Whitepaper

The FTTM core team

Roles, not functions: how heavyweight, lateralized teams become the single greatest contributor to development speed

Team series.

A deep dive into the structure fast development programs are built on: a small, dedicated, role-based core team led by a heavyweight integrator, empowered to own the outcome end to end. This is the lateralization of programs.

Prepared by
lateralworks
Silicon Valley, founded 1988

Subject
FTTM core team
Methodology paper

Online
lateralworks.com
Team series

Table of contents

The FTTM core team

Core thesis. A cross-functional core team is not a group of functions assigned to a project. The FTTM core team carries three critical roles at its center — a chief engineer, a product leader who is the voice of the customer, and a system architect — held together by a heavyweight technical program manager who acts as the integrator. Those roles pull the whole program sideways, across the functional walls, from first definition to final delivery. That lateral pull is what produces speed. Of everything a company can change to get to market faster, the team is the largest single lever.

Overview

Abstract

Speed in product development is decided by how a program is organized, not by how hard people work inside it. Across four decades of fast-time-to-market research in Silicon Valley, one variable separates the programs that hit their market window from the ones that miss it: the structure and empowerment of the team at the center. Tools, schedules, and process discipline all matter, but they operate downstream of the team. Get the team right and the rest becomes tractable. Get it wrong and no amount of tooling recovers the lost time.

This paper is a deep dive into one structure: the FTTM core team. It is easy to confuse with the ordinary cross-functional team, because the words are the same. The difference is not the label. A normal cross-functional team is a set of functions — hardware, software, operations, quality — each sending a representative to defend its interests. An FTTM core team is a small set of program roles that subordinate their functions to a shared outcome. Three of those roles are decisive: the chief engineer, who owns the technical result; the product leader, who owns the voice of the customer; and the system architect, who owns the interfaces and the trade-offs across the whole system. A heavyweight technical program manager — the integrator — leads them and links the pieces into one whole.

We place this structure inside the team typology first described by Steven Wheelwright and Kim Clark [1, 2]: functional, lightweight, heavyweight, and autonomous. Each step transfers more power from the functional hierarchy to the team, and each step buys more speed and more accountability. The FTTM core team lives at the heavyweight and autonomous end of that spectrum. Clark and Fujimoto documented the same pattern in the world auto industry, where heavyweight product managers produced the highest development performance [3]; Donald Reinertsen reached a parallel conclusion from the economics of flow [4, 5]. Our field data converges with theirs, and this paper shows where.

We call the underlying move the lateralization of programs. Functional organizations bias people to think vertically, up and down their silo. Fast programs bias people to think laterally, across the silos, toward the integrated result a customer actually receives. Roles rather than functions is the mechanism that produces the sideways bias. Empowerment — the freedom to decide and act — is what lets the laterally-organized team move at speed. The paper closes on the practical payoff: the core team is the single greatest contributor to fast-time-to-market results, and building it correctly is the highest-leverage decision a program makes.

01

Thesis

The team is the accelerator

Ask a room of executives what makes a development program fast and the answers arrive quickly: a better tool, a tighter schedule, more discipline, more people. Each helps. None is the answer. The answer is the team — its structure, its leadership, and the power it is given to act.

This is not a soft claim about culture. It is the most consistent finding in four decades of fast-time-to-market fieldwork, and it holds across semiconductors, systems, consumer electronics, and greenfield factory start-ups. The team at the center of a program is the single largest lever on the date it ships.

Section 01

The team is the accelerator

Fast-time-to-market work rests on a simple model with three parts: the host, the environment, and the team. The host is the corporation that funds and surrounds the program. The environment is the set of practices and structures the program runs inside — how it plans, decides, and prioritizes. The team is the small group that actually converts a concept into a product. All three matter, but they are not equal in leverage. The host sets the conditions; the environment shapes the flow; the team delivers the result. When we score programs against the FTTM best practices [6], the team practices move the outcome more than any other single category — a pattern that holds across four decades of engagement data [7].

The reason is structural, not motivational. A product has to move laterally through an organization to reach a customer: from market need to definition, through hardware and software and test, into manufacturing, and out to the field. Every functional wall it crosses is a place it can stall. The team is the one entity chartered to carry the product across all of those walls without dropping it. If the team is weak — part-time, unempowered, organized as a set of functional representatives — the product loses time at every crossing. If the team is strong — dedicated, role-based, empowered to decide — the crossings disappear, because the people who own the interfaces are already in the same room.

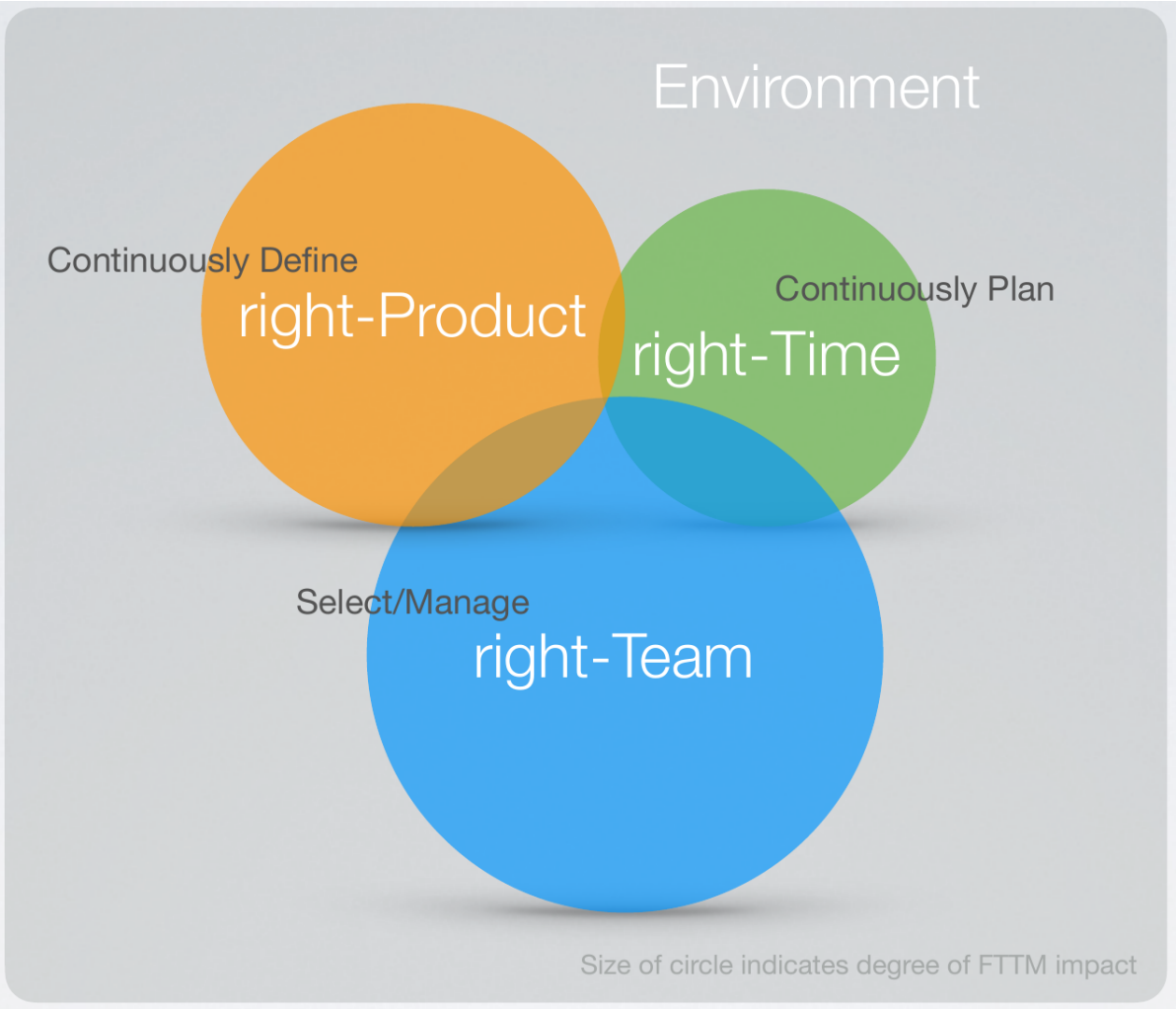

Figure 1. The FTTM system delivers the right product, at the right time, with the right team. The size of each circle indicates its degree of impact on the outcome. Selecting and empowering the right team is the condition that makes the other two achievable.

