

Practical Project Planning: Claude Skills Pack

Oct 9, 2026 · @John

This is the complete source of the Practical Project Planning plugin for Claude: 11 skills built from *A Field Guide to Practical Project Planning* by John Newcomb, "The Biker Project Manager." Install from the ZIP downloads; this document is the readable, printable copy of exactly what those files contain.

What's in the pack

The pack ships as two downloads. Use the plugin ZIP on a paid Claude plan; use the individual skill ZIPs on the Free plan or to install only some skills. The companion guide, *Installing and Using the Practical Project Planning Skills*, walks through both.

Download	What it is	Use it when
practical-project-planning-plugin.zip	All 11 skills as one Claude plugin (version 1.0.0)	You have Claude Pro, Max, Team, or Enterprise
practical-project-planning-individual-skills.zip	A folder of 11 separate skill ZIPs, one per skill	You have the Free plan, or want only some skills

#	Skill	Use it for
1	aaa-method	Any problem, blocker, or decision: Assess, Assemble, Action
2	risk-triage	Scoring risks, accept / mitigate / eliminate, mitigation ROI, reserves
3	stakeholder-communication	Status reports, sponsor emails, steering committee decision asks, escalations
4	four-stage-planning	Running the whole planning model, gate reviews, feedback loops, Agile mapping, scaling
5	stage1-themes-objectives	Themes, measurable objectives, sponsor alignment, flashpoints

#	Skill	Use it for
6	stage2-tasking-schedule	Tasks, dependencies, estimates, internal schedule, Stage 2 workshop
7	stage3-provider-alignment	Making the plan supportable with vendors, SOW, acceptance criteria, billing triggers
8	stage4-spend-schedule	Time-phased spend, budget baseline, contingency, invoice timing
9	wbs	Building, reviewing, and controlling a Work Breakdown Structure
10	red-zone-planning	Go-live readiness, runbooks, rollback, smoke tests, command structure
11	planning-health-check	Diagnosing a troubled project and building a recovery plan

Plugin folder layout:

```

practical-project-planning/
  .claude-plugin/plugin.json
  README.md
  skills/
    aaa-method/SKILL.md
    risk-triage/SKILL.md
    stakeholder-communication/SKILL.md
    four-stage-planning/SKILL.md
    stage1-themes-objectives/SKILL.md
    stage2-tasking-schedule/SKILL.md
    stage3-provider-alignment/SKILL.md
    stage4-spend-schedule/SKILL.md
    wbs/SKILL.md
    red-zone-planning/SKILL.md
    planning-health-check/SKILL.md

```

Plugin manifest and shared response rules

The manifest identifies the plugin to Claude. File: `.claude-plugin/plugin.json`

```
{
  "name": "practical-project-planning",
  "version": "1.0.0",
  "description": "Skills built from John Newcomb's book A Field Guide to Practical Project Planning: the AAA Method, risk triage, stakeholder communication, the Four-Stage Planning Model (and each stage), Work Breakdown Structures, Red Zone Planning, and a planning health check. Feed in a situation and get a to-do list or a step-by-step plan to resolve it.",
  "author": { "name": "John Newcomb", "email": "john@thebikerprojectmanager.com" },
  "homepage": "https://thebikerprojectmanager.com",
  "keywords": ["project management", "planning", "AAA method", "WBS", "PMO", "red zone", "go-live"]
}
```

Every skill file ends with the same response rules. They are printed once here instead of repeating them in each skill below.

Response rules (all Field Guide skills)

- Work from what the user gave. If key facts are missing, state assumptions inline marked ****[assumed]**** and keep going. End with at most 3 questions, and only ones whose answers would change the plan.
- Two modes. "What should I do" / "for me" means a verb-first, ordered to-do list for the user. "Solve it" / "steps to solve" means a full step plan, and Claude actually does every step it can do with the information on hand (drafts the artifact, table, message, or checklist) instead of only listing it.
- Tag each step's owner: ****[You]****, a role such as ****[Tech Lead]****, or ****[Claude]**** for anything Claude can produce right now.
- Every step gets one owner, a concrete output, and a done-when. Shared ownership is no ownership.
- Scale rigor to the situation. A five-minute check stays short; a program-level plan gets full tables. Strategy stays fixed, tactics flex to the user's method (Waterfall, Agile, hybrid, Kanban).
- Plain, direct, practitioner voice. No fluff, no academic jargon. Never use em dashes; use commas, colons, or periods.
- When the problem really belongs to another Field Guide skill, say which one and apply it.

Skill 1: aaa-method

File: skills/aaa-method/SKILL.md (followed by the shared response rules)

```
---
```

```
name: aaa-method
```

```
description: Apply the AAA Method (Assess, Assemble, Action) from John Newcomb's A Field Guide to Practical Project Planning to any project problem, blocker, decision, meeting, or stalled effort. Use when the user says "using the AAA skill", "run AAA on this", "assess assemble action", or asks what steps will resolve a project situation.
```

```
---
```

```
# AAA Method: Assess, Assemble, Action
```

AAA is the decision engine of the Field Guide. Run three steps, in order, every time: assess what is true, assemble what you need, then act with clarity. It scales from a five-minute standup check to a five-week planning cycle. The rigor scales; the discipline does not.

AAA exists to stop three repeat failures. Name whichever one the situation shows:

- **Action without Assess** = reactive chaos (treating symptoms, not causes).
- **Assess without Assemble** = analysis paralysis.
- **Assemble without Action** = preparation without purpose.

```
## Step 1: Assess (clarity before planning)
```

Answer three questions:

1. **What do we know?** Facts, data, prior experience, documented requirements, confirmed constraints.
2. **What don't we know?** Information gaps, unvalidated assumptions, unconfirmed dependencies, areas of uncertainty.
3. **What does the landscape look like?** Organizational context, stakeholder dynamics, market, technical, regulatory.

Then ask **the Magic Question: "Can we start right now?"** If no, list every reason. Sort each reason into:

- **Assemble**: something to gather (people, data, access, tools, provider input).
- **Clarify**: a decision or answer someone owes.
- **Remove**: a blocker to eliminate.

Also in Assess:

- **Assumptions are risks in disguise.** List what the plan depends on and challenge each one.
- **Risk scan.** Triage each risk: low likelihood and low impact, accept; middle, mitigate with a plan; high and high, change the approach to eliminate it. (Use the risk-triage skill for depth.)
- **Flashpoints.** A decision where one choice rewrites everything downstream (build vs buy is the classic). Get it decided before detailed planning. If it cannot be decided yet, either plan both forks or stop planning at the flashpoint. Never plow ahead on a guess.
- Recommended techniques: stakeholder interviews (show up with questions, capture outputs), environmental scan (what changed, which assumptions need revalidating), lessons learned review, assumption and constraint analysis.

