand the intent behind security requirements and organize the backlog accordingly.
SEI12	Ensure agile development processes produce and maintain "just enough" design documentation.

SEI12	Make sure there is at least one person with strong security analysis expertise on the Agile project team.
SEI12	Foster Agile project team and accrediting authority collaboration.
SEI12	Leverage unclassified environments for agile development and community previews.
SEI12	Agile and the information assurance community must join forces to continue improving information assurance processes.
GAO16a	Establish a department policy and process for the certification of major IT investments' adequate use of incremental development, in accordance with OMB's guidance on the implementation of FITARA.
NPS16a	Systems leveraging open architectures and incremental designs can focus on delivering initial capability quickly and then iterate improvements over time. The DoD can tailor acquisition processes for each major type of system to streamline each program's path through focused guidance.
SEI16	Ensure that the RFP contains language that allows the use of Agile. One promising approach that is consistent with Agile is to make sure the original contract is written with Agile in mind and contains sufficient flexibility to permit a wide scope of activity that could be modified as the situation develops. Agile program managers (PMs) could establish contract vehicles that allow for collaborative discussions to resolve and address dynamic developments over the life of the effort.
DSB18	Requests for proposals (RFPs) for acquisition programs entering risk reduction and full development should specify the basic elements of the software framework supporting the software factory, including code and document repositories, test infrastructure, software tools, check-in notes, code provenance, and reference and working documents informing development, test, and deployment.
DSB18	Rec 1: A key evaluation criterion in the source selection process should be the efficacy of the offeror's software factory.
DSB18	Rec 1a: Establish a common list of source selection criteria for evaluating software factories for use throughout the Department.
DSB18	Rec 1b: Competing contractors should have to demonstrate at least a pass-fail ability to construct a software factory.
DSB18	Rec 1c: Criteria for evaluating software factories should be reviewed and updated every five years.
DSB18	Rec 5e: Defense prime contractors must build internal competencies in modern software methodologies.
DSB18	Rec 2: The DoD and its defense industrial base partners should adopt continuous iterative development best practices for software, including through sustainment.
DSB18	Rec 2c: [DoD should] engage Congress to change statutes to transition Configuration Steering Boards (CSB) to support rapid iterative approaches (Fiscal Year (FY) 2009 National Defense Authorization Act (NDAA), Section 814).
DSB18	Rec 2d: [DoD] should require all programs entering Milestone B to implement these iterative processes for Acquisition Category (ACAT) I, II, and III programs.
DSB18	Rec 4a: For ongoing development programs, the USD(A&S) should immediately task the PMs with the PEOs for current programs to plan transition to a software factory and continuous iterative development.
DSB18	Rec 4c: Defense prime contractors should incorporate continuous iterative development into a long-term sustainment plan.
DSB18	Establish a common list of source selection criteria for evaluating software factories for use

	throughout the Department.
FCW18	Contractors would allow government to develop past performance reports with less documentation and less contractor opportunity to appeal their ratings.
USDS	Agile software development is the preferred methodology for software development contracts that contribute to the creation and maintenance of digital services, whether they are websites, mobile applications, or other digital channels.
USDS	Although Part 39 does not directly speak to agile software development practices, it endorses modular contracting principles where information technology systems are acquired in successive, interoperable increments to reduce overall risk and support rapid delivery of incremental new functionality.
USDS	With agile software development, requirements and priorities are captured in a high-level Product Vision, which establishes a high-level definition of the scope of the project, specifies expected outcomes, and produces high-level budgetary estimates.
USDS	Under agile software development, the Government retains the responsibility for making decisions and managing the process; it plays a critical role in the IPT as the Product Owner by approving the specific plans for each iteration, establishing the priorities, approving the overall plan revisions reflecting the experience from completed iterations, and approving deliverables.
USDS	OMB’s 2012 Contracting Guidance to Support Modular Development states that IDIQ contracts may be especially suitable for agile software development because they provide a high level of acquisition responsiveness, provide flexibility, and accommodate the full spectrum of the system life cycle that provides both development and operational products and services. BPAs may work with agile software development using modular contracting methods. Additionally, stand-alone contracts or single-award contracts may be used.
USDS	The Agile process works only if there are appropriate dedicated resources, as the process can be labor intensive. Agencies need to ensure adequate resources are applied to manage their contracts irrespective of the strategy used. Strong contract management ensures projects stay on course and helps prevent the agency from becoming overly reliant on contractors.

Additional Recommendation B5
Cloud Computing

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Remove obstacles to DoD usage of cloud computing on commercial platforms, including DISA CAP limits, lack of ATO reciprocity, and access to modern software development tools.**
Stakeholders	DoD CIO, Service CIOs, USD(A&S)
Background	Lack of ATO reciprocity and current DoD procedures for cloud are obstacles to leveraging modern infrastructure and tools.
Desired State	DoD developers and contractors are able to use modern cloud computing environments and commercial development tools quickly, with a single certification that is transferable to other groups using the same environment and tools.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
B5.1	Rescind Cloud Access Point (CAP) policy and replace with policy that ensures security at scale (including end-to-end encryption).	DoD CIO	Q3 FY19
B5.2	In conjunction with primary Rec B3, allow transfer of ATOs for commercial platforms between programs and Services.	DoD CIO	Q3 FY19
B5.3	Create specifications and certification process for approval of standard development tools (w/ ATO reciprocity).	DoD CIO	Q4 FY19
B5.4	In conjunction with Rec B1, establish a common, enterprise ability to develop software solutions in the "easy-to-acquire-and-provision" cloud that is fully accredited by design of the process, tools, and pipeline.	USD(A&S)	Q1 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Include an approach for enterprise-level DevSecOps and other centralized infrastructure development and management, approach for shared services, and applications management.
Inf	Establish a DoD enterprise ability to procure, provision, pay for, and use cloud that is no different from the commercial entry points for cloud computing.
Inf	DoD should establish a common, enterprise ability to develop software solutions in the "easy-to-acquire-and-provision" cloud that is fully accredited by design of the process, tools, and pipeline.

Related recommendations from previous studies

Sec809	Rec. 43: Revise acquisition regulations to enable more flexible and effective procurement of consumption-based solutions.

Additional Recommendation B6
Certify Code/Toolchain

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Shift from certification of executables for low- and medium-risk deployments to certification of code/architectures and certification of the development, integration, and deployment toolchain.**
Stakeholders	USD(A&S), SAE, DoD CIO, Service CIO
Background	Today, the typical focus of security accreditation on programs is to certify each version of the code that is intended for release. This works against the goal of frequent updates because the more versions of software that are created, the more often the time and expense of the certification have to be borne by the program.
Desired State	The Department will accredit software infrastructures that are capable of producing quality code when used appropriately, enabling each version of the code produced on that infrastructure to be treated as certifiably secure (within appropriate limits, e.g., for versions that do not entail major architectural changes). With this change in certification, DoD will enable rapid fielding of mission-critical code at high levels of information assurance.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
B6.1	Identify and use commercial certification procedures for security assessments and deployment mechanisms that can be used for DoD software programs.	CIO	Q4 FY19
B6.2	Identify three lead programs for initial implementation of certification procedures.	A&S, SAE	Q1 FY20
B6.3	Expand certification procedures to 10 additional sites, spanning all Services and multiple OSD offices; update procedures with each new certification to streamline process.	A&S, SAE with CIO	Q3 FY20
B6.4	Update DoDI 8501.01, Risk Management Framework for DoD Information Technology, to reflect revised certification procedures.	CIO with SAE, A&S	Q4 FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Exempt the DoD from the Clinger Cohen Act, 40 U.S.C. 1401(3)
Inf	DoD should establish a common, enterprise ability to develop software solutions in the “easy-to-acquire-and-provision” cloud that is fully accredited by design of the process, tools, and pipeline.

Related recommendations from previous studies

SEI12	Use common operating environment (COE), software development toolkits (SDKs), and enterprise services to speed up accreditation time.

SEI12	Apply a risk-based, incremental approach to security architecture.
SEI12	Leverage design tactics such as layering and encapsulation to limit impact of change.
SEI13	For an SoS or for the more likely case of a system or component that participates in an existing SoS, an effective risk management approach should: • scale to size and complexity of systems of systems • incorporate dynamics • integrate across full life cycle: requirements to sustainment • focus on success as well as failure

Additional Recommendation B7
Hardware as a Consumable

Line of Effort	Create and maintain cross-program/cross-service digital infrastructure.
Recommendation	**Plan and fund computing hardware (of all appropriate types) as consumable resources, with continuous refresh and upgrades to current, secure operating systems and platform components.**
Stakeholders	USD(A&S), SAE, DoD CIO, Service CIO, USD(C), CAPE
Background	Current information technology (IT) refreshes take 8-10 years from planning to implementation, which means that most of the time our systems are running on obsolete hardware that limits our ability to implement the algorithms required to provide the level of performance needed to stay ahead of our adversaries. Maintaining legacy code for different variants that have hardware capabilities ranging from 2 to 12 years old is an almost impossibly large spread of capability in computing, storage, and communications. From a contracting perspective, this change would require DoD to provide a stable annual budget that paid for new hardware and software capability (see Commandment #3), but this would very likely save money over the longer term.
Desired State	Whenever possible, applications are run in the cloud, so that algorithms can be run on the latest hardware and operating systems. For weapons systems, a continuous hardware refresh mentality is in place that enables software upgrades, crypto updates, and connectivity upgrades to be rapidly deployed across a fleet on an ongoing basis. The adoption rate of the latest hardware and operating system versions is tracked and targets are set for maintaining hardware and operating system "readiness." The paradigm for computing hardware from current Property, Plant, and Equipment categorization (as investments with depreciation schedules) is modified to treat hardware as an expense.
Role of Congress	Provide funding for ongoing replacement of computing hardware as a consumable with a 2–4-year lifetime. Track "readiness" of currently deployed software capability in part by measuring age of the hardware and operating systems on which software is being run.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
B7.1	Establish funds for initial existing weapons platforms involving computing hardware to replace hardware every 2–4 years (like oil).	CIO with USD(C), SAE	Q1 FY20
B7.2	Establish draft guidance for determining when to update hardware and operating systems to balance cost with risk/capability.	CIO	Q2 FY20
B7.3	Work with FASAB to change audit treatment of software/IT with these goals: (1) Separate category for software instead of being characterized as Property, Plant, and Equipment; (2) Default setting that software is an expense, not	USD(A&S), in coordination with USD(C)	Q4 FY20

	an investment; and (3) there is no “sustainment” phase for software.		
B7.4	Modify DoD Financial Management Regulation (FMR) to capture changes in how hardware is purchased and retired from service.	USD(C)	Q1 FY21

SWAP concept paper recommendations related to this recommendation

10C	Move to a model of continuous hardware refresh in which computers are treated as a consumable with a 2-3 year lifetime.
Visits	Make use of platforms (hardware and software) that continuously evolve at the timescales of the commercial sector (3-5 years between HW/OS updates).

Related recommendations from previous studies

Sec809	Rec. 44: Exempt DoD from Clinger–Cohen Act Provisions in Title 40:
Sec809	Rec. 56: Use authority in Section 1077 of the FY 2018 NDAA to establish a revolving fund for information technology modernization projects and explore the feasibility of using revolving funds for other money-saving investments.

