SYM-AM-26-070

Excerpt from the Proceedings

of the

Twenty-Third Annual Acquisition Research Symposium and Innovation Summit

Wednesday, May 6, 2026 Sessions

Volume I

"Accelerating Warfighting Capabilities"

Digital Engineering to Improve the Quality of a Capability Needs Statement

Published: April 30, 2026

Approved for public release; distribution is unlimited.

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

The research presented in this report was supported by the Acquisition Research Program, Graduate School of Defense Management at the Naval Postgraduate School.

To request defense acquisition research, please contact:

Acquisition Research Program
Department of Defense Management
Naval Postgraduate School
E: arp@nps.edu
www.acquisitionresearch.net

Copies of Symposium Proceedings and Presentations; and Acquisition Sponsored Faculty and Student Research Reports and Posters may be printed from the **NPS Defense Acquisition & Innovation Repository** at **https://dair.nps.edu/**.

ACQUISITION RESEARCH PROGRAM
DEPARTMENT OF ACQUISITION, FINANCE, AND MANPOWER
NAVAL POSTGRADUATE SCHOOL

Digital Engineering to Improve the Quality of a Capability Needs Statement

Alfred Schenker—works in the Software Engineering Institute's (SEI's) Software Solutions Division, working there for over 25 years. He works to improve software acquisition and product development practices throughout the armed services and other organizations. He has actively worked in software process, architecture, model-based systems engineering, and metrics. Before joining the SEI, Mr. Schenker spent over 20 years in industry as an active contributor in all phases of product development. Mr. Schenker is also an inventor and has obtained patents for a pressure switch (used in automotive airbag applications) and a manufacturing process to seal gas inside a vessel. [ars@sei.cmu.edu]

Abstract

This paper describes how an acquisition program on the Software Pathway (SWP) can elaborate its software Capability Needs Statement (CNS) within a modern digital engineering environment using Model-Based System Engineering (MBSE) principles. This elaboration (i.e., view) transforms the CNS from a static description into an authoritative, model-based view of mission deficiencies, required enhancements, interfaces, and interoperability. This view enables sponsors and users to prioritize and validate their needs iteratively with traceable evidence. This approach will lead to a more defensible and streamlined software development strategy. It also enables the program to review the capability with stakeholder(s) to ensure adequate validation and develop a quality artifact that can be used for negotiating with contractors.

Keywords: Software Acquisition Pathway, Model-Based Systems Engineering, Capability Needs Statement, Model-Based Acquisition, Agile, Minimum Viable Product, Digital Engineering

Introduction

This paper introduces clear rationale for an acquisition program on the Software Pathway (SWP) to elaborate its software Capability Needs Statement (CNS) within a modern digital engineering environment. This is accomplished by employing Model-Based System Engineering (MBSE) principles that transform the CNS from a static description into an authoritative, model-based view of mission deficiencies, required enhancements, interfaces, and interoperability. This view enables sponsors and users to prioritize and validate their needs iteratively with traceable evidence rather than with documents. The motivation for developing this view is to achieve a digital representation of the system. This representation is available for the express purpose of capability definition, elaboration, and validation, and provides the acquirer with a superior mechanism for independent verification and validation. It also provides the user with a higher likelihood that the technical solutions for the capabilities will better meet their needs.

We at the Software Engineering Institute (SEI) believe this approach will lead to a more defensible and streamlined software development strategy. Although adopting this approach requires program office staff to have a level of model-based competency, using a digital model to review newly proposed capability(ies) provides a simple means to (1) review the capability with stakeholder(s) to ensure that there is adequate validation built into the process and (2) develop a higher quality artifact for negotiating with contractors.

The approach outlined in this paper should have broad applicability to acquisition programs, whether on the SWP or not, although there are specific artifacts referenced that are required for programs on the SWP. The suggested model-based approach elaborates elicited user need statements to enhance their usefulness. The benefits of this approach are illustrated by a use case that demonstrates the use of a model-based method to assess potential implementations of a newly proposed capability. Such an assessment would add value (or

reduce risk) for the users (source of the capability), the developers (implementers of the capability), and the acquirers. This approach represents a potential best practice for connecting stakeholder needs with practical reality using a digital engineering model as a facilitation tool.

The paper starts with a review of SWP and CNS guidance. It follows with an abstract discussion of capability and capability evolution. The next section discusses Digital Engineering (DE) and introduces the concept of a Digital System Model (DSM). The marriage of capability evolution and DSM is described with a notional model-based capability lifecycle. The paper concludes with a use case that illustrates different examples of how an organization might implement a DSM to elaborate its CNS along with a pragmatic discussion of risks and issues.

Software Acquisition Pathway Context

The SWP, established by the Department of Defense (DoD) in 2020 and documented in DoD Instruction (DoDI) 5000.87, is part of the Adaptive Acquisition Framework (AAF; Defense Acquisition University [DAU], n.d.). The AAF was created in response to findings identified in several high-level studies, policy, and strategy documents, including the following:

- 2019 Defense Innovation Board Software Acquisition and Practices (Defense Innovation Board, 2019)
- Fiscal Year 2018 National Defense Authorization Act (NDAA) Section 873 and 874 Agile pilot programs (U.S. Congress, 2017)

Primarily, the SWP was created to accelerate the delivery of software-related capability to the warfighter. The SWP creates a separate mechanism for developing software capability that is independent of the system development lifecycle. The SWP promotes modern development practices, such as Agile and DevSecOps.

There are some important things to consider, especially with respect to Agile. As will be described below, capability evolves incrementally. Agile development practices fit this paradigm very well, as the Agile increments (i.e., sprints) are typically very short and produce capabilities that are testable. Project stakeholders (e.g., users, testers, developers) review the product at the conclusion of a sprint and can assess what is needed (if anything) to enhance the capability.

A digital, model-based representation of a capability should provide the "building blocks" that illustrate what to implement for that capability. These building blocks represent work to be completed. The work descriptions reside in a "Product Backlog." The easiest way to think about this backlog is that it is like an inventory of work to be done. When the stakeholders build the project scope for a sprint, they draw from the backlog. (See a list of stakeholders and their roles in the appendix.)

