Load vs Throughput Lab

Facilitation guide · team, one facilitator

What this is: a 20–30 minute diagnostic you run when the team is flat out but nothing is shipping. You follow one real piece of work all the way through your system, split its life into the time it was *progressing* and the time it was *trapped*, and work out the ratio. That number tells you where the team's capacity actually goes. You finish by naming the biggest sink — and the first thing you'll change on Monday. Your job is to keep it a **trace**, not a log: follow the work, and insist on what really happened.

The one line to hold: teams rarely lose throughput because they're idle — they lose it because work stops moving. **Activity is easy to see. Flow isn't.** You can see who's busy; you can't see the hours the work spent waiting for an approval or a decision, being switched away from, or being redone. This lab makes that visible.

"Why only one ticket?" Expect this question, and answer it plainly: **one honest trace gives one system-level answer.** The states this item got stuck in are the states everything gets stuck in. Averages hide the story; one real work item has a narrative you can act on. You're not measuring the item — you're using it to reveal how the system behaves.

· WHEN DO I USE THIS?

Use this when

GOOD FIT

Everyone is flat out but nothing is shipping. You're being asked for more people and you're not sure that's the problem. Delivery has slowed and the usual explanations don't hold up. You need a diagnostic that names the biggest cause, not another dashboard.

WRONG TOOL

You need to know if you're over-committed › Capacity & Feasibility System. You know *which* piece of work is stuck and need to see where › Constraint Snapshot. You need to know who you're waiting on › Dependency Snapshot. You're letting too much work in at once › Execution Queue Control.

© Witmer

PART 1 RUNNING THE LAB

20–30 minutes

1 · WHAT IS THIS REALLY DOING?

The one idea

Busy people are easy to see, so teams naturally mistake activity for progress. What they can't see is the time work spends waiting, switching or being reworked — those losses leave no trace, so "work harder" becomes the obvious, wrong answer. Ask most delivery leads where their team's capacity disappears each week and they can't tell you — not because they're avoiding the question, but because the system doesn't show it. Following one real work item makes those hidden losses visible, and turns trapped time from a feeling into a number.

THE THROUGHPUT TRACE — ONE FEATURE, THIRTEEN DAYS

the work item's life, day by day:



PROGRESSING — TRAPPED — 11 days

15% progressing · 85% trapped

The effort wasn't missing. It was stuck.

One honest trace. One system-level answer.

The states this item got stuck in are the states everything gets stuck in.

2 · WHAT DO I NEED?

What you'll need

One real work item that recently finished (core)	A ticket, a feature, a deliverable — typical, not the worst case and not the showcase. Choose one that finished recently, while everyone still remembers it accurately.
The people who actually touched it (core)	They know where it really sat, and for how long. Without them you'll reconstruct the ideal workflow, not the real one.
The honest timeline	Every state the work passed through, written *in order*, and how long it sat in each. What actually happened — not what should have happened.
A whiteboard or wall space	Where everyone can see the trace as it builds — and, at the end, the final ratio. The reveal only works if the number is visible to the room.
One facilitator	Follows the work and holds the discipline. The capture sheet needs space for every state, the time spent there, and whether it was progressing or trapped.

Constraints · Manage Load · 20–30 minutes, depending on how many states the item passed through.

3 · HOW DO I RUN IT?

The run-sheet

Run these in order. The bold part is what you say or do; the move under it is where you add the most. The ratio (Step 4) is the finding — everything before it is setup, and everything after it is the intervention.

1. Pick one real work item	**3 min**	*"Which recently finished piece of work is typical of how we deliver?"* Not the worst case, not the showcase. Something the people in the room actually touched.
2. Trace its states	**10 min**	*"Where did this go, from start to finish?"* **Write each state as the team remembers it, in order, before discussing timings.** Then add how long it actually sat in each. Keep asking: **what *really* happened — not what normally happens.**

© Witmer

3. Split progressing from trapped	**7 min**	*Label every state as either **progressing** or **trapped**.* Progressing = someone actively moving it forward. Trapped = waiting (dependency, approval, **decision**), being switched away from, or being reworked.
4. Work out the ratio	**4 min**	*"What percentage of this work's life was it actually progressing?"* **Write the ratio where everyone can see it, then read it out loud.** This is the finding — **don't rush past it.**
5. Name the sink and the Monday change	**6 min**	*"Which single state ate the most time?" Then: **"What's the first thing we'd change on Monday?"*** Name the biggest sink. Then check it's worth fixing: **"would removing this sink change the ratio meaningfully?"** If yes, agree the change and **give it one owner**.

