SYM-AM-25-369

Excerpt from the Proceedings of the Twenty-Second Annual Acquisition Research Symposium and Innovation Summit

Volume III

Digital Engineering, Understanding the Policy and the Engineering in the Minimal Viable Product Approach

Published: May 5, 2025

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

Digital Engineering, Understanding the Policy and the Engineering in the Minimal Viable Product Approach

N. Peter Whitehead, PhD, ME—is a Senior Engineer and Professor of Policy Analysis at RAND, where he conducts research in systems engineering applications & practice, systems analysis, artificial intelligence, cybersecurity, supply chain risk management, and space technology. Prior to coming to RAND, Whitehead conceived of and led an AI at the edge research program at MITRE and supported the DHS HART biometric database program. He was an AAAS S&T Policy Fellow at NSF and OSTP, self-funded his PhD program in systems at UVa, served as Program Manager of an Air Force satellite program at Lockheed Martin, was Test Director of the Air Force DMSP mission sensor, and was a Foreign Service Officer, serving in 60 countries. Whitehead's PhD dissertation in systems developed an AI-supported empirical approach to determining the quality of a systems approach. [nwhitehead@rand.org]

Abstract

This analysis endeavors to clarify the dichotomy of policy and engineering in DoD system acquisition. It considers the Software Acquisition Pathway (DoDI 5000.87) in current DoD policy, approaching that policy from the perspective of good systems engineering practice. It endeavors to provide a bit of guidance on the following: distinguishing policy from engineering – using DoDI 5000.97 Digital Engineering as an example, distinguishing engineering writ-large from software coding, and understanding the importance of working closely with the stakeholder through the minimal viable product (MVP) process. It defines through allusion two distinct flavors (definitions) of MVP – the flavor practiced in commercial industry by many large software companies (systems engineering goal development), and the flavor directed specifically by DoDI 5000.87 (policy). This analysis attempts to show how to use them respectively in the acquisition policy flow and in the systems engineering process.

Summing-up: In developing capability for the DoD, there is a right way, a wrong way, and a policy way, and an acquisition program has to always understand which is in play.

Digital Engineering

The origins of the DoD digital engineering paradigm trace their lineage to the structured software architectures of the 1970s pioneered by such developers as Tom DeMarco and Edward Yourdon.[1] These approaches spawned the concept of computer-aided software engineering tools in the days before recent advancements such as Curser and OpenAI Codex (O'Regan, 2013). In parallel, the paradigm of object-oriented software development evolved to where, in 1995, Grady Booch, Ivar Jacobson, and James Rumbaugh integrated multiple conventions of software engineering and architecture into UML. Their goal was to construct an object-based programming tool where lines of code are replaced with objects, thereby simplifying and expediting the coding process. As it turned out, UML required an extreme level of detail and effort that paradoxically made line-by-line coding more efficient, and so it failed to be adopted for its intended purpose (Bell, 2004; Pandey, 2010). The front-end structured approach to software development was also overshadowed by the Agile approach at the dawn of the new millennium, making architectural frameworks a tool for later documentation but not useful for the new approaches to development (Whitehead et al., 2024).

[1] Systems engineering and the software-based paradigm that became digital engineering diverged primarily through the work of Barry Boehm at USC, the inventor of the systems engineering Vee diagram. Boehm effectively postulated and promoted the assumption that all systems behave like software code, so coding-based systems analyses would be close enough.
The Vee started as a greater than symbol and is wholly derived from software development practices in the 1970s, not established systems engineering practice. See Boehm (1981, 1984).

MODAF and DoDAF

Meanwhile, in Britain, engineers in the Ministry of Defense developed a graphical approach to describing complex systems called the Ministry of Defence Architectural Framework. By the year 2000, this had become the U.S. standard known as DoDAF, and the software tool System Architect was adopted as the industry standard for creating the multi-layered weapon system program perspectives of DoDAF (DoD Chief Information Officer, 2021).

In the early-mid 2000s, a group of software architects, seeing the similarity between DoDAF and the graphical products of UML, created a dialect of UML that they called Systems Modeling Language, thus creating an open-source alternative to System Architect (SysML.org, n.d.). SysML was adopted by a software-centric engineering organization, INCOSE, which coined the term *MBSE* to describe the DoDAF-like architecture use of SysML.[2] In 2006, the Office of the Under Secretary of Defense for Acquisition and Technology, MITRE, Lockheed Martin, Boeing, and others collectively aligned on the MBSE initiative as proposed by INCOSE for system architecture applications in weapon system programs (Hardy, 2006).[3]

Model-Based

The term *model-based systems engineering* might catch some experienced systems engineers off guard, as all systems engineering through thousands of years of practice has been model based, making the term itself sound redundant.[4] Egyptians built scale models of pyramids to study the related mathematics, engineer their construction, and plan the required logistics (Rossi, 2004). Galileo developed mathematical models of the parabolic trajectory of cannon shells that proved to be highly accurate in practice (Naylor, 1976). Bell Labs practiced what Arthur D. Hall (1962) called *systems engineering* and defined it as "organized creative technology and its functions" (p. 3). NASA and military engineers and program managers leveraged thousands of models in successfully putting men on the moon and giving rise to the current perceived value of good systems engineering practice in a complex program (Miles, 1974).