Agile also defines terms, such as *minimum viable product* (MVP), *minimum viable capability release* (MVCR), and *definition of done*, that can have a dramatic impact on the as-defined inventory. For example, if we have a coherent definition of a capability, and we can represent it as "minimum viable," we have a representation of a version of the capability that will enable the product to perform. The capability may not be completely implemented (as the user wished initially), but it may be "good enough." By defining capability in this way, we create a trade space, or flexibility in the way the Program Management Office (PMO) can manage the project.

Priorities may change that require changes in the product backlog. If the PMO has an implementation that is good enough, it can shift the effort originally allocated to this capability to something else that has a higher priority. So, some inventory may become obsolete or new

inventory may be added as a result of trading off priorities. These changes could be quite significant and illustrate how useful the Agile paradigm is in the context of building capability.

On March 6, 2025, the Secretary of Defense issued a memorandum regarding improving the acquisition process that stated, “To meet this challenge, I am directing all DoD Components to adopt the Software Acquisition Pathway (SWP) as the preferred pathway for all software development components of business and weapon system programs in the Department” (Hegseth, 2025).

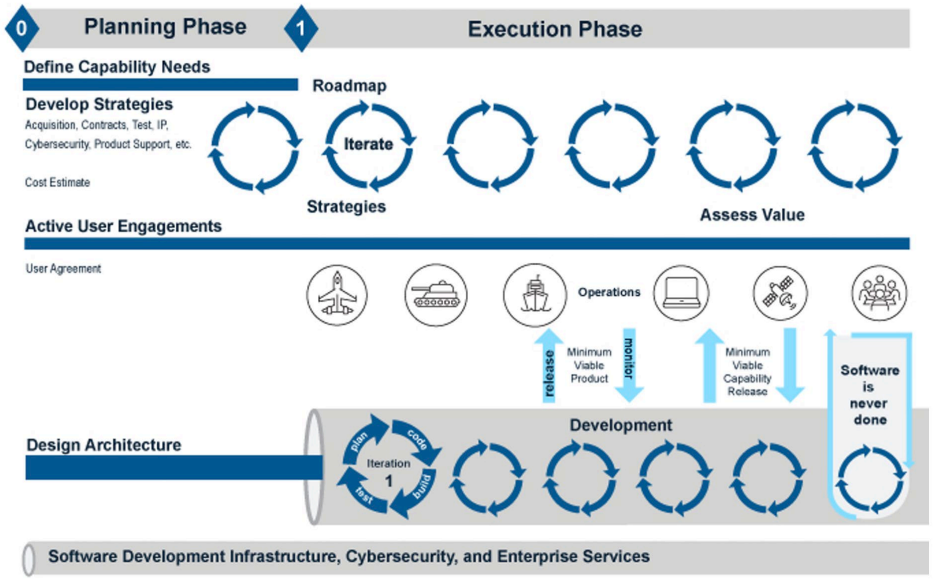
A program on the SWP needs to develop the plan for building the system’s software. The plan will likely be informed by several things, including

- prior implementations
- a set of criteria for establishing development priorities
- a guiding document that describes the desired software capabilities of the system

The last item, also referred to as a CNS, is a required artifact defined by DODI 5000.87:

> a high-level capture of mission deficiencies, or enhancements to existing operational capabilities, features, interoperability needs, legacy interfaces and other attributes that provides enough information to define various software solutions as they relate to the overall threat environment. (DoD, 2020)

DODI 5000.87 includes a graphic that illustrates the expected high-level process for a program on the SWP. Note that the capability needs are shown in the upper left corner, but in the lower right corner, it says, “Software is never done.” The implication is that software capability is never done either. For example, new use cases are imagined after initial capabilities are deployed, threats evolve, hardware is updated, etc. All of these represent potential updates to a CNS. Therefore, it is important to think about the CNS as a living document.



[Source: (Department of Defense, 2020) ©2020 by Department of Defense. Reprinted with permission.]

Figure 1: The Software Acquisition Pathway

Capability Needs Statement

A CNS is a key acquisition artifact that documents the high-level capabilities provided by the software. It is a living document and should be reviewed and updated as needed (at least annually), typically after a system's *value assessment* (Software Engineering Institute, 2026). A value assessment in Agile is the process of measuring the tangible benefits—such as increased functionality, efficiency, or revenue—delivered by teams. Such an assessment is usually reviewed every 10 months (or more often) to ensure its relevance. It focuses on prioritizing working software and customer feedback over rigid plans, using assessments to improve delivery and secure funding (DAU, n.d.).

To enter the SWP, a program must have a draft CNS. This is needed to get an SWP Acquisition Decision Memorandum (ADM) from its Decision Authority (DA). The program's sponsor is responsible for creating the draft CNS with input from users. The draft CNS can be based on a Concept of Operations (CONOPS), a Concept of Employment (CONEMP), or information from a Capabilities Based Assessment or Analysis of Alternatives (Warfighting Acquisition University, n.d.). The CNS is updated as needs change, as the capability is implemented, or as priorities change.

The CNS should also be clear and concise. It should describe the desired mission outcomes and how the software capabilities will enable these outcomes, and it should be fewer than 10 pages long. For guidance on developing your CNS, refer to *Cracking the CNS Code* (Software Engineering Institute, 2026). (See the CNS template for more descriptions [DAU, 2021].)

Capability Definition Is Not Static

When a new capability is provided to users, it rarely works exactly the way the user (who requested the capability) expects. That is not necessarily bad, and this situation can't be solved by policy. The fact is that it is very difficult to define exactly what is needed. Further, the more you use something, the better you get at using it. Capabilities naturally evolve; they should be developed incrementally and be validated by users as they are developed. In this way, useful capabilities can be acquired in a systematic way.

For most capabilities, this means that there ought to be a way to characterize a capability so that it is defined as a "base" set of features plus feature enhancements (that potentially increase the value of the capability). In this way, the user has a chance to evaluate the implementation of the capability while the system is still in development. It is a challenge to precisely define these enhancements a priori, because it is hard to envision how to best use something until you are using it. We have seen this repeatedly with many products.

The safest approach is to first develop what everyone agrees that the capability must perform. Once that has been accomplished, and it is potentially deployed into an evaluation environment, enhancements will be easier to identify, understand, and prioritize.

