TRAINING MODULE

Precedence Diagram Debate Engine

Not a flowchart drawer — a rule-driven design partner

A working module for engineers with 1–2 years on the floor. By the end you will be able to check a line's sequence against real precedence rules, and know exactly what a fixture investment does to that sequence.

Grounded in the classical Precedence Diagramming Method and Assembly Line Balancing's precedence-constraint tradition

BEFORE WE START

What You Will Be Able to Do After This Module

1. Explain what a precedence constraint is, and why some line orders are simply invalid, not just suboptimal.
2. Explain why the rule table lives as editable DATA, not hard-coded logic — and why that matters across industries.
3. Read a floating-tag debate correctly — trade-offs presented, not a single forced answer.
4. Explain how freeing a vise-blocked hand can change a station's place in the sequence, not just its cycle time.
5. Recognize the two most common mistakes junior IEs make when they first review a line's sequence.

Precedence Is a Hard Constraint, Not a Preference

Line Balance (a module you've already trained on) asks “how do we distribute work efficiently?” This tool asks a stricter, earlier question:

EFFICIENCY QUESTION

“Given a valid order, how do we balance the work across stations?” — Line Balance's job.

VALIDITY QUESTION

“Is this order even ALLOWED?” Packing before Quality Check isn't inefficient — it's a defect waiting to ship.

This tool checks validity FIRST — the same way you'd never balance work across a sequence that shipped defects by design.

THE CORE IDEA

Rules Live in a Table, Not in Python

“Packing must be last” isn't an if-statement buried in code — it's a row in an editable table:

Tag	Rule	Type
packing	Must be the LAST station in sequence	Hard constraint
quality	Must PRECEDE packing	Hard constraint
labeling	Floats between assembly and packing	Flexible — debated

WHY THIS MATTERS

A helmet line and a PCB line have completely different valid sequences — but the ENGINE checking them stays the same code. Only TAG_RULES and TAG_KEYWORDS change, per industry, with zero code edits. This is the same design philosophy as the ERB module's editable question library.

WHERE THIS METHOD COMES FROM

Every Balancing Algorithm Assumes a Valid Graph First

Classical line-balancing methods — including the Largest Candidate Rule your Line Balance module uses — all assume the task ORDER going in is already valid. A precedence diagram (a DAG of tasks and their required order) is the formal structure that balancing runs on top of. This tool builds and validates that diagram automatically from your real station data, instead of assuming someone already checked it by hand.

Get the precedence wrong, and every balancing number built on top of it is balancing an invalid sequence.

PRECEDENCE DIAGRAM METHOD

Classical Assembly Line Balancing tradition

THE METHOD, IN ONE PICTURE

Three Auto-Detections, Before Any Rule Is Checked

1

AUTO-TAG

packing / quality / labeling / cleaning / assembly / prep — from the process name, no manual tagging.

2

HAND KINEMATICS

Reuses the Bilateral Splitting Engine's own verb logic — the same list the Ergo Risk Map module uses.

3

TOUCH vs NO-TOUCH

Physically touches the product, or a camera/visual inspection — from the same step text.

Nothing here is manually flagged — all three come from the same real Job-Method step text you already recorded.

1

STEP 1

Check the Hard Rules

Every hard constraint gets checked against the real, current sequence — pass or fail, explicit either way.

A FAIL here isn't a suggestion. It's a real sequencing defect that needs fixing.

A REAL 7-STATION LINE, CHECKED

Sequence: Assembly (Stn 1–4) → Labeling (Stn 5) → Quality (Stn 6) → Packing (Stn 7)

✓ **Packing must be LAST**

Stn 7 is Packing, and it's last

✓ **Quality must precede Packing**

Stn 6 (Quality) is before Stn 7 (Packing)

✓ **Labeling floats between Assembly and Packing**

Stn 5 sits after Assembly (Stn 4), before Packing (Stn 7)

IF QC WERE INSTEAD AT STN 3 (before assembly finishes): FAIL — flagged as "QC precedes final assembly steps at Stn 4; may miss downstream-introduced defects." Not a style note — a real gap.

2

STEP 2

Debate the Floating Tags

A floating tag doesn't get a single forced answer — both valid options are printed, with their real trade-off.

The current placement is checked against the requirement — satisfied either way, here.

LABELING — TWO VALID SLOTS, BOTH DEBATED

OPTION A — Right after Assembly (current)

+ Catches labeling defects before further handling.

− Label exposed to damage risk during subsequent packing handling.

OPTION B — Right before Packing

+ Minimizes damage risk, closest to shipment.

− A labeling error found late means more accumulated rework.

CURRENT PLACEMENT (Option A) already satisfies the “floats between” requirement. The debate isn't telling you to move it — it's making sure you chose it on purpose.

