THE SHAPE OF THE WATER

A Narrative Journey Through Fluid Reliability

Companion Edition | Version 8.3

March 12, 2026

Andrea Valenti

> "*This is the story of an organization that learned to reshape itself continuously—not as a reaction to change, but as its natural state.*"

A Note to the Reader

This is not a how-to guide. This is a story about *why* structure must be fluid.

If you want implementation details, read the Complete Paper. If you want evidence, read Section 6 of that document. If you want ROI calculations, that's Section 7.

This document exists for a different purpose: to make you *care* about the question before asking for the answer. To let you feel what it's like inside an organization learning to flow. To give you the emotional truth that metrics cannot capture.

Read this when you want meaning. Read the Technical Paper when you want proof.

Prologue: The Jazz Lesson

In Brussels, sometime around 2010, I was learning to play saxophone. My friend Ivo was learning bass. We attempted Coltrane's *Equinox* together with a drummer who was the only competent musician among us. We failed spectacularly—but that failure taught me something I wouldn't understand until years later.

Jazz doesn't work when everyone plays their own part in isolation. It works when musicians listen, adapt, respond. When the bass walks somewhere unexpected, the saxophone follows. When the drums push the tempo, everyone adjusts. The sheet music is just a starting point. The real music happens in the spaces between the notes.

I still don't play *Equinox* properly. But I've spent the last 5+ years applying its lesson to organizations.

Part I: The Problem with Stability

The Illusion of Solid Ground

Before the framework existed, we operated like most SRE teams: firefighting, heroics, undocumented knowledge. Engineers were assigned to products and stayed there. Knowledge lived in heads, not systems. When someone left, expertise left with them.

We called it "stability." It was actually brittleness wearing a mask.

The cracks showed during acquisitions. Every two years, a merger arrived—new teams, new cultures, new technical debt, new ways of doing things that conflicted with our ways. Each time, we had to rebuild. Each time, we lost ground. Each time, the pain felt fresh.

> "*The pain doesn't diminish—it's the same every time. It's like a repetitive brain injury from doing the same thing over and over.*"

The traditional response was to fight harder. Document everything. Train everyone. Hope for the best. But hope isn't a strategy, and documentation doesn't transfer culture.

The Moment of Recognition

The recognition came slowly, then all at once: we weren't failing at managing change. We were failing at *being* changeable.

The distinction matters. Managing change implies that change is an exception—an interruption to normal operations that must be controlled and minimized. Being changeable means accepting that change *is* the normal state, and building an organization that moves with it rather than against it.

Water doesn't fight its container. It becomes the shape of whatever holds it. That became our metaphor. That became our name.

The takeaway: Stability is not the goal—adaptability is. The question is not "how do we prevent change?" but "how do we become changeable?"

Part II: Theoretical Anchors

Luhmann's Membranes

Niklas Luhmann saw organizations not as hierarchies but as systems of communication. What matters isn't the org chart—it's the pattern of signals flowing through the network. And crucially, systems maintain identity through *boundaries*: semi-permeable membranes that filter signal from noise, that let essential information through while blocking disruption.

This became our gatekeeping principle. Not gatekeeping as bureaucracy, but gatekeeping as protection—a living membrane that shields engineers from the chaos of interrupt-driven work while remaining permeable to genuine need.

> *"Without a visible and trusted intake model, no protection mechanism holds. Gatekeeping gave us the membrane between urgency and distraction."*

Olivetti's Community

Adriano Olivetti ran a typewriter company in mid-century Italy, but he thought like a philosopher. "The factory cannot look only at the profit index," he wrote. "It must distribute wealth, culture, services, democracy."

We aren't a factory. But Olivetti's insight applies: teams aren't resources to be optimized. They're communities to be cultivated. Autonomy matters. Learning matters. Purpose matters. A team that understands *why* it does what it does will outperform a team that merely executes instructions.

