THE LIVING BOOK

Thinking in Products, Building Systems

From physical design to digital products, projects and enterprise portfolios.

José Ángel Sosa López

Living edition 0.1 — 1 of ~20 chapters published — updated weekly

PDF snapshot generated 2026-09-18 from the same source as the web edition

PART I — UNDERSTANDING THE PRODUCT · CHAPTER 01

What Can Industrial Design Teach Digital Product Leaders?

Materials, tooling, cost and maintenance shape every physical object long before it reaches a user. This chapter argues that a product is 10% idea and 90% effort to prove that idea can survive reality — in wood or in software.

Published Sep 9, 2026 18 min read Edition 0.1 Status: Published

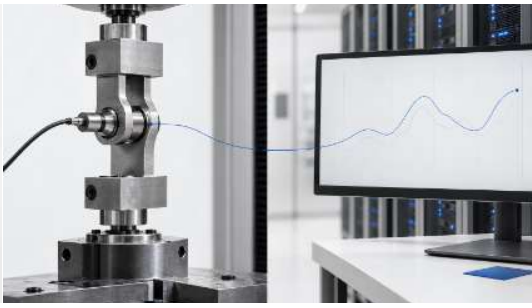
There is a part of product design that happens long before anything exists that we can touch, download, or put in front of a user — less visible than the render or the first working screen, but probably where much of what the product will become gets decided: cost, manufacturability, maintenance, how long it stays relevant.

When I studied industrial design I learned to live with those questions before I got used to calling them product strategy. Years later, working with digital products, platforms and enterprise portfolios, the same questions came back under different names. The materials changed; the problem didn't change much.

If I had to reduce it to a deliberately imperfect ratio: a product is 10% idea and 90% effort to prove that idea can survive reality. It isn't a statistic — it's a way of ordering the experience this chapter unpacks through an 1859 chair, a few uncomfortable hypotheses, and the argument that digital architecture is itself a design material.

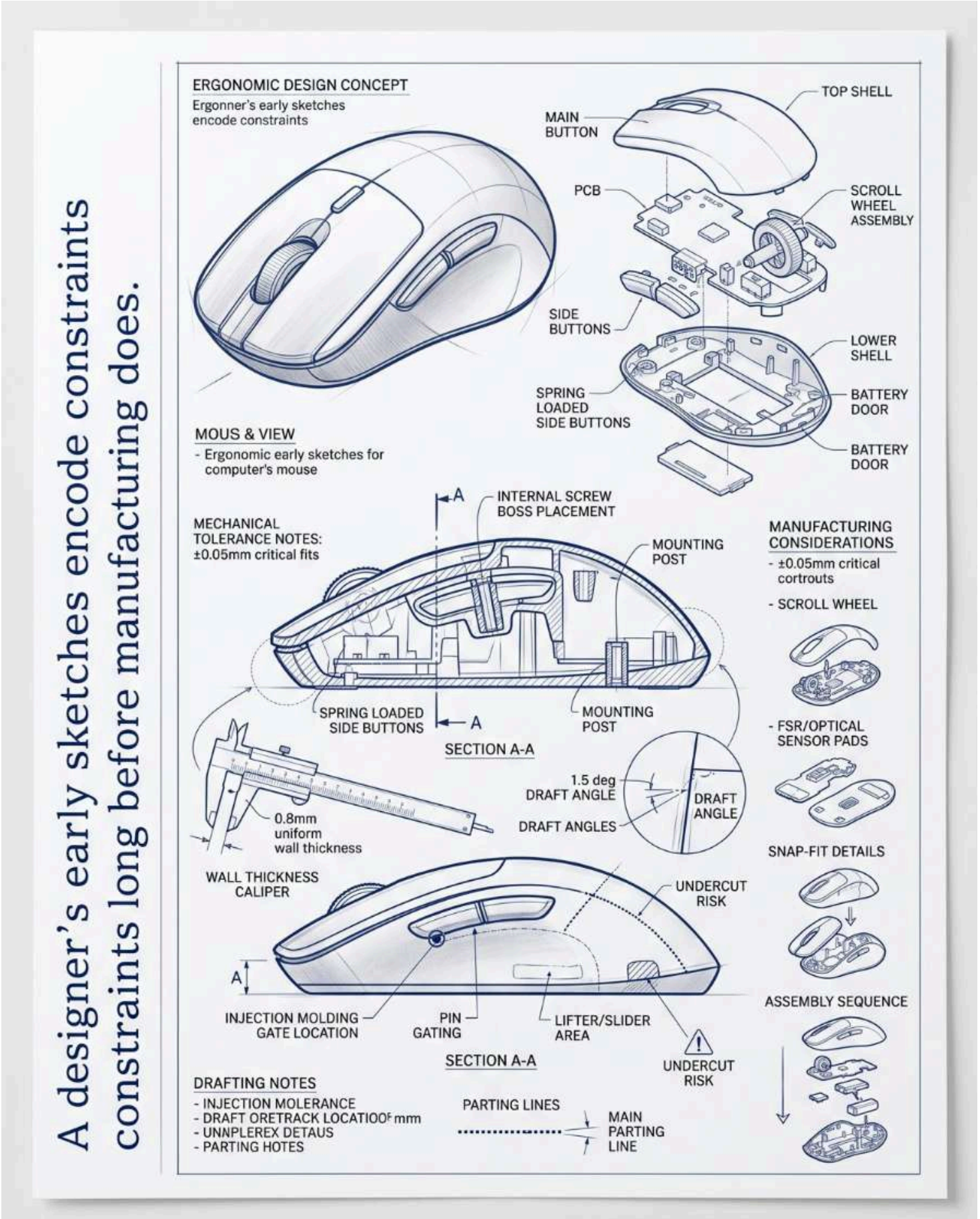
Six pieces, ten screws, two nuts: the architecture Thonet solved in 1859.

Tooling cost is a decision made once and paid for in every unit after.



A hinge under load behaves like an API under traffic: both have a yield point.

The idea is only a hypothesis



A designer's early sketches encode constraints long before manufacturing does — tolerances, draft angles, wall thickness.

Because an idea doesn't yet have a unit cost, an architecture, suppliers, channels, risk, or an expected return. It doesn't have support, production capacity, a backlog, adoption metrics, or a clear model for capturing value. All of that shows up once we stop falling in love with the possibility and start designing the product. In that sense, physical and digital development start from a similar condition: before you can build, you have to turn uncertainty into decisions.

There's a common temptation in product work: confusing the clarity of an idea with its viability. In industrial design the mistake surfaces fast because matter pushes back. A shape can be attractive and still impossible to manufacture with the chosen process; it can require a mold too expensive for the expected volume; it can be too heavy for the logistics channel; it can use an elegant joint that makes repair prohibitive.

The object forces you to negotiate.

Software buys a bit more time before the bill comes due. We can build an experience that works in a demo and still not have a product capable of operating: identity flows, permissions, exception handling, telemetry, billing or support are still missing behind the interface. The absence of physical material doesn't remove the constraints; it just makes them less visible.

Ulrich, Eppinger and Yang frame development as an interdisciplinary discipline where opportunity, planning, customer needs, specifications, concepts, architecture, industrial design, manufacturing and supply chain, prototyping, product economics and project management all coexist. Robert Cooper, from Stage-Gate, places an initial market and feasibility investigation before scaled development, followed by a business case where commercial attractiveness, technical feasibility, risk and financial return intersect.

I'm less interested in defending one method than in pointing out something both models make clear: product development starts well before technical development. Is there really a need, or do we just have a good story to tell about one? Is the problem frequent or costly enough that someone would change behavior? How much would we have to invest before learning we were wrong?

In industrial design those questions lead to materials, suppliers, tooling costs, volumes and price. In a digital product they become estimates for development, cloud, licensing, data, user acquisition and technology evolution. The spreadsheets change. The economic decision stays essentially the same.

"Monetization reshapes the product."

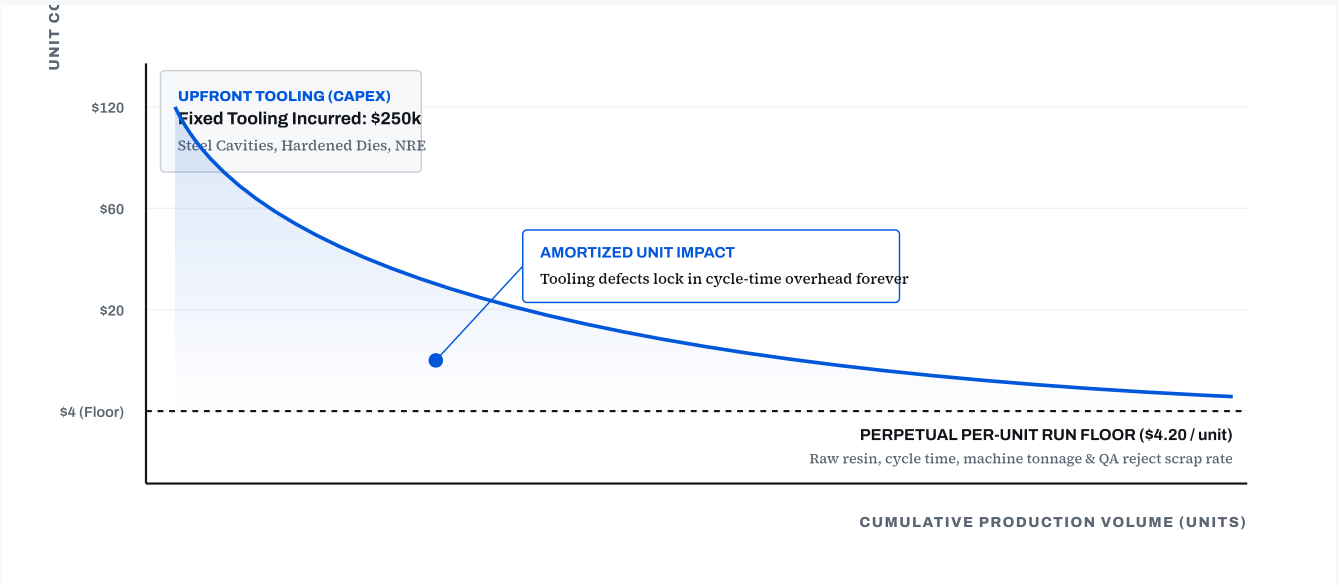
A product doesn't exist because it can be built. It exists when we find a reasonable way to create value, deliver it, and capture enough value to sustain the system that produces it. David Teece defines the business model as the architecture through which a company creates, delivers and captures value — a definition I like because it puts monetization back where it belongs: inside product design, not as a conversation that arrives late.

Designing something sold once is not the same as designing something that has to justify a subscription every month. Selling a machine is not the same as selling the service that machine produces. Offering software by license, by consumption, by seat, by transaction, or through a combined scheme isn't the same either. Every choice ends up shaping architecture, experience, metrics, operations and the customer relationship.

In a physical object this is easier to see because the cost of each unit sits right in front of us. In digital there's a temptation to think that because copying software costs almost nothing, the economics are trivial. But low marginal cost doesn't mean low total cost: keeping systems available, storing data, processing transactions, paying for AI models, running infrastructure, handling incidents and acquiring users are all part of the product too.

That's where another false boundary starts to break down: the one that separates design, technology and business too early.

The cost that freezes before the first sale



Designing is deciding under constraints

Material is one constraint among many — cost, scale, time, regulation, channel.

In school you can fall in love with form. It's natural. But at some point the product stops being only what we want it to be and starts becoming what it can be within a set of constraints. Material is one of them, not the only one: so are cost, process, scale, time, regulation, supplier capacity, packaging, channel and user expectation. Those constraints don't arrive at the end to correct the design. They participate in defining it.

The same is true in digital product. Architecture, data, identity, security, interoperability, latency, availability, privacy or integration capacity shouldn't be a list of problems technology solves after product has finished imagining the experience. They are design materials.

When I say digital architecture is a material, I don't mean it as an aesthetic metaphor. It has properties and limits. An API can support one usage pattern and not another. A data model can make one kind of evolution easy and turn another into a costly migration. An identity decision can let capabilities be shared across products or lock them into separate domains.

The industrial designer learns to ask what the material allows. A product manager should ask a similar question about the system they're building on. This matters because the product that reaches the market isn't just the sum of visible features. It's the accumulated result of prior decisions. Some open possibilities. Others close them.

6

MAIN COMPONENTS — PLUS 10 SCREWS AND 2 NUTS

36

DISASSEMBLED CHAIRS PER CUBIC METER SHIPPED

50M+

UNITS PRODUCED BY 1930

A chair that was actually a system

The Thonet No. 14 chair, sold today as model 214.

There’s a chair that sums up much of this argument better than any contemporary diagram. Michael Thonet spent the first half of the nineteenth century developing techniques for bending wood, and by the 1850s had perfected a solid-wood bending process that made serial furniture production possible. In 1859 the chair known then as No. 14 appeared — the same one Thonet sells today as model 214.

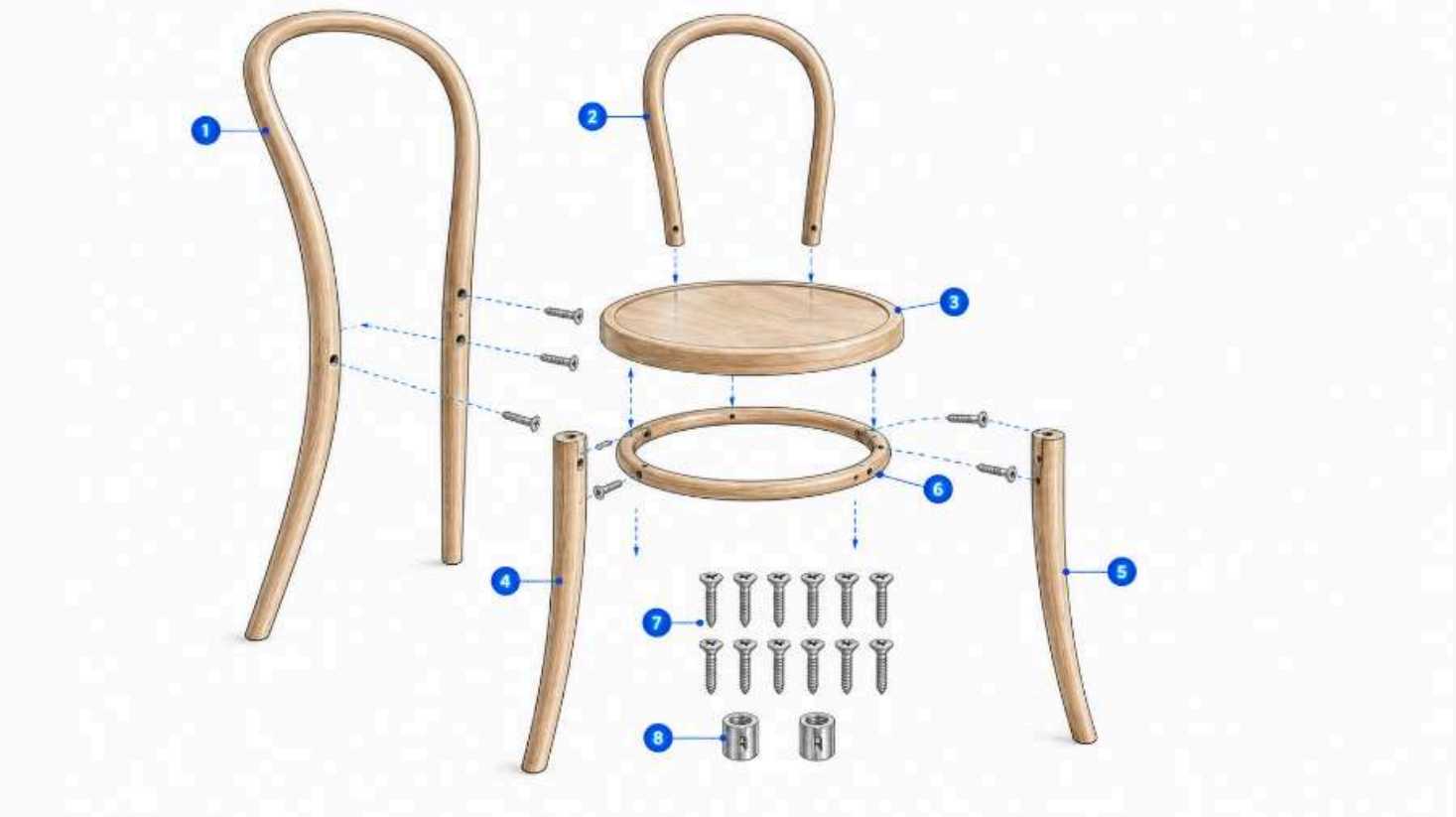
At first glance its success might be attributed to its form: light, recognizable, neutral enough to exist for decades in cafés, restaurants and homes without depending too much on a specific trend. But focusing only on the silhouette misses the more interesting part. The chair was a system of decisions.

