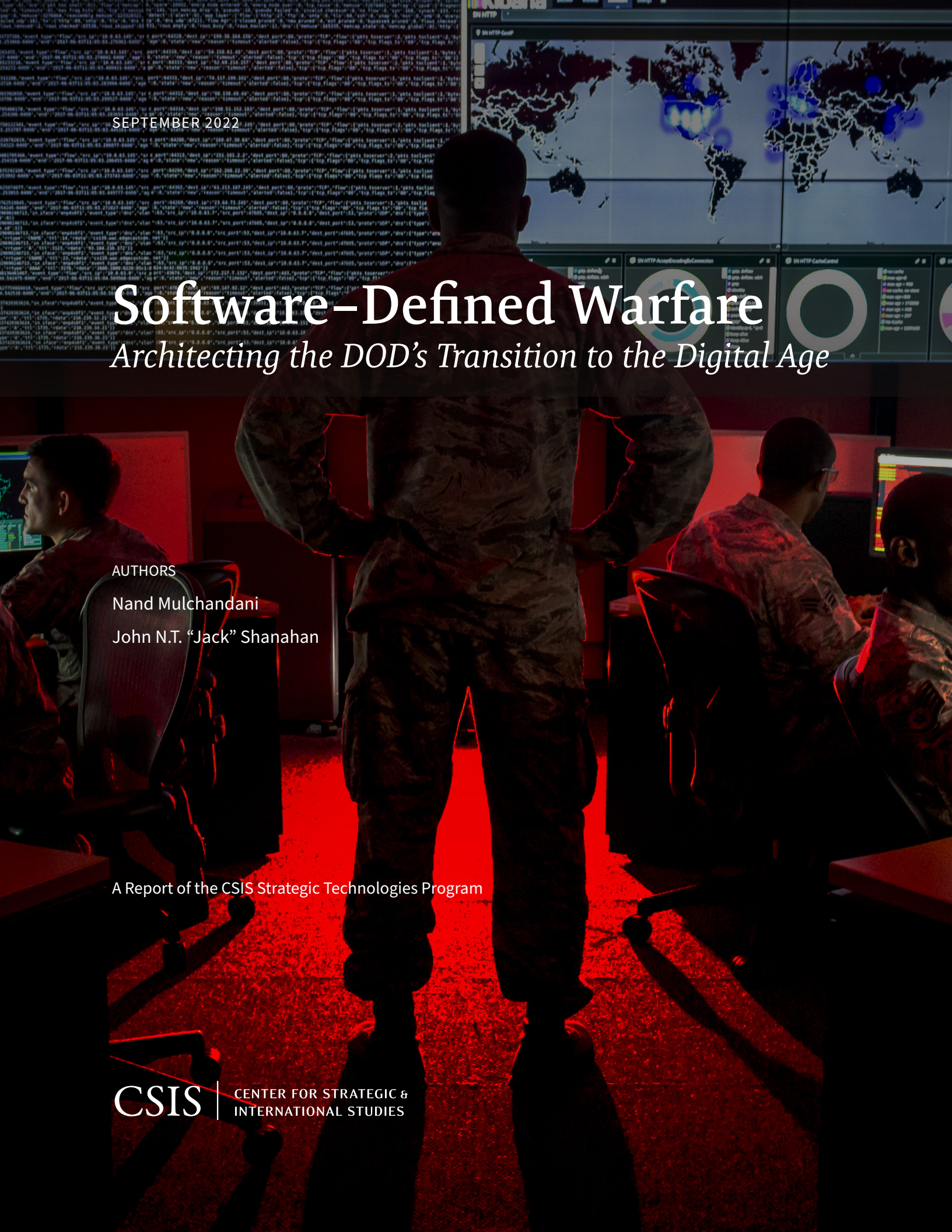


SEPTEMBER 2022

Software-Defined Warfare

Architecting the DOD's Transition to the Digital Age

AUTHORS

Nand Mulchandani

John N.T. "Jack" Shanahan

A Report of the CSIS Strategic Technologies Program

CSIS | CENTER FOR STRATEGIC & INTERNATIONAL STUDIES

SEPTEMBER 2022

Software-Defined Warfare

Architecting the DOD's Transition to the Digital Age

AUTHORS

Nand Mulchandani

John N.T. "Jack" Shanahan

A Report of the CSIS Strategic Technologies Program

CSIS | CENTER FOR STRATEGIC & INTERNATIONAL STUDIES

About CSIS

The Center for Strategic and International Studies (CSIS) is a bipartisan, nonprofit policy research organization dedicated to advancing practical ideas to address the world's greatest challenges.

Thomas J. Pritzker was named chairman of the CSIS Board of Trustees in 2015, succeeding former U.S. senator Sam Nunn (D-GA). Founded in 1962, CSIS is led by John J. Hamre, who has served as president and chief executive officer since 2000.

CSIS's purpose is to define the future of national security. We are guided by a distinct set of values—nonpartisanship, independent thought, innovative thinking, cross-disciplinary scholarship, integrity and professionalism, and talent development. CSIS's values work in concert toward the goal of making real-world impact.

CSIS scholars bring their policy expertise, judgment, and robust networks to their research, analysis, and recommendations. We organize conferences, publish, lecture, and make media appearances that aim to increase the knowledge, awareness, and salience of policy issues with relevant stakeholders and the interested public.

CSIS has impact when our research helps to inform the decisionmaking of key policymakers and the thinking of key influencers. We work toward a vision of a safer and more prosperous world.

CSIS does not take specific policy positions; accordingly, all views expressed herein should be understood to be solely those of the author(s).

Acknowledgments

This report is made possible by general support to CSIS. No direct sponsorship contributed to this report.

Center for Strategic & International Studies
1616 Rhode Island Avenue, NW
Washington, DC 20036
202-887-0200 | www.csis.org

Contents

Introduction

On August 20, 2011, Marc Andreessen published "Why Software Is Eating the World."[1] In the decade since this seminal article was written—an eternity in the technology industry—an entire new generation of "digital native" companies has emerged that forced slower-moving incumbents out of business. Technology has been devouring the world, and it also eats its own.

