Overmatch Software Armory: Overcoming Barriers to Deployment of Modern Naval C2 Software

C. Johnson
NIWC Pacific
Christopher.e.johnson40.civ@us.navy.mil

A. George
NIWC Pacific
Amanda.j.george7.civ@us.navy.mil

D. Jenkins
NIWC Pacific
David.w.jenkins5.civ@us.navy.mil

Abstract

Modern naval command and control (C2) requires software to enable the rapid ingestion, synchronization, and understanding of a multitude of data required to inform naval activities. The naval C2 environment is both highly complex and extremely critical, requiring the Navy C2 systems and software to rapidly react to mission needs to its ships at sea in bandwidth challenged environments. Given the vital nature of C2 software in the modern battlefield, it is necessary for the military enterprises to prioritize the rapid development and delivery of software. This will require a paradigm shift.

For decades software delivery to US Navy ships occurred by engineers and scientist packing up volumes of CD's or 4MM tapes, in some cases scheduling airfare to fly to wherever the ship may be. Weeks would be spent aboard the platform loading the latest version of software, testing it out to ensure everything functioned properly, and then going through a turnover with ships company for acceptance once everything was up and functional. NIWC Pacific has changed that paradigm by realizing the promise of DevSecOps for rapid and responsive software development, and "Over the Air" (OTA) delivery of new software and version updates to Navy ships.

Changes to methods of software development from the traditional "waterfall model" to "the iterative model" then the "spiral model" to "agile", and now "DevSecOps" have evolved the process of software delivery to provide rapid capability advancements to the warfighters wherever they are, whenever they need it. Additional designs and development patterns have changed from monolithic large bang applications to a smaller micro-service architecture where containers and smaller size applications are delivered enabling more efficient use of the limited bandwidth that ships are forced to operate with.

NIWC Pacific has responded to this challenge by creating a robust DevSecOps environment and methods of deploying software "Over the Air" (OTA), with limited to no onboard support for installation. It also provides for the possibility of rolling back to a previous version should a new install prove problematic. The challenge of delivering rapidly responsive software capability OTA for C2 has provided many lessons learned in overcoming challenges that are applicable to the weapons systems software development community.

This paper details the creation of the DevSecOps environment, the Overmatch Software Armory, that provides Command and Control (C2) capability to ships and submarines across the US Navy and provides a set of lessons learned for the community to utilize in overcoming common barriers to C2 software development and deployment across US military services and with international partners. Additionally, this paper proposed a set of metrics to measure the progress of the paradigm change and the benefits of utilizing the Overmatch Software Armory to rapidly deploy C2 software over the air.

1 Introduction

For decades, software delivery to US Navy ships has occurred by packing up volumes of CD's or 4MM tapes, and then flying to wherever the ship happened to be in port. Weeks would be spent aboard the platform loading the latest version of software, testing it out to ensure both that the new program functioned properly, and that this newest install did not break any of the systems it

connected to; and then going through a comprehensive turnover with the Ships Company for final acceptance. This process was both time and labor intensive, for the software developers and the sailors. The US Navy has a limited civilian workforce, thus the time spent by these engineers and scientists traveling to install software came at the expense of new software development, since often the same individuals were tasked with both development and installation activities.

Comparing such practices to how software is installed in the modern world demonstrates a stark contrast. The average user of a computer or a smartphone is a novice with little understanding of software development, yet they are able to effectively install and update software programs without expert help. The ability to receive, install, and even roll back to previous versions of software, without the help of an expert, is a goal that our Navy software development community is striving to achieve. Today, the U.S. Navy's program offices are striving to emulate this commercial example by pursuing development and installation processes that are easy, intuitive, and failsafe.

The current generation of software users require rapid delivery of capability to the warfighter in a flexible, time-rich environment, and their demands have resulted in the maturation of the way software is developed and delivered. Waterfall development has given way to newer, more flexible methodologies, such as adopting practices related to iterative and spiral models, Agile, and DevSecOps. Newer design and development patterns have changed from monolithic "big bang" applications to smaller micro-service architectures. In military applications, where limited bandwidth is often the norm, containerization and smaller footprint applications have further enabled delivery to the warfighter. The U.S. Navy has adopted a paradigm of a "software platform" to serve as a foundation for shipboard applications. As Rear Admiral John W. Ailes, U.S. Navy (Retired, and Susan LaShomb describe in their 2021 *Proceedings* article, "a software platform includes the development environment used to create an application, along with the 'stack' of operation system, middleware, and virtual machine on which it runs."[1]

All of these changes are moving the U.S. DoD towards it's vision to "Deliver Resilient Software Capability at the Speed of Relevance."[2]

This paper focuses on the creation of the DevSecOps environment that provides Command and Control (C2) capability to ships and submarines across the US Navy, the Overmatch Software Armory (OSA). The Navy must be able to rapidly react to mission needs at sea in bandwidth-challenged environments. Such demands have taken advantage of the aforementioned advancements in software development and delivery. In fact, over seventy (70) "Over the Air" (OTA) installs occurred in 2023, requiring absolutely no portable electronic storage at all.

