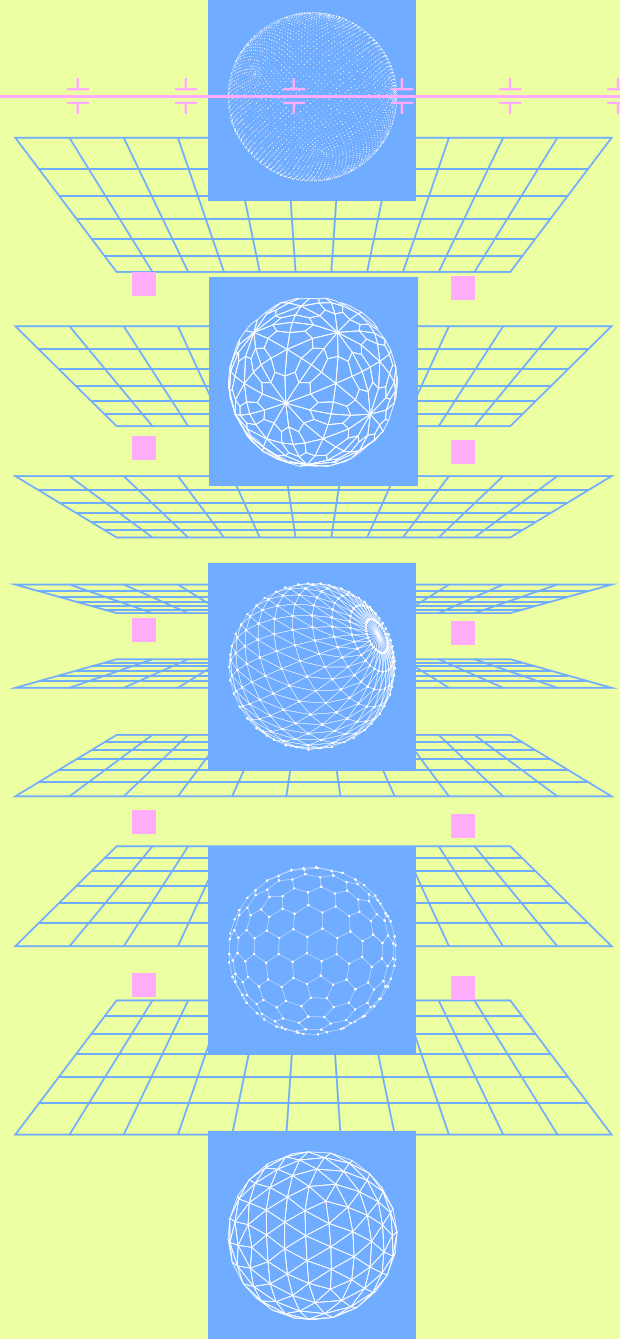


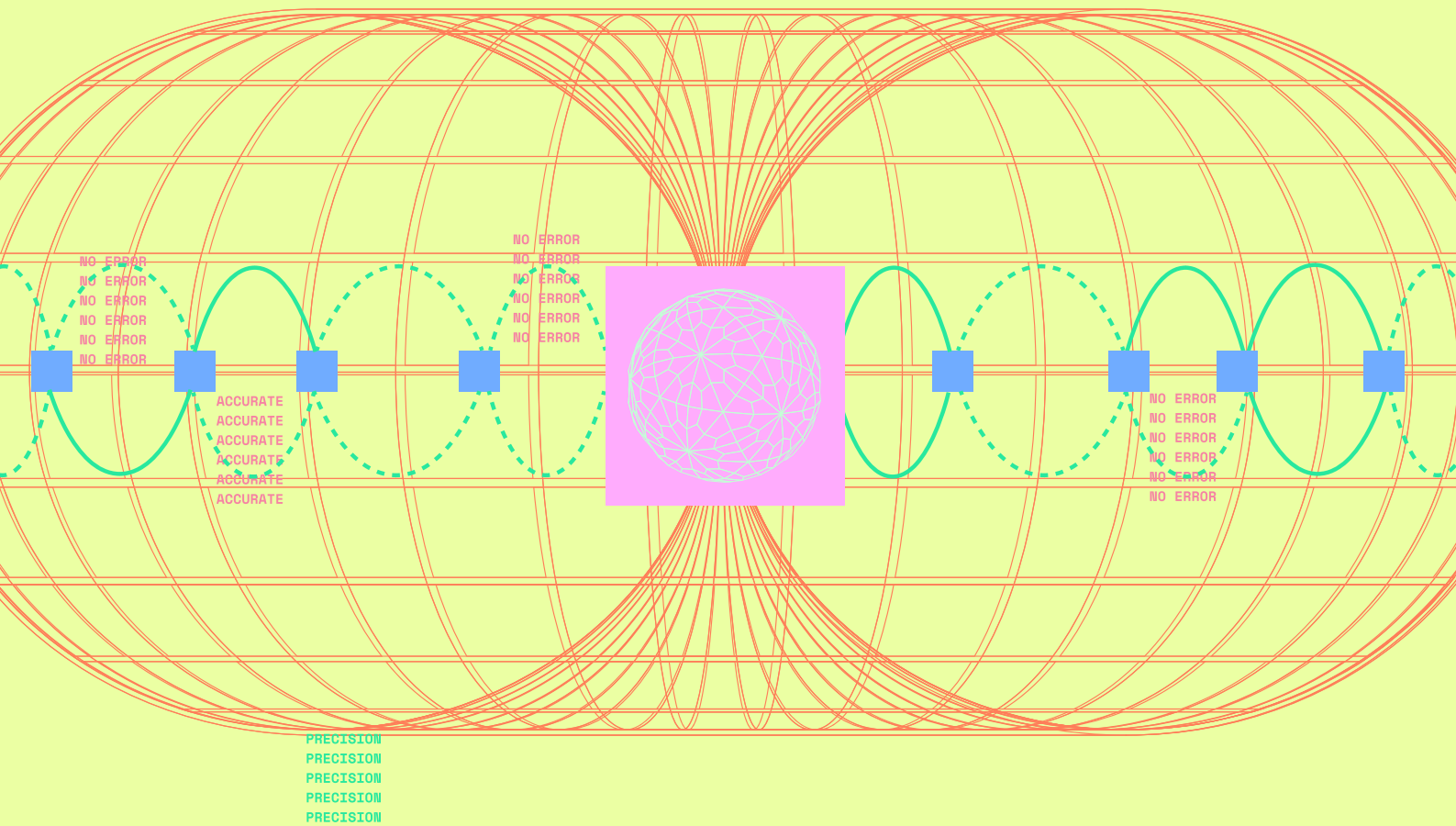


AI code review for financial services companies

01	Introduction: No room for error	03-07
02	Insight 1: Codify judgment as a part of the validation system	08-10
03	Insight 2: Maintaining understanding is PO	11-12
04	Insight 3: Redefine “done” to include detection	13-15
05	Conclusion: Code review has to mean more than reading a diff	16-17
06	References	18-19

Introduction: No room for error

FINANCIAL SERVICE
FINANCIAL SERVICE
FINANCIAL SERVICE
FINANCIAL SERVICE
FINANCIAL SERVICE
FINANCIAL SERVICE
FINANCIAL SERVICE

