Engineering Economics

The Hidden Costs of Technical Decisions

Lloyd Moore

2026

About This Book

Every engineering leader knows technical debt is expensive. This book shows you exactly how expensive – with the calculations.
Engineering decisions are economically invisible. You know losing a senior engineer hurts, but can you prove the $50K retention bonus is worth it? You suspect that "quick optimisation" will backfire, but how do you make that case to the CEO? You see $8M in engineering spend, but is it efficient or wasteful?
This book makes the invisible visible.
The case studies that follow are composite examples drawn from the author's consulting experience, leadership roles, and industry research. Names of individuals and companies have been changed, and certain details have been altered to protect confidentiality. Any resemblance to specific persons or organisations is coincidental. Where a scenario is described, it represents patterns observed across multiple organisations rather than a single company's experience. The calculations are real; the scenarios are representative.
This is not another checklist book. No motivational fluff. No "7 steps to engineering excellence." Every chapter opens with a specific incident and dollar amount, shows the full calculation with auditable

line-by-line breakdowns, and delivers a framework that can be applied to real decisions in the same week.
Written in the analytical style of The Economist for executives who do not want to be patronised.

A Note on Methodology

This book applies economic analysis to engineering decisions. That claim demands some transparency about how the analysis was constructed, where the numbers come from, and what the limitations are.
On composite case studies. Every chapter opens with a specific scenario – a named engineer, a dollar figure, a timeline. These scenarios are composites. They draw on the author's experience as a CTO and consultant, on published industry research, and on patterns observed across dozens of organisations. No single company experienced every detail as described. The composites exist because they are more useful than raw case studies: they isolate the economic pattern without the noise of any particular company's idiosyncrasies. Where a scenario credits "Meridian Systems" with a specific practice, the practice is real; the company name is not.
On economic calculations. Every cost figure in this book shows its working. Fully-loaded salary rates, time horizons, productivity assumptions, and discount rates are stated explicitly. The value of these calculations lies not in the specific outputs – which will differ for every organisation – but in the methodology. A reader whose senior engineers cost $280,000 rather than $210,000 can substitute their own figures and arrive at calculations that reflect their context. Where calculations span multiple years, net present value is applied. Where

estimates are uncertain, conservative and realistic scenarios are distinguished. Readers are encouraged to treat the frameworks as templates, not gospel.

On sourcing. Factual claims are attributed to published research where it exists. The relationship between latency and user behaviour, for instance, draws on decades of human-computer interaction research. The cost structures of engineering attrition reference industry salary surveys. Where published evidence is thin – and in the economics of engineering decisions, it often is – the book draws on professional observation and states this clearly. "Industry experience suggests" means something different from "a 2019 study found," and this book respects the distinction.

On correlation and causation. Engineering economics is not a controlled experiment. When a company redesigns its onboarding programme and observes faster ramp times, the improvement may owe something to the programme, something to a strong hiring cohort, and something to favourable market conditions. This book presents the correlations it observes and the causal mechanisms it finds plausible, but it does not claim laboratory certainty. Where confounding variables are significant, they are identified. The honest position is that these frameworks make costs visible and decisions more rigorous – not that they eliminate uncertainty.

On what this book is not. It is not an academic text. It does not conduct original research or present peer-reviewed findings. It is not a consulting engagement, tailored to one organisation's circumstances. It is an analytical framework, built on a career spent making and studying engineering decisions, for leaders who want to think more rigorously about the economics of their technical choices. The right way to read it is critically: test the frameworks against your own experience, substitute your own numbers, and discard what does not survive contact with your reality.

Part I: The Economics of Engineering Decisions

Chapter 1: The $500K Engineer

In 2018, a senior engineer at a $40M ARR SaaS company spent six months warning that the proposed database architecture would not scale. Product wanted to ship fast. Engineering leadership agreed with him but refused to push back. The decision: ship now, refactor later. He started interviewing that week. Not because of the technical call—those happen. Because his judgment did not matter. Eight months later, the system buckled under daily performance crises. Eighteen months later, the company had haemorrhaged five senior engineers and hired a fractional CTO to diagnose why.

The diagnosis was straightforward. Nobody at the executive level knew engineers were unhappy until they resigned. Exit interviews cited "better opportunities" and "competitive compensation". The CEO approved 15% raises for the remaining engineers. More left anyway. The real problem was not compensation. It was that information does not flow upward through the organisational hierarchy, and by the time problems reach the executive level, they have already metastasised into resignation decisions made months earlier.

Replacing those five engineers cost approximately $1.4M in recruiting fees, lost productivity and knowledge drain. The retention problem could have been solved for a fraction of that cost, but only if executives had known it existed. They did not, because junior engineers had told senior engineers, senior engineers had told managers, and managers

had decided the information was not worth escalating. The executives were the last to know.

That $1.4M represented an average of $280,000 per departure. The actual cost per senior engineer, when calculated with full economic rigour, ranges from $500,000 to over $1M. Most executives hear this figure and assume it is inflated. It isn't. The full calculation, explained in detail later, reveals how these costs compound across recruiting, vacancy, onboarding and tribal knowledge loss.

The question is not whether senior engineer attrition is expensive. It is. The question is why organisations spend millions replacing engineers who could have been retained for thousands—and what happens when they restructure incentives so that retention becomes economically rational for those who control resource allocation.

Why Hierarchies Filter Bad News Out

Organisations create hierarchies to manage complexity. The unintended consequence: information gets filtered at each layer. A junior engineer tells a senior that a technical decision looks problematic. The senior tells the engineering manager. The manager weighs whether this makes him look bad or is worth escalating. The VP of Engineering hears a filtered version, if anything. The CTO hears "we are handling it". The CEO hears "everything on track".

