Whitepaper

Host best practices

How executives create the conditions for fast product development

FTTM methodology series.

A lateralworks Fast Time To Market (FTTM) methodology paper — the 13 host practices that separate fast organizations from slow ones. Revised edition.

Prepared by
lateralworks
FTTM methodology

Date
April 2026
Revised edition

Online
lateralworks.com
FTTM series

Table of contents

Host best practices

Core thesis. Teams deliver products; the host creates the conditions under which they deliver fast. Fix the host and speed becomes the default. Leave the host alone and team-level improvement fades within a year.

Overview
Abstract

Most companies chase faster product development by pushing the team harder. Training, project managers, new tools, tighter schedules — very little improves. The reason is structural. A project team operates inside an environment, a **host**, and the host decides what the team can actually do. A slow host produces slow teams regardless of how skilled those teams are. A fast host lets otherwise-ordinary teams move quickly.

This paper describes the 13 host practices that lateralworks has found, across more than three decades of consulting with technology companies, to separate fast organizations from slow ones. The practices fall into three groups: the **mindset** executives hold about speed, the **environment** they build around teams, and the **portfolio** of projects they manage. Each practice is presented as a contrast: what normal organizations do, and what fast ones do differently. The difference is usually not effort or talent. It is design.

The practices come from direct observation of fast-to-market teams, beginning with a multi-company study of more than 500 practitioners in the early 1990s. They connect to a broader body of research: Reinertsen on product development flow [2], Wheelwright and Clark on team structure [11], Saaty on prioritization [14], House and Price on break-even time at Hewlett-Packard [5], and Ohno on waste elimination [1].

The paper is written for executives who suspect their company is slower than it should be and want a concrete checklist. It is also written for product leaders and program managers who need to make the case to their executives that the environment, not the team, is the constraint. **Appendix A** presents an anonymized case study drawn from a large storage technology company that applied this assessment methodology to its own host.

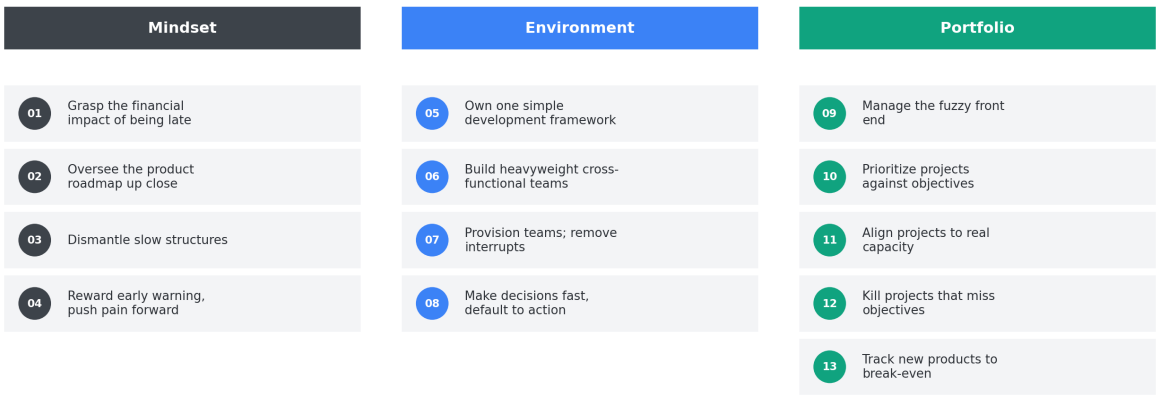


Figure 1. The 13 host best practices at a glance. Mindset, Environment, and Portfolio form a single interlocking system; weakness in one layer undermines strength in the others.

Figures

List of figures

This paper uses figures as first-class explanatory elements rather than decoration. Every figure summarizes or extends a claim made in the surrounding prose and is designed to stand on its own. The list below points to every figure in the paper, in the order it appears.

01

Foundation

The host: what it is, why it matters

A product development team is never the whole story. It sits inside a larger organization: the functional hierarchy, the executive staff, the business units, and the support groups. That organization is the team's **host**. The host decides what resources the team gets, what decisions it is allowed to make, what information reaches it, and what is expected of it. Host behavior either speeds the team up or slows it down; the middle ground is rare.

Two kinds of host behavior dominate the relationship between the organization and the team. The host either **provisions** the team by giving it what it needs to move forward, or it **interrupts** the team, taking away momentum the team had built. Fast organizations are usually not the ones with the most talented individuals; they are the ones whose hosts reliably provision and rarely interrupt.

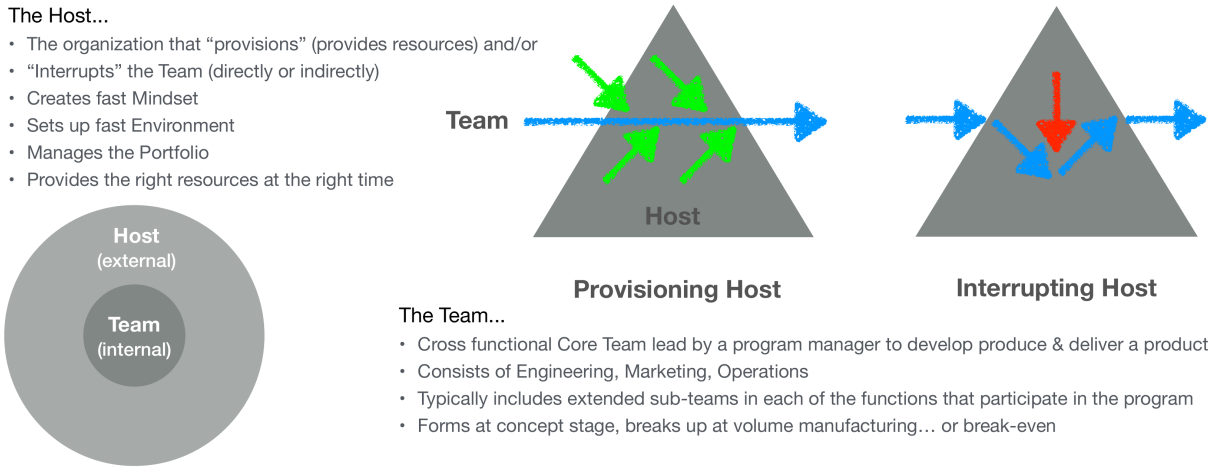


Figure 2. The host is the organization that surrounds the team. Its job is to provision the team — feeding resources, information, and fast decisions into the team's forward path. When it fails, it interrupts the team — deflecting the path and forcing lost time and rework. Same team, two different hosts, two different outcomes.

The idea has a long lineage. Taiichi Ohno built the Toyota Production System around teaching managers to see waste: the time a workpiece spent sitting idle, waiting for something a manager had failed to provide [1]. Donald Reinertsen makes the same point for knowledge work. The cost of delay is nearly always larger than the cost of the resources that would have prevented the delay, and organizations still under-invest in preventing it [2]. Clayton Christensen extended the argument to resource allocation: decisions about who gets what, and when, predict outcomes better than the tactical choices teams make about how to execute [3]. The environment shapes the outcome, and executives design the environment whether they notice they are doing it or not.

Three layers of the host

The 13 practices organize into three layers. The foundation is the **environment**: the development framework, the organization structure, and the decision-making system. Above that sits the **portfolio**: the set of projects the organization has chosen to work on and the discipline that keeps the set prioritized, aligned with real capacity, and pruned when projects stop making sense. At the top sits the **mindset**: the beliefs executives hold about speed, cost, failure, and what good management looks like. Mindset is the hardest layer to change and the most determinative; environment and portfolio follow from it.

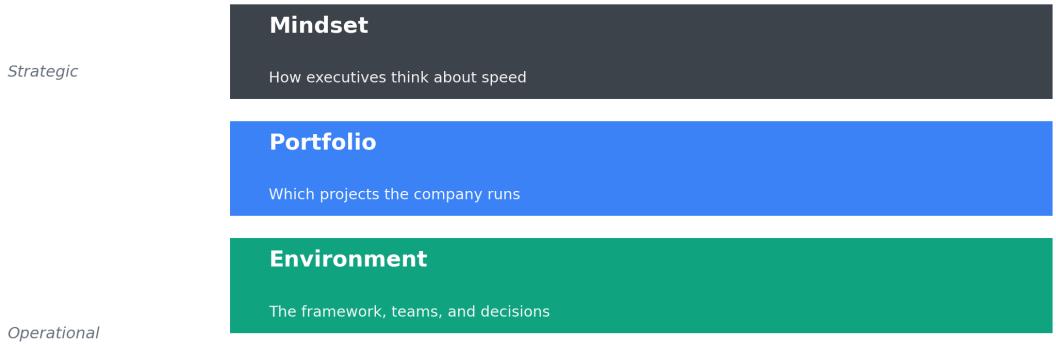


Figure 3. The three layers of the host. Mindset sits on top because it shapes every decision below it. Environment is at the base because it is where the day-to-day work happens.

The FTTM framework: host and team

The 13 host practices in this paper are half of the full FTTM system. The other half lives on the team side, documented in the companion lateralworks *Team Best Practices* paper [34]. Both halves sit under a shared mindset layer. The three-way split is the shape of the full framework.

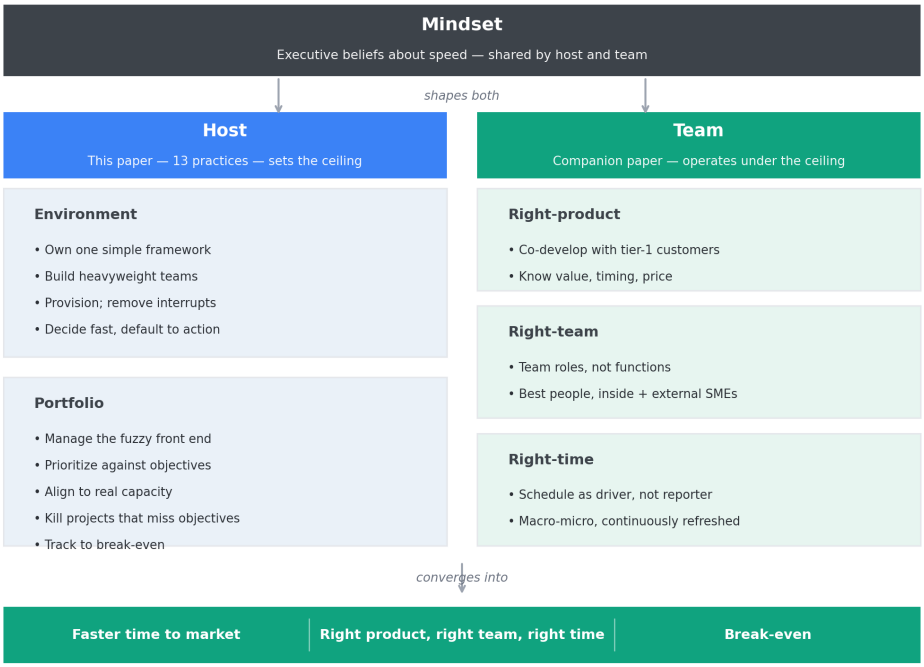


Figure 3b. The full FTTM framework. Mindset is shared between executives and teams and shapes everything below it. Host practices (the focus of this paper) set the ceiling on what a team can achieve. Team practices (the companion paper) describe how the team operates under that ceiling — right product, right team, right time.

This paper focuses on the host side — the practices executives and functional leaders control, which set the organizational ceiling on delivery speed. The team side covers the practices the project team itself controls:

how it co-develops with tier-one customers, how it translates requirements, how it builds and runs its schedule, how it plans aggressively and refreshes continuously. A reader who wants the full FTTM picture should read the two papers together.

The split matters. When a delivery problem surfaces, the host typically looks to the team for an explanation, and the team typically looks to the host for a resource or decision. Naming both sides and documenting both sets of practices makes the conversation tractable: each party knows which practices are theirs to improve, and which belong to the other side of the relationship.

Host provisioning in depth

Provisioning is the active, deliberate work of giving a product development team what it needs to move forward. A host does not just *permit* a team to do its work; it *provisions* it, the way a ship is provisioned before a long voyage: a clear inventory of what the team will need, sourced ahead of time, and delivered on the team's schedule rather than the host's convenience.

In fast organizations, provisioning is not a side activity of the executive staff. It is the executive staff's primary responsibility to the project teams. Executives who touch a product line know what a day of waiting costs the company in dollars, not as a slogan but as a number they can quote.

