Whitepaper

10 best practices of fast teams

Ten proven practices that consistently move products to market faster

FTTM methodology series.

A focused installation sequence drawn from the lateralworks catalogue of thirty-one best practices documented in *FTTM New Product Development Best Practices, Revision 2018.002*. Anchored in thirty-six years of field research across 200+ technology programs — from the team that built the first PlayStation to a $7 billion semiconductor fab that finished first silicon two weeks ahead of schedule.

Prepared by
lateralworks
FTTM methodology

Date
May 2026
Revision 2026.05

Online
lateralworks.com
Silicon Valley, founded 1988

Table of contents

10 best practices of fast teams

Core thesis. Fast time to market is not the result of working harder. It comes from ten disciplined practices, installed together, that turn the schedule into a living instrument and the team into its owner. Speed is not an outcome — it is a habit, protected one Tuesday morning at a time.

Overview

Abstract

Speed is a cultural asset, not a tooling choice. Over thirty-six years of multi-company research and 200+ client engagements [1, 2], lateralworks has identified a consistent set of practices that distinguish teams who deliver the right product at the right time from teams who do not. The practices are simple. Installing them, sustaining them, and defending them against organizational drag is the difficult part.

This paper presents **ten** of those practices drawn from the lateralworks catalogue of thirty-one documented in *FTTM New Product Development Best Practices, Revision 2018.002*. The ten were chosen because they compound — each amplifies the others — and because a team that installs these ten owns the minimum viable FTTM operating system. The remaining twenty-one practices, including fuzzy-front-end management, kill-project discipline, and portfolio-to-capacity alignment [3], become valuable once the core ten are stable.

Where the lateralworks framework parallels published research, this paper says so explicitly. The Freedom Scale adapts Oncken and Wass's monkey-management work [4]. The core team draws on Wheelwright and Clark's heavyweight team framework [5]. The host pillar echoes Reinertsen's queueing math on work-in-process [6] and Hamel and Zanini's diagnosis of excess management cost [7]. End-to-end voice of the customer extends the QFD tradition launched by Hauser and Clausing [8]. Cost of delay at the contributor level builds on House and Price's break-even time framework [9] and Reinertsen's later economic model [6]. The practices are old in parts and original in their integration; the originality is the installation sequence and the discipline that keeps them in place.

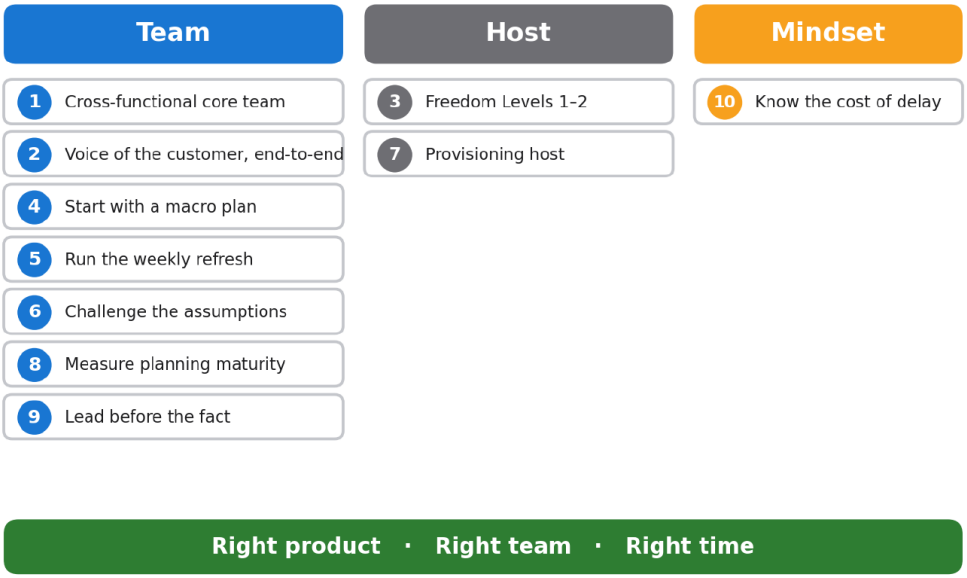

Figure 1. The ten practices, grouped by the three FTTM pillars — Team, Host, and Mindset. Cropped from the source figure.

Three ideas run through every section. **One:** speed is protected or destroyed by the environment the team operates within — the binding constraints sit in the mindset and host cells, not the team cell. **Two:** a team cannot be fast if it is not trusted; the Freedom Scale sits underneath every other practice. **Three:** the schedule is the instrument the team uses to drive itself. A schedule that is refreshed weekly and pulled in continuously behaves very differently from one updated monthly and defended quarterly [10].

01

Framework
The FTTM operating system

The Fast Time To Market framework organizes its best practices along two axes. The horizontal axis names **who owns the practice** — the corporate mindset, the host organization, or the team itself. The vertical axis names **where the practice applies** — portfolio planning, the operating environment, or execution. The result is a nine-cell matrix where every practice lands in one place and every dependency becomes visible [3].

Most companies focus on the team cells. They hire better program managers, install new tooling, and send people to planning workshops. These interventions help at the margin, but they do not unlock the speed available to the organization. The binding constraints almost always sit in the mindset and host cells — the environment around the team — and those cells belong to senior leadership, not the program office.

This pattern repeats across industries. lateralworks observed it at Sony on the first PlayStation [11]. We observed it at Philips on the Velo PDA, whose engineering lead later carried these practices to Apple to ship the iPod, iPhone, and iPad [12]. We observed it at GlobalFoundries Fab8, a $7 billion greenfield semiconductor fab where first silicon landed two weeks early at a measured cost of delay of roughly $5 million per day [13]. The practices are the same; the host environment is what varies.

The framework

Three pillars, one outcome

The three pillars are Mindset, Host, and Team. The team executes the integrated plan. The host provisions the team with people, decisions, and an interrupt-free environment. The mindset establishes speed as the primary asset — the lens through which every staffing, calendar, and portfolio decision is filtered. All three are required. Strong teams inside a normal host plateau within a year; strong hosts wrapped around weak teams produce predictable results but rarely fast ones.

Figure 2. The three FTTM pillars. The team executes the plan; the host provisions the team; the mindset establishes speed as the primary asset. All three are required.

What the framework demands of leadership

Speed is not delegable. An executive team that says the right things about time to market but allows the host to behave normally — reassigning resources mid-program, inserting review gates, running more programs than capacity supports — has not adopted the mindset. The mindset shows up in calendar decisions, staffing decisions, and whether the organization trusts its teams enough to stay out of their way. Hamel and Zanini estimated that excess management cost the US economy roughly $3 trillion a year [7]; most of that cost arrives, on a given Tuesday morning, as a host interrupt.

