Requirement Drift Simulator

Facilitation guide · team, one facilitator

What this is: a facilitated activity you run on a piece of work whose scope has drifted. You make every accepted change physical — one block pulled from a tower per change — and let the team feel the work get unstable until it topples. It ends with a re-baseline: a decision about whether this is still the original project. Your job is to make the tower tell the team's real story, to let it actually fall, and to turn the collapse into one leadership decision with an owner.

The core mechanic — pull, don't stack. The baseline is fixed (the original scope, timeline and resources), and every accepted change pulls a block out of it. You are removing stability, not adding height. And **re-baseline** — the word this whole tool turns on — means formally adjusting the original scope, timeline or resources to match what's actually been accepted. Not "being more careful"; a concrete decision to change the baseline. The activity exists to produce one. **Drift** is the gradual change from the original baseline as those accepted changes build up.

The one line to hold: the problem isn't accepting change — it's accepting change without changing the baseline. Every change was probably reasonable; the customer learnt something, a real need surfaced. The tool doesn't judge the changes. It shows what they cost when the baseline stays frozen underneath them. **Every block is a real, accepted change — never a generic pull. And let the tower fall when it falls.**

· WHEN DO I USE THIS?

Use this when

GOOD FIT

A real project or deliverable that's absorbed a series of reasonable changes and now feels harder than its original scope explains — scope building up over time against a fixed baseline. The one test: can the team name an original baseline and several changes added since? If yes, this is the tool.

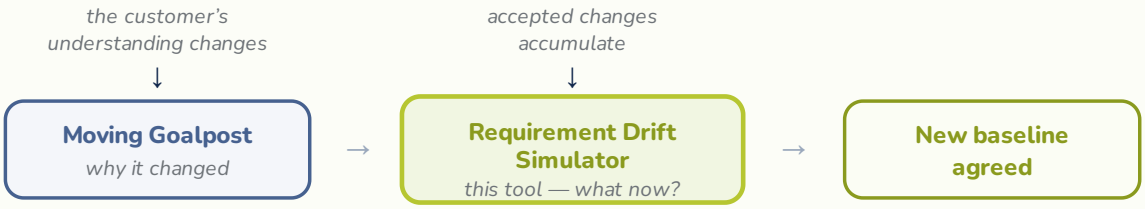
WRONG TOOL

A recurring crunch caused by one early decision cascading › Crunch Cycle Review. A team that won't stop gathering before it decides › Analysis Paralysis Lab (the other tower tool). A change being actively resisted › Surface Honest Resistance. If there's really just one big change, or no baseline to drift from, it's a different problem.

This tool has a partner. Moving Goalpost explains *why* requirements change — the customer's understanding moved. Requirement Drift Simulator shows what that costs when the baseline doesn't move too, and drives the re-baseline. Together they're a sequence, not two neighbours.

© Witmer

TWO TOOLS, ONE SEQUENCE



Moving Goalpost explains *why* requirements changed. This tool shows what that costs when the baseline doesn't.

The problem isn't accepting change. It's accepting change without changing the baseline.

PART 1 RUNNING THE ACTIVITY

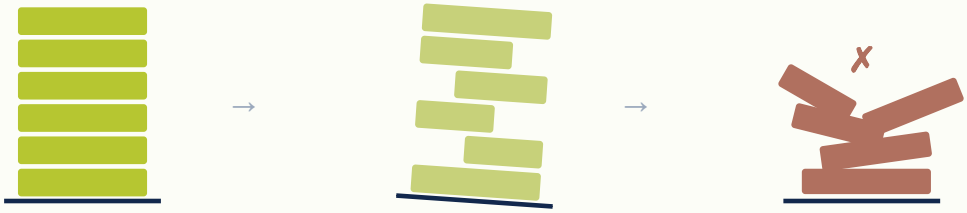
team, one facilitator

1 · WHAT IS THIS REALLY DOING?

The one idea

People judge changes one at a time, and each one passes — the customer learnt something, a real need surfaced, so the team says yes, correctly. But the original baseline stays fixed underneath all of it. No single change caused the instability — the build-up did, carried on a baseline that was only ever built for the first version. This tool makes that build-up physical, so the team stops arguing about which change was wrong and starts asking the real question: are we still building the project we committed to — or does today's work deserve a new baseline?

WHAT THE TOWER SHOWS



The instability is invisible per-pull and obvious in aggregate — which is exactly how drift behaves.

© Witmer

2 · WHAT DO I NEED?

What you'll need

One real project with a drifting scope (core)	A real project that's absorbed a series of changes, and its original baseline (the scope, timeline and resources you first committed to), written down. The team must recognise it.
The block tower (core)	A stacking-block tower (Jenga-style). Each block = one accepted change; pulling it stands for the load that change added to a baseline that never moved.
Change cards (write-on cards)	Name each real change before its block is pulled, so the tower tells the team's own story — not a generic game.
One facilitator	Guides the discussion, tags each block with a real change, keeps the baseline visible, and lets the tower fall when it falls. Never catches it to keep going.

