MADELABS TECHNOLOGY

AI Speed. Engineering Rigor.

PLAYBOOK

The AI Enablement Playbook

Adopting AI in product and engineering teams, from Fear Mapping to the first 90 days

Nicholas Barger, CEO & Co-Founder · Adam Broadbent, CTO & Co-Founder
Published September 25, 2026

CONTENTS

Adopting AI in product and engineering teams

01 Why does AI adoption stall on product and engineering teams?

02 What happens in a Fear Mapping session?

03 What makes AI training stick for engineers?

04 Why do teams need AI office hours between workshops?

05 What are skills, integrations and automations, and why share them?

06 How should AI-assisted engineering work be governed?

07 How do you know whether AI adoption is working?

08 What should the first 30, 60 and 90 days look like?

09 How does MadeLabs help teams put this into practice?

Product and engineering teams adopt AI well when the rollout starts with what the team is worried about, trains on the team's own work, and puts security and review rules in writing from the first month. This guide covers each step as we run it in AI Enablement for Product & Engineering: Fear Mapping, hands-on workshops, AI office hours, shared skills and automations, governance, how to measure progress, and a plan for the first 90 days.

01

Why does AI adoption stall on product and engineering teams?

It stalls because most rollouts deal with the tools and skip the people using them. Engineers have specific, reasonable concerns about AI, and until those are said out loud they quietly decide how much the tools get used.

The usual pattern is easy to recognize. Licenses are bought, a demo is scheduled, and a handful of enthusiasts start shipping faster. Everyone else keeps working the way they did, or tries the tools privately and says little about it. Six months later leadership can see the invoice and can't see the return, and the engineering leader is asked to explain a gap they can feel but can't yet name.

Our Fear Mapping sessions group the concerns engineers raise into eight categories:

Job Displacement: "Will my role still exist?"

Skill Atrophy: "Am I losing the craft?"

Work Ownership: "Whose work is it, really?"

Quality & Trust: "Can I trust what it produces?"

Visibility & Credit: "Does my work still show?"

Management Pressure: "Expectations are shifting."

Speed vs. Quality: "Faster, but worse?"

Surveillance: "Am I being measured by it?"

Only two of the eight are about whether the technology works. Quality & Trust and Speed vs. Quality are engineering questions, and they have engineering answers: review practices, testing, and clear rules for where AI-generated code may go. The other six are about careers, credit and how the team will be managed. A training course won't settle those. What settles them is leadership saying what it will and won't do, and then doing it.

For a CEO this matters because a stalled rollout looks like a tool that didn't work, when the real problem is trust. For a CTO it matters because the engineers who hold back are often the most careful ones, the same people you rely on to catch problems in review. If they are sitting it out, the team's AI-assisted work is being shaped by whoever is least cautious.

02

What happens in a Fear Mapping session?

Your developers spend about 90 minutes naming their concerns about AI, anonymously, then grouping and ranking them together. The session produces a Fear Map: the team's concerns by theme and intensity, and the conditions the team says must be true for AI adoption to feel right.

A session runs remotely. A facilitator from our team shares a screen, and participants join from a phone or laptop with a six-character code. There are no names, no logins and no device IDs, so the tool cannot link a note to a person. We say that out loud at the start, because people need to hear it before they will write anything honest.

The session moves through six phases:

Welcome. Ground rules and the anonymity guarantee. One rule matters most to the room: what's said shapes the engagement, and it never touches anyone's performance review.

Capture. Silent writing, one concern per note, prompted by questions such as "What's the one thing about AI in your work that worries you most?" and "If AI becomes a major part of how your team works, what do you personally stand to lose?"

Group. The notes are clustered into themes and a few are read aloud, word for word. Hearing their own words, unattributed, is often when a room relaxes.

Depth. Open discussion of the two or three largest clusters, including the question that matters most: "What would need to be true for you to feel good about how AI is being introduced here?"

Prioritization. Everyone gets five votes, with no more than three on any one theme. Votes show intensity as well as frequency, so a small group with a strong concern still shows up.

Close. The Fear Map is generated in front of the room, and the facilitator reflects the top themes back and names one concrete commitment.

Leadership receives a client version of the map: the takeaways, the intensity of each theme, and the conditions for success in the team's own words. When several teams take part, their maps are combined into one view, so you can see where one team's concerns differ from another's without anyone being identified.