1.1 Owning the problems

In 2006, Naval Information Warfare Center (NIWC) Pacific, at the time known as Space and Naval Warfare Systems Center Pacific, developed a repository-type approach to support spiral type software development. An important element of this approach was to drive the US Navy toward owning data rights for the software it was contracting for. Recurring issues with Command and Control (C2) software systems onboard ships triggered the need for US Navy software engineers to be able to review and examine the code base for software to try and identify the root causes of the observed issues. Industry partners, rightfully sensitive to intellectual property rights, often challenged requests for access to the source code. Subsequently, contractual language and technical requirements were strengthened, making clear to industry partners that software developed at the cost of the Navy would not only be Navy-owned, but also delivered to a government-managed repository. As a result, Navy software engineers were more quickly able to troubleshoot long standing issues, allowing for faster remediation that improved both the software itself and the overall reliability of ships' systems. This repository stayed up and running for the next twelve years, funded by local software development projects, and over time it became more robust. As software development methods evolved, it added additional tools and capability. For example, it incorporated Atlassian capabilities, to include JIRA and Confluence as a collaboration and planning suite, Jenkins as a build tool, and various other software testing tools to determine vulnerabilities and Information Assurance (IA) issues. Installers were able to access this virtual repository, create installation media, and then proceed to the ship for install. In design, it served as a very early DevOps-like environment, but still without the inherent advantages to an over-the-air (OTA) installation.

1.2 It's getting cloudy outside!

In 2017, the US Navy was embracing the move away from onsite (and expensive) data centers. "Data center consolidation" became a popular initiative to help potentially save the Navy large amounts of money by shifting much of its data storage and infrastructure towards the use of Commercial Cloud and Commercial Cloud Providers (CCPs). Scale, reliability and even perceived security advantages associated with use of

CCPs led senior leaders to embrace their adoption.[3] The advent of the Navy Research and Development Establishment (NR&DE) Commercial Cloud environment was in fact a first step towards providing the Navy with its first OTA capability.

Commercial Cloud providers go through a rigorous accreditation process before being allowed to offer their services to Federal customers. Once accredited, senior leaders were highly motivated to begin the process of system and data migration to the cloud. They soon learned, though, that the desired benefits of cloud were easily missed if systems were not ready for migration. True, commercial cloud compute and storage can be inexpensive compared to providing like-capability on premise; however those savings are only fully realized if systems and data are "cloud-native". Many program managers, anxious to realize advertised cost savings, did not go through the considerable effort to make their systems more cloud-native in architecture prior to migration. Such "lift and shift" endeavors caused many programs to not realize any cost savings, at least not initially. If fact, simply making a copy of their system and dumping it into the cloud, in many cases made their compute and storage bills even more expensive. Other program leaders hit a different pitfall when they selected (and paid for) commercial cloud compute offerings that provided much more services than what their project required, which caused huge spikes in cloud expenses. Over time, however, some programs were able to adopt cloud-native practices and right-size their data usage within the cloud, and thereby realizing the advantages offered by the commercial cloud environment.

Since the NR&DE Cloud environment was developmental in nature, it was built upon the Defense Research and Engineering Network (DREN), which provided transport off-premise. The NR&DE allowed projects to realize the necessary security and advantages of being able to conduct integration and testing of developmental products, while reducing costs of maintaining their own hardware and infrastructure. Each Naval Warfare Center utilizing the NR&DE was able to connect and procure commercial cloud services to fit their use case. This approach allowed everyone to conduct these system activities in a similar manner. The NR&DE also provided the necessary Information Assurance (IA) to be covered by use of the environment via the DREN, and leveraged the accreditation of the commercial cloud providers. This resulted in countless hours saved by project staff not having to obtain a separate Authority to Operate (ATO) accreditation for their work.

1.3 The RITE beginning

Around the same time projects were looking to transition into the commercial cloud, there was considerable talk across the Navy about adopting Development/Security/Operations (DevSecOps) practices, with the goal of enabling Continuous Integration/Continuous Delivery (CI/CD) to its systems. This new development concept would enable programs to deliver software to the customer much faster, while still addressing policy & security considerations. The first of these DevSecOps-type environments was the Rapid Integration and Test Environment (RITE), which provided a software pipeline, complete with tailored tooling to support Static and Dynamic Application Security Testing (SAST/DAST) and software quality testing. Once compiled, software would be sent to a downstream repository for manual integration-type testing prior to release.[4]

For several years this pipeline served a wide variety of C2 software customers. End-state delivery, however, was not yet a feature of this software factory still in its infancy. Technologists were still required to build software media (tapes and CD's) from within the directory once it was approved for release. While state-of-the-art at the time, the major issue with this environment was its inability to connect and deploy software to ships at sea or pier side. Installation teams were still required to spend significant amounts of time onboard ships, working long hours in confined spaces, loading and troubleshooting complex software and hardware issues in order to complete a shipboard operational check.

2 A modern software factory is born

In 2018, NIWC Pacific (known as SPAWAR Systems Center Pacific at the time) invested internal funding to build off the success of RITE and the commercial cloud environment to the Collaborative Software Armory (CSA). This environment was a cloud-native commercial environment utilizing modern tooling and commercial cloud enabled scalability. The lofty goal of CSA was to provide the first dedicated, secure development environment to rapidly build, test, accredit, and deploy applications to the fleet and warfighter. The work began in June of 2018, and was completed and operational in January of 2019. Figure 1 shows the vision of CSA. (As Figure 1 shows, CSA has been absorbed into the larger Overmatch Software Armory; this will be addressed in a later section.)