The U.S. military is considered the best in the world. With a budget larger than the next 10 militaries combined, the Department of Defense (DOD) outspends, out-equips, and out-trains its competitors. It is also an industrial-age, hardware-centric organization that has the "biggest, the mostest, and the bestest" military capital investments: tanks, ships, aircraft, and everything in between. Unfortunately, in today's world, hardware is "old-school"—low-margin, commodity products that are manufactured, stored, shipped, and consumed. After depreciation and excessive wear and tear, hardware is scrapped for parts or ends up in a boneyard.

The military's hardware systems are built to wrap multiple layers of metal and protection around fragile human beings who operate these machines under fire and extreme levels of stress, with the planning and orders coming from groups of people watching screens and integrating information through PowerPoint and Excel. The DOD accomplishes incredible things, but as remarkable as the Department is in many ways, it shares with almost all other federal agencies the common trait of lagging woefully behind the commercial software industry's state of the art—in everything from automating back-office functions to providing digital warfighting services to its customers.

Under normal circumstances, an organization like the DOD would be a huge target for "disruption" by entrepreneurs and investors (a euphemism for "putting out of business"). Fortunately, it has a monopoly on the nation's warfighting functions, which insulates it from the usual forms of

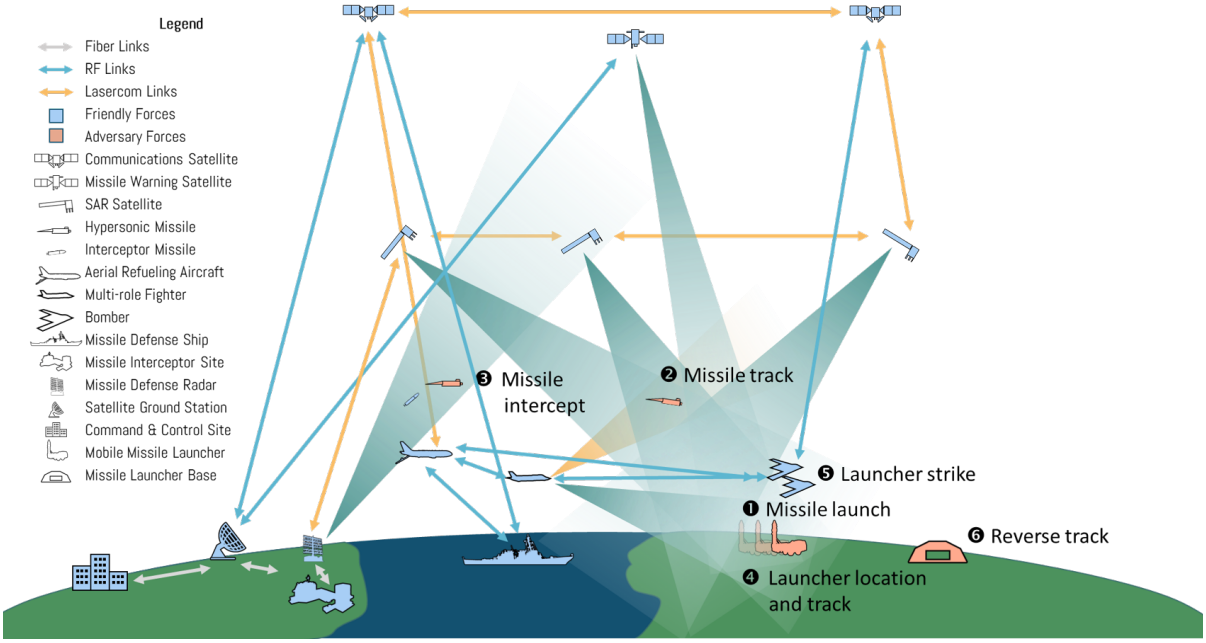
Source: Photo Media/ClassicStock/Getty Images

market competition. Yet this very monopoly is also the root cause of many of the worst problems when it comes to the DOD's failure to adopt new technologies, change its legacy workflows and processes, and design and experiment with new operating concepts. Within the U.S. government, the DOD does not experience the kind of brutal, capitalist, Darwinian journey by which incumbent organizations face off against hungry new start-ups and risk getting pushed to the side—where the start-up takes over as the "new DOD" with brand-new underlying tech. Equally problematic, the massive DOD bureaucracy struggles with the kind of periodic "tech refresh" that has been instrumental to commercial industry success. While it is insulated from market competition within the U.S. economy, the DOD is not immune from the kind of revolutionary, secular, and wide-ranging technological changes happening outside the government. Nor is it immune from the threat of competition with other militaries around the world. Either the DOD will change itself, or its competitors will force it to change—after it might be too late.

For the United States to retain its dominant position in the future—which is not a guaranteed outcome—the DOD needs a new design and architecture that will allow it to be far more flexible, scale on demand, and adapt dynamically to changing conditions. And it must do so at a dramatically lower cost as it delivers its critical services. Central to this will be changing the role of the human in the information and decision loop as well as at the endpoint: increasing speed, accuracy, scale, and techniques to confuse, disorient, and overwhelm the adversary while preventing them from doing the same to U.S. forces. The DOD's systems will need to support dramatically faster decisionmaking and execution speeds; allow for rapidly updating and modifying systems; lower the cost structure of building and deploying these systems; and upend the marginal cost and speed of delivering new functionality.

The idea that the DOD must dramatically change the way it builds warfighting systems is hardly novel. There have been hundreds of articles, think-tank papers, and books that have made a compelling case for radical change. One of the best recent examples is *The Kill Chain* by Chris Brose, who calls upon the DOD to build a "military Internet of Things" that involves "large networks of autonomous systems" and a different mix of weapons systems.[2] Likewise, in crafting the concept of the Third Offset Strategy, former deputy secretary of defense Bob Work has argued for years about the need for major DOD-wide changes, especially the widespread adoption of autonomous systems and AI-enabled capabilities.[3] Other ideas that percolate regularly in defense policy circles revolve around concepts such as networked warfare, mosaic warfare, and next-generation battle networks. Much of this work describes the problem but does not get to the level of proposing concrete technology solutions. In solving extremely hard technology problems, leading commercial technology companies typically start by outlining a set of key architectural standards upon which the entire system will be built. This paper outlines the architecture needed for next-generation warfighting systems.

The key to this architecture lies in software. Software is disrupting and reconstituting entire industries—especially those that rely on manual processes and fail to leverage the power of low or zero marginal costs. This is not about sprinkling "magic software dust" on a company or business model to make it work. Software needs to be at the core of every business and operating model before any business can hope to gain an enduring competitive advantage. The DOD is no different.



