Whitepaper

The long pole

Find the path that sets the end date, challenge the assumptions holding it up, and whittle it down every week

FTTM best practices series.

How fast teams treat the critical path as a target rather than a report: the four-step long pole method, the challenge process behind it, and a wigglechart from a semiconductor fab program that landed first silicon two weeks early after predicting a two-year slip.

Prepared by
lateralworks
FTTM methodology

Date
July 2026
Methodology paper

Online
lateralworks.com
FTTM best practices series

Table of contents

The long pole

Core thesis. The long pole sets the end date. Most organizations watch it grow; fast teams cut it down. Find the real long pole, challenge the assumptions that hold up its duration, cut in big chunks without letting the tent fall down, and whittle it every week through refresh planning. Do this and the end date becomes something the team sets, not something it discovers.

Overview

Abstract

Every project has one path that decides when it ends. The long pole in the tent holds the tent up and sets its height. The long pole in a project is the critical path: the longest contiguous chain of activities from start to finish, the one path with no slack [1]. Whatever happens to the long pole happens to the ship date. Everything else just watches.

The critical path has been formal engineering knowledge since DuPont and Remington Rand published the critical-path method in 1959 [2] and the U.S. Navy built PERT for Polaris a year earlier [3]. Yet most organizations still treat it as a reporting artifact: something to color red in the review deck and watch. Nobody attacks it. Left alone, the pole grows. Work expands to fill the time allotted [4], people start tasks when deadline pressure arrives rather than when the task does [5], and everyone downstream quietly counts on the slip. The record shows how this ends: across a database of more than 16,000 projects, 91.5 percent miss their time or budget targets [6].

This paper sets out the lateralworks long pole method, a core discipline of the FTTM best-practices system [7]. Four moves: find the real long pole and the real gap to target, even when the honest schedule is ugly; determine what makes the pole that long; challenge the assumptions holding up its duration and cut it down in big chunks without letting the tent fall down; then whittle it further every week through refresh planning. Where the work is real invention and durations cannot be known, the method estimates in learning cycles and details only what the last experiment has made knowable.

The evidence runs through a wigglechart from a major semiconductor fab program. When the team built its first honest schedule, the predicted date of first silicon, the first working wafers through the new plant, landed about two years past the January 2012 target. Challenge workshops pulled the prediction back to the target band within months, weekly refresh planning held it there for eighteen months, and first silicon started on December 27, 2011, two weeks ahead of target [8]. The paper also places the method next to the published literature it converges with, and departs from: assumption-based planning, the premortem, and critical chain.

01

Concept

The pole and the tent

A big tent takes its height from one pole. A project takes its duration the same way: dozens of workstreams, thousands of dependencies, and still a single contiguous path of work that decides when everything ends [1]. This section names that path, traces where the idea came from, and explains why on most projects the pole quietly grows.

The critical path, by its right name

Morgan Walker of DuPont and James Kelley of Remington Rand formalized critical-path planning in 1957 and published it in 1959, after trials on chemical-plant construction and maintenance shutdowns [2]. The Navy's Special Projects Office built PERT for the Polaris missile program at the same time [3]. Both methods answer one question: of all the paths through the network of work, which one has zero float? That path is the critical path. Delay it a day and the project ends a day later.

"The long pole in the tent" is the working engineer's name for the same thing, common enough in software culture that Raymond Chen catalogued it as Microsoft "microspeak," the thing everyone else is waiting on [9]. lateralworks uses the term deliberately. "Critical path" sounds like something a scheduler owns. "The long pole" sounds like something a team can cut, and that is the point.

Why poles grow

Left unmanaged, the long pole does not hold its length; it grows. The mechanisms are well documented. Work expands to fill the time available for its completion, Parkinson's original observation [4]. People apply themselves when the deadline gets close, not when the task arrives, the pattern Goldratt called student syndrome [5]. Planners anchor on best cases and inside views, the planning fallacy Kahneman and Tversky identified [10]. And slips accumulate invisibly between schedule updates until they surface all at once.

On a late project a stranger mechanism takes over. The longer the pole gets, the more room every other team has, and the less pressure anyone feels. Everyone is counting on the long pole to keep growing. Nobody thinks about pulling their own work in, because the project end date keeps pushing out anyway; if it all fails, at least they can blame the poor guy on the pole [1]. lateralworks calls this the late-schedule mentality, and it is why exhortation does not fix late projects: the incentives all point the wrong way. The aggregate outcome is grim. Flyvbjerg's database of over 16,000 projects across 20 fields shows 91.5 percent missing time or budget, and only 0.5 percent delivering on time, on budget, and on benefits [6].

Invention sets the floor

Before cutting the pole, respect what makes it long. A consistent rule of thumb from lateralworks engagements: most things can be done in about a year if they involve little or no invention. Add invention and you add years, one to two for incremental invention, three to five for breakthrough invention [8]. A related field rule sizes the learning cycles inside the pole: easy problems need one to three cycles, familiar-but-changed problems three to five, never-done-before problems five to seven [11]. The long pole on a development program almost always runs through the invention, which is why shortening it is an assumptions problem, not a motivation problem. Section 04 shows how to put a number on the cycles.

> **The invention clock.** Little or no invention: about a year. Incremental invention: add one to two years. Breakthrough invention: add three to five years. If the plan shows breakthrough invention finishing in twelve months, the schedule is fiction.

02

Method

Find reality, then cut

The long pole method is four moves, run in order and then repeated weekly. Find reality. Ask why the pole is that long. Challenge the assumptions and cut in big chunks. Refresh the plan every week and whittle. Figure 1 shows the sequence the way lateralworks draws it for teams.