Primary Recommendation C1
Organic Development Groups

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices.**
Stakeholders	USD(A&S), USD(P&R), SAE, Service HR
Background	DoD's capacity to apply modern technology and software practices to meet its mission is required in order to remain relevant in increasingly technical fighting domains, especially against peer adversaries. While DoD has both military and civilian software engineers (often associated with maintenance activities), the IT career field suffers from a lack of visibility and support. The Department has not prioritized a viable recruiting strategy for technical positions, and there is no comprehensive training or development program that prepares the technical and acquisition workforce to adequately deploy modern software development tools and methodologies.
Desired State	DoD recruits, trains, and retains internal capability for software development, including by service members, and maintains this as a separate career track (like DoD doctors, lawyers, and musicians). Each Service has organic development units that are able to create software for specific needs and that serve as an entry point for software development capability in military and civilian roles (complementing work done by contractors). The Department's workforce embraces commercial best practices for the rapid recruitment of talented professionals, including the ability to onboard quickly and provide modern tools and training in state-of-the-art training environments. Individuals in software development career paths are able to maintain their technical skills and take on DoD leadership roles.
Role of Congress	Congress should receive regular "readiness" reports that include organic software development capability and provide budget required to maintain desired capability level and resources for modern software development.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
C1.1	Exercise existing acquisition and cybersecurity hiring authorities to increase the number of software developers in DoD programs with vacant positions.	SAE, PEO, with CIO (cyber excepted service ability)	Immediately
C1.2	Create new military occupational specialty (MOS) and core occupational series plus corresponding career tracks for each Service; use to grow digital talent for modern software development (e.g., Agile, DevSecOps).	J1 and comparable X1 for each Service with USD(P&R)	Q1 FY20
C1.3	Create regulations to allow standard identification, recruitment, and onboarding of experienced civilian software talent, especially on rotation from private sector roles.	USD(P&R)	Q1 FY20

C1.4	Create mechanism for tracking software development expertise and use as preferred experience for promotion into software engineer and acquisition roles.	A&S, CIO	Q2 FY20
C1.5	Obtain additional manpower authorizations for military and civilian SW developers.	USD(A&S), with USD(P&R), SAE	FY20, FY21
C1.6	Stand up one or more software factories within each Service, tied to field needs that can be satisfied through organic software development groups.	SAEs, with PEOs Digital	FY20 (create), FY21 (scale)

SWAP concept paper recommendations related to this recommendation

10C	Establish Computer Science as a DoD core competency.
D&D	Hire competent people with appropriate expertise in software to implement the desired state and give them the freedom to do so ("competence trumps process").

SWAP working group inputs (reflected in Appendix F) related to this recommendation

M&S	The definition of "core capabilities" in 10 USC 2464 should be revisited in light of warfighter dependence on software-intensive systems to determine the scope of DoD's core organic software engineering capability, and we should engage with Congress on the proposed revision to clarify the intent and extent of key terminology used in the current statute.
M&S	Revise industrial base policy to include software and DoD's organic software engineering capabilities and infrastructure. Start enterprise planning and investment to establish and modernize organic System Integration Labs (SILs), software engineering environments, and technical infrastructure; invest in R&D to advance organic software engineering infrastructure capabilities.
Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Overhaul the recruiting and hiring process to use simple position descriptions, fully leverage hiring authorities, engage subject matter experts as reviewers, and streamline the onboarding process to take weeks instead of months.
Wkf	Embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds that may require special authorities to join the Department.
Wkf	Develop a strategic recruitment program that targets civilians, similar to the recruitment strategy for military members, [including] prioritizing experience and skills over cookie-cutter commercial certifications or educational attainment.
Wkf	The Department should incentivize and provide software practitioners access to modern engagement and collaboration platforms to connect, share their skills and knowledge, and develop solutions leveraging the full enterprise.
Wkf	Allow for greater private-public sector fluidity across the workforce while empowering the existing workforce to create a place where they want to work.
Wkf	Modify Title 10, §1596a to create a new Computer-language proficiency pay statute.
Wkf	Pilot a cyber-hiring team with the necessary authorities to execute report recommendations and that can serve as a Department-wide alternative to organization's traditional HR offices and will provide expedited hiring and a better candidate experience for top-tier cyber positions.

Related recommendations from previous studies

DSB87	Rec 26: Each Service should provide its software Product Development Division with the ability

	to do rapid prototyping in conjunction with users.
DSB87	Rec 36: Establish mechanisms for tracking personnel skills and projecting personnel needs.
DSB87	Rec 37: Structure some office careers to build a cadre of technical managers with deep technical mastery and broad operational overview.
SEI10	Improve compensation and advancement opportunities to increase tenure.

Primary Recommendation C2
Acquisition Workforce Training

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., Agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software.**
Stakeholders	USD(A&S), DoD CIO, SAE, Service CIO
Background	Acquisition professionals have been trained and had success in the current model, which has produced the world's best military, but this model is not serving well for software. New methodologies and approaches introduce unknown risks, and acquisition professionals are often not incentivized to make use of the authorities available to implement modern software methods. At the same time, senior leaders in DoD need to be more knowledgeable about modern software development practices so they can recognize, encourage, and champion efforts to implement modern approaches to software program management.
Desired State	Senior leaders, middle management, and organic and contractor-based software developers are aligned in their view of how modern software is procured and developed. Acquisition professionals are aware of all of the authorities available for software programs and use them to provide flexibility and rapid delivery of capability to the field. Program leaders are able to assess the status of software (and software-intensive) programs and spot problems early in the development process, as well as provide continuous insight to senior leadership and Congress. Highly specialized requirements are scrutinized to avoid developing custom software when commercial offerings are available that are less expensive and more capable.
Role of Congress	Prioritize experience with modern software development environments in approval of senior acquisition leaders.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
C2.1	Leverage existing training venues to add content about modern software development practices.	USD(A&S), SAEs with DAU	Q4 FY19
C2.2	Create and provide training opportunities via boot camps and rotations for acquisition professionals to obtain hands-on experience in DevSecOps programs.	A&S with SAEs, USD(P&R)	FY20 (MVP)[2] FY21 (scale)
C2.3	Develop additional training opportunities for key leaders about modern software development practices.	USD(A&S), SAE, DAU	Q2 FY20
C2.4	Create software continuing education programs and requirements for CIOs, SAEs, PEOs, and PMs, modeled after MCLE (Minimum Continuing Legal Education) for lawyers.	A&S, DAU	Q3 FY20

[2] Minimum viable product (first useful iteration)

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Provide training to KOs, PMs, and leadership to understand the value and methods associated with Agile and modular implementation.
Wkf	Create a software acquisition workforce fund (similar to the existing Defense Acquisition Workforce Development Fund (DAWDF)) ... to hire and train a cadre of modern software acquisition experts.
Wkf	Pilot development programs that provide comprehensive training for all software acquisition professionals, developers, and associated functions.
Con	Provide training to KOs, PMs, and leadership to understand the value and methods associated with Agile and modular implementation.
Con	Educate PMs and KOs on Open Source, proprietary, and government-funded code.

Related recommendations from previous studies

DSB09	All CIOs should approve IT acquisition program manager training and certification and advise the personnel selection process.
DSB09	The USD(AT&L) shall direct the Defense Acquisition University, in coordination with the Information Resources Management College, to integrate the new acquisition model into their curriculum.
DSB18	USD(A&S) should task the PMs of programs that have transitioned successfully to modern software development practices to brief best practices and lessons learned across the Services.
DSB18	Rec 5d: The USD(A&S) and the USD(R&E) should direct the Defense Acquisition University (DAU) to establish curricula addressing modern software practices leveraging expertise from the DDS, the FFRDCs, and the University Affiliated Research Centers (UARCs).
DSB18	Rec 5g: DoD career functional Integrated Product Team (IPT) leads should immediately establish a special software acquisition workforce fund modeled after the Defense Acquisition Workforce Development Fund (DAWDF), the purpose of which is to hire and train a cadre of modern software acquisition experts across the Services.
DSB18	Rec 5h: PMs should create an iterative development IPT with associated training. The Service Chiefs should delegate the role of Product Manager to these IPTs.
DSB18	Rec 5b: The Service Acquisition Career Managers should develop a training curriculum to create and train [a] cadre [of] software-informed PMs, sustainers, and software acquisition specialists.
Sec809	Rec 27: Improve resourcing, allocation, and management of the Defense Acquisition Workforce Development Fund (DAWDF).
Sec809	Rec. 59: Revise the Defense Acquisition Workforce Improvement Act to focus more on building professional qualifications.

Additional Recommendation C3
Increase PMO Experience

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Increase the knowledge, expertise, and flexibility in program offices related to modern software development practices to improve the ability of program offices to take advantage of software-centric approaches to acquisition.**
Stakeholders	USD(A&S), SAE, USD(P&R)
Background	Acquisition professionals do not always have experience and insights into modern software development environments, especially in the opportunities (and limitations) for continuous integration/continuous delivery (CI/CD), automated testing (including security testing), and modern cloud-computing architectures. New methodologies and approaches introduce unknown risks, while the old acquisition and development approaches built the world's best military. Program offices are not incentivized to adopt new approaches to acquisition and implementation of software, and inertia represents a barrier to change.
Desired State	Program management offices have staff available with experience in modern software development environments and who are able to make creative (but legal) use of available authorities for acquisition of software to fit the needs of modern software development solutions. Management of most types of software relies on (continuous) measurement of capability delivered to the field rather than being tied to satisfaction of objectives. Time and cost are used as constraints with schedule of delivery of features replanned at each iteration cycle based on warfighter/user feedback.
Role of Congress	N/A

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
C3.1	Establish list of skills and experience needed by program office staff to be considered "fully staffed" for a software program.	A&S with SAEs, USD(P&R)	Q4 FY19
C3.2	Modify Position Descriptions for those in leadership positions in software acquisition programs to prioritize and reward prior experience in software development.	USD(A&S), SAE, Service HR	Q1 FY20
C3.3	Create and provide training opportunities via boot camps and rotations for acquisition professionals to obtain hands-on experience in DevSecOps programs.	A&S with SAEs, USD(P&R)	Q2 FY20 (MVP)[3] FY21 (scale)
C3.4	Modify PM training requirements to obtain DAU Level IIII certification to include hands-on experience with modern software development.	USD(A&S), DAU	Q3 FY20
C3.5	Evaluate readiness level of software (and software-intensive) program offices by comparing experience/skill sets available with the list of needed skills from C3.1	A&S with SAEs, USD(P&R)	Q4 FY20 (MVP) FY21 (scale)

[3] Minimum viable product (first useful iteration)

	(hint: consider tracking those skills sets; see Action C1.2).		

SWAP concept paper recommendations related to this recommendation

D&D	Hire competent people with appropriate expertise in software to implement the desired state and give them the freedom to do so ("competence trumps process").

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Acq	Lead tester from either DOT&E or JITC (preferably both, if JITC is being used as test org) must be a subject matter expert in the subject being tested, similar to how qualified test pilots run test flights (health records, financial systems, etc.).
Wkf	Empower a small cadre of Highly Qualified Experts (HQEs) and innovative Department employees to execute the changes from this report.
Wkf	Establish a software acquisition workforce fund, similar to the Defense Acquisition Workforce Development Fund (DAWDF), but the primary use will be for hiring and training a cadre of modern software acquisition experts.
Wkf	Provide Agile, Tech, and DevSecOps coaches in Program Offices to support transformations, adoption of modern software practices, and share lessons across the enterprise.
Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Modify the existing language in 5 USC Part III, Subpart D, Chapter 53 to add a pilot training program for all software acquisition professionals, developers, and associated functions.
Wkf	Modify Title 10 §1746 to include authorities for the development of a modern academy under the Defense Acquisition University; the HQE cadre (see above) should lead its development. Note: Tied with FY18 NDAA §891 (training on agile and iterative development methods.)
Wkf	Modify Title 5, §§3371-3375 to expand the Inter-Government Personnel Act and allow more civil service employees to work with non-Federal Agencies and Educational Institutions. In addition, modify Title 10, §1599g to expand the Public-Private Talent Exchange Program and modify the language to reduce the "repayment" period from 1:2 to 1:1 ratio.