Source: Todd Harrison, "Battle Networks and the Future Force: Part 2: Operational Challenges and Acquisition Opportunities," CSIS, *CSIS Briefs,* November 3, 2021, https://csis-website-prod.s3.amazonaws.com/s3fs-public/publication/211103_Harrison_Battle_Networks_Part2_0.pdf?vsuBpGNyDDOwNE_hMzckmGEfb8fq13dx.

1

Closing the Kill Chain, Getting inside a Competitor's OODA Loop

The "kill chain," a phrase that long predates Brose's book, refers to a multistep process that involves absorbing information, turning that information into knowledge (actionable intelligence), making a decision, acting on the decision, understanding the consequences of that decision, and refining future actions accordingly. Similarly, John Boyd's "OODA loop" concept refers to the process by which an individual, commander, or team progresses through the same stages: observe, orient, decide, and act.[4] The final part of the process—acting or closing the kill chain—is generally associated with firing and destroying something, but in today's world it could also include a non-kinetic effect such as delivering a cyberattack against a target or deploying a piece of misinformation designed to disrupt the adversary's orientation phase of the OODA loop while protecting one's own. In general, the side that can consistently demonstrate shorter, tighter, and more resilient OODA loops will gain an advantage.

Ultimately, the kill chain and OODA loop are essentially workflows buttressed by human cognition. Today, these workflows are implemented as a confusing mess of manual, semi-manual (including telephone calls), and electronic processes. It is both exciting and shocking to visit a combatant command operations center. In peacetime, it is easy to see how these commands and their subordinate units can handle a relatively limited number of aircraft and ships moving around. What is hard to imagine is how extant manual processes will be capable of tracking hundreds, if not thousands, of moving objects in wartime, all tasked with delivering kinetic and non-kinetic effects from undersea through space, cyberspace, and everything else in between. Executing these workflows so every step of the process and the data flows are done in software, even though decisionmaking is still in the hands of human commanders and operators, can yield a decisive advantage over an adversary: every stage of the OODA loop is tighter, faster, and more well-

Source: U.S. Marine Corps photo, licensed under CC BY 2.0.

informed. Conversely, not taking a digital-age, software-centric approach in the battlespaces of the future could very well mean tactical, operational, and even strategic defeat.

Readers can easily misconstrue such "software cheerleading" as implying the demise of hardware. That misses the point. Military hardware and weapons systems are still vitally important. However, they need to be tethered to the end of internet-scale software systems that handle all the complexity of decisionmaking, targeting, and resourcing—everything but the final step of executing a decision to close the kill chain. Software will fundamentally and irretrievably change the value of hardware by making the processes that lead to the desired effects faster, cheaper, more efficient, scalable, and more accurate.

2

Lessons from Data Center Architecture

Transferring lessons and experience from other domains can help inform breakthrough solutions to big problems in the DOD and other federal government organizations. Years of building large-scale software systems have revealed best principles and practices on how to design computer software and systems for squeezing out the maximum performance at the lowest cost and, where appropriate, removing the most expensive, least scalable, and slowest parts of the process and cost structure: humans, followed by hardware. These same concepts can—and should—be applied to the DOD.

In the computer industry, "software defined" is a broad architectural concept that drives a few different core design decisions. These design decisions turn a bunch of disconnected hardware products into an integrated whole that can be operated and managed as a single platform. It also takes control and complexity, which is typically distributed all over the place, and centralizes it where it can be simplified, managed, and scaled. In military terms, this is centralized direction and decentralized execution: the best of both worlds.

Most of the large-scale computing systems and platforms used in the technology industry are software defined. The largest cloud-computing platforms, such as Amazon Web Services (AWS) and Microsoft Azure, are all "software-defined computing platforms" relying on "software-defined networking" (SDN); their storage is now "software defined," as are fifth-generation wireless (5G) telecom systems, with provisioning and routing all done via software. These large-scale platforms run on commodity hardware (in some cases, built in house) with large-scale orchestration software that pulls the complexity out of the lower levels of the platform to a central system.

It is reasonable to ask whether today's computer systems can handle the scale and size of a battlefield. The answer is a resounding "yes!" Internet-scale companies such as Apple, Meta, Amazon, Alphabet,

Microsoft, Netflix, and others routinely connect billions of users and systems every day, exchanging content and video in real time. While these companies do not typically face the risk of actual kinetic effects destroying their data centers, they do deal with cybercriminals and nation-state–sponsored cyberattacks—and even squirrels or rats eating through cable wires. The best of these architectures are built to be resilient in the event of failure. If these companies can operate at this enormous scale, there is no reason whatsoever why the same key concepts cannot be applied to warfighting to weave together thousands of hardware objects into a single software-defined system.

Concept: Architect for Scaling

Source: Photo Media/ClassicStock/Getty Images

Earlier computer systems and software were designed to allow a single application to run on a single computer system. When a computer system ran out of capacity, the solution was always to throw more resources at the problem: a faster processor, more disk space, a faster network card, and so on. This method is called "vertical scaling." Once a server was running at capacity, you were done, as there are physical limits to the resources you can add to a system. Over time, this evolution led the industry to one of the biggest systems possible: the mainframe.

One of this paper's authors had his first job out of college at a hot, upstart company called Sun Microsystems that was revolutionizing network computing. Sun computers—called "pizza boxes" because they looked like one—were competing against the mighty mainframe. Instead of the "unit of scaling" for the mainframe, which was another mainframe, to scale up you simply added a single "pizza box" worth of additional computing and memory every time you needed some marginal resources for an application. This marginal-expense option is far more attractive than being forced to buy another giant, multi-million-dollar mainframe that might only be 10 percent utilized. This method of small, marginal additions is called "horizontal scaling" and was created to overcome the limits of vertical scaling.

Instead of a single, large computer system running an application, horizontal scaling involves distributing an application's computing workloads across multiple, smaller computer systems. Unfortunately, moving from a vertically to horizontally scaled architecture does not come free. The software itself needs to be re-architected to support this new underlying system topology. Issues such as database storage (which is hard to distribute across multiple systems), file systems, caches and data

consistency, software queues, and load balancers come into play. As companies moved their legacy applications from their on-premises data centers to the cloud, they realized they had to re-factor or even rewrite their applications to take full advantage of what cloud computing could provide.