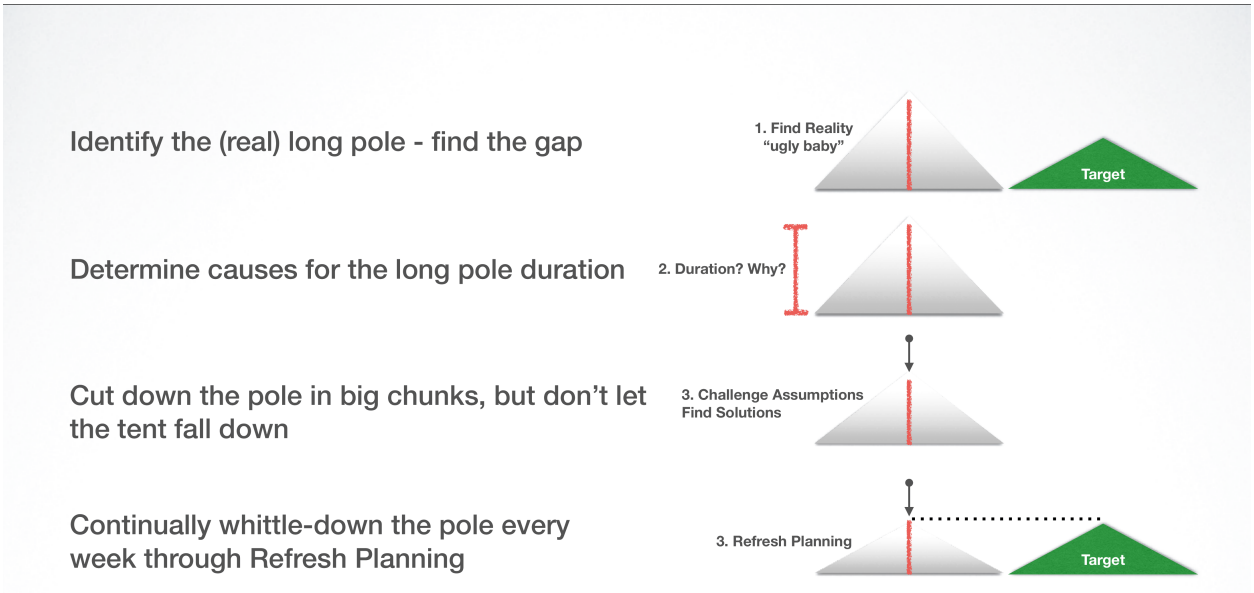

Figure 1. The long pole method: find the real pole and the gap to target, determine what drives its duration, challenge assumptions to cut it in big chunks, then whittle it weekly through refresh planning.

1. Find reality, the ugly baby

Every program carries a schedule everyone knows is fiction: durations manufactured to land on the date management wants, believed by no one, defended by all. The first move is to build the honest schedule instead, and the honest schedule is usually an ugly baby. It shows the project finishing late, sometimes badly late. Show it anyway [11]. When the durations are unknown, estimate conservatively rather than aggressively; if there is a gap to target, the team needs to see it now, while there is still time to change strategy: buy instead of make, outsource instead of hire, resize the product.

Finding reality also means finding the real long pole. On a schedule that has not been scrubbed, the visible critical path is often wrong: missing tasks, stale durations, and unreported slips hide the true driver. The gap between the honest predicted finish and the target is the number the whole method works against.

2. Duration? Why?

With the real pole visible, interrogate its length. What activities compose it? Who owns them? Which durations are measured, which are quoted, and which are guessed? Break the near-term work into tasks of one to five days so slips show up the week they happen rather than months later [11]. The output of this step is a causal account of the pole's duration, specific enough that each segment can be attacked by name.

3. Challenge assumptions, find solutions

Most of any long pole is not physics; it is assumptions with dates attached. Quoted vendor cycle times taken as law. Full-scope qualification nobody questioned. Serial hand-offs that could overlap. Some segments of any pole really are physics; a reliability bake or a fab cycle takes what it takes. The third move runs everything else through the lateralworks challenge process, section 03, and cuts where the leverage is, because most segments only look like physics. Cut in big chunks: a ten percent trim disappears into noise, while removing a whole segment of the pole changes the project. But don't let the tent fall down. The pole holds the tent up. Cuts that gut the product's quality floor or its structural scope buy schedule the customer pays for later. Every cut is bounded and reversible.

4. Refresh planning

The first three moves happen in weeks. The fourth never stops: refresh the schedule weekly at minimum, update what happened, and attempt a pull-in every time: find a way to move the predicted finish earlier, even by days. Teams that slip stay in the room until they have found a way to recover the loss; teams that hold schedule push for a further pull-in anyway. lateralworks calls the surplus banking time: pull in before you slip, and later slips land on the bank instead of the date [11]. Working to the early schedule, doing it now rather than at the last minute, is a core practice of the fast teams lateralworks has studied since the early 1990s [7]. Refresh planning is what makes that behavior mechanical instead of heroic.

03

Practice

Challenge the assumptions

Durations look like facts. Most are conclusions resting on assumptions, and the assumptions were never said out loud. The challenge process is the machinery lateralworks uses to surface those assumptions, test whether the reasons behind them still hold, and convert the broken ones into schedule. This is where the big chunks come from.

The challenge process

The process runs left to right in Figure 2. Pick the area of focus, the long pole segment you want to improve. List the assumptions underneath it: what are we treating as true? Select the ones worth interrogating and ask why. Why do we think this? What are the reasons? Are the reasons still valid? Some assumptions survive: still valid, keep as is. Some resolve instantly: if we can do it now, do it; if we can cut it now, remove it. The rest go to ideas, how it could be done differently, and the ideas are harvested, treated into workable concepts, and acted on. Then repeat, because every acted-on idea exposes the next assumption underneath.