The conditions for success are what make the session worth running. They turn a list of worries into commitments the rest of the engagement is measured against, such as agreeing that AI-generated code is reviewed the same way as everyone else's, or that nobody's output is compared by how often they use an assistant. When your engineers can see their conditions being met, adoption stops depending on persuasion.

03

What makes AI training stick for engineers?

Training sticks when it is built on the team's own work and asks people to commit to an answer before they see it. Generic demos rarely carry over to real tickets; a solved problem from your own backlog does.

Our workshops are mostly virtual, and virtual training has an obvious enemy: the laptop the attendee is watching on is also the thing distracting them. So we make the device part of the session. Participants join a companion app on their phone or laptop and answer at the moments that matter.

Two techniques do most of the work:

Prediction. Before a result is revealed, everyone commits to a guess, such as how long a task will take with an assistant or which request it will refuse. Being wrong in public, among colleagues, is what makes the lesson memorable.

Retrieval practice. Short recall checks, including a warm-up on the previous workshop, so the ideas are recalled rather than only re-read.

Hands-on exercises include a simulated AI coding terminal, where engineers type real commands and work through ordered objectives while the room watches progress on the shared screen. It lets people practise with an assistant before they point one at production code.

The topics come from the Fear Map. If Quality & Trust ranked highest, the first workshop is about reviewing AI-generated code, not about writing more of it. If Skill Atrophy ranked highest, it is about using the assistant to learn an unfamiliar part of the codebase. Either way, workshops are hands-on, with simulated Claude sessions, quizzes and surveys, and they build foundational AI skills for engineers and product professionals.

04

Why do teams need AI office hours between workshops?

Because most real questions come up in the middle of a task, weeks after the workshop that would have answered them. Office hours are a weekly, one-hour open drop-in that gives your engineers time with our engineers while the work is in front of them.

A workshop can teach a pattern. It can't anticipate the moment an engineer is halfway through a migration and unsure whether the assistant's suggestion is safe, or wants to know whether a particular file can go into a prompt at all. Without somewhere to take that question, most people either guess or stop using the tool for that kind of work.

The questions we hear tend to fall into three groups:

Permission. Is this tool approved for this data? May an agent run this command? These go back into the practices we manage with your team, so the next person doesn't have to ask.

Technique. How do I get the assistant to follow our conventions? Why does it keep inventing an API that doesn't exist? These often become a shared skill.

Judgment. Should this be AI-assisted at all? Some work is better done by hand, and saying so is part of using the tools well.

We treat the mix of questions as a signal. Early on, most are about permission. As practices settle and skills are shared, more of them are about technique and judgment, which is the sign that people are using the tools in their real work.

05

What are skills, integrations and automations, and why share them?

They are the reusable pieces that turn one engineer's good technique into the team's standard way of working. When they live in your repositories, they are reviewed like code, and your team keeps them.

Skills are written instructions an AI assistant loads when it needs them: your component conventions, how your team writes a migration, the checks a pull request must pass. They let the assistant follow your standards, and not the average of the internet.

Integrations connect the assistant to the systems your team already uses, such as the repository, the issue tracker and internal documentation, with access scoped to what the task needs.

Automations are the repeated jobs done without anyone asking: drafting a plan for a well-specified ticket, checking a change against your rules, summarizing what a release contains.

Sharing is where most of the value is. An engineer who has worked out how to get reliable results on your codebase usually keeps that knowledge in their own prompts. Once it is written down as a shared skill, the whole team gets the same results, new joiners get them from their first day, and the practice can be reviewed and improved like any other code.

We work this way ourselves. Our own codebase carries skills for component conventions and styling, rules that load automatically when an engineer or an assistant opens billing or permissions code, and automated checks that fail a change which breaks an agreed rule. That structure is what we help your team build around your own work.

What you build together stays with your team. Every skill, integration and automation lives in systems you own, and the aim is that your engineers extend them without us.

06

How should AI-assisted engineering work be governed?

We apply security settings through managed settings for Claude, refining how it's used with allow and deny lists, alongside agreed practices for tool access and code review. Governance runs alongside every other step; it isn't a phase you get to later.

Governance done early means real controls are in place before usage grows. Done late, it usually means an incident review followed by a ban. The practices we agree with teams cover three areas:

Data handling. What may and may not go into a prompt, and which tools are approved for which kinds of data.

