PLAYBOOK · MAINTAINED REFERENCE

The Retrospective Playbook

Ed Schaefer · Version 1.0 · August 2026 · edward-schaefer.com/playbooks/retrospectives · Free to share

A good retro and a bad one can look almost identical on the agenda: same three questions, same block on the calendar. What's different is the execution: whether the team leaves with real value and stays engaged, or spends the hour going through motions everyone's tired of. The most common way it goes wrong: what I call the grump and dump — showing up to vent about what's broken without any of it turning into action. This document works two ways: read it start to finish and run a real retro this cycle, or keep it open as a live reference once the shape is familiar. Point an agent at it and it can do either job.

1. The 60-second version

- **Who's in the room:** team-only. It sits between the review and the next planning session, after one, before the other.
- **Duration:** 90 to 120 minutes is the sweet spot; 60 runs tight, and 30 works only for a long-lived, well-practiced team. (Full ladder, including the Scrum Guide's cap, in Section 8.)
- **Facilitation:** a dedicated expert facilitator, someone with real facilitation training, is preferred over rotating the job through the team.
- **It isn't done until:** the retro produces real actions with owners attached, entered into the backlog. Short of that, it didn't finish.
- **The default failure:** the grump and dump — venting that never turns into anything.

2. The default play

This is my own synthesis, from years of running these and a shelf of retrospective books — a few dozen at this point. Three shaped this structure the most: Esther Derby and Diana Larsen's *Agile Retrospectives*, Sam Kaner's *Facilitator's Guide to Participatory Decision-Making*, and R. Brian Stanfield's *The Art of Focused Conversation*.

Welcome and check-in (when you use one). Sometimes a check-in or an emotion check, sometimes an icebreaker or a team-building warm-up. Not every retro needs one.

Set the stage. Name what this session is for — a look at the last cycle, aimed at real actions — and how it will run: the format, the structure, what happens when.

Idea generation. Diverge early. Silent writing is the default start, so the first person to speak doesn't anchor the room, but it isn't a requirement — small-group structures work too, and most of them (1-2-4-All, from Liberating Structures) start with silent writing anyway. Sometimes all the generation happens up front; sometimes it runs section by section.

Review and group. Put everything up where the room can see it. Group what people think is similar, and check that it actually is: "I was thinking this" — "oh, I was thinking this." Either it's the same point, or it's now clear they're two different ones.

The groan zone. The messy middle: discuss and ask questions until people actually understand what's on the wall. With enough time, this runs on the full set before any narrowing. More often, the grouping above has already narrowed things, and the groan zone runs on what's left.

Converge: vote. Converging is a vote, one way or another. Vote once people feel confident they understand what they're voting on, then take the top five or ten — however many the remaining time can carry.

Work the top items. For each one: what's the experience, what's the problem, what's causing it, and what do we need — in our control, in our influence, or somewhere we need outside help.

The closing ownership beat. Before the room breaks, name what's happening with each item that made the cut, and who owns it. Each one goes into the backlog now.

Plenty of retros run as multiple parts or activities — this first, then that. The sequence above is the default shape, not the only one. The shape traces to the three books named at the top: Derby and Larsen's five-stage skeleton (Set the Stage, Gather Data, Generate Insights, Decide What to Do, Close) underlies the sequence; Kaner's Divergent Zone, Groan Zone, and Convergent Zone shape the middle; and the work-the-top-items conversation is ORID-shaped, the Objective/Reflective/Interpretive/Decisional frame Stanfield lays out.

The reasoning behind who runs it: if nobody's dedicated to facilitating, it usually means one person is sitting out from participating instead, and I'd rather have a facilitator and have everyone participate than the other way around.

3. Script: running the retro

Open with the check-in (when you're using one).

> "Before we get into it, how's everyone doing? Anything on your mind before we start?"

Set the stage.

> "Here's what we're here to do today: dig into what's actually causing the friction from this sprint, and leave with real actions and owners. Here's how we'll run it: ideas first, then grouping, then we vote and work the top items. We're keeping this to the sprint that just wrapped."

Idea generation.

> "Take a few minutes, write down what's on your mind, anything you noticed this sprint. Don't worry about the wording, and don't discuss yet. Let's get it all down first."

Review and group.

> "Let's group the ones that look similar — and check they're actually the same point, not two different ones that sound alike."

The groan zone.

> "This is the messy part: let's talk these through and ask questions until we actually understand what's up here."

Converge: vote.

> "Everyone gets a few votes. Vote for what matters most, and we'll spend our time on the top items."

Work the top items.

> "For each of these: what's the experience, what's the problem, what's causing it, and what do we need? Is it something in our control, something we can influence, or something we need help with from outside the team?"

The closing ownership beat.

> "Before we close: here's what we're doing about each of these, and here's who owns it. These go into the backlog now, same priority as anything else next sprint."

4. Script: rescuing a grump and dump

The retro tips into complaining sometimes, and that's not automatically a crisis. Sometimes you do need that, sometimes people need space to just complain or vent. What actually fails the retro is stopping there, never turning to what's next.

Let it run for a minute, honestly.

> "Okay, I'm hearing a lot of frustration here. Let's take a minute and just get it out: what's actually bothering people?"

Then pivot, deliberately.

> "We've named a lot of what's wrong, and that's real. Here's the question now: what's actually in our control, what can we influence, and what do we genuinely need help with from outside the team?"

Name the trap directly if it's still circling.

> "Right now we're giving up our own control here, saying nothing can be done. Let's pick one thing that's actually ours to fix, and start there."

Land it the same way every retro lands.

> "Same as always: one action out of this, and who owns it?"