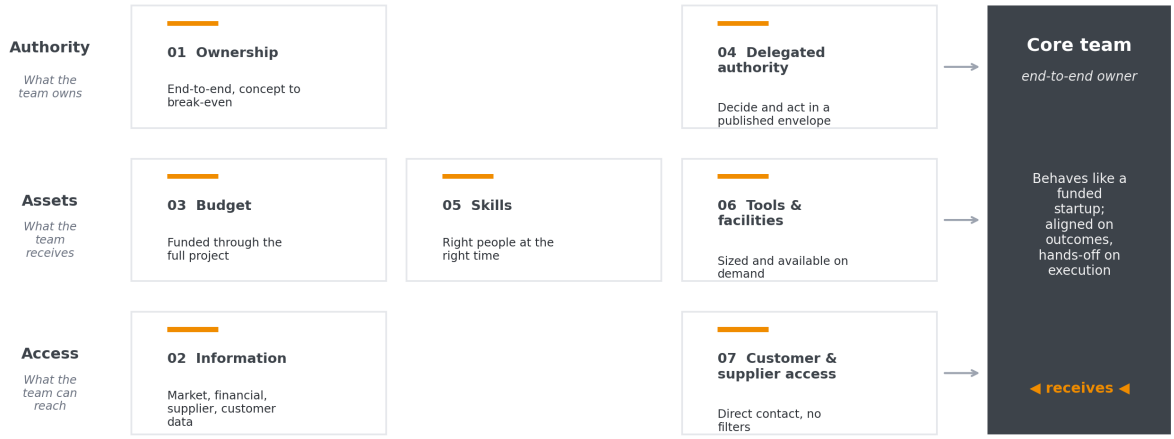


Figure 4. Seven forms of provisioning that the best hosts deliver to a core team. The team behaves like a funded startup — because the host has set it up that way.

Ownership of the product

The team owns the product end-to-end, from concept through break-even, rather than owning a piece of it or handing work to another function. The host treats the team the way a venture investor treats a portfolio company: aligned on outcomes, hands-off on execution.

Information

Market data, competitive intelligence, financial targets, supplier roadmaps, customer commitments, inter-company dependencies: the team receives all of it on time, without having to ask twice. Information

hoarding by functions is one of the most destructive host behaviors.

Budget for the duration

The team has a budget it can count on through the end of the project, not a budget renegotiated every quarter. Budget uncertainty forces teams to pad estimates, hoard resources, and avoid the small, well-placed risks that accelerate delivery.

The right skills, at the right time

The team has the people it needs when it needs them, not three months late or on loan for eight hours a week. Skill provisioning requires the host to forecast and pre-position people, which requires an honest picture of its own capacity. Most slow organizations lack that picture.

Tools, test equipment, and facilities

Engineering tools, test equipment, lab space, and collaboration facilities are sized to the project and available when the team needs them. In a fast host, nobody is negotiating time on a shared test rig.

Delegated authority

The team has the authority to decide and act on anything required to stay on or ahead of schedule, within a published envelope of business direction and technology policy. Without it, everything escalates and everything waits. **Appendix B** gives the full empowerment matrix that fast organizations use to make this envelope explicit.

Direct customer and supplier access

The team talks to customers and suppliers directly, not through sales managers or procurement intermediaries. Every filtering layer adds days or weeks to the feedback loop and injects its own opinions into what the team hears.

> **The provisioning standard.** A provisioned team operates like a funded startup. The host acts like a board of directors plus a shared-services group: committed, informed, helpful, and mostly out of the way.

Host interrupting in depth

An interrupt is anything the host does, or fails to do, that takes momentum away from the team. Interrupts are rarely dramatic and rarely intentional. They are usually a byproduct of structures built for another purpose (cost control, audit readiness, functional specialization, hierarchical decision-making) that keep running regardless of what they cost a given project.

A slow host does not see its own interrupts. It sees missed schedules and attributes them to team execution; it sees over-budget projects and blames estimating; it sees quality problems and points to engineering rigor. What it does not see is the fourteen-signature approval chain, the silently redirected project resources, or the week-long audit. These things are normal to the host, the texture of how the company operates. They are also the reason the project is late.

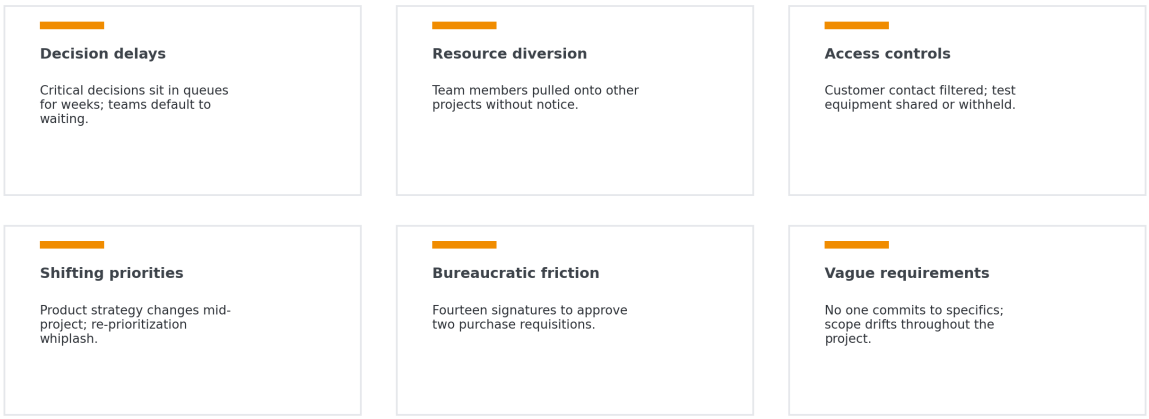


Figure 5. Six recurring categories of host interrupt. All of them are fixable. None of them are the team's fault. Most of them are cheap to eliminate once they are named.

Specific examples from lateralworks engagements

The examples below come from real lateralworks engagements, anonymized. Each one looks small in isolation. Together, they explain how a technically capable team full of smart, motivated people still ships a year late.

Fourteen signatures, four months

The host required 14 signatures and roughly 4 months to approve two engineering purchase requisitions on the project's test plan. The purchases were not controversial, sat inside the approved project budget, and were required by a schedule the executive team had committed to. The signature chain had been designed for audit control; cycle time had never been part of its design. The team lost four months on the critical path, and nobody in the chain considered themselves responsible.

Silent resource diversion

During a business downturn, functional managers redirected critical project resources away from the team without notifying the program manager. The team kept planning against a resource level that no longer existed. The shortfall surfaced at a schedule review three months later; by then the project was eight weeks behind, and the only recovery path was compression that later proved unsustainable.

Audits as denial-of-service

Executive confidence in the team was low, so the host commissioned periodic audits and information requests that shut down productive work for days at a time. The audits surfaced nothing new; they reproduced, expensively, what was already in the team's weekly status reports. The auditing became the reason the next milestone slipped, which justified another audit.

Artificial gates as control theater

The host imposed phase gates and review meetings as a way to stay involved in daily decision-making. The gates did not gate anything; every project passed every one. They operated as reporting occasions, and each cost the team two weeks of preparation and one of recovery. Removing the non-functional gates recovered about six weeks per year per project with no measurable decline in decision quality.

Slow central decisions on fast-moving technical choices

Teams that had framed the decision cleanly (options plus recommendation) waited three to six weeks for a response. By the time the response arrived, market conditions had shifted and the chosen option was no longer the best, forcing another analysis round and another delay. The team eventually stopped escalating and made decisions quietly; that preserved speed but raised the risk of strategic drift.

Whiplash re-prioritization

The host changed product strategy and re-prioritized the "key" projects throughout the project, usually in response to the most recent customer complaint or competitor announcement. Each re-prioritization forced the team to rebuild its plan. The net effect was substantial planning overhead, a frustrated team, and a product that ended up close to the original specification rather than to any of the interim revisions.

Artificial limits on customer contact

Marketing and sales restricted the team’s direct contact with customers. Customer feedback reached the team through sales managers who summarized, filtered, and occasionally fabricated what had been said. The team built features for a fictional customer. At beta, the real customers said something different.

Perpetually vague requirements

Requirements stayed vague because no one in the host would commit to specifics. "We’ll know it when we see it" became the operating model. The team built prototypes, got feedback that contradicted the prior round, rebuilt, and repeated. Each cycle was productive engineering. Aggregated, it was a schedule doubled by requirements drift that a disciplined front end would have resolved in weeks.

Why interrupts persist

Interrupts persist because each one serves someone’s purpose, even when no one intended the aggregate result. Every signature in the 14-signature chain exists because some senior manager once wanted visibility. Every phase gate exists because some project failed and a gate looked like a remedy. Every customer-contact restriction exists because an engineer once said something the sales team wished they had not. Removing these structures means admitting they are not working and negotiating with whoever installed them — two awkward conversations, which is why the structures accumulate.

The lateralworks approach makes the accumulation visible. Each team logs its top three active interrupts weekly. The executive review board acknowledges each one and commits to resolution within seven days. Items that are not resolved in seven days become explicit improvement projects owned at the executive level. Visibility alone fixes a surprising amount; accountability fixes most of the rest.

> **Warning sign.** When executives describe their teams as "never delivering on time" and the teams describe the executives as "the main impediment to speed," the host is interrupting the team. That is not a team problem. It is a host problem, and team-level improvement will not resolve it.

02

Mindset

How executives think about speed

Four practices describe what fast-company executives believe about speed. The beliefs are not motivational. They are economic. They are about how much money is lost when a product is late, about who owns the roadmap, about what kinds of structures actually slow work down, and about when in a project it is best to feel pain.

1. Grasp the financial impact of being late

The executives who run the fastest organizations can describe, in dollars, what a day of schedule slip costs the company. Not in the abstract, not as a percentage. In dollars. They have done the math on the product's market window, its ramp, its expected margin, and the shape of the demand curve. They know that a product that arrives one quarter late does not earn a quarter less; it often earns half as much or less, because it misses the steep part of the adoption curve and enters a market that is already partly supplied. Reinertsen calls this the "cost of delay" and argues that it should be the single most important number used to make trade-off decisions during development [2].

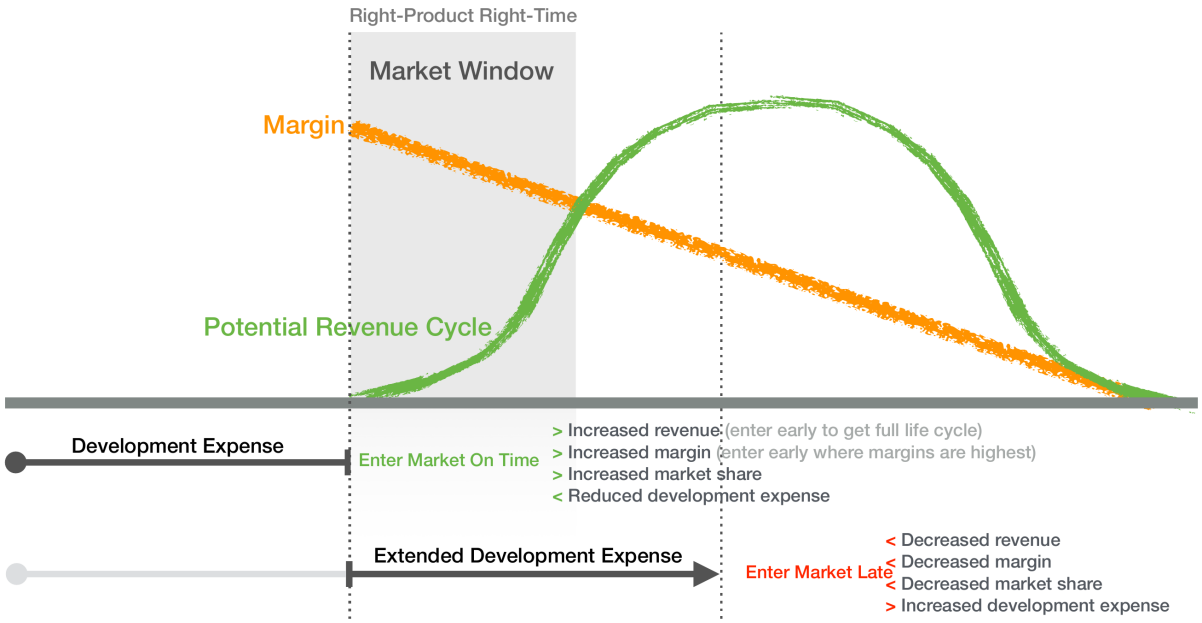


Figure 7. A product that misses the market window captures a smaller, compressed revenue cycle. The lost revenue and shortened life are not recoverable by cutting development cost later.