The model could be resolved with six main pieces, ten screws and two nuts. That reduction wasn't just formal. It let manufacturing and assembly be separated, standardized components, reduced craft labor, and made it possible to ship the chair disassembled. Thonet documents that 36 disassembled chairs could be packed into a one-cubic-meter crate and assembled at the destination. By 1930, more than 50 million units had been produced.

In other words: the chair wasn't designed only to be sat on. It was designed to be produced, repeated, packed, shipped and assembled. That completely changes how you read the object.

There's a technology decision — bent wood; a component decision — few standardized pieces; a joining decision — screws and nuts; a manufacturing decision — repeatable production; and a logistics decision — shipping volume disassembled to be assembled at the end.

Six pieces, one system



Exploded view: six bentwood pieces plus the hardware that joins them.

Illustration in design

A one-cubic-meter crate: 36 disassembled chairs ready to be assembled at destination.

Detail of the solid-wood bending process Thonet perfected in the 1850s.

Screw and nut: the joint that makes the whole system repairable — and repeatable.

Architecture built to repeat

A platform is a collection of assets shared across a family of products.

This is one of the reasons I find it such a useful precedent for understanding digital product. Not because an API is equivalent to a wooden leg, but because the design principle is recognizable: if an architecture lets you reuse components, the next variant costs less than

reinventing the whole system from scratch.

Contemporary research on modular architecture describes exactly this problem. Dahmus, Gonzalez-Zugasti and Otto study product families built from shared, interchangeable modules, aiming to reuse functions across variants. Robertson and Ulrich describe a platform as a collection of assets — components, processes, knowledge, people and relationships — shared across a set of products.

Strip away the industrial vocabulary for a moment and it's hard not to recognize a modern conversation about digital platforms. A shared identity capability can power several experiences. A payments service can be used across different products. A catalog, a data layer, or a

> *"Thonet figured this out in wood. Plenty of companies are still trying to figure it out in software."*

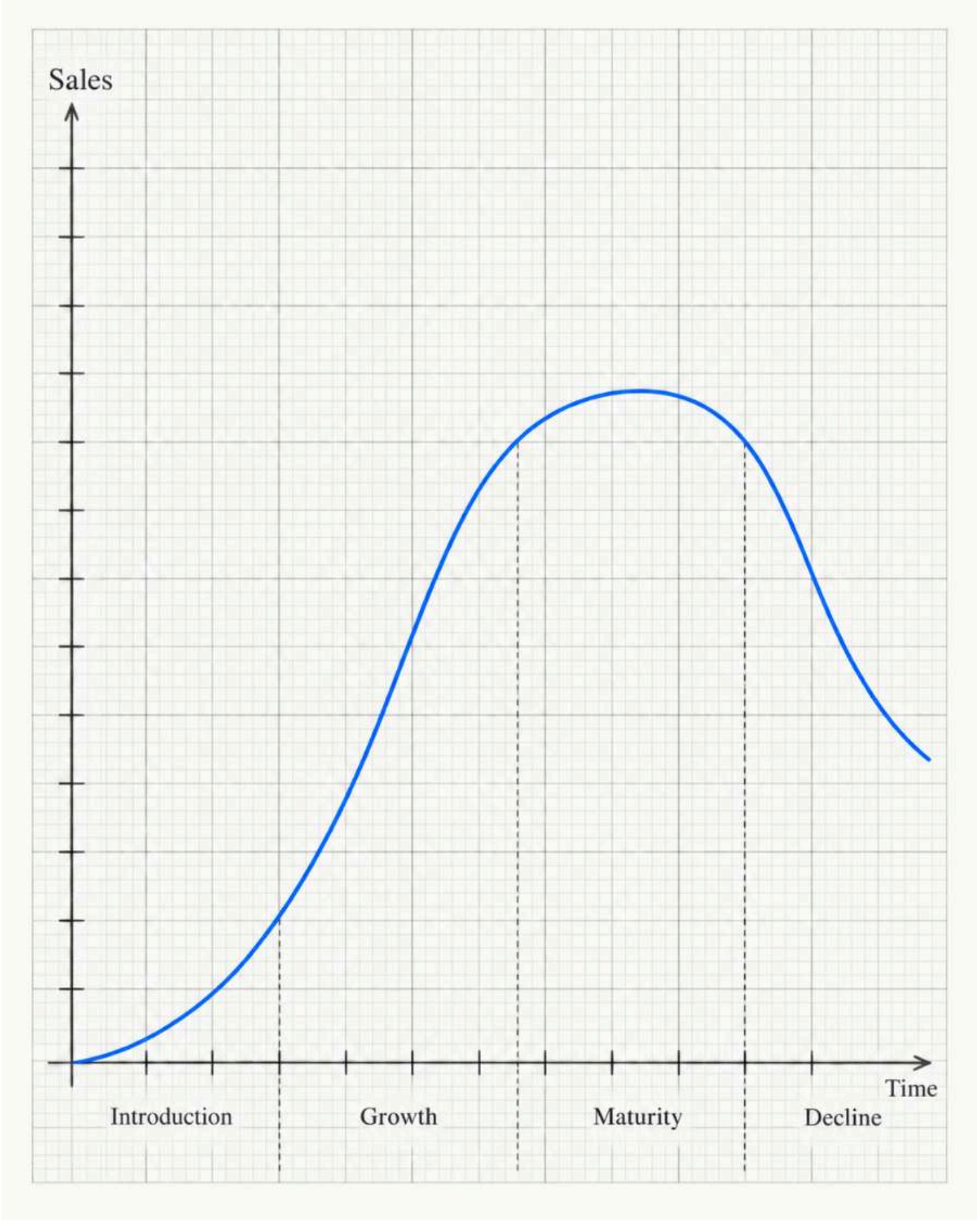
Another lesson from physical product is that a version doesn't need to contain everything the product could eventually become. The industry has worked for a long time with generations, variants, platforms and incremental improvements. We sometimes forget this when we tell the story of digital product as if iteration were invented by Agile. It wasn't.

A U.S. Government Accountability Office study of development practices found that the commercial companies it analyzed reduced risk through evolutionary development: they limited how much new content went into a given generation and pushed capabilities that weren't yet mature enough into later generations, because changes get more expensive as a program advances.

Iteration shouldn't be an excuse to start without judgment; it should be a strategy for deciding what we need to learn now and what can wait until we have evidence. The point of a first version isn't to ship a weaker version of the full vision. It's to build something coherent enough to test the hypotheses most capable of sinking the case if they turn out to be false.

Do people understand the value? Can they complete the task? Do they come back? Do they pay? Does the architecture hold up under real usage patterns? Every product should have its own questions. That's where incremental doesn't mean small. It means controlled.

The lifecycle doesn't start at launch



Launch doesn't open the product. It exposes it.

For a long time, the classic product-lifecycle chart was drawn as a curve running through introduction, growth, maturity and decline. Theodore Levitt turned that idea into a strategy tool in 1965. The curve is useful, but it can create a too-comfortable interpretation if we think product work begins at the left edge of the chart, right at launch.

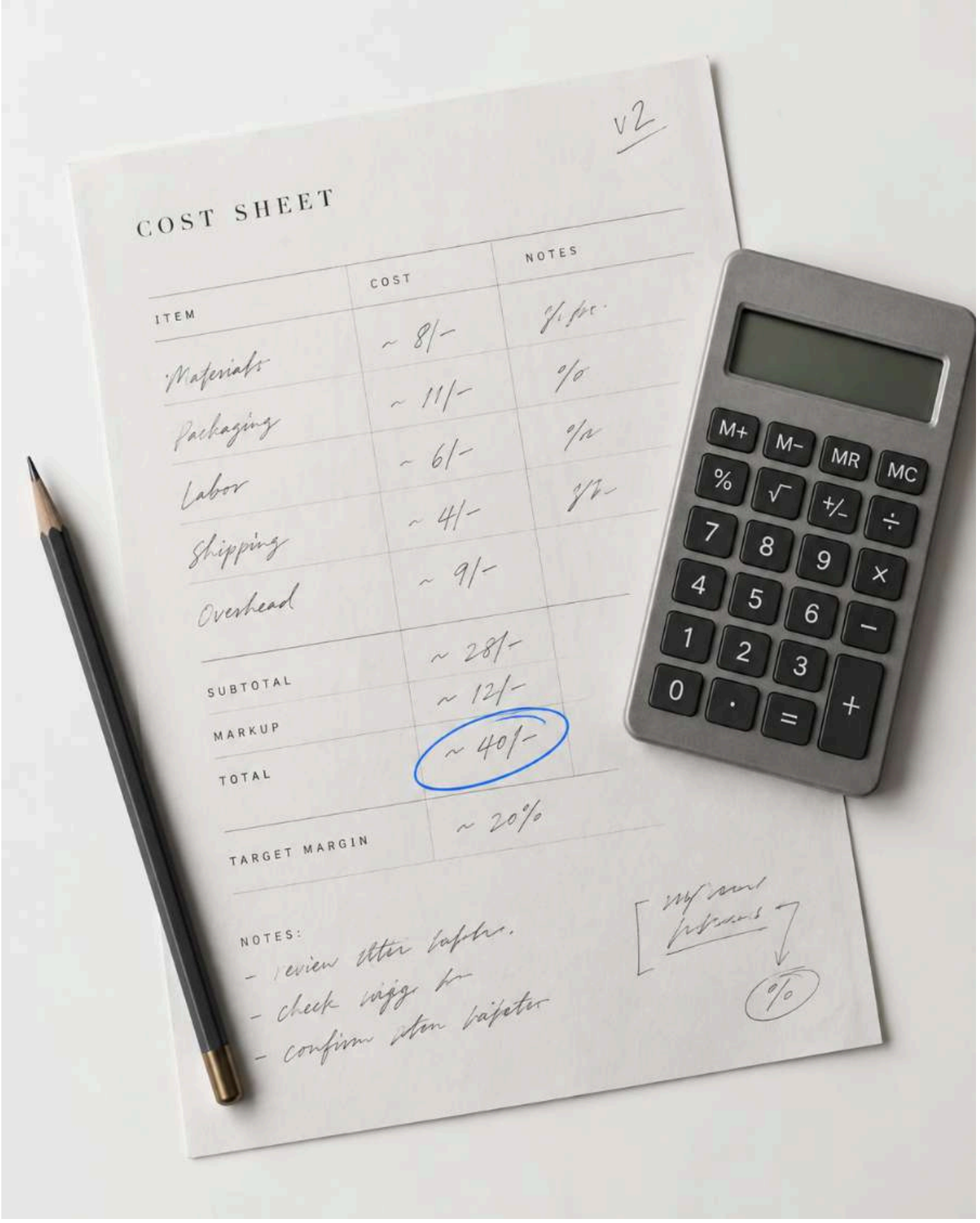
By the time a product gets there, many of the decisions that will limit its trajectory have already been made: cost structure, supply chain, component architecture in the physical world; infrastructure, data model, acquisition, pricing in the digital one.

Launch doesn't open the product. It exposes it. It's the moment when accumulated hypotheses meet users, operations and real economics all at once. Stage-Gate describes launch as the step into operations or full-scale production; in Ulrich and Eppinger's model,

physical development doesn’t end at detailed design either — it includes testing, refinement, and production ramp-up.

The digital equivalent is obvious. A product that works with a hundred users may not work with a hundred thousand. An operation that manually handles ten exceptions can collapse under ten thousand. Before launch we try to answer whether we should and can build. Afterward we try to discover whether we can sustain, scale and evolve what we built.

The profitability design shouldn’t outsource



Design is a negotiation between variables — including the economic ones.

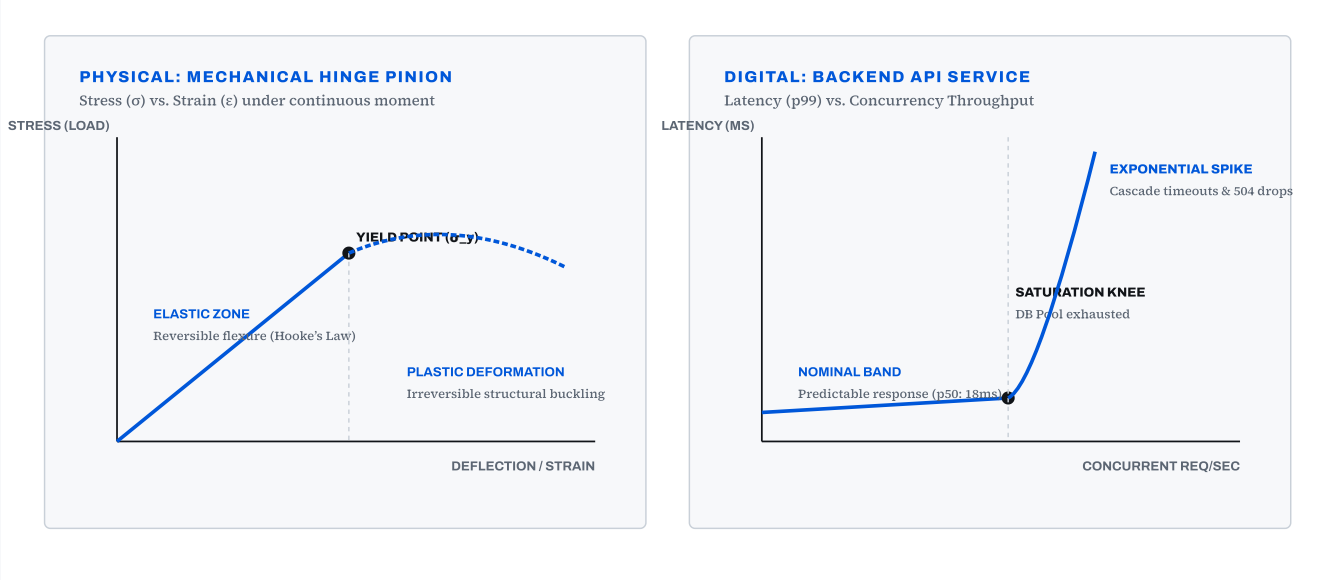
One of the intersections I find most interesting between industrial design and digital product is economics. My experience is that economics is part of the design material: if a product requires a heavy tooling investment, we need a volume capable of amortizing it; if a complex part can become three simple parts, we might increase assembly work but reduce waste or maintenance.