5. What belongs, what doesn't

Segment	Belongs	Doesn't belong
Idea generation	Individual observations, written down before discussion starts, so nobody's opinion anchors the room	Reacting to what's being written (that's the next phase)
Group, groan zone, vote	Grouping similar items and checking they really match; the groan-zone discussion; a vote to narrow to the top items	Jumping from "here's the list" straight to the close without ever converging on what matters most
Work the top items	What's causing it, why it's a problem, and what's in the team's control, influence, or needs outside help, for the items that made the cut	Vague commitments with no owner; problems restated with no next step attached
Follow-through	Actions with named owners, added to the backlog at the same priority as other work	Actions that live only in the meeting notes; fixes tacked onto a sprint plan that already has no room for them

When the only record of an action is the meeting notes, and nothing ever reaches the backlog, that's the clearest sign the retro didn't actually finish.

6. Follow-through

The retro is planned so that part of its outcome, from the start, is the action items: what the team is going to tackle and who owns each one. From there, those become improvement stories in the backlog, the owner carries them, and the sprint's scope gets managed so they get real priority, the same as any other committed work.

Worth being honest about where the canon has moved on this. The 2017 Scrum Guide mandated it: the sprint backlog had to include at least one high-priority process improvement from the previous retrospective. The 2020 guide changed that language to optional: a team may add it. My default keeps the older, stricter version: action items get planned for the same way any other backlog item does. SAFe's Improvement Story, a named backlog item type built for exactly this, sets a concrete cadence: one to three improvement stories per iteration. It's a corroborating precedent for treating it as real, planned work.

Where the sources disagree

Not everyone agrees improvement items belong in the backlog at all. Kelly Lewandowski's rule is that most retro items shouldn't go into Jira: only something an outside party (a customer, the PM, the company) would be disappointed by belongs there; something like "communicate better about blockers" isn't specific enough to be a ticket. It's a reasoned position. My default stands anyway: an action item that doesn't get the same protection as other backlog work usually competes for time it loses, which is the items-identified-no-time-to-do-them failure from Section 7 happening again.

7. Failure modes and fixes

The grump and dump. We show up, complain about what's broken, dump our problems, and vent about how dumb the company or the client is. But none of it turns into anything effective. Underneath it: giving up locus of control, deciding out loud that nothing can be done, so no actions come out of the room. An independent academic match for the same pattern: Matthies and Dobrigkeit's "All Talk–No Action" headache (HICSS-53, 2020), reached from separate fieldwork. Fix: honor the vent, then pivot to what's in the team's control or influence (Section 4), and land on at least one owned action before the meeting ends.

Going through the motions. The same three questions (what went well, what didn't, what should we change) run sprint after sprint with nothing different about the outcome. It's boring, and people would rather be somewhere else. Fix: don't let the close skip the "what can we do about it" step just because the questions feel familiar.

The format changes, the problems don't. Fun retros aren't automatically the fix: they can be the exact same problem with a different coat of paint. Fix: judge any retro by whether the underlying issue changed, no matter how different the format felt.

Items identified, no time to do them. The retro does its job: real problems get named. But the sprint plan has no room for them, so the fixes get tacked on top or quietly dropped. Fix: this is what Section 6 exists to solve. Plan the retro so its output is action items from the start, then protect their priority in the next sprint the way any other committed work gets protected.

The facilitator who's also a participant. Section 2's point applies here too, inverted: if nobody's dedicated to facilitating, someone's usually sitting out from participating. Run it the other way, one person facilitating and fully participating at the same time, and the same trade happens in reverse.

Something suffers, either the facilitation or that person's own voice in the room, because doing both jobs completely at once is rare. Fix: keep the two roles separate, even when it means the facilitator's own material gets less airtime that meeting.

8. Variations

Duration. Ninety to a hundred and twenty minutes is the sweet spot. Sixty minutes runs tight. Most of that time goes to discussion, and outcomes get squeezed. Thirty minutes works, but only for a team that's long-lived and already well-practiced together. Whatever the sprint length, the retro's own ceiling is the Scrum Guide's cap: three hours for a one-month sprint, proportionally less for shorter ones.

Facilitation models. All four ways of doing this work in practice: rotating the job through the team, bringing in an outside facilitator, one dedicated team member taking the role permanently, or a dedicated role built for it, a scrum master or agile coach with real facilitation training. My preference is the last one: a dedicated expert. If a team rotates instead, pairing whoever's up that sprint with real expert support keeps the quality from swinging with whoever's turn it is.

Turn up the good. The failure modes above are the diagnostic half of a retro. There's a positive half too. "Turn up the good" is a phrase I've heard before, not one I came up with; it traces to Woody Zuill. The practice: name what's working and deliberately do more of it. A retro that only ever hunts for problems is only doing half the job.

Other formats and structures. There are tons of retro formats and structures — whole resource libraries of them, and I could start naming a few and think of dozens more. Any of them can work. Judge a format by the test in Section 7: did the underlying issue actually change. Whatever the format, land the same close: actions, owners, into the backlog.

9. AI-era

Having AI listen in during the sprint and surface candidates for the retro would genuinely help — more signal reaching the room. But the people still vote and decide what actually matters. I don't want AI running the "why is this a problem" discussion, and I wouldn't want to replace a human discussion retro with a fully AI one. AI is an addition to the retro, and the discussion stays the team's. Its job is surfacing the elephants we're not discussing, so the team's own conversation can decide what to do about them.

10. This is a fit question, not a recipe

The shape above is a strong default. Where a specific team actually sits against it, and what that's costing, is what the Engineering Ways of Working Diagnostic is built to find.