SYM-AM-23-088

Excerpt from the Proceedings of the Twentieth Annual Acquisition Research Symposium

Acquisition Research:
Creating Synergy for Informed Change

May 10–11, 2023

Published: April 30, 2023

Crossing the Great Software Development Divide within DoN

Chris Johnson—is the Naval Information Warfare Center Pacific Division Head for the Command and Control Systems Division. Chris was the co-creator of the NRDE Commercial Cloud and led the development of the Collaborative Software Armory, now the Overmatch Software Armory (OSA). He is a retired Operations Specialist Chief Petty Officer in the US Navy. Mr. Johnson holds a Bachelors of Science in Information Technology and a Master's degree in Software Engineering from California State University Fullerton. [christopher.e.johnson40.civ@us.navy.mil]

Amanda George—is the Naval Information Warfare Center Pacific Business Portfolio Manager for Business and Force Support (BFS). The BFS Portfolio encompasses a wide range of projects providing innovative and agile business and security solutions to the Warfighter. As the Business Portfolio Manager, Amanda supports the BFS projects by engaging with industry and stakeholders, building internal collaboration between projects, and providing support for strategic planning. Amanda holds both a Master's Degree and Bachelors (Phi Beta Kappa) from the University of California, San Diego (UCSD)**.** [Amanda.j.george7.civ@us.navy.mil]

David Jenkins—is the Deputy for Business Development for Naval Information Warfare Center Pacific's Command and Control Systems Division. In this capacity, David applies his extensive knowledge of contracting, marketing, and project management to support the development of hardware and software-based command and control systems. David holds a Bachelor's degree in International Business from San Diego State University and a Master's Degree in Business Administration from National University. [david.w.jenkins5.civ@us.navy.mil]

Abstract

The US Navy must undergo a software development cultural transformation in order to address outdated considerations associated with pushing its development efforts successfully into the future. The current processes to develop software and acquire key capabilities needed to develop software, and the culture that these processes produce is slowing down the Navy's ability to provide crucial technologies to the warfighter quickly. Naval Information Warfare Center Pacific (NIWC Pacific) has been on the forefront of utilizing cutting-edge software development techniques and enabling technologies, and thus has some key lessons learned to share with the acquisition community. This paper will look at three (3) key areas: 1) Acquisition processes via contracting 2) Software development security approaches, and 3) The Navy's financial ownership of software development. This paper will explain why they are key, and provide recommendations to transform the way ahead for the US Navy in its software development efforts.

Introduction

"In this era of competition and race for digital dominance, we cannot settle for incremental change. The Department must join together to deliver software better and operate as a 21st century force."

- Department of Defense Software Modernization Strategy (2022, p. ii)

The DoD's Software Modernization Strategy succinctly summarized the importance of DoD's ability to deliver software, saying "fighting and winning on the next battlefield will depend on DoD's proficiency to rapidly and securely deliver resilient software capabilities" (2022, p. 1). The key to this is the focus on rapid and secure delivery of software. If the DoD and the Navy can't deliver software rapidly, it will be too late to support the fight. If they can't deliver resilient capability, then we will never succeed in a contested cyberspace. The DoD's Software Modernization Strategy then goes on to identify a practical approach to "unify efforts across DoD and partner with industry-leading software institutions to produce a portfolio of best-in-class software capabilities enabled by DoD processes" (2022, p. 2). While this is in fact necessary, it's not a sufficient approach. Current acquisition and security processes developed to acquire and secure large-scale physical vessels, vehicles, and machinery, along with a lack of financial commitment from the Department of Navy are major blockers that severely degrade the ability of the Navy to support the rapid pace of product delivery required to defeat our adversaries. The largest blocker, however is a cultural divide within the Navy and DoD surrounding whether and how to adopt a new, agile, resilient software development mindset.

The US Navy must undergo a software development cultural transformation in order to address outdated considerations associated with pushing its development efforts successfully into the future. The current processes to develop software and acquire key capabilities needed to develop software, and the culture that these processes produce is slowing down the Navy's ability to provide crucial technologies to the warfighter quickly. Naval Information Warfare Center Pacific (NIWC Pacific) has been on the forefront of utilizing cutting-edge software development techniques and enabling technologies, and thus has some key lessons learned to share with the acquisition community.

