ENTERPRISE GUIDE

The Enterprise Guide to AI-Assisted Development

AI coding agents are already showing up in real work. The harder question is whether the organization can use them in a way that holds up under pressure.

Three problems slow engineering teams down. One operating model fixes them. One team has installed it across 100+ teams.

40%
velocity gain in the first month

95%
team-wide adoption after the workshop

100+
teams trained worldwide

PROBLEM

The cost of getting AI adoption wrong

AI coding tools are already inside your engineering organization. The question is not whether people use them. It is whether the way they use them is making you faster or creating new problems nobody is measuring yet.

Review becomes the bottleneck

Engineers produce more code, faster. Reviewers cannot keep up. Senior people spend their time checking AI-generated output instead of making architecture decisions.

Quality drifts without anyone noticing

AI-generated code looks clean before it is actually sound. Teams confuse polish with judgment and only see the damage later.

Adoption is uneven, so the gains are uneven

Research shows senior engineers get 22% faster with AI tools while juniors get only 4% faster. Without one standard, leverage stays concentrated.

These are not tool problems. They are management problems. The organizations that fix them do not buy a better tool. They install a better operating model.

FRAMEWORK

Delegate. Review. Own.

Everything we teach comes from one observation: the teams that get reliable, repeatable value from AI-assisted development are the ones that make three boundaries explicit.

DELEGATE

Hand off repetitive tasks entirely: boilerplate, migrations, tests, documentation, and logs. Trust but verify.

REVIEW

The agent proposes, you decide: bug fixes, refactoring, optimization, edge cases, and test quality.

OWN

Core architecture, security decisions, business logic, standards, and risk acceptance stay human.

Once the whole team shares these boundaries, agentic development stops being an individual experiment and becomes a team capability.

APPLICATION

Applying the framework across development phases

Delegate-Review-Own applies across the entire software development lifecycle. The table below shows where agents help, what engineers must verify, and what must stay firmly owned.

PHASE	DELEGATE	REVIEW	OWN
Plan	Summarize inputs, draft options, map dependencies	Check assumptions, scope, and constraints	Vision, trade-offs, approval
Design	Generate component skeletons	Check accessibility and performance	System design, architecture
Build	Infrastructure code, configurations, boilerplate	Audit security and cost implications	Architecture decisions
Test	Generate test suites and draft coverage	Reject shallow coverage	Test strategy, data approach
Review	Automated style and pattern checks	Verify intent and audit drift	Standards, risk acceptance
Deploy	Execute deployments, monitor logs	Audit configurations	Release decisions, incident response

When the team disagrees on a cell, that cell should default to Own until standards are clarified.

PRACTICES

The six operational practices

These practices come directly from the workshop curriculum and have been tested across more than 100 teams.

PRACTICE 1

Establish documentation for AI

Agents get better when conventions, testing expectations, repository structure, and business constraints are written down.

PRACTICE 2

Write plans before prompting

Strong teams shape the work first, then ask the agent to help execute it.

PRACTICE 3

Reset early when output goes wrong

AI-generated code is cheap to produce and expensive to repair. Restart weak drafts sooner.

PRACTICES

The six operational practices

These practices come directly from the workshop curriculum and have been tested across more than 100 teams.

PRACTICE 4

Decompose work into small units

One prompt should map to one independently reviewable task.

PRACTICE 5

Use verification checklists

Review gets easier when security, permissions, failure states, and test quality are explicit.

PRACTICE 6

Protect core engineering skills

Architecture, debugging depth, prioritization, and judgment under uncertainty remain human.

WORKSHOP

What the workshop delivers

The white paper explains the operating model. The workshop is where the team applies it to real work on the real codebase.

AREA	WHAT HAPPENS
Real workflows	We start with planning, implementation, review, QA, or release work that already carries pressure inside the organization.
Your codebase	The team applies Delegate-Review-Own to its own codebase using AI coding agents, under NDA, with a minimum of two trainers.
Carry-over	We leave behind review checks, planning patterns, handoff rules, and usage standards so the practice survives after the session.
Execution-first delivery	We focus on applied implementation, not abstract AI theory or generic templates disconnected from your stack, controls, and review standards.

Hands-on application over passive observation. The team leaves with operating patterns, review controls, and decision language it can use immediately.

NEXT STEP

What happens next

If the fit is real, the next step is usually a short strategy session to assess workflow maturity, review bottlenecks, and where Delegate-Review-Own should first be installed.

PERSON	WHAT THEY BRING
Vasilis Tsolis	Co-founder and expert in AI document intelligence and agentic engineering, focused on enterprise AI systems and trustworthy workflows.
Rogier Muller	Technology executive, founder, and Anthropic Ambassador focused on how teams adopt AI without losing control of quality, review, and risk.

The goal is simple: better execution, cleaner review, and a team-wide standard that holds up under delivery pressure.