---

[2]MBSE is "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE, 2007, p. 15). See also SysML.org (n.d.).

[3] Hardy (2006) writes:

MBSE enhances the ability to capture, analyze, share, and manage the information associated with the complete specification of a product, resulting in the following benefits:

- Improved communications among the development stakeholders (e.g. the customer, program management, systems engineers, hardware and software developers, testers, and specialty engineering disciplines).
- Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes.
- Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness.
- Enhanced knowledge capture and reuse of the information by capturing information in more standardized ways and leveraging built in abstraction mechanisms inherent in model driven approaches. This inturn [sic] can result in reduced cycle time and lower maintenance costs to modify the design.

[4] Arthur D. Hall describes the origins of the concept that became labeled *systems engineering* at Bell labs in describing the 1940 development of the TD-2 radio relay system: "the name was new, but the functions were not." He traces *systems analysis*, which includes what we will call system goal definition later in this report, to a philosophy developed in the 1940s by the RAND Corporation. Hall adds the terms *systems thinking* and *systems approach* to the list of supporting concepts. These concepts had existed and evolved over millennia, so, while the authors use the term *systems engineering* here, it is used in a general sense to include the supporting concepts and the history of systems engineering predating 1940 (Hall, 1962, pp. 7, 26).

We know from our study of the state of practice in the DoD that the term *MBSE* describes the leveraging of object-oriented architecture modeling, specifically SysML, as derived from the waterfall, object-oriented software development practice of the late 1990s (Hardy, 2006). In other words, it is a 1990s-era software coding tool repurposed through policy to serve as an engineering tool.[5] We have observed SysML used for architecture, system interface, and organizational modeling predominantly, with teams leveraging the tools for other applications as they see fit. We also know that stakeholders across the DoD and the defense industrial base define the details and scope of MBSE differently, so no observation on our part may be considered universal. Further confusion of systems versus software emanates from the IEEE Software Society standards (e.g., ISO/IEEE 15288 and 24641) that are adopted incorrectly by some in the DoD and in the defense industrial complex as systems engineering approaches (Whitehead, 2024).

Studies conducted by a DoD-sponsored university affiliated research center, SERC, in the 2010s worked to leverage MBSE as defined in SysML onto metamodel optimization concepts originated by such researchers as Markish and Willcox (2003) and Kühne (2006). This work led ultimately to the concept of digital engineering as espoused in the 2018 policy document *DoD Digital Engineering Strategy* and DoDI 5000.97*, Digital Engineering (2023)* (Bone et al., 2019; Shyu, 2023).

Distinguishing Policy from Engineering

Based on the above, digital engineering and model-based systems engineering as prescribed in DoDI 5000.97 are policies – they come to us from the top-down, and their use is not supported with empirical evidence. Policies have terms of justification reflecting the lack of evidence, such as "***can*** modernize how the DoD designs, develops, [etc.]" and "***should*** enable faster, higher-quality decision making" (Shyu, 2023, p. 8).

Engineering emanates from the bottom up, driven by design (goals) and applied science – empirical evidence. Momentum *equals* mass times velocity. Increasing pressure *will* reduce the volume of a gas if the temperature is constant.

Confusion may emanate from the tendency of some documents such as DoDI 5000.97 to readily conflate the two, as in this statement:

> *Digital engineering requires planning and providing financial and other resources for digital methods (e.g., model-based systems engineering (MBSE), product life-cycle management, computer aided design) in support of program activities to the maximum extent possible.* (Shyu, 2023, p. 3)

In that statement, *digital engineering*, *MBSE* and *product life-cycle management* are policies; *computer-aided design* is an engineering tool, the use of which is supported with empirical evidence.

Goal Definition in Concept Development – The Engineering

Engineering starts with system goal definition, a highly complex, human-centric endeavor with no direct parallel in digital engineering policy. Goal definition also brings into play the stakeholder interaction known as minimal viable product (MVP).