The ten practices that follow are the operational expression of that mindset. Each is a concrete behavior that leaders, hosts, or teams either perform or do not. Nothing in the framework is abstract. Everything can be observed on a given Tuesday morning.

Where the binding constraint actually sits. Most acceleration programs put energy into the team cell because that is where the program office can reach. The lateralworks engagement data is consistent across industries: when team-level practices stall, the cause is almost always upstream — portfolio overload, unstable resource commitments, decision queues at the host level, or a leadership team that has not made speed the primary asset. Fix the host and team-level improvement compounds. Leave the host alone and team-level improvement fades within a year.

02

Practices

Ten habits that compound

The ten practices below compound. A team that installs any one in isolation gets marginal improvement; a team that installs all ten gets the time-to-market compression that lateralworks engagements are known for. The dependencies between practices matter — the weekly refresh requires a macro plan to refresh, the macro plan requires a core team to build it, the core team requires a host willing to dedicate people to it. The installation order in section 03 follows those dependencies.

Each practice is presented the same way. A short statement of what the practice is and why it works. A figure cropped from the lateralworks source document. A paragraph that points to where the practice appears in the published research — often a familiar name (Wheelwright and Clark, Reinertsen, Oncken and Wass) attached to an idea the reader already half-recognized. A field example from a lateralworks engagement where the practice paid out in measurable schedule compression. And a callout that names the failure mode normal teams fall into when the practice is absent.

Two threads to keep in mind. **First**, every practice is observable on a Tuesday morning. Nothing here requires reorganization, software purchase, or executive retreat. **Second**, every practice has been done before — the lateralworks engagement data spans 200+ programs since 1988, and the published literature stretches further back than that [1, 5]. None of the ten is novel. The originality is in the integration: which ten, in what order, defended against which failure modes.

Practice 01

Build a cross-functional core team

The core team is the single most important structural decision a program makes. Done well, it produces a small, dedicated, cross-functional group of five to seven heavyweights who own the product from concept through volume production. Done poorly, it produces a steering committee of department heads who protect their functions and assign blame at the integration points.

The best core teams share five characteristics. They form early, during concept, before requirements are finalized. They are led by heavyweights with end-to-end ownership authority, not lightweight project coordinators. They are 100% dedicated to the program. They co-locate, physically or in a disciplined virtual room. And they operate at Freedom Levels 1–2, covered in practice 3.

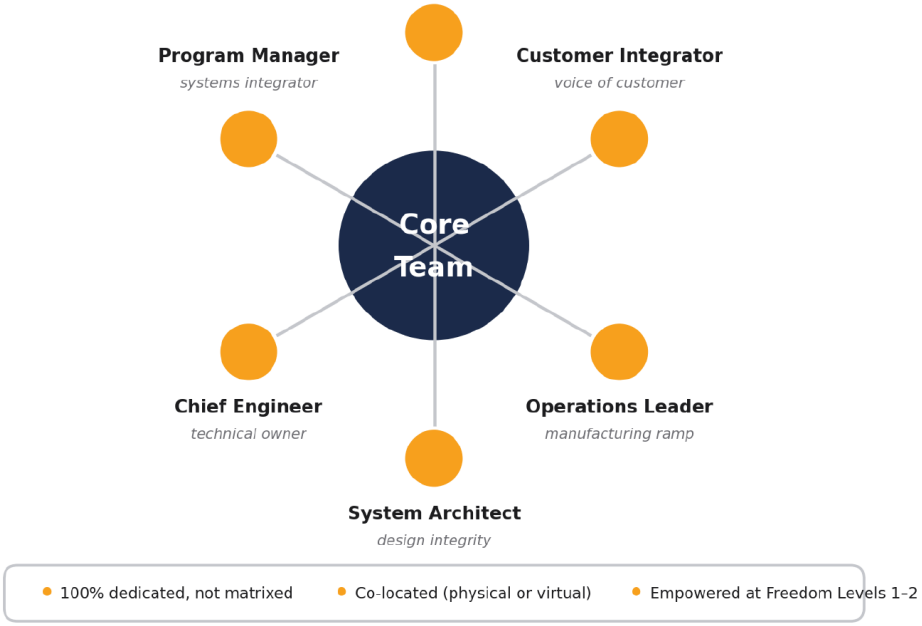

Figure 3. The six heavyweight roles on a mature core team. The roles are project roles, subordinating the individual's functional role while

Figure 3. The six heavyweight roles on a mature core team. The role is the project role, not the functional title — a person can be Director of Silicon Engineering and serve as Chief Engineer on one program. The role comes first.

Why matrixed teams are slower

In the normal matrixed model, core team members serve three to five programs and report primarily into their function. Decisions route up into the function, across to another function, and back down. This pattern adds weeks to every decision. When two programs conflict for a resource, the function resolves the conflict, not the program — the opposite of what portfolio speed requires.

In the literature

Wheelwright and Clark named this pattern in 1992 [5]. Their framework distinguished four team types — functional, lightweight, heavyweight, and autonomous — and showed that only the latter two produced the integration needed for breakthrough products. The heavyweight team, in their formulation, has a project leader with real authority over budget and people, dedicated cross-functional membership, and a single point of accountability for the program outcome. The lateralworks core team is heavyweight in exactly this sense — the vocabulary is slightly different (program director, chief engineer, system architect, operations leader, customer integrator, program manager) but the structure is the same.

From the field

The engineering executive who led the Philips Velo PDA program — a program built using FTTM with a heavyweight core team — carried these practices to Apple, where he became one of the principal authors of the iPod, iPhone, and iPad [12]. The practices traveled with the person. The lateralworks engagement at GlobalFoundries Fab8 used the same model at a much larger scale: lateralworks joined as part of the initial five-person planning team and stayed through first silicon, which landed two weeks early on a $7 billion fab where every day of acceleration translated to roughly $5 million of operational value [13].

> **Failure mode:** the lightweight coordinator. When the program is run by a project manager who reports across to functional VPs rather than down through them, the core team is structurally unable to make trade-off decisions. Every conflict escalates. The schedule reflects whatever the slowest function is willing to commit to. The fix is structural, not behavioral — hire or promote a heavyweight with P&L; authority, then move the program under that leader.

Practice 02

Schedule voice of the customer end-to-end

Voice of the customer is not a workshop held once in the requirements phase. The best teams schedule customer engagement across the full time-to-market cycle — from early strategy counsel through field support — and treat the customer's time as a resource to be planned, not a meeting to be booked.

This practice sits at position two because the output of VOC shapes every subsequent decision on the program. A core team without a scheduled VOC cadence is optimizing for a specification that may not map to what the customer values. When the specification drifts from the underlying requirement — usually because an engineering trade-off looked attractive in isolation — the customer relationship catches it. Programs that engage customers only at the front end have no mechanism to catch this drift and tend to discover it at beta, when correcting it is expensive.

