ML LABS INTELLIGENCE

Why AI Productivity Hits an Entropy Ceiling

AI can raise software output fast while also accelerating architectural mess and operational drag. This article shows how to avoid trading short-term speed for long-term entropy.

STRATEGIC

Author Omar Trejo

Date 2026-01-17

The current enterprise AI narrative is straightforward: more capable models, more autonomous tooling, more output per engineer. The first part is true. Teams are shipping faster. Drafts appear instantly. Refactors that once took days now happen in hours.

The problem is that **output velocity is not the same thing as system productivity**, as **Subbu Allamaraju has argued**. In software, gains made at the point of code generation are often paid back later through operational drag, architectural confusion, and failure recovery. If the rate of change rises faster than the organization's ability to preserve coherence, entropy rises with it.

That is the tension many leadership teams are missing. AI does not just increase throughput. It also increases the speed at which teams can deepen coupling, harden bad early assumptions, and create changes nobody fully understands until the system starts breaking in production.

Why Faster Delivery Often Produces Slower Organizations

Software systems do not degrade because teams stop working. They degrade because teams keep shipping into environments shaped by old decisions, conflicting incentives, and incomplete understanding.

Frederick Brooks argued decades ago in **No Silver Bullet** that some software complexity is irreducible. AI changes the economics of implementation, but it does not remove the essential complexity of operating a growing system under changing business conditions. In practice, the faster teams move, the more important structural discipline becomes.

The organizations struggling with AI adoption are usually not suffering from lack of model access. They are suffering from a widening gap between the speed of local execution and the integrity of the system being changed. That gap is where entropy

accumulates.

> AI does not eliminate software complexity. It compresses the time between introducing complexity and paying for it.

The Four Forces That Raise Entropy

The pattern shows up repeatedly in software organizations adopting AI aggressively. Four forces matter most.

1. Path Dependence Hardens Early Decisions

Early architecture choices rarely stay small. A data model, workflow pattern, integration boundary, or deployment assumption becomes embedded in adjacent systems, team structure, and reporting lines. Once the business starts depending on it, changing it becomes expensive even when everyone knows it is wrong.

Paul David's work on path dependence explains why early choices constrain later ones. In software, the result is familiar: monoliths that cannot be separated cleanly, bespoke workflows that only one team understands, and business logic trapped inside tools chosen for a very different stage of growth.

AI does not neutralize path dependence. In many organizations it intensifies it. Teams can now build on top of flawed foundations much faster, adding more code, more integrations, and more dependencies before anyone pauses to question the base layer. The apparent productivity gain is real in the short term. The lock-in cost grows underneath it.

2. Competing Feedback Loops Pull the System Apart

Organizations do not optimize for one thing. Product teams optimize for growth. Platform teams optimize for stability. Operators optimize for continuity. Finance optimizes for efficiency. AI increases the execution power available to all of them at once.

That sounds positive until those feedback loops begin to conflict. A team using AI to ship features faster can create operational load for the team responsible for keeping the system stable. Another team may use AI to optimize internal workflows in ways that increase coupling or bypass controls. Each decision is locally rational. The system-level result is disorder.

Donella Meadows' systems thinking work is useful here because it frames the issue correctly: organizations are constantly balancing reinforcing loops that drive growth and balancing loops that preserve stability. AI accelerates both the intended work and the tension between them. Without an explicit operating model, that tension resolves into entropy.

3. Delayed Feedback Makes Risk Look Cheap

Some of the most expensive failures begin as invisible drift. A team delays cleanup because feature demand is high. Data quality degrades slowly. A brittle integration keeps working just well enough. Documentation stops matching reality. For a while, nothing breaks visibly, so the organization assumes the tradeoff was acceptable.

Then the repair bill arrives all at once.

Sydney Dekker's analysis of systems drifting into failure matters because it shifts attention from isolated mistakes to the slow accumulation of unresolved signals. AI changes this dynamic by increasing the volume of system change. If maintenance capacity and observability do not scale with that change rate, delayed feedback loops multiply.

This is why many AI productivity calculations are incomplete. They count time saved during creation, but not the future maintenance load created by faster, less-governed change.

