TRAINING MODULE

Priority Recommendation

Five real levers. They do not simply add up.

A working module for engineers with 1–2 years on the floor. By the end you will be able to explain why five independently-real improvement numbers cannot just be summed — and simulate a combined cascade the honest way.

Grounded in Goldratt's Theory of Constraints — a line's output ceiling is always the slowest station, never a sum of independent gains

BEFORE WE START

What You Will Be Able to Do After This Module

1. Explain why five real, individually-valid improvement numbers cannot simply be added together.
2. Identify when two levers are targeting the SAME underlying seconds instead of independent ones.
3. Explain why the line's combined ceiling is always the slowest station, never a sum of separate gains.
4. Explain both baseline bugs this tool fixes: inconsistent “today” and Hidden Factory's NVA misclassification.
5. Recognize the two most common mistakes junior IEs make when they first stack multiple improvement ideas.

Five Real Numbers. Adding Them Is Still Wrong.

Each of these five levers already comes from a tool you've trained on — each number is real, on its own:

C2C Buffer	Bottleneck Exploit	Line Balance	Motion Waste (MTM)	Hidden Factory (NVA)
+4%	+6%	+5%	+3%	+7%

NAIVE SUM: 4% + 6% + 5% + 3% + 7% = 25% line output improvement. This number is almost certainly false.

Two things a plain ranked list hides: some levers target the SAME seconds at the SAME station (double-counted), and the line's real ceiling is always its SLOWEST station — improving a non-constraint doesn't raise output at all, no matter how real that station's own gain is.

THE CORE IDEA — A CALLBACK TO THE BOTTLENECK MODULE

The Line's Output Is a MIN, Never a SUM

Two DIFFERENT stations each get a real, valid improvement. Watch what happens to the LINE's output:

Stn 2

38s → 33s

Line Balance + Motion Waste applied

Stn 3

52s → 52s

UNCHANGED — the real constraint

Stn 5

41s → 35s

Line Balance + Motion Waste applied

Stations 2 and 5 both got real, credible improvements — 5s and 6s off their own cycle times. Station 3 never moved. The LINE's output is set entirely by Station 3, still at 52s. Real combined gain in actual line throughput: ZERO, until Station 3 itself is addressed.

WHERE THIS DISCIPLINE COMES FROM

You Already Learned This in the Bottleneck Module

“An hour saved at a non-bottleneck is a mirage” — Goldratt's line, from the very first module in this series. This tool is that principle applied at whole-suite scale: five separate tools, each honestly reporting a real local gain, don't automatically become five real slices of line-level throughput. Deming made the same point about systems generally — optimizing components independently rarely optimizes the system; the interactions between components are where the real answer lives.

This tool is the suite refusing to let its own five best tools mislead you by omission.

SYSTEMS THINKING

Goldratt · Deming

TWO SUBTLE BUGS, ALREADY FIXED IN THIS VERSION

Before Combining Anything, the Baseline Had to Be Fixed

1. INCONSISTENT BASELINE

C2C Buffer's own “today” already reflects real stoppage losses. Every other lever was silently comparing against the pure capacity ceiling instead — an easier, unfair baseline. Fixed: every lever now measures against ONE consistent, C2C-aware “today.”

2. HIDDEN FACTORY MISREAD

Used to flag anything with an ECRS-review tick as “NVA” — that tick means “could be simplified,” not “produces zero value.” Nearly every station showed ~100% NVA. Fixed: now built on SWCS's real Prep/Process/Park split — only Prep + Park genuinely counts.

Five Levers, Each Borrowed From a Tool You Know

Lever	Borrowed From
C2C Buffer	build_c2c_story.py's own micro-stoppage recovery model — the C2C module.
Bottleneck Exploit	build_bottleneck_story.py's own model — the Bottleneck / TOC module.
Line Balance	generate_line_balance.py's own ECRS-proposed capacity per station.
Motion Waste (MTM)	generate_mtm_investigation.py's own real EXCEED-element gaps.
Hidden Factory (NVA)	generate_swcs.py's own real Prep/Process/Park classification.

No new data or math is invented — this tool calls into the same functions those five tools already use.

A DELIBERATE EXCLUSION

Why Blocking & Starving Isn't Counted a Sixth Time

Blocking & Starving is a CONSEQUENCE VIEW of Levers 2 and 3 — a capacity-difference visualization — not an independent action a person can take.

REAL LEVER	“Reduce Station 3's cycle time by 6 seconds” — an action you can actually go do.
CONSEQUENCE VIEW	“Station 4 sits blocked 6 seconds/cycle because Station 3 is slow” — the SAME fact, seen from downstream.