Related recommendations from previous studies

OSD06	Establish a consistent definition of the acquisition workforce with the Under Secretary of Defense for Acquisition Technology and Logistics, working with the Service Secretaries to include in that definition all acquisition-related budget and requirements personnel.
OSD06	Immediately increase the number of federal employees focused on critical skill areas, such as program management, system engineering, and contracting. The cost of this increase should be offset by reductions in funding for contractor support.
OSD06	Request that the White House Liaison Office create a pool of acquisition-qualified, White House pre-cleared, non-career senior executives and political appointees to fill executive positions, to provide leadership stability in the Acquisition System.
OSD06	Seek legislation to retain high-performance military personnel in the acquisition workforce to include allowing military personnel to remain in uniform past the limitations imposed by the Defense Officer Personnel Management Act and augment

	their pay to offset the “declining marginal return” associated with retired pay entitlement.
OSD06	Realign responsibility, authority, and accountability at the lowest practical level of authority by reintegrating the Services into the acquisition management structure.
OSD06	Fully implement the intent of the Packard Commission. Create a streamlined acquisition organization with accountability assigned and enforced at each level.
SEI10	Assign PMs, DPMs, and other key positions for the program’s duration and into deployment. Use civilians if military rotations are not amenable.
SEI10	Improve qualifications of acquisition staff, emphasizing software expertise.
CSIS15	Rapid acquisition succeeds when senior leaders are involved in ensuring that programs are able to overcome the inevitable hurdles that arise during acquisition and empower those responsible with achieving the right outcome with the authority to get the job done while minimizing the layers in between.
CSIS15	Rapid acquisition is fundamentally an ongoing dialogue between the acquisition and operational communities about what the real needs of the warfighter are and what the art of the possible is in addressing them.
SEI15	5. Government Personnel Experience. Government personnel with extensive experience in developing and managing acquisition strategy and technical architecture should be dedicated and available to a program throughout its duration.
NPS16a	The growth of rapid acquisition organizations gives acquisition executives new avenues to meet their top priority and rapid capability demands. However, these organizations may also have negative effects on traditional acquisition organizations. The DoD’s top talent will flock to the rapid acquisition organizations so that they can work on high-priority programs with minimal restrictions and likely achieve greater success.
NPS16a	Contracting Officers (COs) must function as strategic partners tightly integrated into the program office, rather than operate as a separate organization that simply processes the contract paperwork.
NPS16b	Culturally, the acquisition community needs to embrace the available tools as opportunities, while being selective with procurement methods and adaptive to the market environment.
GAO17	Empower program managers to make decisions on the direction of the program and to resolve problems and implement solutions.
GAO17	Hold program managers accountable for their choices.
GAO17	Require program managers to stay with a project to its end.
GAO17	Encourage program managers to share bad news, and encourage collaboration and communication.
DSB18	Rec 5a: The service acquisition commands (e.g., the LCMC, the NAVAIR, the U.S. Naval Sea Systems Command (NAVSEA), and the AMC) need to develop workforce competency and a deep familiarity of current software development techniques.
DSB18	Rec 5a.2: Services acquisition commands should use this cadre early in the acquisition process to formulate acquisition strategy, develop source selection criteria, and evaluate progress.
DSB18	Over the next two years, the service acquisition commands need to develop workforce competency and a deep familiarity of current software development techniques.

Sec809	Rec. 40: Professionalize the requirements management workforce.
Sec809	Rec. 46: Empower the acquisition community by delegating below-threshold reprogramming decision authority to portfolio acquisition executives.

Additional Recommendation C4
Recruiting (Transient) Digital Talent

Line of Effort	Create new paths for digital talent (especially internal talent).
Recommendation	**Restructure the approach to recruiting digital talent to assume that the average tenure of a talented engineer will be 2-4 years, and make better use of HQEs, IPAs, special hiring authorities, reservists, and enlisted personnel to provide organic software development capability, while at the same time incentivizing and rewarding internal talent.**
Stakeholders	USD(A&S), USD(P&R), SAE, A-1/G-1/N-1
Background	Current DoD personnel systems assume that military and government employees will "grow through the ranks" and that individuals will stay in government service for long periods of time. The attractions of the private sector create personnel-retention challenges that are not likely to be overcome, so a different approach is needed.
Desired State	DoD leverages all individuals who are willing to serve, whether for a long period or a short period, and amplifies the ability of individuals to make a contribution during their time in government. Internal talent is recognized and retained through merit-based systems of promotion and job assignment.
Role of Congress	Support and encourage the use of existing authorities to hire digital talent in creative ways that match the intent of Congress and solve the need for more flexible arrangements in which talented individuals move in and out of government service (without creating unnecessary barriers).

Draft Implementation Plan		Lead Stakeholders	Target Date
C4.1	Exercise existing hiring authorities to increase the number of highly skilled software people in DoD program, such as the Cyber Excepted Workforce.	SAE, PEO, CIO	Starting now
C4.2	In conjunction with Recs C1, create a database of individuals in enlisted, officer, reserve, and civilian positions with software development skills and experience for internal recruiting use to software squadrons & PAOs.	USD(P&R) and Service equivalents	Q3 FY19
C4.3	Within organic software programs, create processes for maintaining release cadence under the assumption of up to 25% turnover per year.	PMOs	Q4 FY19
C4.4	Require software-intensive project proposals to include a plan for maintaining cadence-related metrics in the face of up to 25% turnover of staff.	SAEs	Q4 FY19
C4.5	Identify bottlenecks in providing security clearances for software developers and target granting of interim clearances within 1 month of start date.	DSS	Q1 FY20
C4.6	Revise GS and military promotion guidelines for software developers to allow rapid promotion of highly qualified individuals with appropriate skills, independent of "time in grade."	USD(P&R)	FY20 for FY21 NDAA

C4.7	Obtain additional funding for military, civilian SW developers, including existing personnel, HQEs, IPAs, reservists, and direct commissioning.	USD(A&S), USD(P&R), SAE	FY21

SWAP concept paper recommendations related to this recommendation

10C	Establish Computer Science as a DoD core competency.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Wkf	Develop a core occupational series based on current core competencies and skills for software acquisition and engineering.
Wkf	Overhaul the recruiting and hiring process to use simple position descriptions, fully leverage hiring authorities, engage subject matter experts as reviewers, and streamline the onboarding process to take weeks instead of months.
Wkf	Embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds that may require special authorities to join the Department.
Wkf	Develop a strategic recruitment program that targets civilians, similar to the recruitment strategy for military members, [including] prioritizing experience and skills over cookie-cutter commercial certifications or educational attainment.

Related recommendations from previous studies

DSB87	Rec 34: Do not believe that DoD can solve its skilled personnel shortage; plan how best to live with it, and how to ameliorate it.
SEI10	Divide large acquisition development efforts into multiple smaller, shorter duration programs.
Sec809	Rec. 45: Create a pilot program for contracting directly with information technology consultants through an online talent marketplace.

Primary Recommendation D1
Source Code Access

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Require access to source code, software frameworks, and development toolchains—with appropriate IP rights—for DoD-specific code, enabling full security testing and rebuilding of binaries from source.**
Stakeholders	USD(A&S), CIO, SAE
Background	For many DoD systems, source code is not available to DoD for inspection and testing, and DoD relies on suppliers to write code for new compute environments. As code ages, suppliers are not required to maintain codebases without an active development contract, and “legacy” code is not continuously migrated to the latest hardware and operating systems.
Desired State	DoD has access to source code for DoD-specific software systems that it operates and uses to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most importantly) upgrades (patches and enhancements). DoD is able to rebuild executables from scratch for all of its systems, and it has the rights and ability to modify (DoD-specific) code when new conditions and features arise. Code is routinely migrated to the latest computing hardware and operating systems and routinely scanned against currently known vulnerabilities. Modern IP language is used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors are able to use licensing agreements that protect any IP that they have developed with their own resources. Industry trusts DoD with its code and has appropriate IP rights for internally developed code.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
D1.1	Work with industry to modernize policies for software code ownership, licensing, and purchase. See 2018 Army IP directive as an example.	USD(A&S)	Q3 FY19
D1.2	Modify FAR/DFARS guidance to require software source code deliverables for GOTS and for government-funded software development. Obtain rights for access to source code for COTS wherever possible (and useful).	USD(A&S)	Q3 FY20
D1.3	Modify DoDI 5000.02 and DoDI 5000.75 to make access to code and development environments the default.	USD(A&S)	Q3 FY20
D1.4	Develop a comprehensive source-code management plan for DoD including the safe and secure storage, access control, testing, and field of use rights.	USD(A&S), with CIO	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Every purpose-built DoD software system should include source code as a deliverable.

D&D	Require source code as a deliverable on all purpose-built DoD software contracts. Continuous development and integration, rather than sustainment, should be a part of all contracts. DoD personnel should be trained to extend the software through source code or API access.

Related recommendations from previous studies

DSB87	Rec 22: DoD should follow the concepts of the proposed FAR 27.4 for data rights for military software, rather than those of the proposed DoD 27.4, or it should adopt a new “Rights in Software” Clause as Recommended by Samuelson, Deasy, and Martin in Appendix A6.
DSB18	Rec 6b: Availability, cost, compatibility, and licensing restrictions of [the proposed software factory] framework elements to the U.S. Government and its contractors should be part of the selection criteria for contract award.
DSB18	Rec 6c: All documentation, test files, coding, application programming interfaces (APIs), design documents, results of fault, performance tests conducted using the framework, and tools developed during the development, as well as the software factory framework, should be delivered to the U.S. Government at each production milestone; OR escrowed and delivered at such times specified by the U.S. Government (i.e., end of production, contract reward).
DSB18	Rec 6d: Selection preference should be granted based on the ability of the United States to reconstitute the software framework and rebuild binaries, re-run tests, procedures, and tools against delivered software and documentation.

Primary Recommendation D2
Security Considerations

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough.**
Stakeholders	USD(A&S), CIO, DDS, SAE, DDR&E(AC), DOT&E
Background	Current DoD systems often rely on security-at-the-perimeter as a means of protecting code for unauthorized access. If this perimeter is breached, then a large array of systems can be compromised. Multiple GAO, DoDIG, and other reports have identified cybersecurity as a major issue in acquisition programs.
Desired State	DoD systems use a zero-trust security model in which it is not assumed that anyone who can gain access to a given network or system should have access to anything within that system. Regular and automated penetration testing is used to track down vulnerabilities, and red teams are engaged to attempt to breach our systems before our adversaries do.
Role of Congress	Review (classified) reporting of vulnerabilities identified in DoD systems and provide the resources required to ensure that hardware and operating systems are at current levels (see Recommendation B7, Hardware as a Consumable).

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
D2.1	Adopt standards for secure software development and testing that use a zero-trust security model.	CIO, with DDS	Q3 FY19
D2.2	Develop, deploy, and require the use of IA-accredited (commercial) development tools for DoD software development.	CIO, PEO Digital	Q4 FY19
D2.3	Establish automated and red-team based penetration testing as part of OT&E evaluation (integrated with program development).	DOT&E	Q1 FY20
D2.4	Establish a red team responsible for ongoing vulnerability testing against any defense software system.	CIO with DDS	Q2 FY20
D2.5	Establish security as part of the selection criteria for software programs.	A&S with CIO, SAEs	Q3 FY20

SWAP concept paper recommendations related to this recommendation

10C	Only run operating systems that are receiving (and utilizing) regular security updates for newly discovered security vulnerabilities.
10C	Data should always be encrypted unless it is part of an active computation.
D&D	Create automated test environments to enable continuous (and secure) integration and deployment to shift testing and security left.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Sec	People must learn to appreciate that speed helps increase security. Security is improved when

	changes and updates can be made quickly to an application. Using automation, software can be reviewed quickly.
Sec	The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product.
T&E	Establish a statutory “Live Fire” requirement on software-intensive systems as there is on “Covered Systems” for protecting our warfighters from kinetic threats. “Shoot at it” before design is complete and certainly before it is put into the operational environment.
T&E	Establish a federation of state-of-the-art cyber testing capabilities from non-profit institutions to support trusted, survivable, and resilient defense systems and ensure the security of software and hardware developed, acquired, maintained, and used by the DoD.
T&E	Establish cybersecurity as the “4th leg” in measurement of Acquisition system/program performance: Cost, Schedule, Performance, Cybersecurity.
T&E	Develop mechanisms to enforce existing software and cybersecurity policies (from cradle-to-grave) that are not (now) being adequately enforced.
T&E	Ensure each DoD Component is responsible for representing its own forces and capabilities in a digital modeling environment (e.g., M&S and digital twin), making them available to all other DoD users, subject to a pre-defined architecture and supporting standards. DIA will represent threat forces and capabilities in a digital form consistent with this architecture/standards. Programs are required to use DIA-supplied threat models, unless sufficient justification is provided to use other.

