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.

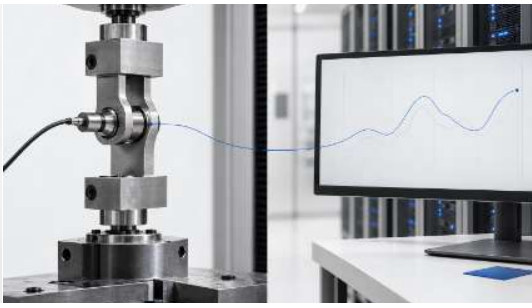
If I had to reduce it to a deliberately imperfect ratio: a product is 10% idea and 90% effort to prove that idea can survive reality. It isn't a statistic — it's a way of ordering the experience this chapter unpacks through an 1859 chair, a few uncomfortable hypotheses, and the argument that digital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



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

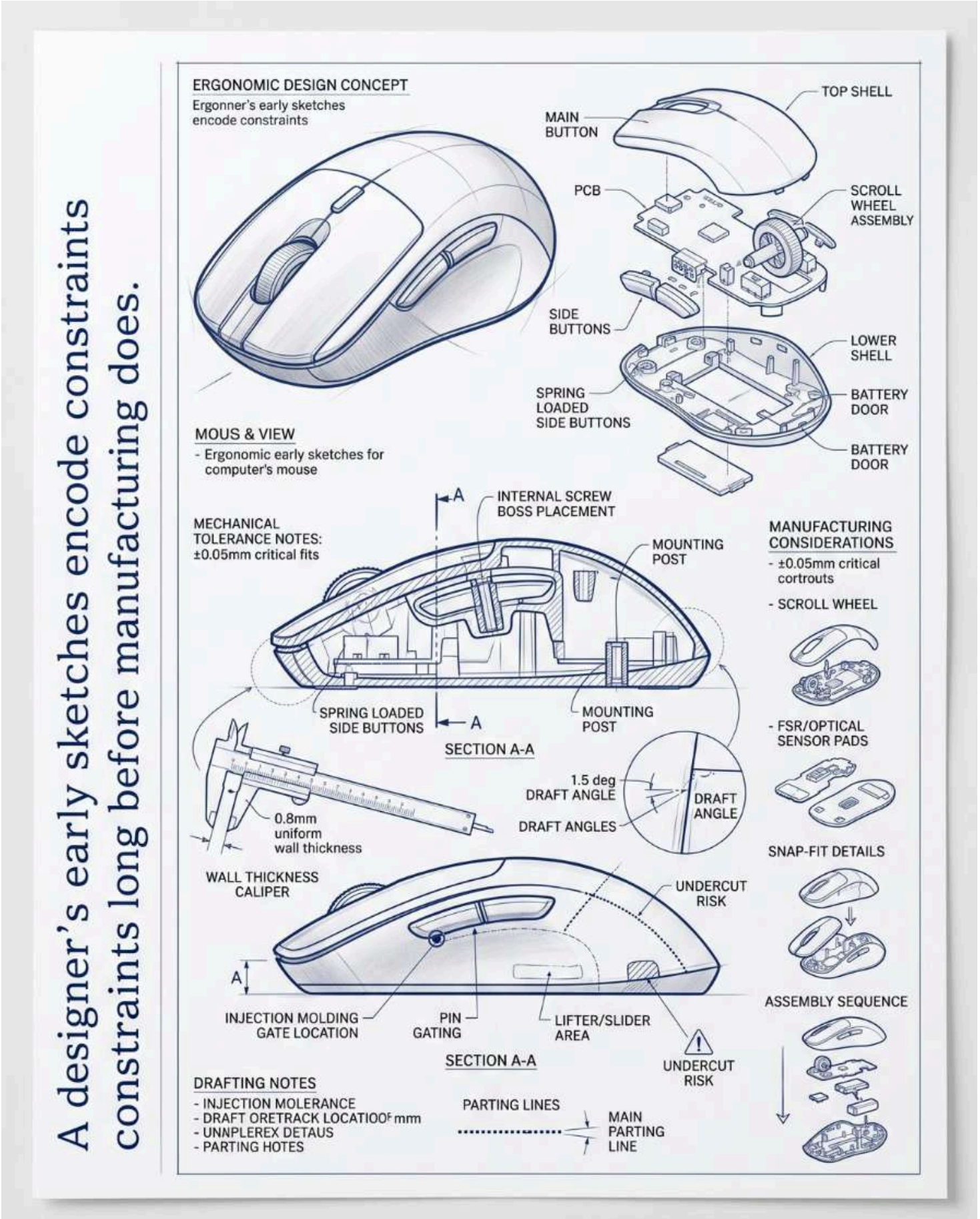


Tooling cost is a decision made once and paid for in every unit after.