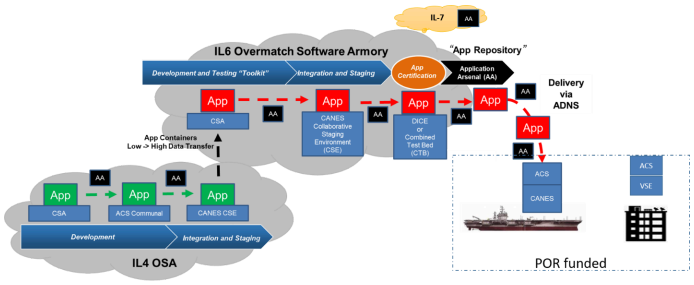

Figure 1: Overmatch Software Factory

Programs utilizing the older pipeline immediately began migration to the new CSA environment. CSA empowered program offices to begin controlling their individual software baselines within a Government owned and managed software factory, a factory designed to provide software to the tactical edge. Project cost savings were achieved thanks to the common use of software tools, configurations, security overlays, and basic development practices.

2.1 Application Arsenal

OSA, and CSA before it, relies on Application Arsenal (AA in Figure 1) for transition software within the pipeline. It is heavily leveraged for receipt and for publishing of applications both throughout the Dev side of the environment but on the Ops side as well. Applications are not considered "ready" until they become visible within the Application Arsenal. Once there, they are understood to be ready to be used for that next level. For example, in the development environment, AA is used to transmit a containerized set of code in the Impact level 4 (IL4) development environment into the Agile Core Services (ACS) Communal environment for testing, then into the CANES Collaborative Staging Environment (CSE) for integration.

Application Arsenal (AA) also provides the "digital environment where approved applications and application updates will be stored and ready for rapid deployment to the fleet."[5] As Figure 1 shows, AA is used to transmit between different aspects of the development pipeline, but also leverages ADNS to transmit the applications to the afloat environment. Similar to a commercial application store like Google Play or the Apple iStore, AA provides an environment where sailors and other users can go to retrieve applications that fit their needs. The sailor on a ship can access AA and download new or newly updated command and control applications in an "on-demand" format. This enables truly automated delivery of capability to the ship, without any additional digital media!

2.2 ACS Communal and CANES CSE

As CSA matured, additional capabilities were added and developed to create a full software pipeline focused on delivery to the shipboard CANES environment. Agile Core Services is a service-oriented architecture for CANEs, including the Navy's Tactical Analytics Framework. Digital Twins of applications, the PaaS, and other dependent items are all kept in a continuous updated cloud environment. This provides ease of configuration management, as well as a running archive of previous builds in the event that roll-backs are required. This environment enabled a whole new level of testing and integration prior to deployment on a shipboard network.[6]

2.3 Rapid Assess and Incorporate Software Engineering (RAISE)

A critical element to securely delivering software via this Navy software factory is based on the Rapid Assess and Incorporate Software Engineering (RAISE) concept. The RAISE concept is a key component of the Navy's Cyber Ready program, as well as enabling speedy delivery of software.[7] This process relies on an Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) inheritance model, where the IaaS is responsible for the majority of the security requirements and associated Authority to Operate (ATO) for both the PaaS and applications to leverage.

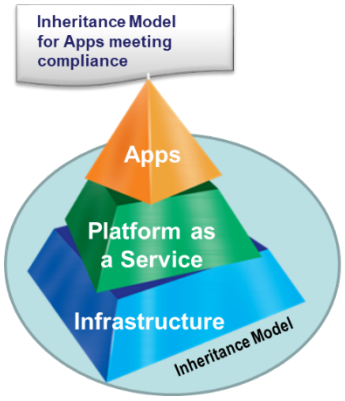

Figure 2: RAISE Inheritence Model

Each level of the pyramid carries certain RMF control requirements, with the Infrastructure layer (IaaS) possessing the majority, the PaaS carrying another portion, and the Apps any remaining controls. Under this model, software that is developed within the IaaS and PaaS enviornment in the software factory inherits the

controls fof those environments. This inheritance enables the software to meet the controls that those environments provide. The software developer is left with a much smaller set of application level controls that they must address individually. This enables the software pipeline executing the various SAST/DAST compliance checks to cover these controls, assess the security posture of the latest software build, and enable rapid deployment to the end state platform requiring the new software delivery. This ability has revolutionized the software development and delivery approach that enable rapid deployment of software to the tactical edge, and still meet the necessary security requirements to keep Navy systems and software compliant.

