Adaptive Execution System

Facilitator guide · 🟢 SOLO 🔵 TEAM

🟢 SOLO · 🔵 TEAM — how to read this guide

Sections marked 🟢 SOLO or 🔵 TEAM apply to that mode only; BOTH sections apply to everyone. If you're running this on your own, you can skip anything marked 🔵 **TEAM only**.

Your job: get the assumptions hidden inside a real plan named out loud, introduce a likely shock, and rebuild the parts that fail so the next change is absorbed instead of breaking the plan. BOTH

The one idea: every plan depends on assumptions. Good plans know what they are. Plans don't fail because you didn't plan hard enough — they fail at the hidden assumptions nobody named. This activity finds those assumptions before the change does.

· WHEN DO I USE THIS?

Use this when BOTH

GOOD FIT

A real plan you're working to now — a sprint, delivery plan or roadmap — in conditions that genuinely change. Concrete enough to name what it depends on staying true.

WRONG TOOL

Feel how small changes pile up › Requirement Drift Simulator. Decide *when* to change course › Pivot Point. A change already landed, needs a full redo › Replan the Plan. Act right now under pressure › Decide Under Pressure.

The one test before you start: can you point to a real plan and say what it assumes stays true? If yes, run it. If the plan is too vague to have assumptions, sharpen the plan first. A plan in a perfectly stable world, or a change that already happened, isn't this tool.

· WHAT WILL YOU LEAVE WITH?

What you'll leave with BOTH

WHAT YOU GET

The assumptions the plan depends on, named and visible. The fragile ones redesigned — each with an early warning signal and a pre-agreed response. And a plan that keeps working even when conditions change.

WHY IT MATTERS

The "unexpected" change turns out to have been sitting inside an assumption nobody named. Change stops feeling like failure and starts feeling planned-for — and there's far less scrambling when conditions shift.

© Witmer

PART 1 BEFORE THE ACTIVITY

get ready

1 · WHAT IS MY JOB HERE?

Your job BOTH

You are running this to one outcome: a plan whose fragile assumptions each carry an early warning signal and a pre-agreed response. Your job is to:

- keep everyone on **assumptions, not tasks** — "build the API" is a task; "the API ships before we integrate" is the assumption
- push for the assumptions **nobody says out loud** — that's where the fragility hides
- read the failure out loud and **name it** — the plan failed exactly where an assumption changed
- refuse "be ready" redesigns — force a **signal plus a response**, agreed in advance

THE ONE RULE

A redesign is a signal plus a response — never "be ready". It only counts if it changes what you'll actually do when the assumption shifts: an early warning signal that it's starting, and a response agreed before the crisis, not during it.

2 · WHO DOES WHAT?

Roles BOTH

🟢 SOLO — for yourself

You run every step yourself from this guide: name your plan's assumptions, draw a shock, test, redesign the fragile ones. The honest question — "what does this need to stay true?" — replaces the group challenge. 30–40 minutes.

🔵 TEAM — you facilitate

You (facilitator) run the activity and keep the team on assumptions, hold the redesign standard, capture, keep time, stay neutral. Participants (3–6) name assumptions, judge which fail, and redesign. 60–75 minutes.

🔵 TEAM ONLY — skip if you're running solo

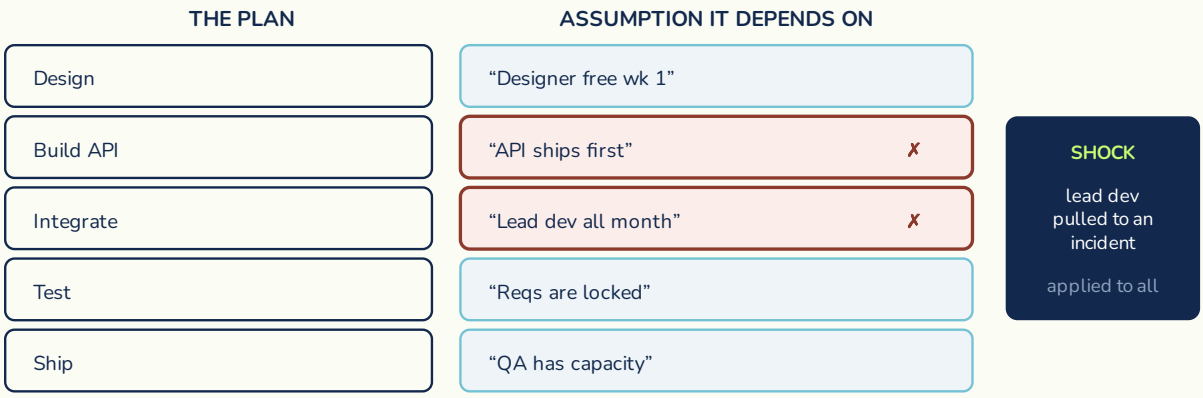
Brief the team in 2–3 minutes, in your own words. Say: "We're going to take our real plan and find the assumptions hiding inside it — the things it quietly takes for granted. Then we'll hit it with a likely change and watch where it fails." Don't read the app page aloud.

