Production Intelligence: From Rules to Reasoning

4/8/2026, 2:40:16 PM

1/28 Production Intelligence: From Rules to Reasoning

Opening slide. The subtitle "From Rules to Reasoning" captures the whole arc: we currently run on hand-written if-then rules; the roadmap moves us toward systems that can reason about trade-offs mathematically and eventually learn from experience.

Glossary:

- **Operations Research (OR):** Using math to find the best answer to a decision problem — like a GPS finding the shortest route, but for factory decisions.
- **AI (in this context):** Systems that learn from data and improve over time, rather than following fixed rules.

2/28 The Question

The key reframe: "AI" is a huge umbrella. Some of the biggest wins come from classical math (optimization), not machine learning. The spectrum runs from "calculator-level math that gives perfect answers" to "systems that learn by trial and error."

Glossary:

- **Deterministic optimization:** Like solving a Sudoku — there's one best answer and the computer finds it. No guessing, no learning needed.
- **Machine learning:** Like teaching a dog new tricks — you show it lots of examples and it figures out the pattern. Needs data, can be wrong sometimes, but handles messy real-world situations.
- **ROI (Return on Investment):** How much value you get back for every dollar spent.

3/28 Where We Are Today

Acknowledge what we have — this is not a "everything is broken" pitch. The gap is that each rule was added to fix one specific problem, without seeing the whole picture. Specific examples if asked:

- **Batching** is purely manual: workers create batches, add items one by one, system validates eligibility (correct status, branch, no duplicates). No auto-sizing, nesting, or grouping optimization. 5 batch types: geometry, roll-tracking, cut-and-sew outsourcing, master box, sticker/postcard.
- AIQC still in data-collection phase (2 garment types, test device disabled in code).
- OEE tracking field exists in Monitor module but is hardcoded to 0.

Glossary:

- **Kaizen:** Japanese for "continuous improvement" — small, frequent changes rather than big overhauls. Our ops team does this regularly.
- **Greedy optimization:** Like packing a suitcase by always putting the biggest item in first — fast and decent, but might miss a better arrangement. Our routing uses this approach with 14 scoring factors.
- **ML demand model:** Our ML team has built a demand prediction model that the Inventory team is already using to help Purchasing Managers make better stock decisions. This is a real asset — for Level 2, we can extend this existing model to production planning (staffing, pre-staging) rather than building from scratch.
- **Local optimum:** Imagine hiking in fog — you reach the top of a hill and think it's the highest point, but you can't see the mountain behind it. Each rule we added found "the nearest hilltop" without seeing the full landscape.
- **Global optimum:** The actual highest peak — what a solver finds by considering the whole landscape at once.

4/28 The Science Behind the Problem

This is the "science says so" slide. Two textbooks, two key insights: (1) there are exact formulas for how queues behave, and (2) our specific scheduling problem is provably too complex for any rule set to guarantee optimal solutions.

Glossary:

- **Little's Law (CT = WIP / TH):** If you have 100 orders on the floor and finish 10 per hour, each order takes 10 hours. Want faster? Either finish more per hour OR have fewer orders on the floor. It's that simple — and that powerful.
- **VUT Equation:** Queue time = Variability × Utilization × process Time. Three knobs you can turn. The key insight: utilization (how busy machines are) has an *explosive* effect — the relationship is a hockey stick, not a straight line. Specifically, the U factor is u/(1−u), so going from 90% to 95% roughly doubles queue time, and 95% to 97% nearly doubles it again.
- **WIP (Work In Process):** All the orders currently on the production floor — not yet shipped. More WIP = longer wait times, always.
- **NP-hard:** A math term meaning "no algorithm can guarantee the optimal solution in reasonable time" for large instances. For our scheduling problem with parallel machines and setup times, the combinations grow astronomically. That's why we need solvers — software that uses clever mathematical tricks (branch-and-bound, constraint propagation) to find near-perfect answers without checking every possibility. Note: small instances (30 jobs, 5 machines) are solvable by modern solvers; the challenge is at production scale with hundreds of jobs and continuous re-planning.
- **Dispatching rule:** A simple rule like "do the most urgent order first" (EDD) or "do the shortest job first" (SPT). Each is provably optimal for exactly one simple problem class but suboptimal for our complex, multi-objective environment. Our Shoptool uses a combination of these.

5/28 The Utilization Trap

This is the "aha" moment for most leaders. The curve is not intuitive — people assume that going from 90% to 95% busy is a small step. In reality, it's catastrophic for wait times.

Glossary:

- **Utilization:** How much of the time a machine (or person) is busy. 90% = busy 9 out of every 10 minutes.
- **Why the hockey stick?** Think of a highway. At 50% capacity, traffic flows smoothly. At 90%, it's getting tight but moving. At 95%, one person braking causes a 20-minute jam. Same physics applies to production lines — the math is identical.
- **The 85% sweet spot:** Below ~85%, you get good throughput with reasonable wait times. Above that, every percentage point costs disproportionately more in delays. If you genuinely need more throughput than your machines can physically produce, you need more machines — no algorithm changes that. But "95% utilized" often includes hidden waste: changeover time, poor load balancing across machines, rework. SMED and better scheduling can recover that waste, effectively moving you LEFT on this curve (e.g., from 95% back to 85%) without losing throughput. That's the real win — not replacing capacity, but recovering the capacity you're already paying for.

6/28 The Roadmap

This is the core slide. The five levels form a maturity model — like going from a bicycle to a car to a self-driving car. Each upgrade is useful on its own. Level 0 is new and deliberate: Factory Physics and Lean both argue that process stabilization must precede process optimization. You can't optimize a system you can't measure, and a solver applied to a high-variability process yields far less than the same solver applied to a stable one.

Glossary:

- **Lean Foundations (Level 0):** Before you can optimize, you need to know your current state. OEE (Overall Equipment Effectiveness) tells you how much of your theoretical capacity you're actually using. SMED (Single Minute Exchange of Die) is a lean technique for slashing setup/changeover times — you separate steps that require the machine to be stopped from steps you can do while it's still running, then eliminate and simplify. Typical result: 50-90% reduction in changeover time with minimal spend. This directly attacks the variability that drives queue times (the V in VUT). CONWIP caps WIP so Little's Law works for you, not against you. This is not "extra overhead before the real work" — this IS the highest-ROI work.
- **Deterministic OR (Level 1):** Math that finds the best answer when you know all the inputs. Like a GPS with perfect traffic data — it gives you THE shortest route.
- **Machine Learning (Level 2):** Software that spots patterns in historical data to predict the future. Like weather forecasting — not perfect, but much better than guessing.
- **Simulation + RL (Level 3):** Building a virtual copy of the factory (digital twin), then letting an AI "play" millions of scenarios to learn the best strategies — like how AlphaGo learned to play Go by playing against itself millions of times.
- **Autonomous System (Level 4):** All layers working together automatically — the system senses, predicts, decides, acts, and explains, with humans supervising rather than operating.

