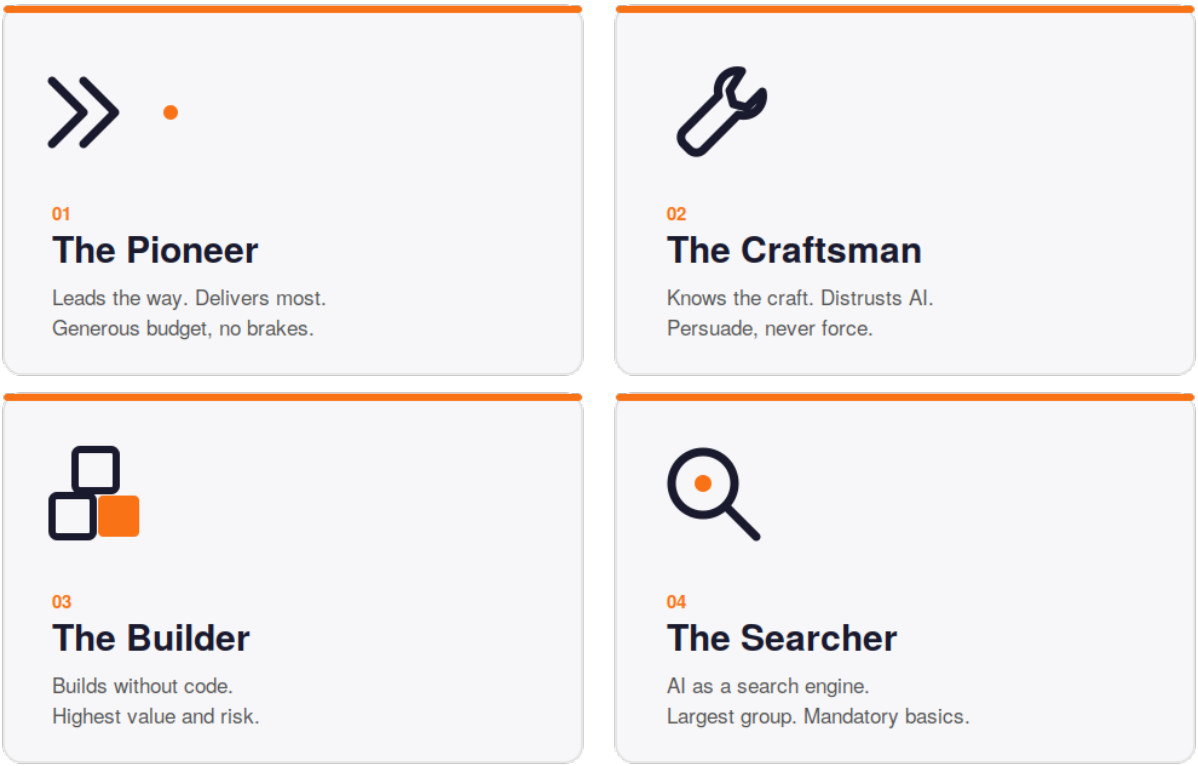
The four AI users in your organization

Why a single AI policy for everyone treats three out of four people wrong



Marc Diks · June 2026

marcdiks.nl

Executive summary

There are four types of AI users in every organization. They respond in opposite ways to the same levers. Treat everyone the same, and you treat three out of four wrong.

In practice I see four groups forming around AI. The Pioneer leads the way and delivers the most output. The Craftsman is the experienced developer who still keeps AI at arm's length. The Builder builds working tools without a coding background. And the Searcher, by far the largest group, uses AI as a better search engine.

The mistake most organizations make is treating these four as one problem with one solution. One policy, one training, one set of rules, rolled out across everyone. But a token budget that suffocates the Pioneer is exactly the focus filter that keeps the Builder sharp. A mandatory training that finally wakes up the Searcher confirms the Craftsman's feeling that AI is being forced on them. Same lever, opposite effect.

This paper describes the four types, how to treat each one to get the most out of them, and the overarching framework: three levers you deliberately turn a different way for each type.

