Whitepaper

The management gap

Twenty years of compounding technology innovation outran the way technology programs are managed. The lateralworks case studies show what works instead.

FTTM methodology series.

A lateralworks position paper. Two decades reshaped the technology; the methods used to manage technology programs did not move with it. The paper traces the gap, looks at the evidence on Agile and complex programs, lays out the FTTM research base, and walks two lateralworks case studies (LSI Logic 500K, 1993–1995; AI-assisted macro planning, 2026) that show the structural alternative in operation.

Prepared by	Date	Online
lateralworks	May 2026	lateralworks.com
Position paper	Rev 1.0	FTTM series

Table of contents

The management gap

Core thesis. Technology capability has grown by orders of magnitude since 2005. The way technology programs are managed has not. Agile, the most visible methodology change of the period, sped up feedback loops inside software teams but did not fix the structural problems that make complex, multi-disciplinary programs late. The lateralworks FTTM methodology, refined since 1988 across 200+ technology programs, is the documented exception. The case studies are the evidence.

Overview

Abstract

Programs that miss their market window usually do not miss because the engineering is slow. They miss because the operating model that organizes the work was built for a slower era.

Between 2005 and 2025 the underlying technology stack moved by orders of magnitude. Compute per dollar, transistor density, network throughput, and now generative AI compute have all jumped on curves no prior era matched. Mainstream program management did not. The methods practiced today (Waterfall in the 1990s, Agile since the early 2000s, hybrids since the 2010s) come from the same root ideas, refined at the margins. The Standish CHAOS series reports a successful-project rate that has moved between 16% and 37% over three decades, with large projects succeeding less than 10% of the time [1, 2, 3, 4].

The argument here is structural. Agile compressed the design-build-test loop inside small software teams and shows real gains there. On complex multi-disciplinary programs the evidence runs the other way: Agile says little about the interfaces between workstreams, about who the driving customer is, or about the critical path, which is usually where complex programs go late. The lateralworks Fast Time To Market (FTTM) methodology, refined since 1988 across 200+ engagements, is the documented counter-example [5, 6].

The paper traces the gap (§01), looks at what Agile did and did not change (§02), lays out the FTTM research base (§03), walks the LSI Logic 500K and the 2026 AI-assisted planning case (§04), distils the reusable pattern (§05), and closes with what this means for technology teams now (§06).

Key finding. The lateralworks case studies are not anecdotes. They are field-tested instances of a structural alternative to mainstream program management: one driving customer, an empowered cross-functional core team, lateral integration across hardware and software, a planning cadence that surfaces gaps early, and (since 2026) AI-assisted synthesis that compresses the planning artifact from weeks to hours [7, 8].

01

The widening gap

The gap: technology accelerated, management did not

Technology capability and program-management method have run on very different clocks since 2005. Technology has compounded. The method used to run technology programs has not.

On the technology side, transistor count in a leading-edge processor has grown from roughly 290 million (Intel Pentium Extreme Edition, 2005) to over 100 billion (NVIDIA Blackwell-class accelerator, 2024), a 350x increase that tracks the long-running Moore's law trajectory [9, 10]. Cloud compute pricing has dropped by more than two orders of magnitude in real terms over the same period. Mobile bandwidth has crossed three generations of standard. Generative AI compute alone has grown faster than anything else in the stack: training-run FLOPS for frontier models doubled roughly every six months between 2018 and 2024 [11].

On the management side over the same window, the textbook used to train program managers in 2025 traces its concepts to PERT and CPM from the 1957–1958 Polaris and DuPont programs, refined by the Project Management Institute in the 1980s, augmented by the Agile Manifesto in 2001 [12], scaled by SAFe in 2011, and rebranded under various hybrid labels since. The intellectual core has not changed. A 2026 program manager and a 2005 program manager work from the same toolkit: a work breakdown structure, a Gantt chart, a risk register, a status report, a stand-up. The tools that operationalize these have improved in usability but not in the underlying model.

