My vision: an approach to the AI transformation of corporate IT

Shock forecast: corporate IT headcount down 50% by 2028 and 70% by 2030, at least in traditional universal banks. Here is how I would run the AI transformation, step by step.

In short. I seriously believe that by the end of 2028 corporate IT headcount, at least in traditional universal banks, can be cut by 50%, and by the end of 2030 by 70% of the end-2026 level. Handing out subscriptions and AI tools won't get you there. You need a shareholder mandate for the cuts, a deep IT audit run by agents, pilots that start with non-critical areas, and savings planned together with the CFO. Here's how I see it.

My shock forecast: by the end of 2028 corporate IT headcount, at least in traditional universal banks, can be cut by 50%, and by the end of 2030 by 70% of the end-2026 level. Screenshot this. These aren't empty words, this is the reality coming at us. For better or worse. The alternative scenario: companies suddenly need many times more IT infrastructure and people for current business projects just to survive in the new world, so IT budgets stay and productivity and workload grow severalfold. Honestly? I don't buy it.

Why exactly 50 and 70%? None of the big consultancies has a straight "minus half of IT staff" forecast. That's my own estimate. But the heavyweights confirm the direction. At Google the share of new code written by AI went from 25% in 2024 to 75% in April 2026, and engineers review part of it. Gartner expects most code to be AI-generated and 80% of engineers to need reskilling by 2027. In 2025 Anthropic's CEO Dario Amodei warned that within one to five years AI could wipe out up to half of entry-level office jobs. And 40% of employers in the World Economic Forum survey plan to cut staff by 2030 wherever AI automates the work. The way I see it, the market has moved from denial ("we don't need to change") to anger that the comfortable old ways are going away. You know what comes next: bargaining. Better to skip straight to acceptance and act now, while there's still time.

My math is simple. **You can cut headcount by 50% with the same workload if every person who stays does twice as much with agents, and by 70% if they do 3.3 times as much.** With agents already writing three quarters of new code, I think a twofold gain across all of IT by 2028 is doable. For threefold by 2030, agents have to get just as deep into Run: administration, support, operations and incidents.

If you can't stop it, lead it. That's why I'd build the AI transformation programme now, for this horizon. Otherwise you'll spend years polishing tools, let competitors run ahead and keep paying the old price for IT. In my experience that price hits 20-30% of all costs at a modern company with a digital core.

Why am I so sure? I know how to cut IT costs without AI: in one year I took a third off a bank's IT spend without stopping a single product. And I've already built corporate software myself with AI tools. The quality blew past my expectations, and since 2024 the models have literally been getting better every month. Truth is, LLMs are perfect translators between languages: give them detailed functional and architecture requirements and they'll calmly translate human requirements into code. That's how it works.

Let me draw the boundaries up front. This is not a complete guide to AI transformation but my take on the key steps: mandate, audit, money, pilots, control and people. Like any transformation, it's a bespoke suit, cut for a specific company, its systems and its people. This article gives the general approach. How to reconfigure the IT org structure and management after AI, and which organisational approach and governance to use for the rollout in each domain, are big topics of their own. I name them here but don't unpack them. Stay active on LinkedIn and I'll keep writing about the stuff that hurts.

Scale: all of Change and all of Run

AI doesn't belong in development alone. It belongs in requirements, architecture, development, testing and DevOps, that is, all of Change. And it belongs in Run: user support, monitoring and incidents, operations, vendor management and even administration and SRE. Plus controls and checks inside every process. **The scale is huge, and you have to see the whole picture from day one.** Scared to let an agent into even your test infrastructure, or to run your corporate cloud through an API? **In two years a competitor will overtake you.** So information security questions have to be solved now, not after the first pilots.