7/28 Level 0: Measure & Stabilize

This slide draws a clear line between what a software team delivers and what it recommends. The software team builds the measurement and control infrastructure; the ops team runs the kaizen events. Both are necessary — you can't do SMED without data on current setup times, and you can't do CONWIP without system-level WIP control. This is a joint effort, and the slide should be presented as such.

The shifting bottleneck problem (critical for POD): In print-on-demand, the bottleneck is NOT a fixed station. It shifts with product mix: DTG-heavy day → printers are the constraint; cut-and-sew surge → sewing stations; mug spike → sublimation presses. Classical TOC assumes a stable bottleneck ("find THE constraint"). That framing doesn't work for POD. Instead:

- **Real-time identification:** The bottleneck is wherever WIP is accumulating RIGHT NOW. Instrumenting queue depths at every station makes this visible in real time. This is a software deliverable.
- **Statistical bottleneck:** The station with the highest average utilization over weeks/months is the "statistical bottleneck" — useful for long-term capacity planning (where to invest, where to focus SMED). But for daily operations, you need the real-time view.
- **CONWIP handles shifting bottlenecks naturally.** This is one of CONWIP's key advantages (Factory Physics is explicit about this): WIP automatically pools before whatever station is currently the constraint. No reconfiguration needed when the bottleneck shifts. Kanban fails here because cards are tied to specific stations/parts.
- **OEE must be demand-adjusted.** In POD, a machine idle because there are no orders is NOT an availability loss — it's the system working correctly. Standard OEE would penalize this. Compute OEE only during periods with pending work at that station.

What the software team owns:

1. **Live bottleneck tracking + OEE:** Instrument queue depths at every station. Build a dashboard showing where WIP is accumulating in real time. This tells ops "the constraint is HERE right now" — and historically "Station X is the constraint 60% of operating hours." The Monitor module's `oeePercentage` field (currently hardcoded to 0) can be repurposed for this, but needs demand-adjustment logic so idle-because-no-orders isn't counted as a loss. **OEE: track everywhere, target at bottleneck.** In standard industrial practice, OEE is calculated at ALL machines — it originated from TPM as a maintenance diagnostic. The key distinction is between *tracking* (everywhere, for maintenance visibility — "is this machine's problem availability, performance, or quality?") and *targeting* (only at the bottleneck, for throughput improvement). A non-bottleneck at 60% OEE might be perfectly healthy — it has spare capacity to absorb variability, which Factory Physics says it SHOULD

have. But if that 60% breaks down as 70% availability × 85% performance × 100% quality, maintenance still wants to know about the availability gap, because in POD this station could become the bottleneck tomorrow when product mix shifts. The mistake is setting a uniform "all stations must hit 85% OEE" target — that pushes non-bottlenecks into the utilization trap, building WIP without increasing system throughput.

2. **CONWIP control logic:** Implementing a WIP cap requires changes to the production control system — when WIP on a line hits the limit, the system holds new releases until a job exits. This is a software deliverable. Factory Physics proves CONWIP achieves the same throughput as push systems with 20-25% less WIP. Critical WIP: $W_0 = r\_b \times T_0$. In a shifting-bottleneck environment, CONWIP is especially valuable because it adapts automatically — WIP pools before whichever station is currently slowest.
3. **Instrumentation:** Adding timestamps at each production stage (order routed, printing started, printing finished, QC started, QC passed, packed, shipped) enables value stream analysis. Measuring setup/changeover durations lets ops teams target SMED kaizens on the worst offenders. This data also feeds Level 1 solver models and Level 2 ML predictions.

What the software team recommends (ops owns execution):

1. **SMED kaizens — but only where "setup" exists:** SMED was invented for press operations with physical die changes. In POD, this concept maps differently by technique:
   - **DTG printing: SMED applies to platen swaps.** Individual prints don't have changeovers (new file + garment swap is the process itself). But switching between platen types (small tee → large hoodie → baby onesie) IS a physical changeover: remove platen, mount new one, calibrate position. This is a legitimate SMED target. It also creates a batching question: how long to keep printing on one platen type before switching? That's the classic setup time vs. waiting time tradeoff (Factory Physics batching law). SMED reduces the changeover penalty, which shifts the optimal batch size smaller — enabling more frequent switches, shorter queues for all order types, and lower WIP. The optimal batch sequence is then a Level 1 scheduling problem (s_{ijk} in Pinedo's notation). Periodic printer maintenance (printhead cleaning, ink purging, nozzle checks) is NOT SMED — that's TPM (Total Productive Maintenance), addressed at Level 2 via predictive maintenance.
   - **Sublimation: SMED applies.** Switching between product types (mug fixtures → flat press → hat press) requires physical fixture changes. These are classic SMED targets.
   - **Cut-and-sew: SMED applies.** Pattern and material changes between product types.
   - **Embroidery: SMED applies.** Thread color changes, hoop swaps. The software team's role: instrument changeover durations by technique so ops knows WHERE setup reduction is worth

pursuing and WHERE it isn't. Don't recommend SMED as a blanket first move — it's irrelevant for the highest-volume technique (DTG).

2. **Value stream mapping:** Walking the floor, timing each stage, identifying where orders wait vs. where they're processed. Software provides the timestamp data; ops interprets it and identifies waste. In POD, this should be done for each major product-mix scenario (DTG-heavy, cut-and-sew-heavy, mixed) since the flow pattern changes.
3. **Standard work:** Documenting the current best method for each task. Software can embed standard work guidance in Shoptool, but the content comes from ops.

The VUT argument — adapted for POD (Factory Physics, Hopp & Spearman): The VUT equation says queue time = V × U × T. In traditional manufacturing, SMED is the primary tool for reducing V (process variability). But for DTG, where there are no meaningful setups, the variability that matters is: unplanned printer downtime, quality failures requiring reprints, and demand arrival variability. This means the V-reduction story for DTG is less about SMED (Level 0) and more about predictive maintenance and AIQC (Level 2). For non-DTG techniques (sublimation, cut-and-sew, embroidery), SMED remains a valid Level 0 tool. The software team's role across all techniques: CONWIP for WIP control, instrumentation for visibility, and data to enable ops improvements wherever they apply.

The CONWIP argument (Factory Physics): Our production is print-on-demand with infinite "part numbers" — kanban is impractical because it requires a card per part number. CONWIP (Constant WIP) is the right pull system: one WIP cap for the entire line. When a job exits, a new one enters. This IS a software deliverable — the production control system must enforce the cap. Factory Physics proves CONWIP achieves 20-25% less WIP at same throughput, and it's more robust to parameter errors. Crucially for POD: CONWIP adapts to shifting bottlenecks automatically, while kanban requires reconfiguration. Operate at 1.5-2× W_0.