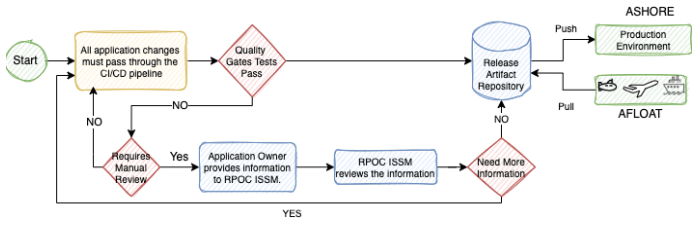

Figure 3: RAISE 2.0 Process for Release

The RAISE 2.0 process requires each application to undergo a series of Gate Tests that must be satisfied prior to release into the repository for production as depicted in the Figure 3 above. There are eight (8) current gate tests as shown in Figure 4 below. Those tests consist of a rigorous suite of tests to include SAST, DAST, SBOM, and classified words testing. Once testing is completed successfully, the RAISE Platform of Choice (RPOC) releases the software for the repository available at that time to end state customers.

ID	Description
GATE 1	must provide Static Application Security Testing (SAST)
GATE 2	must provide Dependency Scanning or Software Bill of Materials (SBOM)
GATE 3	must provide Secret Detection
GATE 4	must provide Container Security Scanning (CSS)
GATE 5	must provide Dynamic Application Security Testing (DAST)
GATE 6	must provide a manual step to allow the RPOC ISSM to review
GATE 7	must sign the release image
GATE 8	must store the release image in an artifact repository

Figure 4: RAISE 2.0 Testing Gates

Findings during the gate testing are collected and must be mitigated within 21 days. All findings and vulnerabilities of an application from these test must be mitigated with a residual risk not exceeding Low/Moderate prior to approval for release.

2.4 Compile to Combat in 24 hours; You Built It, Now Prove It!

Once the CSA software factory hit the streets in January 2019, there were several test events scheduled to test and demonstrate its capabilities. The Compile to Combat in 24 hours (C2C24) framework seen in Figure 5 was highly reliant upon the Collaborative Software Armory (CSA).

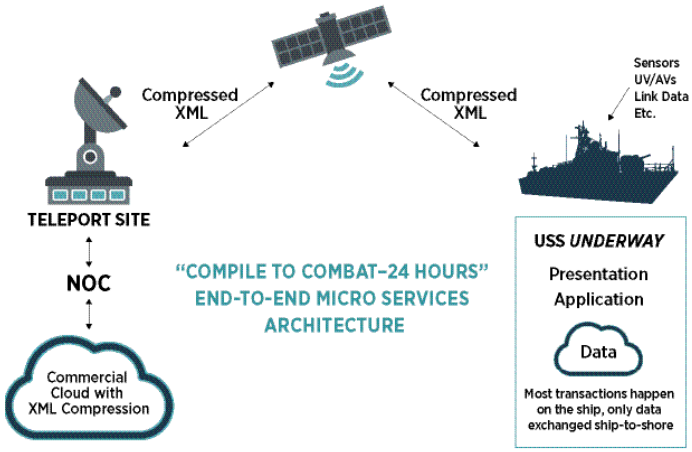

Figure 5: C2C24 Architecture[8]

The very first test involved sending a lightweight, fully containerized app from a Naval Operations Center (NOC) across the NIPRnet to a US Navy ship while it was pier side. For the test to be considered a success, the application had to function correctly once it was delivered to the ship by this new methodology. The test event was completed, the test results verified, and in a US Naval milestone, the software application installed correctly! This humble start proved that Over the Air (OTA) installations aboard a ship were possible and was a giant leap forward towards decreasing or even eliminating the requirement to physically send dedicated installation teams to execute an application install.[9]

2.5 Big Ol' Apps

Currently, Navy software comes in many languages, complexities, and sizes. For many years, software size was determined by Source Lines of Code (SLOC) to determine how large or complex an application would be. The more complex an application, the more lines of code were required. As time has evolved in the computer science community though, many applications have become smaller, requiring less lines of code. Furthermore, based on the Inheritance Model above, applications are not required to bring services already provided by the PaaS or IaaS, which further reduces their size.

Many current Naval applications have begun greenfield efforts to meet this very model, and move away from being monolithic, heavy client-type applications. The more services an application brings itself, the more security issues they open themselves up for during development and deployment. Smaller applications can be more rapidly updated, or even replaced with even newer versions, helping with modularity, maintainability, and sustainability. Such advantages are easily traced to the creation and utilization of a dedicated software factory.

Moving from "Big Ol' Apps" to smaller, modular applications requires a shift in both mindset and practice for government and industry software developers. Often C2 software applications have been procured and produced as large, standalone applications. These applications followed waterfall development, where they provided either 100% capability or none. The move towards smaller modular applications also enables the rapid fielding of C2 software that can easily integrate and provide 70% or 80% of the capability quickly, and then easily iterate to provide the remaining capability in the future. The ability to provide a 70% solution in the face of great need, with low risk of breaking the entire enviornment is a key enabler of modern C2. As Rear Admiral John W. Ailes, (U.S. Navy, Retired) and Susan LaShomb point out in their Proceedings article, "the rapid *fielding* of new technology is key to outpacing U.S. adversaries" (emphasis in the original).[10] These capabilities can be brought to full strength through future spiral development, and will also give "the fleet and opportunity to refine requirements and determine which attributes of a new system are most important and which areas need improvement."[11]

2.6 Overmatch Software Factory

The success of the Collaborative Software Armory led to it being selected by the Navy's Overmatch Effort and renamed the Overmatch Software Armory.[12] The Overmatch Software Armory "is comprised of OSA Tools, Agile Core Services, multiple Collaborative Staging Environments, and the Application Arsenal.[13] As shown in Figure 6, the Overmatch Software Armory has matured to provide a complete pipeline for software development, testing, integration, certification, and deployment.

