Case study

Tilting the organization on its side

How a functional semiconductor company rebuilt four first-of-a-kind programs around heavyweight product delivery teams — and what it took to get its new leaders to lead

FTTM case study series.

A deep-dive case study from a lateralworks engagement with a global semiconductor company delivering power products for AI data centers. Drawn from engagement diagnostics, role charters, alignment sessions, and executive briefings produced during the work. The client's identity, product names, and personnel are withheld.

Prepared by
lateralworks
FTTM methodology

Date
July 2026
Case study

Online
lateralworks.com
FTTM case study series

Table of contents

Tilting the organization on its side

Core thesis. A functional organization becomes fast when it tilts on its side: when a dedicated team owns each product end to end and the functions provision that team instead of directing it. The org chart is the easy part. The hard part is that the new structure asks people to lead in ways their careers never taught them — and until leadership behavior catches up with structural authority, the tilt exists only on paper.

Overview

Abstract

This is a case study about the distance between an org chart and an organization. A global semiconductor company — call it the company — set out to win four first-of-a-kind power products for AI data centers, each for a hyperscale customer, each on a compressed schedule, and each carrying research-grade technical risk that its own roadmap had never retired. Its functional organization, built over decades and excellent at what functional organizations do, could not move laterally at the speed these programs demanded.

Working with lateralworks, the company redesigned the programs around heavyweight product delivery teams (PDTs): dedicated cross-functional core teams, each led by a product delivery leader who owns the product from concept through NPI (new product introduction), qualification, and volume ramp, paired with a program manager who owns the integrated schedule. Functional managers moved from directing daily work to provisioning talent and governing standards. Decision authority moved to the team, made explicit through freedom-scale tables for every role.

Then came the real problem. The newly named PDT leaders — capable, respected managers with careers formed entirely inside functional silos — did not lead. They asked what their role was, waited for definitions, deferred to their PMs in schedule reviews, and escalated decisions the structure had already given them. This paper examines why that happened, why it is the predictable cost of the tilt rather than a failure of the people, and what closed the gap: written role charters with explicit freedom levels, the CEO/COO pairing of leader and PM, honest schedules that plan learning cycles, tiger teams for the research-grade problems, and a general manager armed to demand reality instead of reassurance.

The case draws on the engagement's working documents — role and responsibility charters, alignment meeting notes, executive briefings, and schedule walk-through protocols — and connects them to three decades of published research on heavyweight teams, from Clark and Wheelwright's original taxonomy to Toyota's chief engineer system. It is written for any leadership team asking the same question this company asked: how do you get a highly functional organization to tilt on its side and become a lateral, end-to-end execution system — and what does that demand of people who have never managed in an empowered structure?

01

The setting

A vertical company in a lateral race

The company in this case is one of the world's established semiconductor makers: decades old, global, profitable, and organized the way most successful semiconductor companies are organized — by function. Design, test engineering, product engineering, quality, supply chain, packaging, and manufacturing each ran as a deep vertical discipline with its own management chain, its own standards, and its own career ladders. That structure had served the company well. Functional organizations concentrate expertise, enforce consistency, and scale operational excellence, and for derivative products on proven platforms they are hard to beat.

Then the market moved. AI data centers created sudden, enormous demand for a new class of power products, and two hyperscale customers put four first-of-a-kind programs on the company's table at once — with volume targets in the hundreds of millions of units and schedules set by the customers' build-outs, not by the company's comfort. The prize was generational. So was the risk.

Technology development and product development at the same time

In most companies, perhaps one project in ten carries genuine research risk. Here it was nearly all of them at once: embedded substrate packaging, backside patterning, new suppliers running processes that had never scaled. The packaging was the product, and the research had never been done ahead of it. The company was doing technology development and product development simultaneously, on every program, inside schedules and structures designed for products where the research was already finished. That mismatch — research risk absorbed through roles, schedules, and staffing built for derivative work — turned out to explain almost every problem the engagement would later surface.

The lightweight structure and what it could not do

The company ran its programs through what Clark and Wheelwright, in their classic study of development organizations, called the lightweight team structure [1]. Each program had a project manager, but the PM was a coordinator: no budget authority, no say in performance reviews, no power to commit the functions to anything. Team members sat with their functions, took direction from their functional managers, and were measured by them. The PM's job was to track, chase, remind, and escalate. Work moved through the organization as a sequence of handoffs between silos, and every cross-functional decision climbed a management chain, crossed at the top, and climbed back down.

The evidence that this structure is slow has been on the record for a long time. Larson and Gobeli's study of 540 development projects found that functional and functional-matrix structures consistently underperformed project-centered structures on schedule, cost, and technical success [3]. Takeuchi and Nonaka contrasted the relay race of sequential functional handoffs with the rugby approach — a dedicated cross-functional team moving down the field together — and found the rugby teams consistently faster [5]. Wheelwright and Clark's own field data led them to the same conclusion: for projects that matter, heavyweight teams

outperform lightweight coordination by margins that compound across a portfolio [1, 2].

None of this was news to the company's leadership. The swimlane model they ran — functional workstreams laid side by side in a planning grid — worked as a planning framework. It failed as an execution model for one structural reason: no single person owned a product end to end through NPI, qualification, and volume ramp. Every discovery interview in the diagnostic converged on that same gap. When everything is going to plan, a swimlane chart looks like ownership. The moment a first-of-a-kind problem appears — and on these programs they appeared weekly — the question *who owns this?* had no answer below the general manager himself.