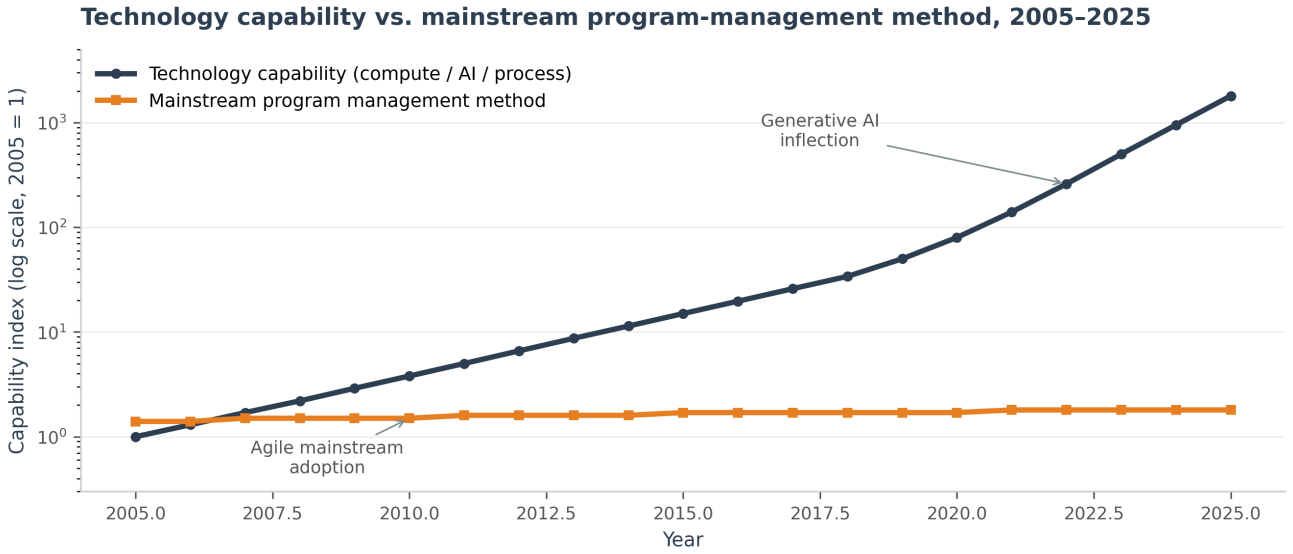

Figure 1. Indexed technology capability vs. mainstream program-management method, 2005–2025. Technology rides a compounding curve driven by Moore's law and then generative AI compute. Mainstream program-management methods are almost flat, with one limited inflection from Agile adoption around 2010.

Why the asymmetry matters

Complex programs pay the bill

How much the gap costs depends on how complex the program is. A simple program (a website refresh, a mobile-app feature, a single-chip respin) is tolerant of stale methods because the surface area is small. A complex program (a new semiconductor product, a multi-partner integration, a regulated product launch) pays for management drag at each interface. The same framework that works on a six-person sprint delivers late on a 200-person, multi-year, multi-partner program because the framework was not designed for that scale of integration.

The asymmetry. Technology capability has grown roughly 1000x in two decades. The mainstream framework for managing programs that use that technology has barely moved over the same period. The drag on Time-to-Market compounds, and hits the most ambitious programs hardest.

02

Methodology critique

Agile: progress on one axis, regression on another

Agile is the most-cited methodological innovation of the last twenty years. It worked on small software programs and ran into trouble on complex ones. The reason is structural.

The 2001 Agile Manifesto laid out four values (individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan) and twelve supporting principles [12]. The values are sound.

Where Agile helped

On small software programs where the customer can be in the room, the product surface is one screen at a time, and the deployment cadence is daily, Agile compressed the design-build-test loop in real and measurable ways. The Standish CHAOS data show Agile projects succeeding three times more often than waterfall on small projects, with lower cost and schedule overruns [13].

Where Agile did not help

Time-to-Market on complex programs tells a different story. The same Standish data, broken out by project size, show that large projects succeed less than 10% of the time regardless of methodology, and that the overall failure rate across IT projects has hovered around 60–70% for three decades. The pattern shows no clear lift from Agile adoption [1, 4, 14]. McKinsey found that 70% of digital-transformation efforts fall short of their targets, and that 17% of large IT projects threaten the survival of the company [14].