© Witmer

3 · HOW DO I PREPARE?

Setup and materials BOTH

THE BOARD AT A GLANCE — ONE SHOCK, APPLIED TO EVERY ASSUMPTION



Each part of the plan carries the assumption it depends on. One shock hits all of them.

The plan fails exactly at the assumptions the shock invalidates (marked ✗) — never randomly.

What	What it is, and when you use it
The real plan	The plan you're working to — a sprint board, delivery plan or roadmap. Step 1: lay it out where it can be seen. The critical ingredient; you bring it.
Assumption cards	Blank cards or sticky notes, one assumption per card. Step 2: write what each part of the plan depends on staying true, placed beside that part.
The Shock List	A set of likely changes — a person pulled away, a deadline moved up, a priority flipped, a dependency slipping. Provided as a PDF. Step 3: draw one and apply it to every assumption.
Capture space	A board or wall to lay out assumptions, mark which fail, and write the redesigned version with its signals. Steps 5–7: this is what leaves the room.

A note on the term: a **shock** is a plausible workplace change — drawn from the Shock List and treated as if it has just happened. Keep it real for your world, not a freak event.

PART 2 DURING THE ACTIVITY

run it

Before you begin — 30-second check

- ✓ There's a real, current plan — concrete enough to name what it assumes stays true
- ✓ Assumption cards, the Shock List, and capture space are laid out
- ✓ 🔵 TEAM the people who'll work to the plan are in the room
- ✓ You're ready to hold the redesign standard — signal plus response, never "be ready"

© Witmer

4 · HOW DO I RUN IT?

How to run it BOTH

The same five moves, whether you run it alone or facilitate a team. What changes is who does the challenging.

SAME FIVE MOVES — RUN ALONE OR WITH A TEAM

SOLO — FOR YOURSELF

you name assumptions on your own plan, draw a shock, and redesign the fragile ones

30–40 min · no facilitation

the honest question replaces the group challenge

TEAM — YOU FACILITATE

the team names assumptions; you apply the shock, hold the redesign standard, keep time

60–75 min · 3–6 people

write privately first, then pool — so quiet doubts surface

🟢 SOLO — for yourself

1. Set up (2 min). Lay out your plan where you can see all of it.

2. Name assumptions (12–15). One per card, beside the part it belongs to. Ask honestly: “what does this need to stay true?”

3. Draw a shock (5). Take one from the Shock List; treat it as real.

4. See which fail (8–10). Mark each card that no longer holds.

5. Redesign (15–20). Signal + response for each fragile one.

6–7. Re-test & commit (8–12). Same shock again; lock what holds.

🔵 TEAM — you facilitate

1. Set up (2 min). Lay out the plan; appoint yourself facilitator.

2. Name assumptions (12–15). Team writes privately first, then pools — so quiet doubts surface. Push for the unspoken ones.

3. Introduce a shock (5). You draw one and read it as if it just happened.

4. See which fail (8–10). Team checks each card; you name the failure out loud.

5. Redesign (15–20). Hold the standard: signal + response, not “be ready”.

6–7. Re-test & commit (8–12). Same shock; lock the version, name who watches each signal.

These times are indicative and will depend on the nature of the problem and the size of the team. The step that overruns is naming assumptions (step 2); the step that gets squeezed is redesign (step 5). Protect step 5 — two assumptions redesigned with a real response beat a full list named and nothing rebuilt.

Naming discipline — three moves that keep step 2 clean: (1) Know when to stop. Stop when every major part of the plan has at least one assumption attached — you don’t need every possible assumption. **(2) Combine duplicates.** If two cards express the same assumption in different words (“requirements stay stable” / “no major requirement changes”), merge them before the shock. **(3) Read them aloud.** Read every assumption out before applying the shock — not just stick cards up — so the whole room shares the same picture.

© Witmer

5 · HOW DOES THE REDESIGN WORK?