Acquisition Processes via Contracting

Providing capability to warfighters and meeting program requirements is often less a technical challenge than one of acquisition. Taking a quick glance around displays at a trade conference, such as WEST 2023, you will see showcased many new capabilities primed for transition into Navy programs. Industry experts in building custom Government applications are looking for teaming opportunities with Navy customers to develop products. Indeed, industry is poised and ready to help solve some of the Navy's biggest technical challenges, but in order to partner, a contract has to be issued. Long contract lead times, however, are working against accelerating delivery of new capability to the warfighter.

The current acquisition system was created to facilitate the acquisition of large-scale physical procurements of everything a military might need from bullets to an aircraft carrier. This process was designed to reduce the risk inherent in procurement. This risk reduction focus has created a culture in the acquisition community that highly prioritizes set requirements. Sacrificing agility in favor of risk reduction is fundamentally opposite to the culture we need. We need a contracting approach that can move agilely, so we can implement industry solutions at the speed industry is creating them. We need a contracting workforce that is empowered to apply the best, tailored contracting approach for the procurement need. We need a contract acquisition environment that matches agile software development principles.

At the present, the acquisition environment is not conducive to working as swiftly as need requires. These days, software development most often occurs on a "two pizza" sized team, building small applications that can be created over a short (less than a year) time frame, in an environment where requirements are not stable and require day-to-day interaction with the warfighting customer to solidify the design and function of Minimum Viable Product (MVP). The Navy's contracting approach should reflect this dynamic. Instead of the current contracting standard requiring highly specific and well-defined requirements, "good enough" requirements and evaluation criteria should be the goal in such an agile development environment, where risk-taking by both the Government and their industry partner should be encouraged and even rewarded.

When agility is required, the time to award to an industry partner must be short in nature. At many Military commands, spending months on requirements development is the norm. Identifying requirement specifics to an acceptable level of detail and documenting justifications for changes to standard processes significantly increase the time needed to execute a contract. Spending months nailing down requirements should be an outlier, not the norm. A contract needs to capture the essence (i.e. most important aspects) of the requirements at the development onset, and it should allow for evolving and emerging requirements all the way through to MVP; it need not eliminate all risk nor capture all requirements needed in perpetuity. Task order award timelines, in particular, must reflect product teams' realities, and should be measured in weeks not months.

What do we need to do to change the mindset?

So how do we, both requirements owners and contract owners, change our mindset? The solution is likely two-pronged: educate the local contract and technical personnel to effectively communicate agile software development requirements, while also providing nontraditional contracting approaches to increase speed and access to nontraditional contract partners. At a local level, NIWC Pacific has focused on working with our technical and contracts personnel to ensure that they each understand the requirements of modern software development. The DoD has made significant efforts, laid out in the 2022 DoD Software Modernization Strategy to "make the acquisition lifecycle and the funding of software programs more agile." (pg. 13) In addition, NIWC Pacific has been looking at options like the Information Warfare Research Project (IWRP) program using Other Transaction Authority (OTA) to bring in prototypes and small efforts more quickly. While this is happening to a degree locally, the larger issue is shifting the culture of contracting away from the idea that a procurement team (to include both requirements owner and contracting team) must spend months on getting requirements exactly right before making a purchase. Agile software development needs smaller increments of improved capability, rolled out at scale with a rapid refresh rate. Such a cultural shift is challenging, but it could happen if it gained momentum at multiple commands and within the larger DoD community.

Software Development Security approaches

"Hey, my code is secure because I filled out the RMF spreadsheet!" - said no one, ever.

