OPERATIONAL READINESS NEWSLETTER

When Training Ends: Integrating AI into the Live Fight

The procedural shifts required for human-machine teaming in complex combat environments.

AI Integration · Operations Doctrine · Human Factors · Combat Systems

> The newsletter for operational commanders who prefer validated doctrine over expensive assumption.

Diagnosis: The Gap Between the Test Bench and the War Zone

There is a profound difference between a piece of technology that works in a controlled environment and an operation that succeeds under pressure. In many high-tech programs, we mistake "capability" for "integration." We celebrate when a drone successfully executes a flight path or processes a sensor feed on the test bench because it proves the hardware functions as designed. But having a functioning tool is not the same thing as having a usable capability.

The failure point in most modern integration projects isn't that the machine fails; it’s that the human operator doesn’t know what to do with the information the machine provides when things get loud and fast. We often build sophisticated systems but fail to build the "manual" for how those systems interact with human decision-making.

In my experience, this is **The Capability Gap**. It occurs when a team assumes that because the AI can identify a target or suggest a route, the operator is now automatically equipped to act on it. They aren't. Without a defined procedure for how that data enters the command loop—who owns the decision, what

are the limits of its accuracy, and at what point does the human override take over—the technology remains an isolated island. We must stop measuring success by whether the machine "works" and start measuring it by whether the operator can confidently act on its output without hesitation.

Why Integration Fails: Process Drift vs. Component Failure

When a system fails in the field, the first instinct is to blame the hardware. If an AI gives a bad recommendation or a drone loses connection, we look for a bug in the code or a flaw in the sensor. While those are real issues, they are rarely why these programs fail to scale. More often, the failure is caused by **Process Drift**.

Process drift happens when the "how" of the operation degrades because it wasn't codified for reality. In the lab, everyone knows what to do because there is no ambiguity. On the line—or in the cockpit—ambiguity is constant. If a pilot receives an automated alert but has to wait three seconds to figure out who has the authority to act on it, those three seconds are a failure of process, not technology.

The Comforting Rationalization	The Underlying Reality
"The AI's algorithm is unreliable."	The human doesn't have a clear protocol for handling low-confidence data.
"The communication link dropped."	There was no pre-planned, rehearsed fallback procedure for a lost connection.
"The operator was overwhelmed."	The decision tree was too complex to execute under high cognitive load.

We don't have a hardware problem; we have a translation problem. We are trying to take technical capabilities and turn them into operational habits. If the instruction is buried in a 200-page manual, it doesn't exist during a crisis. It must be burned into the daily routine until it becomes an automatic reflex.

The Three Operational Tests for New Tech Adoption

To bridge the gap between a successful test and a successful mission, we have to move past "theoretical readiness." A piece of technology is only ready when it passes three specific tests of human-machine interaction. We must stop asking if the machine can do its job and start asking if the team knows how to work with it.

1. **The Trust Test:** Does the operator know exactly what the system *cannot* do? Reliability isn't about 100% accuracy; it’s about knowing the boundaries of the tool so the human doesn't over-rely on it or ignore it out of fear.
2. **The Authority Test:** When the machine provides a "high-priority" alert, who makes the final call? If there is any doubt in the chain of command regarding who has the authority to act on automated data, the system will fail during a high-stress event.
3. **The Recovery Test:** What happens when the technology fails? We must have an immediate, practiced "fallback" mode. If the AI output becomes erratic or the link goes dark, the operator

shouldn't have to think about what to do next; they should simply pivot to the pre-planned manual override.

Making the System Work: From Program Plan to Pocket Procedure

The jump from a "Program Plan" (what we want it to do) to a "Pocket Procedure" (what the user actually does) is where most projects die. A program plan lives in a binder; a pocket procedure lives in the operator's head and on their checklist. To make this work, you have to strip away the complexity until only the essential actions remain.

First, define the **Action Trigger**. Don’t give the user a menu of options; give them a "if X happens, do Y" instruction. If the AI identifies a target with 80% confidence, the procedure should clearly state what that means for immediate action. Second, establish **Decision Boundaries**. Clearly mark where the machine's role ends and the human's responsibility begins. These boundaries must be visible to everyone involved in the chain of command.

Finally, you must move from simulation to "stress-testing" the procedures. You don't test a procedure by seeing if it works when everything is going right; you test it by intentionally breaking parts of the system during training. If the operator can still execute the mission when the data feed is garbled or the primary tool fails, then—and only then—is the process integrated. We want to move from "the technology is working" to "the team knows how to fight with this technology."

Immediate Action Items for Operational Leaders

If you are overseeing a transition to newer systems or integrating complex data feeds into your current workflow, do not wait for the next software update to fix your process. Start implementing these three checks on your next site visit or during your next briefing:

1. **Audit the "Action Gap":** Pick any piece of automated output currently used by your team. Ask the operator at the front line: "If this screen shows an error right now, what is the very first thing you do?" If they have to ask a supervisor for permission or look up a manual, your process has failed.
2. **Simplify the Checklist:** Take the current standard operating procedures (SOPs) for new equipment and cut them by 50%. Remove the "why" and keep only the "how." Create a simplified checklist that can be read and acted upon in under ten seconds.
3. **Mandate Failure Drills:** Schedule a training session where you intentionally disable the high-tech features of your primary tool during an exercise. Force the team to rely on their manual fallback procedures. If they struggle with the transition, it means the integration isn't deep enough yet.

The goal is not to make the technology more sophisticated; it is to make the human role clearer. We want a system where the machine handles the complexity so that the person can focus on the decision.

Call to Action

What non-technical procedure has caused the biggest operational headache in your industry? Share this with a colleague who needs to think beyond hardware specs.

Newsletter replies: newsletter@sixsigmakaizen.com · X: @kaizen_6sigma

Stay curious, stay improving,

David Rodgers
SixSigmaKaizen.com

References: US combat drones move closer to real-life missions with Air Force’s Nevada exercise (interestingengineering.com)