Keep the redesign concrete BOTH

A redesign only counts if it changes what you'll actually do when the assumption shifts. Two parts, every time: an early warning signal, and a response agreed in advance. Ask: "what would tell us this assumption is starting to fail?" — that question produces the signal.

A REDESIGN = EARLY WARNING SIGNAL + PRE-AGREED RESPONSE

VAGUE — REJECT ✗

"We'll keep an eye on the lead dev's load."

no signal · no agreed response

changes nothing when it actually happens

CONCRETE — ACCEPT ✓

SIGNAL

lead dev on any incident longer than half a day

RESPONSE

integration moves to the named backup — agreed now

"Be ready" is not a redesign. A signal tells you it's starting; a response says what you do.

Vague (reject)	Concrete redesign (accept)
"We'll keep an eye on the lead dev's load."	**Signal:** lead dev assigned to any incident longer than half a day. **Response:** the integration task moves to the named backup, agreed now.
"Be flexible if requirements move."	**Signal:** any change request after sprint day 3. **Response:** it goes to next sprint by default, not this one.
"Watch the dependency."	**Signal:** upstream API not in staging by Wednesday. **Response:** switch to the stubbed version; don't wait.

These show the shape — the team writes its own, grounded in its real plan (people, requirements, dependencies, timing).

When to step in — and when not to · a principle, not a step 🔵 TEAM

STEP IN

The team names tasks, not assumptions → ask "what must stay true?"

A redesign is just "we'll be ready" → push for signal + response.

The shock is a freak event nobody believes → swap for a likely one.

They jump to replanning the whole thing → pull back to fragile assumptions.

STAY OUT

They argue whether an assumption holds → that argument is the work.

A redesign exposes a weak plan → that discomfort is the tool working.

They disagree which shock is likeliest → let them pick.

The plan looks fragile → don't fix it; let the failure land.

© Witmer

The clearest tell it's time to step in: the team is listing tasks instead of assumptions. Stop and ask of one item: "what does this part of the plan need to stay true to work?" That answer is the assumption — write that, not the task.

PART 3 AFTER THE ACTIVITY

make it stick

6 · DID WE GET A GOOD RESULT?

What good looks like

BOTH

A GOOD RUN

Every part of the plan carries a named assumption; the fragile ones have an early warning signal and a pre-agreed response. Re-tested, the plan adapts instead of failing.

A HOLLOW RUN

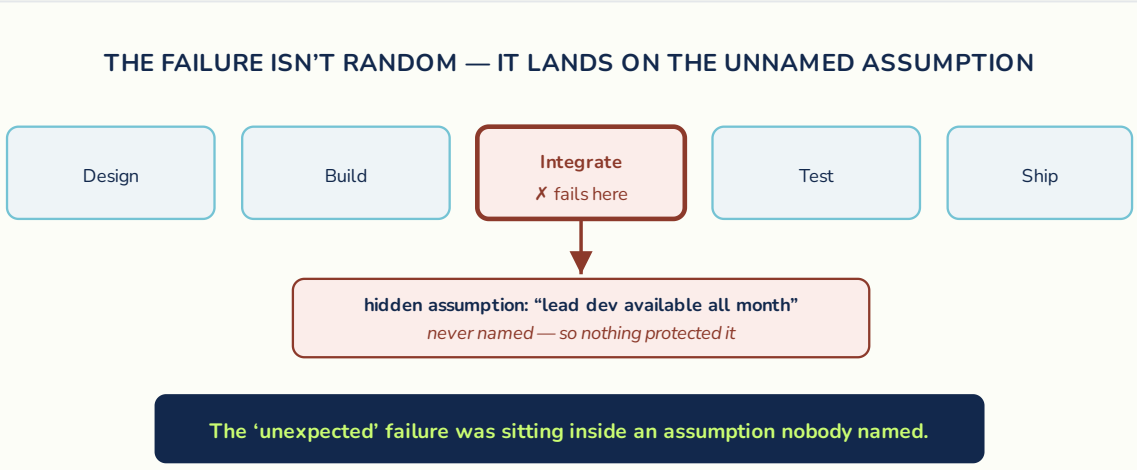
The team lists tasks, not assumptions; redesigns are "be flexible" with no signal; nothing is re-tested. The same shock fails it again. If your board looks like this, go back and ask "what does this assume stays true?"

7 · WHY DOES THIS WORK?