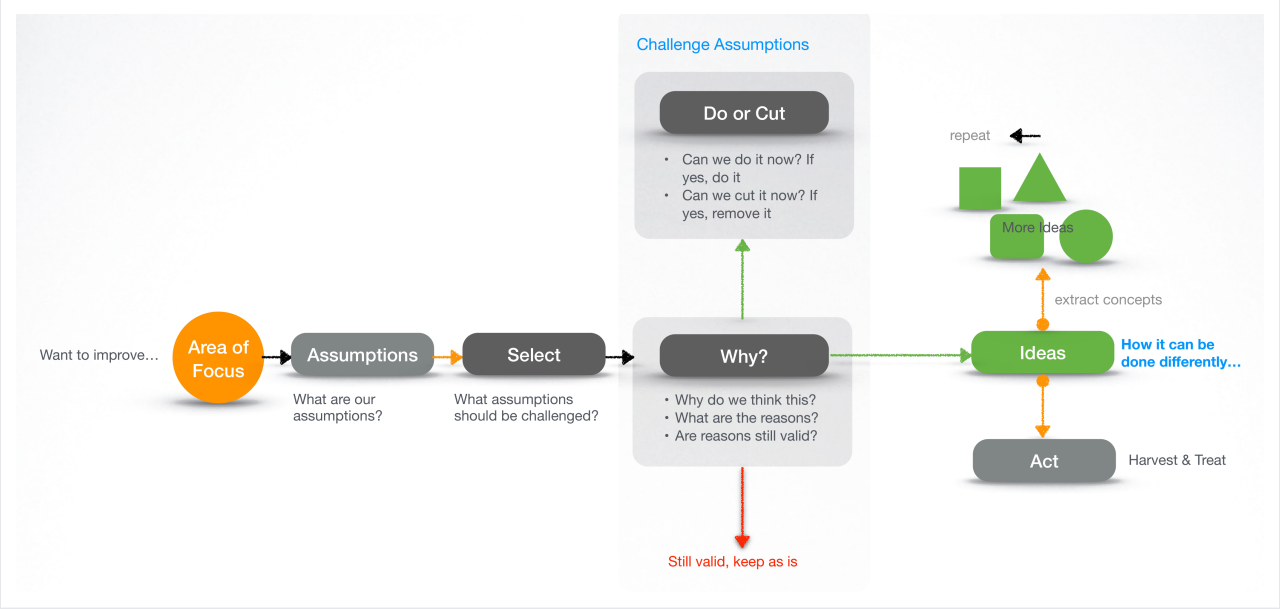

Figure 2. The challenge process: surface the assumptions under the area of focus, select and interrogate the ones that matter, keep what survives, and turn the rest into acted-on ideas.

Say the assumption out loud

Assumptions hide inside plans as unexamined defaults, so the workshop forces them into four sentence forms. **It is essential that…** (it is essential that first qualification meets the full published specification). **…dominates our thinking** (technical completeness dominates our thinking, over schedule). **We must avoid…** (we must avoid starting long-lead fabrication work before design freeze). **We must stay within…** (we must stay within current headcount). The phrasing matters: a belief written as a sentence acquires an owner and a test. A belief left as a default acquires a schedule.

Challenge for leverage

Not every assumption earns workshop time. The filter is pull-in leverage: relax the assumption in your head and watch the long pole. If it moves, challenge the assumption. If it does not, keep walking. On a recent memory-semiconductor qualification program, relaxing “first qualification must meet the full specification” shrank the entire test matrix to the feature set the lead customer actually exercises, weeks off the pole. Relaxing “company X is the first customer” moved nothing. Same register, opposite verdicts [8].

The big chunks on that program all came from challenged assumptions, not harder work: a full-mask respin rescoped to qualification-blocking fixes only; foundry cycle times treated as negotiable rather than quoted, hot lots and expedites bought with a ring-fenced schedule budget; long-lead mask and tooling work pre-staged in parallel with design close; the test program built concurrently against a frozen interface instead of serially after silicon. Each cut was bounded by an explicit quality floor, the tent stayed up, and each one removed a segment of the pole rather than shaving it.

The pattern repeats across engagements [17]. A system-on-chip team that needed six months out of its schedule used the challenge process to generate five acceleration alternatives worth between two and twenty weeks each, built a decision model to pick among them, and took the twenty-week option: a five-month pull-in. A financial-services software program running four months late spent two hours listing its current thinking, picked the five areas blocking the target date, and challenged each one; four of the five yielded a better way, and one more hour rewrote the schedule around them. Five hours of work bought a three-month pull-in, and the product shipped within weeks of the original target [17].

Escape current thinking

Figure 3 shows why the process works when it works. The normal line of thought runs straight from problem to solution inside a channel walled by constraints, and the walls are built from assumptions. Challenging the assumptions is how a team climbs out of the channel, finds the alternative route, and comes back down with a solution the straight line could never reach. This is applied lateral thinking, in de Bono's original sense of deliberately escaping established patterns rather than digging the same hole deeper [12].