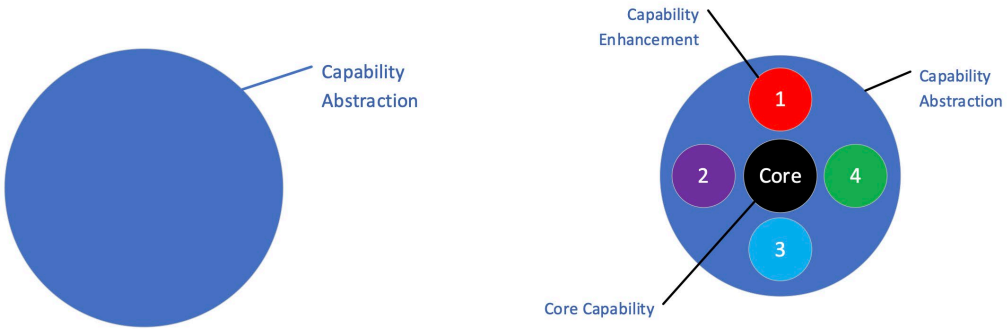


Figure 2: Capability Abstraction

Figure 2 illustrates two ways to represent a capability. Think of a capability as something that needs to evolve. The representation on the left might reflect the initial (user's) characterization of the capability; but to implement it, the capability needs to be broken down into what must be done initially (often referred to with the descriptor "minimum viable") and what enhancements might be added in the future to make the capability better.

The transformation from the blue circle on the left to the discrete picture on the right provides two very important functions. There is a reviewable artifact that shows how the capability might be satisfied and a notional allocation that identifies the core functions and the functions defined as enhancements. Validating this picture with the user ought to be illuminating. The user may not wish to compromise and will insist on an "all (i.e., core) or nothing" strategy, but our experience has been that users don't think about capability in this way. It may be that there is a give-and-take to define the minimum viable capability.

Another way to think about capability evolution is that the developer may be implementing only the core capability but is probably aware of the enhancements that are on the horizon. Accordingly, the developer may propose to scope the minimum viable capability as the core plus some additional infrastructure that would need to be present if/when an enhancement is prioritized. This would be a judgment call for the acquirer.

For example, a fighter jet may need a radar-sensing capability. There are plenty of radars to choose from, and the minimum viable capability should be straightforward to implement. However, it may be that threat analysis suggests enhancements to improve survivability. For a radar system, these enhancements might relate to things such as integrating a jamming capability or increasing the resolution of the sensor to enable the detection of Small Unmanned Aerial Vehicles (SUAVs). Others may emerge during the acquisition that require a change in priority or an entirely new scope. As the capability evolves, enhancements are incorporated into the base as illustrated in Figure 3.

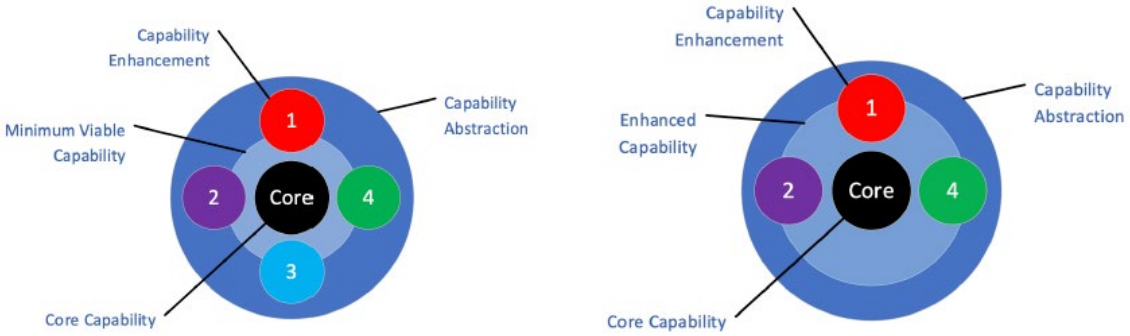


Figure 3: The Evolution of Capability

Based on the plan to incorporate Enhancement #3, the developer may have included some infrastructure to make the enhancement easier to implement. The enhanced capability, represented by the inner light blue circle, is slightly larger than the previous minimum viable capability, as Enhancement #3 has been incorporated into the plan. At any point during system (or software) development, stakeholders may declare that the capability is "done" primarily because it is good enough and other capabilities have a higher priority. This way of rationalizing how capability evolves fits right in with the Agile concept of MVP.

There is a history in DoD acquisition of missing the mark on capability implementation. When compromise isn't possible on defining the core, there is no other choice than to build the capability in its entirety. What usually results is a capability implementation that looks like Figure 4. The capability is implemented with Enhancement #2 and Enhancement #3 incorporated, but Enhancement #1 and Enhancement #4 were not. This might have been caused by running out of time (or money) or some system dependency that was not implemented. Regardless, instead of developing a minimum viable capability, we are left with a partially implemented capability that potentially limits end-to-end system performance, which is not what we want to do.

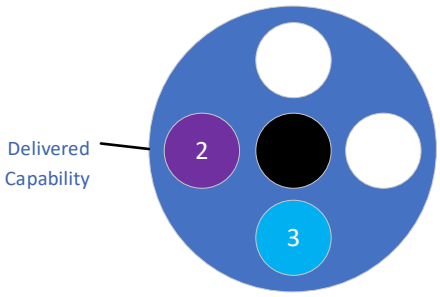


Figure 4: Undesirable Capability Implementation

Digital System Model

Digital Engineering (DE) is becoming pervasive as part of the development of software systems. Initial guidance for DE was issued by the DoD in 2018 and updated in late 2023 as DoDI 5000.97 (DoD, 2023). A handbook titled *Digital Engineering and Modeling Practices* (MIL-HDBK-539A) was just released in March 2026 (Department of War, 2026).

The basic value proposition for a DSM is to enable a process to shift left. Shifting left can be rationalized generally as a trade-off: early investment in a model (shifting left of cost) to prevent late identification of defects or issues. The economic case for DE is based on early defect detection, which was argued by Feiler et al. (2013) and Hansson et al. (2018). In short, these researchers present evidence that, in the domain of embedded safety-critical systems, 35% of errors are introduced in requirements, and a further 35% of errors are injected in architectural design. The cost of correcting an issue in later phases can be 300 to 1,000 times the cost of correcting it in phase (Dabney, 2003).

An organization that wishes to apply DE principles to its process seeks to provide early feedback that guides future development to mitigate project risk. These risks typically are manifested as cost and/or schedule overruns coupled with potential compromises in delivered capability.