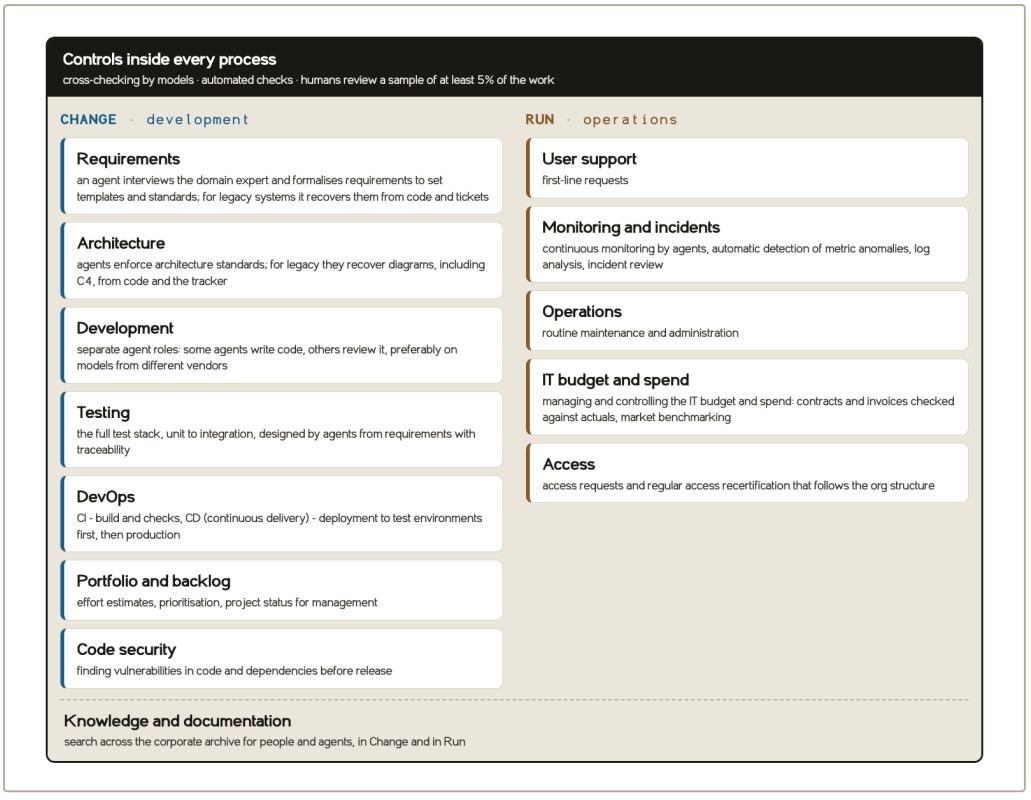


Figure 1. Where to apply AI in corporate IT and what to change now

In parallel I'd build search across the company's knowledge archive, the way we did with FlyArchive: agents need context at every step, and that context lives in documents, tickets and email.

Prerequisites

First, a hard mandate from the CEO and shareholders to change the cost and the status quo of corporate IT. You need the authority to rework processes, team structure and contracts and to decide who goes. Promise up front to keep every department, its budget and the legacy teams with their unique expertise, and the result will hit the ceiling of those promises.

There will be resistance. Managers will explain why their teams are untouchable and why AI doesn't fit their system, and they'll send you memes about AI screwing up. Stay calm. **A good relationship with a middle manager is no reason to keep their budget in a corporation.**

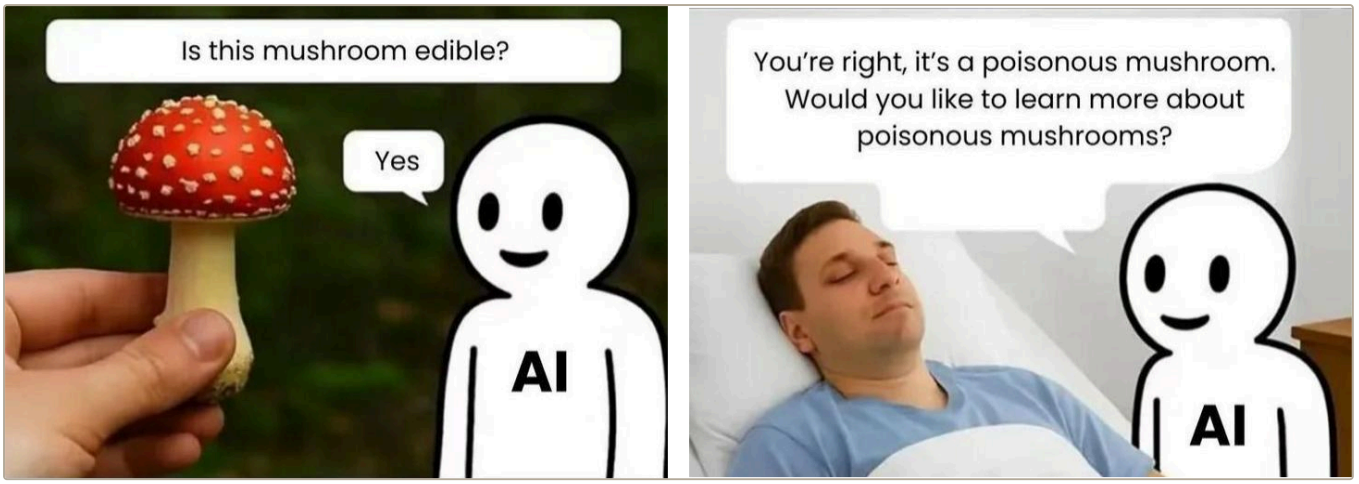


Figure 2. The mushroom meme: you'll get it among the very first

Second, data. How you handle it is agreed with IT infrastructure and information security: either some data may go to external models, or the company runs open models from the DeepSeek, Kimi or GLM families in-house. They're already good enough for 95% of tasks. Size your own AI cluster to the load: a process with 7-10 agents running in parallel needs the capacity for them. I wouldn't buy a cluster before running the numbers; I'd rent one in a cloud data centre to start.