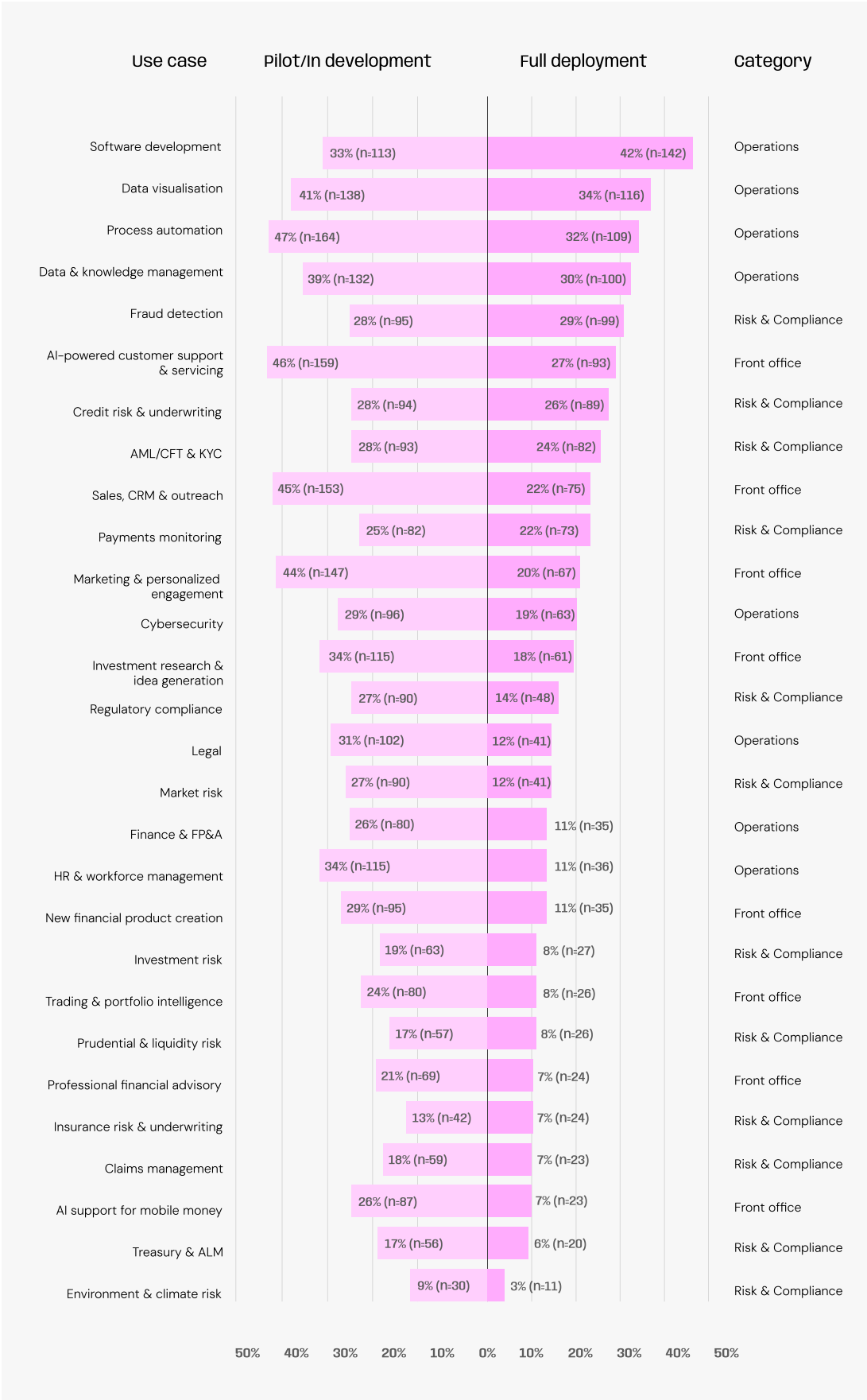


Knight Capital was founded in 1995. By 2012, it had grown into one of the leading high frequency traders in the world until it was derailed by a deployment error. Its software was run on eight servers, and one of them missed an upgrade; it was running an old version. When the old code fired, it sent runaway orders unchecked: 4 million executions, 397 million shares, in 45 minutes. Prices moved 5 to 10 percent or more across dozens of stocks. Knight Capital lost \$460 million and didn't recover [1].

In financial services, code moves money. Code review means making sure money gets where it's supposed to go. When something goes wrong, the stakes can be high, or in the case of Knight Capital, even existential to the business. Knight Capital's failure came down to one missed deployment, on one server, that nobody caught.

AI now writes far more code, far faster, than even a team of people can keep up with. A study from the University of Cambridge involving hundreds of respondents working in fintech and traditional financial institutions found that 42 percent of responding financial firms reported full deployment of AI in software development, with another 33 percent piloting or developing it, the highest full-deployment rate of any use case (Figure 1) [2]. In this sense, software engineering is the most adopted AI use case in financial services.

Figure 1: AI adoption by use case and deployment stage, with software development leading full deployment.

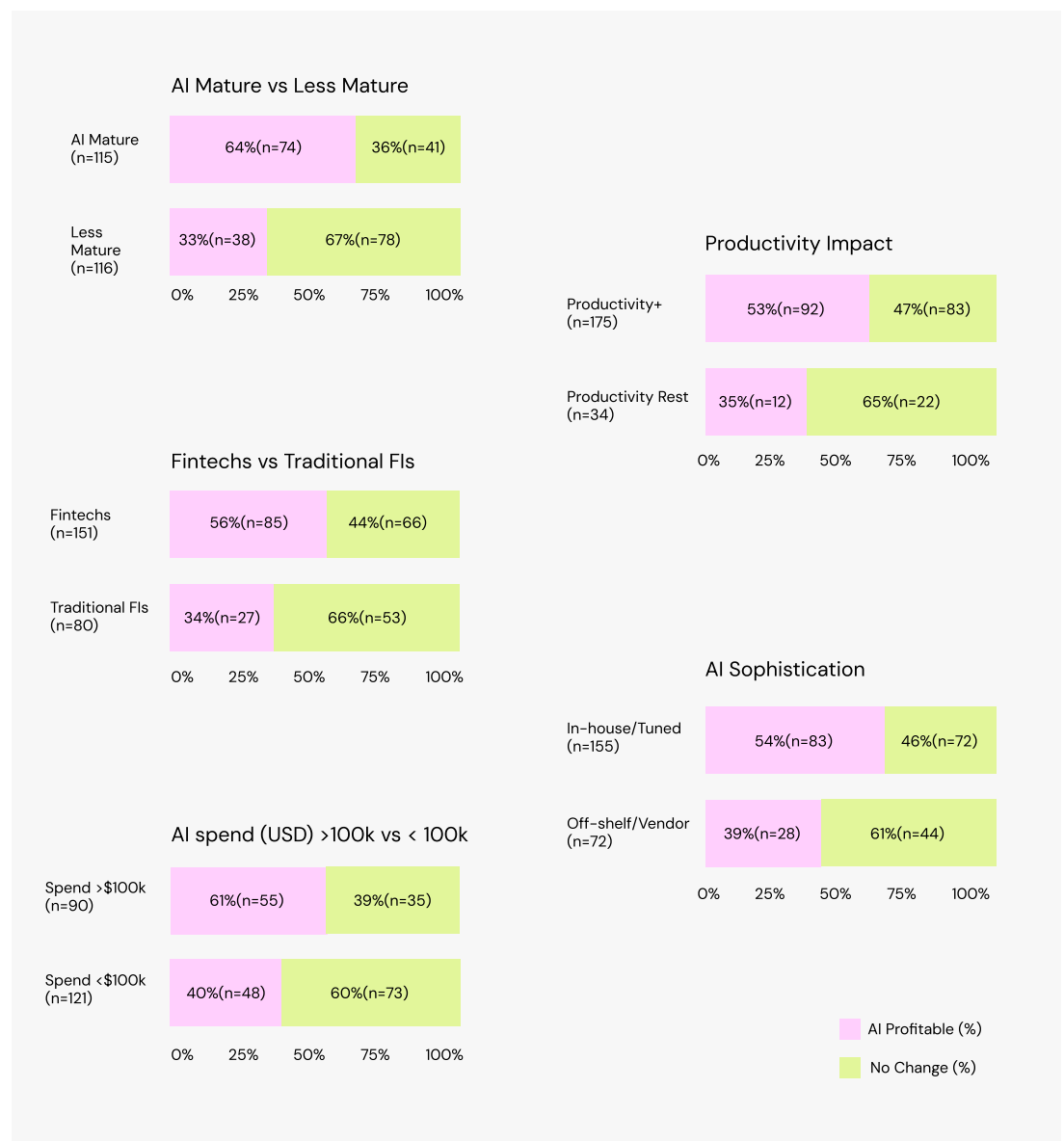


Source: Cambridge Centre for Alternative Finance [2]