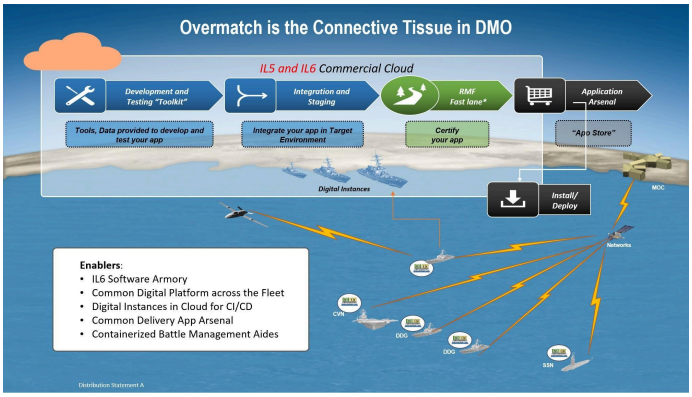

Figure 6: Overmatch Software Factory

Within OSA, software delivery timelines in most cases have shrunk from being measured in months, to days. The visual below demonstrates some of the remarkable accomplishments since the first OTA delivery thirteen (13) months ago. Up to that point, as previously mentioned heavy reliance was on engineers with boots on deck to all Navy platforms providing an onboard software delivery installation, groom and test. This process kept the application installation team tied down to a single platform for up to 2 weeks. The metrics above identifies that to date, we have been able to forego sending engineers shipboard for two plus weeks at a time to a shipboard platform install. Installs are done via Application Arsenal (AA) over the air; this reduces the time spent by an engineer to install the software on the ship from two weeks down to two to three days. The reliance of navy engineers to be available for these installs is basically a comfortability issue where shipboard personnel are still learning the new technologies to support being able to install, update, or rollback any software available via the stack.

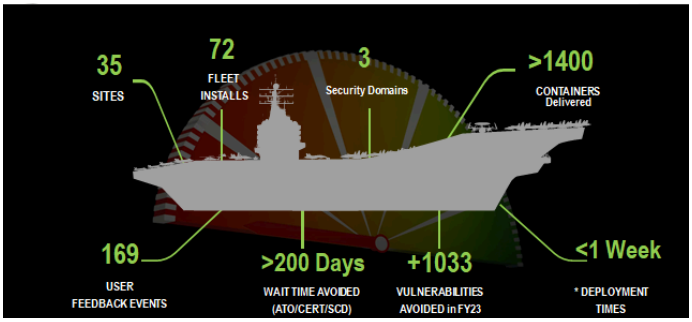

Figure 7: Overmatch Software Armory Metrics

Figure 7 defines the current success that the stack has provided to the Navy since January 2023. Each of these metrics represents a different aspect of the software factory and the outcomes it enables.

2.6.1 Sites

The Sites metric refers to the distinct operational locations (sites) that software has been deployed. This includes individual ships as well as shore sites, like the Naval Operations Centers.

2.6.2 Fleet Installs

The Fleet Installs metric refers to the amount of software programs that have been installed. This includes both completelly new software, as well as updates of software previously installed.

2.6.3 Security Domains

The Security Domains metric refers to the different types of security environments that software has been deployed to via OSA.

2.6.4 Containers Delivered

The Containers Delivered metric refers to the total number of containers that have passed through the deployment process. The utilization of containers in building modular software is a key enabler for doing targeted software updates and deployment over the air in the Navy's often band-width limited environment. Each of the Fleet Installs counted includes one or more containers deployed. This metric provides a more granular assessment of the level of effort that the software designers have put into the software being deployed, and may act as a proxy for the value in new capability and updates delivered to the ship.

2.6.5 User Feedback Events

The User Feedback Events metric refers to the opportunities that OSA has had to collect user feedback. User feedback is a key tenet of agile software development, so increasing this number to provide value through continuous integration/continuous development (CI/CD) is crucial to the Navy.

2.6.6 Wait Time Avoided

The Wait Time Avoided metric refers to the amount of time that software developers often have to spend waiting while the software is reviewed in the accrediation process. This is the amount of time the software is sitting, without receiving improvements, as various Authorities to Operate (ATO) or other accreditions are granted. The reduction in wait time is due to the usage of the RAISE process and the ability for software to inherit the majority of the controls from Iaas and PaaS.

2.6.7 Vulnerabilities Avoided

The Vunerabilities Avoided metric refers to the distinct software vulnerabilities that have been avoided due to the rigorous testing utilizing the tools inherent in the software armory. This is a way to measure the increase in cyber security that OSA provides.

2.6.8 Deployment Times

The Deployment Times metric refers to the total time it

takes for software to move through the Overmatch Software Armory. There is variation in each software's deployment time as different software will have different statistics based on its compile time, vulnerabilities, and other unique characteristics. The less than a week deployment time for OSA indicates the streamlined and agile process that software developers are utilizing.