Standish CHAOS outcomes for IT projects, 1994–2020

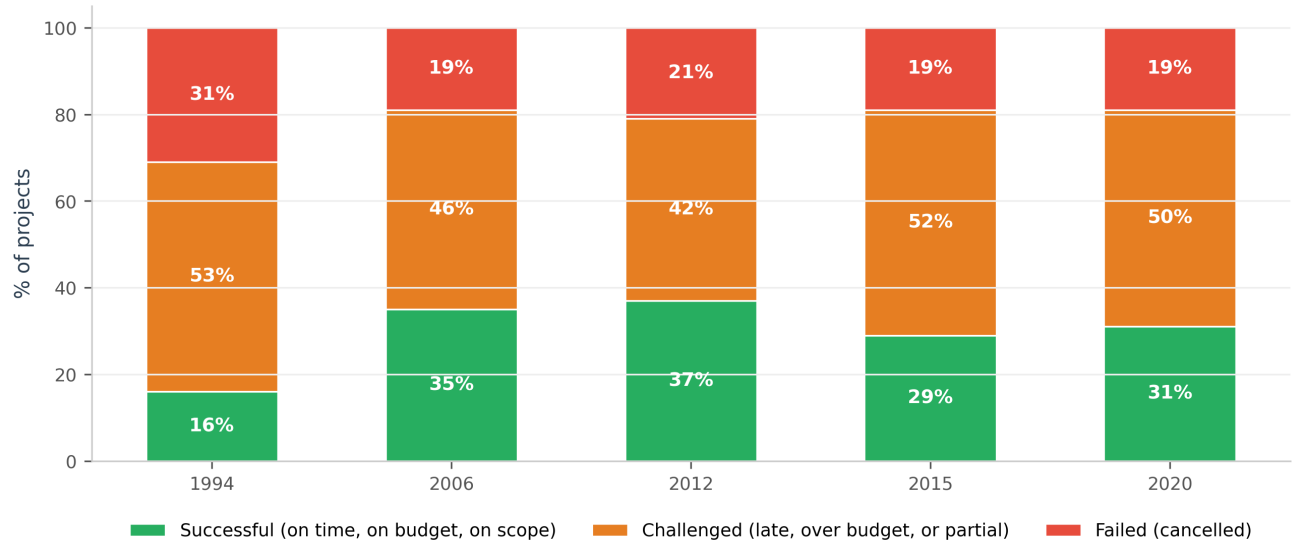

Figure 2. Standish CHAOS outcomes for IT projects, 1994–2020. Success rates have moved between 16% and 37% over three decades. Challenged projects have remained around 42–53% throughout. The overall picture is oscillation, not improvement. Sources: Standish Group CHAOS Report (1994, 2006, 2012, 2015, 2020) [1, 2, 3, 4].

The structural mismatch

Sprint cadence vs. program cadence

Agile helps on a small program and stops helping on a large one because the operating cadence does not match. A sprint runs in two weeks; a fab cycle runs in eight to twelve weeks; a partner contract negotiation runs in two quarters; a qualification campaign runs in three quarters. None of those can be re-scoped at the end of every sprint without compounding cost, and most cannot be re-scoped at all. A team that operates on a sprint cadence but depends on workstreams that operate on a quarterly cadence has a ceremonial sprint, not an operational one.

Three other problems compound the cadence mismatch on complex programs. Agile says little about lateral integration across different workstreams; a sprint optimizes the inside of one software team, while a complex program lives or dies on the interfaces between hardware, software, tooling, library, fab process, packaging, and supply chain. Agile says little about the driving-customer question; serving seven customer segments at once produces sprint-by-sprint requirements churn that Agile neither prevents nor resolves. And Agile says little about the critical path; the team sees the next two weeks and the burndown chart, not the dependency chain that decides when the program ships. The method, not the team, is the limit.