“Every line of code is risk,” says Chintan Turakhia, Head of Wallet at Coinbase [3]. More code being contributed means more code needs review. 51 percent of respondents to the Cambridge study named loss of human oversight as an AI risk, the third most cited overall [2]. Yet Turakhia believes that because of the existing guardrails, financial services products are particularly well-positioned to innovate.

That willingness to innovate pays off. The same Cambridge research found that firms who have scaled or transformed their organizations with AI (which they define as “mature”) were more likely to report profitability gains than ones who are piloting or exploring (“not mature”). 64 percent of advanced adopters reported profitability gains, compared to 33 percent of less mature firms. Fintechs saw gains at 1.65 times the rate of traditional financial institutions (56 percent versus 34 percent), a divergence that tracks the 17-point gap in AI maturity between the two groups (Figure 2) [2].

Figure 2: Firms with more mature AI adoption report higher profitability gains



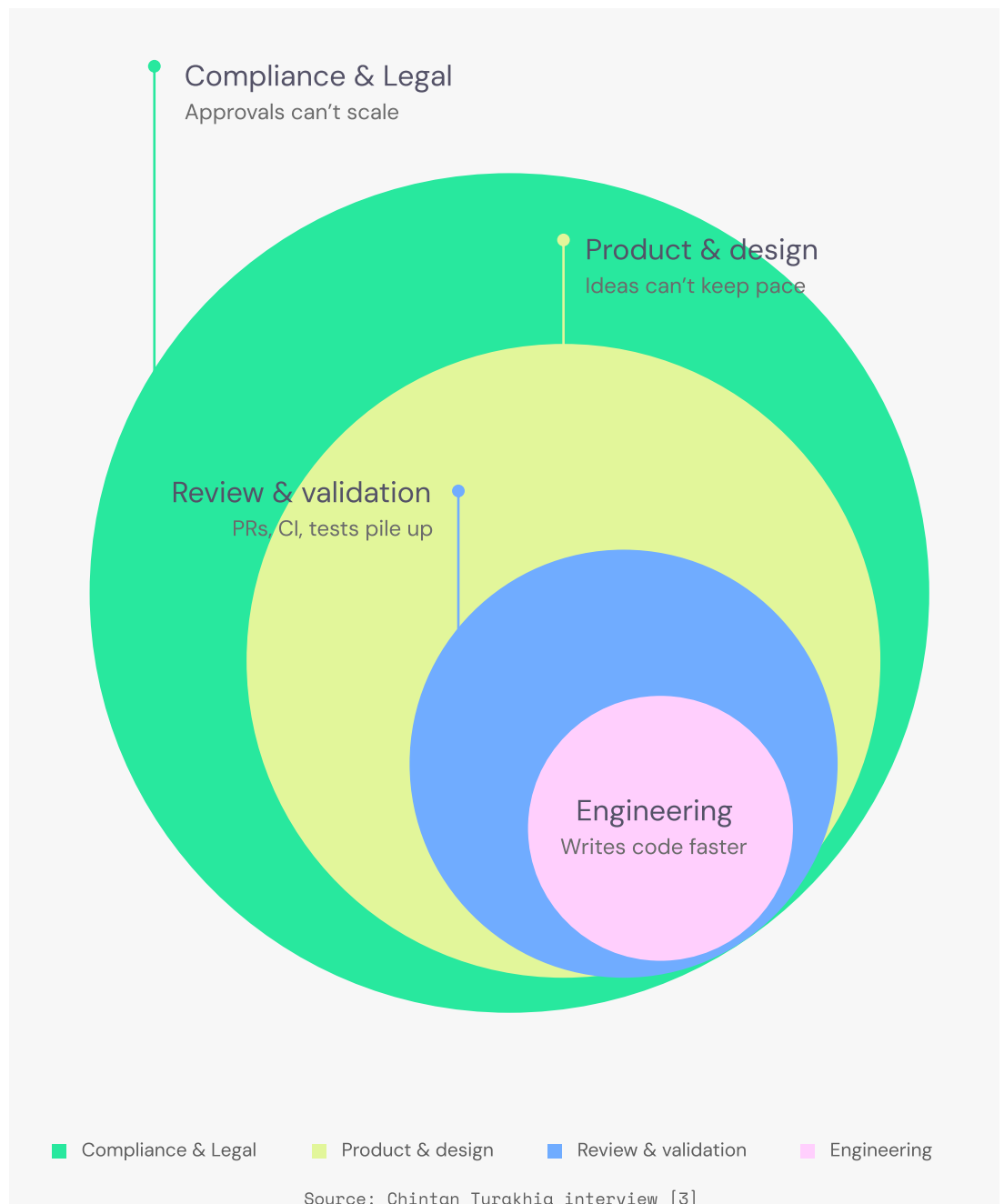
Source: Cambridge Centre for Alternative Finance [2]

More code is a growth opportunity. But once engineering teams can generate that much of it, validation becomes the constraint, not code production. Turakhia thinks about this as a series of nested circles. Each one is a bottleneck until the team solves it, and then the constraint moves outward (Figure 3) [3].

