WHITE PAPER

MEASUREMENT · AUGUST 2026

ECONOMICS OF AGENT VALUE

A True Accounting of Tokenomics

Every enterprise AI program is justified by the same ratio: the value the agent produced, less what its tokens cost, divided by what its tokens cost. The formula looks like rigor. Read the cost structure underneath and it looks more like an instrument pointed at the smallest line on the invoice, one that quietly rewards you for producing less.

AUTHOR

Joshua Davis

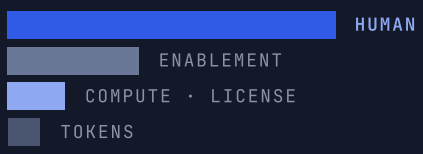
Lead AI GTM Strategist · GitHub

PUBLISHED

August 10, 2026

READING

~30 minutes



Soli Deo gloria

CONTENTS

– Abstract

01 The Formula on the Slide

02 Where the Standard Formula Breaks

03 What the Evidence Requires a Model to Explain

04 The Model

05 Worked Examples

06 Objections

07 Instrumentation

08 Conclusion

– About the Author

– Notes

ABSTRACT

Enterprise programs for AI coding agents are almost universally justified by a ratio: the value the agent produced, less what its tokens cost, divided by what its tokens cost. The formula is structurally correct. It is also the wrong instrument for the decision it is being asked to make, and the reasons why are neither novel nor controversial. They were settled decades ago in capital budgeting, in cost accounting, and in quality engineering, and they are being rediscovered the expensive way.

This paper makes three claims. First, that token cost is a minority of the cost of assisted delivery in nearly every organization, so a denominator built from tokens alone measures the wrong thing. Second, that a ratio with cost in the denominator is scale-blind, which means an organization that optimizes it is structurally rewarded for producing less. Third, that any accounting which treats rejected and reworked output as free will misprice agentic workflows, because rejection is where the cost actually lives.

The correction is a single structural move: put acceptance in the denominator. Measure the total cost of assisted delivery over a stated period, divided by the outcomes that were actually accepted and held. Report that unit cost alongside an absolute net contribution figure, never alone, and decompose the contribution into the part that came from working cheaper and the part that came from producing more. From those numbers falls a decision rule that tells an engineering organization when to spend more per attempt and when to stop.

01 The Formula on the Slide

Every enterprise AI program eventually produces a slide with a formula on it. The formula is almost always this one:

```
Agent ROI  =  ( Value of Agent Output  -  Token Cost )  ÷  Token Cost
```

It is worth being precise about what is right here before saying what is wrong. The shape is the textbook definition of return on investment: gain net of cost, expressed as a proportion of cost. Nobody should be embarrassed by it. It is the formula a CFO expects to see, it is arithmetically sound, and it has the considerable virtue of being computable from data a platform team already has.

The problem is not the algebra. The problem is that the formula imports three assumptions from the domain where it works, and none of the three survives the move into agentic software delivery.

It assumes the denominator captures the investment, which is a question for cost accounting. It assumes the numerator is measurable, which is a question for measurement theory. And it assumes a ratio is the correct basis for a ranking decision, which is a question for capital budgeting. Each of those disciplines has a named, well-documented failure mode for precisely the assumption being made. This paper's argument is that all three failure modes are active at once in AI-assisted delivery, which is why programs measured this way produce conclusions their own operators do not believe.

What follows is not a case against measuring return. It is a case for measuring it on the terms the work actually has.

02 Where the Standard Formula Breaks

2.1 The denominator is not the investment

Token spend is the visible cost of assisted delivery. It is rarely the largest one.

The best available distributional evidence puts this starkly. Analysis of AI spend against engineering payroll finds the top one percent of software companies spending roughly $89,000 per engineer per year on AI, against a fully loaded senior engineer cost near $224,000. Even at that leading edge, all AI spend combined is about 28 percent of the combined human-plus-AI cost, and the token portion of it is lower still, because that figure includes seats and licenses. At the median company the AI figure is $137 per engineer per year, roughly 650 times less than at the leading edge, at which point token spend is a rounding error against payroll.[1] These are modeled figures from an investor's analysis rather than an audited cost decomposition, and should be read for their distribution rather than their precision. The distribution is the point: for every organization outside the top percentile, AI spend of any kind is one to two orders of magnitude below the human cost it is supposed to be leveraging.

The costs the token line does not contain are not exotic. They are human review time, corrective cycles, CI and runner minutes consumed by failed and re-run pipelines, seat and license fees, and the engineering labor spent building the instructions, agent configurations, skills, and evaluation suites that make the agent useful in the first place. A commercial analysis of more than one million pull requests across 2,444 companies attempted to size the downstream portion, reporting that for every dollar spent generating code with AI, roughly forty-four cents is later spent fixing defects in it, twenty-seven cents rewriting it, and eleven cents absorbing review and merge delay: about eighty-two cents consumed before a feature reaches a user.[2] That split is one firm's model rather than an audited standard, and it reaches this paper through a trade-press account rather than a primary publication, so it should be treated as directional only. Two syntheses of published industry research by the present author point the same way from different directions: one puts license fees at roughly a quarter of the total cost of an enterprise AI deployment, with the remainder in integration, governance, training, and rework;[3] the other puts the subscription line at six to fifteen percent of the true annual cost per engineer once overages, review time, and accumulated technical debt are counted.[4] None of the three figures is an audited standard, and they are offered as corroborating shape rather than as measurement. The load in this section is carried by the distributional evidence in the preceding paragraph, which is independent of all three.

This is a known category of measurement error, and it has a literature. Cooper and Kaplan's foundational argument for activity-based costing is precisely that a system which traces direct cost accurately while allocating a large indirect pool by an unrelated driver will systematically distort the decisions made from it.[5] Their later treatment of resource capacity sharpens the point: the cost of resources *supplied* differs from the cost of resources *used*, and a system that fails to surface the difference silently absorbs it.[6] Gartner's own definitional work on total cost of ownership draws the same line, defining a direct cost as one traceable to a purchase order or budget line and an indirect cost as resources consumed elsewhere performing the same function. Their illustrative case is an employee outside IT spending time providing IT support to colleagues.[7] Substitute "engineer reviewing and repairing agent-generated output" and the analogy is exact. Ellram established the same principle as an academic discipline in purchasing: acquisition price is a systematically incomplete basis for a sourcing decision.[8]

A denominator of token cost is an acquisition price. It is the one number in the system that behaves like a SaaS invoice, which is exactly why it gets used, and exactly why it misleads.

2.2 The ratio penalizes the thing the program exists to do

The second failure is the more damaging, because it is invisible until someone optimizes against the number.

