The Unfinished Weapon

Why Our Most Advanced Systems Fail Without Adaptable Software

AUTHORED BY:

Bryon Kroger, Founder & CEO
Rise8
growth@rise8.us

RISE8, INC.
601 S HARBOUR ISLAND BLVD.
STE 109
TAMPA, FL 33602

Executive Summary

Modern conflict is defined by speed, complexity, and unpredictability. Yet while our platforms grow more advanced, our ability to adapt them in real time lags behind. The result: our most sophisticated systems often arrive on the battlefield incomplete—not because of poor design, but because the software running them is frozen in time.

In a fight where every second counts, adaptability isn't a luxury. It's a warfighting requirement. Software that isn't built to adapt becomes a liability.

This paper explores the strategic imperative of continuous software delivery and proposes a model that enables mission-ready software to be shipped, updated, and improved at the speed of relevance.

The New Theater of Dominance

We no longer win by being the biggest or even the fastest. We win by being the most adaptable in conflict. Dominance is no longer decided in planning rooms or procurement timelines. It's decided in active conflict, where threats evolve by the hour and advantage belongs to the force that can learn and adapt faster.

Software is now a primary vector of that advantage. It must be treated like a weapon system built for rapid iteration, tested in production, and continuously improved based on live feedback and mission context.

No software will survive first contact with the enemy.

Static software is a death sentence. Without the ability to adapt mid-mission, even our most exquisite platforms risk obsolescence.

Speed Is Not Enough: The Case for Efficacy Over Efficiency

DoW software efforts often chase cost and schedule efficiency. But failing to perform can never be efficient. Efficacy must come first.

Cheaper, faster delivery of ineffective software is just waste. Dominance in active conflict demands continuous software delivery that improves the mission, not just clears a backlog or delivers to an already outdated plan.

Why Systems Fail: The Adaptability Gap

Fielded hardware often outpaces the software intended to enable it. The result is a widening gap between capability and usability. The problem is both technical and institutional. From rigid acquisition timelines, to siloed teams, and outdated contract structures, the system reinforces the delivery of static products over evolving capabilities.

The consequence is an "unfinished weapon:" a system deployed to the field with software that lacks the flexibility to adapt in response to the adversary. The result is operational risk.

The Mission O/S Model

To compete in modern conflict, we must re-architect how we deliver software. Winning demands a new operating model. Mission O/S is a framework to deliver software as a continuously evolving weapon based on three non-negotiable imperatives:

1. Continuous Delivery to Production (first)

Before we can build GOTS, or safely evaluate COTS, we need the ability to ship software quickly, safely, and sustainably to production. Too many teams celebrate demos. But real users don't operate in demo environments. They operate in production.

The production environment is the only place to truly validate assumptions, measure performance, and learn at mission speed. Continuous delivery affords us:

- Frequent, incremental, secure, releases
- Testing with real users, under real conditions, on real missions
- Risk management through iteration

This isn't just technical practice. It must be a warfighting competency: shrinking the feedback loop to accelerate evolution. Every release is an opportunity to validate assumptions, course-correct, and strengthen the system in context. You either learn in production or you fall behind.

2. Mission-Integrated Teams

To deliver software with mission impact, software teams must be embedded in the mission—not orbiting around it via stakeholder meetings or quarterly reviews. Integration requires co-location for shared context and continuous collaboration.

Mission-integrated teams:

- Understand the user's operating environment, pressure points, and language
- Build empathy that drives urgency and relevance
- Are mission-obsessed and respond to operational change in real-time

Integration collapses the distance between problem discovery and solution delivery. When delivery teams see and feel the impact of their work, they move faster, fix smarter, and build better.

3. Outcome Ownership

Teams aren't just responsible for delivery; they're also accountable for mission impact. That means tracking what happens after release, not just hitting deadlines or clearing backlogs.

