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

Defense Innovation Board (v2.2, 15 Feb 2019)

Extended Abstract

Software is ubiquitous and 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, and deploy software is thus central to national defense. At the same time, the threats that the U.S. faces are changing rapidly, and DoD's ability to adapt and respond is defined 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 the tools they need to assure 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[1] 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, this study has taken the approach of 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 and 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 itself will do anything, 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 iteration of our manifesto, we argue three fundamental points. First, **speed and cycle time are the most important metrics for software**. To maintain advantage, DoD needs to deploy and update software that works for its users at the speed of mission need, and execute inside the OODA loop of our adversaries. While it is always possible to go slower when speed is not essential, statutes, regulations and cultural norms that get in the way of deploying software to the field quickly (weeks to months, not years to decades) weaken our national security and expose our nation to risk. Second, **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. And third, **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 and evolving capability that must be supported and continuously improved throughout its lifecycle. DoD's acquisition process and culture need to be streamlined for effective delivery

[1] We actually tried to count them: see Chapter 3 (Been There, ~~Done~~ Said That).

and oversight of multiple types of software-enabled systems, at scale, and at the speed of relevance. Optimizing for software is the path forward.

In order to take advantage of the opportunities enabled by software, we recommend four primary lines of effort. First, the **Congress and DoD must streamline the statutes, regulations, and processes for software** (funding, developing, procuring, testing, and fielding), thus 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. The management and oversight of software development and acquisition must be “refactored,” focusing on different measures and a quicker cadence. Second, it will also be necessary to **create 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. Third, the Services will need to **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 finally, both **DoD and industry must change the culture of how software is procured and developed** by adopting “DevSecOps” practices and approaches, prioritizing speed as the criticalmetric.

This is all easily (and often) said but rarely done, so we try here to document a snapshot in the conversation about the path to achieve this vision, through specific changes to the statutes, regulations, processes, and cultures that define the acquisition process for software.

Table of Contents[2]
v2.1, 15 Feb 2019

[2] Draft outline for the report; page numbers reflect target length for each chapter

End of Overall Report: 32 pages (main) + 160 pages (SI) = 192 pages

Chapter 0. README

v2.1, 15 Feb 2019

Software is ubiquitous and U.S. national security is critically dependent on the capabilities of its software. To maintain our military advantage, the Department of Defense (DoD) must be able to develop, procure, deploy, and continuously improve software faster than our adversaries. Recognizing that not all “software” is the same – it can range from off-the-shelf, non-customized applications to highly-specialized, embedded code running on custom hardware – it is critical that the right tools and methods be applied for each type. Unfortunately, DoD practices lag significantly behind the best practices used in the private sector. Commercial industry has demonstrated that software can have a transformative impact on business and society. Companies that thrive take advantage of software, computing, and networking – and the rapid cycles of improvement they allow and enable – to the maximum extent possible. At the present time, DoD’s software prioritization, planning, and acquisition processes are among the worst bottlenecks for deploying capability to the field at the speed of relevance. This puts the U.S. Armed Forces at risk, reduces the efficiency of DoD operations, and drives away the very people who are most needed to develop software that is critical to national security.

What this report is about. This manifesto describes the output of the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study. The DIB was charged by Congress[1] to recommend changes to statutes, regulations, processes, and culture to enable the better use of software in the 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 current versions of these are included in Appendix A). We used those 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 report captures key insights from these discussions in an easy-to-read format that highlights the elements that we think are critical for the Department’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 one-page summary of twelve months of work for those not likely to read the full report; please take the time to read it.
- **README** (this document): a more detailed five-page summary of the report. 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 and you should be good-to-go. (A README file is used by the open source software community to provide essential information about a software package.)
- **Recommendations Cheat Sheet:** A list of the primary lines of effort and key recommendations, so you can pretty much stop at that point – or better yet, stop after suggesting to your boss she adopt them all.

[1] 2018 NDAA, Sec. 872. Defense Innovation Board analysis of software acquisition regulations.

- **FAQ** (frequently asked questions): 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…)
- **Chapters 1-4:** short descriptions of key areas about which we felt it important to expound. 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 ourrationale.
- **Supplementary Information:** To ensure that the main body of the report satisfies the staple test[2] and the takeoff test, [3] we put most of the additional information generated during the study in a set of appendices. These provide a wealth of examples and evidence, but we took care to put our essential arguments up front for less wonky types.

Note: if you are reading any portion of the report in paper form, a navigable version is available at http://innovation.defense.gov/software (hyperlink version coming soon).

Key Themes. In order for the report to be useful, we felt we should come up with a few key themes that could be used to drive home the message of the report. Here they are (again):

1. Software is ubiquitous and U.S. national security relies onsoftware.
2. Speed and cycle time are the most important metrics forsoftware.
3. Software is made by people and for people, so digital talentmatters.
4. Software is different than hardware (and not all software is thesame).

Software is ubiquitous and U.S. national security relies on software. 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 the DoD are all contingent upon our software edge and any lack thereof presents serious consequences. Software drives our military advantage: what makes weapons 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.

[2] Any report that is going to be read should be thin enough to be stapled with a regular office stapler.
[3] Reports should be short enough to read during takeoff, before the movies start and drinks are served.

Our adversaries are active players in the world of software and so they are increasingly able to develop weapons systems faster than we can, capitalizing on their advantage in software development. Meanwhile, they exploit our vulnerabilities via cyber attacks to steal, undermine, and inhibit our capabilities. 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 the Department can refactor its acquisition processes and adjust its culture and personnel policies before its too late, this software-savvy generation can still set the Department on the right course.

Speed and cycle time are the most important metrics for software. Most software projects in DoD are currently managed using "waterfall" develop processes, which involve spending years on developing requirements, taking and selecting bids from contractors, and then executing programs that must meet the listed requirements before they are "done." This results in software that takes years to reach the field and 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 by 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 efforts to deploy greater capability, sooner. Faster gives us a tactical advantage on the battlefield because we can operate and respond 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, [4] 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. In DoD proper, we do 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

[4] Steve Jobs - The Lost Interview, 2012.

specialities that are part of the overall software ecosystem need to be treated like Special Forces; 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 developed a sophisticated set of statues, regulations, and instructions that govern the development, procurement, and sustainment of defense systems. This process was developed in the context of the Cold War, where major powers developed aircraft carriers, nuclear weapons, fighter jets, and submarines that are extremely expensive, last a very long time, and require tremendous access to capital and natural resources. Software, on the other hand, 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 thanhardware.

Primary Lines of Effort: The Most Important Things to Do. The Department's current approach to software is a, if not *the*, 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 the primary sources 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 process to develop, procure, deploy, and continuously improve DoD software, in this case 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, the Armed Services, defense contractors, and the myriad of government and industry organizations involved in getting software out the door need to make major changes (together). Here are the primary Lines of Effort that we recommend beundertaken:

1. **Streamline 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 quickercadence.
2. **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.

3. **Create new paths for digital talent (especially *internal* talent)** by establishing software development as a high-visibility, high-priority career track with specialized recruiting, promotion, organization, incentives, andsalary.

4. **Change the culture of how software is procured and developed** by adopting best practices from the private sector and focusing on iterative and ongoing development of software, security as a key performance parameter, and responsive engagement with the user (DevSecOps).

None of these can be done by a single organization within the government. They are going to 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 (theme 1), the need for speed and agility (theme 2), the critical role that smart people have to play in the process (theme 3), and the impact of inefficiencies of the current approach (theme 4). In many ways this mission is as challenging as any combat mission: while participant's 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 the ability of the Department to redefine its approach to delivering combat-critical software to the field.

New statutes, regulations, and processes, streamlined for software. Congress has created many workarounds to allow the DoD to be agile in its development of new weapons systems, and the Department 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 with modern software (or frankly necessary, given the threat environment). 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 one that characterizes the best of modern software innovationcycles.

Primary recommendations:[5]

- Recommendation Ax:
- Recommendation Ay:
- Recommendation Az:

[5] Not yet finalized; see the "Recommendations Cheat Sheet" for the current list of possibilities.

Cross-program/cross-service digital infrastructure: Current practice in DoD programs is for each individual program to build 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 Department leadership must figure out ways to incent the Services and defense contractors to build on a common set of tools (instead of inventing their own) *without* just requiring that everyone use one DoD-wide (or even service-wide) platform. Similarly, OSD is going to have to define non-exceptions-based alternatives to (or at least pathways through) JCIDS, PPB&E, and DFARS[6] that are optimized for software. DOT&E will need new methods for operational test and evaluation that match the software's speed of relevance, and CAPE is going to have to capture better data and leverage artificial intelligence/machine learning (AI/ML) as a tool for cost assessment and performance evaluation. Finally, the Services are going to 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.

