SYM-AM-25-301

Excerpt from the Proceedings

of the

Twenty-Second Annual Acquisition Research Symposium and Innovation Summit

Wednesday, May 7, 2025 Sessions
Volume I

Sustained Innovation Through Composable Systems

Published: May 5, 2025

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

The research presented in this report was supported by the Acquisition Research Program at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF DEFENSE MANAGEMENT
NAVAL POSTGRADUATE SCHOOL

Sustained Innovation Through Composable Systems

Hon. Nickolas H. Guertin, PE—performs consulting and is also part-time faculty at Virginia Tech's National Security Institute as a Senior Research Fellow. He is on the board of Southern New England Defense Industrial Association. He most recently served as an Assistant Secretary of the Navy for Research, Development and Acquisition (RDA), to include sustainment and oversight of the contracting community for total of $130 billion. Prior to that role, Mr. Guertin served as the Director, Operational Test and Evaluation, a senior advisor to the Secretary of Defense, reporting independently to Congress on Department of Defense weapon systems. Mr. Guertin has an extensive four-decade combined military and civilian career in submarine operations; ship construction and maintenance; development and testing of weapons, sensors, combat management products including the improvement of systems engineering; and defense acquisition. Prior to his confirmation positions, he was performing applied research for government and academia at Carnegie Mellon University's Software Engineering Institute. He received a Bachelor of Science in Mechanical Engineering from the University of Washington and an MBA from Bryant University. He is a retired Navy Reserve Engineering Duty Officer and is a licensed Professional Engineer (Mechanical). [nickolashg@vt.edu / nickolas.guertin@transformus.us]

CAPT Gordon Hunt USN (Ret.)—is a Naval Reserve Engineering Duty Officer recently retired as a naval combat system software solutions and design principles. He is the co-founder and vice president of the company Skayl focusing on system of systems integration and semantic data architectures, which enable increased systems flexibility, interoperability, and cyber security. Hunt's experience in building real systems with current and emerging infrastructure technologies is extensive. His technical expertise spans embedded systems, distributed real-time systems, data modeling and data science, system architecture, and robotics & controls. He is a recognized expert in industry on Open Architecture and data-centric systems and as a regular author and presenter, he speaks frequently on modern combat-system and command and control architectures. As a CAPT Engineering Duty Officer in the U.S. Navy, he supported combat system development and system integration scalability challenges. Hunt earned his BS in Aeronautical Engineering from Purdue University and his MS in Aerospace Engineering & Robotics from Stanford University. [gordon@skayl.com]

Robert Matthews—is a Technical Fellow and Advanced Concepts Engineer at L3Harris Technologies, advocating for and architecting MOSA developments for critical pursuits. Bob has a Bachelor of Science in Aerospace Engineering from the University of Maryland and earned his MBA at Florida Tech. He joined the U.S. Army Reserves and served 8 years as a CH-47 Chinook Helicopter Mechanic and Flight Engineer. Before joining L3Harris, he enjoyed a 20-year career as a civil servant with the Naval Air Systems Command. While at NAVAIR, Bob led the Avionics Architecture Team that coordinated with Army Aviation and The Open Group to found the FACE Consortium. Bob was elected the inaugural FACE Steering Committee Chair, serving in that role for 5 years. During that time, Bob and his team also initiated the HOST standard and collaborated with the Air Force, The Open Group, and the FACE Consortium to establish the SOSA Consortium. Bob has received multiple Navy and DoD-level awards for his work on Open Architecture. [bob_matthews@live.com]

Abstract

The enemy gets a vote. Our adversaries seek to outpace us as we seek to win in any clime and place. Winning future conflicts isn't just about innovation, it's about operational excellence and delivering useful innovation to the warfighter faster. The long-standing paradigm of building expensive, highly complex, monolithically integrated weapons systems that take years to plan and upgrade, are extremely vulnerable to asymmetric innovation (Schmidt, 2016). A simple zero-day hack can undo a decade of development and billions of dollars of taxpayer investment, leaving warfighters exposed and our economy irreparably harmed. The Hollywood climaxes of a Jedi against a Death Star or a small cell of rebels injecting a virus into networked alien attacker remain far too plausible an outcome against our inflexible and increasingly networked systems. Commercial technology cycles are outpacing DoD's ability to integrate, giving our opponents the critical time needed to exploit the same technology against us. There is no doubt that emerging technologies like AI, autonomy, quantum computing, and others just entering our imagination, will be critical to overmatch, but only if we can field it first and change it faster. Technology Superiority

only worked as a strategy when seismic innovation was generational and we could ensure disproportionate access. The next conflict won't be decided by a particular technology, but by how quickly it is adapted for military use. We must shift our strategy from technology superiority to implementation superiority. This paper investigates the common pain points that have often impeded programs and proposes a set of acquisition, design, and deployment practices that shift toward composable systems to foster sustained, disruptive, and rapid innovation that outpaces our adversaries despite increasingly egalitarian access to technology.

Recent worldwide activities where American firepower has been put to the test show that the products we have built so far have been equal to the challenges presented in limited engagements, regional conflicts, or by unsophisticated opponents (Bath, 2025) but with increasingly smaller margins. In addition, the Department of Defense (DoD) is continually challenged by delays in capability delivery as well as program cost overruns. Rigidity in the current systems and the patterns of delivering improvements have been responsive to evolving operational engagements under only the most extreme and extraordinary circumstances, not as a matter of course and a reflection of purposeful design. Clearly, we do not lack technological innovation; it is the acquisition process that is broken. The authors have observed a wide variety of design teams that have been hampered by insufficient focus or funding two fundamental aspects of executing any large-scale cyber-physical system: the speed of integration and up-front considerations for future adaptability. Projects are often structured as end-item completion tasks, or a "one and done" approach to design. This is antithetical to an environment where capability is delivered by complex systems that need to be periodically improved as a part of their life cycle (Shenker, 2021). The paper highlights the detrimental impact of tightly coupled, monolithic products that end up being fragile to changing capability requirements and highlights the need to establish a requirements strategy predicated on flexibility and long-term growth. In this context, the full range of acquisition architecture must include approaches and design patterns for future-proofing systems where designs are purpose-built to change over time. A design pattern for continuous improvement (McCarthy et al., 2024).