Four people, four worlds

Walk into any organization and watch how people use AI. You won't see one behaviour, you'll see four. One developer runs Claude Code and Codex side by side and clears their backlog at record pace. A colleague right next to them only reaches for AI for a snippet or a bug, then closes the tool again. Someone in a business team builds, without any coding knowledge, a tool that saves their whole team time. And the largest group types in the occasional question as if it were Google.

These four people don't need the same thing. They need opposite approaches. Yet organizations default to rolling out one AI policy across everyone. The result: the vanguard is held back, the sceptic is driven further into the trenches, the builder creates uncontrolled risk, and the largest group stays stuck at the surface.

In this paper I walk you through the four types. For each one I describe who they are, what the value and the risk is, and above all: how to treat them to get the most out of them. Then I bring it together in one framework that explains why a single policy cannot work.

Type 1 — The Pioneer

The innovative vanguard developer

Who is this?

The Pioneer is the developer who runs out ahead. While the rest are still reading about a new tool, they've already been running it for a week. This is the one using Claude Code and Codex side by side, trying a new model on release day, and setting three approaches next to each other without hesitation to see which one wins.

What's distinctive isn't how much this person talks to AI, but how deep AI sits in the work. Not a chatbot occasionally picked up, but a fixed part of the development process. This developer works from a backlog and uses AI to clear it faster. The result is visible: more tickets, more code, in the same time.

Yes, this group burns a lot of tokens. In absolute terms it's the most expensive group in the organization. But that consumption isn't waste, it's tuition and production capacity at once. Whoever consumes the most also delivers the most here.

> **The Pioneer's token consumption is not a cost but output. The most expensive group is also the one that delivers the most. Measure it against results, not against consumption.**

What's the value?

The ROI here is the most directly measurable of all four types: tickets processed per sprint, code delivered, and token consumption as a proxy for activity. This group shifts the productivity frontier of your entire development department. And they do more than deliver code: they discover what works, so the organization doesn't have to reinvent the wheel with every new tool.

How do you treat them?

- **Give a generous ceiling, no brakes.** Set the token budget so high it's rarely hit. Every minute spent requesting more budget is a minute not spent on output. Here the budget is a safety net, not a threshold.
- **See token consumption as investment, not cost.** The bill is the highest, but so is the return. Measure it against output. Twice the consumption with three times the tickets is not a cost problem but a return.

- **Give them a mentor role.** They don't just deliver output, they pull others along. Make training colleagues an explicit part of their role. This is your most important lever to get the Craftsman moving.
- **Facilitate, then stay out of the way.** This group generally knows what it's doing. The biggest risk is getting in their way with needless approvals. Give access, budget and trust, and remove obstacles.

The goal is not a small elite, but growth: lifting as many developers as possible to this level. Today's Pioneer is the mentor for tomorrow's Pioneer.

Type 2 — The Craftsman

The experienced developer who keeps AI at arm's length

Who is this?

This is the developer who does reach for AI, but only for isolated tasks. A snippet, a piece of explanation, a bug. After that the tool closes again. Often these are the older, experienced developers. They learned their craft without AI and they're good at it. That's exactly where the friction sits: they don't trust AI yet, and see it as a threat rather than a help.

This is not ignorance and not laziness. It's distrust, fed by the sense that something is at stake.

Why this group is so valuable

Don't underestimate this type. These people still know how to really build. They know the fundamentals, recognize a bad architectural choice, and can still properly validate an AI's output. That's precisely the skill that disappears fastest in the other groups.

> **Convert the Craftsman into a Pioneer and you get the best AI developer there is: the speed of AI combined with the judgment of a seasoned developer.**

What's the risk if they stay put?

Two problems reinforce each other. The productivity gap with the Pioneers grows every month, until they're simply overtaken. And you pay twice: the organization buys AI licenses this group barely uses, while the work goes slower than it could. Standing still isn't free here.

How do you treat them?