Related recommendations from previous studies

DSB09	In the Services and agencies, the CIOs should also have strong authorities and responsibilities for system certification, compliance, applications development, and innovation.
DSB09	The DOD CIO, supported by CIOs in the Services and agencies, should be responsible for certifying that systems and capabilities added to the enterprise do not introduce avoidable vulnerabilities that can be exploited by adversaries.
Sec809	Rec. 77: Require role-based planning to prevent unnecessary application of security clearance and investigation requirements to contracts.

Primary Recommendation D3
Software Features

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics to avoid requirements creep, overly ambitious requirements, and program delays.**
Stakeholders	USD(A&S), Joint Staff, SAEs
Background	Current DoD requirements processes significantly impede DoD's ability to implement modern SW development practices by spending years establishing program requirements and insisting on satisfaction of requirements before a project is considered "done." This impedes rapid implementation of features that are of the most use to the user.
Desired state	Rather than a list of requirements for every feature, programs should establish a minimum set of requirements required for initial operation, security, and interoperability and place all other desired features on a list that will be implemented in priority order, with the ability for DoD to redefine priorities on a regular basis.
Role of Congress	Modify relevant statutes to allow the use of evolving features over rigid requirements and develop alternative methods for obtaining information on program status (See Rec A2, Action A2.4).

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
D3.1	Modify requirements guidance by memo to shift from a list of requirements for software to a list of desired features and required interfaces/characteristics.	USD(A&S), with CMO	Q4 FY19
D3.2	Update CJCSI 3170.01H (JCIDS requirements process) to reflect contents of guidance memos.	Joint Staff	Q1 FY20
D3.3	Modify DoDI 5000.02 and DoDI 5000.75 (or integrate into new DoDI 5000.SW).	USD(A&S)	Q2 FY20
D3.4	Define and use new budget exhibits for software programs using evolving lists of features in place of requirements (see also Rec A2).	USD(A&S), with USD(C), CAPE, HAC-D, SAC-D	Q3 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
10C	All software procurement programs should start small, be iterative, and build on success—or be terminated quickly.
D&D	Accept 70% solutions in a short time (months) and add functionality in rapid iterations (weeks).

Related recommendations from previous studies

SEI01	Ensure that all critical functional and interoperability requirements are well specified in the contract (statement of work, Statement of Objectives).
SEI01	Handle requirements that have architectural consequences as systems engineering issues—up front.

SEI12	Ensure requirements prioritization of backlog considers business value and risk.
GAO17	Match requirements to resources—that is, time, money, technology, and people—before undertaking new development efforts.

Additional Recommendation D4
Continuous Metrics

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Create and use automatically generated, continuously available metrics that emphasize speed, cycle time, security, user value, and code quality to assess, manage, and terminate software programs (and software components of hardware programs).**
Stakeholders	USD(A&S), CAPE, USD(C), SAE, Service Cost Orgs
Background	Current program reporting requirements are largely manual and time consuming, and they provide limited insight into the SW health of a program. New metrics are required that match the DevSecOps approach of continuous capability delivery and maintenance and provide continuous insight into program progress.
Desired State	Program oversight will re-focus on the value provided by the software as it is deployed to the warfighter/user, and it will rely more heavily on metrics that can be collect in an automated fashion from instrumentation on the DevSecOps pipeline and other parts of the infrastructure. Specific metrics will depend on the type of software rather than a one-size-fits-all approach.
Role of Congress	N/A (but see Rec A3)

Draft Implementation Plan		Lead Stakeholder	Target Date
D4.1	Modify acquisition policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and useful functionality.	USD(A&S)	Q3 FY19
D4.2	Modify cost estimation policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code.	CAPE	Q3 FY19
D4.3	Develop specific measure of software quality, value, and velocity and the tools to implement the automatic generation and reporting.	DDS, with CAPE, CIO, USD(C)	Q4 FY19
D4.4	Modify DoDI 5000.02, DoDI 5000.75, and DoDI 5105.84 to reflect use of updated methods and remove earned value management (EVM) for software programs.	A&S	Q1 FY20

SWAP working group inputs related to this recommendation

Acq	Revise DFARS Subpart 234.201, DoDI 5000.02 Table 8, and OMB Circular A-11 to remove EVM requirement.
Con	Allow for documentation and reporting substitutions to improve agility (agile reporting vs. EVM) (Cultural and EVM Policy).
Con	Establish a clear definition of done targets for software metrics for defense systems of different types (code coverage, defect rate, user acceptance).
D&M	Congress could establish, via an NDAA provision, new data-driven methods for governance of software development, maintenance, and performance. The new approach should require on-demand access to standard (and perhaps real-time) data with reviews occurring on a standard

	calendar, rather than the current approach of manually developed, periodic reports.
D&M	DoD must establish the data sources, methods, and metrics required for better analysis, insight, and subsequent management of software development activities. This action does not require Congressional action but will likely stall without external intervention and may require explicit and specific Congressional requirements to strategically collect, access, and share data for analysis and decision making.
T&E	Establish requirements for government-owned software to be instrumented such that critical monitoring functions (e.g., performance, security) can be automated as much as possible, persistently available, and such that authoritative data can be captured, stored, and reused in subsequent testing or other analytic efforts.

Related recommendations from previous studies

DSB87	Rec 19: DoD should develop metrics and measuring techniques for software quality and completeness and incorporate these routinely in contracts.
DSB87	Rec 20: DoD should develop metrics to measure implementation progress.
Sec809	Rec 19: Eliminate the Earned Value Management (EVM) mandate for software programs using agile methods.
MITRE18	Elevate Security as a Primary Metric in DoD Acquisition and Sustainment.

Additional Recommendation D5
Iterative Development

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Shift the approach for acquisition and development of software (and software-intensive components of larger programs) to an iterative approach: start small, be iterative, and build on success or be terminated quickly.**
Stakeholders	USD(A&S), CAPE, USD(C), USD(P&R), SAE, Service HR
Background	Current-language DoD acquisition guidance is largely based around a hardware-centric paradigm, with a well-defined start and end and sequential life cycle activities.
Desired State	Software acquisition in DoD follows an iterative approach, with frequent deployment of working software, supported by a DevSecOps infrastructure that enables speed through continuous integration/continuous delivery. Software projects are continuously evaluated by the quality of their deployed capability and are terminated early if they are found to be non-performant. Software is never “complete.” Programs are viewed as an ongoing service rather than a discrete project.
Role of Congress	Authorize and track software programs that utilize iterative methods of development rather than milestone-based progress. Recognizing that the distinction between RTD&E, procurement, and sustainment is not appropriate for many types of software, identify new ways of providing oversight while enabling much more flexibility for programs.

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
D4.1	Issue guidance immediately changing the default for acquisition programs to use iterative software development methodologies (e.g., DevSecOps, agile development).	USD(A&S)	Q3 FY19
D4.2	Issue guidance changing the default for acquisition programs to be iterative software development methodologies.	SAE	Q3 FY19
D4.6	Select three software programs widely perceived to be in dire straits and go through a program termination exercise to identify new potential solutions and the blockers to more effectively terminating non-performing programs.	USD A&S	Q1 FY20
D4.3	Modify DoDI 5000.02 and 5000.75 (or DoDI 5000.SW) to reflect more iterative approaches for software development.	USD(A&S)	Q2 FY20
D4.4	Modify Service acquisition policy to reflect more iterative approaches for software development.	SAE	Q2 FY20
D4.5	Build a Congressional Reporting Dashboard that would be available to the four Defense Committees to show the progress of DoD and Services DevSecOps programs, including speed and cycle time, code quality, security, and user satisfaction.	USD(A&S)	Q4 FY20

SWAP concept paper recommendations related to this recommendation

10C	Adopt a DevOps culture for software systems.
10C	All software procurement programs should start small, be iterative, and build on success—or be terminated quickly.
D&D	Accept 70% solutions in a short time (months) and add functionality in rapid iterations (weeks).
D&D	Take advantage of the fact that software is essentially free to duplicate, distribute, and modify.
D&D	Treat software development as a continuous activity, adding functionality continuously across its life cycle.
Visits	Spend time upfront getting the architecture right: modular, automated, secure.

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Con	Treat procurements as investments; "What would you pay for a possible initial capability?"
Con	Leverage incentives to make smaller purchases to take advantage of simplified acquisition procedures.
Con	Use modular contracting to allow for regular investment decisions based on perceived value.
Con	Streamline acquisition processes to allow for replacing poorly performing contractors.
T&E	Develop the enterprise knowledge management and data analytics capability for rapid analysis/ presentation of technical RDT&E data to support deployment decisions at each iterative cycle.

Related recommendations from previous studies

OSD06	Change DoD's preferred acquisition strategy for developmental programs from delivering 100 percent performance to delivering useful military capability within a constrained period of time, no more than 6 years from Milestone A. This makes time a Key Performance Parameter.
OSD06	Direct changes to the DoD 5000 series to establish Time Certain Development as the preferred acquisition strategy for major weapons systems development programs.
GAO17	Follow an evolutionary path toward meeting mission needs rather than attempting to satisfy all needs in a single step.
GAO17	Ensure that critical technologies are proven to work as intended before programs begin. Assign more ambitious technology development efforts to research departments until they are ready to be added to future generations (or increments) of a product.
NDU17	Prioritize technical performance and project schedules over cost. Maintain aggressive focus on risk identification and management across all elements of the open system, and resolve technical problems as rapidly as possible.
DSB18	Rec 2a: [DoD programs should] develop a series of viable products (starting with MVP) followed by successive next viable products (NVPs).
DSB18	Rec 2b: [DoD programs should] establish MVP and the equivalent of a product manager for each program in its formal acquisition strategy and arrange for the warfighter to adopt the initial operational capability (IOC) as an MVP for evaluation and feedback.
DSB18	Rec 3a: The MDA (with the DAE, the SAE, the PEO, and the PM) should allow multiple vendors to begin work. A down-selection should happen after at least one vendor has proven they can do the work, and should retain several vendors through development to reduce risk, as feasible.

Additional Recommendation D6
Software Research Portfolio

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of ML and AI into software development, cost estimation, security vulnerabilities, and related areas.**
Stakeholders	USD(R&E), USD(A&S)
Background	Software is essential to national security, and DoD needs to stay ahead of adversaries on emerging SW development practices.
Desired State	DoD benefits from a feedback loop between research and practice, in areas important to retaining the ability to be able to field innovations in software-enabled technologies. Mission needs and a practical understanding of the acquisition ecosystem inform research programs in emerging technologies. Results emerging from research impact the department's warfighting and other systems thanks to high-quality and modular software systems, a DevSecOps infrastructure capable of moving fast, and other enablers. Model-based engineering of software (including "digital twin" approaches) is routinely used to speed development and increase security.
Role of Congress	N/A

Draft Implementation Plan		**Lead Stakeholders**	**Target Date**
D6.1	Designate a responsible person or organization to coordinate software research activities.	USD(R&E)	Q4 FY19
D6.2	Stand up a Chief Engineer for Software to direct the implementation of next-generation software methodologies and tools.	SAEs	Q4 FY19
D6.4	Direct the Principal Civilian Deputy to the SAE to implement the acquisition infrastructure for DevSecOps, allowing quick incorporation of new technologies into DoD systems, implemented by someone with software development experience.	SAEs	Q4 FY19
D6.6	Create a documented DoD Software strategy, perhaps patterned on the DoD cyber strategy,[4] with ties to other existing national and DoD research strategies, and with involvement of A&S and the Services.	USD(R&E)	Q4 FY19
D6.5	Make acquisition data collected continuously from DevSecOps infrastructure and tools available to researchers with appropriate clearances, as a testbed for AI, ML, or other technologies. (See Recs A3, D4)	USD(A&S)	Q4 FY20

Related recommendations from previous studies

DSB18	Rec 7a: Under the leadership and immediate direction of the USD(R&E), the Defense Advanced

[4] https://media.defense.gov/2018/Sep/18/2002041658/-1/-1/1/CYBER_STRATEGY_SUMMARY_FINAL.PDF

	Research Projects Agency (DARPA), the SEI FFRDC, and the DoD laboratories should establish research and experimentation programs around the practical use of machine learning in defense systems with efficient testing, independent verification and validation (IVV), and cybersecurity resiliency and hardening as the primary focus points.

