



DOME SYSTEMS

The New Shadow IT

How velocity engineering is creating structural governance tension and why the answer is consistent operations, not restriction

A product manager builds an agent on a Tuesday	3
The zero-cost code thesis	4
Vibing in the pursuit of velocity	5
The tension of speed and control	6
Why restriction fails as a regressive instrument in an era defined by speed	7
Getting from here to there	8

A product manager builds an agent on a Tuesday

A product manager at a Fortune 500 company has a problem. Her team spends eight hours a week compiling data from three internal systems into a weekly report. On Tuesday afternoon she opens Claude, describes the workflow, and has a working prototype by Wednesday morning. The agent queries the CRM, pulls metrics from the data warehouse, formats the report, and drops it into Slack.

It works. The team loves it. She shares it with two other PMs.

By Thursday, three teams are running agents that access production data systems. No one in platform engineering knows they exist. No one in security has reviewed what data they access. No credentials have been formally issued. No audit trail exists.

This is not negligence. This is not a rogue actor circumventing policy for personal gain. This is the rational behavior of someone with a real problem and a tool that solves it in hours instead of months. And it is happening inside every enterprise with more than one team and access to a large language model.

The zero-cost code thesis

The cost of writing working software is collapsing. Large language models have compressed the cycle from idea to functioning prototype from weeks to hours. This is not a marginal improvement in developer productivity. It is a structural change in who can build software and how fast.

However, the relationship between Speed, Cost, and Risk continues to exist. With AI, then everything is sacrificed to achieve almost limitless speed, with at least some promise of cost reductions of doing so. The constraint is no longer "can we build this?" The constraint becomes "should we run this, and under what conditions?". That question who is allowed to access what, under which policies, with what oversight requires tooling to answer. Risk is now the core challenge for an enterprise adopting AI.

Most enterprises do not have the controls to manage this kind of risk. The tooling they have was designed for a world where software was built by engineering teams and deployed through controlled pipelines. The assumption that production systems pass through a governance checkpoint before going live is no longer holding.

Vibing in the pursuit of velocity

This reach for speed isn't 'vibe coding' in the YOLO sense that it has come to mean. Rather, it's using available practices to achieve maximum speed. So, what emerges when code is cheap and nobody is operating it safely?

A 'velocity engineer' is anyone inside the enterprise who builds functional agent systems rapidly and independently, optimizing for delivery speed rather than operational governance. They are not bad actors. They are product managers, data scientists, sales engineers, analysts: people with real problems and tools that let them build solutions faster than they can get them approved through existing channels.

The pattern is shadow IT, accelerated by an order of magnitude.

Shadow IT in the 2010s was a team buying a SaaS tool on a corporate credit card. The tool operated within its own boundary. The blast radius was limited to one vendor's application. The governance team could discover it, evaluate it, and bring it into compliance with a vendor review and an SSO integration.

Velocity engineering is different in kind, not just in degree. An AI agent is not contained within a vendor boundary. It reaches out. It calls tools. It queries databases. It accesses APIs. It acts on behalf of the enterprise in ways that create real operational and compliance exposure and it can be built and deployed in the time it takes to schedule the meeting where you would have discussed whether to build it.

The tension of speed and control

The tension is structural, not adversarial.

Velocity engineers are measured on delivery speed. They have a real problem, a working solution, and a team that needs it yesterday. The governance question: "is this agent operating within acceptable boundaries?", is genuinely not a question they can answer with the tools available to them, even if they wanted to. There is no self-service path to governed agent deployment in most enterprises. No portal where you register an agent, request tool access, and get credentials that come with policy enforcement and audit logging built in. The only path available is the ungoverned one.

Enterprise security and platform teams are measured on risk reduction. They need to know what agents exist, what each can access, and whether any of them are creating exposure. They cannot approve what they cannot see. And what they can see, when they eventually discover an ungoverned agent, they can only respond to with restriction: shut it down, block the access, add a policy that prevents it happening again.

Restriction destroys value. The agent was solving a real problem. The team that built it will find another way and probably one that is even less visible. The operations team knows this. The velocity engineer knows this. Neither has the tooling to resolve it.

Both sides are doing their jobs. Both are making individually rational decisions. The outcome is collectively irrational: valuable innovation that is either ungoverned or blocked, with no option in between.

Why restriction fails as a regressive instrument in an era defined by speed

The instinct to restrict is understandable and wrong. And we've seen it before in prior industry transformations.

Blocking velocity engineering does not prevent agents from being built. It prevents them from being built visibly. The cost of building an agent is now so low that any team with a problem and an LLM can do it. Restricting access to one tool pushes usage to another. Restricting one pathway pushes innovation underground. The barrier to entry is a browser. The barrier to deployment is a cron job.

Retroactive governance amounting discovering ungoverned agents after the fact and bringing them into compliance, is worse. It requires reconstruction from fragments: access logs that were never designed to track agent behavior, interview notes, best guesses about which data flowed where. The compliance evidence was never generated because the tooling to generate it was never in place. You cannot audit what was never recorded.

And the cost of retroactive governance scales with time. Every week without governance tooling is a week of discovery debt accumulating. The ungoverned agents do not pause while the governance team builds its response. They multiply. They connect to more tools. They process more data. The reconstruction problem compounds.

The only answer that works at scale is an Agent Operations Platform that makes governed access the path of least resistance. A platform that enables by saying "yes, under these conditions, with this identity, audited by default".

Getting from here to there

What does this platform look like from both sides?

From the velocity engineer's perspective: you build your agent, connect it through the platform, and it works. Credentials are issued automatically. Policies are enforced transparently. The agent has access to the tools it needs, subject to the rules the enterprise has defined. You did not file a ticket. You did not wait three weeks for a security review. You connected through a platform that was designed for exactly this — and your agent was governed from the first tool call.

From the enterprise team's perspective: every agent is registered. Every tool call is visible. Every action is auditable. When a new agent connects, you see it immediately. You can scope its access, apply policy, and monitor its behavior from the moment it starts operating. When something goes wrong, the evidence chain is already there not because someone remembered to set up logging, but because the platform produces it as a natural consequence of the agent operating through a governed path.

The velocity engineering problem is structural. It is not a failure of discipline or a gap in training. It is the inevitable consequence of cheap code meeting absent operations. Every enterprise with more than one team will produce this dynamic. The enterprises that provide the platform will be the ones where innovation and governance coexist where velocity engineers can move fast, and enterprise teams can say yes. The ones that do not will oscillate between ungoverned innovation and innovation-killing restriction, never resolving the tension, because the resolution was never a choice between the two sides. It was always the platform.