Why coding agents fail to boost productivity?

Nik Tkachev

Head of Product, JetBrains Air

Make it happen. With code.

At JetBrains, code is our passion. Ever since we started, we have strived to make the strongest, most effective developer tools on earth.

JetBrains Today

15M+
users trust our tools

3.9K
new users every day

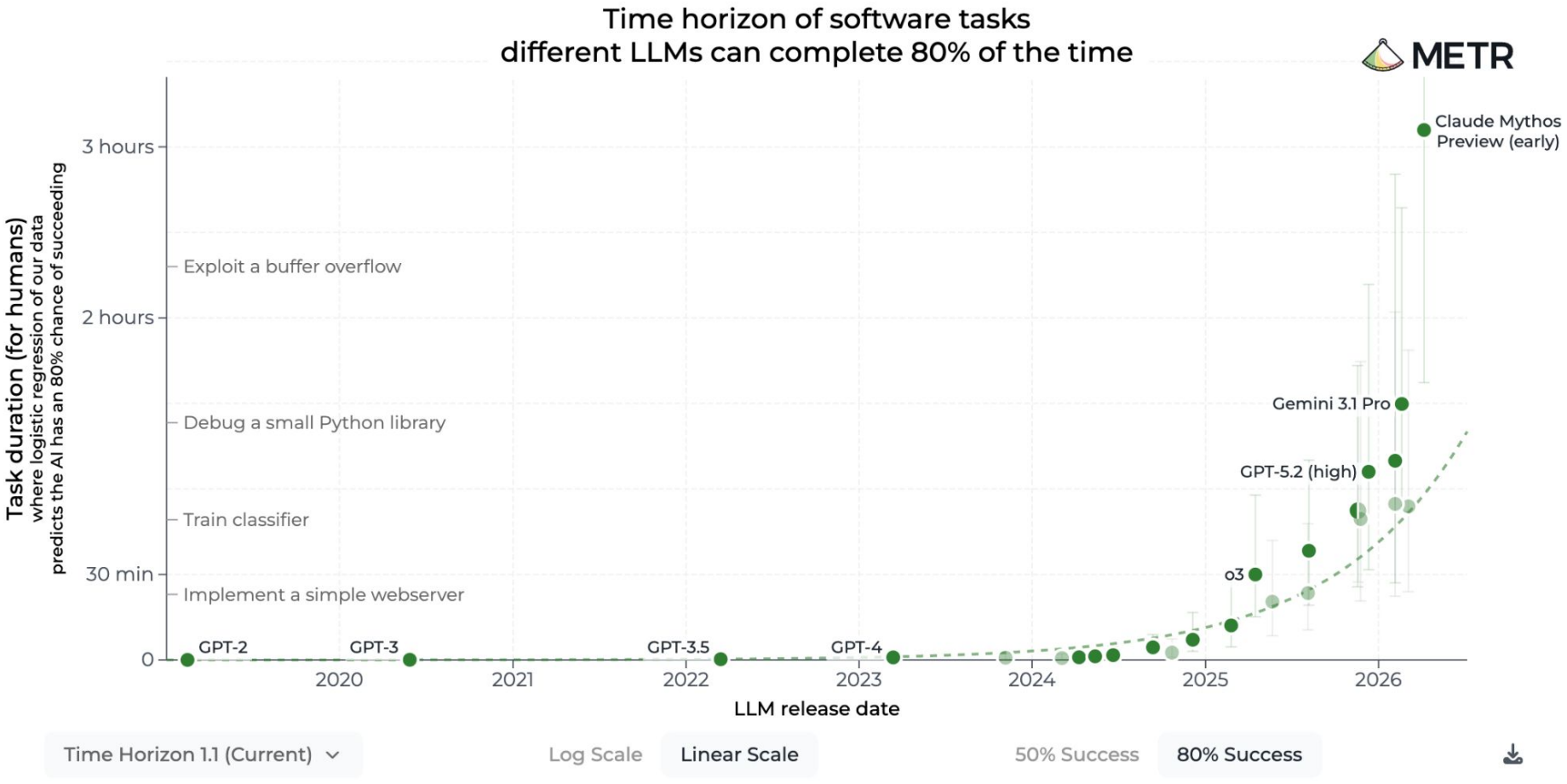
287K
business customers

2,800+
employees worldwide

AI Optimism

AI Realism

AI Pessimism

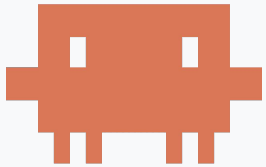


Sources: METR

Define Coding Agent

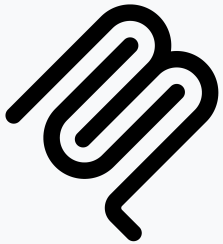
LLM model
(Gemini 3.5 Flash)

- understanding input
- reasoning
- providing output



Harness
(Claude Code CLI)

- collect context
- perform tool calls
- iterate until task is done



Extensions

- MCP tools
- AGENTS.md
- Skills
- ...