Each layer filters detail and diminishes urgency. By the time something reaches the executive level, it is either a crisis or it has vanished entirely. The filtering is not malicious. Middle managers believe they are doing their jobs by "handling problems at their level". They see escalation as failure. Presenting solutions rather than problems feels like professionalism. It functions as information suppression.

A software company with 120 engineers illustrates the pattern. The frontend team spotted performance issues with the new dashboard in March. By April, engineers were openly discussing them in code reviews. The engineering manager knew by May and began investigating solutions. The VP of Engineering heard about it, framed as "we are optimising dashboard performance". The CTO learned in July that "some performance work" was happening. In August, the company's largest customer complained the dashboard was unusable. The executive team treated this as a sudden crisis requiring an emergency response. Engineers had known for five months.

The upshot: executives making resource allocation decisions operate on information that is months stale and systematically scrubbed of bad news. Engineers know in August that the database will not scale. Managers know in October that engineers are frustrated. VPs sense a morale problem in December. The CTO learns in February that engineers are resigning. Each group thought they were handling the problem appropriately. None realised they were creating information latency that made the problem unsolvable.

The convenient fiction of the chain of command makes this worse. Many organisations treat skip-level conversations—executives talking directly to engineers several levels down—as inappropriate. The objections are consistent: it undermines middle management authority, breaks the chain of command, creates a perception of distrust, or represents micromanagement. These concerns are presented as organisational health considerations. They are actually organisational antibodies protecting middle management from accountability.

A CTO who talks directly to junior engineers hears what is actually happening. A CTO who relies on filtered reports hears what management wants them to hear. The taboo against skip-levels ensures the latter. Middle managers prefer this arrangement because it gives them control over the flow of information. They can present

problems they have solved and suppress problems that reflect poorly on them. The organisation pays for this preference through information latency that turns small problems into crises.

The economic argument against skip-levels is that executive time is expensive and should be devoted to strategy, not tactics. This logic collapses when strategic decisions rest on systematically wrong information. A CTO who spends several hours per week talking to engineers across all levels learns what is actually broken, what is actually blocking productivity, and which engineers are mentally checking out. This information has direct strategic value: it reveals where to allocate resources, which managers are effective, and which technical bets are failing. The alternative—relying on reports filtered through three or four layers—is cheaper per hour and vastly more expensive per bad decision.

The time investment need not be enormous. Allocating several hours per week (the specific amount matters less than the consistency) for direct conversations across levels provides ground truth. As organisations grow, the frequency per person decreases, but the coverage remains. A CTO might talk to each engineer once per quarter rather than once per month. This maintains a connection to reality without consuming the executive calendar. The engineers who get these conversations tell executives things they would never tell their manager. Executives hear problems three to six months earlier than they would through proper channels. Early warning makes intervention cheaper.

A payments company discovered this after losing three senior engineers in a quarter. The new CTO instituted weekly office hours: any engineer could book 30 minutes, no agenda required, no manager approval needed. The first month uncovered three festering problems. The deployment system was so unreliable that engineers avoided deploying on Fridays, fearing weekend pages. The monitoring system

spewed so many false alerts that engineers had learned to ignore it. The API documentation was so outdated that new engineers spent their first month learning through trial and error. None of this had reached the executive level through normal channels because each manager thought they were handling it. The CTO allocated budget to fix all three problems within the quarter. Six months later, voluntary attrition had dropped to near zero.

Why Engineers Leave (And It Is Not Usually Compensation)

Engineers leave when the expected value of staying falls below the expected value of alternatives—and exit interviews measure the wrong variables. Compensation gets cited because it is measurable and socially acceptable. The actual reasons are harder to articulate and potentially awkward. Three patterns recur.

The first is loss of agency. Engineers are told to build systems they know will fail. Technical judgment is consistently overruled for non-technical reasons. They watch decisions being made that will cause problems, raise concerns and are told to implement anyway. When the predicted failure arrives, they are blamed for not preventing it. This is not about ego or engineers wanting to be in charge. It is about being asked for expert judgment, then watching it ignored.

A fintech company decided to build their own authentication system rather than use established solutions. Senior engineers argued this was an unnecessary risk for a non-differentiating feature. Product management insisted on a custom implementation to support specific user flows. Engineering leadership agreed with the seniors but did not push back. The custom authentication system shipped with three security vulnerabilities in the first month, required emergency patches, and consumed six months of engineering time that could have gone to

revenue-generating features. The senior engineer who had argued against it started interviewing the week the decision was finalised. He did not wait for the failures he knew were coming.

The economic toll: wasted implementation time, ongoing maintenance burden and engineer disengagement. The company spent $180,000 building something that could have been bought for $12,000 annually. The engineer they lost would have built features estimated to generate $400,000 in annual recurring revenue, based on product roadmap priorities. The decision to override his judgment cost approximately $580,000 in the first year alone—not counting the cost of replacing him.

The second pattern is technical debt becoming unpayable. The team identifies critical infrastructure work: the database needs replication, the deployment system needs automation, the monitoring needs improvement. These get deprioritised quarter after quarter behind feature work. Engineers watch the system strain under load, knowing exactly what is coming and when it will break. When it fails, they are asked why they did not prevent it. They stopped arguing for the infrastructure work after it was deprioritised the third time. They leave before the inevitable crisis because they do not want to be blamed for a failure they predicted and could not prevent.

An e-commerce company deferred database infrastructure work for eighteen months. Engineers requested budget to implement read replicas and improve query performance. Each quarter, feature development took priority. Load climbed 40% annually while infrastructure remained static. In month 19, the database became the bottleneck for the entire platform. Response times degraded from 200 milliseconds to 4 seconds during peak hours. Revenue fell as conversion rates dropped. The company spent $240,000 on emergency infrastructure upgrades and lost an estimated $1.2M in revenue during the degraded performance period. Two of the three senior engineers who had requested the infrastructure work had already left. They had

correctly predicted the failure timeline and refused to be present for the crisis.