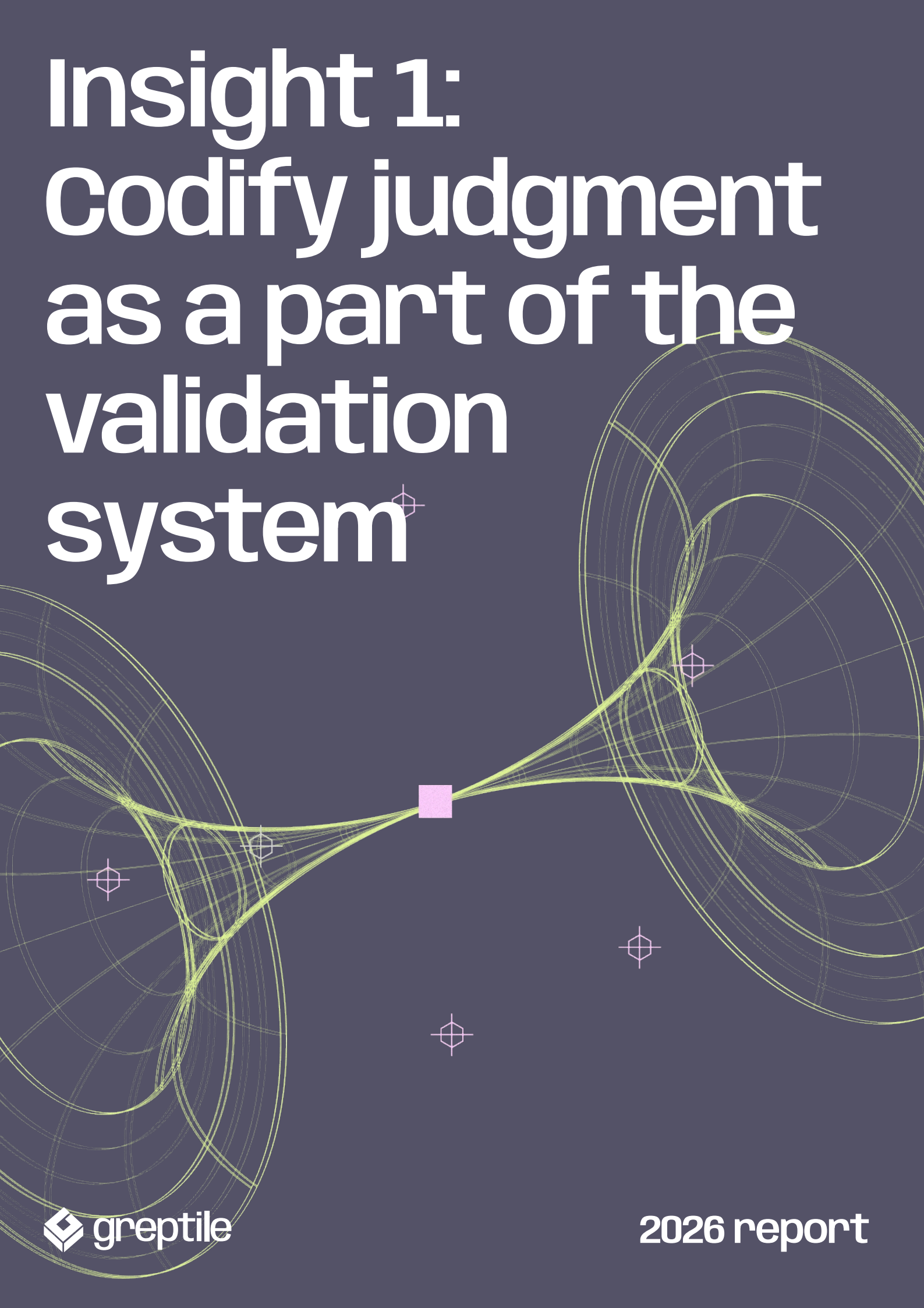
Once a team can ship faster, review and validation become the limit. Once validation keeps pace with that velocity, the limit becomes product ideas and design capacity. Solve that, and the limit becomes compliance and legal.

The leaders in financial services who figure out how to make AI-generated code trustworthy, to the point of zero errors, are the ones who widen each of these constraints in turn.

Figure 3: Nested circles of constraints



Insight 1: Codify judgment as a part of the validation system



Before he spent nearly a decade building software in financial services companies, Sesh Nalla worked on software for air traffic control. The stakes were literally life or death. “You need to think of everything that can possibly go wrong and you have to prevent it,” he says [4]. “There is no, ‘Oh, we’ve learned this, next time we’ll fix it.’ It’s too fatal.”

The stakes are slightly lower for wealth management software, but still very consequential. “You could lose a few hundreds, but you can’t lose billions,” he says. Doing code review well means understanding which parts of the products can cost billions, and getting more rigorous with those parts.

At Jane Street, production engineer Mark Doss describes the firm’s trading hot loop as a critical part of the system. With the hot loop, there isn’t even room for 0.01% of errors, so event-based monitoring replaces SLOs. That level of rigor isn’t necessary everywhere else; for less critical systems, Jane Street takes a more standard approach to monitoring [5].

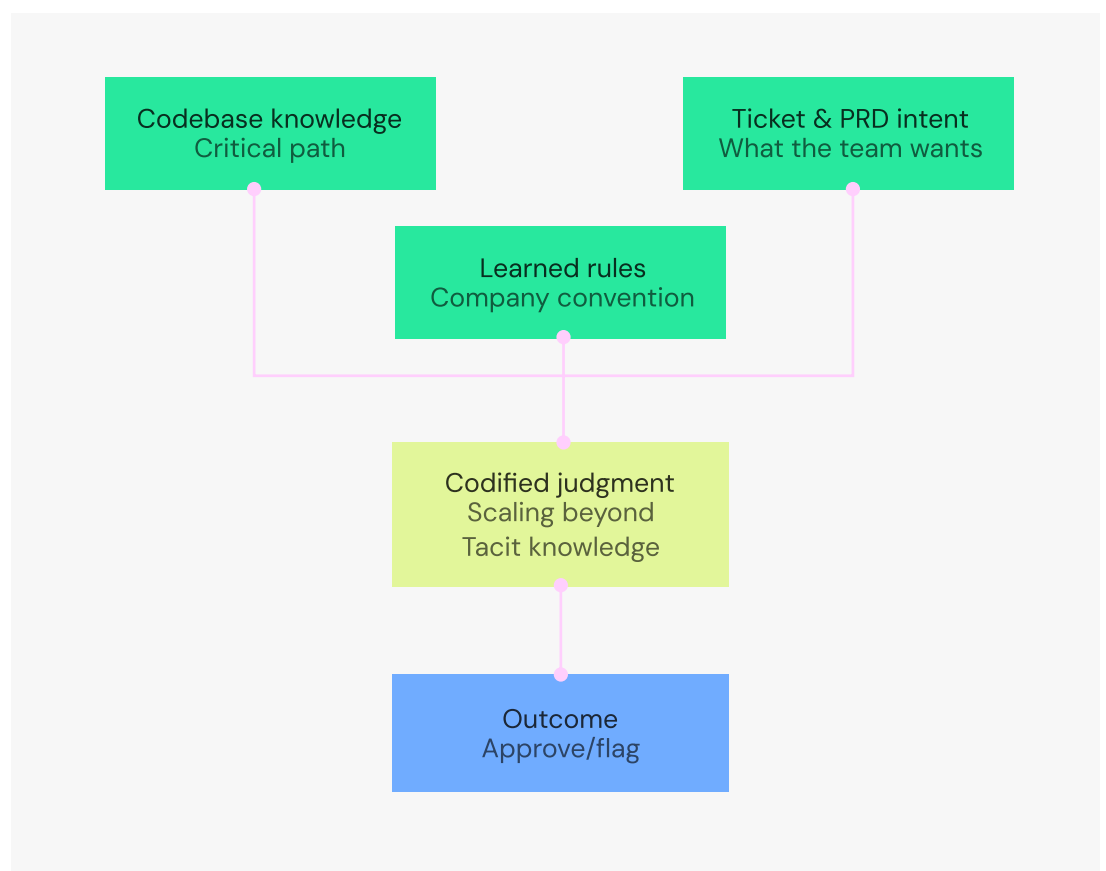
In order to make the most of your AI code review, you need to make sure this sense of judgment is codified into the tool. One artifact of codified judgment is knowledge of the codebase itself: which files are fragile, which patterns have broken before, where the known vulnerabilities live. As Linear’s Head of Product Nan Yu put it, the right shape for this is less a static document and more something closer to agent memory, a running catalogue of patterns and intentions, with a clear trigger to escalate to a human reviewer when something genuinely new shows up [6].

A review limited to the diff can't account for that context; Greptile builds this through a full repository index [7]. It continues with rules learned over time, house standards accumulated from how a team actually works, not a static style guide [8]. It also considers the intent behind a change: what a PRD or ticket says the team is trying to achieve, in addition to what the diff does (Figure 4).