Three rights, one team

The FTTM system is often summarized as three rights: the right product, at the right time, built by the right team. The first two are outcomes. The third is the mechanism that produces them. A team that is close to the customer defines the right product; a team that owns its schedule and can act without waiting hits the right time. Remove the team from the center and the two outcomes lose the only agent that can reconcile them — which is exactly what happens on programs where marketing writes a requirements document, throws it to engineering, and the two never sit at the same table again.

This paper argues the case in eight steps. It begins with the confusion that has made the word "team" nearly meaningless (Section 02), places the FTTM core team inside the established typology of team structures (Section 03), and then isolates what actually makes it different: roles instead of functions (Section 04). It examines the three critical roles at the core and the integrator who leads them (Sections 05 and 06), the empowerment that lets them move (Section 07), and the direct line from this structure to development speed (Section 08).

The thesis

The greatest single lever

The greatest single contributor to fast time to market is not a tool or a schedule. It is the team.

lateralworks

FTTM best-practice research, 200+ programs since 1988

02

Failure mode

How everything became a team

When more than two people gather to work on something today, they call it a team. There are executive teams, launch teams, process-improvement teams, and development teams. Most of them are reconfigured functional silos wearing a new label.

To see why the FTTM core team is different, start with what a normal cross-functional team actually is — and why, despite good intentions and hard work, it defaults to slow.

Section 02

How everything became a "team"

The history is worth a moment. Engineering department heads used to meet to synchronize work moving through their functions. Then the word "team" arrived, and every gathering of more than two people became one. Then "cross-functional" arrived, and a cross-functional team came to mean inviting all the department heads or technical leads to a project meeting. The same people ended up on every cross-functional team in the company, which turned that small group into the information bottleneck for the entire organization [8].

Watch a normal program form and the pattern repeats. A project is announced at a kick-off. A team is assembled from people who are already over-allocated — the best contributors are on the most teams, because everyone wants them, so the better you are the more ineffective you become. The group is told the goals are lofty, hiring is frozen, and they will have to reach deep and deliver one more success. Once assigned, the team quietly reproduces the functional hierarchy it came from: a small copy of the departmental structure, now called a team, layered on top of the ten projects everyone already carries. Because leaderless teams rarely work, someone inserts a program manager who is given no control of schedule, budget, or people, and is then expected to deliver on time.

The engineering core team

The most common failure mode is the core team that is really an engineering team. The logic sounds reasonable: engineering is the hardest phase, so the core team is built from engineering disciplines — a hardware member, a software member, test and verification, a few design leads, and a cluster of technical subject-matter experts. Marketing writes the requirements document and hands it off. Manufacturing is left out because its role comes later. The team swells to twenty or thirty people, because no one wants to be excluded from technical decisions, and its meetings degrade into long, poorly structured debates where the strongest expert wins the argument.

Two costs follow. First, the requirements conflicts are never resolved, because each side keeps plausible deniability — marketing can blame engineering for building the wrong thing, engineering can blame marketing for describing it poorly, and what ships is usually what engineering could build rather than what the customer wanted. Second, manufacturability is designed in late or not at all, because operations was never in the room during the gestation phase. The emphasis sits on the function and its current problems, not on the product and the overall schedule. The program manager keeps a record and occasionally raises a flag, but few pay attention, because the schedule "was unrealistic anyway."

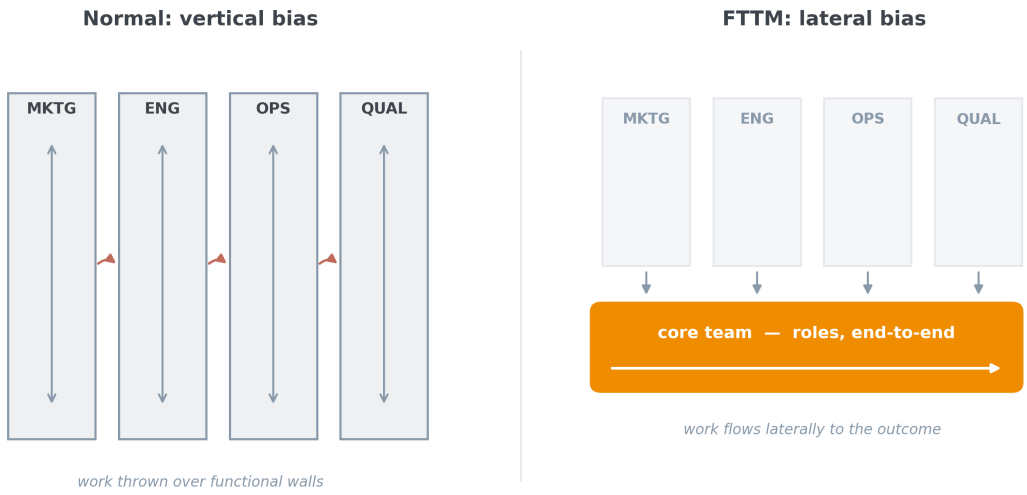

Figure 2. Vertical bias versus lateral bias. In a normal organization, functional roles push each person to think up and down their silo, and work is thrown over the walls between functions. The FTTM core team gives people program roles that pull attention sideways, across the functions, toward the integrated result.

This is the deeper problem, and it has a name: compartmentalized thinking [9]. Each person focuses on a small part and no one holds the complete system. It is not a hypothetical risk. Boeing's 737 MAX review found employees who described a compartmentalized approach, each focused on a small part of the plane, which left them without a complete view of a critical and ultimately dangerous system [10]. The missing role was a system architect overseeing the change and its implications for the whole. Functional teaming manufactures exactly this blind spot. Lateral teaming is the correction.

03

Typology

Four structures, rising power

The move from functional to lateral is not binary. It runs along a well-documented spectrum of team structures, each transferring more power from the hierarchy to the team, and each buying more speed and more accountability in return.

Steven Wheelwright and Kim Clark named four types: functional, lightweight, heavyweight, and autonomous [1, 2]. We have worked with all four, and with many hybrids between them, and we have added two axes to their model — speed and accountability [11].

Section 03

Four structures, rising power

The four structures are best read as a continuum of empowerment. The question at every step is the same: how much power — over budget, over resources, over decisions, over people — has the functional hierarchy been willing to transfer laterally to the team? The more that transfers, the faster the program can move and the more the team owns the result.