STEP 3 — FIXTURE INVESTMENT, TIED BACK TO SEQUENCE

Freeing a Vise Hand Can Move More Than Just CT

1. Station 3's LH is dead-blocked as a static vise — same verb-logic detection as the Ergo Risk Map module.
2. A locating fixture could hold the part instead of the hand — freeing LH for real work.
3. Station 3's CT could drop: 46s → 38s.
4. That CT change can shift WHICH station is the line's actual bottleneck — a Line Balance / Bottleneck consequence, not just an ergonomics win.
5. The precedence debate re-checks: does the new CT change where this station makes sense in a takt-bound resequencing?

This is why the fixture recommendation lives INSIDE the precedence debate, not as a separate ergonomics-only note.

STEP 4 — THE PROCESS–PRODUCT MATRIX

Which Station Touches Which Component — and Why That Gap Matters

Component “bracket-X” is touched at Station 2 (install) AND Station 6 (QC re-check) — with Stations 3, 4, 5 never touching it at all.

Stn 2	Stn 3	Stn 4	Stn 5	Stn 6
install	—	—	—	**re-check**

FLAGGED FOR CFT REVIEW

Non-adjacent component touch: why does bracket-X need re-touching at Station 6 instead of being fully verified at Station 2, adjacent to its install? Possible redundant handling, or a real process gap worth a second look.

STEP 5 — PFMEA CROSS-CHECK, WRITTEN AS A CFT LOG

Real Distance, Real Failure Mode, Real Discussion

PFMEA: “Component missing/double-loaded” at Stn 4, AP=HIGH. Quality gate is at Stn 6 — 2 stations of distance to travel before it's caught.

IE	Flags the 2-station gap between the failure point and the quality gate.
Quality/PFMEA Owner	Confirms AP=HIGH — this gap carries real risk, not a formality.
Production/Line Owner	Notes moving QC to Stn 5 would require re-sequencing labeling.
Maintenance/Tooling	Offers a cheaper option: an in-line sensor at Stn 4 itself, no resequencing needed.

OUTCOME: In-line sensor at Stn 4 approved. OWNER: Maintenance/Tooling. STATUS: Pilot scheduled — real minutes, not a single engine's verdict.

BEFORE YOU GO TO THE FLOOR

Five Mistakes That Give Junior IEs Away

- ✗ Treating a floating-tag debate as a demand to move the station — it's a trade-off presentation, not an instruction.
- ✗ Fixing a hard-rule FAIL with a workaround instead of actually resequencing — the constraint doesn't go away by ignoring it.
- ✗ Approving a fixture purely for ergonomics without re-checking what the resulting CT change does to the sequence.
- ✗ Treating a non-adjacent component touch as automatically wrong — it's flagged for review, not automatically a defect.
- ✗ Skipping the CFT log format and writing a single-voice verdict instead — losing the real cross-functional trade-offs that were actually raised.

FROM PENCIL-AND-PAPER TO THE SOFTWARE

What generate_precedence_debate.py Does For You

Every layer on this module, run automatically across your real line sequence.

1	**Auto-tags, auto-derives, auto-detects**	Station tags, hand kinematics, and touch/no-touch — all from real Job-Method text, zero manual flagging.
2	**Validates against TAG_RULES**	Every hard constraint checked, pass or fail, explicit — and every floating tag debated with real trade-offs.
3	**Builds the Process–Product Matrix**	From Components_Used or extracted part nouns — flags non-adjacent touches for CFT review automatically.
4	**Cross-checks PFMEA against real gate distance**	Real failure mode, real AP, real current control — not a generic rule of thumb.
5	**Writes the CFT Discussion Log**	Four-role exchange, ending in Outcome / Owner / Status — real meeting minutes, not one engine's opinion.

KEY TAKEAWAYS

Recap — Say This Back In Your Own Words

- Precedence is a hard constraint — some sequences are invalid, not just inefficient, and that's checked before any balancing math runs.
- Rules live in an editable table, not code — the same engine works across any industry by changing data, not logic.
- A floating-tag debate presents real trade-offs — it never forces a single answer where more than one is genuinely valid.
- Freeing a vise-blocked hand can change which station is the line's actual bottleneck — fixture decisions live inside the sequence question, not beside it.

TALK IT THROUGH — 3 QUESTIONS FOR YOUR NEXT LINE WALK

1. Walk your own line's sequence — can you name which stations carry a hard precedence rule, and which are genuinely floating?
2. Find a station where one hand is locked as a vise — if you freed it, would the resulting CT change actually shift the bottleneck?
3. Pick a PFMEA finding on your line — how many stations of distance sit between the failure point and the nearest real quality gate?