
Closing the Delivery Discount in B2B SaaS

Why vertical software that runs like a services business is priced like one, and the operating model that closes the gap as a company grows from 50 to 300 people

September 2026

CHAPTER 1

Executive Summary

The early-2026 software sell-off was read as a verdict on the sector. It was a repricing of criteria: the market now pays for durable earnings and marks down whatever looks old. Entrenched vertical B2B platforms, the ones owning the workflow, the switching costs and the regulatory depth, are the right shape for that regime — and are marked down anyway, because from the outside they look like legacy.

That mispricing makes vertical B2B software the most interesting play in the sector for private equity: the moat is real, and the discount sits on the delivery model underneath it. Software that runs like a services business is priced like one: in private M&A, non-SaaS software sold at around 2.3–2.6x revenue and IT services at around 1.1x between 2024 and Q1 2026, against 3–3.8x for SaaS. **Owning a vertical moat and being able to execute on it are different things, and the market prices the second.**

The gap has a cause. **The accounts measure the custom work a company billed. The discount is priced on the custom work it absorbed** — a decade of customer-specific commitments built into the core product, funded invisibly from the R&D line. The financial cleanup most teams have already run cleaned the wrong line. The rebuild most boards have already funded produced a newer version of the same asset.

This report sets out an operating model for resolving that condition, built for vertical B2B software companies growing from 50 to 300 people, the stage band in which the condition forms. Resolution is an order, not a method: mandate, casting, team, then rebuild and routing. And AI has raised the stakes in both directions: on a governed operating model it makes the software multiple reachable inside a hold period; on an ungoverned one it industrialises the decay.

The destination is specific. Exhibit 1 shows the shape the market pays a software multiple for, beside the shape it discounts. The gap between the two columns, priced on the same revenue, is what this report names the delivery discount.

EXHIBIT 1 The shape the market pays for, beside the shape it discounts

TWO SHAPES, TWO PRICES

The same revenue. The market pays for the shape, not the number.

	PRICED AS SOFTWARE ~3–3.8x revenue	PRICED AS SERVICES ~1.1–2.6x revenue
Gross margin	80%+	~60%
Services share of revenue	<15% of mix · ~5% of CoGS	20%+ of mix · >10% CoGS
Net revenue retention	110%+	below 100%
R&D share of revenue	20–30%, with the roadmap to show for it	30%+, without one
Rule of 40	met	failed
ARR per head	\$200k+	~\$115k

The gap between the two columns is the delivery discount, priced on the same revenue.

Chapters 2 and 3 diagnose the condition and show how to detect it from outside the codebase. Chapters 4 to 6 cover what AI changed, the architecture of resolution, and why no external party has fixed this. Chapters 7 and 8 cover what resolution is worth and where to start.

CHAPTER 2

Why the Pro-Serv/Subscription Cleanup Didn’t Move the Multiple

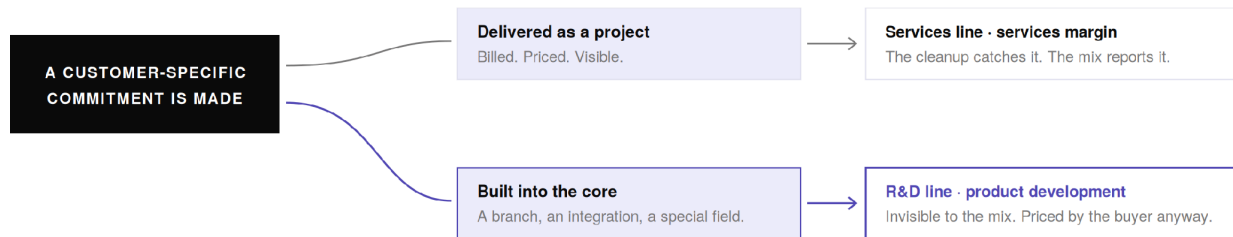
A software business on the deck; a services business in the P&L. Every sponsor has seen the pattern, and the standard remedy is well known: drive professional services below 15% of mix, separate the income statement, and the software multiple should follow.

The remedy rests on a flawed diagnostic. Revenue mix measures only the custom work a company billed. It is blind to the custom work the company absorbed.

EXHIBIT 2 The same commitment, two routes through the accounts

BILLED VERSUS ABSORBED

The same commitment, two routes through the accounts — and only one is visible to the cleanup.