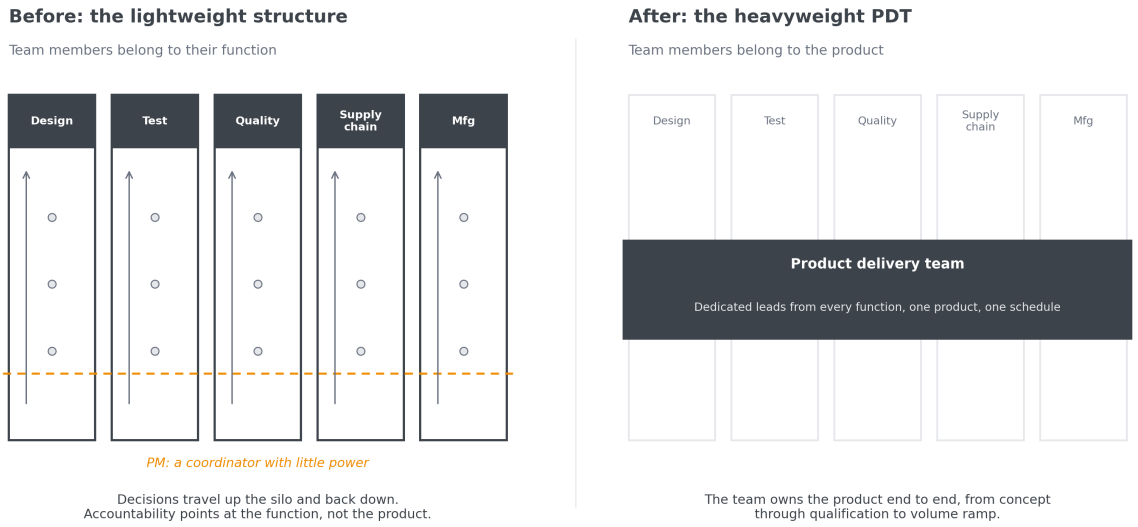

Figure 1. The tilt. In the lightweight structure, people belong to functions and decisions travel vertically. In the heavyweight PDT, a dedicated team owns the product end to end and decisions travel with the work.

The cost of staying vertical

By the time the engagement began, the symptoms were visible at every level. Milestones that had held for months were slipping together. The few experts who could solve the hardest problems — a die-crack mechanism, on-resistance shortfalls, SMT fallout at a new supplier — were split across every product line, attending every meeting, solving nothing to completion. Escalations stacked up at the top of the functions. And the general manager, who owned all four programs, was in effect the only integrator in the building: the one point where the lateral view of any product came together, on top of running the business. The structure was asking him to be the core team, and that does not scale to four concurrent first-of-a-kind programs. The company did not have a talent problem. It had a structure problem, and the structure problem was about to be named.

02

The redesign
The heavyweight product delivery team

The redesign replaced coordination with ownership. Each of the four products got a product delivery team: a dedicated, co-located, cross-functional core team with the authority to carry its product from concept through NPI, qualification, production release, and volume ramp. The functions did not disappear — they kept standards, technical governance, and career development — but execution authority moved out of the silos and into the teams. In Clark and Wheelwright's terms, the company moved from lightweight to heavyweight [1]. In plainer terms, it tilted the organization on its side: the primary axis of management stopped being the function and became the product.

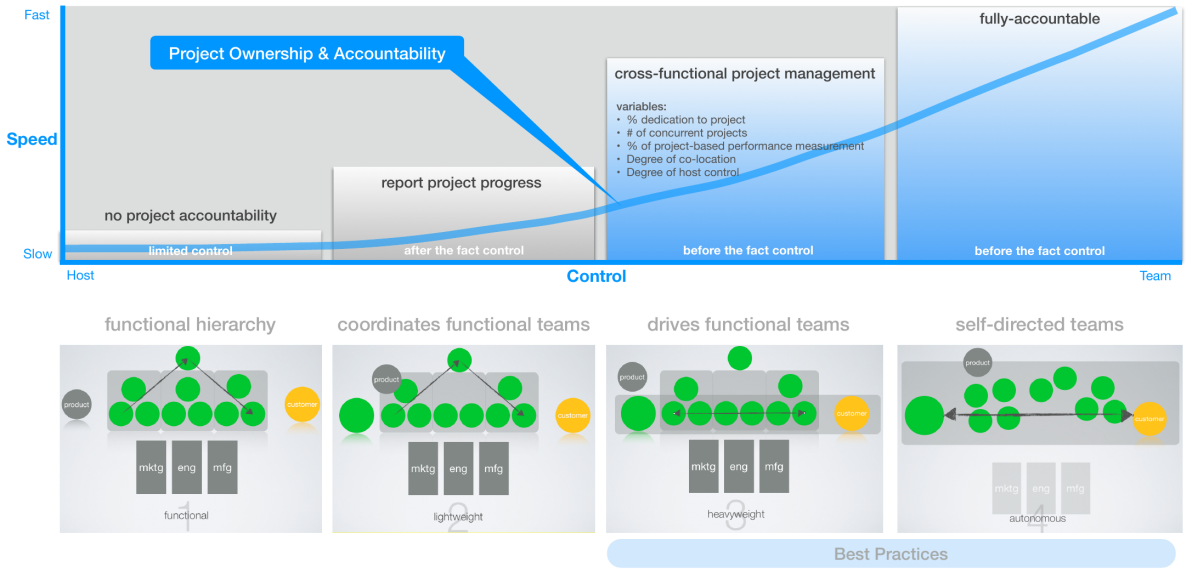

Figure 2. The four team structures, from functional hierarchy to self-directed autonomous teams [1]. Speed rises as project ownership and accountability shift from the host to the team, and control moves from after-the-fact to before-the-fact. The company's move: from lightweight (coordinates functional teams) to heavyweight (drives functional teams) — the best-practice zone.