Consider a program that describes a maturation path: an AI-assisted engineer who uses the agent as a tool, and an AI engineer who directs agents as a workflow. Assume the assisted engineer produces 100 units of value on 10 units of token cost, and the AI engineer produces 300 on 50.

	VALUE	TOKEN COST	ROI	NET VALUE
AI-assisted engineer	100	10	900%	**90**
AI engineer	300	50	500%	250

The AI engineer delivers nearly three times the net value and reports a materially worse ROI. Anyone taking the ratio seriously and optimizing it should stay at the assisted level and decline to mature.

This is not an artifact of the numbers chosen. It is inherent to any efficiency ratio with cost in the denominator, and it is among the best-documented results in corporate finance. Brealey, Myers, Allen, and Edmans enumerate it among the named pitfalls of the internal rate of return rule under the heading "Mutually Exclusive Projects": the project with the higher rate need not have the higher net present value, because a rate is blind to the size of the investment that earns it.[9] Ross, Westerfield, and Jaffe give the same failure its two standard causes, the scale problem and the timing problem, and prescribe incremental analysis as the remedy.[10] Damodaran states it most plainly: net present value is denominated in dollars and does not factor in project scale, while the internal rate of return is a percentage standardized *for* scale, so rankings by the former favor large projects and rankings by the latter tilt toward projects requiring smaller investments.[11] Bierman and Smidt devote an entire chapter to it.[12]

> **THE CONCESSION THIS ARGUMENT REQUIRES**
>
> Every one of those authorities also identifies the condition under which a ratio *is* the correct ranking device: a binding constraint on the resource in the denominator. Under genuine capital rationing, maximizing value per unit of the scarce resource is right, and the profitability index exists for that purpose.[13]

That concession must be made carefully, because a constraint binds by policy, not by size. An AI budget capped by procurement or by a committed-spend agreement binds at whatever level it is set, even if it is trivial against payroll. Organizations do hit those caps: one large operator exhausted its full-year 2026 AI budget within the first few months.[14] The honest position is therefore not that token spend is never a binding constraint. It is that when a cap genuinely binds, the correct instrument is value per unit of *that specific capped resource*, computed against the cap, and applied only for the duration the cap is in force. A generic ROI ratio on token cost is not that instrument, and reporting one does not discharge the obligation to know which resource is actually scarce.

For most organizations, the resource that is scarce is engineering attention and review capacity, not tokens. The measured direction of travel supports that reading. Industry telemetry from a delivery-analytics vendor, covering roughly 22,000 developers over two years, reports defects per developer up 54 percent, incidents traced to individual pull requests up 242 percent, and pull requests merged without review up 31 percent.[15] The last of those three is the diagnostic one: review is being skipped, which is what a capacity constraint looks like before it is named. GitHub's platform telemetry shows more than one in five code reviews now involving an agent across more than sixty million reviews, a tenfold increase in under a year, which indicates that review volume has grown fast enough to require automating the review function itself.[16] Both sources are vendor-published and should be weighted accordingly.

Section 4.5 returns to this and specifies the constrained-ranking form for organizations that need it.

2.3 The numerator is not measurable as stated

"Value of agent output" has no defined unit and no stated measurement method. In a discipline whose premise is measurement rigor, an unfalsifiable numerator is the same anti-pattern as an unsubstantiated savings claim; whoever wants a favorable number can produce one.

This is not a hypothetical risk. It is the specific failure Goodhart described for monetary indicators: any observed statistical regularity tends to collapse once pressure is placed upon it for control purposes.[17] Strathern compressed the result into the form the management literature actually uses, that when a measure becomes a target it ceases to be a good measure.[18] Her surrounding argument is the more pointed one here. Audit is not a neutral observer; it has a life of its own that reshapes the activity it audits. An ROI target with an undefined numerator will be met by redefining the numerator.

The measurement problem is compounded by a documented perception gap. METR's randomized controlled trial of experienced open-source developers working in repositories they already maintained found that AI tooling *increased* task completion time by 19 percent, with a confidence interval of +2 to +39 percent, while the same developers had forecast a 24 percent speedup beforehand and still estimated a 20 percent speedup after experiencing the slowdown.[19] Two caveats belong with that figure. It describes early-2025, largely non-agentic tooling, and METR's own larger late-2025 replication produced wider, cohort-divergent results that the organization has publicly declared an unreliable signal, prompting a redesign of the experiment.[20] The durable finding is not the 19 percent. It is the roughly forty-point gap between perceived and measured effect, which is what disqualifies developer self-report as a value input.

A numerator sourced from self-reported time savings is therefore not merely soft. It is biased in a known direction, by a known magnitude, in favor of the tool. The consequences of that gap beyond measurement, for skill retention and for operational dependence on the tool, are a separate subject and are treated elsewhere.[21]

2.4 There is no period

Minor beside the others, and still fatal in practice: the formula states no time period. Cost accrues continuously, value arrives in lumps, and enablement investment is expensed long before the capability it buys shows up in output. A number without a period is not comparable to any other number, including its own prior value.

This matters more than it appears, because the omission interacts with a real economic phenomenon. Brynjolfsson, Rock, and Syverson's productivity J-curve describes exactly this shape: general purpose technologies require substantial complementary intangible investment, that investment is poorly captured in measurement, and the result is understated productivity in the early years followed by overstatement later as the intangibles are harvested.[22] Their earlier work on the AI productivity paradox concludes that implementation lags are the largest contributor to AI's absent productivity signal, because the full effects of a general purpose technology are not realized until waves of complementary innovation are built around it.[23] DORA's 2026 modeling of AI-assisted development returns includes an explicit J-curve for the same reason, attributing the early dip to learning curves, code verification overhead, and process adaptation; its authors characterize their own figures as a high-uncertainty estimate rather than a formula.[24]

An unbounded ROI ratio measured during the trough of a J-curve reports a program failure that is in fact a program on schedule.

03 What the Evidence Requires a Model to Explain

Before proposing a replacement, it is worth stating what the empirical record shows, because the correct model is the one that accounts for it without special pleading.