Additional Recommendation D7
Transition Emerging Tools and Methods

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Invest in transition of emerging tools and methods from academia and industry for creating, analyzing, verifying, and testing of software into DoD practice (via pilots, field tests, and other mechanisms).**
Stakeholders	USD(A&S), USD(R&E), Service Digital PEOs
Background	Software is essential to national security, and DoD needs to stay ahead of adversaries in implementing emerging SW development practices. Research work at universities and in the private sector, along with best practice implementation from the private sector, can provide valuable tools and methods to be deployed across DoD.
Desired State	Development and test technology, tools, and methods that are being created and used in the private sector and academia and are known and visible to the PEOs Digital who enable transition into Service programs. DoD labs are investing internally and externally to mature software development and analysis tools.
Role of Congress	N/A

Draft Implementation Plan		Lead Stakeholders	Target Date
D7.1	Create a community of practice, code repositories, and other mechanisms to keep all practitioners knowledgeable about the latest trends and capabilities in software development, testing, and deployment.	USD(A&S)	Q4 FY19
D7.2	Invest in and engage with academic and private sector efforts to transition tools to do software engineering: creating, analyzing, verifying, testing, and maintaining software.	Service Digital PEOs, USD(R&E)	FY20

SWAP working group inputs (reflected in Appendix F) related to this recommendation

Req	OSD should consider identifying automated software generation areas that can apply to specific domains.

Related recommendations from previous studies

OSD06	Direct the Deputy Director for Research and Engineering to coordinate service science and technology transition plans with the appropriate military service.
OSD06	Direct the Deputy Director for Research and Engineering to actively participate in the Joint Capabilities Acquisition and Divestment process to reemphasize technology push initiatives.

Additional Recommendation D8
Collect Data

Line of Effort	Change the practice of how software is procured and developed.
Recommendation	**Automatically collect all data from DoD national security systems, networks, and sensor systems, and make the data available for machine learning (via federated, secured enclaves, not a centralized repository).**
Stakeholders	USD(A&S), USD(P&R), SAE, CMO, CAPE, DOT&E, DDR&E(AC)
Background	DoD discards or does not have access to significant amounts of data for its systems and has not established an infrastructure for storing data, mining data, or making data available for machine learning. Current analytical efforts are siloed and under-resourced in many cases.
Desired State	DoD has a modern architecture to collect, share, and analyze data that can be mined for patterns that humans cannot perceive. Data is being used to enable better decision-making in all facets of the Department, providing significant advantages that adversaries cannot anticipate. Data collection and analysis is done without compromising security, and DoD, with minimum exceptions, should have complete data rights for all systems (developed with industry).
Role of Congress	N/A

Draft Implementation Plan		**Lead stakeholders**	**Target Date**
D8.1	Develop comprehensive data strategy for DoD, taking into account future AI/ML requirements,	CDO with USD(A&S), SAE	Q1 FY20
D8.2	Implement a minimum viable product (MVP) that collects and analyzes the most critical data element for one or more programs.	CDO with USD(A&S), SAE	Q3 FY20
D8.3	Create digital data infrastructure to support collection, storage, and processing.	CDO with USD(A&S), SAE	Q1 FY21
D8.4	Require that all new major systems should specify a data collection and delivery plan.	A&S	Q2 FY21
D8.5	Implement data collection requirements for new sensor and weapon system acquisition.	A&S	FY21

SWAP concept paper recommendations related to this recommendation

10C	All data generated by DoD systems—in development and deployment—should be stored, mined, and made available for machine learning.

Related recommendations from previous studies

DSB18	Rec 7b: [USD(R&E)] should establish a machine learning and autonomy data repository and exchange along the lines of the U.S. Computer Emergency Readiness Team (US-CERT) to collect and share necessary data from and for the deployment of machine learning and autonomy.
DSB18	Rec 7c: [USD(R&E)] should create and promulgate a methodology and best practices for the construction, validation, and deployment of machine learning systems, including architectures and test harnesses.

Appendix B: Legislative Opportunities in Response to 2016 NDAA Section 805

(Template Language for Recommendations A1 and A2)

This appendix provides a template for the type of legislative language that could represent a new category/pathway to procure, develop, deploy and continuously improve software for DoD applications, aligned with Recommendations A1 and A2 in Chapter 5. This template is designed to serve as an example of how the types of changes we envision might be implemented and has not been reviewed or endorsed by the Department. It is written to be consistent with 2016 NDAA Section 805 (Use of alternative acquisition paths to acquire critical national security capabilities).

SEC. [???]. SPECIAL PATHWAYS FOR RAPID ACQUISITION OF SOFTWARE APPLICATIONS AND UPGRADES.

(a) GUIDANCE REQUIRED.—Not later than [90, <u>180</u>, 270] days after the date of the enactment of this Act, the Secretary of Defense shall establish guidance authorizing the use of special pathways for the rapid acquisition of software applications and upgrades that are intended to be fielded within one year.

(b) SOFTWARE ACQUISITION PATHWAYS.—

(1) The guidance required by subsection (a) shall provide for the use of proven technologies and solutions to continuously engineer and deliver capabilities in software. The objective of an acquisition under this authority shall be to begin the engineering of new capabilities quickly, to demonstrate viability and effectiveness of those capabilities in operation, and continue updating and delivering new capabilities iteratively afterwards. An acquisition under this authority shall not be treated as an acquisition program for the purpose of section 2430 of title 10, United States Code or Department of Defense Directive 5000.01.

(2) Such guidance shall provide for two rapid acquisition pathways:

(A) APPLICATIONS.—The applications software acquisition pathway shall provide for the use of rapid development and implementation of applications and other software and software improvements running on commercial commodity hardware (including modified or ruggedized hardware) operated by the Department; and

(B) EMBEDDED SYSTEMS.—The embedded systems software acquisition pathway shall provide for the rapid development and insertion of upgrades and improvements for software embedded in weapon systems and other military-unique hardware systems.

(c) EXPEDITED PROCESS.--

(1) IN GENERAL.—The guidance required by subsection (a) shall provide for a streamlined and coordinated requirements, budget, and acquisition process that results in the rapid fielding of software applications and software upgrades to embedded systems in a period of

not more than [one year] from the time that the process is initiated. It shall also require the collection of data on the version fielded and continuous engagement with the users of that software, so as to enable engineering and delivery of additional versions in periods of not more than one year each.

(2) EXPEDITED SOFTWARE REQUIREMENTS PROCESS.—

(A) Software acquisitions conducted under the authority of this provision shall not be subject to the Joint Capabilities Integration and Development System Manual and Department of Defense Directive 5000.01, except to the extent specifically provided in the guidance required by subsection (a).

(B) The guidance required by subsection (a) shall provide that—

(1) Requirements for covered acquisitions are developed on an iterative basis through engagement with the user community, and utilization of user feedback in order to regularly define and prioritize the software requirements, as well as to evaluate the software capabilities acquired;

(2) The requirements process begins with the identification of 1) the warfighter or user need, 2) the rationale for how these software capabilities will support increased lethality and/or efficiency, and 3) the identification of a relevant user community;

(3) Initial contract requirements are stated in the form of a summary-level list of problems and shortcomings in existing software systems and desired features or capabilities of new or upgraded software systems;

(4) Contract requirements are continuously refined and prioritized in an evolutionary process through discussions with users that may continue throughout the development and implementation period;

(5) Issues related to life-cycle costs and systems interoperability are considered; and

(6) Issues of logistics support in cases where the software developer may stop supporting the software system are addressed.

(3) RAPID CONTRACTING MECHANISM.— The guidance required by subsection (a) shall authorize the use of a rapid contracting mechanism, pursuant to which—

(A) Aa contract may be awarded within a [90-day] period after proposals are solicited on the basis of statements of qualifications and past performance data submitted by contractors, supplemented by discussions with two or more contractors determined to be the most highly-qualified, without regard to price;

(B) a contract may be entered for a period of not more than one-year and a ceiling price of not more than [$50 million] and shall be treated as a contract for the acquisition of commercial services covered by the preference in section 2377 of title 10, United States Code;

(C) a contract shall identify the contractor team to be engaged for the work, and substitutions shall not be made during the base contract period without the advance written consent of the contracting officer;

(D) the contractor may be paid during the base contract period on a time and materials basis up to the ceiling price of the contract to review existing software in consultation with the user community and utilize user feedback to define and prioritize software requirements, and to design and implement new software and software upgrades, as appropriate;

(E) a contract may provide for a single one-year option to complete the implementation of one or more specified software upgrades or improvements identified during the period of the initial contract, with a price of not more than [$100 million] to be negotiated at the time that the option is awarded; and

(F) an option under the authority of this section may be entered on a time and materials basis and treated as an acquisition of commercial services or entered on a fixed price basis and treated as an acquisition of commercial products, as appropriate.

(4) EXECUTION OF RAPID ACQUISITIONS.--The Secretary shall ensure that —

(A) software acquisitions conducted under the authority of this provision are supported by an entity capable of regular automated testing of the code, which is authorized to buy storage, bandwidth, and computing capability as a service or utility if required for implementation;

(B) processes are in place to provide for collection of testing data automatically from [entity specified in (A)] and using those data to drive acquisition decisions and oversight reporting;

(C) the Director of Operational Test and Evaluation and the director of developmental test and evaluation participate with the acquisition team to design acceptance test cases that can be automated using the entity specified in (A) and regularly used to test the acceptability of the software as it is incrementally being engineered;

(D) acquisition progress is monitored through close and regular interaction between government and contractor personnel, sufficient to allow the government to understand progress and quality of the software with greater fidelity than provided by formal but infrequent milestone reviews;

(E) an independent, non-advocate cost estimate is developed in parallel with engineering of the software, and is based on an investment-focused alternative to current estimation models, which is not based on source lines of code;

(F) the performance of fielded versions of the software capabilities are demonstrated and evaluated in an operational environment; and

(G) software performance metrics addressing issues such as deployment rate and speed of delivery, response rate such as the speed of recovery from outages and cybersecurity vulnerabilities, and assessment and estimation of the size and complexity of software development effort are established that can be automatically generated on a [monthly, weekly, continuous] basis and made available throughout the Department of Defense and the congressional defense committees.

(5) ADMINISTRATION OF ACQUISITION PATHWAY.—The guidance for the acquisitions conducted under the authority of this section may provide for the use of any of the following streamlined procedures in appropriate circumstances:

(A) The service acquisition executive of the military department concerned shall appoint a project manager for such acquisition from among candidates from among civilian employees or members of the Armed Forces who have significant and relevant experience in modern software methods.

(B) The project manager for each large software acquisition as designated by the service acquisition executive shall report with respect to such acquisition directly, and without intervening review or approval, to the service acquisition executive of the military department concerned.

(C) The service acquisition executive of the military department concerned shall evaluate the job performance of such manager on an annual basis. In conducting an evaluation under this paragraph, a service acquisition executive shall consider the extent to which the manager has achieved the objectives of the acquisition for which the manager is responsible, including quality, timeliness, and cost objectives.