Five layers, one product axis

The operating model defines five layers, each with a distinct relationship to the product. What follows is the model as the company chartered it, in a role and responsibility document written jointly with lateralworks and issued under the general manager's name.

The product delivery leader is the single point of accountability for an entire product — the CEO of the product. The role owns the business outcome, the customer relationship, the technical trade-offs, and the execution schedule: every milestone from working samples through qualification to production release, the

P&L including cost and yield economics, and the authority to make start/stop/continue recommendations for the product. One interviewee in the diagnostic called it a unicorn role: someone who makes quick decisions, commands respect, has skin in the game, and is close to the business, the customer, and the execution teams at once. The charter is deliberate about the heavyweight test: the leader directs all embedded leads day to day and conducts or shapes most of their performance reviews. This mirrors Toyota's chief engineer — the shusa — who owns the vehicle program end to end while the functions supply people and expertise [6, 7].

The PDT program manager is the execution engine — the COO to the leader's CEO. Where the leader owns the what and the why, the PM owns the how and the when: building and continuously driving the integrated schedule that connects every functional workstream into one critical-path plan. The charter is explicit that this is not a lightweight PMO coordinator or a status reporter. The PM refreshes the schedule weekly, recalculates the critical path after every refresh, models pull-in scenarios, tracks the ten to twenty secondary critical paths behind the main one, and knows at all times the gap between the target date and the real schedule — using that gap to generate urgency before the fact rather than damage control after it.

The embedded functional leads — eleven disciplines per team, from silicon validation and module design through test, packaging, supply chain, and product quality — are assigned 100% to one product. No multiplexing across programs. Each lead carries local authority to make product-specific decisions in their domain without asking their functional home, reports operationally to the product delivery leader, and keeps a dotted line to the function for standards and career development.

The functional managers keep three jobs and lose one. They keep talent: recruiting, developing, and provisioning the leads the teams consume, with hiring lead times planned against program milestones. They keep standards: the corporate quality specifications, procurement frameworks, and process standards every product must meet, with roughly a 20% veto on enterprise-level decisions. They keep knowledge transfer: communities of practice that move lessons between teams so dedication does not breed silos. They lose daily work direction. The charter states it without hedging: functional managers do not direct the daily priorities of embedded leads, do not override product-level technical trade-offs, do not impose functional timelines on the integrated plan, and do not conduct performance reviews unilaterally.

The program director sits above the four teams and owns what no single team can: the integrated master schedule across all four products, cross-team resource conflicts, program-level risk, and the operating cadence. The role coordinates rather than commands — product-level trade-offs stay with the product delivery leaders — and reports directly to the general manager, replacing the de facto arrangement in which the GM was the integrator of last resort for everything.

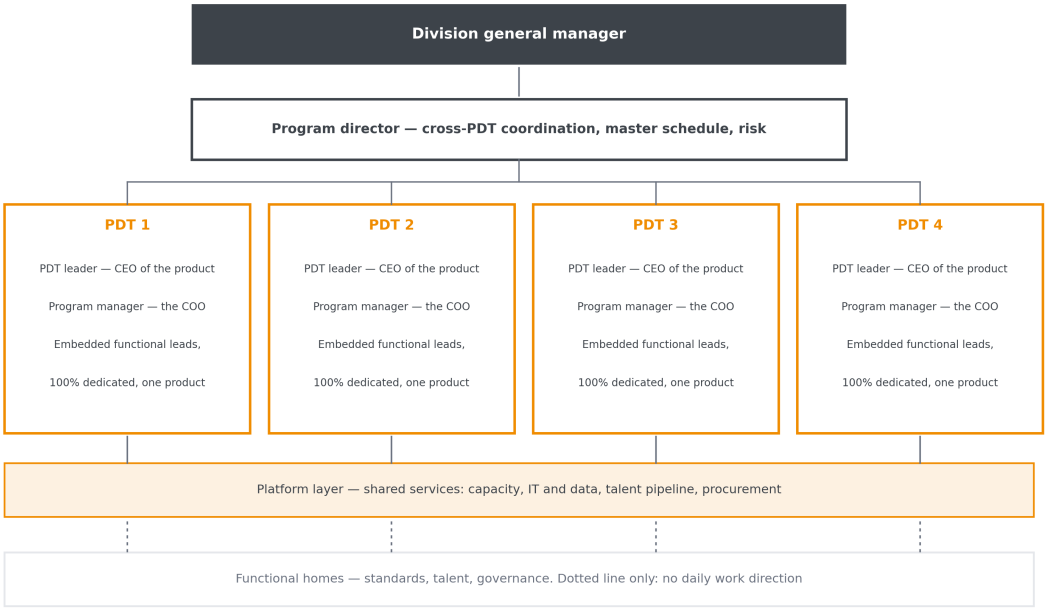

Figure 3. The five-layer operating model. Execution authority concentrates in the four PDTs; the platform layer and the functional homes provision them.

Making authority explicit: the freedom scale

Announcing that a team is empowered changes nothing by itself, because empowerment fails at the level of individual decisions. The charter therefore attaches a freedom-scale table to every role: for each decision area, an explicit level from 1 (act, routine reporting only) to 5 (wait until told). A product delivery leader runs at Level 1 on product technical trade-offs, customer commitments, schedule, and the direction of embedded leads. The PM runs at Level 1 on everything schedule-related. Embedded leads run at Level 1–2 on product-specific decisions in their domain. Functional managers run at Level 1 on enterprise standards — their core authority — and stay out of daily direction entirely. Appendix A reproduces the scale in full.

The scale does two things a conventional RACI chart does not. It quantifies trust — each level is a specific answer to “how much can this person do without asking?” — and it makes drift measurable. The charter’s diagnostic rule: if a role consistently operates two or more levels below its suggested freedom, that is a structural problem, not a people problem. That rule turned out to be the engagement’s most important early-warning instrument, because the first thing it detected was the leadership gap this paper is about.