90%	−7.2%	~⅓	2.75×
AI usage at work, against 30% reporting little or no trust in its code	Delivery stability per 25% increase in AI adoption (DORA 2024)	Of substantive agent output not accepted; features at 66.1%	Total bill increase as token prices halved and consumption grew 450%

Adoption is near-total and trust is not. DORA's 2025 research, drawn from nearly 5,000 technology professionals, reports 90 percent AI usage at work, up from 76 percent the prior year, with a median of roughly two hours per day interacting with AI. More than 80 percent report increased productivity. And 30 percent report little or no trust in AI-generated code.[25] Stack Overflow's 2025 survey found trust actively inverted, with 46 percent distrusting accuracy against 33 percent trusting it, and identified the single largest developer frustration as "AI solutions that are almost right, but not quite," cited by 66 percent, with a further 45 percent reporting that debugging AI-generated code takes longer than writing it.[26]

Throughput and stability have decoupled. DORA's 2024 report found that a 25 percent increase in AI adoption was associated with a 1.5 percent decrease in delivery throughput and a 7.2 percent decrease in delivery stability, even as roughly three quarters of respondents reported feeling individually more productive.[27] The 2025 edition found the throughput relationship had reversed to positive while the stability relationship remained negative, with DORA's own explanation being that acceleration exposes downstream weakness: more change volume produces more instability.[25] The framing is worth borrowing. AI does not fix a delivery system; it amplifies what the system already was.

Roughly a third of substantive agent output is not accepted. The largest task-stratified study available, an unrefereed preprint covering 7,156 agent-authored pull requests, reports acceptance rates that vary far more by task type than by agent. Documentation tasks reach 82.1 percent acceptance and chores 84.0 percent, while features land at 66.1 percent, bug fixes at 66.0 percent, and performance work at 55.4 percent.[28] By agent the spread is narrower, from 61.6 to 77.9 percent. The economically important reading is the task-type cut: on the categories that constitute most engineering value, about one attempt in three is discarded, and the cost of discarding it is human review time.

Accepted is not the same as good. Work accepted to MSR 2026 finds agent-authored pull requests carry 1.87 times the redundancy of human ones, and that human authors delete substantially more code than agents do. The finding that should concern any measurement designer is the sentiment inversion: reviewers express *more* neutral or positive emotion toward AI contributions than human ones, which the authors read as surface-level plausibility masking redundancy and producing silent accumulation of technical debt.[29] Commercial, non-peer-reviewed code-analytics data covering the same adoption window points the same way, showing duplicated code blocks rising sharply and refactoring, measured as moved lines, collapsing to a small fraction of its pre-adoption share; that work establishes correlation with the adoption period rather than causation.[30]

This finding has a direct consequence for any model built on acceptance, and §4.2 addresses it head-on rather than deferring it to an objection.

Falling token prices have raised, not lowered, total spend. Token prices fell by more than 90 percent between 2023 and mid-2026, and spend on large language models roughly doubled in the same window; analysis covering December 2024 to December 2025 found token costs halved while token consumption grew 450 percent, a roughly 2.75-times increase in total bill.[31] Enterprise generative AI spend went from $1.7 billion in 2023 to $11.5 billion in 2024 to $37 billion in 2025, with the AI coding category alone rising from about $550 million to $4.0 billion year over year.[32] This is the Jevons paradox operating on inference, and its consequence for measurement is direct: unit cost, not total spend, is the only quantity an organization can control, because total spend responds to price cuts by growing.[33]

A model that cannot explain how a program is simultaneously faster, less stable, more expensive, and more trusted by its users than it deserves to be is not a model of this work. The one that follows is built to hold all four at once.

04 The Model

4.1 Notation

The notation divides into three groups: what is being measured, what it costs, and what gets reported.

GROUP 1 · SCOPE AND OUTCOMES

What is being counted, and over what window. These are the definitions that must be frozen before measurement begins.

SYMBOL	DEFINITION
`P`	The reporting period. Every figure carries one.
`O`	Outcomes attempted in P. A task, pull request, ticket, or loop run. The definition must be frozen.
`R`	First-pass acceptance rate. The fraction of attempts accepted without a corrective cycle and still standing at the end of a stated stabilization window.
`A`	Accepted outcomes in P, where `A = O × R`.

GROUP 2 · COST COMPONENTS

What the organization actually spends. Together these sum to the total cost of assisted delivery.

SYMBOL	DEFINITION
`C_tok`	Token and request consumption.
`C_hum`	Human time at loaded rate: prompting, reviewing, verifying, correcting.
`C_cmp`	Compute: CI/CD minutes, runners, actions, re-runs.
`C_lic`	Seat and license cost, apportioned to P.
`C_ena`	Enablement: instructions, skills, agent configurations, evaluation suites, amortized over expected life.
`TCAD`	Total cost of assisted delivery in P.

GROUP 3 · REPORTED MEASURES AND BASELINE

What the program publishes, plus the counterfactual the contribution figure depends on.

SYMBOL	DEFINITION
`U`	Unit cost per accepted outcome.
`N`	Net contribution in P.
`V`	Value delivered in P.
U_0	Pre-adoption baseline unit cost, on the same frozen outcome definition and the same cost scope.

4.2 Defining acceptance so it cannot be flattered

The model puts acceptance in the denominator, which makes `R` the most consequential definition in the framework and the most attractive one to manipulate. The evidence in §3 makes this concrete rather than hypothetical: reviewers respond more warmly to agent-authored code than its measured redundancy warrants.[29] A merge rate measured on lenient review is a biased instrument, and a model that divides by it inherits the bias.

Three definitional requirements follow, and they are not optional.

Acceptance requires a stabilization window. An outcome counts as accepted only if it is still standing after a stated period, typically two to four weeks. A change that is merged and then reverted, hotfixed, or substantially rewritten was not accepted; it was deferred. The window converts a point-in-time merge decision into a durability test, and durability is much harder to flatter than a merge button.

Corrective cycles disqualify first-pass acceptance. `R` is *first-pass* acceptance. An outcome that required a corrective round trip is not a first-pass acceptance even if it eventually merges. This is the definition that makes `R` behave like first-pass yield rather than eventual yield, and it is the whole reason `R` carries information about cost.

The definition is frozen before measurement and re-baselined only on a recorded change. Definition drift is the primary mechanism by which programs report improvement they did not achieve.

None of this eliminates reviewer leniency. It bounds it, and it makes the residual bias a stated limitation rather than a hidden one. An organization that cannot instrument a stabilization window should say so and treat its `U` as an optimistic bound.

4.3 Cost

$$TCAD = C_{tok} + C_{hum} + C_{cmp} + C_{lic} + C_{ena}$$

Rework is deliberately not a separate term. It is a decomposition of the terms already present: rework consumes human time, compute, and tokens, and appears in each of them. Adding a rework line would double-count it. The mechanism by which rework raises cost is handled structurally in the next equation rather than additively here, and that is the point of the design.

4.4 The core equation

$$U = TCAD \div A \qquad \text{where } A = O \times R$$

The total cost of assisted delivery, per accepted outcome, over a stated period.