2.7 The first official over the air install shipboard!

In January 2023, the US Navy was able to conduct its first operational over the air (OTA) install. Just like the operational test, this represented a monumental accomplishment. Over a 72-hour period, software was successfully transmitted over the dedicated network from the shore-based repository to a ship floating pier side in San Diego.[14] NIWC Pacific engineers and scientists were shipboard to document and assist with the numerous technical challenges encountered. Many lessons were learned through this first evolution. Some of the larger containers had issues with timeouts, requiring the engineers to subsequently find solutions. Queue adjustments and container sizing were adjusted and applied. On subsequent installs, the solutions applied resulted in software delivery to be successfully transmitted in just a few hours. Since January 2023, more than seventy (70) C2 software applications installed successfully, demonstrating a powerful new technical reality for Sailors and the ships they operate.

3 Managing the overmatch software armory (OSA)

Managing an enterprise-scaled service designed to provide capability to the Nation's fleet is quite the undertaking. It requires seasoned engineers who are knowledgeable on the latest industry software development tools, as well as what constitutes an efficient software development pipeline. They also must be able to react to individual project issues that arise when things don't work as expected. This effort is led by some of the finest Navy civilian software professionals in existence, and the Navy relies on equally capable industry partners that are well-versed both in software development and with specific software development tools. Finding such talent, both within the Government or through Industry, is a significant challenge in and of itself, especially given the all-too-common problem of finding sponsors to cover the cost of maintaining a Naval software factory.

Financing enterprise software factories presents multiple challenges. Enterprise licenses and the labor to actually maintain a continual capability are not cheap. While there are single time "up front costs" from the architecture, there are also significant continuing costs for "just keeping the lights on" and maintaining the environment. Many Navy Commands are not keen or able to invest large sums of money into "infrastructure IT plumbing" such as a software factory. Factories therefore, often have to be self-sustaining, charging the factory customers based on how much they actually use the factory tools. The charges to the factory customers must take into consideration such issues as the costs for new testing tools, making the architecture more efficient, and sometimes shifting to cloud-native products vice third party vendor products. Including these innovation and planning costs into the rates is essential, but it also poses a significant challenge every year while budgeting.

The approach of "everyone develops to the same pipeline configurations" can also present contractual issues when companies are required to develop products inside a Government owned/controlled environment. In some cases, there have been considerations made for contractors to initially develop within their own environment. Later in the development process, however, the code must migrate and compile within the prescribed factory. This requires contracting officers to include appropriate clauses spelling out the dedicated process of how the Government will receive and accept software in accordance with the contract.

The architecture associated with an enterprise software factory relies heavily on the underlying transport backbone of how the factory is built within the commercial cloud, and how developers are able to connect to the factory. Network components such as routers, switches, gateways, and comm links, can each cause connection issues at any given time. In short, a factory is often only as good as the network it resides upon. When network problems occur, end users sometimes mistakenly blame the factory itself as being broken, undermining their confidence in the software factory concept. Efforts are underway, therefore, to improve the reliability of the underlying network in order to reduce or even eliminate future outages preventing access to the factory itself.

4 lessons learned and ongoing challenges

While the Overmatch Software Armory is up and running, delivering C2 capability to the fleet, there are still a number of challenges for the NIWC Pacific community and the U.S. Navy. The lessons learned and ongoing challenges laid out in this section of the paper need to be addressed by the entire C2 community. While some of the challenges are structural, based on the way the U.S. Navy funds its C2 infrastructure, many of them are also cultural. The United States Navy and partner nations are

strong innovators. The need to have modular and configurable software based solutions that enable command and control both within and between national forces has only increased in the last decade. This need should drive us to continue to vigorously implement the lessons learned while fiercely tackling the ongoing challenges.

4.1 Technical Software Debt

The Navy has a number of older, monolithic software programs that have not yet gone through the modernization processes needed to make them agile and containerized. The programs are facing a choice now: do they invest in a re-development effort to become containerized, lightweight, intuitive to use, and RAISE focused, or do they continue with business as usual and focus on work arounds? The crux of the choice comes down to whether they can garner the resourcing support and identify a path forward for modernization. As this paper has shown, the cultural barriers and the lack of understanding of the true need for and true cost of agile software development and employment makes this question much more complex than it would seem at first inspection.

4.2 DIY Culture

While greenfield software development may be the easiest to work with in a mature software factory, it's also the most prone to utilizing a "Do It Yourself (DIY)" approach to a software factory. For a software developer without access to a mature software factory, the temptation is to start building and testing code with the resources they have. Then as time goes on, and budgets offer small amounts of funding, the software developer acquires resources and tools, eventually creating their own bespoke software factory-like tool set and process. These software developers, both individuals and small teams, may look at the larger software factories and feel that they don't need all the capabilities provided in the package that they would need to pay for. Thus, the innovation mindset that underpins U.S. Navy software development can also lead individual software developers and small teams to focus "DIY" approaches, rather than understanding the benefits of utilizing a software factory at scale. This culture must first be understood, then the needs of these developers must be addressed to ensure that they are able to balance innovation with the need to utilize the cyber security, testing, and collaborative staging environments that will become crucial as the software matures for deployment.