The same is true in digital. We can offer a free tier to acquire users and charge for advanced capabilities; we can charge per seat and later discover value actually depends on transaction volume; we can combine subscription and consumption. There's no universal model. What does exist is an unavoidable relationship between how we capture value and what product we end up building.

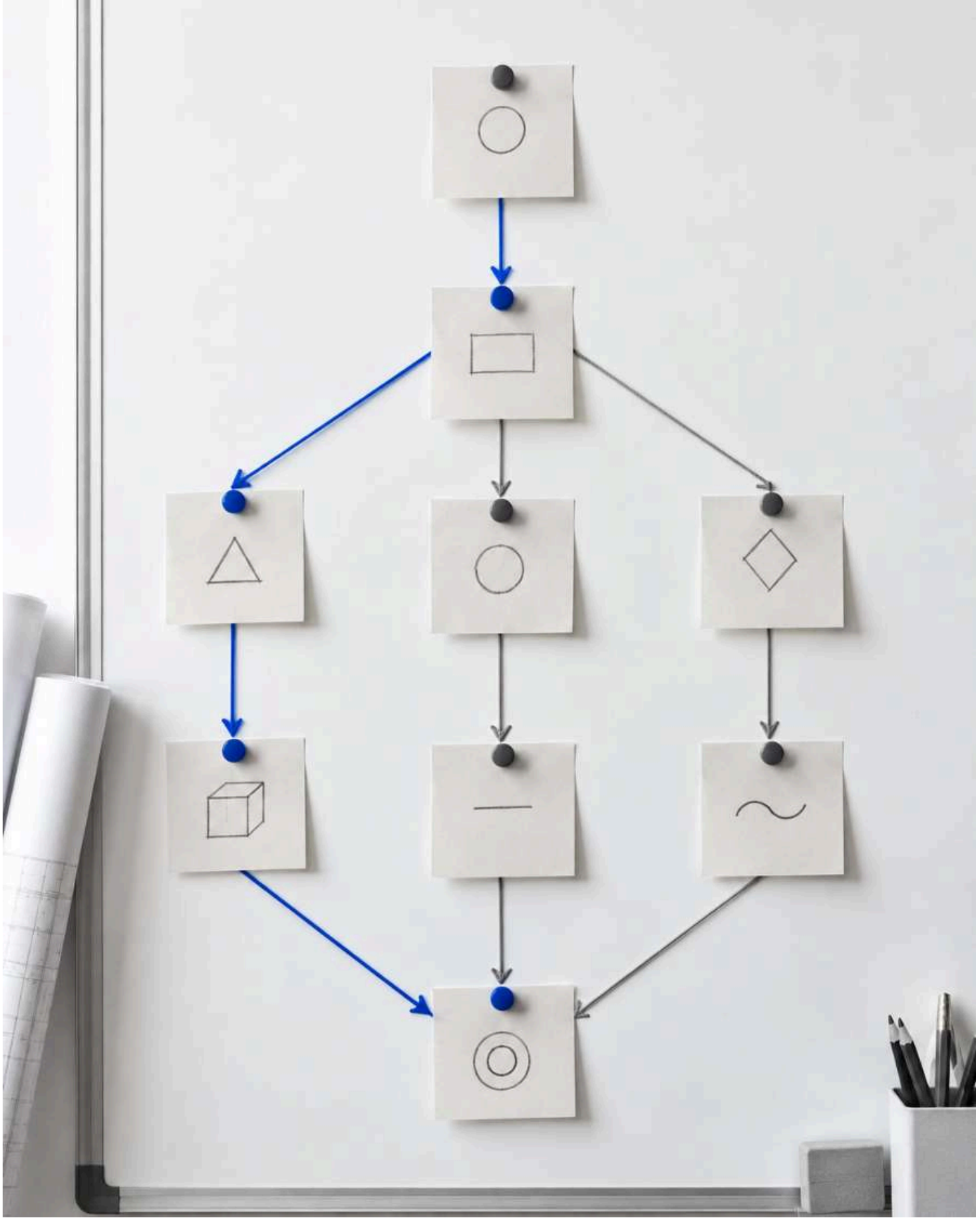
That's why a monetization plan shouldn't be a slide for investors. It should become a set of explicit hypotheses: who pays, why they pay, when they pay, why they'd keep paying, and which costs grow as usage grows. can

In a physical product, the sale makes the economic exchange visible immediately. In digital, we can accumulate usage for months before discovering whether behavior translates into sufficient revenue. A user is not yet a business model. Neither is a download. And a viral idea

Physical yield, digital saturation



Product and project: related, not identical



Managing product means knowing where you need control and where you need to learn.

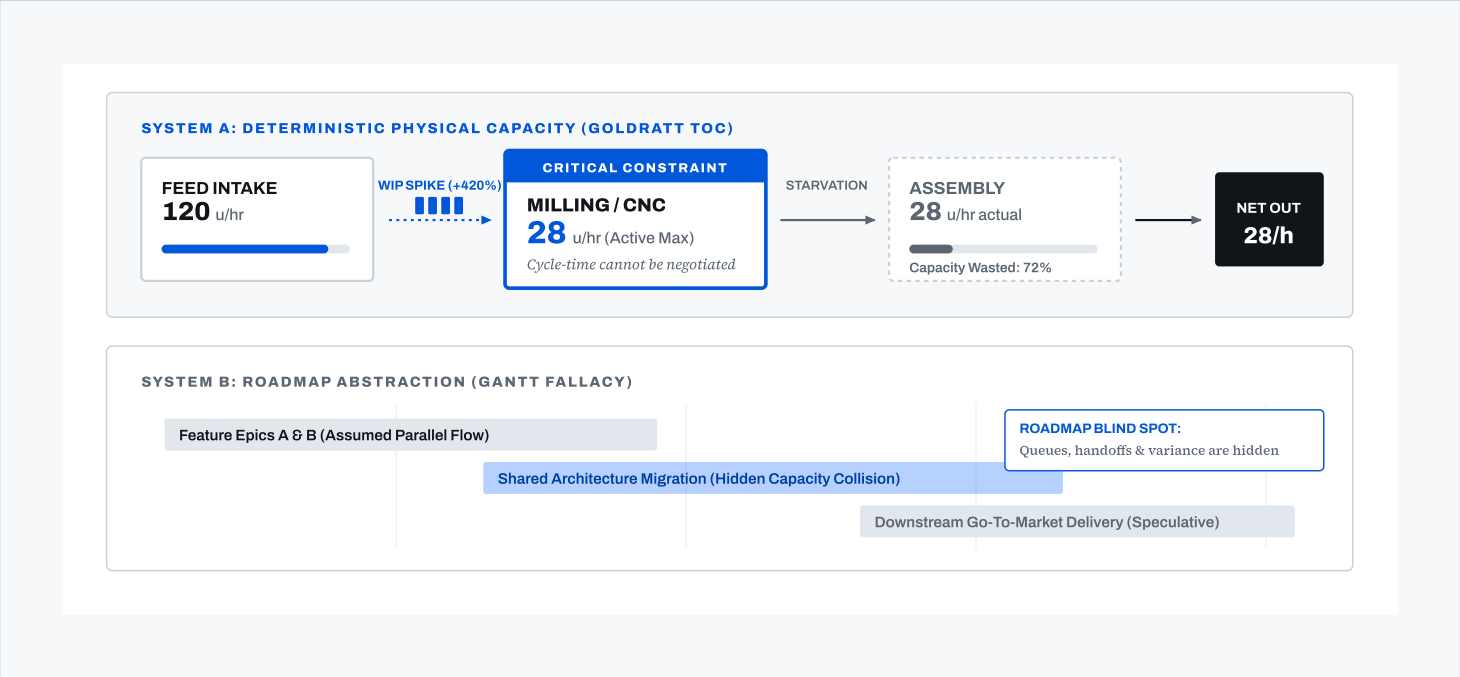
A project exists to produce a result within a specific combination of scope, time, resources, dependencies and risk. A product, by contrast, can move through dozens of projects over its life: launching it might take one, migrating its architecture another, entering a different market a third.

Research by Jetter, Albar and Sperry for PMI studied how project-management practices and product development complement each other in technology environments. Their conclusion wasn't that traditional control should be imposed on innovation, but that management mechanisms need to adapt to the level of uncertainty — a distinction far more useful than the old Agile-versus-Waterfall debate.

Not every element of a product carries the same level of uncertainty. I can have a fully deterministic contractual deadline and a technical solution that's still open. I can know the technology but have no certainty about adoption. Managing product also means knowing where you need control and where you need to learn.

In industrial design this becomes obvious on the path to production: some elements have to be frozen before investing in tooling or scale. In software we have more freedom to change after launch, but that freedom isn't infinite — migrating data, breaking compatibility, or replacing core components can also be extraordinarily expensive.

The visible constraint, the roadmap that hides it



What an 1859 chair still teaches

Its value wasn't only in the finished object, but in the system able to repeat it.

The Thonet chair condenses nearly the entire chapter. It didn't begin as just an attractive shape: it needed a mature manufacturing technology, an architecture of few pieces, a joining method, repeatability, a logistics strategy. And afterward it needed the ability to use shared principles and components to produce variations without starting from zero.

In digital product we often obsess over the visible launch: the app, the screen, the new feature. But the products capable of surviving usually depend on something less visible: an architecture that can evolve, an economy that can sustain itself, an operation that can repeat, and an organization that can learn faster than it accumulates complexity.

Both start from imperfectly known needs. Both turn ideas into specifications. Both decide on an architecture. Both face technical and economic constraints. Both need prototypes, need to solve production or operations, and reach a market that might ignore them. Both eventually disappear if the value they produce stops justifying the cost of maintaining them.

The biggest difference probably isn't in the nature of the product but in the speed and cost with which we can correct certain decisions. Software lets us modify a lot of things after launch — that advantage is huge. It can also make us careless. When change looks cheap, we postpone structural decisions and pile up temporary fixes, until we discover that digital tooling exists too: contracts, data, integrations,

Four steps to name a tolerance

A short method, used throughout this book, applied here for the first time to a single product decision.

1

Name the constraint

State the real limit — cost, latency, headcount — in one sentence.

2

Write the number

Assign a measurable value, even a rough one, not an adjective.

3

Assign an owner

A person or team who answers when it's exceeded.

4

Verify before launch

Check against the number, not against how the demo felt.

“Making an idea able to exist more than once.”

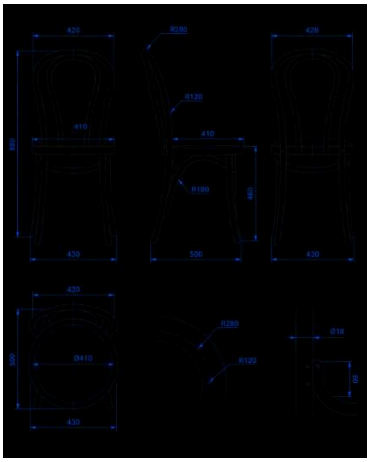
— J. A. Sosa

When I think about the products I’ve watched get built over the past several years — physical, digital, platforms, enterprise capabilities — I find myself caring less and less about the original idea and more about the chain of decisions that turned it into something sustainable. The idea matters. Nothing starts without it. But it’s rarely the scarce part.

What’s scarce is finding a need important enough, translating it into a clear proposal, proving we can build it, designing an architecture that doesn’t condemn us too soon, finding an economy capable of sustaining it, coordinating the people needed to bring it to market, and keeping enough flexibility to accept that some of our first assumptions will be wrong. That’s where my 10/90 comes from — not as an academic formula but as a personal reminder that initial enthusiasm is a very small fraction of the road.

The industrial designer runs that negotiation with materials, processes, suppliers, cost and manufacturing. The digital product leader runs it with technology, data, behavior, operations and economics. Both are trying to solve the same fundamental problem.

A prototype proves something can exist once. A product proves we can repeat the value. And a well-designed system lets that product change without having to be reinvented whole. That’s where the next chapter begins.



Transparent object space: contain, never cover — the full diagram stays visible regardless of container width.

REFERENCES

1. Ulrich, K., Eppinger, S., Yang, M. — *Product Design and Development.*
2. Cooper, R. — Stage-Gate new-product development practices.
3. Teece, D. — Definition of the business model as the architecture of value creation, delivery and capture.
4. Thonet — Historical documentation of the No. 14 / model 214 chair, bentwood technique and packing method.
5. Dahmus, J., Gonzalez-Zugasti, J., Otto, K. — Research on platform architecture and modular product families.
6. Robertson, D., Ulrich, K. — Planning for Product Platforms.
7. U.S. Government Accountability Office — Commercial evolutionary-development and risk-reduction practices.
8. Levitt, T. (1965) — Exploit the Product Life Cycle.
9. Jetter, A., Albar, F., Sperry, R. — PMI research on project management and product development in technology environments.
10. Author's field notes on program management, retail POS and enterprise portfolios, 2016–2026.

PART I — UNDERSTANDING THE PRODUCT · CHAPTER 02

Product Strategy, Materials and the Cost of a Decision

Every product begins as a promise made before enough is known. This chapter is about what that promise costs to keep.