The structural change is the presence of `R` in the denominator. Quality is no longer an argument made alongside the economics; it is inside them. A drop in acceptance raises unit cost mechanically, without anyone having to assert that poor quality is expensive. A workflow that produces twice as many attempts while its acceptance rate halves reports as more expensive per outcome, because it is.

This is not an invention. It is the software equivalent of a metric manufacturing settled on decades ago. First-pass yield and rolled throughput yield are standardized, certification-level process metrics in the Six Sigma body of knowledge,[34] and the idea they encode is the operative definition of cost of poor quality: all costs that would disappear if there were no deficiencies, no errors, no rework, no field failures.[35] A unit that had to be reworked was not produced at the cost of a unit that did not. Crosby's argument that quality is free rests entirely on booking the price of nonconformance as a real line of production expense rather than an unfortunate externality.[36]

Placing R in the denominator is how that idea is made arithmetic rather than rhetorical.

One property of U is worth stating early, because it becomes important in §5: U requires no baseline, no counterfactual, and no valuation of software. It is computed entirely from costs the organization actually paid and outcomes it actually kept. Of the two reported measures, it is the one that cannot be argued with.

4.5 The two reported measures

Efficiency is U, the cost per accepted outcome. It is a ratio, it is comparable across periods, and it should fall as a program matures. This is the administrator's number.

Contribution is net value:

$$N = V - TCAD$$

It is an absolute, denominated in currency. It rises with scale and capability. This is the engineer's number.

> **Report both. Never a ratio alone.**

The reason is specific and worth stating precisely rather than rhetorically. Under the baseline-relative value definition in §4.8, $N = A \times (U_0 - U_1)$. The two measures are therefore not independent: they always agree on *sign*, because N is positive exactly when $U_1 < U_0$. What they do not agree on is *direction of change*, because U is a rate and N scales with volume. A program that improves its unit cost while shrinking its accepted output reports better efficiency and worse contribution. A program that grows output while holding unit cost flat reports unchanged efficiency and improved contribution. Those are different programs, and a single number cannot distinguish them.

That is the whole argument for the pairing, and it is the direct application of the capital-budgeting result in §2.2. The ratio rewards restraint; the absolute rewards contribution. Reported alone, efficiency argues against growth and contribution argues against discipline. The precedent in software delivery measurement is *Accelerate*'s construction of the four key metrics, which deliberately pairs throughput measures with a stability measure so that speed cannot be purchased with quality.[37] The pairing here does the same work at the level of money.

The constrained form. Where a resource genuinely binds, as discussed in §2.2, add a third figure: value per unit of the constrained resource. Where the constraint is review capacity, that is $V \div C_{hum}$. Where it is a hard AI budget cap, it is $V \div (C_{tok} + C_{lic})$ computed against the cap. This is the profitability index applied correctly, to the resource that is actually rationed. Report it in addition to U and N, name the constraint it assumes, and remove it when the constraint lifts.

4.6 Compound acceptance in multi-stage workflows

Agentic workflows are rarely a single attempt. A typical loop plans, generates, tests, and submits for review, and each stage has its own acceptance behavior. Where stages are sequential and a failure at any stage discards the attempt, the effective acceptance rate is the product of the stage rates, exactly as rolled throughput yield is computed as the product of the yield at each step of a process:[38]

$$R_{rolled} = R_1 \times R_2 \times \ldots \times R_n$$

The consequence is not intuitive and it is why long agentic chains disappoint. Four sequential stages at 95 percent acceptance yield a rolled rate of 81 percent. Four stages at 90 percent yield 66 percent. A workflow whose every individual stage looks healthy can have an effective acceptance rate that makes its unit cost indefensible, and no

amount of token optimization will touch it.

This also gives a design rule with teeth: shortening a chain, or making a stage's failure recoverable rather than fatal, is an economic intervention of the same class as changing model tier, and is frequently the cheaper one.

4.7 The escalation rule

The most operationally useful result in the framework falls directly out of the core equation. Spending more per attempt is justified only when acceptance rises faster than cost:

$$U_2 < U_1 \iff C_2 \div C_1 < R_2 \div R_1$$

Here C is the **full cost per attempt**, not the price of a model tier. That distinction is the entire content of §5.2 and it is where most escalation decisions go wrong: an attempt whose token cost triples may have a full cost that rises by a tenth.

An attempt costing three times as much is worth it only if it raises first-pass acceptance by more than three times proportionally. Adding an evaluation suite, a test gate, or a second reviewer is justified by the same single test, which is what makes the rule useful: one criterion covers every intervention that trades cost for quality.

One caution, from §4.6. Where an added stage is *fatal* on failure rather than corrective, it does not simply raise C; it also multiplies R downward through the rolled rate. The escalation test still applies, but R_2 must be the rolled rate of the new configuration, not the acceptance rate of the added stage in isolation. Adding gates is not automatically a quality improvement.

The rule has a ceiling, and the ceiling is the part worth internalizing. Because $R \leq 1$, the maximum justifiable cost multiple is bounded by $1 \div R_1$. At a first-pass acceptance rate of 0.30, escalation up to 3.33 times the cost can be justified. At a mature 0.85, the ceiling is 1.18 times: almost nothing.

> **The room to escalate is widest exactly where acceptance is worst.** Immature workflows should be optimized for capability; mature ones for cost. That is a portfolio strategy, and it is derived rather than asserted.

The rule also resolves the ambiguity that ROI slides typically leave open when they annotate the formula with some version of "increasing this often means decreasing this." Two readings are usually available and they teach opposite lessons: either better quality costs more tokens, a tension to be managed, or better quality reduces token spend through less rework, a synergy to be exploited. Both are defensible. The honest answer is that both occur, and the escalation rule quantifies which one is operating: quality raises R, which lowers U, and often raises C, which raises U. The net direction is an empirical question with a computable answer. It should be a decision rule, not a slogan.

4.8 Defining value

V must be defined or the framework inherits the very flaw it corrects. Two forms are acceptable.

Baseline-relative. This form requires no absolute valuation of software:

$$V = A \times U_0$$

Gross value is what the same accepted output would have cost at the pre-adoption unit cost. Because U_1 is defined as $TCAD \div A$, it follows that $TCAD = A \times U_1$, and contribution reduces to:

$$N = V - TCAD = A \times (U_0 - U_1)$$

That reduction is a restatement of the definitions, not a discovery, and it makes two things explicit that a reader is entitled to know.

First: N is a cost-avoidance figure, not cash. It states what the same accepted output would have cost at the old unit rate, less what it did cost. On fixed headcount it does not appear in any budget line and does not become cash unless spend, headcount, or scope actually changes. It should be labeled as cost avoidance in any document that reaches finance. Reporting it as realized savings is the same offense as an unfalsifiable numerator, committed in the other direction.

