What 35 Years in Technology Taught Me About How Organizations Really Work

Jeff Finlay | Purple Unicorn Coaching

I Did Not Start With a Framework

I did not begin with a model. I began with a keyboard, a codebase, and a set of problems that needed solving. My first technology role was as an intern, and I was already developing a habit that would define the next three decades: I was becoming indispensable.

My manager at the time pulled me aside and said something I have never forgotten. He called it the beer truck problem. If Jeff gets hit by a beer truck tomorrow, can this company continue? He was not being morbid. He was being precise. What he was naming was the structural risk of concentrated knowledge, the fragility that forms when one person becomes the load-bearing wall of an organization's technical operations.

I was an intern when I first heard that framing. I carried it with me for 35 years.

What followed was a career spent inside technology organizations of different sizes, industries, and levels of maturity. I was an engineer. Then an architect. Then a leader. I built systems, inherited systems, and watched systems fail in ways that were not supposed to be possible. I firefighted. I scaled. I got the 3 AM call. I made the 3 AM call.

None of that experience started with a framework. It started with pattern recognition. Over time, I began to notice that the same structural behaviors were repeating themselves across organizations that had nothing else in common. Different products. Different industries. Different leadership teams. The same pressure responses. The same failure modes. The same invisible risks hiding beneath apparent success.

At some point, pattern recognition stopped feeling like intuition and started feeling like evidence. That is when I began building a formal system to describe what I was seeing.

The Patterns Were Structural, Not Personal

Early in my career, I made the same mistake most technical leaders make: I attributed organizational problems to people. A team was struggling because someone was not stepping up. Delivery was slipping because someone made the wrong call. A platform was fragile because someone had written bad code.

Those attributions were not entirely wrong. But they were systematically incomplete.

What I eventually learned to see was the architecture underneath the behavior. Strong teams struggled not because they lacked capability but because they were operating inside structures that could not scale. Talented engineers were blamed for friction that was actually baked into how decisions moved, or failed to move, through the organization. Heroics that looked like competence were actually concealing fragility. Governance processes that felt like discipline were slowing momentum without producing better outcomes. Leaders who appeared effective in isolation were quietly creating bottlenecks at scale.

I watched teams burn out chasing fires that the system itself was generating. I watched organizations blame individuals for conditions that individuals did not create. I watched executives misdiagnose structural problems as talent problems and then spend years solving for the wrong variable.

The turning point was recognizing that these were not random failures. They were predictable. The same patterns produced the same outcomes with near-mechanical regularity. That regularity meant they were structural, not personal.

The behavior was a symptom. The structure was the cause.

Once I saw that clearly, I could not unsee it. And it changed how I diagnosed every situation I walked into afterward.

Recognizing the Archetypes in the Wild

Before I had formal names for what I was observing, I was already recognizing the operating modes.

I worked in organizations where one person, sometimes myself, was the answer to every critical question. Production issue? Find Jeff. Architectural decision? Find Jeff. Customer escalation touching the platform? Find Jeff. The organization functioned. It even felt fast. But the velocity was entirely dependent on a single point of concentration. When that person was unavailable, or overwhelmed, or eventually burned out, the fragility became visible all at once. The Hero-Driven Organization does not announce itself as a risk. It announces itself as a high performer, right up until it does not.

I worked in other environments where decision-making was centralized by design. One leader, one approval path, one architectural voice. This produced clarity in the short term and rigidity over time. The organization knew where authority lived, but it could not scale without that authority becoming a bottleneck. I have sat on both sides of that dynamic, as the leader who was the bottleneck and as the engineer waiting for a decision that should not have required my manager's manager to make.

I saw organizations where the engineering culture had developed a kind of aesthetic reverence for technical elegance that progressively decoupled the technical work from the business it was supposed to serve. Beautiful architecture. Slow delivery. A widening gap between what the engineers were building and what the organization actually needed. The craft was real. The connection to outcomes had weakened.