Computer systems in older data centers are called "pets" because each computer system would have a unique name, typically part of a broader naming scheme. Popular naming schemes were the planets in the solar system or cats (e.g., "Fluffy"). Just as if a pet got sick and had to be taken to the vet, if one of these computers had a problem, engineers would try their best to fix it. In contrast, today's computers running at a major cloud vendor have a virtually meaningless string of characters as a name (e.g., "i-0123456789abcdef")—unfortunately, just like anonymous cattle. Cloud-computing instances are created and destroyed all the time, with little sentimental value.

Applying this to the United States' warfighting systems architecture is simple: The DOD needs a massive number of cheap, disposable, and easy-to-manufacture endpoint systems that it can concentrate, distribute, and scale up or down as the need arises. In many cases, these systems will require little or no maintenance. If one breaks, you just replace it with an identical clone, driving down overall complexity in logistics and maintenance. Parts of the DOD are already thinking about this concept (with smaller drones, for example), but the Department needs to start applying this architectural thinking to many more of the aspects of hardware, vehicles, and weapons systems that it builds or buys.

Concept: Eliminate Single Points of Failure

Source: Photo Media/ClassicStock/Getty Images

The current design points for the U.S. military are hardware centric, human centric, and big. Size matters. The mighty aircraft carrier exemplifies this paradigm. It costs billions of dollars to build, carries thousands of sailors, and hosts dozens of different kinds of fighter aircraft. It sits in the middle of a carrier strike group (CSG), with many ships and submarines tasked to protect it. The protection offered by the CSG makes it extremely hard to sink a U.S. aircraft carrier. However, taking out an aircraft carrier destroys the entire CSG architecture, making it nearly useless for the purpose for which it was designed: to project force around the world.

An aircraft carrier, given its size and footprint, is like a very large mainframe or a large database system. An aircraft carrier is vertically scaled, as the unit of scaling for the system is yet another aircraft carrier. Many of these vertically scaled systems are also single points of failure. Since these

systems are very expensive, there are normally not any redundancies from running two systems at the same time. The crash of a mainframe or database system crashes the entire system, as there is no backup. Obviously, architectures that have such single points of failure are extremely brittle and eventually fail.

Identifying and eliminating these single points of failure are critical to building a resilient architecture, both in commercial industry and in the DOD. The aircraft carrier and the concept of the CSG were created at a time before China developed carrier-killer missiles. At $10 million a missile, shooting a few of these and hitting a U.S. carrier worth upward of $10 billion would generate an eye-watering return on investment (ROI). Or, as some of the fighting in Ukraine seems to presage, you can get an even more impressive ROI by attacking a CSG with hundreds (if not thousands) of small, attritable, armed, autonomous swarming drones, all of which can be horizontally scaled on demand to attack the most vulnerable parts of the enemy's position.

Concept: Virtualization

Virtualization technology has revolutionized data centers by transforming a computer server into a virtual machine that can be copied, cloned, snapshotted, and moved around. It has helped create the revolution in cloud computing and "software as a service" by allowing users to rent computers by the minute and operate them remotely at an incredibly competitive cost.

In its generic form, virtualization is the act of creating an abstraction and translation layer between two distinct systems, such as hardware and software, or taking an older system and allowing it to be manipulated and managed as part of larger digital workflows. Initially, virtualization's most popular trait was the ability to take older/legacy Windows operating systems (such as Windows 2000) and continue to run them on newer hardware, as well as the ability to consolidate the systems into a central data center.

In the case of legacy military platforms, sensors, and workflows, virtualizing the system involves wrapping it in a set of callable Application Programming Interfaces (APIs) that allow it to participate in any electronic workflows without the interruption of a manual process. An API is a standardized software layer with a set of well-defined and well-documented software entry points that a programmer can call from their code to receive data and initiate processes. Once kill chains have been converted to end-to-end electronic workflows, the effect of a weapons system at the end of such a workflow can be implemented as an API call. Ideally, these APIs should be two-way: inbound ones that allow for the control

Source: U.S. Navy photo by Mass Communication Specialist 2nd Class Michael H. Lehman/Released

and execution of the system and outbound ones that provide status, capacity, usage statistics, and the like that can be fed back into the orchestration system for planning, resupply, and resource management. Even if a system can only enable some APIs—for instance, outbound APIs that deliver status and supply-level information—that is still a substantial improvement, as its calculations can be networked into the broader orchestration systems.

During previous DOD Joint All-Domain Command and Control (JADC2) demonstrations, the military services demonstrated API-based access to weapons systems. This is a crucial first step. In effect, it means that those systems could be integrated as part of electronic kill chains and brought under the management of the overall software orchestration engine. This functionality, however, needs to be moved into production and to the field instead of just used once in demonstrations or experiments. The importance of changing the DOD's mindset from relying on individual hardware systems and one-off software programs to using a system centered on the role of APIs cannot be overstated.

The second dimension of virtualization is to set up the endpoint hardware to run multiple software stacks simultaneously, all completely isolated from each other logically but sharing the underlying hardware platform. There are numerous cases where fixed, one-time hardware costs dominate. The ability to dynamically reprogram or switch to a completely different workload stack—for example, based on the time of day—can drive up the utilization of the hardware. Small examples are cell phones, with a user potentially wanting to use multiple phone numbers (tied to SIM cards) on a single device. A more complex and expensive example is a satellite, for which the initial costs of building and launching the system are often extremely high, while the variable costs of operating it are fairly predictable. Squeezing out maximum utility from such an expensive system can make the difference between a profitable investment and one that is losing money. In addition, thinking about a constellation of satellites as a "computing grid" able to deploy multiple/dynamic workloads can fundamentally transform the traditional paradigm of satellites as having a fixed function. Instead, they can be used in the same way as any flexible cloud-computing system, giving actual "computing in the cloud."

Concept: Low-Cost Commodity Hardware and Stateless Endpoints

In the original Top Gun movie, it is hard to miss "MAVERICK" stenciled on Pete Mitchell's plane—which is very cool until you start wondering what happens if it breaks down. One can just imagine this line from Ed Harris (playing the admiral in the latest film): "Maverick, I've got some bad news. Your airplane is going to be down for maintenance. The good news is that you can take Iceman's jet if he agrees to give it up!" That might be great for Maverick, but he would have to face off against Iceman, who would rather die than give up the chance to fly his own jet. On top of that, Iceman won't appreciate Maverick messing with his seat settings, air freshener, and radio presets!