Normal organization

Schedule slips are treated as unfortunate but normal. "We don't want to ship something that isn't ready." Cost and quality are always the reasons for the slip, and they are accepted because the alternative (the cost of being late) has never been quantified. When a project slips, the team adds resources late, which raises development cost without recovering the revenue that has already been lost. The business runs on the implicit assumption that slip is free.

Best organization

The cost of delay is quantified before development starts and reviewed at every major checkpoint. Executives know what a day is worth in dollars. When a trade-off must be made — pay for an extra engineer or slip a week — the math is explicit and the answer is usually "pay." The business operates on the explicit principle that a day saved is worth more than a dollar saved, because the day cannot be bought back.

Connection to the broader literature. Smith and Reinertsen's *Developing Products in Half the Time* [4] showed that for most technology products, a six-month delay to market is more damaging to lifetime profit than a 50 percent development cost overrun. Hewlett-Packard's internal Break-Even-Time (BET) metric [5] operationalizes this by tracking a product from idea to the point at which cumulative cash flow crosses zero — the executives who ran the best HP divisions in the 1990s managed to BET, not to development cost.

2. Oversee the product roadmap up close

Fast companies have one person (an executive, not a committee) who owns the product roadmap. That person can be held accountable for what the company chooses to build. Everyone below them knows who recommends, who must agree, who provides input, who decides, and who is responsible for executing each decision. The roadmap is refreshed on a regular cadence, usually monthly, and the refresh is driven by customer input rather than internal opinion.

Slow companies diffuse roadmap ownership across marketing, engineering, and business-unit leadership. The decision-making process is complex enough that only a few people understand it, and when things go wrong no one can be held responsible. This ambiguity is comfortable: it protects careers. It is also expensive, because in the absence of a clear owner, the roadmap defaults to whatever the loudest person in the room asked for last.

Normal organization

Roadmap responsibility is shared between marketing and engineering and sometimes product management. No single executive can say yes or no on behalf of the company. The planning cycle is annual and produces a thick document that is out of date by the time it is signed. Customer input is filtered through internal experts who "know what customers really need." Decisions default to the CEO as tie-breaker, and the CEO does not have time to understand the details.

Best organization

One executive owns the roadmap, with clearly defined recommend / agree / input / decide / execute (RAIDE) roles for every step. The roadmap is refreshed monthly, fluid, and tied directly to customer signals and competitive data. Prototypes validate assumptions on paper before they become commitments. A feedback loop from failure analysis feeds back into roadmap revisions.

Connection to the broader literature. The RAIDE model of decision rights is a descendant of the RACI framework used in process management [6]. Bain & Company's research on decision effectiveness [7] showed that companies where decision rights were explicit outperformed their peers by a factor of roughly two on measures of execution speed. The ambiguity that slow companies tolerate is not neutral — it is measurably expensive.

3. Dismantle slow structures

Most organizations, faced with delivery problems, respond by adding control. More phase gates, more sign-off levels, more reporting templates, more PMO checkpoints. The intuition is that tighter control produces better outcomes. The evidence is the opposite: past a certain point, control slows work down without improving quality, and the cost of the added control dwarfs the cost of the problems it was meant to catch.

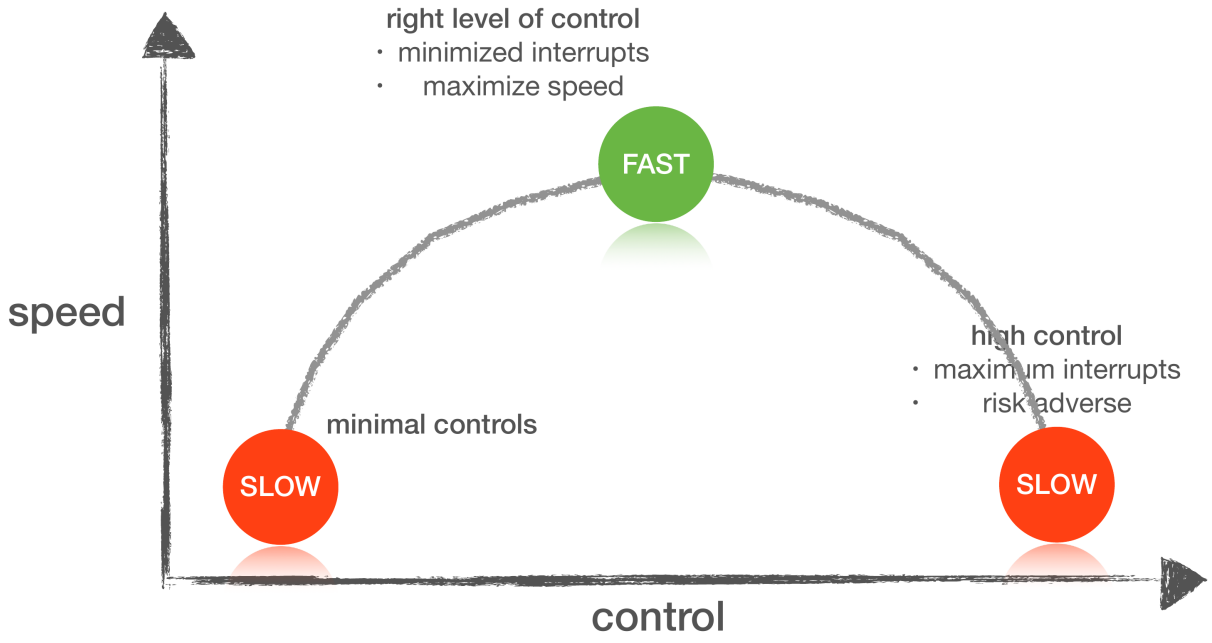


Figure 8. Speed does not rise monotonically with control. Organizations with too little structure move slowly because nothing is coordinated; organizations with too much structure move slowly because everything requires approval. The fastest organizations sit in the middle — enough process to prevent chaos, not so much that it becomes the work.

Fast organizations treat their own management systems as products that need continuous improvement. They inventory the interrupts that teams experience — missing resources, missing decisions, missing information, shifting priorities — and they systematically remove them. They raise approval thresholds so lower-level managers can authorize spending without waiting in a queue. They map their business processes end-to-end and cut the steps that add no value. They report by exception, leaving projects alone when they are within tolerance of their targets.

Normal organization	Best organization
The dominant behavior is cost control. Development is governed by heavy phase-gate life cycles that were designed to limit spending, not to accelerate delivery. A PMO exists to enforce the gates. Teams spend substantial time feeding information into tracking systems that few people read. When projects run late, more money is thrown at them — by then it is too late to recover, because there was no early warning. Removing the interrupts would eliminate a convenient source of excuses, so no one removes them.	The default is speed, not control. Executives know that the cost of delay far exceeds any savings from tight OPEX governance. Interrupts are inventoried and eliminated. Approval levels are raised. Reporting is by exception. Process improvement is horizontal, mapping the end-to-end flow rather than vertical silos. Control is pushed down to the team, which moves faster and, counter-intuitively, spends less.

Connection to the broader literature. Hamel and Zanini's *Humanocracy* [17] synthesizes survey data from more than 10,000 managers and estimates that excess bureaucracy costs the US economy approximately $3 trillion per year in lost output — more than the GDP of most G20 economies. Graeber's *The Utopia of Rules* [18] documents the same pattern from a different angle, showing that rule-making expands faster than the problems it is meant to solve, because adding a rule is cheap for the executive who proposes it and expensive for every team that must comply with it thereafter. Both lines of research support the same conclusion: control structures accumulate by default, and dismantling them requires deliberate, sustained executive effort.

4. Reward early warning, push pain forward

Every development project has a certain amount of pain built into it. Problems that will be discovered, trade-offs that will have to be made, assumptions that will turn out to be wrong. The only question is when that pain will be felt. Normal organizations push it into the future. Problems are minimized, risks are downplayed, aggressive forecasts are rewarded, and the person who raises a concern early is labeled a non-team-player. The pain arrives anyway, usually at the worst possible moment, close to the target ship date, when there is no time left to recover.

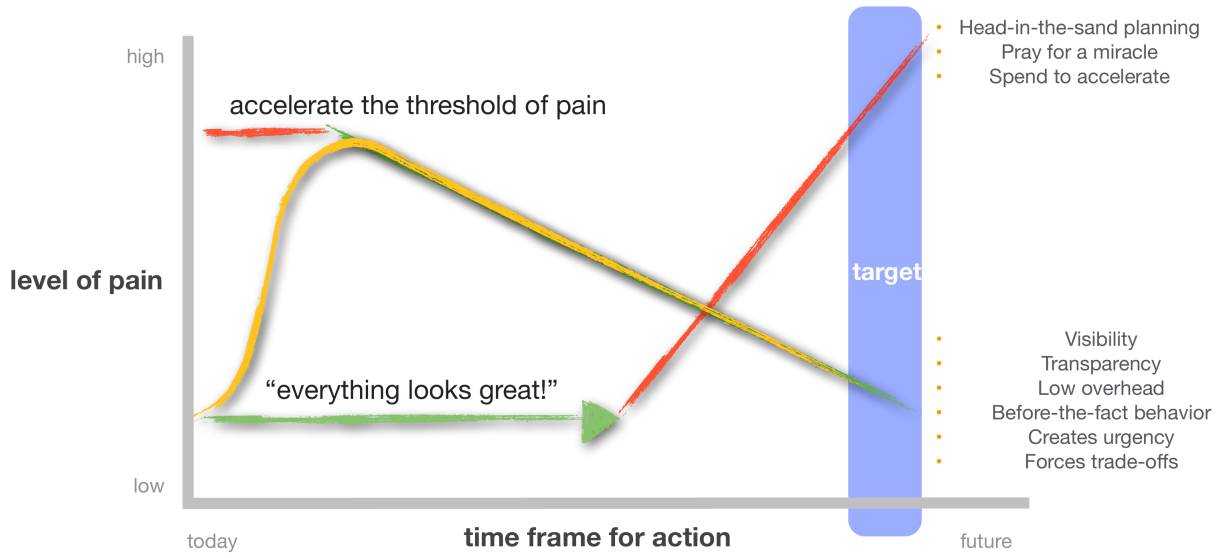


Figure 9. In normal organizations, everything looks great until the crisis hits right before ship. In best-in-class organizations, pain is pulled forward into the project's early phases, where there is time and money to respond to it.

Fast organizations invert this. They build management systems that reward people for identifying problems early. They treat failure as a source of learning rather than blame. The R&D budget includes explicit funding for learning cycles, because the team that has budgeted to fail three times on its way to a working solution will get there faster than a team that is trying to succeed on the first try.

Normal organization

Pain is pushed later. Pointing out future problems is seen as defeatist. Aggressive forecasts and "can-do" optimism dominate the early phases. When the inevitable problems arrive late in the project, they are attributed to unforeseeable technical difficulty. Learning from failure is discouraged because failure implies blame.

Best organization

Pain is pulled forward. Management systems reward people for identifying potential problems before they materialize. Learning cycles are budgeted into the schedule. Transparency and honest reporting are encouraged and rewarded. Teams live by the rule "the sooner we find it, the more time we have to fix it."

Connection to the broader literature. Eric Ries's *The Lean Startup* [8] formalized the idea of learning cycles as a first-class budget item. Edmondson's research on psychological safety [9] showed that teams which reliably report bad news early outperform teams that protect their leaders from it, sometimes by large margins. The practice of pulling pain forward is a deliberate act of engineering psychological safety into the management system rather than leaving it to individual manager personalities.

03

Environment
Where teams live

The environment is where teams live. It is the development framework they are required to follow, the organizational structure they sit inside, and the decision-making systems they interact with daily. An executive mindset that is fully aligned with speed cannot produce fast outcomes if the environment is built to slow work down. Environment is where mindset becomes material.

5. Own one simple development framework

A product development framework should fit on one page. It should describe the end-to-end flow from idea to end-of-life, as a single integrated system rather than a sequence of organizational hand-offs. And it should be owned by one person — not a committee, not a rotating chair — whose job it is to continuously redesign the framework to make development go faster.