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."

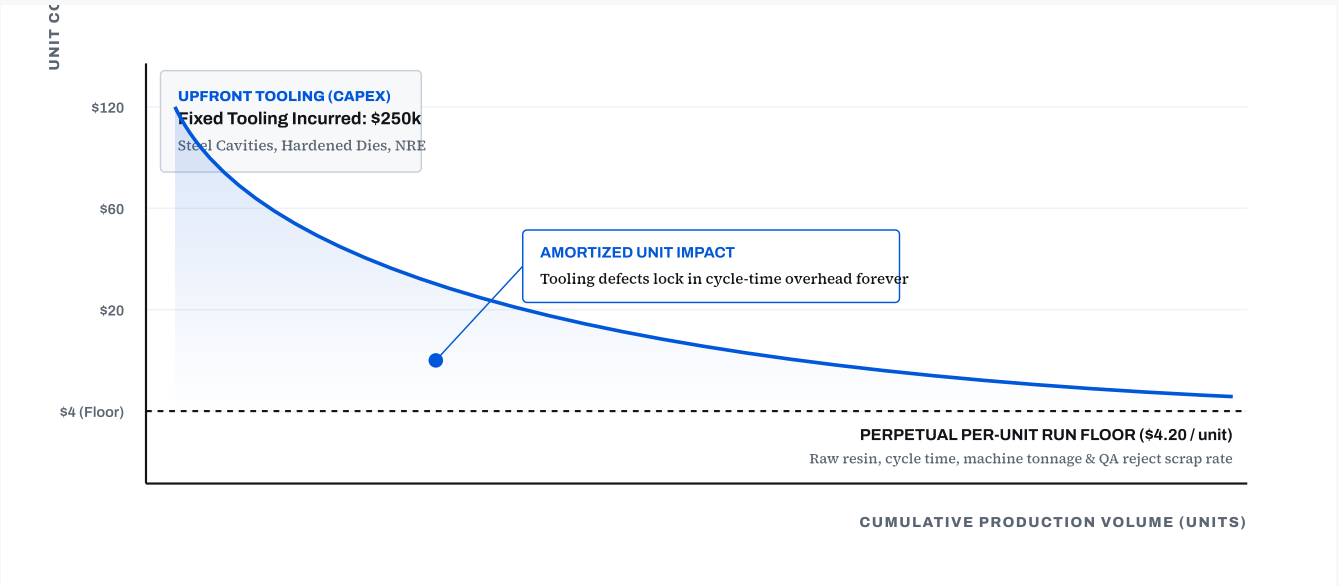
A product doesn't exist because it can be built. It exists when we find a reasonable way to create value, deliver it, and capture enough value to sustain the system that produces it. David Teece defines the business model as the architecture through which a company creates, delivers and captures value — 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.