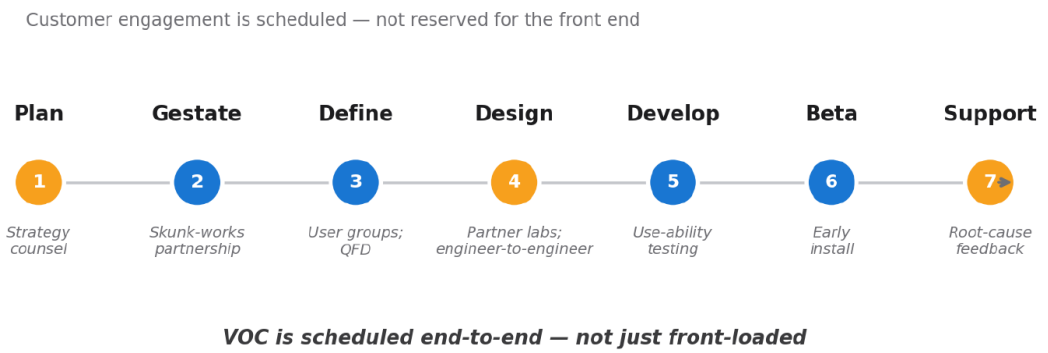

Figure 4. Voice of the customer activities mapped to TTM phases. Different engagement techniques fit different phases; no single technique carries the whole cycle.

Different tools for different phases

Early planning calls for strategic counsel from a handful of trusted lead customers. Gestation benefits from skunk-works collaboration — engineer to engineer, under NDA, with prototypes that are explicitly early and unfinished. Requirements call for user groups and structured techniques like Quality Function Deployment [8]. Design calls for partner labs and direct designer-to-designer contact. Development benefits from usability testing on real customer workflows. Beta installs expose integration issues that internal test cannot reproduce. Field support and root-cause analysis close the loop and feed the next program.

In the literature

Hauser and Clausing launched Quality Function Deployment into the Western product development literature in 1988 [8], introducing the "house of quality" matrix that translates customer requirements into engineering specifications. QFD remains the single best-documented technique for the requirements phase of VOC, and FTTM practice 3.5 — understanding customer wants versus hows, prioritizing drivers that maximize satisfaction — is the lateralworks operationalization of it [3]. The broader insight — that customer engagement must be continuous rather than front-loaded — is consistent with the lean product development tradition captured by Reinertsen [6] and with the design thinking literature [14].

> **Insight:** fear of sharing real schedule and performance data with customers is a classic normal-team pattern. The best teams share it deliberately. The customer becomes a partner in hitting the window rather than an audience for a polished commitment — and is willing to help when the program needs a decision that only the customer can make.

Practice 03

Operate at Freedom Levels 1–2

Empowerment is a word leaders use often and an instrument they rarely calibrate. The Freedom Scale, adapted from Oncken and Wass [4], makes the calibration explicit. It names five levels of decision authority, each with specific leadership behavior on one side and specific team behavior on the other.

Level 1 is full authority: the team acts and reports what happened. The leader has after-the-fact control. Level 2 is the team acts and advises at once: the leader has during-the-fact control. Level 3 is the team recommends and acts unless redirected: the leader has before-the-fact control. Levels 4 and 5 — ask what to do, wait until told — are trainee behaviors. Any experienced professional operating there is being under-used, often at significant cost to the organization.

Figure 5. The Freedom Scale. Levels 1–3 are where leverage and growth happen. Levels 4–5 are trainee status only, and should be escaped, not tolerated.

UNIDIR: default to action

On one of the fastest programs lateralworks ever studied, team members opened emails with the acronym **UNIDIR** — Unless Otherwise Directed, I Intend On. The sender stated what they were about to do and moved. The recipient had a small window to redirect. Most of the time nobody redirected, and the work happened on the sender's schedule rather than on whoever was slowest to reply. UNIDIR is Freedom Level 2–3 behavior encoded into an email convention. It is trivial to adopt and remarkably effective.

In the literature

Oncken and Wass's 1974 *Harvard Business Review* article on management time — the original "monkey on the back" metaphor — introduced the idea that a manager's job is to keep the next action with the subordinate rather than take it on themselves [4]. The article ran in HBR in November–December 1974 and was reissued as a Classic in November–December 1999 with a commentary by Stephen Covey. The lateralworks Freedom Scale extends Oncken's framework by making the level of delegation explicit and situational — a team may legitimately run at Level 1 on technical decisions and Level 3 on hiring, but the level must be named.

> **Warning:** the most common disempowerment is self-inflicted. Teams ask for permission they already have because they have not paused to notice the level they were granted. A Freedom Level audit in the first month of a program consistently finds two to three categories where the team is working below the level the host has authorized.

Practice 04

Start with a macro plan

The default approach to program planning is bottom-up. Each function builds its detailed schedule, the pieces are stitched together, and the integrated result is presented to senior management weeks or months into the project. The integrated result is almost always unrealistic, internally inconsistent at the hand-off points, and too late to inform the strategic decisions that determine whether the program finishes on time.

The FTTM approach is top-down. In a one- or two-day offsite, the five- to ten-person core team builds a macro plan of no more than fifty tasks that covers the program from concept to volume production. The plan is organized around four to six major cross-functional integration points. The outputs of each integration point are negotiated and agreed across functions on the spot. This forces integration to happen top-down, before any function has built a detailed plan that assumes a convenient handoff.

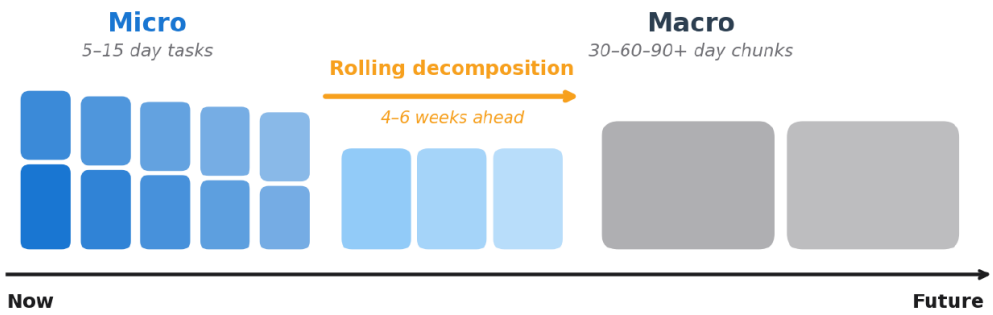

Figure 6. Macro-to-micro planning. Near-term work is broken into 5–15 day micro tasks; future work stays at 30–60–90 day macro chunks and is decomposed only as the team learns.

The gap is the real deliverable

The primary output of the macro planning workshop is not the plan. It is the **gap** — the difference between when the program needs to be done and when the critical path says it will be done. Most programs are late by 30–50% of their overall duration, and most teams do not discover this until the back half of the project. Macro planning produces the gap on day two. That early discovery creates the urgency for the strategic decisions — scope cuts, resource additions, parallel paths — that would otherwise arrive too late to matter.

