PART I · THE ARGUMENT

The Category Error

OPENING CASE

Two subscriptions, one correction

DATED: JUNE 2025 TO JUNE 2026. VENDOR ACTIONS, DATES, AND FIGURES ARE SOURCED. THE BUYING ORGANIZATION IS A COMPOSITE.

Twenty dollars per developer, per month. In the spring of 2025, that was the price of Cursor, an AI coding assistant. A software team bought one Cursor license for each developer who would use it, just as it did with every other software tool. The expense went into the budget beside the company's ticketing system, design software, and password manager.

Cursor was useful, and it was cheap. Once the agreement was signed, nobody in finance had a reason to examine the line item again. Nothing about a twenty-dollar software seat demanded attention. But beneath every seat, a meter was running.

On June 16, 2025, the twenty dollars stopped being a price and became a balance. Cursor replaced the Pro plan's monthly allowance of five hundred requests to external models with a twenty-dollar credit for frontier-model usage. Each request now drew down that credit at the model provider's API rate. The subscription still cost twenty dollars a month, but what that price bought had changed completely.

Light users barely noticed the change because their activity remained within the included credit. Heavy users had a different experience. A handful of prompts could exhaust the credit, after which additional usage was billed at the underlying API rates. The change therefore affected the customers who relied on the product most and often received the most value from it. Many faced charges they had not anticipated or budgeted for.

On July 4, less than three weeks after the change, Michael Truell apologized and promised refunds for charges incurred during the transition. Truell is chief executive of Anysphere, the company behind Cursor. His explanation reduced the dispute to arithmetic. The newest models consumed far more tokens on long-horizon tasks than a flat monthly price could cover.[1]

On June 18, 2025, two days after Cursor changed its plan, GitHub began enforcing monthly premium-request allowances for Copilot and letting customers pay for usage beyond them. It then spent the next year preparing a larger change.

On June 1, 2026, GitHub stopped measuring standard usage in requests and began measuring it in credits. Those credits reflected the tokens each customer consumed and the published rate of the model that customer selected. Customers received advance notice and could preview their charges before the bill arrived. Every plan included an allowance, and annual subscribers kept premium-request pricing until their subscriptions expired. The subscription price did not change by a dollar. What changed was what that dollar bought.

Microsoft's scale changed how the pricing correction arrived, but not whether the underlying economics required it. On January 28, 2026, four months before that change, the company told investors on its earnings call that Copilot had passed 4.7 million paid subscribers and was growing 75 percent year over year. If a provider can fund the gap between a flat price and the cost of the resource that price buys, then it controls the timing and the form of the correction. That control is what buys advance notice, a published schedule, and tooling that shows customers the effect before it lands. If a provider cannot fund the gap, the

1. Michael Truell, "Clarifying Our Pricing," *Cursor*, July 4, 2025, accessed July 29, 2026; Maxwell Zeff, "Cursor Apologizes for Unclear Pricing Changes That Upset Users," *TechCrunch*, July 7, 2025, accessed July 29, 2026. The Cursor post describes the change of June 16 and the clarification of June 30. The TechCrunch report carries Truell's role and title.

gap sets the timing instead. Scale made the correction orderly. It did not make the variable cost disappear.[2]

From the buyer's side, the two pricing corrections looked very different. Cursor changed its plan without warning, then apologized and issued refunds after customers objected. GitHub announced its change in advance, published a schedule, and gave customers tools to preview the effect on their bills. The execution differed. The economic correction was the same.[3]

Both products had been sold and purchased like conventional software: one seat, one flat price, and a cost fixed when the contract was signed. Beneath that subscription, however, the provider carried a variable cost. Every request consumed computing resources, and heavier use increased that cost while the revenue from each seat remained fixed. Once the cost of serving heavy users exceeded what the subscription could support, the provider changed the terms.

The provider determined when and how that correction occurred. Customers did not have to approve the change, or even know how many tokens they were consuming, for it to affect them. In both cases, the movement was in one direction: away from unlimited use at a flat price and toward plans that tied allowances, credits, and additional charges to consumption.

The buyers had not chosen the wrong vendors, and the tools had not failed. They made an **error of category**. They bought access to a metered resource but managed it as conventional licensed software, as-

---

2. GitHub, "Update to GitHub Copilot Consumptive Billing Experience," *GitHub Changelog*, June 18, 2025, accessed July 28, 2026; GitHub, "GitHub Copilot Is Moving to Usage-Based Billing," *The GitHub Blog*, April 27, 2026, accessed July 29, 2026; Microsoft Corporation, "FY26 Second Quarter Earnings Call," January 28, 2026, accessed July 29, 2026. The Microsoft earnings call carries the subscriber figure and the growth rate.