(D) The project manager shall be authorized staff positions for a technical staff, including experts in software engineering to enable the manager to manage the acquisition without the technical assistance of another organizational unit of an agency to the maximum extent practicable.

(E) The project manager shall be authorized, in coordination with the users of the equipment and capability to be acquired and the test community, to make trade-offs among life-cycle costs, requirements, and schedules to meet the goals of the acquisition.

(F) The service acquisition executive or the defense acquisition executive in cases of defense wide efforts, shall serve as the decision authority for the acquisition.

(G) The project manager of a defense streamlined acquisition shall be provided a process to expeditiously seek a waiver from Congress from any statutory or regulatory requirement that the project manager determines adds little or no value to the management of the acquisition.

(6) OTHER FLEXIBLE ACQUISITION METHODS.—The flexibilities provided for software acquisition pathways under this section do not preclude the use of acquisition flexibilities otherwise available for the acquisition of software. The Department may use other transactions authority, broad agency announcements, general solicitation competitive procedures authority under section 879 of the National Defense Authorization Act for Fiscal Year 2017, the challenge program authorized by section 2359b of title 10, United States Code, and other authorized procedures for the acquisition of software, as appropriate. Such authorities may be used either in lieu of or in conjunction with the authorities provided in this section.

(d) FUNDING MECHANISMS.—

(1) SOFTWARE FUND.—

(A) IN GENERAL.—The Secretary of Defense shall establish a fund to be known as the [‘‘Department of Defense Rapid Development of Effective Software Fund’’] to provide funds, in addition to other funds that may be available for acquisition under the rapid software development pathways established pursuant to this section. The Fund shall be managed by a senior official of the Department of Defense designated by the [Under Secretary of Defense for Acquisition and Sustainment]. The Fund shall consist of amounts appropriated to the Fund and amounts credited to the Fund pursuant to section [???] of this Act.

(B) TRANSFER AUTHORITY.—Amounts available in the Fund may be transferred to a military department for the purpose of starting an acquisition under the software acquisition pathway established pursuant to this section. These funds will be used to fund the first year of the software acquisition and provide the Department an opportunity to field software capabilities that address newly discovered needs. A decision to continue the acquisition on other funds will be made based upon the progress demonstrated after the first year. Any amount so transferred shall be credited to the account to which it is transferred. The transfer authority provided in this subsection is in addition to any other transfer authority available to the Department of Defense.

(C) CONGRESSIONAL NOTICE.—The senior official designated to manage the Fund shall notify the congressional defense committees of all transfers under paragraph (2). Each notification shall specify the amount transferred, the purpose of the transfer, and the total projected cost and funding based on the effort required each year to sustain the capability to which the funds were transferred. The senior official will also notify the congressional defense committees at the end of the one-year timeframe and report on the fielded capabilities that were achieved. A notice under this paragraph shall be sufficient to fulfill any requirement to provide notification to Congress for a new start.

(2) PILOT PROGRAM. The Secretary may conduct a pilot program under which funding is appropriated in a single two-year appropriation for life-cycle management of software-intensive and infrastructure technology capabilities conducted under the authority of this section. The objective of the appropriation software pilot program would be to provide 1) greater focus on managed services versus disaggregated development efforts, 2) additional accountability and transparency for information centric and enabling technology capabilities, and 3) flexibility to pursue the most effective solution available at the time of acquisition; 4) much greater insight into the nature of software expenditures across the DOD enterprise; 5) an improved ability to measure costs and program performance;

Appendix C: An Alternative to P-Forms and R-Forms: How to Track Software Programs

Background. DoD's Planning, Programming, and Budgeting System (PPBS) establishes the basis for the budget submission to Congress. Multiple statutes, instructions, and directives must be addressed in order to change the way the budget is put together, adjudicated, enacted and managed. Exhibits are prepared by OSD and DoD Components to support requests for appropriations from Congress and help justify the President's budget. These include a number of forms that are aligned with the existing appropriations process:

- P-Form: Procurement
- R-Form: Research, Development, Test, and Evaluation (RDT&E)
- O-Form: Operations and Maintenance
- M-Form: Military Personnel
- C-Form: Military Construction

As described by the Section 809 panel, the competing objectives of the acquisition system make it very difficult for Congress and the Department to effectively budget and manage defense projects, as illustrated in the following diagram (from the Section 809 panel, Volume 3):

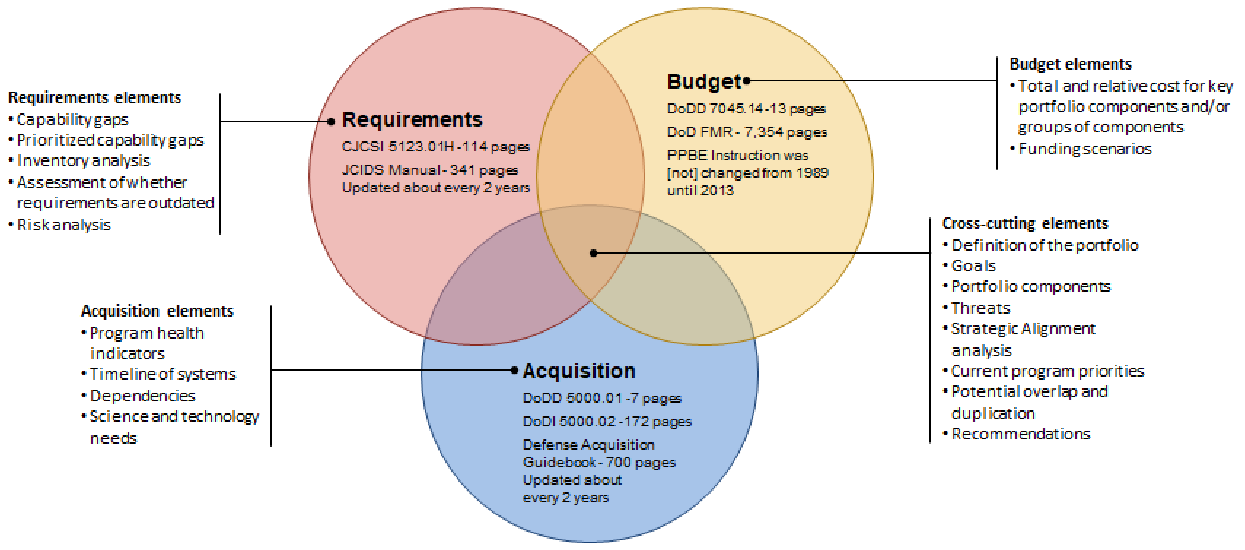

Requirements elements
• Capability gaps
• Prioritized capability gaps
• Inventory analysis
• Assessment of whether requirements are outdated
• Risk analysis
Requirements
CJCSI 5123.01H - 114 pages
JCIDS Manual - 341 pages
Updated about every 2 years
Budget
DoDD 7045.14-13 pages
DoD FMR - 7,354 pages
PPBE Instruction was [not] changed from 1989 until 2013
Budget elements
• Total and relative cost for key portfolio components and/or groups of components
• Funding scenarios
Cross-cutting elements
• Definition of the portfolio
• Goals
• Portfolio components
• Threats
• Strategic Alignment analysis
• Current program priorities
• Potential overlap and duplication
• Recommendations
Acquisition elements
• Program health indicators
• Timeline of systems
• Dependencies
• Science and technology needs
Acquisition
DoDD 5000.01 - 7 pages
DoDI 5000.02 - 172 pages
Defense Acquisition Guidebook - 700 pages
Updated about every 2 years

Figure C.1. Multi-layered DoD budget environment.

In this appendix, we describe a different type of mechanism for budget management for software programs, one that is tuned to the nature of software development. We envision this design to reflect and be interweaved with our primary recommendations—in particular A1 (new acquisition pathway for software) and A2 (new appropriations category for software). It could be also be used for software programs that are making use of other pathways (e.g., traditional DoD 5000.02, mid-tier [Sec 804] acquisition, other transaction authority [OTA] based pathways, or operations and maintenance [O&M]).

Key Characteristics. It is useful to list some of the properties that the new process should satisfy before presenting a specific approach for new methods of managing the budget for software programs. The characteristics that we believe are most important are that the process be:

- *Iterative*: In proposing a new approach for approval and oversight of software programs, we envision a process very similar to the way that software itself is developed: Congress and DoD should articulate what their needs are for oversight and approval of software programs, then try out different ways to gain transparency in proposing and monitoring of software programs. Oversight processes can evolve iteratively, ultimately achieving better oversight

- *Efficient:* The current budget process requires the separate creation of standalone forms and documents that are not a part of the regular information that is maintained and tracked as part of the planning and execution of the software program. Instead, we emphasize the use of automated and machine-readable budget information that is interoperable with financial management tools (with translation to human-readable form when useful).

- *Insightful.* The process should provide insights to both DoD and Congress about the planned and current capabilities of the program and opportunities for portfolio optimization. This includes making use of metrics that are appropriate for software (cycle time, rollback time, automated test coverage, etc.), extracting those metrics in an automated fashion wherever possible, and treating software as an enduring capability.

- *Electronic.* Consistent with the nature of software and software development, the budget artifacts used by Congress and DoD should be largely electronic in nature. By "electronic" we do not mean electronic forms that are "printable" (e.g., PDF and Word files), but rather information that is available in electronic form and requires no further processing to be ingested into analysis systems.

Budget Information for Ongoing Software Programs. Since software is never done, the most important budget artifacts will be those for ongoing programs. The information that is required depends on the type of software, so we briefly describe here our advice for what information should be most relevant in evaluating and renewing the budget of an ongoing program.

- *Type A (commercial off-the-shelf apps):* By its nature, ongoing expenses for COTS apps will be based on the commercial price of the software or service. Existing mechanisms for budgeting materials, supplies, and consumables for DoD functions should be used: usage, spend rate, attainment of (volume) price discounts, etc. It is also important to track resources (money and people) needed to perform upgrades made mandatory by vendor version updates and obsolescence.

- *Type B (customized software)* and *Type C (COTS hardware/operating system):* These classes of software represents custom software that is developed, assured, deployed, and maintained by either organic developers or a contractor/vendor for DoD-specific purposes. Type B software will require primarily configuration management and customization, whereas Type C software will involve customized coding. These types of software are perhaps the least well-suited to the traditional spiral development/hardware-focused acquisition and budgeting process, since they often represent an enduring capability in which new features are continuously added.

The diagram below shows the expected cost profile of a software program of this type, in which the annual cost starts small (and may terminate, if not successful), rises as the software is scaled to its full extent, and then falls as it is optimized and continuously improved.

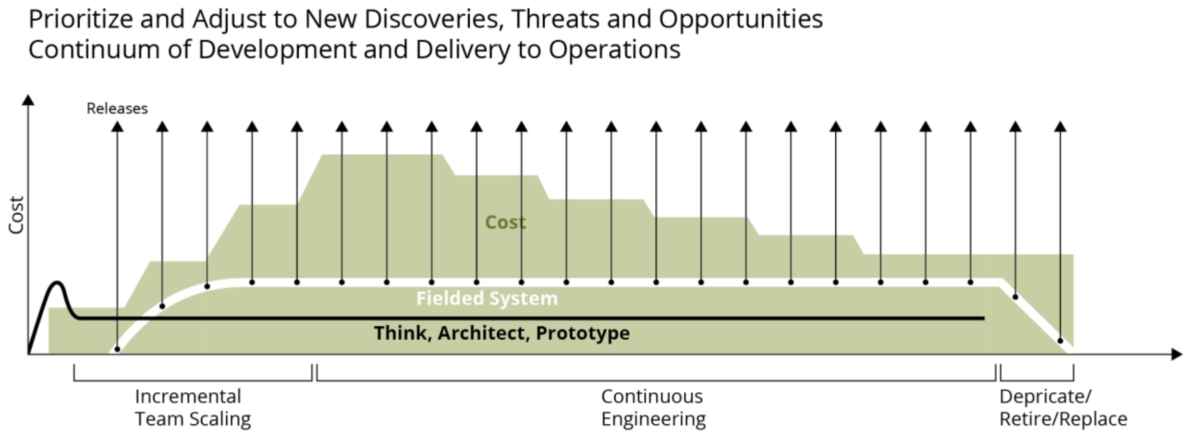


Figure C.2. DevSecOps life cycle cost profile.