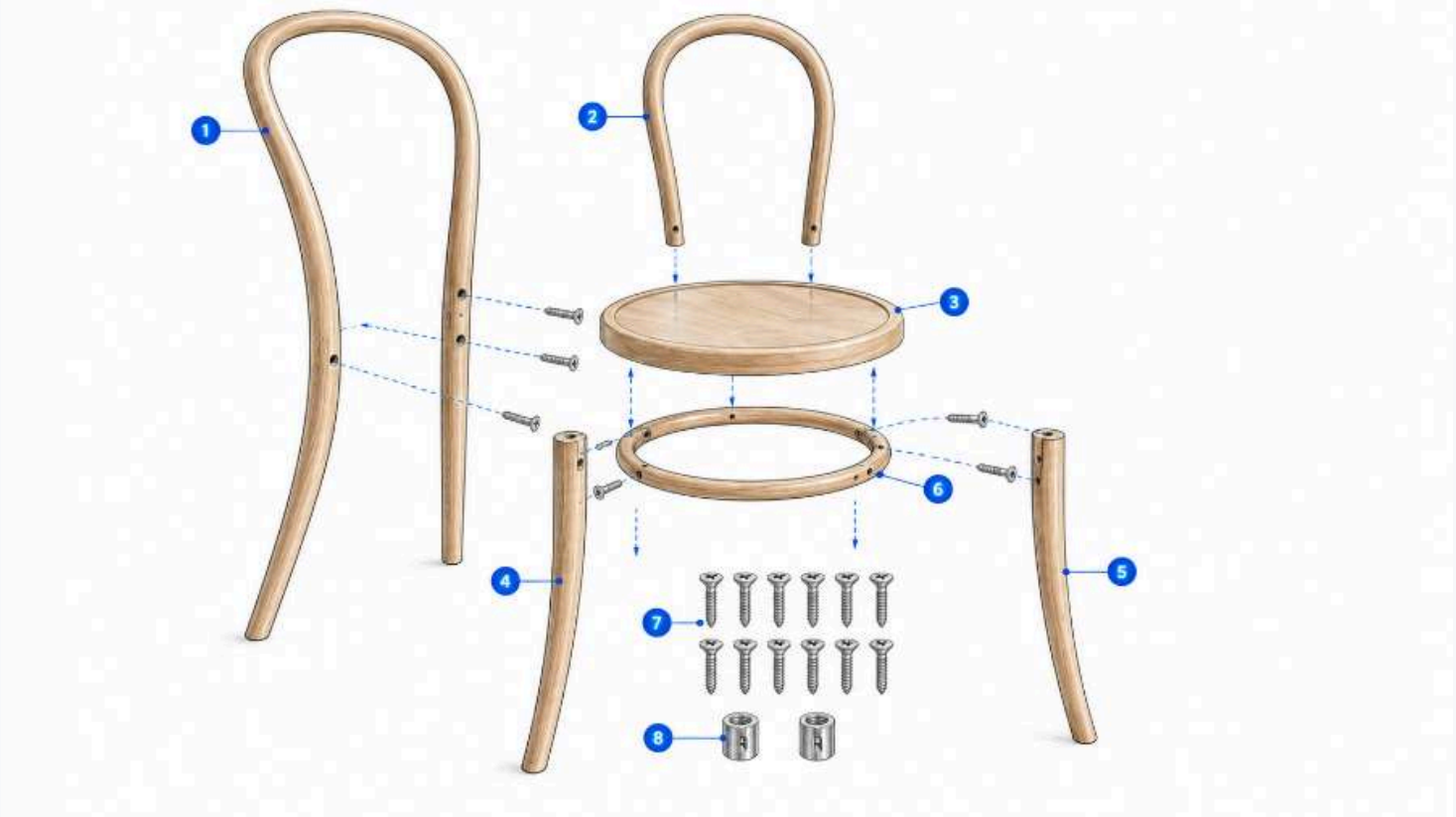
At first glance its success might be attributed to its form: light, recognizable, neutral enough to exist for decades in cafés, restaurants and homes without depending too much on a specific trend. But focusing only on the silhouette misses the more interesting part. The chair was a system of decisions.

