REDSTONE · FOUNDER STORIES

“That's an eighty times cost difference, on a process that runs thousands of times a day.”

How one drawing, shared on a call, convinced us the era of the autonomous agent was already over.

Alexander Oelling, Matthias Kainer, Jesper Bylund and Kamil Klüber
Michael Brehm
INXM · Berlin

Error compounds

Alexander joined the call and shared his screen before I had asked a single question. Not a slide with the company name on it. A drawing: a small robot at the top of a family tree, branching down into six smaller robots, "*error compounds*" underlined twice in red, next to a steering wheel wired into a grid of a thousand tiny mechanical parts, each one checked off or flagged. Jesper had posted a version of it on LinkedIn a few weeks earlier. Alexander pulled it straight back up.

"*That's the whole pitch, if you want it in one drawing,*" he said. "*Everyone else is building the robot family tree. We built the grid.*"

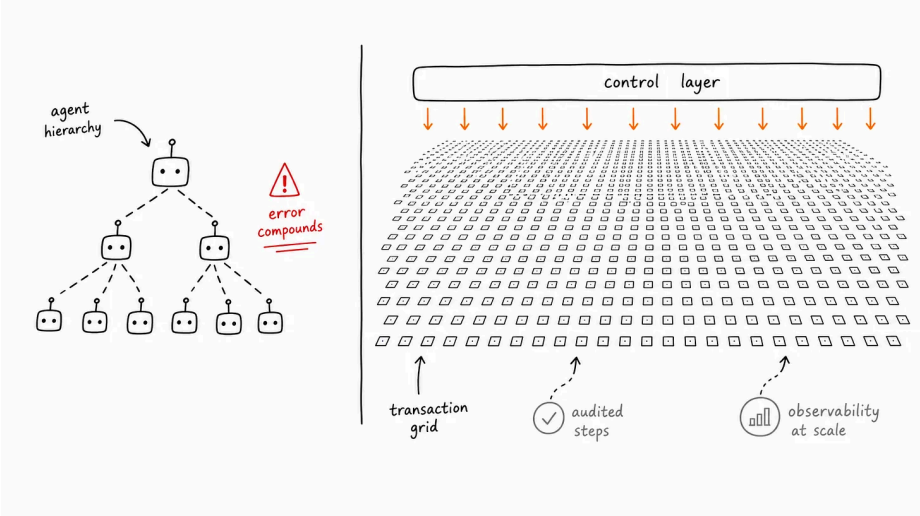
Alexander is INXM's co-founder and CEO. He spent years before this as Chief Digital Officer at Isar Aerospace and Volocopter, leading digital transformation in regulated, safety-critical environments, after founding and scaling Sensorberg into one of Europe's largest proximity infrastructure providers. I had come to hear how a serial founder, a systems architect, a product designer and a manufacturing veteran had ended up building the same company, and why they think the entire industry is arguing about the wrong bottleneck.

Four founders, one failure

None of the four arrived at this from a typical software background, which turns out to matter more than it sounds. Matthias Kainer, the CTO, has spent more than twenty-five years in software development and systems architecture, including stretches at Microsoft, AutoScout24, ThoughtWorks, Volocopter and Isar Aerospace. Jesper Bylund, the CPO, led design at Volocopter and product design at n8n before that. Kamil Klüber, the CSO, spent nearly seven years at Siemens driving digital transformation across aerospace, defence and new space, and

years before that at HELLA, working the entire product development value chain in automotive. Between the four of them: rockets, air taxis, product design, factory floors. Four different industries, and, as it turned out, the same failure sitting underneath all of them.

"*We compared notes across our backgrounds,*" Alexander said, "*and we kept landing on the same pattern. Not a lack of intelligence or data. The intelligence was there. What was missing was execution, reliable, auditable, repeatable execution at the process level. That isn't sector-specific, but structural. In that moment we stopped saying this is interesting and started saying this is the problem.*"



It is a specific kind of tolerance, and Alexander argues most software teams never develop it, because they have never operated anywhere failure has physical consequences. Rockets require determinism, you cannot patch a launch failure in production. Air taxis require certification, you cannot ship a workaround. Factory floors require reliability at scale, where a system that works ninety seven times in a hundred is a system that fails on the other three, and that failure has a cost attached to it.

