The future of AI engineering is not more code, but better decisions about what gets built

Most companies are applying AI to the wrong bottleneck.

They are treating AI engineering as a code generation problem. It is becoming something else: a decision problem. What should start? What should wait? What is shaped enough to hand to AI? What is too risky, too vague, or too cross-cutting to accelerate safely?

That shift matters because AI output is rising faster than most organizations' ability to govern, verify, and absorb change. DeliveryTower's public framing captures this well: it sits above the coding agent, assesses work in context, routes it into the right execution lane, and learns from delivery history such as review friction, regressions, and follow-on work.

The result is a new failure mode in software delivery. It is no longer just "developers cannot produce enough code." It is "the system can produce more change than humans can responsibly review and verify." DeliveryTower itself frames the problem as one of strategy, verification, blast radius, similarity to prior changes, and learned signals from PRs and incidents rather than simple issue tracking.

AI is increasing output. It is not automatically increasing delivery capacity.

This is the mistake many companies are making right now.

They automate implementation before they improve work selection.

They increase the rate at which code can be produced, but they do not improve the quality of decisions about which backlog items deserve to become implementation work in the first place. They confuse coding speed with delivery capacity.

Those are not the same thing.

Delivery capacity is constrained by reviewer attention, verification capacity, and the organization's ability to decide whether a change is aligned, understood, and safe enough to absorb. DeliveryTower's operating model explicitly centers on these questions: whether work is aligned, verifiable, safe for AI, and likely to remain safe after release.

That is why more AI-generated code can make a team slower overall. Not always, and not because AI is inherently harmful. It can make a team slower when the change volume grows faster than review quality, verification discipline, and work selection improve.

In that environment, velocity becomes deceptive. Code output rises. PR counts rise. But so do regressions, reviewing churn, follow-on fixes, and the amount of scarce human judgment spent on work that was never a strong candidate to begin with.

Not every backlog item deserves to be implemented

This is the uncomfortable point many teams will need to accept.

A backlog is not a queue of equally valid candidates for automation.

Some work is strategically important but not yet shaped. Some is locally attractive but weakly aligned. Some have unclear acceptance criteria. Some have no credible verification path. Some look small, but historically produce cross-cutting regressions, support pain, or rollback risk. Some is simply not worth the reviewer's attention; it will consume the reviewer's time.

In an AI-assisted environment, those distinctions matter more, not less.

When coding capacity becomes abundant, the highest-leverage decision is no longer “how do we get more implementation done?” It is “which work deserves implementation at all?”

That is the category DeliveryTower points to. Its public site describes the product as a decision surface for product work, not another issue board, with the goal of producing a lane recommendation, reasons, and an AI-ready brief grounded in context and history.

The new scarce resources are reviewer attention and verification capacity

The real limit on AI engineering is not token supply or model capability alone.

It is the human and organizational capacity to decide whether a change is good, safe, and worth merging.

Reviewer attention is scarce. Verification capacity is scarce. Security review is scarce. Clear acceptance criteria are scarce. The ability to observe what happened after release is scarce. Those constraints do not disappear because implementation got cheaper.

In fact, they become more valuable.

That means prioritization itself has to change. Prioritization cannot just answer “is this important?” It has to answer “is this the right use of scarce reviewer and verification capacity?” and “is this urgent enough, valuable enough, and ready enough to justify turning it into change?”

This is where many current AI adoption patterns break down. They assume the backlog is ready. They assume the bottleneck is implementation. They assume more generated code is progress.

Often, it is just more surface area.

The next layer in AI engineering is work selection with hindsight

The strongest response is not “review more PRs better.” It is “stop low-value, weakly shaped, and historically risky work from becoming PRs in the first place.”

That requires a new system upstream of implementation.

A useful model looks something like this:

First, extract a structured summary and weighted keywords from the item being considered, including its title, description, and comments.

Second, find similar delivered work using semantic similarity over those summaries and keywords.

Third, score the item across the dimensions that actually matter to delivery quality: strategic value, customer value, regression risk, security exposure, and readiness.