[5] Coding and engineering writ-large are two very different practices. The computer does as instructed. Code may be complicated, but it does not deal with complexity in the systems engineering sense. Complexity, as dealt with in systems engineering, involves humans, politics, the axiological as well as the applied science and the interaction of often incompatible sub systems. This will be addressed further in the goal development section.

Before initiating the planning phase, the goal definition phase of the program can be the most important phase of any development program (Gibson et al., 2016). This phase examines where the program must functionally go, how to measure progress, who will benefit, and how they will benefit. It provides the foundation for planning the way the system will operate, the path, and the methodologies (Buede & Miller, 2016). In software acquisition, goal definition does not end at any milestone but continues iteratively through the entire system lifecycle. The software is never finished, and goals evolve. Getting the preliminary step of goal definition correct will reduce program risks and streamline both complex and rudimentary aspects of the acquisition process (Whitehead, 2014).

Goal definition defines where to go and how we can tell when we arrive. Engineering necessarily follows with how to get there. Policy puts necessary constraints on engineering.

Identifying Stakeholders

The originating office must not define the system goals in a vacuum, no matter how well they may understand the problem. At the outset of goal definition, they should establish a hierarchical list that includes all of the system stakeholders (Gibson et al., 2016, pp. 55–75). At the top of the hierarchy are the end users, the customers. Next are the entities that support the end-users directly, to include their help desk functions, financial representatives, and the many, varied sources of data and models for their simulations. Next are the enterprise entities that will be responsible for training, cloud operations, future planning, access, security, and integration across the military enterprise. The originating or coordinating office is not necessarily the provider of the facilities or resources needed to make the program go but is the critical center of this and most other following activities.

Elucidating Stakeholder Goals via Scenario Development

In facilitated exercises conducted multiple times, the coordinating office aligns and integrates the goals of all the respective stakeholders into a common set of functional goals.[6] These goals take the form of multiple scenarios describing the use, function, and implementation of the system (Alexander & Maiden, 2005). Systems engineering shows us a litany of approaches for doing this, generally parsed into preliminary surveys, in-person exercises, hotwash, iteration, and final drafts socialized for stakeholder comment.[7] An important principle the coordinating office should adhere to is to focus the process of scenario development on a clear articulation of the desired end state, rather than identifying specific parts of the design and/or specific development approaches (Weinberg, 1982). Instead, the development team must continually refine the development features and process in the (later) planning and execution phases in the context of the stakeholder-informed scenarios (Reis, 2011, pp. 99–113). Figure 1 suggests how a classical system development process—one that does not specify exactly how the desired system will be used and by whom—can ultimately miss the desired end state, the conflict of policy and engineering.

[6] An objective, outside entity could provide neutral facilitation based on systems principles. These functional goals are likely to differ from the conceptual system architecture shown in Figure 1.

[7] Instead of seeking approval from all stakeholders, the coordinating office will likely adjudicate comments using a formal process such as Department of Army Form 7874, the Army-Wide Staffing Comment Resolution Matrix.