WHERE IT SURFACES INSTEAD

R&D rising as a share of revenue without the roadmap to show for it · delivery dates behaving like estimates of estimates · a cost of change that climbs every year for reasons that never appear as a line item.

The revenue mix measures the first route. The delivery discount is priced on the second.

Where the Custom Work Went: The R&D Line

When a B2B platform secures a major enterprise contract, the demands for bespoke workflows, custom integrations and tailored onboarding do not disappear because the board capped the services P&L. The cost of meeting them is reallocated. Without strict architectural governance, engineering builds single-customer edge cases directly into the core product.

- **Bloated R&D.** The budget stops funding scalable product innovation and quietly funds unbilled professional services.
- **Degraded cadence.** Engineering bandwidth is consumed managing the technical debt of competing custom configurations, and releases slip from monthly to quarterly.
- **Margin erosion.** True gross margin deteriorates as the cost of maintaining hidden bespoke branches scales with the customer base.

Sophisticated buyers see through a clean revenue mix. They examine R&D as a percentage of revenue, ARR per head, and release cadence. When they find bloated engineering cost and sluggish deployment, they apply the delivery discount, correctly: the asset is a services business the accounts have not caught up with.

The Empty Chair: Nobody Owns the Product–Services Boundary

The convenient explanation, that engineering was never consulted, is usually wrong. Technical leadership is often in the room. The chair that is empty belongs to a function most of these companies have never split out: ownership of the boundary between the core product and its services layer.

Consider the room where a €250k annual enterprise renewal is negotiated. Sales, account management, often the chief executive, frequently a technical leader too. Everyone is incentivised to reach yes. But nobody owns the distinction between what belongs in the product and what belongs beside it, so nobody can say: we will build this, outside the core, priced, on its own contract.

Two structural facts keep the chair empty even when technical leadership is present. **The team that built version one is rarely the team that carries the product to the next stage** — the job changed underneath them, which is why rebuild attempts from the incumbent team stall. And the oldest strata of the platform have often gone untouched for 10–20 years, because every quote for

changing them starts at three years. The boundary question was settled by default, long ago, in favour of absorption.

The chair is not a title. It is a decision right with a clock: whether a commitment enters the core product, or is built beside it, priced, on its own contract. Standards do not collapse in one decision. They erode through forty small ones nobody counted.

Everyone in that room is paid on the yes. The function that would carry the cost has not been invented yet.

CHAPTER 3

Why It Happens in Vertical B2B

Code bloat is rarely a symptom of engineering incompetence. It is the accumulated interest on a decade of rational commercial survival.

When a €250k contract hinges on a bespoke workflow, the rational short-term decision is to say yes and build it into the core. The delivery discount is the long-term price of those short-term decisions. And it does not happen randomly: it fractures predictably across scaling stages. The reliable marker is not headcount alone but the number of real management layers. Where headcount and layers disagree, believe the layers.

The published scaling roadmap maps ten stages, 0 through 9. Three of them are where this audience lives — and where the discount forms, compounds, and finally prices. For each: the binding constraint, what it looks like from inside, and what graduation requires, function by function.

EXHIBIT 3 The three stages that matter here — constraint, shape, and graduation by function

Stage	The constraint	Product	People	Delivery & operations
CATEGORISE 50–99 · 3 layers	No function has an accountable owner. Every consequential decision routes through the same few people, and commitments arrive without a cost attached.	Looks like: roadmap by hunch; specs live in meetings; sales commitments land on engineering unpriced. Graduates by: splitting product from delivery, giving the intake an owner, and installing written specs with a ready-for-dev gate.	Looks like: generalists everywhere; the loudest opinion carries; founders remain the last decision-maker on everything. Graduates by: functions with named owners, mapped decision rights, and a first management layer that is trained rather than merely promoted.	Looks like: one queue, everything urgent, releases slipping quarter on quarter. Graduates by: SOPs and standardised artefacts — specs, user stories, release notes — a published calendar, and bug work planned rather than emergent.
SPECIALISE 100–249 · 4 layers	Focus is too broad to hold. Nobody can see the accumulated cost of the commitments already made — which is where the absorbed discount hides.	Looks like: four products, one backlog; nobody can say what shipped last year without touching the core. Graduates by: product operations installed, evidence-led prioritisation, and the absorbed-cost audit — which becomes possible at this stage — actually run.	Looks like: leaders promoted into roles they have never seen at this scale; micromanagement appears, the reliable tell. Graduates by: specialised roles, teams and stacks — and for each stretched leader, development under a mentor who has held the role at this stage, or replacement.	Looks like: squads competing for one shared bench; the senior layer consumed by cross-team mediation. Graduates by: product-led autonomous teams, each owning a surface with its own engineers, tech lead and roadmap.