Agents are already there

Sources: Pragmatic Engineer. AI Tooling for Software Engineers in 2026, JetBrains Developer Ecosystem Survey 2026 (draft)

Notes: AI tools include chat-based apps like ChatGPT, AI assistants like JetBrains AI Assistant and Copilot, AI-first IDEs like Cursor and Coding agents like Junie and Claude Code

95% Use at least one AI tool at work

80% Use coding agents at least several times a week

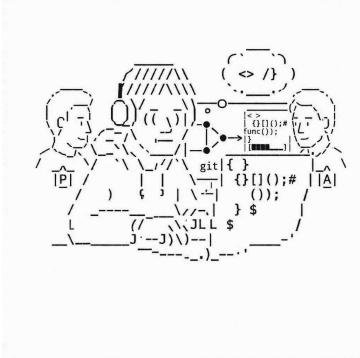
69% Agree that coding agents increased their productivity

High adoption ≠ High productivity

What hinders adoption?

Three perspectives

Individual



Team

Organisation

Individuals

Mental Fatigue

Coding agents can reduce coding effort but increase mental fatigue through gamification-driven overuse, constant decision-making between AI-generated options, and the ongoing burden of reviewing and validating generated code.

Forming new skills

Working effectively with coding agents requires new intuition and orchestration skills. Developers must learn when to trust, guide, or intervene, as an agent may either solve a problem immediately or spend hours stuck in a loop.

Stress

Developers face growing anxiety from fear of missing out on rapid AI advances, concern about agent-driven incidents or mistakes, and fear that relying too heavily on automation could erode their own technical skills over time.

Addictive

→ positively unexpected results lead to a dopamine hit

→ one more prompt to solve it!

→ “you are absolutely right!”