3. GitHub, "Update to GitHub Copilot Consumptive Billing Experience," *GitHub Changelog*, June 18, 2025, accessed July 28, 2026; GitHub, "GitHub Copilot Is Moving to Usage-Based Billing," *The GitHub Blog*, April 27, 2026, accessed July 29, 2026. Act one sits on the GitHub Changelog, dated June 18, 2025. Act two was announced April 27, 2026, with a preview bill experience launched in early May ahead of the June 1 transition.

suming that the subscription price fixed the cost. What had looked like fixed-price software became a consumption-based resource all at once.

1.1 The purchase that is not one

The buyer understood the transaction through the conventional software model. Under that model, an organization buys licenses that give a defined number of employees access to a program. Once the organization pays for a seat, additional use creates almost no additional cost. An employee who opens the program a thousand times a day costs the same as one who opens it once a week.

The organization therefore fixes its cost when it signs the contract. Procurement negotiates the number of seats and the price of each seat. That per-seat amount is the **access price**. After the purchase, the organization manages access rather than consumption: who has a login, how many seats are active, and when the contract renews.

DEFINITION

Access price

The stated amount an organization pays for the right to use an AI capability, usually per seat or subscription. It does not include additional costs based on how much AI the organization actually uses.

DEFINITION

Software access model

A purchasing model in which an organization pays a fixed price for access to software, usually per seat or subscription. Because additional use creates little or no additional cost, the organization manages who has access rather than how much of the software each user consumes.

The software access model is not flawed. It accurately describes licensed software, and organizations have refined it through decades of enterprise purchasing. The problem begins when buyers apply that model to AI.

An AI assistant may be sold as a program licensed by the seat, but each task calls a model and consumes computing resources. The amount consumed depends on factors such as the model selected and the size

of the request and response. Each additional use therefore creates an additional underlying cost, whether the provider charges the buyer immediately or absorbs that cost for a time. AI access may be packaged like licensed software, but AI use behaves like a metered resource.

The result is a mismatch between what the organization manages and what its AI systems consume. Access management asks who can use the tool and how many seats are active. Resource management asks how much AI the work consumes, what that consumption costs, and what value the work returns.

An organization that manages only access to a metered resource is measuring the wrong quantity. It counts seats even though usage and cost are determined by tokens, requests, and credits. The organization is managing the right tool through the wrong economic model.

1.2 The consumption event

The discipline developed in this book begins with a single unit of measurement: **the consumption event**. Each time an employee, workflow, or product calls an AI model, the system consumes computing resources and creates one of these events. The meter records them, and the invoice sums their cost.

DEFINITION

Consumption event

A single use of an AI system that consumes metered computing resources and creates an underlying cost greater than zero, whether or not the buyer is charged for it separately. A consumption event may be measured in tokens, requests, credits, compute time, or another equivalent unit. It is the basic unit of consumption tracked in AI Operations Management.

The anatomy of a consumption event is simple, and Figure 1.1 shows its parts. Something enters the model: a prompt, a document, or a conversation history. The model performs a computation. Something returns: a completion, a suggestion, an answer, or a tool call.

As this work occurs, a meter records the resources consumed. Providers typically measure this consumption in input and output tokens, the units into which the model divides the material it reads and generates. The provider records those units in a usage ledger, even when the buyer never sees the individual event.

This creates an important difference in visibility. The provider sees a stream of metered consumption. The buyer may see only a flat subscription charge at the end of the month. Token counts can rise into the millions while remaining hidden behind that fixed price. The buyer often has no reason to examine the underlying consumption until usage exceeds an allowance or additional charges begin to appear.