Then do the most important thing: do not collapse all of that into a vanity score. Use it to recommend a lane.

That lane might be AI Auto for work that is clear, bounded, low-risk, verifiable, and historically well understood. It might be AI-assisted when a human defines the problem and acceptance criteria, but AI implements them. It might be Needs Shaping First when the work is valuable but underspecified. It might be Human Led when risk or exposure is too high. It might be Needs Historical Review First when similar items failed before, and the organization does not yet understand why.

This is very close to how DeliveryTower already publicly frames its model: AI-first, hybrid, discovery, or human-led routing based on clarity, verification strength, and blast radius, sharpened by what actually happened in prior PRs and releases.

Learning from prior delivery mistakes is the missing feedback loop

Most AI development discussions focus on prompting, coding agents, and review workflows.

What is often missing is hindsight.

Not abstract hindsight. Operational hindsight.

What kinds of work led to reverts?
What kinds of changes caused regressions after the merge?
Which items triggered heavy reviewer pushback?
Which ones generated multiple review rounds?
Which ones created a support burden or hidden follow-on work?
Which apparently small changes turned out to be hard to verify safely?

Those are not just postmortem questions. They are intake questions.

A mature AI engineering system should learn from delivery mistakes, verification failures, and review mistakes, in that order. It should use those outcomes to warn the next decision-maker before the work starts.

DeliveryTower's public model explicitly points to these learned signals: historical PR friction, review rounds, regressions, rollback risk, support and rollout follow-on work, and evidence from similar changes.

That matters because the most expensive form of AI slop is not ugly code. It is an organizationally expensive change: work that consumes review effort, creates risk, and still fails to produce enough value.

Better AI engineering starts with shaping, not just routing

One of the biggest opportunities here is not only to say no earlier. It is to shape better.

When work is valuable but not ready, the answer should rarely be "throw it to the agent and see what happens." It should be "improve the work packet."

That means forcing better questions:

What exact problem are we solving?
How will we know this worked?
What is the smallest viable slice?

What could go wrong?
What systems or dependencies are affected?

It also means producing better artifacts before implementation starts:
a clearer problem statement,
explicit acceptance criteria,
a realistic verification path,
and, where needed, a smaller and safer slice.

This is where the long-term leverage appears. In the early days, the main value was better work selection. Over time, the payoff is that a higher percentage of work becomes safely AI-assistable because the organization has learned to shape work more systematically.

In other words, the goal is not to slow AI down. It is to make more work genuinely fit for acceleration.

The business case is stronger than “fewer bad PRs”

The real business case is not just operational hygiene.

It is strategic leverage.

A system like this helps teams align engineering effort more closely with explicit company objectives, allocate scarce review time to safer, more valuable work, and reduce the chance that AI output crowds out more important priorities. DeliveryTower’s public message is already aimed there: it reads strategy and customer context, scores strategic fit and customer value alongside change risk and verification strength, and tries to turn backlog items into delivery-ready work packets before coding begins.

That gives leaders two levels of return.

Early on, they should expect:
higher percentage of started work reaching production successfully,
less wasted review effort,
and fewer regressions and rollbacks.

Over time, they should expect:
a higher percentage of work being safely handled in AI-assisted lanes,
and faster cycle time for valuable, well-shaped work.

That is a much better story than “we generated more code this quarter.”

A better model for AI delivery

The companies that win with AI engineering will not be the ones that merely let agents code faster.

They will be the ones who build a better system for deciding what should start, what should be shaped first, what deserves historical review, what belongs in a safer lane, and where scarce reviewer attention should actually go.

That is the real next layer of software delivery.

Not more automation at the bottom of the funnel; Better judgment at the top.

The future of AI engineering is not more code, but better decisions about what gets built.

About DeliveryTower.com

Delivery Tower is a new product that aims to improve Agentic Engineering outcomes by taking an analytical approach to intake and delivery. It is a partnership between Troy Magennis and Larry Maccherone, both “veterans” of engineering and security processes. If you are interested in working with us as early adopters or have comments, just go to our website and request access or a call. Or email troy.magennis@focusedobjective.com