> *A system that sometimes works and sometimes doesn't isn't a system. It's a liability. That's exactly the lens we apply to enterprise AI. An AI that occasionally hallucinates its way through a process doesn't reduce operational risk. Instead, it introduces a new kind of risk, one that's even harder to audit than the problem it was meant to solve.*
>
> ALEXANDER OELLING

The notary

We went back to the beginning, or rather to the specific afternoon it stopped being an idea. "*It became real at the notary,*" Alexander said. "*Until then, it's all ideas. You meet a future customer, an investor, you talk a bit here and there, and it stays a pleasant, hypothetical conversation, because you can't raise money before there's a company to raise it into. Then the notary turns to you and says, sign here, you're founding the company now. That's the moment it stops being virtual.*"

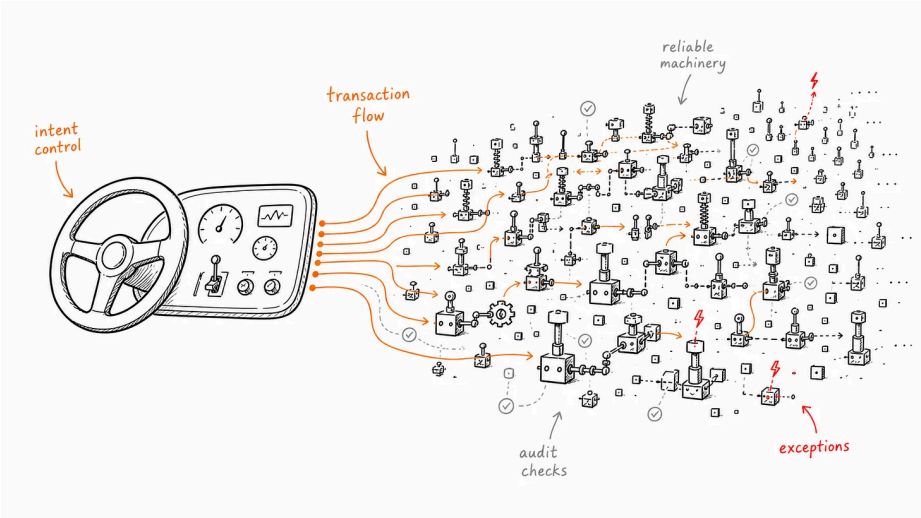
He is careful not to dress this up as a leap of faith. "You don't know yet whether it will fly. You have a feel for the market, real experience in it, enough conversations behind you, and a conviction. At some point you simply find it, when the team is together and you understand the market, then you can see where it's heading. So you say, now we build it." The conviction, he told me, came from a gap he had been staring at for years without a name for it. Nobody was actually serving the enterprise customer, and given the sheer number of industries and niches involved, no single company ever would on its own.

The bet

This is where the actual bet gets interesting: people initially thought the team was being too conservative, too technical, or simply late to the party. While most of the market raced toward autonomous agents, INXM bet on something narrower, and in their telling, more useful. AI that plans, and an orchestration layer that executes.

"*Not because we didn't want AI to do the work. We absolutely did,*" Alexander said. "*But autonomous and reliable weren't going to arrive at the same time, and in enterprise you can't ship the gap.*"

The product is called INXM Orchestrator, and the underlying approach is what the team calls Compiled AI, a term with an origin story Alexander is oddly insistent on getting right. "*The honest answer is we didn't coin it alone. It comes from a research paper, on deterministic code generation for LLM-based workflow automation. When Matthias read it, his reaction was simple, these are the words I've been looking for. We'd already built the thing. We just didn't have the word for it.*" The mechanism, once you see it, is straightforward. An LLM is used in a compilation phase to generate executable, auditable code for a given process. That code gets validated. Only then does it run, deterministically, with zero tokens spent at runtime.



A workflow tool executes the steps someone defined last year, built by professional process engineers in complex tools. Orchestrator executes the steps that are right today, in the customer's own systems and language, without needing an engineer to rebuild the whole process every time something changes. "*An operator on a factory floor doesn't want to interact with AI,*" Alexander said. "They want their task done. And the plant manager doesn't want to trust a black box. They want to see what ran, when, why, and what they can change."