A DSM that employs a systems modeling language (SysML) can document requirements and requirement elaborations. These elaborations typically take the form of block diagrams, sequence diagrams, and interface descriptions. An important element of a SysML model is that it maintains traceability between the diagrams and the requirements. This is very important

when reviewing new requirements or changes to existing ones. In the context of this paper, the DSM we refer to is one that is created and maintained by the acquirer.

The representation of a CNS in a DSM, created by the government, would be a break from tradition. We refer to this representation as a Government Reference Architecture (GRA). The GRA enables the PMO to elaborate capabilities and/or requirements in a model-based environment. To be clear, for this type of model application, the program office is assumed to represent the requirements with more than text. In other words, the acquirer is expected to elaborate on how it envisions the requirement to be implemented. The elaborated model-based representation can then be used for different purposes (e.g., requirement validation [with the requirement provider], requirement implementation guidance [for the project team]).

Model Development Lifecycle

One of the most important things to do when attempting to develop a useful model is to proactively identify the way the model is intended to be used by its stakeholders. The reason for this is that a model development effort can go "off the rails" (generally because the model tool is so flexible) if model developers start to use the model in ways that it was not intended for. Model development represents a process, and there will likely be a trade-off in development between the normal process improvement activity of process review and improving and changing the intended use of the model. For example, the process review could identify that the model would have been more useful if a certain type of information was added. This may be true, but it would come at a cost (more time, more money, training, etc.). The model team should be very careful about changing the use cases of the model. A prototyping effort might be conducted very early in the project to validate the approach.

If model use case(s) are defined up front, then the process practitioners will have a structure they can work with to improve the way they use the model to perform their job. Changing the use cases after significant investment has been made in the model is generally not a good idea. So, we strongly advocate that the PMO should identify very early how it intends to use the model. Likewise, the PMO should communicate model use case(s) with others who will have a role in the process.

Beyond these tasks, the model developer should anticipate inputs and develop the process for (1) updating the model, (2) verifying that model updates reflect new inputs, and (3) releasing the updated model to stakeholders.

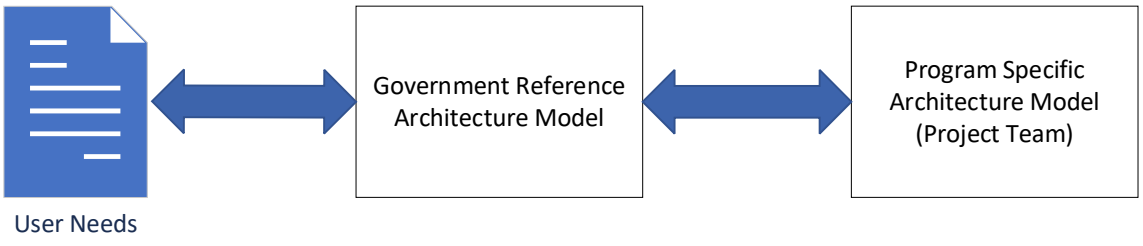


Figure 5: GRA Model Development Overview

A very simple representation of the process is depicted in Figure 5. Initially, the process flows from left to right, as the acquirer represents and elaborates the stakeholder's capability needs in the GRA model. However, as the product is built, the process will flow in the other direction, reflecting design decisions and constraints that may impact the implementation of some of those needs. In this way, the user, the acquirer, and the contractor all participate in a verification and validation process that maintains consistency between user needs and the development of the system.

What the Model Could Be Used to Do

The following are the most likely uses of the GRA:

1. **Validate User Needs.** The PMO will use its domain knowledge to build a notional model-based representation of each user need. This may involve anticipating specific hardware, software components, and interfaces. As part of the validation, the PMO attempts to characterize the relative priority of each user need. Documenting specific constraints (e.g., performance, cost, schedule, technology dependencies) should also be incorporated.
2. **Communicate User Needs to the Contractor.** The PMO incorporates the GRA into the contract it negotiates with its development contractor.
3. **Serve as a Communication Mechanism.** The PMO uses the GRA to communicate with users and the project team as development progress is made and priorities change.
4. **Provide Traceability Between User Needs and Project Backlog.** The PMO translates from the CNS into the GRA, identifying elements that need to be implemented. The work associated with building the capability traces from the GRA to the Project Backlog tool.

Discussion of Capability Evolution and a Capability Lifecycle

Sometimes capabilities are explicit, and all stakeholders understand the capability. Sometimes the capability is something new. Let's examine the capability of an automobile's cruise control. At this point, everyone understands the concept behind the capability. The driver can maintain a constant speed without using the accelerator or the brake. The concept is very old; the first patent for a "Speedostat" was granted in 1950 to Ralph Teeter, and it basically accomplished the task of controlling the speed of the car (Sears, 2018). Cruise control was first released in 1958 by Chrysler. By the mid-1970s, cars with cruise control were being sold by American Motors, General Motors, Chrysler, and Ford.

The core capability in this case is to automatically maintain a constant speed of a car. Implementing the capability was initially accomplished in several different ways. A breakthrough occurred in the late-1980s when Motorola developed an electronic cruise control unit referred to as the Automotive Speed Control Processor. When integrated with an electronic throttle control, this unit became the starting point for modern cruise control, as it integrates the cruise control function with the engine management systems.

For several years after the release of the electronic unit, this capability remained fairly static. However, in the past 20 to 30 years, there has been a steady evolution of new capability enhancements. For the most part, these enhancements were enabled by new technology, but some were simple software enhancements. An example of a software enhancement is the way the cruising speed is adjusted, whether by 1 mph or by 5 mph, with each press of the up/down arrow.

When new technology (e.g., GPS, low-cost radar, lidar, or imaging systems) became available, new cruise control capabilities could be imagined. It is now common for cars to have *adaptive* cruise control. This capability automatically adjusts the speed of the vehicle to the one in front of it to maintain a safe driving distance. It is also normal now for the vehicle to have some sort of safety-oriented capability that the new cruise control capability integrates with.

New technology will continue to drive the evolution of this capability. An example of this might be a predictive adaptive cruise control system that gathers intelligence about the road ahead and adjusts the speed to deal with the specific driving context (e.g., reduce speed if a policeman is reported ahead or if weather conditions ahead may cause an unsafe driving condition).

