

Growth Has a Tail.

10

GROWTH HAS A TAIL

Revenue arrives before the full cost of earning it does.

Growth looks clean on a chart. Revenue rises, customers accumulate, markets open and headcount follows, while the company appears to be doing exactly what ambitious companies are supposed to do: becoming bigger, stronger and more valuable.

Then the chart meets the company.

The enterprise customer that made the quarter requires its own implementation. The feature that closed an important deal becomes another branch of the product to maintain. The new country needs different contracts, pricing and support, while a temporary workaround survives because replacing it would interrupt growth. Teams form around the exceptions, managers around the teams and systems around the resulting complexity.

Almost none of these decisions is foolish. Most were perfectly rational when they were made, which is precisely why their accumulated consequence is difficult to see. Growth creates obligations before their economics become visible: revenue is counted when it arrives, while the organisation required to keep earning, delivering and supporting it reveals itself over time.

Growth has a tail, and eventually you have to operate it.

The sale ends before the economics do

Imagine two companies adding the same \$10 million of revenue. One does it through an increasingly repeatable product, where implementation gets faster, knowledge carries from one deployment to the next and the commercial model becomes easier to reproduce. The other reaches the same number through custom configurations, pricing exceptions, specialist support and implementation teams that grow alongside the customer base.

From far enough away, both are growing; up close, they are becoming different kinds of companies. The first is converting growth into capability, while the second may be converting it into obligation. That distinction appears gradually through implementation time, support demand, engineering capacity, sales productivity and cost-to-serve before eventually reaching margins and cash generation. Revenue can improve while the machine producing it quietly deteriorates.

Bain has described a “growth paradox” in which growth creates organisational complexity that can ultimately impede further growth, while research by Nicholas Bloom, John Van Reenen and others has found strong associations between management practices and productivity, profitability, sales growth and firm survival.

Yet complexity is only part of the story. The deeper problem is that the benefit of growth and the burden it creates arrive on different clocks. A customer comes first; the support requirement follows, the exception becomes precedent and eventually somebody is hired to manage what was once unusual. By the time the full cost becomes visible, the decisions that created it have disappeared into the history of the company.

Getting bigger is not scaling

Much of my career has involved businesses that were already growing, which sounds preferable to fixing one that is shrinking and usually is, except that growth creates its own camouflage. When demand is strong, almost every problem can initially be solved by adding capacity: another salesperson handles the pipeline, another engineer handles the variation and another manager coordinates the people added to solve yesterday's constraint. For a while, revenue outruns the inefficiency.

At Amazon, I encountered this inside Brand Innovation Lab. Bespoke creative work could be enormously valuable because it responded closely to individual advertisers, but that same responsiveness could make every success expensive to reproduce. The challenge was not simply producing more work, but turning what repeatedly worked into a more productised commercial system so that growth did not require reinventing the machine every time.

That reinforced a distinction I have come to regard as fundamental: **repetition is not scale unless repetition improves the system.**

The hundredth customer should benefit from what was learned serving the tenth. Brand should reduce the cost of establishing trust, product knowledge should reduce implementation effort, distribution should make demand easier to reach, and software should allow output to grow faster than the labour required to produce it. A business should become not merely capable of producing more, but better at producing more.

Which gives us a harder test than the growth rate itself: **does every additional \$1 of revenue make the next \$1 easier or harder to earn?**

If the next dollar becomes easier because the company has accumulated reusable product, knowledge, distribution, infrastructure or trust, something is compounding. If it becomes harder because each increment requires proportionately more customisation, coordination and intervention, the company may be getting bigger without truly scaling.

Capital can conceal the difference. It cannot abolish it.

Never confuse heroics with architecture

Early in my career as a software engineer, I learned that badly designed systems can appear perfectly healthy under light load because clever people know the shortcuts, remember the dependencies and work around the weaknesses. Architecture reveals itself under pressure, and companies behave much the same way.

At the beginning, the founder remembers why every promise was made, the best salesperson knows how to get the unusual deal through and an engineer knows which part of the product cannot safely be touched. The company feels fast because people carry context in their heads and cross organisational boundaries informally. That is often a strength; the mistake is allowing heroism to become architecture.

As the company grows, the same people become routing layers. Decisions travel upward because context has not travelled downward, customer issues depend on institutional memory and the strongest people become bottlenecks precisely because they are so valuable.