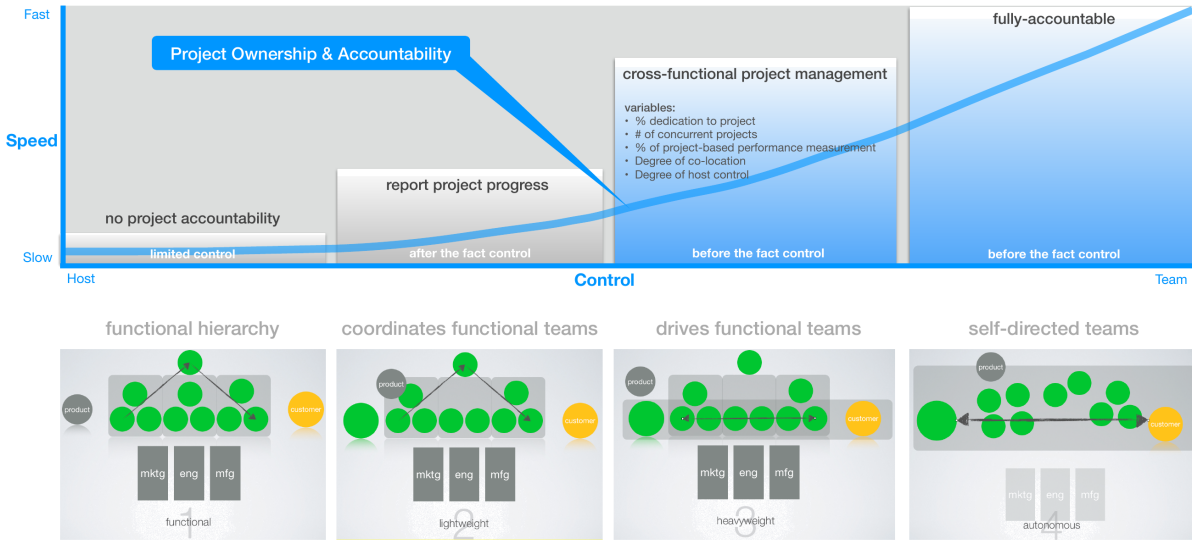

Figure 3. Project ownership and accountability across the four team structures. Top: as control shifts from the host (left) to the team (right), the team moves from no project accountability under limited, after-the-fact control to full accountability under before-the-fact control — and speed rises with it. Bottom: the same four structures — functional hierarchy, lightweight (coordinates functional teams), heavyweight (drives functional teams), and self-directed autonomous. Power, and with it speed, moves toward the team as you read to the right.

Functional hierarchy

The classic structure. Information moves up and down the hierarchy, each function sitting in a silo with high walls between them. The development effort has to navigate between the silos, and only a few people ever touch the market or the customer, so customer information arrives filtered. Power is concentrated in the hierarchy, the organization is inwardly focused, and few people are accountable for the product — most are accountable only to their function. Development is slow, and quality and margin suffer.

Lightweight team

The traditional response to functional inefficiency: get some lateral coordination without surrendering vertical power. A project manager is appointed, but the title outruns the authority. Budget still lives in the functions, so information still flows up and down to move sideways. These managers are really coordinators — given responsibility for delivery but little authority to execute, spending their days negotiating and pleading for cooperation from functional managers who control the resources. It is not unusual for one person to sit on ten such teams and for a manager to run five at once. Accountability to the product rises slightly, but the program stays slow, because the under-capacity problem is simply masked by a lot of people trying very hard.

Heavyweight team

The game changer. Here the hierarchy shares real power with the lateral team. The program manager shares budget and resource decisions with the functions and has meaningful input — often half or more — into the performance reviews of team members. Decisions get faster because information moves laterally and stays inside the team, which now has the authority to set technical direction. The team has a direct line to the customer and the market, which creates customer pull and accelerates development. It costs more, because people are more dedicated to a single program, but that added R&D; expense is repaid many times over in cycle-time improvement and in products that actually meet customer needs — and that command a higher price at introduction precisely because they arrived on time.

This is the structure Clark and Fujimoto found at the top of the world auto industry. Studying development performance across global automakers, they concluded that the highest design quality and the shortest lead times came from heavyweight product managers who carried the product from concept to market and integrated the functions around it [3]. Toyota's equivalent — the shusa, or chief engineer — is the archetype. The FTTM core team is a heavyweight team by this definition, built for technology programs rather than automobiles.

Autonomous team

The most advanced stage. These teams are fully dedicated and, in most cases, physically separate — self-contained and led by senior general managers. They have every function of a small business, like a start-up, with a direct customer line and continuous voice-of-the-customer through the development cycle. Functions blur; each person knows a great deal about everyone else's domain, and marketing and engineering become nearly indistinguishable. The hierarchy's job inverts: it exists to provision and protect the team, not to control it. These teams are very fast and tend to have the highest hit rate on products that ship on time and exceed customer needs.

Donald Reinertsen reaches a parallel conclusion from a different direction. Where Wheelwright and Clark argue from organizational structure, Reinertsen argues from the economics of flow: development is dominated by uncertainty and queues, and speed comes from decentralizing control to the people closest to the work and quantifying the cost of delay so they can act on it [4, 5]. Centralized functional hierarchies are, in his terms, optimized for capacity utilization rather than flow — which is why they feel busy and deliver late. The empowerment axis we add to Wheelwright and Clark is the same variable Reinertsen calls decentralized control.

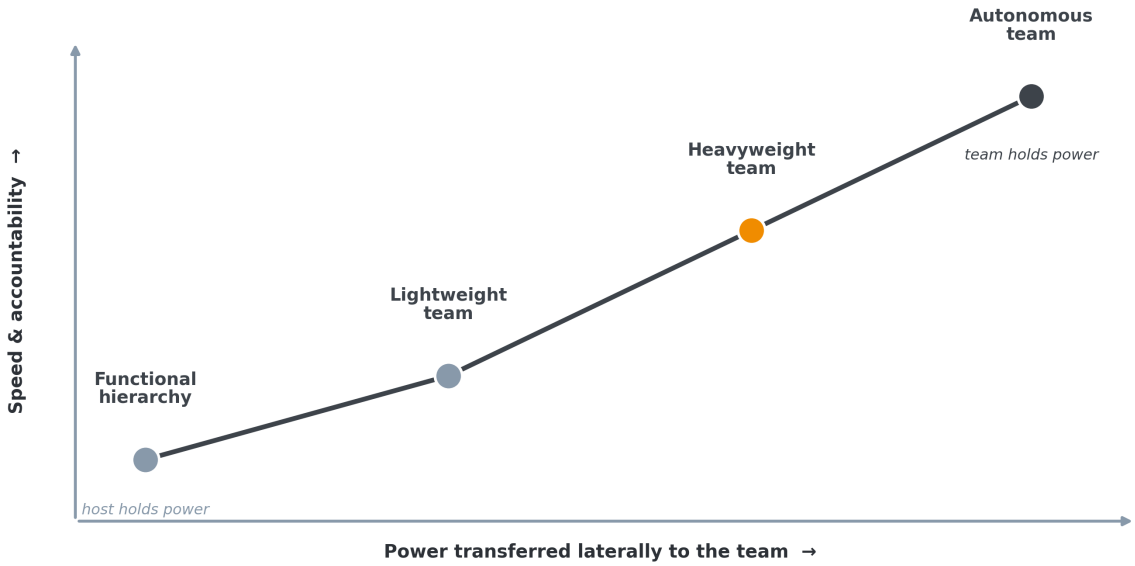

Figure 4. The empowerment continuum, distilled. The same progression as Figure 3, read as a single line: each structure transfers more power laterally to the team, and speed and accountability rise together as it does. The FTTM core team sits at the heavyweight-to-autonomous end. Most organizations rate themselves further right than they operate; the hierarchy tends to pull them back left.