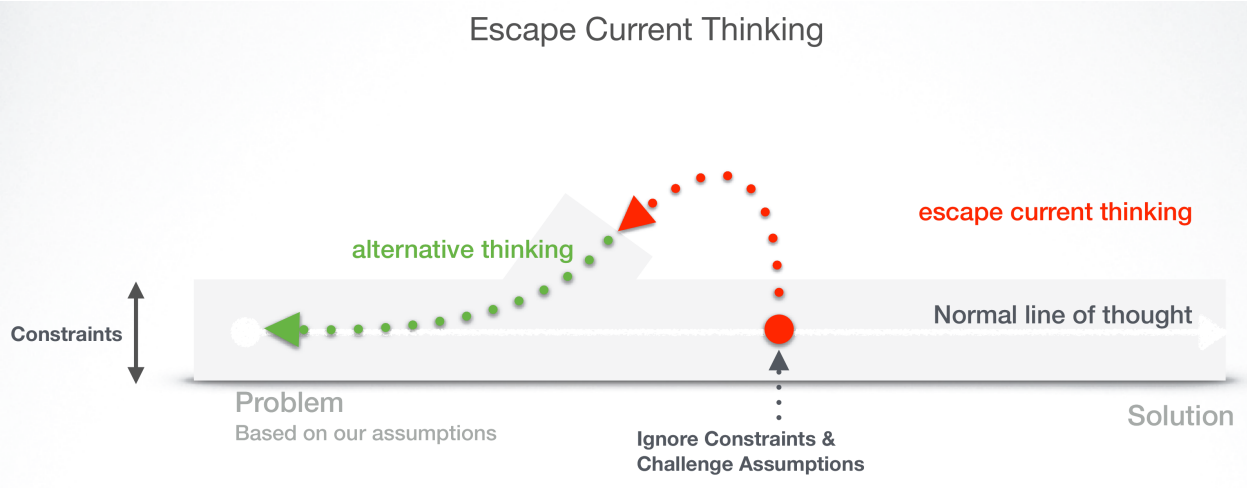

Figure 3. Escape current thinking: the normal line of thought runs inside constraint walls built from assumptions. Challenging assumptions is the way out.

The same machinery, pointed at speed

The challenge process has close relatives in the published literature. RAND's assumption-based planning identifies the load-bearing assumptions under a plan, asks which are vulnerable inside the planning horizon, and posts signposts and hedges against their failure [13, 14]. Klein's premortem asks a team to assume the project has already failed and explain why; the prospective-hindsight framing it exploits raises the ability to identify reasons for outcomes by roughly 30 percent [15, 16]. Both converge on the same diagnosis lateralworks reached in the field: plans fail through assumptions nobody examined.

The difference is the direction of fire. Assumption-based planning and the premortem are defensive: they protect a plan from surprise. The challenge process attacks a duration. The question changes from "what could make us fail?" to "which belief, if we dropped it, would make us faster?"

The method

Rule one of the long pole

“Cut down the pole in big chunks, but don’t let the tent fall down.”

The long pole method

lateralworks, FTTM best practices

04

Innovation

Plan what you can’t know

The strongest objection to the long pole method arrives on innovation programs: you cannot plan what you do not know. Experiments have to return results before the plan can firm up, and, most uncomfortable of all, the honest pole often gets longer as the team learns. This section takes the objection seriously and shows how learning-cycle estimation plans the unknown anyway [19].

Ninety percent is known

Most of an innovation project is not innovation. Smith and Reinertsen observed that about 90 percent of a typical development project is known work, and less than 10 percent is genuinely new [20]. The discipline is to isolate the 10 percent, staff it with the right skills, and give it a wide window, while the other 90 percent runs on conventional planning. The failure mode is the mirror image: teams become transfixed by the 10 percent, wait for the breakthrough, and finish late because the easy 90 percent quietly slipped behind them [19]. The approach scales to extreme unknowns. On one advanced-node semiconductor program, a new manufacturing process, new tools, a new product design, new design kits, and a new fab were all being developed simultaneously, and the program was still planned, predicted, and managed to an accelerated pace [19].

Estimate in learning cycles

The 10 percent is planned in learning cycles: plan, do, check, act. Nobody knows how long the solution will take, but most professionals can say whether the problem is easy, hard, or difficult, and that judgment converts to a cycle count: easy problems resolve in one to three cycles, hard ones in three to five, difficult ones in five to seven [11, 19]. Assign an average duration per cycle and the unknown acquires a defensible number. A problem judged hard, on the easy side, at three cycles of fifteen days is a 45-day pole segment (Figure 4). These are quantified guesses, but a plan is nothing else anyway, and quantified guesses beat hope [19].

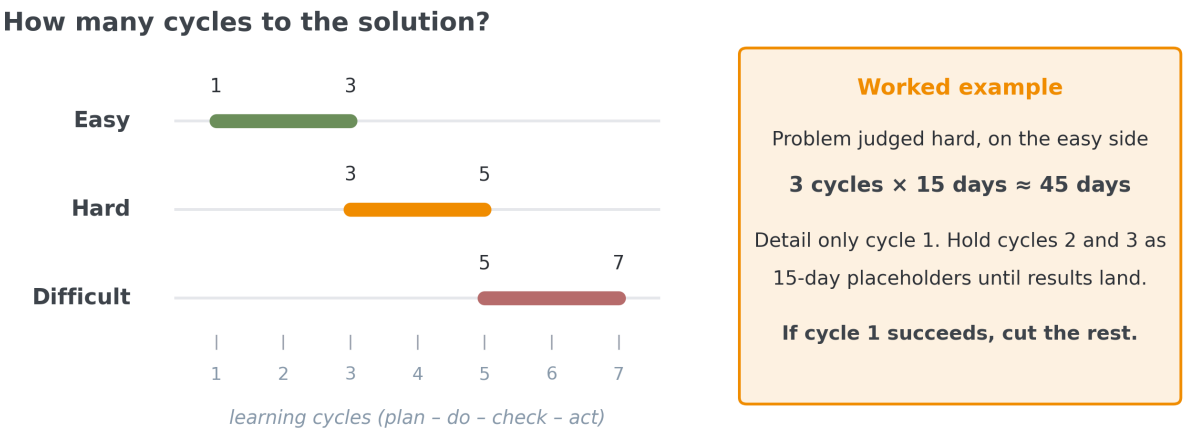

Figure 4. Learning-cycle estimation: judge the difficulty, convert it to a cycle count, assign an average cycle duration, and the unknown becomes a plannable pole segment. Detail only the cycle in front of you.

Then plan only what you know. Cycle one gets broken into real tasks; cycles two and three stay as summary placeholders until cycle one returns results, and each cycle's results decide the shape of the next. If cycle one succeeds, the remaining cycles get cut, which makes optimizing the cycle count one of the sharpest acceleration levers an innovation program has [19].