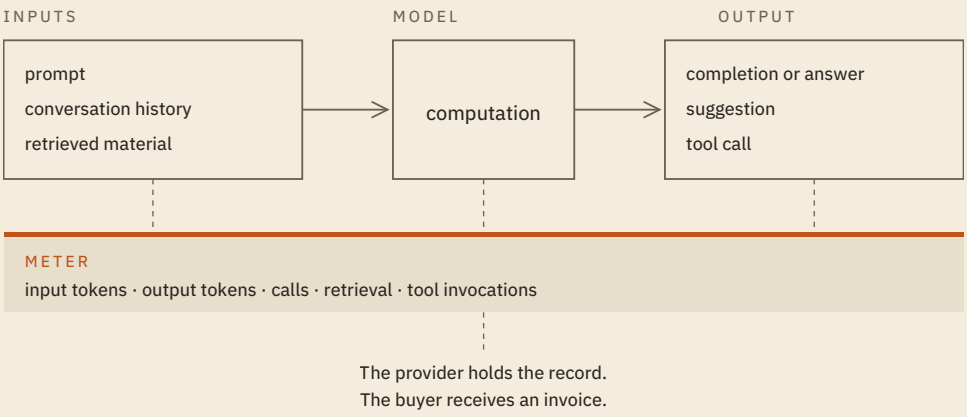

Figure 1.1 *Anatomy of a consumption event. The system assembles an input, the model performs the computation, and the system returns an output. A meter records the resources consumed by the event. In most cases, the provider retains this event-level record, while the buyer receives only an aggregate usage total or charge.*

AI Operations Management must measure cost in the same unit used to record consumption. Three units appear possible: the seat, the task, and the consumption event.

A seat measures access, not use. Two employees with identical seats can consume vastly different amounts of AI. A task is the right unit for measuring business value, because it connects AI use to completed

work. It cannot, however, measure cost reliably, because one task may require a single model call while another requires hundreds.

The consumption event provides the necessary unit for cost. It is the smallest recorded use of AI that consumes computing resources and creates an underlying cost. It is also the unit captured by the provider's usage meter. The organization can attribute costs to users, tasks, workloads, and workflows only by connecting those categories to the events that produced the costs. Later chapters group individual events into these larger operating units. Because each total begins with recorded events, the organization can trace it back to actual consumption and reconcile it with the provider's invoice.

A seat measures access. A task measures value. A consumption event measures use and cost.

Figure 1.2 *Two purchase models. Under the seat model, the buyer pays a fixed price for access, so cost remains flat as usage rises. Under the event model, each consumption event adds to the total, so cost rises with usage.*

The seat model and the event model do not describe the same cost in different ways. They describe two different cost structures. The seat model is the software access model defined in 1.1. The event model is the **resource consumption model**, in which each use is a metered consumption event. Figure 1.2 makes visible the difference that a flat subscription price can hide. Under the software access model, the buyer pays a fixed price for access, regardless of how much the software is

used. Under the resource consumption model, every use adds to the total cost.

A contract may package AI as fixed-price software, but that packaging does not make the underlying cost fixed. If the buyer manages seats while consumption governs the economics, the mismatch remains invisible until the provider changes the terms or additional charges appear. The correction then arrives on the provider's schedule, not the buyer's.

1.3 The flat-rate objection, answered

One objection stands against everything said so far: "We pay twenty dollars per seat each month. Our price is fixed, and the provider never bills us by the token. From our perspective, this is simply software."

The objection is valid. A buyer who pays a flat monthly price is not secretly being billed by the token. But this does not mean that the underlying resource is unmetered. Flat pricing relocates the meter from the buyer to the provider. It does not abolish it. This is **meter relocation**.

The provider continues to measure what each customer consumes. It sets the flat price based on an estimate of how much the average customer will use and what that usage will cost. The price is therefore built on an assumption. It remains viable while subscription revenue covers the total cost of serving the customer base.

The arrangement becomes unstable when actual usage departs from that assumption. If a small group of heavy users consumes a disproportionate share of the resource, the flat price collected across all customers may no longer cover the provider's cost. The provider must then bring the price or the product limits back into line with consumption.

It can raise the subscription price, charge for usage above an allowance, redefine the allowance in tokens or credits, or cap consumption outright. Chapter 4 examines these instruments in detail. The method may vary, but the direction does not: what the customer receives becomes more closely tied to what the customer consumes.

The buyer does not choose when this correction occurs. The provider holds the meter, bears the variable cost, and decides when the existing terms have become unsustainable. A flat price can hide the meter from the buyer. It cannot prevent the provider from acting when the numbers no longer work.

This result does not depend on a vendor being careless or badly managed. It follows from the economics of deployed AI, and the AI Operations Management registry states it as a theorem.

THEOREM 1 · THM-009

AI Use Is Resource Consumption, Not Merely Software Access

Within a defined deployment and cost boundary, if:

(i) an AI activity requires compute, and executing it consumes resources;

(ii) production scaling expands the surface over which that activity is used;

(iii) the buyer's total cost extends beyond the access price; and

(iv) measurement makes resource use, or the cost-bearing activity, visible;

then that AI use is a resource-consuming operating activity, not merely software access.

A theorem in this book is a statement established within a defined system from earlier propositions and lemmas. It is not a generalization drawn from the Cursor and Copilot episodes. Those cases illustrate the result; they do not prove it. Like every theorem in this book, Theorem 1 applies only when its stated conditions hold.

