Adam Boas
adamboas.com | anboas@gmail.com
July 31, 2026

Digital Engineering Is Not a Department

Connecting systems engineering, software, data, acquisition, and operations around mission decisions.

Digital engineering is easy to misunderstand.

It can look like a new organization. A new tool stack. A model repository. A systems engineering modernization office. A better way to produce artifacts before the "real" acquisition and delivery decisions happen somewhere else.

That version will disappoint people.

Digital engineering creates value only when it becomes part of the mission decision system. Models, software, data, architecture, technical reviews, acquisition choices, test evidence, cyber constraints, and operational feedback have to reinforce each other. If they remain separate lanes, digital engineering becomes another department with better diagrams.

The point is not to digitize engineering artifacts.

The point is to improve mission decisions.

The Artifact Trap.

Every engineering transformation has an artifact trap.

The organization starts measuring whether the new artifacts exist instead of whether they change decisions. The model exists. The architecture view exists. The data standard exists. The dashboard exists. The repository exists. The review package exists.

But the program still makes decisions the old way.

The model is not trusted. The software team is not connected to it. The acquisition team does not use it. The cyber team reviews late. Operators give feedback through informal channels. Test evidence arrives after major choices are already locked. The architecture becomes documentation after the fact.

That is not digital engineering. That is digital paperwork.

DoD Instruction 5000.97 establishes digital engineering policy and responsibilities across acquisition lifecycle activities [1]. The important word is lifecycle. Digital engineering cannot sit in one box because the decisions it should improve happen across the lifecycle.

A simple failure pattern makes this clear: a program builds a model-based architecture repository, but the release backlog, cyber authorization evidence, test plan, and sustainment tradeoffs still live in separate review channels. At the next milestone, leaders receive model views in one briefing and software delivery risk in another. The decision does not change because the evidence never became one decision environment.

What Changes in a Mission Decision System.

A useful digital engineering system starts with the decisions that matter.

What operational outcome is under pressure? What mission thread matters? What decision has to get faster, safer, more resilient, or more accurate? What systems, software, data, people, interfaces,

constraints, and authorities shape that decision?

Once those questions are clear, the model has a job.

It should expose dependencies. It should show where software changes affect mission performance. It should identify integration risk. It should connect data quality to operational confidence. It should make test evidence easier to interpret. It should help acquisition leaders understand what is mature, what is uncertain, and what must be funded next.

The decision system is not abstract. It changes who is in the room, what evidence is required, and when a program is allowed to move.

If an interface change affects mission latency, the model should show the operational consequence before the change becomes a late integration surprise. If a software release changes system behavior, the digital thread should connect that change to cyber posture, test coverage, user feedback, and sustainment risk. If a program wants to scale a prototype, the authoritative evidence should show whether the capability has a transition owner, realistic data access, validated performance, and a path through acquisition.

That is the difference between a model that illustrates the system and a model that changes the decision.

Digital Thread Is Not a Translation Layer.

DoD digital engineering language puts real weight on the digital thread and the authoritative source of truth. That is right, but the words can hide a failure mode.

Many implementations build a digital thread that still functions as a translation layer. The model translates for systems engineers. The backlog translates for software teams. The acquisition package translates for leadership. The test report translates for verification. The risk register translates for governance.

The result is a better-documented version of the old stovepipes.

An authoritative source of truth has to be authoritative for decisions, not merely authoritative for storage. It should reduce hand-carried interpretation between engineering, software, cyber, test, acquisition, and operations. It should make assumptions inspectable. It should make conflicts visible before they become program surprises.

If the digital thread cannot show where a mission requirement, interface, data dependency, software change, test result, cyber finding, and operational observation touch each other, it is not yet a decision system. It is a record system.

Records matter. Decisions matter more.

Systems Engineering and Software Have to Share a Loop.

Defense programs often treat systems engineering and software delivery as different worlds.

Systems engineering manages requirements, architecture, interfaces, reviews, and technical baselines. Software teams work in backlogs, pipelines, releases, user stories, test automation, observability, and incident response.

Both worlds are necessary. The problem is the gap between them.

Software is now a major path by which mission capability changes after fielding. The DoD Software Modernization Strategy recognizes software as central to resilient capability delivery and emphasizes

cloud, DevSecOps, cybersecurity, and workforce as major enablers [2]. That means digital engineering cannot only model the system at acquisition milestones. It has to connect to how software actually evolves.

If a model cannot see software change, it cannot see the mission system.

If a software team cannot see mission-thread, interface, safety, cyber, and acquisition context, it will optimize locally.

The loop has to be shared.

In practice, that means software delivery evidence should update the engineering picture, and engineering constraints should shape the software backlog. Release telemetry, defect patterns, integration failures, cyber findings, and operator feedback should not remain after-action material. They should become engineering evidence.