The pole that grows as you learn

Here is the contradiction. Sections 02 and 03 are about cutting the pole; on invention work the pole often grows instead, because each experiment surfaces work that was always there. A pole that lengthens as results land is reality arriving on schedule. The alternative is worse, and it is the norm: schedules with zero

learning cycles, mandated by right-first-time policy, on problems the team has never once solved right the first time. Ask a team “have you ever done it right the first time?” and the answer is no. They plan that way because adding the real cycles would show a large slip, and because they fear planned cycles become a self-fulfilling prophecy. Hard problems are rarely solved on the first pass whether the plan admits it or not; the only choice is whether the gap shows now, while there is time to rescope, or at the end, when there is not [19].

The planning process absorbs the growth three ways. The cycle count states the real gap up front, which creates near-term urgency and forces scoping decisions while they are still cheap. Weekly refresh re-syncs the pole with actual learning progress, so growth arrives in weekly increments rather than a cliff. And the challenge process still cuts, except the target is the cycle count itself: can cycle one be designed to answer more questions, can two experiments run in parallel, is cycle three still necessary after what cycle one returned? The innovation pole is cut in cycles, not days.

> **Zero learning cycles.** A schedule with no learning cycles on a never-done-before problem is not optimistic. It is a slip announcement with a delivery date. Planning the cycles does not cause them; it prices them.

05

Evidence

Watch the wiggle

A schedule reviewed quarterly is an obituary. The wigglechart is the weekly plot of the schedule’s predicted finish against the target, and it is where the long pole method becomes visible: reality surfacing, challenge biting, refresh planning holding the line. One chart from a semiconductor fab program tells the whole story.

Reading the chart

Figure 5 plots, week by week from October 2009, the predicted date of first silicon at a new fab, the black line, against the committed target of January 10, 2012, the blue line [8]. Within weeks of the first honest refresh, the prediction spiked into 2014: the ugly baby, about two years past target, in public. Nothing had slipped. The truth had surfaced. The spike is step one of the method printed on real data, and the team responded with steps two and three: root-causing the pole's duration and challenging the assumptions under it pulled the prediction back into the target band by mid-2010.

Then eighteen months of step four. The line wiggles, that is the name, because an updated schedule moves every week; each slip shows, gets attacked in the room, and recovers within weeks. Notice the blue target line itself stepping down mid-chart: the team banked pull-ins and reset its commitment earlier. The endgame is the method in miniature: a final ten-day pull-in from reduced tool processing time completed on December 27, 2011, and first silicon started two weeks ahead of the original target [8].

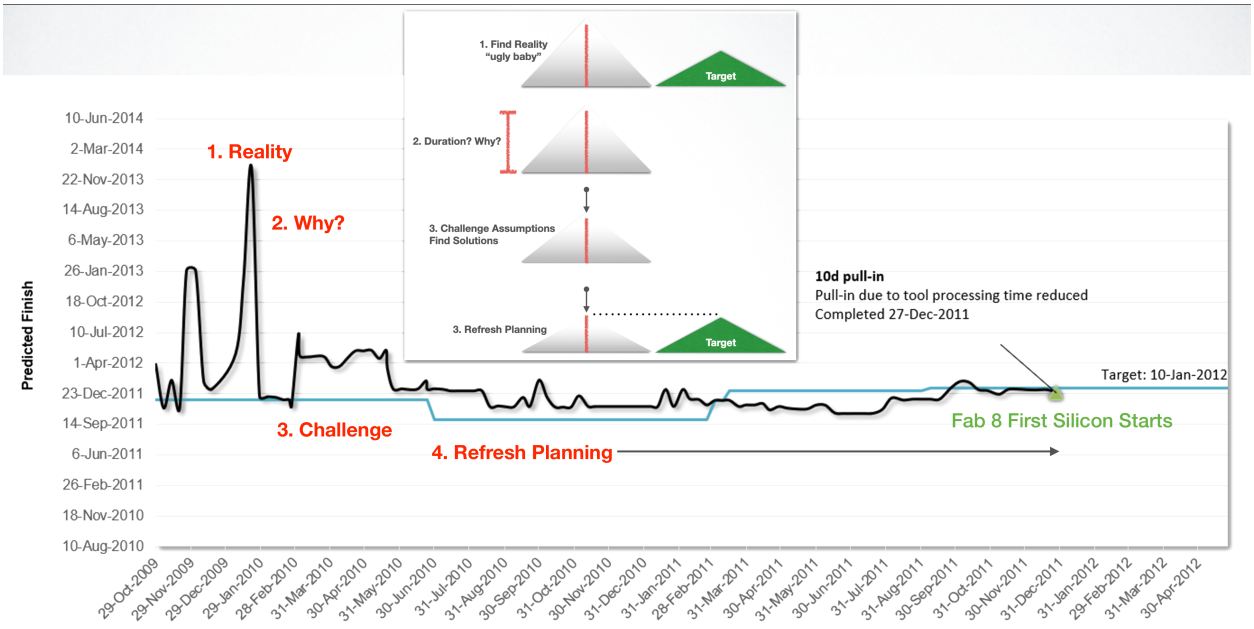

Figure 5. The wigglechart: weekly predicted finish (black) against target (blue) for first silicon at a semiconductor fab, October 2009 to early 2012. Reality surfaces as a two-year spike; challenge pulls it back; refresh planning holds it; the program finishes early. Method steps annotated in red.

A flat line is a lie

Most projects produce the opposite chart: a predicted finish that sits flat on the target for months and then cliffs at the end, when recovery is impossible. Edmonds catalogs the causes from lateralworks engagements [11]: schedules planned aggressively to please management; schedules never updated, so slips accumulate silently; multi-month activities whose owners report “on schedule” until the last possible week; work missing from the plan entirely; experiment-driven work planned with zero learning cycles, the fiction section 04 dismantles; and plain not telling the truth, because reporting a slip is punished. Each cause has a mechanical fix, honest planning, weekly refresh, one-to-five-day near-term tasks, owner-built schedules, learning-cycle estimates, and each fix depends on the same cultural one: a reported slip starts a recovery conversation, not a beating.