Each team became a learning cell—responsible for its rituals, its self-improvement, its peer-to-peer development. We gave them autonomy within direction, and they gave us engagement.

Gramsci's Organic Intellectuals

Antonio Gramsci, the Italian political philosopher, writing from prison, developed the concept of the "organic intellectual"—not the detached academic theorist, but the thinker who emerges from within a community, who articulates its values and spreads its culture.

We call them culture carriers.

In every team, there are people who quietly normalize high standards. They don't have special titles. They don't give speeches. But their influence ripples outward. When they pair with someone new, that person learns not just technical skills but cultural expectations. When they rotate to another team, they carry the culture with them.

We learned to identify these carriers. Not to promote them away from the work, but to amplify their influence deliberately.

Rogers and the Spread of Innovation

Everett Rogers spent his career studying how new ideas spread through social systems. His insight: adoption happens through legitimacy and participation, not mandate. People don't change because they're told to. They change because they see peers they trust succeeding with the new approach.

This shaped how we introduced every practice. No big-bang rollouts. No mandates from above. Instead: willing pilots, shared outcomes, internal ambassadors. By treating change as a social process rather than a deployment, adoption became contagious.

The takeaway: Every practice rests on theory—boundaries that protect, communities that learn, carriers who spread culture, and adoption that happens through legitimacy rather than mandate.

Part III: The Three Cycles

Cycle One: The Laboratory (2019–2021)

The first cycle began with a single product team. A small group, full autonomy, a desperate need to rebuild engineering culture around resilience.

We didn't have a framework yet. We had experiments.

The gatekeeping rotation emerged first—a weekly sacrifice in which one engineer handled all the interrupts so others could focus. It was, as we still say, "a horrible week." But horrible weeks shared equally build solidarity. And engineers who saw the chaos firsthand understood viscerally why automation mattered.

Kanban-style triage boards surfaced operational load transparently. For the first time, we could see what was really happening—not what we imagined, not what the reports claimed, but the actual flow of work through the system.

Quarterly reviews became ritual. Not status updates, but honest examinations: Does our structure match our strategy? Are we defending systems or enabling teams?

By the end of this cycle, SRE had transformed from reactive firefighters to proactive engineers. Morale climbed. Trust grew. We had something that worked.

Then the merger came.

Cycle Two: The First Merger (2021–Mid 2022)

Three acquired products. Three cultures. Three ways of doing things. Suddenly we weren't a small autonomous team anymore. We were a franchise responsible for platforms we hadn't built.

The pain was real. We had to put our beautiful advanced projects aside and go back to basics. Where is your visibility? Where is your gatekeeping? Where is your documentation? Let's build it again.

> *"I have this beautiful jazz standard I want to play. But my new bandmate hasn't developed their embouchure yet—everything sounds weak and squeaky. So we go back to basics: long tones, fundamentals, patience."*

But something surprising happened. The rebuild went faster. The patterns we'd developed weren't just practices—they were *portable* practices. Rotation across product boundaries distributed SRE culture. Shared Kanban boards enabled visibility across silos. The model proved interoperable.

Six months where it had taken two years before.

Cycle Three: The Convergence (Mid 2022–Present)

Another wave of acquisitions. A larger convergence—multi-product, multi-cloud, service footprints we'd never touched before. New domains. New technical stacks. New cultures to absorb.

By now, we had institutional memory. All five phases reapplied, now with experience. Gatekeeping scaled with automated triage pipelines. Infrastructure as Code standardized environments across formerly divergent stacks. Resource pooling matured into dynamic staff deployment backed by data.

The framework had proven something important: it wasn't just a way to build one team. It was a repeatable organizational metabolism that adapts and grows with its environment.

The takeaway: Each cycle shortened—two years, then one year, then six months. The framework doesn't eliminate disruption; it reduces recovery time.

Part IV: The Five Practices (A Human Account)

Protect: The Shield That Enables

Protection sounds defensive. It isn't.