4.3 Economies of Scale

The U.S. Navy's resourcing model for Software Factories continues to be a challenge. There is currently no program office providing steady resourcing for the development of one or more software factories. Software factories have to split the costs among the set of current customers, creating sometimes large charges that make it very difficult for small projects to utilize the capabilities. In order for software factories to be financially viable they must achieve economies of scale where they have sufficient users to share the burden of cost. In the last couple of years, this problem has been recognized, and a variety of solutions have been piloted, with varying degrees of success. This has been further exacerbated by the Navy's lack of agility when it comes to purchasing software subscriptions. The Navy's acquisition community is still wrestling with the conundrum of how to agilely contract for a capability where the demand is not known up front (software licenses) in a manner that does not create waste. Again, in the last couple of years, this problem has been surfaced, but has not yet been fully solved.

4.4 A Plethora of Software Factories

There is a unique tension between the need to have sufficient numbers of tailored software factories to address inherently different production and operational environments, and the need to keep the number of software factories small so that they can achieve economies of scale. The production or operational environment that C2 software will deploy to is varied. Some C2 software will deploy to temporary forward operating bases (as in the case of planning software), while others will deploy aboard afloat on CANES; still others will deploy in shore-based cloud production environments. This variety in final production environment is driven entirely by where the US Navy needs to operate it's C2 software. The variety in operational environment ensures that a one-sized fits all software factory is not the answer. The Navy probably needs more than one software factory, but the actual number and when it makes sense to build anew or modify an existing environment is still a question the community is addressing. Fortunately, there has been a significant cultural shift lately within the Navy's Software Factory Community towards increased transparency and even self-organization in specialized areas.

4.5 Measuring Success

The current metrics that the Overmatch Software Armory uses are based on the results that software developers can achieve by utilizing the factory. These metrics, however, do not directly measure the effectiveness of the

software factory itself. The identification and definition of metrics for the usefulness of a software factory itself is an ongoing challenge. It is difficult to separate the usefulness of the factory itself from a number of confounding factors like: software developer skill level, usage of tools within the factory, the initial state of the code (legacy vs. new), and team resourcing. Variation in any of these confounding factors will likely change the success of the code development (and metrics) within the factory. For example, if a software developer is entirely unfamiliar with one of the tools, the first time they use the software factory they will likely have to spend some time training themselves on the tool. This additional time could significantly slow the deployment speed of the software, and if the developer is ineffective, it could also reduce overall software security. Thus, a number of other metrics might be useful to help understand and measure the actual usability of the software factory itself. This is an ongoing area that these authors are continuing to study.

4.6 Burden Sharing and Training

One common technical and cultural problem is determining which activities are the responsibility of the software factory, and which are the responsibility of the software development team. At first glance, the division of responsibility may seem easy: the software factory is responsible for providing the tools and environment, while the software development team is responsible for using the tools. In practice, these lines often get blurred. Software developers need to have the skills to use the tools provided. When a software factory switches out a tool in it's suite, then it transfers additional burden onto the development teams. Who pays for the additional training that maybe inherent in utilizing a new tool? In addition to this conundrum, there is often great variation between the skill sets of various teams. Some teams utilizing the OSA are experts with the tools, platform, their code, and are effectively resourced; these teams need little additional support and are able to fully utilize the factory. Other teams utilizing the OSA have differing skill sets within their teams and may lack knowledge of one or more aspects of the platform. Particularly when the platform was less mature, it was more common for it to be utilized by software developers who had never used a common software factory. While there continues to be a healthy dialogue between the OSA and its users regarding training on tooling and new capabilities, some of these challenges have been solved. NIWC Pacific offers a "Concierge Team" that enables projects that are lacking basic skill sets, or are using the factory for the first time, to onboard and get started with OSA. This team provides a "concierge-like" service, identifying the software developer teams' needs and filling in these gaps. This Concierge Team has been designed specifically to address the common problems that novice software developer teams often encounter when onboarding into a software factory for the first time. If further support is needed, OSA also offers quarterly training days, and makes connections between the software developer teams and the tool providers so the team can schedule additional training. Finally, if a software development team finds that they are in need of specialized software developer resources, NIWC Pacific has a Magic Team that provides seasoned software developers to help bridge gaps. The Magic Team is not designed to be a permanent augmentation of the software development team, but can bridge any gaps that may appear for a finite length of time. Offering these additional services, at an additional cost, has helped the novice software developer teams to be able to fully utilize the software factory, even while they are discovering the gaps in their own experience and training.

4.7 Trust the Process

A culture of low trust in the benefits of the software factory, rapid and agile software deployment, and sailor capable installs can undermine the entire process. The utilization of an enterprise environment often goes against a software development team's innovative culture, their DIY focus, and is outside their experience. While the U.S. Navy is learning to effectively and efficiently provide software platforms "stacks" it is also learning to effectively utilize them! There are many things that are happening in the IaaS and PaaS process that contribute to cyber security, controls that the cyber security expert on the team may be used to inspecting in a manual manner. Moving to a continuous accreditation environment requires a new skill set in utilizing the tools and a new trust of the outputs. Increasing trust in the process is a slow moving, but essential step.