Read this way, the wigglechart is an early warning system, the thing every late project wishes it had bought in time [11]. A line that wiggles is a schedule being told the truth weekly, while the gap is still small enough to close. A line that never moves is a schedule that has stopped speaking.

06

Discipline
Peel the onion

Cutting the first pole is only the first layer. Underneath every long pole is the next one, and underneath the mechanics is the cultural inversion that makes fast teams fast: they assume success, and they plan like they mean it.

The next pole is already growing

Shorten the longest path and a new path becomes the driver. lateralworks calls working this sequence peeling the onion: put Team A on long pole one, and simultaneously put Team B on long pole two, on the explicit assumption that Team A will succeed. Repeat weekly through refresh planning, walking the schedule back one critical path at a time, until five teams are working acceleration in parallel [1]. The second and third paths deserve the attention for another reason: they often sit within ten days of float, ten days of slack, behind the critical path, close enough that one bad week makes them the new pole. Fast teams treat near-critical paths as critical.

Figure 6 shows what the onion looks like on a live program: the top seven critical paths on the memory-semiconductor qualification program of section 03, ranked by gap to a June 30, 2027 target [8]. CP1, the respin-one samples path, holds zero buffer and lands December 6. Behind it, six more poles hold between 13 and 34 days. Every path in the stack is late against the target, and any of them can take over the tent.

Figure 6. The onion on a live program: the top seven critical paths ranked by gap to target, with buffer held and nine-week trend for each. CP1 drives the date; CP2, slipping fastest, is the one to watch. Redrawn from a weekly fastProjectAI critical-path analysis [8].

Watching the next poles is measurable, not a matter of vigilance. Current buffer says where each path stands; buffer-burn rate, slip days per week over the last few weeks, says when it takes over. Divide buffer by burn rate and the paths rank themselves by time-to-overtake [18]. Read CP2 in Figure 6 that way: 13 days of buffer against 25 days of slip over nine weeks, 13 of them in the last week alone. At the recent rate CP2 overtakes CP1 within two weeks, and unlike CP1 it has no pull-in actions open. Moving attention to CP2 now, before it becomes the long pole, is the before-the-fact behavior that defines fast teams. Slow teams never look behind CP1; the first they hear of CP2 is the week it takes over, with no pull-in plan and the slip already baked in [18].

This analysis is a standard weekly output of fastProjectAI, the lateralworks scheduling platform: it surfaces the top critical paths, computes buffer-burn rates, ranks the paths by time-to-overtake, and flags the exact pattern in Figure 6, pull-in attention concentrated on a stabilized CP1 while the next path drifts unattended [18]. In a weekly schedule review it turns the silent slippers into Monday's conversation instead of next month's crisis.

Fourteen days from a new long pole

Figure 7 reduces the mechanics to three paths. CP2 runs 13 days behind CP1, so it holds 13 days of float. If CP1 pulls in fourteen days, CP2 is the new long pole. If CP2 slips fourteen days, same result. The distance between a managed project and an unmanaged one is that thin: two ordinary weeks, in either direction, hand the end date to a path nobody was managing. That is why peeling the onion shortens CP2 through CP5 in step with CP1 rather than waiting for the handoff.

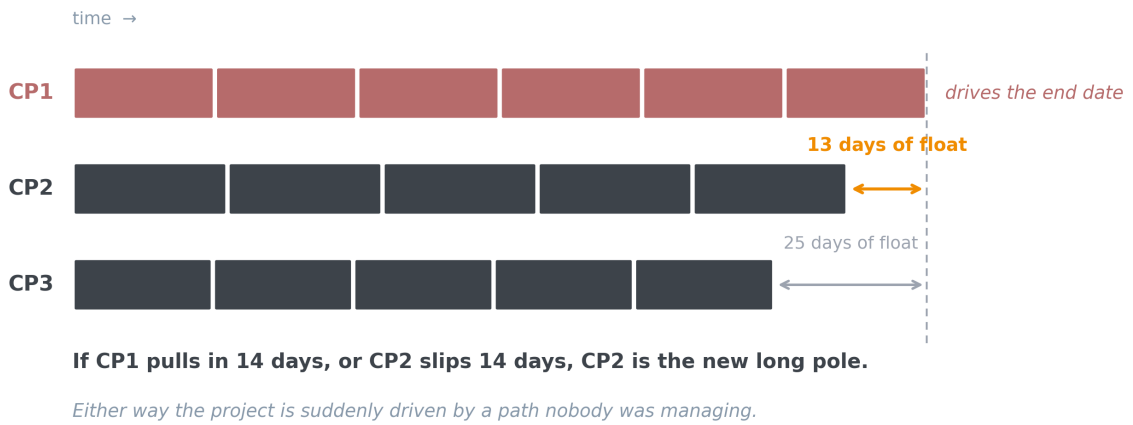

Figure 7. Fourteen days from a new long pole: CP2 holds 13 days of float behind CP1. One good week on CP1 or one bad week on CP2 makes CP2 the driver. Fast teams shorten both in step.

Assume success

Peeling the onion only works under an inverted cultural assumption. Slow projects hope CP1 slips: every pole behind it inherits the room, the owners quietly absorb it and start working to the later date, and the late-schedule mentality feeds itself. Fast projects assume CP1 is going to pull in, so every path behind it must accelerate to stay out of its way; the whole team is focused on acceleration and believes the project will get shorter, not longer [1, 18]. This is the behavioral engine behind working to the early schedule, the practice the FTTM research has found in fast teams for three decades [7]. It runs contrary to human nature, which is exactly why it needs a weekly mechanism rather than a poster.