- **Never force them.** This is the most important rule for this type. Mandatory tracks and top-down pressure backfire. Force confirms exactly the feeling that AI is a threat being imposed on them. Here you persuade, you don't force.
- **Let colleagues provide the proof.** The most convincing argument comes not from management, but from a trusted colleague who shows that it works. Peer proof beats persuasion.
- **Build trust through small, shared wins.** Pair the Craftsman with a Pioneer on real work from their own backlog. Every win they experience themselves chips away at the distrust. It's about their own experience, not promises.
- **Address the threat directly.** Because the core is fear, name it explicitly. Make clear the goal is to enrich their work and increase their impact, not to replace them.

The goal is growth into Pioneer. This type is the most important pool from which new vanguard developers come: the technical foundation is already there, only the trust is missing.

Type 3 — The Builder

The building business user

Who is this?

This is the business user who builds working tools and apps themselves, without a formal coding background. The Builder doesn't wait for the IT backlog, but solves their own problem. Whoever knows the problem best builds the solution here, with prompting, agents, and no-code and low-code side by side.

What's powerful is the combination of domain knowledge and drive. But exactly what makes this type so valuable also makes it the most risky. It's the only type where the highest value and the highest risk come together in the same person.

> **In the Builder, the highest value and the highest risk sit in the same person. That's why governance here is not a brake, but the condition for safely harvesting that value.**

What's the value?

The value comes from four directions at once: speed without waiting on the IT backlog, relieving scarce developers, innovation close to the business, and proof in the form of working prototypes that produce a far better conversation than a slide deck full of assumptions.

What's the risk?

Here you have to be sharp, because all four major risks play out at once. **Shadow IT**: tools outside the view of IT and governance. **Security and data exposure**: no awareness of where data goes or which door is left open. **Maintainability**: if the builder leaves, no one can take it over. **Scalability**: a prototype pushed uncontrolled into production. With this type, governance is not a brake but a condition for safely harvesting the value.

How do you treat them?

- **Set the frame, let them build freely within it.** IT defines the boundaries around data, tools and security. Within those frames the Builder gets all the room. You protect the organization without removing the speed that makes this type so valuable.
- **Have developers review before production.** The Builder may build a prototype themselves, but the step to production runs past a real developer who checks security, maintainability and scalability.

- **Let IT take over at scale.** When a tool catches on, hand it over. The Builder proves the problem, IT makes it robust.
- **Use budget as a focus filter.** Opposite to the Pioneer, the Builder gets a tight budget. Whoever wants more explains to their manager what they want to build. That threshold filters serious projects from hobby projects that never go live.

The distinguishing skill

What's the difference between a good and a bad Builder? Not technical knowledge. The centre of gravity sits up front: being able to define a sharp problem, then writing good prompts, then knowing when to bring in IT, and only then judging the output. Good building starts with clear thinking, not with reading code. The scarce skill here is business insight, and that's exactly what this type already has by nature.

Type 4 — The Searcher

ChatGPT as a search engine

Who is this?

This is by far the largest group, the majority. For the Searcher, AI is mainly a better search engine: a question in, an answer out. Plus some light text work, like summarizing or drafting an email. No follow-up questions, no context, no second iteration.

The key thing to understand: they don't stay at the surface out of fear, but out of unfamiliarity. They simply don't know what more is possible, have no incentive to learn further, and their work doesn't seem to demand it at first glance. It's not resistance, it's a blind spot.

> **Craftsman and Searcher look alike, but they're blocked for opposite reasons. For the Craftsman it's emotional: distrust. For the Searcher it's practical: unfamiliarity. That's why the same intervention backfires with one and falls flat with the other.**

What's the value and the risk?

The value per person is the lowest here. But because this is by far the largest group, it holds collectively the greatest untapped potential of the whole organization. The risk is subtle: these people think they're already an AI user. You can't create hunger in someone who thinks they've already eaten.

How do you treat them?

Here the approach turns a hundred and eighty degrees compared to the Craftsman. Where you must never force the experienced developer, this group should actually get a mandatory baseline. Without that push, the combination of unfamiliarity and lack of incentive never breaks loose.