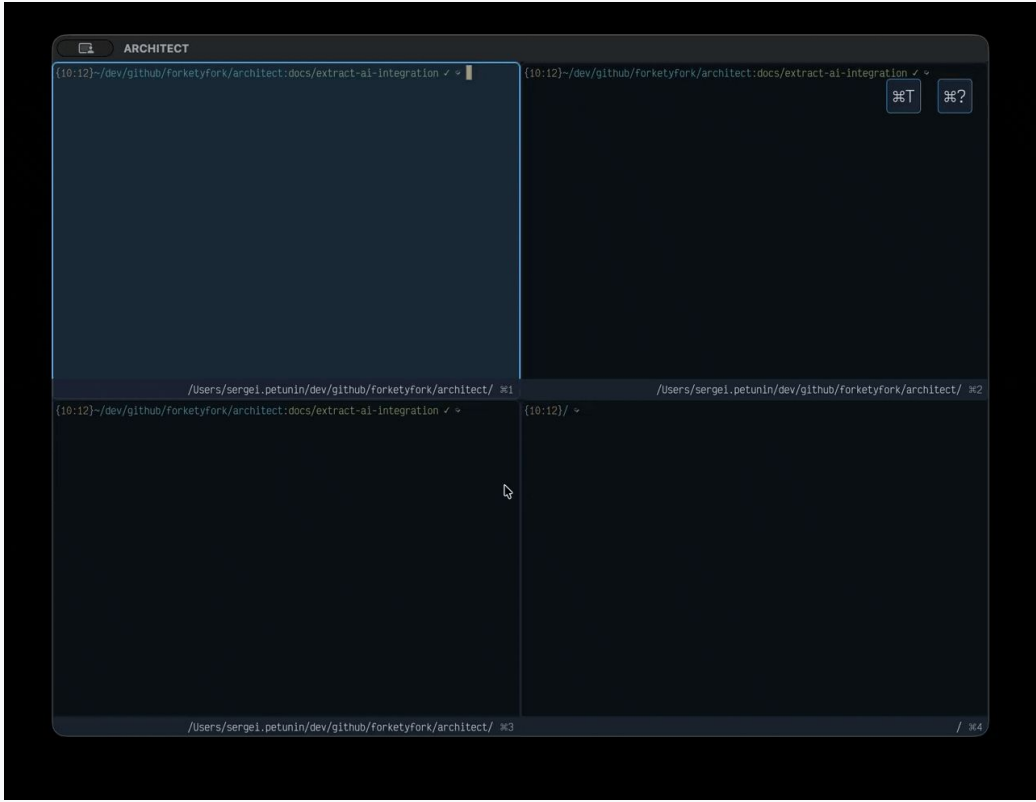
	
You buy **chips**	You buy **tokens**
You spin the slots	You press "Generate"
You might hit the **jackpot**, or **nothing**	You might get a **bug-free app**, or **total garbage** that won't even run
Flashing lights, seductive animation	“Great idea!”, “Of course!”, “Perfect solution just for you!”
I've got my own **strategy**	I'm a **prompt engineer**
One more spin — I'll win it all back!	One more prompt and this bug will disappear
The **casino** is always in profit	The **Cursor** is always in profit
Easy money: I hit the **jackpot**!	Easy coding: I built a **SaaS in 1 day!**
Where did the last 4 hours go?	Wait, did I just spend 4 hours writing prompts for a function I could've written in 20 minutes?

Exhausting

→ Decisions to allow agent to do X

→ Context switching between multiple tabs

→ Review tons of AI generated results



Architect, pet-project of one of Air developers


GEMINI

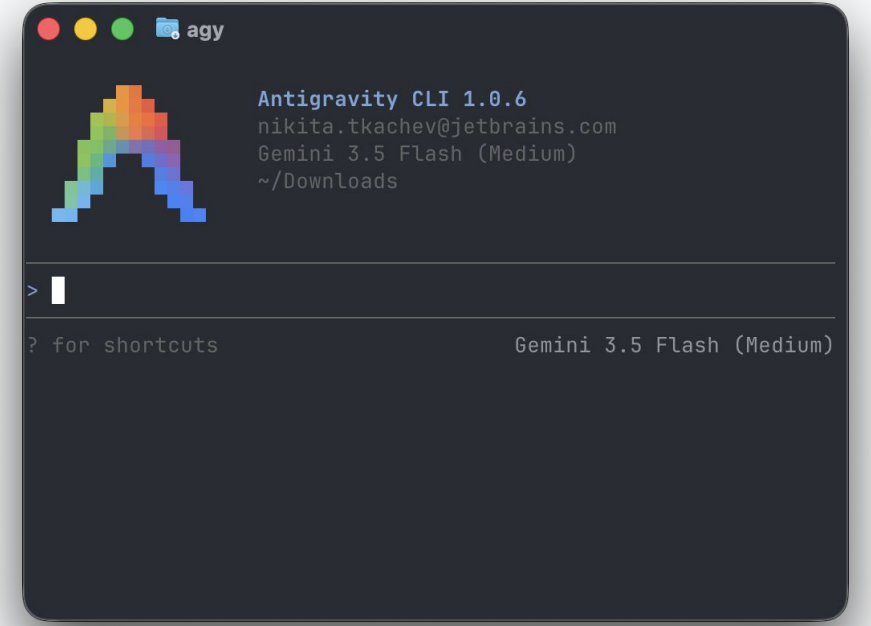

Budovicium - $0.01/$0.02 USD - No Limit Hold'em
Deflotte - $0.01/$0.02 USD - No Limit Hold'em
Alcor III - $0.01/$0.02 USD - No Limit Hold'em
Laodica III - $0.01/$0.02 USD - No Limit Hold'em
Eurymedon III - $0.01/$0.02 USD - No Limit Hold'em
Antiope III - $0.01/$0.02 USD - No Limit Hold'em
Alcith III - $0.01/$0.02 USD - No Limit Hold'em
Imperatrix II - $0.01/$0.02 USD - No Limit Hold'em
Helenos II - $0.01/$0.02 USD - No Limit Hold'em
Beograd II - $0.01/$0.02 USD - No Limit Hold'em
Araunocia - $0.01/$0.02 USD - No Limit Hold'em

Skill demanding

→ “Ask me anything” onboarding

→ Steep learning curve

→ Planning development instead of developing by plan



“There was a learning curve in a sense of what is possible, which use cases are good for AI, and which are not. I invested a lot of my time figuring out what is working and what is not...

currently, after **6 months**
I have an impression I more or less understand...”