Glossary:

- **Shifting bottleneck:** In POD, the constraint station changes with product mix. DTG-heavy days → printers constrain; cut-and-sew surges → sewing constrains. "Find THE bottleneck" is the wrong framing — instead, instrument queue depths to see where WIP accumulates in real time (current bottleneck) and track which station has highest average utilization over time (statistical bottleneck for capacity planning).
- **Demand-adjusted OEE:** Standard OEE = Availability × Performance × Quality. Track it at ALL stations (standard industrial practice — OEE originated from TPM as a maintenance diagnostic), but only TARGET improvement at the current bottleneck. A non-bottleneck at 60% OEE may be healthy — spare

capacity absorbs variability. But maintenance still wants the A×P×Q breakdown to spot problems before this station becomes the bottleneck (which in POD can happen tomorrow with a product-mix shift). In POD, must also be demand-adjusted: compute only during periods with pending work, since idle-because-no-orders is correct behavior. The mistake is a uniform "85% OEE everywhere" target — that's the utilization trap applied to metrics.

- **SMED (Single Minute Exchange of Die):** A lean technique for reducing physical changeover times. In POD: DTG platen swaps (small tee → hoodie platen) are a real SMED target — reducing swap time shifts the optimal batch size smaller, enabling more frequent switches and shorter queues for all order types. Sublimation fixture changes, cut-and-sew pattern changes, and embroidery thread/hoop changes also apply. Note: periodic DTG maintenance (printhead cleaning, ink purging) is NOT SMED — that's TPM, addressed at Level 2. Key connection to Level 1: SMED reduces the changeover penalty, but the OPTIMAL platen sequence and batch sizes require a scheduler that balances changeover cost vs. waiting cost across all pending orders (s_{ijk} in Pinedo's notation).
- **CONWIP (Constant Work In Process):** A pull system where you set a maximum number of jobs allowed on the production floor. When one finishes, the next enters. Unlike kanban (which needs a card per part number — impossible for print-on-demand), CONWIP uses one cap for the whole line. This IS a software deliverable — the production system must enforce the cap. Key advantage for POD: CONWIP adapts to shifting bottlenecks automatically — WIP naturally pools before whichever station is currently slowest.
- **Value stream mapping:** Traces a single order's journey from arrival to shipment, recording process time AND wait time at every step. Example: Routing (0.5m) → Queue (45m) → Printing (8m) → Queue (20m) → Dryer (4m) → Queue (35m) → QC (2m) → Queue (15m) → Packing (3m). Value-added time: ~17.5 min. Total lead time: ~2+ hours. That means ~85% of the time the order is just sitting in a queue — invisible waste nobody planned. The map shows WHERE time is lost: the longest queue sits before the current bottleneck; a station with 2-min process time but 40-min queue isn't slow, it's understaffed or batching unnecessarily. **POD caveat — shifting flows:** A single map isn't enough because the flow changes with product mix. You'd need maps for main scenarios (DTG-heavy, cut-and-sew-heavy, mixed). But the real pitch isn't "do a paper VSM exercise" — it's that the software team's instrumentation work (timestamps at every stage) creates a **continuous, automated value stream map as a dashboard**. Instead of a one-time walk-the-floor exercise with a stopwatch, you get live wait times vs. productive times broken down by product type, updated in real time. This doesn't go stale, captures every product-mix scenario automatically, shows trends over time, and directly feeds the Level 1 solver with real processing and queue time data. **POD caveat — multi-component products:** Simple products (single-component DTG) have a linear flow that's easy to track. But multi-component products (cut-and-sew) have a DAG (directed acyclic graph), not a line. Example: a cut-and-sew hoodie

has front panel, back panel, and sleeves — each printed on a different roll, cut separately, then all converging at a sewing station. Component A finishes cutting at 10:00, but sewing can't start until ALL components arrive. If Component C finishes at 10:45, then A and B sat idle for 45 and 20 minutes respectively. That wait isn't caused by the sewing station being busy — it's **synchronization waste** (waiting for the slowest sibling component). The dashboard needs to track: (1) per-component path and timing, (2) synchronization wait at assembly/convergence points, (3) which component path is the critical path (determines total lead time), and (4) which path is most often critical across orders (where to focus improvement). This also has a Level 1 scheduling implication: the solver should coordinate component completion times so they arrive at sewing roughly together, minimizing synchronization waste. This is a real constraint for the MIP — not just "minimize tardiness" but "minimize tardiness while coordinating parallel component paths." **Dashboard vs. paper exercise:** The one-time paper VSM is useful for building initial understanding with ops. But the software deliverable (stage timestamps + dashboard showing wait vs. productive time by product type, with multi-component synchronization tracking) is what creates lasting value. VSM alone doesn't improve anything — same caveat as OEE. It makes waste visible. Ops has to act on it.

- **Standard work:** Documenting the current best-known method for each task. Software can embed guidance in Shoptool; ops creates and maintains the content.

8/28 Level 1: Optimize

Level 1 is where the biggest, most certain gains are. None of this is AI — it's math that's been used in airlines, logistics, and manufacturing for decades.

Glossary:

- **Greedy heuristic:** Our current routing approach. For each order it scores 14 penalty factors per candidate facility (fulfillment time, shipping cost, cross-region penalty, load balancing, branding availability, special customer rules, and more), then uses a greedy-optimized split algorithm to assign items to branches — iterating to improve the assignment but never exploring all combinations. It's good, but it optimizes one order at a time, not the whole batch at once. A MIP solver considers all pending orders simultaneously and finds the provably best assignment.
- **MIP (Mixed-Integer Programming):** A solver that uses mathematical techniques (branch-and-bound, cutting planes) to find provably optimal or near-optimal solutions without enumerating every possibility. "Mixed-Integer" means some decisions are yes/no (assign this order to Facility A or B) and some are continuous (how much capacity to allocate). Think of it as a super-powered Sudoku solver for business decisions.
- **Constraint Programming (CP):** Like MIP but better at handling complex rules — "Machine A can't run while Machine B is in setup" or "these two jobs can't overlap." Perfect for scheduling.
- **Bin packing:** Given items of different sizes, fit them into the fewest bins (or onto sheets with minimal waste). Our geometry batching today is manual grouping — a solver could optimize substrate utilization automatically.
- **OR-Tools:** Google's free, open-source optimization toolkit. Used by Google itself for fleet routing, scheduling, etc. Zero license cost.

 Level 1: The Evidence

These numbers come from academic textbooks, not vendor marketing. They show what solvers achieve on the same *class* of problem we face — not a direct prediction for our facility. A Q1 pilot (re-solving 30 days of historical orders) will give us our own numbers.