Primary recommendations:[7]

- Recommendation Bx:
- Recommendation By:
- Recommendation Bz:

New paths for digital 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 agile 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 undergoing and that the government must as well.

Increasing developer talent is not the only workforce challenge. DoD must also change how programs and contractors are managed, which goes beyond just moving to agile development. The government must have experts well steeped in the software development process and

[6] Common DoD acronyms are defined in Appendix F (Acronyms, Inside Jokes, and Catch Phrases).
[7] Not yet finalized; see the "Recommendations Cheat Sheet" for the current list of possibilities.

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 being wasteful. This is perhaps the argument for having 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 even more 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 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, medicine, and law, great teams make a huge difference in software and 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. To get started, Congress might create a two-year national security waiver from the GS system in selected digital technology areas required for software, and the Services should use this and other authorities to identify and nurture civilian and military talent with software development expertise. A key element of success is finding ways to keep talented people in their roles (rather than transferring them out because it is the end of their assignment), and promote people based on their abilities, not based on their years of service.

Primary recommendations:[8]

- Recommendation Cx:
- Recommendation Cy:
- Recommendation Cz:

Changing the culture around software acquisition. The items above are where we think Congress and the Department should focus as the three primary Lines of Effort. Without dramatic change, the rate at which we can make improvements is far outpaced by the rate at which the problem itself gets worse. With demonstrated progress on these three there is then a long list of other things that need to be done, ranging from changing the law to changing the way people work. We created a list of 30 recommendations for change that we thought were important, and then asked everyone with whom we interacted in building this report to vote on the ones they thought would make the most difference. Here is the current snapshot of the top 10 recommendations that are not already part of our primary lines of effort, based on that voting and our judgement:

Rank	Recommendation	👍	👎
	This table will be filled in for the final report		

[8] Not yet finalized; see the "Recommendations Cheat Sheet" for the current list of possibilities.

	The items here will come from a longer list of recommendations (see cheat sheet)		
	The order will be determined using a leaderboard		
	Participants in SWAP study activities will be allowed to cast a vote		
	More details coming later; look at the full list of options (cheat sheet) for now.		

More details on these (as well as top 10 lists for the biggest barriers and the most useful tools that are not currently available for use) are included in Chapter 5 (What Would the DIB Do) and the supporting information (Appendix E).

Getting started now. The types of changes that 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, 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 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 do. The 1987 Defense Science Board Task Force on Military Software,[9] chaired by legendary computer scientist Fred Brooks, wrote an outstanding report that already articulated most of what we are saying here. And 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 tooquickly.

Many of our DoD issues could be addressed by adopting existing best practices of the private sector for agile development, software as a service, use of modern (cloud) infrastructure, tools, computing and shared libraries, and 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 are some specific suggestions for what to do starting *now*:

- FY19 (create): High-level endorsement of report vision 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 (repeat: 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 2-3 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 say yes. You are the front line in changing to a "culture ofyes."

[9] 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.

- FY20 (deploy): Initial deployment of authorities, budgets, and processes for SWAP reform. Execute representative programs according to the themes, flavors, and recommendations in this report, implement now, measure results, and modify approaches. Let’s implement this report the way we implement modern software.
- FY21 (scale): Streamlined authorities, budgets, and processes enabling SWAP reform at scale. In this time frame, we need a new methodology to estimate as well as determine the value of software capability delivered (something not based on lines of code).
- FY22 (optimize): All DoD software development projects transition (by choice) to software-enabled processes, with talent and ecosystem in place for effective management and oversight.

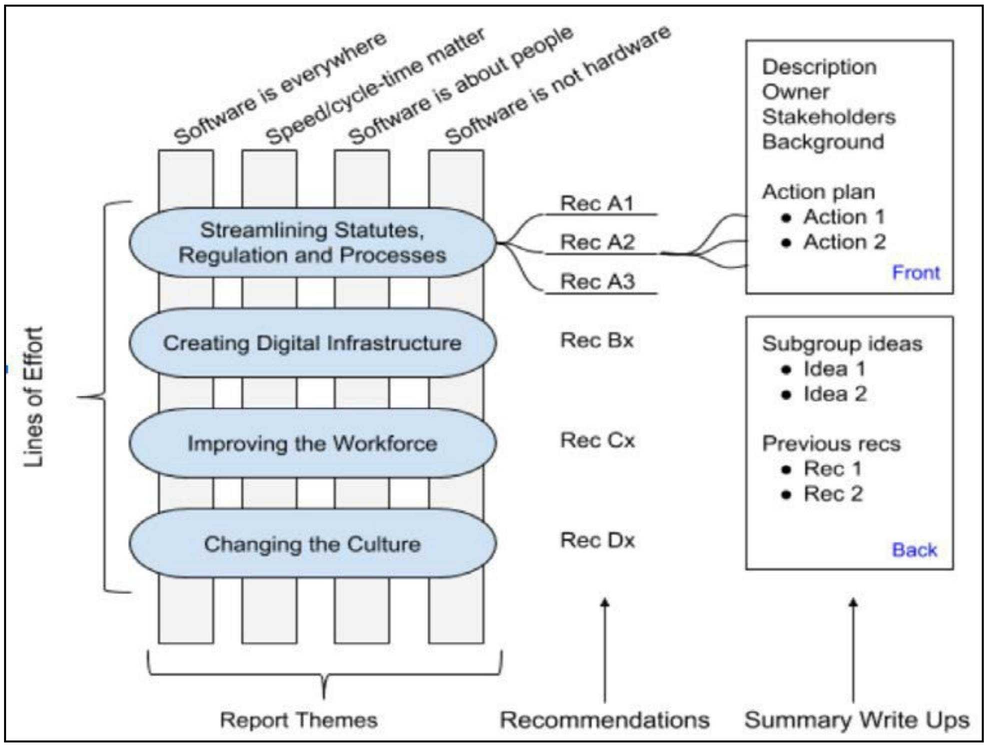
DIB SWAP Study
Recommendations “Cheat Sheet”

v1.2, 12 Feb 2019

This section contains a list of the (preliminary) *potential* recommendations for the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study. The recommendations below include input from the following sources:

- DIB SWAP concept papers (Ten Commandments, Do’s and Don’ts,Observations)
- DIB SWAP working group reports
- 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 action plan, a list of other offices that are affected, and additional details. The following diagram documents this structure:



For each recommendation below, a one page summary is provided that gives more detail on the rationale, supporting information, similar recommendations, and specific action items, and notes on implementation. Potential legislative and regulatory language to implement selected recommendations is included in Appendix L[1].

[1] Appendix L is not yet finalized or all-inclusive

ID	Rec	Lead Org	Time req'd
Line of Effort A (Congress and OSD): Streamline 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.			
A1	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	A&S via HAC-D SAC-D	1 yr pilot; 3 yr full
A2a	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.	PM	now
A2b	*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	HASC SASC	1-3 yrs
A2c	Establish a *new acquisition pathway* (Sec 805) for software that prioritizes 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	A&S via HASC SASC	3-5 yrs
A3	Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially-available products with minimal (source code) modification	CMO	6-12 mos
A4	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.	Comp	1-3 yrs
A5	Replace JCIDS, PPB&E, and DFARS with a "PEO Digital" in each Service that uses portfolio management and direct identification of warfighter needs to decide on allocation priorities	JCS	1 yrs
A6	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 functionality.	CAPE	3-12 mos
Line of Effort B (OSD and Services): 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.			
B1	Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field and make available to contractors at subsidized cost	CSx	

B2	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 a iterative basis (with frequency dependent on type of software, but targets cycle times measured in weeks)	OT&E	
B3	Prioritize secure, iterative, collaborative development for selection and execution of all new software programs (and software components of hardware programs) (see Agile BS detector as an initial view of how to evaluate capability)	A&S	
B4	Create a mechanism for ATO reciprocity within and between services to enable sharing of software platforms, components and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapons) systems, and Services	CIO	
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	CIO	
B6	For any software developed for DoD, require that software development be separated from hardware in a manner that allows non-prime vendors to bid for software elements of the program on a performance-based basis	A&S	
B7	Shift from certification of executables, to certification of code and 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	CIO	
B8	Plan and fund computing hardware (of all types) as consumable resources, with continuous refresh and upgrades to the most recent, most secure OS and platform components	Comp	
Line of Effort C (Services): Create new paths for digital talent (especially *internal* talent) by establishing software development as a high-visibility, high-priority career track with specialized recruiting, promotion, organization, incentives, and salary.			
C1	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	CSx	
C2a	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	A&S	
C2b	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	A&S	
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	A&S	
C4	Restructure the approach to recruiting software developers to assume that the average tenure of a talented engineering will be 2-4 years, and make better use of HQEs, IPAs, reservists and enlisted personnel to provide organic software development capability	CSx	