For the Department of War (DoW), we should think of capabilities in these terms. There is a base capability: the ability to maintain the set point of the car's speed. Then we can think of enhancements to meet various purposes (e.g., accuracy, user interaction, new technology incorporation). The timeline for capability development might not take as long as it has for cruise control. In fact, new enhancements might be enabled within a single increment of development. This fits in perfectly with the Agile project management construct described earlier. However, thinking about capability in this way is different from the way it has been thought of in the past.

Model-Based Capability Definition

For a specific acquisition, we can imagine a software capability that is required for the system. In most cases, this will not be an entirely new capability, rather it will be some legacy capability that needs to be enhanced. Enhancements are typically documented as requirements. These requirements come from users or, as in the case of security and/or safety, from a regulatory agency or the desire for compliance with DoW policies.

The main point of using interconnected DSMs is to ensure that mission needs, requirements, design, and verification are end-to-end traceable and analyzable using a structured modeling language rather than disconnected documents. A notional workflow for the capability translation might include the following steps: analyze mission threads in a model, identify gaps and perform trade studies, define new capabilities, analyze key performance characteristics and measures, allocate new capabilities to systems and identify impacted interfaces, and construct a high-level verification strategy.

Most systems being developed for the DoW are not greenfield (i.e., entirely new), although some system capabilities may be. What this means is that, in the most common case, the PMO has a very good picture of what it is acquiring.

For example, the Army has attempted to replace the Bradley Fighting Vehicle (BFV) several times over the last 25 years. Each time it tried, a PMO was stood up to manage the acquisition. The BFV replacement (especially from a software perspective) will have a similar capability as the existing BFV. There will be new technology, advanced computing capability, improved network communications, etc., but it should be easy for the acquirer to logically organize the BFV replacement vehicle based on the current BFV. For example, consider the navigation capability. It may be that the BFV did not incorporate GPS into its driver navigation, but a BFV replacement would require it. Regardless, the core logical navigation functions (e.g., waypoint navigation, reporting position) should be very similar.

If we can imagine how the PMO staff could logically construct the vehicle based on functional capabilities, it does not seem to be a stretch for them to use a model-based approach to perform the logical construction. The model may or may not act as a constraint for the project team, depending on how much control the PMO wants to exercise.

An example where the acquirer might constrain the contractor could be how to represent the BFV replacement's underlying infrastructure to comply with a modularity requirement. The PMO may want to employ modular components with a common infrastructure to simplify upgrading technology and other future modernization efforts. This decision will constrain the project team's ability to define modularity boundaries, along with the underlying infrastructure, and seek PMO approval as part of its process. While this may seem to be a completely normal part of the design process, there may be additional work performed by the project team to meet this requirement, and it is likely that the project team may not wish to comply.

In general, the capabilities can be represented in black boxes, with little to no constraint on what occurs within those boxes. Compliance with a data model may be required, and the boundary will be managed by an interface description. However, the PMO will focus on defining

the performance characteristics of the capability. In most cases, the project team is responsible for the precise details that describe how the black box capability is implemented.

The following sections define a notional process for how the discrete capability needs (as summarized in a CNS and defined by project stakeholders) could be incorporated into a model and used as a critical point of communication between the project team and the user community.

Model-Based Capability Notional Process

Figure 6 illustrates a possible process for model-based capability. It's not a coincidence that operational needs are at the center of this diagram. Operational needs typically result from using the capability, observing some aspect of the capability that fell short, and recommending an enhancement to prevent a future recurrence of this deficiency. For the warfighter, this may have come from performing missions. A platform could have been used to perform some mission-related function with a negative result. An after-action review might have identified the problem and initiated the creation of a new need (or enhancement).

The pace of technology advancements seems to increase every day. This has always been a problem for DoD weapons systems, as these systems inevitably are outdated on the day they are released to the warfighter. One way to mitigate this risk is to actively elicit new operational requirements during the development of a new or updated system. This elicitation mitigates the risk of fielding obsolete products but increases the likelihood that the program will exceed its schedule and/or budget. It seems there will be some learning required to accomplish this process element effectively.

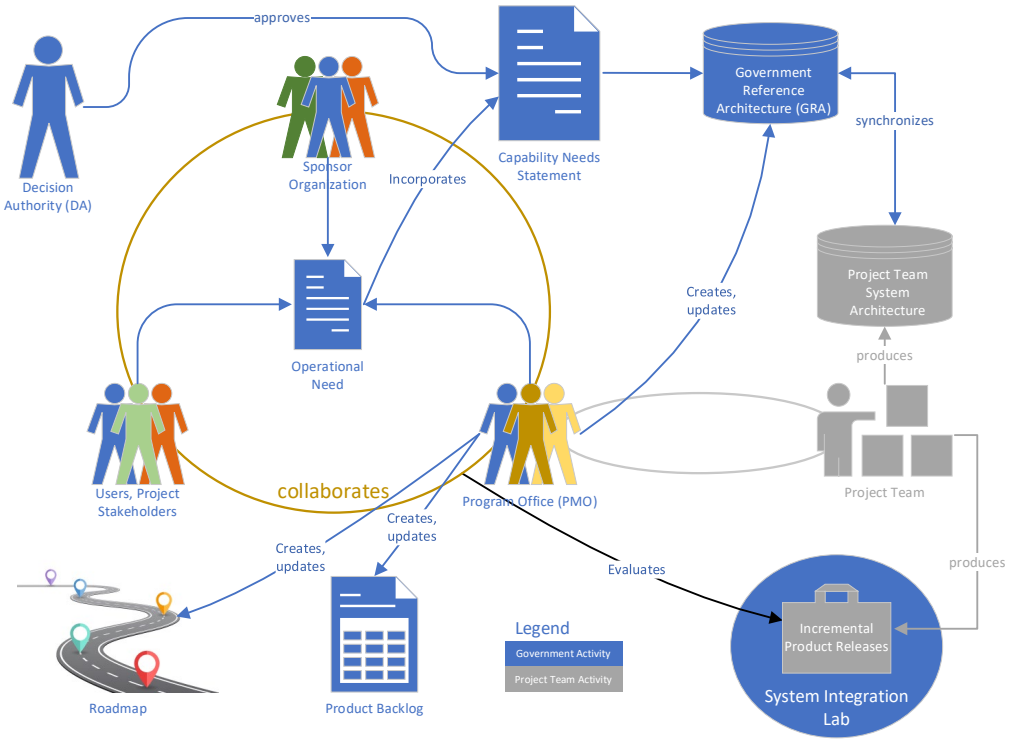


Figure 6: Model-Based CNS Notional Process Overview

For operational needs to make their way into the development pipeline, they need to be communicated to the PMO, and they need to be advocated by the sponsor. This activity implies that there is active collaboration between these process stakeholders. It also implies that there

are trade-offs that can be made *during an acquisition* to accomplish the new operational need without compromising the overall system capability.