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."*

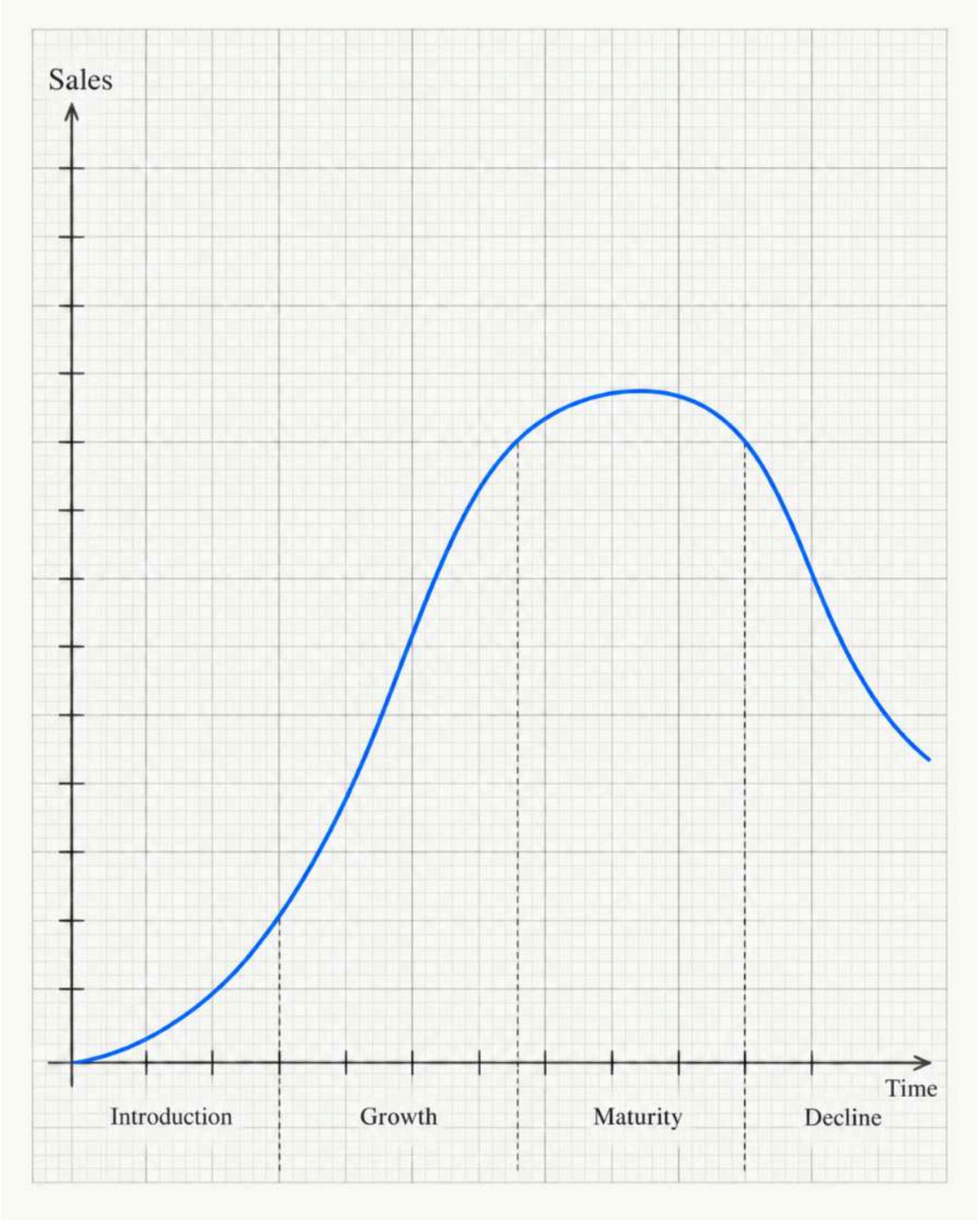
Another lesson from physical product is that a version doesn't need to contain everything the product could eventually become. The industry has worked for a long time with generations, variants, platforms and incremental improvements. We sometimes forget this when we tell the story of digital product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