Published Sep 16, 2026 14 min read Edition 0.2 Status: Published

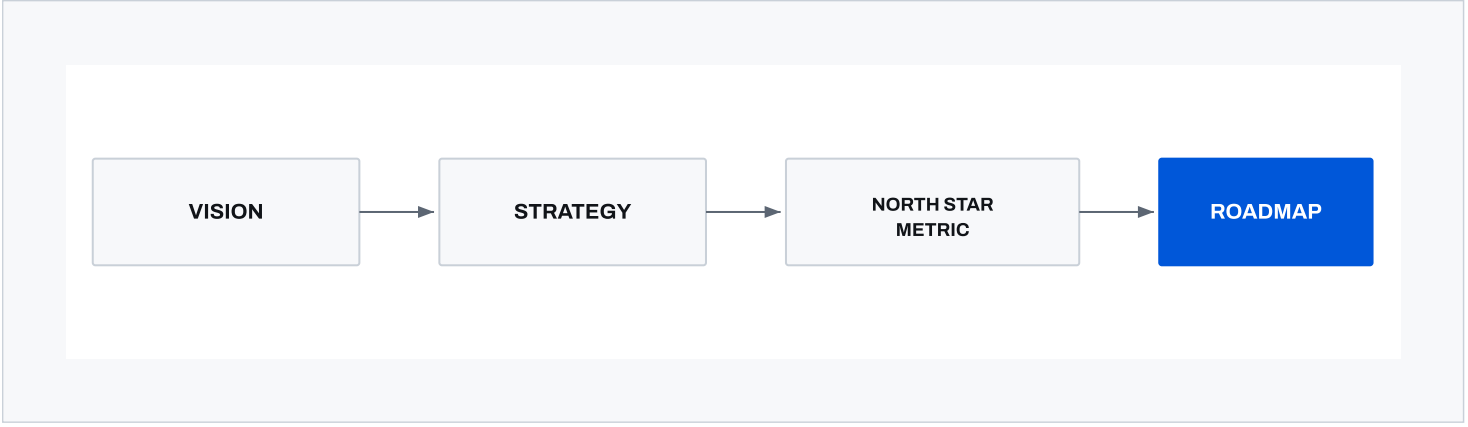
A physical product may begin with a sketch: a surface, a structure, a proposed relationship with the human body. A digital product begins with a vision of change — a customer who should be able to do something faster, more safely, or with less effort than before. Neither begins with a roadmap. A roadmap is already an act of translation, arriving only after the organization has chosen a direction.

Before it can exist, the organization must define its Product Vision, choose a Product Strategy and decide how delivered value will be recognized. A North Star Metric gives that value a measurable direction, and only then can a sequence of product bets be organized over time. At this point the product still exists mostly as intent — the team may understand the problem without knowing the real cost of solving it.

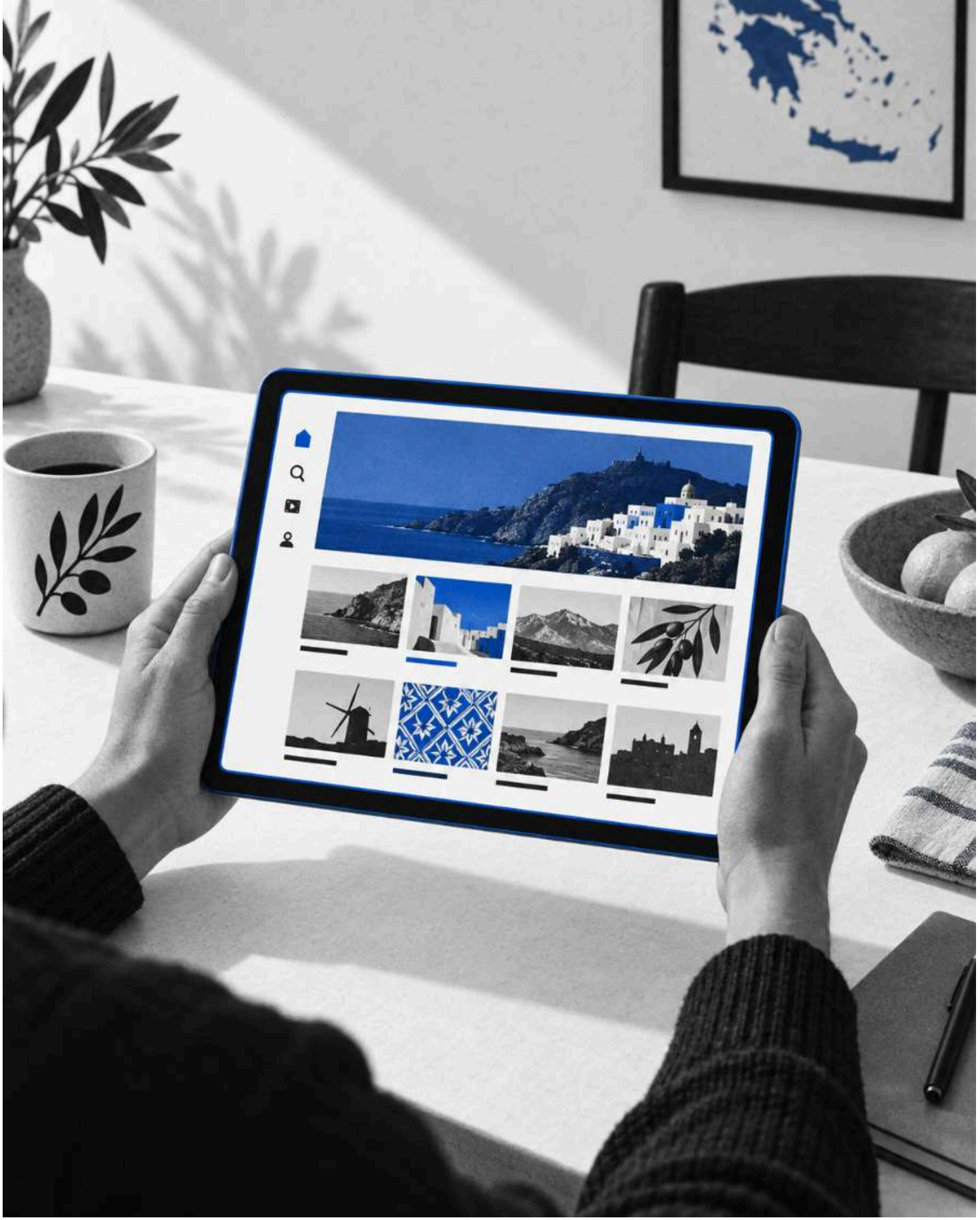
This is where feasibility begins — not the narrow question of whether something can be built, but an examination of whether it can be produced, operated, maintained and adapted under conditions the organization and the market can sustain. A technically possible solution is not automatically a feasible product. The rest of this chapter asks what that difference actually costs.

4	6	80%
LENSES A PRODUCT MUST PASS THROUGH	COST FAMILIES FROM BUILD TO RETIREMENT	TYPICAL MATERIAL YIELD IN THE WORKED EXAMPLE

Direction precedes planning



What a Product Strategy Actually Commits To



Choosing a regional audience over global reach — a strategy is as much about what a product will not do.

A Product Vision states a direction of change. A Product Strategy decides how the organization will get there under real constraints — which customers it will serve first, which problems it will refuse to solve, and how it intends to win against alternatives. Vision points; strategy commits.

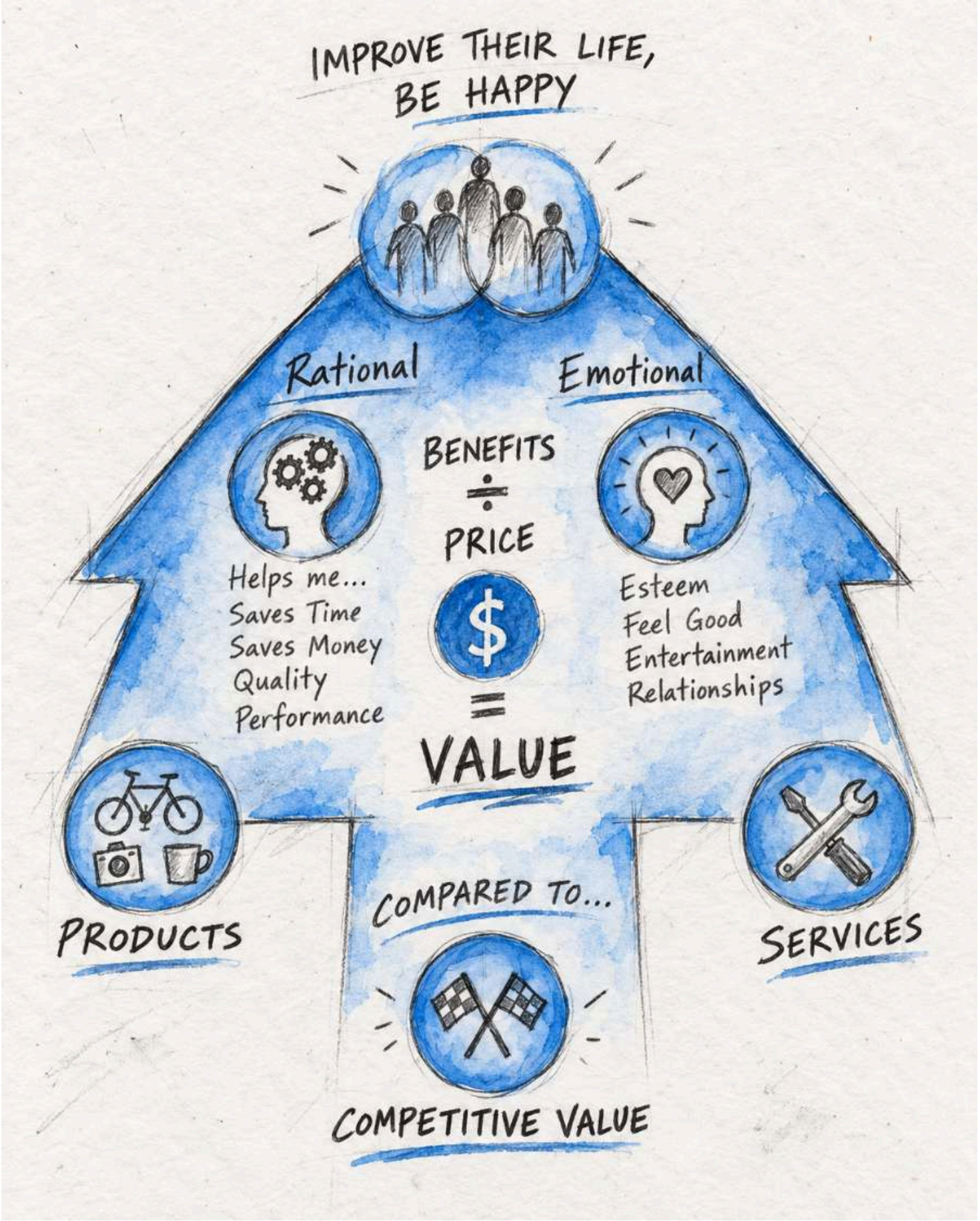
Consider a fictional streaming product called Northline. Its vision is to give families immediate access to relevant regional stories on any connected device. That vision alone says nothing about how Northline competes with services that already have larger catalogues and larger budgets.

Northline's strategy narrows the vision into a bet: serve a specific regional audience through a curated, locally produced catalogue, rather than compete on the sheer volume of global content. It deliberately trades reach for relevance and a lower cost of content acquisition.

A strategy is therefore as much about what a product will not do as about what it will. Northline will not license international blockbusters, will not attempt every device profile in its first year, and will not chase every geography a global competitor already owns.

This is the strategy the rest of the chapter tests against reality — because a strategy that cannot be produced, operated and paid for is not yet a strategy. It is a preference.

The Customer Value Equation



Benefits up, price down, value grows — the same wedge that reshaped the smartphone market in 2007.

People buy a product because it improves their life or their business more than the alternatives available to them. That comparison is never made in isolation — it is always made against whatever else the customer could choose instead, including doing nothing.

A simple way to hold this in mind is a value equation: a product's value equals its benefits, rational and emotional, relative to its price, judged against competitors. A product that improves this ratio faster than its competitors tends to keep the customers it has and attract new ones.

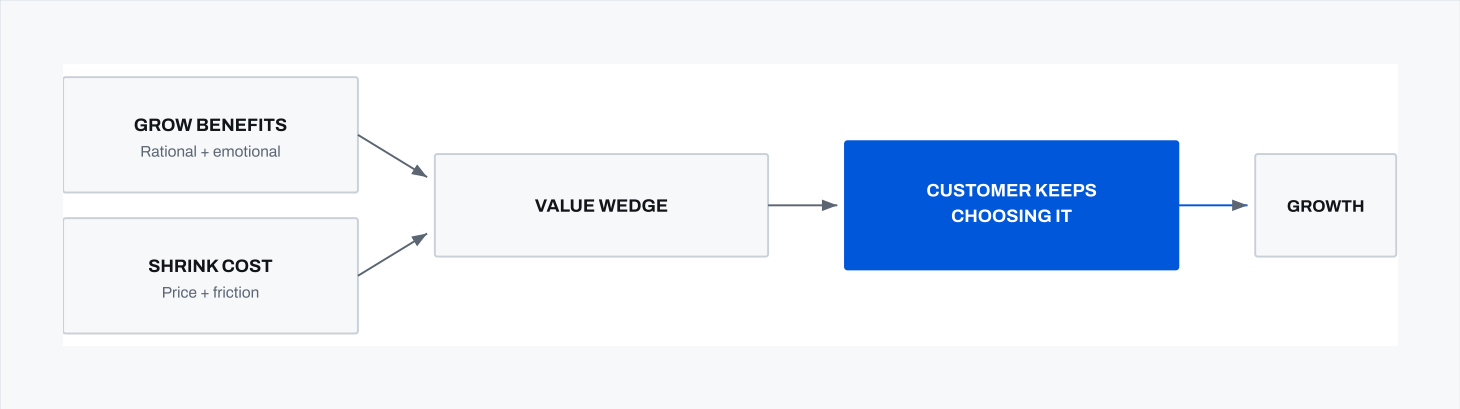
> *Value = (Rational Benefit + Emotional Benefit) ÷ Price — relative to the competitive set*

For Northline, the rational benefit is straightforward: content a household cannot easily find on a larger, more generic platform. The emotional benefit is closer to identity — seeing a region's own stories told on its own terms. Neither benefit matters if the price, in money or in friction, erases the advantage.