The third pattern is smart people forced to do stupid work. Senior engineers earning $180,000 annually are assigned to maintain systems that should be deprecated. Busy work is prioritised over meaningful problems. Process theatre consumes time: estimation ceremonies that produce numbers no one believes, architecture reviews that change nothing, documentation that no one reads. The economic waste is paying premium rates for work that could be done at half the cost, while burning out senior talent on tasks beneath their capability. They leave to find work that uses what they are good at.

One SaaS company had a reporting system built six years earlier that served twelve customers generating $80,000 annual revenue. The system required constant maintenance due to technical debt. A senior engineer earning $190,000 annually spent approximately 60% of her time maintaining this system. The company was effectively paying $114,000 per year to support $80,000 in revenue. When she proposed deprecating the system and migrating the twelve customers to the new platform, she was told those customers were "strategic relationships" that required continued support. She left three months later. Her replacement cost $220,000 to recruit and onboard. The company could have migrated all twelve customers for approximately $40,000 and freed her to work on the core product.

Compensation is a real factor in some resignations. The signal is timing. If an engineer raises compensation concerns directly before job searching, and their compensation is genuinely below market for their scope, it is a compensation problem. If they cite compensation only when pressed during exit discussion, it is probably not. They were already looking for other reasons and found an offer that provided justification.

Legitimate compensation issues have patterns. The engineer is significantly below market for their actual scope. They are watching junior engineers hired at their salary. Their compensation has not adjusted for expanded responsibility. Critically, they bring this up directly before job searching. "I believe I am underpaid relative to market. Can we discuss this?" This is a compensation problem that can be solved with a compensation solution.

Compensation as excuse looks different. The engineer has been interviewing for months, receives an offer for 30% more, and cites market rates during exit discussion. When pressed about whether a matching offer would keep them, they equivocate. "I will consider it, but I have already accepted the other role." Compensation was not the motivating factor. It was the justification factor. The engineer had already decided to leave for the reasons described earlier but uses compensation as the socially acceptable explanation.

The test: would a 20% raise keep them? If yes, it is compensation. If no, it is something else and they are being diplomatic. Most engineers who leave for loss of agency, unpayable technical debt or meaningless work cannot be retained with money alone. They have concluded the work environment is broken. Compensation becomes the explanation because "I am leaving for more money" is easier than "I am leaving because my judgment does not matter and I am tired of building things I know will fail".

A developer tools company learned this distinction after conducting thorough exit interviews with departing engineers. Of the eight engineers who left citing compensation, the company made counter-offers to five (the ones they most wanted to retain). Three declined the counter-offers despite the offers matching or exceeding their new roles. Follow-up conversations revealed that compensation had not been their real concern. They were leaving because the product roadmap kept pivoting, making their work feel pointless.

They cited compensation in exit interviews because it was easier than explaining that they did not believe in the company's direction.

The Early Signals Executives Miss

The warning signs exist months before resignation but surface at different organisational levels at different times. Executives typically see only the final stage, when intervention is no longer possible. Junior and mid-level engineers see the first signals six to twelve months before senior engineers leave. Senior engineers stop fighting for technical decisions. Architecture reviews become rubber stamps. "We tried that, leadership said no" becomes a common response. The technical debt backlog grows with no allocation. Junior engineers notice because they are the ones implementing the decisions that senior engineers stopped arguing against. They see the seniors quietly disengaging.

A junior engineer at a logistics company observed this pattern over six months. The senior architect who had previously written detailed code review comments began approving everything with "LGTM" (looks good to me). In architecture discussions, he stopped proposing alternatives and simply agreed with whatever product wanted. When the junior asked privately why, he said "I have learned that my opinion does not change outcomes, so I am saving time by not giving it". The junior realised the senior was leaving before the senior had even started interviewing. Three months later, the resignation came. It surprised management completely.

Senior engineers experience different signals four to eight months before they leave. Pattern recognition: this will fail and leadership will not listen. Moral injury: being forced to build what one knows is wrong. Loss of faith: my expertise does not matter here. They stop

arguing in meetings, start rubber-stamping technical decisions, reduce code review commentary from detailed to perfunctory. This often looks like maturity to executives—they are not arguing as much, they seem more aligned. It is actually resignation.

The behavioural change is visible to anyone who works closely with the engineer but often goes unremarked because it reduces friction. A senior engineer who previously pushed back on unrealistic timelines and questioned technical approaches becomes agreeable. Product managers are pleased. Engineering managers interpret this as the engineer becoming a better team player. The engineer has actually checked out mentally and is conserving energy for interviewing. They have concluded that arguing is pointless and are simply executing instructions while they search for an exit.

Managers see changed behaviour two to four months before resignation. Less engagement in one-on-ones. The engineer updates their LinkedIn profile, attends more industry events, documents their work more thoroughly than before, trains junior engineers on their systems. Managers often fail to escalate because high attrition reflects poorly on their management. They hope to solve it quietly. By the time the pattern becomes obvious, the engineer has offers.

An engineering manager at a media company noticed one of his senior engineers had started documenting everything: architecture decisions, deployment procedures, system quirks. The manager thought this was positive—finally, the tribal knowledge was being captured. What he missed was the motivation. Engineers typically document thoroughly when they are leaving and feel guilty about the knowledge gap they will create. The documentation is part of their mental transition out. The manager realised this only in retrospect, during the exit interview.