This idea that Maverick can only fly the plane with his name on it gets us back to the concept of "pets vs. cattle" systems—that a single piece of hardware has to be bound to a single pilot, and we cannot dynamically switch the pilot over to another piece of hardware through a "just-in-time" software deployment. Core to the architectural shift from vertical to horizontal scaling is making the actual endpoint systems both stateless and disposable, which introduces the possibility of just-in-time software deployment and configuration. The advantages of this approach include the ability to move to dramatically lower-cost hardware over time (i.e., ride the commoditization curve); scale up and down as needed; rapidly provision new systems; bypass bottlenecks in emergencies (if something

breaks, just provision to a new system and fix things later); and introduce new security advantages, since the endpoint systems always have the latest software and do not retain state. Data, which should be clearly compartmentalized away from system and application software, lives in the cloud and synchronizes to the endpoint as needed for local processing.

Beyond these obvious benefits, the DOD should start thinking about how to integrate newer technologies like artificial intelligence (AI) into existing systems and how to design new systems differently from the ground up. Currently, the DOD thinks of deploying a "global" hardware platform—for example, an unmanned aerial vehicle (UAV)—across multiple areas of responsibility (AORs). In a pre-AI age, this did not present any major problems. As the DOD integrates more AI-based software in these systems, however, the data the algorithms are trained on for each AOR can vary widely. For example, the data from Eastern Europe, which has snowy or muddy fields, and the South China Sea, which has open water and scattered islands, can be very different. AI software performs optimally when it is retrained from new data collected in an operational setting. In the digital age, fielded AI models must be updated regularly. Beyond Project Maven and the DOD Joint AI Center, there are few operational examples in the DOD of AI software fielded via continuous integration/continuous delivery (CI/CD) even though the approach was routinized in commercial industry a long time ago. The idea that fielded AI software is never finished and requires continuous updates is not a niche idea. At some point, almost every piece of software will be AI-enabled and, if embedded into hardware systems, will need to be designed for continuous and even over-the-air remote updates. As the DOD operationalizes this at scale, it will have to become much smarter about deploying just-in-time software on local systems custom-tailored to the mission.

Similarly, once a crisis or conflict begins, bugs and problems in fielded systems will require fast fixes. This is no different than how commercial products move rapidly from "minimum viable product" to demonstrated high-performance production products. The ability to turn around a patch or "dot release" of software and deploy them to fielded hardware systems will become a key differentiator in future conflicts. Just as Apple runs operating systems such as macOS®, iOS® (for iPhones), and iPadOS®, the DOD should envision a day where it has a TankOS®, FighterOS®, and ShipOS®—each running individual hardware systems. Each armed service should be able to deliver over-the-air patches and new apps to each OS for improving functionality and fixing bugs. There is no question that in the first phase of any war, the United States' adversaries will be launching a number of "zero-day" cyberattacks against its weapons systems vulnerabilities. The country's ability to get back into the fight will be dependent on the DOD's ability to spin patches for these vulnerabilities, clean up its endpoints, and do rapid updates of its systems before booting them up again. If the DOD adopted a stateless, just-in-time software deployment model, the time needed to close this loop will be dramatically shorter than having to clean up malware off of a stateful software/hardware system and ensure that it is malware free.

Concept: Design for Instrumentation

Since anyone can track the delivery of a hamburger or pizza on their phone in real time, it is unfathomable that the DOD's combatant commanders cannot gain real-time visibility into the endpoint systems that are under their management. Today, almost every product used is heavily instrumented to measure a variety of data: real-time control of the endpoint system, status of the endpoint, and (on the development side) understanding what features end users are utilizing. There are hundreds of great

products to choose from to both instrument and track this usage, and the DOD's endpoint hardware systems need to be designed and built with instrumentation integrated into them from the outset. (A critique of user interfaces/user experiences in the DOD is outside the scope of this paper. Unfortunately, UI/UX best practices remain a foreign concept to most DOD hardware and software system developers.)

The most important design element is to include sensors for every part of the system that could be relevant for remote control and orchestration, as well as any other components that are crucial to the operation of the overall system. The sensors should provide the ability to read such information as on/off status, unit values, and the like. The simplest and most telling example is that many of the resources and munitions on all airplanes and tanks are still not tracked automatically—with few exceptions and rarely in real time.

When a particular component includes one or more sensors, the next question is whether those sensors should be paired with the ability to automate the component. Obviously, any component with a read-only sensor cannot be automated, but it should be possible to automate the others. This includes test and evaluation (T&E) sensors capable of reporting to users when a fielded system is not performing as expected. This is especially important for systems that integrate so-called "black-box" AI capabilities.

The close friend of instrumentation is real-time connectivity and reporting. If real-time connectivity is not possible, then the next best option is an ability to synchronize prioritized data when the system regains a connection. Currently, the DOD's version of an iPhone sync is pulling out a hard drive from a helicopter, plane, or drone, then shipping it cross-country to a secure data center where it can be downloaded and processed. The military cannot continue to operate this way.

Concept: Simulation, Testing, Verification

Planning, exercises, and wargaming have always been an integral part of warfighting preparation. Scenes of military leaders and policymakers standing around tables with large maps that include board pieces representing personnel, tanks, aircraft, and ships—from the United States and its allies and partners, as well as the adversary—are sufficiently familiar. Each side makes a move, the outcomes of which are then assessed by impartial observers before the next round of moves. Military wargames like this take place regularly but not nearly frequently enough to account for the rapid pace of change unfolding in today's digital environment. All militaries have plans for different kinds of scenarios and contingencies, but few of these plans are tested in advance. Why?

First, simulation systems vary widely depending on what someone is trying to simulate. The major factors that affect a simulation are its size and scale (typically the number of objects being handled), the fidelity of each object (what level of detail do you simulate), and the resources you are willing to throw at an object. For instance, a first-person shooter game can act as a simulation for a relatively small number of soldiers in very high fidelity. The gun's operating can be replicated extremely accurately, along with a highly realistic small-scale simulation environment. On the other hand, complexity increases exponentially when attempting to run a simulation across the entire Indo-Pacific theater, with thousands of weapon systems and objects all interacting dynamically. Such a simulation would be incredibly complex, expensive, and probably unfeasible.