Slow organizations accumulate frameworks the way ships accumulate barnacles. Each generation of leadership adds its own phase gates, its own review boards, its own documentation requirements. Nothing is removed, because no one owns the whole thing. The resulting process is complex enough that individual contributors stop trying to understand it and instead try to work around it.

Normal organization	Best organization
The development framework is complex, owned by no one, and designed primarily to control spending and concentrate decision-making at the top of the hierarchy. Platforms and reuse are discussed but not implemented, because implementation would require orchestration across silos that does not exist. Life cycles are rarely redesigned; when they are, the work is done by people on the sidelines and is never adopted.	The development life cycle is owned by one person or function, reports to the executive staff, and is continuously redesigned to reduce friction. The framework fits on one page. It is macro and flexible — all projects look the same at the top level, but each team fills in its own detail. Design for reuse, platforms, and fast derivatives are built into the framework itself rather than being retrofitted project by project.

Connection to the broader literature. Womack and Jones's description of value-stream mapping [10] makes the same case from the manufacturing side: a value stream has an owner whose job is to remove waste across functional boundaries, because no individual function is in a position to see the whole flow. The lateralworks insight is that the same logic applies to product development — the framework needs a value-stream owner, not just a PMO.

6. Build heavyweight cross-functional core teams

Not every cross-functional team is the same. Wheelwright and Clark identified four different organizing forms for product development teams [11]: functional, lightweight, heavyweight, and autonomous. They are not interchangeable. Each sits at a different point on the curve of speed versus team ownership, and the fastest organizations deliberately choose heavyweight or autonomous structures for their most important projects.

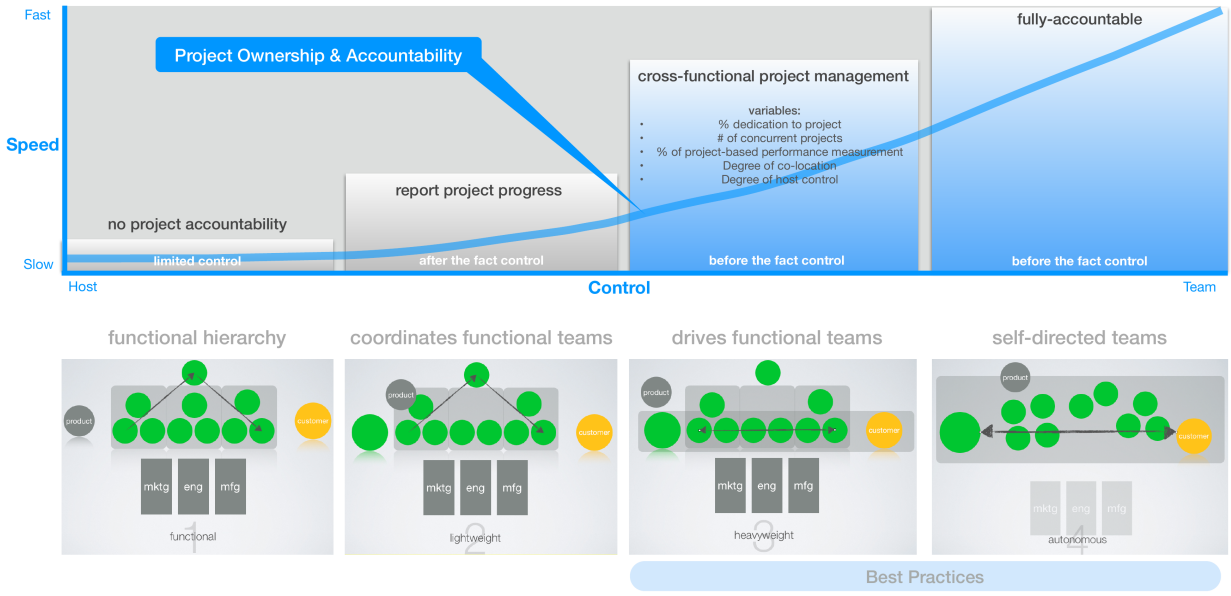


Figure 10. The four team structures, adapted from Wheelwright and Clark. Functional teams have no lateral flow. Lightweight teams coordinate but do not drive. Heavyweight teams drive the project across functions. Autonomous teams operate as startups inside the company. Best practice is heavyweight or autonomous on the projects that matter most.

The word "heavyweight" refers to the authority of the team leader and the dedication of the team members, not to the number of people on the team. A heavyweight team is led by a senior engineer or program manager with enough technical and business experience to make difficult trade-off decisions in real time. The team is dedicated (ideally 100 percent) to a single project, and is colocated, at least at the core. Team members receive the majority of their performance review from the team leader, not from their functional manager. The power structure shifts from the functions to the team.

Normal organization

Cross-functional teams exist in name only. Members are on eight to fifteen projects simultaneously, take direction from their functional hierarchy, and attend team meetings as a courtesy. The "real work" happens back in the functions. Core teams become bloated with 20+ part-time representatives. Nobody is responsible for the integration of functional deliverables into a product. The team's role is coordination, not delivery.

Best organization

Heavyweight teams form early (during feasibility) on the projects that matter. Members are dedicated, colocated, and led by a senior program manager with real authority. A single engineering leader owns technical trade-offs across the system. Performance reviews flow through the team. The team behaves like a startup: it owns the product, it shares in the reward of success, and it is accountable for failure.

Connection to the broader literature. Wheelwright and Clark's *Revolutionizing Product Development* [11] documented a factor-of-three speed difference between lightweight and heavyweight teams across a large sample of Japanese and American auto and electronics manufacturers. Toyota's chief engineer system, studied by Morgan and Liker [12], is the canonical example of a heavyweight structure: a single engineer owns a vehicle program end-to-end, with authority that cuts across the functional hierarchy.

7. Provision teams; remove interrupts

The foundation of this practice is laid out in detail in the two earlier sections on host provisioning and host interrupting. The short version: even the best heavyweight team can be paralyzed by an interrupting host. If decisions take weeks, if resources arrive late, if test equipment is shared with five other teams, if customer access is gated by a sales function that does not trust engineers, then the team's structural advantage is spent on working around the organization rather than on delivering the product. **The host is the ceiling on what any team can achieve.**

Normal organization

The host views the team as a mechanism for hand-offs between functions. It controls the budget, stretches resources too thinly across too many projects, and provides information, decisions, and customer access on its own timeline. Equipment and tools are shared, dated, or unavailable when the team needs them. The relationship is adversarial: the team views the host as the main impediment to speed, and the host views the team as a source of missed commitments.

Best organization

The host gives the team true ownership of the product, as if it were a funded startup. It provides timely market, financial, supplier, and customer information. The team has enough budget for the duration of the project. The right skills arrive at the right time. Tools, test equipment, and facilities are sized to the need. The team has delegated authority to decide and act on everything required to stay ahead of schedule.

The interrupt and provisioning matrix

lateralworks teams use a pair of simple matrices to make the provisioning and interrupting dynamic explicit. Each week during the core team refresh-planning meeting, the team logs its top three active **interrupts** — the specific things slowing it down right now — and its top three **provisioning needs** — the specific things it needs from the host in the coming week. Every line item has a seven-day life. If an interrupt or provisioning need is still open after a week, it becomes a managed improvement project outside the refresh cadence.

The matrix is reviewed weekly by the executive review board (ERB). Each team gets ten minutes to walk the board through its matrix, make its case, and leave with either a resolution or a commitment to one within seven days. The ERB is a communication forum, not a problem-solving forum; teams are coached to resolve team-level interrupts inside the team and bring to the ERB only the items that require host action. The matrix makes the abstract idea of provisioning concrete, auditable, and time-boxed.

Interrupt matrix

Top active interrupts	Source	Team / host	Syst. / sit.	Negative impact	Preventive action	Responsible	Target
PR for test equipment in approval queue 6 weeks	Finance / procurement	Host	Systemic	3-wk critical-path slip on qualification test	Raise sub-$25K signature authority to PM	CFO + PM	Within 7 days
Two HW engineers pulled to sustaining work without notice	HW functional mgr	Host	Situational	Board bring-up delayed; schedule slipping 1 day/day	Return engineers or formally cancel competing project	VP engineering	Within 3 days
Customer feedback filtered through sales; beta contradicts specs	Sales org	Host	Systemic	Requirements churn; rebuild of two subsystems likely	Establish direct engineer-to-customer channel	VP mktg + PM	Within 7 days
Shared signal-integrity bench booked; no slots for 3 weeks	Lab scheduling	Host	Situational	2-week slip on eye-diagram characterization	Reprioritize bench; add overnight shift or 2nd bench	Lab manager	Within 5 days
Product spec open on interface choice; engineering waiting	Product mgmt	Host	Systemic	Design work stalled; growing rework exposure	Force decision at next ERB; document default	Product GM	At next ERB

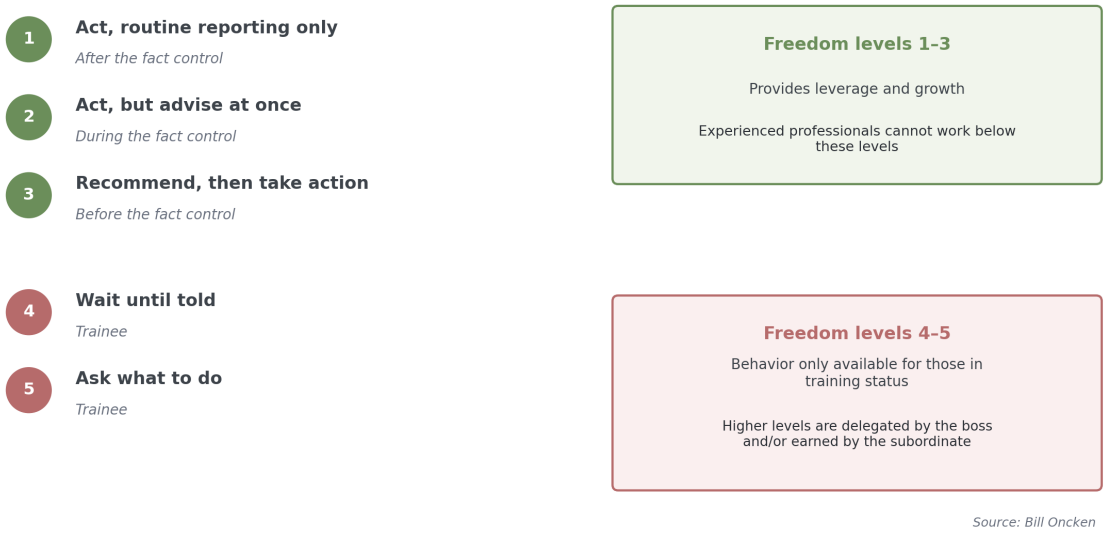
Provisioning matrix

Top provisioning needs	Schedule impact	Responsible provisioner	Provisioning actions	Target
Dedicated ASIC verification engineer for 12-week tapeout sprint	High — on critical path	VP engineering	Pull engineer from lower-priority program; backfill sustaining	Within 10 days
Direct lead-customer access for feature-prioritization workshop	High — drives scope	VP marketing	Introduce PM and lead engineer to customer CTO; bypass account team	Within 7 days
Second signal-integrity bench for parallel characterization	High — 2-wk slip avoided	Lab mgr + CFO	Approve capital purchase; stand up bench in existing lab	Within 3 weeks
Supplier roadmap and committed delivery for critical component	Medium — affects build plan	VP operations	Executive-to-executive call with supplier; secure allocation	Within 7 days
Delegated spending authority raised to $25K for core team	Medium — removes friction	CFO	Update spending policy; publish empowerment matrix change	Within 14 days

Figure 6. The interrupt and provisioning matrices with representative entries. Teams fill in the top three active items each week and commit to a seven-day resolution cycle. The rows shown here are anonymized examples from recent lateralworks engagements; each team populates its own matrix with its own current interrupts and provisioning needs.

Empowerment and the freedom scale

Provisioning is only half the story. The team also needs a clear, explicit envelope of authority — what it can decide and act on without asking permission, and where it must still recommend or wait. lateralworks uses Bill Oncken's *freedom scale* to make this envelope concrete. The scale has five levels that describe the delegation relationship between a team and its host.