Resilience · Adapt to Change · 30–45 minutes. Protect Step 2 (listing every change) and Step 5 (the decision) — a complete change-list and one re-baseline beats a dramatic collapse with no decision made.

3 · HOW DO I RUN IT?

The run-sheet

Brief the team, then run these in order. A briefing you can say almost word for word: "We started with a clear scope. Since then we've accepted changes — small ones, big ones, each reasonable at the time. We're going to make every change physical: one block pulled per change, and watch what the build-up does. Then we decide — freeze, re-baseline, remove, or grow. About 45 minutes."

1. Name the work and the baseline	**4 min**	*"What's the project, and what was the original scope — the baseline we committed to?"* Write the baseline where everyone can see it. The built tower represents that baseline: complete and stable.
2. List every change since	**8 min**	*"Now — every change we've accepted since. Every 'small' one too."* Write each on a card — one change per card; don't merge similar changes, or the mechanism weakens. Mine the small ones — the corridor, email and stand-up yeses. A long pile of small cards is a good sign. Don't judge whether each was justified; just capture it.

© Witmer

3. Build, then load change by change	**6 min**	*Read each change card aloud and pull one block for it. "Would you have rejected that change on its own?" — everyone says no.* After each, name it: "still standing — that change was reasonable too." You're building the case that no single change is the culprit.
4. Feel the drift — and let it fall	**10 min**	*Keep pulling. The tower leans, then a pull brings it down — don't steady it.* Sit in the silence a beat. Then: "which change brought it down? Was it unreasonable? No — and neither were the others."
5. Re-baseline — a leadership decision	**10 min**	*"Is this still the original project — or does today's work deserve a new baseline?"* Agree one response only: freeze, re-baseline, remove, or grow the baseline. Commit one decision, with an owner.

THE COLLAPSE IS THE WHOLE TOOL — DO NOT CATCH IT

The instinct is to keep the game going or soften the fall. Resist it. The team needs to see a *reasonable* change be the one that topples the work. After it falls: "which change brought it down? Was it unreasonable? No — and neither were the others." When the tower falls, read it back: **"but that last change was tiny!"** — exactly the point; the size of the last one isn't the cause, the build-up is. **"We had good reasons for each."** — yes, and that's why no one ever said no. **"So we should've rejected them?"** — not necessarily; you'd have re-baselined. If the tower won't fall, you ran out of changes first — name it honestly, then ask "how many more before it goes?" The lean is still the finding.

PART 2 GETTING IT RIGHT

the decision, the boundary & mistakes

4 · TURNING THE COLLAPSE INTO A DECISION

Re-baseline — the four options

End on a decision, not a realisation. The felt collapse is powerful but it isn't the output — the re-baseline is. If the team just nods and says "we should be more careful," push: "that's a realisation, not a decision. What do we stop, re-scope, remove or grow — and who owns it?" This is a leadership call: whether the original commitments still hold. A script for forcing it: "We're not leaving with 'we should be more careful.' We're leaving with a decision — freeze, re-baseline, remove, or grow — and who owns it." There are four honest ways to re-baseline — agree one:

© Witmer

FOUR WAYS TO RE-BASELINE — CHOOSE ONE

FREEZE SCOPE	NEW BASELINE	REMOVE WORK	GROW BASELINE
deliver the baseline; stop accepting	agree a new, honest baseline; re-set date	something must come out to make room	expand scope, time and resource together
when the date matters most	*when the changes are necessary*	*when date and resource are fixed*	*when all changes are genuinely needed*

5 · WHY "PULL, DON'T STACK"?

Why it's not the other tower tool

This tool and Analysis Paralysis Lab both use a tower, but the move is opposite — and so is the lesson. Here the baseline is fixed and every change *takes* from it; the instability rises. There, blocks are *stacked* per piece of analysis and the height rises. If you find yourself stacking blocks, you're running the wrong tool.

TWO TOWER TOOLS — OPPOSITE MOVES

REQUIREMENT DRIFT (this tool)
PULL a block per change
the fixed baseline weakens with every reasonable change
felt moment: the wobble, then the fall
lesson: changes accumulate into collapse

ANALYSIS PARALYSIS LAB
STACK a block per analysis
the height grows as the team keeps over-gathering
felt moment: the teetering height
lesson: you'll never feel "ready"

6 · WHAT WILL THE TEAM SAY?

What they say — and your move

These are the lines that stall the run during the collapse. Each has a one-sentence response.

What they say	Your response
"This is just Jenga / a game."	"The blocks are your real changes. Watch what topples it — that's your project."
"The changes were the client's fault."	"Maybe — but we accepted them. This is about seeing the load, not assigning blame."
"We can't un-accept changes."	"Right — so we re-baseline. That's the decision, not undoing the past."
"It fell by accident, bad pull."	"A stable tower survives a clumsy pull. This one was already unstable."

© Witmer

7 · WHAT IF IT GOES WRONG?