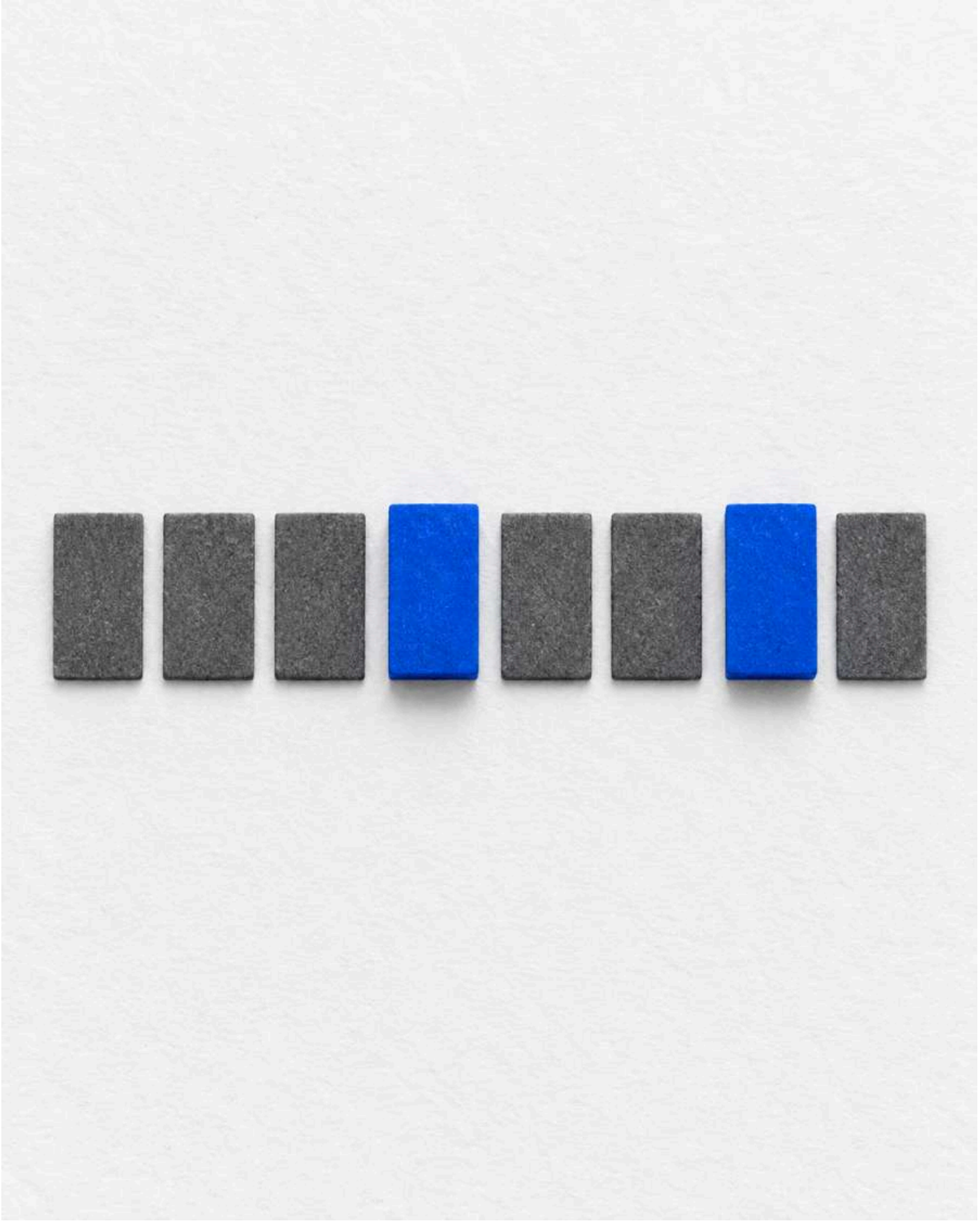
A product strategy therefore has two levers, not one: grow the benefits customers actually notice, or shrink the cost of delivering them. Improving both at once — better content, cheaper delivery — is what turns a narrow regional bet into a durable margin rather than a one-time novelty.

This is the same value wedge that reshaped the smartphone market in 2007, when a new entrant stopped competing on battery life and messaging speed and instead removed the unmet friction of using the internet from a phone at all. The lesson was never the device — it

The value wedge



Which Dimensions Actually Compete



Picking two dimensions, not eight — tightening what customers barely notice is expensive over-specification.

Every product competes on a shortlist of dimensions: features, usability, performance, durability, reliability, serviceability, conformance and aesthetics. Not every dimension matters equally to every customer, and treating all eight as equally important is itself a strategic mistake.

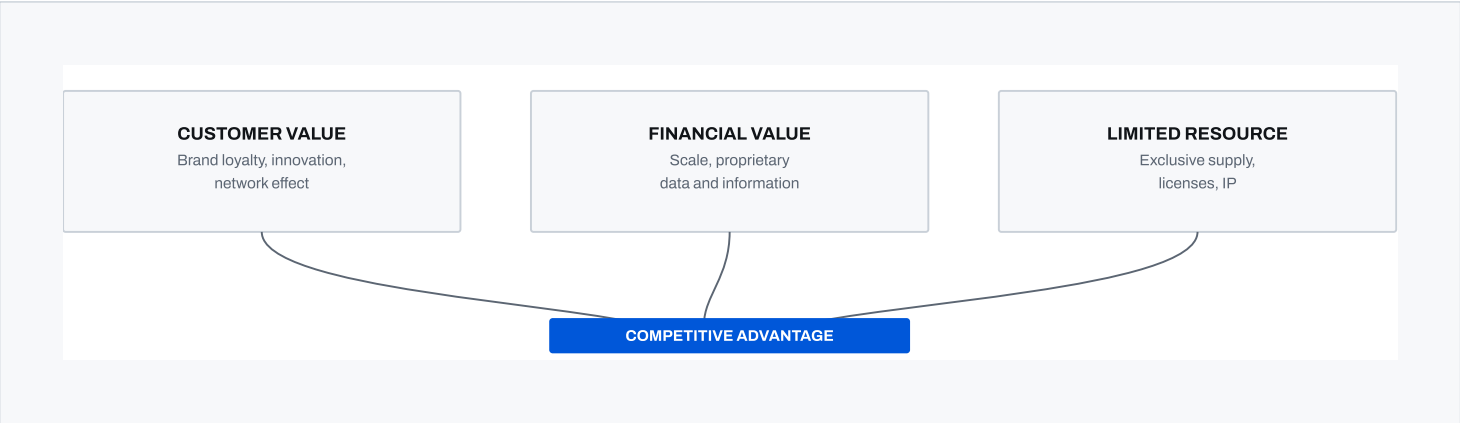
A regional streaming product like Northline gains little by competing with a global catalogue on raw feature count. Its real contest is on usability — how fast a household finds something worth watching — and on reliability — whether the stream that starts on a Tuesday evening keeps playing.

Durability and serviceability, dimensions that matter enormously for a physical product, barely register for a streaming service; conformance to a regional content rating system, by contrast, may matter more to Northline's audience than to a global competitor's.

This is the same idea introduced earlier as tolerance, applied one level up: a strategy does not just set a tolerance for each dimension, it first decides which dimensions deserve a tolerance at all. Tightening a dimension customers barely notice is the strategic equivalent of over-specifying a bolt no one will ever inspect.

Deciding which two or three dimensions to compete on — and which the product can afford to be merely adequate at — is one of the clearest outputs a product strategy can produce.

Sources of competitive advantage



Sustainable advantage tends to come from one of three kinds of barrier: what customers value, what the balance sheet can fund, or what a competitor simply cannot access.

Four Steps to a Product Roadmap

A roadmap is not a wish list in date order. It is the output of a disciplined narrowing, from insight to sequence.

1

Generate Insights

Product, market and customer signals.

2

Develop Opportunities

Use cases, features, a snapshot.

3

Prioritize

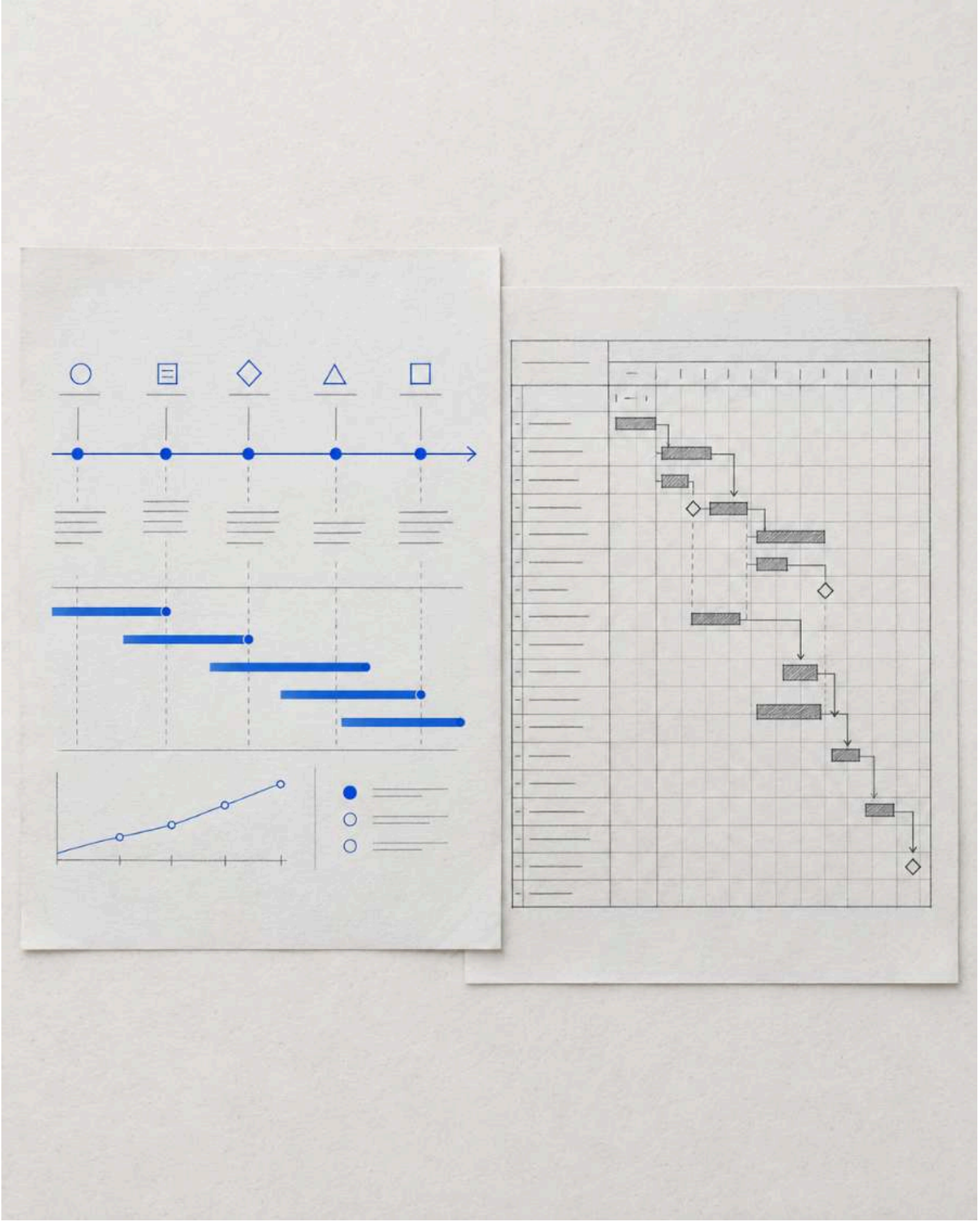
Value against effort, on one matrix.

4

Build the Roadmap

Sequence bets as a living document.

Two Outputs: the What and the How



A roadmap and a delivery plan, side by side — negotiated together, not handed off.

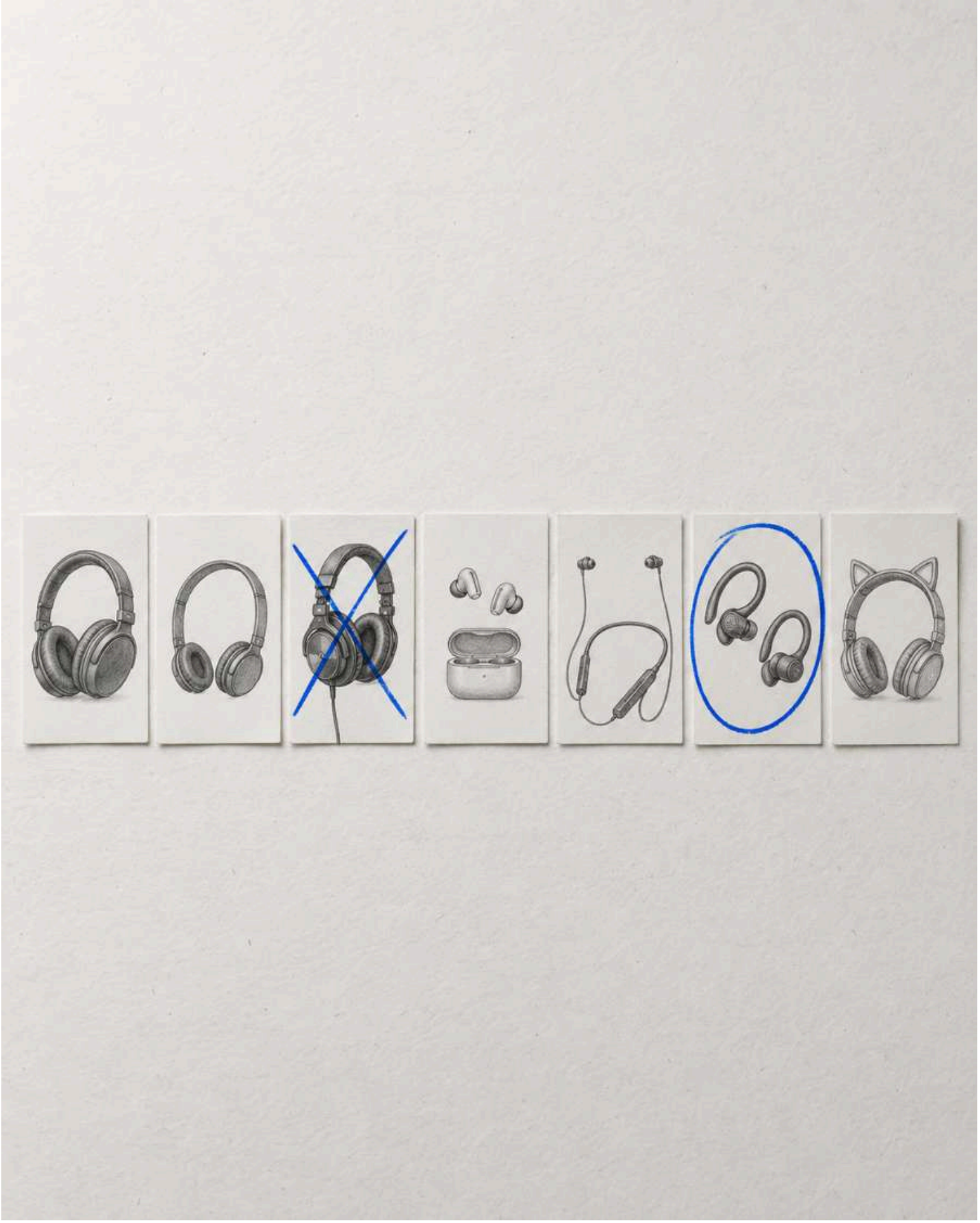
A product strategy produces two distinct documents, and conflating them is a common source of confusion. The roadmap is the what — the sequence of product bets the organization has chosen to make. The product development lifecycle strategy is the how — the goals and initiatives that determine whether the organization can actually deliver that sequence.

The lifecycle strategy itself has three parts: effectiveness goals, which describe the outcomes the roadmap should produce; efficiency goals, which describe what it should cost in time and resources to produce them; and the initiatives needed to close the gap between current and target performance on both.

For Northline, an effectiveness goal already exists in this chapter: weekly hours of content successfully reproduced by active households. An efficiency goal exists too: the cost per successfully streamed hour. A roadmap without both is just a list of features; goals without a roadmap are just aspirations.

This is why a credible product strategy cannot be written by the product team alone, sealed, and handed to engineering and finance afterward. The roadmap and the lifecycle strategy have to be negotiated together, because a roadmap that ignores delivery cost is not a strategy — it is the upper-left quadrant from the decision quadrant below, unexamined.

Rationalize, Improve, or Build



A portfolio, edited on purpose — rationalizing is the least comfortable of the three moves, and the most often skipped.

Every product portfolio decision reduces to one of three moves: improve what already exists, build something new, or rationalize — retire what no longer earns its cost.