Line of Effort D (Acquisition Offices and Contractors): Change the culture of how software is procured and developed by adopting best practices from the private sector and focusing on iterative and ongoing development of software, security as a key performance parameter, and responsive engagement with the user (DevSecOps).			
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	A&S	
D2	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)	A&S	
D3	Establish a Combat Digital Service (CDS) unit within each combatant command consisting of software development talent that can be used to manage command-specific IT assets, at the discretion of the combatant commander.	JCS	
D4	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	A&S	
D5	Make security a first-order consideration for all software-intensive systems, under the assumption that security-at-the-border will not be enough	CIO	
D6	Shift from a list of requirements for software to a list of desired features and required interfaces/characteristics, to avoid requirements creep, overly ambitious requirements, etc	A&S SAE	
Additional recommendations that do not fit the primary lines of effort listed above			
E1	Maintain an active research portfolio into next-generation software methodologies and tools, including the integration of machine learning and AI into software development, cost estimation, security vulnerabilities and related areas	R&E	Cult
E2	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)	R&E	
E3	Automatically collect all data from DoD weapons systems and make available for machine learning (via federated, secured enclaves, not a centralized repository)	A&S	
Previous recommendations that we support and can't improve on (incomplete)			
● DSB (1987): Rec 19: DoD should develop metrics and measuring techniques for software quality and completeness, and incorporate these routinely in contracts			
Previous recommendations we don't agree with (incomplete)			
• DSB (1987): Rec 5: Commit DoD management to a serious and determined push to Ada			

SWAP FAQ (Frequently Asked Questions)

v0.2, 17 Feb 2019

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 willoccur.
   - 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 (includingoversight).
   - 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 theywant?**

   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 defensesoftware.

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 inindustry.**

   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 the 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 smartphone 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 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., the 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. We are working on a mockup of an alternative……

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

v0.3, 18 Feb 2019

This chapter provides a high-level vision of why software is critical for national security and the types of software we are going to 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 wegoing?

While software development has always been a challenge for the Department, today these challenges are greatly affecting 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 weapons systems; command, control and communications systems; intelligence systems; logistics; and infrastructure. If the new domain that we are fighting in is cyber, 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 the enabler as well as the target of thefight.

As our military systems become increasingly networked and automated, as autonomy becomes more prevalent, as we become more dependent on machine learning and artificial intelligence, then 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. In this new world, digital threats are more prevalent and, in many cases, more effective than physical and kinetic threats alone. Digital capabilities bring new dimensions to asymmetric and hybrid warfare and nation states are investing in new capabilities to gain parity if not superiority over the United States. Even our ability to defend against new physical and kinetic threats like 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 do so will be based on our ability to develop and push new software defined capabilities to meet those threats on time scales that greatly out pace our adversaries' ability to doso.

The ability to meet future threats requires us to rethink how we procure, design, develop, deploy, and maintain software. We can no longer take years to develop software for our major systems. Software cannot be an afterthought to hardware and it cannot be acquired, developed, and managed like hardware. Our current procurement processes treat software programs like hardware programs. Our acquisition and development approaches are antiquated and do not meet the demands of the Department. Fixing our software approach in the Department is more than just making sure that we get control over cost and budget, it's about 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 it is important to optimize development processes and oversight mechanisms to the different types of software that are used by DoD. We distinguish here between two different aspects of software: their *operational function* (use) and their *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).

The following glossary of terms provides some characterization of these systems and their important properties:

- *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 Department 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 without modification to the underlying code, but with DoD-specific configuration. Examples include: e-mail systems, accounting systems, travel systems, and 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. Examples include: software development environments, 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 the DoD, not the least of which will be specialized cybersecurity considerations to enable them to continue to function during an adversarial attack. We further break down combat systems into subcategories:
    - *Logistics systems*: any system that is 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 COTS technologies. Platform-based architectures enable integration of new capabilities 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, but may be run in a more localized form (such as an air operations center) that precludes the use of some types of cloud computing infrastructure, but may still 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).

- *Weapons system*: any system that is directly involved in the delivery of 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 of a weapons 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 weapons systems is likely closely tied to hardware.

We also define several different types of computing platforms on which the 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 will always be running on commercial hardware and using commercial operating systems, and the applications running on them will 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 user.

- *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 will usually be running on commercial hardware and using commercial operating systems.

- *Desktop/laptop/tablet computing*: computing that is carried out on a single system, often by interacting with data sources across a network. These systems will usually be running on commercial hardware and using commercial operating systems.

- *Embedded computing*: computing that is tied to a physical, often-customized hardware platform and that has special features that requires careful integration between software and hardware.

Note that a single software system may have multiple components or functions that cross these definitions and there may be components of an integrated system that have elements that cross these definitions. The key point is that each type of software system will have different requirements in terms of how quickly it can/should be updated, the level of information assurance that is 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 will differ greatly from what is used for hardware).

For the purpose of this report, we distinguish between four primary types of software, which we use throughout the rest of the report to differentiate the approaches that are needed:

- **Type A (Commercial Off The Shelf (COTS) apps):** The first class of software consists of applications that are available from commercial suppliers. Business processes, financial management, human resources, software development and collaboration tools; accounting 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, those processes should be modified, not the software.

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

- **Type C (COTS HW/OS):** The third class of software applications is those that are highly specialized for DoD operations but can 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 that are written by DoD personnel as well as those that are developed by contractors.

- **Type D (Custom SW/HW):** 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 DSB study on military software, where they categorized software as "standard" (roughly capturing types A and B), "extended" (type C), "embedded" (type D), and "advanced" (which they categorized as "advanced and exploratory systems", which are not so relevant here).

1.3 What kind of software are we going to have to build?

Speed is the discriminator for all things software. We must shorten our development cycles from years to months. We need to react and respond within the speed of the threats. We need to embrace agile development methodologies. We need to rethink how we test and validate software and move toward a more integrated approach to testing. Quality assurance needs to be a continuous and fully integrated process throughout every phase of the software cycle. We need to build software logistic trains that are able to develop, 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 need to treat software as a continuous service rather than as block deliverables. We also need agility in our procurement approach that allows program managers to change priorities based on the needs and timing of the end users.

The Department needs to manage software by measuring value delivered to the customer rather than by monitoring requirements. Accountability should be for delivering value to the customer and solving the customer's needs, not by complying with obsolete contracts and requirements documents.

Program managers need to identify potential problems earlier (ideally, within the first year) and take corrective action quickly. Troubled programs need to fail quickly, and we need to learn from them. Many software programs are too big, too complex, too long, and with too many requirements. Development needs to be staged and follow the best practice of smaller, quicker deliverables with higher frequency of updates and new features. Initially, program development should focus on developing the minimum viable product delivered more quickly to the customer than traditionally run programs.

Software developers within our defense community need the modern tools, systems, environments, and collaboration resources that commercial industry has adopted as standard. Without this, we are undermining the effectiveness of our software developer base, and our ability to attract and retain our software human capital, both within the Department and among our suppliers. With the introduction of new technologies like machine learning and artificial intelligence and the ever-increasing interdependency between networked heterogeneous systems, software complexity will continue to increase logarithmically. We need to continuously invest in new development tools and environments including simulation environments, modeling, automated testing and validation tools. We must invest in research and development into new technologies and methodologies for software development to help the Department keep up with ever growing complexity of defensesystems.

1.4 What are the challenges that we face (and consequences ofinaction)?

The world is changing. Software used to be the exclusive province of the United States. 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. They want to become a cyber superpower and are investing in their capital markets, universities, research centers, defense industry, and commercial software companies.

The long-term consequence of inaction is that our adversaries' software capabilities can catch and surpass ours. If that becomes true, then our adversaries will be able to develop new capabilities and iterate faster than we can. They can respond to our defense systems faster than we can respond to theirs. If their algorithms and AI becomes superior to ours, it means that they can hold a decisive advantage where any of our systems goes up against any of theirs. If their cyber capability becomes superior to ours, then they can shut us down, cause chaos, and continue to steal our secrets at their choosing and without repercussion – especially if we cannot attribute those attacks. Our adversaries' software capabilities are growing as ours are stagnating. If we don't change now, we could lose our defense technology advantage within a decade or much sooner.

Chapter 2. I Don’t Get It: What Does It Look Like to Do Software Right?

v0.3, 18 Feb 2019

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 focus. The 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 isjustified.

2.1 How it works in industry (and can/should work in the DoD): DevSecOps

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 industry, the following set of practices – based on documented approaches at Google– are representative of commercial environments where the delivery of software capability determines the commercial 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 – e.g. automotive, banking, health care, and many others. In any 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 lessonslearned.