The overlay roles: a triad at the core

Beyond the thirteen core team roles — the leader, the PM, and the eleven embedded leads — the charter defines three overlay roles that describe how the team integrates rather than what each member does. The system integrator looks laterally across the program and manages every interface, so that a decision made in one domain does not surprise another. The chief engineer makes the final technical call — when the silicon, driver, and module teams disagree, one person listens to the data and decides, and cannot be overruled on technical matters inside the product. The system architect protects the total system from component-level sub-optimization, allocating performance budgets and rejecting local improvements that degrade the whole. These are hats worn by existing core team members, not new hires — a module design lead who also serves as chief engineer still owns module design. Each prevents a specific failure mode: integration surprises, committee-diluted technical decisions, and locally perfect components in a product that misses its targets.

One caution from the literature belongs in any account of this model. Christensen and Overdorf warn against one-size-fits-all team structures: heavyweight teams are the right tool when a program must create new capabilities, and an expensive one where existing processes suffice [8]. The company's derivative products did not get PDTs. The four first-of-a-kind programs — where the existing processes were precisely what could not deliver — did.

03

The leadership gap

Nobody had ever been the boss

The structure went in. The behavior did not. Months after the model was announced, the newly named PDT leaders were still asking what their role was. In schedule reviews, the leader would turn to the PM for answers about their own product. Decisions the charter had placed squarely inside the team were still being walked up the old functional chains for blessing. The freedom-scale tables said Level 1; the behavior said Level 4 — ask what to do. By the charter's own diagnostic rule, a two-level gap between structural and behavioral freedom is a structural alarm. This one rang across all four teams at once.

It would be easy to read this as a failure of the individuals, and it would be wrong. These were capable, respected managers. What they had never done — what the company had never asked anyone below the general manager to do — was own a cross-functional outcome. Their careers had taught them to run a discipline deeply and to escalate anything that crossed a boundary. The new structure asked them to organize people they did not hire, spend budget they had never controlled, make trade-offs between disciplines they had never worked in, and answer for a result no function could deliver alone. Nobody had ever been the boss before, because the old structure had no such job.

What the gap looked like up close

The engagement notes record the pattern in specifics. Leaders reported missing skills on their teams without a plan to acquire them — describing the job rather than doing it, in the phrase that stuck. Leaders treated the role as a part-time assignment stacked on an existing functional job, because that is what every prior special assignment had been. Leaders waited for a written role definition before acting, while the problems in front of them went unorganized. And when schedules slipped, leaders reported the slip rather than mobilizing against it. The briefing to the general manager put the standard plainly: leaders given a problem start organizing people to solve it. They do not complain about missing skills; they find out what is missing and fix it.

The deeper mechanism is the one Oncken and Wass described five decades ago: every time a subordinate asks the boss what to do, the monkey jumps onto the boss's back, and the subordinate's initiative shrinks by exactly that much [9]. The freedom scale names the same dynamic from the other side — people disempower themselves by asking. The more a leader asks, the more they get told, and the closer they drift to waiting until told. A career of asking is not undone by an announcement.

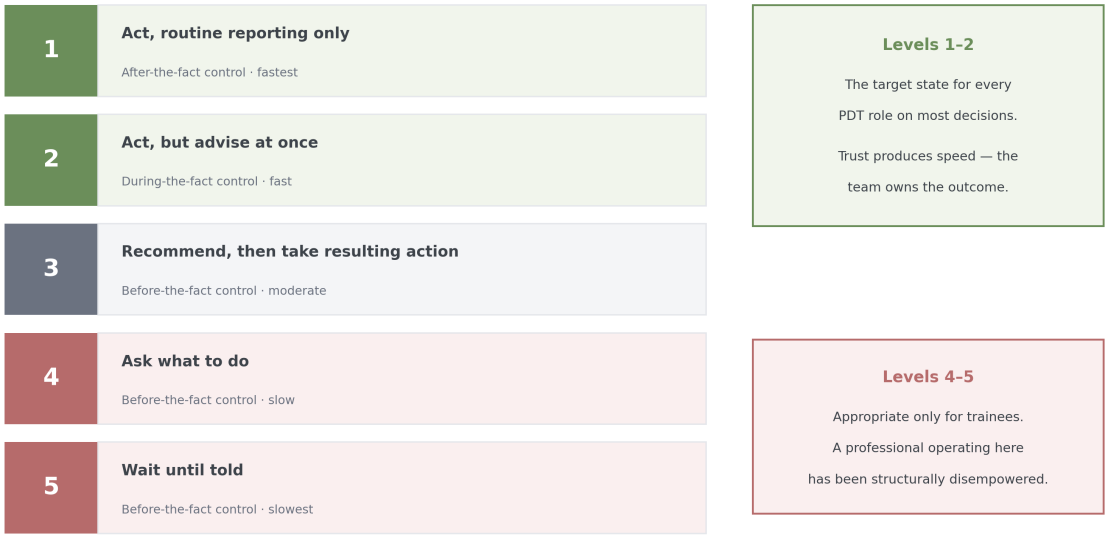

Figure 4. The freedom scale. The structure put the PDT roles at Levels 1–2; the inherited behavior sat at Level 4. Closing that gap was the engagement's central work.

There was also a harder edge to it, and the diagnostic did not look away from it: the company's history had punished people who stepped out. The people who stepped out into the wind, as one summary put it, got blown away. In an organization with that memory, hanging back was learned rationality. Edmondson's research on psychological safety reaches the same conclusion from the academic side: people take interpersonal risks, surface problems, and act without permission only where the climate has demonstrated it is safe to do so [10]. Empowerment on paper does not overwrite a decade of evidence about what happens to the empowered.