Executives see the final stage zero to one month before resignation: "The senior engineer gave notice." Complete surprise. But the information existed all along at lower levels. It simply did not flow

upward because the organisation's information architecture filters bad news out of executive visibility.

Disengagement and attrition share common antecedents—compensation dissatisfaction, limited advancement, weak management—which makes isolating the causal mechanism difficult. What the economic analysis reveals is that the costs materialise regardless of which factor dominates.

The economics of a single departure are damaging enough. The cascade effect makes them catastrophic.

Why One Departure Becomes Five

One departure is a data point. Three is a pattern that remaining engineers notice. If a respected colleague is leaving, maybe I should look too. Engineers assume departing colleagues know something they do not. The assumption is often correct. The departing engineer has concluded the situation is not fixable and has already explored alternatives. Those who remain wonder what they are missing.

External recruiters notice the pattern. LinkedIn activity increases from the team. Multiple engineers from the same company appear in recruiter searches. Outreach intensifies. A retention problem becomes a recruitment problem. Engineers who were not previously considering leaving begin taking calls. Some discover opportunities they find attractive. The rate of departures accelerates.

A cloud infrastructure company lost two senior engineers within three weeks. Over the following three months, five more departed. Post-mortems revealed that the first two departures triggered widespread questioning. Engineers asked: "Why are they leaving? Do they know something about the company's financial health we don't?" The reality: the first two had left for the reasons described earlier—loss

of agency and technical debt. But their departures created information uncertainty that recruiters exploited. The cascade cost approximately $2.2M in replacement costs and delayed three major product initiatives by a combined eight months.

The pattern appears differently at different company stages. A Series A startup with 22 engineers lost two senior engineers within a month. Both cited the same issue: the technical roadmap kept changing based on investor feedback, making their work feel pointless. The impact was disproportionate. With only four senior engineers total, losing two meant losing 50% of architectural expertise. Remaining engineers absorbed crushing workload while the company scrambled to hire replacements in a competitive market. Two projects were abandoned entirely because nobody remaining understood the technical context. The startup's runway shortened by three months as hiring costs consumed cash reserves faster than planned. For startups operating with limited runway, attrition creates existential risk that established companies do not face.

The knowledge drain is expensive in ways that are hard to quantify but surface as mistakes over the following quarters. The senior engineer who left knew which three lines of code were critical and which 3,000 could be deleted. Knew which customers had special requirements and why. Knew which systems were stable-but-ugly versus actually broken. This knowledge leaves with them and gets rediscovered through expensive mistakes: deleted code that should not have been touched, customer workflows that break, technical debt that turns out to have been load-bearing.

A payments company discovered this six months after losing a senior engineer who had built their reconciliation system. The engineer had never documented that certain transaction types required special handling due to a quirk in one payment processor's API. The quirk was not mentioned in the processor's documentation; the engineer had

discovered it through trial and error three years earlier. When a new engineer modified the reconciliation logic, transactions from that processor started failing. The company spent two weeks debugging, ultimately discovering the undocumented quirk through support tickets with the processor. Those two weeks cost approximately $45,000 in engineering time and caused $180,000 in failed transactions that required manual reconciliation.

These stories accumulate. The question is whether organisations understand what they add up to.

The Full Economic Cost of Losing a Senior Engineer

When a senior engineer earning $200,000 annually departs, the total cost ranges from $500,000 to over $1M. Most executives assume this figure is inflated. It is not. The costs decompose into four categories, each invisible to standard financial reporting.

Direct replacement costs total $85,000 to $100,000: recruiting fees (20-25% of first-year compensation), sign-on bonuses ($20,000-$40,000 in competitive markets), and internal recruiting expenses. These figures vary by company stage—a Series A startup handling recruiting internally appears to spend $15,000 but consumes founder time worth $36,000 in opportunity cost. Appendix A provides detailed breakdowns by company stage.

Vacancy costs total $50,000 to $100,000 over a typical three-to-six-month hiring period. When a senior engineer departs, approximately 60% of their work gets absorbed by remaining engineers, creating context-switching overhead and 10-15% productivity loss per engineer carrying the load. The remaining 40%—platform improvements, technical debt reduction, architectural planning—quietly vanishes from the roadmap. The deferred work

creates compounding costs: database optimisations postponed degrade into emergency interventions costing multiples of the original scope.

Onboarding and ramp-up costs total $92,000 to $125,000. A replacement operates at roughly 25% productivity in month one, 50% in months two and three, 75% in months four and five, reaching full effectiveness around month six. But most senior engineers take closer to twelve months to match the departed engineer's domain expertise. Add the cost of engineers doing the onboarding—10-15 hours per week in month one, 5-8 hours in months two and three—and the total exceeds $90,000.

Tribal knowledge loss is the hardest to quantify: $100,000 to $300,000 over the following year. The departed engineer knew which parts of the codebase were fragile, which customers had special requirements, and why that database query was written in a seemingly inefficient way (because the "obvious" optimisation causes data corruption under conditions discovered three years ago). This knowledge leaves with them. Mistakes from missing context occur three to five times per departure in the first year, each costing $12,000-$15,000 in investigation and repair. Slower decision-making—questions that took the departed engineer 30 seconds now requiring three hours of code archaeology—adds $14,000 over six months.

Total cost per departure: direct replacement ($85,000-$100,000) plus vacancy ($50,000-$100,000) plus ramp-up ($92,000-$125,000) plus tribal knowledge ($100,000-$300,000). Conservative total: $327,000-$625,000. Realistic total accounting for project delays: $500,000 to $1M. The ranges represent the 25th to 75th percentile of observed outcomes; individual departures occasionally fall outside in both directions.