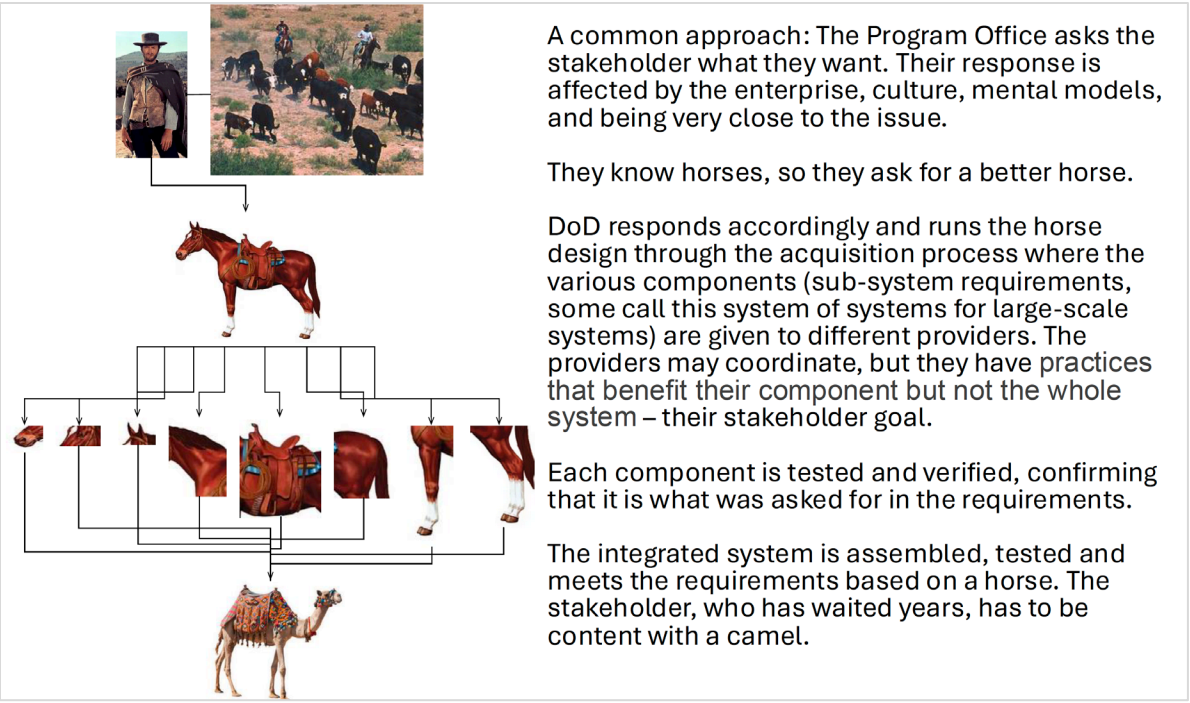

Figure 1. Classical System Development Failure

Scenarios generally have four parts described in plain language or with relatable examples. First, describe the environment where the system will be used, including accessibility hardware and the operational conditions (e.g., in a remote location on a tablet with limited, unclassified connectivity or in a training center in the continental United States with high-speed, classified connectivity).

Second, the users of the simulation are characterized to scope their familiarity with the intended use. This can be achieved with various traits but should be definable and explicit, such as domain-specific training, service experience, and typical operating tempo, among many others.

Third, the immediate user-goals and intended activities are described (e.g., test the simulated effectiveness of a new counter unmanned aerial vehicle system or improve the logistics and timeline of a deployment of armored personnel carriers to Europe).

Finally, the scenario should specify concrete outcomes and/or data expected from the simulation activity, such as gaining skill training, refining a conceptual design, developing or validating novel concepts of operation, planning an upcoming operation, or assessing weapon effectiveness (Alexander & Maiden, 2005).

The set of goal definition scenarios is intended to be as complete as possible, in recognition of the iterative minimum viable product (MVP) process, illustrated in Figure 2.[8] The MVP process in industry is characterized by the phrase, *not like that, more like this,* as options are presented to the stakeholders by the developers (Reis, 2011, pp. 99–113). Each successive hypothetical design is less wrong, iteratively both refining and explaining the mental model of

[8] MVP is often interpreted differently in different circles. The first part of this chapter loosely follows the industry definition of an MVP, one that is developed closer to the goal definition phase and provides the simplest product viable for commercialization. The second part of this chapter focuses on the DoD definition, where iterative products are delivered during the execution phase.

the stakeholder to the developer in terms that they both understand, as in Figure 2.[9] Not only does the creation of end-user scenarios help to refine the initial design, but it establishes stakeholder buy-in for the program, not unimportant in DoD culture. Assumptions, beliefs, and unwritten cultural factors will inevitably have an impact on the development. These axiological factors represent a critical subject for discussion in the goal definition process that should not be ignored.

Note that this is the *first* of the two flavors of the MVP process to be employed during the system development and acquisition process flow. This is the system goal definition phase before the DoDI 5000.87 Capability Needs Statement/planning phase. Subsequently, the DoD flavor of MVP will be employed during the DoDI 5000.87 execution phase.