Its claim is deliberately narrow. The theorem does not say that flat-rate AI subscriptions are impossible, nor does it predict that every provider will reprice them. It says that when an AI activity requires computation,

each execution consumes resources. As employees, workflows, and products use that activity at greater scale, its resource consumption grows. When the stated cost boundary includes the cost of those resources, the economics of the activity cannot be described by the access price alone.

The fourth condition asks only that the resource use be measured somewhere, not that the buyer can see it. A provider's meter satisfies that condition even when the invoice shows a single flat charge. The business is operating a resource-consuming activity, not merely accessing software.

A flat monthly subscription to ChatGPT does not contradict this conclusion. The subscription price states what the buyer pays for access under a particular plan. It does not show how much computing capacity the buyer consumes, what that consumption costs, or whether the provider absorbs the cost within the subscription. Buyers may classify the charge as another SaaS expense because that is how it appears on the invoice. But the invoice describes the commercial arrangement, not the operating economics beneath it. The subscription buys access. Each use still consumes a resource.

DEFINITION

Meter relocation

The placement of consumption metering on the provider's side of a flat-rate subscription. The buyer pays a fixed price for access, while the provider continues to measure actual use. The provider sets the flat rate based on expected consumption and may change the price, allowance, or usage limits when actual consumption exceeds that expectation.

The public record already shows this pattern running its course in several forms. Some providers have moved directly to usage-based pricing. Others have retained a flat subscription while adding allowances, credits, or hard limits on consumption. The commercial instrument differs, but the correction is the same: what the buyer receives is brought back into line with what the buyer consumes.

DATED: JANUARY 2025

OpenAI provided an early example. Chief executive Sam Altman said publicly that the company was losing money on its two-hundred-dollar Pro subscriptions because subscribers used them more than the price had assumed. He also acknowledged that he had personally set the price. The problem was not the price of access. Actual consumption had exceeded the assumption built into it.[4]

DATED: JULY 2025

Claude Code customers encountered the same economic correction through tighter limits rather than a higher subscription price. Subscribers reported that they were reaching their plans' usage limits sooner, although Anthropic had given no notice of a change. Many had not realized that their subscriptions were subject to such limits at all. Anthropic acknowledged the reports but did not confirm that it had altered the plans. Those reports were published on July 17, 2025.

Eleven days later, the company announced two new weekly usage caps in addition to its existing five-hour limits. The caps would take effect the following month for all Pro and Max subscribers. Max subscribers could continue using Claude Code beyond those limits by purchasing additional usage at standard API rates. The subscription price remained intact, but the amount of consumption included within it now had a clearer boundary.[5]

The Cursor and GitHub Copilot episodes that opened this chapter show the same pattern. Within twelve months, two widely used AI coding products revised their subscriptions so that what customers received more closely reflected what they consumed. Cursor made the correction

4. Sam Altman (@sama), post on X on ChatGPT Pro subscription economics, January 5, 2025, accessed July 28, 2026; Kyle Wiggers, "OpenAI Is Losing Money on Its Pricey ChatGPT Pro Plan, CEO Sam Altman Says," *TechCrunch*, January 5, 2025, accessed July 29, 2026. The TechCrunch report reproduces both statements.

5. Anthropic (@AnthropicAI), post on X announcing new weekly rate limits for Claude Pro and Max, July 28, 2025, accessed July 28, 2026; Maxwell Zeff, "Anthropic Unveils New Rate Limits to Curb Claude Code Power Users," *TechCrunch*, July 28, 2025, accessed July 29, 2026; Russell Brandom, "Anthropic Tightens Usage Limits for Claude Code Without Telling Users," *TechCrunch*, July 17, 2025, accessed July 29, 2026. The announcement set two weekly caps, effective August 28, 2025, in addition to five-hour limits that were already in force and that subscribers reported, in an account published July 17, 2025, had been tightened without notice.

abruptly and responded to complaints with an apology and refunds. GitHub announced its change in advance and introduced it with billing tools and a transition period. The method differed. The economic correction did not.

In both cases, the provider continued to measure usage beneath the flat subscription price. The provider determined when the existing terms no longer worked and changed the arrangement on its own schedule. The flat rate had not removed the meter. It had placed the meter, and control over the correction, on the provider's side of the transaction.