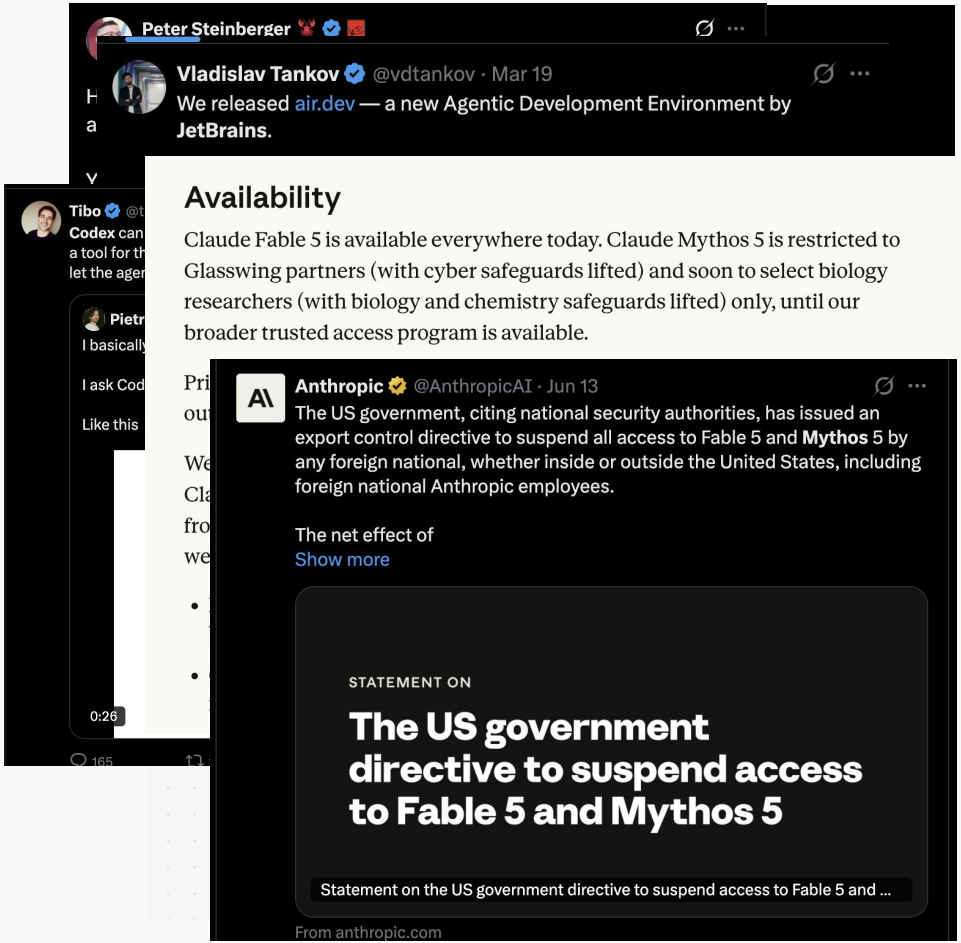
Senior engineer, Finance industry

Unsettling

→ I'm so behind! (FOMO)

→ What if the agent screw up in prod?

→ What if I forget how to code?



Break free with delegation

Developers are able to delegate to agents once they learn, build trust, and have proper tools:

Completion

Ask the agent to provide an isolated code snippet directly to the IDE editor pane. Accept / Decline code.

Prompting

Ask the agent to perform a task with editing rights to files in the project. Step by step steering.

Goal

Ask the agent to achieve a target state with a set of constraints. Fire and forget, review results.

Automation

Same as fire and forget but with a trigger to start the task without explicit user input.

What to do?

Mental Fatigue

⟶ Keep only one complex task at a time
⟶ Make skills for workflows which are repetitive and doesn't require much steering

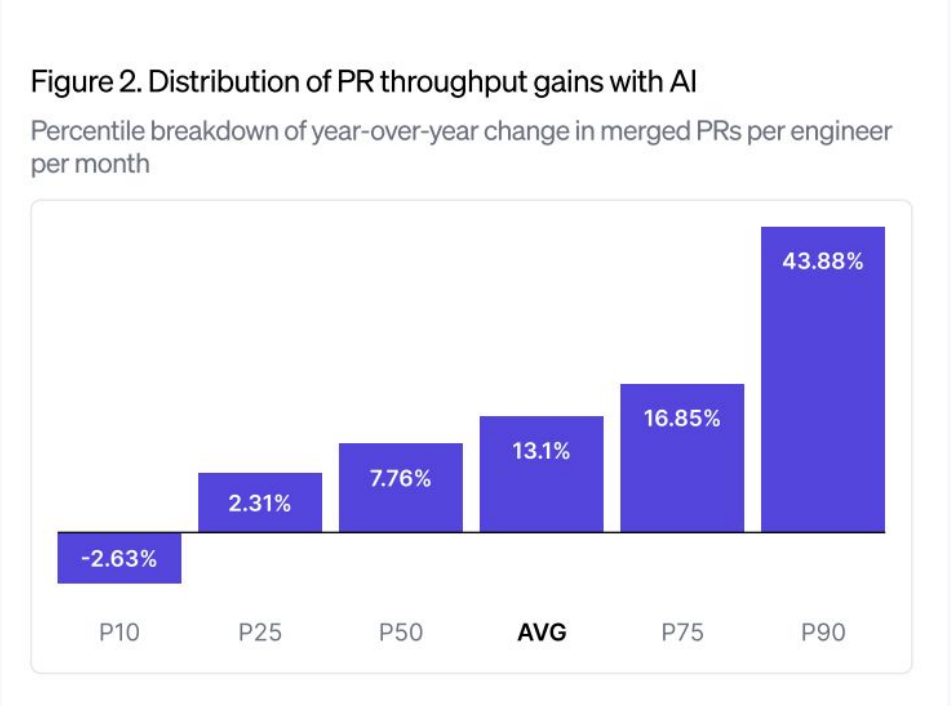
Forming new skills

⟶ Treat agent as a pair programmer, discuss, plan, design before implementing
⟶ Try things you already do but with agents, build your intuition