Counting it as a 6th lever would credit the SAME 6 seconds of opportunity twice — once as “fix the bottleneck” and again as “fix the blocking it causes.” They are one opportunity, described from two angles.

Different Lenses on the Same Overlapping Seconds

Station 3 (today's constraint, 52s) gets THREE separate lever recommendations at once:

Bottleneck Exploit	**– 6s**	Remove NVA + high-C2C elements at the constraint
Motion Waste (MTM)	**– 4s**	Fix an EXCEED-flagged reach element at this station
Hidden Factory (NVA)	**– 3s**	Prep/Park time identified at this station

NAIVE SUM: 6+4+3 = 13s off a 52s station. Almost certainly overstated — the MTM gap and the NVA time likely overlap with what Exploit already removes.

CASCADE APPROACH: apply the levers as an ORDERED stack against the station's real 52s, capping at what's actually there — never letting three lenses double-spend the same seconds.

WORKED EXAMPLE 2 — TWO STATIONS, ONE CEILING

Real Local Gains, Zero Line-Level Gain — For Now

This is the exact scenario from Slide 4, walked through as a recommendation would present it:

Lever	Station	Local Claim	Real Line Impact
Line Balance	Stn 2	38s → 33s (−5s)	0% — Stn 2 was never the ceiling
Motion Waste	Stn 5	41s → 35s (−6s)	0% — Stn 5 was never the ceiling
Bottleneck Exploit	**Stn 3**	**52s → 46s (−6s)**	**+12% — THIS is the line's real ceiling**

A ranked list would show Line Balance and Motion Waste as “good ideas” and might rank them ABOVE Bottleneck Exploit by raw seconds saved. The cascade sequences them correctly: Bottleneck Exploit first, because it's the only one that moves the actual ceiling — the other two become valuable only AFTER the constraint shifts.

HOW THE CASCADE ACTUALLY COMBINES THEM

Simulated, Not Just Summed

1. Establish ONE consistent baseline — the C2C-aware “today,” for every lever.
2. Apply levers in sequence, exactly like LineSim.computeOutput would if you clicked them on one by one in the browser.
3. After each lever, recompute the WHOLE line's new bottleneck — it may have moved to a different station.
4. Cap any overlapping claim at the same station — never let two lenses double-spend the same seconds.
5. Report the REAL cumulative line-output gain — not a sum of five independent percentages.

BEFORE YOU GO TO THE FLOOR

Five Mistakes That Give Junior IEs Away

✗ Adding five improvement percentages together and presenting the sum as the expected line-output gain.

✗ Ranking levers by raw seconds saved instead of by whether they actually move the line's real ceiling.

✗ Claiming credit for the same seconds twice — once from Motion Waste, again from Hidden Factory, at the same station.

✗ Treating Blocking & Starving as an independent 6th opportunity instead of a view of levers already counted.

✗ Comparing every lever's “gain” against a different, inconsistent “today” baseline.

FROM PENCIL-AND-PAPER TO THE SOFTWARE

What generate_priority_recommendation.py Does For You

Simulates the real combined cascade automatically — across your whole line, every time it's run.

1	**Pulls each lever from its own validated tool**	No new math invented — calls directly into the same functions the C2C, Bottleneck, Line Balance, MTM, and SWCS tools already use.
2	**Establishes one consistent baseline**	Every lever measured against the same C2C-aware “today” — Bug Fix 1, applied automatically.
3	**Uses SWCS's real Prep/Process/Park split**	Hidden Factory reads genuine NVA time, not an ECRS-review checkbox — Bug Fix 2, applied automatically.
4	**Simulates the cascade, not just a ranked list**	Applies levers in sequence, recomputes the bottleneck after each, caps overlapping claims — the honest combined answer.

KEY TAKEAWAYS

Recap — Say This Back In Your Own Words

- Five real, individually-valid improvement numbers still cannot simply be added together.
- The line's output ceiling is always the SLOWEST station — improving a non-constraint doesn't raise output, no matter how real the local gain is.
- Multiple levers can target the same overlapping seconds at one station — that's double-counting, not five independent wins.
- Blocking & Starving is a view of the bottleneck, not a sixth opportunity — counting it separately double-counts the same seconds again.

TALK IT THROUGH — 3 QUESTIONS FOR YOUR NEXT LINE WALK

1. If your own line had five improvement ideas on the table right now, could you say — honestly — which ones actually move the real ceiling?
2. Have you ever seen a project plan that just summed several independent “% improvement” numbers? What would the real combined number likely have been?
3. Pick two levers on your own line that might be looking at the same overlapping seconds — how would you actually tell them apart?