Stage	The constraint	Product	People	Delivery & operations
CAPITALISE 250+ · 5+ layers	Growth now requires capital allocation rather than operating effort — and each acquisition imports a new boundary problem.	Looks like: a portfolio of overlapping products, each deal adding another. Graduates by: one platform thesis, buy-versus-build discipline, and integration standards applied per acquisition.	Looks like: an org chart that mirrors past deals rather than the operating model. Graduates by: leadership allocated to constraints, not legacies, with succession built inside.	Looks like: parallel stacks and duplicated capability per acquired unit. Graduates by: shared core with per-asset edge, and capital pointed at the highest-return modernisation first.

Headcounts are indicative; where headcount and layers disagree, believe the layers. The delivery discount is a disease of the first two stages above, which is why the model is built for the journey from 50 to 300; by Capitalise, the question is whether the boundary discipline survives acquisition.

Why B2B Specifically

In consumer software the customer is a population. You instrument, test, and let evidence arbitrate; product operations are the only available method. In enterprise B2B the customer is a person with a name, a renewal date and an opinion, and the requirement genuinely is specific.

The account director who says yes looks like a commercial asset. The product manager who proposes a discovery cycle looks like an obstacle. The services shape wins by default, and in the short run it works: responsive, renewals closed, customers satisfied. The cost stays invisible for years. By the time it is visible it is architectural — which is why the condition is commonest in exactly the companies with the strongest moats. Deep workflow embedding creates the customer specificity that makes the services shape feel necessary.

The Six Indicators

The codebase is not required. Six indicators identify an absorbed discount before a buyer does — and five of the six sit outside the revenue mix entirely.

EXHIBIT 4 The six indicators

#	Indicator	What it reads	Why it matters
1	ARR per head	Well below the ~\$141k median while revenue still grows; top tier \$200k+	Headcount rose and the operating model did not. The cleanest external tell
2	R&D share of revenue	Rising, with no roadmap surface to account for it	Where absorbed customer work lands once it leaves the services line
3	Configuration absorption	The share of last year's commitments satisfied without a core change	Most companies cannot answer this at all, which is itself the finding
4	Support and success cost	Above 12% of revenue while churn holds, against a ~9% norm	Delivery is generating incidents faster than support closes them
5	Commitment-to-release time	The elapsed time, and whether it is lengthening	Measures the intake directly rather than through a financial proxy
6	Services mix	Above ~15% of revenue, or running at or below break-even	Lagging — the one everyone already watches

Five of the six sit outside the revenue mix. By the time the sixth moves, the other five have been true for years. Blended gross margin below 70% remains the clearest confirmation once the condition is advanced — and the most lagging signal of all.

CHAPTER 4

AI: Faster Fix, Faster Bloat, Faster Decay

Core architectural rebuilds have historically been un-underwriteable by institutional capital: three to four years of senior engineering effort, well beyond a standard hold period. AI changed that. It also changed the failure mode.

The Three-Year Trap: Paying for Two Platforms

The trap is not that the asset gets parked. It is that it runs. The company pays twice: R&D to build the replacement, and R&D to maintain the original, because customers keep renewing, keep raising tickets, and keep asking, and none of that stops for the rebuild. Carrying both cost lines for three years is what makes the programme un-underwriteable, and why boards defer the decision until diligence forces it at the worst possible price.

What Data Shows Around AI Potential for Bloat

- GitClear’s 2026 analysis of 623M code changes found reuse falling and duplication rising on every signal tracked. In 2022 developers were roughly 2x as likely to refactor as duplicate; by 2026 they were 5x more likely to duplicate than refactor. Duplicated blocks reached the highest level on record; maintenance of year-old code fell by three-quarters.
- Two caveats, stated plainly: GitClear sells code-analysis tooling, and the finding is correlational.
- METR’s 2025 randomised trial found AI made experienced developers 19% slower while they believed they were 20% faster. The 2026 revisit of the same cohort estimates ~18% faster. A 37-point swing in 12 months is a warning against confident forecasts.