Second: N carries a volume effect that must be shown separately. Because V is linear in A, every additional accepted outcome is valued at the full counterfactual rate U_0. A change in N between two periods therefore decomposes into three parts:

$$\begin{aligned} \Delta N &= (A_2 - A_1) \times (U_0 - U_1) && \text{rate held, volume grows} \\ &+ A_1 \times (U_1 - U_2) && \text{volume held, rate improves} \\ &+ (A_2 - A_1) \times (U_1 - U_2) && \text{interaction} \end{aligned}$$

Report the decomposition, not just the total. A program whose contribution growth is almost entirely volume has not become more efficient; it has become larger, and larger is only valuable if the additional outcomes were wanted. Where an engineering organization is demand-constrained rather than capacity-constrained, the volume term should be discounted or excluded outright.

U_0 is the fragile term, and it must be scoped. Two rules govern it. The cost scope of U_0 must match the cost scope of U_1 exactly: if U_1 counts only assisted-delivery cost, U_0 must be computed on the same basis and not on fully loaded team cost, or contribution is inflated by whatever the scopes differ by. And U_0 must be re-derived, not merely frozen, whenever the underlying labor market or toolchain shifts materially. A counterfactual frozen indefinitely grows more flattering every year through wage inflation alone.

Directional. Where an organization already prices time and defects:

$$V \approx (\text{Hours Displaced} \times \text{Loaded Rate}) + \text{Cycle-Time Value} + \text{Defect-Avoidance Value}$$

The cycle-time term has a legitimate basis. Reinertsen's central methodological claim is that product development decisions require a quantified economic framework, and that if an organization quantifies only one thing, it should quantify the cost of delay.[39] The term is admissible where an organization has already done that work. Where it has not, inventing a number for this paper's sake is precisely the failure of §2.3.

Directional is acceptable if labeled directional. Undefined is not. And self-reported hours saved is not an acceptable input to either form, for the reasons established in §2.3.

4.9 Time rules

RULE	REQUIREMENT
Period	Every U and N carries a stated P. No period, no number.
Minimum window	At least one month for unit cost. At least two quarters for any maturation claim.
Stabilization	R measured against a stated durability window, typically two to four weeks, per §4.2.
Amortization	C_ena over the configuration's expected life, default two quarters. C_lic apportioned to P.
Matching	Accrual, not cash. A workflow built in P1 that delivers in P2 has its enablement amortized across both.
J-curve	Expect U to rise for the first four to eight weeks of a maturation step, as C_ena lands before R improves. A single-month measurement will show a loss.
Baseline	Freeze O and R definitions before adoption. Match U_0 scope to U_1 scope. Re-baseline on definition change or material market shift, and record the change.

The J-curve rule is the one most often violated and the most expensive to violate, because it terminates programs during their expected trough. It is the operational form of the productivity J-curve result: complementary intangible investment is expensed in the period it is incurred, while the output it enables arrives later.[22] Any measurement window shorter than the lag will report a correctly executing program as a failed one.

4.10 Per-consumer forms

The model applies unchanged across maturity levels, but the dominant cost term and the dominant lever move.

Assist. The engineer uses the agent as a tool. O is tasks or pull requests. C_hum dominates, typically 70 to 85 percent of TCAD. The consequence deserves to be stated to customers explicitly: token optimization at this level moves a small share of total cost. A program promising material savings from token efficiency alone, at this maturity, is over-promising by construction.

Engineer. The engineer directs agents through defined workflows. O is workflows completed. C_ena becomes significant and is amortized across runs, which is what makes workflow reuse an economic lever rather than an engineering preference. A configuration used by one team is an expense; the same configuration used by twelve is a fixed cost divided twelve ways.

Agentic. Loops run on schedule or on trigger, with human involvement reduced to triage and approval. C_hum per attempt falls sharply. It does not vanish, and it is frequently still the largest single term: the loop in §5.2 spends 61 percent of its cost on three minutes of triage per run. Substituting A = O × R:

$$U_{agentic} = \text{Cost Per Run} \div R$$

Here acceptance rate is the dominant lever, decisively ahead of token efficiency, and the reason is visible in that same 61 percent. Every rejected run consumed triage time as well as tokens. As R approaches zero, unit cost approaches infinity regardless of how cheap each run is. Cheap loops with poor acceptance are the worst position in the entire model, and they are invisible in seat-based reporting, because a seat-based report shows a fixed cost and no output column at all.

05 Worked Examples

5.1 Maturation across three quarters

A team of ten. Outcome is a merged pull request, frozen, measured against a four-week stabilization window. Loaded rate $100 per hour. Enablement in P2 and P3 is a $60,000 workflow build amortized across those two quarters; the $2,000 in P1 is the residual amortization of pre-existing prompt and instruction assets.

The baseline U_0 is $1,400 per merged pull request, derived pre-adoption on the same cost scope. It is an input, not a result, and §5.1.1 shows how much of the reported contribution depends on it.

	P1 · ASSIST	P2 · TRANSITION	P3 · ENGINEER
C_tok	8,000	20,000	30,000
C_hum	100,000	95,000	80,000
C_cmp	4,000	8,000	12,000
C_lic	6,000	6,000	6,000
C_ena	2,000	30,000	30,000
TCAD	**120,000**	**159,000**	**158,000**
O	150	180	240
R	0.80	0.75	0.85
A	120	135	204
V (= A × 1,400)	168,000	189,000	285,600
U	**$1,000**	**$1,178**	**$775**
N	**$48,000**	**$30,000**	**$127,600**

Four observations.

Human time dominates. C_hum is 83 percent of TCAD at the assist level and tokens are 7 percent. Any claim that token optimization is the primary FinOps lever for an assist-level program is arithmetically unavailable.

The J-curve is visible in P2. Unit cost rises and net contribution falls, because enablement lands before acceptance improves. Judged on that quarter alone, maturation looks like a failure. This is why the time rules require at least two quarters for a maturation claim.

The example assumes human effort per attempt falls. Implied review-and-correction time per attempted change is 6.67 hours in P1 and 3.33 hours in P3, a halving. That is the single most consequential assumption in the table and it is stated here rather than buried, because §3's evidence points the other way for organizations that adopt without process change. The assumption is defensible only for a team that has actually built the workflows, gates, and evaluation suites the C_ena line pays for. A team that spends the enablement money and does not get the time reduction has a very different table.

The acceptance rates are above the published population figures. R of 0.80 to 0.85 exceeds the 66 percent feature and bug-fix acceptance reported for agent-authored pull requests at population scale.[28] The example measures acceptance across all attempted changes including routine ones, where published rates run above 80 percent, and it describes a team operating with deliberate workflow design. It is illustrative, not typical.