Owning outcomes requires:

- Stable, long-lived teams with historical context and decision-making authority
- Cultural investment—mission-focused incentives and feedback
- Production-based metrics that surface real operational performance (see: Measuring What Matters), not just IT metrics

Outcome ownership builds resilience. When teams are responsible for results, they can't shift blame so they solve problems. If teams measure success by compliance or velocity, they optimize for that. But when impact defines delivery success, they'll align around everything that matters most and build what the mission actually needs, in the way it needs it.

Why Continuous Delivery Reduces Risk

Big bang releases carry big unknowns. Frequent, incremental updates de-risk delivery by exposing failure points early, validating assumptions, and allowing for rapid course correction to keep the system mission-aligned. Continuous delivery isn't riskier—it's how we manage risk responsibly.

From Output to Outcome: Measuring What Matters

Success must be measured by outcomes in production. We define outcomes as changes in user behavior that produce mission impact: lives saved, hours returned, faster decisions, mission acceleration. Traditional IT metrics like story points or feature counts are insufficient. The mission must be the source of the one metric that matters at any given time for a team:

- **Mission Impact Metrics** – Custom KPIs tied directly to operational goals—e.g., targeting accuracy, decision time reduction, hours of manual input eliminated.

Things like continuous delivery metrics still matter, alongside other product metrics, but only in so much as they contribute to rapid mission metric improvements:

- **Deployment Frequency** – Are we delivering value often enough to keep up with the mission?
- **Lead Time for Change** – How quickly can a new idea or fix go from concept to production?
- **Mean Time to Recovery (MTTR)**– How fast can we detect and recover from failures?
- **Change Failure Rate** – Are we learning fast without breaking the mission?
- **Reliability**– Do our systems perform as intended over time, under pressure?
- **Time to First Outcome** – How long does it take for the first user to gain value from the system?
- **User Satisfaction (e.g., Net Promoter Score)**– Are we building tools that users love and rely on?

The latter are grounded in DevOps research (like DORA) and are increasingly used in high-performing software teams across all sectors, including defense. Strong performance across these indicators correlates with faster delivery, fewer failures, and more mission-critical outcomes and impact. But at the end of the day, we have to measure the mission impact.

Reclaiming the Software Factory

The software factory model works, but only when it's anchored in continuous delivery and mission alignment. The goal of software factories was to deliver better outcomes faster, safer, and more aligned to mission needs. Mission O/S offers the model for execution.

Mission O/S isn't a platform or tech stack. It's a framework anchored in continuous delivery, enabled by mission-integrated teams, accountable to outcomes in production. Many efforts have drifted toward platform obsession and innovation theater, building complex internal tooling no warfighter will ever touch. Mission O/S shifts this focus. To reclaim its original promise, the software factory must:

- Stay focused on delivering mission impact
- Adopt commercial tools for commodity capabilities (below the value line)
- Compete between contracts (in production), not just for contracts
- Avoid DIY in areas where commercial software outperforms government-built alternatives

Mission O/S recenters delivery around what matters most: working software, in production, delivering real impact. It's how "factories" stop producing theater and start delivering complete weapons systems.

Creating a Culture of Competition and Learning

The best solutions don't win in pitch decks. They win in production.

Instead of competition for contracts, imagine competition between contracts—in production. Multiple teams solving the same problem measured by real-world outcomes: the changes in human or system behavior that produce mission impact.

This surfaces the best solution faster and fosters a culture of learning, accountability, and innovation.

This may seem inefficient, but it applies rigor toward efficacy. Pouring money into a perfect spec that fails in conflict is the real inefficiency. Bad ideas can't hide in production like they do in powerpoint.

Conclusion: Lethality Demands Continuous Adaptation

Modern warfighting requires systems that can adapt mid-mission. Without continuous delivery of software that evolves in conflict, even the best hardware becomes static and ineffective.

Built-to-adapt isn't a tech buzzword—it's a strategic requirement and the capability that enables all others. To win the next fight, we must equip our warfighters with systems that learn, ship, and adapt at the speed of mission needs.

Because if we ship software that can't change, we're not fielding a capability. We're fielding unfinished weapons that will fail missions and get people killed.

Contact:
For questions or to learn more about how Rise8 enables continuous delivery of mission-impacting software, contact **growth@rise8.us**