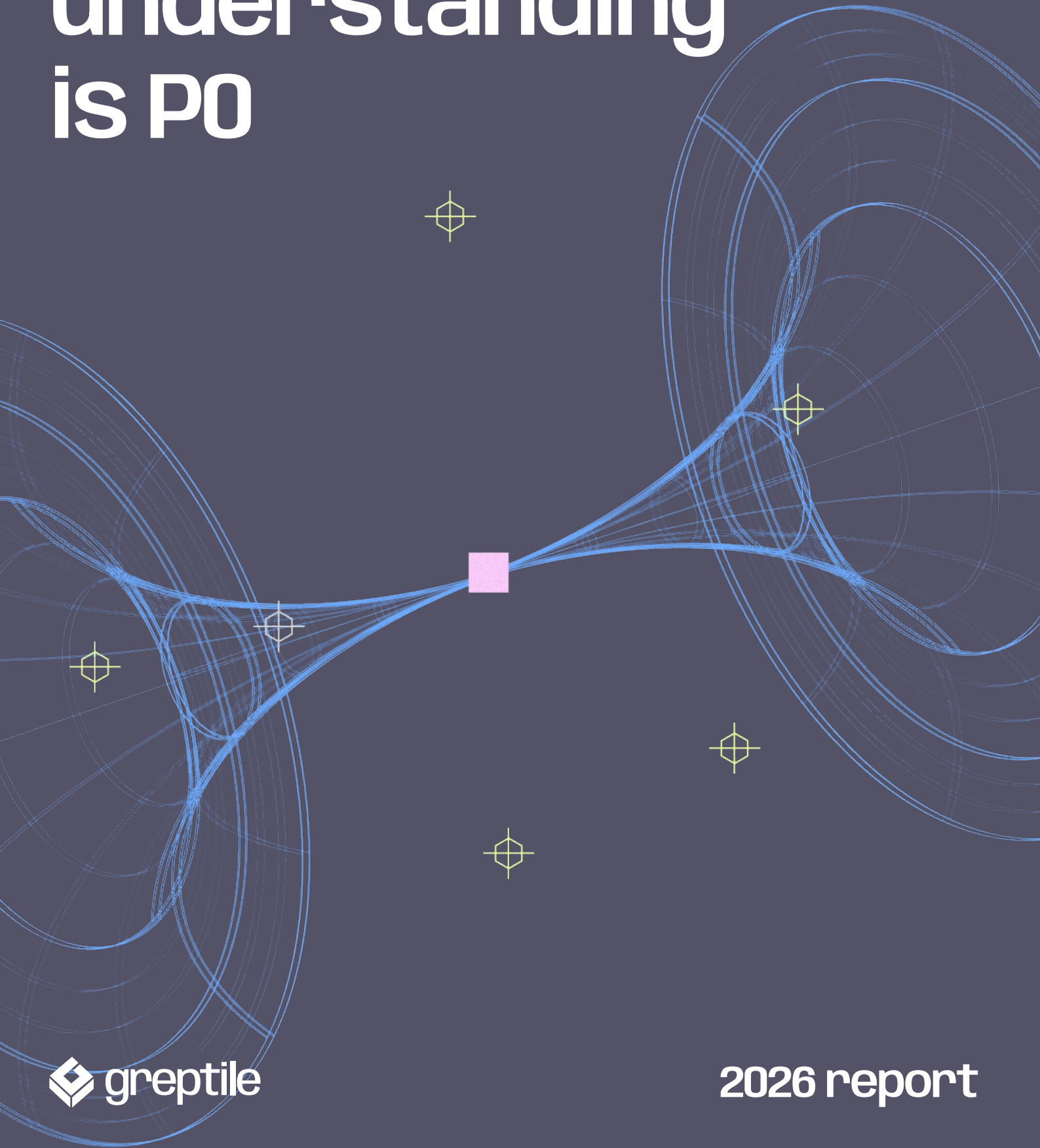
Coinbase's team uses a security risk framework to score every change and adjust reviewer requirements accordingly, from a single AI reviewer at the low-risk end to multiple mandatory human reviewers at the high end [3]. Not all code needs the same scrutiny, but every line needs some. A tool like Greptile can apply this principle by auto-approving low-risk pull requests (in beta) [9].

Sinthanai Natarajan, head of engineering at JPMorgan Chase, says that before deployment, you need to decide which parts of the system should be handled by procedural agents constrained to a fixed set of actions, and which can be handled by generative agents with more autonomy. Once that pattern is established, it can be standardized and handed off to engineering teams to build on, giving them real room to work, but within a defined structure rather than open-ended freedom [10].

Figure 4:
Inputs to
codified
judgment



Insight 2: Maintaining understanding is P0



Some companies may implement “slop free zones,” slang describing critical parts of software where AI coding tools aren’t allowed and human reviews are required. Sesh Nalla doesn’t conflate slop and AI. Now working as the VP of Engineering at Datadog, he says, “The aspect that I would emphasize more on, [whether] you used coding agents to produce it or engineers organically produced it, [is] the comprehension of, do we know when things go wrong at 3:00 AM.... We can understand what went wrong and then fix it.” [4]

The need to explain AI-augmented code hasn’t caught up to what regulators require. Cambridge’s study found that 79 percent of surveyed regulators rate explainability as critical or important to their objectives, but only 50 percent of industry respondents have adopted explainable AI methods [2].

The prior assumption was the software developer who typed out the code also understood it. “When a developer writes code themselves, they hold the mental model,” says Moshe Sambol, VP of customer solutions at Lightrun to American Banker [11]. “They know why they made each choice, where the assumptions are, and how the edges should fail. AI-generated code has none of that. When it breaks in production, debugging becomes a process of reverse-engineering a decision you didn’t make.”

AI code review needs to document the evidence a system produces when it runs, not just the code itself. Every artifact needs to give a reviewer enough to verify the run independently: the screenshots, logs, traces, and scripts that let a person or a downstream agent look at exactly what happened and confirm it. Bad evidence is worse than no evidence. Greptile’s TREX feature (which stands for Test, Run, and Execute) runs code and captures screenshots, logs, traces, and scripts along the way.

Maintaining this level of understanding, and being able to explain what’s actually happening, is a top priority for AI code generation and review.