The Agile paradox. Agile sped up feedback inside a software team. On complex multi-disciplinary programs it tracks with longer Time-to-Market, not shorter, because the method does not cover the interfaces, the driving customer, or the critical path — which are where complex programs usually go late.

03

The research base

FTTM: thirty-six years of research

lateralworks was founded in 1988 to answer one question: why do some technology teams ship on time, or early, on programs that any reasonable person would call impossible, while other teams with similar talent and similar budgets ship late on programs that look easier? The methodology that has come out of that work is called Fast Time To Market, or FTTM. FTTM is not a consulting framework; it is a set of observed practices, refined through direct engagement on 200+ programs across semiconductor, hardware, software, and integrated-systems work [5, 6].

The research base is what sets FTTM apart. Many management frameworks are codified from a small number of authoritative case studies and then extended by inference. FTTM works the other way: it starts from direct observation of 500+ engineers across hundreds of programs, and only codifies what the observation supports [5]. Four findings, each contradicting a mainstream assumption, sit at the centre.

The critical path is rarely where you think it is

Teams usually watch the obvious critical path. The second or third critical path is the one that derails the program: a partner decision, a fab process step, a single under-staffed function. The discipline is to find the non-obvious chains and instrument them [5].

Buffer is not slack; it is a managed resource

Positive buffer (margin ahead of a milestone) and negative buffer (debt behind a milestone) are not symmetric. They consume at different rates, signal different things to the executive layer, and call for different management. FTTM treats buffer as an inventory account, not a rounding error [5].

Speed comes from decision quality, not decision speed

Fast teams in the observation base do not decide faster; they decide better and revisit decisions less often. The fastDecisionAI methodology built on this finding treats decision quality, not throughput, as the lever [5].

Portfolio drag is the silent schedule killer

Too many active projects at the organization level starve each one of the resources it needs to ship. FTTM Portfolio methodology attacks this drag by treating project starts as the controlled variable, not project ends. The implication is that the most important schedule decisions sit at the portfolio layer, not inside any single program [5].

These findings sit underneath the program-level practices that show up in the case studies: a single driving customer, a cross-functional core team, an integrated master schedule run as the operating system of the program, lateral integration across hardware and software, weekly Voice-of-Customer, and (since 2026) AI-assisted planning that compresses synthesis without surrendering judgment [7, 8].

04

Field evidence

Two case studies

The lateralworks case studies are the main evidence for the argument here. They are field-tested, publicly documented at lateralworks.com/results/case-studies [6], and not retrospective narrative — the program records, schedule artifacts, and outcome data are preserved and citable. Two cases follow.

Case A — Sony PlayStation

The LSI Logic 500K ASIC, 1993–1995

In 1993 Sony Computer Entertainment was the next challenger in a console market that had already absorbed four failed entrants (Panasonic 3DO, Atari Jaguar, Philips CD-i, Commodore Amiga CD32) [7]. The PlayStation depended on a single custom ASIC that no semiconductor vendor had committed to deliver: a 500,000-gate, single-die system-on-chip integrating a 32-bit RISC CPU, a 3D geometry engine, a motion decoder, DMA controllers, and bus interfaces. LSI Logic, then a sub-$500M ASIC company, took the design [7]. LSI's prior largest part was 100,000 gates. The Sony part was a 5x leap, on a new process node, with new design tools, a new cell library, and a new packaging approach, to be delivered to volume production in 24 months.

The LSI marketing organization had specified a "super-product" combining the requirements of seven vertical segments into one design. Internal engineering estimated five-plus years to build that specification. Wilf Corrigan, the LSI CEO, was asking for two [7]. The program as written could not ship.

lateralworks restructured the program around three reinforcing moves. The super-product was abandoned. Pick one driving customer whose requirements cover roughly 80% of general market need, focus the entire program on that customer, expand to the remaining segments only after launch. The driving customer chosen, over the initial objections of a marketing organization that dismissed gaming as a "toy" market, was Sony. The first cross-functional core team in LSI's history was stood up: seven senior subject-matter experts, co-located, under a single Project Monarch (Jean-Louis Bories), with Brian Halla as senior accountable executive. And the program ran on a high-frequency refresh-planning cadence: macro plan refreshed weekly, micro plans refreshed at workstream level across hardware and software, weekly Voice-of-Customer with Sony sustained for the full two years [7].