- **Start with mandatory basic training for everyone.** Because the block is unfamiliarity, you give everyone the same starting point. Mandatory works here, because you're filling an empty space, not pushing against a wall.
- **Then let colleagues provide the inspiration.** After the basics, motivation comes from low-threshold examples. “Look what I did with this in ten minutes” does more than any module.
- **Hand out ready-made prompts and use cases per role.** Not “go discover for yourself”, but “here are five things that deliver value for your work right away”.

- **Build AI into the tools they already use.** If the function sits in their existing environment, use becomes self-evident instead of an extra action to remember.

For this group there's no leap, but a staircase: first the basics safely in place, then a few powerful use cases per role, then from incidental to habit. A modest step per person is, at these numbers, the largest collective leap the organization can make.

The framework

Why a single AI policy for everyone doesn't work

Most organizations treat AI adoption as one problem with one solution. One policy, one training, one set of rules, rolled out across everyone. That's exactly why it goes wrong. The four types respond in opposite ways to the same levers. What moves the Pioneer forward holds the Builder back. What gets the Searcher moving drives the Craftsman further into the trenches.

> **Treat everyone the same, and you treat three out of four wrong.**

The three levers you turn differently per type

The framework revolves around three variables. Not settings you fix once, but levers you deliberately turn a different way for each type. The orange cells below show where you're strict or steering: they sit scattered, not in one column.

	Pioneer	Craftsman	Builder	Searcher
Budget	Generous ceiling	Follows naturally	**Tight + accountability**	Not relevant
Pressure	No pressure, facilitate	Never force, persuade	**Guardrails fixed, building free**	**Mandatory basics are fine**
Governance	Minimal	Minimal	**Maximal**	Basic safety

The three levers per user type. Orange = strict or steering, light = room.

The sharpest contrast sits between Pioneer and Builder on budget. For one, a generous budget is an investment in output; for the other, a tight budget is an investment in focus. Same lever, opposite setting, both for the right reason. On pressure, Craftsman and Searcher flip: for the Searcher, mandatory works, because there's no wall to push against, just an empty space to fill.

The growth movement

The four types aren't boxes people are stuck in. They're positions on a movement with a direction. The Searcher moves up a staircase, and the motivated among them can grow towards Builder. The Craftsman is the most important pool for new Pioneers: the technical foundation is already there, only the trust is missing. The Pioneer is not an end station but a growth group you want to make as large as possible.

The goal is not to make everyone a Pioneer. The goal is to let everyone move up one position. For the largest group that's collectively the biggest leap; for the Craftsman it's qualitatively the most valuable one.

The Pioneer as the pivot

One element binds the whole framework together: the Pioneer is not only the most productive group, but also the engine behind the movement of the others. For the Craftsman the Pioneer is the living proof and the hands-on mentor who removes the distrust. For the Builder the Pioneer is the developer who reviews and scales prototypes. For the Searcher the Pioneer is the colleague who shares low-threshold wins after the basic training.

That's why the Pioneer's mentor role is not a bonus but a core function. Whoever deploys the Pioneers only as individual production machines leaves their biggest lever unused: their ability to get the other three types moving.

Conclusion and recommendations

AI adoption is not a matter of flipping one switch for the whole organization. It's a matter of turning the right lever the right way for each group. Concretely that means:

1. **Stop with one AI policy.** Design per type, because the levers run opposite.
2. **Differentiate budget deliberately.** Generous for Pioneers, tight with accountability for Builders. Not as a cost cut, but as steering.
3. **Apply pressure selectively.** Mandate where there's no resistance (Searcher), persuade where there is (Craftsman).
4. **Put governance where the risk is.** Heavy on the Builder, light on the rest.
5. **Make the mentor role explicit.** The Pioneer is your most important adoption instrument, not just your fastest developer.
6. **Steer on movement, not on destination.** Moving up one position per type is the win, not making everyone a Pioneer.

About the author

Marc Diks writes about AI strategy, governance and digital transformation for organizations. He builds production applications with AI tools himself and shares his hands-on experience on his blog, LinkedIn and Substack.

Website: marcdiks.nl | LinkedIn: linkedin.com/in/marcdiks | X: x.com/marcdiks