TRACE WHAT ACTUALLY HAPPENED — NOT WHAT SHOULD HAVE HAPPENED

This is the discipline the whole lab depends on. Teams reconstruct the *ideal* workflow remarkably fast: "well, normally it goes to review, then…" That's the process on paper, not the life this item actually had. Keep pulling them back: **"What really happened to *this* one?"** never **"What normally happens?"** The gap between the two is usually where the throughput went. And watch for the state nobody names: the days it sat waiting for someone to *decide* — not blocked technically, not being reworked, just waiting on a decision. That sink is invisible precisely because no one owns it. **Decision waiting hides because nobody owns it — so ask explicitly: "who could have decided this sooner?"** That question turns an invisible delay into a named, fixable one.

PART 2 GETTING IT RIGHT

the boundary & common mistakes

4 · WHICH TOOL IS THIS?

Where this sits — the Execution MRI

The Constraints tools form a diagnostic suite, each reading a different failure. The boundary is drawn by **level**, not by symptom: **this tool reads the whole system**; Constraint Snapshot reads a single workflow; Dependency Snapshot reads a specific dependency. Run them in that order — system first, then zoom in.

© Witmer

THE EXECUTION MRI — EACH TOOL READS A DIFFERENT FAILURE

CAPACITY & FEASIBILITY SYSTEM	“Are we committing to more than we can deliver?”
LOAD vs THROUGHPUT LAB (this tool)	**“Where is our capacity disappearing each week?”**
CONSTRAINT SNAPSHOT	“Where exactly is this piece of work stuck?”
DEPENDENCY SNAPSHOT	“Who are we waiting for?”
EXECUTION QUEUE CONTROL	“Are we starting too much at once?”

This tool reads the *system*. If it points at a problem area, Constraint Snapshot investigates that area in depth.

5 · WHAT IF IT GOES WRONG?

Common mistakes

What happened	The fix
The team traced the ideal workflow	The most common failure. “What normally happens” is the process on paper. Keep asking what really happened to *this* item.
You logged metrics instead of following the work	This is a trace, not a ledger. Follow the item on its journey, state by state — that’s what produces the narrative.
You never worked out the ratio	Without the progressing-vs-trapped number, this is reporting, not a diagnosis. The ratio *is* the finding.
Someone asked “why only one ticket?”	Answer it: one honest trace gives one system-level answer. The states it got stuck in are the states everything gets stuck in.
The item chosen was the worst case	An outlier tells you about the outlier. Pick something typical — that’s what reveals the system.
It became a blame conversation	The trace shows the leak, not a person. Redirect: “we’re not asking who — we’re asking where the work stopped moving.”
You named a sink but nothing changed	A diagnosis without an intervention is a dashboard. End with the Monday change — and an owner.

© Witmer

Before you close the lab

- ✓ You traced one real item, state by state — what actually happened
- ✓ The progressing-vs-trapped ratio is written where everyone can see it
- ✓ The single biggest sink is named — waiting, switching or rework
- ✓ You have the first change you'll make on Monday, and one owner for it

6 · HOW DO I MAKE THIS STICK?

Take it back to work

THE HABIT

When delivery slows and the answer sounds like "we need more people," ask first:
"What percentage of the time is our work actually moving?"

If most of it is trapped, more effort just feeds the same leak. Fix the flow before you add the capacity.

Appendix · Reference

You should not need this to start. Read it before, or if a run goes sideways.

7 · A WORKED RUN

Watch the facilitator move

The item: "a feature that took three weeks to deliver."

Stage	What happened
The assumption	Everyone remembers working hard throughout, so the team assumes it simply needed more capacity.
The trace	State by state: picked up · waiting on design sign-off · in dev · waiting on a decision about the API · parked for an incident · back in dev · waiting for review · rework · shipped.
The split	Two days progressing. Eleven days trapped — most of it waiting for approvals and one decision nobody owned.
The ratio	Written on the board: **15% progressing, 85% trapped.** Read out loud. The room goes quiet.
The sink and the change	"Waiting for decisions ate the most time." Monday change: a standing 15-minute decision slot with the sponsor. Owner: the delivery lead.

The team walked in asking for another engineer. They walked out knowing that a second engineer would have spent 85% of their time trapped too — and that the fix was a decision slot, not a hire.

© Witmer

· WHERE DOES THIS FIT?

The rest of the suite

The question	Use this
Where is our capacity disappearing each week?	**This tool — Load vs Throughput Lab.** Whole system. Trace one item, get the ratio, name the sink.
Are we committing to more than we can deliver?	**Capacity & Feasibility System** — commitments vs capacity, before you say yes.
Where exactly is this piece of work stuck?	**Constraint Snapshot** — one workflow, in depth. Use it *after* this tool points at the area.
Are we starting too much at once?	**Execution Queue Control** — intake and WIP. Nothing new enters until something finishes.

The move underneath: when a busy team isn't delivering, don't add effort — find out where the effort is going. Follow one real piece of work, split its life into moving and stuck, and let the ratio tell you the truth. The problem is rarely too little work. It's too little flow. **Fix the biggest sink first. Then trace another item** — the ratio is how you'll know whether it worked.

© Witmer