In the literature

The macro-to-micro pattern is one expression of rolling-wave planning, documented in the PMI Project Management Body of Knowledge [15] and in the agile literature's progressive elaboration concept [16]. Reinertsen's flow principles add the economic argument: in a process with variability, detailed early plans waste effort on work whose inputs are still uncertain [6]. The lateralworks contribution is the specific scale — fifty tasks, four to six integration points — and the discipline of producing the gap as the deliverable rather than producing the plan.

Detail does not equal accuracy. A 3,500-task schedule built in the first week of a program is almost always less predictive than a fifty-task macro schedule built in the first two days, because the detail is based on information the team does not yet have.

Practice 05

Run the weekly refresh

The macro plan and micro decomposition are static artifacts until the team installs the discipline that makes them living instruments. That discipline is the weekly refresh: a standing meeting, same day, same time, every week, at which every activity owner reports progress against scheduled tasks, the schedule is updated live, and the critical path is recalculated.

The refresh is not a status review. Status reviews communicate upward; refreshes drive forward. In a refresh, the team answers three questions about each activity: what is the actual remaining duration, what has changed since last week, and is there any opportunity to pull in the near-term schedule. The critical path reveals the answer. The team walks out knowing whether the end date has held, slipped, or pulled in.

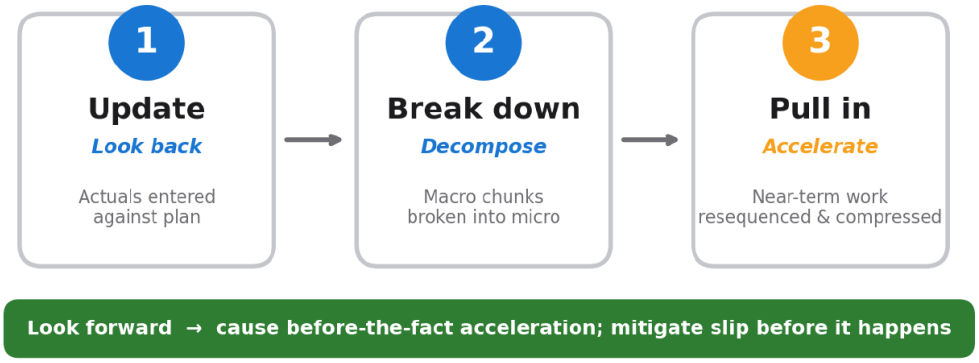

Figure 7. The three-step weekly refresh: update the plan against actuals, break down the next rolling window, look for pull-in opportunities on the near-term critical path.

Cadence over content

The power of the refresh is the cadence, not the content of any single session. Controllability is the point. A team cannot control whether silicon fails at characterization or whether a supplier misses a window, but it can control whether the refresh happens every Tuesday at 9:00 a.m. On a three-year joint development program lateralworks coached in the UK, the refresh was missed exactly once in three years — on Christmas Day.

In the literature

The weekly cadence echoes McChesney, Covey, and Huling's *4 Disciplines of Execution* [10], which makes a similar argument for a weekly accountability rhythm at the team level. The schedule-as-instrument framing connects to the lean product development tradition: Reinertsen's flow principles emphasize fast feedback loops as a prerequisite for managing the variability inherent in product development [6]. The lateralworks adaptation is the specific structure — update, break down, pull in — and the insistence that the cadence itself, not the content of any one session, is the active ingredient.

From the field

lateralworks has documented the weekly refresh as a structured seven-page artifact pulled directly from the schedule and the pull-in meeting transcript [17]. Compiling this artifact by hand takes six to eight hours of PM time, which is why most programs cut it. The fastProjectAI implementation drops the assembly time to a few minutes, which is the difference between a team that runs the practice and a team that knows it should.

Practice 06

Challenge the assumptions

Every schedule rests on assumptions. Some are explicit and named — "we will get silicon by March". Some are implicit and invisible — "we must deliver the full feature set," "we cannot hire additional resources," "marketing owns the customer relationship". The implicit assumptions are usually the binding ones, and they are almost never questioned until the gap has become so large that questioning them is the only remaining option.

The lateralworks Challenge workshop surfaces and questions those assumptions deliberately, before the gap forces the conversation. It is a structured team exercise that separates current thinking into four categories: what is essential, what dominates our thinking, what must be avoided, and what boundaries we must stay within. The team selects a subset — typically about eight — and asks three questions of each: why do we think this, are the reasons still valid, and how could it be done differently.

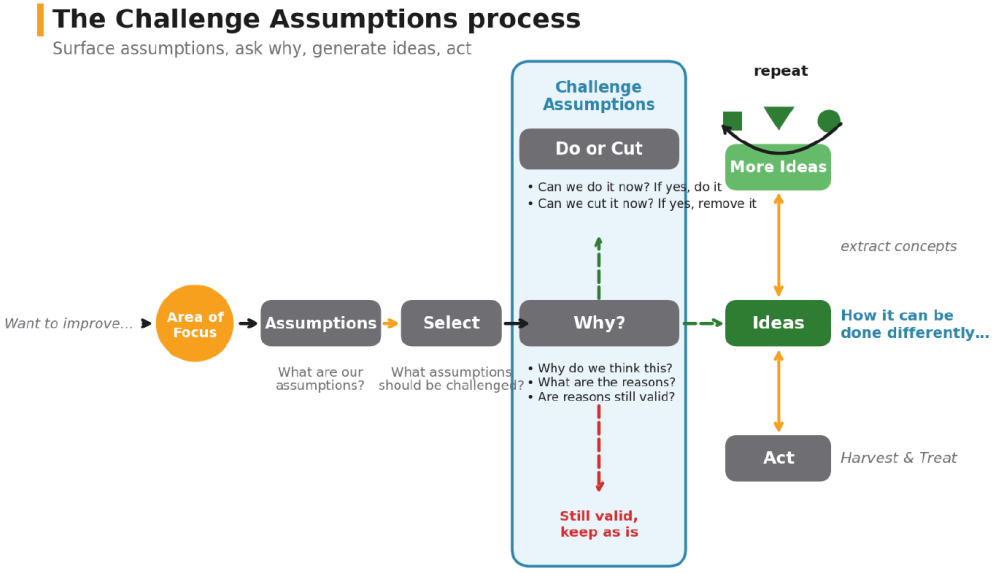

Figure 8. The Challenge Assumptions process. The workshop moves from a focus area through the team's assumptions to a decision: Do or Cut now, keep as-is if still valid, or harvest the idea as a reusable concept for later cycles.