The host sets the ceiling

lateralworks' framework makes a distinction that organized the whole engagement: the team and the host. A delivery team never works in a vacuum. It sits inside a host — the functional groups, the portfolio managers, the executive staff — and the host decides what the team can actually do: what resources it gets, which decisions it owns, what information reaches it, and how fast a question gets answered. The host either provisions the team or interrupts it. Fast organizations are rarely the ones with the most talented engineers; they are the ones whose hosts reliably provision and rarely interrupt [14, 17].

Read through that lens, the leadership gap was only half a people problem. The other half was host behavior. PDT leadership was being run as a side job because hiring had not caught up with the operating model — dedication was treated as a preference rather than the design. The leader/PM split had never been written down, so the forward-looking half of the leader's job went undone by default. And the hardest technical problems cut across every product while the experts who could solve them stayed divided by product. McChrystal's account of the same transition in a very different institution lands on the same point: moving authority to teams fails unless the center changes what it does — from directing moves to provisioning shared consciousness and getting out of the way [11]. The ceiling on the PDT leaders was set above them.

Closing the gap

The fix was a package, not a speech, and it worked on both halves at once. On the role side: the leader/PM split was written down and aligned in a working session with every leader and PM in the room. The split is by horizon, not by task. The PM is the COO — head down, near term, owning the detail. The leader is the CEO — looking ahead, clearing obstacles, provisioning the team. The model sentence the session settled on: *tell me what you need to pull in three weeks.* Leader and PM run in constant contact, not one meeting a week, and the leader reads the schedule in fastProject as an analysis tool — a second set of eyes, not a schedule-building duty.

On the host side: all four leaders were dotted-lined to a single overall owner, ending the ambiguity about where the product axis converged. Hiring proceeded with dedication as a requirement, and new leaders were protected from second hats. The freedom-scale tables became onboarding instruments — new leaders allowed to start at Level 3 while building context, with a planned trajectory to Level 1–2 within ninety days, so that autonomy became an expectation with a date on it. And the general manager took on the piece only he could do: telling each leader directly, you own the outcome — organize people around the problem, stop waiting for a role definition — and then judging them on whether they found and fixed what was missing, not on how well they reported it.

None of this is exotic. All of it is specific, written, and enforced — which is what separated it from the announcement that had preceded it.

You are flying in the clouds with instruments you are not being shown.

lateralworks briefing to the division general manager

Case engagement, 2026

04

The schedules
Getting reality into the schedules

Leadership gaps hide in abstractions; schedules make them concrete. The most measurable symptom of the leadership gap was that none of the four programs had a real schedule — and every leader half-knew it. The schedules had been built from derivative-product templates: one pass through each step, no learning cycles, no allowance for the iteration that first-of-a-kind work always demands. For months the milestones held, because early phases forgive optimism. Then they began to slip together. The slip was not the teams failing. It was the schedule finally catching up with reality.

Confirmation bias, institutionalized

A schedule with no gap to target feels like good news and is almost always the opposite. The teams had convinced themselves they were on schedule even though the people closest to the work knew they were not, because a happy schedule felt safer than a real one — safer to present, and safer to live inside for one more week. FTTM inverts the reading: a schedule that matches the target on day one has been reverse-engineered from the answer, and a visible gap between target and reality is not a failure but a tool. The gap is the engine of before-the-fact urgency — it justifies the resources, decisions, and parallel paths that close it, and hiding it does not shrink it; it only moves the discovery to a date when nothing can be done [4, 15].

Planning the learning cycles

The structural fix was to plan iteration explicitly. Every technical problem on the critical paths was rated easy, medium, or hard, and budgeted accordingly: roughly two full learning cycles for easy problems, three or four for medium, five or more for hard — each cycle a complete design-build-test-learn loop sized with known DOE (design of experiments) and fab cycle times. An educated guess with a method beats a schedule that pretends. The companion convention closed the other hole: when a developing problem had no answer yet, the schedule could no longer say nothing. An agreed worst-case placeholder went into the plan, keeping it closed-loop, to be pulled in as the problem resolved. “No answer yet” had been quietly becoming “no task in the plan,” and open-loop unknowns are how a schedule lies while every individual line item tells the truth.

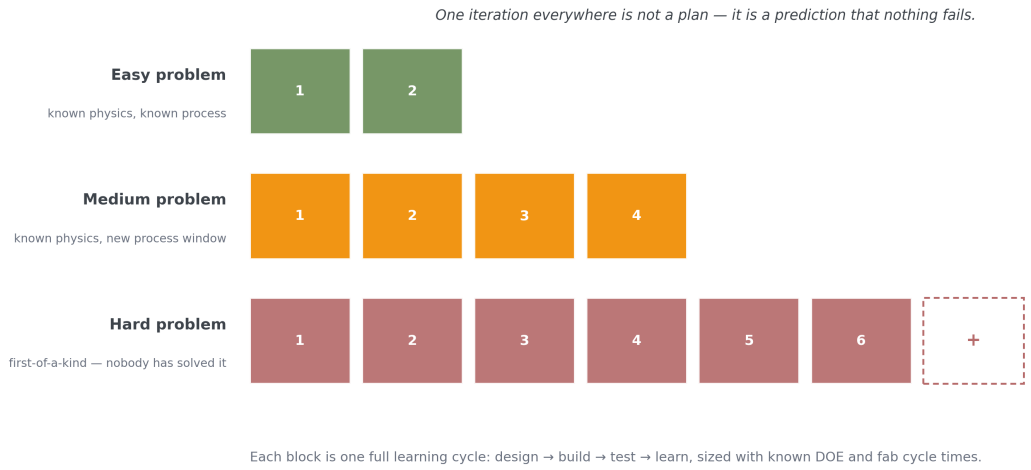

Figure 5. Learning cycles budgeted by problem difficulty. A single pass through design-build-test is a prediction that everything works the first time — a prediction that has never come true on first-of-a-kind work.