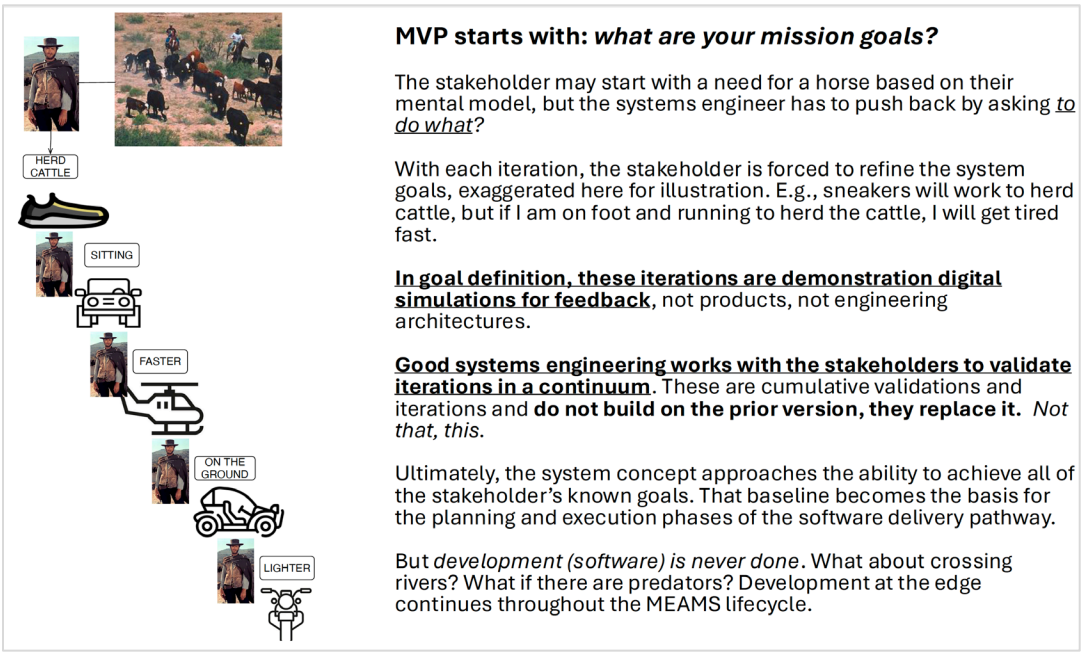

Figure 2. Minimum Viable Product System Development – Commercial Industry Flavor

System Scope

The system scope is expanded by the developed scenarios but bounded by the management triad of schedule, cost, and quality. How much can be afforded, when do stakeholders need it, what are the minimum quality attributes that get the job done for them, and what are acceptable program risks? In the goal definition phase, the coordinating office accumulates these data for planning and costing, including all the logistics and support for deployment and lifecycle operation. Financial bounds have to be a part of the goal definition process to obtain an on schedule and within cost (i.e., viable) end state.

Indices of Performance

Indices of performance (IoPs) represent metrics that relate to the respective goals. These metrics can be technical, descriptive of a process, or concerning the engineering or execution activities themselves. IoPs will be subject to refinement during the MVP process, but the key, longitudinal IoPs should be maintained throughout the lifecycle, and the units must be consistent. Critical aspects of IoPs are that they be measurable, objective, nonrelativistic, meaningful, and understandable to the stakeholders (Gibson et al., 2016, pp. 41–45).

[9] Systems engineers have known this approach for a very long time; see Churchman (1968).

Software Acquisition Pathway – The Policy

Once system goals are well defined, the program advances in the DoD acquisition cycle. DoDI 5000.02, *Operation of the Adaptive Acquisition Framework*, lays out the different pathways that can be used to acquire solutions for end users throughout the DoD (Office of the Under Secretary of Defense for Acquisition and Sustainment, 2022). Figure 1 of DoDI 5000.02 includes software acquisition which is described in detail in DoDI 5000.87 and reproduced in Figure 3.[10] Compared with the other acquisition pathways, which are largely unidirectional and marked by milestone events, software acquisition is iterative, implying that software components must be continuously improved (via MVP iteration and CI/CD) during the entire system lifecycle.