To accomplish these trade-offs, the Agile notion of minimum viable capability must be embraced by all collaborators. In the best of worlds, a system is procured within its cost and schedule constraints and meets all of its functional requirements. When new requirements are actively elicited, there must either be a means of increasing the scope of the acquisition, or the new requirements need to be traded off against the existing project scope.

Capability Abstraction

Capabilities are derived from stakeholder needs and refined into requirements. Some capabilities are so fundamental or widely accepted that they exist as baseline expectations even if not explicitly specified. Regardless, as operational needs are identified, an allocation is required to determine how the requirement will be satisfied. This allocation is an iterative and recursive problem, and it gets more complicated as the system matures. For example, the proposed implementation of a specific requirement may add memory or computational load to an already overallocated component, which might cause a new task or require refactoring the component to resolve the overallocation. It's important to know as early as possible what the implementation plan is and what options exist in the trade space.

The accumulation of requirements into capabilities results in the CNS. As stated earlier, the CNS is a living document that is required for a program to be on the SWP. We at the SEI strongly recommend creating the CNS collaboratively by the team members shown in Figure 6.

Model-Based Capability Translation

For simplicity, let's assume that a GRA already exists for the system being acquired. As new operational needs are identified, the PMO can update the GRA, suggesting ways that a new need can be satisfied. Throughout this process, the PMO identifies risks early in the lifecycle, including suggesting possible mitigations (e.g., prototyping, a capability trade-off analysis, alternate implementations).

In principle, the GRA represents all of the system's capabilities and provides bidirectional traceability between the requirements and the specific elements of the GRA (e.g., block diagrams, behavior diagrams, interfaces, sequence diagrams). Although this representation is much more detailed than the CNS, it is important to remember that the linkage between the GRA and the CNS is explicit. The design choices or trade-offs that are in the GRA should be specifically traceable to the CNS.

Capability Definition Validation

The proposed implementation of the operational need, represented in the GRA, is reviewed by the government team. The user validates that the proposed implementation is acceptable or identifies possible gaps in the implementation. The sponsor validates the implementation and can suggest other solutions from other projects it sponsors.

Updating the Roadmap and Product Backlog

As updates are made to the CNS and priorities change, the PMO must update the Roadmap and the Product Backlog. These are artifacts of the Agile project management process, and they should accurately reflect the program's priorities and expectations. The Roadmap is a strategic artifact that represents big targets for the project team, and it is much more static than the Product Backlog. The Product Backlog should reflect tactics and low-level implementation strategies.

Model-Based Acquisition

The PMO provides the GRA to prospective bidders for the system. This is probably done initially through a Request for Information (RFI), but it eventually becomes one of the Statement of Work (SOW) attachments. There are nuances that must be addressed to use a DE model that results in a successful acquisition. For example, the PMO must decide whether the GRA will continue to be updated after the contract is awarded (i.e., the Authoritative Source may transition to the Program Specific Model). The PMO might also require the project team to use the model-based contract attachment as a contract deliverable (the Contract Data Requirements List [CDRL]).

GRA to Program-Specific Model Synchronization

Assuming that the PMO decides to maintain its GRA as the project team iterates its own system model, the PMO will need to update the GRA over the course of the acquisition to reflect the implementation decisions made and constraints identified (and approved by the PMO) by the project team. This work may require revalidating the capability with other project stakeholders, such as users.

The project team's system architecture will likely contain much more detail than the GRA, so some effort and competency will be required to accurately perform the synchronization.

Project Product Validation

In the lower right portion of Figure 6, there is a representation of the delivery of the product into the government System (or Software) Integration Lab (SIL). This delivery is primarily shown for completeness, since programs on the SWP are required to deliver functional versions into the operational environment (or SIL) at least annually. However, project stakeholders can also evaluate the product in the SIL. In this way, they can provide the ultimate validation of the capability or identify gaps that need to be resolved to make the capability acceptable.

Use Case – New Capability

The following use case illustrates the way a digital-engineering-based CNS can be integrated into the normal operations of an acquisition program. As you read the use case and the scenarios, think about the criteria you would use to assess the different implementations. For now, let's assume the following criteria:

- elapsed time for the capability to get to the warfighter (Faster = Higher)
- reliance on PMO technical competency (More Reliance = Lower)
- programmatic risk (i.e., cost, schedule, technical; More Risk = Lower)
- quality of the technical solution (Higher Quality = Higher)

Use Case Context: New Capability

The user community for infantry fighting vehicles has observed an increase in the number of vehicles that have been damaged by undetected SUAVs and needs to deploy a capability that protects the vehicle against this emerging threat. This community communicated this need to the sponsor of one of the PMOs for the ground vehicles, and the sponsor updated the CNS for the vehicle. The CNS update was provided on the normal update cycle. There were no other significant changes in the updated CNS.

The PMO has established a roadmap for the software effort with biannual releases of the platform software into its systems integration lab. The software contractor has adopted an Agile

software development lifecycle with sprints every 2 weeks. The PMO was already aware of this threat and had begun investigating alternatives for SUAV detection on the vehicle.

For this capability upgrade, the PMO assigns a team to investigate the alternatives for detecting SUAVs. The identified options include a modification to a legacy piece of equipment, a new piece of equipment, and a combination of a new piece of equipment and a modification to a legacy piece of equipment.

Scenario 1: Organic Model-Based Engineering

The PMO team established to investigate alternatives for detecting SUAVs includes representation from Lethality, Systems, Software, and DE. The PMO invested in a GRA representation of the vehicle at the outset of the program. The team members were previously involved with the DE group when they initially built the GRA, so they are familiar with the model.