Sources:

- **Scheduling 48%:** Postek, Zocca, Gromicho, Kantor — *Hands-On Mathematical Optimization with Python* (Cambridge, 2025), Example 3.8. FIFO heuristic: 31 units past-due; MIP solver: 16 units past-due on identical input. Same problem class as ours: jobs with release dates, durations, and due dates on shared machines.
- **Shift planning 30%:** Same textbook, Example 3.3. MIP shift scheduler found 7 of 10 shift-slots were sufficient to cover all demand. This is about matching worker availability to demand patterns, not about cutting headcount.
- **CONWIP 20-25%:** Hopp & Spearman — *Factory Physics* (3rd ed.). In a 5-station example, push systems required 25% more WIP to achieve the same throughput as CONWIP.
- **Constrained assignment:** Our routing has merge constraints (items in same shipment should co-locate), eligibility constraints (not every facility handles every product), and multi-factor scoring. A MIP solver handles all of these natively.

10/28 Level 1: Where It Applies

This slide makes Level 1 concrete. Each row maps a real part of our facility to a textbook optimization problem. If someone asks "where would we start?" — pick the row with the biggest pain point.

DTG / DTFilm — Platen scheduling: The platen swap problem we discussed: switching between small tee, large hoodie, baby onesie platens takes time (s_{ijk} = sequence-dependent setup times). A solver determines the optimal sequence and batch sizes: "print all small tee orders first, then switch to hoodies, then onesies" vs. "interleave to meet due dates." The tradeoff is changeover time lost vs. orders waiting in queue. SMED reduces the swap time; the solver optimizes the sequence given whatever the swap time is.

DTFilm — Component tracking: DTFilm is a two-phase process: (1) print designs on film sheets, which are stored as components, then (2) later pressed onto garments. This creates an inventory management problem: when to print films (batching for efficiency) vs. when to press them (driven by garment order due dates). A solver can optimize the print schedule to minimize the inventory of stored film components while ensuring films are ready when the corresponding garment orders need pressing. This is a classic inventory/scheduling hybrid problem.

Sublimation rolls / DTFabric — 2D nesting / cutting stock: When printing on rolls (sublimation for cut-and-sew, or the new DTFabric printer), the solver determines how to arrange multiple print designs on a roll to minimize wasted fabric. This is the "cutting stock problem" — one of the most well-studied problems in operations research, with excellent solvers. The same applies to DTFabric's new roll printer. Industry benchmarks show 5-15% material savings from optimized nesting vs. manual arrangement.

Cut&Sew — Multi-component synchronization: As discussed earlier, a cut-and-sew product has multiple components (front panel, back panel, sleeves) that may be printed on different rolls, cut separately, and then must all converge at a sewing station. If components arrive at different times, early ones create synchronization waste (waiting for the slowest sibling). A solver coordinates the scheduling of all components so they arrive at sewing roughly together. This is a scheduling problem with precedence constraints and assembly operations — well-studied in job shop scheduling literature (Pinedo Ch. 7).

Mugs — Fixture scheduling: Sublimation mugs require specific fixtures (11oz, 15oz, travel mug). Switching between fixture types takes time. Same optimization logic as DTG platen scheduling: batch by mug type to minimize fixture swaps, but balance against due date pressure. Smaller problem size than DTG (fewer fixture types), so potentially easier to solve optimally.

Canvases / Posters / Notebooks — Sheet nesting: Multiple canvas or poster prints can fit on a single substrate sheet. This is classic 2D bin packing: given prints of various sizes, arrange them on sheets to

minimize the number of sheets used (= minimize substrate waste). OR-Tools has a dedicated bin packing solver. UV printing on flat items follows the same logic.

Key point for Q&A: Every row maps to a textbook optimization problem with mature solvers. We're not inventing new algorithms — we're applying known solutions to our specific constraints. The risk is in implementation and data quality, not in whether the math works. We deliberately excluded areas like warehouse slotting and sorting consolidation — those are better served by simple heuristics (ABC analysis, accumulate-and-pack) and don't need solver-level optimization.

11/28 Level 2: Predict

Level 2 is where machine learning enters — but for prediction, not decision-making. We're using AI to see the future a few hours ahead, then feeding those predictions into the Level 1 solvers.

Glossary:

- **Demand forecasting:** Predicting how many orders (and what type) will arrive in the next 4-24 hours. Like a weather forecast for orders. Even rough accuracy helps — if you know a surge is coming, you pre-staff the bottleneck station. Key point: we already have this model in-house — the ML team built it for Inventory/Purchasing. Extending it to production planning is adapting an existing asset, not starting from zero.
- **Predictive maintenance:** Using sensor data and historical failure patterns to predict "this printer will likely jam in 3 hours." Like how your car dashboard warns about oil changes before the engine seizes — but for production equipment. In TOC terms, an unplanned breakdown at the bottleneck is the worst thing that can happen; predicting it lets you re-route work before it hits.
- **ML regression:** A type of machine learning that predicts a number (e.g., "this order will take 4.2 hours") based on patterns in historical data. Our current ETA model already uses a queue-depth ratio (pending items ÷ average daily throughput, updated hourly — essentially Little's Law). ML regression adds features the current model ignores: product mix complexity, time of day, day of week, recent machine downtime. The gain is incremental, not a full replacement.
- **AIQC (AI Quality Control):** Our own AI QC system, currently in **data-collection phase** — the `AIQCEventService` sends item data (Kornit preprocessor params, garment color/size, print file S3 URLs) to a printserver SQS queue, and the Shoptool `TestController` collects operator-captured photos paired with QC outcome labels (pass/hold/damaged). The test device is currently disabled (`shouldEnqueue` returns `false`). Only **2 garment families** are supported (Bella+Canvas 3001 and Gildan 5000). Level 2 work is: (1) complete data collection and launch the model in production, (2) expand SKU coverage to the full catalog, and (3) build a feedback loop connecting defect outcomes back to upstream parameters (ink batch, press settings, preset) to prevent issues before printing.

12/28 **Level 2: Why Prediction Matters Mathematically**

This connects Level 2 back to the science from slide 4. The VUT equation says queue time is driven by three factors — and each prediction capability attacks a different one.

The VUT Equation — full breakdown

The VUT equation comes from **Factory Physics** (Hopp & Spearman, Ch. 8). It's derived from queueing theory — specifically the Kingman/VUT approximation for a G/G/1 queue (general arrivals, general service, single server). The full form is:

$$Q_t = \left(\frac{c_a^2 + c_e^2}{2}\right) \times \left(\frac{u}{1-u}\right) \times t_e$$

Where:

- **Qt** — the **expected queue time**: how long a job waits in line *before* it starts being processed. This is pure waiting, not productive work. It's the single biggest component of cycle time in high-utilization systems — often 80-90% of total time an order spends on the floor is just waiting.
- **V = (ca² + ce²) / 2** — the **variability** factor. ca is the coefficient of variation of inter-arrival times (how unpredictably orders arrive), ce is the coefficient of variation of processing times (how unpredictably a station processes). Both are unitless ratios (standard deviation / mean). When both are exactly 1, this factor equals 1 — that's the "random" (Markovian) baseline. Below 1 = more predictable than random. Above 1 = more erratic.
- **U = u / (1 − u)** — the **utilization** factor. u is utilization (fraction of time the station is busy, 0 to 1). This is the hockey-stick term: at u=0.5 it's 1, at u=0.9 it's 9, at u=0.95 it's 19, at u=0.99 it's 99. This is why the utilization trap exists — the relationship is hyperbolic, not linear.
- **T = te** — the **mean effective process time** per job at the station. Longer individual process times mean longer queues, linearly.