A note on hourly rates used throughout this book: fully-loaded rates range from $90 per hour for mid-level engineers to $230 per hour for VPs at Series A to Series C companies. The formula: annual

compensation divided by 2,080 hours, multiplied by 1.25-1.35 for employer overhead. The ratios hold regardless of specific dollar amounts.

These costs scatter across budgets and hide in noise. Consider how a $500,000 departure appears in financial systems: the $47,000 recruiting fee sits in HR's budget; the $30,000 sign-on bonus in engineering's; the $75,000 vacancy cost manifests as reduced velocity (which appears nowhere); the $100,000 onboarding cost gets charged to whatever projects the onboarding engineers work on; the $200,000 tribal knowledge loss surfaces months later as production incidents and project delays. No single budget line shows "$500,000 attrition cost." The CFO sees $77,000 in direct costs and approves them as reasonable. The remaining $423,000 never reaches financial review.

This creates systematic underestimation. When a VP proposes $200,000 in retention investment, the CFO compares it against visible replacement costs ($80,000) and concludes it does not pencil out. The correct comparison: $200,000 investment against $500,000 per departure times three annual departures ($1.5M), yielding a 6.5-to-1 return. A healthcare technology company with 95 engineers discovered this gap firsthand: visible attrition costs per departure were $180,000; comprehensive tracking revealed the true cost was $467,000. The company had been losing nearly 3% of annual revenue to preventable attrition without realising it.

Building an Attrition Cost Calculator

The simplest credible estimate uses a research-validated multiplier: 2.5× annual salary. Academic studies of knowledge worker turnover consistently find replacement costs between 1.5× and 3× salary; for

senior engineers, the multiplier tends toward the higher end. A $200,000 engineer therefore costs approximately $500,000 to replace. This 30-second calculation wins budget battles. When a CFO questions retention investment, the response is immediate: "We lose six senior engineers per year at $210,000 average compensation. At 2.5× multiplier, annual attrition costs $3.15M. The retention investment costs $400,000. We need to prevent one departure to break even; we are losing six."

But the multiplier does not reveal where costs come from—which makes it difficult to identify which interventions would reduce them. For that, component-level analysis is required.

Where the Costs Hide

Attrition costs accumulate across six categories, each invisible to standard financial reporting:

Direct replacement costs are the most obvious: recruiting fees (20-25% of salary for external hires), sign-on bonuses, relocation packages. For a $210,000 engineer using an external recruiter with a $28,000 sign-on bonus, direct costs reach $75,000 before the new hire arrives.

Vacancy costs accumulate during the four to six months between departure and replacement. Work redistributed to remaining engineers creates context-switching overhead—typically 10-15% productivity loss across two to four team members. Work abandoned entirely (technical debt, infrastructure improvements, architectural planning) represents 30-40% of the departed engineer's capacity. A four-month vacancy easily costs $60,000 in lost productivity.

Onboarding drag extends six to twelve months after the replacement arrives. New senior engineers reach 25% productivity in month one, 50% by month three, 75% by month five. The productivity gap,

combined with senior engineer time spent onboarding, costs another $60,000 before the replacement reaches full effectiveness.

Tribal knowledge loss manifests as mistakes, slower decisions and deferred projects over the following year. Questions the departed engineer could answer in 30 seconds now require hours of code archaeology. Incidents that the departed engineer would have prevented occur because the replacement lacks context. A conservative estimate: $30,000 to $100,000 depending on documentation quality and system complexity.

Team disruption increases attrition risk for remaining engineers. When a respected senior engineer departs, colleagues notice and begin questioning whether they should also be looking. The probability-weighted cost of triggering additional departures: 20% of the sum of all other categories.

Opportunity cost compounds everything above. The departed engineer would have contributed architectural decisions, mentoring and code review that improved the entire organisation's output. These second-order effects are real but difficult to quantify.

The standard calculation totals these components for a specific departure. Using conservative assumptions for a $210,000 senior engineer (four-month time-to-hire, minimal tribal knowledge loss, 20% secondary attrition risk), the total reaches $260,000. Using realistic assumptions (six-month time-to-hire, moderate tribal knowledge loss, 30% secondary attrition risk), the total approaches $500,000. Comprehensive analysis of load-bearing engineers—staff and principal levels who own critical systems—routinely exceeds $800,000.

Appendix A provides detailed calculation worksheets with formulas for each component, worked examples at three complexity levels and guidance on customising estimates with company-specific data.

When Retention Investment Pays Off

With attrition cost quantified, the break-even calculation becomes straightforward. For a company with 40 senior engineers, 15% annual attrition (six departures) and $450,000 average attrition cost, current annual attrition expense is $2.7M.

If a retention intervention—technical debt sprints, deployment tooling, staff engineer career tracks—reduces attrition by 50%, it prevents three departures and saves $1.35M annually. Any intervention costing less than $1.35M pays for itself in year one.

Intervention effectiveness varies by implementation quality and organisational context. No retention programme prevents all departures; some engineers leave for reasons the company cannot address. But meaningful interventions that target actual causes of dissatisfaction typically reduce attrition by 30-60% in the first year. At the conservative end, preventing two departures at $450,000 each still justifies $900,000 in annual retention investment—more than most companies spend.

Why Executive Incentives Drive Attrition

Understanding the costs of attrition does not explain why organisations continue accepting them. The explanation lies in how executive incentives are structured. Consider the decision calculus facing a VP of Engineering in October, three months before annual reviews, six months before engineers' equity vests.

Her senior platform engineer wants to spend six weeks refactoring the authentication system. Technical debt is mounting. The current implementation has become brittle. Two security researchers have flagged concerns. But there are no active incidents. No customer

complaints. No revenue impact. Just engineers saying "this needs to be fixed before it becomes a crisis".