The Oncken freedom scale. Levels 1–3 are the range in which experienced professionals can add durable value. Levels 4–5 are appropriate only for trainees.

Levels 1–3 provide leverage and growth. Experienced professionals cannot work below these levels of freedom — if they are forced to, they stop adding value, and the organization pays the senior-level salary for junior-level output. Levels 4 and 5 are appropriate only for trainees. When a fast-moving heavyweight team is stuck at level 4 or 5 on any of its important decisions, that team is, by definition, not heavyweight.

The freedom scale is not applied as a blanket label. It is applied decision category by decision category, in an **empowerment matrix** that the host and the team negotiate and publish together. A heavyweight team might sit at level 1 for day-to-day design decisions, level 2 for minor architectural changes, and level 3 for product-spec changes after the initial specification has been frozen. The matrix makes this envelope visible to everyone.

> **How to deploy the empowerment matrix.** (1) Assign one ERB member to own the matrix for each heavyweight team. (2) Tailor the decision categories to the team's context — add or change categories as needed. (3) Where a single "X" in a cell is ambiguous, add a concrete example of a decision that falls into that category at that level of freedom. (4) Ask the team to self-assess its current level of freedom, per category. (5) Meet with the team to reconcile host expectations against the team's understanding, and publish the reconciled matrix. (6) Use the matrix situationally, as a coaching tool, to train the behavior the host wants to see.

The full empowerment matrix that lateralworks uses on new engagements is reproduced in **Appendix B**. It covers seven decision categories — staffing, resource allocation, performance appraisal, product definition, customer requirements, capital and expense spending, and design / technology — and specifies a default

freedom level for each. Teams and their hosts adapt the defaults to their specific context during the deployment conversation.

> **Connection to the broader literature.** The interrupt matrix has a direct analog in the *andon cord* of the Toyota Production System [1]: a standing mechanism for individual contributors to surface problems immediately, with a standing commitment from management to respond. Spear and Bowen's description of Toyota's "DNA" [13] emphasizes that the andon cord is not primarily about stopping the line — it is about forcing management to reveal problems it would otherwise prefer not to see. The empowerment matrix plays a similar role for decision rights: it forces an implicit, often unhealthy, delegation relationship to become an explicit, negotiable one.

8. Make decisions fast, default to action

Speed of decision-making is not the same as quality of decision-making. A fast decision made with 80 percent of the information is almost always worth more than a slow decision made with 95 percent, because the 20 percent gap closes quickly once the team starts executing and the extra weeks spent gathering information are never recoverable. Fast organizations know this in their bones. They default to action.

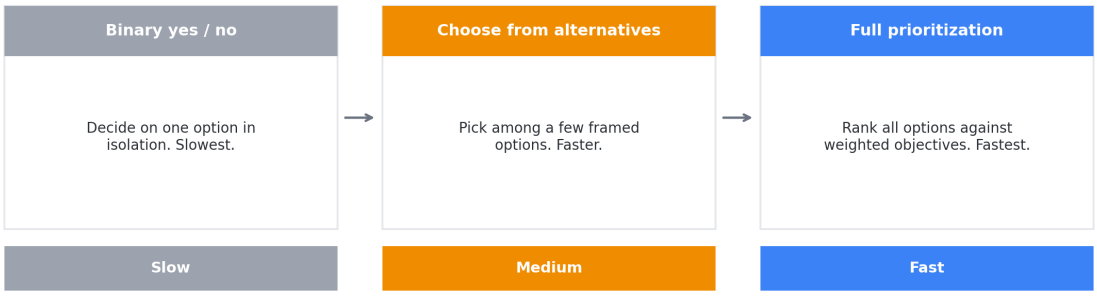


Figure 11. Three modes of decision-making, from slowest to fastest. Binary yes/no decisions are slow because they pretend each option exists in isolation. "Choose among alternatives" is faster. Full prioritization — ranking every option against weighted business objectives — is the fastest because it forces the trade-offs to be explicit and resolved at once, rather than being re-litigated on every decision.

Fast organizations run a fast-decision forum with a standing commitment to respond to team escalations within 24 hours. They publish the company's business direction and technology policy in a short, clear document that teams can consult to make their own tactical decisions without escalating. They use "strategic bypasses" — where a team needs guidance and the normal forum is not available, the product champion defaults to action after consulting the key decision-makers informally. The rule is: if you're waiting more than a day for a decision, either the decision doesn't really need to be made or the organization is broken.

Normal organization

Critical decisions require approval from upper management and sit in queues for weeks. Teams lack an overall understanding of business direction and technology policy, so they cannot make defensible tactical decisions on their own. They default to either waiting or generating more data. "Default to delay" is the organizational norm.

Best organization

Decision-making is fast because business direction and technology policy are clear, documented, and widely understood. Teams can act inside that envelope without escalation. When escalation is needed, a fast-decision forum responds within 24 hours. The default is action; the exception is waiting, not the rule.

Connection to the broader literature. A 2019 McKinsey survey of more than 1,200 managers and executives [19] found that only about 20 percent rated their organizations as good at decision-making — and the lost productivity from poor decision processes was estimated at more than 500,000 days of managers' time per year in a typical Fortune 500 company. The framework that best distinguishes fast from slow organizations is Bezos's *Type 1 vs Type 2* decision taxonomy [20]: Type 1 decisions are irreversible and deserve deliberation; Type 2 decisions are reversible and should be made quickly, because the cost of a wrong Type 2 decision is low and the cost of waiting is high. Fast organizations recognize that most development decisions are Type 2 and treat them accordingly. Slow organizations treat every decision as Type 1.

The ceiling

The host is the ceiling on what any team can achieve.

04

Portfolio

Pointing resources at the right work

A company with the right mindset and the right environment can still fail at speed if its portfolio is broken. Too many active projects for the available capacity; too few being killed when they stop making sense; starts that drift because no one owns the front end; teams tracked only to the handoff rather than all the way to the money. The portfolio practices close the loop: they make sure the resources that the environment frees up are pointed at the projects that matter most.

9. Manage the fuzzy front end

The period between when a product idea is first discussed and when a full team is actually staffed is called the fuzzy front end. In normal organizations it is invisible and unmanaged. Ideas drift for months or years. Marketing champions some; engineering quietly prototypes others. Neither group has the resources to do serious feasibility work, so the real technical investigation is deferred to the eventual development team, which inherits product goals that were set without enough information and are treated as unchangeable.

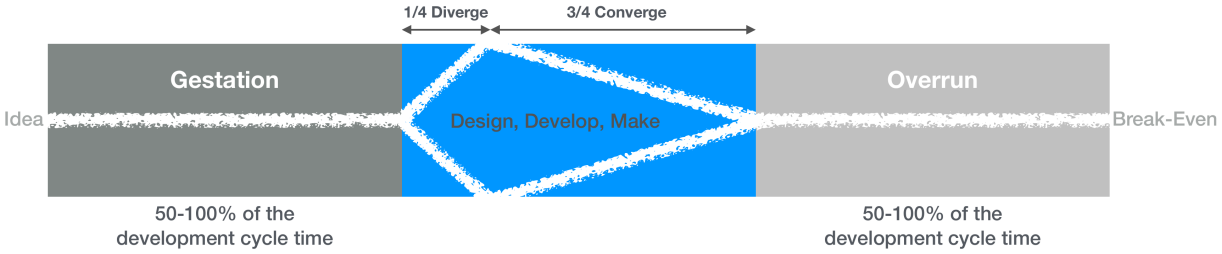


Figure 12. A typical product cycle has three phases. Gestation (idea to full team) and Overrun (target date to break-even) together often consume as much or more calendar time as the tracked design-develop-make phase. In most organizations, only the middle is measured.

Fast organizations treat the fuzzy front end as a real phase, not a prelude. A small multidisciplinary team — typically two to four people from engineering and marketing — is assigned to study feasibility within a fixed time budget, often under two months. The output is a go/no-go decision with clear economics: the business case, customer requirements, market positioning, and macro schedule. Because the time budget is tight, the work is focused. Because it is real, the goals it produces are grounded in actual investigation rather than wishful thinking.

Normal organization

The front end is unowned and under-staffed. Ideas meander for long periods. Marketing and engineering rarely work on feasibility together. The real discovery happens later, on the development team, which was formed late and is under-resourced. Goals set in the front end are inherited by the development team and never revisited, including unrealistic expectations for performance and schedule.

Best organization

The front end is a formal phase with a time budget, a small dedicated feasibility team, and a clear go/no-go gate. It is managed with the same discipline as the rest of the development process. Because it is fast and focused, it shrinks total time-to-market substantially — gestation is often the largest single opportunity for cycle-time reduction.

Connection to the broader literature. Khurana and Rosenthal's MIT Sloan research [21] studied the front end in 12 companies and found that successful front-end processes had three common features: an explicit product concept document, an identified product champion with executive sponsorship, and a formal go/no-go review before full development commitment. Companies that treated the front end as informal or implicit saw roughly twice the rate of late-stage requirements change. Koen and the PDMA's follow-on *New Concept Development Model* [22] documented the same pattern across 23 companies. Both bodies of work support the lateralworks finding that the front end is usually the single largest cycle-time opportunity in the development pipeline.

10. Prioritize projects against business objectives

Without an explicit prioritization system, project selection defaults to whoever has the most political influence. With an explicit system, project selection becomes a data-driven consensus. Fast organizations use formal methods — most commonly an Analytic Hierarchy Process (AHP) model — to score every candidate project against a set of weighted business objectives. The result is a ranked list, not a yes/no verdict. When something new enters the list, something else has to move down. Zero-sum ranking is the whole point.

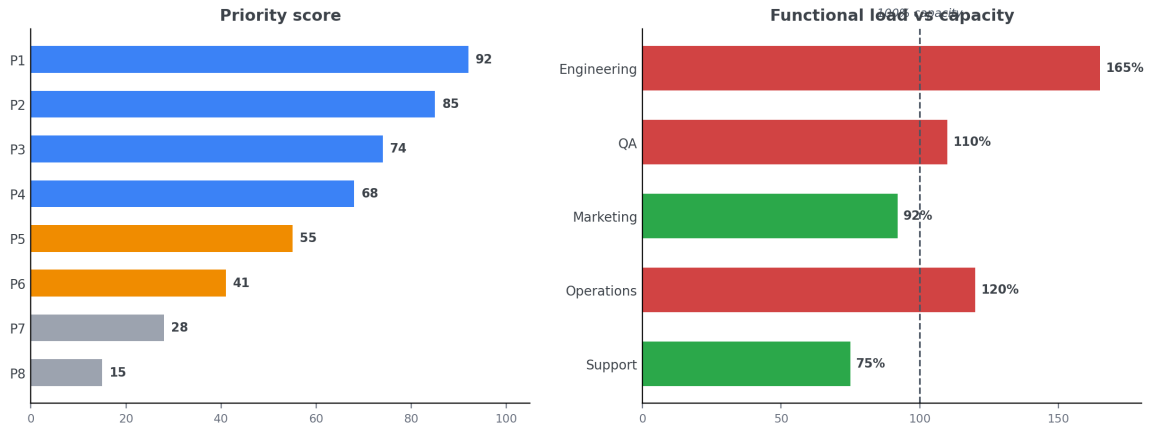


Figure 13. The two halves of portfolio discipline. On the left, every project is ranked against weighted business objectives to produce a prioritized list. On the right, the list is then checked against real functional capacity — the project that is second on the priority list may have to wait if the engineering group is already at 165 percent capacity on the first one.

Normal organization

There is no formal prioritization system. Marketing drives project selection from revenue plans; engineering filters based on available capacity. The two planning systems never reconcile. The business unit with the most political influence wins the most projects. The pipeline expands without regard for capacity. Projects rarely get killed, even when they should.

Best organization

A formal AHP or equivalent model scores every project against weighted business objectives. Marketing and engineering are forced to reconcile what is wanted with what can be delivered. The portfolio is refreshed monthly or quarterly. New ideas are ranked against the current portfolio; a new start must displace an existing project of lower priority.

Connection to the broader literature. The Analytic Hierarchy Process was developed by Thomas Saaty in the 1970s [14] and is now a standard technique in decision science. The Project Management Institute has published detailed guidance on using AHP for portfolio prioritization [15]. The method is widely used in defense procurement, infrastructure investment, and technology development precisely because it forces explicit trade-offs where political negotiation would otherwise dominate.