I saw organizations where alignment had become the primary activity. Where the answer to any pressure was another meeting, another working group, another round of consensus-building that deferred accountability without resolving it. Decisions happened, eventually. But the cost of the process was often higher than the cost of the decision itself.

And, occasionally, I worked in organizations that had figured out something different. They were not frictionless. They were not perfect. But they had developed a structural capacity to learn from what was happening, to surface constraints honestly, and to make decisions that were connected to actual tradeoffs rather than politics or habit. Those organizations behaved differently under pressure. They did not fragment. They adapted.

I did not have a taxonomy for these observations yet. But the patterns were accumulating.

Realizing Leadership Systems Maturity Was the Hidden Variable

There is a class of organizational problem that looks like an execution problem but is actually a leadership system problem. I spent a significant portion of my career inside that category before I understood what I was looking at.

I have been in organizations that performed well by every external measure. Revenue was growing. Systems were stable. Engineers were shipping. And yet something felt wrong. There was a brittleness underneath the surface. The organization could execute, but it could not adapt. It could run, but it could not steer.

What I eventually identified was the gap between the complexity the organization was facing and the maturity of the leadership system it had in place to manage that complexity. The technical systems had scaled. The delivery systems had scaled. The leadership systems had not.

Promoted engineers were leading teams without frameworks for making the tradeoffs their new roles required. Experienced technical leaders were making architectural decisions without a structural way to connect those decisions to business risk. Executive sponsors were managing organizations whose internal operating conditions were invisible to them because no shared language existed to describe what was actually happening.

The maturity gap did not produce dramatic failures. It produced slow degradation. It produced the kind of organizational friction that accumulates quietly, that gets attributed to personality conflicts or cultural problems, that never gets traced back to the structural mismatch between organizational complexity and leadership system capability.

Recognizing that gap, and understanding it as a variable that could be named, measured, and addressed, was the conceptual foundation for what would become the Unicorn's ASCENT Framework™.

Connecting Architecture and Leadership

One pattern I observed consistently across organizations was the degree to which technical architecture and decision architecture mirror each other. This is not a metaphor. It is a structural reality.

Organizations that concentrate technical authority tend to produce systems that concentrate complexity. Systems that are difficult to modify without specialist intervention often exist inside organizations where decisions require the same. Technical debt accumulates in direct proportion to the structural conditions that prevent it from being addressed. Platform instability in high-pressure environments rarely has a purely technical cause. The instability reflects the decision conditions that governed how the platform was built and maintained.

I spent years watching organizations invest heavily in technical modernization while leaving the leadership systems that governed how technical decisions were made entirely unchanged. The new architecture would arrive. The old decision patterns would reassert themselves. Within 18 months, the new system would have the same problems as the old one, expressed in newer syntax.

The inverse was equally instructive. When an organization developed genuine structural capacity for distributed decision-making, when leaders at different levels had frameworks for connecting technical choices to business context, the technical systems reflected that capacity. They became easier to modify, easier to understand, and more tolerant of the pressure that comes with growth.

The technical architecture was legible as a record of how the leadership system had been operating. Once I saw that clearly, diagnosing a technical environment became partly a leadership system diagnosis. The codebase was evidence. The platform conditions were indicators. The deployment frequency, the incident patterns, the places where work slowed or stopped were all organizational signals, not just engineering signals.

Adaptive technical systems do not emerge by accident. They are produced by leadership systems that have developed the structural capacity to make good decisions under complexity and to maintain that capacity as complexity grows.

Why I Finally Systematized It

For most of my career, the pattern recognition I had developed was experiential and informal. I could walk into an organization and quickly develop a working model of how it was operating. I could identify where decisions were concentrating, where structural fragility was building, where the gap between leadership system maturity and organizational complexity was widest. But I was doing this intuitively, and intuition does not transfer.