One caution from the field: almost everyone rates their own structure further to the right than it really is. The desire to be heavyweight is real, but the functional hierarchy exerts a constant pull back toward lightweight, because lightweight is comfortable and cedes no power. Some organizations stand up an autonomous team once, as an event, and then quietly revert. Moving right is a deliberate, sustained choice — and if speed is critical, it is not optional.

04

Difference

Roles, not functions

Everything so far has described a spectrum. Now the specific claim: what makes the FTTM core team different from an ordinary heavyweight team, and radically different from a functional one, is a single substitution. It organizes around roles, not functions.

That one change is the mechanism of lateralization. It is small to state and hard to do, and it is the difference between a team that owns a product and a set of functions that defend their pieces of it.

Section 04

Roles, not functions

A core team is typically five to ten key people responsible for developing a new product. In a normal organization they are organized around functional responsibility — the hardware person, the software person, the operations person — and each carries a vertical bias toward their specialty. That bias simply reproduces the functional silos one level down, at the core-team table, and puts the emphasis on the parts rather than the whole. The best product teams do the opposite. They organize around program roles — chief engineer, system architect, system integrator, customer integrator — and each member subordinates their functional role to their program role. The functional hat still exists; it is simply no longer dominant. Those program roles create a lateral focus from the beginning of the project to the end.

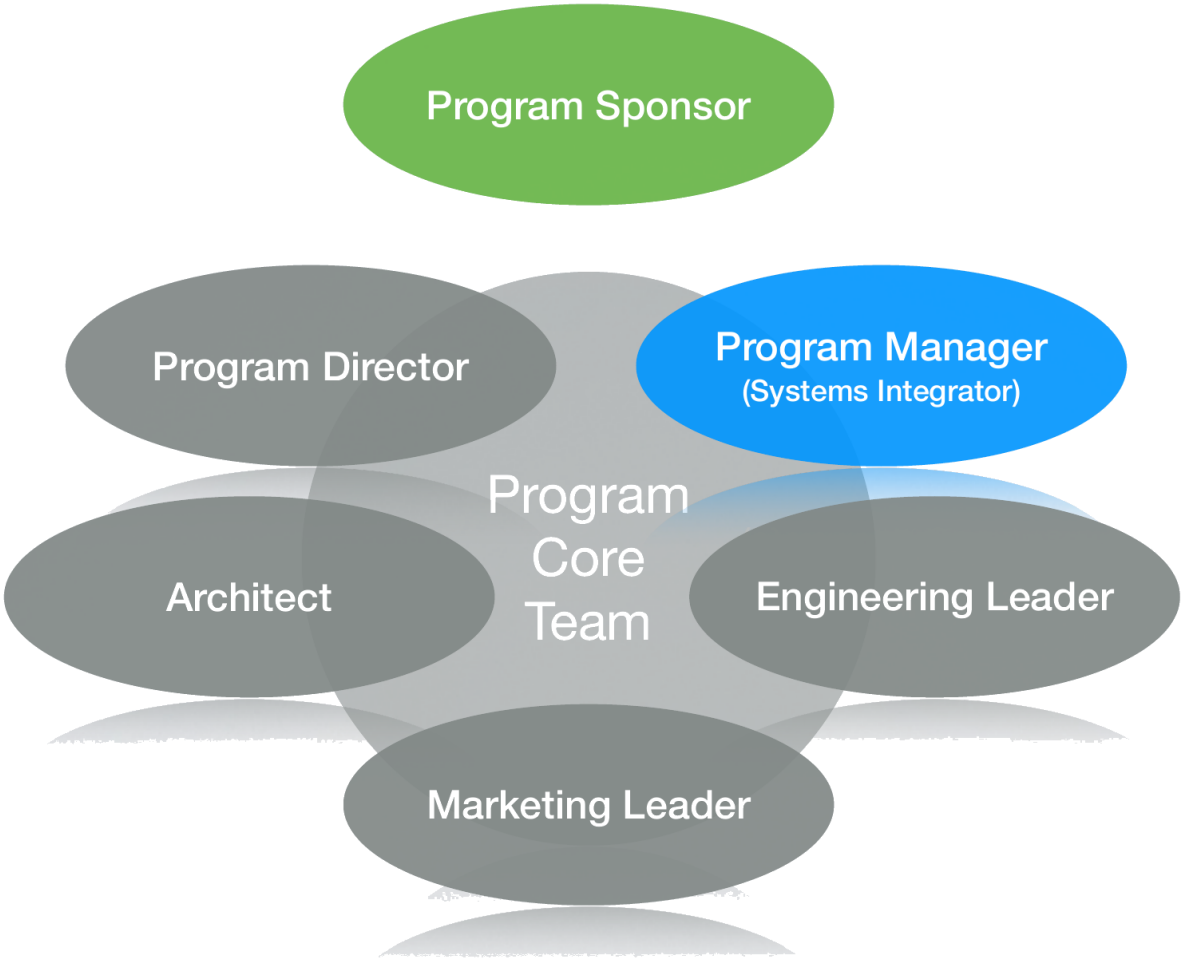

Figure 5. An FTTM core team organized around roles. Fewer than six people sit at the center — a program director, a program manager acting as system integrator, an architect, an engineering leader, and a marketing leader as voice of the customer — with a program sponsor providing clout from outside. Each holds a program role first and a functional role second.

The size is deliberate. An FTTM core team is fewer than six people, not the twenty-five that accumulate when no one wants to be left out. It is not a miniature of the functional hierarchy; it is the management team of a small business. This is the same team you would assemble to do a start-up — and start-ups almost

never organize around functions at the beginning, because the hierarchy is too slow. Large companies reproduce the known structure by reflex. Fast teams refuse to.

The difference, line by line

The contrast is easiest to see laid out point by point. The table below compares a normal core team and an FTTM core team across thirteen dimensions — how the leader is positioned, how the work is structured, how members are assigned and measured, who owns the outcome, and how the whole thing is held together. No single row is decisive; the difference is the accumulation. Notice how the rows reinforce one another. Part-time membership (row 5) undercuts ownership (row 9); functional reporting and compensation (rows 10 and 11) pull members back toward their silos regardless of what the org chart says; and a coordinator with no authority (row 1) cannot hold an integrated system together (row 2). Change one row in isolation and the others drag it back. The FTTM column only works as a set.

