The Rise of the Finance Engineer

What it is, why it's happening now, and how to become one.

By Stephen Hedlund, Head of Finance, Rillet

Rillet

Finance Professionals Are Becoming Builders

A new kind of finance professional is emerging. They started as accountants, controllers, and FP&A analysts. They didn't come from engineering. Most of them had never touched a terminal before. And they're building tools their software vendors haven't shipped yet.

We're calling them Finance Engineers.

Historically, scaling a company meant scaling finance headcount alongside operational complexity.

Finance Engineering changes that equation.

One Finance Engineer can now automate workflows that previously required analysts, consultants, ERP admins, spreadsheet operators, or engineering support to maintain manually. The result is not that finance teams disappear. The result is that finance teams become dramatically more leveraged.

The finance organizations that adopt these capabilities early will close faster, operate leaner, and spend more time designing systems instead of manually operating them. The ones that don't will increasingly find themselves scaling headcount to compete with workflows software is already capable of handling.

Why Finance Teams Couldn't Build (Until Now)

For years, finance teams knew exactly what they needed. They just couldn't build it.

Finance professionals spend enormous amounts of time inside broken workflows they didn't choose and can't change. Historically, fixing those workflows required engineering resources, ERP vendors, consultants, or years-long implementations.

AI-native tooling changes that equation. Now the people who actually understand the operational pain can build the solution directly.

If you've worked in finance for more than three years, you know the drill:. You hit a manual workflow, you know exactly how to fix it, and you file a ticket to engineering. Then you wait.

Backlog. Sprint planning. Someone else's priorities. Six weeks later, either nothing happened or you got something close enough to live with.

That gap defined the constraints of finance operations for a decade. Your stack was whatever your vendors decided to build, and your workflows were whatever you could manage in Excel.

AI Changed Who Can Build Accounting Software

The hard part was never just writing code. It was understanding intercompany eliminations, deferred revenue, auditability, approval hierarchies, close workflows, and the operational reality of how finance teams actually work. AI-native tooling dramatically lowered the engineering barrier. The accounting expertise was already there.

What Is a Finance Engineer?

Finance professionals who build internal software using AI.

A Finance Engineer is a finance professional who builds the systems, agents, and automations that make every part of the finance function faster. They started as accountants, controllers, or FP&A analysts, and taught themselves to use AI tools to architect workflows that used to require engineering.

Most of these Finance Engineers are not writing traditional production code. They're building through AI-native tooling, and conversationally iterating on workflows, integrations, automations, and internal software using coding agents.

Call it vibe coding if you want, but the appeal of Finance Engineering is all the same: instead of spending your career working around broken workflows, you get to retool them.

> Finance leaders using AI work within existing systems → Finance Engineers build new ones.

- When the ERP doesn't talk to the billing system the way it should, Finance Engineers build the connection.
- When a reconciliation could be automated but the vendor hasn't shipped it, they ship it themselves.
- When there's a manual close task that runs the same way every month, they look at it and think: this should run itself.

That's a different mental model than FinOps or systems accounting. Those roles keep the infrastructure running: clean data, working integrations, and reliable processes. Finance Engineers build on top of that foundation to create entirely new capabilities.

Finance Engineers are measured on the same KPIs as any other finance role—close speed, forecast accuracy, audit readiness—but their output is systems, not spreadsheets.

The role is emerging now the way Marketing Engineer, GTM Engineer, and Data Engineer did in previous eras. It doesn't appear on most org charts yet, but it's already doing real work at real companies.

What Is a Finance Engineer?, Continued

Before	After
File a ticket with engineering	Have it built by the weekend
8-week wait for a workflow fix	Shipped by next close
Reconciliation still manual	Automated, runs every month
Finance dependent on engineering's roadmap	Finance designing their own roadmap
Best operators doing repetitive work	Best people designing the systems that eliminate it
Spreadsheet-heavy workflows	Internal software built around the business
ERP limitations define operators	Operators define the tooling

How Finance Engineers Actually Build

The stack is simple: AI coding agents, APIs, spreadsheets, and finance systems.

Most Finance Engineers are building with tools like Claude Code, Claude Cowork, Cursor, Lovable, Replit, and MCP servers. They connect those tools to systems like Rillet, Stripe, Ramp, Salesforce, Brex, or their data warehouse through APIs, browser automation, CSV exports, or direct integrations.

The workflow usually starts with a very specific operational pain point:

- Reconciliation that takes 4 hours every month.
- Forecast that depends on manually pulling reports from five systems.
- Close checklist buried across spreadsheets and Slack threads.

Then, they open an AI coding tool and start iterating conversationally: “Build me a workflow that pulls open invoices from Rillet, matches them against Stripe payouts, flags discrepancies, and posts exceptions into Slack.”

Most of the actual development happens through back-and-forth prompting, not traditional software engineering. The Finance Engineer describes the workflow, tests the output, notices edge cases, refines the logic, and gradually turns an operational process into software.