While you likely laughed, this statement reflects how we currently manage software security requirements. A large part of what we do as software developers is focused on filling out certain Risk Management Framework (RMF) artifacts, as if by doing so we have secured our code and made our systems safer. In years past, after programs went through months of coding development that software would be integrated into the larger-scale system to ensure module-to-module interoperability. Concurrent to this activity happening, the security evaluation would begin by a separate engineering team tasked to ensure that DoD RMF guidance locked down security vulnerabilities, in accordance with policy, to ensure no High Risk vulnerabilities were present which would prevent the application from acquiring the all-important Authority to Operate (ATO). All-too-often, the activity of locking down these identified security vulnerabilities created new issues that prevented the applications from performing as intended. The application would then be banished back into development for re-work, re-test, then more re-work, etc. This was not only frustrating and slow for all parties involved, but it has also been hugely expensive, as can be attested by virtually everyone that has ventured to release a DoD application.

Another primary concern with software development associated with security approaches, was the common approach of packaging up enterprise or infrastructure elements (i.e. databases, service buses) and incorporate them into their new application package. This gave a false promise to programs that doing so would assure success when going live in the production environment. Unfortunately the opposite often held true. More often than not, the applications instantly inherited the vulnerabilities of the associated service or infrastructure dependency, and once that happened, they were doomed at being able to successfully achieve an up-check security accreditation. Removing such dependencies proved equally complex and difficult, and many programs were cancelled or abandoned over such issues, sometimes resulting in negative publicity for the Navy.

Our desire as a forward-leaning software community is to provide a way to gain early, accurate insight into the security posture of software during the build phase of development. So early, in fact, that immediate feedback can be internalized by our development teams and immediately address high and moderate risk issues. Today's Development-Security-Operations (DEVSECOPS) environments are designed with tooling that provides a wide array of testing that covers many aspects of security issues and vulnerabilities. Results, then, are consolidated and analyzed while still in the software factory to evaluate any potential false positives, and at essentially the same time ready replacements can be offered. These powerful and always changing tools help alleviate the potential for wasted work years chasing issues that were minor, at best.

These tools provide the developers a direct pointer to the very lines of problematic software code, allowing them to focus their effort on writing better code rather than searching through endless lines of code for an identified vulnerability. In some cases, these tools take it even a step further, and provide not only the exact line of code that is broken, but also can implement a prescribed correction to fix the issue. Such prescribed fixes can save developer time from having to research security fix precedence on something that may be unique or something they are not familiar with. Such an approach is a more secure way to address the cyber posture of our software; and, once again, this can all be done in near real-time. Using these tools, evaluators and decision makers who control fielding decisions allowing production software to be fielded can quickly see how secure the software is within minutes of going through the build phase. In fact, actual fielding decisions could be made months ahead of time, providing for hour-to-hour fielding decisions if required. During the transition, and for those "old-schoolers" that still want a spreadsheet to review, these automated tools can provide those at the touch of a "print report" button.

In addition to helping software developers go faster, the other part of the software security approach challenge is how we as a community adopt a reciprocity approach that would help lessen the burden of fielding systems today. In a current production model using cloud, there are three levels: 1) Infrastructure as a Service (IaaS), 2) Platform as a Service (PaaS), and 3) Mission Applications. Each of these, by definition, also carry a certain level of requirement for cyber accreditation. The majority of the requirements for any ATO reside within the IaaS. All core components of the infrastructure are contained within the IaaS, such as databases, enterprise services, etc.

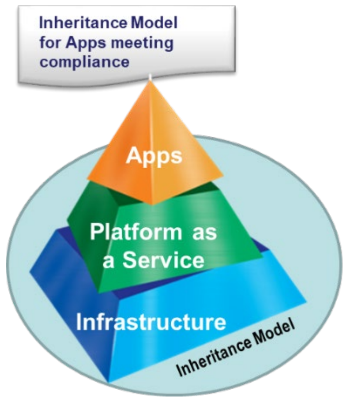

Figure 1 Tri-Tier Security Inheritance Model

In Figure 1, infrastructure is the base of everything that is required for the system and applications to operate. In a typical production example, such as a shipboard environment, the CANES infrastructure serves as the shipboard IaaS, and is the holder of the ATO for the environment. It also contains the bulk of associated RMF security requirements. The other dependent tiers - the PaaS, and Mission Applications - rely on the IaaS to have these items available and then leverage those items for their ATO package.