Third, numbers before you start. Agree the IT cost baseline, the quality and availability limits and the goal with the CEO, the shareholders and the CFO right away: lower costs, faster change, business growth served by the same team.

Stage 1. An audit run by AI agents

Start by reconciling the whole fabric of IT: code, processes, budgets, contractor deals, what the company orders, what it actually gets and what it pays. Agents do the audit. They need read access to:

- code and test environments;
- trackers and knowledge bases;
- architecture and integrations;

- monitoring and incident history;
- board decisions;
- budgets and contracts;
- process regulations.

Every conclusion from one model gets checked against the primary sources by a model from another vendor, and every conclusion comes with links to the code, the ticket and the document. An agent's verdict is a reason to check. A people decision needs confirmed facts.

In practice the "fat" usually sits in three places:

- bloated teams;
- support and development contracts;
- work done out of habit.

For most projects a team of 3-5 people working with agents is plenty today. A strong manager can defend a budget like a pro, but don't confuse that with an effective department. Not everyone goes: I'd trim the top seniors with AI skills and the middle management layer but keep a lot of them, because the first hold the hard skills and nobody else will sort out the friction between departments. **Here's the surprise: as models get better, the talkers and communicators inside companies will be worth more than the quiet coders.** Again, not all of them, and people will still need technical chops to control the machines. You just need far fewer of those people.

The output of this stage is a map of IT domains with costs, owners and problems, plus a short list of proposed changes: which process we change, what effect we expect, how we check it, who owns it and when we pull the plug.

Stage 2. Money and TTM from day one

Start measuring the effect from day one. If you're honest with shareholders, the effect is mostly money and TTM (time to market). **This is where a toy with nice tools ends and a real business change begins.**

Outside research gives you a reference point, not an answer. In experiments with 4,867 developers at Microsoft, Accenture and a Fortune 100 company, an AI assistant delivered about 26% more completed tasks on average (Microsoft Research), but that was way back in 2025. **So measure in-house.**

People used to buy external benchmarks for this. I used decent tools like BlueOptima, which measure developer productivity from source code changes and compare companies. Now, in my view, you can do it yourself: agents go through the repository and the tracker, count delivery by team and figure out on their own how to compare it fairly. You don't need an external benchmark for that anymore.

So you don't drown in metrics, I'd give the whole programme two North Stars:

1. **Total cost of IT**: C = payroll + outstaffing + contractors + licences + infrastructure + vendor system support + AI spend. It has to go down in gross terms every quarter. Savings on people eaten by subscriptions and the cluster don't count.
2. **Delivery time (TTM)**: from an agreed task to a working result, including waiting for sign-offs. It must not grow, otherwise the cuts just pushed the work into a queue.

Quality isn't a star, it's a guardrail: post-release defects and rollbacks don't grow. For development, time and quality are easy to track with DORA metrics. The DORA report for 2025 links AI use to higher development throughput and lower delivery stability, so checks get tighter as the volume of changes grows. Tokens and sessions show cost, not value: nobody cares how many tokens a task burned, the business cares about the result.

For a shareholder, two formulas fall out of the first metric:

1. **Effect for the company** = d × ΔC, where d is IT's share of company costs and ΔC is the drop in IT cost. With d = 25% and IT costs cut by a third, the whole company's costs fall by about 8%.
2. **Payback in months**: T = R / (monthly savings - monthly AI spend), where R is the one-off cost of the switch: training, severance, running the old and new process side by side.

The effect needs an owner who works hand in hand with finance. Freed-up hours mean a decision: cut contractors, stop hiring, reduce headcount or hand the capacity to the business. **The same hours can't be booked both as savings and as extra development.** A pretty summary percentage won't cut it. The key is spotting the moment free time appears.

Stage 3. Pilots ordered by the cost of a mistake

Based on stage one, go for the low-hanging fruit. Peripheral systems and repetitive support operations go first: mistakes there are cheap and results show up fast. Then the more important systems. Mission-critical systems, the ones whose failure stops core operations, come in as results pile up. A system's criticality and an agent's rights are two different things: an agent can analyse a critical system's code long before it gets permission to touch anything in it. And as I said, the audit and analysis can and should be done with agents right now.