Second is the actual setup of the simulation itself. In almost every case, actual operational systems are not used for simulation—meaning real-world operational data is rarely ported over to the simulation systems. Even when this happens, the data transfer is complex, hindered by data size and protection

of classified information. This creates very high barriers to entry that are extremely hard to bridge. One such example is how the military services have been working for the past decade on ways to integrate live, virtual, and constructive (LVC) weapon systems. The Air Force's Virtual Flag program, for example, was designed from the beginning as a LVC joint and combined exercise. Despite considerable progress over the past few years, Virtual Flag still suffers from the combined challenges of scale, integration of multi-level classified systems, and fidelity. One promising option is to find ways to easily export or pull information from operational systems into simulation systems that have been designed to handle low-fidelity/high object-count or high-fidelity/low object-count simulations—essentially a dial that someone can use to manage the trade-off between these choices. Once built, it would make sense not only to be able to run testing and verification on top of these systems, but also to include concepts such as "Chaos Monkey" (see below).

Operational and contingency plans (to include branches and sequels) can also be described and explored as part of the system. There are numerous extremely lucrative technologies such as generative adversarial networks and digital twinning that can be built and run on these platforms. This approach would allow the DOD to use AI-enabled technologies and other capabilities to test different adversarial strategies and techniques, as well as explore ways to counter them (some of the most promising work in this area includes game-theoretic competitions). The problem is that these can only be developed if the DOD has the right platforms, designed with the right APIs, upon which to build. So far, this remains much more of a pipe dream than reality.

Concept: Design for Failure and Let the Chaos Monkey Loose

Netflix built an open-source product called "Chaos Monkey" it used for testing complex and highly available software systems. Like a crazy monkey having a good time, it would randomly kill parts of a software system to see if they would recover and continue running. This has now been expanded to include the "Simian Army" family of tools that expand the testing concept.

Running Chaos Monkey on your software is the equivalent of the "power-cord test" in a physical data center—essentially, randomly pulling a power cord of a running computer system or router to see if all the high-availability features of the site remain operational.

The DOD should consider Chaos Monkey in two forms: first as a traditional testing product that the Department can use to test its backbone with warfighting software systems and, second, to test warfighting plans in a simulation environment to see how they actually unfold in situations characterized by random and uncontrollable changes (the inevitable friction that happens in the real world). Both approaches would allow the DOD to begin to design more for resiliency and anti-fragility than for optimizing performance under the kind of perfect conditions that simply never exist. This is, of course, only possible if the DOD can build software simulation systems that accurately represent its fielded systems under the full spectrum of realistic operating conditions.

Concept: Build Autonomy and Automation at the Edge

When any system is software defined, it is typically the case that the control path and data path are pulled apart. Control becomes centralized, leaving the actual endpoint system to execute its mission with complex control logic, driving up cost and complexity. It is obviously not realistic or optimal

for every small control decision to be made centrally. The latency between the endpoint and central control logic would be too high, and if the communication link were severed, the endpoint could not continue to function. This is also a case where the types of endpoints found in a military context (such as an aircraft, vehicle, or ship) are much larger and more complex than a typical computer network.

The argument against this architecture is that if both the larger and smaller systems require manual management, then simple math shows that the DOD will require many more people to scale because it has more systems. As a matter of fact, the limiting factor in drone flights today are the crews to pilot the craft, not the actual hardware. For instance, each MQ-1 Predator or MQ-9 Reaper remotely piloted aircraft (RPA) system requires a pilot and sensor operator, along with a fairly large behind-the-scenes team of maintenance personnel and intelligence analysts (up to about 100 people total). As a business model, this does not scale well at all, as even with a relatively inexpensive piece of hardware, the human cost of operating the system remains quite high. Adding UAVs to the current architecture—wherein one pilot flies a single RPA—does not yield much of a cost or operating advantage (that is, there are relatively limited economies of scale). If anything, the DOD ends up with negative economies of scale: as the Department flies more systems, the harder and more expensive operating each aircraft gets. Unless they have an infinite supply of money, commercial companies with negative economies of scale quickly go out of business. The way around this is to figure out how to remove the expensive overhead of human involvement through more automation and autonomy, up to and including AI-enabled systems.

Granting autonomy to these endpoint systems seems to run counter to the entire discussion of centralizing control through orchestration. The right way to think about it is to consider the scope of orchestration any particular system needs to have. An individual drone, for instance, should be responsible for the autonomy functions that are relevant to it, including taking off, landing, flying, and some route planning. If it is programmed to perform a certain function—perhaps identifying and tracking certain objects—then it should also be able to perform those functions without extensive human intervention or a central control system managing even the smallest tasks. But a single drone should not be responsible for orchestrating the functions of hundreds of other drones, nor should it be responsible for coordinating between a swarm of drones and armored vehicles on the ground. Those orchestration functions should be done by levels of software above, with a singular endpoint playing the right role and function within the broader context of all the systems being orchestrated.

Concept: Integrate Development Environments with Production

Currently, the DOD's core code-development environments are not tightly integrated with the deployment of this code at the endpoints. The Department has an entire spectrum of development environments that are all trying to address similar issues, but there are no simple pathways from code development—which is done in these specific enclaves—to production environments. It will be even more challenging to deploy code to an actual endpoint (a piece of hardware such as a drone, ship, vehicle, or manned aircraft). While these types of deployments are not trivial, given that they require substantial pre-deployment testing and evaluation, accelerating this development–production–fielding–sustainment cycle will be critical to future success and gaining a competitive battlespace advantage. This underscores an earlier critical point: whereas the DOD could once claim the mantle of being bigger than anyone in the world and, as such, could rationalize refusing to adhere to commercial best practices, today the world's biggest tech companies deal with eye-watering speeds and scale well beyond anything the DOD regularly sees.

Containerization, which allows for the right packaging of code that can be "sealed" and deployed to the endpoints in a predictable manner, plays an important enabler role. This will require endpoint system developers to enable newer systems to handle containers and to build these systems to be receptive targets of development, security, and operations processes.

This becomes even more important when a capability such as AI is integrated into these systems. Unlike traditional software, AI software is never done. The more operationally relevant data that is gathered and the more that an AI-based system performs in the operational environment, the more retraining and updating is needed. This loop of gathering data, retraining, and redeploying code is an important one to master and can only be enabled if these development environments are built the right way, with consideration of the end production environments.

