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

Defense Innovation Board, 21 March 2019

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

Extended Abstract

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

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

In this iteration of our report, we emphasize three fundamental themes:

1. **Speed and cycle time are the most important metrics for managing software**. To maintain advantage, DoD needs to procure, deploy, and update software that works for its users at the speed of mission need, executing more quickly than our adversaries. Statutes, regulations and cultural norms that get in the way of deploying software to the field quickly weaken our national security and expose our nation to risk.

2. **Software is made by people and for people, so digital talent matters**. DoD's current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly fast or deep enough to meet its mission needs. New mechanisms are needed

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

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.

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

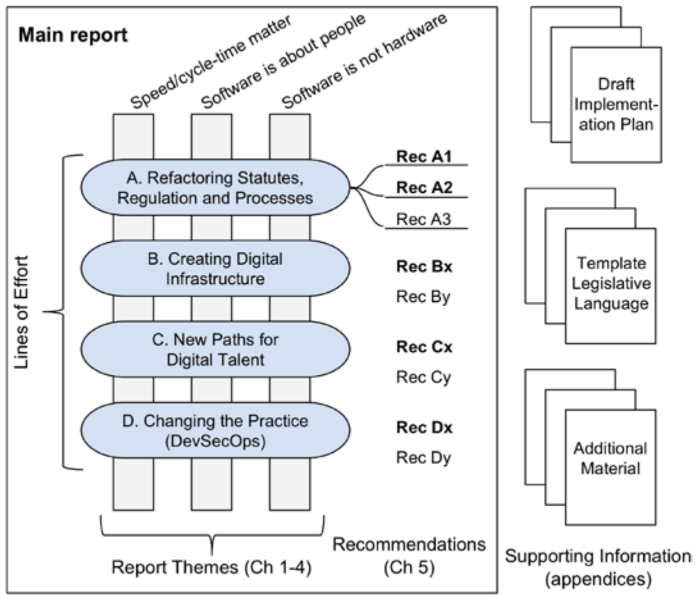
To take advantage of the power of software, we recommend four primary lines of effort:

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

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

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

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


Main report
Speed/cycle-time matter
Software is about people
Software is not hardware
Lines of Effort
A. Refactoring Statutes, Regulation and Processes
B. Creating Digital Infrastructure
C. New Paths for Digital Talent
D. Changing the Practice (DevSecOps)
Rec A1
Rec A2
Rec A3
Rec Bx
Rec By
Rec Cx
Rec Cy
Rec Dx
Rec Dy
Report Themes (Ch 1-4)
Recommendations (Ch 5)
Draft Implement-ation Plan
Template Legislative Language
Additional Material
Supporting Information (appendices)

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

Table of Contents

Supporting Information

Chapter 0. README (Executive Summary)

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

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

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

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

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

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

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

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

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

This report is organized as follows:

- **Extended Abstract:** a two-page summary of the key takeaways from the report.
- **README** (this document): a more detailed executive summary of the report. 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.
- **Chapters 1-4:** short descriptions of key areas and topics. If you attach the extended abstract to any one of these as a preface, it should be comprehensible.
- **Chapter 5:** a more detailed description of the recommendations and our rationale.
- **Supporting Information:** To ensure that the main body of the report satisfies the staple test[6] and the takeoff test,[7] 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. Some highlights:

[4] James Vincent, "Putin says the nation that leads in AI 'will be the ruler of the world'", *The Verge*, Sep 4, 2017: https://www.theverge.com/2017/9/4/16251226/russia-ai-putin-rule-the-world

[5] Section 872 of the FY18 National Defense Authorization Act (NDAA) directed the Secretary of Defense to "direct the Defense Innovation Board to undertake a study on streamlining software development and acquisition regulations." The DIB-SWAP members were charged to "review the acquisitions regulations applicable to, and organizational structures within, the Department of Defense…; review ongoing software development and acquisition programs…; produce specific and detailed recommendations…; and produce such additional recommendations for legislation." See Section 872 of the FY18 NDAA at https://www.congress.gov/115/plaws/publ91/PLAW-115publ91.pdf (Appendix J)

[6] Any report that is going to be read should be thin enough to be stapled with a regular office stapler.

[7] Reports should be short enough to read during takeoff, before the movies start and drinks are served.

- **Draft implementation** (Appendix A)**:** For each recommendation, a summary of the background, desired state, stakeholders, role of Congress, and actions to be taken.
- **Legislative language** (Appendix B)**:** Template language for new or revised statutes, aligned with our recommendations.
- **An Alternative to P-docs and R-docs** (Appendix C): A different mechanism for budget submissions for software programs.
- **FAQ** (frequently asked questions, Appendix D): a list of the most common questions that we get about the study and our attempt to answer them. (Question #1: hasn't all of this been recommended before? A: yes…).

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

Key themes. The rise of electronics, computing, and networking has forever transformed the way we live: software is a part of almost everything with which we interact in our daily lives, either directly through embedded computation in the objects around us or indirectly through the use of information technology through all stages of design, development, deployment, and operations. Our military advantage, coordination with allies and partners, operational security, and many other aspects of 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.