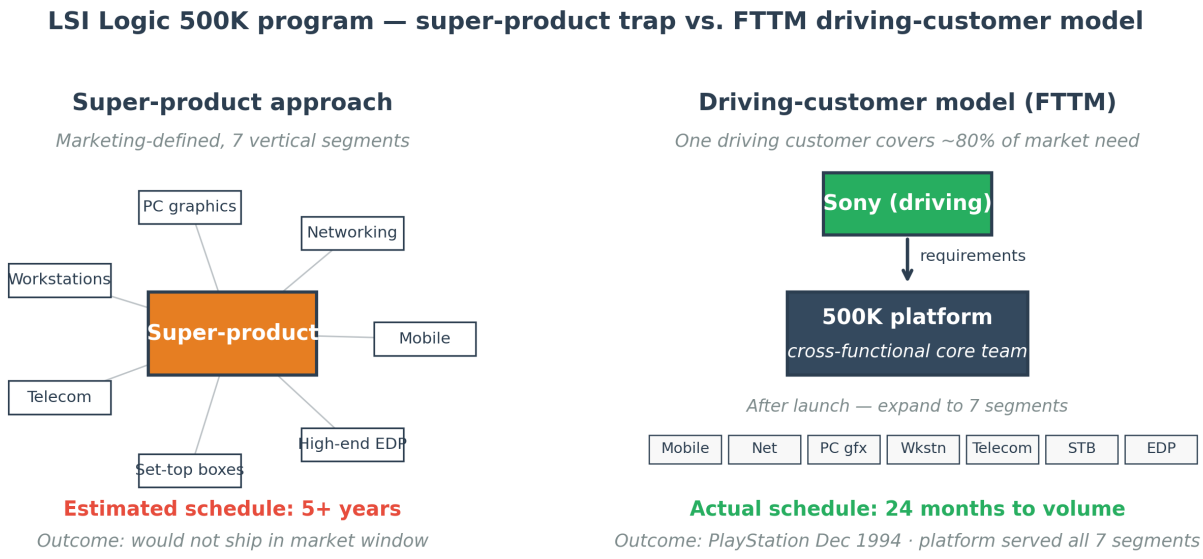

Figure 3. The super-product trap on the left vs. the FTTM driving-customer model on the right, as applied to the LSI Logic 500K program. The super-product would have served seven segments — if it had shipped. The driving-customer model shipped to Sony in 24 months and went on to serve all seven segments anyway, over the next seven years. Source: lateralworks case study [7].

The outcome

Engineering samples shipped to Sony 12 months ahead of the original system schedule. Volume production began 24 months after kickoff. The first PlayStation reached the Japanese market on December 3, 1994, sold 100,000 units the day it hit stores, and by November 1995 was outselling Sega Saturn three to one in the United Kingdom [7]. The original PlayStation shipped 102M lifetime units, the first console to clear that threshold. The Sony engagement roughly doubled LSI Logic in a single year, and LSI's stock price moved from $4 7/8 to $126 over the engagement window [7]. The 500K platform served as the technology foundation for LSI's product line over the next seven years, and eventually expanded to all seven of the original super-product's vertical segments. The focus had come first; the breadth came after.

Case B — AI-assisted macro planning

A 2026 semiconductor program

The second case is more recent and tests a different question: can the FTTM operating model be run faster? The case comes from a 2026 semiconductor program planning a multi-year, multi-partner stacked-memory device, peak engineering headcount in the hundreds and a budget in the tens of millions [8]. The conventional planning cycle for that program would have taken three to four weeks of meetings, drafts, and reviews, and the plan would have been obsolete the moment a partner decision shifted.