Keywords: MOSA, Integration, Open Architecture, MBSE, Rapid-Fielding, Frameworks, Rapid, Composable, Flexible, Artificial Intelligence, lock, integrated, weapon systems, implementation superiority, acquisition process, loose coupling, cyber-physical, interoperability, composable systems, key architecture drivers

Introduction

This paper argues for the adoption of composability requirements and a strategy that prioritizes flexible architectures. It does so in the context using newly evolving methods that take advantage of selected standards that have evolved over the past few years to facilitate fluid integration and capability improvement that has long been sought, but few times achieved. Such an approach has long been projected to mitigate the pain points we will discuss, and to deliver innovation faster, more reliably, and at scale (Boydston et al., 2019). The maturation of the development models and supporting environments are now mature enough to, if properly applied, to ensure the rapid deployment of capabilities to warfighters while also maintaining a competitive edge through the ability to swiftly integrate cutting-edge technologies across diverse systems. The focus, therefore, is not merely on the technologies themselves but on architecting two different classes of systems needed for capability transition; the pipeline that creators use to build (how we identify and adapt innovation) and the products that get fielded to the users (how we design for composability and rapidly integrate). Together, these will seamlessly and affordably accommodate future innovations (Hughes & Jackson, 2021).

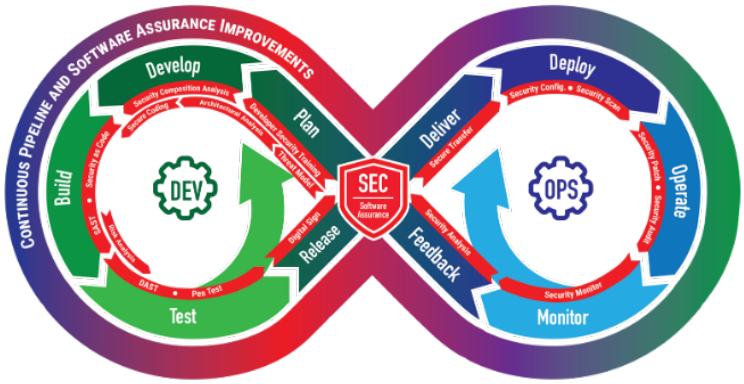

Figure 1: Continuous Evolution of Development and Capability Deployment (Chick et al., 2023)

In 2024, the three secretaries of the armed services jointly signed a memo advocating the use of MOSA (Del Toro et al., 2024). In it, they attest that the DoD Armed Forces face rapidly evolving threats across the world.

> "The dynamic and rapid change of adversary capabilities observed in current conflicts necessitates a critical warfighting capacity to integrate advanced capabilities to counter and maintain a warfighting advantage. To meet this threat, Modular Open Systems Approach (MOSA) shall be implemented and promulgated among the Military Services to facilitate rapid transition and sharing of advanced warfighting capability to keep pace with the dynamic warfighting threat."
>
> Memorandum for Service Acquisition Executives and Program Executive Officers, 2024

While this is good for guidance at the very top, we must defend it and implement it effectively. National Defense Authorization Act legislation has in the past prioritized the foundations of composable systems as critical to winning future conflicts. There is a tension with those in industry who have maintained market share by controlling key aspects of integration for major acquisition program and are profit-motivated to undermine a strategy where integration is engineered to the point where vendor-lock procurement practices can no longer be justified.

The problem is dauntingly complex. The concept is simple, the standards are written, the technology is rapidly maturing, but the work to modernize architectures and adopt cohesive and composable acquisition strategies will be hard. Utilizing the practices discussed here will have an impact to the market is as monumental as Ford's moving assembly line to cars, standard electrical outlets to appliances and the market app stores to smartphones. Innovation will skyrocket for DoD if we can muster the will to unlock it. The greatest challenge will be overcoming the business model environment that establishes long-term vendor lock conditions that makes sources of alternatives and innovation difficult to bring into the solution space for our national security.

Problem Statement

Current DoD tactical edge solutions that are highly interactive (cohesive) software systems and closely tied to physical events (cyber-physical), are not delivering capabilities affordably or fast enough to maintain overmatch (Eckstein, 2024).

Integration issues and latent defects are found and corrected through intense manual intervention, taking years and millions of dollars and repeated many times over a system lifecycle, across most DoD ACAT programs. Over time, the result has become a collection of exquisite, one-off designs that have limited interoperability and are not interchangeable at any level with other systems in the battlespace. We don't have the time or money for that anymore. In order to change this state to a new environment fielding high-quality products fast, the foundational elements of good design must be in place as a linchpin element of success. Going fast, and staying fast, takes upfront work to set the right conditions for success and continue to evolve. Instant success can happen by chance, sustained success takes deliberate design (Flyvbjerg & Gardner, 2023).

Thinking through different aspects of how the portfolio of products came to be, to include mission, procurement, business, intellectual property, technical, data, deployment and improvement, have all evolved in an ad-hoc fashion. At the very core of our acquisition process, we seek to identify capability gaps and derive new material solution requirements independent of any other solution or financial solution. Title 10 authority is delegated to program managers who often share institutional bias to avoid shared risk, common components, or interdependency on another program. These manifest in contracts to a single provider who is inherits that bias to support or create as many economic barriers to disruptive competition as possible. Our current acquisition process is focused on a 50-year-old problem that is no longer relevant and our current product architectures reflect that process. If we truly want a different outcome we must rethink our process, analyze the unintended consequences that resulted, and do the hard work of thinking through in advance a new and adaptable process that is more explicit and creates incentives for composable systems, better manages interdependencies and is more measurable to ensure we produce not just repeatable but positive outcomes. The result is a more flexible acquisition process that ensures we solve today's challenges and adapts itself to address tomorrow's. This will create an enabling environment for sustainable innovation.