METR measures speed. GitClear measures structure. For an asset whose problem is unpredictable delivery cost, structure is the number that governs — and structure is the dimension that has degraded.

Architecture Fundamentals and AI Guard-rails

Undisciplined adoption fails in three predictable ways. Architectural drift: agents optimise for the prompt, not the system, and drift accumulates one feature at a time until no engineer can reason about the whole. Compounding defects: AI-written code looks right because it reads right, and tests written by the same AI miss the same blind spots. A governance void: the vendor does not own the output, the engineer did not fully understand the tool, and the next hire inherits code with no author.

Any team can ship AI-written code fast. Very few can ship code the next hire will maintain, the next auditor will sign off on, and the next model release will not render obsolete. Five guard-rails separate the two:

EXHIBIT 5 Five guard-rails — the failure each one answers

Failure mode	Guard-rail
Models recommend deprecated infrastructure	A curated golden-path infrastructure context
Teams default to the most expensive model	Tiering rules with spend visibility
Agents re-implement functions that already exist	Enforced reuse and token-efficiency gates
Knowledge stays locked in long-tenured staff	An AI-mapped codebase; every change carries its plan

Failure mode	Guard-rail
Ungoverned, invisible LLM spend	One managed control surface for models, prompts and tools

The evidence for how much the guard-rails matter is a controlled comparison: two rebuilds inside a single organisation, same period, comparable team size, identical AI tooling, opposite discipline. Both shipped on time.

EXHIBIT 6 Two builds, same tooling — the architectural side-by-side

Dimension	Build A — UI over legacy	Build B — disciplined greenfield
API contract	Hand-written validation, no formal schema	Generated clients from a formal OpenAPI schema
Type discipline	Permissive rules, many escape hatches	Strict typing, generated types throughout
State management	Client-driven fetching and caching	Server-first, hydrated to the client
Authentication	Custom security middleware	Battle-tested third-party provider
Test coverage	Limited	Full end-to-end suite
Codebase volume	~2x for comparable scope	Baseline
Backend	Unchanged legacy monolith	Co-evolved, event-driven, multi-tenant
Maintenance debt	High, accumulating per feature	Low, absorbed by the system

The differences surfaced after launch, which is exactly why they are not caught in time. The undisciplined build carries materially higher defect density. Each new feature reinvents patterns the design system would have provided. The custom security layer becomes an audit risk at exit. And the deepest finding: the interface was modernised while the backend stayed untouched, so every original scaling complaint survived, and worsens as new front-end adoption drives more concurrent load onto the unchanged backend. Shipped, visibly newer, structurally the same asset.

Same organisation, same period, same tooling, comparable teams. The difference is architectural discipline. The discipline is the product; the tooling is downstream of it.

18–24 Month Flagship Product Redesign Delivered

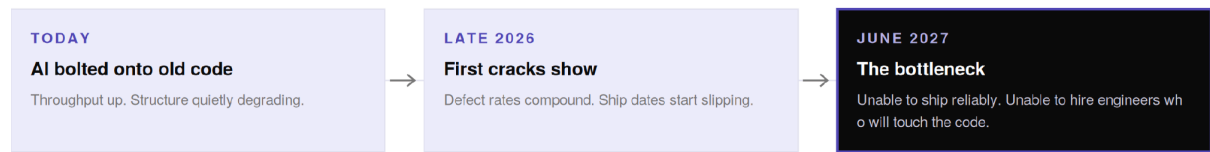
The disciplined build is also the evidence for the execution window. The modernised platform released at month 18, with client migrations running through month 24 and beyond. Classification: realised, stated without naming the company. Around it, on the same engagement: delivery ran about a month ahead of schedule after three prior rebuild attempts had failed; the senior team was retained against active poaching; monthly hosting cost fell ~80% on a comparable load, on a greenfield product (hosting only, not total cost of ownership; operational); and a key account's multi-year trust crisis resolved into a confirmed migration decision. One caveat carried openly: that account was later acquired and the programme stalled; the claim is the recovery itself, not the revenue that followed.

Release cadence on the delivered build ran at 6x against the sector norm: from the quarterly releases typical of legacy vertical software to 2 production releases a month, with four development releases a day. Cadence measures how often the product ships, not output per engineer, and this report attaches no per-engineer multiple and no operator-to-agent ratio. Cadence is the number a buyer can verify. Classification: operational, against an external norm.