Isolating the hard 10%

Each program carried a small set of problems nobody had solved before — the die-crack mechanism, the on-resistance shortfall, the SMT fallout — holding up the 90% of work the teams had done a hundred times. Left inside the product schedules, these research-grade problems leaked into every meeting, consumed the leaders, and split the few experts who could solve them across four programs at once. One problem cost a month of large-meeting confusion before structured validation even began. The FTTM answer is isolation: pull the hard 10% out of the product schedules into dedicated tiger teams — the best people, roughly ten, three months, nothing else on their plate — running their own tightly refreshed plans with executive visibility, their status feeding back into each product schedule. Solutions come from focused people, not from large meetings. The companion discipline is a trigger: when an issue shows the telltale signs, structured problem solving starts immediately, with a named technical owner driving the fix and the PM facilitating — not three weeks later, after the meetings have failed.

The refresh as a keystone habit

Honest structure decays without cadence. The operating rhythm that holds the schedules honest is the weekly refresh: the schedule current with reality by Tuesday evening, non-negotiable, before the Wednesday working meeting; the PM applying first-level scrutiny to every functional input — if a deliverable is a month off its date, the PM questions it before the room does; the critical path recalculated after every refresh; and pull-in decisions made against the top three to five paths, not just the longest one [15, 16]. Trend charts across the four programs distinguished the controllable slips from the research-driven ones, so the teams could fix what was theirs to fix now rather than have it bite later, once the big problems cleared. The refresh is where the leadership gap and the schedule gap turned out to be the same gap: a leader who cannot narrate their own critical path has not yet taken ownership of their product, whatever the org chart says.

05

The general manager
Arming the owner of the outcome

A heavyweight structure changes the general manager's job as much as anyone else's — a fact the literature on empowered teams mostly skips. The GM in this case owned all four programs. Under the old structure he had been the de facto integrator of everything; under the new one his job was to hold the ceiling up: provision the teams, protect the model, and demand reality. lateralworks prepared two working instruments for him, and they are worth examining in detail because they show what executive support for a lateral structure actually consists of.

Seven observations, seven actions

The first instrument was a one-page briefing: seven observations from inside the teams, the root cause behind each, and the actions only the general manager could take. Its opening line set the register: the program is on a collision course with failure if reality does not get into the schedules soon. The problems themselves are solvable — problems exposed early almost always are; problems discovered in the eleventh hour rarely are. That is the first principle of FTTM: pull the pain forward.

- **Leaders are not leading.** Tell each leader directly: you own the outcome. Judge them on whether they find and fix what is missing, not on how well they report it.
- **Confirmation bias on schedules.** Ask each leader two questions: does your schedule reflect reality today, and does it include every iteration you expect to need? Treat a schedule with no gap to target as a red flag, not good news.
- **The critical 10% is unmanaged.** Isolate it. Stand up dedicated tiger teams and give them everything they ask for.
- **None of the four schedules is real.** Direct every PM to plan learning cycles explicitly and put a worst-case placeholder on every open unknown.
- **No safe way to surface bad news.** Publicly back the first person who surfaces an ugly gap. Reward gap-surfacing in reviews. Say it out loud, often: bad news early is good news.
- **The resource excuse is untested.** Make every leader quantify the ask — names and numbers, not headcount totals. Fund the specific asks fast. Kill the general excuse.
- **Excuses are still on the table.** Remove every named excuse in writing — people, skills, decisions, access — then hold delivery accountability with no exceptions.

The sequence in the last item matters more than any single action. Accountability without provisioning is unfair; provisioning without accountability is charity. FTTM does both, in strict order: first remove every excuse, so nothing is left to point at when the date arrives, then hold the fire. Teams that have been counting on their excuses will test whether the sequence is real.

Ten questions, once a month

The second instrument was a walk-through protocol: each leader and their PM present their schedule to the GM live in the scheduling tool, walking the first five critical paths. One hour per program, no slides, schedule current as of that morning. The PM drives the tool; the leader answers the questions. Where are we against the target, plus or minus — in weeks, as a number? Walk me through critical path one, task by task. Which tasks assume everything works the first time? What are the top three technical problems between here and delivery, and is solving them all that those people do? What do you need that you are not getting — how many people, which skills, which decisions? Each question comes with a listen-for: “on track” without a number means the gap is hidden; a flat trend with no pull-ins means drift is being absorbed silently; a vague call for resources is the excuse, not the constraint.

The protocol’s designers were explicit that the answers matter less than the preparation the questions force: leaders will get into their own schedules before they will stand in front of the general manager with one. Run monthly, as a cadence rather than an event, the walk-through became the primary development mechanism for the leaders themselves. By the second round, the answers to the iteration and risk questions are expected to be in the schedule, not in the room. The graduation test is behavioral: when a leader answers every question from their own plan without looking at the PM, the pain has been pulled forward — and the general manager is no longer flying in clouds with instruments he is not being shown.

Beneath both instruments runs the same cultural mechanic: the general manager’s first reaction to bad news determines how much bad news reaches him afterward. In an organization whose history taught people that stepping out was dangerous, protection has to be visible and repeated before behavior changes [10]. The briefing made that his personal work — not a delegable communications task.