Research Focus

Failure analysis, while often painful, is an indiscernible tool to practical engineers. This research draws upon the more than 100 years of collective military, civilian, and industry experience of the authors. This includes in-depth knowledge of hundreds of DoD programs of record and career perspectives from enlisted maintainer to senior officer, GS-7 to Presidential Appointee, and junior software developer to technical fellow. Despite the vastly different experiences between authors, the failure modes discovered are extensively repeated and almost universal across programs. Our investigation resulted in six common program "pain points" that have driven DoD to program ever increasingly expensive, late and brittle systems. While many initiatives and projects espouse the benefits of modular open systems, integrability, portability and many other positive attributes, most fail to demonstrate any measurable benefit at scale because they failed to address the same common pain points. This paper and our research conclusions address how composable systems can address these frequent program pain points:

1. **The "one-and-done" design trap**: Acquisition training is replete with case studies of failed programs and examples of what not to do. The culture in almost every new program office includes a mantra of needing to "do it right the first time." They have a strategy, a list of (most times) overly ambitious key performance parameters and have in the past focused their energy in designing a point solution that does not address capacity for future growth and implementation discovery. But the operational needs of these systems are too complex to ever get it right the first time and because our enemies are never static, neither can be our requirements. Even if they created that perfect solution, the cyber-physical elements would likely be obsolete before operational evaluation (OPEVAL) is complete and the cyber-security

environment is ever-changing. Sadly, many of these programs do not include an upgrade strategy or are able to get those strategies put into their initial budget plans. So even programs that start with MOSA requirements and good architectural design patterns often traded them off to achieve marginal improvements in initial capability.

Instead, every program must start with the premise that the only constant is change and therefore all systems must be cheap and replaceable, or resilient to change through adaptable architectures. Key architecture drivers (KAD) must explicitly address enterprise, domain and system composability requirements, must be measurable and made equal to, or of greater importance, than any initial system KPP values. This will ensure programs prioritize how a system is put together for iterative improvement as opposed to initial capability.

2. **10-pounds of requirements in a 5-pound budget**: While "one-and-done" is principally a schedule-driven acquisition fault mode, the drive to overachieve in initial implementation is a cost driven failure mode. That said, the symptoms are often identical. At some point, almost every development program faces significant cost challenges, forcing program leadership to choose between preparing for the future or achieving as much as can be done in the near-term. Where there is program cost pressure, fear of program cancellation always follows. This drives program leaders to focus on key program measures to demonstrate success and without KAD requirements, program leadership focuses near-term capability while allowing designs to become less open, more tightly coupled and more rigid, making it even harder and more costly to add or change capability in the future. While conventional wisdom dictates that larger programs are less susceptible to pressure to pursue short-term gains at the expense of long-term flexibility, practical experience shows the opposite to be true.

The all-or-nothing uber-programs frequently become "too big to fail" which almost universally becomes the next Defense Acquisition University (DAU) lesson in what not to do (Flyvbjerg & Gardner, 2023). Sadly, programs called "too big to fail" have proven time and again to be too big to ever fully succeed. Loose coupling, modularity, and open systems provides the means to replace underperforming, low-value solutions with competitive alternatives. If an underperforming component is truly critical, it can be spun off as a separate program, developed and matured independently, while a less capable but more mature alternate "80% solution" is substituted to allow the rest of the program to proceed at lower risk. If instead of spinning-off problem elements, we use composability to segregate an uber program into smaller programs with lower risk, we have a more resilient enterprise portfolio, with a higher probability of each program success and higher probability of overall mission success. This does create new risks for interdependency between the different programs that make up the system, but this risk is manageable as long as alternatives are available. Interdependencies between programs often present themselves as Government Furnished Equipment (GFE) to the program responsible for systems integration. The programs that spin up to provide modules for larger systems must also employ composability to offer tailored, modular, and competitive solutions to better manage risk.

3. **Overbuilt by design, not need**: This is an insidious failure mode. In proprietary, monolithic and tightly coupled systems, it can be difficult, if not impossible, to identify and isolate critical safety and security boundaries. During requirements refinement, significant cost and schedule impacts can be identified that last the life of the program. For example, many safety related requirements require deterministic time behaviors that necessitate a Real-Time Operating System (RTOS). When only 10–20% may be related to deterministic time behaviors, the requirement for an RTOS is applied to 100% of the code. This prevents the use of much more flexible and affordable software environments that can rely on modular software techniques using mainstream development and deployment approaches (e.g., run on Linux, built with containers). This means 80–90% of the code is over-built and over-tested using an RTOS and associated development environment that can drive cost and schedule by a factor of 2-5x during

initial design and every time the code is modified for the life of the system. If, however, the systems team had architected with composability and put the malice of forethought to use that to segregate and isolate safety concerns, the stringent safety requirements and RTOS usage could be constrained to a single module within the system with additional requirements to ensure a high reliability (HIREL) design that requires few upgrades over the life of the system. This minimizes the impact of the stringent requirements over the life, while also allowing the rest of the system to progress much more efficiently. Security-related concerns follow a nearly identical design pattern of needing to be physically and logically isolated to the smallest possible security boundary.

4. **Custom for custom's sake**: This failure mode is frequently coupled with "overbuilt by design" by merging requirements inflation and tight coupling to the point where the only feasible answer to any make-buy decision drives into custom make. Seasoned MOSA architects recognize this as rationalization based on false pretenses where ample opportunities to use Commercial-Off-The-Shelf (COTS) or Military-Off-The-Shelf (MOTS) (M/COTS) abound. The rapid evolution of commercial processing has dramatically increased the amount of high-performance M/COTS processors that are suitable for military environments. As a result, DoD processing hardware modules are commoditizing, yet first and second tier defense suppliers are remarkably resistant to the trend. Based on the author's experience across many programs, a second 80/20 rule has emerged. Despite 80% of our processing is now being general purpose, we still custom build 80% of our hardware. Application of composability to isolate the 20% of our system with specialized processing needs is a simple concept technically, but overcoming the bias to put all solutions into the same pipeline turn out to be incredibly challenging. To optimize DoD systems to maximize use of M/COTS solutions, our Composability KADs must align military system interfaces with M/COTS interface standards. The solution isn't hard, it's just hard to accept.

5. **Trouble letting go**: As discussed throughout the paper, initial program requirements tend to focus on technical superiority, which leads to setting overly optimistic performance of key and/or emerging technologies. Fervent adherence to overly ambitious key performance parameters (KPPs) can and has killed programs. Early performance improvements are frequently stalled by diminishing returns as the dreaded 80/20 rule takes hold. In this context it means you tend to get 80% improvement with the first 20% of the budget, then spend 80% of the budget trying to achieve the last 20% of the performance required. A lot of architectural sins are committed trying to squeeze out the last margin of performance improvement. For example, a program team may start with good design patterns for modularity and abstraction to decouple hardware from software, but performance improvements can be had if you allow hardware to be directly controlled by the software. So as pressure mounts to achieve performance requirements, we have seen engineers skip abstraction layers to directly manage hardware to gain marginal improvement but incur long-term technical debt. Now, any time you try to upgrade the hardware or port the software, you will have to modify the software for the specific hardware. In many cases the marginal performance gains that cost so much and eroded composability gets "baked in" to the next technology upgrade.

