PROCESS FLOW VISIBILITY · CYBERSECURITY OPERATIONS

PFV PRACTITIONER GUIDE

Map it. Diagnose why it waits. Act on evidence.

Second edition · Matches the shipped toolkit at csi.dewood.org · Seven steps, one process, about an hour end to end

> **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.

> **How to read this guide.** Each step matches a section of the toolkit page, top to bottom. Do them in order the first time — the pipeline is one-directional on purpose: the diagnosis feeds the recommendations, and the recommendations feed the calculator.

STEP 1 — MAP THE PROCESS (ABOUT 30 MINUTES)

Pick one process you run repeatedly; alert triage, incident response, vulnerability remediation, firewall change approval, and detection engineering are all good candidates. In the Process Flow Worksheet, name it and add its steps as columns (up to twenty; five to eight is typical). You can also load a benchmarked sample from the dropdown and edit it rather than starting from a blank sheet.

For each step record the performer, how work arrives, predecessor tasks, **processing time** (hours of hands-on work), **queue time** (business days it waits before the step starts), the defect rate, and how output leaves. Count the waiting honestly — a normal week, not a good one. If a step's output comes back to be redone, activate its rework loop so the redo consumes capacity in the maths.

STEP 2 — SCORE, THEN CROSS-CHECK

Cells colour green, yellow, and red as you type. Red is not shame; it is signal. The summary computes per-step duration and whole-flow efficiency.

Before believing any of it, run the **Little's Law cross-check**: open items ÷ items finished per year ≈ lead time per item. If that disagrees with your mapped total, the item count is right and the map is wrong — go back and find the waiting you missed.

STEP 3 — SEE WHERE, AND ONLY WHERE

The largest queues tell you *where* to investigate. Resist the reflex to fix anything yet. The biggest wait does not tell you *why* it exists, and it does not prove you found a bottleneck. Note the task numbers of your top one or two waits; the next stage carries them forward so every finding traces back to its heat-map column.

STEP 4 — DIAGNOSE YOUR TOP WAITS

The "Diagnose your top waits" section shows one card per dominant wait, labelled **TASK N** with its queue days. Each card asks two questions:

- **Q1 — what most often causes work to wait here?** Pick from the six mechanisms, or "Unknown — needs evidence."
- **Q2 — which of these do you actually see?** An evidence checklist that adapts to your Q1 answer. Check only what you have genuinely observed.

The output names the likely mechanism and your confidence. A mechanism alone reads **Hypothesized**; two or more evidence indicators read *evidence-supported (self-reported)* — deliberately not **Verified**, because the tool cannot read your logs. It also gives a first reversible test, what to measure, and the direction to expect if the hypothesis holds.

THE SIX MECHANISMS AT A GLANCE

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.

A DECISION PATH

- Held for a schedule or window? → **Batching.**
- Blocked on an external prerequisite or team? → **Dependency.**
- WIP piling up before a highly-utilised step? → test for **Capacity.**
- Many items active at once, or constantly interrupted? → **WIP overload.**
- Work repeats or moves backward? → **Rework loop.**
- The wait is required by design? → **Policy / control.**
- None clearly apply? → **Unknown** — collect evidence before prescribing.

The order matters: it rules out the cheap-to-confirm, non-capacity explanations first — the ones most often misread as "we need more people."

COMMON FALSE DIAGNOSES

- **Longest wait = bottleneck.** Length locates; it does not explain.
- **Idle-approver batching read as capacity.** A long queue with a free resource is almost never capacity.
- **Rework read as understaffing.** A busy step may be redoing work, not doing new work.
- **Policy delay treated as pure waste.** Ask what the control is for before you cut it.
- **Forcing a verdict.** "Unknown, collect evidence" beats a confident wrong answer.

STEP 5 — REVIEW THE MATCHED ACTIONS

"Improvement Actions for Your Diagnosis" prescribes; it does not diagnose. Before you complete Step 4 it simply asks you to diagnose first. Afterwards it lists the curated actions matched to each mechanism you chose — labelled with the task, your confidence, and the calculator lever the mechanism feeds.

Treat every action as a *candidate experiment*, not a guaranteed fix: run one small and reversible, and re-measure the same number — queue time, WIP, throughput, rework rate, or blocked time — to see whether it moved in the predicted direction.

STEP 6 — MODEL THE RECOVERY

The Investment & Return Calculator opens behind a one-time gate: *"Before the dollars — read this once."* It restates the four confidence levels and requires you to acknowledge you are looking at an estimate. Then:

- Each diagnosed mechanism routes to its lever — capacity / batching / dependency / WIP → **Reduce Waiting**; rework → **Reduce Rework**; policy → **Eliminate Unnecessary Work** — and the lever is highlighted with your mechanism and confidence.
- The modelled improvement **defaults to a confidence-bounded point inside that lever's fixed band**: a hypothesis lands at the conservative end (Reduce Waiting 15%, Reduce Rework 35%); evidence-supported earns the mid-band (25% / 50%). The aggressive end is reserved for log-verified data and cannot be reached from estimates.
- You can always type your own number. A manual override wins and persists, even if you change the diagnosis afterwards.

> **What the model never does.** The bands themselves never move; no diagnosis unlocks the aggressive end; and **Improved** has no code path into the numbers. A measured post-fix gain is something you report — it is not an input the calculator will project from.

Verify before you commit budget. Pull the flagged steps from your status-transition logs your systems already record — Jira issue changelogs, ServiceNow task audit tables, SOAR case timelines, and check both the times and the mechanism. Verification is what turns a self-reported diagnosis into a measured cause — and moves your numbers up the ladder to **Verified**.

STEP 7 — EXPORT, ACT, AND CLOSE THE LOOP

Export the fully formatted .xlsx (colours, rework loops, and summary preserved) for the record and the review meeting.

Then run the chosen experiment and **measure again**. If the number moves as predicted, the mechanism held — and only now have you earned the fourth level, **Improved**: the intervention produces measurable operational gains. If it does not move, the mechanism was something else; return to the decision path. If you want this to stay true, adopt instrumented ongoing measurement — an optional maintenance practice, not a higher grade of confidence, and the way you catch a fixed process quietly drifting back.

This is the loop the whole method runs on: **map, measure, diagnose, test, measure again.**

> Everything in this guide is self-serve at **csi.dewood.org**. Your process data stays in your browser unless you explicitly export it.