OPERATIONAL SYSTEMS THINKING

Integration isn't about moving data. It’s about shared operational context.

Why connecting your systems is not the same as fixing your supply chain process.

Supply Chain · Digital Transformation · Operations Management · Governance

> The newsletter for VPs who prefer operational context over technological complexity.

The Hard Truth: Your Systems Are Talking to Each Other—But Not To You.

I have sat in many meetings where IT teams and executives celebrated "successful integrations." They would point to beautiful dashboards, automated data feeds from the warehouse management system (WMS) into the ERP, and high-speed API connections that replaced manual entry. On paper, it was a masterpiece of modern engineering.

Then I walked onto the floor.

I watched an operator at a packing station stare at a screen where a "system error" wouldn't let them process a shipment because the inventory count was off by three units. The system and the warehouse software were talking to each other perfectly—they had exchanged the data packets flawlessly. But they weren't telling the operator what to do next. They didn't provide an escalation path, a reason for the discrepancy, or a clear instruction on whether to wait for a supervisor or override the count.

Integration is not just about moving bits and bytes from point A to point B; that is merely plumbing. True integration is about creating shared operational context. If your software "talks" but your people

are still left guessing when something goes sideways, you haven't integrated your process—you've only automated the confusion. You have replaced a manual data entry error with an automated information gap.

The goal isn't to make the computers work better together; it is to ensure that when a reality on the floor deviates from the plan in the office, the system provides enough context for a human being to make a correct decision immediately. If you have to pick up a radio or call a manager every time "the system" hits a snag, your integration has failed at its most basic purpose.

What is Actually Happing in Fragmented Supply Chains?

The primary reason these projects fail to deliver actual results is **The Integration Illusion**. This is the belief that because two pieces of software are connected by an API or a middleware layer, the underlying business process has been "solved." It isn't. You have merely obscured the cracks in the foundation with a fresh coat of digital paint.

In many cases, companies treat data transfer as the end goal rather than just a prerequisite for operational flow. They assume that because the information is moving faster, the decision-making will become easier. But if the person receiving the data doesn't have the authority to act on it, or doesn't understand the "why" behind the numbers, the speed of the data is irrelevant.

To see this clearly, look at the difference between what leadership thinks they are buying and what actually happens when a shipment is delayed:

The Comforting Rationalization	The Operational Reality
"The system will automatically alert us to inventory discrepancies."	A red light flashes on a screen, but no one knows who is responsible for calling the supplier.
"Automated data flow eliminates manual errors."	Data flows perfectly, but because it's entered into two systems at once, nobody knows which one is the 'source of truth' during a conflict.
"Integration creates a seamless supply chain."	Information moves instantly, but since there are no defined decision rights, the floor team waits for an email from HQ before they can move a single pallet.

When we ignore these realities, we end up with what I call **The Resolution Gap**. The data is present, but the "next step" is missing. You have a high-tech way of identifying a problem without a low-tech plan to solve it.

The Three Missing Pillars of Modern Integration

To move past the integration illusion, you must look beyond the connection and focus on what makes that connection useful for someone with dirt on their boots or grease on their hands. True integration requires three pillars:

1. Governance (The Rules of Engagement)

Governance isn't a corporate policy document; it’s the answer to the question: "Who decides?" When two systems disagree—for example, if the warehouse shows 50 units but the sales order only allows for 40—the system must have a pre-defined rule. Who has the authority to override? What is the limit of that authority? Without governance, your team will default to "calling the manager," which creates a bottleneck and slows down everything else.

2. Workflow Orchestration (The Map)

Orchestration is the sequence of events triggered by data. If an order is delayed in transit, does the system automatically notify the production floor? Does it adjust the labor schedule for the next shift? Integration means that one piece of information moving through the pipe triggers a series of related actions across different departments without manual intervention. It’s about ensuring the "next step" is always visible and actionable.

3. Human Ownership (The Accountability)

This is where most IT-led projects fail. You can have the best software in the world, but if no human being owns the exception handling, the process will eventually break. There must be a designated person or team responsible for "cleaning up" the data when it hits a snag. If an automated alert goes unacknowledged because "it's just a system notification," you don't have an integrated system; you have a digital junk drawer.

How to Design for True Operational Context: The 5 Steps

If you are currently struggling with fragmented systems that won't talk, do not start by buying more software or hiring more consultants. Start by mapping the reality of your floor. Use these five steps to build an integration that actually works for the people doing the work.

1. Map the Physical Handoffs

Before looking at a single API, walk the line. Identify every point where a physical item moves from one person's responsibility to another. At each handoff, ask: "What information does the next person need to do their job?" If they have to stop and call someone else to get that info, you have found a gap in your operational context.

2. Define Decision Rights

For every piece of data entering your system, define who is allowed to change it and under what conditions. Create an "exception matrix." For example: *If the order is X amount or less, the floor lead can approve a substitution; if it is over X, it requires supervisor approval.* This removes the need for constant "permission-seeking" during peak hours.

3. Identify and Document Exception Paths

A perfect process doesn't exist in manufacturing. You must plan for when things go wrong. For every automated step, write out a manual "fallback" procedure. If the internet goes down or the data is corrupted, what is the specific set of steps the operator takes to keep moving? This should be taped to the machine or printed on a placard, not buried in a digital manual.

4. Standardize the Trigger Points

Determine exactly what event triggers an action. Instead of "the system tells us when we are low," use "when the count hits 10 units, the system automatically generates a restock request and notifies the purchasing clerk." Be specific. A trigger must be a clear signal that demands a specific response.

5. Validate via Gemba Walks

Once you have designed the new flow, don't check it from your office. Go to the station where the work happens. Watch an operator try to execute the process as it is written on paper. If they hesitate, if they look confused, or if they reach for their phone to call someone for help, the integration isn't finished yet. You haven't provided enough context.

Practical Takeaways: What You Can Start Doing Tomorrow

You don't have to overhaul your entire IT infrastructure by Monday morning. However, you can begin closing the gap between "data movement" and "operational context" immediately with these steps:

1. **Audit Your Alerts:** Look at the top five most common "system alerts" your team receives. For each one, ask: "Does the person receiving this know exactly what they are supposed to do next?" If the answer is anything other than a clear 'yes,' write down the missing step and communicate it to the team.
2. **Create an Exception Log:** Instead of just fixing problems as they arise, keep a log for one week of every time an employee has to "stop" work because they didn't have enough information from the system. Use this list to identify where your integration is failing to provide context.
3. **Define One "Decision Right":** Pick one common friction point—like minor inventory discrepancies or shipping delays—and formalize a rule for it. Give the floor staff a clear boundary of what they can approve without calling a manager.
4. **Map One Handoff:** Choose one specific transition (e.g., from packing to loading) and map out exactly what information must be present at that moment. If that information isn't currently in your system, work on getting it there.
5. **Stop Calling It "Integration":** When talking with your IT team or vendors, stop using the word "integration" as a catch-all. Start using the term **"Operational Context."** Force them to talk about how the data helps the person at the machine make a decision, not just how it moves between databases.

Call to Action

What single process boundary in your operation requires true integration? Share this if it helped clarify the difference between connectivity and operational capability.

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

Stay curious, stay improving,

David Rodgers
SixSigmaKaizen.com

References: logisticsviewpoints.com/2026/07/24/why-supply-chain-modernization-is-increasingly-an-integration-program/