	Normal core team	FTTM core team
1	Project leader is a part-time coordinator, works on multiple projects, has little power; control of the project rests within each function's hierarchy.	Project leader is full-time, works only on this project, and has decision-making authority for technical and managerial topics — the project is the job.
2	The project runs as a series of independent work-streams, each focused on its own deliverables; the leader attempts to manage the interfaces and the team is largely uninvolved.	The project runs as an integrated system, with core-team members focused on the integration points and working together to achieve shared outcomes.
3	Roles on the team are functional.	Each member has a functional role, but three carry program roles: chief engineer, architect, and systems engineer / integrator.
4	Team members have a function-bias, protect their function, and their functional requirements always trump the project.	Team members have a project-bias and focus on project outcomes. It is a mission, not a job.
5	Team members are part-time and work on multiple projects.	Functions dedicate members 100% to the project; they have one job, the project.
6	Functions provide a representative to sit on the team, yet that person lacks decision authority and technical expertise.	Team members have decision authority, can speak for their function, and are technical subject-matter experts.
7	Much time is spent avoiding blame and covering for the function if things go wrong.	Focus is forward-looking — how to accelerate, learning cycles, incremental improvement. The team shares failures and uses them to learn.
8	Team members are a conduit of information back to the functional hierarchy, which becomes their main purpose for being on the team.	Members report back to their functions with what is needed and when, and the critical interfaces — not a channel functions use to "spy" on the team.
9	The team does not own the overall outcome or feel responsible for it — only its functional deliverables. The project leader is the only owner.	The team owns the outcome, is accountable for its success or failure, and is rewarded together — from early definition to final delivery.

	Normal core team	FTTM core team
10	Team members report to their functional managers, not to the core team or project leader.	Team members report to the team leader, sometimes with a dotted line to their function. Most effective when the member is assigned to the team leader.
11	Compensation is based on goals defined within the function, not tied to project success or failure.	Compensation is based on the project hitting its stated targets for product performance and schedule.
12	Core-team members are virtual or geographically distributed.	Core-team members are co-located. When that is not possible, they meet in person regularly — a number of days each month.
13	The project is something people participate in; the "real work" happens in each person's function, where it is also managed by the functional hierarchy.	The project is like a small business whose members are equity owners. Their main focus is the project. This is all they work on.

Read the right-hand column as a whole and a different entity comes into view. It is not a committee of departments. It is a small business whose founders own equity in the outcome. When people own the result and believe they can control it, they work faster and more effectively, because they can see the consequence of their own contribution. That ownership is what the role-based structure unlocks and the function-based structure suppresses.

Roles, not functions. That one substitution lateralizes the entire program.

05

The core
The triad and the integrator

At the center of the core team sits a smaller core still: three roles that own the two outcomes that matter most, and one role that binds them together. We call it the triad — the core of the core team.

These are not a group of functions. They are the chief engineer, the product leader, and the system architect, led and integrated by a heavyweight technical program manager. Get these four right, and add names afterward, and most of the team is already working.

Section 05

The triad and the integrator

The triad owns delivery of the right product at the right time. The integrator manages the program and its interfaces and owns right-time performance. The technical leader is the final technical decision-maker and owns product performance. The product leader is the voice of the customer, helps make the trade-offs, and owns the right product — and in some programs is also the system architect. Around this triad the rest of the core team assembles, but the triad is the load-bearing structure. When we see a program fail technically or slip badly, one of these three seats is usually empty or occupied by someone without the authority to fill it.

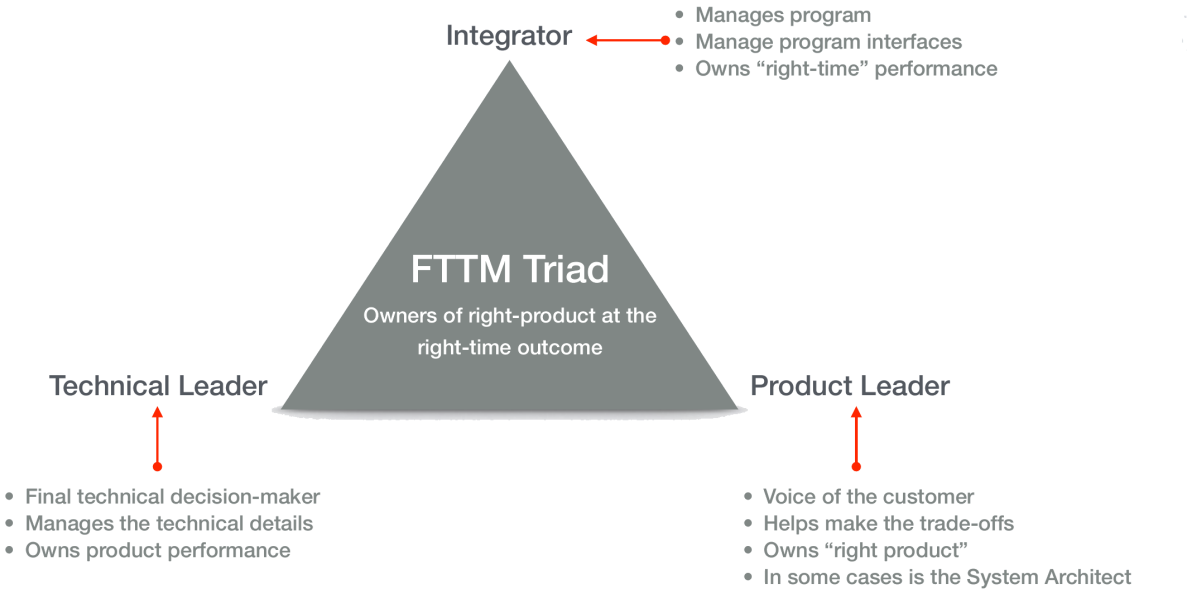

Figure 6. The FTTM triad — the core of the core team. The integrator manages the program and owns right-time performance. The technical leader is the final technical decision-maker and owns product performance. The product leader is the voice of the customer and owns the right product. Together they own the right-product-at-the-right-time outcome.

The chief engineer — the technical leader

The chief engineer is the single person who understands all of the subsystems and how they interact. This is a generalist who knows how to work with the specialist experts — the technical functional leads — and who listens to the debate, evaluates the data, and then makes the final technical call. Two properties define the role. The chief engineer has ultimate responsibility for the success or failure of the total system, and the chief engineer cannot be overruled on technical matters. Without that second property the role collapses back into a committee. The chief engineer drives the design process, integrates the test plans against the requirements, and acts as the first customer of manufacturing for the initial build. This is the technology-program analogue of Toyota's chief engineer, the role Clark and Fujimoto identified as the engine of heavyweight development [3].

The product leader — the voice of the customer

The product leader is the customer's advocate inside the company and the team's link to the market. This person gets, understands, and continuously refreshes customer requirements, then converts them into product requirements and specifications, and prioritizes those specifications by customer value so the right product is what actually gets built. The role carries the cost-of-delay message to the team and the host, so the program keeps its sense of urgency. Crucially, the product leader understands the customer's business problem — what the customer is actually trying to solve — rather than a list of requested features, and reconciles what the market wants against what can be delivered inside the window. This is the seat that a functional core team leaves empty when marketing writes a document and walks away.

The system architect — owner of the interfaces

The architect owns the whole system and how its elements interface. The complexity in most systems lives at the interface points, and the architect owns those connections and sees the complete product end to end. Like the architect of a building, the system architect defines and manages the interfaces and makes the trade-offs — cost, technical, resource — needed to optimize the total system rather than sub-optimize its components. The architect may also wear a functional hat, but the program role dominates. One of the most common missing roles on failed technical programs is exactly this one: no one making trade-offs for the good of the whole, so decisions get made by functional bias and the product is quietly compromised. The Boeing 737 MAX case is the cautionary version — a critical system changed without a system architect overseeing the implications for the whole [10].