She has two options. Option A: approve the refactoring. Accept six weeks of reduced feature velocity. Risk missing quarterly OKRs. Explain to the CEO why the product roadmap is delayed for "technical work" customers will not see. Potentially jeopardise her Q4 bonus, which is tied to shipping three features the sales team has already committed to prospects. The benefit arrives in twelve months, maybe eighteen—retaining that senior engineer who will stay because the company took technical debt seriously.

Option B: prioritise the features. Acknowledge the technical debt as "important" and commit to addressing it "next quarter". Ship the roadmap. Hit OKRs. Collect the bonus. The senior engineer stays on for now because equity has not vested. If the authentication system eventually breaks, that is a future-quarter problem. If the engineer leaves in six months, recruiting can backfill the role.

Option B wins every time—until it does not. Until a critical system fails during a product launch. Until the organisation loses five senior engineers in eighteen months and the CFO starts asking why the company is spending $1.4M annually to rehire for preventable problems.

The fundamental misalignment: executive compensation optimises for quarterly results; engineer retention requires long-term investment. Information flow does not solve this. Economic restructuring does. The hidden costs remain invisible in ways that make rational actors behave irrationally. Direct replacement costs, vacancy costs, onboarding costs, and tribal knowledge loss—all calculated earlier—create a total cost per departure of $500,000 to $1M. But these costs are distributed and deferred. The decision to defer technical debt creates immediate, visible wins. Sales gets the demo they need.

Marketing gets the launch announcement. The CEO gets to tell the board that engineering shipped on schedule.
This creates what behavioural economists call normalisation of deviance. Each individual departure feels manageable, each instance of deferred technical work seems rational, each quarter's trade-offs appear justified in isolation. By the time the pattern becomes obvious—18% annual senior engineer attrition, mounting technical debt, cascading system failures—the organisation has normalised dysfunction.

When Interventions Will Not Work

These interventions will not work everywhere—and in some organisations the economics do not support attempting them at all. Three scenarios predict failure. First, high-churn business models operate under different economics. Consulting firms and contracting shops expect 20% to 40% annual turnover; their business models price in replacement costs and bill rates assume limited institutional knowledge. Retention interventions designed for product companies make no sense where client rotations drive natural departures and partner tracks deliberately create up-or-out pressure. Early-stage startups pre-product-market-fit may experience engineer departures that signal necessary pivots rather than retention failures.
Second, implementation theatre produces worse outcomes than inaction. A technical advisory board without genuine veto authority becomes a release valve that diffuses engineer concerns without changing decisions. Executive on-call rotations that do not connect incidents to root-cause decisions create performative empathy without accountability. Cost-of-attrition accounting that gets calculated but never appears in executive dashboards remains an academic exercise.

Half-implemented interventions signal that leadership wants the appearance of caring without the commitment to structural change. Third, these interventions assume a cultural prerequisite that many organisations lack: leadership must genuinely want behaviour change, not reputational management. The diagnostic test is simple: propose tying 25% of executive variable compensation to senior engineer retention. If leadership immediately identifies reasons this will not work, the answer is clear. They want solutions that do not cost them personally.

Companies unwilling to grant engineers veto authority, tie executive pay to retention or account for attrition costs in quarterly financial reviews are not ready for structural change. The interventions work when leadership recognises that $1.4M in annual attrition costs more than the interventions required to prevent it.

What Recovery Looks Like

The companies that do recover share a common characteristic: they treat a crisis as an inflection point rather than an anomaly to explain away.

The SaaS company that lost $1.4M to attrition never recovered. The fractional CTO diagnosed the problem; the executives nodded; nothing changed. That pattern is common. A different organisation's experience—a mid-sized payments processor that suffered a similar pattern but responded differently—reveals what structural change actually looks like.

Fourteen months before the Black Friday catastrophe, a senior platform engineer at that payments processor raised a specific concern: the transaction processing system could not handle projected holiday traffic. She submitted a detailed proposal for database sharding and

queue optimisation, estimated at six weeks of engineering time and $80,000 in infrastructure costs. The VP of Product deprioritised it. Two feature launches took precedence. The engineer's quarterly review praised her "proactive identification of potential issues" while her architectural proposal gathered dust in Jira.

She left four months later for a 15% raise at a competitor. Her replacement, hired after a three-month search consuming $47,000 in recruiting fees, took five months to reach productivity. By then, two more senior engineers had departed—one citing frustration with technical debt, another accepting a principal engineer role unavailable at the payments company. The database concern resurfaced in an architecture review nine months after the original warning. By then, the engineering team had lost institutional knowledge of the proposed solution. A junior engineer was assigned to "investigate options".

Black Friday arrived. Transaction volume spiked at 9:47 AM Eastern. The database began rejecting writes at 10:23 AM. By the time engineers identified the bottleneck—precisely the one flagged 14 months earlier—the system had failed to process $2.5M in transactions. The five-hour recovery effort involved emergency infrastructure scaling that cost $180,000 and permanent architectural changes requiring three engineers working through the holiday weekend at overtime rates.

The post-mortem, presented to the executive team on December 3rd, included a new section mandated by the CTO: "Previously Raised Concerns". It documented the original engineer's warning, the deprioritisation decision and the subsequent departures. The CFO calculated total costs.

Engineer attrition for three senior engineers: using the replacement cost framework, each departure cost approximately $235,000 in measurable expenses. $47,000 recruiting (external recruiter at 22% of $210,000 salary), $30,000 sign-on bonus, $83,000 vacancy cost

(four-month average time-to-hire), $75,000 ramp-up and onboarding. Three engineers: $705,000.