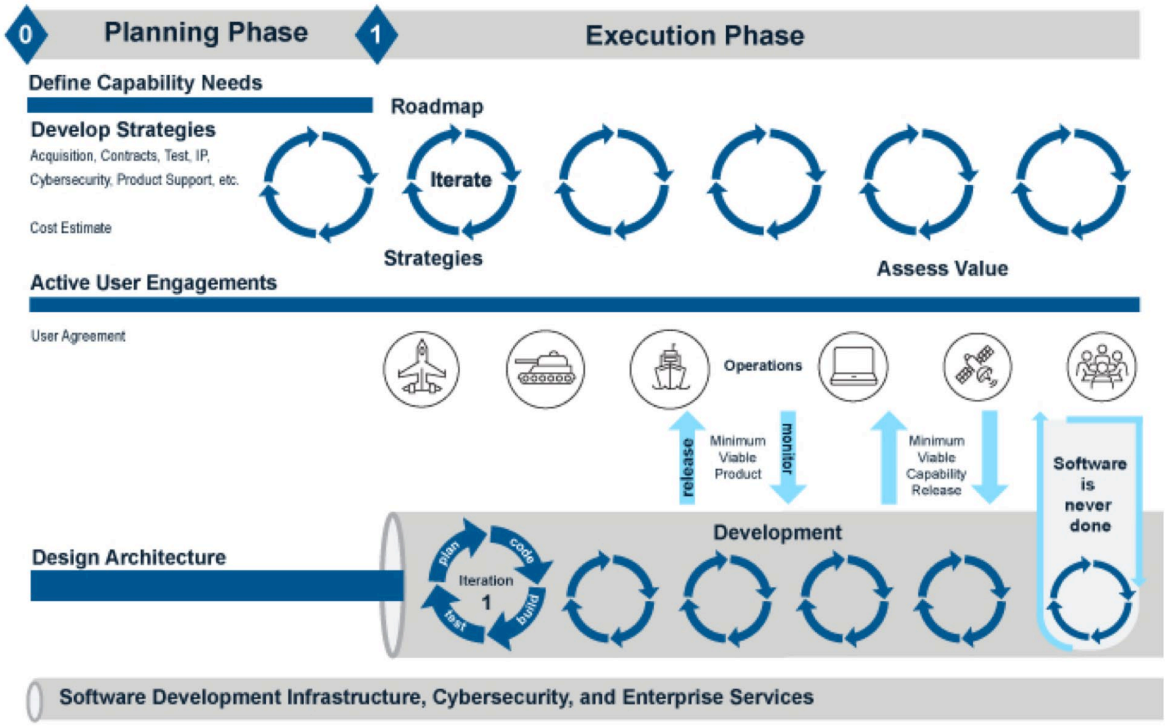

Figure 3. The Software Acquisition Pathway
(Lord, 2020, p. 8)

As shown in Figure 3, the software acquisition pathway is divided into two phases, planning and execution. During the initial planning phase, market analysis is conducted to determine if a commercial off-the-shelf (COTS) software solution to address the system goals is available for purchase. If available, the COTS option must be pursued in accordance with Title 10, Section 3453. If COTS is not available, then the planning phase of the framework proceeds to understand end user needs and establish methodologies to deliver to the users the correct software capabilities (Lord, 2020, p. 9).

Programs using the software acquisition pathway will be identified in competent DoD program lists and databases within 60 calendar days of initiating the planning phase in accordance with the DoD's implementation of Section 913 of Public Law 115-91 on acquisition data analysis (Lord, 2020, p. 10).

[10] The actual process requirements will largely be defined by the overall cost, application, and/or ownership of the final system. This, in turn, will specify the funding source, proponent, and coordinating office.

The Planning Phase

The planning phase of the Software Acquisition Pathway is guided by a draft Capabilities Need Statement (CNS) that is developed by the operational community via the MVP process described above. Through the process, requirements in the CNS are re-prioritized to facilitate effective software development, and user engagement is utilized to update the CNS accordingly. The decision authority, the Under Secretary of Defense for Acquisition and Sustainment, selects the project manager to strategize and govern the software acquisition process (Lord, 2020, p. 9). Software design and architecture attempt to use existing enterprise services as much as reasonably possible. However, this should be guarded by focusing on the system goals discussed earlier in the chapter; planners should realize when the re-use benefit of existing solutions is outweighed by the gaps to goals introduced when forcing alignment. Planning considers and documents in appropriate artifacts a range of factors including but not limited to development environment, automation tools and capabilities, cybersecurity threats, risk-based lifecycle management, testing, and evaluation (Lord, 2020, p. 10). Once the decision authority validates that the appropriate acquisition artifacts are complete and the strategies, analysis, and resources are in place, the process transitions to the execution phase. From planning and through execution, the program develops and tracks metrics of success and keeps cost estimates, costs, and software data reporting up to date.

Other Required Planning Documents

In conjunction with the CNS, a user agreement, acquisition strategy, intellectual property strategy, test strategy, and cost estimate must be approved to transition to the execution phase. The sponsor and program manager must develop a user agreement to ensure commitment, involvement among parties, and delegate decision-making authorities (Lord, 2020, p. 11). Decisions include capabilities defining, capabilities prioritization, software feature trade-offs, software cadence, user acceptances, and readiness for operation deployment. In addition, the user agreement will commit proper resourcing to engage users and create a system for feedback and ways to shape requirement details.

The Acquisition Strategy

The acquisition strategy identifies how to acquire, develop, deliver, and sustain software capabilities for the end users' needs (Lord, 2020, p. 11). Active collaboration between the program manager, program stakeholders, and functional experts ensure the acquisition strategy addresses current environments, priorities, risks, and approaches. The acquisition strategy will be revised by the program manager until it is sufficient for the decision authority to approve development and continue to mature it through the acquisition lifecycle. The acquisition strategy will be approved by the decision authority to include process and documentation tailoring. Key elements of the acquisition strategy are risk-based business and technical management, roadmap and cadence for delivery, flexible and modular contract strategy, planned government personnel and resources, tailoring to use modern practices, high-level test strategies, architecture strategy to enable open modular systems, intellectual property training, product support strategies, and program manager strategy to ensure all is in accordance with law and regulations. If software is embedded, then it must align with the platform acquisition strategy.