Challengeable and non-challengeable

Not every assumption is worth the team's time. An assumption is **challengeable** if it expresses a belief that could reasonably be different — "we cannot reduce the feature set," "DT performance is non-negotiable," "we must ship on the original committed date". An assumption is **non-challengeable** if it is a fact about the physical world or a legal constraint. The Challenge workshop focuses on the first category.

In the literature

The technique draws on the de Bono tradition of lateral thinking [18] and on Eli Goldratt's Theory of Constraints [19], both of which name the surfacing of hidden assumptions as the leverage point for breakthrough improvement. Senge's work on mental models in *The Fifth Discipline* [20] makes the same point at the organizational level. The lateralworks contribution is the structured workshop format and the four-category sort — essentials, dominating thinking, avoidance, boundaries — that turns the abstract idea of "challenge your assumptions" into a Tuesday-morning exercise the team can actually run.

> **Practitioner note:** the Challenge workshop is most productive when the team is stuck on something and believes there is no obvious path forward. It is least productive when applied prophylactically at the start of a program, before the team has enough context to know which assumptions are binding. Use it when it hurts.

Practice 07

Make the host a provisioner

The host is the organization outside the project team. It is the functional management hierarchy, the executive leadership, the shared services, the finance and HR organizations. It controls the budget, the people, the equipment, and the decision queues that the team depends on. The host creates the conditions for success or failure long before the team does anything wrong.

In a normal organization, the host interrupts. Fourteen signatures are required to approve a critical engineering requisition. Project resources are redirected mid-stream to cover a higher-priority emergency. Artificial gates and review cycles are imposed to maintain control over daily decisions. The portfolio contains more projects than capacity supports, so every project is starved of some resource it needs. The cost of these interrupts is measured in quarters, not days.

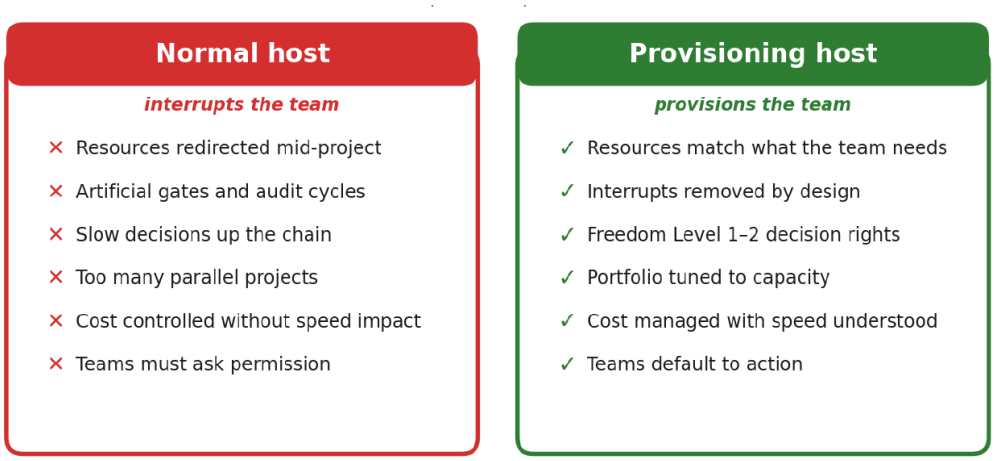

Figure 9. The environment the host creates determines portfolio speed. Interrupting hosts make individual programs slow; provisioning hosts make the whole portfolio fast.

The 25% productivity floor

lateralworks has repeatedly measured engineer time on normal programs and found less than 25% spent on engineering work. The other 75% is consumed by meetings that could have been emails, reviews that produce no decisions, context switches between programs, and rework driven by late information. Every one of those is a host interrupt. The provisioning host does not fix them one at a time — it redesigns the environment so they do not appear.

In the literature

Reinertsen's queueing analysis is the formal model behind the host pattern [6]. At 60% utilization, queues are short and predictable; at 90% they have multiplied roughly nine-fold; at 95% they spiral. Small overcommitment produces large schedule damage. Starting fewer projects is how more projects finish on time. Hamel and Zanini estimated that excess management cost the US economy roughly $3 trillion a year [7] — a number that, broken down to the program level, matches the lateralworks 75% interrupt floor. The Project Management Institute's Pulse of the Profession data shows the same pattern from the portfolio side: organizations with mature portfolio management complete significantly more projects on time [21].

From the field

The lateralworks portfolio briefing on starts control [22] documents the cure: a six-step zero-based budgeting loop that ranks projects against weighted criteria using Saaty's Analytic Hierarchy Process [23] and then gates entry against real resource capacity. Projects load in priority order until cumulative engineering demand crosses the ceiling; everything below the line is deferred, descoped, or killed. The line is the artifact that prevents the active list from quietly refilling between quarterly reviews.

Practice 08

Measure planning maturity

Most quality systems measure outputs. On-time delivery, defect rates, cost variance. Planning maturity measures something earlier — the health of the instrument the team uses to drive itself. A team with a healthy planning instrument tends to produce healthy outputs. A team with a broken instrument produces outputs that look fine in reports and miss the target in reality.

lateralworks uses a four-level maturity metric — ML0 through ML3 — that can be assessed in an afternoon by observing the team's planning artifacts and standing meetings. The metric is self-consistent with the practices already described: ML2 is the level at which the weekly refresh works, ML3 is the level at which before-the-fact leadership operates.

Figure 10. The four planning maturity levels. Most teams live at ML1. The move to ML2 requires months of discipline; the move to ML3 requires the team to be looking forward, not back.

What each level looks like on a Tuesday

ML0. The schedule exists but is not current. Nobody can say with confidence when the program will finish. The gap between target and real end date is unknown. Status reports describe activity, not trajectory. **ML1.** A schedule exists and is accurate to what the team knows today. The gap is identified and named. The critical path is known. Updates happen but are infrequent and driven by external requests. **ML2.** The weekly refresh is installed and running. The schedule is a living instrument. Trends are tracked across refreshes. The team runs the refresh without the coach in the room. **ML3.** The team spends as much time looking forward as looking back. Pull-in is a standing discussion, not an exception. ML3 is rare and hard-won, and correlates strongly with programs that finish ahead of schedule.

In the literature

The maturity-model construct comes from the Software Engineering Institute's Capability Maturity Model [24], later generalized as CMMI. The SEI levels measure process repeatability across an organization; the lateralworks ML scale measures something narrower and more behavioral — the team's relationship to its own schedule. The structural similarity is intentional: both scales are diagnostic before they are prescriptive, and both name the rare top level as the one that requires the team to be improving its own process.

What moves a team up the scale. The factors are all behavioral, not technical. Team structure (dedicated, reporting to a heavyweight leader). Time committed to planning (main focus or afterthought). Contact with the customer (driving genuine urgency). Discipline (the refresh rhythm does not move). Executive pull (senior management asks questions that reward schedule maturity). When one of these is missing, the team plateaus.