Insight 3: Redefine “done” to include detection

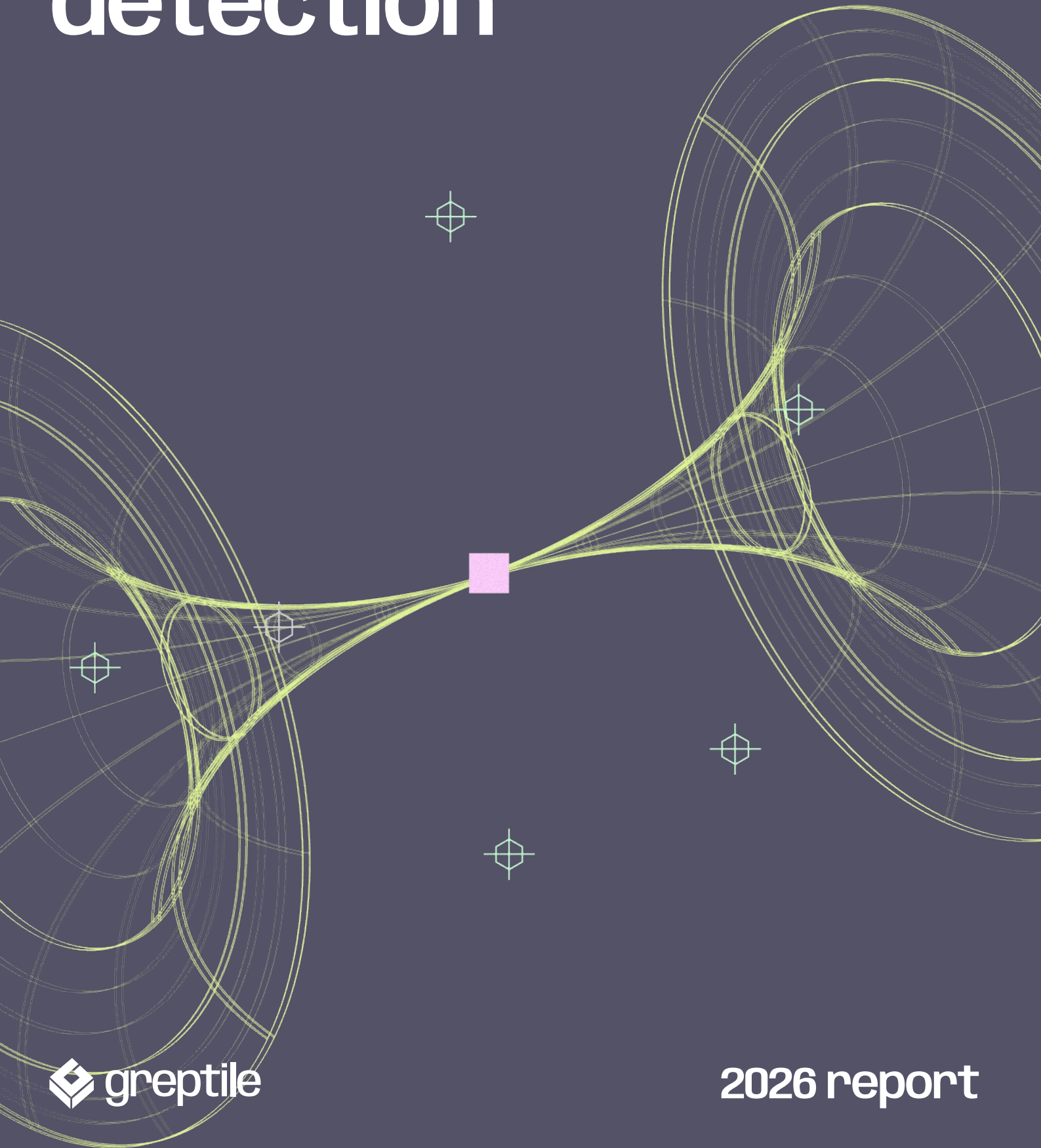
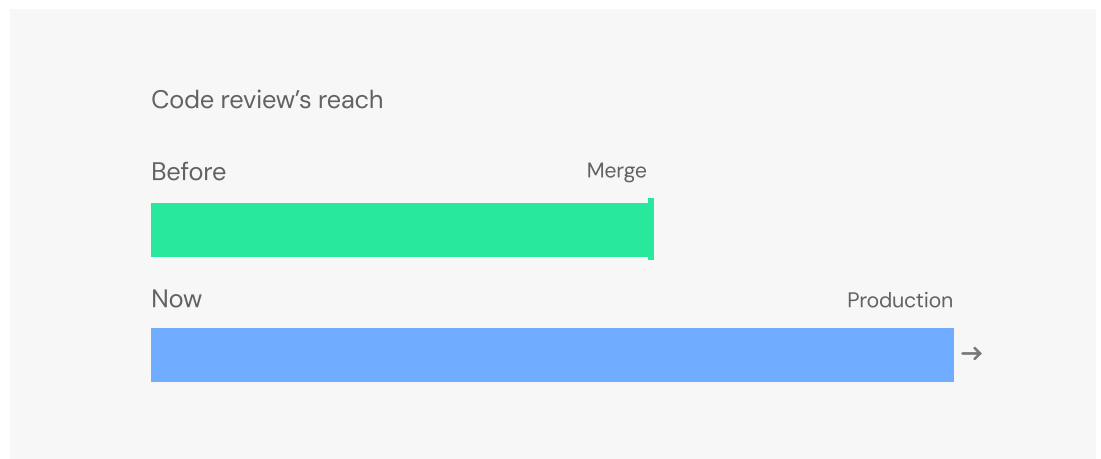


Figure 5: Code review's reach



The boundary of traditional code review was drawn at merge. It omitted failures that surfaced only once code is running, under real load, against real conditions and real data. Covering that means code review has to extend past the merge (Figure 5).

In practice, this means the software engineers working in validation would benefit from a wider breadth of experience. Paul Richards, Senior Vice President of Technology Operations at Citizens Bank, suggests that every developer should start in operations. He says, "When you have that mindset of what it takes to keep things running, you code different, you think differently in terms of the outcome, not just, 'Did I satisfy the requirements of this story?' If you take the mindset of, 'I'm going to be the one on call at 2 a.m. and I don't want to be called, so I'm going to write good code,' I think that's the mind shift" [10].

Jane Street operates this way structurally. "The people who are developing the software are also deeply integrated into writing the alerting," says Mark Doss. Production teams monitor their own systems; the people writing the core logic are also the ones writing the alerting around it [5].

At Jane Street, the highest-leverage signals are outcome-based, not service-based. They work under the premise that a healthy system doesn't mean nothing's wrong; instead, they watch out for a handful of signals that catch what the team didn't anticipate.

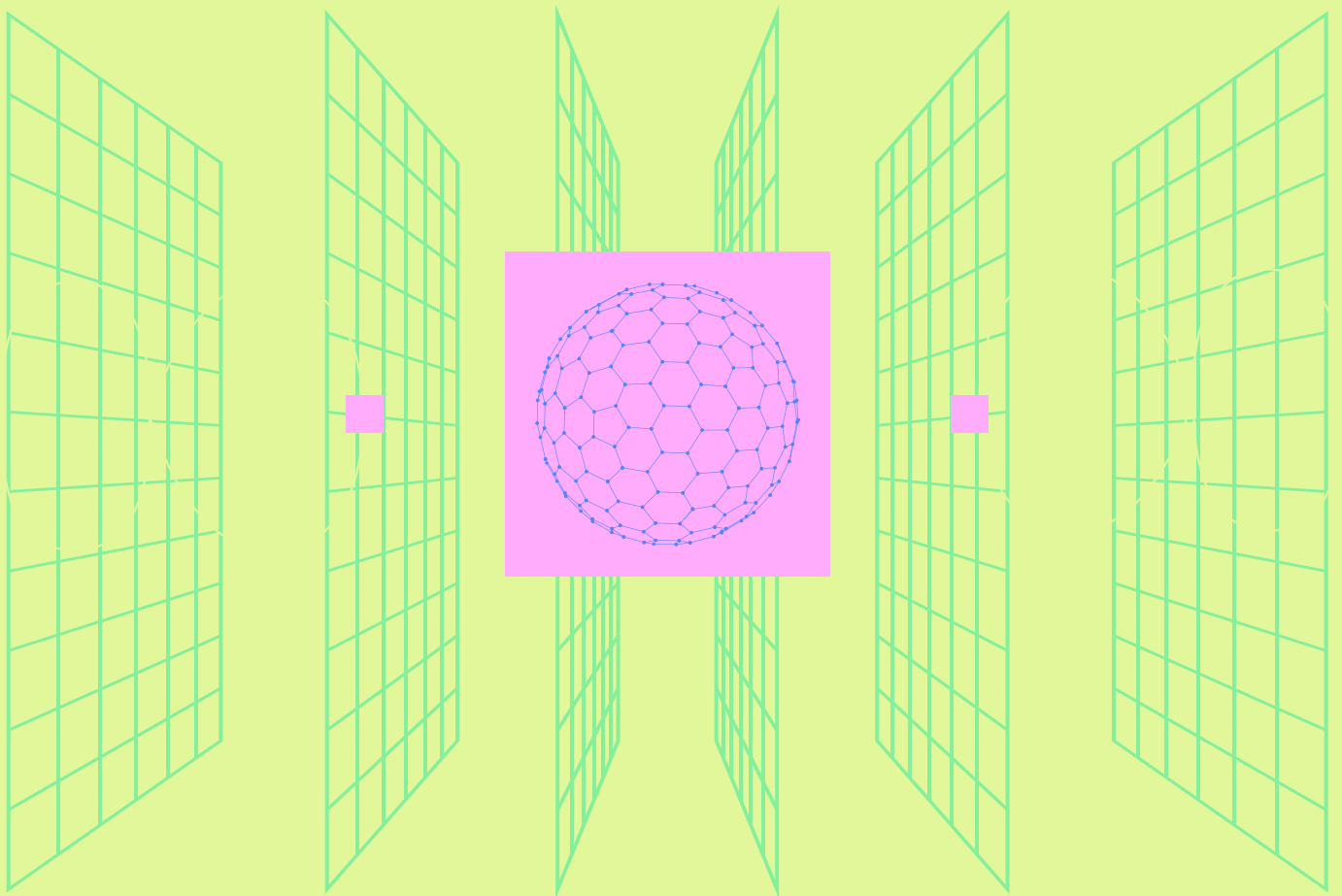
One example of this signal is what Jane Street calls "fill-too-good": an alert that fires when the firm makes too much money on a trade. "It can catch so many issues in so many different places in our tech stack," says Doss. "If you have problems with market data, if you have problems with instrument generation, if you have problems with the order engine, if you have a rogue trader, if a trader has fat-fingered something, anywhere in your trading stack, something can be caught by this one alert" [5].