The equation gives the **expected time a job spends waiting in queue** (not being processed — just waiting). Total cycle time at a station = queue time + process time = Qt + te.

How each prediction capability maps to the equation — honestly

Demand forecasting → primarily attacks U (utilization), NOT V. A common overclaim is that demand forecasting "reduces arrival variability." It doesn't — customer orders still arrive when they arrive; the spikes are real. What forecasting actually does: if you know a spike is coming in 2 hours, you can pre-staff

additional operators, pre-stage materials, or spin up extra capacity. This keeps utilization (u) from spiking into the hockey-stick zone. At u=0.90 the U factor is 9; if you add capacity to keep u=0.80, the U factor drops to 4 — more than halving queue time. The forecast doesn't smooth demand; it lets you **match capacity to demand before you're overwhelmed.**

There IS a secondary V-reduction mechanism, but only when combined with controlled work release (CONWIP from Level 0): if you know demand is coming, you can pace how work enters the floor rather than dumping it all at once. This smooths the *effective* arrival pattern at downstream stations, reducing ca. But that's CONWIP doing the smoothing — the forecast just tells you what to prepare for.

Predictive maintenance → attacks V (process variability, ce). This one IS a genuine V-reduction. Unplanned machine breakdowns inject massive spikes into ce — a station that normally takes 2 minutes per job suddenly takes infinity (it's down). Predictive maintenance converts unplanned downtime into planned downtime: you schedule maintenance during low-demand windows instead of suffering random breakdowns during peaks. This directly reduces ce, which directly reduces V. This is the strongest VUT-based argument for prediction on this slide.

Quality prediction → attacks T (effective process time, te). When a defective item is caught late in the process, it has to be reworked — effectively doubling (or more) its process time. Quality prediction catches defects earlier or prevents them upstream, reducing the average effective process time te. There's also a V-reduction component: rework is highly variable (some items need rework, most don't), so reducing rework rate also reduces ce.

Why it matters for Level 0: SMED reduces ce (less variability in effective process time by reducing changeover variation). CONWIP controls WIP directly (Little's Law), which indirectly stabilizes arrival patterns to downstream stations. Standard work reduces ce by making processing times more consistent.

Glossary:

- **Arrival variability (ca):** How unpredictable order arrivals are from a station's perspective. Demand forecasting does NOT reduce this directly — orders still arrive when customers place them. But controlled work release (CONWIP) can smooth the effective arrival pattern at stations.
- **Process variability (ce):** How unpredictable processing times are. Predictive maintenance genuinely reduces this — converting random breakdowns into scheduled maintenance windows.
- **TOC (Theory of Constraints):** The management philosophy that says: find the ONE biggest bottleneck, protect it, and make everything else serve it. If the bottleneck goes down unexpectedly, the whole plant suffers. That's why prediction matters most AT the bottleneck.

13/28 Level 2: We Already Have a Vision

Link to the Google Doc

This slide references Voldemārs Kalve's document "Driving process automation through forecasting outcomes." The point is that Level 2 isn't speculative — we already have an internal product manager who mapped out concrete automation opportunities that depend on trusted, automated forecasting. The document covers:

1. **Order routing:** Currently Printful and Printify use separate routing logic, making forecasting harder. The key insight is that routing should maximize profit, not minimize cost — different margins on manufacturing vs. shipping mean the cheapest option isn't always the most profitable. He proposes tracking "expected winner" (optimal route) vs. "actual winner" (where it was routed) to quantify money left on the table. This aligns directly with our Level 1 routing optimization — the MIP solver can optimize for profit instead of cost with the same constraint structure.

2. **3P forecasting:** Automated price-change impact calculations for print providers (the Share Promise deal with Monster Digital and SwiftPOD generated 5+ million USD additional profit — that calculation was done manually but could be automated). Auto-regeneration of forecasts when key thresholds change (e.g., PP changes pricing by more than X%).

3. **1P planning forecasts (technique level):** From forecast → daily volumes → required headcount per station. This is the workforce scheduling pipeline: forecast demand, calculate required capacity, assign employees by skill, and handle real-time reassignment when shortages occur. All of this maps to Level 1 (MIP shift optimizer) fed by Level 2 (demand forecasting).

4. **DTG queue optimization:** Tag pickup trolleys and printer destinations to direct employees to the right printer. Keep popular pallets loaded all day to eliminate changeover time. This connects to Level 0 (SMED — reducing setup time) and Level 1 (scheduling — optimal pallet sequence).

5. **1P purchasing forecasts:** Replace backward-looking weeks-on-hand calculations with forward-looking forecasts. Automated bulk buys that factor in facility space availability. Predict non-blank consumable usage (ink, paper, glue) based on product category popularity and machine count.

The strategic point: we're not proposing Level 2 in a vacuum. An internal stakeholder has already mapped the downstream automation opportunities that forecasting enables. The infrastructure we build for Level 2 directly unlocks this existing roadmap.

14/28 Level 3: Adapt

Level 3 is the most ambitious near-term investment. The digital twin is the cornerstone — it's a virtual copy of the plant that lets you test changes without risking real production. Think of it as a flight simulator for the factory.

Glossary:

- **Digital twin:** A software model that mirrors the real plant in real time — every machine, every queue, every worker. When a printer breaks in real life, it breaks in the model too. You can then ask "what if?" questions: "What if we re-route these orders?" — and the twin shows the outcome in seconds.
  - Examples for existing software:
    - FlexSim
    - AWS IoT TwinMaker
- **Reinforcement learning (RL):** An AI that learns by trial and error — like training a dog, but in a simulator. It tries thousands of dispatching strategies in the digital twin, gets "rewarded" when throughput goes up and SLAs are met, and gradually learns a strategy that's better than any hand-crafted rule. Crucially, all the mistakes happen in the simulator, not on real orders.
- **Policy tuning:** Our current dispatching rules have parameters (thresholds, weights, priorities). Today these are set by gut feel. Policy tuning means running thousands of simulations with different parameter values to find the best ones — like tuning a radio to the clearest frequency.
- **Rolling horizon:** Instead of making a schedule for the whole day and sticking to it, re-solve every few minutes with updated information. Like re-checking GPS every 5 minutes — if traffic changed, you get a new route.

15/28 Level 3: The Policy Evolution

This table shows that our current Shoptool rules, lean process improvements, the solvers we want to add, and future AI are all just different "strategies" for the same game. Level 0 is deliberately placed before optimization — in Powell's framework, Level 0 is about stabilizing the system so that any policy class (PFA, CFA, VFA) operates on a cleaner, lower-variability foundation. Specific percentage gains will come from our own pilots — the textbook benchmarks set expectations but aren't direct predictions for our facility.