An improvement should differentiate the product from competitors, drive better customer value, or open a new use case or segment. A new product should change the game, help sell more of the current lineup, or fill a real hole in the portfolio, not a hypothetical one.

Rationalization is the least comfortable of the three, and the most often skipped. Retiring a low-value feature or an underused content tier frees the budget and attention a genuinely new bet requires — the same sunk-cost trap the decision quadrant below was built to expose.

For Northline, this might mean retiring a device profile with negligible usage, improving discovery for its core regional catalogue rather than chasing catalogue size, and building exactly one new capability — localized recommendations — that a global competitor has no strategic reason to build first.

A Product Must Survive Four Questions

A feasibility review in progress — no single lens can compensate indefinitely for the absence of another.

An engineer may confirm that a chair can be fabricated from a specific steel section. A software architect may demonstrate that a platform can process the expected transactions. Neither answer establishes whether customers want the product, whether the organization can produce it consistently, or whether its economics will remain sustainable after launch.

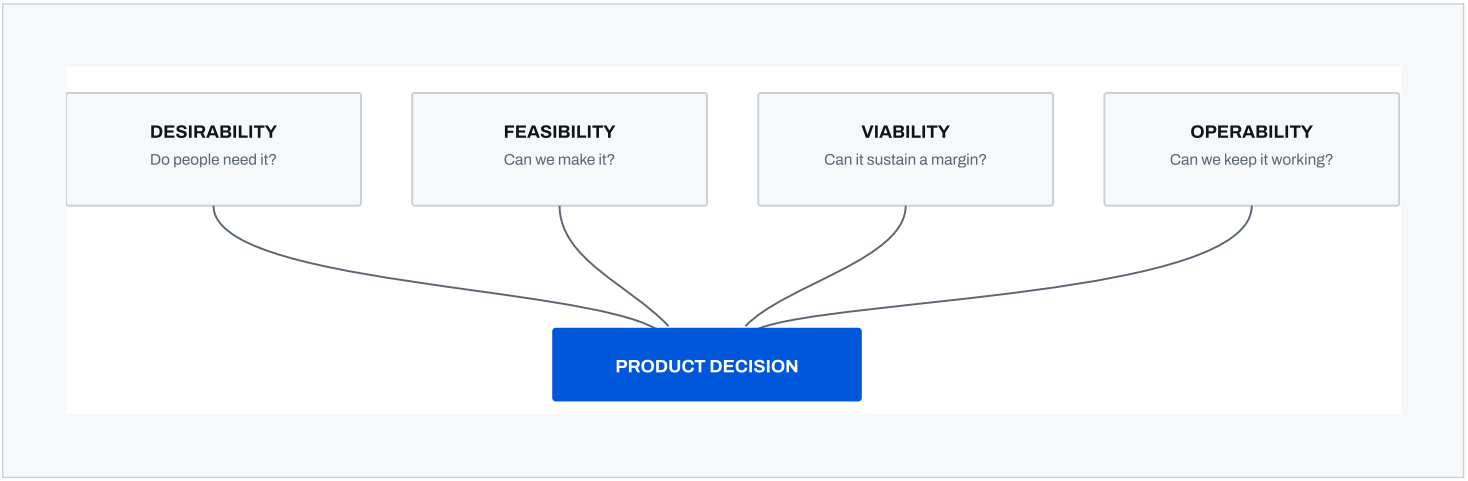
The familiar design-thinking model evaluates innovation through desirability, feasibility and viability. Desirability asks whether the solution matters to people. Feasibility examines whether it can be produced with the available technology and capabilities. Viability considers whether it can survive economically.

For products intended to live beyond their first release, a fourth lens is necessary: operability. Operability asks whether the product can be supplied, monitored, repaired, supported and eventually retired without destroying the value it was designed to create.

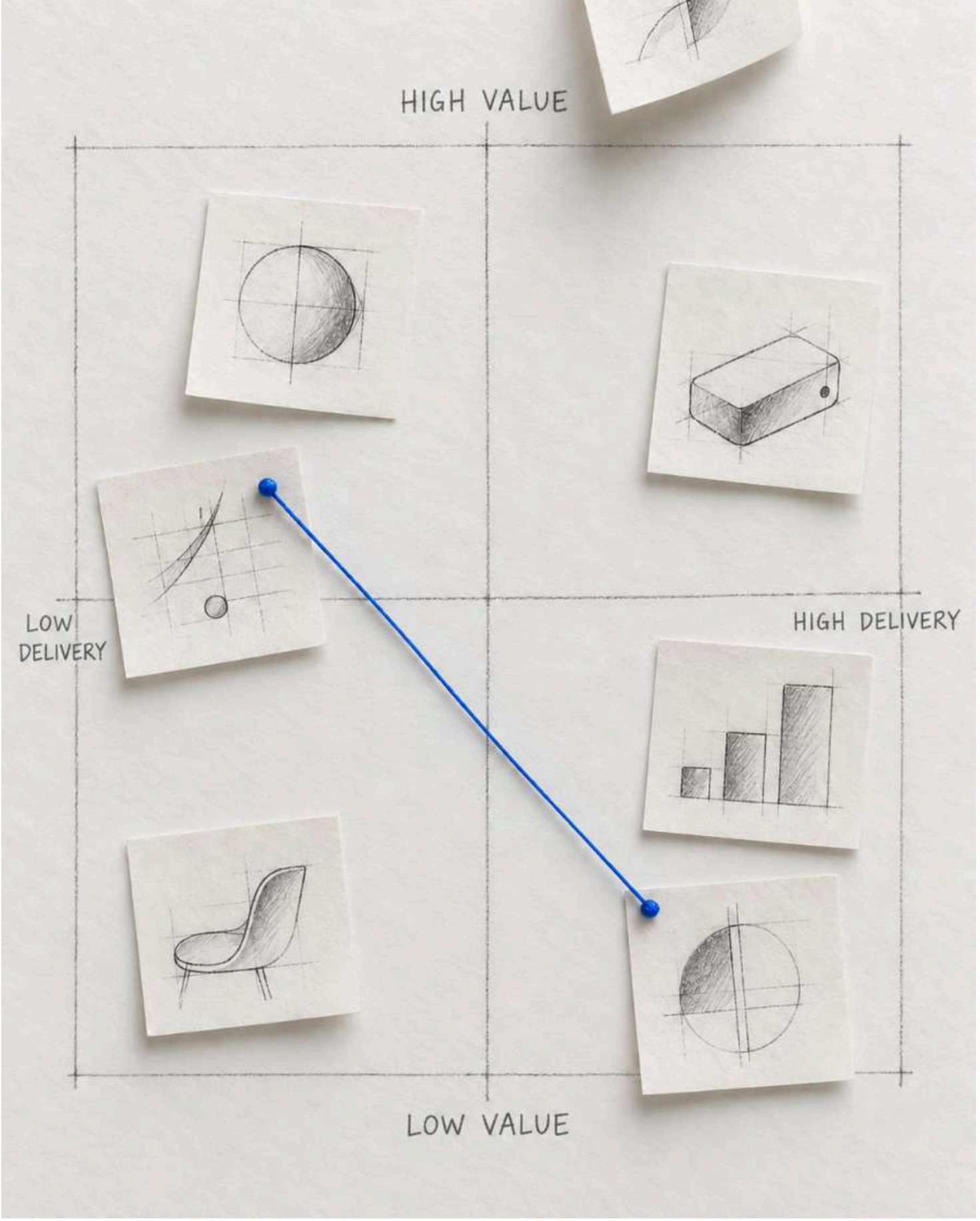
These questions are connected. A desirable product may be impossible to manufacture at the expected price. A technically feasible platform may require operating costs greater than the revenue it generates. A profitable first release may accumulate maintenance obligations that make each later improvement slower and more expensive.

Feasibility is therefore not a gate crossed once. It is a condition examined again as the product, its market and its production system change.

The four lenses of product feasibility



From Direction to an Investment Decision



Sorting ideas by value and difficulty — a living decision instrument, not a ceremonial workshop artifact.

Before calculating detailed costs, the team needs to decide which product assumptions deserve investment. A simple decision quadrant can compare expected value with total delivery difficulty.

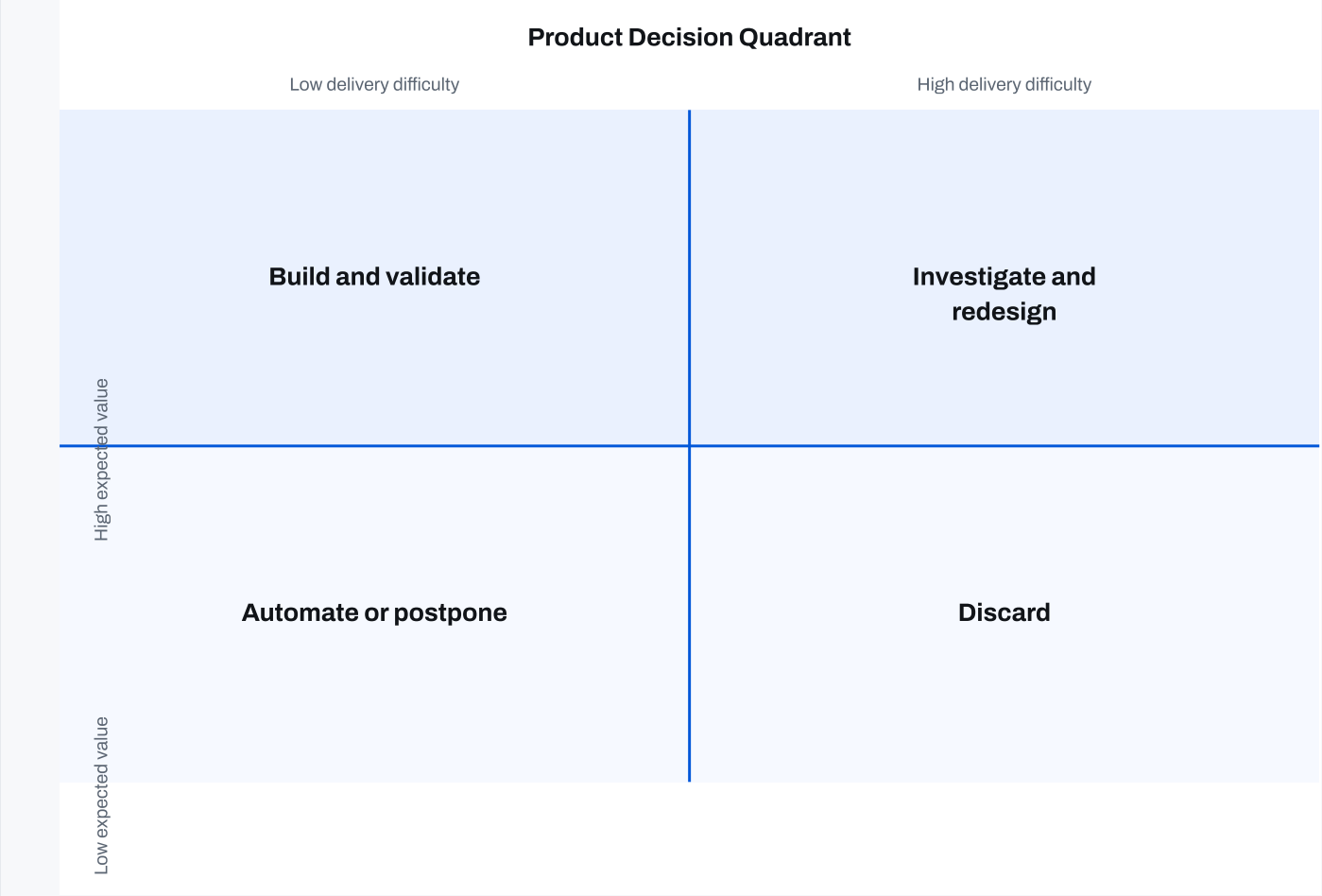
Expected value includes customer benefit, strategic contribution and economic potential. Difficulty includes cost, time, technical uncertainty, dependencies and operational risk. The quadrant is not a mathematical proof of viability — its purpose is to expose the relationship between benefit and effort.

An idea in the upper-left combines high expected value with low difficulty and is a strong candidate for validation. An idea in the upper-right may still be valuable, but the team should divide it, investigate its largest uncertainties, or search for another way to produce the result.

The lower-right area deserves particular attention. Organizations often continue funding expensive, low-value features because work has already begun; the quadrant makes visible what sunk-cost reasoning attempts to hide.

The position of an idea changes as evidence appears. Research may raise its expected value, a prototype may reduce technical uncertainty, and a supplier quotation or a load test may reveal that the original production model is not viable. The diagram is a living

Product decision quadrant



High-value ideas are not automatically approved; difficult ideas require stronger evidence, decomposition or a different production model.

The Anatomy of Cost

Six families, one shared language — development, production, operation, storage, maintenance, retirement.

Once a product hypothesis deserves further exploration, its cost must be decomposed. A first model can use six families: development, production, operation, storage, maintenance and retirement.

These categories apply to both physical and digital products. The objects inside them change, but their economic function remains surprisingly similar — a fact the table below makes explicit.

TABLE 2.1 — A SHARED COST LANGUAGE

Cost family	Physical product	Digital product or service
Development	Research, prototypes, engineering, tooling	Discovery, UX, architecture, development, testing
Production	Materials, labor, assembly, quality control	Compute, API execution, data processing, AI inference
Operation	Energy, machinery, supervision, distribution	Cloud services, security, observability, support
Storage	Raw material, work in progress, inventory	Media, databases, logs, backups, inactive data
Maintenance	Repairs, spare parts, tooling replacement	Corrective work, upgrades, technical debt, incidents
Retirement	Returns, disposal, recycling	Migration, archival, data deletion, decommissioning

This decomposition matters because early estimates frequently include development and omit the rest of the product's life. A prototype can demonstrate that a concept works; it rarely demonstrates what it will cost to keep working for five years.

Materials Are Visible Decisions

Offcuts, yield and the true cost of a sheet of material — the purchase price is only the beginning.

In a physical product, material cost appears tangible — wood, steel, leather, adhesives and finishes can be measured, weighed and quoted. Yet purchase price is only the beginning.

Material selection also affects waste, machining time, energy consumption, defect rates, packaging, transportation and maintenance. If one hundred dollars of material produces only eighty dollars of usable output after cutting and defects, the product does not have a material cost of one hundred.

rath

$$C_{usable\ material} = C_{purchased\ material} \div yield$$

At an 80% yield, one hundred dollars of purchased material becomes an effective usable-material cost of 125 dollars.

A designer can change this relationship before purchasing negotiations begin. Reducing the number of parts, changing a cutting pattern, standardizing a section, or designing several components from the same stock may save more than a small supplier discount.

Digital products consume materials too, although they do not remain visible in the final interface. Computing capacity, network transfer, storage, licensed data, third-party APIs and model tokens are transformed into a service — the material simply arrives through a meter

Tolerance Is an Economic Variable

A tolerance stack, drawn to scale — precision creates value in some places and merely cost in others.

A tolerance defines how much variation the product can accept and still perform its intended function. In manufacturing it may describe a dimension, an angle, surface roughness, color variation or material strength.