Practice 09

Lead before the fact

Three modes of leadership produce very different results. After-the-fact leadership responds to problems that have already happened. During-the-fact leadership responds as they happen. Before-the-fact leadership responds to the weak signals that predict them. The three modes correspond loosely to the three productive Freedom Scale levels.

Before-the-fact leadership is not about working harder or longer. It is about paying attention to a different set of signals. A slight slip in a non-critical task that feeds the second critical path. A quality metric trending in the wrong direction but still inside spec. A supplier responsive but slightly less responsive than last month. None of these is urgent. All of them are predictive.

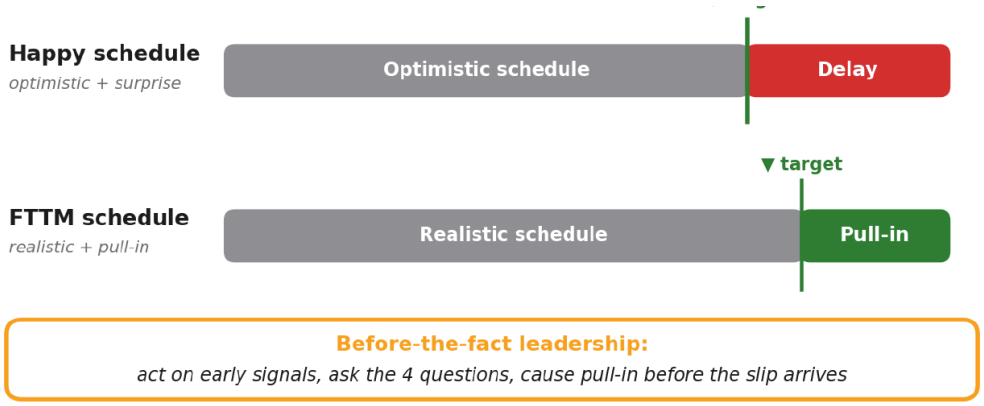

Figure 11. Happy schedules produce end-of-project surprises. Realistic schedules with continuous pull-in absorb the surprises before they become delays.

The four questions

lateralworks teaches four questions that form the core of before-the-fact leadership. At each refresh, and whenever a decision point arises, the core team asks: **1.** What is the worst thing that could happen between now and the next refresh? **2.** What early signal would tell us it is starting? **3.** What action would we take if we saw that signal? **4.** Can we take any of that action now, while it is cheap? The fourth question is the one that separates mature teams from reactive teams. There is almost always something that can be done now, while the problem is small and the cost is contained.

The second critical path

The second critical path — the longest path through the schedule that is not quite the critical path — is where most program slips actually come from. A small slip on the primary critical path is visible and gets attention. A small slip on the second critical path is invisible until the second path becomes the first. The lateralworks briefing on this pattern [25] shows how buffer-burn rate (slip days per week over the last two or three weeks) ranks the near-critical paths by time-to-overtake, surfacing the silent slippers in Monday's conversation rather than the one a month later when the new CP1 is locked in.

In the literature

The before-the-fact framing maps to what Klein and others call recognition-primed decision making [26] — the expert's ability to act on weak signals before they become strong. It also connects to the high-reliability organization literature [27], which documents how teams in domains where failure is catastrophic (aircraft carriers, nuclear plants, ER trauma units) cultivate a preoccupation with failure that lets them act on signals other teams would dismiss as noise. The lateralworks framing brings this discipline to program management, where the failure mode is a missed window rather than a crash.

Core discipline: before-the-fact leadership requires the team to believe its own schedule. If the schedule is known to be optimistic, no weak signal feels weak enough to warrant action. This is why realistic planning (practice 4) and the weekly refresh (practice 5) must be in place before before-the-fact leadership can operate.

Practice 10

Cost of delay at the contributor level

Every day a product is late, revenue walks out the door. Most companies know this at the portfolio level but never translate it for the individual contributor. The CFO has a number. The engineering manager can quote it from last quarter's review. The engineer writing code on Tuesday morning has no idea. That gap between corporate-level awareness and contributor-level behavior is where most of the available schedule compression hides.

In one lateralworks engagement, a high-volume consumer product was losing $1.6 million in profit for every day it was late. That number lived in a finance model nobody on the development team had seen. When it was shared, a queue of decisions that had been stuck for weeks cleared in days. A $50,000 piece of test equipment that would save a week of characterization was suddenly obvious. A two-week hiring delay suddenly had an $11 million price tag attached. The mindset the 2018.002 catalogue calls "pay to save a day" only becomes available once the daily number is visible to the people making daily decisions.

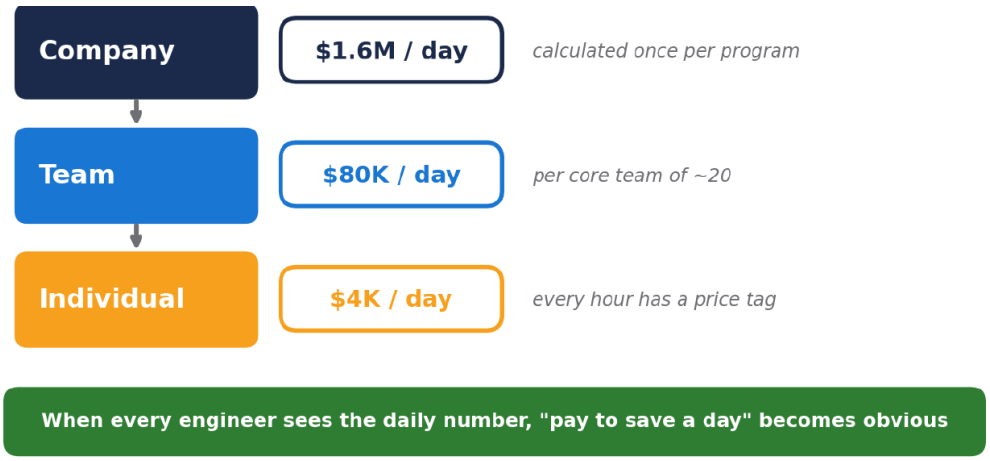

Figure 12. The daily cost of delay, cascaded from the company-level number to a per-engineer number. The example uses $1.6M/day from a real lateralworks engagement with a high-volume consumer product.

What goes into the calculation

Eight factors feed a credible cost-of-delay number. Each is usually partially known by finance, partially known by sales, and completely unknown by the engineering team: lost gross margin on units not sold in the delay window; lost revenue from customers who choose a competitor; declining average selling price as the product enters a more mature market; extended run-rate of development costs; lost market share that does not come back; late entry into the sales life-cycle; lost customers and markets foreclosed by the delay; and, for enabling technologies, loss of access to the customer's own product cycles. The exact number is not the point. An order-of-magnitude number shared with the team outperforms a precise number hidden in a finance deck every time.