1.4 What follows if this is true

Once an organization recognizes deployed AI as a resource it consumes, the management requirements become familiar. Businesses already know how to manage resources used in daily operations. Consider a manufacturer that relies on steel. Counting how many employees are authorized to order steel would measure access to the resource, but it would not manage the resource itself. The company must determine how much steel its operations require, what specifications it must meet, what it costs, where it is used, and what value it helps create. The same management logic applies to AI.

Before purchasing steel, the company forecasts demand, sets a budget, and evaluates suppliers against its requirements. During production, it compares actual consumption with the plan and tracks how much steel each product line uses. If supply becomes limited, it allocates the available steel according to business priorities. It then includes the cost of that steel when calculating the profitability of each finished product. If a product consumes more steel than its margins can support, management can see the problem and respond.

These practices are not unique to manufacturing. They apply to any resource that costs money as it moves through an organization. To manage such a resource, leaders must be able to answer five questions:

What is the organization buying, and what requirement must it meet?

How much does it expect to consume, and how will it know when actual use departs from the plan?

Where is the resource being used?

Who decides how it is allocated when there is not enough?

What value does the organization receive in return?

If an organization cannot answer these questions, it is not managing the resource, regardless of what its organizational chart or policies suggest. It is simply receiving invoices and paying them.

Deployed AI is one of these resources. Every time an employee, workflow, or product uses AI, the organization consumes computing resources and incurs a cost. Total consumption rises with the volume of work, and as a deployment succeeds, that volume usually grows.

Most organizations, however, do not manage AI usage through a single, coordinated discipline. Responsibility is divided across several existing functions. Procurement negotiates the contract. Cloud cost management monitors infrastructure spending. Engineering tracks the systems it operates. Finance allocates the costs it can trace. Each function manages one part of the resource, but none manages it from purchase through consumption to business return.

These functions were designed before deployed AI created a resource that crossed all of their boundaries. As a result, AI usage touches several owners but belongs fully to none. Chapter 14 distinguishes the subject of this book from these neighboring disciplines. What is missing is not attention. It is a management system that brings the separate responsibilities together.

Organizations that would never allow material to move through a plant without a record often allow AI consumption to pass through daily work unrecorded. The five questions therefore go unanswered, and the reason is visible in three places.

First, the organization tracks the wrong quantities. A company accustomed to seat-based software tracks headcount, license utilization, and renewal dates. None of these measures shows how much AI the organization consumes or what produces the bill.

Second, the organization lacks a record of use. Under a seat-based contract, each additional use costs nothing at the margin, so the buyer has little reason to record it. Under an event-based model, every use consumes resources and can add to the bill. Yet a company that has not built a meter cannot show which work produced that consumption. The invoice does not provide this record. It reports the total cost, not the uses that created it.

Third, the organization has no basis for accountability. Finance cannot assign a cost to the team, workflow, or product that incurred it unless the company can trace the cost back to the work. A cost assigned to no one is defended by no one.

It would be a mistake to treat these gaps as negligence. The organization inherited them from the seat-based software model, just as it inherited the absence of a clear owner. It did not decide to stop metering software use. It never began, because the software it had purchased for three decades did not require it. Enterprise software procurement built a management system around the economics of the license: count the seats, track utilization, and manage renewals. That system worked.

AI often entered the organization in familiar packaging. It came through the same vendors and procurement channels, initially at a price low enough to approve without much debate. The organization therefore managed it as another software license. The existing system did not fail. It did exactly what it was built to do, continuing to report seats, utilization, and renewal dates even after those measures had stopped explaining consumption and cost.

Scale makes this inherited arrangement untenable. While AI remains a pilot, the organization can afford to manage it as a software license. One team uses a few seats, and the bill is too small to matter against the larger software budget.

This all changes when the pilot enters production. AI usage then grows with the volume of work the system performs. A contact center does not merely add licensed users. It applies AI across the conversations and workflows those users handle, generating a new consumption event each time the system performs work.

The contact-center deployment examined later in this chapter illustrates the difference. It operates with roughly five thousand seats but generates tens of millions of consumption events each month. The craft section calculates this ratio in full. At pilot scale, seat count is a harmless simplification. At production scale, it is a blindfold.

The problem established in this chapter defines the work of the rest of the book. Once AI consumption moves through daily operations and produces variable cost, the organization needs a system for measuring, assigning, and governing it. No existing management practice provides that system in assembled form. The remaining chapters build it.

1.5 What this book is not

This book addresses what happens after an organization decides to deploy AI. It does not explain how models work internally, so it carries no account of architectures, training, or weights. It does not teach prompt engineering or offer techniques for improving a model's output. It also does not help leaders identify problems that AI might solve. Each of these subjects has its own purpose and literature, and Chapter 3 explains where each ends and AI Operations Management begins.