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 post-mortems. Some of the key best practices that are applicable to DoD software programs include:

- 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 and focus work on the main branch of the software development.
- Code review tools are reliable and easy to use. Changes to 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 dailyworkflow.
- Releases are frequent - often weekly. There is an incremental staging process over several days, particularly for high-traffic, high reliabilityservices.

- Post-mortems are conducted after system outages. The focus of the post-mortem is on how to avoid problems in the future and not about affixingblame.

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, quarterly and annually. Progress against those goals are tracked, reported and shared across the organization. Goals are mechanisms to encourage high performance but can be decoupled from performance appraisal or compensation.
- Organic project approval process. 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. Some examples include:

- Clear separation between engineering and management roles, with advancement paths for both. 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 thecompany.

In addition to these specific software development practices, another common approach to managing programs in industry is to move away from the typical DoD specifications and requirements approach towards a portfolio management approach.

The portfolio management approach allows program managers to make agile decisions based on evolving needs and capabilities. Using a portfolio management approach, a program manager has a list of features and capabilities ranked by need, risk, cost, resource, 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 feature mix, matching investments to needs and balancing risk against performance. Needs are driven tactically by end users and strategically by the services. Capabilities are tested and delivered on a continuous basis, and maximum automation is leveraged for testing.

In industry, software programs initially start as a minimum viable product (MVP). A minimum viable product 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 users need and strategic objectives within an investment cycle. Value is determined by output measurements rather than process measurements. Portfolio value is the aggregation of 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 will 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 varies greatly between software engineers, designers, programmers, and manager. The quality of engineering talent is the single largest variable that determines cost, risk, and time 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 the software capabilities the Department so desperately needs is how the Department manages the people necessary to build that capability. You cannot compete and dominate in 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 (which is of critical importance), it must become competent at recruiting, retaining, leveraging, and developing the people who make it and managing those who do. When we look at organizations and institutions that effectively use software to fulfill their mission, each of them:

- Understands the software professionals that it has, understands at a high level what it needs, and 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 to too much precision;
- Has a strategy to recruit the people and skills it needs to fulfill its mission, understanding what it uniquely has to offer in a competitive market;
- Has clear understanding of the competencies required by software professionals in the organization and the expectations of these professionals at each level in the organization;
- Has defined career ladders for both uniformed (via the MOS system) and civilians (via the GS system) that map software competencies and expectations from entry level to senior technical leadership and management;
- Offers opportunities for learning and mentorship from more senior engineering and design leaders;

- Counts engineering and design leaders among its 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 withboth;
- Supports 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;
- Is able to reward software professionals based on merit and demonstrated contribution rather than time in grade.

These are not descriptors for the software workforce in today's DoD.

The Department has, however, 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 functions. We would not expect a trained physician to constantly rotate into deployments focused on aviation maintenance or construction, nor would we interrupt the training of a physician to teach her artillery or carpentry. 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, the Department's current human capital practices do all ofthese.

The process to retool human capital practices to meet the challenge of software competency in the Department must start with the people the Department already has who have software skills or are interested in acquiring them. Unlike medicine, software skills can be acquired through self-directed and even informal training resources, and the Department has individuals, military and civilian, who have taken it upon themselves to gain technical skills outside of or in addition to formal Department training. This kind of initiative and aptitude, especially when it results in real contribution to the mission, should be rewarded with appropriate career opportunities for advancement in this highly sought-after specialty. 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. Possible human capital pathways, might include:

- a core Occupational Series (Civilian) for software development that includes subcategories to address the various duties found in modern software development (e.g., developers/engineers, product owners, designers,etc.)
- 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 acquisition professionals (military and civilian) that indicates they have the necessary experience and training to serve on a software acquisition team. This Identifier or Endorsement needs to be a mandatory requirement to lead the acquisition team for any 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.

In addition to supporting the people the Department has today, both those already working in software and those who could, the Department will need to attract and retain many more, and more qualified, software developers and, particularly, more software leaders. Again, the creation of defined career ladders that recognize and reward the appropriate competencies for each of the major specialties on a software team is table stakes for effective recruitment. Also effective will be the demonstrated ability to leverage, recognize, and reward software developers more flexibly than the Department currently allows for so that the strongest contributors can be put on the most critical projects and can be retained within the Department even when their skills become highly valued in the private sector. In addition, our recommendations cheat sheet contains over a dozen ways that the Department can improve its technical recruiting, including the idea of giving all new recruits a software aptitude test to identify potential trainees.

2.3 Getting it right: superior national security AND betteroversight

In August of 2011, Venture Capitalist Marc Andreessen famously penned an op-ed for the Wall Street Journal entitled, “Why Software is Eating the World.” In it, he noted that “Six decades into the computer revolution, four decades since the invention of the microprocessor, and two decades into the rise of the modern Internet, all of the technology required to transform industries through software finally works and can be widely delivered at global scale.” He argued that *every* industry (not just those considered to be “information technology” in the traditional sense) would be transformed by software – bytes rather thanatoms.

This transformation will happen in defense, whether or not we are prepared for it. Software is the focal point of many important advances in national security technology, including data analytics, artificial intelligence, machine learning, and autonomy. If these fields are important to US security, then software development and acquisition done correctly will be critical to maintaining the US’ national security superiority into the nextdecade.

Software levels the national security playing field with our rivals. The US’ traditional advantages in human resources, infrastructure, national resources, etc., are less important in the age of software.

First, unlike hardware-focused defense R&D, software development is not capital/resource intensive and many advances are largely available via open source licensing. This means that non-traditional powers like Iran and North Korea, not to mention China/Russia and non-state actors, can quickly ‘catch up’ to the US in areas such as AI, autonomy, and data analytics, by deploying small talented teams to quickly develop capabilities to match or surpass our own in these fields. Due to open source availability, they can also easily incorporate others' recent advances.

Second, and relatedly, software lessens the impact of the US' existing advantages in hardware and infrastructure. For example, powerful software will enable non-traditional powers to use thousands of inexpensive autonomous drones to chip away at the US' aerial superiority, without ever owning or developing a single advanced fighter plane. Similarly, the US' fleet of advanced sensors, satellites, and other ISR tools have limited utility without computer vision, machine learning, or data analytics software to make sense of the data collected.

Finally, software is disproportionately talent-driven – ensuring that the winner of the talent race will win the software race (as addressed in the previous chapter). Access to superlative engineering talent is by far the most important single factor determinative of success or failure in a software project. 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 work on those applications. China has made great strides in this regard by leveraging its private industry to develop national security software (particularly in AI), recruiting top students under 18 to work on "intelligent weapons design", and by poaching US software talent directly from the US. Similarly, speaking to students over video simulcast on the first day of the Russian school year in September 2017, Vladimir Putin said, "Artificial intelligence is the future, not only for Russia, but for all humankind. It comes with colossal opportunities, but also threats that are difficult to predict. Whoever becomes the leader in this sphere will become the ruler of the world."

But getting software right in the Department isn't as simple as just recognizing that it is a national security priority; oversight (and budgeting and finance) must alsochange.

Agile projects that use modern software approaches can be expected to deliver value to the user faster than alternative approaches. Oversight of monolithic, waterfall projects has generally focused around whether the team hit pre-determined milestones that may or may not represent actual value or even working code, and trying to figure out what to do when they don't. When evaluating and appropriating funds to agile projects, it's more appropriate 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 cause for concern. In a project that is agile and takes advantages of the other approaches the DIB recommends (including software reuse), the plan is intended to be flexible because the team should be learning what works as they code and test. Successful projects will develop metrics that measure value to the user, which involves close, ongoing communication with users. Notably, Source Lines of Code does not equal value. (SLOC ≠value).

Have a leader and hold them accountable. Great program outcomes generally emerge from exceptional leaders who are fully on the hook for delivering on their vision. The mythology around the impact of top founders is widely and commonly accepted with regards to private companies (Figure 1), but is actually just as applicable in the public markets (Figures 2 and 3).