Step 2: Assemble (preparation with purpose)

Key question: **"Do we have what we need to act with confidence?"** Check every category:

- **Teams:** right skills, experience, availability windows. Confirmed commitments, not a list of names.
- **Data and information:** documents, specs, requirements, historical performance data.
- **Tools and infrastructure:** systems, environments, access. Nothing kills a kickoff faster than missing access.
- **Provider inputs:** scope, roles, assumptions, points of contact, delivery constraints.
- **Stakeholder alignment:** the people who need to agree actually agree, before execution.

Tools: resource assembly matrix (who and what, when needed, status of securing it), intentional team composition, RACI, focused information-gathering working sessions.

Warning: **never confuse urgency with readiness.**

Step 3: Action (move with clarity)

Key question: **"Are we turning preparation into real progress?"**

- Decisions are finalized, documented, and communicated. Debate ends.
- Every action has **one owner, a clear output, and a due date.**
- Timelines are committed on real constraints and real capacity, then baselined.
- Planning artifacts are produced and published.

- Escalation paths are defined: trigger, target, expected response time.

Tools: action register (the accountability backbone), task board, escalation protocol, decision log (stops re-litigating settled questions).

AAA is a cycle

After Action, re-Assess: did it work, what changed, what is next. Always set the re-assess checkpoint.

Procedure when handed a situation

1. Restate the problem in one line and name the decision(s) that must be made.
2. Place it: lifecycle swimlane (Initiating, Planning, Execution, Delivery, Closing) and scale (quick check, meeting, phase, program).
3. Run Assess. Produce Know / Don't Know / Landscape, the Magic Question answer and its reasons, triaged risks, and any flashpoint.
4. Run Assemble. Turn each reason from the Magic Question into an assemble item with owner and needed-by date.
5. Run Action. Produce action register rows, the decision(s) to log, and the escalation path.
6. Set the re-assess checkpoint.
7. Route to a companion skill if the root problem lives there: risk-triage, stakeholder-communication, four-stage-planning, stage1-themes-objectives, stage2-tasking-schedule, stage3-provider-alignment, stage4-spend-schedule, wbs, red-zone-planning, planning-health-check.

Output format

****Problem (one line):**** ...

****Where we are:**** swimlane, scale, AAA failure mode observed (if any)

****ASSESS****

- Know: ...
- Don't know: ...
- Landscape: ...
- Can we start now? No, because: 1) ... 2) ... (each tagged Assemble / Clarify / Remove)
- Risks: risk, triage call (Accept / Mitigate / Eliminate)
- Flashpoint: name it and the decision path, or "none found"

****ASSEMBLE****

```
| Item | Type (People / Info / Tools / Provider / Alignment) | Owner | Needed  
by | Status |
```

****ACTION****

```
| # | Action | Owner | Output | Due | Done when |
```

- Decisions to log: ...
- Escalation path: trigger, to whom, response time

****Re-assess checkpoint:**** when and what to check

****What I can do now:**** artifacts Claude can draft immediately (then draft them if the user asked to solve it)

Skill 2: risk-triage

File: `skills/risk-triage/SKILL.md` (followed by the shared response rules)

```
---
```

```
name: risk-triage
```

```
description: Triage and respond to project risks using the qualitative and  
quantitative approach from John Newcomb's A Field Guide to Practical Project  
Planning. Use when the user asks to assess, score, rank, or respond to risks,  
build a risk register, decide if a mitigation spend is worth it, size  
contingency or reserves, or says "using the risk skill".
```

```
---
```

```
# Risk Triage
```

Risk management is not eliminating uncertainty; that is impossible. It is understanding it well enough to make informed response decisions. Risk lives in the Assess step of AAA, and it directly shapes what must be assembled. Risk is not always bad news: some risks are upside. Don't "ride squid" (no gear) on a high-stakes initiative.

```
## 1. Find the risks
```

- Pull from: the situation described, the stated plan, constraints, dependencies, providers, timing windows.
- ****Convert assumptions into risks.**** Every "we assume X" becomes "if X turns out false, then ...".
- Write each risk as cause, event, effect: "Because [cause], [event] may happen, which would [effect]."
- Flag upside risks (opportunities) separately.

2. Qualitative triage (two questions)

How likely is it? How bad (or good) is it if it happens? Rate each Low / Medium / High.

Likelihood / Impact	Response	What it means
Low and Low	Accept	Log it, watch it, do nothing else.
Anything in the middle	Mitigate	Write the plan for what you will do if it shows up: trigger, response, owner.
High and High	Eliminate ("nope")	Do not mitigate. Change the approach so the risk no longer exists.

Low likelihood with High impact: mitigate, and consider a reserve (step 3).
For upside risks: plan how to capture the benefit and who owns it.

3. Quantitative options (put real numbers to the pain)

A. Spend a little to shrink a big exposure. Compare mitigation cost to the exposure it removes.

Example from the book: a risk costs \$3M if it hits; spending \$20K now caps the exposure at \$500K. That buys \$2.5M of protection for \$20K: a good deal. Show the math: exposure before, exposure after, mitigation cost, reduction per dollar. State whether you are using maximum exposure or probability-weighted (expected) exposure.

B. Hold reserve. Set aside specific cash, capacity, materials, or people (including a staffing plan if you fall behind) so a known risk does not become a full stop. Contingency is not padding; it is a disciplined response to quantified uncertainty. Every reserve names the risks it covers, the release condition, and who approves its use. (Feeds the Stage 4 contingency reserve plan.)

4. Timing matters

The closer you get to go-live, the less you can afford surprises. Re-triage near launch: items you accepted early may need mitigation now. For releases within about six weeks, also apply the red-zone-planning skill.

Output format

Risk register

ID	Risk (cause, event, effect)	Up/Down	Likelihood	Impact	Response (Accept / Mitigate / Eliminate / Reserve)	Response action	Owner	Trigger to watch
----	-----------------------------	---------	------------	--------	--	-----------------	-------	------------------

****Quantitative calls**** (for any risk worth a number): exposure before, mitigation cost, exposure after, verdict.

****Reserve plan**** (if any): reserve, amount or capacity, risks covered, release condition, approver.

****Approach changes required**** (all High/High items): what changes in the plan.

****Action list:**** owner-tagged steps with due dates.

Skill 3: stakeholder-communication

File: `skills/stakeholder-communication/SKILL.md` (followed by the shared response rules)

```
---
name: stakeholder-communication
description: Use AAA (Assess, Assemble, Action) to plan and draft stakeholder
communication, following John Newcomb's A Field Guide to Practical Project
Planning. Use for status reports, executive updates, sponsor emails,
escalations, steering committee prep and decision asks, communication plans,
or when the user says "using the stakeholder communication skill" or "how do
I tell the sponsor".
---
```

Stakeholder Communication with AAA

Communication failures are leadership failures. Most escalations could have been prevented, or softened, by better communication planning. Communication does not take care of itself. Run AAA on it.