The integrator binds the three

The three owners need one person to link them into a whole and drive the program on the calendar. That is the integrator — the heavyweight technical program manager. The integrator manages the program, manages the interfaces between the pieces, and owns right-time performance. Think of the program director as the CEO of the project and the integrator as the COO. The integrator is the lateral linker: the person who continuously connects the work-streams into one integrated program, runs the tactical rhythm, provisions the team, and prevents or works around the interrupts that would otherwise stall it. The next section is about why this role, done right, is so rare — and so decisive.

The mindset

A small business

These are not a group of functions on a project. They are a small business and the team owns the outcome.

lateralworks

The integrated core team

06

Leadership

The heavyweight integrator

Are program managers worth having? For much of the industry the honest answer is often no — the role is filled by someone who is a competent manager or a competent engineer, rarely both, and the team experiences the result as overhead.

The integrator is a different animal. It is a heavyweight technical program manager: someone with the technical clout to lead the triad and the management skill to drive the program. This section is about what the role is, and why it is the hardest seat to fill.

Section 06

The heavyweight integrator

Across decades of technical programs, most program managers are either good managers or good engineers; perhaps one in ten are both. A marginal manager with an outdated grasp of the technology is the worst of both worlds, which is why technical teams so often describe their program manager as overhead or a time sink, and why some argue — with a grain of truth — that removing that person would speed the project up. Non-technical program managers become proxy managers for the lead engineers and designers who actually control the work. The bigger the company, the larger this overhead tax on speed [12].

The successes look different. The fastest programs we have worked on were led by an engineer who had learned to manage — someone with technical credibility who could also plan, run people, and operate at both altitudes: on the ground in the detail when needed, and at thirty-thousand feet on the macro plan when not. These leaders understand the market and the customer as well as they understand the core technology. They are rare, and they are worth their weight in gold. The role is not made of a certification; it is made of clout plus craft.

What the integrator actually does

Operationally, the integrator owns the program's macro plan of record and drives the program to it. The job is concrete: build and continuously refresh the master schedule; coordinate every participating team and cross-functional group; resolve resource contention between the milestone teams and the functions; run the tactical meetings that surface blocking issues and get them solved; and provision the team while shielding it from interrupts. The integrator drives development against a schedule that is refreshed weekly, not a schedule that is written once and admired. The title varies — program manager, systems integrator, program systems integration manager — but the function is constant: laterally link the pieces of the program into one integrated whole and hold it to the calendar.

This is the heavyweight product manager of the research literature, translated to technology. Wheelwright and Clark describe the heavyweight leader as someone given responsibility for the total effort and its overall success, who integrates the functions and brings the voice of the customer into the process, and who is senior enough to make it stick [2]. Clark and Fujimoto found that this concentration of responsibility in one credible, empowered leader was a primary driver of superior development performance [3]. The integrator is that role. The difference between a lightweight coordinator and a heavyweight integrator is not effort — both work hard — it is authority and technical standing.

07

Empowerment
Freedom to act

A well-designed team that cannot act is still slow. Structure sets up speed; empowerment releases it. The lack of freedom to act is a root cause of delay, apathy, and a general sense of helplessness — and it runs from the top of the hierarchy to the bottom.

Fast organizations push power down. We quantify how far down with a simple instrument adapted from Bill Oncken: the freedom scale [13].

Section 07

Freedom to act

There are five levels of freedom, and the lower the number, the faster decisions get made and the more ownership people feel. Level 5 is "wait until told" — a fully passive state, in theory reserved for trainees, though we have watched executive vice presidents live there. Level 4 is "ask what to do," marginally better and just as self-defeating, because the more you ask, the more you are told, and the faster you slide back to 5. Level 3, "recommend, then take action," moves a person out of trainee mode and gives the manager before-the-fact control while still encouraging action. Level 2, "act, but advise at once," is real-time management with during-the-fact control. Level 1, "act, routine reporting only," is the most advanced — and experienced professionals struggle to work below it. The more Level 1 behavior an organization pushes down, the more leverage it gets from the combined judgment of its people.

Figure 7. The freedom scale, adapted from Bill Oncken [13]. Levels 1 to 3 provide leverage and growth and are where fast cultures operate; Levels 4 and 5 are for trainees and slow environments. Fast teams take Level 1 and 2 — and the more they take it, the more empowerment they are granted.

One field observation captures the whole idea. On a very fast program, people began emails with the acronym UNIDIR — "unless otherwise directed, I intend to…" — and then acted [14]. That is default-to-action in place of default-to-delay-and-discuss. It only works in a culture that grants the requisite permission to fail and learn, because Level 1 is risky if you are punished for being wrong. The fastest cultures treat failure as a source of learning cycles, which is precisely what makes Level 1 safe enough to take. This is the same variable Reinertsen names decentralized control [5]: push authority to where the information is, and the system speeds up.

The steering arm and provisioning

Empowerment does not mean abandonment. FTTM core teams report to a steering arm — typically three senior executives representing marketing, engineering, and operations. Their role is strategic and deliberately light: they break ties on cross-functional issues the team gets stuck on, they provision the team with information and resources, and otherwise they interact on an exception basis. The team is left alone unless it asks for help or deviates from an agreed metric such as budget or schedule. If a program slips past its buffer, the team reports its corrective actions; if it is inside the parameters, it runs itself. One steering arm can oversee many programs at once precisely because it manages by exception rather than by presence.

The other half of the steering arm's job is to fight interrupts. Slow organizations default to delay because the work waits for the hierarchy to decide and act — they are quick to interrupt the team and slow to provision it. Fast hosts invert that: they provision quickly and protect the team's attention, because context-switching and interruption are among the largest hidden taxes on a development schedule. The integrator prevents interrupts where possible and works around them where not; the steering arm removes the organizational ones the team cannot reach.

08

Speed
The key to fast development

Every thread in this paper converges on one claim: the core team is the key to speed, and the greatest single contributor to fast-time-to-market results. Structure, roles, and empowerment are not ends in themselves. They exist to compress the calendar.

This closing section connects the team to the outcome — through the customer, through the economics of delay, and through a program where it was tested under real pressure.

Section 08

The key to fast development

Speed comes from removing the waits, and most waits are lateral. Work stalls where one function hands to another, where a decision needs an authority who is not in the room, where a requirement is ambiguous and no one owns the reconciliation. A role-based, empowered core team removes those waits by construction: the owners of the interfaces sit together, the decision authority is inside the team, and the voice of the customer is a permanent member rather than a document. That is why the team, and not the tool or the schedule, is the largest lever. Tools and schedules help a team that can already act; they cannot rescue a team that cannot.

Speed and the customer relationship

The product leader's role points at a second source of speed: the depth of the customer relationship. Programs that co-develop with tier-one customers — continuously refining requirements rather than freezing a specification and hoping — move faster, because they discover the wrong turns early instead of at launch. The relationship itself is a speed variable. Adversarial and purely transactional relationships are slow; cooperative, co-development partnerships are fast, because requirements converge in real time and the team never spends months building the wrong thing.