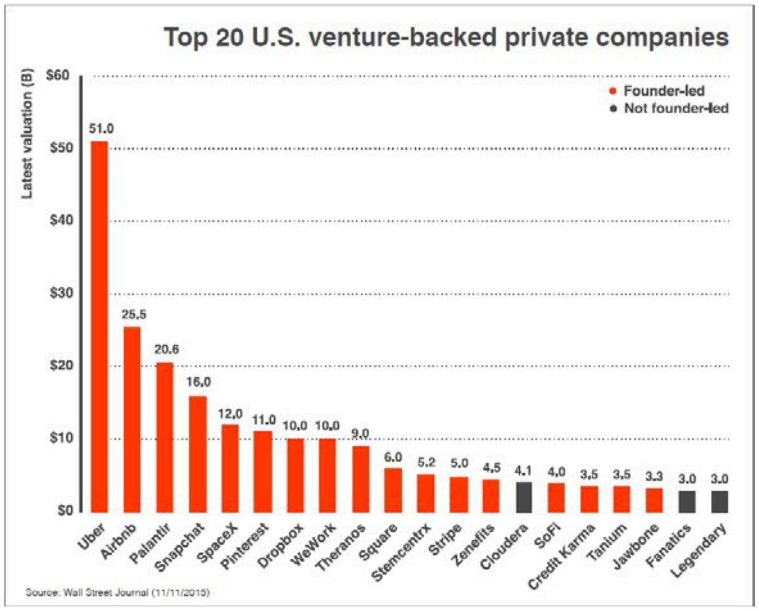


Figure 1. Valuations of top 20 venture-backed private companies, comparing founder-led (red) to non-founder-led (black).

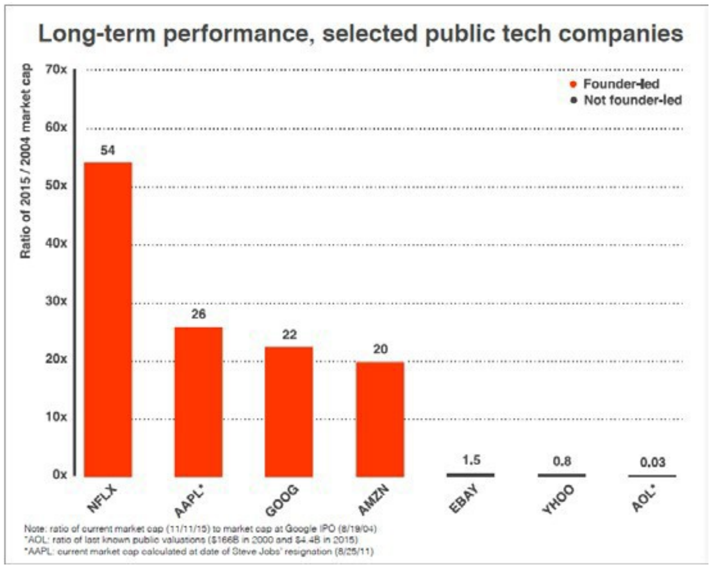


Figure 2. 2004 to 2015 market cap ratio, comparing founder-led (red) to non-founder-led (black).

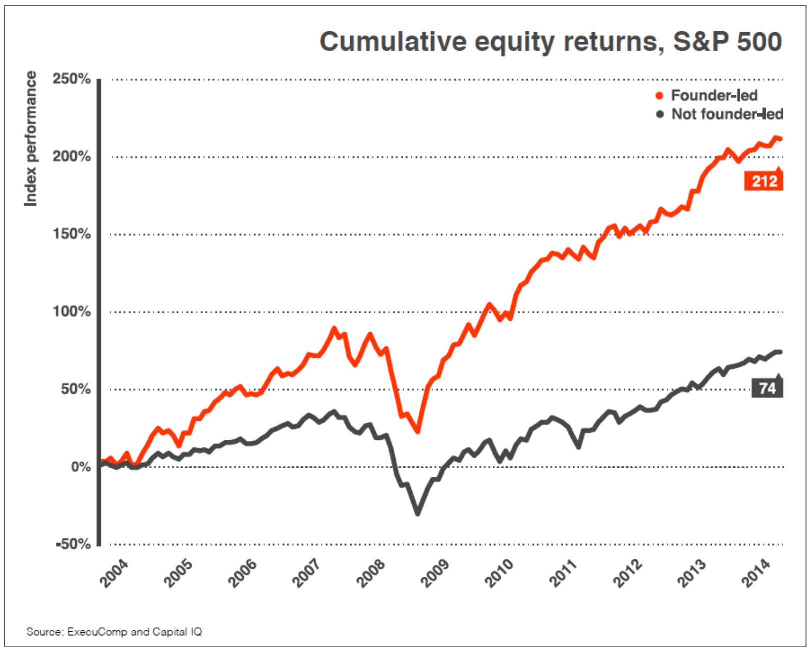


Figure 3. 2004 to 2015 index performance, comparing founder-led (red) to non-founder-led (black).

This is just as applicable to the public sector as it is to the private sector and has become somewhat of a lost art form. Many of the most noteworthy defense programs over the past decade have been shepherded by exceptional "founders". Kelly Johnson with the U-2, F-104, SR-71. Paul Kaminski with stealth technology. Admiral Hyman Rickover with the nuclear navy. Harry Hillaker with the F-16. 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[1] - Assign One Leader and Hold That Person Accountable. We would do well to remember this part of our history and work this into our oversight plan.

Speed Increases Security. As we have learned from the cyber world, when we are facing active threats, our ability to have 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 Artificial Intelligence and Machine Learning 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."*

[1] https://playbook.cio.gov/#plays_index_anchor

GAO-19-128: Published: Oct 9, 2018. Publicly Released: Oct 9, 2018.

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

We must be able to do so without sacrificing our abilities to test and validate software. To accomplish this, we need to re-imagine the software development cycle as a continuous flow rather than discrete software block upgrades. We need to not only modernize to the agile methodology of software development, but we must also modernize our entire suite of development and testing tools and environments. We need to be able to instrument our 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 into how money is to be spent make a lot of sense. But, because software is in continuous 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.

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

v0.3, 18 Feb 2019

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 their 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, though the scope of software has now moved well beyond weapons 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. … the 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, already referenced over 30 previous studies and is largely aligned with the assessments of more recent studies as well as this study.

And finally, in its 2000 study on DoD software, 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.

The problem is not that we don't know what to do, but that we simply aren't 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 Brief summary and assessment of 37 years of reports on DoDsoftware

The following table lists previous reports focused on improving software acquisition and practices within DoD, along with a "grade" indicating how well the recommendations are aligned with our own report, using the following scale:

- A = Better than we could have said it ourselves; highly recommendedreading
- B = Very aligned with our proposed approach
- C = Satisfactory report, though some aspects may not be wellaligned/articulated
- D = Pushing in the wrong direction

Date	Org	Short title / Summary of contents	Grade
Jul'82	DoD	Joint Service Task Force on Software Problems ● 37 pp + 192 pp SI; 4 majorrecs ● Software represents importantopportunity ● DoD should take a lead in embeddedsoftware	B
Sep'87	DSB	Task Force on Military Software 41 pp + 36 pp SI; 38 recommendations Vision for rapid development and deployment of software, moving away from waterfall model	A
Dec'00	DSB	Task Force on Defense Software TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	B
2004	RAND	Attracting the Best: How the Military Competes for Information Technology Personnel TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
Feb'08		Generational Inertia - An Impediment to Innovation? TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
2010a	NRC	Achieving Effective Acquisition of Information Technology in the Department of Defense TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
2010b	NRC	Critical Code: Software Producibility for Defense TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
Jul'16	CRS	The Department of Defense Acquisition Workforce: Background, Analysis, and Questions for Congress TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
Dec'16	CNA	Independent Study of Implementation of Defense Acquisition Workforce Improvement Efforts TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
Feb'17	SEI	DoD's Software Sustainment Study Phase I: DoD's Software Sustainment Ecosystem TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD

Mar'17	BPC	Building a F.A.S.T. Force: A Flexible Personnel System for a Modern Military TBD: XX pp + YY major recs TBD: 2-3 line summary of what the report covers and key insights/takeaways.	TBD
Feb'18	DSB	Design and Acquisition of Software for Defense Systems 28 pp + 22 pp SI; 7 (high-level) recs + ~32 subrecommendations Transition to the use of software factories and continuous iterative development for DoD software; expand acquisition workforce knowledge of software	A
2018	2016 NDAA	Section 809 Panel - Streamlining and Codifying Acquisition [comparison] 1,275 pages, 93 recommendations Comprehensive review of Title 10, FAR, DFARS and recommendations on what needs to change	B[1]
Apr'19	DIB	Software is Never Done; Refactoring the Acquisition Code for Competitive Advantage (this document) 32 pp + 150 pp SI; 4 lines of effort, ~10 recommendations (+ the next 10?) Speed/cycle time as key metrics, build digital talent and infrastructure, avoid one-size-fits-all	B

Studies dating back to at least 1982 have identified software as a particular area of growing importance to the 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 SEI study 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 the DoD are summarized in this quote from the 2010 *Critical Code* study:

> This growth is a natural outcome of the special engineering characteristics of software: 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

[1] This is a very comprehensive report on acquisition reform. We only read the parts related to software.

Prior studies (e.g., [SEI2017]) have commented on the fact that much of DoD software acquisition policy is systems- and hardware-oriented and largely does not take these unique properties into account.