Assess: understand the stakeholder before you try to influence them

- ****Purpose:**** what must this communication accomplish? Inform, get a decision, escalate, ask for help, reset expectations.
- ****Audience:**** who exactly, and what do they care about most and least? Examples: an ROI/budget stakeholder wants spend to date, forecast, and when the return arrives. A resource/capacity stakeholder wants schedule status, the right owners on the work, and over-allocation.
- ****Format and channel:**** how do they like to receive information (one-pager, email, deck, verbal, dashboard)?
- ****What "good" looks like to them.****

If several stakeholders, profile each one separately; one message rarely fits all.

Assemble: turn data into directed intelligence

This is where most status reports go to die. "Task 36a was completed" is a fact executives do not care about. They want:

1. Are we on track?
2. What is at risk?
3. Do you need anything from me?

Lead with what matters to this audience. Put supporting detail behind it. Cut the noise. Signal, not data dump.

Action: deliver in a form they will consume

- ****Status report:**** send it in their preferred format, bottom line first.
- ****Decision meeting / steering committee:**** open with why they are there and the decision you need. Make the ask explicit. Provide only what they need to say yes, say no, or choose A vs B. Record the decision and assign follow-ups.
- Close every communication with a clear ask, a clear message, and a clear next step.

Steering committee prep (book pattern)

- Assess: current status, which risks are real now, decisions needed, data supporting each recommendation.
- Assemble: status, risks, supporting data, the deck; pre-brief key participants; confirm decision-makers will attend.
- Action: run the meeting, secure decisions, document outcomes, assign follow-ups.

Templates Claude drafts from

****Executive status (one screen)****

- Bottom line: Green / Amber / Red, one sentence on why.
- What is at risk: top 1 to 3 items, impact, what we are doing.
- What I need from you: specific ask and by when (or "nothing this week").
- Changes since last update.
- Details (behind the fold): milestones, spend vs plan, upcoming.

****Decision ask****

- Decision needed: one sentence, needed by date, why now.
- Options: A and B (and C if real), each with impact on scope, schedule, cost, risk.
- Recommendation and rationale.
- The ask: "Approve option A today."

****Escalation****

- Issue, impact if unresolved, what has been tried, the specific help needed, deadline.

```
## Output format
1. **Audience profile** per stakeholder: cares about, ignores, format, purpose of this message.
2. **Message plan:** what leads, what goes behind, what gets cut.
3. **Draft(s)** ready to send (Claude writes them when data is provided).
4. **Delivery and follow-up:** channel, timing, who sends, follow-up actions with owners.
```

Skill 4: four-stage-planning

File: `skills/four-stage-planning/SKILL.md` (followed by the shared response rules)

```
---
name: four-stage-planning
description: Run or guide the full Four-Stage Planning Model from John Newcomb's A Field Guide to Practical Project Planning (Stage 1 Themes and Objectives, Stage 2 Tasking and Internal Schedule, Stage 3 Professional Services Alignment Review, Stage 4 Spend Schedule). Use for "plan this project", "build a project plan", "where are we in planning", stage gate or transition reviews, adapting planning to Agile or hybrid, scaling planning to project size, or when a change in one stage ripples into others.
---
```

Four-Stage Planning Model

The centerpiece of the Field Guide. Win on paper before you try to win in execution. The model lives inside the **Planning** swimlane of the five-stage lifecycle (Initiating, Planning, Execution, Delivery, Closing). Swimlanes overlap; they are not rigid gates.

The four stages and the connected flow

Each stage's outputs become the next stage's inputs. Nothing stands alone.

Stage	Purpose	Key outputs	Becomes input to
1. Themes and Objectives	Set intent: themes and measurable objectives, with sponsor alignment	Approved Themes and Objectives artifact	Stage 2: objectives are decomposed into tasks
2. Tasking and Internal Schedule	Turn objectives into tasks and an internal schedule the team believes	WBS, task register, dependency map, supportable internal schedule	Stage 3: tasks needing external support are

reviewed with providers |

| 3. Professional Services Alignment Review | Make the plan supportable with provider reality (not procurement) | Updated plan, agreed services scope/SOW, invoiceable work and commercial inputs | Stage 4: invoiceable costs feed the spend schedule |

| 4. Spend Schedule | Time-phased financial plan tied to the work | Spend schedule, budget baseline, contingency reserve plan, financial dashboard | Execution |

Primary AAA emphasis: Stage 1 heavy Assess and Assemble; Stage 2 Assemble and Action, then Assess checks quality; Stage 3 full cycle; Stage 4 Action and Assess.

For depth on any stage, apply its skill: stage1-themes-objectives, stage2-tasking-schedule, stage3-provider-alignment, stage4-spend-schedule. For decomposition, apply wbs.

Traceability (the thread that keeps the plan honest)

You must be able to walk the chain both ways: spend line (Stage 4) to services scope (Stage 3) to the task needing external support (Stage 2) to the objective that justified it (Stage 1). If a link breaks, something is disconnected, and disconnected planning gets expensive.

Procedure

1. **Locate the project.** Which stages are done, which artifacts exist, which are missing or weak? Is this pre-planning (no charter yet means Initiating work first)?

2. **Size it** and set formality:

- Small (1 to 5 people, 1 to 3 months): Stage 1 can be one sponsor conversation; flat 2 to 3 level WBS; skip Stage 3 if no external partners; Stage 4 is a simple spreadsheet by week; AAA informal but consistent.

- Medium (5 to 20 people, 3 to 12 months): all four stages, moderate formality; 3 to 4 level WBS; 1 to 2 partners; monthly spend by category; AAA at formal decision points; use Red Zone Planning to lock the finish.

- Large (20+ people, 12+ months): each workstream may run its own four-stage cycle; 5+ WBS levels with a dictionary; many partners; weekly spend with earned value and rolling forecasts; formal decision logs; Red Zone at scale.

3. **Check for flashpoints** before Stage 2 detail. Decide first, or plan both forks, or stop at the flashpoint.

4. **Run the next stage** using its skill.