Glossary:

- **Level 0 (Stabilized rules):** Not a new "policy" in Powell's sense — it's about reducing the variability and noise that ALL policies must cope with. SMED reduces process variability (c_e in VUT). CONWIP controls WIP (Little's Law). OEE provides the measurement baseline. These are the preconditions that make every subsequent level more effective.
- **PFA (Policy Function Approximation):** A fancy name for "if-then rules." Our Shoptool today: IF order is DTG AND facility has capacity, THEN route there. Simple, fast, but limited.
- **CFA (Cost Function Approximation):** Instead of rules, you solve an optimization problem. "Given these 200 orders and 5 facilities, find the assignment that minimizes total cost." The solver does the thinking, not the rule writer. This is the industrial workhorse — airlines use CFAs to assign planes to routes.
- **VFA (Value Function Approximation):** The RL approach. The system learns "how valuable is it to be in this state?" — e.g., "having Machine 3 available with 10 orders in queue is worth X." Then it makes decisions that lead to high-value states. This is "real AI" — it learns and adapts.
- **Measurable in simulation:** Every step can be tested by running both the old and new approach on the same historical data in the digital twin. No guessing — you see the numbers before committing.

16/28 Level 4: Orchestrate

The full vision — this is aspirational (2-3 year horizon). Each layer does one job and feeds the next. Think of it as a nervous system for the factory: senses (L1), anticipates (L2), plans (L3), acts (L4), communicates (L5).

Glossary:

- **MES/ERP integration (Layer 1):** Connecting to our existing business systems (order management, inventory) so the digital twin knows what's happening in real time.
- **IoT sensors (Layer 1):** Devices on machines that report status — temperature, vibration, ink levels, job counts. The "eyes and ears" of the digital twin.
- **Constraint Programming / MIP Solver (Layer 3):** The math engines from Level 1, now running continuously — re-solving the schedule every few minutes as conditions change.
- **TOC Bottleneck Logic (Layer 3):** The scheduling layer always prioritizes the bottleneck. If the DTG printers are the constraint today, everything else subordinates to keeping them fed.
- **Metaheuristic Fallback (Layer 4):** When the RL agent faces a situation it hasn't learned yet, it falls back to fast approximate methods (genetic algorithms, simulated annealing) — like a backup autopilot.
- **LLM Copilot (Layer 5):** A ChatGPT-like interface for supervisors. "Why was Batch 247 prioritized?" → "Because Machine 7 has predicted maintenance in 2 hours and these orders have tight SLAs."

17/28 What Changes for Workers?

This is the "what about our people?" slide. Important for SLT who will immediately worry about workforce impact.

Glossary:

- **Augmentation vs. replacement:** We're not replacing workers with robots. We're giving workers better tools — like upgrading from a paper map to GPS. The worker still drives; the system just shows the best route.
- **Skill equalizer:** Research from JD.com and others shows AI-assisted tools help new employees perform closer to experienced employees. Think of it like spell-check — it helps weak spellers more than strong ones, raising the floor without lowering the ceiling.
- **Override without penalty:** Workers MUST be able to say "the system is wrong" and do something different without getting blamed. If they can't override, two bad things happen: (1) they follow bad recommendations blindly, and (2) they stop trusting the system entirely.
- **Trust is fragile:** In one study, hospital radiologists used an AI tool for months, then abandoned it after a few disagreements. Trust takes months to build and moments to destroy. That's why we build incrementally.

18/28 ROI Framework

Level 0 is the fastest path to visible results. It requires no new software, no hiring, and no risk. A SMED kaizen on the bottleneck station can deliver results in a single week (Art Byrne's documented case: 93% setup reduction, $332 spend). OEE is a database query on data we already collect. CONWIP is a policy decision, not a technology purchase.

If asked "what would it take to try?" — Level 0 needs a kaizen team and a week. Level 1 would need 1-2 engineers and open-source tools for 3-6 months. That's the minimum experiment. Everything else is optional and would only make sense if earlier levels show real gains.

Glossary:

- **Confidence levels explained:** "Highest" = this is the Toyota Production System, proven at thousands of companies over 50+ years. Wiremold: enterprise value increased 2,467%. American Safety Razor: 3.5× investment return. Virginia Mason Medical Center: operating profit from $700K to $40.9M. "Very high" = proven math used by airlines since the 1950s, we know it works. "High" = many companies have done this and measured 15-35% improvements. "Medium" = works in research and some companies, but needs our specific simulation to validate. "Exploratory" = cutting-edge, ambitious, 2-3 year vision.
- **"Each level funds the next":** Level 0's WIP reduction frees working capital and shortens lead times. If Level 1 saves X% in scheduling waste, that justifies Level 2's data pipeline. If Level 2's predictions save Y% in unplanned downtime, that justifies the digital twin for Level 3. No big upfront commitment needed.
- **SMED ROI:** Art Byrne documents setup reductions from 90 minutes to 6 minutes in one kaizen week. If a bottleneck machine does 3 changeovers/day, that's 4.2 hours recovered — equivalent to adding half a machine shift for $332.

19/28 What We're NOT Proposing

This slide preempts the "we've heard this before" objection. The Tayur quote is from a Carnegie Mellon professor who's skeptical of AI hype but supportive of targeted applications — perfect credibility anchor.

Glossary:

- **Pilot purgatory:** When a team builds a cool demo that works in a lab, but it never makes it to production because nobody can explain the business case. "Look, the AI can predict machine failures!" → "Great, but how much money does that save and what does it cost to maintain?"
- **Automating broken processes:** If your process has high variability (long setups, no WIP control, no OEE visibility), putting a solver on top produces modest gains at best. The VUT equation proves this: V dominates queue time at high utilization. Fix V first (Level 0: SMED, CONWIP), then optimize (Level 1). Art Byrne's Lean Turnaround: "Present Capacity = Work + Waste." Remove the waste before you optimize the work.
- **"Optimizing without measurement is guessing":** We can't improve what we don't measure. OEE is hardcoded to 0 in our Monitor module. We need to know our Availability × Performance × Quality before we can know where to apply solvers.
- **"RL without optimization is like AI without data":** You can't train a learning system if you don't first know what "good" looks like. The Level 1 solvers define what optimal looks like. The Level 3 RL system then learns to approximate that optimal in real-time conditions.

20/28 AI Readiness: Prerequisites

Level 0 has the lowest barrier of all. The software deliverables (OEE, instrumentation, CONWIP logic) are straightforward engineering work on existing systems. The ops recommendations (SMED, value stream mapping) need our data but are executed by ops teams on the shop floor. No new hires needed for Level 0.

If asked "what do we need to start?" — Level 0 SW work can be done by existing engineers. The ops recommendations go to the kaizen teams. Level 1 needs 1-2 engineers with optimization modeling skills. We already have the data in our routing/batching modules.