Tighter tolerances normally require more precise equipment, additional inspection, better process control and higher rejection costs. Digital products have tolerances too — maximum response time, acceptable error rate, service availability, recovery time after failure, data freshness, synchronization delay and result accuracy.

A requirement for 99.99% availability is not simply a decimal placed in a document. It can imply redundancy, monitoring, automated recovery, more complex deployments and permanent operational capacity.

A tolerance should come from the consequence experienced by the user. A financial transaction and a film recommendation do not require the same certainty; a medical component and a decorative cover should not carry the same dimensional control.

The Barcelona Chair: apparent industrial simplicity did not create a low-cost method of production.

"Cost optimization does not always mean making the cheapest possible product."

— On the Barcelona Chair

The Barcelona Chair is often treated as a symbol of industrial modernity — its crossed steel frame appears reduced, rational and ready for reproduction. Its production tells a more complicated story.

Ludwig Mies van der Rohe and Lilly Reich designed the chair for the German Pavilion at the 1929 Barcelona International Exposition, for a representative setting where the Spanish royal couple would be received: an important, monumental chair, not inexpensive seating for a mass market.

The chair combines a curved metal frame, supporting straps and carefully upholstered leather cushions, assembled from multiple individually treated sections rather than a single mechanically formed surface.

The chair could not compete with ordinary seating through low price or high-volume efficiency. Its viability required a different equation — one the framework below makes explicit.

Place the same chair in a low-cost, high-volume strategy and the product becomes economically incoherent without changing a single line of its form.

Premium Viability, Decomposed

Rather than removing every expensive operation, the Barcelona Chair's business model preserved them and found a market willing to pay for the result.

1

Craft Value

Labor the market chooses to pay for.

2

Authorship

A recognizable design signature.

3

Durability

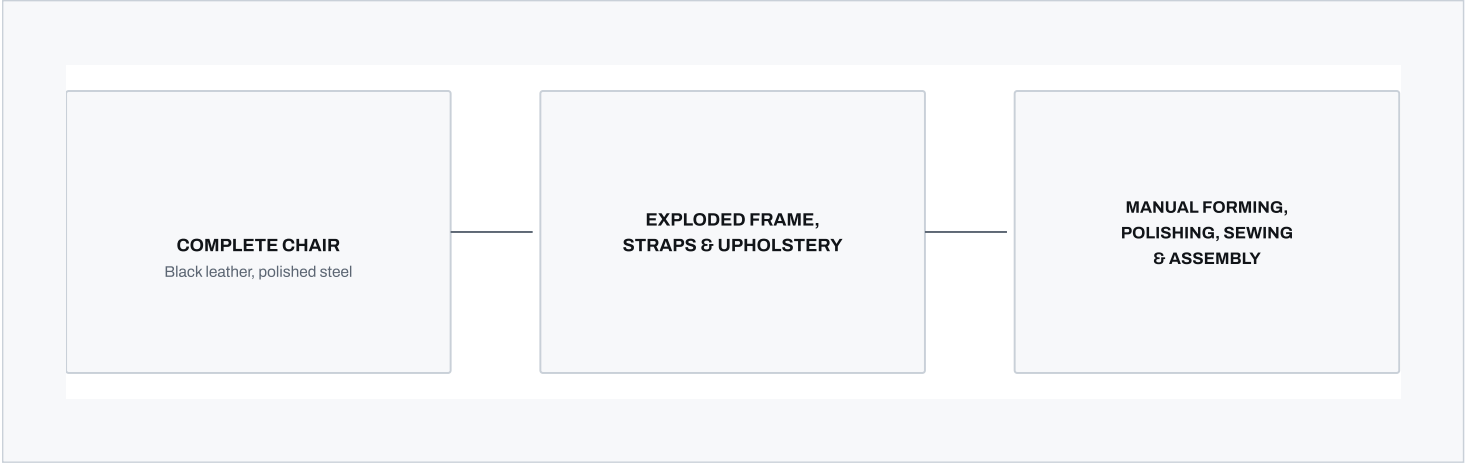
Built to outlast a trend cycle.

4

Cultural Meaning

An object people choose to keep.

Form versus production



Barcelona Chair: form versus production. Apparent industrial simplicity did not create a low-cost method of production.

Labor and the Illusion of Linear Output

A team, not a headcount — hours worked are an input, not finished pieces.

Manufacturing labor can often be connected to observable operations — cutting, bending, welding, sewing, assembling, inspecting. Even there the relationship is not perfectly linear: setup time, batch size, learning, defects and model changes affect the cost per piece.

In digital products the connection between labor and completed units becomes even less direct. Ten developers working for a month do not necessarily produce twice as much valuable software as five — adding people also adds onboarding, communication, integration, review and coordination.

A feature that appears small on the interface may require changes to identity management, data models, cybersecurity controls and several existing integrations. Another may be assembled rapidly from capabilities the organization already owns. Hours worked are an

input, not finished pieces.

Software-cost estimation therefore considers functionality, complexity, criticality, technical conditions and risk — not headcount alone. This is one place where an experienced Product Manager creates measurable value: not certainty, but recognition of what a first calculation omits.

Legacy constraints behind the interface, unavailable or poor-quality data, external approvals, security and regulatory work, operational readiness, adoption, migration and the cost of failure — the Product Manager connects technical uncertainty with customer value and economic consequence.

100K	$0.33	$140K
ACTIVE HOUSEHOLDS IN THE MODEL	COST PER SUCCESSFUL HOUR	ESTIMATED MONTHLY CONTRIBUTION

A Streaming Service and the Cost of One Successful Hour

Northline, a fictional regional streaming product — a vision and a strategy still have to survive contact with a cost structure.

Return to Northline, introduced earlier as a strategy bet on regional relevance over global reach. A vision and a strategy are not yet a business — they still have to survive contact with a cost structure, starting with the metric that will measure whether the bet is working.

Its North Star Metric is not the number of registrations or downloads — neither guarantees customers receive value. A more useful metric is weekly hours of content successfully reproduced by active households.

For Northline, “successfully reproduced” also requires a tolerance: playback must start within the expected time and continue without a failure severe enough to make the viewer abandon it. The metric therefore connects customer experience with technical and economic

performance.

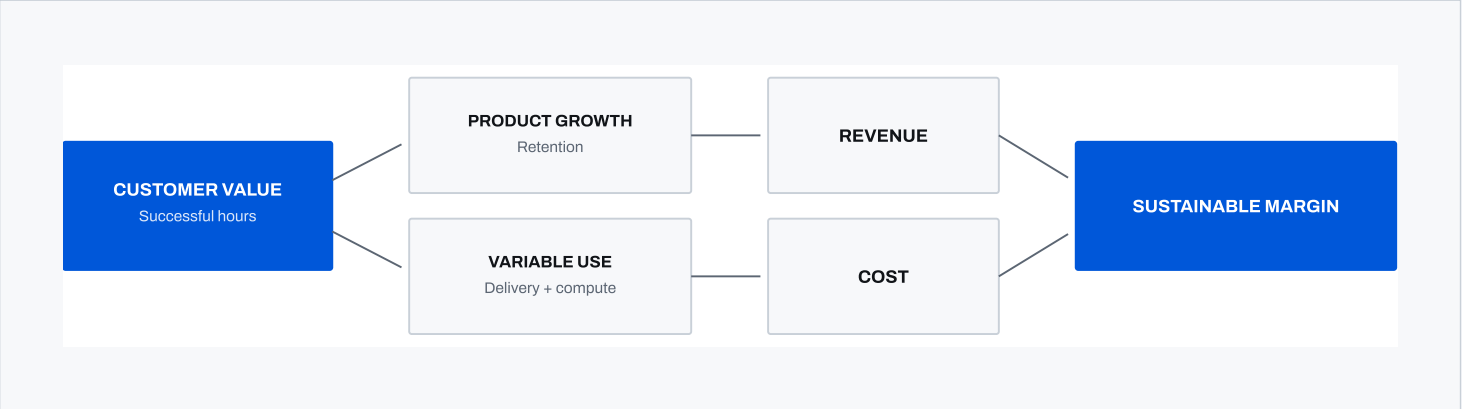
ILLUSTRATIVE MONTHLY SCENARIO

Variable	Assumption
Active households	100,000
Average successful viewing	20 hours
Successfully streamed hours	2,000,000
Subscription revenue per household	$8.00
Monthly revenue	$800,000
Fixed and step-fixed costs	$420,000
Variable cost per streamed hour	$0.12
Total variable cost	$240,000
Estimated contribution	**$140,000**

These numbers do not represent Netflix or a particular cloud provider — they form a transparent model for understanding cost behavior. The estimated monthly contribution is revenue minus fixed and variable costs: 800,000 minus 660,000, or 140,000 dollars. The cost per successfully streamed hour works out to $0.33.

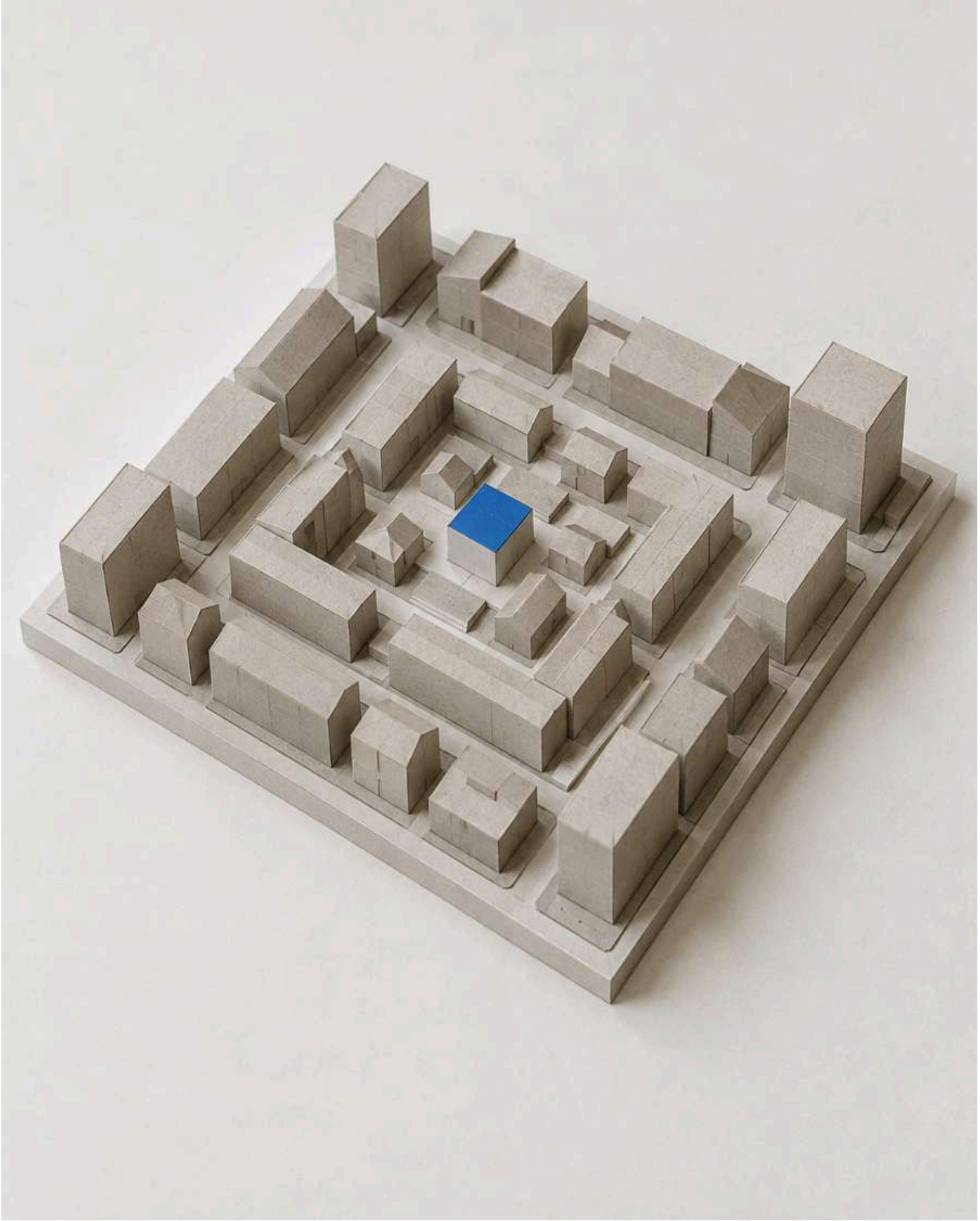
Now imagine viewing rises to thirty hours per household without a price change. Engagement improves, but variable consumption also increases: an additional million viewing hours costs 120,000 dollars more, so the North Star Metric moves upward while the margin moves downward. This does not mean the metric is wrong — it means a North Star must be accompanied by economic guardrails.

A North Star Metric needs economic guardrails



Greater engagement creates value but may also increase the cost to serve.

When Growth Changes the Product



A system redesigned around real demand — architecture is part of the business model.

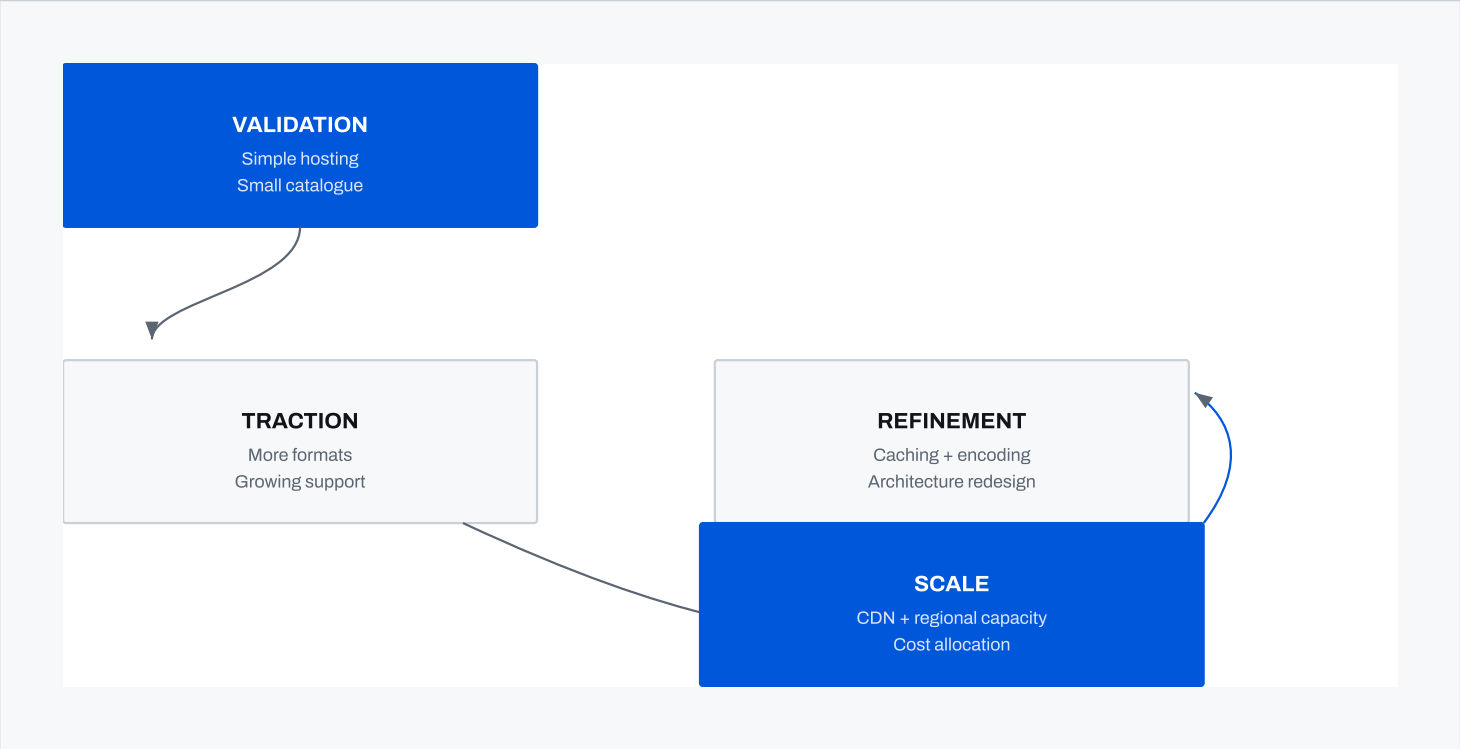
Northline's first architecture might be intentionally simple — a limited catalogue stored in one cloud region, encoded into a few playback formats, delivered through external services. This may be the fastest, least expensive way to test whether the audience exists.