The subject of this book is the deployed resource itself. It examines what AI consumes as employees, workflows, and products use it; what that consumption costs; and how the organization measures, assigns,

and governs both. The decision to deploy establishes the need. This book explains how to manage what follows.

CRAFT SECTION

The consumption-event inventory

THE CONTACT-CENTER SETTING AND APPROXIMATE AGENT POPULATION ARE DRAWN FROM THE CITED STUDY. THE EVENT ARCHITECTURE AND ALL VOLUME ASSUMPTIONS ARE STIPULATED FOR THIS EXERCISE.

A seat count shows how many people can use an AI system. It does not show how much AI activity their work generates. **The consumption-event inventory** replaces that administrative count with an operating view of the deployment. It identifies each kind of consumption event, what resources the event consumes, where that consumption is measured, and what the seat count leaves hidden.

This is the first artifact the reader produces in this book. Later artifacts use it to trace consumption, assign costs, and compare those costs with the value created. The inventory therefore needs to be specific enough to run against an actual deployment.

The procedure has four steps.

Step 1: Enumerate the event types. List every distinct kind of consumption event generated by the deployment. Begin with the work the system performs, not with the number of employees who have access. A single deployment may classify a request, retrieve information, generate several responses, call a tool, and summarize the completed interaction. Each is a separate event type when it triggers its own measurable use of resources.

Step 2: Identify the resource drivers. For each event type, state what causes its resource consumption to rise. Common drivers include input tokens, output tokens, model calls, retrieval operations, tool invocations, and the amount of context processed. Do not assume that two events consume the same resources merely because they occur inside the same workflow.

Step 3: Locate the meter. State which system records the consumption and which party controls that record. The meter may sit with the model provider, the application vendor, the organization's cloud environment, or an internal platform. A flat-rate contract does not remove this step. It means only that the provider may hold the operational meter while the customer sees a fixed price.

Step 4: State what the seat count conceals. Estimate the volume of each event type over a defined period. Show the operating assumptions used in the calculation, then compare the resulting activity with the number of seats. The purpose is not merely to produce a larger number. It is to reveal the work, consumption, and cost drivers that cannot be seen from the seat count alone.

Consider a customer-support organization that has deployed a generative AI assistant across its contact center. For each incoming customer message, the assistant drafts a suggested reply that the agent may edit and send; the assistant draws on a knowledge base and on the conversation so far. The deployment covers a large agent population, on the order of five thousand agents, and the organization currently accounts for it as a per-seat tool.[6]

MODEL INVENTORY

Step 1. Event types. The deployment generates at least three: (a) a suggested-reply generation, triggered each time an agent asks the assistant to draft a response to a customer message; (b) a knowledge retrieval, triggered when the assistant pulls reference material to ground a suggestion; and (c) a conversation-close operation, in which the deployment summarizes or tags the resolved conversation.

Step 2. Resource drivers. For suggested-reply generation, the drivers are input tokens (the system instructions, the retrieved knowledge, and the conversation history assembled into the prompt) and output tokens (the drafted reply). The volume driver is the number of customer messages that receive a drafted reply,

---

6. Erik Brynjolfsson, Danielle Li, and Lindsey Raymond, "Generative AI at Work," *Quarterly Journal of Economics* 140, no. 2 (2025): 889-942. The study supplies the setting and the scale of the agent population. The event architecture and the volumes used below are stipulated for this exercise and are not reported by the study. It returns in Chapter 6 as the book's anchor case on realized value.

which scales with contacts multiplied by the number of turns per contact. For knowledge retrieval, the drivers are the number of retrieval calls and the tokens those calls embed and return. For the conversation-close operation, the drivers are input tokens (the full conversation) and output tokens (the summary or tags); their volume scales with the number of resolved conversations.

Step 3. The meter. The generation and close operations are metered on the provider's side, by token. Retrieval is metered wherever the retrieval service runs. In the architecture stipulated here, the retrieval service is the provider's managed index, metered by call and by token; an organization that operates its own vector store meters that step itself. Meter location is a property of the architecture rather than of the event type, which is why Step 3 directs the reader to locate the meter instead of assuming it. Under its per-seat plan, the organization receives a monthly invoice that shows only a seat count.