Fears

⟶ Save new tools and methods to try later in one go
⟶ Assume agentic engineering will become a basic skill
⟶ Have fun

Teams are not getting 10x productivity



Sources: DX longitudinal study of 400+ companies (2024-2026)

Teams

Bidirectional pressure

Expected to deliver more, faster, and with minimal additional cost through coding agents, while remaining fully accountable for software quality, delivery predictability, and business outcomes across the entire SDLC.

SDLC breaks

Without new clear processes and governance, teams risk a zoo of agents, inconsistent workflows, and a flood of uncoordinated, low-context pull requests.

Shipping wrong things

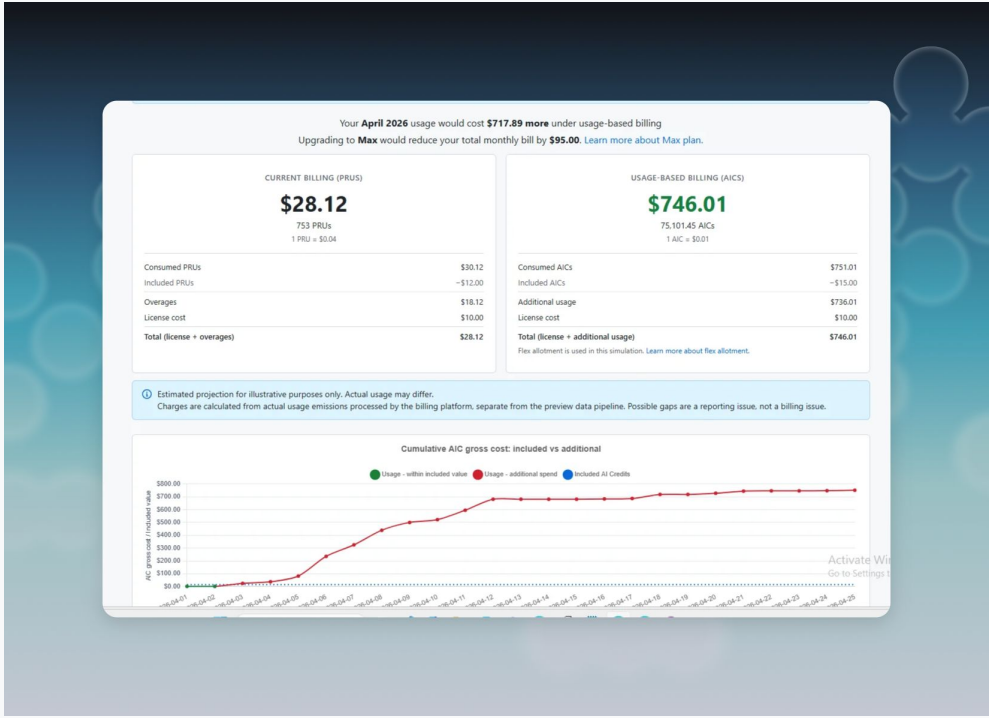
When code becomes cheap, teams risk shipping the wrong things faster. Agent-driven work can reduce discussion, shared understanding, and ownership, leading to parallel execution without clear alignment on the underlying problem.