A second example: if a desk normally trades 3 percent of market volume and suddenly hits 60 percent, the team immediately knows something's gone wrong, even if they're not sure what the cause is yet [5].

Expanding the jurisdiction of code review pays off in production. Banco BV cut troubleshooting time from about a week to about an hour after adopting LLM observability, a ~97% cut, while running more than 30 AI solutions in production and growing 30 percent a month [12].

Even if you're not running a trading hot loop, the underlying principle still holds in financial services businesses of all types and scales. Detection has to be designed on purpose, not left to default tooling, and ownership has to extend past merge.

Conclusion: Code review has to mean more than reading a diff

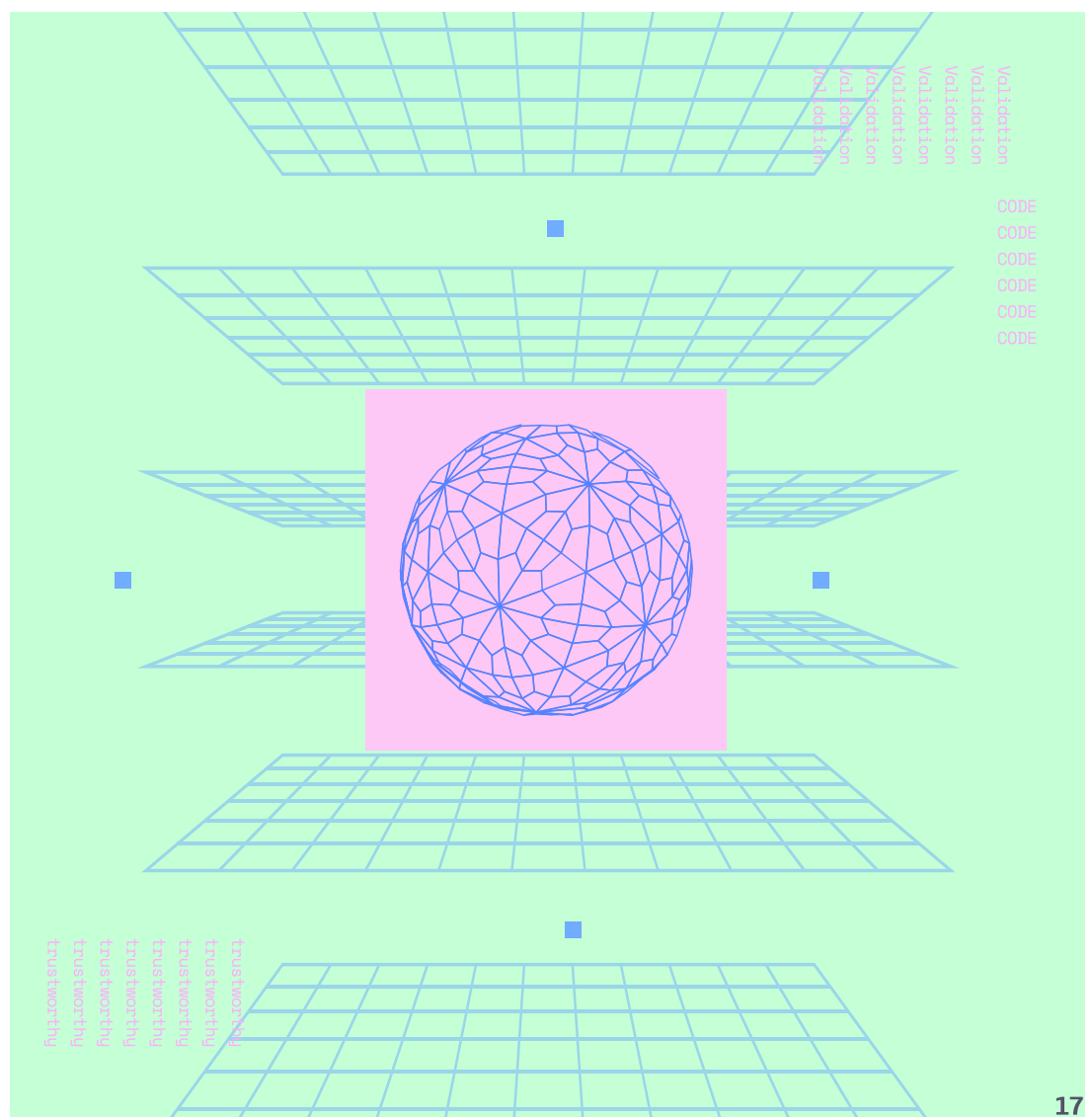


Code moves money in financial services. Code review is the process that makes the software trustworthy.