4. Stale Models Make Good Intentions Dangerous

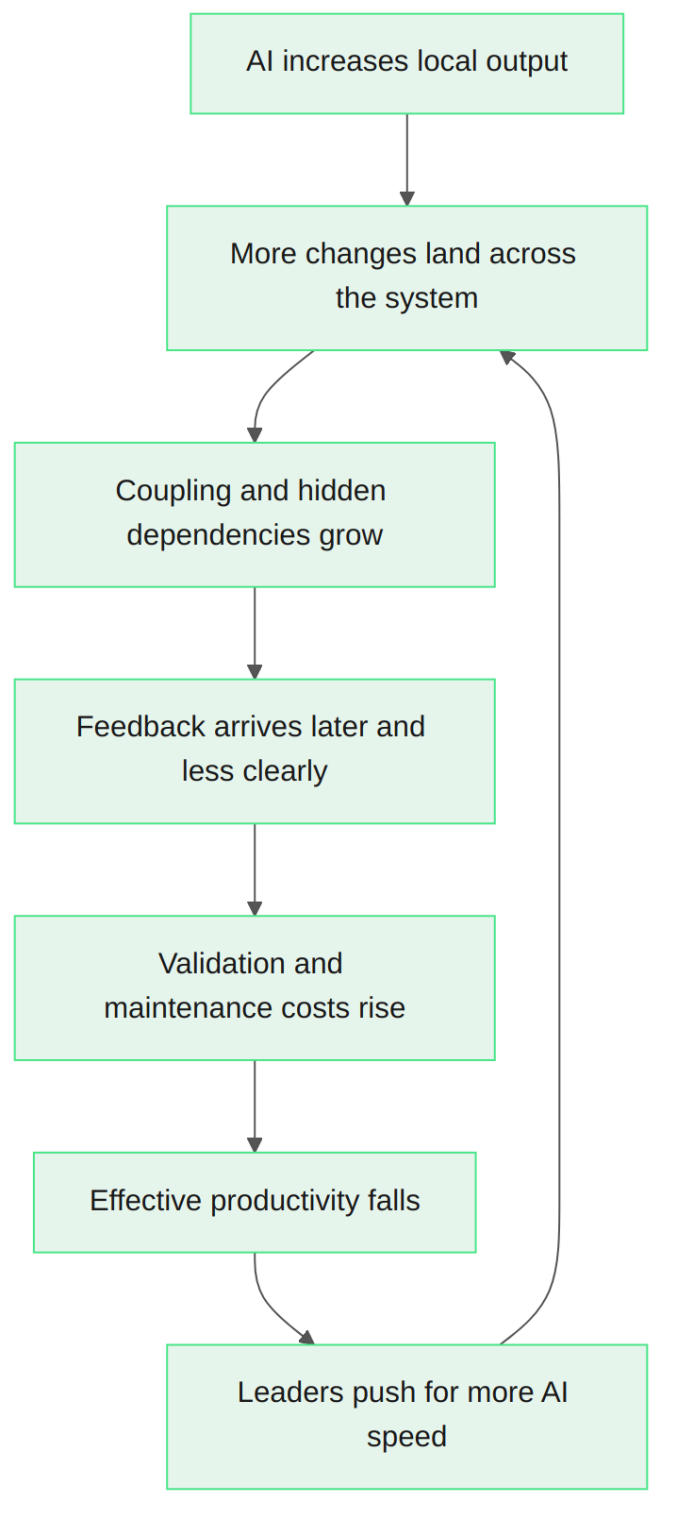
Every engineer, team, and executive holds a partial model of how the system works. Those models are never perfectly aligned. As software evolves, the gap widens. The original designer sees one architecture. A newer engineer sees the surface behavior. Operations sees runtime fragility. Leadership sees delivery metrics.

AI inherits the same problem. It acts on an inferred model built from the code, prompts, surrounding files, and local context available at the moment. That model can be useful without being complete. When multiple people use AI against the same codebase, the organization is effectively introducing multiple high-speed actors operating from slightly different maps of the same terrain.

The result is not just occasional code defects. It is model drift at organizational scale: more changes made confidently, fewer shared assumptions, and higher validation cost for every consequential decision.

Where the Productivity Ceiling Appears

The ceiling does not appear when the model becomes less capable. It appears when the system can no longer absorb the rate of change being applied to it.


AI increases local output
More changes land across the system
Coupling and hidden dependencies grow
Feedback arrives later and less clearly
Validation and maintenance costs rise
Effective productivity falls
Leaders push for more AI speed

This loop is why some teams feel dramatically faster for one quarter and materially less effective two quarters later. The throughput increase is real, but the system coherence required to sustain it never caught up.

The wrong conclusion is that AI "stopped working." The more accurate conclusion is that the organization converted velocity into entropy faster than it converted velocity into durable capability.

The Operating Model That Prevents the Slide

If AI adoption is going to produce durable gains, the management model has to change with it. The answer is not to slow everything down indiscriminately. The answer is to constrain high-speed change with stronger coherence mechanisms.