5. **Run the stage transition review** (substantive quality gate, not a ceremony):

- 1 to 2: Themes strategically sound? Objectives measurable? Stakeholders agree? Ready to decompose?
 - 2 to 3: WBS complete? Dependencies identified? Estimates credible? Which tasks need external support?
 - 3 to 4: Service scopes specific and measurable? Outputs and acceptance criteria clear? Billing triggers defined enough to build a spend schedule?
 - 4 to Execution: Spend schedule aligned with project schedule? Contingency adequate for the risk profile? Budget formally approved?
6. ****Handle feedback loops.**** New information should change the plan; the only failure is pretending nothing changed.
- 3 to 2: provider reality forces task restructure or resequencing.
 - 3 to 1: provider options or constraints prompt new or revised objectives.
 - 4 to 3: a different commercial model (milestone vs time-and-materials) manages risk better.
 - 4 to 1: cost exceeds budget; reprioritize objectives or defer lower-priority themes.
- Treat every revision as a decision: what is changing, why, and the impact on scope, schedule, and cost. Log it.
7. ****Keep it alive.**** Execution retros update Stage 2 estimates and Stage 4 forecasts; provider performance reviews update Stage 3; stakeholder reviews may reopen Stage 1. A plan never updated is a historical document.

Workshop mechanics

- ****Chaos time:**** groups need about 30 minutes to settle, trade context, and align before producing clean output. Rule of thumb: need one hour of real work, schedule ninety minutes. Longer if the team is new to the project or the model.
- Send the ****Stage 2 and Stage 3 planning guides at least five days**** before their sessions.
- Stage 4 needs no separate prep; run it in the same working session as Stage 3.

Agile and hybrid mapping (same stages, different artifacts)

Stage	Waterfall	Agile	Hybrid
---	---	---	---
1	Charter, scope statement, requirements	Epics and themes, product vision, iteration objectives	Themes up front; objectives refined iteratively
2	WBS, detailed Gantt	Story breakdown, sprint planning, backlog refinement (rolling wave)	High-level WBS up front, detail per iteration
3	Provider sessions, SOW refinement, acceptance criteria	Ongoing provider alignment, capacity agreements	Framework agreement, iteration

```

scoping, checkpoints |
| 4 | Detailed baseline, full cost projection | Rolling-wave budgets,
incremental funding | Phase baseline, iteration refinement |
Agile is not the absence of planning. It is planning in slices.

```

```

## PMO note

```

```

Keep stage standards and reviews lightweight. One efficient process with real
value beats a hundred box-checking ones.

```

```

## Output format

```

```

**Planning roadmap**

```

```

| Stage | Status (Done / Partial / Not started) | Artifacts present | Gaps |
Next actions | Owner | Target date |

```

```

**Flashpoints:** ...

```

```

**Gate review** for the current transition: each question, Pass / Fail, fix.

```

```

**Feedback-loop changes** (if any): change, reason, impact on scope /
schedule / cost.

```

```

**Next step:** the single most important thing to do now, then the owner-
tagged action list.

```

Skill 5: stage1-themes-objectives

File: `skills/stage1-themes-objectives/SKILL.md` (followed by the shared response rules)

```

---

```

```

name: stage1-themes-objectives

```

```

description: Stage 1 of John Newcomb's Four-Stage Planning Model (Themes and
Objectives). Use to turn a charter, idea, or vague goal into project themes
and measurable objectives, fix vague or unmeasurable objectives, prepare a
Stage 1 visioning workshop, align sponsors and stakeholders on success
criteria, or spot and handle flashpoints (such as build vs buy) before
decomposition.

```

```

---

```

```

# Stage 1: Themes and Objectives

```

```

Stage 1 sets the intent. Define themes, turn each into measurable objectives,
and get alignment before anyone burns time on decomposition.

```

```

Key principle: "If you can't articulate the theme, you're not ready to
plan."

```

Definitions

- **Theme:** a headline describing what the project is trying to accomplish; value-driven grouping tied to organizational goals. Example: "Improve Customer Experience".
- **Objective:** the measurable success criteria for a theme. At project end you can clearly say achieved or not. Example: "Reduce average customer support response time from 48 hours to 4 hours by Q3." "Improve customer experience" is NOT an objective.

Preparation

Review the charter. Interview the sponsor and key stakeholders. Build a Stage 1 prep guide with draft themes and objectives so the session starts from something concrete.

Activities

1. **Theme drafting** with sponsor and key stakeholders. For each theme: short description, strategic rationale, link to organizational goals. Themes should reinforce each other; a theme with no link to the strategic rationale does not belong.
2. **Success criteria workshop.** For each theme, write objectives using: verb + metric + baseline + target + date. Add owner, and priority ranking if useful.
3. **Stakeholder and sponsor alignment.** Review, resolve disagreements, confirm priorities, and **document approvals**.

AAA in Stage 1

- **Assess:** Why does this project exist? Which organizational priorities does it serve? What are the constraints? What has been tried before?
- **Assemble:** Right stakeholders and right inputs in the room (documents, data, context). If the right people aren't there, you don't get a plan; you get wishes.
- **Action:** Document approved themes, commit to measurable objectives, communicate them to all stakeholders.

Objective quality test (apply to every objective)

- Measurable metric? Baseline known? Target set? Date set?
- One accountable owner?
- Traces to exactly one theme?
- Would the sponsor sign it as written?
- Could two people disagree on whether it was achieved? If yes, rewrite.

Flashpoints

A flashpoint is a fork where one choice rewrites the whole downstream plan (build vs buy, platform A vs B, in-house vs outsourced, phased vs big-bang).
If you see one:

1. Name it and who decides.
2. Get the decision before Stage 2 detail.
3. If it cannot be decided yet: plan both forks completely, or stop planning at the flashpoint and resume after the decision. Never plow ahead on a guess.

Pitfalls

Vague objectives; disconnected themes; skipping stakeholder input (it will be challenged during execution).

Gate to Stage 2

Themes strategically sound? Objectives measurable? Stakeholders agree (documented)? Team ready to decompose?

Agile note

Themes and objectives become Epics with clear business value statements, product vision, iteration objectives. Shorter horizon, identical discipline.

Output format

Themes and Objectives artifact

#	Theme	Description	Strategic rationale	Org goal supported	
ID	Objective	Metric: baseline to target	By date	Owner	Priority

Theme |

(Number objectives 1.1, 1.2, 2.1 so they become WBS branches.)

- Values Claude inferred are marked **[assumed]** for the sponsor to confirm.

Flashpoint log: flashpoint, options, decision owner, needed by, plan approach (decide / plan both / stop).

Alignment issues: disagreements and who resolves them.

Workshop plan (if requested): attendees, pre-read, agenda with 30 minutes of chaos time built in.

Gate check and next actions.

Skill 6: stage2-tasking-schedule

File: `skills/stage2-tasking-schedule/SKILL.md` (followed by the shared response rules)

name: stage2-tasking-schedule

description: Stage 2 of John Newcomb's Four-Stage Planning Model (Tasking and Internal Schedule). Use to break objectives into tasks, build a task register, map dependencies, estimate effort, sequence work into an internal supportable schedule, prepare or run a Stage 2 planning workshop and planning guide, or fix a schedule nobody believes.