End of subsidies

→ Most team workflows are pushed to API pricing

→ Budget management going down to teams

→ Measurement moves from people to initiatives/workflows/specific tasks

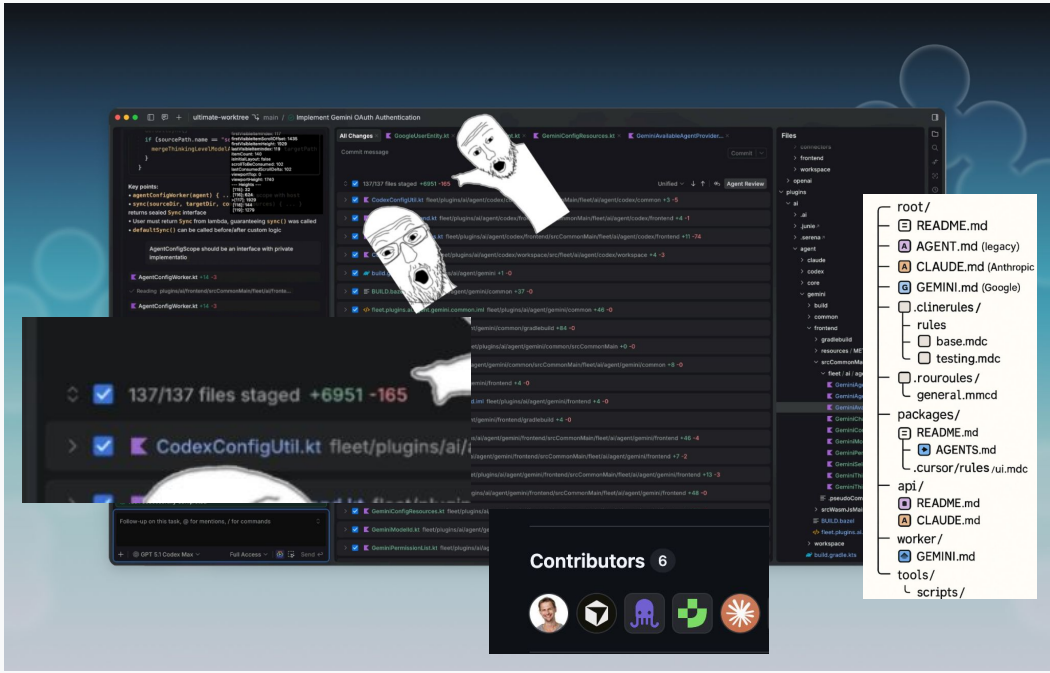


Agentic Zoo

→ Agents are maintained in silos, configs start to interfere (skills, instructions, tools)

→ Issues are caught only on review, no clear process to encode the feedback into the agent setup

→ Agents aren't connected to all the tools the developer uses or connected without security in mind

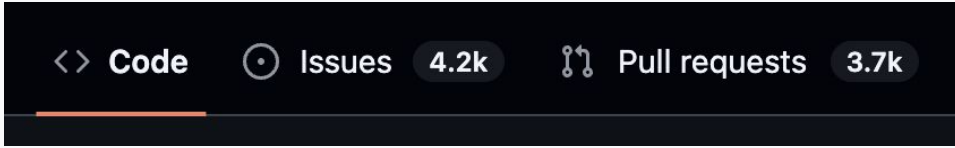


“Velocity” trap

→ Not everything in the backlog worth developing

→ Unless review stage is automated it create a significant bottleneck on the later step instead of improving the velocity

→ Code is a liability, each new thing increase the cost over time



Team maturity model

Teams are evolving through stages:

L1 Exploring

A few people experimenting.

Nothing shared across the team.

L2 Scaling

Most of the team uses AI.

No shared setup, no shared practice.

L3 Practicing

Shared workflows exist.

Someone owns them.

L4 Optimizing

Agents are a team capability.

Metrics-driven.

Team maturity model

Teams are evolving through stages:

L1 Exploring

A few people experimenting.

Nothing shared across the team.

L2 Scaling

Most of the team uses AI.

No shared setup, no shared practice.

L3 Practicing

Shared workflows exist.

Someone owns them.

L4 Optimizing

Agents are a team capability.

Metrics-driven.

Most of the teams are stuck there now

Once you pass the individual adoption, you get a complete new challenge

Champions exist but can’t scale

Someone figured it out. The team can't inherit it yet.

SLDC has to adapt to advance

Investment is required before scaling

Problems L2 teams are dealing with

REMEMBER — Context & Knowledge

- Prompting skills stay individual because nobody shares what works
- Each person's agent writes code differently
- Everyone solved the same setup problems alone
- Process knowledge lives in someone's head
- Fixing shared agent setup is important but never urgent enough to do

CONNECT — Visibility & Collaboration

- PRs pile up faster than reviews
- Bugs look right until production
- Good engineers accelerate, weak ones create landmines
- Agent writes too much code, so review becomes cutting instead of checking
- Coding got faster but delivery didn't
- Agents suggest destructive production changes that look routine

ACT — Monitoring & Automation

- Monitoring noise becomes someone's job
- Small operational fires break focus
- One owner becomes the system's single point of failure
- Repeated manual work drains attention
- Logs hide what actually happened
- Team coordination stays manual despite agent's adoption

What to do?