The information available as part of the budget process should reflect the following data on the current and desired state of the program:

- List of features implemented and those planned for future releases
- Number of active users and level of satisfaction of the user base
- Time required to field high priority functions (specifications → operations) or fix newly found security holes (discovery → operations)
- Time from code committed to code in use
- Time required for full regression test and cybersecurity audit/penetration testing (and the percentage of such testing that is automated)
- Time required to restore service after outage
- Percentage test coverage of specs/code, including percentage of tests that are automated
- Number of bugs caught in testing versus in field use
- Change failure rate (rollback deployed code)
- Percentage code available to DoD for inspection/rebuild

The cost data associated with the program should include the following information:

- The size and annual cost of the development team, along with the percentage of programmers, designers, user interface engineers, system architects and other key development categories.
- The size and annual cost of the program management team, including both government and contractor program management (if applicable).
- Software licensing fees
- Computing costs (including cloud services)
- Other costs associated with the program

These metrics should be tracked over time, with reports of the past three years of data as well as targets for the coming two years. Annual budget submissions should compare the projected metrics and costs of the program from the past fiscal year with the actual metrics and costs for that period, as well as rolling updating the time horizons to drop the oldest year of tracking data and add the newest year of projected data.

- *Type D (custom hardware and software, including embedded systems):* Embedded systems associated with custom hardware that is still in the development phase is most likely to be reported as part of the hardware development program (using traditional budget items). However, once the software/hardware platform and form factor has been designed then the continued development of the software should be reported in a manner similar to Type C (COTS hardware/operating systems).

Budget Information for New Software Programs. Creating new software programs involves estimation of the cost of the software over at least the initial procurement and deployment phases. Such programs should start small, be iterative, and build on success—or be terminated quickly. Whenever possible, new software programs should have small budgets, require early demonstration of results, and then be turned into ongoing programs (with budget justification as described above). We remark briefly on specific considerations based on the type of software.

- *Type A (commercial off-the-shelf apps)* and *Type B (customized software).* For commercial software of these two types, the most relevant information is the features to be provided by the software, the number of instances of the software expected over time, and the cost of that software (either as purchase cost or licensing costs). For Type B software, additional information should be provided regarding the staffing needs for software configuration, in a manner that is similar to customized software (Type C), though with less intensive development costs.

- *Type C (COTS hardware/operating system).* For custom software running on commodity hardware and operating systems, there are two primary questions that must be addressed: (a) is the software functionality available in commercial products that meets the (primary) needs of the Department and, if not, (b) how large should the initial development effort be in order to create a minimum viable product (MVP) and then begin to scale the initial deployment if successful.

  For comparing customized software to commercially available software, the following information should be provided:

  - A list of features that are desired and an indication of which of those features are available in commercial packages versus those that are DoD-specific.
  - A list of commercial software packages providing similar functionality and the cost of purchasing or licensing that software for initial and full-scale deployment.
  - A justification for why DoD processes cannot be adopted to the development and operations practices of standard commercial approaches and/or why a smaller software development program focused on interfacing DoD specific cases to commercial packages cannot be accomplished.

The goal of providing this information is to ensure that commercial processes/software can be adopted and implemented as standard business practices within DoD. If a DoD-customized software is needed, this information also serves as a good comparison point on the rough costs that are available for related commercial software (when it exists).

- *Type D (custom hardware and software, including embedded systems):* The initial phases of development for custom hardware and software are likely to track hardware development, although in some cases it may be possible to begin software development using emulation and simulation. Care should be taken that embedded software truly requires custom solutions: the trend in commercial software is to establish a layer between hardware and software that allows software to be hardware agnostic (converting Type D into Type C). This approach is quite prevalent in consumer electronics (smart phones and other mobile devices) and transportation systems (automobiles, aircraft).

Software Program Budget Exhibits. Since software programs will be integrated into larger programs and elements of larger programs will have software component, it will be necessary to provide budget exhibits that are compatible with other budget processes used by Congress and DoD. As described above, we believe that the primary information used for tracking ongoing programs should be electronic in nature, and that it should be pulled from existing databases and systems rather than compiled specifically for the budget process.

Following the format used by R-docs, we believe that software programs budget exhibit can be broken down into 5 levels, as shown in the diagram to the right. Each of the exhibits should reflect the information described above (depending on the type of software program) and should exist primarily as electronic databases whose information can be presented in a form consistent with the information that Congress desires.

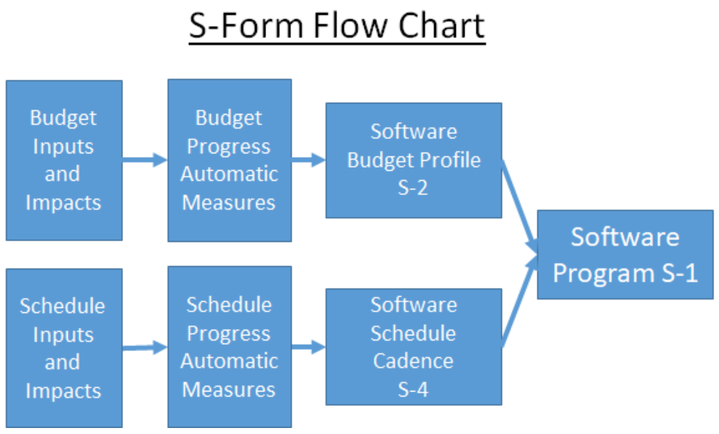

S-Form Flow Chart
Budget Inputs and Impacts
Budget Progress Automatic Measures
Software Budget Profile S-2
Software Program S-1
Schedule Inputs and Impacts
Schedule Progress Automatic Measures
Software Schedule Cadence S-4

Figure C.3. S-Form inputs.

The individual exhibits are as follows:

- S-1 Exhibits: the basic document for presenting DoD’s software program information. The S-1 is prepared at the OSD-level, with one exhibit for each separate software appropriation account/portfolio. Because the S-1 is a summary document, all other software exhibits submitted for a program element must reconcile to the numbers shown on the S-1. The S-1 form should be automatically generated from information maintained by the Component headquarters based on information provided (electronically) be individual software program elements.

- S-2 Exhibits: feeds into the S-1 and are automatically populated to provide summary funding information, program description, metrics, and budget justification for each software program element.

- S-4 Exhibits: generate a display of major program releases. This exhibit is required for each project. If a program element consists of only one project, then the S-4 is prepared for the entire program element.

Multi-Element Program Budgets. For the purpose of establishing a new funding authority that will address the continuous improvement nature of software, a coordinated set of budget exhibits must be put in place. Capability elements that are solely software are relatively rare. The hardware platform that the software must run on will either be provided by a different program under a platform-as-a-service (PaaS), or involve computing hardware that is necessarily coincident to a military vehicle (carried in a ship, aircraft, ground or space). When physical space, power, weight and cooling needs for the computer services have to be managed at the vehicle level, a coordination of the design and implementation of the hardware/software environment must be established and managed over a long period—several epochs of lifespans for computer equipment on which the continuously changing software must run. This is a fundamentally different environment than hardware and must be accommodated in a new software budget exhibit, at the right time of development, while managing within the appropriate form-factor.

Fortunately, the PPBS environment has a mechanism for managing this—the multi-Program Element Project. The coordination of research (R-Form), Procurement (P-Form) and Operations (O-Form) with software program information (electronically generated S-Form) can be accommodated in a single project or set of projects in the PPBS. The most limiting case is the one that requires the greatest level of coordination in software-intensive and embedded products. The figure below shows a parallel timeline for the ideation, creation, scaling and implementation phases of software with the spiral nature of hardware for research, engineering/manufacturing development, procurement, operations, sustainment and disposal.

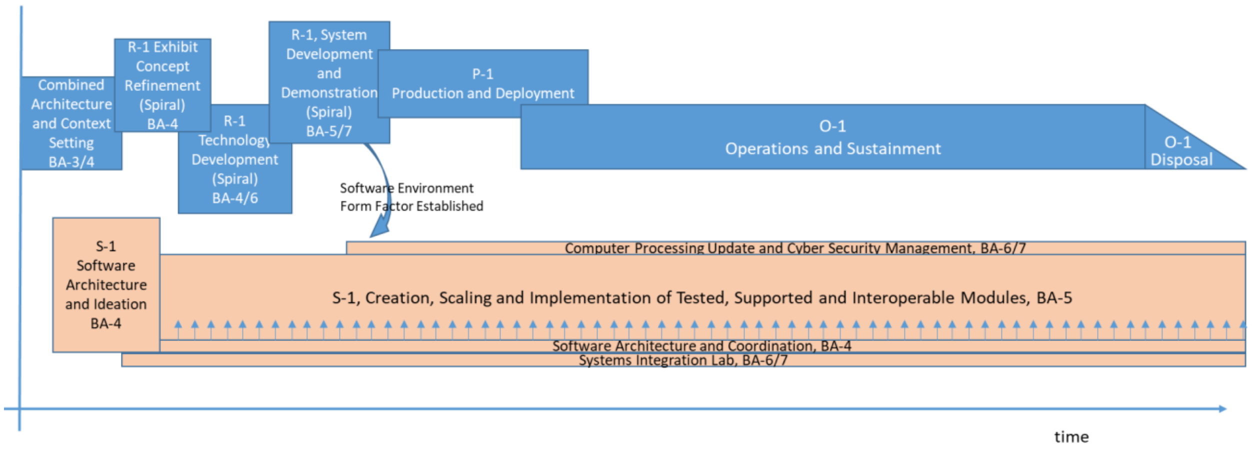

Combined Architecture and Context Setting BA-3/4
R-1 Exhibit Concept Refinement (Spiral) BA-4
R-1 Technology Development (Spiral) BA-4/6
R-1, System Development and Demonstration (Spiral) BA-5/7
P-1 Production and Deployment
O-1 Operations and Sustainment
O-1 Disposal
Software Environment Form Factor Established
S-1 Software Architecture and Ideation BA-4
Computer Processing Update and Cyber Security Management, BA-6/7
S-1, Creation, Scaling and Implementation of Tested, Supported and Interoperable Modules, BA-5
Software Architecture and Coordination, BA-4
Systems Integration Lab, BA-6/7
time

Figure C.4. Budget exhibits by program phase.

Sample Budget Exhibits. To illustrate the type of information that could be presented to Congress as part of the budgeting process, we provide below a sample of some "S-Forms" that might be used to describe a hypothetical software program. For the purposes of illustration, we focus here on a Type C (custom software on commercial hardware/operating system). Other types of software could make use of similar exhibits. We again emphasize that the desired state is that these documents are automatically populated based on electronic databases used within program offices and maintained as part of ongoing development activities.