Stage 2: Tasking and Internal Schedule

Stage 2 turns objectives into a plan the team can execute: tasks, sequence, effort, timeline, then an internal review until the team agrees it is supportable.

Key principle: "Every task must trace back to an objective. If it doesn't, it doesn't belong." This stops gold plating, scope creep, and work nobody asked for.

Preparation

Send the **Stage 2 planning guide** at least five days before the workshop. Build it from the Stage 1 Themes and Objectives artifact and include every known constraint and dependency (key dates, freezes, PTO, holidays, other projects competing for the same people).

Activities

1. **Task breakdown:** decompose each objective into the tasks that achieve it. Goal: complete, assignable, estimable. (Apply the wbs skill for structure, numbering, and quality checks.)
2. **Dependency mapping and sequencing:** "In what order must this happen?" Use dependencies, resource constraints, and business priorities.
3. **Effort estimation:** person-hours or person-days, estimated by the people who will do the work, not only those who manage it.
4. **Schedule build:** convert tasks, dependencies, and effort into a timeline.
5. **Internal supportability review:** walk the plan with the internal team; adjust for capacity, holidays, PTO, and constraints until the team agrees it can be supported.

Workshop setup (wall or virtual board)

- Zones: Themes, Objectives, a **backlog area** for tasks as they are created, and a zone for PTO, key dates, and dependencies.
- A timeline in intervals that fit the work: monthly buckets for many teams;

two-week or one-week for fast efforts.

- Generate tasks into the backlog, then move them onto the timeline to form the schedule.
- Budget ****chaos time**** (about 30 minutes) before real output. Need one hour of work, plan ninety minutes.
- Attendees: technical leads, SMEs, experienced practitioners. The PM structures the process; the experts provide the content.

AAA in Stage 2

- ****Assess:**** Stage 1 objectives and scope boundaries. What must be decomposed? What constraints and dependencies apply?
- ****Assemble:**** the decomposition team plus Stage 1 outputs as starting inputs.
- ****Action:**** produce the WBS, task register, dependency map, preliminary schedule.
- ****Assess again:**** do tasks trace to objectives? Do estimates hold up? Peer review.

Pitfalls

- Gold plating: tasks beyond the approved objectives are scope creep, however valuable.
- Missing dependencies: schedule fantasies that collapse in execution.
- Insufficient decomposition: if a task cannot be estimated with reasonable confidence, break it down further.
- Decomposing without the right people.

Gate to Stage 3

WBS complete? Dependencies identified? Estimates credible? ****Which tasks require external support?**** For each task ask "Can we deliver this with internal capability?" A "no" makes it Stage 3 services scope.

Agile note

The WBS becomes a backlog structured by Epic and Theme. Decompose in rolling waves: detail for the next iteration, higher level beyond.

Output format

****Task register****

WBS ID	Task	Traces to objective	Owner	Effort	Depends on	Internal / External	Notes
--------	------	---------------------	-------	--------	------------	---------------------	-------

****Dependency map:**** predecessor to successor list, critical chain called out.

****Internal schedule:**** tasks placed in time buckets, with PTO / freeze / key dates overlaid.

****Supportability concerns:**** over-allocation, unrealistic estimates, unowned

tasks, and the fix for each.

****Tasks flagged for Stage 3**** (external support needed).

****Workshop kit**** (if requested): planning guide outline, attendee list, board layout, agenda with chaos time.

Skill 7: stage3-provider-alignment

File: `skills/stage3-provider-alignment/SKILL.md` (followed by the shared response rules)

```
---
```

```
name: stage3-provider-alignment
```

```
description: Stage 3 of John Newcomb's Four-Stage Planning Model
(Professional Services Alignment Review). Use when reviewing an internal plan
with vendors, contractors, or professional services providers; refining a SOW
or services scope; defining provider outputs, acceptance criteria, handoffs,
and signoffs; confirming rate cards, invoicing triggers, and change-order
mechanics; or when a provider cannot support the plan, change orders are
piling up, or a vendor was brought in late.
```

```
---
```

Stage 3: Professional Services Alignment Review

Stage 3 takes the internally built Stage 2 plan and reviews it ****with the provider(s) who will help deliver it****, then modifies scope, sequencing, tasks, acceptance criteria, and assumptions until the plan is supportable in the real world.

****It is NOT procurement.**** No RFPs, provider selection, or market comparisons here. That is wasted motion in this stage.

****Key principle:** "Providers are partners in delivery, not afterthoughts in planning."****** Late alignment is one of the most expensive planning failures: mismatched expectations, change orders, delivery friction.

Preparation

Send the ****Stage 3 planning guide at least five days ahead****: the Stage 2 plan and timeline plus every constraint and dependency, so the provider can surface problems before the session.

Activities

1. ****Provider plan review and feasibility check.**** Walk the WBS,

dependencies, and schedule together. Hunt for where assumptions break:

- lead times, access, tooling, environments, data readiness, regulatory steps, provider capacity.

2. ****Scope and output refinement.**** For each externally supported task: what the provider produces, what the internal team produces, and what "done" means (acceptance criteria, quality bar, handoff, who signs off).

3. ****Delivery approach and sequencing alignment.**** Adjust sequence, dependencies, and milestones to how the provider actually works: workstreams, integration points, review cycles, change control, signoffs.

4. ****SOW / services scope finalization.**** Map the SOW cleanly to Stage 2 tasks: outputs, responsibilities, assumptions, governance. Ideally no surprises by now.

5. ****Commercial and risk alignment.**** Confirm rate cards, invoicing triggers (milestones, T&M cadence, acceptance points), change-order mechanics, and risk ownership so Stage 4 is built from real invoiceable work.

AAA in Stage 3 (full cycle)

- ****Assess:**** stress-test the Stage 2 plan. Risky assumptions? Dependencies on provider availability? Lead times, access, or review cycles that could break the schedule?

- ****Assemble:**** internal workstream leads plus provider(s); WBS, task register, dependency map, schedule, assumptions, constraints, existing contract or SOW terms.

- ****Action:**** update the plan so both sides can support it; confirm responsibilities; finalize the services scope so it can be costed and invoiced.

Pitfalls

Treating it like procurement; involving the provider too late; unclear handoffs and acceptance criteria (they create disputes); building Stage 4 on non-invoiceable assumptions.

Feedback loops

- Back to Stage 2 when tasks must be restructured, resequenced, or rescope.
- Back to Stage 1 when provider options or constraints suggest new or revised objectives.

Log each as a decision: what changed, why, impact on scope, schedule, cost.

Gate to Stage 4

Service scopes specific and measurable? Outputs and acceptance criteria clear? Billing triggers defined well enough to build a spend schedule?

