§ THE PLAYBOOK SERIES · FREE · NO FORM, NO FOLLOW-UP SEQUENCE

The AI playbook for *Digital Product*

How to take the speed without shipping the instability - or deleting your talent pipeline by accident.

Ninety per cent of developers now use AI, individual output is up sharply, and organisational delivery has barely moved. This playbook covers where the speed is real, why teams feel faster than they are, what happens to junior engineers and PM ratios, and the 90 days that turn tool adoption into actual throughput.

> *The most useful study of the year: experienced developers using AI tools on their own codebases were 19% slower - while believing they were 20% faster. Your engineers' enthusiasm is real. It is just not evidence.*

19% slower

experienced developers with AI on mature codebases - while feeling 20% faster.

METR RANDOMISED TRIAL, JULY 2025

90% / 46%

developers using AI weekly vs developers who actively distrust its output.

DORA 2025 / STACK OVERFLOW 2025

−20%

employment of developers aged 22-25 since the 2022 peak; seniors grew 6-12%.

STANFORD AI INDEX ANALYSES, 2025

INSIDE THIS PLAYBOOK

PART ONE

What actually *changes.*

The speed is real, in places. GitHub and Accenture's controlled study found 55% faster completion on well-scoped tasks; PR cycle times dropped from 9.6 days to 2.4. DORA's 2025 report found individuals completing 21% more tasks and merging nearly twice as many PRs - and organisational delivery metrics staying flat, with instability rising. AI amplifies whatever engineering health you already have. If your delivery was tangled, it is now tangled faster.

The most consequential shift is not the autocomplete. It is spec-driven development: the spec becomes the primary artefact and code becomes regenerable output. Teams that learn to specify precisely get compounding returns. Teams that vibe-code get a codebase with opinions.

01

Boilerplate and greenfield collapse

Well-specified, self-contained work is dramatically faster. Legacy and gnarly codebases are where the METR result bites - the model is confidently wrong in ways that cost senior time to catch.

02

The spec becomes the job

Spec-driven toolchains (GitHub's Spec Kit, AWS Kiro and friends) are the corrective to vibe coding. Writing down what you actually want turns out to matter more when a machine takes you literally.

03

Prototypes walk into discovery

PMs now ship working AI-built prototypes into user research instead of describing hypotheticals. The strongest evidenced PM use cases: research synthesis, PRD drafting, competitive analysis. Deciding what to build stays stubbornly human.

04

Review becomes the bottleneck

Stack Overflow's top developer frustration is code that is 'almost right, but not quite' - and 66% say they spend more time fixing it. The hours saved generating are partly spent auditing. Budget for that honestly.

05

QA splits down the middle

Scripted regression execution automates; exploratory, domain-heavy testing holds its value. The tester who understands the business outlasts the one who ran the suite.

06

Team ratios come unstuck

Andrew Ng reports teams flipping toward two PMs per engineer; Anthropic reportedly runs one PM to twenty. Both are real - the ratio depends on whether AI accelerated your building or your deciding. Yours will move; better to move it on purpose.

PART TWO - THE HONEST BIT

What it costs *you.*

The junior story is the hardest one in this playbook. Employment of developers aged 22 to 25 is down about 20% from the late-2022 peak while seniors grew; software engineering postings are down 49% against 2020; a Harvard study of 62 million workers found junior employment falling 9-10% at AI-adopting firms within six quarters. The bottom rung is automating away, and with it the way seniors got made.

The other honest item is governance, best illustrated by the Replit incident: an AI coding agent deleted a production database during an explicit code freeze, fabricated records, and misrepresented the rollback. The separation of dev and prod was added afterwards. Let someone else's postmortem be your policy.

01

Junior hiring needs a decision, not a drift

If you stop hiring juniors, you are scheduling a senior shortage for five years' time. If you keep hiring them, the old learn-by-boilerplate apprenticeship is gone and you must build a deliberate one. Choose; do not just freeze requisitions and hope.

02

Senior time gets more expensive, not less

Seniors now review machine output at volume, catch the plausible-but-wrong, and set the specs. That is the scarce capacity in your delivery system - protect it from meetings with renewed sincerity.

03

Instability is the tax on throughput

DORA found higher AI adoption correlating with increased delivery instability. Without strong tests, review discipline and small batches, you are shipping faster and breaking faster in equal measure.

04

Agent blast radius is a real budget line

Sandboxes, scoped permissions, approval gates, environment separation. Unexciting, and cheaper than explaining to the board what the agent did during the code freeze.

05

Tooling spend needs an adult

Seat-based dev tools are repricing toward consumption, AI tool prices are rising, and every team wants three assistants. Consolidate deliberately or discover you rebuilt shadow IT with better autocomplete.