Data, AI, and Autonomy Belong in the Same Evidence Environment.

Digital engineering also fails when data strategy is treated as a side channel.

Mission decisions depend on data: source quality, latency, lineage, access controls, semantics, telemetry, test results, operational feedback, and authoritative references. A model that cannot connect to the data used by software teams, operators, and acquisition stakeholders will become stale.

This matters even more for AI and autonomy.

The NIST AI Risk Management Framework treats AI risk management as a lifecycle discipline across govern, map, measure, and manage functions [3]. Digital engineering should make those lifecycle questions visible inside the same mission decision system. What is the operational context? What data is used? What assumptions are embedded in the model? What performance evidence exists? What risks are mapped? What changes after deployment? What monitoring shows drift, misuse, or degraded confidence?

AI assurance and digital engineering should not become parallel governance rituals. They should meet in the same evidence environment.

For autonomy, that evidence environment also has to show authority. What can be delegated? Under what conditions? What human judgment remains required? What override path exists? What telemetry proves the system stayed inside its authority boundary? What happens under degraded connectivity?

If digital engineering cannot answer those questions, autonomy governance becomes a policy overlay instead of an engineering discipline.

Acquisition Needs Evidence Earlier, and Incentives Have to Change.

Acquisition decisions are engineering decisions with money and authority attached.

That means digital engineering has to support acquisition, not merely report to it.

Technology maturity, integration risk, software cadence, cyber posture, test evidence, sustainment assumptions, and transition pathways should be visible before a program is forced into a late binary choice. The GAO Technology Readiness Assessment Guide is valuable because it pushes organizations toward evidence-based maturity assessment [4]. Digital engineering should make that evidence easier to create, inspect, and update.

The Software Acquisition Pathway also matters here. DoD Instruction 5000.87 provides a pathway for software-intensive capability that emphasizes iterative development, user engagement, cybersecurity, and continuous delivery practices [5]. If digital engineering cannot support that rhythm, it will become too slow for software-heavy systems.

But this is not only a tooling problem. Programs often keep producing artifacts because artifacts are politically safer than changed decisions. Milestone reviews reward polished packages. Funding lines separate work that should be integrated. Model trust is uneven. Cyber, test, acquisition, and engineering communities carry different incentives. Workforce skill gaps make translation layers feel safer than shared execution.

Digital engineering that matters changes those incentives. It rewards evidence reuse, earlier risk surfacing, shared ownership of assumptions, and decision traceability. It makes the cost of hiding uncertainty higher than the cost of exposing it early.

Operational Feedback Closes the Loop.

The most important digital engineering input often arrives after the system touches reality.

Operators find workarounds. Maintainers see failure patterns. Cyber teams see new attack paths. Software teams see telemetry. Test teams see edge cases. Program offices see where governance slows delivery. Users reveal that the modeled workflow was not the actual workflow.

If that feedback does not enter the engineering system, the model becomes a museum.

Digital engineering should shorten the distance between operational truth and technical decision. That means the system needs feedback mechanisms, not just baseline control. It needs a way to distinguish a local issue from a portfolio pattern. It needs a way to update assumptions without destroying configuration discipline. It needs a way to make evidence reusable across programs.

This is where digital engineering connects to growth and modernization. A company or program that learns across delivery efforts compounds advantage. It stops recreating the same integration analysis. It sees transition barriers earlier. It turns operational evidence into better capture, better architecture, and better investment decisions.

What Good Looks Like.

Good digital engineering has a few recognizable properties.

It starts with mission decisions, not tool adoption.

It makes the digital thread useful for authority, funding, test, cyber, software, sustainment, and operational choices.

It treats the authoritative source of truth as decision evidence, not just a shared repository.

It connects systems engineering and software delivery in one loop.

It treats data lineage, telemetry, and AI assurance as first-class engineering concerns.

It informs acquisition decisions before commitments harden.

It supports iterative software delivery without abandoning system discipline.

It makes cyber, test, safety, and sustainment constraints visible early.

It allows operational feedback to update architecture and investment choices.

It produces artifacts because the artifacts serve decisions, not because the transformation needs

artifacts.

Digital engineering is not a department.

It is a way to make mission systems understandable enough to change responsibly.

When it works, models, software, data, acquisition, and operations stop speaking through translation layers. They become parts of the same decision system.

That is where the value is.

Sources

1. U.S. Department of Defense, DoD Instruction 5000.97, Digital Engineering.
2. U.S. Department of Defense, DoD Software Modernization Strategy, 2022.
3. National Institute of Standards and Technology, AI Risk Management Framework.
4. U.S. Government Accountability Office, Technology Readiness Assessment Guide, GAO-20-48G.
5. U.S. Department of Defense, DoD Instruction 5000.87, Operation of the Software Acquisition Pathway.