If the product gains traction, its economic conditions change. More users create more concurrent sessions, a larger catalogue increases storage and encoding, and expansion to new countries adds content rights, regional infrastructure, support complexity and new device profiles.

Large streaming platforms bring content closer to viewers because repeated long-distance delivery is expensive and can damage performance. The specific architecture is not universal — the principle is that as demand becomes observable, the production system can be redesigned around actual patterns rather than initial assumptions.

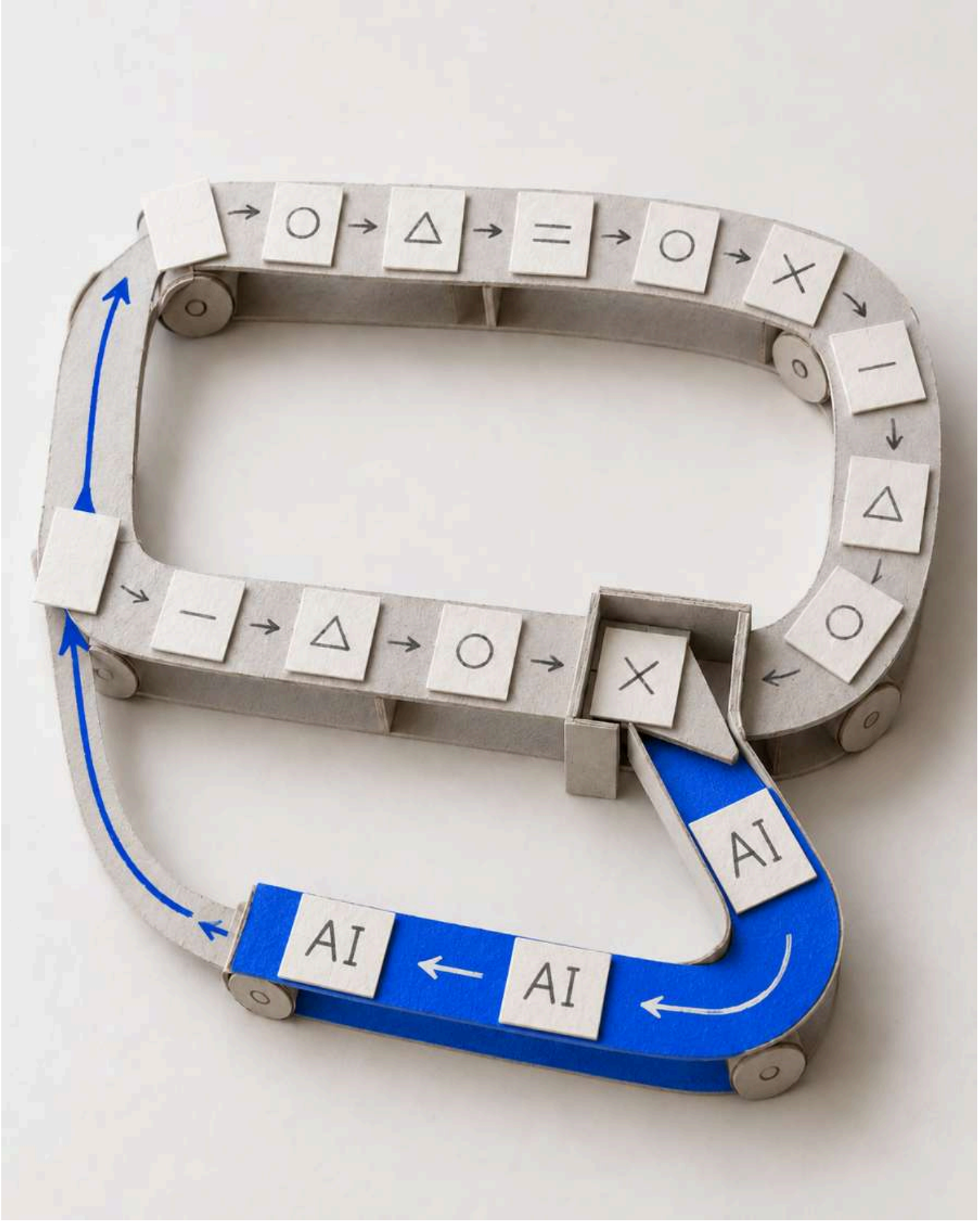
Northline might pre-encode its most-watched titles, place popular assets closer to concentrated audiences, change storage tiers, eliminate low-use formats, cache repeated requests, or scale variable services according to demand.

Product growth changes the production architecture



Refinement continues after traction — scale informs a return to refinement, not a finish line.

AI Changes the Shape of Labor



Retries, review and the real cost of speed — faster generation is not the same as proportionally cheaper production.

Artificial intelligence can reduce the time required for research, coding, content tagging, interface exploration, testing and documentation — and let a team explore more alternatives before selecting one. But faster generation is not the same as proportionally cheaper production.

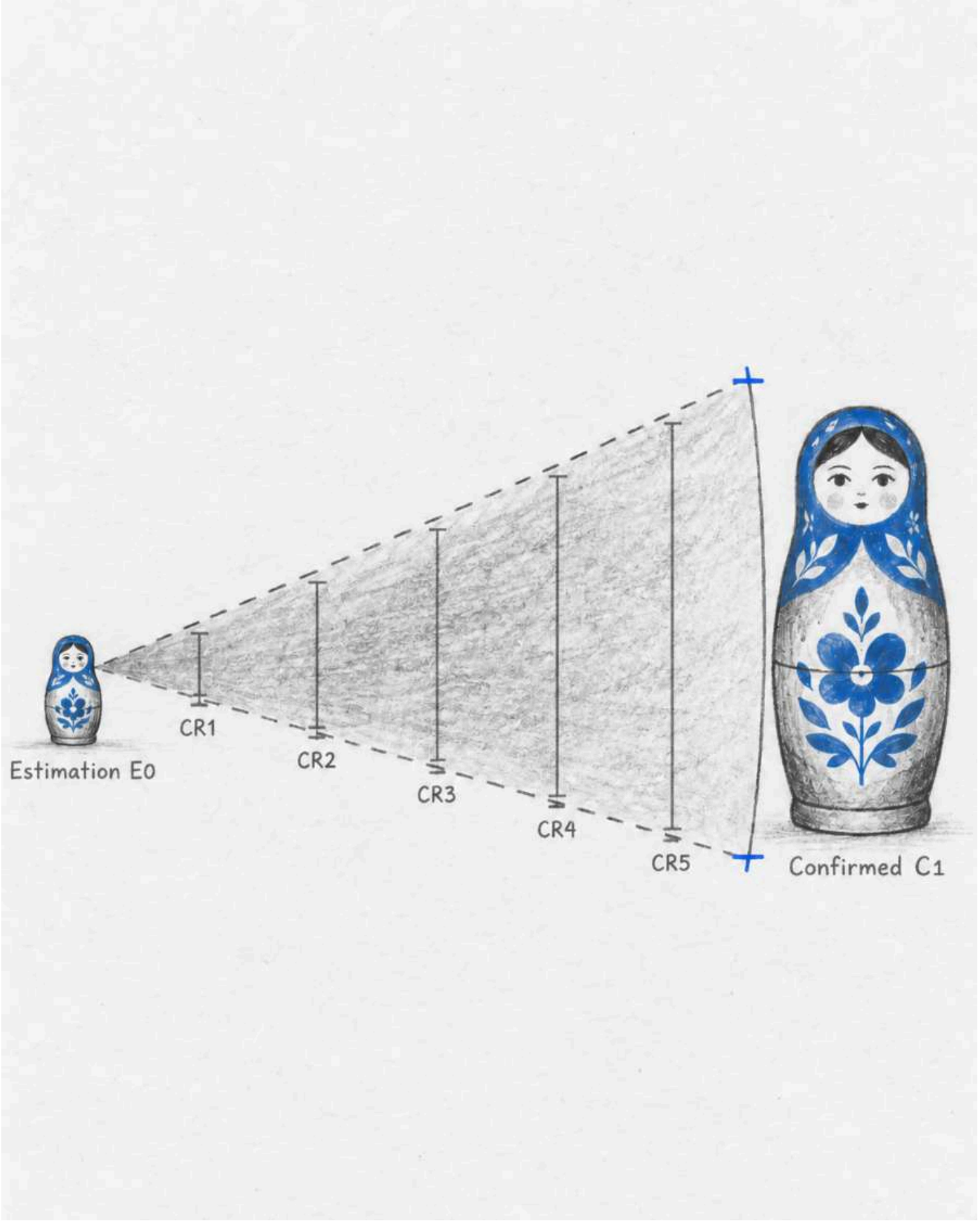
> $C_{AI\ workflow} = C_{inference} + C_{retrieval} + C_{evaluation} + C_{retries} + C_{human\ review} + C_{operation}$

A model with a lower price per token may produce a higher cost per successful task if its outputs require more retries or manual correction. A generated component may appear complete while still requiring security review, integration, testing and long-term ownership.

This is why AI-assisted estimates can remain far from reality — the tool may estimate the effort visible in the request while missing the environment the work must survive in.

Cloud providers recommend measuring unit costs — cost per inference, per data point, per completed task — alongside business-value measures, and continuously comparing costs and outcomes as the system and its resource allocation are refined. AI changes the speed of p

Estimation Is a Range, Not a Promise



A forecast, not a receipt — an estimate narrows as confidence ranges close in on a confirmed cost.

An early estimate will be wrong. That does not make it useless — its purpose is not to disguise uncertainty with a precise figure, but to reveal which assumptions could change the investment decision.

A credible estimate begins with a technical baseline, decomposes the work, records assumptions and applies more than one estimating method, then tests sensitivity and risk before comparing the forecast with actual results.

Instead of saying the product will cost $500,000, a team can say: under the current catalogue, device coverage and viewing assumptions, the first release is expected to cost between $450,000 and $650,000, with content preparation and delivery volume as the greatest

uncertainty.

The second statement is less comfortable but more useful — it identifies where research, prototyping or negotiation can reduce uncertainty. Different methods can be combined as product knowledge improves.

TABLE 2.2 — ESTIMATION METHODS EVOLVE WITH PRODUCT MATURITY

Method	Best use
Analogous estimation	Comparing with similar products or previous initiatives
Bottom-up estimation	Costing known components and activities
Parametric estimation	Applying relationships such as cost per hour, unit or transaction
Three-point estimation	Modeling optimistic, probable and pessimistic scenarios
Activity-based costing	Assigning shared costs to the activities that consume them
Sensitivity analysis	Identifying assumptions with the largest economic effect
Rolling forecast	Replacing assumptions with actual performance over time

In physical manufacturing, prototypes and pilot runs reveal actual material yield, assembly time and process defects. Digital products need the equivalent — instrumented releases that measure delivery cost and operational effort, not adoption alone. An MVP that validates demand while ignoring operation validates only half of the product.

The Cost Catalogue

No licensed or generated image matched this section yet — left as an honest placeholder rather than a mismatched one.

Cost visibility should not remain inside a spreadsheet owned only by finance or cloud engineering. Product decisions need a shared catalogue that connects expenditure with design choices.

Every significant cost should identify the product or capability that creates it, its owner, whether it is fixed, variable or step-fixed, the unit that causes it to grow, the evidence behind the estimate, the level of uncertainty, and the date it was last reviewed.

A cloud invoice organized only by technical service is insufficient — it can show that data transfer increased without explaining which customer behavior, feature or market produced it. Product cost management requires allocation by meaningful units: active household, completed order, successful transaction, processed document or resolved request.

FinOps practices formalize this collaboration between product, engineering and finance. The objective is not simply to reduce cloud spending but to connect technology consumption with business value and assign responsibility for both.

Dashboards for forecasting, tagging, allocation, budgets and anomaly detection can support this work, but their value depends entirely on the quality of the product model underneath them. A sophisticated dashboard cannot repair an undefined unit of value.

Design Continues After Launch

A product, still being designed — the most resilient products are those built to change when reality arrives.

The Barcelona Chair and Northline appear to belong to different worlds — one transforms steel and leather into a physical object, the other transforms data, infrastructure and content rights into hours of entertainment. Both reveal the same relationship between design and economics.

The Barcelona Chair preserved expensive craftsmanship and found viability through a premium position. Northline cannot solve increasing consumption simply by charging more for every technical inefficiency; it must continually redesign how content is encoded, stored and delivered.

Materials change, suppliers disappear, usage patterns evolve, technology ages, regulations introduce new controls, and growth reveals bottlenecks that were invisible in the prototype. Cost refinement is not evidence that the original product failed — it is part of the product's development.

The failure occurs when the organization protects the original design after the evidence has changed. A product is not only the object, interface or service the customer experiences; it is also the system capable of producing that experience repeatedly.

The most resilient products are not those whose teams predicted every future cost. They are those designed with enough visibility and adaptability to change when reality arrives. Every cost begins with a decision.

REFERENCES

1. Stratechi (Joe Newsum) — On the customer value equation, the value wedge, product dimensions, sources of competitive advantage and the four-step roadmap process.
2. IDEO — Design thinking: desirability, feasibility and viability as evaluation lenses.
3. Museum of Modern Art — On the Barcelona Chair, designed by Ludwig Mies van der Rohe and Lilly Reich, 1929.
4. Knoll — The Barcelona Chair in licensed production: craftsmanship and upholstery method.
5. NASA Software Engineering Handbook — Project attributes used in software-cost estimation.
6. Amplitude — On defining and using a North Star Metric.
7. Netflix — Open Connect, Netflix's content-delivery system.
8. AWS — Auto Scaling documentation.
9. Google Cloud — Well-Architected Framework, AI and machine-learning workload cost guidance.
10. GAO — Cost Estimating and Assessment Guide.
11. FinOps Foundation — FinOps practices and principles.