If the program team had instead defined their minimum-viable-product to the 80% solution and allowed technology to mature independently to achieve the last 20%, overall cost and schedule would have been reduced while maintaining system composability. Here again, an Agile approach to KPPs combined with composability KADs produces better, faster and lower risk life-cycle performance improvement than attempts to achieve a performance measure ahead of the where the current technology is actually capable of going. Using Agile processes, we can lock in early gains and continuously evaluate incremental gains until we reach the point of diminishing returns. Investments can then be more efficiently and effectively focused on the next technology

that will provide performance with a much greater return. This approach also maintains system composability, so the next technology can be affordable integrated into the system.

6. **Silos of excellence**: In defense systems integration, the term “silos of excellence” refers to highly capable yet isolated systems that are engineered for performance within a single program but built with tightly coupled architectures that resist reuse, extension, or interoperability. These systems often contain custom-built interfaces, undocumented assumptions, and implementation-specific dependencies that form a “walled garden” of functionality. These silos enshrine some of our most exquisite, extraordinary and technologically superior capabilities we have today, but they are too costly and rigid. While they may perform exceptionally within their intended context, adapting them for use in other platforms, missions, or domains typically requires reengineering, code rewrites, and extensive testing, an effort that undermines agility and inflates life-cycle cost. Well-defined, open and modular interfaces are the key to breaking down these silos. Standards enable systems to communicate through shared, semantically-rich contracts that clearly define what data is exchanged, how it should be interpreted, and how components behave. By separating the “what” from the “how,” systems become loosely coupled but can have high cohesion. Each module can evolve independently, be reused in new contexts, and be integrated more rapidly without deep knowledge of the internal implementation. This interface-centric architecture supports versioning, validation, and automation, making it possible to test, deploy, and update components with confidence. As MOSA efforts mature, the defense ecosystem shifts from fragile, bespoke integrations to composable, interoperable systems of systems, enabling faster innovation and mission readiness.

The Government Accountability Office (GAO) got it wrong: As we conclude our analysis of pain points and pivot toward our path forward, it is important to question how prior failure analysis missed the mark. The GAO’s conclusion to many troubled programs points to a lack of early requirements definition/understanding and recommends more upfront analysis to better define requirements. That answer is at best, insufficient. Spending more time, better defining a more refined set of overly optimistic KPPs is just going to lead to more fervent application of the six pain points. Their conclusion actually makes matters worse, because it presupposes that the tomorrow’s requirements are knowable and relatively stable when neither condition is true. More requirements refinement upfront is only better if the emphasis is placed on KADs that allow us specify system architectures and requirements that are resilient to changing performance and capability requirements. While the concept of composable architectures is simple, actually architecting for composability and doing it well for our complex and highly cohesive military systems is quite difficult. To be clear, good architecture is not free and integration of composable military systems is not simple. The good news is that the tools and standards needed to make composability a tractable problem are rapidly maturing.

Aspects of Architecture – Implications on Design

The story of CAD (Computer-Aided Design) modeling begins in the 1960s with simple wireframe graphics, evolving rapidly in the 1970s and 1980s into parametric solid modeling. Early tools like Sketchpad, CATIA, and Pro/ENGINEER transformed engineering from drafting boards to digital workstations, improving productivity and enabling 3D visualization of complex assemblies. However, these models remained largely isolated and focused on geometric representation and disconnected from simulation, manufacturing, or systems-level design.

In the 1990s and early 2000s, the integration of CAD with CAM (Computer-Aided Manufacturing) began a digital manufacturing revolution. Parametric modeling and associative design meant that design changes automatically updated tooling paths, reducing errors and shortening production cycles. This era saw the emergence of Product Lifecycle Management (PLM) tools and standards like STEP (ISO 10303) that enabled better sharing of product models

across design and manufacturing systems. Still, CAD-CAM workflows were largely mechanical in focus, with limited integration of electrical, software, or systems-level logic.

The growing complexity of modern systems—particularly in aerospace, automotive, and defense—pushed engineers to think beyond parts and assemblies. As products became mechatronic (blending hardware, electronics, and software), the limitations of traditional CAD-centric workflows became apparent. This gave rise to Model-Based Systems Engineering (MBSE): an approach where a *digital model of the entire system* becomes the authoritative source of truth across life-cycle phases—from concept through disposal.

MBSE expands on CAD by incorporating behavioral models, functional logic, data flows, and inter-domain dependencies. Tools that use related modeling standards, like SysML (Systems Modeling Language) and UML (Unified Modeling Language) provide the backbone for modeling logical architecture, requirements, interfaces, and verification pathways. These models are not only descriptive but executable, allowing for early validation through simulation and integration with analysis tools.

The transformation from CAD-centric to model-based digital engineering has been powered by a constellation of standards and interfaces designed to facilitate interoperability and automation. Keys standards and interfaces enabling integration and automation include:

- STEP (ISO 10303): The foundational standard for exchanging 3D product data, especially geometry and product structure. Recent updates (like AP242) support PMI (Product Manufacturing Information), kinematics, and electrical harnesses.
- JT (Jupiter Tessellation): A lightweight 3D visualization format widely used in PLM systems for CAD data exchange and digital mock-up.
- SysML (OMG): The primary modeling language for MBSE, enabling modeling of system requirements, structure, behavior, and parametric constraints. Its extension into SysML v2 aims to bridge semantics more closely with engineering analysis and software execution.
- PLCS (Product Life Cycle Support – ISO 10303-239): Focused on long-term support and configuration management of complex systems, tying together engineering data over the entire life cycle.
- QIF (Quality Information Framework): Standard for metrology and quality data, enabling closed-loop quality control from design through inspection.
- MTConnect and OPC UA: Standards for machine-to-machine communication in manufacturing environments, enabling real-time integration between design models, MES (Manufacturing Execution Systems), and shop floor equipment.
- Functional Mock-up Interface (FMI): Enables co-simulation of models from different engineering domains, such as thermal, control, and mechanical systems—critical for virtual integration and digital twin development.

