PROCESS FLOW VISIBILITY

Finding the hidden time — and diagnosing why it waits

White Paper · Second edition · Reflects the shipped diagnose → prescribe → model toolkit at csi.dewood.org

> **Start in the elevator.** Two questions, fifteen seconds: how many tickets, changes, or findings are open right now — and how many finish in a week? Divide the first by the second. That is how long each one really takes (Little's Law guarantees it, from counts alone — no estimates involved). Then the only question that matters: **does that cycle time work for your business?** If not, everything that follows shows you where the time goes, and why.

> **Governing principle.** Find where time is trapped. Ask why. Check the evidence. Then choose what to change.

Core distinction, enforced throughout. A long wait is **observed**. A mechanism is **hypothesized**. Only operational evidence makes it **verified**. A measured post-fix gain makes it **improved**. "Bottleneck" is reserved for an evidenced throughput constraint — never a synonym for "largest wait."

EXECUTIVE SUMMARY

In most security operations processes the work is fast and the elapsed time is not. Items spend the overwhelming majority of their life waiting — between steps, for approvals, for other teams, for scheduled windows. Process Cycle Efficiency (PCE), the ratio of hands-on time to total elapsed time, routinely lands below 10%. The gap is invisible in activity metrics and utilisation dashboards, because busy people and slow processes coexist comfortably.

PFV makes the gap visible with a per-step decomposition: a worksheet a practitioner completes in about thirty minutes, a heat map showing where the waiting concentrates, and a Little's Law cross-check that catches self-deception in the estimates. This edition adds the step that was missing between finding a delay and spending against it — **a structured diagnosis of why each top wait exists**, improvement actions matched to that diagnosis, and a financial model whose starting assumptions are bounded by the strength of the evidence behind them.

What is at stake is not abstract: MTTD, MTTR, patch latency, and exposure windows are all, in large part, queue time.

THE PROBLEM: THE BIGGEST COST HIDES BETWEEN THE STEPS

Ask a team where the time goes and they describe the work. Ask the timestamps and they describe the waiting. The two answers rarely match, because human attention tracks effort, not elapsed time. A step with two hours of work and three days of queue is remembered as "two hours." Multiply that across a workflow and an organisation systematically misreads where its time — and therefore its money and its risk — actually sits.

The failure mode is familiar: improvement effort flows to the visible work (tools, training, headcount at busy steps) while elapsed time barely moves, because the elapsed time was never in the work. Across alert triage, incident response, vulnerability remediation, firewall change approval, and detection engineering, this is the difference between optimising effort and recovering time.

THE METHOD: PER-STEP PCE DECOMPOSITION

PFV asks for two numbers per step — processing time (hands on the task) and queue time (elapsed wait before the step starts) — plus a defect rate and the handoff structure. From these it computes per-step and whole-flow PCE, colours every cell against thresholds, and flags the dominant waits. Rework loops are modelled explicitly: work that returns re-enters the flow and consumes real capacity.

Two disciplines keep the estimates honest. First, a **Little's Law cross-check**: open items divided by throughput gives an independent estimate of lead time; if it disagrees with the mapped total, the map is missing waiting. Second, the **trust ladder**: every number is labelled with the evidence behind it, and the toolkit will not let an estimate masquerade as a measurement.

For example, the worksheet may show that a firewall change spends 1.5 business days queued at CAB approval. That finding is real and useful. It is also, on its own, only a *location*.

FROM DELAY LOCATION TO DELAY MECHANISM

Per-step decomposition answers *where* the time goes. It does not, by itself, answer *why*. A dominant queue is a diagnostic signal — the place to investigate — not a proven cause, and certainly not a proven bottleneck. A bottleneck (a step whose limited capacity constrains the throughput of the whole system) is only one of several reasons a queue grows. The same symptom can arise from six different mechanisms, and each demands a different response. Hiring another approver does nothing for a queue that is really a weekly batch; adding capacity to a step that is waiting on another team just moves the idle time.