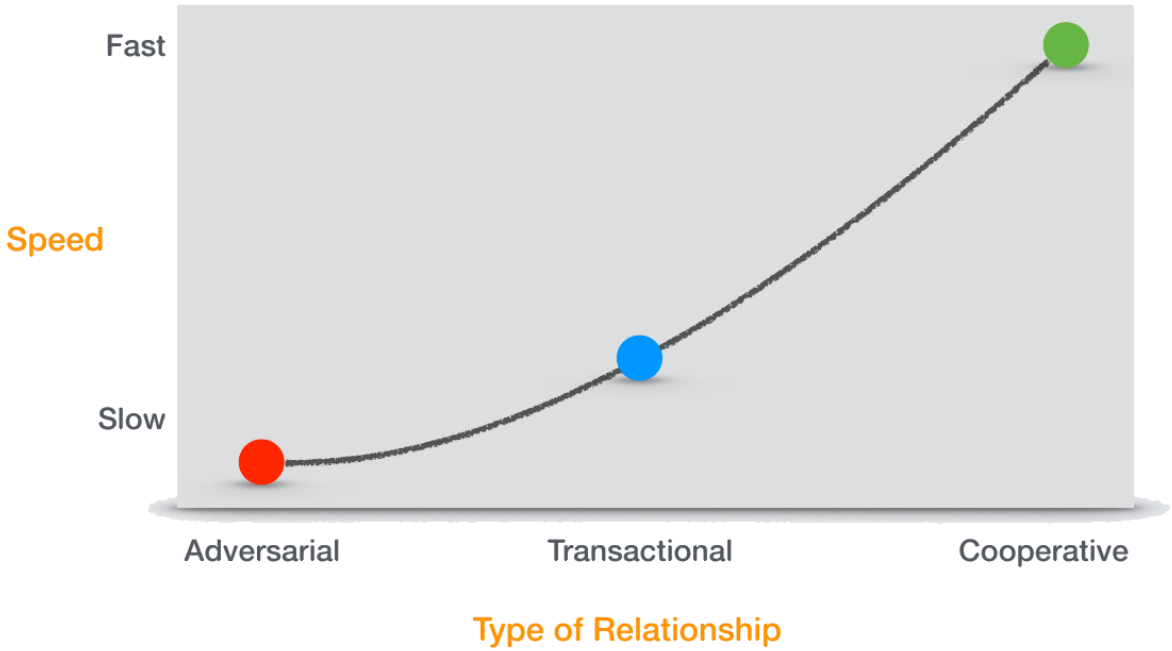

Figure 8. Development speed rises with the depth of the customer relationship. Adversarial and transactional relationships keep the team guessing; a cooperative co-development partnership converges requirements in real time. The product leader on the core team owns this relationship.

The economics: cost of delay

The reason speed is worth this much structural effort is economic. Late products lose profit at a rate most organizations never quantify. When the cost of delay is understood at the individual-contributor level — not just in a finance model — it changes behavior, because everyone can see what a week is worth. Reinertsen made the quantification of delay cost the center of modern development economics [4, 5]: if you do not know what a week of delay costs, you cannot set priorities, size queues, or justify the dedicated, empowered team that prevents the delay. The core team is expensive relative to a part-time functional committee. Against the cost of delay it is cheap.

A program under pressure

A greenfield semiconductor fab makes the case concretely. The charter was to build and start up an advanced fab in under two years, on a multi-billion-dollar budget, in a remote location, with a lead customer committed to high-volume production in two years for a major product — and a cost of delay above five million dollars per day in lost profit. The program was run on the FTTM structure: a core team of five establishing the top-down plan and the integrated master schedule, expanded to roughly fifteen with cross-functional sub-teams and milestone teams organized around integration points, planning refreshed weekly, and macro plans pulled in month over month. The first qualified process wafer of the high-volume line was delivered two weeks ahead of a schedule that had been judged impossible. The structure — role-based, empowered, integrated — is what made an aggressive date achievable [7].

The lateralization of programs

Step back and the whole argument is one idea. Functional organizations bias people to think vertically — and, as Melvin Conway observed, organizations tend to build systems that mirror their own communication structure [15], so a vertically-organized company ships vertically-fractured products. Fast programs bias people to think laterally, across the walls, toward the integrated product a customer receives. Roles instead of functions is the substitution that produces the sideways bias; the triad and the integrator are the roles that carry the most weight; empowerment is what lets the laterally-organized team move at speed. This is the lateralization of programs, and the FTTM core team is its instrument. Of every decision a company makes about how it develops products, how it builds and empowers this team is the one that moves the date the most. Build the team right, and speed becomes the default rather than the exception.

09

Portfolio

The best-practice portfolio

The core team does not run on charisma. It runs on a specific set of practices, and those practices are part of a larger portfolio. Seeing the whole portfolio makes it clear why the team sits where it does — and which practices, precisely, make it effective.

The FTTM framework organizes roughly thirty best practices into a single map. This section lays out that map and then isolates the handful of team practices that decide whether a core team performs [6].

Section 09

The best-practice portfolio

The portfolio is organized by two questions: who acts, and what the action delivers. Three actors run down the left. Mindset is the set of beliefs a fast organization holds — that late products destroy profit, that cost of delay belongs in the hands of individual contributors, that slow structures should be dismantled on purpose. Host is what the corporation does to enable speed: how it prioritizes the portfolio, provisions and protects teams, and measures performance. Team is what the core team itself does: get close to the customer, define the right product, and drive the schedule. Across the top run the delivery dimensions — the development environment and organization structure, the portfolio discipline, and execution split into the right product, the right team, and the right time. Every practice lands in one cell of that grid.

	Mindset	Environment	Portfolio	Execution		
	For Speed	Development Framework Organization Structure Decision-Making	Prioritized Aligned Managed	Right-Product	Right-Team	Right-Time
Host	Grasp enormous financial impact late products have on profitability and growth	Fast development framework	The fuzzy front end is managed		Cross-Functional Core Teams (CFCT); Heavyweight / dedicated	Projects get killed that fail to meet objectives
	Create fast culture through close-in oversight of product roadmap	Provision teams	Projects are prioritized based on business objectives		Project-based performance measurement	
	Systematically dismantle slow structures	Fast decision-making system	Projects continuously aligned with resources and skills available		Freedom-level 1 empowerment	
	Accelerate the “threshold of pain,” reward early warning behavior		New products are tracked to break-even			
Team	Cost-of-delay is understood at individual contributor level; used to drive urgency	Co-develop with tier 1 customers; continuously refine requirements		Get customer requirements; translate to product definition; reconfirm through development life cycle	CFCT roles not functions; lateralized	Macro-micro planning; micro near, macro far...short interval scheduling
		Schedule as driver (vs reporter)		Know what they value, when they want it, how much they will pay for it	Get the best people; mix of inside and external SMEs; the virtual team	Team planning, simulation, continuous scrubbing, breakdown, ownership
				Understand customer wants vs. hows; prioritize the drivers that maximize customer satisfaction		Market/Customer driven vs. resource constrained
				Start design while refining requirements; never freeze specs		Know the gap; use to drive before the fact behavior/ urgency
						Refresh Planning; continuous schedule pull-in

Figure 9. The complete FTTM best-practice portfolio. Actors run down the left (mindset, host, team); delivery dimensions run across the top (environment, portfolio, and execution — right product, right team, right time). The highlighted column is right-team: the practices, split between what the host provides and what the team does, that determine whether the core team is effective.

Most of the portfolio sets the conditions for speed. The practices that build the core team itself sit in a single column — right team — and they come in two halves. The host owns the first half: it has to stand up a heavyweight, dedicated core team, measure the team on the project rather than on functional goals, and grant it real freedom to act. The team owns the second half: it has to organize around roles rather than functions, and it has to be staffed with the best people, blending inside talent with external subject-matter experts. Neither half works without the other. A team organized around roles but measured on functional goals will drift back to its functions; a team granted freedom but staffed with part-time generalists will not earn the trust that freedom requires.