At the heart of today's efforts is the Digital Thread—a traceable, integrated chain of data that connects requirements, design models, simulation results, manufacturing plans, and operational feedback. This concept builds on decades of CAD and PLM progress, now extended by MBSE to enable automation and agility across the entire engineering ecosystem (AIAA, 2023).

In this environment, model updates ripple through simulation, requirements, and even machine tool paths without manual translation. The result is faster development, fewer integration errors, and enhanced ability to manage complexity and change. As AI and generative design begin to influence engineering processes, the robust digital infrastructure enabled by

CAD, PLM, and MBSE standards is what makes the next wave of intelligent, adaptive systems development possible.

Software & System Architecture Consideration

Why is software different? It doesn't have to be. Like hardware we need to architect software for integration, standardizing integration surfaces for agility and adaptability. Unlike hardware, however, software implementation can be performed without the necessary up-front work of preliminary design and management for future change.

Modern software-intensive systems operate in environments where rapid capability delivery and frequent technology refreshes are paramount. Yet, these very systems are often shackled by integration challenges that consume resources and slow progress. To overcome these constraints, software architects must shift their perspective, treating integration not as an afterthought, but as a primary design objective. This requires careful and intentional definition of integration surfaces that enable modularity, upgradability, and resilience in software architectures.

Integration surfaces, traditionally thought of as application program interfaces (APIs), must be broadened in scope. APIs represent only one form of interface; data representation, protocol adherence, timing dependencies, and the internal use of data also form critical aspects of the integration landscape. By considering all these surfaces, architects can create infrastructures that not only reduce the complexity of connecting components but also enable those connections to evolve independently. For instance, aligning application-level data models with protocol-level and signal-level representations creates an architectural clarity that minimizes the impact of change.

The goal is to reduce the brittleness typically associated with integration. Too often, systems are designed with tight coupling between components, where even minor modifications to a single module ripple across the entire ecosystem, requiring widespread retesting, recertification, and costly engineering effort. This "tyranny of commonality" constrains flexibility and undermines the very goal of rapid capability insertion (Lunde, 2023). Instead, by clearly defining software mating surfaces—boundaries at which change can be isolated and controlled—systems become more adaptable and maintainable.

A helpful analogy is that of mechanical assembly: in a well-designed machine, parts are interchangeable and interfaces well characterized. A transmission upgrade should not require recasting the entire engine block. In software, this principle is rarely followed. Without disciplined architectural separation, changes in business logic, transport protocols, or data models often require rebuilding and redeploying the entire system. This leads to high update costs and extended downtimes.

To counter this, integration surfaces must be designed to support software update flexibility as a first-class objective. We do this by enabling software updates to occur through architected integration design. Software update flexibility is a critical enabler of operational agility. In domains like defense, aerospace, and industrial systems, the ability to upgrade individual components without disrupting the entire platform is essential—not just for capability growth, but also for sustainment and compliance. This demands an infrastructure-first architecture where integration surfaces are purposefully constructed to absorb change rather than propagate it.

Such an approach requires architects to decouple infrastructure, protocols, and data semantics from application logic. Software components should not be statically compiled with assumptions about their runtime environment, dependencies, or mission set. Instead, interfaces

should be abstracted, configurable, and well-documented, allowing components to be replaced or upgraded without altering the underlying infrastructure.

A significant source of integration cost today stems from the repeated manual mapping of messages, fields, units, and primitives between systems. This repetitive activity, often buried in engineering documentation, consumes enormous effort and leads to brittle integration points. Automating these mappings through model-driven architectures, mediation layers, or canonical data models can eliminate large portions of non-value-added work.

Moreover, placing integration boundaries at appropriate levels of abstraction allows teams to manage risk more effectively. Moving integration surfaces closer to the application core—rather than at the infrastructure edge—can contain changes within a module and reduce recertification burdens. For example, modifying a data transport mechanism (e.g., switching from shared memory to TCP) should not require changes to the business logic, provided the integration surface is clearly delineated.

Ultimately, the hallmark of a robust software architecture is not only its initial design, but how gracefully it accommodates change. Architects must deliberately plan for evolution—not by hardening interfaces to resist change, but by making them flexible, modular, and expressive. This means rethinking how systems are integrated: using standards wisely, avoiding over-reliance on uniformity, and focusing on *interoperability*, not just compatibility (Carlton et al., 2021).

By taking an architecture-driven approach to integration—one that acknowledges all the software surfaces where change occurs—teams can break the cycle of rework and build systems that are ready for the demands of today and tomorrow (Allport et al., 2016).

DevOps and the Illusion of Integration: Automation Alone Doesn't Solve Architectural Interoperability

Over the past decade, DevOps has transformed software engineering by enabling rapid, automated build, test, and deployment cycles. Tools such as Jenkins, Kubernetes, GitLab CI, Docker, and Terraform have become the backbone of continuous integration and continuous delivery (CI/CD) in modern workflows (Kim et al., 2016). However, while DevOps enables faster deployment and operational responsiveness, it does not address the deeper architectural challenges required for true system-of-systems (SoS) integration, particularly in defense, aerospace, and other complex, multi-domain systems.

As emphasized in the 2021 ASNE Intelligent Ships paper (Hunt et al., 2021), DevOps automates the *Build and Deploy* stages of development, but defers critical integration work related to system behavior, semantic meaning, and interface design. These challenges are not about speed of deployment, but about cohesion, interoperability, and composability—aspects that DevOps, by itself, does not solve.

Where DevOps Ends: The Limits of Tooling in System Integration

DevOps tools provide value through automation but make key assumptions; that component interfaces are well-specified, that semantics are shared across systems, and that integration logic is known ahead of time. These assumptions rarely hold in large, evolving systems. For example:

- CI/CD Pipelines automate building and testing but don't resolve semantic mismatches or behavioral alignment (Fitzgerald & Stol, 2017).
- Kubernetes and other orchestration frameworks deploy containers but are blind to how those containers exchange and interpret data.

- Infrastructure-as-Code (e.g., Terraform) provisions compute resources, but doesn't enforce or even describe functional interactions between services.

Even within DevOps, the emphasis is on reducing time-to-deploy and increasing testing coverage. The DoD DevOps Reference Design (DoD CIO, 2019) defines Continuous Integration as automated testing and security scanning but does not prescribe methods for aligning system interfaces or documenting data semantics.