The cost of misdiagnosis in technology organizations is not abstract. When structural problems are attributed to people, organizations replace people instead of changing structures, and the problems persist. When fragility is invisible until it fails, the failure arrives at the worst possible moment. When leadership system gaps go unnamed, they cannot be addressed deliberately. Organizations spend years managing symptoms because they do not have the language to describe causes.

I also became increasingly concerned about the damage done to capable individuals who were held responsible for structural conditions they did not create. Engineers blamed for incidents rooted in architectural decisions made years before they arrived. Leaders held accountable for organizational dynamics that were the product of structural design, not individual failure. The absence of a structural language does not protect people from blame. It just makes the blame arbitrary.

I systematized the model to create precision where there was previously only intuition. The Tech Organization Operating Model™ is a diagnostic framework built on two axes. Decision Velocity and Decision Clarity describe how an organization forms and executes decisions under pressure. Individual Reliance and Systemic Capability describe where organizational reliability actually lives. The intersection of those dimensions produces four structural conditions that I have observed in every technology organization I have worked in or studied.

The archetypes that emerge from those quadrants are not judgments. They are operating descriptions. They describe what an organization is doing structurally at a given point in time. They explain why certain strengths produce predictable distortions under scale. They provide a shared language for conversations that previously had no vocabulary.

The Unicorn's ASCENT Framework™ overlays that structural diagnostic with a model of leadership system evolution, describing how leadership capabilities must develop as organizational complexity increases. ASCENT explains why organizations move across the matrix and what conditions enable or prevent that movement.

Together, these two systems can describe what a technology organization is doing and why it is doing it. That distinction matters. Diagnosis without explanation produces observation. Explanation without diagnosis produces theory. The combination produces the basis for structural intervention.

What This Is, and What It Is Not

I want to be direct about the nature of this work, because precision here matters.

The Tech Organization Operating Model™ is not a culture framework. Culture is an output of structural conditions, not a root cause. Organizations that treat culture as the primary lever for change often find themselves investing in surface adjustments while the underlying structural dynamics remain unchanged. This model operates at the level of structure.

It is not a personality framework. The archetypes describe operating modes, not character types. The same individuals can operate inside different archetypes depending on the structural conditions they are working within. Attributing archetype behavior to individual disposition produces the same misdiagnosis problem that motivated the development of the model in the first place.

It is not a generic leadership development program. Leadership development that is disconnected from the specific structural conditions of technology organizations tends to produce generically competent leaders operating inside structures that have not changed. ASCENT describes leadership system evolution inside technology environments, calibrated to the specific demands of those environments.

It is not agile methodology, or any derivative of it. The model does not prescribe process frameworks. It diagnoses structural conditions. What an organization does with that diagnosis is a separate question.

This is a systems-based structural model derived from direct observation of how technology organizations operate under pressure. It is designed to give executives and technical leaders a precise shared language for describing what is actually happening inside their organizations, as a precondition for addressing it deliberately.

An Invitation to Look at Technology Differently

Technology organizations are not mysterious. They are structural systems that behave in observable and largely predictable ways. The patterns I began noticing as an intern have appeared, in one form or another, in every technology organization I have engaged with since. Different companies. Different scales. Different industries. The same structural dynamics, expressed in different contexts.

Once you have the language to describe what you are seeing, the diagnosis becomes faster and more reliable. You stop asking why talented people keep producing the same outcomes and start asking what structural conditions are generating those outcomes. You stop attributing organizational friction to personality and start examining decision architecture. You stop treating platform fragility as a technical problem and start recognizing it as an organizational signal.

That shift is not about adopting a new framework for its own sake. It is about developing the structural literacy to intervene at the right level. Most organizational problems in technology are not solved by the interventions that feel most obvious. They are solved by addressing the structural conditions that produce the behavior in the first place.

After 35 years, the thing I am most certain of is this: technology organizations behave rationally given their structural conditions. When those conditions are visible and accurately named, the path to changing them becomes clear. When they remain invisible, organizations stay locked inside patterns they cannot explain and therefore cannot escape.

Clarity changes how you intervene. That is why this work matters.