The team replaced the conventional cycle with an AI-assisted three-step flow. A senior project manager fed source documents (charter, milestone-scrub minutes, risk register, partner emails, meeting transcripts) into a large language model and prompted three connected artifacts: a one-page milestone timeline for the executive sponsor, a milestone matrix for the team, and a macro schedule that could be ingested by the project-management tool. Each artifact took hours to generate. The team scrubbed each draft, fed the

corrections back in, and used the time saved to argue about the program rather than the artifact. The first complete plan was on the table in three days [8].

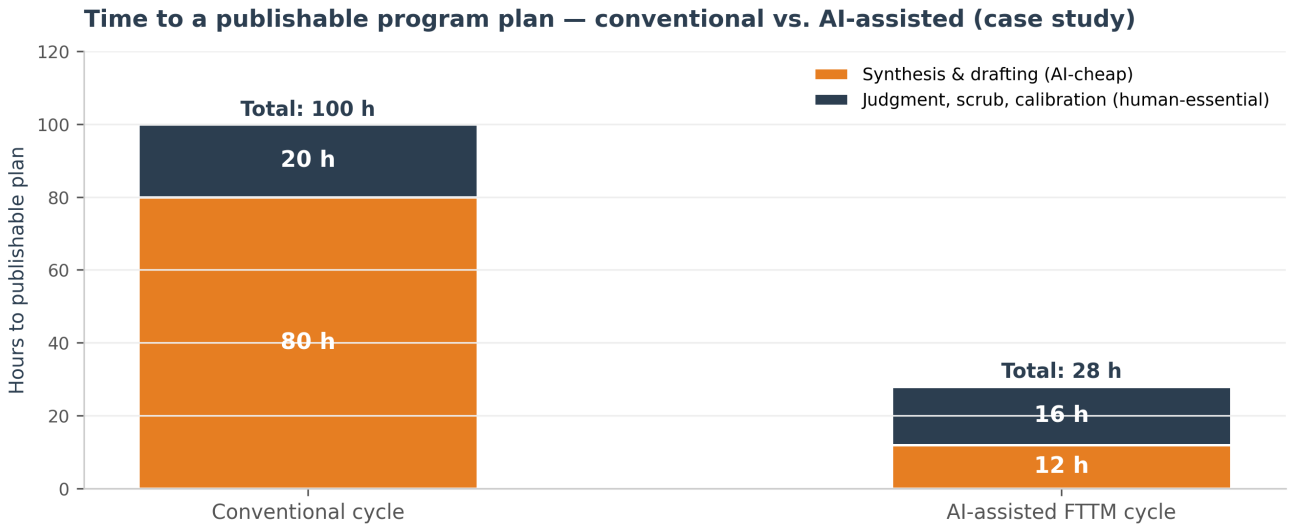

Figure 4. Time to a publishable program plan, conventional vs. AI-assisted FTTM cycle. AI compresses the synthesis and drafting work (the high-volume, low-judgment portion) from approximately 80 hours to approximately 12 hours. The judgment portion (scrub, calibration, local context, decisions) stays human and is only modestly compressed because it cannot be. Source: lateralworks case study [8]; consistent with the broader literature [15, 16, 17].

The case-study findings sit inside a broader research literature on AI-augmented knowledge work. The Harvard / Boston Consulting Group field experiment with 758 consultants found that, on tasks inside the AI's capability frontier, AI-equipped consultants completed 12.2% more tasks, took 25.1% less time, and produced 40% higher quality output than the control group [15]. The Stanford / MIT study of 5,179 customer-support agents found a 14% average productivity gain, with novice agents seeing a 34% lift [16]. The MIT writing experiment in Science found a 40% reduction in time on task and an 18% quality gain across 453 professionals [17]. The case study's working principle (AI does roughly 80% of the synthesis work in hours, the team scrubs the remaining 20% over days) is consistent with all three studies.

What the two cases share. Both are instances of structural change in how a technology program is run. The 500K case replaced a seven-segment super-product with a driving customer and the first cross-functional core team in LSI's history. The AI-assisted-planning case replaced a three-to-four-week conventional planning cycle with a three-day AI-assisted flow. Two different programs and two different decades, but the same move: find the structural constraint, change the operating model around it, and run the team at the cadence the program actually demands.