MCP servers and browser-use tools dramatically expand what these agents can access, letting finance teams interact directly with ERPs, BI tools, internal systems, and web applications without needing formal engineering infrastructure.

Real Examples of Finance Engineers at Work

None of these people were software engineers. All of them shipped real internal tools.

The builds vary, but the profiles across the board are consistent: accountants who decided the bottlenecks they faced were a system problem, not a workload problem.

Here's how they solved it.

Brock Beyer, Controller at Jump: Studied accounting, earned his CPA, had never written a line of production code. He built a hoteling desk software for his company entirely through vibe coding tools. It went live in production. **It saved Jump $20,000 a year.**

Alex Altman, Head of Finance at Coram AI: Picked up Claude Code roughly 90 days before joining one of our sessions. By the time he showed up, he'd connected his ERP to Claude via MCP, **built a live 13-week cash forecast** pulling from AR and AP in real time, and gotten QuickBooks connected by having Claude Cowork operate his browser and fill out the developer connection form.

Samuel DeZube, Head of Strategic Finance at Workshop: Built a full CPQ tool — connected to PandaDoc, with automatic proposal generation, multi-option deal structuring, and Slack-based approval routing for out-of-policy discounts — using only Claude Cowork. No terminal. All in about **10 back-and-forth prompts.**

Colin Schwendt, Controller at CaptivateIQ: Built a **full event budgeting app in Lovable**, connected it to the Ramp API to pull home cities and auto-calculate travel allowances, integrated GSA rates for hotel and per diem, and wired the output directly to Ramp's virtual card system so pre-authorized budgets load onto employee cards before the event. It's live, deployed to the company, secured via magic-link auth. He built it while doing his actual job.

The accounting knowledge is what makes these builds hold up. All of these people know what a journal entry needs to look like. They know what controls need to exist. They know when an automation that looks clean is producing something that would never survive an audit.

The Finance Engineer's Toolkit

What's actually in the stack.

Here's an inside peak into what Finance Engineers are using:

1. **Claude Code:** An agentic coding environment where you describe what you want to build in plain English and Claude writes, runs, and debugs the code. Best for building applications, automations, and data pipelines.
2. **Claude Cowork:** Claude operates your browser on your behalf. Useful for tasks that require navigating existing software interfaces, filling out forms, or connecting systems that don't have clean APIs. No terminal required.
3. **MCP Servers:** Model Context Protocol. Connects Claude directly to your live data — your ERP, your CRM, your billing system. Gives AI tools the context they need to provide better outputs.
4. **Lovable:** A visual app builder that uses AI to generate full web applications from descriptions. Good for tools that need a UI and need to be shared across a team.
5. **Rillet:** The ERP underneath all of it. Real-time architecture means every build that connects to Rillet is pulling data that's actually current. That's not a given with legacy systems built on batch processing.

How to Start

Look at your calendar. Find the thing you do every month that is rules-based and repeatable.

You don't need to know what a terminal is. You don't need to understand APIs before you start. The first build teaches you more than any amount of reading about it.

Good first projects:

- A journal entry that runs the same way every close
- A reconciliation that pulls from two systems
- A quarterly report rebuilt manually in Excel every month
- A close checklist spread across Slack and spreadsheets

Anything where the instructions would be the same document every time is probably a candidate for automation.

Richard Atkins, Accounting Director at Crisp, started because his company asked the team to find AI use cases. He looked at his actual work calendar, found vendor renewal management sitting directly in front of him, and automated it—including competitive analysis, RFP scoring, and build-versus-buy evaluation factoring in engineering time, hosting, compliance, and audit risk.

He opened Claude Code. Claude walked him through the rest.

Become a Certified Finance Engineer

The first cohort will define the standard.

At Rillet, we think Finance Engineers will become one of the most important operating roles inside modern finance organizations over the next decade. The finance professionals who learn to build now will design the workflows everyone else eventually works inside.

We created Rillet to eliminate the bottlenecks of legacy ERP systems. The people pushing that transformation forward inside companies are Finance Engineers: operators who don't just work around broken workflows, but rebuild them entirely.

That's why we're building the first certification for Finance Engineers.

How to qualify for Rillet's Certified Finance Engineering program:

1. Attend at least 3 weekly vibe coding sessions (or 1 live session)
2. Build at least one AI-powered workflow that a real finance team uses in production
3. Submit your build through the certification waitlist

The first cohort will set the benchmark for what the role means. Every Finance Engineer who comes after them will be measured against what this group built.

We run sessions every Friday at rillet.com/vibes. Recordings of past sessions are available there as well. If you want to see what Finance Engineers are actually building week by week, this is where to start.

Let's build together.

Rillet

Rillet is the AI-native ERP built by accountants for accountants, designed from the ground up around Continuous Close architecture. To learn more, visit rillet.com.