Next, the PaaS contains the RMF security items necessary to operate the services and elements associated with that PaaS. Again, building on the RMF inheritance model, this enables the mission applications to focus on delivering just the mission component without having to worry about adding in missing services necessary to communicate with the IaaS or other PaaS components. This keeps the Mission Application requirements for RMF lightweight, and easily achievable when tools within DEVSECOPS pipelines can readily provide security insight and accelerate the development-to-delivery timeline. The Navy needs program offices that ***trust*** the results provided by the software pipeline tooling. This is the very essence of what RMF (emphasis on the "R") means, and such trust is imperative to delivering code to the warfighter quickly. Trusting these tools to do the job they are built for would represent a healthy shift in culture that will result in an accelerated delivery of software capability and fixes to the Fleet. All too often, the Naval community is not willing to allow for ***any*** risk, and everything down to lower-level vulnerabilities have to be accounted for with a Plan of Action and Milestone. While these reports are important, very few of them have anything to do with the actual operating posture of our applications and how secure they are.

Navy's financial ownership of software development

Software development is a highly skilled complex task; building a viable software application is equally as hard as building a ship, flying an airplane, or designing a submarine. Similar to building our physical assets—ships, jets, missiles—it is an endeavor that takes tremendous resources of people and specialized tools. In addition, like a Navy shipyard, software development relies on a set of infrastructure tools that need to be provided to the software developers.

For years the Navy has relied on industry building Navy software, and for years we have struggled with the fact that much of that software is self-contained. It does not interoperate well with other applications delivered by other vendors. The code platform might be different, the services might be unique, or the design pattern might not fit the architecture. We typically discover these problems the minute we try to integrate some application into a larger system.

Industry controlled software development was a paradigm long overdue to shift, and recently has, toward Government-owned and operated environments, using the best available industry-developed tools. Requiring each developer to develop within the same Government environment is an acceleration method to ensure that software can interoperate with other applications sooner in the lifecycle. Having a common set of tools helps to confine or bin the development environment, ensuring that there is a smaller risk of language and service diversity inside the applications. To do this, however, requires the Government to be willing to undertake the cost of owning and maintaining those development environments.

A typical environment consists of a cloud service provider, code repositories, cyber security tools, agile project management tools, and engineering to ensure that all the tools are set to the correct security level, are accessible to users, and are integrated. Unfortunately, the tools required to produce these environments are not "plug and play," particularly given the Government's security requirements. Therefore, each environment comes with a financial cost that is split between Government labor, contractor labor, and licensing fees.

Given this cost, the Navy must choose to either fund the entire software development pipeline, or share that cost among the users of the software development pipelines. The DoD and the Navy have chosen the latter, enabling and encouraging a multitude of software development pipelines to form. As the GAO states in a recent report (2022), "[a]s of August 2022, the DOD has established 29 software factories across the department" (p. 19) Each of the software pipelines is supported, in large part, by the programs and projects utilizing their services.

The cost of each software factory is initially bourn by the organization that needs it. As the RAND Corporation addressed extensively in its 2020 report "Personnel Needs for the Department of Air Force Digital Talent A Case Study of Software Factories":

> Because the software factories are start-up organizations, funding for most of the factories at the time of this study (FY 2020) appeared to be ad hoc, with the parent or owning organization providing initial funds and billets but the majority of funding coming from customers that use their software development and platform capabilities. Although software factory missions primarily focus on serving their parent or owning organization, they also have customers that expand beyond the owner, including the broader DoD.(p. 2)

The software factories spread the cost of the engineering support and tool licenses across all the users. This model, where the command provides an initial investment, then the programs pay as they need the services, creates a significant challenge. If there aren't enough programs that utilize the unique offerings of a software pipeline, economies of scale are not achieved, resulting in the pipeline becoming unaffordable for projects.