The comparison. Now the same data under the original formula, with value measured against token cost alone:

	ROI, TOKEN-ONLY	U	N
P1 · Assist	2,000%	$1,000	$48,000
P2 · Transition	845%	$1,178	$30,000
P3 · Engineer	852%	$775	$127,600

The token-only formula reports maturation as a collapse from 2,000 percent to 852 percent, and reports P2 and P3 as effectively identical despite a 23 percent difference in unit cost and a four-fold difference in contribution. The corrected model reports unit cost down 22.5 percent, accepted output up 70 percent, and net contribution up 166 percent from P1 to P3, with the P2 trough visible in both U and N exactly as the J-curve rule predicts. Same team, same quarters, opposite conclusion, and only one of the two reports anything the business can act on.

5.1.1 What the contribution figure actually contains

Applying the decomposition from §4.8 to the change in N from P1 to P3, a total of $79,600:

COMPONENT	AMOUNT	SHARE
Volume: 84 additional accepted outcomes at the P1 margin	$33,600	**42%**
Rate: P1 volume at the improved unit cost	$27,059	**34%**
Interaction	$18,941	**24%**
Total ΔN	**$79,600**	**100%**

Roughly two thirds of the contribution growth traces to producing more, and that portion is only worth what the organization would have paid for those 84 additional merged changes. A demand-constrained team should discount it heavily. A team with a backlog it cannot clear should not.

This is also why U is reported first. Unit cost is invariant to U_0 and to volume; it is computed from money actually spent and outcomes actually kept. Contribution is the more persuasive number and the more fragile one.

Sensitivity to the baseline. Because $N = A \times (U_0 - U_1)$, the headline contribution growth is highly sensitive to a baseline that is asserted rather than measured:

U_0	N AT P1	N AT P3	REPORTED GROWTH
$1,100	$12,000	$66,400	+453%
$1,400	**$48,000**	**$127,600**	**+166%**
$1,600	$72,000	$168,400	+134%

The unit cost figures in the table above do not move at all across that range. Any organization publishing a contribution number owes its reader the baseline it used and the method by which it was derived.

5.2 Escalation on an agentic loop

A scheduled dependency-triage loop. Four hundred runs per month. Triage costs three minutes per run at $100 per hour. Build cost $6,000 amortized over six months. No incremental seats. Before automation, the same triage work was performed manually at $45 per correctly actioned finding, which is the U_0 for this workflow. The question is whether to move the loop from a lower model tier to a higher one.

	LOOP A · LOWER TIER	LOOP B · HIGHER TIER
C_tok	200	600
C_hum (triage)	2,000	2,000
C_cmp	100	100
C_lic	0	0
C_ena	1,000	1,000
TCAD	**3,300**	**3,700**
Cost per attempt	$8.25	$9.25
R	0.30	0.45
A	120	180
V (= A × 45)	5,400	8,100
U	**$27.50**	**$20.56**
N	**$2,100**	**$4,400**

Both measures agree, and they agree for different reasons, which is the point of reporting both. Unit cost falls 25 percent because acceptance improved faster than cost. Contribution more than doubles because the same improvement also produced 60 more accepted outcomes. Neither number alone would have shown both effects.

Applying the escalation rule correctly, on full per-attempt cost: $C_B \div C_A = 1.12$ against $R_B \div R_A = 1.50$. The test passes with room to spare.

Applying it to tokens alone, as the original formula would have it: token cost triples, which would require acceptance to rise above 0.90 to justify the change. The upgrade is rejected—on the same data that shows it working.

The ceiling check confirms the headroom. At $R_A = 0.30$ the maximum justifiable multiple is $1 \div 0.30 = 3.33$ times, so a 1.12 multiple is nowhere near the limit. Contrast a mature workflow already running at $R = 0.85$, where the ceiling is 1.18 times and almost no escalation can be justified.

Finally, note the shape of Loop A's cost. Tokens are 6 percent of it and human triage is 61 percent. Optimizing tokens is close to pointless while an acceptance rate of 0.30 is discarding 70 percent of every run, and every discarded run consumed triage time as well as tokens. Acceptance is the lever. Token price is not.

06 Objections

"This is just net present value with extra steps."

In spirit, yes, and that is the argument. The claim is not that a new financial instrument is required but that the existing one is being misapplied, and that the correction is the one finance already prescribes for the mutually-exclusive-projects pitfall: evaluate the absolute, and use a ratio only where a genuinely binding constraint makes value-per-unit-of-constraint the decision variable.[9][12] Section 4.5 provides that constrained form explicitly.

"Acceptance rate is gameable."

It is, and §4.2 is the answer rather than a footnote. The stabilization window is the load-bearing mitigation: an organization can lower its review bar and inflate merge rate, but it cannot easily prevent the resulting changes from being reverted, hotfixed, or rewritten inside four weeks, and the definition of `R` counts those against it. Frozen definitions and recorded re-baselining handle drift. Reporting `N` alongside `U` prevents improvement by shrinkage. What none of this fully eliminates is reviewer leniency at the moment of review, and the paper states that as a residual limitation rather than claiming a cure.

"Human time is hard to attribute."

It is harder than reading a token invoice and considerably easier than the alternatives organizations already accept. Review time, corrective cycles, and triage are visible in the same systems that produce the outcome count. The activity-based costing literature's answer applies without modification: the difficulty of tracing an indirect cost is not an argument for excluding it, because excluding it does not make the cost stop being incurred—it makes the decision made from the number wrong.[5]

"Every claim here says AI costs more than advertised."

It does not. The model is symmetric and reports whichever answer the data supports. Example 5.2 is precisely a case where the narrow formula rejects an escalation the correct one approves. Industry telemetry across 275,000 engineers reports pull request throughput at 1.7 times for the highest-adoption cohorts, and revert-rate increases at those tiers that are real but small; that source is vendor-published.[40] A model that only ever produces bad news is not a model. It is a position.

"The ninety-five percent failure statistic proves the whole category is unproven."

That figure, commonly rendered as 95 percent of enterprise AI pilots producing no return, should be handled carefully by anyone who wants to be believed. It comes from a non-peer-reviewed report by an MIT Media Lab research initiative, its actual wording is that "95% of organizations are getting zero return," and Kevin Werbach of the Wharton School has stated publicly that he has read the document repeatedly and cannot understand where the claim comes from.[41][42] It is cited here only to note that it is contested. A framework that needs a disputed statistic to make its case does not have a case.

07 Instrumentation

The model imposes five measurement obligations, and none requires a new tooling category.