The idea underneath

BOTH

Every plan depends on assumptions, whether they're written down or not. When one changes unexpectedly, the plan appears to fail — but the failure isn't random: it lands precisely on the assumption the change invalidated. Naming the assumptions and testing them in advance turns hidden weaknesses into something you can see and design around, before real life exposes the weakest point.



© Witmer

Before you finish — the check

- ✓ Every part of the plan carries a named assumption — not a task
- ✓ The fragile assumptions are marked, where the shock made them fail
- ✓ Each fragile one has an early warning signal **and** a pre-agreed response
- ✓ The redesigned plan was re-tested against the same shock — and adapts
- ✓ 🔵 TEAM each signal has a named owner who watches it
- ✓ The board/plan has been captured before anyone leaves

8 · HOW DO I MAKE THIS A HABIT?

Take it back to work BOTH

THE HABIT

"Before committing any plan, ask — which part of this assumes nothing will change?"

The lasting change: you stop treating change as bad luck and start naming assumptions as you plan — protecting the fragile ones before the change arrives, not after. Every plan depends on assumptions; good plans know what they are.

Appendix · Reference

You should not need this while you run the activity. Read it before, or use it if a run goes sideways.

9 · A WORKED RUN

Watch the reveal land BOTH

A five-person squad planning a two-week sprint to ship an integration. The plan looks tight. Everyone's confident.

Naming Beside each part, they write what it depends on: "designer free week one", "upstream API ships before we integrate", "lead dev available all month", "requirements locked", "QA has capacity". Several were never said out loud.

The shock The facilitator draws: "the lead dev is pulled onto a production incident for a week." Likely — it happened last quarter. They check each card against it.

The reveal: the plan fails in exactly one place — the integration step, which silently assumed the lead dev was available all month. Not design, not QA. As someone says: "the 'unexpected' thing was sitting inside an assumption we never named. Our fragility wasn't bad luck — it was built in." The redesign: integration moves to a named backup on the warning signal; re-tested, the same shock barely dents the plan.

The learning moment to name out loud: ask the room — "if we'd never tested this assumption, when would we have discovered it?" The answer (mid-sprint, in a crisis) is exactly what the activity just bought them out of.

© Witmer

10 · WHAT GOES WRONG MOST OFTEN?

Common mistakes BOTH

Mistake	Why it fails the run
Listing tasks, not assumptions	The commonest failure. “Build the API” is a task; “the API ships before we integrate” is the assumption. Only assumptions can fail.
Redesigns that say “be ready”	That changes nothing. Force an early warning signal plus a response agreed in advance.
Unlikely shocks	A freak event nobody believes teaches nothing. Use changes realistic for this team’s world.
Replanning the whole thing	The point is to protect the fragile assumptions, not redo the plan. A full redo after a change has landed is Replan the Plan.

11 · WHAT IF SOMETHING GOES WRONG?

Edge cases BOTH

What happens	What to do
The plan survives every shock	Either genuinely strong (rare) or the shocks are too soft. Draw a harder, still-likely shock before deciding it can cope.
Everything fails	The plan is highly coupled to one assumption. Good — that’s the single most valuable thing to redesign first.
The team identifies dozens of assumptions	Very common. Group similar ones together, then test the major assumptions first — otherwise the session overruns before you reach the redesign.
The team lists tasks, not assumptions	Ask of each task: “what does this need to stay true?” Write that answer instead.
🔵 One person dominates	Have everyone write cards privately first, then pool — so quiet doubts surface.
They argue whether an assumption holds	Let it run briefly — the argument surfaces the real risk — then mark it “contested” and redesign for both cases.
Time runs out	Redesign only the single most fragile assumption, fully. One real redesign beats five named and none protected.
A change has already happened	Wrong tool — this builds resilience before the change. Use Replan the Plan to redo the plan now.

© Witmer

12 · WHERE DOES THIS FIT?

Where this sits in the Gym BOTH

If instead...	Use this
the team needs to feel how small changes accumulate into chaos	**Requirement Drift Simulator** — makes the slow build-up of change visible.
the question is *when* a plan should change at all	**Pivot Point** — finds the signal that it's time to change course.
a change has already landed and the plan needs a full redo	**Replan the Plan** — redesigns execution after reality has shifted.
the team must act right now under uncertainty	**Decide Under Pressure** — the fast call when the change is already here.

Rule of thumb: this tool builds resilience *before* the change — it names and protects fragile assumptions while the plan still holds. Once a change has already landed, it's Replan the Plan; deciding whether to change course is Pivot Point.

© Witmer