Bidirectional pressure

⟶ OWN the token spend AND the impact
⟶ Productive agentic workflows should surface on the team level and cost tracked
⟶ Group agentic sessions with real product/business tasks (e.g. Jira issue) to evaluate the impact

SDLC breaks

⟶ Treat agent configuration as code. Clear ownership, versioning, guidelines
⟶ Have a chat and discuss key architecture decisions and pitfall to avoid, ask an agent to extract guidelines
⟶ Assess agent access regularly and create safety-nets for failures

Shipping wrong things

⟶ Invest into maintenance first to reduce toil
⟶ Understand how to build faster consistently (repeatable workflows)
⟶ As more checks and quality gates are in place, the review becomes faster

Organisations

Shadow AI

An agentic zoo can rapidly increase tooling and infrastructure costs while reducing visibility, governance, and organizational control over how AI is used.

Agent integration

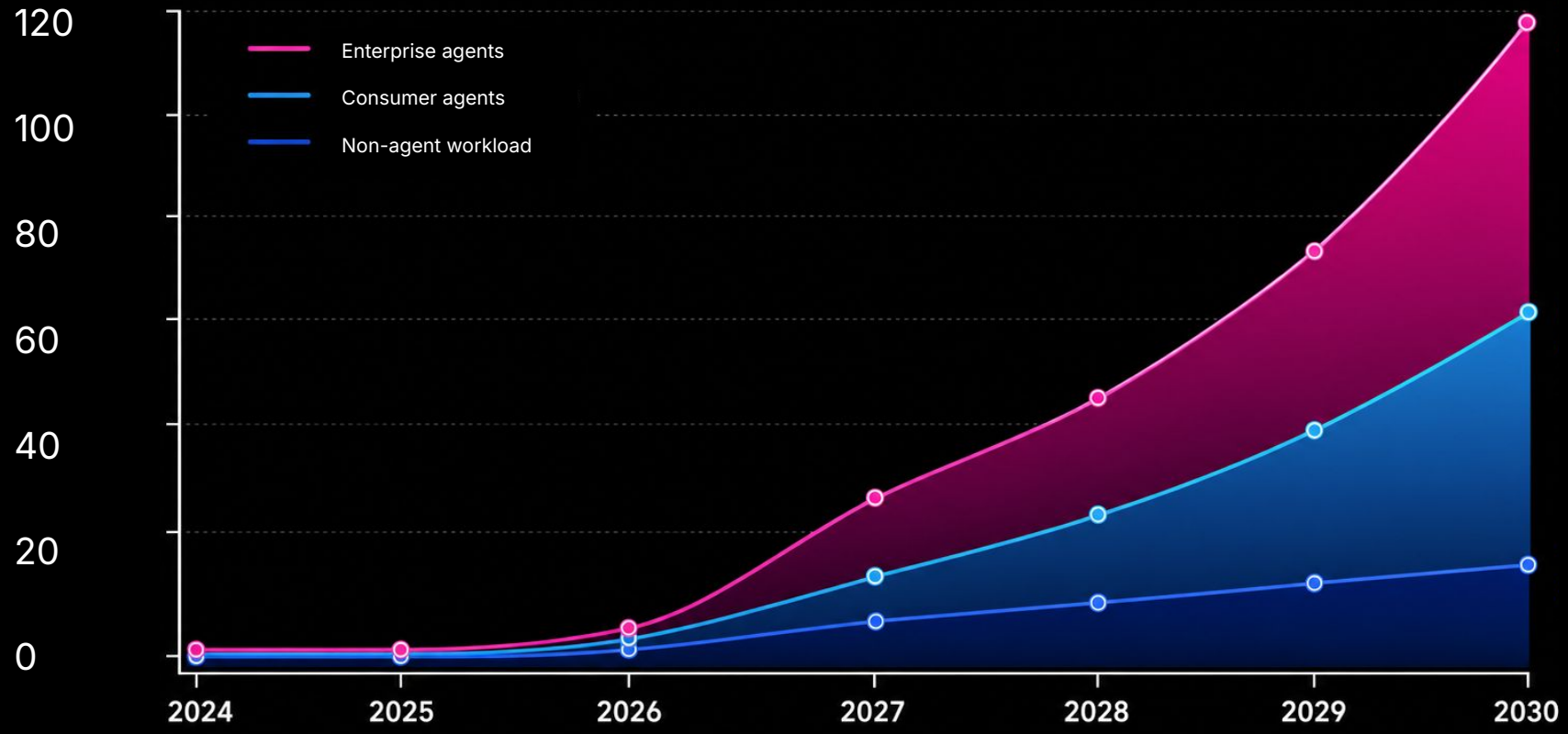
Integrating agents into the SDLC requires rethinking tooling and infrastructure, enabling agents to operate independently of developers while remaining connected to the systems, workflows, and controls they need.

ROI

As AI adoption scales, measuring ROI becomes critical. AI usage can quickly become a major cost center for software-intensive organizations, making visibility into value creation essential.