EXHIBIT 7 From adoption to consequence

FROM ADOPTION TO CONSEQUENCE

The lag runs roughly 12 months. Most of the adoption happened in 2025–26.



A dated, falsifiable prediction. The window between adoption and consequence is what the operating model has to beat.

A DATED PREDICTION

Enterprise codebases bloated on undisciplined AI adoption will reach a structural bottleneck by approximately mid-2027: unable to ship reliably, unable to attract engineers willing to maintain them, unable to unwind the accumulated debt at the speed the market expects.

The lag between adoption and consequence runs approximately twelve months. The majority of enterprise AI adoption occurred during 2025 and 2026.

CHAPTER 5

Modernising in Place: Rebuild the Core Without Stopping the Business

A complete rip-and-replace of a legacy B2B platform is a death sentence inside a hold period. Enterprise customers are deeply integrated against existing APIs, and freezing the product for a multi-year rewrite triggers churn at exactly the wrong moment. The alternative is to modernise in place — and it is an authority problem before it is a technology one.

The Board Mandate

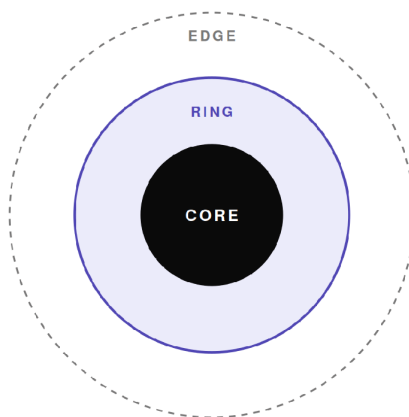
Without a board-level governance directive, modernisation fails. The mandate changes how enterprise deals are approved and delivered: technical leadership gains the authority to refuse custom code in the core product, and enterprise sales targets are decoupled from core engineering commitments. The test is simple. When a material customer asks for an exception in month 9, who decides, and does the commercial team already know the answer?

Core, Ring, Edge

The product is restructured into three strict layers. Customer requests are never refused; they are routed.

EXHIBIT 8 The three layers

THE THREE LAYERS



Customer requests are never refused; they are routed. Nothing lands in the core uninvited.

THE CORE — LOCKED

Product code only, standardised, identical for every customer. Strict typing, full test coverage. No single-customer edge case touches it.

THE RING — CONFIGURATION

Customer-specific need absorbed as rules and data, never code branches. Cleanly separated — exactly where agentic tooling builds custom configuration cheaply and safely.

THE EDGE — PRICED AND OWNED

Bespoke integrations, adapters, legacy bridges. Outside the main application, priced, under its own statement of work, with a named owner.

This forces the operational separation between scalable R&D and bespoke delivery, so the R&D budget stops subsidising one-off services and returns to building enterprise value. Three tests distinguish a real ring from a claimed one, because every vendor believes they already have a platform layer: what share of last year's commitments were satisfied inside it with no core change; whether a ring change can be made by someone who is not a core engineer; and whether a ring change can be removed without residue.

CHAPTER 6

Why No External Party Has Fixed This

Most holders of fatigued assets have tried already. A rebuild was funded. Consultants came. An offshore team was stood up. Any credible next attempt owes the room an account of why the previous ones failed that does not amount to calling anyone incompetent, because they were not.

- 1. In-house.** The team serving unhappy customers cannot be pulled off to rebuild the product those customers are complaining about. The rebuild starves or the service degrades, usually both.
- 2. Point AI at it.** Faster generation on the same architecture accelerates the bloat, for the reasons in Chapter 4.
- 3. Buy offshore volume.** Capacity at the lowest available rate, when capacity was never the constraint.

Offshore: Judgement, Not Rates

The usual objection is quality, and it is wrong. A low-rate bench executes specified work, often well. Architecture decisions are not specified work. They are judgement about consequence, made by someone who knows why the last three decisions were taken and will tell a customer that a requirement costs more than it is worth. That capability does not survive being written into a ticket.