The Architectural Gap: Why Modularity and Interface Rigor Still Matter

To achieve scalable interoperability, systems must be designed around modular components with precise, semantically clear interfaces. This level of rigor is absent from most DevOps workflows. The Interface Documentation Maturity Levels (IDML) framework (Hunt & Allport, 2018) identifies levels of interface documentation maturity, showing that most DevOps efforts operate at levels 2 or 3 (human-readable or syntactic interface specifications). However, true integration requires IDML 5–7, where semantic context and compositional behavior are machine-readable and automatable.

Similarly, the Levels of Conceptual Interoperability Model (LCIM; Tolk, 2004) identifies that true composability is not achieved until LOI 5 (Conceptual)—where both the meaning and use of data and behavior are explicitly modeled. DevOps only addresses LOI 1 (Technical) and LOI 2 (Syntactic), leaving the most challenging and impactful aspects of integration unaddressed.

Beyond Pipelines: Architectures for Integration

To fill the architectural gap left by DevOps, the defense and aerospace communities are investing in open architecture frameworks that promote interface standardization, semantic modeling, and composable system design. Notable efforts include:

- FACE™ Technical Standard – A modular approach to airborne software architecture (The Open Group, 2021).
- SOSA™ (Sensor Open Systems Architecture): A collaborative standard that defines modular hardware and software interfaces for sensor systems, ensuring plug-and-play interoperability and rapid integration across vendors (The Open Group, 2023).
- Air Force GRA (Government Reference Architecture): A formal architecture effort aimed at ensuring that Air Force systems follow reference models for modularity, data exchange, and scalability across platforms and domains (Department of the Air Force, 2022).
- UDDL (Universal Data Description Language): UDDL is a machine-readable language designed to support data model documentation and semantic interoperability across systems. It provides a structured way to describe data entities, attributes, relationships, and semantics beyond traditional schema formats like XML or JSON Schema. UDDL is particularly useful in model-based integration, where precise, reusable, and shareable data definitions are essential for automating the composition of system interfaces. Unlike syntax-only representations, UDDL enables semantic layering, allowing systems to not only exchange data but also understand the context and intended use of that data.
- OMS Open Mission Systems (OMS): The standard is a U.S. Air Force initiative that defines common interfaces for integrating mission systems across different aircraft. It promotes a modular, open architecture that makes it easier and faster to add or upgrade capabilities. By separating software from hardware, OMS allows components to be reused and updated without full system redesign. It often works with the Universal Command and Control Interface (UCI) to improve communication between subsystems.

- UMMA (Unmanned Maritime Mission Architecture): The Navy's evolving reference architecture for modular and interoperable unmanned maritime systems, designed to enable faster capability insertion across unmanned surface and undersea platforms (Naval Sea Systems Command, 2022).

These frameworks address what DevOps cannot: the architectural separation of concerns, explicit documentation of behavior, and semantic mapping of interfaces required for composable and evolvable systems.

DevOps is necessary but not sufficient. While it automates deployment and life-cycle management, it does not resolve integration at the architectural level. For system-of-systems environments, the most expensive and brittle parts of development occur not in deployment, but in the misalignment of data semantics, undefined interface behavior, and manual integration rework. Addressing these challenges demands a shift toward explicit, machine-readable models of interface and behavior, guided by open architecture frameworks like SOSA, Air Force GRA, and UMMA.

Without this architectural rigor, DevOps simply accelerates the delivery of disjointed components. With architectural rigor, we can build adaptive, interoperable systems at scale.

Applying MOSA

Modular Open Systems Approach is well recognized as a strategic pilar of improving defense acquisition. This has been a multi-year approach with many laudable achievements and deserved accolades. However, there is much progress to be made, especially in achieving the business goals of MOSA (Guertin et al., 2015). There have been several attempts to assess compliance or to characterize a maturity model for what it means to achieve the business and technical goals of MOSA (Schenker et al., 2024).

When a contract includes the phrase "do MOSA to the maximum extent practical," it often implies a lack of genuine commitment to the Modular Open Systems Approach (MOSA). Essentially, it indicates that the contracting party is not willing to invest the necessary resources or funding to fully implement MOSA principles. Instead, they are opting for a more superficial or minimal compliance, rather than embracing the full potential and benefits of MOSA.

When a contract states "shall comply with MOSA policy," it often means that the contracting party is merely fulfilling a requirement for the sake of formality. From this language, it is not clear if the acquiring party is genuinely interested in the practical application or benefits of MOSA. Instead, it would appear that they are including the phrase to meet a bureaucratic or regulatory requirement, absent a measurable commitment for implementing MOSA.

The Open Systems Management Plan (OSMP) is a critical document in defense contracting that outlines the strategy for implementing the Modular Open Systems Approach (MOSA). By documenting the MOSA strategy in the OSMP, defense contractors can demonstrate their commitment to delivering innovative, cost-effective, and sustainable solutions that can evolve with changing mission requirements and technological advancements

However, when a contract includes the phrase "shall document MOSA strategy in an Open Systems Management Plan," it often implies a lack of clarity or understanding about how to implement the Modular Open Systems Approach (MOSA). Essentially, it indicates that the contracting party may lack the knowledge or implementation guidance on how to effectively request these strategies. This can lead to confusion and or superficial compliance, rather than a traceable commitment.

If the OSMP needs to be delivered with the proposal, it indicates that the contracting party is genuinely interested in the MOSA strategy and its implementation. This shows a commitment to understanding and integrating MOSA principles from the outset, making the answer to the MOSA strategy question significant and impactful.

Instead, if the OSMP is delivered after the contract award, it implies that the MOSA strategy is not a critical factor in the decision-making process for awarding the contract. In this scenario, the contracting party is not prioritizing MOSA principles during the initial stages of the contract, and the answer to the MOSA strategy question becomes less relevant and less impactful.

These sentences reflect a broader issue within the DoD acquisition process, where the true potential of MOSA is often overlooked or underutilized due to superficial compliance and lack of genuine commitment.

Real changes in achieving the business objectives of MOSA would be most likely to be achieved if the program crafting the contract solicitation also had sufficient depth of understanding of the available standards and technical approaches needed to perform to the desired outcome of the contracts. Key indicators that this is the case would be seen if the requirements specified a technical compliance to standards that are known to be useful for the intended purpose. Figure 2 graphically depicts a set of MOSA-related standards and how they can be applied in system design decisions.