Agile note

Alignment becomes ongoing: framework agreement, capacity agreements, iteration-level scoping and checkpoints.

Output format

****Provider alignment matrix****

Task ID	Task	Provider	Internal	Shared	Provider output	Internal output	Acceptance criteria	Signoff by	Billing trigger	Assumption or constraint
---------	------	----------	----------	--------	-----------------	-----------------	---------------------	------------	-----------------	--------------------------

****Plan changes**** from the review: change, reason, impact.

****SOW gap list:**** missing or vague clauses mapped to the task they should cover.

****Stage 4 inputs:**** rate cards, milestones, expected invoice timing and amounts.

****Session kit**** (if requested): planning guide outline, attendee list, agenda with chaos time.

Skill 8: stage4-spend-schedule

File: `skills/stage4-spend-schedule/SKILL.md` (followed by the shared response rules)

name: stage4-spend-schedule

description: Stage 4 of John Newcomb's Four-Stage Planning Model (Spend Schedule). Use to build a time-phased spend forecast, budget baseline, contingency reserve plan, cash flow forecast, or financial dashboard tied to the work; to answer when invoices will hit; or when budget and schedule disagree, spend cannot be traced to work, or total cost exceeds the available budget.

Stage 4: Spend Schedule

Stage 4 is where planning meets financial reality. Map every dollar to the timeline so the budget stops being a single number and becomes a time-phased plan you can monitor.

****Key principle:** "A plan without a spend schedule is a wish list."

Inputs (assemble from prior stages)

- Finalized Stage 3 schedule and invoiceable tasks and milestones (rate cards, billing triggers, invoice timing).
- Internal labor costs from Stage 2 effort estimates, if tracked.

- Overhead and infrastructure costs from organizational standards.

Run Stage 4 in the same working session as Stage 3; it needs no separate prep.

Activities

1. **Bottom-up cost estimation** from invoice-driving tasks and milestones. More accurate than top-down allocation because it is grounded in real work.
2. **Budget allocation by period or phase.** Spend aligns with the schedule: it should not precede the work or lag behind it.
3. **Cash flow forecasting.** When money leaves, based on schedule, billing terms, and resource loading. Prevents liquidity surprises.
4. **Contingency planning.** Reserves sized from risk analysis and estimate uncertainty. Contingency is not padding. (Use risk-triage for sizing.)

AAA in Stage 4

- **Assess:** Estimates consistent with historical actuals? Service costs aligned with benchmarks? Contingency sufficient for the risk profile? Any disconnect between schedule and spend? Run sensitivity on the biggest assumptions.
- **Assemble:** the inputs listed above.
- **Action:** build the spend schedule, set the baseline, write the contingency plan, produce the dashboard, present for approval.

Red flags to call out

- Spend piling up where it does not map to real work: stop and look harder.
- Invoices expected before the work that triggers them, or long after.
- Lines with no task or objective behind them (traceability break).
- Total exceeds budget: feedback to Stage 1 to reprioritize objectives or defer lower-priority themes.
- Commercial model creating risk (e.g. T&M where milestones would protect you): feedback to Stage 3.

Gate to Execution

Spend schedule aligned with the project schedule? Contingency adequate for the risk profile? Budget formally approved?

Scaling and Agile

- Small: simple spreadsheet mapping spend to weeks.
- Medium: monthly, category-level detail.
- Large: weekly, detailed categories, earned value tracking, rolling forecasts.
- Agile: rolling-wave; high level overall, detailed for current and next iteration; incremental funding.

If no dashboard standard exists, listing costs against the timeline is usually enough to add value.
 If the user wants a working spreadsheet, build one (a Sheets artifact, or .xlsx if they ask for Excel).

```
## Output format
**Spend schedule**
| Period | Category (Labor / Services / Materials / Other) | Task or
milestone ID | Provider | Amount | Invoice expected | Traces to objective |
Totals by period and cumulative.
**Budget baseline:** total, contingency reserve, management reserve.
**Contingency reserve plan**
| Reserve | Amount | Risks covered | Release condition | Approver |
**Financial dashboard template:** planned vs actual, forecast at completion,
variance, (earned value for large programs).
**Red flags found** and the feedback-loop action for each.
```

Skill 9: wbs

File: `skills/wbs/SKILL.md` (followed by the shared response rules)

```
---
name: wbs
description: Build, review, or fix a Work Breakdown Structure using John
Newcomb's Theme, Objective, Task structure from A Field Guide to Practical
Project Planning. Use for "build a WBS", "break this down", "decompose this
work", "review my WBS", task numbering, WBS dictionaries, scope change
control, or when estimates are unreliable, scope keeps growing, or work keeps
showing up that nobody planned.
---
```

Work Breakdown Structure (WBS)

If the Four-Stage Model is the backbone, the WBS is the engine. It is the hierarchical decomposition of scope into work that can be owned, estimated, sequenced, and delivered. It is **not a schedule**; it is the scope map answering "What work must happen for us to hit these objectives?" If it is not in the WBS, it is not in scope. When the WBS is weak, the schedule is fiction and the budget is guesswork. In Agile, read "backlog" for WBS; same discipline, different cadence.

Structure and numbering

```
| Level | Name | Description | Example |
|---|---|---|---|
| Project | Project (no number) | Top node | Enterprise CRM Implementation |
| 1 | Theme | Value-driven grouping | Improve Customer Experience |
| 1.1 | Objective | Measurable success criteria for the theme | Reduce
support response time from 48h to 4h by Q3 |
| 1.1.1 | Task | Assignable, estimable unit of work | Build legacy data
extraction scripts |
Start from Stage 1 outputs: each theme is a branch, each objective sits under
its theme, tasks achieve objectives. Always use IDs ("Task 1.1.3" beats a
twelve-word description).
```

Decomposition techniques

- **Top-down:** themes to objectives to tasks. Best when intent is clear and agreed.
- **Bottom-up:** list tasks, then group upward. Fast for experienced teams, but always validate against Stage 1 objectives or you lose the strategic thread.
- **Rolling wave:** detail near-term work, keep later work high level, refine as you go. Ideal for Agile and hybrid.

Stop decomposing when a task can be owned by one person, estimated with confidence (days, not months), and verified as done. If it cannot, go another level. Decompose each branch only as deep as confident estimation requires, and note why levels differ.

Always include

- Integration, system testing, deployment, and transition work (not just component outputs).
- A **Project Management** branch: plan, status reports, decision log, lessons learned, closeout.
- **Hidden work:** coordination, support, validation, handoffs.
- **External work** clearly flagged (feeds Stage 3). Ask per task: "Can we deliver this with internal capability?"