11. Align projects continuously with real capacity

Priority is half of the portfolio discipline. The other half is capacity. A prioritized list of projects that collectively require 150 percent of the available engineering hours is not a portfolio — it is a queue of future disappointments. Fast organizations model the resource constraint explicitly, and they refresh the model regularly. When a new project is proposed, the model shows which existing projects would slow down if the new one were added. The conversation about whether to start the new project becomes a conversation about whether it is worth the slowdown on the existing ones.

Slow organizations have no such model. When people point out that the pipeline exceeds capacity, they are told to be more productive. When projects slip, the explanation is always team execution, never resource overcommit. Executives who try to surface the capacity problem are career-limited; the safer path is to nod along with the "can-do" narrative until one is promoted out of the role.

Normal organization

No organized system for balancing resources. The tools that exist are ad hoc, consider only one functional view, and are updated infrequently. The pipeline expands because there is no shared picture of what the expansion costs. Executives who raise the issue are penalized. The "can-do" culture suppresses honest signals about capacity.

Best organization

A shared model tracks functional capacity against the active portfolio. New projects cannot be added without showing the impact on existing ones. The model is refreshed monthly; all stakeholders participate. When reality forces trade-offs, they are made explicitly, with data, rather than implicitly, by letting projects slip.

Connection to the broader literature. The cognitive research on task-switching — most notably Rubinstein, Meyer, and Evans [23] — consistently shows that shifting between tasks carries a 20–40 percent efficiency penalty per switch, and the penalty compounds with complexity. An engineer on eight concurrent projects is losing several hours per day to context switching alone, independent of any other inefficiency. DeMarco's *Slack* [24] makes the organizational case: a system running at 100 percent utilization has zero capacity to respond to anything unexpected. Little's Law from queuing theory [25] predicts mathematically that increasing the number of concurrent items in a fixed-throughput system proportionally increases cycle time. The multiplexing data in the case-study appendix is a field confirmation of this prediction.

12. Kill projects that miss objectives

Projects started with good intentions do not always remain good projects. Markets shift, competitors move, technical feasibility changes, business objectives evolve. A project that was in the top quartile of the portfolio last quarter may be in the bottom quartile this quarter. In normal organizations, this rarely matters; projects, once started, tend to keep running. In the fastest organizations, the prioritization model is re-run every quarter, and projects that have dropped below a threshold are challenged. Some are rescoped. Some are deferred. Some are killed outright. The resources they release are redirected to higher-priority work.

Normal organization

Projects rarely get killed. When they are stopped, they often reappear under a new name. Business objectives are vague enough that no project can clearly fail against them. Projects pushed by senior management or politically powerful sponsors are especially hard to stop, even after they have missed their economic window.

Best organization

Projects are assessed at least quarterly against current business objectives. Projects that fall below a threshold or have missed their economic window are challenged. The decision-making process is data-based and agnostic to the sponsor's position in the hierarchy. Killing a project is treated as good portfolio hygiene, not as failure.

Connection to the broader literature. Cooper's research on stage-gate product development [16] found that the single most predictive behavior of high-performing product organizations was willingness to kill projects at early gates. Low-performing organizations tended to pass projects through gates on faith, then discover late in development that the projects should have been killed at gate one. The correlation between kill rate and portfolio productivity held across industries.

13. Track new products all the way to break-even

What an organization measures is what the organization optimizes. Most companies measure development projects by development cost and by time-to-first-shipment. Both metrics end at the engineering handoff to operations. The people who designed the product never see whether it made money,

yielded well, satisfied customers, or captured market share. They optimize for the metric they can see — the handoff — and make decisions accordingly.

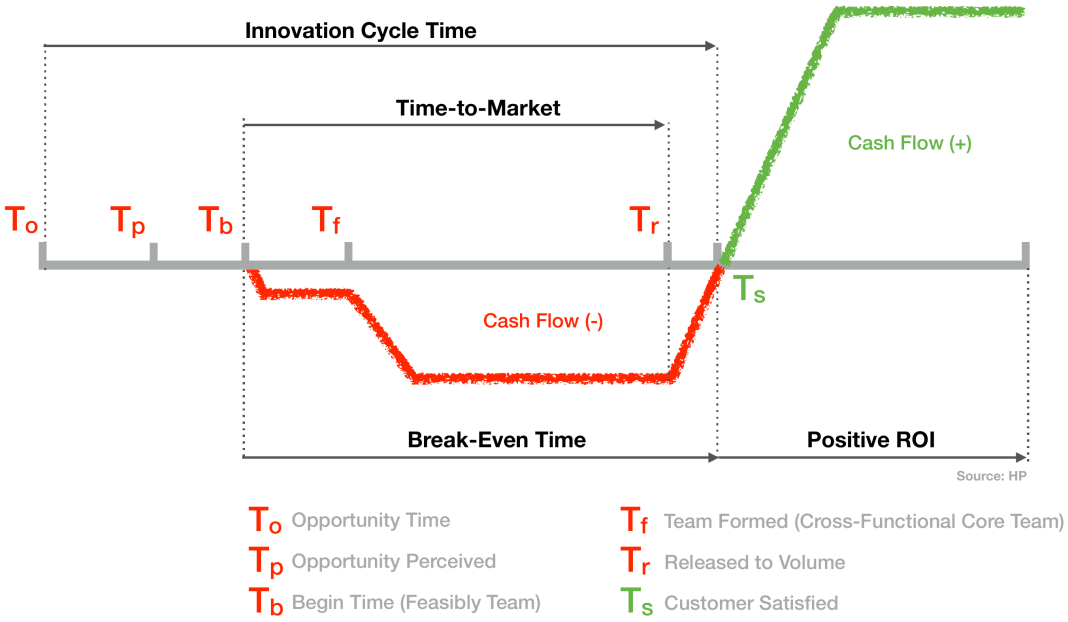


Figure 14. Break-even time (BET) tracks a product from the moment the opportunity is first perceived through to the moment cumulative cash flow crosses zero. Time-to-market is a subset of BET; it starts when the full team forms and ends at release. Teams that optimize for BET make different design choices from teams that optimize for time-to-market.

Fast organizations measure new products by break-even time — a discipline pioneered at Hewlett-Packard in the 1980s [5]. BET starts when the opportunity is first perceived and ends when cumulative cash flow crosses zero. It includes the gestation phase, the design phase, the ramp phase, and the early market phase. Teams that know they will be held accountable to BET make different design choices from teams that are held accountable to time-to-market. They think harder about manufacturability, about yield, about cost at volume, about serviceability — because all of these show up in cash flow after launch.

Normal organization

Projects are tracked by function. Marketing tracks what it starts. Engineering tracks design-to-release. Operations tracks the ramp. Somebody in finance eventually notices whether the product made money. The engineers who designed it never see the economic consequence of their design choices.

Best organization

Products are tracked end-to-end by the core team from idea through to break-even. The team sees the economic consequence of its own decisions. This visibility changes design choices upstream: yield, manufacturability, and service cost are considered early, when they are cheap to influence, rather than late, when they are expensive.

Connection to the broader literature. Kaplan and Norton's *Balanced Scorecard* [26] framed the general principle: what an organization measures drives what it optimizes. Goldratt's theory of constraints [27] makes the stronger case for throughput accounting — measuring the rate at which the system makes money rather than the cost at which each function operates. BET is a throughput-accounting metric applied to product development: it measures when the investment turns positive, not how much was spent getting there. The HP executives who pioneered BET [5] understood that a development team held accountable to BET would make different design choices than a team held accountable to development cost, and their division-level results in the 1990s validated the prediction.

05

Implementation
Putting it to work

The 13 practices are not a menu. They are a system. Attempting one of them in isolation — a new decision forum while the portfolio is still overcommitted, or a heavyweight team in an interrupting host — rarely produces more than a temporary improvement. The host behaviors that slowed the old system down will pull the new practice back into the old shape within a quarter or two. Real change requires moving on several fronts at once, in a deliberate sequence, with visible executive sponsorship throughout.

What follows is the six-step sequence that lateralworks uses on host transformation engagements. The sequencing is not rigid — companies with strong portfolio discipline but weak team structure would reorder the middle steps — but the sequence as written works for the majority of companies that come to lateralworks with a "we're too slow" diagnosis.

1. Start with mindset

Quantify the cost of delay on the top three current projects. Put the number in dollars per day in front of the executive staff. Make sure everyone understands, at a gut level, what a week of slip costs the company. Without this anchor, every other change is negotiable; with it, speed becomes the obvious default.

2. Install the interrupt and provisioning matrix

Start with one heavyweight team. Have the team log its top three interrupts and top three provisioning needs, weekly. Have the executive review board commit to a seven-day resolution cycle. Protect the mechanism from being diluted. Within 30 days the pattern of interrupts will be clear enough that systemic fixes become obvious; within 90 days the team's velocity will have measurably increased; within 180 days, other teams will be asking to be included.

3. Convert the most important teams to heavyweight

Pick the top two or three projects on the current prioritized list. Staff them as heavyweight teams — dedicated, colocated, with a senior program manager who has real authority. Protect these teams from the old functional pull. This will be unpopular in the functions; do it anyway.

4. Redesign the development framework

Assign one executive to own the end-to-end flow. Their job is to produce, within 60 days, a one-page framework that every project can use, and to begin retiring the accumulated review structures, phase gates, and sign-off chains that the framework is replacing. Expect resistance from whoever built the retired structures.

5. Install portfolio prioritization and capacity alignment

Stand up an AHP-based prioritization model. Run it against the full project portfolio. Match the ranked list against real functional capacity. Kill or defer whatever is below the capacity line. This step tends to surface resistance from whoever championed the projects that end up below the line; hold the line anyway. The discipline is meaningless if exceptions are granted based on executive volume.

6. Switch the scorecard to BET

Move the executive scorecard off development cost and time-to-market and onto break-even time. This is the longest-lead change because it requires agreement from finance, changes to performance management, and a culture shift away from functional measurement. It is also the change that most reliably sustains the other five: a scorecard built around BET rewards the right behaviors everywhere else in the system.

Expect resistance. Every organization that has grown up with slow habits has built a set of internal structures that benefit from those habits — risk-averse mid-level managers whose role exists to gatekeep, finance functions whose control mechanisms depend on slow decision chains, functional leaders whose power derives from owning resources that heavyweight teams need. These people are not malicious. They are doing the job the organization asked them to do. Changing the job requires negotiating with them, not

overruling them. The engagements that work are the ones where the CEO or general manager makes the change a personal priority and stays with it for 12 to 18 months — long enough for the new structures to outlast the political half-life of the old ones.

Key finding. Across lateralworks engagements, companies that complete all six steps of this sequence have consistently achieved 30–50 percent reductions in time-to-market on the projects run under the new system, within 12–18 months of the start of the engagement. Companies that attempt only one or two steps typically see a short-term improvement that fades within a year as the old structures reassert themselves. The system effect is real, and it is the system, not any individual practice, that produces the durable result.

Sources

References

[1] Ohno, T. *Toyota Production System: Beyond Large-Scale Production.* Productivity Press, 1988.

[2] Reinertsen, D. G. *The Principles of Product Development Flow: Second Generation Lean Product Development.* Celeritas Publishing, 2009.

[3] Christensen, C. M. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Harvard Business Review Press, 1997.

[4] Smith, P. G., and Reinertsen, D. G. *Developing Products in Half the Time: New Rules, New Tools.* 2nd ed. Wiley, 1998.

[5] House, C. H., and Price, R. L. "The Return Map: Tracking Product Teams." *Harvard Business Review,* January–February 1991.

[6] Jacka, J. M., and Keller, P. J. *Business Process Mapping: Improving Customer Satisfaction.* Wiley, 2009.

[7] Rogers, P., and Blenko, M. "Who Has the D? How Clear Decision Roles Enhance Organizational Performance." *Harvard Business Review,* January 2006.

[8] Ries, E. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* Crown Business, 2011.

[9] Edmondson, A. C. *The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth.* Wiley, 2018.

[10] Womack, J. P., and Jones, D. T. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation.* 2nd ed. Free Press, 2003.

[11] Wheelwright, S. C., and Clark, K. B. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality.* Free Press, 1992.

[12] Morgan, J. M., and Liker, J. K. *The Toyota Product Development System: Integrating People, Process, and Technology.* Productivity Press, 2006.

[13] Spear, S., and Bowen, H. K. "Decoding the DNA of the Toyota Production System." *Harvard Business Review,* September–October 1999.