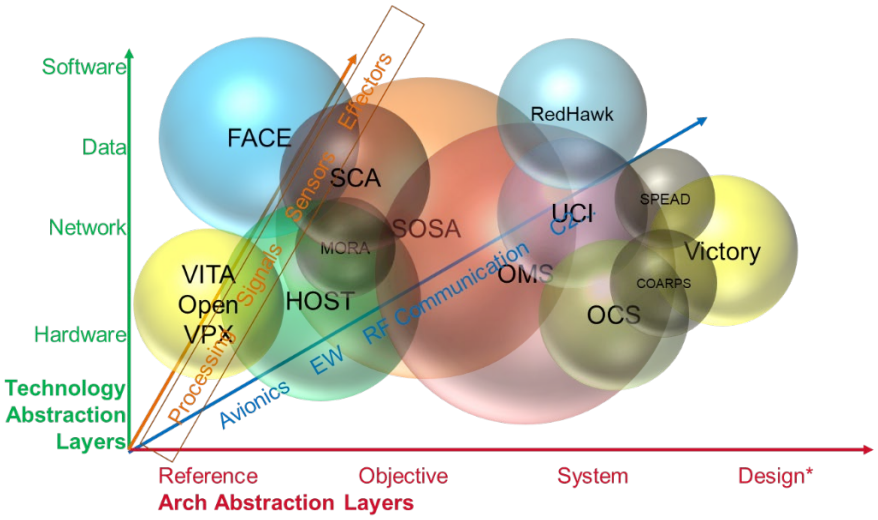

Figure 2. Mapping of Standards to Abstraction Layers for Technical MOSA Strategies

Test objectives and demonstrations of modular design criteria or interchange of components should be clearly spelled out and unambiguous to the reader. One of the critical enablers of this approach would be an open dialogue with industry about the goals of the product for life-cycle performance improvement and compliance to open standards. Industry/Government collaboration prior to release of the request for proposal (RFP) is key to achieving the Governments objectives.

FORGE Act Connection

The time is ripe for rethinking the transformation of effort and resource allocation into battlefield advantage. The crescendo of calls for achieving different outcomes is at a fever pitch and the willingness to act on this confluence is heartening; however, in our drive to be different,

we risk culling protections that would lead us back to the pit of the six pain points discussed above.

The most recent draft National Defense Authorization Act (NDAA) seeks to remove barriers to innovation and puts a heavy premium on commercial products and flexible acquisition approaches. While the thrust of this reform effort is laudable, it runs the risk of making it easy to present tight vertical integration. Knitted carefully into the FORGE Act language are key sentences that would promote large “one and done” procurements that would create unbreakable barriers to entry to all but the largest companies and once again risk vendor-lock solutions.

DoD risks losing access to American innovation if it does not become a better customer and carefully cull the counterproductive language from the FORGE Act. Doing so requires process improvements and investing in the management of the architecture of architectures it needs to create and maintain competitive pressure from large and small businesses. These are the methods by which we ensure the product is both sound and has many market alternatives. In doing so, it can also facilitate the use of commercial product development strategies while preserving fair opportunities with industry. In the years after World War II, it was common for DoD to create austere prototypes of several platforms across organizations. Then, it would put into production only the very best of the bunch. However, in doing so, we must preserve the learning that came from the long road we have traveled to ensure the government can make sound procurement decisions and acquisition flexibility to ensure we attract competitive alternatives and reward innovation on an open playing field.

Changes Needed for Certification for Use and Operational Test and Evaluation

We have made the assertion that the operational community needs to have products updated and re-fielded, fast. We also need to achieve our objectives against a thinking enemy who brings new things to the fight. To be employed at the speed of need, the system must also be composed to reduce fragility through modularity and loose coupling, facilitate rapid testing to find defects, and then quickly push corrections out to operational employment. Also, if a latent problem is found in the field, the system must be built to fall back to a known good state, report the issues, and be receptive to a subsequent update (McQuade et al., 2019). In future conflicts, a zero-day attack cannot wait months or even years for a proven remedy, yet that is routinely the case today.

Not fully addressed in the pain points above, the changing nature of software-intensive and cyber-physical systems use requires that testing have a higher level of prominence in the cost-trades associated with advancing a new and innovative design (Fields, 2018).

The concept of a “testable architecture” necessarily involves a contextual awareness of how the system performs its objectives and an approach to include the perspectives of a wider array of participants than the developers alone. This must be achieved by considering the testability of architecture as a first principle (Guertin & Hunt, 2017). With systems undergoing regular updates and deployment, the full range of stakeholder needs have to be present all along the path of creativity.

The application of artificial intelligence (AI) and machine learning (ML) is being considered for use across the defense portfolio. These designs require their own attention to architecture precepts for testing and monitoring of in-use behaviors that comport to structured frameworks for how testing will be performed as a life-cycle consideration, not just for producing improvements, but also for assessing how performance is changing in the deployed state (McCarthy et al., 2024).

- The application of AI is done in the context of solving a problem.

- Data has to be curated and managed to be effectively applied and integrated.
- Identification of training data and segregating of test data along with the best-fit use of algorithms is critical to success.
- Understanding of the operational environment and how the trained algorithm will behave in the use-case can be better understood through modeling and simulation.
- Once the AI/ML model is integrated into the system, then full useability and operational workflow can be assessed for effectiveness, survivability, and suitability.
- Lastly, how an AI-enabled system is continuously updated, recalibrated, evaluated for behavior drift and the need to trigger re-assessment needs to be established.

The tools and methodologies for testing must incorporate a cost-risk balance between automation and defect detection. Automation of test does not come for free, but advancing tools employed in this environment can identify areas ripe for investment. An area that calls for further study is to use the integrated decision support key (IDSK) as a possible framework for identifying parameters that will need to be evaluated as a life-cycle continuum as the system being built is improved over time.

Summary and Recommendations

Winning the future fight requires more than innovation and demands implementation superiority. This paper has explored how tightly coupled systems, siloed development practices, and legacy acquisition approaches limit our ability to respond with speed and agility. The solution lies in composable architectures, where modular design, rigorous interface definition, and semantic interoperability enable continuous evolution, rapid fielding, and affordable upgrades. But composability doesn't happen by accident; it must be deliberately architected, measured, and enforced.

To realize these benefits, we must elevate architecture to a first-class engineering discipline. Architecture is not just about structure—it is the mechanism that makes complexity manageable and change possible. It is not easier than design, but it is more essential. Without well-formed architectural strategies, programs fall into familiar traps: brittle integration, vendor lock, and systems that are too costly to adapt or sustain.