Domain starter branches (Level 2 ideas)

- IT / software: Infrastructure, Application Development, Data Migration, Integration, Testing, Training.
- Construction / facilities: Site Preparation, Foundation, Structure, Mechanical, Electrical, Finishes, Commissioning.
- Professional services: Discovery, Analysis, Design, Implementation Support, Knowledge Transfer, Project Management.

Governance

- **Peer review** before baselining: completeness, level consistency, clear wording, traceability.
- **WBS dictionary** (lightweight is fine; full one on complex projects): ID, description, owner, outputs, acceptance criteria, effort and duration, dependencies, associated risks.
- **Change control** after baseline: effective over formal. Check impact on scope, schedule, cost, and quality before approving. Update the WBS whenever a change is approved.
- Build it **with the team** (SMEs, technical leads), never alone.

Common mistakes and fixes

Mistake	Fix
---	---
Vague or inconsistent names ("Do integration")	Name so it is ownable, estimable, verifiable; otherwise decompose
Inconsistent levels without reason	Decompose to the level needed for confident estimates; note why
Missing integration work	Add integration, system test, deployment, transition
Missing PM artifacts	Add a Project Management branch
Never updated	Tie updates to change control
Built alone	Run decomposition sessions with SMEs

WBS quality checklist (score every item Pass / Fail)

1. Scope completeness: all required work represented.
2. No duplication.
3. Work packages named clearly and specifically.
4. Traceability: every task traces to an objective.
5. Consistent decomposition levels.
6. Each work package estimable with confidence.
7. Single owner per work package.
8. Integration, testing, deployment, transition included.
9. Project management work included (planning, reporting, governance, closeout).
10. Hidden supporting and coordination work exposed.
11. Provider-supported work identified.
12. Numbering convention applied consistently.
13. Peer reviewed by a qualified peer or SME.
14. Definitions documented (descriptions, dependencies, acceptance criteria) as needed.

Modes

- **Build:** produce the WBS from objectives or a description, then run the checklist on it.
- **Review:** run the checklist and the mistakes table on the user's WBS; rank findings by impact and give the fix for each.
- **Change:** assess a proposed change for scope, schedule, cost, and quality impact and state approve / reject / needs info.

Output format

| WBS ID | Element | Level | Type (Theme / Objective / Task) | Owner | Traces to | Internal / External | Est. effort |

Checklist results: 14 items, Pass / Fail, fix for each Fail.

Dictionary entries for complex or risky tasks (if useful).

If the user wants a working file, build it as a spreadsheet (Sheets artifact, or .xlsx if asked).

Skill 10: red-zone-planning

File: `skills/red-zone-planning/SKILL.md` (followed by the shared response rules)

name: red-zone-planning

description: Red Zone Planning from John Newcomb's A Field Guide to Practical Project Planning, for the final two to six weeks before a go-live, cutover, launch, migration, or major release. Use for go-live readiness, runbooks, cutover plans, rollback plans, smoke tests, go/no-go criteria, launch command structure, launch communications, or a project that is "95% done" but drifting toward a risky launch.

Red Zone Planning

You can be 95% complete and still fail on go-live day. The Red Zone is that last stretch, where timing tightens, small mistakes get expensive, and risk tolerance drops. The main plan gets the ship to the harbor; Red Zone Planning is the **harbor pilot** that docks it. It sits at the end of Execution or the front edge of Delivery.

Goal: **replace assumptions with verification, and vague ownership with explicit accountability.** Core question: **Can we finish with control?**

What it is and is not

- IS a short-horizon plan with day-of detail: sequence, owners, dependencies,

decision points.

- IS a readiness discipline that surfaces hidden work, decision debt, and late risk.
- Is NOT a full re-plan. Tighten the finish; do not rewrite the past.
- Is NOT a pep talk. No concrete output, it does not count.

When to use (trigger is readiness risk, not the calendar)

High-impact release (production, customer, revenue, compliance, or safety risk); coordination-heavy (many systems, teams, partners, tight cutover windows); the project is drifting and needs a reset before the window closes; or unresolved decisions, unclear ownership, or unvalidated assumptions that could surface in launch week.

Five exposure areas to hunt

1. ****Unmade decisions:**** approvals, cutover choices, scope trade-offs, go/no-go criteria, final authority.
2. ****Missing information:**** access, environment readiness, external dependencies, runbook inputs, monitoring coverage, open assumptions.
3. ****Hidden work:**** assumed, forgotten, never decomposed, or "someone else's problem".
4. ****Unowned steps:**** shared, vague, or no operator on the day-of plan.
5. ****Verification gaps:**** no proof the system is healthy, data is correct, or rollback is possible.

Inputs (come in ready)

Current milestone plan and release scope (in, out, must-land); open risks, issues, decisions with owners and status; constraints and timing windows (maintenance windows, change freezes, staffing limits, approval timing); access and environment readiness; provider and dependency commitments; draft runbook and validation approach; preliminary go/no-go criteria.

Non-negotiable outputs (the go-live package)

1. Full accounting of all remaining work.
2. Runbook / rollout plan: exact sequence, timing windows, prerequisites, validation points.
3. Go-live checklist: every action with an owner, a time or window, and a definition of done.
4. Rollback plan: what abort means, who can call it, when, how to back out, how to prove rollback worked.
5. Smoke test plan: minimum health checks with pass/fail criteria not open to debate.
6. Communications plan: who gets what, when, which channel, who sends it, before, during, after.

7. Command structure: who runs the event, where issues go, who is on call, who can say go, hold, or roll back.

Plus: final readiness review date and go/no-go criteria.

Mini Stage 1 and Stage 2

- Mini Stage 1: reconfirm the go-live objective, acceptance criteria, success signals, and the risks you will not accept.
- Mini Stage 2: decompose what is left into an assignable, sequenced, dependency-aware, time-bound checklist. If a step cannot be owned and verified, it is not planned.

AAA at the finish line

- ****Assess:**** What is truly left? What is at risk? Which assumptions are unvalidated? What could fail on go-live day, and how would we know?
- ****Assemble:**** right people, access, environments, approvals, partner support, monitoring, comms channels, before launch day.
- ****Action:**** lock the checklist, assign owners, set timestamps, confirm comms, set the command structure. Execute and document deviations.

Workshop format (90 to 120 minutes)

Participants (minimum): PM (facilitator), release/deployment lead (runbook owner), technical leads for affected systems, QA/testing lead, operations/support lead, and a business owner empowered to make scope and readiness trade-offs.

Pre-work sent in advance: milestone plan and scope, open risks/issues/decisions, constraints, draft runbook and validation approach.