Tribal knowledge loss: $2.2M. The departed engineers possessed critical understanding of the database architecture, the specific failure modes that had been identified, and the proposed solutions. When the Black Friday incident occurred, the remaining team had to rediscover the problem, research solutions from scratch, and implement fixes under emergency conditions. This knowledge gap transformed what should have been a planned migration into a crisis response, multiplying costs across investigation time, failed approaches, emergency vendor engagement, and merchant remediation.

Failed transactions: $2.5M in payment processing volume that failed during the five-hour outage. The company's take rate was 2.9%, so direct revenue loss was $72,500, but the contractual obligation was to process all transactions. Failed processing triggered SLA penalties ($180,000 credited to affected merchants) plus merchant support costs ($45,000 in support team overtime and account management time to prevent churn).

Emergency infrastructure: $180,000 for emergency database scaling (additional read replicas, upgraded instances, expedited vendor support), load balancer reconfiguration and CDN optimisation to handle the traffic that had been projected 14 months earlier.

Recovery and remediation: $90,000 in overtime costs. Three senior engineers worked 72 hours over the holiday weekend at 2.5 times overtime rate ($200,000 annual divided by 2,080 hours, times 2.5 multiplier, times 72 hours, times three engineers, equals $51,923), plus two weeks of follow-up work for the broader engineering team to implement permanent fixes (five engineers, times two weeks, times $200,000 divided by 52 weeks, equals $38,462). Total: $90,385, rounded to $90,000 in the post-mortem.

Total incident cost: $3.47M. Cost to have implemented the original fix: $80,000 (six weeks of engineering time for one senior engineer plus infrastructure costs).

The ratio—$3.47M cost versus $80,000 prevention—appeared on the first page of the post-mortem. It was the number that changed the conversation.

The CEO, facing questions from the board about the incident, commissioned a retention analysis. It revealed that senior engineer attrition had reached 34% annually—more than double industry averages for profitable companies. Exit interviews, previously filed without executive review, showed a consistent pattern: talented engineers left when technical concerns were acknowledged but not acted upon.

Implementation of Six Interventions

The company implemented six interventions over eighteen months. Each met resistance; each produced measurable results.

The CEO presented the post-mortem and retention analysis at a four-hour off-site. The CFO objected to the $1.2M implementation cost. The CEO's response: the company had spent $3.47M on a preventable incident and $2.8M annually on preventable attrition. The question was not whether they could afford $1.2M but whether they could afford $6M every 18 months.

Intervention 1: Attrition cost tracking. The finance team built a monthly report showing total attrition cost alongside customer acquisition cost and MRR. The initial report: three senior departures costing $685,000—displayed on the executive dashboard directly below customer acquisition cost of $14,500. The juxtaposition was deliberate.

Intervention 2: Post-mortem accountability. A mandatory "Previously Raised Concerns" section required searching Jira, Slack

and architecture notes for prior warnings about any failure mode. The first post-mortem under the new template revealed a four-hour API outage caused by connection pool exhaustion—flagged three months earlier as two days of work. Outage cost: $45,000. Fix cost: $1,800. Ratio: 25-to-1. Over the following quarter, 73% of production incidents had been predicted by engineering concerns that were deprioritised.

Intervention 3: Executive on-call rotation. One non-engineering executive on-call per week, receiving alerts attributed to previously deprioritised technical debt. The VP of Product drew the first week. She received 11 pages—seven for the same connection pool issue. After the fifth, she messaged engineering: "What is it going to take to fix this permanently?" Two days of work. She approved it immediately. The experiential lesson changed prioritisation discussions in ways that engineering advocacy never had.

Intervention 4: Retention-linked compensation. Twenty percent of executive variable compensation tied to maintaining 90% retention of senior engineers, on a sliding scale. The compensation committee initially rejected the proposal, arguing executives could not control whether engineers received better offers. The VP of Engineering's response: "Engineers leave when their concerns are ignored and when technical debt makes their jobs miserable. These are things executives do control." Within two weeks of the policy taking effect, executives began asking "what is the attrition risk if we defer this?" during prioritisation discussions—before the incentive had paid out once.

Intervention 5: IC career track. Three levels with management-equivalent compensation: Staff Engineer ($210,000-$260,000, matching Engineering Manager), Principal Engineer ($270,000-$340,000, matching Director), Distinguished Engineer (matching VP). Promotion criteria focused on technical impact rather than team size. Nine engineers met Staff criteria; two met Principal. Average compensation increase: 18%. Three engineers

who had been interviewing for Principal roles at other companies withdrew from those processes.

Intervention 6: Protected technical debt allocation. Twenty percent of engineering capacity—one week per month—reserved for infrastructure work. Connection pooling fixed. Memory leaks fixed. False alert rate down 60%. Build times slashed from 25 minutes to 7. New engineer setup from three days to four hours. The CI/CD improvements alone recovered $230,000 annually.

The first fiscal year concluded with 91% retention (five departures out of 56 senior engineers), up from 66% the previous year. Four of the five departures were driven by factors the company could not address (relocation, startup founding, career scope, health). Attrition costs fell from $2.8M to $1.2M—a $1.6M reduction that exceeded the $1.2M implementation cost, producing positive ROI in year one.

Eighteen-Month Outcomes and Cultural Shifts

Eighteen months after the Black Friday incident, the company's retention metrics had stabilised at 9% annual attrition (91% retention). This rate placed them in the top quartile for companies at their stage and size. But the more significant changes were cultural and operational.

Architecture review process had been restructured. Technical debt proposals now included explicit risk assessments: "If we defer this work, what is the probability of production incident within six months, and what is the estimated cost?" Proposals that identified high-probability, high-cost risks were automatically prioritised. The "Previously Raised Concerns" section of post-mortems created accountability for deferral decisions. Executives who deprioritised technical work that later caused incidents had to explain those

decisions publicly. This created healthy caution around overriding technical concerns.