In the literature

House and Price introduced break-even time in *Harvard Business Review* in 1991 as the central metric for product development economics [9] — the point at which cumulative profit equals cumulative investment. Reinertsen extended the argument with the cost-of-delay framework in *The Principles of Product Development Flow* [6], reporting that roughly 85% of product managers do not know their cost of delay and that intuitive estimates from the same team vary by a factor of fifty. The lateralworks practice operationalizes both: calculate the number, then share it down to the contributor level so the economic argument is available at the point where daily decisions get made.

From the field

GlobalFoundries Fab8 operated at roughly $5 million per day of operational value [13]. At that rate, the two-week schedule pull-in lateralworks helped deliver translated to roughly $70 million of value — a return on a multi-year consulting engagement that compounds the case for the practice. The same logic scales down: on a $200 million consumer product program at $1.6M/day, every Tuesday meeting that fails to make a decision is a measurable economic loss, not a calendar artifact. Cost-of-delayAI is the lateralworks software function that surfaces this number at the point of decision [28].

> **Leadership test:** a simple test for whether cost of delay has reached the contributor level is to ask three randomly chosen engineers on the program what one day of schedule slip costs the company. If none of them can answer within an order of magnitude, the practice is not installed. The number does not have to be exact — it has to be visible.

03

Installation
Putting it together

The ten practices compound. Installed in isolation, each one produces marginal improvement; installed together, they produce the time-to-market compression that lateralworks engagements are known for. Order of installation matters, because the practices depend on each other. The weekly refresh has nothing to refresh without a macro plan. The macro plan has no team to build it without a heavyweight core. The core team has no air to breathe without a host willing to dedicate people to it. The sequence below respects those dependencies.

No two engagements run the same playbook. Some practices can be parallelized; some organizations start further up the maturity curve than others; some have already installed pieces of FTTM under a different name. The phasing below is the median across the engagement base — the shape most engagements end up resembling, not a rigid template.

Installation

The lateralworks sequence

Phasing is by elapsed weeks from program start, not by calendar week. Activities overlap deliberately — the cost-of-delay number is computed in parallel with core team formation, the VOC schedule is built while the macro plan is being scrubbed. The point is that no practice gets installed in isolation, and the team feels the compounding effect within the first quarter rather than the first year.

Phase	Activity	Outcome
Week 1–2	**Training and mindset**	Leadership and core team attend FTTM foundation training. Speed is named as the primary asset; the language of the framework becomes shared vocabulary.
Week 1–2	**Cost of delay**	Finance and program leadership calculate the daily $ number and commit to sharing it with the full team. This is the single most leveraged early move.
Week 2–4	**Core team & freedom levels**	Heavyweight roles assigned. Core members moved to 100% dedication. Freedom level named explicitly for each major decision category in the team charter.
Week 3–5	**Macro plan workshop**	One- or two-day offsite produces the 50-task macro plan, four to six integration points, and the initial gap analysis. The gap, not the plan, is the deliverable.
Week 5–8	**VOC schedule built out**	Customer engagement mapped onto the macro plan across every phase from gestation to support. Lead customers and reference accounts identified by name.
Week 5–8	**Macro plan scrubs**	Two to three one-hour sessions per week refine the macro plan, test assumptions, align functional detail into the cross-functional structure.
Week 8+	**Weekly refresh installed**	Standing meeting at the same day and time every week. Coach attends the first six to eight sessions, then steps back. The cadence never moves.
Month 3+	**Challenge workshop on demand**	When the team encounters a binding constraint, a Challenge workshop surfaces and questions the assumptions. Used when it hurts, not prophylactically.
Month 6+	**ML2 certification**	Coach assesses planning maturity. Teams that reach ML2 run the refresh independently. Before-the-fact leadership habits are reinforced in the look-forward portion of each refresh.

Installation

What to measure

Schedule compression is the outcome. It is not available as a leading indicator. The metrics that matter during installation are behavioral, not schedule-based. When the four leading indicators are healthy, schedule compression follows. When one or more is broken, the team plateaus regardless of how much schedule pressure the host applies.

Refresh attendance. Does every core team member attend every week? Anything below 90% means the refresh has been demoted to "another status meeting" and the compounding effect is already breaking down.

Refresh duration. A mature refresh converges toward 30–45 minutes. Two-hour refreshes mean the team is using the refresh as the primary working session rather than the integration point. Five-minute refreshes mean nobody is being honest.

Critical-path trend. Is the end date moving or holding across consecutive refreshes? The wigglechart is the visual instrument. A line that has not moved in eight weeks signals a structural gap; a line trending down signals the team is winning back time.

Freedom-level utilization. Are decisions being made at the named level or escalating above it? Audit by sampling: pick three decisions made in the last two weeks and ask which freedom level was used. Drift toward escalation predicts the next plateau.

Beyond the ten

The ten practices in this paper are the minimum viable FTTM operating system. The full 2018.002 catalogue contains twenty-one additional practices worth installing once the core ten are stable [3]. The high-leverage additions, in order of pay-out: fuzzy-front-end management (practice 2.1), which stops programs from starting before they are ready; kill-projects-early discipline (2.5), the single largest portfolio-speed lever most organizations have not pulled; fail-fast learning cycles (3.7), which replaces do-it-right-the-first-time thinking with incremental convergence; and project-based performance measurement (2.9), which aligns individual incentives with program outcomes rather than functional outputs [22]. These are commonly addressed in later phases of an engagement.

Final word. The practices in this paper and the twenty-one more in the full catalogue are not secret. They appear in various forms across the agile, lean, and program management literatures. What separates the teams that achieve fast time to market from the teams that do not is installation discipline. The practices are simple; installing them against the organizational drag that every company produces is hard. Speed is a habit, protected one Tuesday morning at a time.

Self-diagnostic

Normal vs best at a glance

The ten practices contrast sharply with how a normal team operates. The table below is the contrast on a single page — useful as a self-diagnostic for teams that want to identify which practices to install first, and as a reference for leaders trying to explain the FTTM mindset in one view.

#	Practice	Normal team	Best (FTTM) team
1	**Cross-functional core team**	Matrixed members spread across 3–5 programs at 20–30% attention.	5–7 heavyweights, 100% dedicated, empowered as roles not functions.
2	**VOC scheduled end-to-end**	One customer workshop during requirements; quiet until beta.	Customer engagement scheduled across every phase of the TTM cycle.
3	**Freedom Levels 1–2**	Team defaults to Level 3–5; permission requested for most decisions.	Freedom level named explicitly for each decision category; team at 1–2.
4	**Start with a macro plan**	Detailed bottom-up schedule built function-by-function over weeks.	Top-down 50-task macro plan built by the core team in a 1–2 day offsite.
5	**Run the weekly refresh**	Monthly or quarterly status reports; schedule updated on demand.	Same day, same time, every week; missed only for force majeure.
6	**Challenge the assumptions**	Assumptions unspoken until the gap forces the conversation.	Structured Challenge workshop surfaces assumptions before crisis.
7	**Provisioning host**	Host interrupts, redirects resources, adds gates, starves projects.	Host provisions resources on time; removes interrupts by design.
8	**Measure planning maturity**	Schedule health not measured; team plateaus at ML0 or ML1 indefinitely.	ML0–ML3 assessed in an afternoon; target ML2, ambition ML3.
9	**Lead before the fact**	Respond to problems after they arrive; second critical path invisible.	Act on weak signals; ask the 4 questions every week; pull in continuously.
10	**Cost of delay at contributor level**	Daily cost known only to finance; engineers see overhead, not economics.	Every engineer knows the daily $; "pay to save a day" is an obvious trade.