3

Putting It All Together

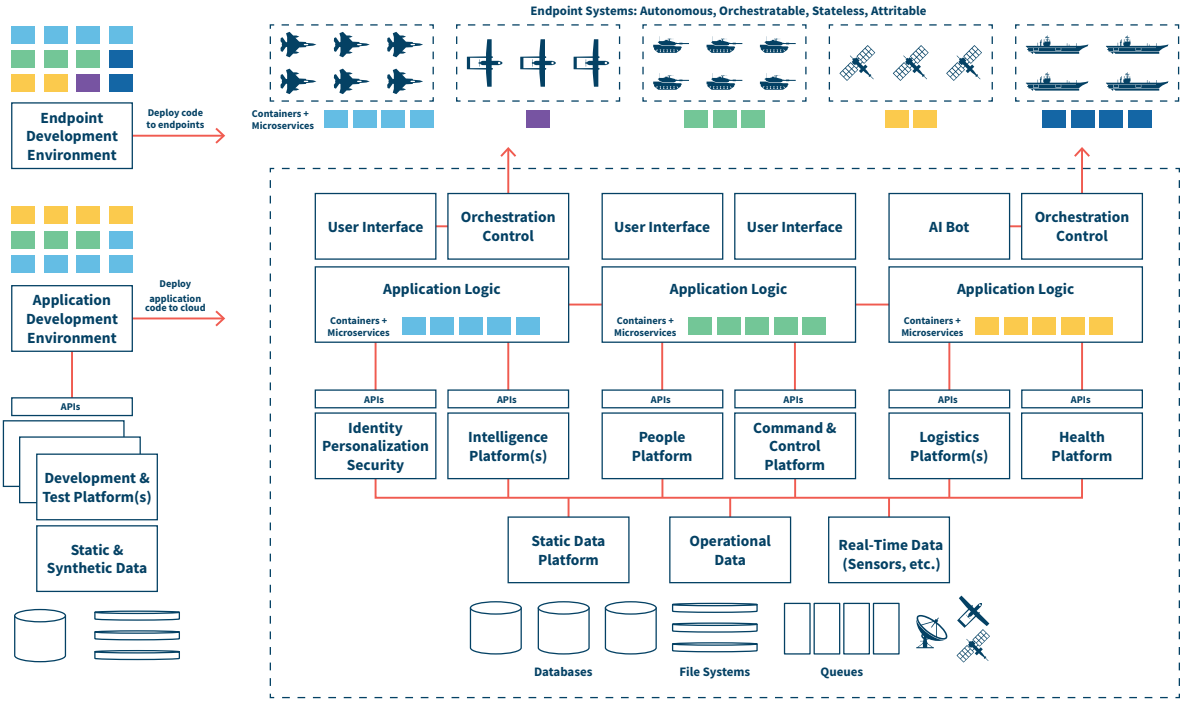
Applying these core principles across the DOD can seem impossible. Large technology transformations at legacy, industrial-era organizations present seemingly insurmountable obstacles. Indeed, most of the successful examples of organizations that adopted the aforementioned software best practices have involved digital-native companies that did not have to deal with legacy systems. And these companies almost never faced the byzantine DOD, in which there are thousands of hardware and software systems built by hundreds of different vendors, few of whom had to adhere to any core architectural standards that would be considered commercial software best practice.

This report provides a blueprint for the way forward. The DOD should adopt these core concepts as it designs, develops, fields, and sustains all its weapon systems and supporting hardware. Think of a software "core" with an outer hardware "shell"—in essence, running the logic, planning, and optimization in the software plane, with the hardware tethered to the end of a long chain of software. In this idealized architecture, end-to-end software architecture connects all the relevant sensors (inputs) and outputs (weapons systems). It would also make it easier to model all the right workflows (kill chains), along with predictive intelligence systems, logistics, targeting, and fires, all of which would be integrated into a single architecture built on a few large-scale core platforms.

At the heart of all these large networks of systems—whether autonomous or manually controlled—will be complex, internet-scale, highly available software that pulls everything together. The underlying software systems will help bind all the DOD's people, weapons, logistics, and intelligence—and, equally important, pull together the Department's development and production environments to allow for more rapid code development, deployment, and sustainment.

There are two critical steps required to pull off this kind of transformation. First, the larger problem needs to be broken down into smaller and smaller parts that are independent, allowing the creation of structured interfaces and APIs around them. This is especially important since each armed service runs its own infrastructure and applications, which will probably be the first to be cut, followed by segments underneath them. Second, the DOD should build the core scaffolding around which the other parts of the system can be "glued on" over time. Such a "loosely coupled, highly aligned" architecture could be expanded over time by bringing over and integrating more parts into the system instead of trying to upgrade or rewrite every part of the problem at once.

Aspirational Core System Architecture for Software-Defined Warfare



Source: Designed by Nand Mulchandani.

As emphasized earlier, the most important part of this process will be to define an overall architecture with the right interfaces and abstractions. In addition, the DOD should use well-established and proven industry-standard and open-source best practices to build these types of interfaces (APIs, REST), data-exchange formats (JSON, XML), storage and representation methods (time-series databases, event/pipeline management systems), application packaging (containers), and the runtime architecture (microservices). At the same time, this approach should disconnect the data platform from the application logic, as well as the user interface and presentation services from the backend data and application logic (through APIs and other interfaces). This will allow each layer of the system to evolve independently without having to build a highly interconnected but fragile stack wherein even small changes will ripple through the architecture to every part of the system, making code updates extremely hard and expensive.

At the heart of the DOD's software architecture should be various software platforms to be built and operated across the department. These platforms provide a necessary divide between the data/core-

logic layers and the applications. Separating the two provides an environment in which building a new application is dramatically simplified since all the heavy lifting of aggregating, cleaning, and integrating data is already done at the platform level. Additionally, the DOD should insist that any vendor implementing a platform and APIs build these as open interfaces that are well documented, version controlled, rent free, and availably freely to any vendor who wants to build upon them.