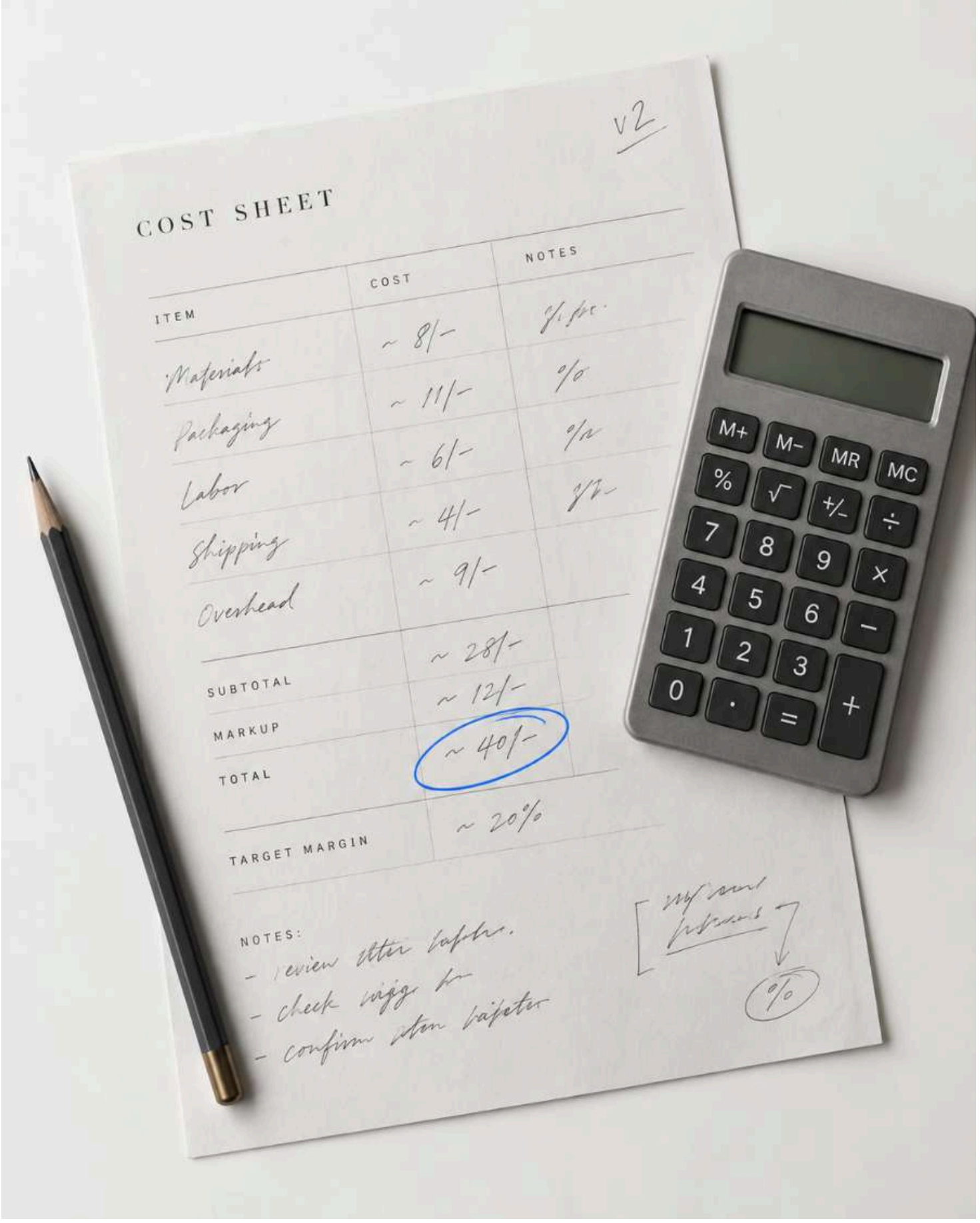
By the time a product gets there, many of the decisions that will limit its trajectory have already been made: cost structure, supply chain, component architecture in the physical world; infrastructure, data model, acquisition, pricing in the digital one.

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.

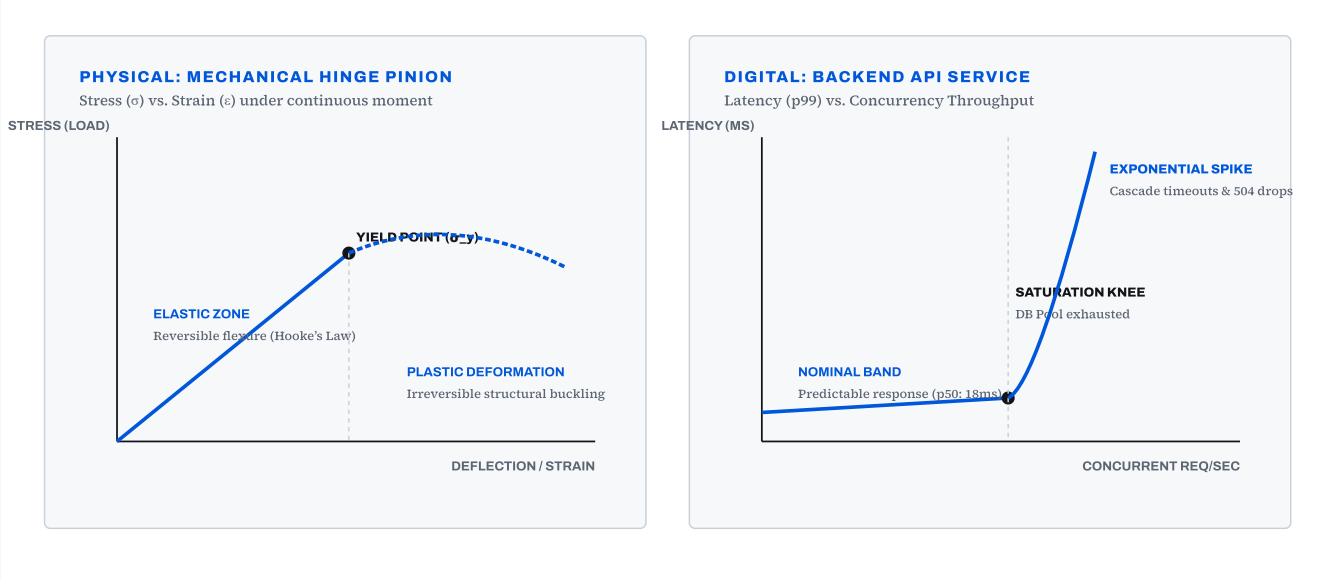
One of the intersections I find most interesting between industrial design and digital product is economics. My experience is that economics is part of the design material: if a product requires a heavy tooling investment, we need a volume capable of amortizing it; if a complex part can become three 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.