Architectural Guardrails

Define where AI-assisted changes are allowed to move quickly and where they are not. Stable interfaces, bounded contexts, approved integration patterns, and explicit ownership boundaries matter more in an AI-heavy environment because they limit the spread of local decisions.

If every team can change everything faster, the architecture degrades faster. If teams can move quickly inside well-defined boundaries, AI becomes a multiplier instead of an accelerant for disorder.

Shared System Models

Critical systems need living documentation that reflects actual runtime behavior, data dependencies, and ownership. This is not bureaucracy. It is the minimum viable mechanism for keeping engineers, operators, and AI tooling oriented around the same system model.

The important standard is not perfect documentation. It is enough shared truth that consequential changes can be evaluated against something more reliable than memory and local interpretation.

Feedback Compression

As change velocity increases, feedback loops must get shorter. That means stronger observability, explicit maintenance budgets, tighter post-deployment review, and routine auditing of assumptions that no longer hold.

Organizations that treat maintenance as optional overhead while accelerating AI-assisted development are making a structural mistake. They are increasing system mutation while weakening the functions that detect and absorb mutation.

Review by Reversibility, Not by Volume

Not every AI-assisted change needs heavyweight review. The right standard is reversibility. Small, reversible changes can move fast. Changes that alter schemas, workflows, permissions, or integration boundaries need stronger review because they increase future path dependence and recovery cost.

This lets organizations preserve speed where it is safe while slowing only the subset of work most likely to compound entropy.

Expected Results

Organizations that adopt this operating model usually see an initial correction period. Some work slows down. A few optimistic productivity assumptions get revised. Teams discover more hidden coupling than expected.

That correction is healthy. It is the organization converting apparent velocity into real control.

Once the boundaries are clear, AI-assisted execution becomes materially more valuable. Teams can automate more confidently because they know where the edges are. Review burden falls on the changes that matter most. Maintenance becomes visible earlier. The quality of decisions improves because everyone is reasoning from a more coherent view of the system. In practical terms, concentrated expertise paired with AI inside a disciplined operating model can deliver more than a larger team operating without one.

When This Framing Gets Misused

Entropy is not an excuse for paralysis. Some organizations respond to these risks by creating so much governance that AI becomes irrelevant to execution. That is the opposite failure mode.

The point is not to prove that rapid AI adoption is dangerous. The point is to recognize that speed without structure is self-defeating. The winning organizations will not be the ones that avoided AI, and they will not be the ones that turned every engineer into an unbounded change generator. They will be the ones that paired acceleration with tighter control over architecture, feedback, and decision rights.

First Steps

1. **Map your highest-entropy systems.** Identify the codebases and workflows where coupling, unclear ownership, and delayed feedback are already present. Do not start AI acceleration there without extra controls.
2. **Classify changes by reversibility.** Define which AI-assisted changes can move automatically, which require peer review, and which require explicit architectural approval.

3. **Create one shared operating model for critical systems.** Document runtime dependencies, data boundaries, ownership, and recovery constraints in a form that both humans and AI tooling can use.

The ceiling on AI productivity is not set by the model alone. It is set by how much disorder the surrounding system can absorb before speed turns into drag.

Practical Solution Pattern

Treat AI productivity as a systems problem, not a code generation problem. Increase delivery speed only in proportion to your ability to preserve architectural boundaries, compress feedback loops, and maintain a shared model of the system being changed. The organizations that compound value from AI are not the ones generating the most changes. They are the ones generating the most controlled changes.

This works because entropy is what eventually converts fast local output into slow organizational performance. Path dependence, competing incentives, delayed signals, and incomplete system models do not disappear when AI arrives; they become easier to intensify accidentally. Installing boundary-based governance, reversibility-aware review, and explicit maintenance capacity addresses the actual bottleneck instead of optimizing the visible surface layer alone. If AI output is rising faster than system coherence, an **AI Engineering Retainer** can tighten architecture, review, and delivery discipline across the live workflow.

References

1. Allamaraju, S. **Productivity and Entropy**. *Writing is clarifying*, 2026.

2. Brooks, F. P. **No Silver Bullet: Essence and Accidents of Software Engineering**. *Computer*, 1987.
3. David, P. A. **Clio and the Economics of QWERTY**. *The American Economic Review*, 1985.
4. Meadows, D. H. **Thinking in Systems**. *Chelsea Green Publishing*, 2008.
5. Dekker, S. **Drift into Failure**. *Ashgate/Routledge*, 2011.