On focus

What the driving customer does

"Focus on the market leader and the rats and dogs will follow."

— Pat Brockett, National Semiconductor

Recounted in the lateralworks LSI Logic 500K case study [7]

05

Synthesis

The reusable pattern

Seven operating principles repeat across the case studies and the broader lateralworks engagement record. Together they describe what structural innovation in technology program management looks like.

01 / Find the driving customer.	One customer whose requirements cover approximately 80% of general market need. Focus the program on that customer. Expand to other segments after launch. The super-product approach (many customers, one specification) fails to ship in time for any of them.
02 / Stand up one cross-functional core team.	Senior subject-matter experts, co-located, single Project Monarch, primary job is the program. No functional hierarchy inside the team. The team is the integration point, not a status reporter.
03 / Run the schedule as the operating system.	The integrated master schedule is the organizational framework, the simulation engine, the early-warning system, and the interface manager — not a status report. Refresh weekly at the macro level, more often at the workstream level.
04 / Integrate laterally across all workstreams.	Hardware, software, tools, libraries, fab process, packaging, supply chain, and partner organizations belong on one dependency graph. Sequential handoffs are how late programs ship.
05 / Sustain Voice of Customer.	A specification frozen at kickoff is obsolete by the third month of a two-year program. Run weekly VOC. Distinguish wants (the customer's problem) from hows (the supplier's solution). Renegotiate gaps as explicit trade-offs.
06 / Expect organizational resistance.	A program that breaks the prevailing functional model generates internal resistance proportional to its strategic importance. Plan for the escalation. Get executive sponsorship in writing.
07 / Use AI to compress synthesis, not judgment.	AI compresses the first 80% (synthesis, structure, draft) in hours. The remaining 20% (calibration, local context, judgment) stays with the team and is the basis for trust in the output. The order matters: AI first, humans second.

The pattern is reusable. The 500K case shipped in 1995. The AI-assisted planning case ran in 2026. Thirty-one years separated them; the operating structure was the same. The case studies on lateralworks.com show the pattern at work in seven other program contexts [6].

06

Implications

What this means for technology teams now

Three closing observations follow from the argument so far: the gap from §01, the Agile evidence from §02, the FTTM research base from §03, and the case studies from §04.

The gap widens

Technology capability growth is now led by AI compute, which is doubling on a shorter clock than the underlying semiconductor stack ever did [11]. Mainstream program management is not on a comparable curve. Teams running on the 2015 toolkit are losing ground year by year.

The answer is not faster Agile

Sprint cadence does not match the natural cadence of complex programs, and running the sprint faster does not change that. The fix is structural. A driving customer. A cross-functional core team. An integrated master schedule. Weekly Voice-of-Customer. And the discipline to refuse the super-product trap. These are the moves the lateralworks case studies show working on programs from a $500M semiconductor partner to a $7B fab [7, 18].

AI changes the operating economics

The 2026 case study shows what becomes possible when synthesis collapses from weeks to hours [8]. The methodology is faster to operate, executive reviews are faster to prepare, and the plan is faster to refresh when a partner decision shifts. None of that changes what the right structure is. It changes how cheaply the right structure can be maintained, which is what now puts the FTTM operating model in reach for programs that previously could not afford the planning overhead.

Closing thesis. Two decades of compounding technology innovation, and almost none on the management side. The lateralworks case studies are the documented exception: a refined operating model, field-tested across 200+ programs since 1988 and now augmented by AI-assisted synthesis. The case studies on lateralworks.com are evidence, not theory, of what structural innovation in technology program management actually looks like [6].

Sources

References

The argument draws on three independent streams of evidence: the Standish project-outcome series, the post-2023 literature on AI-augmented knowledge work, and the lateralworks engagement archive. Citations are in order of first appearance.

[1] Standish Group International. *The CHAOS Report*. 1994. The original study of 8,380 software projects. 16% delivered on time and on budget; 31% cancelled; 53% challenged.

[2] Standish Group International. *CHAOS Summary 2009*. 2009. 32% success rate; 44% challenged; 24% failed.

