The Forward Deployed Engineering Model Is Backward

By: Nick Weir PhD, VP of Mission Engineering

Forward-deployed engineers (FDEs) have become a defining role in modern defense tech. Unlike classic commercial enterprise settings, defense software operates in fragmented, security-constrained environments where customer organizations often lack integrated technical SMEs. In that context, FDEs help connect the technical, operational, and human pieces needed to deliver mission impact.

The success of a few leading defense technology companies has reinforced the idea that scaling FDE teams is the path to success. Today, most defense software companies either employ FDEs or are building organizations modeled after them. This trend is a welcome correction to an industry that has long delivered software disconnected from operational reality.

However, many businesses have drifted from the stated purpose of forward deployed engineering. Instead of using FDEs to bridge the gap between mission needs and product development, they use FDEs as the primary product development mechanism. These FDEs become a crutch for poor product development and user experience. The test when buying AI for mission capabilities is whether FDEs build a bridge to customer-owned capability or become a permanent dependency.

At Legion, we believe that FDEs should improve the product by tightening feedback loops, not by building the product themselves. They should rely on builder tools *within the product* to deliver mission impact, meaning FDE output scales as the product matures. This also makes it easier to achieve mission value *without FDEs*, as customers can use these intuitive, mature builder capabilities themselves. This reduces dependency on vendor staff and makes each successive use case cheaper for customers.

Why Forward Deployed Engineering Exists

National Security software companies face a difficult challenge. Every customer's environment is different. Networks differ, security controls differ, available data sources differ. Even organizations performing similar missions often execute them in fundamentally different ways. Traditional enterprise software companies typically respond by limiting flexibility. They standardize workflows, constrain configuration, and force customers to adapt to the product. Because most enterprise customers have

teams or staff dedicated to integrating software into their business operations, this works, but rarely works in national security environments.

Forward deployed engineers emerged as a response to this reality: By embedding technical personnel directly with users, companies could understand operational requirements in context and rapidly adapt software to support mission execution. Done correctly, this creates a powerful feedback loop: FDEs experience real operational problems first-hand, identify where the product falls short, and product teams improve the system. The next customer benefits from those improvements without requiring the same level of custom work. Forward deployed engineers should accelerate product learning. In many organizations, however, FDEs have become a substitute for product maturity.

Where the Model Breaks

The most common failure mode of the modern FDE model is organizational. As customer demands increase, it is often easier (and beneficial to top-line revenue) to assign additional FDEs to deliver functional prototypes than to improve the underlying product itself. A skilled FDE can solve a problem this week. Building a reusable product capability, then shipping it to customer environments, can take longer. It doesn't help that legacy government contracting models, which are still far more pervasive than newer software-friendly licensing models, generally favor services-based execution.

The immediate results are usually positive, with customers receiving tailored solutions quickly and program milestones met. Engineers build strong relationships with users. However, this comes with a price: the company accumulates a growing inventory of custom workflows, integrations, applications, and operational processes that exist primarily within individual customer environments. As FDEs need to scale capability, they build underlying developer functionality *for themselves*, not based on broader product vision and end user experience. New use cases require additional adaptation, additional support, and additional engineering effort (i.e., more FDEs). The result is a delivery model that scales primarily by adding people.

This has implications for both vendors and customers. For vendors, it drives increasing support burden and growing complexity across deployments (which, in government contracting, isn't always seen as a bad thing, though it should be). For customers, it creates a dependency on the services personnel who understand how the system actually works. In this scenario, there is no path to maturity or eliminating dependency on FDEs.

The Better Solution: Production Engineering Builds and Small FDE Teams Deliver

The real question is where product development ends and forward deployed engineering begins. Legion's view is that forward deployed engineers should build capability within the product using the same tools available to end users, not outside of it.

Perhaps the most important aspect of this model is organizational discipline. Legion believes forward deployed engineering organizations should remain intentionally small, even on large programs. This is not about internal operating costs (FDE costs are generally passed through to customers in contracts, another downside of large FDE teams). It is about the incentives the structure creates. Large FDE organizations allow companies to compensate for weaknesses in their product, whereas small FDE organizations force companies to improve the product itself. Over time, the amount of mission value delivered by each FDE should increase while the amount of direct support required by customers should decrease.

Programs requiring large, permanent FDE presences are often an indication that important capabilities have not been successfully productized. This is because configuration, integrations, new workflow development, and routine operational changes are difficult and require specialized knowledge. No product should operate this way.