The cost of operating in a software development pipeline for small software development efforts is often much greater than their resourcing. In the face of large price tags, small software development efforts often "go without" such pipelines and an ad hoc, build-it-as-you-go approach, often resulting in unforseen costs and fewer capabilities. The culture needs to change so that these small projects can identify the software development pipelines as a clear requirement to their programs, and thus budgeted for. Smaller programs have to accept as a given that top-tier DevSecOps is a "must have" rather than a "nice to have."

Larger-sized programs often have a different response to the large price tag associated with software development pipelines; they simply decide that it would be easier and cheaper to build their own software development infrastructure. These projects view the large cost associated with a software pipeline, then underestimate the costs of procuring, integrating, and securing their own environment. These programs then end up spending a great deal more than anticipated due to unrecognized engineering requirements, unanticipated ATO accreditation requirements, and license fees. These larger-sized programs needs to recognize up front the inherent benefits of using an existent software development pipeline.

Many pipelines have been created to address unique challenges that DoD software developers may face, such as where the software will deploy, be it a fighter jet, an unmanned system, or a Naval ship. One NIWC Pacific-developed software pipeline, for example, is designed to support deployment of software on Naval vessels. This specific pipeline is tailored toward delivering software to afloat units where software delivery is extra complicated due to low bandwidth availability of shipboard networks, and the potential for at-sea degraded communications. This can and does present the software developer extra challenge, since they must ensure that not only must they pack all the features necessary into only the containers destined for a shipboard delivery, but also not eat up precious satellite bandwidth for prolonged periods. Building and maintaining such an environment is a tremendous undertaking. It requires unique understanding of the problem space, resources that understand not only how DEVSECOPS works, but also how to rapidly accredit and release software that is secure and of low risk to the warfighter. Unfortunately, these unique requirements also come with a strong demand for specialized tools and strong engineering support for the software development environment, and ultimately, a large price tag.

At the moment, the Navy is currently dealing with a number of disparate software pipelines, each with a small base of users, procuring licenses in silos, and developing their own technical expertise. Given the increasing importance of software in all parts of the Navy, it makes business sense for the Navy to start viewing support of software development pipelines as a larger platform, with a dedicated funding stream to ensure the software pipelines are secure, agile, and don’t cripple smaller programs with cost. Getting to this ideal requires the Navy to stop thinking of software development as an add-on or enabler, and start seeing it as a key asset in the force. Software pipelines may not be as tangible and photogenic as our fighter jets and aircraft carriers, but they are just as important in ensuring we will prevail in any future conflict.

The Navy needs to commit to “owning” their various software factories to truly realize the power they could bring to rapid delivery of capability, interoperability, and common service adoption. Such benefits easily justify the investment. The Navy engineering community universally understand the need to have such development environments, as well as how to manage and operate them. The aforementioned NIWC Pacific-software pipeline was created utilizing scant innovation dollars, as were most other pipelines. The Navy now needs to commit programmed dollars toward adopting and maintaining these software factories at scale.

Summary

We are writing our software code in an agile manner, but we aren’t acquiring it, securing it, or financing it in a similar manner. By moving our software engineering community to an agile mindset, it exponentially improved the speed and quality of the code we produce. Now, we need to move our acquisition, security, and investment communities towards that same agile mindset! We can’t buy, secure or produce software to combat 21st century problems with a 20th century purchase, security and investment mindset.

References

Keller, K.M, Lytell, M.C., & Bharadwaj, S. (2022). *Personnel Needs for Department of the Air Force Digital Talent: A Case Study of Software Factories.* RAND Corporation. https://www.rand.org/content/dam/rand/pubs/research_reports/RRA500/RRA550-1/RAND_RRA550-1.pdf

Office of the Deputy Secretary of Defense. (2022). *Department of Defense Software Modernization Strategy.* https://media.defense.gov/2022/Feb/03/2002932833/-1/-1/1/DEPARTMENT-OF-DEFENSE-SOFTWARE-MODERNIZATION-STRATEGY.PDF

United States Government Accountability Office. (2023). *Software Acquisition: Additional Actions Needed to Help DoD Implement Future Modernization Efforts.* https://www.gao.gov/assets/gao-23-105611.pdf