[14] Saaty, T. L. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation.* McGraw-Hill, 1980.

[15] Vargas, R. V. "Using the Analytic Hierarchy Process (AHP) to Select and Prioritize Projects in a Portfolio." *PMI Global Congress Proceedings,* 2010.

[16] Cooper, R. G. *Winning at New Products: Creating Value Through Innovation.* 5th ed. Basic Books, 2017.

[17] Hamel, G., and Zanini, M. *Humanocracy: Creating Organizations as Amazing as the People Inside Them.* Harvard Business Review Press, 2020.

[18] Graeber, D. *The Utopia of Rules: On Technology, Stupidity, and the Secret Joys of Bureaucracy.* Melville House, 2015.

[19] De Smet, A., Jost, G., and Weiss, L. "Three Keys to Faster, Better Decisions." *McKinsey Quarterly,* May 2019.

[20] Bezos, J. "1997 Letter to Shareholders" and subsequent annual letters. Amazon Investor Relations.

[21] Khurana, A., and Rosenthal, S. R. "Integrating the Fuzzy Front End of New Product Development." *MIT Sloan Management Review,* Winter 1997.

[22] Koen, P. et al. "Providing Clarity and a Common Language to the Fuzzy Front End." *Research-Technology Management,* March–April 2001.

[23] Rubinstein, J. S., Meyer, D. E., and Evans, J. E. "Executive Control of Cognitive Processes in Task Switching." *Journal of Experimental Psychology: Human Perception and Performance,* Vol. 27, No. 4, 2001, pp. 763–797.

[24] DeMarco, T. *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency.* Broadway Books, 2002.

[25] Little, J. D. C. "A Proof for the Queuing Formula: L = λW." *Operations Research,* Vol. 9, No. 3, 1961.

[26] Kaplan, R. S., and Norton, D. P. *The Balanced Scorecard: Translating Strategy into Action.* Harvard Business School Press, 1996.

[27] Goldratt, E. M. *The Goal: A Process of Ongoing Improvement.* North River Press, 1984.

[28] Argyris, C. "Teaching Smart People How to Learn." *Harvard Business Review,* May–June 1991.

[29] Detert, J. R., and Edmondson, A. C. "Implicit Voice Theories: Taken-for-Granted Rules of Self-Censorship at Work." *Academy of Management Journal,* Vol. 54, No. 3, 2011.

[30] Oncken, W., Jr., and Wass, D. L. "Management Time: Who Has the Monkey?" *Harvard Business Review,* November–December 1974 (reprinted 1999).

[31] Parker, G. M. *Cross-Functional Teams: Working with Allies, Enemies, and Other Strangers.* 2nd ed. Jossey-Bass, 2002.

[32] lateralworks internal methodology documentation. "The FTTM Methodology." 2024–2026.

[33] lateralworks internal assessment database. Host best practice maturity assessments across semiconductor, storage, networking, and industrial automation engagements, 1992–2025.

[34] lateralworks. *Team Best Practices: Right-Product, Right-Team, Right-Time — the team side of the FTTM framework.* Companion paper to this document, covering co-development with tier-one customers, requirements translation, cross-functional core team roles, macro-micro scheduling, and continuous schedule pull-in. lateralworks methodology series, 2026.

A

Appendix A

Case study: a large storage technology company

Context. This appendix reports anonymized results from a lateralworks engagement with a multi-billion-dollar storage technology company. The company had missed several major product launches, was losing share to a faster competitor, and asked lateralworks to assess why. The assessment covered the 13 host practices, used both structured interviews (50+ practitioners across engineering, marketing, and operations) and quantitative analysis of project-staffing data (65 engineers across 11 active projects). What follows is what the assessment found and what the company did about it.

A.1 Methodology — the 5-level maturity scale

Each of the 13 practices is scored on a 5-level maturity scale standard in lateralworks host assessments. The scale is designed to force honest conversation: level 1 is explicit absence, and level 5 requires evidence of continuous improvement, not just presence of the practice.

Level	Label	Meaning
1	**Does not occur**	The practice is absent. Behavior actively contradicts it.
2	**Occurs at random**	The practice appears occasionally, driven by individual initiative, not institutional design.
3	**Occurs consistently**	The practice is in place and reasonably reliable, but not systematically managed.
4	**Defined and managed**	The practice is documented, measured, and owned by someone accountable.
5	**Continually improved**	The practice is periodically reviewed, tuned, and strengthened as the organization learns.

A.2 Assessment results — perception vs reality

Two parallel assessments were run. The project teams scored themselves on each of the 13 practices, using the same 5-level scale. In parallel, the lateralworks consultants scored the organization based on structured interviews with more than 50 practitioners. The results are shown below. The most instructive finding is not any individual score — it is the systematic gap between the two assessments.

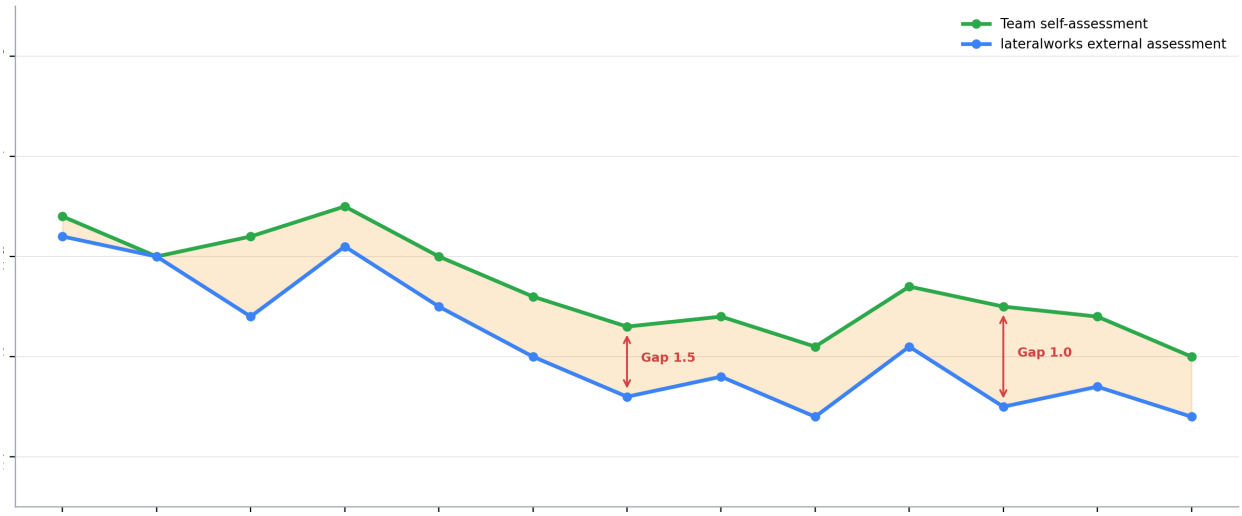


Figure A1. Maturity of the 13 host practices — team self-assessment (green) vs external lateralworks assessment (blue). The green line describes how the teams believed the company was operating; the blue line describes what the lateralworks interviews actually found. Red arrows mark the two largest perception gaps.

What the chart tells us

Four observations matter more than the individual scores.

First, the gap itself is the finding.

In a healthy organization, the external and internal assessments converge within a level. When the external assessment is systematically lower than the internal one, the organization has lost contact with what it is actually doing. That is a leading indicator of trouble — the kind of disconnect that shows up later as missed launches and lost market share. The remedy is not to argue about the scores; the remedy is to close the perception gap so that executives and teams are seeing the same system.

Second, the gap is selective.

Teams and lateralworks agreed, roughly, on financial-impact awareness, roadmap ownership, early-warning behavior, and the fast development framework. Where they disagreed was on the practices most directly tied to how projects are staffed and run — cross-functional core teams, capacity alignment, killing projects, and tracking to break-even. This is telling. Teams overestimated the maturity of the operational practices because those practices touch their daily work, and no one likes to describe their own environment as broken.

Third, the cluster of weaknesses is diagnostic.

The practices that scored lowest — provisioning teams, making decisions fast, managing the fuzzy front end, prioritizing projects, aligning to capacity, tracking to break-even — are exactly the practices that define a fast host. They are not random. Together they describe an organization whose mindset is oriented toward speed in the abstract but whose operational structures are oriented toward functional control, overcommitment, and lagging measurement.

Fourth, the near-zero scores are not effort problems.

Several practices score at or below "occurs at random" on the external assessment. These are not problems that can be fixed by working harder. No amount of team-level effort will make the fuzzy front end well-managed if the host has not staffed it. No amount of team discipline will align projects to capacity if the portfolio itself is overcommitted. These are structural problems that require executive-level structural intervention.

What the chart rules out. The chart rules out "the team is the problem" as an explanation for the missed launches. On every practice where the team has primary agency (mindset, early warning, development framework) the scores are reasonable. On every practice where the host has primary agency (provisioning, portfolio, BET tracking) the scores are low. A team problem produces the opposite pattern.

Connection to the broader literature. The systematic gap between self-assessment and external assessment is a well-documented phenomenon. Argyris [28] called it "Model I behavior" — the organizational tendency to protect self-image at the cost of accurate self-understanding. Detert and Edmondson [29] documented the specific mechanism: mid-level managers learn early in their careers that delivering bad news to executives is career-limiting, so the news that reaches the top is systematically better than the news at the bottom. In this company, the pattern was not pathological — it was ordinary and predictable. The value of an external assessment is that it bypasses the filter.

A.3 Themes from the interviews

The following themes emerged consistently across the more than 50 structured interviews with practitioners. Comments are anonymized and, where reproduced, paraphrased to remove identifying detail. Each theme maps to one or more of the 13 practices.

On roadmap churn (practice 2)

"The roadmap changes every time the VP of product attends a customer dinner." This was said by several engineers, independently, in different divisions. The pattern the interviews described was whiplash re-prioritization driven by the most recent customer complaint, without a structural process for deciding which complaints actually justified a roadmap change.

On slow structures (practice 3)

"I spend more time preparing review materials than doing the work that will be reviewed." The approval and review structures had accumulated over a decade without anyone being responsible for pruning them. Engineers estimated that 20–30 percent of their time went to reporting rather than producing. The reporting did not catch problems earlier; it merely made problems visible later.

On cross-functional teams (practice 6)

"I'm on eight projects. I have no idea which one is my priority." Several engineers reported being formally assigned to eight or more concurrent projects, with no single team leader empowered to set their priority. The functional manager nominally set priority but in practice deferred to whichever project had the most recent executive attention.

On provisioning (practice 7)

"The characterization bench is the critical path for my project and it's booked by three other teams." Shared test equipment, shared software-development tools, and shared customer-access channels were all being rationed across too many concurrent projects. The cost of an additional bench or license was a small fraction of the cost of the slips caused by sharing.

On capacity alignment (practice 11)

"Every quarter we start two more projects than we finish. It's been this way for five years." Capacity was never explicitly modeled. The pipeline expanded continuously because there was no mechanism to force the trade-off between a new start and the existing work. The multiplexing data in the next section quantifies what that expansion had produced.

On killing projects (practice 12)

"Projects here don't die. They get renamed." Several interviewees described projects that had been stopped and then restarted under a new code name, with the same sponsor and the same team. The renaming preserved political face at the cost of portfolio discipline.

On tracking to break-even (practice 13)

"I designed three products in a row. I have no idea whether any of them made money." The measurement system ended at the handoff from engineering to operations. Engineers who wanted to know whether their products had been commercially successful had to ask a friend in finance as a personal favor; the data was not routinely shared.

A.4 Quantitative finding: multiplexing

In parallel with the interviews, lateralworks collected the project-staffing data for 65 engineers across the 11 active projects in the division. Each engineer's count of concurrent active projects was calculated from the

formal staffing records. The distribution is shown below.