That has left me with a simple operating instinct: **when repeated heroics appear inside a growing company, admire the person and investigate the system.**

Success preserves what failure would have killed

Young companies should do things that do not scale. They should sit with customers, improvise and customise because a business that standardises before understanding its market can become beautifully efficient at producing something nobody particularly wants.

I experienced the other side of this while building Unknown Origins. Services businesses naturally reward responsiveness, and done well that proximity produces extraordinary insight into what customers actually value. The trap appears when every lesson remains a service rather than becoming reusable capability.

Moving from creative output towards repeatable offerings reinforced something I had seen elsewhere: bespoke work is not the enemy of scale when it generates learning. The problem begins when the learning has finished but the exception survives.

An exception should teach the product something, justify its own economics or disappear.

Successful companies find this particularly difficult because failure makes its case immediately, while success leaves evidence in defence of almost everything that produced it. A customer was won or a quarter was made, so the workaround acquires legitimacy simply because it was present at the scene of the victory. What helped win the first \$10 million can eventually make the next \$100 million harder to earn.

Scale requires deciding what not to scale

Microsoft gave me a much larger laboratory for the same problem. Building businesses across countries, industries, products and partner ecosystems makes one truth unavoidable: global scale does not come from reproducing everything everywhere.

Something works in one market, so the organisation reproduces it in another. Local requirements appear, teams adapt, propositions diverge and operating models accumulate variations until the global business risks becoming a federation of perfectly reasonable local decisions whose aggregate economics are anything but simple.

Blind standardisation is no better because markets really are different. Customers buy differently, regulations vary and distribution structures differ. The hard work is determining **which differences contain information and which merely contain history.**

Building Microsoft's industry and transformation businesses made that judgement central to the job. The objective was never uniformity, but enough common architecture across product, commercial motion, partners and accumulated learning that entering the next market did not require inventing another company.

Standardise everything and you become deaf to the market; localise everything and you become hostage to it. Scale depends on knowing the difference.

Complexity has to earn its keep

This is why I distrust simplistic calls for organisational simplicity. Complexity is not inherently bad; unexamined complexity is. Some complexity protects value, some unlocks markets and some creates differentiation. The managerial task is not to eliminate it but to distinguish complexity that earns a return from complexity that merely consumes one.

A product variant that repeatedly unlocks a valuable segment may deserve to become product, while twenty variants accumulated through unrelated customer requests may represent product debt. A specialist implementation team may create competitive advantage because customers value the expertise, or it may exist because the underlying product remains unnecessarily difficult to deploy. A local operating model may encode valuable market knowledge, or simply reflect the first way the company happened to enter that country.

The useful question is economic rather than aesthetic: **what does this complexity allow us to earn, protect or learn that we could not without it?**

If nobody can answer convincingly, it has probably stopped paying rent.

Growth can hide deterioration

The most treacherous moment is often when the company still appears successful, because fast growth can conceal weakening operating leverage while new revenue arrives faster than accumulated inefficiency becomes visible. Bookings rise, so implementation expands; customers increase, so support grows; products multiply, so coordination increases. Each decision can make sense independently even as their combined effect makes the next increment of growth progressively harder.

I saw another version of this at PE-backed PeopleCert, where the challenge was not simply generating demand but connecting brand, demand, customer economics and operating cadence as the business expanded internationally and prepared for IPO readiness. That environment sharpens the distinction between growth and the **quality of growth**: how repeatable is demand, what happens to acquisition economics, how does customer value develop after the initial transaction, and can commercial momentum translate into durable financial performance?

Different business models legitimately carry different cost structures, so there is no universal ratio that proves a company is scaling. The danger lies in assuming scale economics will somehow arrive later.

They emerge only when learning and reusable capability accumulate faster than the obligations created by growth.

AI makes complexity cheaper too

Artificial intelligence can lower the marginal cost of software development, content production, analysis, customer service and administrative work. The obvious conclusion is that companies will produce considerably more with smaller teams, and in many activities they probably will.

The less comfortable possibility is that cheaper production also makes complexity cheaper to create.

I saw the beginning of this shift at Amazon as generative AI started changing the economics of creative production. When the cost of producing something falls dramatically, the immediate temptation is to make more: more variants, more personalisation, more campaigns, more experimentation. Yet the bottleneck moves rather than disappears because somebody still has to decide what deserves to exist, understand what is working and stop what is not.

The same problem now reaches across the enterprise. More software can mean more software to maintain, more agents can mean more automated interactions to govern, and more personalised propositions can create more commercial variants whose downstream economics remain obscure. A company can make every department more productive while making the enterprise harder to operate.