The team updates its model-based vehicle representation to build out various implementation options to support the capability. The options include the following four:

- Option 1: adding a new sensor expressly for the purpose of identifying SUAVs combined with existing lethality options
- Option 2: repurposing an existing sensor to use with existing lethality options
- Option 3: the same as Option 1 for sensing but with a new lethality option
- Option 4: the same as Option 2 for sensing but with a new lethality option

All of these options involve significant modifications to existing workflows on the vehicle, including target prioritization and target tracking. These workflows are elaborated by the team in the GRA, so the modifications can be represented as “red lines” over the existing content. The team assesses the modified workflows for each option using the information from the GRA and builds a table of potential impacts to the existing system. Beyond the size, weight, and power impacts, the team identifies that there may be impacts to the computing infrastructure, as all of the options require a machine learning algorithm with high-speed image processing. The team also identifies that significant impacts to the vehicle’s Tactics, Techniques, and Procedures (TTPs) will be required for some of the options, especially if repurposing existing equipment.

The team updates their assessment of the implementation options to include programmatic information (i.e., identification of cost and schedule budgets and project risks). It brings the GRA and its assessment to the stakeholders, reviews the potential implementations, and gathers information on how to define and develop the GRA representation of the core capability (as opposed to the enhancements). Throughout this process, the team observes that rapidly fielding an initial capability is the highest priority, so it focuses on the implementations that use existing hardware.

While reviewing the impact of the SUAV threat with the user from a mission performance perspective, it becomes clear that if the vehicle were able to share target data with its neighboring vehicles, it would significantly improve its ability to deal with this threat. This peer-to-peer communications capability does not currently exist, so it could not be included in the option set, but it represents a very desirable future enhancement to the counter-SUAV capability. The team integrates the new communications capability into the GRA as an enhancement.

The team works with its contracts team to initiate an RFI to identify possible candidates to implement this new capability and passes the high-level DSM to the candidates to help them decide whether to compete. The team elicits feedback from the candidates, which will be provided as part of its response.

The procurement and implementation will proceed normally, although it is highly likely that the winning offeror will be the one most familiar with the platform and its architecture, as selection of the technical solution will be weighted towards solutions that use existing hardware.

Scenario 2: Using an External Software Architect

This scenario is functionally identical to Scenario 1 (above). The main difference is that the PMO uses an external subject matter expert (SME), a software architect with ground vehicle expertise, to manage the software architecture and update the GRA as required. This is a critical distinction between these two scenarios. Both DE tool-related skills and appropriate software domain expertise are required of the software architect to perform successfully in this role. It's entirely possible that the PMO could outsource the architecture work at the start of the acquisition and aim to transition this role to organic PMO staff as the project becomes more stable.

It's also possible that the sponsor organization (e.g., the Principal Executive Office [PEO]) may be looking to develop an architecture position for its portfolio of programs and may provide an incentive to the PMO for this purpose.

Regardless, the PMO team working on the new counter-SUAV capability develops and investigates the alternatives, working with the external software architect, and proceeds in much the same manner as what is described in Scenario 1.

Scenario 3: Software Architect by Project Team

In this scenario, the PMO relies on the project team to develop the alternatives. This scenario is viable, but it places a burden on the PMO to ensure the identified alternatives are not overly dependent on solutions that favor the project team's technology. This is a natural conflict of interest (COI) for the project team.

The PMO needs to establish a consulting contract with the project team to support the PMO's need for changes to priorities and new requirements. In this way, the PMO can avoid this likely COI, as the project team could staff this consulting contract with a team that is firewalled.

Scenario Discussion

These three scenarios are all valid ways for a PMO to implement a DSM to improve the quality of a CNS. The first two implement a GRA that is maintained by the PMO. The third is a contractor-maintained architecture, which is potentially the same as the system architecture but at a higher level of abstraction.

The first scenario relies on the competency of the PMO staff to identify the technical solution(s) and to provide the DE skills to develop the GRA. Assuming these competencies were properly provided, this approach should get a higher quality technical solution to the warfighter quicker. However, there is some risk, as the project team will need to cooperate.

The second scenario is less reliant on PMO competency but requires a contract action (to get the external software architect tasked to support this). This might add a bit more time when compared to the first scenario. To a certain extent, using an external architect is like what a person might do when implementing a home improvement project. If you don't want to design the improvement yourself, or don't trust your contractor to do it, you use a third party that is knowledgeable to provide an independent perspective.

The third scenario relies on the project team to provide the independent high-level design. This would also require a contract action of some kind to get the project team working on the new capability. The potential impact to the overall project might be higher in this case, as there is the possibility that the firewalled design team might not be as firewalled as we expect,

which increases the likelihood of a COI. There is also a predisposition on the part of a contractor-oriented project team to not be as open about their design solutions as a government-oriented one. There probably are several reasons for this, but this author's opinion is that it is mostly cultural.

Summary

It's a big decision for a program to move its software development onto the SWP. It's an even bigger decision for the program to adopt a DE process to validate its CNS. This section of the paper summarizes the key considerations for the PMO.

The most important aspect of this approach is to accelerate the delivery of capability to the warfighter. We suggest that this is accomplished by maintaining a higher quality and validated representation of the CNS. On the surface, this sounds straightforward. However, the "devil is in the details." This approach adds burden to the PMO, and the process has lots of moving parts. As the system matures, dependencies emerge that need to be managed. If done correctly, the validation process will uncover capability-related risks and issues. When these risks and issues are identified early (e.g., before contract award), the project team has the best chance to mitigate them or make trade-offs with other capabilities.

To effectively use a DSM for this purpose, the PMO needs to keep some basic principles in mind. The PMO must be able to represent and communicate new capabilities within the DSM. This requires some model-based competency with a tool (e.g., Cameo), some specific software domain experience (for the particular software being developed), and the ability to translate the capability implementation alternatives from SysML into a form that the users understand. DE model use by PMOs has been increasing recently, typically because the PMO has required a DSM as a contract deliverable to define aspects of the product design (e.g., requirements, product architecture, component definitions, interfaces). As a result of this need to review the DSM, using DE model tools has become a necessary competency for some PMO staff.

One other point to consider is the amount of information in the DE model. As the DE model evolves, especially when it is under the control of the project team, a lot of details will be added. Most of these details are not necessary to understand the function of the software; they are needed to precisely specify the implementation. This low-level specification causes the model to be less useful for the purpose of interacting with users. The users need simpler (although functionally accurate) ways to envision end-to-end software threads. Keeping the representation simple helps the PMO explain alternative implementations in a way that users understand and can help to elicit critical aspects of the capabilities that may have been missed in non-model representations. The user-oriented model might be a different artifact than the project team model.