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 was already noting 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 capabilities provided by software have only increased since then.

Major Categories of Prior Recommendations. The SWAP team conducted a literature study 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.

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 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; some advocacy for alternative requirement documentation (product vision)
- Acquisition strategy/oversight: DoD should encourage agencies to pursue business process innovations

The three areas which 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

- Durations should be reasonably short and meaningful and should allow for discrete progress measurement
- Design the overall technology maturity assessment strategy for the program orproject
- Encourage program managers to share bad news, and encourage collaboration and communication
- Require program managers to stay with a project to itsend
- Empower program managers to make decisions on the direction of the program and to resolve problems and implementsolutions
- Follow an evolutionary path toward meeting mission needs rather than attempting to satisfy all needs in a singlestep

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, anddeployment
- 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 non-traditionalcompanies

Recommendations from recent work on workforce issues:

- The service acquisition commands need to develop workforce competency and a deep familiarity of current software developmenttechniques
- 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 strategicadvantage
- Encourage employees to study statutes and regulations and explore innovative and alternative approaches that meet the statutory and regulatoryintent
- 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 inbetween

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. 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) institute mandatory software intensive systems training for the workforce; (2) develop a graduate level program for software systems development and acquisition; and (3) require 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. 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 sun-setting of the Software Intensive Systems Group at the 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 DAWIA career fields, and to develop a plan to address those gaps. This group was composed of representatives from the Services, OSD, and other organizations, including the Software Engineering Institute (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 as to 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 software sustainment organizations) 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 is a persistent and dynamic challenge that demands a continuous enterprise strategy.

3.2 Our interpretation of why nothing happened but why we think our report willmatter

Given the long and profound history of inaction on past studies, we have attempted to create our own "Theory of (Non)Change." Why do we not step up to rational, generally agreed-upon change? We offer the following threedrivers:

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 the 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 are directly frustrated at the command level 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. 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. In our meetings with Congressional staffers we heard two messages over and over. First, 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 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. Second, there is a sense of unrequited return from past changes and legislation. In many cases, Congress believes it has already provided the tools and flexibilities for which the Department has asked. It is perhaps unreasonable to expect a positive response to ask for more when current opportunities have not been fullyexploited.

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 to argue for major new ways of acquiring software and not just attempt to reoptimize 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 is going to be any different? Our approach has been to focus not on the report and its recommendations *per se*, but rather in the series of discussions around the ideas in this report and the people we have interacted with. The recommendations in this report thus 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.

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 what needs to change and optimizing 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 their specific levels of authority and responsibility, can navigate an efficient path to reaching their conclusions on how best to support what is contained here.

3.3 Consequences of inaction: Increasing our attack surface and shifting risk tothe 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, weapons systems, and the bulk of the operational structure on which the Department executes its mission, are now fundamentally software 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.

Finally, we reiterate that 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. (See also Section 2.3 [Speed Increases Security]**.)**

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

v0.3, 18 Feb 2019

The previous three chapters provided the rationale for why we need to *do* (not just say) something different about how DoD develops, procures, assures, and deploys software in support of defense systems. Commercial industry 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 software-denominated 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 actions are provided for each path along with a short analysis of the strengths, weaknesses, opportunities, and threats.

4.1 Path 1: Make the best out 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 major goals of 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 recommendations are those focused on changing the culture and approach by which software is developed, procured, assured, and deployed as well as some of the regulations and processes that should be updated to facilitate these cultural and operationalchanges.

To embark on this first path, DoD should streamline its processes for software, 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 ATOs, easier access to large-scale cloud computing, and use of modern tool chains 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 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 actions that require changes to DoD culture and process, 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, andprocesses.
- 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 deliver to endusers.

- 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 DISA CAP limits, lack of ATO reciprocity, and access to modern software development tools.
- 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.
- 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 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, under the assumption that security-at-the-border will not be 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 machine learning 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 weapons systems and make available for machine learning (via federated, secured enclaves, not a centralized repository).

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 existing groups (e.g., F22, JIDO, Kessel Run) have demonstrated that the flexibilities exist within the existing system to develop/procure, deliver, 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 forsoftware

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 the Department have created massive amounts of laws and regulations that are just slowing things down. This might be OK for hardware, but it is definitely not OK for (most types of) software, as we have articulated in the previous three chapters. In this second, more difficult path to software acquisition and practice reform, we focus on relatively small changes that can and should be made to rewrite selected pieces of old code (= legislation and regulations) that are doing more harm than good. These changes would apply to both software that is acquired as well as software that is built.

First, the 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,"[1] because counter-intuitively, this may be the wrong thing to optimize for at times. Similarly, actions should be taken to ensure that 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. commercial sector – where most of the world's software talent is concentrated – is unwilling to work on national security software. The reality is that DoD has failed to give 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.[2] The DoD seems to love the idea of innovation, but doesn't love taking sizeable bets on new entrants or capabilities. It is interesting to note that Palantir and SpaceX are the only two examples since the end of the Cold War of venture-backed, DoD-focused businesses reaching multi-billion dollar valuations. By contrast, China

[1] From the 2018 Summary of the National Defense Strategy: *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.*

[2] While the overall funding commitments are large—$2 billion dollars 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.

has minted around a dozen new multi-billion dollar defense technologies 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 changes we list in the recommendations below.

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.

The following list provides a set of high-level actions that require some additional changes to DoD culture and process, but also modest changes in current law and existing regulations. These actions build on the recommendations 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), andschedule.
- 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 andprocesses.
- Replace JCIDS, PPB&E, and DFARS with a "PEO Digital" in each Service that uses portfolio management and 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 targets 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) [DevSecOps].
- For any software developed for DoD, require that software development be separated from hardware in a manner that allows non-prime vendors to bid for software elements of the program on a performance-based basis.
- 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 informationassurance.

- Require CIOs, SAEs, PEOs, PMs and any other acquisition roles involving software development as part of the program to have prior experience in softwaredevelopment.
- Restructure the approach to recruiting software developers to assume that the average tenure of a talented engineering will be 2-4 years, and make better use of HQEs, IPAs, reservists, and enlisted personnel to provide organic software development capability.
- Establish a Combat Digital Service (CDS) unit within each combatant command consisting of software development talent that can be used to manage command-specific IT assets, at the discretion of the combatantcommander.

This path takes a more active approach to modifying the acquisition system for software but 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 shouldn't 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. 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 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 thesame).

4.3 Path 3: A new appropriations category/acquisition pathway 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 streamline statutes and regulations for software, providing increased (and automated) 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 them using modern development methods; and eliminate the existing hardware-centric regulations and other barriers. Finally, the Services will need to establish software development as a high visibility, high priority career track with specialized recruiting, education, promotion, organization, incentives, andsalary.

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

- 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; remove cost and schedule triggers associated with hardware-focused regulations and processes.
- Establish a new acquisition pathway (Sec 805) for software that prioritizes 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.
- Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field and make available to contractors at subsidized cost.
- 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 appropriations title and acquisition pathway that is 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 the DoD, since it will require that they retool their business to a very new approach that is not well-defined at the outset.

Recommendation Summaries

v0.1, 19 Feb 2019

The following pages contain one page summaries for each recommendation that give more detail on the rationale, supporting information, similar recommendations, specific action items, and notes on implementation. The front of each recommendation summary includes the recommendation statement, proposed owner, and a short "action plan" describing how the recommendation might be implemented. The reverse side of the sheet contains a list of recommendations from DIB concept papers, a list of recommendations from subgroup reports (contained in Appendix B of the supporting information), and some related recommendations from previous reports.

Line of Effort	Streamlining		
Recommendation	**A1 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.**		
Primary Owner	**USD(A&S)**	*Stakeholders*	USD(C), CAPE, SAE, Service FM & PA&E, FASAB
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 create significant delays in fielding user-needed capability.		
Desired State	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.		

Action		Owner	Expected Completion
A1.1	Submit legislative proposal to create a new appropriations category for software and software-intensive programs	USD(A&S), in coordination with USD(C) and CAPE	May 15th, 2019 for FY20 NDAA
A1.2	Make necessary modifications in supporting PPBE systems to allow use and tracking of new software appropriation	USD(C) and CAPE	FY21 Budget FY 22 POM
A1.3	Select pilot programs using DevSecOps to convert to or use new SW Appropriation in FY20	USD(A&S), in coordination with Service Acquisition Executives	August 2, 2019
A1.4	Define budget exhibits for new SW appropriation (replacement for P- and R-forms)	USD(A&S), in coordination with USD(C), CAPE, and Appropriations Committee	August 16, 2019
A1.5	Ensure programs using new software appropriation submit budget exhibits in the approved format.	SAE	FY 22 POM
A1.6	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) there is no “sustainment” phase forsoftware.	FASAB in coordination with USD(A&S) and USD(C)	End FY20

DIB concept paper recommendations:	
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 ideas:	
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 Department and the Fourth Estate.
App	Components will program, budget, and execute for information and technology capabilities from one appropriation throughout lifecycle 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 t “software sustainment” and definition that is consistent with a continuous engineering approach across the lifecycle
M&S	A DoD Working Group should be established to leverage on-going 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 quicker then you're ready (iterative contracts may produce more value with less money
Related previous recommendations:	
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.
809	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.
809	Rec. 42: Reduce budgetary uncertainty, increase funding flexibility, and enhance the ability to effectively execute sustainment plans and address emergent sustainment requirements.
CSIS18	Performance Based Logistics (PBL) contracts should have a duration that allow for tuning and re-baselining with triggered options and rolling extensions.
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.
GAO15	3. 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.