The real fight will start mid-pilot. This is the bureaucracy that can eat all your savings. If an agent sped up the coding but a change still waits weeks for sign-offs, the economics haven't moved. You'll have to change queues, team boundaries and who decides what. This is where the comfy experiment with tools ends. You'll have to change processes, because the bottleneck may not be in analysis, development or testing at all, but in signing papers.

Stage 4. Control

My model is an automated car plant: robots do the work, people watch quality and stop the line when they see a defect. A person who mechanically clicks "approve" isn't control.

The model is a tool. Responsibility for the result stays with people. So the people who stay in IT become operators, like on that plant floor. They own the models they work with, know where those models slip, and can double-check the output with another model, an automated check and their own expertise.

Control stands on three legs:

1. **Cross-checking between models.** My favourite. A model from another vendor first forms its own view, then works through the differences with the executor. I've checked Claude's output with Codex and the other way round many times. Works like a charm.
2. **Automated checks:** tests, static analysis, reconciliations, permissions. Whether an accounting entry balances is checked by calculation, not by a model's story. Tools like SonarQube aren't going anywhere, models or not.
3. **Selective human review:** new kinds of tasks, disagreements between models, a random slice of routine work. High-stakes actions always get human sign-off. **And a human should spot-check at least 5% of a model's work.**

An agent's authority grows in steps: read-only, changes in a sandbox, changes for approval, limited autonomy. **People have to learn to control the machines.** A big problem: agents inside a corporation know nothing about the org chart, rules and restrictions until someone spells them out. An agent doesn't care whether a junior or a senior analyst is allowed to see a data mart. So while you build control, you also build layered protection against agents that go off the leash. But that's a story for another day.

Stage 5. People and hard decisions

The 2028 horizon means you can't wait for everyone to pick up AI voluntarily. **Most likely you'll have to make hard calls: let go of the people who didn't adapt while keeping the department's workload.** Then the ones who stay have to work with AI tools, or they won't keep up.

Help them, though. Training starts with the people already comfortable with agents and ends with real work done on real tasks: a resolved ticket, an accepted change, a verified report. Carve out time for it, because on top of a full workload it stays a plan forever.

After that, teams get targets on cost, time and quality, and hitting them changes the budget and the headcount. A manager who got the tools and the time owns the result. As long as a manager's status depends on headcount, they have every reason to keep staff, so the success criteria for managers change too.

Not every objection is sabotage: reproduce the specific risk and check it. But "it's impossible here" with no example doesn't stop the programme. **Those who can't or won't work the new way after the transition period have to go, managers included.** Someone with hidden knowledge of a system leaves only after that knowledge is in the documentation and the checks. Layoffs cost money and time, so build that into year one.

After the cuts, the IT org structure and management will have to be reconfigured: different roles, different reporting lines, model operators instead of doers. That's a big topic and needs its own deep dive.

Conclusion

Step up to the bar in your company now. Don't wait. Otherwise it'll be the mid-2010s all over again, when a bunch of well-known top managers flew to Silicon Valley, discovered autonomous agile teams there, and the whole thing swept across the CIS. As I see it, very soon we'll get the first corporate examples like that, they'll shake shareholders and business to the core, and everyone will want the transformation. So the time to start is now.

A few topics didn't make it into this article, and each deserves its own piece:

1. Protecting yourself from agents that go off the leash: permissions, the org chart and layered control.
2. Your own AI cluster or the cloud: how to size the load and run the numbers.
3. Data and the regulator: what can go to external models and who answers for an agent's mistake.
4. How to run layoffs: labour law, knowledge transfer, talking to the team.
5. How to measure development productivity with agents on the repository instead of external benchmarks.
6. The new IT org structure and management after AI: roles, layers, reporting lines.
7. The organisational approach to the rollout in each domain: what governance you need and who decides what.
8. How to motivate and inspire the people who stay.

You may disagree with me. Tell me in the comments on LinkedIn whether you agree and which of these topics I should take on next.

For those who made it to the end, a gift: an archive with ai-sdlc-kit 0.1.3, a set of skills for AI agents that reverse-engineers any of your legacy code into requirements and architecture. One agent reads an existing system's code and writes its description: what it's made of, what processes it runs, what it must do and how to check it. A second agent checks that description against the code and sends back whatever doesn't match. On my own project I described 25 processes this way, and along the way about 480 remarks on the code itself piled up, including real security bugs.



Andrey Sabynin

Founder of FlyOptima, an AI-native developer of solutions for banks and fintech.