Engineers under constant interrupt cannot think deeply. They cannot innovate. They cannot do the work that prevents future fires. Protection isn't about avoiding work—it's about enabling the *right* work.

The gatekeeping rotation creates that shield. Yes, the gatekeeper has a terrible week. But that terrible week buys focus for everyone else. And because everyone rotates through, everyone understands the cost. Everyone has skin in the game.

> *"Sharing the pain is a key factor in a team. We all have to share that burden equally."*

Prepare: Learning to Move

Preparation means training for rotation before rotation happens. It means developing "organizational languages"—platform, customer ops, product, architecture—so engineers can translate across boundaries.

We built context packets and onboarding guides. We created "tours of duty" that moved people through teams before they needed to perform there. We replaced manager resistance ("I'll lose my best person!") with confidence in coverage ("I'll get them back enriched, and someone equally capable will fill in").

The goal was never cross-skilling for redundancy. It was cross-skilling for adaptability. An engineer who has worked in three domains thinks differently than one who has worked in one.

Pulse: The Quarterly Question

Every quarter, we ask the same questions:

- Are our team boundaries, responsibilities, and leaders aligned with current business intent?
- Are we defending systems or enabling teams?
- Are rotations producing clarity or confusion?

These aren't retrospectives. They're structural reviews. We're not asking "how did the sprint go?" We're asking "is our organization still correctly shaped?"

Sometimes the answer is yes, and we confirm our structure. Sometimes the answer is no, and we reshape. The ritual matters more than any individual outcome.

Pool: Talent Without Possession

Resource pooling terrifies managers. "I'll lose my people." "They won't come back." "The knowledge will leave."

We addressed this structurally. Engineers remained affiliated with a home team but rotated with clear return paths. Assignments flowed through data and process, not politics. The gatekeeper absorbed L1 work, so rotations didn't create coverage gaps.

When federal compliance certification arrived, we didn't hire consultants. We reassigned internal talent temporarily, then returned them—enriched with compliance knowledge that stayed in-house. The cost savings were substantial. The capability building was priceless.

Promote: Amplifying Culture

Culture isn't an accident. It's an asset to be engineered.

We identified culture carriers—not by title but by influence. We made their impact intentional through rituals: rotation graduations, postmortem storytelling, shared dashboards that normalized transparency.

And we tied promotion not just to delivery, but to contribution to systemic resilience. The engineer who made three others better was worth more than the engineer who performed alone.

The takeaway: The five practices form a system—protection enables focus, preparation enables movement, pulse enables correction, pooling enables flexibility, and promotion enables sustainability.

Part V: What The Numbers Mean

The companion technical paper contains the full quantitative validation: 25,498 tickets across 5+ years, statistical significance at $p < 0.001$, TOIL reduction from 59.7% to 44.7%.

But numbers without narrative are just numbers.

What does 44.7% TOIL mean? It means that more than half of engineering time now goes to work that creates enduring value—improvements, innovations, capabilities that compound. It means we crossed Google's benchmark for healthy SRE organizations.

What does absorbing three mergers — 6 to 38 engineers — with zero critical knowledge loss mean? It means the framework scales. It means rotation works. It means culture transfers without requiring heroes.

What does 15-20% retention advantage versus peer teams mean? It means engineers *want* to stay. Despite the rotations. Despite the terrible gatekeeper weeks. Despite the constant change. Maybe because of them.

> *"I've never been in an org that planned for morale instead of reacting to burnout."*

The takeaway: Numbers validate what experience teaches—this framework produces measurable improvement in TOIL, retention, and scalability.

The AI Paradox

And yet, the industry keeps looking for shortcuts.

In 2024, DORA reported that software delivery throughput at AI-adopting organizations dropped by at least 1.5%. In 2025, a randomized controlled trial by METR made it worse: 16 experienced developers — working on their own repositories, code they'd maintained for years — were **19% slower** with AI tools. Before the study, they predicted a 24% speedup. After measuring the slowdown, they still believed AI had made them 20% faster. The gap between perception and reality was 43 percentage points. And 69% kept using the tools anyway.