Technical judgment regained authority within the organisation. Engineers reported feeling that their concerns were taken seriously rather than being acknowledged and ignored. This changed behaviour in subtle but important ways. Engineers invested more effort in identifying and documenting risks because they believed the documentation would influence decisions. When risks materialised, the documented concerns created learning opportunities rather than blame cycles.

Project delivery improved despite allocating 20% of capacity to technical debt. This paradox—shipping more features while doing more infrastructure work—resulted from removing friction. When deployment takes 7 minutes instead of 25 minutes, engineers deploy more frequently. When local development setup takes 4 hours instead of 3 days, new engineers become productive faster. When false alerts drop 60%, engineers trust monitoring again and respond faster to real issues. The compounding effect of small improvements exceeded the direct cost of the time allocated to making them.

The original platform engineer who had raised the database concern? She returned as a Principal Engineer in month 16 at a salary of $295,000, representing a 40% increase from her departure compensation. The company recruited her specifically to lead infrastructure scaling and database architecture. Her return served two purposes: she brought expertise the company desperately needed, and her willingness to return signaled to remaining engineers that the cultural and structural problems had been genuinely addressed. The company that ignores technical concerns does not successfully recruit back the engineers who left because their concerns were ignored.

The economic calculus had fundamentally shifted. When the VP of Product proposed deprioritising infrastructure work in a Q3 planning

meeting, the CTO asked a single question: "What is the attrition risk if we defer this?" The question stopped the conversation. The VP of Product knew that every senior engineer departure reduced his bonus by roughly $4,000 (his 20% retention component of a $100,000 target bonus divided by 50 senior engineers, times the number of departures above the 90% retention threshold). Losing two engineers to preventable frustration would cost him $8,000 directly, plus the opportunity cost of projects delayed by the vacancies. He approved the infrastructure work.

The interventions worked because they aligned incentives. Executives cared about retention when their bonuses depended on it. Technical debt got fixed when executives experienced the consequences of ignoring it. IC career tracks retained ambitious engineers who would otherwise have left for Principal roles elsewhere. The changes were structural, not aspirational—which is why they persisted beyond the initial crisis that motivated them.

Information Is Cheaper Than Replacement

Departing engineers know what is broken. They know which technical decisions are failing, which processes waste time and which management practices create disengagement. This information exists within the organisation but does not reach executive level through normal channels because hierarchies filter information and middle management has incentive to suppress bad news.

The question is whether executives will extract what is broken from engineers before they leave, or absorb it from the problems those engineers would have prevented. The organisations that get this right treat information flow as a strategic priority. They create skip-level channels that bypass filters. They recognise that executive time spent

hearing ground truth has higher ROI than executive time spent on strategy based on filtered information. They understand that retention is cheaper than replacement, but only if problems are visible early enough to fix.

The organisations that get this wrong optimise for management comfort over information accuracy. They treat skip-levels as inappropriate, rely on reports systematically scrubbed of bad news and learn about retention problems when resignation letters arrive. They spend millions replacing engineers who could have been retained for thousands and they never quite understand why their best people keep leaving.

The calculation is straightforward. Allocating several hours per week for direct conversations with engineers costs roughly $50,000 annually in executive time. If this prevents one senior engineer departure, the intervention pays for itself five times over. In practice, it prevents multiple departures and provides information that improves resource allocation across the organisation. The alternative—learning about problems through resignation letters—is cheaper per hour and catastrophically more expensive per outcome.

The $40M SaaS company that lost him never implemented structural changes. They hired the fractional CTO, received the diagnosis, nodded through the recommendations and changed nothing. Annual attrition stabilised at 24%—not catastrophic enough to force a reckoning, but persistently expensive. They are still hiring fractional CTOs to diagnose problems their full-time leadership could see if it chose to look.

The senior engineer who left in 2018 is still in the industry. He now works at a company where the CTO maintains regular skip-levels and engineering holds veto authority on technical architecture. He is not interviewing. Neither are his colleagues. The company runs at 12% voluntary attrition, less than half the industry average for companies at

their stage. The difference between the two organisations is not talent strategy or compensation philosophy. It is that one structured its incentives so that executives bear the cost of ignoring technical judgment, and the other structured its incentives so that engineers bear the cost instead.

That asymmetry explains nearly everything. A senior engineer departure costs $500,000 to $1M. Prevention costs a fraction of that. Yet most organisations exposed to these numbers continue to absorb the higher expense, because the people who make resource allocation decisions do not personally pay when engineers leave. The VP who defers infrastructure work collects a bonus for shipping features on schedule; the $500,000 replacement cost scatters across budgets she never sees. The barrier to retention is not economic complexity. It is that the economics punish the wrong people.

Organisations prefer to frame attrition as a labour-market problem—"engineers are expensive and competitive"—because that framing demands no internal reckoning. The alternative explanation is harder: that decision-making processes systematically discount technical expertise, that information architectures prevent executives from learning about problems until they metastasise, and that the resulting departures represent a rational response by engineers who have concluded their judgment carries no weight. The question is not whether structural interventions work. The payments processor proved they do, at a 6.5-to-1 return. The question is whether leadership will restructure incentives before the next $3.47M incident, or whether they will continue paying $500,000 per resignation for the privilege of not knowing what their own engineers could have told them.

The cost of losing engineers is only half the equation. The other half—the cost of failing to hire the right ones in the first place—turns out to be even harder to measure and even more expensive to ignore.

The analysis continues.

Seven more chapters, each ending with a spreadsheet calculator for running the numbers on your own organisation.

codegood.co/book

PDF direct, or Kindle, paperback and hardcover on Amazon.