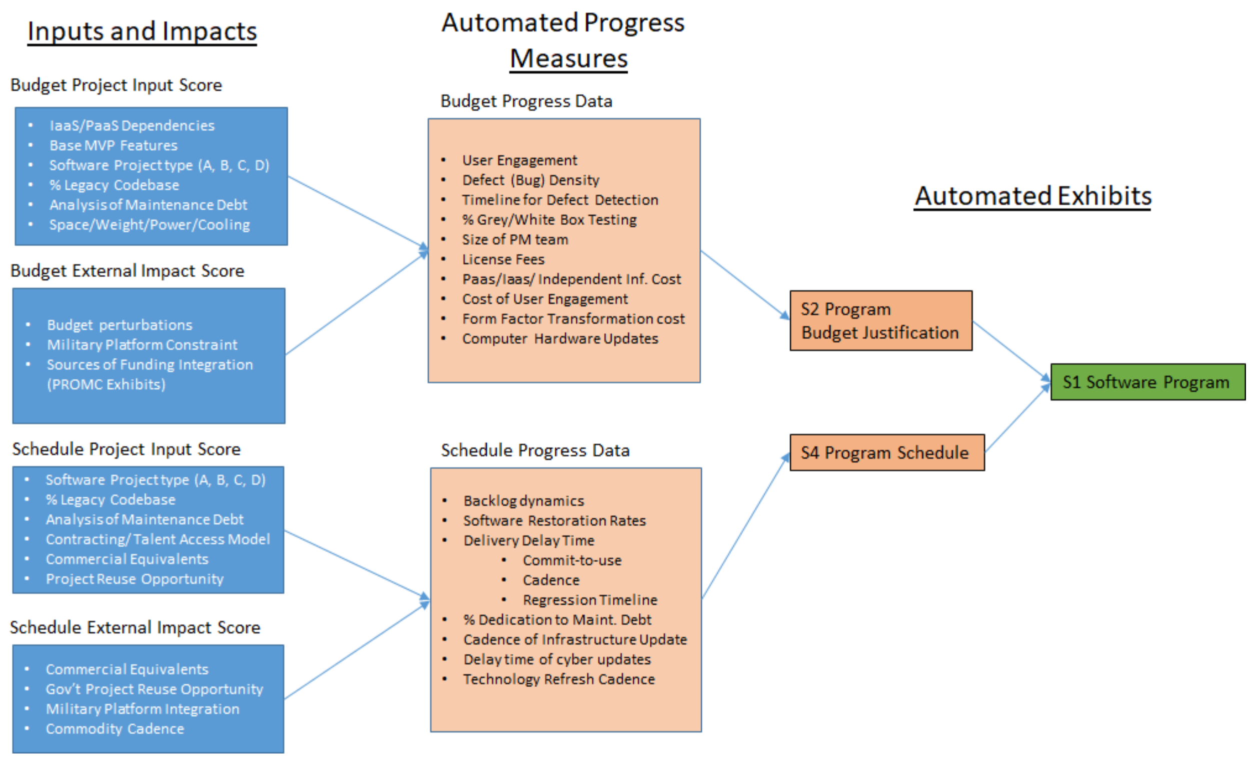

Inputs and Impacts
Budget Project Input Score
IaaS/PaaS Dependencies
Base MVP Features
Software Project type (A, B, C, D)
% Legacy Codebase
Analysis of Maintenance Debt
Space/Weight/Power/Cooling
Budget External Impact Score
Budget perturbations
Military Platform Constraint
Sources of Funding Integration (PROMC Exhibits)
Schedule Project Input Score
Software Project type (A, B, C, D)
% Legacy Codebase
Analysis of Maintenance Debt
Contracting/Talent Access Model
Commercial Equivalents
Project Reuse Opportunity
Schedule External Impact Score
Commercial Equivalents
Gov't Project Reuse Opportunity
Military Platform Integration
Commodity Cadence
Automated Progress Measures
Budget Progress Data
User Engagement
Defect (Bug) Density
Timeline for Defect Detection
% Grey/White Box Testing
Size of PM team
License Fees
Paas/Iaas/ Independent Inf. Cost
Cost of User Engagement
Form Factor Transformation cost
Computer Hardware Updates
Schedule Progress Data
Backlog dynamics
Software Restoration Rates
Delivery Delay Time
Commit-to-use
Cadence
Regression Timeline
% Dedication to Maint. Debt
Cadence of Infrastructure Update
Delay time of cyber updates
Technology Refresh Cadence
Automated Exhibits
S2 Program Budget Justification
S4 Program Schedule
S1 Software Program

Figure C.5. Software progress metrics and budget exhibit crosswalk

Appendix D: Frequently Asked Questions (FAQ)

This document captures some of the common questions and comments that we have received as we discussed the report with various groups.

1. **Haven't all of these ideas already been recommended in previous studies? Why is this study/report any different?**

   Yes, the vision for how to do software right has existed for decades and most of the best practices that we and others have recommended are common practice in industry today. Chapter 3 (Been There, ~~Done~~ Said That) summarizes previous work and provides our assessment of why things haven't changed. Here are the parts we think are new and different:

   - The recommendations in this report serve primarily as documentation of a sequence of iterative conversations and the real work of the report is the engagements before and after the report is released.
   - Our engagements in the process, and the iterative ways we have worked on this study (just like good software!) have created a willing group of advocates (inside the Department) ready to move forward. If we permit them, we believe change will occur.
   - We focus on speed and cycle time as the key drivers for what needs to change and recommend optimizing statutes, regulations, and processes to allow management and oversight of speed at scale. This won't fix everything, but if you optimize for speed then many other things will improve as well (including oversight).
   - This report is shorter and pithier than previous reports, so we hope people will read it.

2. **Shouldn't Congress just get out of the way and let DoD run things the way they want?**

   This is not the way that the Constitution works. The Legislative branch is an equal branch of government and has a responsibility to see that the Executive branch performs its duties well and properly uses taxpayer resources. This makes implementation of many of the ideas in this report a challenge, but we believe that oversight of software is actually *easier* than oversight of hardware, and Congress can and should take advantage of the insights provided by optimizing speed and cycle time to perform oversight of defense software.

3. **Military software is different than commercial software since lives and national security are at stake, so we can't just do things like they do in industry.**

   Not all (defense) software is the same. Some software requires different consideration in DoD compared with industry, but some software is very much equivalent. Foreign governments perform espionage against U.S. companies and those companies should be protecting themselves in the same way as the U.S. government should (and in many cases, companies are doing better at protecting their code than the government, in our experience).

   And even for those types of software that are very different from what we would find in the commercial world, the broad themes of modern software development are the same: software is never done, speed and cycle time are critical measures, software is by people and for people, and software is different from hardware. In all cases we believe that the acquisition of software must recognize these broad themes to take advantage of the opportunities provided by modern software development practices.

While certainly agreeing that the role of military is different, there are many areas of the private sector in which health, economic well-being, and life safety are critically dependent on software - aircraft, hospitals, traffic management, etc.

4. **Embedded software (in weapons systems) is different than commercial software since it is closely tied to hardware, so we can't just do things like they do in industry.**

   Not all software is the same, and embedded systems have different requirements for testing and verification that may not be present in other types of systems. The broad themes of modern software development also hold for embedded systems: software is never done, speed and cycle time are critical measures, software is by people and for people, and software is different from hardware. The issue of cycle time is the one that usually raises the most concern, but we note that embedded software can also have bugs and vulnerabilities and figuring out how to deploy patches and updates quickly is a valuable feature (think about hardware-coupled features in a mobile device or a Tesla as examples of where this is already being done in industry).

5. **For military systems, training is an essential element and we can't change the software quickly because we can't retrain people to use the new version.**

   Not all software is the same and many types of software have functions that are not directly evident to the user. Indeed, there are some types of software where you might want to update things more slowly to avoid creating confusion for a human operating under stress and having to rely on their training to avoid doing something wrong. For those systems, it will be important to figure out how to couple software updates with training so that warfighters have access to the latest version of the software that provides the functionality and security required to carry out their mission. It is also important to continuously evolve our training regimes to take advantage of what may be increased flexibility and adaptability of "digital natives."

6. **Providing source code to the government is a non-starter for industry. How will they make money if they have to give the government their code?**

   It is critical that DoD have access to source code for purpose-build software: it is required in order to do security scans to identify and fix vulnerabilities, and only with access to the source code and build environment can the government maintain code over time. However, providing source code is different than handing over the rights to do anything they want with that code. Modern intellectual property (IP) language should be used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors should be able to use licensing agreements that protect any IP that they have developed with their own resources.

8. **Won't Congress simply reject modern continuous, incremental software programs believing that "software is never done" is just an open invitation to make programs last forever?**

   "Software is never done" specifically highlights that certain capabilities will be enduring, e.g., DoD will always need the capability to ingest data from overhead assets, process that data, and disseminate it and the information it contains. In this situation sensors will change, new analyses will be developed and new products will be required by decision makers. In the traditional DoD software world, a highly defined requirement would be defined, a program would be launched and years later a (likely) out-of-date capability would be delivered, followed immediately by a new, large scale, highly definable requirement, blah, blah, blah. In a world where this need will endure, a continuously funded, incrementally managed software program works better. We must be comfortable that we will spend a certain amount of money each year, we let the program use modern tools for delivering value to real end users incrementally, and we measure success by real-time metrics delivered by the development infrastructure and through direct feedback from the user community. This is the best way to provide Congress with the oversight it deserves.

9. **Have you read a P-Form and an R-Form?**

   We have! To us, these do not seem to be able to provide the type of insight into a software (or software-intensive) program that would be required to make a sound judgement about whether a program is in trouble. In addition, they appear to require substantial manual effort to generate and that effort has relatively little added value, they are missing key metrics that are important to understand whether a software program is on track (speed, cycle time, bugs found in test versus in the field, etc.), and the information they contain is updated to infrequently.

   In Appendix C of our report we describe a different type of mechanism for budget submissions for software programs, one that is tuned to the nature of software development. We believe that it is possible to implement a mechanism for managing software program that makes use of digitally generated information that is part of the ongoing data that are used in the software development process and that provides improved insight into how well that program is delivering value to the end user.

10. **Government will never hire software developers that are as good as industry.**

    While it is certainly true that the vast majority of the highest capability software developers are in the private sector, it is also true that we found extremely capable and dedicated people in the Department—just not nearly enough of them. Actions as consistently detailed in our study can help to address this gap. First, the government should continue to partner with industry and to make use of contractors as a mechanism for obtaining the talent that it needs to develop software that meets its needs. For those cases where it makes sense to use organic (government) software development, the government should make use of existing or new hiring authorities to offer salaries that are as competitive as possible. It is highly unlikely that these will match commercial salaries, but it will show that DoD values software development expertise and that it recognizes that this expertise is in high demand and short supply. On top of this, DoD should anticipate that they will not be able to attract software developers for their entire career. Instead, DoD should have a plan and a set of mechanisms that allow it to hire people for shorter periods of time (e.g., 2-4 years), a period which we believe individuals who are interested in serving their country will be willing to devote. Recommendation C4 (Recruiting (Transient) Digital Talent) provides some ideas for how this might be implemented.

11. **What is the purpose of the use of commercial services guidance in the new acquisition pathway that you propose (Recommendation A1 and Appendix B)?**

    Commercial item procurement was established in 1994 by Congress as a way of encouraging new entrants into the industrial base. While the law was directed at Silicon Valley it also included the vast majority of other types of commercial products at the time — eventually to expand into a greater number of services. Procedures were established (under FAR Part 12) to exempt these types of fixed price contracts from a significant portion of defense-unique acquisition requirements. A preference was also established for the government to buy commercial products and solutions where they existed over defense unique solutions.

    The rapid contracting mechanism in Appendix B would essentially treat all purchases through this mechanism as a commercial item covered under FAR Part 12 to limit DoD from applying unique accounting and oversight procedures applicable to traditional defense contracts. Defining these purchases as commercial item purchases triggers two things: (1) a purchasing preference and (2) relief from regulatory burdens, including government-unique contract clauses and data requirements. The purpose of this language is to ensure this favorable treatment for the alternative acquisition pathway without requiring the contractor to make any proof that is a "commercial" vendor.

12. **Would the use of the proposed acquisition pathway (Recommendation A1) and/or proposed appropriation category (Recommendation A2) be required for all software programs?**

    No. We envision this as becoming the *preferred* pathway for software because it is optimized for software. However, traditional acquisition pathways would still be available.

CONTACT

Jeff Boleng | SWAP Study Program Manager
jeffrey.l.boleng.civ@mail.mil | Office (703) 571-9029

SWAP Study Staff
Courtney Barno | courtney.e.barno.ctr@mail.mil | (703) 614-5147
Devon Hardy | devon.k.hardy2.ctr@mail.mil | (703) 614-5148
Sandy O'Dea | Sandra.l.odea.ctr@mail.mil | (703) 614-5139