06

Lessons
What it takes to tilt

The question this case answers is the one its general manager asked at the start: how do you get a highly functional organization to tilt on its side and become a lateral, end-to-end execution system that moves quickly — and what does that do to the people who have never managed in an empowered, holistic structure? Eight lessons carry from this engagement to any organization attempting the tilt.

- **1 · Structure moves in a day; authority moves in months.** The org chart is the fast part. Plan for the lag between announced empowerment and exercised empowerment, measure it with the freedom scale, and treat a persistent two-level gap as a structural alarm, not a personnel verdict.
- **2 · Write the operating contract down.** Role charters with explicit freedom levels per decision area, a written leader/PM split, and named overlay roles. Every ambiguity left unwritten reverts, under pressure, to the old model — because the old model is what everyone already knows how to do.
- **3 · Nobody learns to lead alone.** The CEO/COO pairing is the unit of leadership development, and the mutual accountability it creates is what the team literature has always identified as the engine of performance [12]. A first-time product boss paired with a strong PM, in constant contact, learns the job at speed. The same person alone reverts to their function.
- **4 · The host sets the ceiling.** Most of what looks like team underperformance is host behavior: part-time roles, unprovisioned asks, slow decisions, interrupts. Fix provisioning before judging the team [14, 17].
- **5 · Dedication is the design, not a preference.** 100% assignment for core team members, no second hats for leaders, hiring plans that assume it. Programs of this economic size justify it many times over.
- **6 · Separate research from execution.** First-of-a-kind problems get their own tiger teams, their own schedules, and the best people full-time. Product schedules plan the learning cycles that remain.
- **7 · Protect the messenger, personally and publicly.** The general manager's first reaction to bad news is the strongest culture signal in the company. Bad news early is good news — said out loud, often, and backed the first time it costs something.
- **8 · Use the schedule as the leadership curriculum.** Monthly walk-throughs where the leader, not the PM, narrates the critical paths convert schedule discipline into ownership faster than any training course.

Where it stands

This case is written from inside a live engagement, and it makes no claim the programs cannot yet support. What can be reported is the state of the tilt at the time of writing. The role charters are issued and aligned, with every leader and PM having worked the leader/PM split in the same room. The Tuesday-evening refresh is a standing expectation across all four teams. The next schedule build plans learning cycles and worst-case placeholders from the start. The leaders are being dotted-lined to a single overall owner as hiring lands, with dedication now a requirement of the hiring plan rather than a preference. The research-grade problems are moving out of the product schedules into project-managed task forces. And the general manager holds the briefing and the walk-through protocol reproduced in section 05, with the first monthly cadence on the calendar. The honest test of the case — leaders answering every question from their own plans — sits a few walk-throughs in the future, which is exactly where a real transformation timeline says it should.

What the tilt asks of people

Every role in the tilted organization gives something up. The functional manager gives up daily command of their best people and receives, in exchange, the standards authority, the talent pipeline, and the cross-program knowledge role — a real job, but a different identity, and the charter has to say so explicitly or the old identity reasserts itself through the dotted line. The embedded lead gives up the safety of their silo and receives local authority they may at first not want. The PM gives up the coordinator's alibi — I can only track what they tell me — and becomes accountable for a schedule that reflects reality. The leader gives up the option of describing problems and becomes the person of whom organizing is expected. And the general manager gives up the integrator role he had grown into and becomes the keeper of the ceiling: provisioner, protector, and the one voice that can make bad news safe.

None of these exchanges completes on announcement day. The engagement's realistic clock — Level 3 to Level 1 in ninety days for a new leader, a behavior change visible by the second monthly walk-through, hiring that lands over quarters — is itself a finding: the tilt is a program, with a schedule, and it should be managed like one [18]. The companies that get this right do not have better people. They have a host that provisions, an operating contract in writing, and a cadence that keeps both honest.

> **The bottom line.** Empowerment is not a value statement; it is a measurable operating condition. Put the authority in writing, pair the leaders, provision the teams, plan the learning cycles, and make bad news safe — then hold the fire. That is how a vertical company earns a lateral speed it was never built for.

A

Appendix
The freedom scale

The freedom scale is a framework for quantifying empowerment, developed from lateralworks' study of fast product development organizations and rooted in the delegation ladder Oncken and Wass sketched in 1974 [9, 17]. It measures the degree to which individuals and teams are trusted to act independently. Lower values mean faster decisions and greater ownership. Best-in-class product organizations run their teams at Levels 1–2 on most decisions; normal organizations run at Levels 3–5, where functional managers keep before-the-fact control — slower, bottlenecked, and disempowering.

The five levels

- **Level 1 — act, routine reporting only.** After-the-fact control; the fastest level. Reserved for trusted professionals on decisions within their domain.
- **Level 2 — act, but advise at once.** During-the-fact control. Collaborative and fast; right for decisions with cross-functional impact where visibility matters but speed cannot wait.
- **Level 3 — recommend, then take resulting action.** Before-the-fact control. The "unless otherwise directed, I intend to…" level; a reasonable starting point for people new in role.
- **Level 4 — ask what to do.** Appropriate only for trainees. Creates dependency on the boss and disempowers the individual.
- **Level 5 — wait until told.** Completely passive. Anyone operating here on routine decisions has been structurally disempowered.

Key principles

- Freedom is situational, not absolute: a person may hold Level 1 on technical decisions and Level 3 on hiring. Levels attach to decision areas, not to people.
- People disempower themselves by asking. The more someone asks, the more they get told, and the closer they drift to waiting until told. Best practitioners break the cycle by defaulting to action.
- Trust produces speed; control produces slowness. Level 1–2 teams make real-time decisions without hierarchical bottlenecks — and management gains leverage, not loses it, because the team owns the outcomes.
- Suggested levels are the target state, not the starting point. New hires may run at Level 3 while building context, with a planned trajectory to Level 1–2 within ninety days.
- If a role consistently operates two or more levels below its suggested freedom, that is a diagnostic signal: the individual needs development, the culture is overriding the structure, or the role definition needs revisiting.