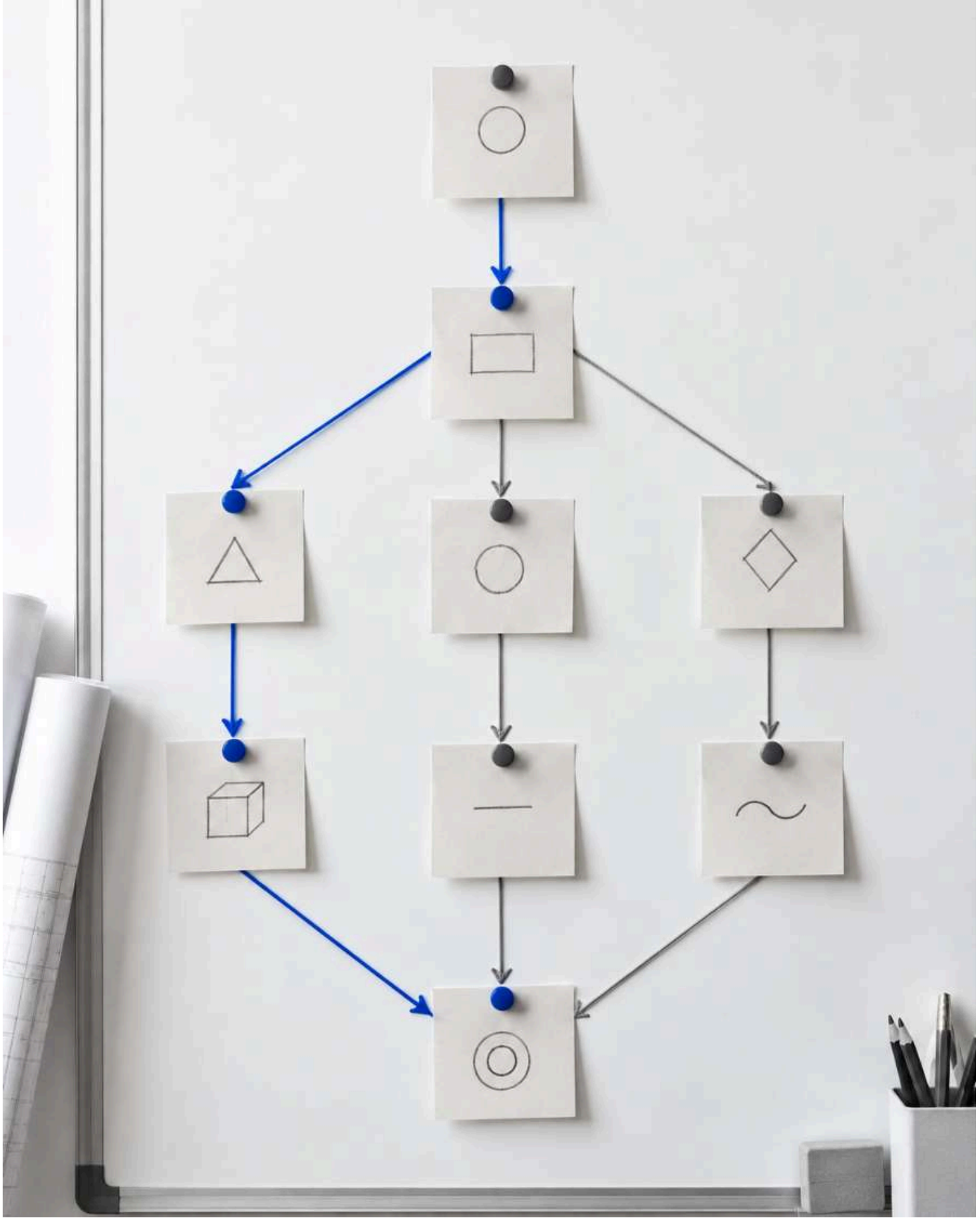
That's why a monetization plan shouldn't be a slide for investors. It should become a set of explicit hypotheses: who pays, why they pay, when they pay, why they'd keep paying, and which costs grow as usage grows. can

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.