Banked time beats buffers

Goldratt's critical chain reached the same diagnosis of schedule pathology, student syndrome, Parkinson, safety hidden inside every estimate, and prescribed pooling: strip the padding out of tasks, aggregate it into a project buffer, and manage the buffer to protect the committed date [5]. The long pole method agrees with the diagnosis and inverts the prescription: pull in before you slip, bank the time, and let later slips land on the bank instead of the date [11]. A buffer defends a commitment; a bank beats one. On FTTM programs the

target is a floor, and the wigglechart in Figure 5, where the team stepped its own target earlier mid-program, shows what the difference looks like on real data.

Two organizations

The whole method compresses into a contrast lateralworks sees on every engagement.

	Normal organization	Best organization
The schedule	An aggressive fiction built to please management	Honest, even when it shows a late date
The long pole	A reporting artifact everyone counts on growing	The weekly target of attack
Slips	Accumulate silently, surface at the cliff	Surface within a week, recovered in the room
Assumptions	Invisible defaults inherited as constraints	Written, ranked, and challenged for pull-in leverage
Other teams	Assumed to fail, so I have more time	Assumed to succeed, so I have less
The target	Defended with buffers and blame	Beaten with banked time

> **Running it this week.** Build the honest schedule and let it be ugly. Name the long pole and the gap to target. Break the next six weeks of pole work into one-to-five-day tasks. Run one challenge workshop on the largest segment. Then refresh the schedule next week, and every week after, and attempt a pull-in each time.

Speed is a weekly behavior directed at the one path that decides the date. Find the pole. Ask why it is that long. Challenge what holds it up, cut in big chunks, and keep the tent standing. Then come back next week and whittle. Teams that do this finish early on purpose. Teams that leave the pole alone have handed it the end date, and an unattended pole only grows.

A

Appendix

Critical path analysis, live

This appendix walks through one week of critical path analysis from a live memory-semiconductor qualification program, produced in fastProjectAI [8]. Two views: the top seven critical paths ranked against the target, and the schedule detail behind the first three. Figures A1 and A2 are redrawn from the program screenshots, with owner names removed; every number is as reported.

The top seven poles

Figure A1 shows the long pole and the six poles behind it. CP1, respin-one samples, lands December 6, 2027 against a June 30 target: 159 days late, zero buffer, 79 days of slip over nine weeks, and 14 more in the last week. It carries open pull-in actions and it is being managed. The rows beneath it are the watch list, each with its own nine-week trend, week-over-week change, buffer, and predicted finish.

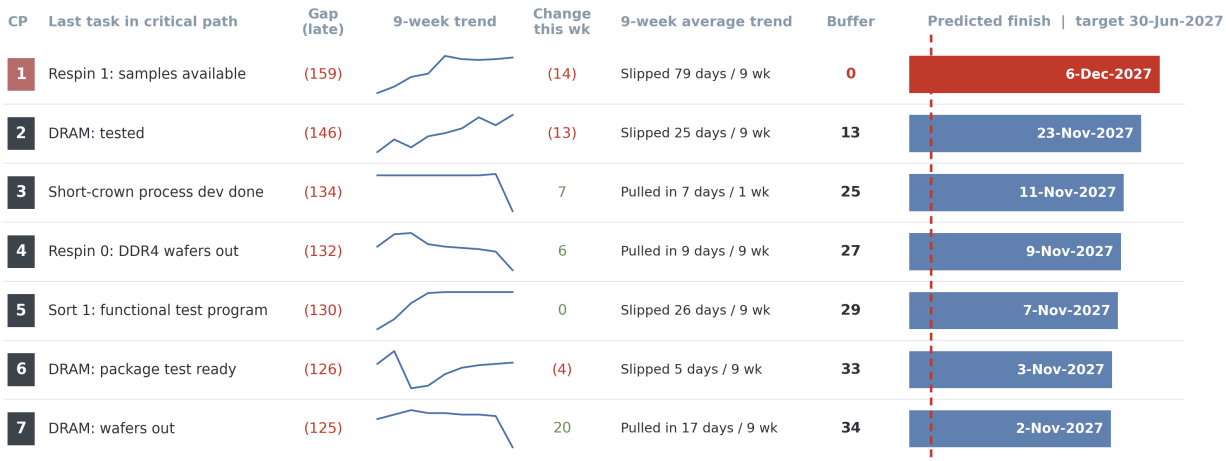

Figure A1. fastProjectAI weekly critical path analysis: the top seven paths ranked by gap to the June 30, 2027 target, with nine-week trend, change this week, average trend, buffer, and predicted finish. Redrawn from a live program screenshot [8].

The row that matters most is not CP1. CP2, the DRAM package-test path, has slipped 13 days in the past week and 25 days over the past nine weeks, and holds 13 days of buffer. It is likely to slip more unless attention moves to it now, before it becomes the long pole. That is the before-the-fact behavior characteristic of fast teams. Slow teams never see what is happening behind CP1; they meet these shorter poles for the first time as a crisis, the week one of them takes over the tent.

The schedule behind the poles

Figure A2 shows the same program's schedule detail for critical paths one through three. CP1 runs from sort development through the respin-zero package and design validation chain. CP2 ends 13 days behind CP1, so it holds 13 days of buffer. If CP1 pulls in 14 days, CP2 becomes CP1. If CP2 slips 14 days, CP2 becomes CP1 and is driving the project. Two ordinary weeks in either direction change which chain of work owns the end date.

Which direction those weeks run is cultural. Slow projects hope CP1 slips, because everything behind it gets more time. Fast projects assume CP1 is going to pull in, so the paths behind it must pull in too; everyone is focused on acceleration, and the team believes the project will get shorter, not longer. The same 14 days, opposite ends of the tent.