Every left-column entry is a pattern lateralworks has observed across hundreds of engagements. Every right-column entry is a behavior observed on programs that consistently delivered fast. The gap between the two is not intelligence, budget, or technology — it is practice. The teams in the right column installed these behaviors one at a time, starting from a version of the left.

A

Sources

References & further reading

The lateralworks FTTM framework is grounded in thirty-six years of original field research across 200+ technology programs [1, 2]. Where the framework converges with published academic and industry literature, this paper cites both — the lateralworks source for the operationalized practice, the external source for the underlying idea. Readers familiar with Wheelwright and Clark, Reinertsen, Oncken and Wass, or House and Price will recognize the foundations; readers new to the literature can use the references list as a reading path.

Entries 1–3 cite the lateralworks methodology base. Entries 4–10 cite the core external literature on heavyweight teams, delegation, cost of delay, voice of the customer, and weekly cadence. Entries 11–13 are lateralworks engagement examples drawn from lateralworks.com. Entries 14–28 are supporting external sources cited in specific practice sections.

Sources

References & further reading

[1] lateralworks. *The FTTM framework: thirty-six years of original research.* https://www.lateralworks.com/methodology. Accessed May 2026.

[2] lateralworks. *About lateralworks: founded 1988, Silicon Valley.* https://www.lateralworks.com/about. Accessed May 2026.

[3] lateralworks. *FTTM New Product Development Best Practices, Revision 2018.002.* Internal methodology catalogue. 101 pages; 31 practices organized across Mindset (6), Host (10), and Team (15).

[4] Oncken, W., Jr., and Wass, D. L. “Management time: who’s got the monkey?” *Harvard Business Review*, November–December 1974. Reissued as an HBR Classic with commentary by Stephen R. Covey, November–December 1999. https://hbr.org/1999/11/management-time-whos-got-the-monkey.

[5] Clark, K. B., and Wheelwright, S. C. “Organizing and leading ‘heavyweight’ development teams.” *California Management Review*, volume 34, number 3, Spring 1992, pp. 9–28. https://cmr.berkeley.edu/1992/05/34-3-organizing-and-leading-heavyweight-development-teams/.

[6] Reinertsen, D. G. *The Principles of Product Development Flow: Second Generation Lean Product Development.* Celeritas Publishing, 2009. ISBN 978-1-935401-00-1.

[7] Hamel, G., and Zanini, M. “Excess management is costing the U.S. $3 trillion per year.” *Harvard Business Review*, September 5, 2016. https://hbr.org/2016/09/excess-management-is-costing-the-us-3-trillion-per-year.

[8] Hauser, J. R., and Clausing, D. “The house of quality.” *Harvard Business Review*, May–June 1988, pp. 63–73.

[9] House, C. H., and Price, R. L. “The return map: tracking product teams.” *Harvard Business Review*, January–February 1991, pp. 92–101.

[10] McChesney, C., Covey, S., and Huling, J. *The 4 Disciplines of Execution: Achieving Your Wildly Important Goals.* FranklinCovey / Free Press, 2012.

[11] lateralworks. *Sony PlayStation: the 500K ASIC.* Case study, 23 pages. https://www.lateralworks.com/cases/sony-playstation-500k-asic.pdf.

[12] lateralworks. *Results: the methodology that built the iPod.* Tony Fadell and the Philips Velo program. https://www.lateralworks.com/results. Accessed May 2026.

[13] lateralworks. *Results: GlobalFoundries Fab8.* $5M/day saved; first silicon two weeks ahead of schedule on a $7 billion greenfield semiconductor fab. https://www.lateralworks.com/results. Accessed May 2026.

[14] Brown, T. *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation.* HarperBusiness, 2009.

[15] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Seventh edition, 2021. Rolling-wave planning is documented in Section 2.6.

[16] Cohn, M. *Agile Estimating and Planning.* Prentice Hall, 2005. Chapter 13 covers progressive elaboration and rolling-wave planning in an agile context.

[17] lateralworks. “The weekly schedule refresh.” May 8, 2026. https://www.lateralworks.com/ideas/the-weekly-schedule-refresh.

[18] de Bono, E. *Lateral Thinking: Creativity Step by Step.* Harper & Row, 1970. Foundational work on the systematic challenge of assumptions.

[19] Goldratt, E. M. *The Goal: A Process of Ongoing Improvement.* North River Press, 1984. Theory of Constraints applied to manufacturing; the methodology has since been extended to project management in *Critical Chain* (1997).

[20] Senge, P. M. *The Fifth Discipline: The Art and Practice of the Learning Organization.* Doubleday, 1990. Mental models as organizational constraint.

[21] Project Management Institute. *Pulse of the Profession 2023: Power Skills, Redefining Project Success.* PMI annual benchmark report. https://www.pmi.org/learning/library/pulse-of-the-profession-2023.

[22] lateralworks. “Portfolio prioritization and starts control.” May 11, 2026. https://www.lateralworks.com/ideas/portfolio-starts-control-briefing.

[23] Saaty, T. L. “Decision making with the analytic hierarchy process.” *International Journal of Services Sciences*, volume 1, number 1, 2008, pp. 83–98.

[24] Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V. “Capability Maturity Model, Version 1.1.” *IEEE Software*, volume 10, number 4, July 1993, pp. 18–27. Foundational paper from the Software Engineering Institute at Carnegie Mellon.

[25] lateralworks. “The critical path you’re not watching.” May 7, 2026. https://www.lateralworks.com/ideas/the-critical-path-youre-not-watching.

[26] Klein, G. *Sources of Power: How People Make Decisions.* MIT Press, 1998. Recognition-primed decision making in naturalistic settings.

[27] Weick, K. E., and Sutcliffe, K. M. *Managing the Unexpected: Sustained Performance in a Complex World.* Third edition, Jossey-Bass, 2015. The five principles of high-reliability organizing.

[28] lateralworks. *fastProjectAI software suite: fastProjectAI, fastDecisionAI, and Cost-of-delayAI.* https://www.lateralworks.com/software. Accessed May 2026.