Read that again. Engineers feel faster. Measurement says slower. They see the data. They keep believing.

The paradox compounds at team level. Teams with high AI adoption merge 98% more pull requests — but review time increases by 91%. AI-generated code has a 32.7% acceptance rate versus 84.4% for human-written code. The bottleneck didn't disappear. It moved. From writing code to reviewing it. From the individual to the membrane.

By February 2026, a study of 6,000 executives confirmed what Robert Solow observed about computers in 1987: "*You can see the computer age everywhere but in the productivity statistics.*" The vast majority report little measurable impact from AI on operations.

This is the mitochondrial fallacy wearing a new mask. AI is the latest promise that you can solve an organizational problem with a technical solution. That if you just give people better tools, the structure will take care of itself.

It won't. We proved it won't. Our TOIL went from 83.9% to 44.7% — not because we adopted better tools (though we did), but because we changed how people relate to the work, to each other, and to the organization itself. The tools were necessary. They were never sufficient.

Google's own 2025 DORA report finally acknowledged the qualifier: AI correlates with higher throughput *only when the entire delivery system adapts*. Not the coding step. The system. The membrane, the rotation, the quarterly question of *does our shape still match our strategy*. Without that, faster tools just produce faster drift — more changes flowing through unchanged review processes, more incidents from insufficiently vetted code, more TOIL from automations that shift work rather than eliminate it.

The takeaway: Three years of evidence — from DORA surveys to controlled experiments to executive studies — confirm what we learned the hard way: tools without structure accelerate chaos as efficiently as they accelerate progress.

Part VI: The Philosophical Heart

Why Water?

Water doesn't fight its container. It flows around obstacles. It finds the path of least resistance—not through weakness, but through adaptability. Pour it into a cup and it becomes the cup. Pour it into the ocean and it becomes the ocean.

This is the opposite of brittleness. Brittle systems break under stress. Fluid systems reshape.

Traditional organizational thinking treats structure as scaffolding: build it right once, and it will hold forever. But scaffolding is temporary by design. It's supposed to come down when the building is done. Organizations are never done.

The Cyclic Nature

We don't progress linearly from chaos to order. We cycle:

Stagnation → Disruption → Optimization → Flow → Stagnation

Each merger resets us to disruption. Each recovery builds new capability. The cycles aren't failures—they're the mechanism of growth.

> *"Those shifts aren't so disruptive anymore because we've been doing this continuously."*

The first cycle took two years. The second took one. The third took six months. We're not eliminating pain. We're reducing recovery time.

The Irony

Here's the irony that took years to appreciate:

We're a team that preaches immutable infrastructure—containers that never change in place, configurations that rebuild rather than patch. And yet we argue that immutable organizational structure is a false promise.

The contradiction is only apparent.

We don't actually achieve immutability in infrastructure. We achieve *idempotent rebuildability*. The container is disposable. The Dockerfile is stable. We preserve the ability to recreate capability, not the capability itself.

Apply this to organizations: we don't preserve the team. We preserve the ability to recreate the team. Rotation isn't loss—it's the mechanism of perpetual reconstruction.

The takeaway: Like water, the goal is not to resist change but to flow with it. Like infrastructure, the goal is not to preserve instances but to preserve the ability to rebuild them.

Part VII: What We Learned the Hard Way

You Cannot Isolate Excellence

When a team reaches high performance, the temptation is to protect it. Isolate it from the chaos. Let it continue its advanced work while others rebuild.

We tried this. It failed.

> "*People often asked me: why not isolate the high-performing teams and let them keep flowing? Because we reached that level of maturity precisely by not isolating ourselves.*"

Excellence isn't self-sustaining. It requires constant renewal—fresh eyes, new challenges, the friction of different perspectives. An isolated team calcifies. Its practices become rituals divorced from purpose. Its culture carriers leave, and the culture leaves with them.