The Intellectual Property Strategy

The intellectual property strategy (IPS) identifies and describes the management of delivery and license rights for all software and related material necessary to meet requirements (Lord, 2020, p. 12). The IPS must support and be consistent with government strategies and implemented through requirements in contracts. Rights and obligations of the government and industry should be understood by the program manager to handle strategy and negotiation for software deliverables and license rights. The IPS includes negotiation for and periodic delivery

of software components (Lord, 2020, p. 13). The IPS should address collaboration with other developers and users of software, in the case of government will take delivery and/or modify source code, to reduce duplication. The program manager should attempt to avoid the creation of program-specific versions of software components. Commercial or proprietary software not previously included in the IPS will be approved by the program manager before insertion into software developed for the government. The IPS identifies where intellectual property may result from government investments and treat them appropriately. The program manager should require delivery of all source code at the government's expense and any other requisite documentation. Timelines for delivery should be planned around transitions to new contractors or the government.

Test Strategy

The test strategy defines the process by which capabilities, features, user stories, use cases, and elements will be tested and evaluated to demonstrate if criteria are satisfied. The test strategy identifies the independent test organizations, testing artifacts that will be shared, tools and resources for data collection, and transparency. Tests should assess software performance, reliability, sustainability, and other key metrics. For embedded software, safety assessments and mitigation strategies should be included for any implications to the overarching system. The schedule for embedded software should also align with test and integration for the overarching system. To the extent practical, testing and operational monitoring should be automated for user evaluation. The test strategy should include information in accordance with applicable modeling and simulation policies. The decision authority will approve the test strategy, and the Director, Operational Test and Evaluation, will be the final approver for programs on their oversight list.

Cost Estimate

The cost estimate, in accordance with DoDI 5000.73, *Cost Analysis Guidelines and Procedures*, estimates and considers the technical content of the program described by the other software acquisition pathway required documents. The initial cost estimate must be completed before the execution phase and then updated annually. Where applicable, cost and software data reporting, to include software resources data reports, must be submitted in accordance with DoDI 5000.73 policies and procedures.

The Execution Phase

Software capabilities that correspond with the needs of the end users are developed and delivered during the execution phase. The program assembles components from enterprise services and contracts. Existing connections are preferred to new ones and based on the acquisition and intellectual property strategies. The program maximizes automation of processes related to the project when possible. Consideration should be given for lifecycle objectives and managing technical debt. The sponsor and program office develop and maintain a product roadmap, while the product owner and office maintain a backlog detailing a prioritized list of user needs. The product roadmap and backlog are shaped by continual user feedback.

That continual user feedback takes place in the second iteration of the MVP process, this time during acquisition execution. The program manager and sponsor use an interactive human-centric design process to define the MVP as user needs evolve. If the MVP does not have sufficient capabilities or performance to deploy into operations, then the program manager and sponsor define an MVP release. The MVP release delivers initial capabilities to enhance mission outcomes and must be deployed to an operational environment within a year of the first obligated funds given to acquire or develop new capabilities. Subsequent MVP releases should be delivered at least annually per policy. Through execution, the program should continually update the development process and take user feedback to inform short-term capability

deliveries and long-term solutions. Testing should be guided by risk strategies, and cyber testing and monitoring should be automated to be used to support a conditional authority to operate or accelerated accreditation process.

Cybersecurity policies and assigned authorized officials guide this process. Recurring cybersecurity assessments should be performed on all components of the process. Program managers work with stakeholders to provide controls to enable conditional authority to operate where needed and ensure secure development, cybersecurity and assurance capabilities, and secure lifecycle management. Intellectual property strategy considerations should be marinated through execution. Programs develop and track metrics of success of the program, keep cost estimates, costs, and software data reporting up to date from planning and through the execution phase. The sponsor and user community conduct value assessments at least on delivered software. The results of the assessments inform progress and updates to the process.

Summary/Recommendations

Both policy and engineering impact acquisition programs. Please, never confuse the two.

Understand the limits of software coding tools such as UML and SysML and don't conflate the policy directive to use them in programs with systems engineering practice – despite the confusing verbiage.

The DoD Software Acquisition Pathway includes plenty of space to accommodate a wide variety of sound systems engineering approaches to deliver capability to the warfighter. We have attempted to show one systems engineering approach to doing so via MVP.

Program success will likely depend on the regular engagement of the stakeholders in the development and iteration process via MVP and the budgeting of continuous improvement over the lifecycle of the program.