So the question is never the rate. It is four conditions, and none of them is geography. First, overlap — and full-day overlap is the wrong test, because it describes a team that needs supervising. An hour a day with the stakeholders who make product decisions is sufficient,

provided the team owns the work between those hours: owners in the product development process, not engineers taking orders. Second, seniority in the only sense that matters here — the ability to form an opinion and hold it under executive or customer pressure. A senior who folds when the account director pushes is a junior with tenure, and the boundary falls exactly there. Third, employment into the company itself, with code ownership and customer-facing accountability, rather than a vendor bench. Fourth, seniority as the unit purchased, because capacity pointed at an unowned intake produces absorbed work faster. Markets meeting all four exist at 40–55% of tier-1 cost, at top-of-market rates locally. The saving is geographic. The seniority is not discounted.

Follow the Incentives

The deeper explanation is structural. Every conventional remedy is sold by a party whose economics require the problem to persist. An outsourcer’s margin depends on junior headcount staying billable: more heads and longer engagements are the product, not a side effect. A consultancy is paid for the report, and then for the next report. Neither is acting in bad faith; both are acting rationally.

The boundary collapses under commercial pressure because nobody in the room is compensated on whether it holds. Which is why the incentive structure of whoever executes the transformation matters more than their methodology. A firm paid by the day carries no exposure to whether the discipline survives month 14. A firm that keeps the delivery team on its own books is selling an annuity, and the annuity requires the core to remain absorbent.

CHAPTER 7

True SaaS Margin Shape: What the Re-Rating Is Worth

The financial argument does not rely on aggressive growth or a favourable exit market. It rests on a mechanical EBITDA improvement, funded by redirecting delivery OPEX the asset is already spending, with the re-rating as upside rather than baseline.

What Moves the Multiple: The Published Drivers

IT services trade at around 1.1x revenue and non-SaaS software at 2.3–2.6x, against 3–3.8x for SaaS in private M&A; services carry 20–35% gross margin against 75–80%. The drivers below are published on revenue multiples, not EBITDA, and Exhibit 9 shows them as published.

EXHIBIT 9 What the published drivers move · revenue multiples · external

Driver	Published comparison	Source
NRR above 120% vs below 100%	11.7x vs 4.1x revenue	SEG, public, Q4 2024
Gross margin above 80% vs below 60%	7.2x vs 3.5x revenue	SEG, public, Q2 2025
Gross margin above 80% vs below 80%	7.6x vs 5.5x revenue	SEG, public, Q4 2024
SaaS vs non-SaaS software	+19% to +52% on revenue	Aventis, private M&A, 2024–Q1 2026
Rule of 40, +10 points	About +1.0x revenue	Aventis, public, Q3 2025

Each driver is measured on its own and they are not summed; stacking them is how modelled work becomes dishonest. They are revenue effects. On EBITDA the penalty compresses, because low gross margin shrinks profit as well as price.

The companion case prices its company on that basis: entry at around 1.5x revenue, exit at 10–11x EBITDA, deliberately below current private medians. Modelled.

Separability and Self-Funding

- **Separability.** Once custom work lives outside the core, a buyer underwrites product and services each on its own terms rather than discounting the blend. The perpetual maintenance obligation, usually unpriceable and therefore priced punitively, acquires a defined scope and a cap.
- **Self-funding.** The delivery pool — R&D, hosting, infrastructure, services cost of goods — runs at 36–44% of revenue in assets like this. Resolution is paid for out of spend already being lost. A property of the problem, not a pricing device.
- **Attribution.** What is rewarded inside a sponsor is attributable improvement. A gain nobody can trace to a decision funds nothing at the next investment committee, which is why a measurement baseline set in the first 100 days is worth more than a larger gain nobody can allocate.

CHAPTER 8
The Starting Move: The Three-Pile Audit

Do not begin with a consultancy or a model of future multiples. Begin by exposing the true cost of delivery hiding inside the current operation. It takes an afternoon.

Pull the active R&D backlog and the engineering sprint board for the last 90 days. Require engineering and product leadership to sort every ticket, without exception, into one of three piles.

EXHIBIT 10 The three-pile audit

Pile	Definition	What it signifies
1. Standard product	Features applying natively to all customers, zero bespoke configuration	Scales at near-zero marginal cost. This is what a software multiple pays for
2. Single-customer bespoke	Code written for the workflow, integration or onboarding of exactly one customer	Services delivery, funded from the R&D line and invisible in the revenue mix
3. Break-fix and technical debt	Remediation required to keep the platform stable	Usually the codebase buckling under the weight of Pile 2

In a healthy pure-play SaaS business, Pile 1 dominates engineering capacity. If the audit shows bandwidth consumed by Piles 2 and 3, the financial reality is settled: the board may believe it holds a product company, but the R&D budget is operating a services business. Most companies cannot produce the sort at all. The ones that can usually do not like the third pile.