B

Appendix
The thirteen problems

Midway through the engagement, lateralworks summarized what it saw slowing the four teams as thirteen problems in four groups. The list is reproduced here in anonymized form because its shape is the case's argument in miniature: only one of the thirteen is a conventional org-chart problem. The rest are mindset, front-end management, and decision discipline — the parts of a tilt that no reorganization slide can deliver, and the reason the fuzzy front end, where half to all of a development cycle can disappear before anyone notices, is the cheapest place in the whole project to take out time [4, 13].

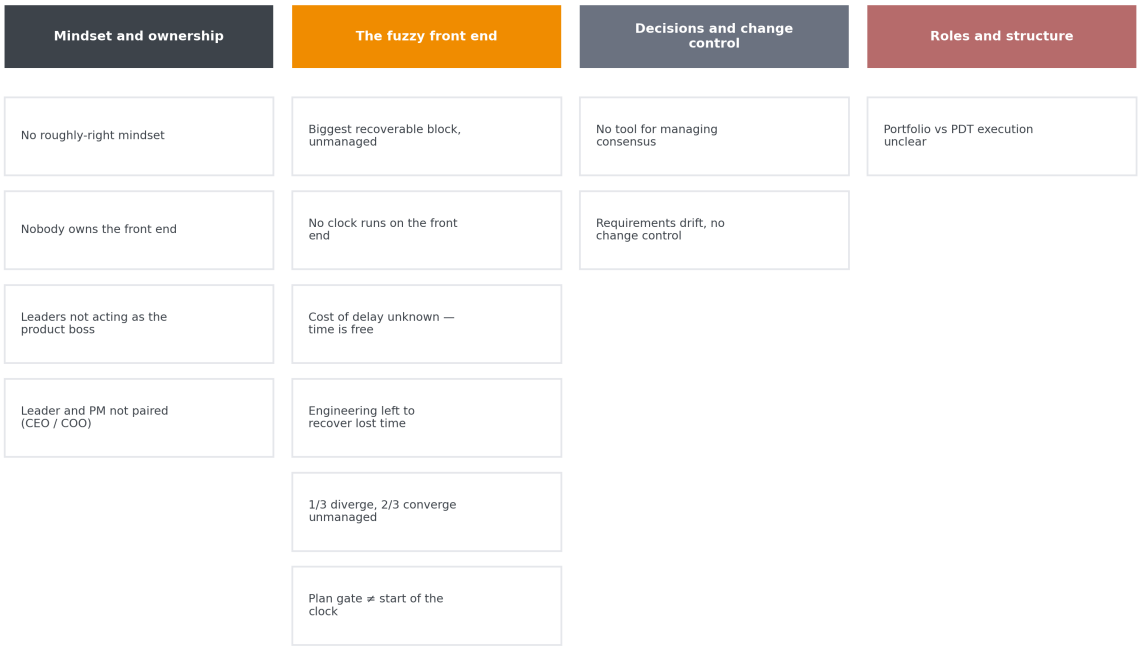

Figure 6. The thirteen problems, grouped. Six of thirteen sit in the fuzzy front end — the stretch between the first mention of an idea and a staffed team working to a committed plan.

Sources

References

[1] Clark, K. B., and Wheelwright, S. C. “Organizing and Leading Heavyweight Development Teams.” *California Management Review*, Vol. 34, No. 3, Spring 1992, pp. 9–28.

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

[3] Larson, E. W., and Gobeli, D. H. “Organizing for Product Development Projects.” *Journal of Product Innovation Management*, Vol. 5, No. 3, 1988, pp. 180–190.

[4] Smith, P. G., and Reinertsen, D. G. *Developing Products in Half the Time.* Van Nostrand Reinhold, 1991.

[5] Takeuchi, H., and Nonaka, I. “The New New Product Development Game.” *Harvard Business Review*, January–February 1986, pp. 137–146.

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

[7] Sobek, D. K., Liker, J. K., and Ward, A. C. “Another Look at How Toyota Integrates Product Development.” *Harvard Business Review*, July–August 1998, pp. 36–49.

[8] Christensen, C. M., and Overdorf, M. “Meeting the Challenge of Disruptive Change.” *Harvard Business Review*, March–April 2000, pp. 66–76.

[9] Oncken, W., and Wass, D. L. “Management Time: Who’s Got the Monkey?” *Harvard Business Review*, November–December 1974.

[10] Edmondson, A. “Psychological Safety and Learning Behavior in Work Teams.” *Administrative Science Quarterly*, Vol. 44, No. 2, 1999, pp. 350–383.

[11] McChrystal, S., Collins, T., Silverman, D., and Fussell, C. *Team of Teams: New Rules of Engagement for a Complex World.* Portfolio/Penguin, 2015.

[12] Katzenbach, J. R., and Smith, D. K. *The Wisdom of Teams: Creating the High-Performance Organization.* Harvard Business School Press, 1993.

[13] Reinertsen, D. G. *Managing the Design Factory: A Product Developer’s Toolkit.* Free Press, 1997.

[14] lateralworks. “Fast autonomous teams.” lateralworks.com, 2019. https://lateralworks.com/ideas/2019/1/17/fast-autonomous-teams

[15] lateralworks. “The long pole.” lateralworks.com, July 2026.

[16] lateralworks. “Make your bed.” lateralworks.com, July 2026.

[17] lateralworks. Internal assessment database. Engagement data across client programs, 1988–2026.

[18] Kotter, J. P. *Leading Change.* Harvard Business School Press, 1996.