Agenda:

1. Reconfirm the target: readiness definition, acceptance criteria, go/no-go authority.
2. Review current state: done, in flight, remaining.
3. Surface gaps: missing work, decisions, information, unowned steps.
4. Build the day-of checklist: tasks, owners, dependencies, timestamps, validation points.
5. Finalize rollback: triggers, decision path, steps, rollback validation.
6. Lock communications: cadence, channels, audiences, message owners.
7. Publish outputs and set the next checkpoint and final readiness review date.

Runbook template

Time	Step	Owner	Validation / Proof	Fallback
T-60	Open bridge / war room; confirm attendance	[Name]	All critical roles present or accounted for	Delay start / escalate

```
| T-30 | Final readiness check (access, approvals, monitoring) | [Name] | All
must-have criteria satisfied | No-go decision |
| T-0 | Begin deployment step 1 | [Name] | Step success signal captured |
Rollback trigger defined |
| T+30 | Smoke test (minimum health check) | [Name] | Smoke tests pass;
monitoring green | Rollback if fail |
| T+60 | Send "go-live complete" or "holding" update | [Name] | Message sent
and confirmed | |
Rules of the road: write steps as observable actions; use real timestamps
(the runbook is a schedule, not a narrative); one owner per step; define
rollback triggers in advance ("we'll know if it feels bad" is not a trigger);
log go/no-go, rollback, and scope decisions with time, owner, rationale.
```

```
## Failure modes this prevents
```

```
Hidden work, decision debt, unowned steps, communication gaps, no escape
route (a vague rollback plan is not a rollback plan).
```

```
## Output format
```

```
**Readiness gap list** by the five exposure areas: gap, owner, close-by date.
**Go-live package drafts** (Claude drafts each from what is known, marking
[assumed] items): remaining work list, runbook table, go-live checklist,
rollback plan, smoke test plan, comms plan, command structure, go/no-go
criteria.
**Workshop kit** (if requested): invite list, pre-work request, agenda.
**Readiness call:** Ready / Ready with conditions / Not ready, with the
reasons.
```

Skill 11: planning-health-check

File: `skills/planning-health-check/SKILL.md` (followed by the shared response rules)

```
---
name: planning-health-check
description: Diagnose a troubled, at-risk, or untested project or plan
against John Newcomb's unified planning system (AAA, Four-Stage Planning
Model, WBS, Red Zone) and prescribe recovery steps. Use for "my project is
off track", "what's wrong with this plan", "review this plan", project rescue
or recovery, scope creep, budget overruns, missed dates, misaligned outputs,
traceability audits, or PMO questions about planning standards, scaling, and
building planning culture.
---
```

Planning Health Check and Recovery

Most project failures are not execution failures. They are planning failures. A missed date is rarely a slow team; the work was never decomposed, dependencies were never found, or scope was never defined. An overrun is rarely reckless spending; the financial plan was built on assumptions. This skill finds which link in the system broke and prescribes the fix.

The system being checked

Lifecycle (Initiating, Planning, Execution, Delivery, Closing) gives context. Inside Planning, the Four-Stage Model gives structure. AAA gives decision discipline at every point. The WBS gives decomposition rigor inside Stage 2. Red Zone Planning secures the finish. Four-Stage without AAA = structure without decision quality. AAA without Four-Stage = good decisions without structure. WBS without Four-Stage = detail without strategic connection.

Symptom to root cause to fix

Symptom	Likely root cause	Apply
Nobody can say what success looks like; endless debate on priorities	Vague objectives	stage1-themes-objectives
Big plan swings after a key decision; rework	Unmanaged flashpoint	stage1-themes-objectives
Dates slip repeatedly; estimates wildly off; surprise work	Insufficient decomposition, missing dependencies	wbs, stage2-tasking-schedule
Work nobody asked for; scope creep	Broken traceability, gold plating, weak change control	wbs, four-stage-planning
Vendor can't deliver as planned; change orders; disputes over "done"	Late provider alignment, vague acceptance criteria	stage3-provider-alignment
Budget looks fine but doesn't match reality; invoice surprises	Disconnected financial planning	stage4-spend-schedule
Team acts fast then reverses	Action without Assess	aaa-method
Lots of analysis, nothing moves	Assess without Assemble	aaa-method
Everyone is ready but nobody commits	Assemble without Action	aaa-method
Escalations, surprised sponsors, status nobody reads	Communication planning gap	stakeholder-communication
Known risks keep becoming issues	No triage, no reserve	risk-triage
Near launch: open decisions, unclear owners, vague rollback	Decision debt, hidden work, no escape route	red-zone-planning
Plan never updated since kickoff	Plan treated as a document, not a living process	four-stage-planning (feedback loops)

Traceability audit

Build a light Requirements Traceability Matrix: Objective | Tasks | Validation check | Spend lines. Then:

- Any task with no objective: scope creep or gold plating.
 - Any objective with no tasks: unplanned commitment.
 - Any spend line with no task: unexplained cost.
 - Any provider clause with no task: SOW drift (use a cross-reference log).
- Pick the largest cost or most contested task and walk the chain from Stage 4 back to Stage 1 out loud.

Procedure

1. ****Intake:**** what is happening, lifecycle position, size, methodology, time to next major milestone, what has been tried.
2. ****Map symptoms**** to root causes with the table. Rank by impact.
3. ****Score the system**** Green / Amber / Red: Stage 1, Stage 2 and WBS, Stage 3, Stage 4, AAA discipline, communication, risk, and Red Zone readiness if launch is within about six weeks.
4. ****Recovery plan via AAA:**** Assess (facts vs assumptions, the Magic Question "Can we start right now?"), Assemble (who and what is needed to fix it), Action (owned, dated steps).
5. ****Treat every fix as a decision:**** what changes, why, impact on scope, schedule, cost. Log it and communicate it.

Scaling and PMO guidance (when the question is organizational)

- Scale formality to size: small projects run the stages lightly; large programs run a cycle per workstream with dictionaries, earned value, and formal decision logs.
- PMO role: establish a common, lightweight system (guidance, templates, training, governance), not rigid one-size rules. Standards without mentorship produce compliance without competence. High-performing teams need authority, information, responsibility, and accountability. The real legacy is a planning culture that outlasts the leader.
- Necessary governance, never process for process's sake.

Output format

****Health scorecard****

| Area | Rating | Evidence | Root cause |

****Top root causes**** (ranked, max 3 to 5).

****Recovery plan****

- Immediate (this week): owner-tagged actions.

- Next (2 to 4 weeks): owner-tagged actions.
- Sustain: practices that stop it recurring.
- **Traceability findings** (if an audit was run).
- **Skills to apply next**, and what Claude can draft now.

Copyright

Copyright 2026 John Newcomb. All rights reserved. Based on *A Field Guide to Practical Project Planning* (ISBN 979-8-9962259-2-7). More resources at thebikerprojectmanager.com; questions to john@thebikerprojectmanager.com.