Glossary:

- **OEE implementation:** The Monitor module has an `oeePercentage` field hardcoded to 0. The maintenance log data already exists. Computing OEE = Availability × Performance × Quality is a database query, not an ML project. This is the single most important metric for production — without it, we're optimizing blind.
- **Value stream map:** A visual map of every step from order to shipment, showing processing time vs. waiting time at each stage. In most factories, value-added time is <5% of total lead time. The map reveals where time is wasted and where the true bottleneck sits.
- **SMED kaizen:** A focused week-long event where a cross-functional team attacks setup times on one machine. Typical result: 50-90% reduction. The Lean Turnaround documents 93% reduction (90 min → 6 min) in one week for $332.
- **OR-Tools / Pyomo:** Software libraries (like Excel but for optimization). A developer writes "minimize total late orders, subject to: each order goes to exactly one facility, no facility exceeds capacity" and the solver finds the best answer. Pyomo is the modeling language, OR-Tools/Gurobi are the engines.
- **Data pipeline:** A system that automatically collects, cleans, and stores data — processing times, failure logs, demand patterns. Like plumbing for data: you need it before you can run ML models.
- **Digital twin / DES (Discrete-Event Simulation):** A computer program that replays factory operations event by event — "order arrives at 10:01, starts printing at 10:05, printing finishes at 10:12, moves to QC at 10:13..." Used to test scheduling changes without touching the real factory.

21/28 Recommended First Moves

Phase 0 has two tracks: software deliverables (OEE, instrumentation, CONWIP logic) and a joint recommendation to ops (SMED kaizens, value stream mapping). The SW items are ours to build. The ops items need our data but are executed on the shop floor by ops teams. Phase 1 runs in parallel on historical data. Neither blocks the other. Phase 0 SW work has zero production risk (OEE is a metric, CONWIP is a controlled pilot) and Phase 1 has zero production risk (running solvers on historical data).

Why Phase 0 matters for Phase 1: A solver optimizing a high-variability process (long setups, unpredictable machine availability, no WIP control) will produce modest gains — maybe 10-15%. The same solver on a stabilized process (short setups, controlled WIP, known bottleneck) can achieve the full 20-48% benchmark range. Factory Physics proves this: the VUT equation shows that variability (V) is a linear multiplier on queue time. Halving V doubles the effective benefit of any scheduling improvement.

Why we added "tune existing rules" to Phase 1: Before building a MIP, test whether systematic parameter tuning of our current dispatching heuristics (using ATCS-style composite priority rules from Pinedo's Scheduling Theory) closes part of the gap for free. Powell's Chapter 12 frames this as "policy search over PFAs" — optimizing the parameters of existing rules is itself a valid optimization approach. If tuning closes 50% of the gap, the remaining 50% justifies the solver investment.

Glossary:

- **Live bottleneck tracking + demand-adjusted OEE:** In POD, the bottleneck shifts with product mix (DTG-heavy → printers; cut-and-sew surge → sewing). "Find THE bottleneck" is the wrong framing — instead, instrument queue depths at every station so ops can see where WIP is accumulating RIGHT NOW. The Monitor module's `oeePercentage` field (hardcoded to 0) can be repurposed, but must be demand-adjusted: compute only during periods with pending work, since idle-because-no-orders is correct POD behavior, not a loss. Track at candidate bottleneck stations, not everywhere. The statistical bottleneck (highest avg utilization over time) guides long-term investment; the real-time view guides daily ops.
- **Value stream map (as a dashboard):** Traces orders' journeys, recording process time AND wait time at every step. Typical finding: value-added time is <15% of total lead time — the rest is queue time nobody planned. The real deliverable is the software instrumentation: stage timestamps create a continuous, automated VSM dashboard showing wait vs. productive time by product type. For multi-component products (cut-and-sew), the dashboard must also track synchronization waste — how long early components wait for siblings at assembly points — and identify the critical path (slowest component that determines total lead time). This data directly feeds the Level 1 solver: scheduling should coordinate component completion times to minimize synchronization waste.

- **SMED kaizen:** Applicable wherever physical changeovers exist: DTG platen swaps (small tee → hoodie → onesie), sublimation fixture swaps, cut-and-sew pattern changes, embroidery thread/hoop changes. NOT applicable to individual DTG prints (new file + garment swap is the process, not a setup) or periodic printer maintenance (that's TPM, Level 2). A one-week kaizen separates internal steps (machine stopped) from external steps (can prep while running), typically yielding 50%+ reduction. Key benefit: reducing platen swap time shifts the optimal batch size smaller — more frequent switches mean shorter queues for ALL order types. The optimal platen sequence then becomes a Level 1 scheduling problem.
- **CONWIP pilot:** Set a maximum WIP count for one line. When a job exits, release the next. Factory Physics proves this achieves the same throughput as push systems with 20-25% less WIP. Critical WIP: $W_0 = r\_b \times T_0$. Start at $2\times W_0$ and gradually reduce.
- **ATCS tuning:** "Apparent Tardiness Cost with Setups" — the most sophisticated dispatching rule in scheduling theory (Pinedo, Ch. 5). It combines urgency, processing time efficiency, and setup minimization into one composite priority score. Systematically tuning its parameters (using grid search or Bayesian optimization on historical data) can close a significant portion of the gap between our current FIFO and a full solver — at near-zero cost.
- **Routing MIP pilot:** Take 30 days of historical orders and re-route them with a MIP solver using the same penalty factors the current engine uses (shipping cost, fulfillment time, cross-region, etc.). Compare total score vs. what the greedy split actually chose. The gap = improvement potential.
- **Bin-packing MIP:** Same idea for batching — take yesterday's geometry batches, re-solve them with a solver, and compare substrate utilization. Note: first confirm which code component makes the item-grouping decision (the current `GeometryBatchService` is a data access layer; the decision logic may be upstream).

22/28 Summary

Close by tying the threads together. The closing statement paraphrases core Factory Physics principles (variability buffering law + Little's Law + WIP control) into one sentence — it's a synthesis, not a direct quote. This is a conversation starter, not a final plan.

The key narrative shift: the original pitch was "solvers first." The revised pitch is "stabilize first, then solvers." This is what Factory Physics and Lean both demand. The VUT equation proves it: a 50% reduction in variability (V) through SMED and process stabilization halves queue time at any utilization level — no solver needed. Then applying a solver to the now-stable system captures the full 20-48% benchmark potential. Without Level 0, solvers operate on noisy data and high-variability processes, delivering maybe half their theoretical benefit.