Token use by AI agents is expected to multiply 24 times by 2030

Estimated monthly token count for agentic AI applications, in quadrillions



Source: Goldman Sachs Research

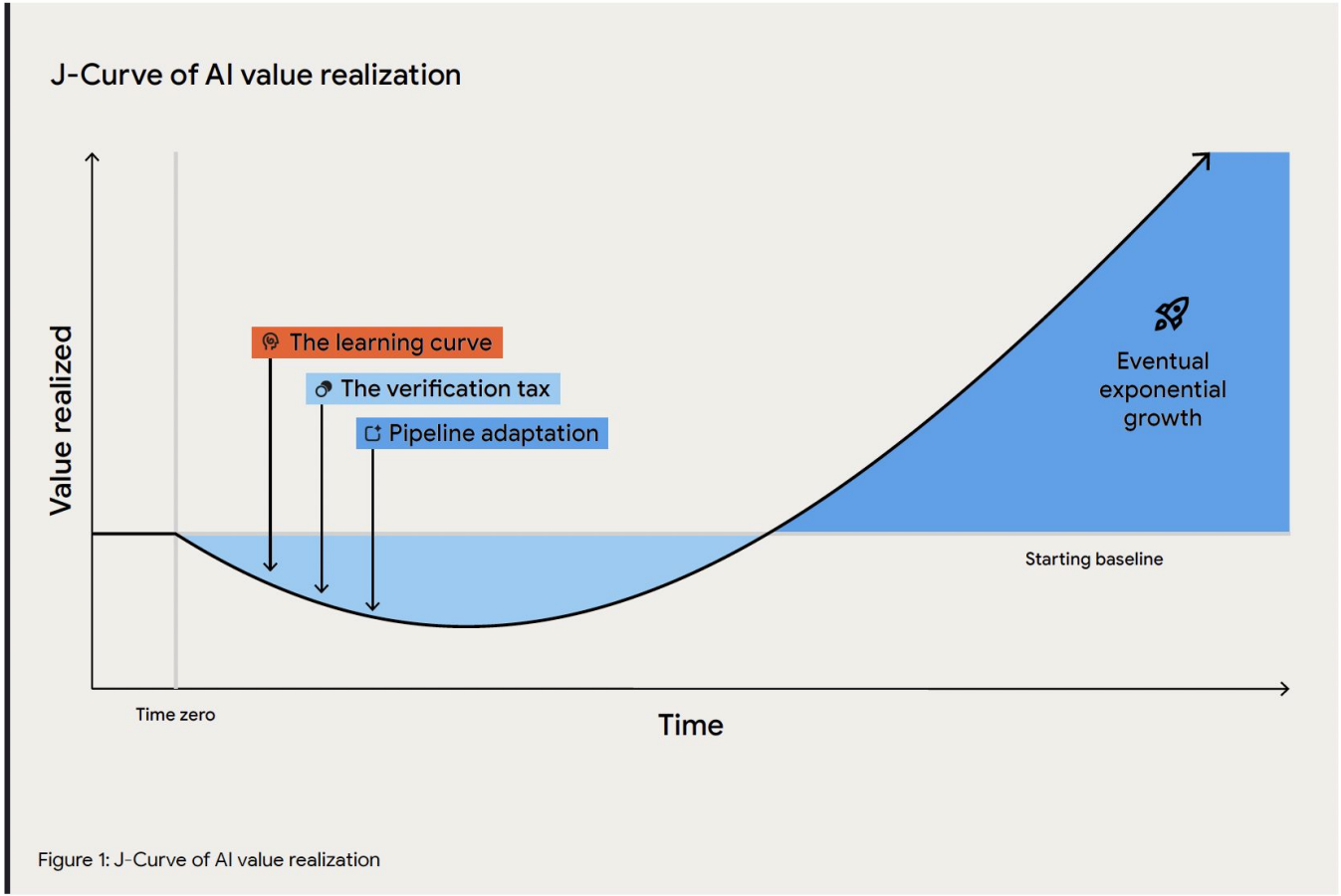


Figure 1: J-Curve of AI value realization

What to do?

Shadow AI

⟶ Commit to one vendor or embrace the vendor-agnostic approach. There is no middle ground on scale
⟶ Make it public, share clear guidelines, do’s and don'ts
⟶ Give a mandate to experiment if you want real upsides

Agent integration

⟶ Tool integration is a security and compliance process, be proactive to drive the clearance
⟶ Agents require monitoring and observability, make sure you can trace what is happening

ROI

⟶ Avoid AI theater and token spend leaderboards
⟶ Make sure to measure e2e metrics to avoid bottleneck traps
⟶ Promote ownership for the impact, not code delivery

JETBRAINS

Be among the first to try JetBrains Central.

The control and execution plane for agent-driven software production.

Nik Tkachev
Product, JetBrains Air