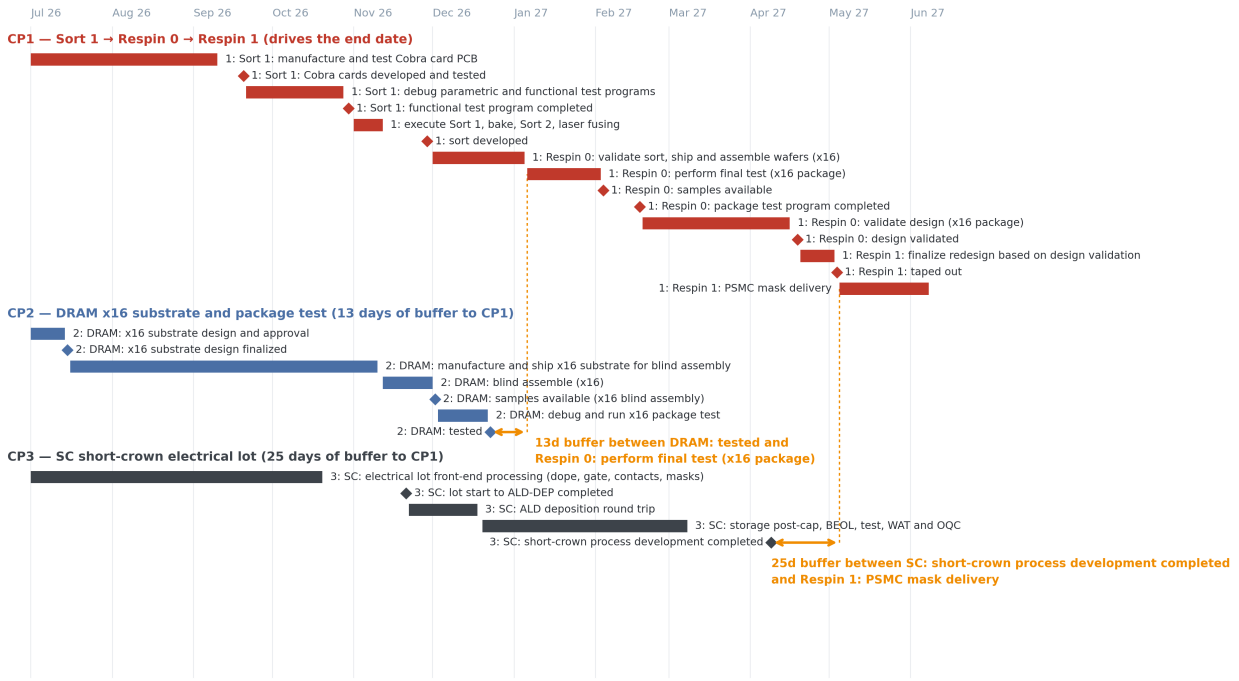

Figure A2. Schedule detail for CP1 through CP3, condensed from the fastProjectAI Gantt. The red chain drives the date. Each feeder path carries its buffer at the point it merges into CP1: 13 days between DRAM tested and Respin 0 final test, 25 days between short-crown completion and Respin 1 mask delivery. Redrawn from a live program screenshot [8].

Sources

References

[1] Mitchell, N. “The Long Pole.” lateralworks, January 2019. https://lateralworks.com/docs/the-long-pole

[2] Kelley, J. E., and Walker, M. R. “Critical-Path Planning and Scheduling.” Proceedings of the Eastern Joint Computer Conference, 1959.

[3] Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W. “Application of a Technique for Research and Development Program Evaluation.” Operations Research, Vol. 7, No. 5, 1959.

[4] Parkinson, C. N. “Parkinson’s Law.” The Economist, November 1955.

[5] Goldratt, E. M. Critical Chain. North River Press, 1997.

[6] Flyvbjerg, B., and Gardner, D. How Big Things Get Done. Currency, 2023.

[7] lateralworks. “10 best practices of fast teams.” Whitepaper, lateralworks FTTM Best Practices series. https://lateralworks.com/papers/10-best-practices-of-fast-teams.pdf

[8] lateralworks. Internal engagement data: semiconductor fab first-silicon program wigglechart (2009–2012) and memory-semiconductor qualification challenge register (2026).

[9] Chen, R. “Microspeak: The long pole.” The Old New Thing, Microsoft Developer Blogs, August 5, 2008. https://devblogs.microsoft.com/oldnewthing/20080805-00/?p=21373

[10] Kahneman, D., and Tversky, A. “Intuitive Prediction: Biases and Corrective Procedures.” TIMS Studies in Management Science, Vol. 12, 1979.

[11] Edmonds, M. “Crystal Ball or Early Warning System?” lateralworks, July 2019. https://lateralworks.com/docs/crystal-ball-or-early-warning-system

[12] de Bono, E. Lateral Thinking: Creativity Step by Step. Harper & Row, 1970.

[13] Dewar, J. A., Builder, C. H., Hix, W. M., and Levin, M. H. Assumption-Based Planning: A Planning Tool for Very Uncertain Times. RAND Corporation, MR-114, 1993.

[14] Dewar, J. A. Assumption-Based Planning: A Tool for Reducing Avoidable Surprises. Cambridge University Press, 2002.

[15] Klein, G. “Performing a Project Premortem.” Harvard Business Review, September 2007.

[16] Mitchell, D. J., Russo, J. E., and Pennington, N. “Back to the Future: Temporal Perspective in the Explanation of Events.” Journal of Behavioral Decision Making, Vol. 2, No. 1, 1989.

[17] Mitchell, N. “Breakthroughs by Challenging Assumptions.” lateralworks, January 2019. https://lateralworks.com/ideas/breakthroughs-by-challenging-assumptions

[18] Mitchell, N. “The critical path you’re not watching.” lateralworks briefing paper, May 2026. https://lateralworks.com/papers/the-critical-path-youre-not-watching.pdf

[19] Mitchell, N. “Learning Cycles.” lateralworks, January 2019. https://lateralworks.com/ideas/learning-cycles

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