Crucially, we must stop trying to align system-specific designs to standards in isolation. Instead, we must align architecture concepts across standards, establishing a normalized foundation of abstraction, separation of concerns, and modularity. Each standard should clearly document where architectural principles are preserved and where they are refined into constraining design specifications. This allows for interoperability between standards, not just between systems.

The tools, models, and frameworks are now mature enough to support this approach. The opportunity is real, but success depends on clear leadership intent, acquisition reform, and a shared understanding that composability is not a technical preference, but a strategic necessity. Systems built today must be designed to evolve—because if we don't build to change, we won't be able to compete.

The recommendations are clear and need courage and commitment to realize and achieve our goals. We are on a similar path to the decades long model-based transformation which has occurred for hardware and material systems manufacturing. To see this to the end, we must:

- Architect first. Elevate architecture to drive integration, not lag behind it.
- Prioritize composability as a Key Architecture Driver (KAD) equal to performance and cost.

- Align architectures across standards and use standards to refine—not redefine—those concepts.
- Measure and enforce interface rigor through appropriately architected interfaces boundaries.
- Reject superficial compliance. Build systems that are testable, modular, and ready to evolve.

Architecture is not a checkbox. It is the battlefield where flexibility, speed, and superiority are won.

References

Allport, C., Hunt, G., & Revill, G. (2016). Rethinking system integration. *Electronic Engineering Journal*. http://www.eejournal.com/archives/articles/20160609-interoperability

American Institute of Aeronautics and Astronautics. (2023). *Digital thread: Definition, value and reference model*.

Bath, A. (2025, January 16). Navy fired more than 200 missiles to fight off Red Sea shipping attacks, admiral says. *Stars and Stripes*. https://www.stripes.com/branches/navy/2025-01-16/houthis-navy-red-sea-missiles-drones-16500246.html

Boydston, A., Feiler, P., Vestal, S., & Lewis, B. (2019, September). *Architecture centric virtual integration process (ACVIP): A key component of the DoD digital engineering strategy*. U.S. Army.

Carlton et al. (2021). *Architecting the future of software engineering, a national agenda for software engineering research & development*. Carnegie Mellon University, Software Engineering Institute. https://insights.sei.cmu.edu/documents/1309/2021_014_001_741214.pdf

Chick, Pavetti, & Shevchenko. (2023). *Using model-based systems engineering (MBSE) to assure a DevSecOps pipeline is sufficiently secure* (Technical Report CMU/SEI-2023-TR-001). Carnegie Mellon University, Software Engineering Institute. https://doi.org/10.11184/R1/22592884

Colclough et al. (2024). Software development practices are changing the character of the test and evaluation enterprise. *ASNE Naval Engineer's Journal*, *136*(1 & No. 2).

Del Toro, Kendall, & Wormouth. (2024). *Modular open systems approach for Department of Defense weapon systems*. https://www.cto.mil/wp-content/uploads/2024/12/Tri-Service-Memo-Signed-17Dec2024.pdf

Department of Defense Chief Information Officer. (2019). *DoD enterprise DevSecOps reference design*. https://dodcio.defense.gov

Department of the Air Force. (2022). *Digital engineering playbook: Government reference architecture (GRA)*. Air Force Digital Transformation Office.

Eckstein. (2024, March 21). US Navy making Aegis updates, training changes based on Houthi attacks. *Defense News.* https://www.defensenews.com/naval/2024/03/21/us-navy-making-aegis-updates-training-changes-based-on-houthi-attacks/

Fields. (2018). *Designing and acquisition of software for defense systems*. Defense Science Board.

Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Flyvbjerg, B., & Gardner, D. (2023). *How big things get done: The surprising factors that determine the fate of every project*. Currency.

Guertin & Hunt. (2017). *Transformation of test and evaluation: The natural consequences of model-based engineering and modular open systems architecture*. Naval Postgraduate School, Acquisition Research Program.

Guertin, N., & Schenker, F. (2021). Using value engineering to propel cyber-physical systems acquisition. In *Proceedings of the 18th Annual Acquisition Research Symposium*. Naval Postgraduate School.

Guertin, N., Schmidt, D. C., & Sweeney, R. (2015). How the Navy is using open systems architecture to revolutionize capability acquisition. In *Proceedings of the 12th Annual Acquisition Research Symposium*. Naval Postgraduate School.

Hand, S., Lombardi, D., Hunt, G., & Allport, C. (2018). *Interface documentation maturity levels (IDML): An introduction*. Open Group FACE™/U.S. Army Technical Interchange Meeting.

Hughes & Jackson. (2021). *A framework for DevSecOps evolution and achieving continuous-integration/continuous-delivery (CI/CD) capabilities*. Carnegie Mellon University, Software Engineering Institute.

Hunt, G., Allport, C., Foley, S., & Hand, S. (2021). Achieving warfighting lethality via system and semantic system-of-systems model-based engineering integration. *Proceedings of the ASNE Intelligent Ships Symposium*.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations*. IT Revolution Press.

Lunde. (2023). *The true cost of software complexity*. https://ronlunde.com/post/costofcomplexity/

McCarthy et al. (2024). Key steps to fielding combat credible AI-enabled systems. *ASNE Naval Engineer's Journal*, *136*(1 & 2).

McQuade et al. (2019). *Software is never done: Refactoring the acquisition code for competitive advantage* [The Software Acquisition and Practices [SWAP] Study].

Naval Sea Systems Command. (2022). *Unmanned maritime mission architecture (UMMA) overview*. Presented at NDIA Expeditionary Warfare Conference.

The Open Group. (2023). *Sensor open systems architecture (SOSA) technical standard edition 1.0*. https://www.opengroup.org/sosa

Schenker, A., Guertin, H., & Schmidt, D. (2024). *A model for evaluating the maturity of a modular open systems approach.* Carnegie Mellon University, Software Engineering Institute.

Schmidt, D. C. (2016, July 11). *A design for maintaining maritime superiority: A naval perspective on open-systems architecture* [Blog post]. SEI Blog. https://insights.sei.cmu.edu/sei_blog/2016/07/a-naval perspective-on-open-systems-

Tolk, A. (2004). Composable mission spaces and M&S repositories—applicability of open standards. *Spring Simulation Interoperability Workshop*. IEEE.

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net