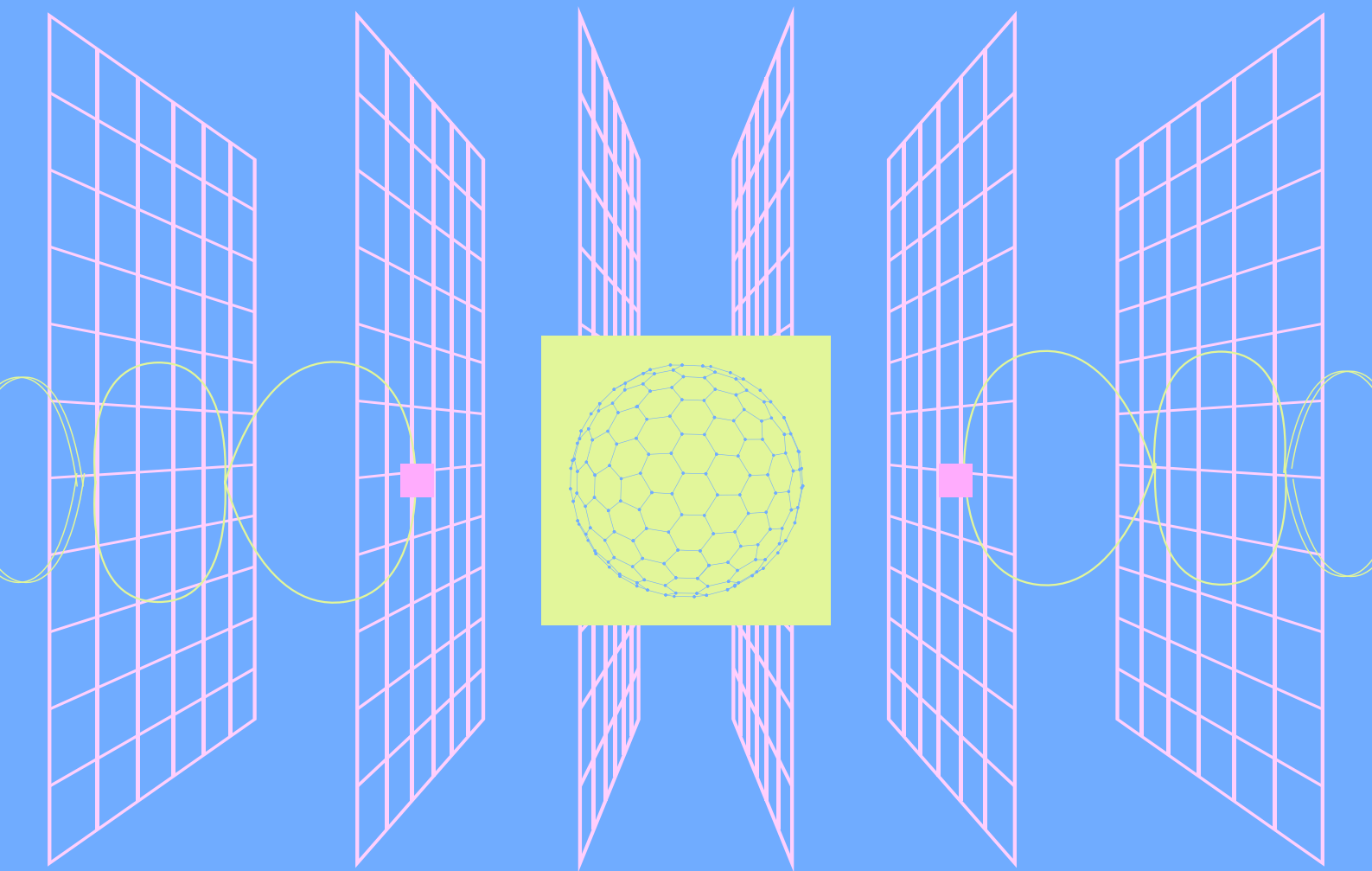
As AI introduces a much higher volume of code production, code review needs to keep up as well. People won't be able to review every line. Instead, teams will need to leverage AI to help with their code review.

In order to make the most of these tools, teams will need to codify their judgment before the code ships and make sure the AI code review tool has context. Team members will need to be able to explain the code after it ships, even if it was written by AI. As velocity increases, teams will need to catch the bugs that didn't show up in testing before production.

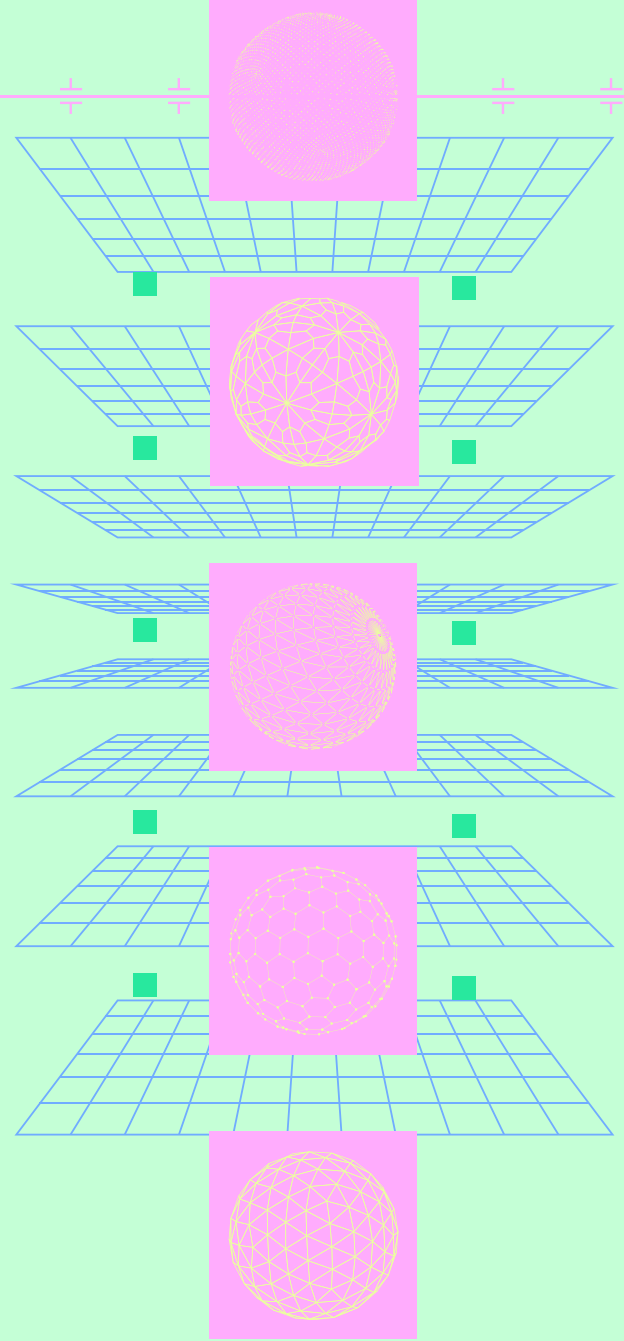
These three insights help mitigate risk and introduce opportunity. The firms that level up their validation with AI code review tools will be the ones whose software engineers can move fast without dreading what happens next.



References



- [1] Knight Capital Americas LLC, Securities and Exchange Commission Administrative Proceeding, Release No. 70694, October 16, 2013. sec.gov/files/litigation/admin/2013/34-70694.pdf
- [2] The 2026 Global AI in Financial Services Report: Adoption, Impact and Risks. Cambridge Centre for Alternative Finance, University of Cambridge Judge Business School, April 2026. jbs.cam.ac.uk/wp-content/uploads/2026/04/ccaf-2026-04-28-global-ai-in-financial-services-report.pdf
- [3] Chintan Turakhia, Head of Wallet, Coinbase. Interview conducted by Daksh Gupta, Greptile, 2026.
- [4] Sesh Nalla, Vice President of Engineering, Datadog. Interview conducted by Daksh Gupta, Greptile, 2026.
- [5] Mark Doss, Production Engineer, Jane Street. Production engineering talk. youtube.com/watch?v=zR9PpXWsKFQ
- [6] Nan Yu, Head of Product, Linear. x.com/thenanyu/status/2081121226265633159
- [7] Greptile documentation, "Full codebase context." greptile.com/docs/code-review/key-features#full-codebase-context
- [8] Greptile documentation, "Memory and learning." greptile.com/docs/how-greptile-works/memory-and-learning
- [9] Greptile documentation, "Auto-approve low-risk PRs." greptile.com/docs/code-review/key-features#auto-approve-low-risk-prs-beta
- [10] Sinthanai Natarajan, Head of Engineering, JPMorgan Chase; and Paul Richards, Senior Vice President of Technology Operations, Citizens Bank. youtu.be/hlyOUefamVo
- [11] Moshe Sambol, Vice President of Customer Solutions, Lightrun, quoted in American Banker. americanbanker.com/news/pressure-fomo-some-big-banks-are-rolling-ai-out-too-fast
- [12] Banco BV customer story, Datadog. dash.datadoghq.com/sessions/agent-observability-at-scale-governing-monitoring-and-securing-ai-agents-in-production



AI code review for financial services companies