[3] Standish Group International. *CHAOS Manifesto 2013*. 2013. 39% success rate; 43% challenged; 18% failed. Project size identified as the single largest factor in resolution outcome.

[4] Standish Group International. *CHAOS 2020: Beyond Infinity*. Edited by Johnson, J. 2021. 31% successful; 50% challenged; 19% failed. Large projects succeed less than 10% of the time.

[5] lateralworks. 'The FTTM Framework' (methodology page). https://www.lateralworks.com/methodology. Accessed May 13, 2026. 36 years of research; 500+ engineers observed; 200+ programs.

[6] lateralworks. 'Case studies' (results page). https://lateralworks.com/results/case-studies. Accessed May 13, 2026. Seven field-tested case studies across semiconductor, hardware, and complex-systems programs.

[7] lateralworks. 'Sony PlayStation: The 500K ASIC.' Case study, 23 pages, May 13, 2026. https://lateralworks.com/cases/sony-playstation-500k-asic.pdf. Engineering samples 12 months ahead of system schedule; volume production in 24 months; PlayStation launched December 3, 1994; 102M lifetime units; LSI Logic roughly doubled in a single year; stock $4 7/8 → $126 over the engagement window.

[8] lateralworks. 'AI-assisted macro planning.' Case study, 44 pages, May 2026. https://lateralworks.com/results/case-studies. Conventional planning cycle compressed from three-to-four weeks to three days using an AI-assisted three-step flow.

[9] Moore, G. E. 'Cramming more components onto integrated circuits.' *Electronics*, vol. 38, no. 8, April 19, 1965.

[10] Wikipedia. 'Transistor count.' https://en.wikipedia.org/wiki/Transistor_count. Documents the trajectory from Intel Pentium Extreme Edition (~290M transistors, 2005) through NVIDIA Blackwell B200 (208B transistors, 2024).

[11] Sevilla, J. et al. 'Compute trends across three eras of machine learning.' Epoch AI, 2022, with updates through 2024. Documents the post-2018 acceleration with doubling times of approximately six months.

[12] Beck, K. et al. 'Manifesto for Agile Software Development.' 2001. https://agilemanifesto.org.

[13] Standish Group International. *CHAOS Manifesto 2011*. 2011. Agile projects succeed three times more often than waterfall projects across the period 2002–2010.

[14] McKinsey & Company. 'Common pitfalls in transformations: A conversation with Jon Garcia.' 2020. 70% of digital-transformation efforts fall short; 17% of large IT projects threaten company survival.

[15] Dell'Acqua, F., McFowland III, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K. C., Rajendran, S., Krayer, L., Candelon, F., and Lakhani, K. R. 'Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of Artificial Intelligence on Knowledge Worker Productivity and Quality.' Harvard Business School Working Paper 24-013, September 2023. 758 BCG consultants; 12.2% more tasks completed, 25.1% less time, 40% higher quality inside the AI's capability frontier.

[16] Brynjolfsson, E., Li, D., and Raymond, L. R. ‘Generative AI at Work.’ *Quarterly Journal of Economics*, vol. 140, no. 2, 2025, pp. 889–942. NBER Working Paper 31161, April 2023. 5,179 customer-support agents; 14% average productivity gain; 34% for novice agents.

[17] Noy, S., and Zhang, W. ‘Experimental evidence on the productivity effects of generative artificial intelligence.’ *Science*, vol. 381, no. 6654, July 2023, pp. 187–192. doi:10.1126/science.adh2586. 453 college-educated professionals; 40% reduction in time on task; 18% quality gain.

[18] lateralworks. ‘Programs that changed industries’ (home page results). https://www.lateralworks.com. Documents the Fab8 engagement at GlobalFoundries Malta, NY — a $7B greenfield fab, $5M/day saved by first silicon arriving two weeks ahead of target; the Philips Velo engagement that produced the engineering executive who later built the iPod at Apple (Tony Fadell); and the Charles Schwab IT Infrastructure Rationalization program, $15M in annual savings, internal Program of the Year.