Step 4. What the seat count conceals. The seat count treats the agent population as a set of equal units: five thousand seats at one price. Put the volume drivers from Step 2 against it and the two quantities separate immediately. Stipulate, for this exercise, five thousand agents, each handling forty contacts on a working day, with six drafted replies per contact, one retrieval per drafted reply, and twenty-one working days in the month. Suggested-reply generations then run at 5,000 × 40 × 6 × 21, or 25.2 million events a month. Retrievals match them one for one at 25.2 million. Conversation-close operations run once per contact, at 4.2 million. The deployment generates roughly 54.6 million consumption events a month, which is about 10,900 events behind every seat.

Change any stipulated figure and the total moves; change none of them and the seat count still does not move, because the seat count is not a function of any of these quantities. Two agents on identical seats can consume amounts of the resource that differ by an order of magnitude: a high-volume agent handling long, reference-heavy conversations generates many times the consumption of a low-volume agent, at the same seat cost. The quantity that drives the bill is tokens per resolved contact. It does not appear on the seat line, and until the inventory names it, the organization is managing a number that does not govern its cost.

The inventory has done its work when the organization can see, for the first time, the shape of what it is buying. Not five thousand seats. A flow of tens of millions of consumption events whose volume and intensity, not headcount, determine the bill.

Chapter summary

The category error is now clear. Under the software access model, the organization fixes its cost when it signs the contract and manages the number of seats. Under the resource consumption model, each metered event adds to total cost, and the organization must manage the resulting flow of consumption. These are different operating models, even when the provider packages both as software subscriptions.

The consumption event is the atomic unit of the second model. It is the smallest use of AI that consumes resources and creates an underlying cost. It is also the smallest unit whose consumption the provider's meter records. Seats measure access. Consumption events measure use and cost.

A flat price does not erase this distinction. It relocates the meter to the provider, which continues to measure consumption and decides when the commercial terms must change. This conclusion does not require a prediction about any particular vendor. It follows from Theorem 1: deployed AI is a resource-consuming operating activity, not merely software access.

The reader can now apply the consumption-event inventory to a deployment description. The inventory identifies the types of events the deployment produces, the resources each event consumes, where that consumption is metered, and which cost drivers the seat count conceals. It makes the flow visible. It does not yet explain how to manage that flow. Seeing it whole comes first.

Key terms

Category error

The error of managing a metered resource as if it were licensed software. It rests on a false assumption: that the access price fixes the total cost. The tool is the right one and the vendor is not at fault. The economic model applied to the tool is wrong.

Consumption event

A single use of an AI system that consumes metered computing resources and creates an underlying cost greater than zero, whether or not the buyer is charged for it separately. A consumption event may be measured in tokens, requests, credits, compute time, or another equivalent unit. It is the basic unit of consumption tracked in AI Operations Management.

Resource consumption model

The economic model in which deployed AI is consumed as a metered resource. Each use creates a consumption event, and cost accrues with each event. It contrasts with the software access model.

Software access model

A purchasing model in which an organization pays a fixed price for access to software, usually per seat or subscription. Because additional use creates little or no additional cost, the organization manages who has access rather than how much of the software each user consumes.

Access price

The stated amount an organization pays for the right to use an AI capability, usually per seat or subscription. It does not include additional costs based on how much AI the organization actually uses.

Metered resource

A resource whose consumption is measured per unit of use, here in tokens or their equivalents.

Flat-rate objection

The claim that a flat per-seat price makes AI equivalent to licensed software; answered by meter relocation.

Meter relocation

The placement of consumption metering on the provider's side of a flat-rate subscription. The buyer pays a fixed price for access, while the provider continues to measure actual use. The provider sets the flat rate based on expected consumption and may change the price, allowance, or usage limits when actual consumption exceeds that expectation.

Discussion questions and problems

DISCUSSION QUESTIONS

1 A colleague argues that your organization pays a flat per-seat price and never receives a token bill, so AI is a seat-priced good and the consumption model does not apply. Explain why the contract does not change the underlying economic model. Your answer should not depend on predicting how any specific vendor will price its product in the future.

2 The Cursor and GitHub Copilot episodes can be read as two companies changing their prices. Taken together, what do they reveal about where consumption was being measured all along? Why does the evidence become stronger when the two cases are considered together?

3 Section 1.2 treats the consumption event, rather than the user or the task, as the discipline's atomic unit. Make the strongest case for using the task instead. What could a task-based discipline explain or manage more effectively? What would it lose by moving away from the event?

4 Construct the strongest flat-rate objection a capable skeptic could make after reading this chapter. Then answer it. Your response should explain why flat-rate pricing does not eliminate the underlying consumption problem without assuming that any particular subscription will eventually be repriced.

