The Agentic Coding

White Paper

How we teach teams to actually ship faster with AI

Cursor Workshop

cursorworkshop.com

Most teams get it wrong

They adopt Cursor or Copilot. Juniors generate code they can't verify. Seniors get marginal speed bumps. Nobody agrees on what the AI should handle versus what requires human judgment.

Jellyfish ran the numbers. Senior engineers are 22% faster with AI tools. Juniors? 4% faster. The tool is the same. The gap is how people use it.

AI multiplies existing skill. If you're a 10, you become a 15. If you're a 3, you become 3.2.

So how do you get everyone closer to the 15?

Three zones

Every task in agentic coding falls into one of three buckets.

Delegate to AI

The AI does the work. You give it a prompt, it executes.

Examples: boilerplate, test scaffolds, migrations, docs, log analysis.

Why seniors are better at this: they write tighter prompts with less ambiguity. The AI has less room to guess wrong.

Review

The AI proposes. You verify. You're the quality gate.

Examples: checking AI code for edge cases, security holes, architectural fit, meaningful test coverage.

Why seniors are faster: they know what "correct" looks like. They spot problems in seconds. Juniors need investigation time.

Own

The AI can't decide this. Human judgment is the moat.

Examples: product vision, architecture trade-offs, risk acceptance, the AGENTS.md that governs how AI works in your repo.

The more you own the right things, the more you can safely delegate everything else.

Seven phases

Apply the three zones across your workflow:

Phase	Delegate	Review	Own
Plan	Draft PLAN.md, map dependencies, slice stories	Validate scope and estimates	Vision, trade-offs, sign-off
Design	Suggest components, generate skeletons	Check accessibility, performance	Design system, pipeline choices
Build	Generate IaC, Dockerfiles, CI	Audit secrets, IAM, cost	Architecture, security policy
Test	Generate Playwright/Vitest tests	Reject over-mocked tests	Strategy, data seeding
Review	Auto-review via AGENTS.md rules	Verify intent, audit drift	Rule maintenance, risk acceptance
Document	Generate AGENTS.md, changelogs	Curate accuracy	Doc structure, update policy
Deploy	IaC execution, tail logs	Secrets audit, billing check	The pager, feature flags, stop button

---

Four things you can do today

Create an AGENTS.md file

AGENTS.md is the new standard for giving AI agents context about your codebase. Think of it as a README for agents. Over 60,000 open-source projects already use it.

The file tells agents how to build, test, and write code in your repo. Put your conventions in one place: "Always use Zod for validation. Run tests with pnpm test. Follow the patterns in src/components."

The real advantage: one file works across every major AI tool. Claude Code, OpenAI Codex, Cursor, Copilot, Devin, Aider, Jules, Zed. Write it once, and every agent your team uses follows the same rules. No more per-tool configuration.

Keep a CHANGELOG.md for agents

Agents lose context between sessions. When you start a new chat, the AI has no idea what you refactored yesterday or why you made certain decisions.

A running CHANGELOG.md (or refactor.md) fixes this. After each significant change, document: what changed, why, which files, any follow-up tasks.

When you start a new session, the agent reads the changelog and picks up where you left off. No more re-explaining context. Works across tools, works across team members.

Break work into small units before prompting

Don't ask for "the entire feature." Ask for one verifiable piece at a time.

Bad: `"Create a user authentication system"`

Good: `"Create a JWT validation middleware that extracts user_id from the Authorization header and returns 401 for invalid tokens"`

Give juniors verification checklists

If they can't spot issues by instinct, give them explicit checks:

- No hardcoded secrets
- Error states handled
- Tests cover failure cases
- Permissions are least-privilege

Protect the gym

Some skills shouldn't be delegated. Problem decomposition, debugging intuition, system design, trade-off judgment.

Let the robot lift heavy things at work. Keep training the muscles that matter.

What happens when teams apply this

We've trained teams at Fortune 100 and top EU companies. What we see:

- 40% faster velocity within the first month
- Cleaner codebases (AGENTS.md enforces rules across all tools)
- Fewer review cycles
- Faster onboarding for new engineers

The framework works. The question is whether your team knows how to use it.

What you learn in training

This cheat sheet is the surface.

In our on-site and off-site programs, your team will:

Learn the tools

Cursor (Tab, Composer, multi-file context, MCP servers), Claude Code (terminal-native workflows), Codex (background agents at scale).

Build the infrastructure

Write an AGENTS.md that works across Cursor, Claude Code, and Codex. Set up MCP servers for GitHub, Postgres, AWS, Figma. Create templates your team can reuse.

Develop the judgment

When to delegate versus own. How to verify AI output quickly. How to train juniors without creating dependency.

Ship on your codebase

We don't use toy examples. You apply the framework to real code and leave with internal playbooks.

Next step

Explore training programs

cursorworkshop.com/training

On-site sessions at your office. Executive offsites for deep immersion. Enterprise programs for larger teams.

We respond within 24 hours.

About Cursor Workshop

We've trained engineers at Fortune 100 companies, high-growth startups, and enterprise teams across Europe. Our workshops consistently rate above 4.8/5.

Rogier Muller
Co-founder. Background in AI strategy and developer tooling. Previously built products used by thousands of engineers.

Vasilis Karaiskos
Co-founder. Deep expertise in agentic workflows and enterprise engineering practices.

Get in touch

Email: hello@cursorworkshop.com
Web: cursorworkshop.com
LinkedIn: linkedin.com/company/cursorworkshop

Cursor Workshop trains engineering teams to ship faster with AI.