What the host must provide

Practice	Why it makes the core team effective
2.8 Cross-functional core teams — heavyweight and dedicated	Puts real budget, resource, and decision authority in the team and dedicates its members to one program. This is the structural difference between a lightweight coordinator and a heavyweight integrator (Sections 03, 06).
2.9 Project-based performance measurement	Ties reward to the program hitting its product and schedule targets, not to functional goals. Without it, members protect their function and the project loses (Section 04).
2.10 Freedom-level 1 empowerment	Pushes decision authority down to the team so it can act and report rather than ask and wait — the empowerment that converts good structure into speed (Section 07).

What the team must do

Practice	Why it makes the core team effective
3.8 Cross-functional core team — roles not functions; lateralized	The core substitution. Members take program roles — chief engineer, architect, integrator, voice of the customer — and subordinate their functions, which pulls the whole program laterally toward the outcome (Sections 04, 05).
3.9 Get the best people; mix of inside and external SMEs	Staffs the triad and the extended team with credible experts who can speak for their function and make decisions, so the team carries real technical authority rather than a reporting relay (Section 05).

These five are the load-bearing practices, but the core team is also the place where much of the rest of the portfolio gets executed. The team co-develops with tier-one customers and continuously refines requirements [8]; it translates customer needs into product definition and starts design without freezing specs; it runs schedule as the driver rather than the reporter, with macro-micro and refresh planning pulling the near-term schedule in. The host, for its part, provisions the team quickly and eliminates interrupts. Read the portfolio as a system and the pattern is clear: the core team is the convergence point where mindset, host, and team practices meet and turn into a delivered product.

Core thesis, restated. The portfolio has roughly thirty practices, but only a handful build the team itself — three the host must provide (heavyweight and dedicated, project-based measurement, freedom to act) and two the team must own (roles not functions, the best people). Get those five right and the core team becomes the engine that executes the rest of the portfolio at speed.

A

Appendix A

Core team roles and charters

A core team is defined by its roles before it is staffed by its people. Get the roles and responsibilities right first; add names second. The charters below are a template to start from, adapted from lateralworks engagement practice — trim or add roles to fit the program.

Appendix A

Core team roles and charters

Program sponsor usually external to the core team

Charter. Provide strategic direction, act as the tiebreaker on issues the core team cannot resolve, and report by exception.

Key responsibilities. Acts as the venture capitalist inside the corporation; is accountable for the business success of the program from approval through general availability; secures the budget and resources from the company; and provides advisory support and clout as the team competes for resources and mindshare.

Program director CEO of the project

Charter. Provide and maintain consistent focus on the vision for the program.

Key responsibilities. Defines program goals and outcomes clearly; sets and manages the critical success criteria and modifies them as the product evolves; manages core-team members against their individual charters; and breaks ties on key decisions by putting a clock on them. Keeps the "what" and the "when" in front of a team that will otherwise drift toward interesting technical problems.

Program manager — system integrator COO of the project

Charter. Co-own the overall program plan of record and drive the program to it.

Key responsibilities. Owns and continuously refreshes the master (macro) schedule; coordinates all participating teams and cross-functional groups; settles resource contention between milestone teams and functions; runs the weekly tactical and refresh-planning meetings to surface and clear blocking issues; provisions the team and minimizes interrupts from the host; and laterally links the work-streams into one integrated whole. Owns right-time performance.

Architect owner of the system

Charter. Provide leadership for all technical aspects of the program and hold the top-level systems-integration vision of the end product.

Key responsibilities. Maintains the end-to-end technical vision; makes the system trade-offs so the whole is not compromised for a part; manages the subsystem budgets of time, cost, and performance; owns the interfaces and their completeness at engineering release; and works closely with the chief engineer as the two technical decision-makers on the program.

Engineering leader — chief engineer final technical decision-maker

Charter. Ensure the product and plan conform to the requirements and will work in the customer environment; provide primary technical leadership and decision-making.

Key responsibilities. Is the single person who knows all the subsystems and how they interact; a generalist who works with the specialist experts, listens to the debate, evaluates the data, and makes the final technical decision — and cannot be overruled on it. Has ultimate responsibility for the success or failure of

the total system, drives the design process, integrates the test plans to the requirements, and acts as the first customer of manufacturing for the initial build.

Marketing leader voice of the customer

Charter. Provide the external link to market and customer for the internal development team.

Key responsibilities. Gets, understands, prioritizes, and continuously refreshes customer requirements; converts them into product requirements and specifications; prioritizes those specifications by customer value so the right product is what gets built; understands the customer's business problem rather than a feature list; carries the cost-of-delay message to team and host; and reconciles what the market wants against what can be delivered inside the window.

Milestone-team leader end-to-end team leader

Charter. Manage a milestone team's plan of record and its integrated deliverable.

Key responsibilities. Defines the milestone, its outcomes, and its critical success criteria and manages to them; coordinates the cross-functional participants who must deliver the integrated milestone; drives the sub-project against a continuously updated schedule; runs tactical meetings to clear blocking issues; and interfaces with the core team — providing status, variances, planned recovery actions, and requests for resources. Milestone teams form around integration points, deliver, and disband; they are the only second layer of hierarchy on a fast team.

A note on using the template. Adapt it to the program. Add or remove roles as the scope requires. Get the roles and responsibilities right before adding people by name. Normal-sized programs run a core team that interacts with the functional resources; large programs run a core team plus milestone teams, with hundreds of contributors arranged in that flat, two-level configuration — a small group at the center and multiple sub-teams revolving around it.

Sources

References

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

[2] 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.

[3] Clark, K. B., and Fujimoto, T. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry.* Harvard Business School Press, 1991.

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

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

[6] lateralworks. *FTTM New Product Development Best Practices*, Revision 2018.002. Practice 3.8, "Cross-functional core team: roles not functions, lateralized"; Practice 2.7, "Provision teams, eliminate interrupts."

[7] lateralworks. "Internal assessment and engagement database." Field observations across advanced-technology programs, 1988–2026.

[8] lateralworks. "Integrated core team." Ideas library, lateralworks.com. https://lateralworks.com/ideas/integrated-core-team

[9] lateralworks. "The failure of compartmentalized thinking." Ideas library, lateralworks.com. https://lateralworks.com/ideas/the-failure-of-compartmentalized-thinking

[10] Kitroeff, N., Gelles, D., and Nicas, J. "Boeing Built Deadly Assumptions Into 737 Max, Blind to a Late Design Change." *The New York Times*, June 1, 2019. https://www.nytimes.com/2019/06/01/business/boeing-737-max-crash.html

[11] lateralworks. "Fast autonomous teams." Ideas library, lateralworks.com. https://lateralworks.com/ideas/fast-autonomous-teams

[12] lateralworks. "Are program managers of value?" Ideas library, lateralworks.com. https://lateralworks.com/ideas/are-program-managers-of-value

[13] Oncken, W., Jr. *Managing Management Time: Who's Got the Monkey?* Prentice Hall, 1984. (Origin of the levels-of-initiative / freedom scale adapted here.)

[14] lateralworks. "Freedom to act." Ideas library, lateralworks.com. https://lateralworks.com/ideas/freedom-to-act

[15] Conway, M. E. "How Do Committees Invent?" *Datamation*, vol. 14, no. 4, April 1968, pp. 28–31. (Conway's law: organizations design systems that mirror their own communication structure.)