Mechanism	What it is	Evidence that confirms it	Watch out
Capacity constraint	Demand persistently meets or exceeds what the step can complete.	WIP accumulates before the step; the resource is highly utilised; arrivals ≈ completions; the queue grows over time.	Concentration is not capacity. The longest wait is not automatically a bottleneck.
Batching / schedule	Work is held for a meeting, review cycle, window, or minimum batch size.	Items wait while capacity is free; completions cluster around the cycle; wait tracks arrival timing within it.	The most common false capacity signal. An idle approver with a long queue is almost always batching.
Dependency / blocking	Work cannot proceed until another team, vendor, input, or decision arrives.	Blocked status; external ownership of the next action; queue age tracks their response time, not this step's capacity.	Adding capacity here does nothing — the step was never the limiter.
WIP overload / switching	Too many items are active at once; multitasking ages everything.	High active-item count per person; many started, few finished; frequent expedites; aging WIP while new work keeps starting.	Staffing can look adequate while the queue grows. Attention, not headcount, is the scarce resource.
Rework loop	Work repeats because inputs were incomplete or defects surfaced.	Items move backward; reopenings and resubmissions; high rejection or clarification rate.	Rework consumes real capacity and reads as "not enough people."
Policy / control	The wait is required by design — approval, separation of duties, compliance.	The delay is intentional; the control owner can name its risk purpose; removing it would change a control objective.	Not automatically waste. Ask whether the objective can be met in less elapsed time.

A seventh verdict — **Unknown, needs evidence** — is legitimate, common, and honest. When the indicators do not support a classification, go and look before prescribing anything.

> **PFV identifies where time is trapped. Delay diagnosis determines why it is trapped. Improvement follows diagnosis — it never precedes it.**

CONFIDENCE, AND WHAT EACH LEVEL CAN CARRY

THE CONFIDENCE LADDER — FOUR LEVELS

Every claim the method produces stands on one of four levels, and the toolkit labels which one you are on:

- **Observed** — a large wait was found. The worksheet located it; nothing about its cause is known yet.
- **Hypothesized** — a likely mechanism. You have named a plausible "why" from the six, but not yet backed it with evidence. Citing concrete indicators you actually see raises this to *evidence-supported*,

which is still self-reported.

- **Verified** — your logs confirm it. The same timestamps that verify per-step times also show what the work was actually waiting on.
- **Improved** — the intervention produces measurable operational gains. You changed something, re-measured the same number, and it moved as predicted.

> **The first three levels are confidence in the *cause*. "Improved" is a different kind of claim** — evidence that the *remedy* worked, not that the diagnosis was right. The two can come apart: a fix can succeed for reasons other than the mechanism you named. For exactly that reason, "Improved" never feeds the financial model. It is the level you report from, not the level you project from.

Never take a hypothesized cause upstairs as a proven one, and never call the largest wait a "bottleneck" without throughput evidence. More than one mechanism can be true in a single wait.

Instrumented ongoing measurement is not a rung. It is an optional continuous-improvement and maintenance capability: it is how you re-measure after an intervention — which is what earns **Improved** — and how you notice when a process that was fixed quietly drifts back.

HOW THE TOOLKIT RUNS IT

THE PIPELINE: MAP → DIAGNOSE → PRESCRIBE → MODEL

The toolkit runs the argument in one direction, and each stage feeds the next.

- **Map.** The Process Flow Worksheet captures up to twenty steps of a real process — processing time, queue time, defect rate, handoffs — and renders a heat map. Per-step decomposition answers *where* the time goes.
- **Diagnose.** For the one or two largest waits, "Diagnose your top waits" asks two questions per step: which of the six mechanisms most often causes work to wait here, and which concrete evidence indicators you actually see. Each card carries the step's **task number**, so every finding traces back to the exact heat-map column. Picking a mechanism yields **Hypothesized**; two or more evidence indicators upgrade it to *evidence-supported (self-reported)* — deliberately not "Verified," because the tool cannot read your logs.
- **Prescribe.** "Improvement Actions for Your Diagnosis" has no diagnosis engine of its own. It reads your diagnosis and lists the curated actions matched to each mechanism — framed as candidate experiments to test, never guaranteed fixes — and names the recovery lever each one feeds.
- **Model.** The Investment & Return Calculator routes each diagnosed mechanism to its lever: capacity, batching, dependency, and WIP feed **Reduce Waiting**; a rework loop feeds **Reduce Rework**; policy

overhead feeds **Eliminate Unnecessary Work**. Your confidence then positions the modelled improvement *inside* that lever's fixed band.