This last point raises an interesting conundrum. The implication is that there are two models being used to represent the same thing. This approach violates the DE Authoritative Source of Truth (ASoT) principle, which seeks to advocate a single source of truth as a DE goal. In the case discussed in this paper, the models are being used for different purposes. However, both models need to accurately represent the evolving system. This may be viewed as redundant by the PMO and other stakeholders who might ask, "Why am I paying for two models of the same system?"

No doubt there will be bumps along the way, but we need to remember that the reason why we are recommending this is to get software capability to the warfighter quicker. The enabling process is to improve the quality of capability definitions, so that we have a better understanding of minimum viable capability and can make informed decisions about trade-offs. This is accomplished by employing MBSE principles that transform the CNS from a static

description into an authoritative, model-based view of mission deficiencies, required and optional enhancements, interfaces, and interoperability. This view enables sponsors and users to prioritize and validate their needs iteratively with traceable evidence rather than with documents.

Appendix: Relevant Stakeholders

The following are relevant stakeholders to be considered as part of the processes described in this paper (Software Engineering Institute, 2025).

Decision Authority

The DA is the designated individual who has overall responsibility for a program. This individual has the authority to approve a program's progression through the acquisition process and is responsible for reporting cost, schedule, and performance results. For programs on the SWP, the DA documents the rationale and approves the decision to go on the SWP. In general, the DA approves strategies, reviews progress, and provides go/no-go decisions. In the context of this paper, the DA approves the CNS and reviews/approves the PMO's roadmap.

Program Sponsor

The sponsor is the individual who, or organization that, holds the authority and advocates for needed end user capabilities and associated resource commitments. The sponsor should be part of a collaborative team, made up of the sponsor, users, and PMO personnel, that jointly elaborates operational needs and develops (or updates) the CNS.

Users, Project Stakeholders

The users and stakeholders are individuals who ultimately use the software. A user (or end user) conveys operational concepts, requirements, and needs; participates in capability prioritization and sprint (or increment) demonstrations; and provides feedback on developed capabilities. The users should be part of a collaborative team made up of the sponsor, users, and PMO personnel jointly developing the CNS.

Program Management Office

The PMO should be part of a collaborative team made up of the sponsor, users, and PMO personnel jointly developing the CNS. The PMO needs to ensure the CNS is written at a high enough level to allow trade space during Agile software development. The PMO also ensures that relevant areas (e.g., logistics, training, cybersecurity, certifications) are included in the CNS. The PMO, along with the sponsor, must ensure that any changes in needed capabilities or threats trigger an update to the CNS.

Contractor(s) or Project Team

A multidisciplinary team is typically funded to develop a system, software, or other warfighting capability through a formal mechanism (e.g., contract, agreement, interdepartmental request). This team can be a government team or a contractor/vendor team. It includes roles such as project manager, development team member, system integrator, finance specialist, contracting specialist, and product test and evaluation team member. Depending on the program, this team also might include a product owner who shares that role with or interacts with the product owner in the PMO.

References

Dabney, J. B. (2003). *Return on investment of independent verification and validation study* (Preliminary Phase 2B Report). NASA.

Defense Acquisition University. (n.d.). *Adaptive acquisition framework*. Retrieved March 11, 2026, from https://aaf.dau.edu/

Defense Acquisition University. (n.d.). *Define capability needs*. Retrieved March 11, 2026, from https://aaf.dau.edu/aaf/software/define-capability-needs/

Defense Acquisition University. (2021, May 14). *Capabilities Needs Statement (CNS) template*. https://aaf.dau.edu/storage/2022/06/CNS-Template-14-May-2021.docx

Defense Innovation Board. (2019). *Software Acquisition and Practices (SWAP) study.* DoD. https://media.defense.gov/2019/Mar/07/2002097482/-1/-1/0/SWAP_STUDY_VIGNETTES.PDF

DoD. (2020). *Operation of the software acquisition pathway.* https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF?ver=virAfQj4v_LgN1JxpB_dpA%3D%3D

DoD. (2023). *Digital engineering.* https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500097p.PDF?ver=beePIqKXaLUTK_Iu5iTNREw%3d%3d

DoW. (2026). *Digital engineering and modeling practices.* https://quicksearch.dla.mil/Transient/8606E424EB0D44868358B4367CE254E1.pdf

Feiler, P. H., Goodenough, J. B., Gurfinkel, A., Weinstock, C. B., & Wrage, L. (2013). *Four pillars for improving the quality of safety-critical software-reliant systems.* Software Engineering Institute. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=47791

Hansson, J., Helton, S., & Feiler, P. H. (2018). *ROI analysis of the system architecture virtual integration initiative.* Software Engineering Institute. https://www.doi.org/10.1184/R1/12363080.v1

Hegseth, P. (2025). *Directing modern software acquisition to maximize lethality.* https://media.defense.gov/2025/Mar/07/2003662943/-1/-1/1/DIRECTING-MODERN-SOFTWARE-ACQUISITION-TO-MAXIMIZE-LETHALITY.PDF

Sears, D. (2018, March 8). The sightless visionary who invented cruise control. *Smithsonian Magazine*. https://www.smithsonianmag.com/innovation/sightless-visionary-who-invented-cruise-control-180968418/

Software Engineering Institute. (2025). *Software acquisition go bag user’s guide.* PA: Software Engineering Institute. https://www.sei.cmu.edu/documents/6412/Go_Bag_Users_Guide.pdf

Software Engineering Institute. (2026). *Cracking the CNS code.* https://www.sei.cmu.edu/documents/6465/Cracking_the_CNS_Code.pdf

U.S. Congress. (2017, December 12). *National Defense Authorization Act for Fiscal Year 2018*. Office of the Law Revision Counsel. https://www.congress.gov/bill/115th-congress/house-bill/2810

Warfighting Acquisition University. (n.d.). *Analysis of Alternatives (AoA)*. Retrieved March 11, 2026, from https://www.dau.edu/acquipedia-article/analysis-alternatives-aoa-0

Warfighting Acquisition University. (n.d.). *Capabilities-Based Assessment (CBA)*. Retrieved March 11, 2026, from https://www.dau.edu/acquipedia-article/capabilities-based-assessment-cba-0


Acquisition Research Program
Department of Acquisition, Finance, and Manpower
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net