When this happens, the application writer need only be concerned about their own application logic, as they can freely rely upon any of these backend platform services for data and some common logic. Since they will not have to re-import, clean, and rebuild the entire data environment from scratch and recode common business logic, they will be able to rapidly build new functionality at a dramatically lower cost. Multiple vendors can also build competing products on the platform, driving down costs and increasing the quality of the applications for the DOD customer. Finally, when the Department owns the interfaces (the APIs, not the actual applications themselves), vendor lock-in will be lessened: no vendor will be able to hold the customer (DOD) hostage since all the data is captive in a system, with no way of exporting or accessing it. This is known as the "Roach Motel" problem, in honor of an aggressive late-night infomercial focused on cockroach removal that boasted, "Roaches check in, but they don't check out!"[5] The problem in this case is that data does not check out. In addition to enabling external vendors to build new applications, this architecture would also make it dramatically easier for internal DOD coders and data scientists to build smaller applications rapidly or enable new analytics on top of existing data. The same barrier of cleaning, importing, and normalizing data that exists for a smaller vendor is even larger for an individual internal developer. Immediately opening this up to the broader internal developer community will likely spur a dramatic increase in the kind of "micro applications" that are simply not developed today due to high costs and high barriers to entry.

4

Conclusion

Warfighting has always been viewed as some esoteric combination of art and science that is incredibly complex to manage. The size, scale, and speed with which events take place and change continuously, even chaotically, can make it seem nearly impossible to master either the art or the science. In the future, warfighting will only become more complex, even more chaotic, and even faster. The art side of the art-science equation remains as important as ever. However, with the proliferation of new technologies and a shift to a data-centric world, the science part of the equation is becoming increasingly consequential. In future crises and conflicts, the side that adapts faster and demonstrates the greatest agility—to include rapidly updating and promulgating fielded software and AI models—is likely to gain a significant competitive advantage.

The only way for the DOD to stay competitive in this new warfighting environment is to ensure that it uses the most potent weapon available: technology, and more specifically, software. The commercial computer industry used to manage data centers the way the DOD has always conducted warfighting but realized that it did not scale. The industry came up with new tools and technologies to virtualize, automate, monitor, and scale data centers, which in turn have allowed the industry to offer services at a fraction of the cost with almost no downtime.

One of the greatest challenges the DOD and other U.S. federal agencies face is that they were built from the ground up as industrial-age, hardware-centric organizations. Making the transition to digital-age, software-centric, more risk-tolerant organizations is exceedingly difficult. But it is also the only path to future success. The template exists; implementation is now the hard part.

While many of these concepts from the software industry may not have perfect analogues to military hardware and warfighting, it is important to understand the core concepts outlined in this paper and

the effects they have on architecting large-scale, complex systems. The key is to begin thinking of military objects and hardware as part of a larger system that is woven together using software. This will not only prevent the most expensive and scarce resources from being subject to mind-numbing legacy workflow processes but will also dramatically accelerate information and decision cycles. At the same time, by automating most of the routine, mundane functions that currently absorb far too much of everyone's time, implementing these software best practices will allow military leaders to spend more time focusing on the critical decisions they need to make.

Software-defined warfare is the way of the future. Of course, this should never mean that humans are redundant or uninvolved. To the contrary, the approach described in this report opens up incredible new opportunities for human-machine teamwork, in which the roles of humans and machines are optimized based upon what each does best. Business objectives, goals, and the overall architecture of the systems are still all set by humans. The actual operations are handled by software, based on parameters established by humans. Any exceptional conditions or issues are still escalated to humans. The best way to characterize this change is to "take the robot out of the human" by electronically connecting end-to-end processes, leaving the human with the time to concentrate on making the most important and consequential decisions. In a fully AI-enabled future, UI/UX will advance to the point where users will train smart machines so systems can adapt to an individual's performance, the pace of their cognitive development, and even their past behaviors. Given the state of its hardware and software, there is no possible way the DOD can do this today.

We wish the best for our country and, in our time at the Department of Defense, have tried our best to institute and implement best practices to orient the United States in this new direction. We hope this work continues and these ideas continue to make their way into the thinking and design of our next-generation warfighting systems.

About the Authors

Nand Mulchandani is the inaugural chief technology officer (CTO) of the United States Central Intelligence Agency (CIA). Prior to joining the CIA, he served as the CTO and acting director of the Department of Defense Joint Artificial Intelligence Center. He also cofounded and was CEO of several successful start-ups: Oblix (acquired by Oracle), Determina (acquired by VMWare), OpenDNS (acquired by Cisco), and ScaleXtreme (acquired by Citrix). He has a bachelor's degree in computer science and math from Cornell, a master of science in management from Stanford, and a master in public administration from Harvard.

Lt. General (Ret.) John N.T. "Jack" Shanahan retired as a lieutenant general in the United States Air Force. From 2018 to 2020, he served as the inaugural director of the Department of Defense Joint Artificial Intelligence Center (JAIC). Prior to leading the JAIC, he led the creation of the Algorithmic Warfare Cross Functional Team (aka Project MAVEN) as the director for defense intelligence (warfighter support), in the Office of the Under Secretary of Defense for Intelligence. Over his 36 years of service in the Air Force, he accumulated more than 2,800 flight hours and flew F-4D/E/G, F-15E, and RC-135 aircraft. He is a graduate of the University of Michigan, the Naval War College, the National War College, and North Carolina State University.

Endnotes

1 Marc Andreessen, "Why Software Is Eating the World," *Wall Street Journal,* August 20, 2011, https://www.wsj.com/articles/SB10001424053111903480904576512250915629460.

2 Christian Brose, *The Kill Chain: Defending America in the Future of High-Tech Warfare* (New York: Hachette Books, 2020).

3 Bob Work, "Remarks by Deputy Secretary Work on Third Offset Strategy," U.S. Department of Defense, speech delivered April 28, 2016, https://www.defense.gov/News/Speeches/Speech/Article/753482/remarks-by-deputy-secretary-work-on-third-offset-strategy/.

4 Frans Osinga, *Science, Strategy and War: The Strategic Theory of John Boyd* (Delft: Eburon Academic Publishers, 2005), http://www.projectwhitehorse.com/pdfs/ScienceStrategyWar_Osinga.pdf.

5 See recordman33, "Black Flag Roach Motel Commercial- 1983," YouTube video, uploaded May 23, 2012, https://www.youtube.com/watch?v=d55RPr5W0ys.

COVER PHOTO
U.S. AIR FORCE PHOTO BY J.M. EDDINS JR.

CSIS | CENTER FOR STRATEGIC & INTERNATIONAL STUDIES

1616 Rhode Island Avenue NW
Washington, DC 20036
202 887 0200 | **www.csis.org**