Throughput Is Not Value

We were a Kanban team from the beginning. Kanban is built around flow—work in progress limits, cycle time, throughput. We got very good at moving tickets.

But tickets aren't outcomes.

> *"It's like doing laundry just to say you did laundry—washing ten socks nobody needs. The activity happened, but no value was created."*

The shift from throughput to value delivery is still ongoing. We're learning to ask not "how many phases did you complete?" but "what can you show us that works?" It's harder to measure. It's harder to report. But it's the only thing that matters.

Autonomy Requires Alignment

We gave teams autonomy. Micro-teams of three or four, fully independent, choosing their own cadences and methods. But autonomy without direction is chaos.

The solution: intent-based leadership. We define the outcome. Teams define the path. "I don't care how you get there. I care that you get there."

This only works if the intent is clear. If the outcome is measurable. If teams can validate their own progress without waiting for approval.

The takeaway: Excellence requires renewal, throughput is not value, and autonomy without alignment is chaos. These lessons came from failure, not theory.

Part VIII: The Invitation

This narrative companion doesn't stand alone. It pairs with the technical paper, which contains the methodology, the data, the statistical validation.

But frameworks don't transfer through documentation alone. They transfer through practice, through rotation, through the slow accumulation of shared experience.

If you recognize your organization in these struggles—the brittleness masquerading as stability, the heroics masquerading as excellence, the change fatigue that never quite goes away—then perhaps something here is useful.

Not as a blueprint to copy. But as a story to learn from.

Epilogue: Still Learning Long Tones

I still don't play *Equinox* properly. Fifteen years later, and Coltrane remains out of reach.

But I've learned something about practice. The advanced stuff—the substitutions, the polyrhythms, the extended techniques—they all rest on foundations. Long tones. Scales. The embouchure that only comes from thousands of hours of patient work.

Organizations are the same. The advanced capabilities—the seamless rotations, the cultural resilience, the ability to absorb mergers without losing ground—they rest on foundations too.

Visibility. Protection. The shared pain of gatekeeping. The quarterly rhythm of honest assessment. The humble recognition that we're never done, that the cycle continues, that tomorrow brings another opportunity to reshape ourselves.

The water keeps flowing.

The shape keeps changing.

And somewhere in that constant motion, we find something that looks like reliability.

Where to Go from Here

If this resonated: Read the Complete Paper. It contains the evidence, the economics, and the falsifiable claims. What you felt here is validated there.

If this challenged you: Good. The framework doesn't ask for belief—it asks for experimentation. Try one practice. Measure the results. Let the data convince you.

If this hurt: You may be living in an organization that has forgotten how to flow. The pain of recognition is the first step toward change. You are not alone; 82% of SRE professionals report burnout symptoms.

If you want to try this: Start with gatekeeping. It requires no permission, no budget, no organizational change. Protect one person's focus for one week. See what happens.

If you want to talk: The framework needs external validation. Your context, your failures, your adaptations—they're all valuable data. Reach out.

For the technical framework, quantitative validation, and implementation details, see: ***The Shape of the Water: Fluid Reliability Framework — Complete Edition v8.3***

Author

Andrea Valenti is a Senior Director specializing in organizational transformation and Site Reliability Engineering. The Fluid Reliability Framework represents 5+ years of applied research validated through 25,498 operational tickets across three major organizational cycles.

Acknowledgments

To Ivo Velitchkov, who asked the right questions and made me articulate what I had only practiced. To the engineers who rotated through terrible weeks and emerged stronger. To the culture carriers who spread standards without speeches.

And to the saxophone, which taught me that the music lives in the spaces between the notes.

Version 8.3 | March 12, 2026

v8.3: Inclusive language review — replaced exclusionary terminology, removed organization-identifying references

---

"*The organization that cannot metabolize adversity cannot sustain reliability. We learned to metabolize adversity. And in doing so, we learned that adversity is not the opposite of reliability—it is the teacher of reliability.*"