Tool access. What an assistant or agent is allowed to read, change and run, scoped to the least access that does the job. An agent that can open a draft pull request has no need to merge one.

Review. How AI-generated code is reviewed before it merges. Our own rule is that an engineer peer-reviews the AI-generated code we ship, and it is the standard we recommend.

A fourth practice is worth copying from how we run our own systems. Every AI capability in our client portal and internal tools is registered in one place, and an administrator can switch any of them off from a settings page without a code change. Our software refuses to run an AI feature that hasn't been registered, so nothing slips in unreviewed. Whatever the equivalent looks like on your team, you should be able to answer two questions quickly: where is AI in use, and how do we turn it off?

Governance is also how the Fear Map's conditions get met. Surveillance and Management Pressure are governance concerns, and they are answered by commitments the team can see: how usage data will and won't be used, and that no one's performance is measured by how much they lean on an assistant.

We apply security settings through managed settings for Claude with your team, refining the allow and deny lists as the office-hours questions show where they need to change.

07

How do you know whether AI adoption is working?

Look at how the team behaves and how confident it is, not at license counts. The useful measures are whether the concerns in the Fear Map are easing, whether shared practices are being used, and whether AI-assisted work is holding up in review and in production.

License counts and prompt volumes are easy to collect, and they say little. A team can be using a tool heavily and badly. Worse, measuring individuals by usage feeds the Surveillance concern directly, and undoes the trust the rollout depends on.

The measures we find useful are these:

Concern intensity. A later Fear Mapping session with the same team shows whether the top themes have eased, and whether the conditions for success have been met in the team's view.

Shared practice in use. How many skills, integrations and automations the team maintains, and whether people beyond their authors are using and improving them.

Review outcomes. Whether AI-assisted changes pass review at a similar rate to the rest, and what reviewers keep catching. A recurring catch is a skill waiting to be written.

Delivery measures you already track, such as cycle time or the time from ticket to pull request, compared with your own history instead of a benchmark.

The kind of questions in office hours: fewer about permission, more about technique and judgment.

For the business leader, the summary is simple: is the team delivering more, and is the work holding up? For the technology leader, the detail above is what backs that answer when it is questioned.

08

What should the first 30, 60 and 90 days look like?

Spend the first 30 days learning what's in the way and agreeing the rules, the next 30 training on real work, and the third 30 building the shared skills and automations that make the change permanent.

Days 1 to 30: learn and agree.

An introductory conversation about your team, its tools, how it ships today and what you want AI to change.

A first Fear Mapping session, with one per team if several are involved.

The first security settings for Claude, with allow and deny lists for data handling, tool access and review, shaped by what the Fear Map found.

One visible commitment from leadership in response to the top theme, stated to the team.

Days 31 to 60: train on real work.

Workshops built on the themes that ranked highest and on tasks from your own backlog.

Office hours running, with the questions feeding back into the written practices.

The first shared skills, usually your conventions and your review checklist, committed to your repositories.

Days 61 to 90: make it the team's standard.

Integrations and automations for the repeated jobs the team has identified, with access scoped and reviewed.

A review of the measures above against where the team started.

A decision on the next quarter: which concerns remain, which practices need tightening, and whether an opportunity has emerged that deserves its own project.

By the end of the third month the team should be using AI in the same way from one engineer to the next, under rules it helped write, with shared tools it maintains itself.

09

How does MadeLabs help teams put this into practice?

Through AI Enablement for Product & Engineering, a monthly advisory that runs every step in this guide with your team. It starts with a conversation about your team and a first Fear Mapping session.

Each month combines Fear Mapping sessions, training workshops, AI office hours, security considerations, and building and sharing skills, integrations and automations. It is for product and engineering teams, and the capability stays with your team when the work is done.

We use AI throughout our own delivery, with an engineer peer-reviewing the code it writes. What we teach is the way we already work.

If your team is using AI unevenly, cautiously or not at all, any of those is a good place to start.

AI Enablement for Product & Engineering www.madelabs.io/services/ai-enablement

Start a conversation www.madelabs.io/contact

The AI Enablement Playbook. Read the current version online at www.madelabs.io/insights/ai-enablement-guide.

MadeLabs · AI Speed. Engineering Rigor.