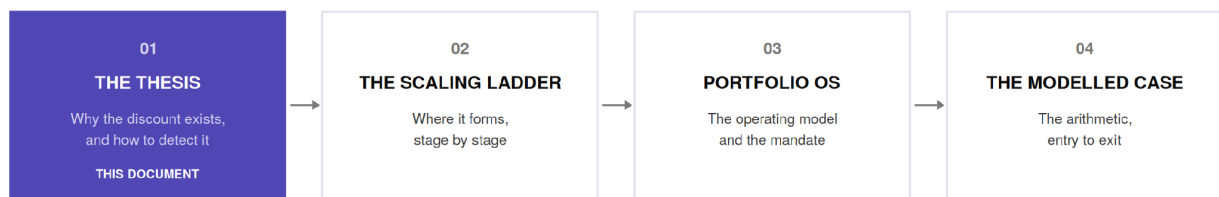
NEXT STEP: DOWNLOAD THE OPERATING MODEL

Run the three-pile audit. If Piles 2 and 3 dominate, the condition in this report applies to your company, whatever the services line says.

The operating model that closes it — the four steps, the transition mechanics, the engagement economics — is set out in a companion document: renaissance.tech/research/portfolio-os. The arithmetic, run line by line on a constructed company, is in a second.

Or bring the audit result to a 30-minute diagnostic conversation. Sponsors and founders enter the same sequence; only the scoreboard differs.

EXHIBIT 11 Where this document sits



Read in either direction: the ladder locates the stage; the thesis prices it; the operating model resolves it; the case shows the arithmetic.

Precision beats power. Alignment beats scale. Truth beats urgency.

About the figures. Every figure carries one of four classes: realised (cash or contractual outcome, verifiable against records); operational (measured system improvement on comparable workload, not traced to P&L); modelled (calculated against a stated baseline — an estimate, not a banked saving); externally verifiable. The classes are never summed. Multiple-expansion figures are modelled. From one engagement, stated without naming the group, its sponsor or its customers: the 18–24 month window and the key-account recovery are realised; the 6x release cadence, against the sector norm, and the ~80% hosting reduction, on a comparable load on a greenfield product and covering monthly hosting only, not total cost of ownership, are operational. Full bases: a separate evidence note, available on request. Valuation multiples are cited on the basis their publisher reports them, revenue or EBITDA, and never mixed. No published dataset cuts multiples by professional-services share; the link between services-heavy delivery and the discount is drawn through gross margin and retention.

A Renaissance Group company.

Sources. Valuation compression and sector bifurcation, Q1 2026: Aventis Advisors SaaS valuation series; Bain & Company software commentary, February 2026. Code structure: GitClear, The Maintainability Gap: AI Code Quality in 2026 (623M analysed changes). Developer productivity: Becker, Rush, Barnes and Rein, METR, July 2025 (arXiv:2507.09089), and METR's 2026 revision. Operating benchmarks: SaaS Capital, 2026 Spending Benchmarks for Private B2B SaaS; Benchmarkit, B2B SaaS Performance Metrics; SaaSRIse, 2026 SaaS Benchmarks Report; Aleph, ARR per Employee 2026. Valuation: Software Equity Group, 2Q26 SaaS M&A and Public Market Report and gross-margin and NRR cohort analyses (2024–2025); Aventis Advisors, SaaS, Software and IT Services Valuation Multiples 2015–2026; Hampton Partners, Enterprise Software M&A Report 2H2026; Multiples.vc, Public Software Valuation Multiples (October 2026). Scaling stages after Acquisition.com's published \$100M scaling roadmap, re-expressed for vertical B2B software.

Important information

This report is for general information only. It is not investment, financial, legal or tax advice, and it is not a recommendation about any particular company. Market figures are published third-party data, cited as their publishers report them. Modelled figures illustrate stated assumptions; they are not forecasts. Results from past engagements do not predict results elsewhere. Nothing in this report is a promise or guarantee of any outcome. Take advice from your own advisers before acting on it.

The full terms, including how figures are classed and sourced, are at www.rein essence.tech/legal.