01 **Freeze the outcome definition.**

Decide what an outcome is: a merged pull request, a closed ticket, a completed workflow run. Write it down before measuring. Definition drift is the single most common source of non-comparable period-over-period figures.

02 **Instrument acceptance with a durability window.**

`R` is the fraction of attempts accepted without a corrective cycle and still standing after a stated window. Both halves are derivable from systems the organization already runs: merge and close outcomes and review-cycle counts give the first, and reverts, hotfixes, and rewrite commits against the original change give the second. This is not a survey question, and it is deliberately not merge rate alone.

03 **Attribute human time by sampling, not by timesheet.**

A periodic sample of review and correction time against a known outcome population is sufficient to establish the loaded-cost coefficient, and is more reliable than continuous self-report, which the perception-gap evidence disqualifies for this purpose in any case.[19]

04 **Amortize enablement on a register.**

Every configuration, skill, evaluation suite, and workflow build gets a cost and an expected life. Without this, C_ena either disappears entirely or lands as a single-quarter shock that manufactures a false J-curve.

05 **Derive and document U_0 once, on the matching scope.**

Record how the pre-adoption unit cost was computed, on which cost scope, over which period, and revisit it when the labor market or toolchain shifts. Every contribution figure the program will ever publish rests on it.

Everything else in the model is arithmetic performed on those inputs plus invoices the organization already receives.

These five obligations govern *measurement*. They do not constitute a spend-governance program. Workload classification, budget hierarchy and caps, real-time consumption instrumentation, and the standard optimization levers of caching, model routing, and batch processing are a separate discipline, and one this framework assumes is already in place.[33]

08 Conclusion

The standard formula is not wrong so much as too narrow to survive contact with an enterprise. It measures one cost term against an undefined benefit, over no stated period, using a ratio that rewards restraint over contribution. Its implicit promise, that optimizing token consumption is the path to better returns, is contradicted by its own cost structure in nearly every organization that runs it, and by a market in which per-token prices have fallen more than 90 percent while total spend has multiplied.[1][31]

Corrected, the model says something simpler and considerably more useful: the total cost of assisted delivery, divided by the outcomes actually accepted and held, over a stated period.

Putting acceptance in the denominator is the structural change, and it is the one that does the work. Quality stops being an argument made alongside the economics and becomes part of them. A workflow that produces twice as many attempts while its acceptance rate halves shows up as more expensive per outcome, because it is. And when the workflow runs in stages, the compounding is multiplicative rather than additive, which is why chains that look healthy stage by stage can be indefensible end to end.

Two numbers are reported because one alone misleads. Efficiency falls with maturity and is the administrator's number; it needs no baseline and cannot be argued with. Contribution rises with capability and is the engineer's number; it is the more persuasive figure and the more fragile one, it is cost avoidance rather than cash, and it must be decomposed into what came from working cheaper and what came from producing more. Reported alone, efficiency argues against growth and contribution argues against discipline. Together they describe the same delivery from the two positions that must agree on it.

And they carry a decision rule. Spend more per attempt only when acceptance rises faster than cost, measured against the full cost of the attempt rather than the token line. The rule has a ceiling worth knowing: the room to escalate is widest where acceptance is worst. Optimize immature workflows for capability and mature ones for cost.

None of this is a novel financial instrument. It is capital budgeting's treatment of scale, cost accounting's treatment of indirect cost, and quality engineering's treatment of first-pass yield, applied to a category of work that has so far been measured as though none of that history existed. The organizations that adopt it will not discover that AI agents are worth less than claimed. They will discover which of their workflows are worth more.

ABOUT THE AUTHOR

Joshua Davis is a Lead AI GTM Strategist for Americas—Enterprise at GitHub. Across four decades, he has built software, architected enterprise cloud, and led teams of all sizes. Today, he helps organizations do far more than sell AI: he helps them implement it, manage it, and govern it at enterprise scale. He advises leaders who need AI strategy, adoption, and governance to move as one.

To contact or request advisory options: jdav.is/advisory

NOTES

42 sources · peer-reviewed research, industry telemetry, company disclosures, and reporting, 1979–2026.

1 Tomasz Tunguz, "When AI Costs More Than the Engineer," Theory Ventures, June 29, 2026. Figures modeled from Goldman Sachs, the Ramp AI Index, Levels.fyi, KeyBanc, and Epoch AI; presented by the author as directional analysis rather than a measured cost decomposition. The per-engineer figures are total AI spend, inclusive of seats and licenses, not token spend alone.

2 "For Every $1 Spent on AI, Companies Pay $0.44 Fixing Bugs, $0.27 Rewriting Code, and $0.11 on Review Delays," *Tech Startups*, July 6, 2026, reporting Entelligence AI Research analysis of 2,444 companies and more than one million pull requests. Discussed at length, with the caveat reproduced above, in Joshua Davis, "The Second Invoice: What AI-Generated Code Costs After the Meter Stops," jdav.is, July 16, 2026.

3 Joshua Davis, "Scaling AI in the Enterprise: What the Evidence Says Works," jdav.is, July 1, 2026. A synthesis of published industry research by the present author, not independent survey work.

4 Joshua Davis, "The Developer Premium: What Eighteen Months of AI Replacement Attempts Actually Cost," jdav.is, June 29, 2026. A synthesis of published industry research by the present author.

5 Robin Cooper and Robert S. Kaplan, "Measure Costs Right: Make the Right Decisions," *Harvard Business Review* 66, no. 5 (September–October 1988): 96–103.

6 Robert S. Kaplan and Robin Cooper, *Cost and Effect: Using Integrated Cost Systems to Drive Profitability and Performance* (Boston: Harvard Business School Press, 1998), chap. 7.

7 Lars Mieritz and Bill Kirwin, *Defining Gartner Total Cost of Ownership*, Gartner Research G00131837 (Stamford, CT: Gartner, December 8, 2005).

8 Lisa M. Ellram, "Total Cost of Ownership: An Analysis Approach for Purchasing," *International Journal of Physical Distribution & Logistics Management* 25, no. 8 (1995): 4–23.

9 Richard A. Brealey, Stewart C. Myers, Franklin Allen, and Alex Edmans, *Principles of Corporate Finance*, 14th ed. (New York: McGraw Hill, 2023), chap. 5, sec. 5-3, "Pitfall 3—Mutually Exclusive Projects."

10 Stephen A. Ross, Randolph W. Westerfield, and Jeffrey Jaffe, *Corporate Finance*, 10th ed. (New York: McGraw-Hill/Irwin, 2013), 149–54.

11 Aswath Damodaran, *Applied Corporate Finance*, 3rd ed. (Hoboken, NJ: John Wiley & Sons, 2010), chap. 6, "Project Rankings—NPV and IRR."