PROBLEMS

P1 · WORKED

Five hundred seats, one meter

A company purchases an AI coding assistant for five hundred employees at a fixed annual price per seat. The contract contains no token charges, and the finance team records the entire purchase as software expense. Over the next year, some employees use the assistant occasionally while others use it throughout the workday.

Using the resource consumption model, explain why the absence of a usage-based charge on the company's invoice does not make seat count the underlying unit of AI consumption. Identify where the meter is likely to reside, what it measures, and what the per-seat price conceals.

MODEL ANSWER

The company pays for the AI coding assistant by the seat, but the seat is the commercial pricing unit, not the underlying unit of consumption. The five hundred employees do not consume the same amount of AI simply because the company purchases five hundred licenses. An employee who uses the assistant several times a month generates far less activity than one who relies on it throughout the workday.

The underlying meter therefore sits behind the seat price. Each time an employee uses the assistant, the system processes a request and consumes computing resources. The provider can measure those consumption events even when it does not expose them as separate charges on the customer's invoice. The flat annual price is built around assumptions about how much consumption the average seat will generate.

Seat count therefore conceals the actual cost driver. Two companies may each purchase five hundred seats while generating very different levels of AI consumption. The same difference can exist inside one company: a small group of heavy users may account for a disproportionate share of total consumption even though every employee carries the same accounting cost per seat.

Under the resource consumption model, the meter has not disappeared. It has moved to the provider's side of the transaction. The company sees a fixed software price, while the provider sees the flow of consumption events that the price must cover.

ANNOTATED REASONING

The response begins by separating the commercial pricing unit from the underlying consumption unit. The company buys five hundred seats, but equal seat count does not imply equal AI use. It then makes the hidden mechanism visible: each request consumes computing resources, and the provider can measure those events even when the customer never sees a usage-based charge. From there, the response explains what the seat price conceals. Users, teams, and companies with the same number of seats can generate very different levels of consumption. The conclusion follows directly: flat pricing changes where the meter is visible, not whether consumption exists.

P2 · WORKED

Inventory a coding-assistant deployment

A software company has deployed an AI coding assistant to three hundred developers. The assistant provides inline code completions, answers questions in the editor, and can perform multi-step agentic tasks. Produce the consumption-event inventory for the deployment.

MODEL INVENTORY

Event types:

(a) Inline completion: triggered as a developer types and the assistant generates a code suggestion.

(b) Editor chat query: triggered when a developer asks the assistant a question.

(c) Agentic task run: triggered when a developer delegates a multi-step task to the assistant.

Resource drivers:

Inline completions consume input tokens from the surrounding code context and output tokens from the generated suggestion. Each event may be small, but the total volume can be high because completions occur repeatedly throughout the workday.

Editor chat queries consume input tokens from the developer’s question and any code or file context supplied with it. They also consume output tokens from the assistant’s response.

Agentic task runs consume resources across multiple model calls rather than a single request. They may also invoke tools, inspect files, generate code, and repeat steps before the task is complete. For that reason, one agentic run can consume substantially more resources than a single completion or chat query.

Meter: The meter sits primarily on the provider's side. The provider can measure token consumption and, for agentic tasks, additional activity such as tool calls or repeated model requests. Under a seat-priced plan, the buyer may see only the number of seats and the resulting subscription charge.

What the seat count conceals: Seat count does not show how much AI the developers actually consume. Two developers with identical seats may generate very different levels of usage. A developer who runs long agentic tasks throughout the day may consume far more resources than several developers who use only occasional inline completions.

The important cost drivers are therefore the number, type, and intensity of consumption events. In this deployment, agentic task volume and length are likely to be especially important. The seat count reveals none of that variation.

P3 · COMPLETION

Inventory a document-review deployment

A legal operations team has deployed an AI assistant that extracts key clauses from each contract submitted, compares them against a policy playbook, and drafts a redline memo. Complete the inventory below by filling the blank column.

EVENT TYPE	RESOURCE DRIVERS	METER LOCATION	WHAT THE SEAT COUNT CONCEALS
_____	Input tokens (full contract text), output tokens (extracted clauses); scales with contract length	Provider side, per token	Cost scales with document length, not reviewer headcount
_____	Input tokens (extracted clauses plus playbook), output tokens (comparison result); retrieval calls against the playbook	Provider side, per token and per call	A long playbook inflates every comparison regardless of seat count
_____	Input tokens (clauses, comparison, contract context), output tokens (the drafted memo); highest output-token event	Provider side, per token	The drafting step, not the reviewer, is the dominant cost driver

Interleaving: none. This is the first chapter; problem sets begin reaching back to earlier chapters in Chapter 2.