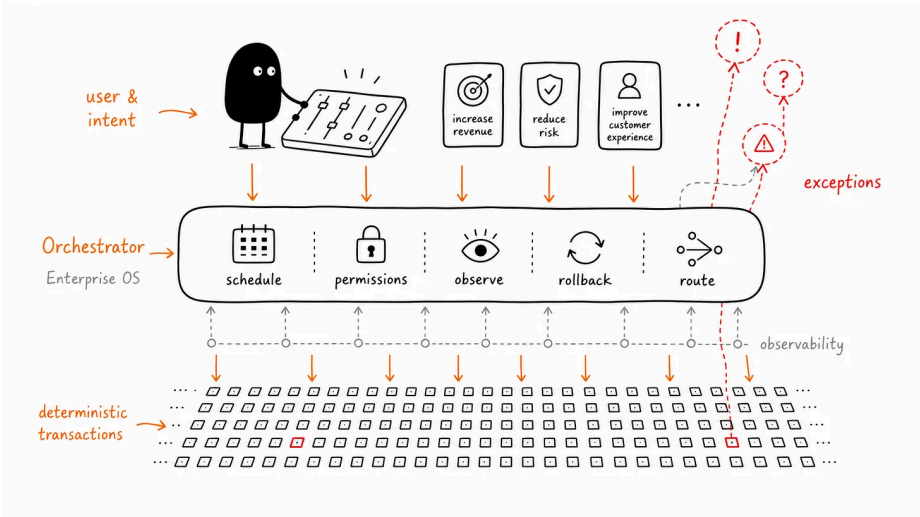
The number

The figure Alexander keeps returning to is the cost curve. "*A conventional LLM approach runs roughly a hundred thousand or more tokens per SAP transaction. Our compiled execution runs the same transaction in one thousand to twelve hundred and fifty tokens, because the intelligence was spent in the compilation phase, not the execution phase. That's an eighty times cost difference, on a process that runs thousands of times a day.*"

He is careful to frame this as an architecture decision rather than a bet against the models improving. "*Even if execution reliability eventually becomes a native property of the models themselves, the cost structure is already decided. It's not a bet that today's models stay flawed. It's a bet that the economics compound regardless.*"

I asked the obvious sceptic's question. A CIO has just spent ten million euros on SAP, why does he now need another layer on top of it? Alexander didn't flinch. "*Because the ten million didn't buy you a process layer. It only bought you a system of record. Your SAP knows what happened. Your orchestration layer decides what happens next, coordinates it across every system involved, and executes it without a developer every time a process changes. Two*

different problems. We don't compete with SAP. Instead, we make the SAP investment actually executable in an AI world."



What it took

The part of the conversation that stayed with me longest had nothing to do with tokens or orchestration layers. I asked what he draws on when things get hard, and he didn't reach for a business anecdote.

> *I've been doing this a long time, over twenty years as a serial founder, and more resilient with every one of them. The sharpest turning point was getting seriously ill. I had cancer, and it forced me to look at everything from a completely different angle. You learn a certain humility toward the universe, and at the same time a hard realism. That's what it takes to make it as a founder. It's a constant struggle with yourself, and I won't pretend I've fully mastered it.*
>
> ALEXANDER OELLING

He credits staying restless, building teams that do not need him standing over them, and, deliberately, keeping the rest of his life stable. "*None of it works without a stable environment and the backing of your family. You can't afford too many construction sites in your life at once. With a small child at home, you don't need a second source of chaos on top of the startup.*"

That same instinct for where not to spend energy shows up in how he talks about the hardest problem the company has actually had to solve, which, tellingly, was not the natural-language pitch that sits at the centre of INXM's own product. "*Honestly, that isn't the hard part. The*

hard part is understanding the customer's use cases and putting the customer first. For that you need process and industry experts, not the software developers the same project used to require." The genuinely difficult part, in his account, is product-market fit in the plainest sense, getting honest about whether technology that impresses everyone in a demo can be pressed into a form that creates measurable value for a specific buyer.

Signing off

Before we wrapped up, I asked what has to be true in three years for any of this to have worked. Alexander didn't reach for a growth number. "*Customers, customers, customers. Projects that are simple to connect. AI that's not only safe, practical and transparent, but genuinely easy to adopt.*" Zoom out to the ten year version and the ambition gets larger without getting vaguer, a company that behaves the way AWS does inside a business today, not a tool anyone evaluates on a shortlist, but the layer the business quietly runs on. "*Companies don't think about AWS as a vendor but as the layer their business runs on. That's the position we're building toward.*"

It is a question we spend more time on inside Redstone than founders probably assume. We argue about it constantly across the team, how much of the stack above the model gets absorbed by the model itself, and how much stays a separate, defensible layer that someone has to build on purpose. Early on that felt like a question with a comfortable answer arriving eventually. The longer we sit with it, the harder it gets to say with any confidence which side of that line INXM's own bet lands on, and I have come to think that is less a gap in our own thinking than the honest state of the argument right now.

When we hung up, the drawing was still open on my screen, the robot family tree still had error compounds underlined in red. I found myself looking at the grid next to it instead, small, checked, auditable transactions, unglamorous by design. INXM's argument, in the end, is a quiet one. The industry has spent two years asking its models to be smarter, when the thing actually breaking in production was never intelligence at all.

Redstone invested in INXM because the company solves the execution problem of enterprise AI, a structural bottleneck that persists no matter how much model intelligence improves. INXM arrives at the precise moment enterprises are moving from AI pilots to production deployments, where reliability, auditability and cost per transaction decide who actually keeps the budget.

Redstone is one of the most active European early-stage VCs and holds a top-decile track record across the sector funds.