12 Harold Bierman Jr. and Seymour Smidt, *The Capital Budgeting Decision: Economic Analysis of Investment Projects*, 9th ed. (New York: Routledge, 2007), chap. 4.

13 Brealey et al., *Principles of Corporate Finance*, sec. 5-4, "Choosing Projects When Resources Are Limited"; Bierman and Smidt, *Capital Budgeting Decision*, chap. 6.

14 Sasha Rogelberg, "After Blowing through Its Entire 2026 AI Budget in Months, Uber CTO Says, 'We're Coming to the End of the So-Called Tokenmaxxing Era,'" *Fortune*, August 7, 2026. No dollar figures are disclosed in the account.

15 Neely Dunlap, "The Gap between AI Spend and Engineering Outcomes," Faros AI, June 18, 2026. Telemetry across approximately 22,000 developers and 4,000 teams over two years; vendor-published.

16 Andrea Griffiths, "Agent Pull Requests Are Everywhere. Here's How to Review Them," The GitHub Blog, May 7, 2026. First-party platform telemetry from a vendor with an interest in the trend it reports.

17 C. A. E. Goodhart, "Problems of Monetary Management: The UK Experience," in *Monetary Theory and Practice: The UK Experience* (London: Macmillan, 1984), 96.

18 Marilyn Strathern, "'Improving Ratings': Audit in the British University System," *European Review* 5, no. 3 (1997): 305–21. The compressed formulation is Strathern's gloss on Goodhart, not Goodhart's own sentence.

19 Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," arXiv:2507.09089 [cs.SE], July 2025. Sixteen developers, 246 tasks, in repositories they already maintained.

20 Joel Becker, Nate Rush, Tom Cunningham, David Rein, and Khalid Mahamud, "We Are Changing Our Developer Productivity Experiment Design," METR, February 24, 2026.

21 Joshua Davis, "The Atrophy Risk: What Happens to Developers When AI Goes Dark," jdav.is, July 6, 2026. Reviews the METR result alongside a controlled trial on comprehension effects and the operational consequences of tool dependence.

22 Erik Brynjolfsson, Daniel Rock, and Chad Syverson, "The Productivity J-Curve: How Intangibles Complement General Purpose Technologies," *American Economic Journal: Macroeconomics* 13, no. 1 (2021): 333–72.

23 Erik Brynjolfsson, Daniel Rock, and Chad Syverson, "Artificial Intelligence and the Modern Productivity Paradox: A Clash of Expectations and Statistics," NBER Working Paper 24001 (Cambridge, MA: National Bureau of Economic Research, November 2017).

24 DORA and delta innovation practice, *The ROI of AI-Assisted Software Development (2026.01)* (Google Cloud, May 11, 2026). The report's authors characterize their figures as a high-uncertainty estimate intended to start a conversation rather than a rigid formula.

25 DORA, *State of AI-Assisted Software Development 2025* (Google Cloud, September 23, 2025). Survey of nearly 5,000 technology professionals, fielded June–July 2025.

26 Stack Overflow, "AI," in *2025 Developer Survey* (Stack Overflow, July 2025).

27 DORA, *Accelerate State of DevOps Report 2024* (Google Cloud, 2024). The 2025 edition reports standardized effect sizes rather than comparable percentages; the 1.5 and 7.2 percent figures belong to the 2024 edition only.

28 Giovanni Pinna, Jingzhi Gong, David Williams, and Federica Sarro, "Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance," arXiv:2602.08915v2 [cs.SE], May 7, 2026. Dataset of 7,156 pull requests. Preprint; not peer-reviewed at time of writing.

29 Haoming Huang, Pongchai Jaisri, Shota Shimizu, Lingfeng Chen, Sota Nakashima, and Gema Rodríguez-Pérez, "More Code, Less Reuse: Investigating Code Quality and Reviewer Sentiment towards AI-Generated Pull Requests," arXiv:2601.21276 [cs.SE], January 29, 2026, accepted to Mining Software Repositories 2026.

30 William Harding, *AI Copilot Code Quality: Evaluating 2024's Increased Defect Rate via Code Quality Metrics*, version 2025.2.5 (GitClear, February 2025); GitClear, *The Maintainability Gap: AI Code Quality in 2026* (GitClear, January 2026). Both are commercial, non-peer-reviewed analyses using proprietary metric constructs.

31 Sasha Rogelberg, "Tokens Are Getting Cheaper, but Companies Are Spending Even More on AI as a Result, Top Economist Warns," *Fortune*, June 17, 2026, quoting Torsten Slok of Apollo and reporting Bain & Company analysis.

32 Tim Tully, Joff Redfern, Deedy Das, and Derek Xiao, "2025: The State of Generative AI in the Enterprise," Menlo Ventures, December 9, 2025.

33 Joshua Davis, "The Token Is the Unit: A Financial Leader's Guide to Enterprise AI Economics," jdav.is, June 15, 2026.

34 American Society for Quality, *Certified Six Sigma Black Belt (CSSBB) Body of Knowledge* (Milwaukee, WI: American Society for Quality, 2022), sec. V.F.6.

35 Frank M. Gryna, "Quality and Costs," in *Juran's Quality Handbook*, 5th ed., ed. Joseph M. Juran and A. Blanton Godfrey (New York: McGraw-Hill, 1999), sec. 8.

36 Philip B. Crosby, *Quality Is Free: The Art of Making Quality Certain* (New York: McGraw-Hill, 1979), chap. "Cost of Quality."

37 Nicole Forsgren, Jez Humble, and Gene Kim, *Accelerate: The Science of Lean Software and DevOps; Building and Scaling High Performing Technology Organizations* (Portland, OR: IT Revolution Press, 2018), chap. 2.

38 Spencer Graves, "Six Sigma Rolled Throughput Yield," *Quality Engineering* 14, no. 2 (2001): 257–66.

39 Donald G. Reinertsen, *The Principles of Product Development Flow: Second Generation Lean Product Development* (Redondo Beach, CA: Celeritas Publishing, 2009), principle E3.

40 Jellyfish, "AI Engineering Trends," Jellyfish, July 2026. Ongoing benchmark across more than 1,000 companies, 275,000 engineers, and 90 million pull requests; vendor-published.

41 Aditya Challapally, Chris Pease, Ramesh Raskar, and Pradyumna Chari, *The GenAI Divide: State of AI in Business 2025* (MIT NANDA, July 2025).

42 R. Scott Raynovich, "Why We Don't Believe MIT NANDA's Weird AI Study," Futuriom, August 26, 2025, quoting Kevin Werbach of the Wharton School.