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 transform its culture and personnel policies before it is too late, this software-savvy generation can still set the Department on the right course.

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

1. Speed and cycle time are the most important metrics for software.
2. Software is made by people and for people, so digital talent matters.
3. Software is different than hardware (and not all software is the same).

Speed and cycle time are the most important metrics for software. Most DoD software projects are currently managed using “waterfall” development processes, which involve spending years on developing requirements, taking 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 because it demands focus on the critical functionality rather than over-specification or bloated requirements. It also means we can identify trouble earlier and take faster corrective action which reduces cost, time, and risk. Faster leads to increased reliability: the more quickly software/code is in the hands of users, the more quickly feedback can focus on efforts to deploy greater capability. Faster gives us a tactical advantage on the battlefield by allowing operation and response inside our adversaries' observe–orient–decide–act (OODA) loops. Faster is more secure. Faster is possible.

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

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

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

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

Bringing about the type of change that is required to give DoD the software capabilities it needs is going to take a significant amount of work. While it is possible to use the current acquisition system and DoD process to develop, procure, assure, deploy, and continuously improve DoD software, the statutes, regulations, processes, and culture are debilitating. The current approach to acquisition was defined in a different era, for different purposes, and only works for software projects through enormous effort and creativity. Congress, the Office of the Secretary of Defense, 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 four primary lines of effort that we recommend be undertaken:

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

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

C. **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.

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

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, the need for speed and agility (theme 1), the critical role that smart people have to play in the process (theme 2), and the impact of inefficiencies of the current approach (theme 3). In many ways this mission is as challenging as any combat mission: while 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.

Refactor statutes, regulations, and processes, streamlined for software. Congress has created many workarounds to allow DoD to be agile in its development of new weapons systems, and DoD has used many of these to good effect. But the default statutes, regulations, and processes that are used for software too often rely on the traditional hardware mentality (repeat: software is different than hardware) and those practices do not take advantage of what is possible 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 innovation cycles (see the "DIB Ten Commandments of Software" in Appendix E for more information about the DIB's guiding principles for software acquisition).

Create and maintain cross-program/cross-service digital infrastructure. Current practice in DoD programs is 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 DoD 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) Joint Capabilities Integration and Development System (JCIDS), Planning, Programing, Budget and Execution (PPB&E), and Defense Federal Acquisition Regulation Supplement (DFARS)[9] that are optimized for software. The Director, Operational Test and Evaluation (DOT&E) will need new methods for operational test and evaluation that match the

[9] Common DoD acronyms are defined in Appendix I (Acronyms and Glossary).

software's speed of relevance, and Cost Assessment and Program Evaluation (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.

Create new paths for digital talent (especially internal talent). The biggest enabler for great software is providing great people with the means to contribute to the national security mission. While the previous recommendations speak to providing the tools and infrastructure DoD technologists need to succeed, it is equally important that the Department's human capital strategies allow them to even do this work consistently in the first place. Driving the cultural transformation to support modern, cloud-based technology requires new types of skills and competencies, changing ratios of program managers to software engineers, moving from waterfall development to 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 the government manages its programs and contractors, which goes beyond just moving to agile development. The government must have experts well steeped in the software development process and architecture design to adequately manage both organic activities and contracted programs. They must have the skills to detect when contractors are going down the wrong path, choosing a bad implementation approach, or otherwise being wasteful. This is perhaps the argument for ensuring we have software development experience natively in the government, rather than relying primarily on external vendors; unless there are software-knowledgeable members on the core team, it is impossible to effectively monitor and manage outsourced projects. This is 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, 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. 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.

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

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

Getting started now. The types of changes 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 therefore the approach to implementing change for software should be different as well.

Indeed, most (if not all) of the changes we are recommending are not new and not impossible to do. The 1987 Defense Science Board Task Force on Military Software,[10] chaired by legendary computer scientist Fred Brooks, wrote an outstanding report that already articulated much of what we are saying here. And the software industry has already implemented and demonstrated the utility of the types of changes we envision. The problem appears to be in getting the military enterprise to adopt a software mindset and implement a DevSecOps approach in a system that was intended to make sure that things would not move too quickly.

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*:

- Within 60 days after delivery of this report to Congress: Define a detailed implementation plan and assign owners for each of the top recommendations to begin right now.
- FY19 (create): High-level endorsement of the vision we articulate here, and support for activities that are consistent with the desired end state (i.e., DevSecOps and enterprise-level architecture and infrastructure). Identify and launch programs to move out on the priority recommendations (start small, iterate quickly). If you are reading this and are in a position of leadership in your organization, pass this on to others with your seal of approval and a

[10] 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.

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 be on the front line of changing DoD to a “culture of yes.”