Glossary (recap for Q&A):

- **Variability:** Unpredictability — machine breakdowns, demand spikes, quality failures. Factory Physics proves it's buffered by inventory, capacity, or time — and the only way to reduce all three buffers is to reduce variability itself.
- **WIP control:** Limiting how much work is on the floor at once. Less WIP = faster throughput (Little's Law). CONWIP is the practical mechanism: one cap for the whole line, robust to parameter errors.
- **Stabilize first, then optimize:** Level 0 (SMED, CONWIP, OEE) attacks variability at source — the V in VUT. Level 1 (solvers) attacks scheduling efficiency — the U and T in VUT. Together they compound: reduced V means the solver operates on a cleaner problem with better data and larger achievable gains.
- **Intelligence:** The spectrum from lean process improvement (Level 0) to math solvers (Level 1) to learning systems (Level 3). Human intelligence (Lean/kaizen) is always part of the picture — AI augments it, doesn't replace it.

23/28 Questions?

Keep this open-ended. If no questions, move to appendix only if there's interest. Common questions to expect:

- "How do we know the 20-48% number is realistic?" → Textbook benchmarks on the same problem class + we can prove it on our own historical data. But those numbers assume a stable process — which is why Level 0 comes first.
- "Do we need to hire?" → Level 0 needs nobody new — existing ops teams and kaizen facilitators. Level 1 needs 1-2 engineers with optimization skills. Level 3 may need RL expertise (hire or partner).
- "What's the risk?" → Level 0 is zero risk (lean fundamentals, proven for decades). Level 1 is near-zero risk (proven math, tested offline). Risk increases with each level but so does the evidence base.
- "How does this affect headcount?" → This is about throughput and SLA improvement, not headcount reduction. Lean transforms: Wiremold quadrupled sales while growing headcount modestly. Workers do the same jobs with better guidance and less firefighting.
- "We already do lean/kaizen — why is this new?" → The question is whether the fundamentals are fully embedded. OEE hardcoded to 0 suggests measurement gaps. If setup times haven't been attacked with SMED, and WIP isn't controlled via CONWIP, there are high-ROI opportunities before we need any solver.
- "Why not just jump to AI?" → Factory Physics: a solver on a high-variability system delivers modest gains. The VUT equation proves variability (V) is a linear multiplier on queue time. Halving V through process improvement doubles the effective benefit of any downstream optimization.

 A1: The Scheduling Problem Formally

Only go here if someone asks "how do you know the problem is hard?" This is the mathematical proof.

Glossary:

- **FJc (Flexible Job Shop):** Multiple work areas (printing, cutting, packing), each with several machines. Different orders take different paths through these areas. This is literally our production floor.
- **r_j (release dates):** Orders arrive throughout the day — you can't schedule an order before it exists. This makes the problem "online" — you're constantly replanning.
- **s_ijk (sequence-dependent setup times):** Switching from a DTG job to a sublimation job takes time (ink change, substrate change). The time depends on BOTH what you were doing and what you're switching to. The sequencing component reduces to the Traveling Salesman Problem — one of the hardest problems in math.
- **M_j (machine eligibility):** Not every printer handles every product. This adds constraints that make the problem harder.
- **sum(w_j * T_j) (weighted tardiness):** Our goal: minimize late orders, weighted by priority. A rush order that's 1 day late matters more than a standard order that's 1 day late.
- **"Strongly NP-hard":** No pseudo-polynomial algorithm exists either — the problem remains hard even when all numbers are small. At production scale, exact optimal solutions are out of reach. That's OK — solvers find solutions that are 95-99% optimal in seconds, which is far better than our current rules.

26/28 A2: CONWIP --- The Right Pull System for Us

Go here if someone asks about Lean pull systems or kanban. The key point: kanban doesn't work for us because every print is unique.

Glossary:

- **Kanban:** A pull system from Toyota where each part has a card. When you use a part, the card goes back to the supplier as a signal to make more. Problem: you need a card for every part number. Toyota makes thousands of the same bumper — we print millions of unique designs. We'd need infinite cards.
- **CONWIP:** "CONstant Work In Process." One simple rule: set a maximum number of orders allowed on the production floor. When one finishes and ships, the next one enters. That's it. No cards per product, no complex tracking. Think of it as a "one in, one out" policy for the whole line.
- **Why CONWIP beats push:** In a push system (like MRP), you release work based on demand forecasts. When forecasts are wrong (and they always are), you flood the floor with WIP, which makes everything slower (Little's Law). CONWIP prevents this by design — the floor can never be overloaded.
- **W0 = rb × T0:** The "critical WIP level" — the minimum amount of work on the floor needed to keep the bottleneck fully fed. Below W0, you're starving the bottleneck. Above 2×W0, you're just creating congestion. The sweet spot is 1.5-2× W0.

27/28 A3: Powell's Four Policy Classes

Go here if someone asks "how does RL relate to optimization?" Powell's framework shows they're the same family — just different levels of sophistication.

Glossary:

- **Sequential decision problem:** Any situation where you make a decision, see what happens, make another decision, repeat. Dispatching orders on a factory floor is exactly this: assign Order A to Machine 1, see new orders arrive, assign Order B to Machine 3, a machine breaks, reassign...
- **PFA (rules):** "Always do the most urgent order first." Fast, simple, but blind to trade-offs.
- **CFA (optimization):** "Solve a math problem to find the best assignment of all current orders to all machines." Better, but assumes you know everything about the current state.
- **VFA/RL (learning):** "I've seen a million similar situations in simulation, and I've learned that when the queue looks like THIS, the best action is THAT." Handles uncertainty and novel situations, but needs extensive training.
- **DLA (lookahead):** "Let me simulate the next 4 hours and pick the action that leads to the best predicted outcome." Like a chess engine thinking several moves ahead.
- **CFA is the sweet spot for industry:** Powell notes that most real-world systems (airlines, logistics, warehousing) use CFAs — optimization models with a few tunable parameters. They're practical, explainable, and effective. Pure RL is rare in production systems.

28/28 A4: Technology Tier Map

Reference slide — only pull this up if someone asks "what specific technologies are we talking about?" Read top to bottom: Level 1 (green) is proven and actionable now, Level 2 (blue) has industry evidence, Level 3-4 (purple/red) is the future vision.

Glossary:

- **Constraint Programming (OR-Tools):** Google's free solver. Tell it "here are my machines, here are my jobs, here are the rules about what can run where" and it finds the best schedule. Already used at Google, Amazon, and thousands of companies.
- **CONWIP:** The simple WIP cap explained in A2. Level 1 because it's cheap to implement and has guaranteed benefits.
- **Discrete-Event Simulation:** The foundation of the digital twin. Models the factory as a sequence of events (order arrives, printing starts, printing finishes, QC starts…) and lets you fast-forward through a whole day in seconds.
- **Multi-Agent Systems:** Each machine/station acts as an independent "agent" that negotiates with other agents. Useful at very large scale when a central optimizer is too slow. Think of it like a marketplace: each machine "bids" for orders based on its current queue and capabilities.
- **Queueing Theory:** The math behind "why do lines form and how long will they be?" Not a technology we implement — it's the analytical framework that explains why the VUT equation works. Useful for back-of-envelope calculations and intuition.