The end users of the software, often sailors on ships, need to learn to trust the process of downloading a new software update or application on their ship, without the support of a team of engineers. They are naturally risk adverse as the consequences are large in the shipboard environment. Circling back to the analogy of a cell phone user, the consequences of downloading an application or update that causes your phone to do an unexpected restart are not high; if we have to restart our cellphones, we stand to lose between 1-5 minutes of our time. If a sailor on a ship downloads an application or update that causes CANES to shutdown and restart, the consequences could be catastrophic. These sailors, moreover, have had no experience or training with the

software development process and so do not always know the rigorous testing that applications go through with digital twins in the collaborative staging environments. Thus, asking the sailor to trust the download of an application to a ship, without a team of engineers standing by requires a significant amount of trust.

The U.S. Navy and NIWC Pacific are working on increasing trust in the process through a combination of proving success and providing high level insights into the process of developing software for over the air installs. The goal is for this to be taken as the norm, but today we are still far from that state.

5 SUMMARY

The path to creating the Navy's first over the air software deployment capability was a long one, and required determination and ingenuity. Lessons were learned on each step of that journey, and each milestone achieved served as a stepping stone toward the next elevated capability. The journey is far from complete, and in the coming months and years we look forward to being able to deploy software to ships on-the-move in any place and at any time. NIWC Pacific is now looking towards the future when the Navy personnel aboard a ship will feel confident, and have the tools they need, to do an application installation or update on their own, without a NIWC engineer supporting them. Today's dynamic and evolving warfighting environment demands that we keep pressing forward, building on yesterday's successes toward tomorrow's victory.

REFERENCES

[1] Ailes, John W. (Rear Admiral U.S. Navy, Retired) and LaShomb Susan. "The Software Payload is the Platform." *U.S. Naval Institute Proceedings*. May 2021. https://www.usni.org/magazines/proceedings/2021/may/software-payload-platform

[2] Department of Defense Software Modernization Strategy. *Department of Defense*. 2022. https://media.defense.gov/2022/Feb/03/2002932833/-1/-1/1/DEPARTMENT-OF-DEFENSE-SOFTWARE-MODERNIZATION-STRATEGY.PDF

[3] "SPAWAR Shares Vision for the Information Warfare Platfrom with Industry." *U.S. Navy Press Office*. 17 July 2017.

[4] Galdorisi, George (Capt. U.S. Navy, Retired), George, Amanda, and Morris, Michael. "Finding the 'RITE' Acquisition Environment for Navy C2 Software." *Proceedings of the Twelfth Annual Acquisition Research Symposium*. Naval Post Graduate School. 30 April 2015. https://apps.dtic.mil/sti/tr/pdf/ADA623193.pdf

[5] Gamboa, Elisha. "NAVWAR Deploys the Navy's First Application Arsenal." *Naval Information Warfare Center Systems Command (NAVWAR) Public Affairs*. 19 August 2021. https://www.navy.mil/Press-Office/News-Stories/Article/2735561/navwar-deploys-the-navys-first-application-arsenal/

[6] PMW 160 Tactical Networks Program Office. *PEO C4I*. https://www.peoc4i.navy.mil/Portals/98/Documents/Tear-Sheets/2023_PMW 160_TearSheet.pdf?ver=NQQlLrVs5P9y46gw81nHJA%3d%3d

[7] "Early Cyber Ready Success: RAISE 2.0 and Overmatch Software Armory." *Department of Navy Chief Information Office*. 28 July 2023. https://www.doncio.navy.mil/ContentView.aspx?ID=16441

[8] Barrett, Danelle (Rear Admiral U.S. Navy) "C2C24 Transforms Navy Operations. U.S. Naval Institute Proceedings. August 2018. https://www.usni.org/magazines/proceedings/2018/august/c2c24-transforms-navy-operations

[9] McDermott, Kara. "SPAWAR Supports Navy's Digital Transformation with 'Compile to combat in 24 Hours' Training Series" Department of Navy Chief Information Office. April-June 2019. https://www.doncio.navy.mil/mobile/ContentView.aspx?ID=12451&TypeID=21

[10] Ailes, John W. (Rear Admiral U.S. Navy, Retired) and LaShomb Susan. "The Software Payload is the Platform." *U.S. Naval Institute Proceedings*. May 2021. https://www.usni.org/magazines/proceedings/2021/may/software-payload-platform

[11] Ailes, John W. (Rear Admiral U.S. Navy, Retired) and LaShomb Susan. "The Software Payload is the Platform." *U.S. Naval Institute Proceedings*. May 2021. https://www.usni.org/magazines/proceedings/2021/may/software-payload-platform

[12] "Navy Development Security Operations (DEVSECOPS) Guidance." *Department of the Navy*. https://www.navy.mil/Resources/NAVADMINs/Message/Article/2460115/navy-development-security-operations-devsecops-guidance/

[13] NIWC Pacific Overmatch Sofware Armory. *Naval Information Warfare Center Pacific*. https://www.niwcpacific.navy.mil/Technology/Overmatch-Software-Armory/

[14] Gast, David W. (Capt. U.S. Navy) and Baptiste, Philip. "Maritime Tactical Command and Control program advances with containerization technology." *CHIPS*. 15 March 2024.

https://www.doncio.navy.mil/CHIPS/ArticleDetails.aspx?ID=16689