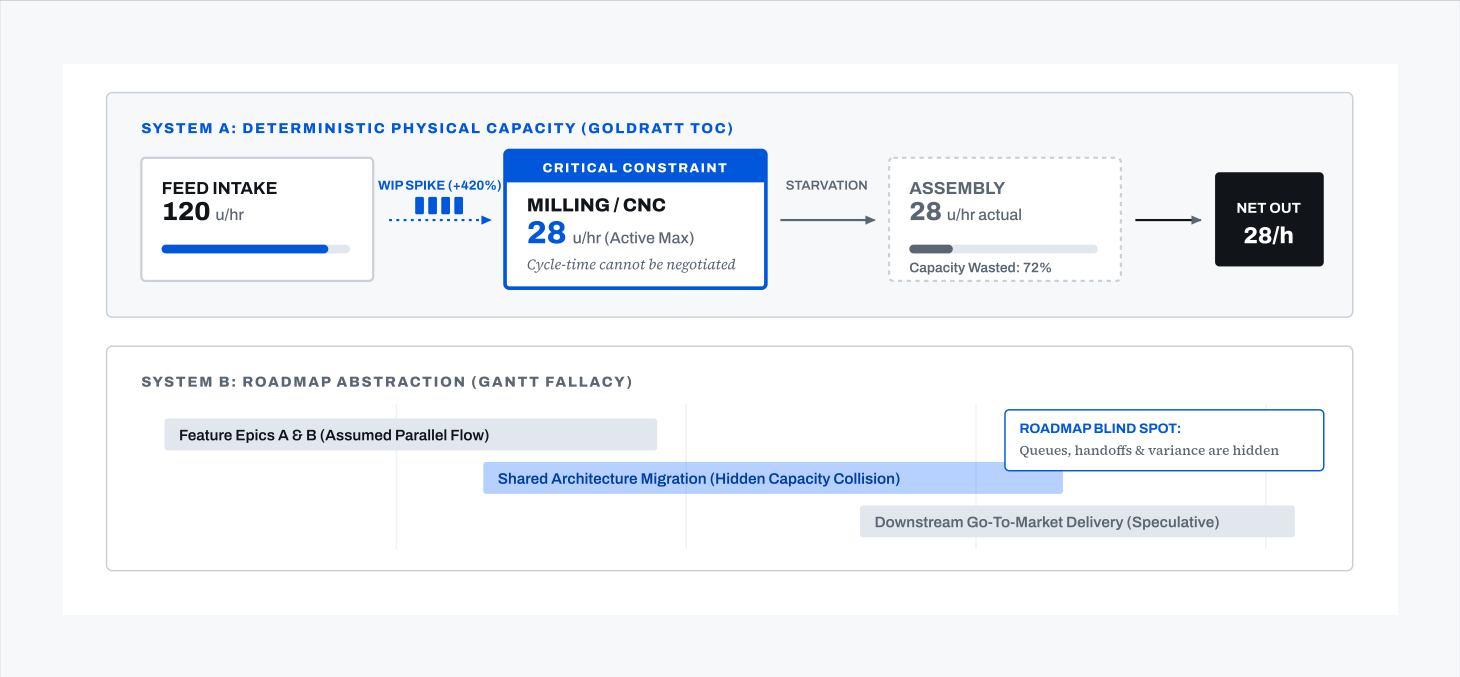
A project exists to produce a result within a specific combination of scope, time, resources, dependencies and risk. A product, by contrast, can move through dozens of projects over its life: launching it might take one, migrating its architecture another, entering a different market a third.

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.”

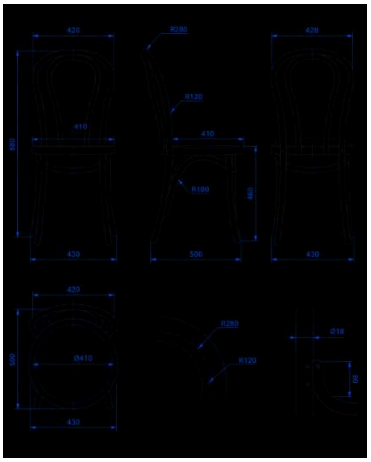
— J. A. Sosa

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.