That is not an argument against AI; it is an argument against confusing automation with redesign. Whenever technology removes a constraint, management should ask what was built around that constraint and whether it still deserves to exist.

Making unnecessary work cheaper does not make it necessary.

The organisation eventually reaches the customer

Operating models are usually discussed as internal machinery, but customers experience them through response times, implementation, reliability and whether somebody appears able to make a decision. Internal complexity eventually escapes the building as fragmented ownership becomes another hand-off, slow decisions become delayed answers and product complexity becomes an implementation problem.

The reverse is more valuable. When learning from one customer improves the experience of the next, the operating model is creating product advantage; when deployment becomes faster through repetition, it is creating margin; and when the same platform can enter another market without recreating the company around it, it creates strategic optionality.

Perhaps the most durable lesson I have taken from moving between engineering, global technology businesses, entrepreneurial ventures and PE-backed environments is that **the operating model is not what happens after strategy**. Product determines what value can be created, go-to-market determines whether it can be captured, and the operating model determines whether doing both repeatedly compounds the value or consumes it.

Eventually, the customer feels the difference. So does the investor.

The multiple eventually discovers the tail

Investors naturally spend enormous energy understanding revenue: its growth, recurrence, retention, pricing and pipeline. There is another question worth asking alongside them: **what obligation travels with the revenue?**

Two dollars of revenue are not economically identical if one can be delivered through an increasingly capable system while the other carries growing requirements for implementation, customisation, support and senior intervention.

The relevant question is therefore not whether the company has complexity, because every serious company does, but whether the value created by growth is increasing faster than the organisational burden required to sustain it. No single organisational measure determines enterprise value, and stronger management does not create some universal law of valuation. Competitive position, market structure, business model and capital intensity still matter enormously.

The mechanism is nevertheless difficult to escape. When growth creates reusable capability that reduces the marginal effort required for future growth, the conditions for stronger operating leverage improve; when obligations accumulate faster than capability, pressure travels in the opposite direction.

Sooner or later, the economics notice.

MAKE THE NEXT \$1 EASIER

After enough versions of this problem, a few principles become difficult to unlearn.

Let exceptions buy knowledge. Bespoke work is valuable when it reveals repeatable demand, but an exception that neither teaches the product nor justifies its economics should not quietly become permanent architecture.

Follow revenue beyond the sale. A customer is not economically understood when the contract is signed. Follow what that revenue subsequently consumes in implementation, support, engineering attention and management intervention, because the tail is part of what you sold.

Turn experience into capability. The hundredth deployment should contain the accumulated intelligence of the first ninety-nine. If the company keeps learning the same lesson through different people, it is accumulating experience without accumulating advantage.

Know what must vary. Markets and customers are genuinely different, but not every difference deserves its own operating model. Preserve variation where it creates value and remove it where it merely records history.

Watch the heroes. Exceptional people are an advantage, but needing exceptional people to make ordinary things happen is a warning. Repeated rescue missions usually reveal a capability the organisation has failed to build.

Delete before automating. AI makes execution cheaper, which makes judgement more valuable. Before accelerating a workflow, establish whether the workflow deserves to survive.

Above all, **make the next \$1 easier.** Not cheaper regardless of quality or standardised regardless of customer value, but easier to sell, faster to implement, more reliable to support and more economically attractive to retain. Growth compounds when today's customer leaves behind something that helps win and serve tomorrow's.

What growth leaves behind

At the beginning of a company, the great uncertainty is whether anybody wants what you have made. If you are fortunate, people do, and that victory creates a different problem: you have to become the company capable of keeping the promise.

I began my career as a software engineer, and an instinct from that world has stayed with me. You learn surprisingly little about architecture when a system is under light load because its hidden dependencies reveal themselves only under pressure. Growth does the same thing to a business, exposing where knowledge really lives, where decisions stop, which exceptions have become dependencies and whether success is creating reusable capability or merely more work.

After years spent building, scaling and rebuilding businesses, that has made me less interested in growth as an isolated number than in what growth leaves behind. The boardroom question is not simply how much larger the company can become, but what would happen if it became three times larger: **what would become three times better, and what would quietly become three times harder?**

Revenue tells you that you won the customer; **the tail tells you what the growth was worth.**

Roy Sharples
September 2026

Roy Sharples is the founder of Lucan Ventures, where he builds and reinvents businesses through product innovation, go-to-market and operating model design.

[Lucan Ventures](#) | hello@lucanventures.com

© 2026 Lucan Ventures. All rights reserved.