Common mistakes & edge cases

What happened	The fix
Debating whether a change was justified	It doesn't matter — the point is the load, not the verdict. Changes are often legitimate; that's the premise, not the debate.
Debating whether each change was "worth it"	Stop the debate. The activity isn't judging the changes — it's showing what they accumulated into. Changes are often legitimate; that's the premise.
Arguing about who approved it	Turns the room into a blame session and kills the insight. Redirect to the accumulation.
Catching the tower to keep it going	The collapse is the lesson — let it fall. A facilitator who catches it has thrown away the point.
Ending at the collapse	The felt moment isn't the output; the re-baseline is. Force the choice: freeze, re-scope, remove or grow — and an owner.
Generic change cards	Every block must carry a real, named change, or it's just Jenga.
The team lists only big changes	Push for the small ones — the corridor yeses. The trickle is what this tool surfaces.
The work genuinely needs all the changes	Then re-baseline up — formally grow scope, time and resource to match. "Grow the foundation," not only "stop."

Before you close the activity

- ✓ Every block was a real, named change — not a generic pull
- ✓ The team felt the drift arrive suddenly, and saw the toppling change wasn't the culprit
- ✓ The room asked the leadership question: is this still the original project?
- ✓ You left with one decision — freeze, re-baseline, remove or grow — and an owner

© Witmer

8 · HOW DO I MAKE THIS STICK?

Take it back to work

THE HABIT

Before accepting the next change, ask:

“Are we changing the scope — or changing the baseline?”

Four questions to run before the next change: **what does this add? what does it delay? what must come out if this comes in? are we changing the scope, or the baseline?** The last one is the real lesson: the failure was never accepting changes — it was failing to re-baseline when we did.

© Witmer

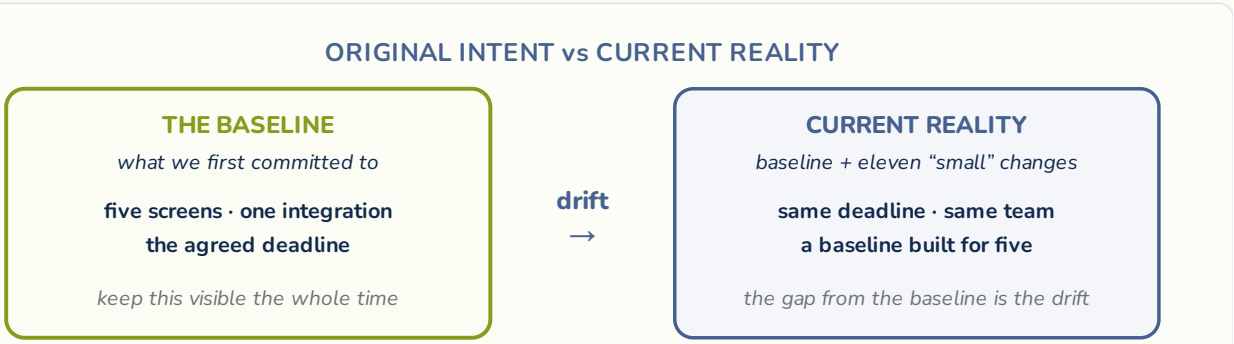
Appendix · Reference

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

9 · A WORKED RUN

Watch the facilitator move

The work: “the Q3 portal release. Baseline: five screens, one integration.” The gap between what was committed and what the team is now carrying is the drift:



Stage	What happened
Changes listed	“Extra report. SSO. A second user role. ‘Quick’ audit log. Mobile view. Another approval step…” — eleven cards.
Loading (read flat)	Pull for ‘extra report’ — stands. Pull for ‘SSO’ — stands. “Still fine. Every one of these was reasonable.”
The collapse	On ‘mobile view’ the tower leans hard. Don’t steady it. Next pull — ‘another approval step’ — it falls. “That change was tiny. It’s not the cause. The other ten are.”
The decision	“Freeze scope. Re-baseline to original plus SSO and the second role; defer the rest to phase two.” Owner: Sam.

The collapse made visible what eleven reasonable yeses had hidden — the baseline was only ever built for five. The team didn’t regret the changes; they re-baselined to carry them. They accepted every change — what changed was the baseline, not the customer’s learning.

© Witmer

· WHERE DOES THIS FIT?

Where this sits in the Gym

The shape of the problem	Use this
Many accepted changes accumulate until the work is unstable	**This tool — Requirement Drift Simulator.** Accumulation made felt. Pull, don't stack.
The customer's understanding keeps changing — and you need to know *why*	**Moving Goalpost** — the partner tool. It explains why the requirements moved; this one shows what that costs. A sequence: why → what now.
One early decision cascades into a recurring crunch	**Crunch Cycle Review** — causation, not accumulation. One thing setting off the next.
The team won't stop gathering before it decides	**Analysis Paralysis Lab** — the other tower tool. Set a threshold; you'll never feel ready.

The move underneath: when reasonable changes pile onto a fixed baseline, don't debate which one was wrong — make the build-up physical, feel it topple, and decide whether the baseline itself must move. That shift — from "was that change okay?" to "is this still the original project?" — is the whole tool. Accept the change; then change the baseline too.

© Witmer