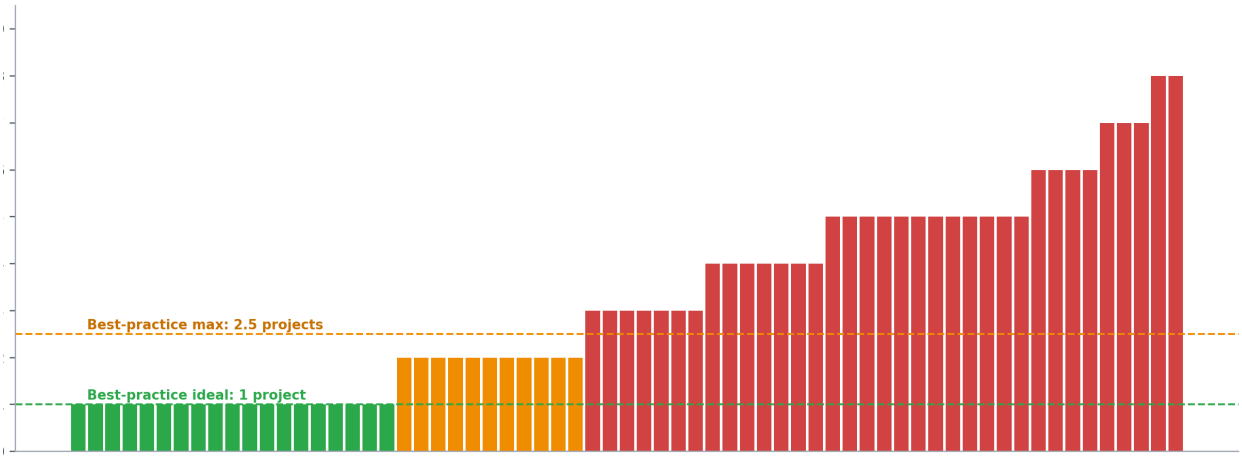


Figure A2. Concurrent project load per engineer (n = 65, anonymized and ranked by load). Green dashed line: best-practice ideal of 1 project per engineer. Amber dashed line: best-practice maximum of 2.5 projects per engineer. Most of the population is above the maximum; a significant tail is well above it.

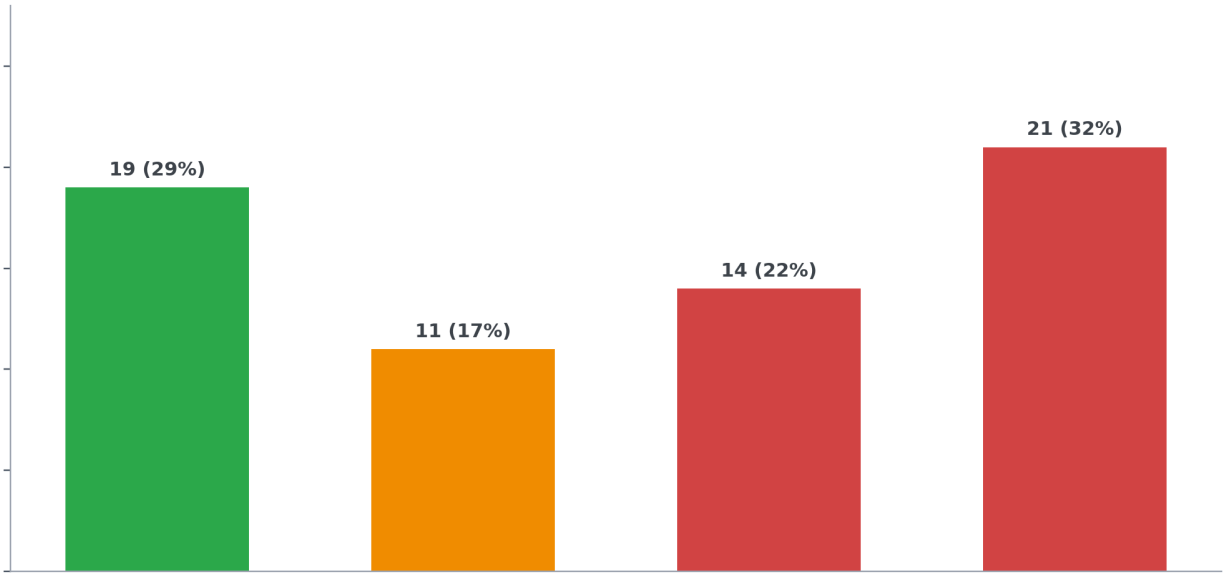


Figure A3. The same data as a bucketed distribution. Only 19 of 65 engineers (29 percent) are working on a single project. 32 percent are at or above five concurrent projects — a load at which task-switching alone accounts for 30–50 percent efficiency loss before any other factor is considered.

Multiplexing at this level has three compounding effects. First, **direct task-switching cost**: each time an engineer shifts projects, roughly 20–40 percent of the next productive hour is spent getting re-oriented to the problem. An engineer on eight projects may spend 2–3 hours per day context-switching, independent of any other inefficiency. Second, **decision latency**: when an engineer is needed for a decision on project A, they are often mid-task on project B, and the decision waits for the context-switch. Cumulative decision latency on a heavily multiplexed team can add 30–40 percent to schedule. Third, **commitment softening**: engineers on many projects cannot make reliable commitments on any one of them, because they cannot predict which project will demand attention next. Commitments drift, plans become fiction, and the team loses the ability to forecast its own work.

A.5 Analysis — what the data added up to

Three findings emerged from reconciling the qualitative interview themes with the quantitative multiplexing data and the assessment chart.

Finding 1: The host is the ceiling — and the host cannot see itself

The perception gap between the team self-assessment and the external assessment (Figure A1) is not a scoring disagreement. It is evidence that the organization has lost accurate sight of how it operates. The executive staff believed the company was running a heavyweight-team model, prioritizing against business objectives, and killing underperforming projects. The data said otherwise. Until the executive picture aligned with the operational reality, no amount of team-level improvement would produce a sustained change in outcomes.

Finding 2: Provisioning is a capacity problem first

The interview themes and the multiplexing data point to the same root cause: the company had chronically overcommitted itself across too many projects for too long. The pipeline expansion reported by engineers ("two more starts than finishes every quarter, for five years") is exactly what produces the Figure A2 distribution. Every provisioning problem the teams identified — shared test equipment, shared engineers, filtered customer access — is downstream of the capacity overcommitment. Fixing any one of them without fixing the portfolio would produce a temporary improvement that reversed when the pipeline re-expanded.

Finding 3: The core teams are heavyweight in name only

Formally, the company had "core teams." The staff structure listed team leaders, team members, and charters. Operationally, the core teams had no authority the functions did not concur with; team members reported to functional managers, had their performance reviewed by functional managers, and took direction from functional managers whenever there was a conflict with team priorities. In every case where the interviews described a core team breaking down, the breakdown came from a functional pull that the team leader had no authority to resist. The structure was lightweight dressed as heavyweight.

A.6 The improvement plan

The company and lateralworks jointly proposed a seven-point improvement plan. The plan was phased over 12 months and sequenced deliberately: mindset-level changes first, then structural changes, then measurement changes.

(1) Install the interrupt and provisioning matrix on the top three projects. Commit to a 7-day resolution cycle, owned by the executive review board. Publish weekly.

(2) Rebuild the top three core teams as true cross-functional heavyweight teams. Assign senior program managers with real authority. Move performance review of team members from the functional managers to the team leaders.

(3) Publish an empowerment matrix for each of the three teams, following the Appendix B template. Reconcile host expectations with team understanding and publish the agreed matrix.

(4) Enforce a ceiling of 2.5 concurrent projects per engineer across the division. Kill, defer, or deprioritize whatever work does not fit under the ceiling.

(5) Install an AHP-based portfolio prioritization model. Run it against the current project list. Match the prioritized list against real functional capacity. Share the model with all executives and publish monthly refreshes.

(6) Stand up a 24-hour fast-decision forum for cross-functional decisions that would otherwise escalate. Publish the envelope of decisions the forum is empowered to make.

(7) Add break-even time (BET) to the executive scorecard. Begin tracking BET for every new project alongside the existing development-cost and time-to-market metrics, and phase out the development-cost metric over 18 months.

A.7 Early results

Twelve months into the engagement, the company reported the following results on the three pilot projects that had been rebuilt as true heavyweight teams inside a provisioning host:

- **Time-to-first-prototype** on the pilot projects reduced by 35–45 percent compared to the baseline of the previous three products the division had shipped.
- **Schedule predictability,** measured as the gap between committed and actual milestone dates, improved by roughly 3x. Milestone slippage fell from an average of 4–6 weeks per milestone to 1–2 weeks.
- **Engineer multiplexing** on the pilot projects fell from an average of 5.2 concurrent projects per engineer to 1.8. Division-wide multiplexing fell more slowly (to 3.1) because the portfolio-kill-or-defer step (#4) took longer than expected.
- **Interrupt resolution** reached steady-state at about 85 percent of logged interrupts resolved within the 7-day window, up from an effectively unmeasured baseline.

The BET metric (#7) had not yet produced results; the first product launched under the new system was still in its early ramp at the 12-month report point. The company committed to continuing the engagement for another 12 months to see the BET signal and to extend the model across the remaining divisions.

> **Lesson.** The pattern at this storage company is the pattern lateralworks sees again and again: a team that appears to be the problem turns out to be an ordinary team performing to the ceiling of an interrupting host, and the executives who commissioned the assessment turn out to be the primary people with the authority to raise that ceiling. The teams could not fix this. The assessment did not make the executives more talented. It made the structural problem visible enough that it could be addressed at the level where the leverage actually existed.

B

Appendix B
Empowerment matrix

The empowerment matrix is the document that makes the delegation relationship between a heavyweight team and its host explicit. Each row is a category of decision the team will face; each column is a level on the Oncken freedom scale introduced in practice 7. The cells show, for each decision, the level of freedom the team has by default. The matrix is not a rigid rulebook — it is a starting point for the conversation the team and the host need to have before the project begins.

The matrix below is the lateralworks default for heavyweight teams in technology companies. It is offered as a template. Companies with different risk profiles — heavily regulated industries, safety-critical products, early-stage startups — will adjust the defaults.

Decision category	Sub-decision	L1 Act, report	L2 Act, advise	L3 Recommend
Staffing	Retain contractors / consultants		✔	
	Relocate team members within the team			✔
Resource allocation	Assign members to tasks			✔
	Determine member availability			✔
	Re-assign members out of the team	✔		
Performance appraisal	Team lead reviews of members	✔		
	Member reviews of each other			✔
Product definition	Close initial specs	✔		
	Change specs post-freeze		✔	
Customer requirements	Obtain requirements from customer			✔
Capital / expense	Set project budget			✔
	Manage spending within budget	✔		
	Spending under $5K	✔		
	Spending over $5K			✔
Design / technology	Product architecture — minor		✔	
	Product architecture — major			✔
	Day-to-day design decisions	✔		
	Technology trade-offs	✔		

B.1 How to read the matrix

The three freedom levels map to the Oncken scale described in practice 7:

- **L1 — Act, report:** The team acts on its own judgment and reports the decision after the fact. Appropriate for decisions where speed is critical, reversibility is high, and the team has enough context to decide well. Most day-to-day design and spending-under-limit decisions live here.
- **L2 — Act, advise:** The team acts but notifies the host immediately. Used for decisions that are reversible but where the host needs the information to coordinate with other teams or external parties. Minor architecture changes and contractor retention typically live here.
- **L3 — Recommend:** The team frames the decision, presents options, and recommends. The host decides. Used for decisions that are hard to reverse, carry material financial impact, or commit the company beyond the team's scope. Budget setting, major architecture changes, and external customer requirements live here.

Levels 4 and 5 ("wait until told" and "ask what to do") do not appear in a heavyweight-team matrix. A team operating at those levels on any substantive decision is, by definition, not heavyweight; it is a coordination body reporting to the functional hierarchy, which is a different kind of organism and should be labeled accordingly.

B.2 Adaptation notes

Adaptation notes. The defaults in this matrix work for most heavyweight product development teams in unregulated or lightly regulated industries. Adapt them as follows. (1) **Regulated industries** (medical devices, aerospace, financial services): move product-definition and spec-change rows to L3 across the board. (2) **Early-stage startup inside a larger company:** move most rows to L1 or L2; the point of an intrapreneurial team is speed, and L3 defaults defeat the purpose. (3) **Platform teams serving multiple product teams:** move resource-allocation and availability rows to L3, because the trade-offs span multiple stakeholders. (4) **Safety-critical products:** move design-decision rows to L2 or L3 and require a named safety authority. The matrix is a starting point, not a universal answer — use it to force the conversation, then write down what the conversation produced.

A filled-in empowerment matrix should be published — on a team wiki, a shared drive, or printed and pinned to the team space — and referenced when decisions arise. The point is not the document itself; the point is to make the delegation relationship so explicit that both the team and the host know what the team is allowed to do, and neither has to guess.