Under this model, FDEs remain deeply engaged with users. They identify mission requirements. They still design workflows, configure solutions, and help organizations operationalize capability. However, they do so exclusively through governed product capabilities. When a customer requirement cannot be met with existing platform functionality, the answer is to improve the platform rather than build a parallel solution. The answer is to improve the platform itself.

This creates a deliberate forcing function inside the company. Every meaningful customer requirement becomes pressure on the product organization to expand the platform's capabilities rather than pressure on FDEs to create one-off implementations. Production engineering teams should own platform development, including reliability, scalability, governance, instrumentation, testing, security controls, release management, and platform architecture. They should build the capabilities that FDEs use to solve customer problems. Otherwise, FDEs for each customer operate a separate "shadow" organization with different build requirements, different security posture expectations, and different roadmap, resulting in different customers receiving fundamentally different products. Prospective customers in this situation should carefully evaluate similarities and differences with other deployments before using them as evidence the vendor meets requirements.

When product engineering teams build the system, the product becomes more capable, more reusable, and easier to build on. First, technically adept end users and non-vendor support personnel begin developing capabilities (and not with months of training). Next, the engineering and product team further simplifies configuration, enabling less technical end users to enable themselves. There's less and less FDE support, not more and more. Most importantly, the improvements benefit every customer rather than remaining isolated within a single deployment. This is as it should be when the customers sit within the same business or US Government Department.

Customer-Funded Capability and Product Ownership

Another challenge involves the relationship between customer-funded development and product evolution. In many deployments, government funding supports engineers who build integrations, workflows, data models, mission applications, and other capabilities that extend the underlying platform. Over time, some of the patterns that emerge from this work may be incorporated into the vendor's broader product.

Acquisition leaders should clearly understand the distinction between customer-funded implementation work and vendor-owned product development. What capabilities are being delivered specifically for this deployment? Which of those capabilities are expected to become part of the broader product? What rights does the government retain when customer-funded work informs future product development?

These questions are not unique to the FDE model, but they become increasingly important as more capability development occurs within customer environments rather than within the vendor's product organization.

A Decision Framework for Defense Buyers

Forward deployed engineering is not inherently good or bad. The question is whether it creates a more capable customer or a more dependent one.

When evaluating AI platforms and other software-intensive systems, acquisition leaders should look beyond the size of a vendor's FDE organization and instead evaluate how that organization creates enduring capability. Several questions help distinguish between a scalable software platform and a services business built around software.

Can your own personnel adapt the system?
Determine what changes government operators, administrators, or support contractors can make without vendor involvement. Routine workflow changes, integrations, data sources, and mission applications should increasingly be accomplished using the product itself rather than requiring engineering support.

Does the need for FDEs decrease over time?
Forward deployed engineers should be most valuable during initial deployment and adoption. As the platform matures, the amount of vendor support required to sustain and evolve capability should steadily decline. If every new mission requirement requires additional vendor engineers, the product is not becoming more capable.

How does operational feedback become product capability?
Customer requirements should improve the underlying platform, not remain isolated within a single deployment. Ask how field experience influences the product roadmap, how frequently those improvements are released, and whether future customers benefit from work performed today.

What governs field-developed capability?
Understand whether mission applications, workflows, and integrations are built using governed platform capabilities or through one-off engineering inside the customer environment. Platform-based development is easier to secure, test, maintain, and upgrade, while ad hoc implementations create technical debt and long-term support risk.

Can capability move with the mission?
Capability developed for one organization should be portable to another. Workflows, applications, and operational knowledge should transfer across units, commands, security domains, and deployment environments with minimal rework. If capability remains tied to the engineers who built it or the environment where it was first deployed, the organization is accumulating services, not software.

The answers to these questions reveal the trajectory of the platform. One model produces increasing vendor dependence, growing services costs, and fragmented capability across customers. The other produces reusable software, lower sustainment costs, and customers who become progressively more self-sufficient.

The objective of forward deployed engineering should not be to maximize the number of engineers supporting a deployment. It should be to maximize the amount of mission capability that remains after those engineers leave.

Continue the series

Follow Legion Intelligence on LinkedIn: linkedin.com/company/legion-intel

Explore all Command Papers: legionintel.com/command-papers


LEGION
INTELLIGENCE