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

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

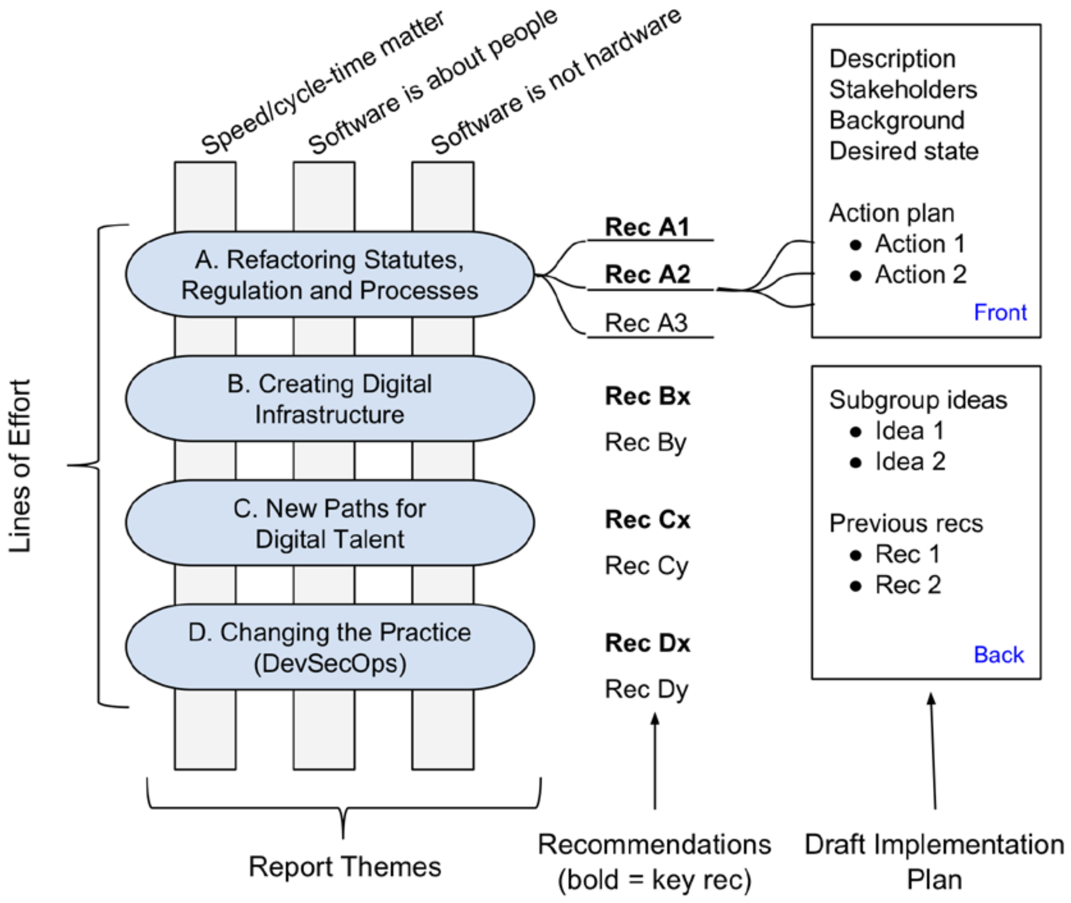
A two page summary (“cheat sheet”) of the lines of effort and recommendations are given next.

DIB SWAP Study
Recommendations "Cheat Sheet"

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

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

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



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

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

Line of Effort A (Congress and OSD): Refactor statutes, regulations, and processes for software	
A1	Establish new acquisition pathway(s) for software that prioritizes continuous integration and delivery of working software in a secure manner, with continuous oversight from automated analytics
A2	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
Line of Effort B (OSD and Services): Create and maintain cross-program/cross-service digital infrastructure	
B1	Establish and maintain digital infrastructure within each Service or Agency that enables rapid deployment of secure software to the field and incentivize its use by contractors
B2	Create, implement, support, and use fully automatable approaches to testing and evaluation (T&E), including security, that allow high-confidence distribution of software to the field on an iterative basis
B3	Create a mechanism for Authority to Operate (ATO) reciprocity within and between programs, Services, and other DoD agencies to enable sharing of software platforms, components and infrastructure and rapid integration of capabilities across (hardware) platforms, (weapons) systems, and Services
Line of Effort C (Services): Create new paths for digital talent (especially *internal* talent)	
C1	Create software development units in each Service consisting of military and civilian personnel who develop and deploy software to the field using DevSecOps practices
C2	Expand the use of (specialized) training programs for CIOs, SAEs, PEOs, and PMs that provide (hands-on) insight into modern software development (e.g., agile, DevOps, DevSecOps) and the authorities available to enable rapid acquisition of software
Line of Effort D (Acquisition Offices and Contractors): Change the practice of how software is procured and developed	
D1	Require access to source code, software frameworks, and development toolchains – with appropriate IP rights – for DoD-specific code, enabling full security testing and rebuilding of binaries from source
D2	Make security a first-order consideration for all software-intensive systems, recognizing that security-at-the-perimeter is not enough
D3	Shift from the use of rigid lists of requirements for software programs to a list of desired features and required interfaces/characteristics, to avoid requirements creep, overly ambitious requirements, and program delays

Additional context provided in Chapter 5 and draft implementation plans in Appendix A.