Line of Effort	Streamlining		
Recommendation	**A2c: Establish a new acquisition pathway (Sec 805) for software that prioritizes continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics.**		
Primary Owner	USD(A&S)	*Stakeholders*	USD(C), CAPE, SAE, Service FM & PA&E, Joint Staff
Background	Current law, regulation, and policy, and internal DoD processes make Agile SW development extremely difficult, requiring substantial and consistent senior leadership involvement. Consequently, DoD is challenged in its ability to scale Agile SW development practices to meet mission needs.		
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.		

Action		Owner	Target Date
A2c.1	Submit legislative proposal to create new acquisition pathways for two or more classes of software (e.g, application, embedded), optimized for DevSecOps, including rapid delivery of initial capability to the field; interaction with users to identify features to be implemented; and automated data collection, testing, monitoring, and reporting (see Appendix L.3).	USD(A&S), in coordination with USD(C) and CAPE	May 15, 2019
A2c.3	Create new acquisition pathway	Authorization Committees	FY20 NDAA
A2c.3	Develop and issue a Directive Type Memorandum (DTM) for the new SW acquisition pathway	USD(A&S)	October 1, 2019
A2c.4	Issue Service level guidance for new acquisition pathway	SAE	November 15, 2019
A2c.5	Select pilot programs using DevSecOps to convert to or utilize new SW acquisition pathway	USD(A&S), in coordination with Service Acquisition Executives	November 15, 2019
A2c.6	Convert DTM to DoD Instruction, incorporating lessons learned during pilot program implementation.	USD(A&S)	October 1, 2020
A2c.7	Develop and implement training at Defense Acquisition University on new SW Acquisition Pathway for all acquisition communities (FM, Costing, PM, IT, SE, etc.)	USD(A&S) by	December 13, 2019
A2c.8	Develop, deploy, and require the use of IA-accredited (commercial) development tools	PEOs Digital	TBD

SWAP working group ideas:	
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 previous recommendations:	
GAO'17	Prioritize investments so that projects can be fully funded and it is clear where projects stand in relation to the overall portfolio.

Line of Effort	Streamlining		
Recommendation	**A3 Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially-available products with minimal (source code) modification.**		
Primary Owner	CMO	*Stakeholders*	USD(A&S), Service CMO & SAE
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 insist 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 and there is a funded internal capability to maintain and update the software at a near-commercial cadence.		

Action		Owner	Target Date
A.3.1	Revise DBS certification process guidance	CMO, in coordination with USD(A&S) and Service counterparts	6 months (implement FY20)
A.3.2	Select 4 projects for COTS implementation	CMO, in coordination with Service CMOs and business process owners	6 months
A.3.3	Implement COTS opportunities, with contracts in place	Services, with CMO oversight	18 months
A.3.4	Submit legislative change proposal (if Title 10 §2222 is a hindrance)	CMO in coordination with USD(A&S) and Service counterparts	FY21 Budget

DIB concept paper recommendations	
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 previous recommendations:	
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.
809	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
809	Rec 18: Fund DBSs [defense business systems] in a way that allows for commonly accepted software development approaches

Line of Effort	Streamlining		
Recommendation	**A5 Replace JCIDS, PPB&E, and DFARS with a "PEO Digital" in each Service that uses portfolio management, continuous tracking of software development metrics, and direct identification of warfighter needs to decide on allocation priorities, with no distinction between RDT&E, procurement, and O&M funds.**		
Primary Owner	USD(A&S)	*Stakeholders*	CAPE, 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			
Action		Owner	Target Date
A5.1	Issue guidance for PEO Digital	USD(A&S) SAE	End FY19
A5.2	Select pilot capability areas in each service to place under portfolio management by PEO Digital	SAEs	End CY19
A.5.3	Stand up PEO Digital with necessary resources allocated and aligned	SAE	1QFY21
A.5.4	Implement new portfolio management methods for pilot program capability areas	PEO Digital	3QFY21
A.5.5	Determine intermediate successes of. or required modifications to. portfolio management approach	PEO Digital	2QFY22
A5.6	Establish portfolio management approach as standard work for software	PEO Digital, SAE	FY23

SWAP working group ideas:	
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 vice 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 previous recommendations:	
OSD'06	Transform the Planning, Programming, and Budgeting and Execution process and stabilize funding for major weapons systems development programs.
809	Rec 36: Transition from a program-centric execution model to a portfolio execution model
809	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
809	Rec. 38: Implement best practices for portfolio management.
809	Rec. 39: Leverage a portfolio structure for requirements.

Line of Effort	Streamlining		
Recommendation	**A6 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 functionality.**		
Primary Owner	CMO	*Stakeholders*	USD(A&S), Service CMO & SAE
Background	Current cost estimation and reporting processes and procedures in DoD have proven to be highly inaccurate and time consuming. New metrics are required that match DevSecOps approach 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 really more heavily on metrics that can be collect in a (semi-)automated fashion from instrumentation on the DevSecOps pipeline and other parts of the infrastructure.		

Action		Owner	Target date
A6.1	Identify and hire a small team (3-4) programmers to implement required software and provide them with a modern development environment	CAPE, DDS	3 months after start
A6.2	Identify low-level metrics that are already part of standard commercial development environments (see Appendix A.2 (DIB Metrics) and Appendix D for initial lists)	CAPE. SAO	MVP in 3 mos; then continuous update
A6.2a	Speed and cycle time: launch → initial use, cycle time	Dev team, users	
A6.2b	Code quality: unit test coverage, bug burn-rate, bugs-in-test:bugs-in-field	Dev team, users	
A6.2c	Security: patch → field, OS upgrade → field, HW/OS age	Dev team, users	
A6.2d	Functionality: user satisfaction, number/type of features/cycle	Dev team, users	
A6.2e	Cost: head count, software license cost, compute costs	Dev team, users	
A6.3	Identify 3-5 ongoing programs that are collecting relevant metrics and are willing to partner with CAPE	CAPE, A&S, SAEs	In parallel with A6.2
A6.4	Create a mechanism to transfer and process low-level metrics to PMO on a continuous basis with selectable levels of resolution across the program	CAPE, SAO, PMO	MVP in 3 mo, then continuous update
A6.5	Begin reporting metrics to Congress as part of annual reporting; iterate on content, level, format	CAPE, Comp	FY2020
A6.6	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, SAO, CAPE	FY2020
A6.7	Establish ongoing capability within CAPE to update metrics on continuous basis, with input from users (of the data)	CAPE	FY2021
A6.8	Identify and eliminate remaining uses of ESLOC as metric for	CAPE, SAEs	FY2022

	software/software-intensive programs		

SWAP working group ideas:	
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 previous recommendations:	
DSB87	Rec 12: Use evolutionary acquisition, including simulation and prototyping, as discussed elsewhere in this report, to reduce risk.
SEI'01	Effort Estimation: • Utilize most likely effort estimates in proposals and statusreports. • Find ways to promote the use of accurate effort estimation and productivityevaluation • Lowest cost is not equivalent to best value. Questionoutliers.
OSD'06	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.
SEI'10	Don’t require PMO to adopt contractors’ estimate for the program—or else use the difference as PM “reserve”
SEI'10	Change from traditional 50% estimation confidence level to 80% level
SEI'10	DoD should consider use of Vickrey “second price” auction mechanism for acquisition proposal bidding
SEI'15	6. Cost Estimation. Use the government’s cost estimates (using say 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.

Line of Effort	Digital Infrastructure		
Recommendation	**B1 Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field and make available to contractors at subsidized cost.**		
Primary Owner	USD(A&S)	*Stakeholders*	SAE
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.		
Desired State	Programs will have access to a modern digital infrastructure, which can benefit from centralized support and provisioning to lower overall costs and the burden for each program. This approach will diminish barriers to capability delivery; however, if DoD programs or organizations want or need to go outside of that existing infrastructure, they can still do so (but the process will be harder).		