> **The bands never move.** A hypothesis defaults to the conservative end of its band (Reduce Waiting 15%, Reduce Rework 35%); evidence-supported earns the mid-band (25% / 50%). The aggressive end is reserved for log verification and is deliberately unreachable from estimates alone. You can always type your own number — a manual override wins and persists. No diagnosis can manufacture a bigger promise than the band allows, and no self-attestation unlocks the top of it. Before any dollar figure appears, a scenario gate ("Before the dollars — read this once") requires you to acknowledge you are looking at an estimate.

WORKED CONTRAST: THE SAME SYMPTOM, OPPOSITE FIXES

Firewall change approval. The longest wait in the flow — yet the approvers are idle much of the week while changes wait for the weekly CAB. This is batching, not capacity. The useful experiment is a second review window, or risk-tiered standing approvals for low-risk changes — not another approver. Adding capacity to an idle step changes nothing.

Detection engineering. The team asks for headcount, but rules keep moving backward from production to tuning. The constraint is rework, not staffing: each redo consumes capacity that reads as new demand. The experiment is tighter entry criteria and pre-deployment test coverage.

Both look identical on the heat map. Only the diagnosis separates them — which is why the method refuses to let the map alone choose the remedy.

PRIOR ART, AND WHAT IS NEW

None of the underlying mathematics is novel, and the method does not pretend otherwise. Process Cycle Efficiency and value-stream decomposition come from the Lean Six Sigma tradition; the queue-time discipline and batch-size economics from Reinertsen; the estimation and calibration habits from Hubbard; the flow-conservation cross-check from Little's Law; and the cost-of-delay-versus-holding-cost framing traces to Harris's 1913 economic order quantity work. The sources, in full:

- Michael L. George, *Lean Six Sigma: Combining Six Sigma Quality with Lean Production Speed* (McGraw-Hill, 2002) — Process Cycle Efficiency and value-stream decomposition.
- Donald G. Reinertsen, *The Principles of Product Development Flow: Second Generation Lean Product Development* (Celeritas Publishing, 2009) — queue economics, batch-size effects, WIP constraints, and cost of delay.

- Douglas W. Hubbard, *How to Measure Anything: Finding the Value of "Intangibles" in Business* (Wiley, 2007; 3rd ed. 2014) — calibrated estimation and 90% confidence intervals. With Richard Seiersen: *How to Measure Anything in Cybersecurity Risk* (Wiley, 2016) — the same discipline applied to security decision-making.
- John D. C. Little, "A Proof for the Queuing Formula L = λW," *Operations Research* 9(3), 1961 — the flow-conservation law behind the worksheet's cross-check.
- Ford W. Harris, "How Many Parts to Make at Once," *Factory, The Magazine of Management* 10(2), 1913 — economic order quantity; the earliest ancestor of the toolkit's delay-cost-versus-batch framing.

PFV's contribution is the packaging and the posture: per-step PCE decomposition applied across the security operations portfolio, self-serve in a browser with nothing installed, an explicit evidence label on every number, a delay-mechanism diagnosis standing between the map and the money, and an integrated business case whose defaults are bounded by confidence. The claim is not new mathematics. It is a defensible route from a thirty-minute estimate to a number a CFO can interrogate — one that never overstates what is known.

LIMITS

- Estimates are estimates. Worksheet outputs stay labelled **Observed** / **Hypothesized** until logs verify them.
- The diagnostic's ceiling is evidence-supported self-report. It cannot read your logs; only you can promote a mechanism to **Verified**.
- The calculator models scenarios, not forecasts. Its bands are conservative by construction, capped, and gated behind an explicit acknowledgment.
- **Improved** is reported, never projected. Post-fix gains enter the story only after you re-measure — they are not an input the model will project from.
- Exposure-window language stays conditional and scoped: shortening a detection or remediation flow narrows the window in which a threat can act. PFV does not claim a causal reduction in breach probability.

CONCLUSION

The method now runs end to end: map a process in thirty minutes; see where the time is trapped; diagnose why each top wait exists; test the action matched to that diagnosis; and model the recovery with assumptions the evidence can actually carry. Every number stays labelled with the rung it stands on. Try it on one process at **csi.dewood.org** — the worksheet, the diagnostic, and the calculator are self-serve, and your process data stays in your browser.