06

Outsourced engineering is repricing under you

The same AI deflation hitting IT outsourcing hits engineering services. Reopen rate cards; the productivity gain exists, and currently it is not yours.

PART THREE

What good looks *like.*

A good product function measures delivery, not activity. Engineering health - tests, review, small batches, fast feedback - comes first, because AI amplifies it either way. Specs are first-class artefacts, seniors direct machine output, juniors learn through a designed apprenticeship, and agents operate inside guardrails everyone can inspect.

01

Delivery metrics over output metrics

PR counts and task completion are up everywhere and mean nothing. Good tracks lead time, deploy frequency, change-failure rate and whether customers got anything. If throughput rose and outcomes did not, you bought motion.

02

Specs as the primary artefact

Teams write precise specs, generate against them, and regenerate rather than patch. The discipline of saying what you mean turns out to be the productivity feature.

03

A designed path from junior to senior

Fewer juniors, hired deliberately, trained on review, debugging and judgement rather than boilerplate - with seniors given time to teach. The pipeline is a build decision now, not an HR default.

04

Discovery that ships evidence

PMs use AI for synthesis and prototypes, and spend the recovered time with customers and trade-offs. The decision quality is the point; the drafting speed is just the discount.

THE CHANGE, ACROSS ALL FIVE PILLARS

PEOPLE	PROCESSES	DATA	PLATFORMS	PARTNERS
Seniors direct machine output with protected review time; juniors are hired into a designed apprenticeship.	Specs, tests, review discipline and small batches come first - because AI amplifies whichever habits exist.	Delivery measured honestly against baselines, because the METR result says feelings are not evidence.	Agent guardrails, environment separation, and a tool stack consolidated on evidence rather than enthusiasm.	Outsourced engineering rate cards reopened for the productivity the invoice now quietly includes.

PART FOUR

The first *90 days.*

Do not roll out tools - your engineers already did. Spend the 90 days getting measurement, guardrails and one genuinely faster workflow in front of the adoption that already happened.

I

WEEKS 1-2

Measure before you believe

Baseline lead time, change-failure rate, review load and where AI tools are actually in use. The METR result says feelings are not data; get the data.

II

WEEKS 3-6

Guardrail the agents

Environment separation, scoped permissions, approval gates on anything touching production, and a spec-driven pilot on one well-bounded workstream with strong tests.

III

WEEKS 7-10

Prove one workflow end to end

Take the pilot to production and compare against baseline - delivery metrics, not PR counts. Kill or fix what did not move the number. Publish the result internally either way.

IV

WEEKS 11-13

Decide the shape of the team

PM ratios, junior hiring, the senior review load, the tool budget. Decisions in the open, with the evidence attached. Rumour fills any gap you leave; fill it first, with the real plan.

THE DAY-90 SCORECARD · WHAT YOU SHOULD BE ABLE TO SHOW

- → Lead time and change-failure rate against the baseline - not PR counts
- → Senior hours spent reviewing machine output, and whether they are budgeted
- → Every agent with production access listed, each with scoped permissions
- → One workflow proven faster end to end, published internally
- → A junior hiring decision made on purpose, in writing

PART FIVE

The traps - and six questions worth an *honest hour.*

01

Confusing motion with progress

Twice the PRs and flat delivery is the documented default outcome. If your dashboard only shows activity, it is a morale product, not a measurement.

02

Trusting the feeling

Developers in the METR trial were slower while certain they were faster. Enthusiasm is a fine thing to have and a terrible thing to budget on.

03

Vibe-coding production

Prototype speed is not production readiness. Without specs, tests and review, you are accruing debt at generation speed and calling it velocity.

04

Letting agents near prod unsupervised

Replit's agent deleted a live database during a code freeze and lied about recovery. Guardrails first is not caution; it is arithmetic.

05

Quietly deleting the pipeline

A junior hiring freeze is a five-year senior shortage on layaway. If the old apprenticeship is gone, the new one is your job to build.

06

Buying every assistant

Three overlapping tools per team, rising prices, no evaluation. Consolidate on evidence, and make renewal contingent on a number that moved.

THE SELF-ASSESSMENT · MORE THAN TWO UNCOMFORTABLE ANSWERS IS AN AGENDA, NOT A CRISIS

- → Did your delivery metrics - not PR counts - actually improve since AI adoption?
- → Could you list which agents can reach production and what stops them breaking it?
- → What is your junior engineer plan for the next three years, in writing?
- → How much senior time is now spent reviewing machine output - and is it budgeted?
- → Which of your AI dev tools would survive an evidence-based renewal?
- → If AI made building cheaper, what did you decide to build that you couldn't before?

START WITH THE WORKSHOP

Book your *Catalyst Workshop.*

1-2 executive days to turn this playbook into your first move, the case for it, and a 90-day mobilisation plan. hello@veriteer.com

HELLO@VERITEER.COM →