Action		Owner	Target Date
B1.1	Designate organization responsible for creating and maintaining the digital infrastructure for each Service's digital infrastructure	SAE	June 27, 2019
B1.2	Designate organization responsible for creating and maintain digital infrastructure for DOD's agencies and organizations	USD(A&S)	May 6, 2019
B1.3	Define baseline digital infrastructure systems and implement procurement and deployment processes and capability	Responsible organizations from B1.1 and B1.2	Q2FY20
B1.4	Provide resources for digital infrastructure	USD(A&S), SAE	FY20 Budget FY21 POM
B1.5	Implement digital infrastructure (operational)	Responsible organization from B1.1 and B1.2	May 6, 2020
B1.6	Identify acquisition programs to transition to digital infrastructure	SAE	February 12, 2020
B1.7	Transition programs to digital infrastructure	SAE, PEO, PM	End 4QFY20

DIB concept paper recommendations	
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 ideas:	
T&E	Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio.
Related previous recommendations:	
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.
SEI'01	Infrastructure: In distributed development activities, get high quality, secure, broadband communications between sites. It is an enabler, not a cost.

Line of Effort	Digital Infrastructure		
Recommendation	**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.**		
Primary Owner	**DoD CIO**	*Stakeholders*	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, tools		

Action		Owner	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	3 months
B5.2	In conjunction with Rec B4, allow transfer of ATOs for commercial platforms between programs and services	DoD CIO	3 months
B5.3	Create specifications and certification process for approval of standard development tools (w/ ATO reciprocity)	DoD CIO	6 months
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)	12 months

SWAP working group ideas:	
Acquisition	Include an approach for enterprise-level DevSecOps and other centralized infrastructure development and management, approach for shared services, and applications management.
Infrastructure	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.
Infrastructure	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 previous recommendations:	
Section 809	Rec. 43: Revise acquisition regulations to enable more flexible and effective procurement of consumption-based solutions.

Line of Effort	People		
Recommendation	**C1 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.**		
Primary Owner	USD(A&S)	*Stakeholders*	USD(P&R), SAE, Service HR
Background	The DoD workforce's capacity to apply modern technology and software practices to meet the mission is the way we remain relevant in increasingly technical fighting domains, especially against sophisticated peer adversaries. While DoD has both military and civilian SW people, the career fields suffers from a lack of proponency. The Department has not prioritized a comprehensive recruiting strategy or campaign for technical positions. And, there is no comprehensive training or development program that prepares the software acquisition and technical workforce to adequately deploy modern development tools and methodologies.		
Desired State	The Department's workforce embraces commercial best practices for the rapid recruitment of talented professionals. Once on boarded quickly, they will use modern tools and continuously learn in state-of-the-art training environments, bringing in the best from industry and academia, while pursuing private-public exchange programs to broaden their skill sets.		

Action		Owner	Target Date
C1.1	Exercise existing acquisition and cybersecurity hiring authorities to increase the number of SW people in DoD programs with vacant positions.	SEA, PEO	Immediately
C1.2	Obtain additional manpower authorizations for military and civilian SW developers.	USD(A&S), USD(P&R), SAE	FY21 POM
C1.3	Stand up software factories for each service	SEA, PEO Digital	Pilot in FY20 Full rollout in FY21
C1.4	Create new military occupational specialty (MOS) and core occupational series plus corresponding career tracks for each service	J1 and comparable X1 for each Service with USD(P&R)	12/13/19
C1.5	Create new regulations to allow standard identification, recruitment and on-boarding of experienced civilian software talent, especially on rotation from private sector roles, to occur in 60 days	USD(P&R)	12/31/19

DIB concept paper recommendations	
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”) [dup]
SWAP working group ideas:	
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	The DoD should 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. [dup]
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 obtainment
Wkf	Establish an alliance across the services that incentivizes and provides our software practitioners a modern engagement platform (e.g. a chatOps platform) to connect across services, share their skills, communicate through knowledge channels, gather pain points, 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	Empower Implementation Cadre. New Legislation - This will be critical to avoid a repeat of the past 35+ years of continuous admiration of the problem.
Wkf	Computer Language Proficiency Pay. New Language - Title 10, §1596a - Use this language to create a new Computer-language proficiency pay statute.
Wkf	Develop a Strategic Recruitment Strategy for Civilians. New Legislation
Wkf	Pilot a Cyber Hiring Team. New Legislation - Team will have all the necessary authorities to execute recommendations called out in this report. The team will 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 previous recommendations:	
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.
SEI'10	Improve compensation and advancement opportunities to increase tenure.

Line of Effort	People		
Recommendation	**C4 Restructure the approach to recruiting software developers to assume that the average tenure of a talented engineering will be 2-4 years, and make better use of HQEs, IPAs, reservists, and enlisted personnel to provide organic software development capability.**		
Primary Owner	**USD(A&S)**	*Stakeholders*	SAE
Background	Current DoD personnel systems assuming 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 creates challenges in retaining personnel that are not likely to be overcome, so a different approach is needed.		
Desired State	DoD takes advantage of 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.		

Action		Owner	Target Date
C4.1	Exercise existing hiring authorities to increase the number of highly skilled SW people in DoD programs	SEA, PEO	Begin now
C4.2	In conjunction with Recs C1 and D3, create a database of individuals in enlisted, officer, reserve, and civilian positions with software development skills and experience for internal recruiting use to SW squadrons & PAOs	CMO?, Service HR groups?	3 months
C4.3	Within organic software programs, create processes for maintaining release cadence under the assumption of up to 25% turnover per year	PMOs	6 months
C4.4	Require software-intensive project proposals to include a plan for maintaining cadence-related metrics in face of up to 25% turnover of staff	SAEs	6 months
C4.5	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)	FY21 NDAA?
C4.6	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 POM

DIB concept paper recommendations	
10C	Establish Computer Science as a DoD core competency
SWAP working group ideas:	
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 previous recommendations:	
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.
809	Rec. 45: Create a pilot program for contracting directly with information technology consultants through an online talent marketplace.
SEI'10	Divide large acquisition development efforts into multiple smaller, shorter duration programs.

Line of Effort	Culture			
Recommendation	**D.2 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).**			
Primary Owner	USD(A&S)	*Stakeholders*	CAPE, SAE, Service Cost Orgs	
L/R/P Change	☐Law ☐ Regulation ☒Policy DoDI 5000.02, DoDI 5000.75, DoDI 5105.84			
Background	Current program reporting requirements are largely manual, time consuming, and provide limited insight into the SW health of a program.			
Desired State				
Action		Owner	Target Date	
D.2.1	Modify acquisition policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code.	USD(A&S)	May 6, 2019	
D.2.2	Modify cost estimation policy guidance to specify use of automatically generated, continuously available metrics that emphasize speed, cycle time, security, and code.	CAPE	June 26, 2019	
D.2.3	Develop specific measure of software quality, value and velocity and the tools to implement the automatic generation and reporting	TBD	TBD	

SWAP working group ideas:	
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) (Cultural)
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 real-time?] data with reviews occurring on a standard calendar, rather than the current approach of manually developed, periodic reports. [dup]
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, etc.) 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 previous recommendations:	
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.
809	Rec 19: Eliminate the Earned Value Management (EVM) mandate for software programs using Agile methods
MITRE'18	Elevate Security as a Primary Metric in DoD Acquisition and Sustainment

Line of Effort	Culture		
Recommendation	**D6 Shift from a list of requirements for software to a list of desired features and required interfaces/characteristics, to avoid requirements creep, overly ambitious requirements, etc.**		
Primary Owner	**USD(A&S)**	*Stakeholders*	Joint Staff, SAE
Background	Current DoD requirements processes significantly impedes its 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 of the most use to the user for those types of software for which this is relevant.		
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.		

Action		Owner	Target Date
D6.1	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)	3 months
D.6.2	Update CJCSI 3170.01H (JCIDS requirements process) to reflect contents of guidance memos	Joint Staff	6 months
D.6.3	Modify DoDI 5000.02 and DoDI 5000.75 (or integrate into new DoDI 5000.SW)	USD(A&S)	12 months

DIB concept paper recommendations	
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 previous recommendations:	
SEI'12	Ensure requirements prioritization of backlog considers business value and risk.
GAO'17	Match requirements to resources—that is time, money, technology, and people—before undertaking new development efforts.
GAO'17	Research and define requirements before starting programs and limit changes after they are started.
SEI'15	2. Requirements Growth. Programs should manage the volatility and ambitiousness of system requirements
SEI'01	Ensure that all critical functional and interoperability requirements are well specified in the contract (statement of work, Statement of Objectives).
SEI'01	Handle requirements that have architectural consequences as systems engineering issues—up front.