Bibliography

Alexander, I. F., & Maiden, N. (2005). *Scenarios, stories, use cases: Through the systems development life-cycle*. John Wiley & Sons.

Bell, A. E. (2004). Death by UML fever: Self-diagnosis and early treatment are crucial in the fight against UML fever. *Queue*, *2*(1), 72–80.

Boehm, B. W. (1981). Software engineering economics. *New York*, *197*.

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering,* (1), 4–21.

Bone, M. A., Blackburn, M. R., Rhodes, D. H., Cohen, D. N., & Guerrero, J. A. (2019). Transforming systems engineering through digital engineering. *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, *16*(4), 339–355. https://doi.org/10.1177/1548512917751873

Buede, D. M., & Miller, W. D. (2016). *The engineering design of systems: Models and methods* (3rd ed.).

Churchman, C. W. (1968). *The systems approach* (Vol. 1). Dell.

Defense, D. C. I. O. U. S. D. o. (2023). *DODAF - DOD architecture framework version 2.02.* https://dodcio.defense.gov/library/dod-architecture-framework/

DoD. (2020). *Operation of the adaptive acquisition framework*.

DoD Chief Information Officer. (2021). *The DODAF architecture framework version 2.02*.

Friedenthal, S., Griego, R., & Sampson, M. (2007). INCOSE model based systems engineering (MBSE) initiative. *INCOSE 2007 symposium*.

Gibson, J. E., Scherer, W. T., Gibson, W. F., & Smith, M. C. (2016). *How to do systems analysis: Primer and casebook*. John Wiley & Sons.

Hall, A. D. (1962). *A methodology for systems engineering*. D. Van Nostrand Company.

Hardy, D. (2006). Model based systems engineering and how it aids DoD acquisition & systems engineering. *9th Annual Systems Engineering Conference*.

INCOSE. (2007, September). *Systems engineering vision 2020.*

Kühne, T. (2006). Matters of (meta-) modeling. *Software & Systems Modeling*, *5*(4), 369–385.

Light, T., Younossi, O., Clayton, B., Whitehead, P., Wong, J. P., Pfeifer, S., & Triezenberg, B. L. (2022). *A preliminary assessment of digital engineering implications on weapon system costs*. T. R. Corporation.

Lord, E. M. (2020). *Operation of the software acquisition pathway* (DoD Instruction 5000.87). https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF

Markish, J., & Willcox, K. (2003, October). Value-based multidisciplinary techniques for commercial aircraft system design. *American Institute of Aeronautics and Astronautics Journal*, *41*(10), 2004–2012. https://doi.org/10.2514/2.1890

McQuade, J. M., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software is never done: Refactoring the acquisition code for competitive advantage, final report of the software acquisition and practices (SWAP) study conducted by the Defense Innovation Board*. https://media.defense.gov/2019/Apr/30/2002124828/-1/-1/0/SOFTWAREISNEVERDONE_REFACTORINGTHEACQUISITIONCODEFORCOMPETITIVEADVANTAGE_FINAL.SWAP.REPORT.PDF

Miles, R. F., Jr. (1974, January 15). *A contemporary view of systems engineering*. Jet Propulsion Laboratory.

Naylor, R. H. (1976). Galileo: The search for the parabolic trajectory. *Annals of Science*, *33*(2).

Office of the Under Secretary of Defense for Acquisition and Sustainment. (2022, June 8). *Operation of the adaptive acquisition framework* (DoD Instruction 5000.02, change 1).

O'Regan, G. (2013). Ed Yourdon. In *Giants of computing: A compendium of select, pivotal pioneers* (pp. 277–279). Springer.

Pandey, R. (2010). Architectural description languages (ADLs) vs UML: A review. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 1–5.

Reis, E. (2011). The lean startup. *Crown Business*, *27*, 2016–2020.

Rossi, C. (2004). *Architecture and mathematics in ancient Egypt*. Cambridge University Press.

Shyu, H. (2023). *Digital engineering* (DoD Instruction 5000.97). DoD. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500097p.PDF?ver=bePIqKXaLUTK_Iu5iTNREw%3D%3D

SysML.org. (n.d.). *SysML open source project: What is SysML? Who created it?*

Weinberg, G. M. (1982). *Rethinking systems analysis and design*. Little, Brown.

Whitehead, N. P. (2014). *The dimensions of systems thinking - An approach for a standard language of systems thinking* [Doctoral dissertation, University of Virginia]. https://libraetd.lib.virginia.edu/public_view/3x816m893

Whitehead, N. P., Light, T., Luna, A., & Mignano, J. (2024). *A framework for assessing the costs and benefits of digital engineering*.

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

www.acquisitionresearch.net