A FIELD MANUAL

# The *Operator* Method

*For running content operations at agency scale, without scaling the team that runs them.*

CONTENTS

# In *order*.

01 · WHY THIS EXISTS

# Most agencies don't have a *tooling* problem.

Most agencies running content for clients don't have a tooling problem. They have a coordination problem dressed up as a tooling problem.

You can tell because every six months the founder reorganises the workspace. Asana to ClickUp. ClickUp to Notion. Notion plus Slack plus a shared drive plus a spreadsheet for the parts the other tools can't hold. Each migration burns two weeks of the team's time and resolves nothing structural.

The reason is that the system isn't the tool. The system is the set of decisions and handoffs that move a brief from "the client said something" to "the work is shipped." That system exists in your team whether you've written it down or not. Most of the time it lives in the founder's head, which is why the founder is the bottleneck.

> *This manual is the system, written down.*

It came out of watching 52 paying agencies and 2,000+ teams in trial run Clipflow. The patterns repeat. The leaks repeat. The fixes repeat. We've taken the parts that are tool-agnostic — the operating principles that work in any system — and laid them out in the order that actually fixes things.

The last section covers how Clipflow does this specifically. You can ignore that section and the rest of the manual is still useful. That's the test.

02 · WHO THIS IS FOR

# If most of *these* are true, the manual will land.

The agencies running on this framework all share a few traits. If most of these are true for you, the manual will land.

- **Team size: 3 to 20.** Below three, you don't have enough handoffs to need a system. Above twenty, you have the budget for an ops department and the manual stops being enough on its own.
- **Multiple clients in flight.** Five or more, typically. The framework's hardest at the seam where one client's process meets another's.
- **Content as the deliverable.** Not campaigns, not ads, not strategy decks. Things that get written, designed, edited, approved, and published on a recurring cadence.
- **Founder still inside the work.** If you've already promoted out of the operator role and have a head of ops who runs the day-to-day, this manual is for them, not you. Forward it.

If you're outside that profile, two specific cases:

- **In-house brand teams.** The framework still applies but the language is wrong — replace "client" with "stakeholder" throughout and most of it survives.
- **Solo operators or freelancers.** The pillars are the same but you can collapse Choreograph into Capture because you're the only handoff. Skip ahead.

03 · THE SHAPE OF THE PROBLEM

# Four symptoms. *One* underlying problem.

Before the framework, the diagnostic.

Most agency founders describe their core problem as one of four things:

01 "We can't scale without hiring more PMs."

02 "Our quality drops when we get busy."

03 "We're losing time to revisions and rework."

04 "I'm the bottleneck on everything important."

These sound like four different problems. They're four symptoms of the same underlying problem: the operating system in your team is informal, founder-resident, and doesn't scale.

The cost of an informal operating system isn't visible on a P&L. It shows up as:

→ Founder time absorbed into routing, clarifying, and re-explaining

→ New hires taking three months instead of three weeks to become productive

→ Client churn caused by inconsistent delivery, not poor work

→ Quiet attrition — people leaving because their role is "do whatever the founder needs today"

You don't fix this by hiring another PM. Hiring another PM into an informal system gives you two people with the same context-routing problem instead of one. The system has to exist outside any individual's head before more people can use it.

That's what the four pillars do. They name the operating system. Once it's named, it can be improved. Until it's named, every fix is a workaround.

04 · THE FOUR PILLARS

# Four operations. *One* loop.

Every content operation runs the same loop, whether it's been designed or not.

**Capture** is how work enters the system. The brief, the request, the "quick favour."

**Choreograph** is how work moves through the system. The handoffs between strategist, writer, editor, designer, reviewer, client, scheduler.

**Compound** is what the system remembers. The brand voice, the client preferences, the things that should never need to be re-asked.

**Capacity** is how much the system can hold. Who's available, who's drowning, what next month actually looks like.

The four pillars connect:

```
Capture → feeds → Choreograph
Choreograph → feeds → Compound (every completed project adds to memory)
Compound → informs → Capacity (better memory = better forecasting)
Capacity → gates → Capture (you stop accepting work you can't deliver)
```

> *The leak in any one pillar shows up as overload on the founder.*

That's the pattern. The founder fills whatever gap the system has. Find the gap and you find the founder's hidden second job.

The next four chapters cover each pillar in detail: what it is, how it breaks, what good looks like, and how to recognise the signal in your own team.

# 01. Capture.

*The point at which work enters the system. The front door. The moment a request crosses from the client's head into your team's queue.*

## What it is

Capture is the front door. It's the moment a brief, request, or task crosses from the client's head into your team's queue. Done well, it produces a unit of work that has everything needed to start: scope, deadline, deliverable spec, success criteria, named owner.

Done badly, it produces a Slack message saying "can we talk about the Q3 thing later" that requires three follow-up exchanges before anyone knows what's actually being asked for.

## How it breaks

The most common capture failures, in order of frequency:

- → **Multiple front doors.** Briefs arrive via email, Slack DM, the project portal, "she mentioned it on the call," and a shared Notion page. Each one has a different format and a different person reading it. Most agencies don't realise how many doors they have until they map it.
- → **Optional fields that should be mandatory.** The intake form exists but doesn't enforce scope, deadline, or asset spec. Half the briefs come in incomplete and require chasing.
- → **No named owner at intake.** Work enters the queue but it's not assigned. It sits in "needs triage" until someone notices.
- → **Capture happens in the conversation that creates the work.** The PM or strategist is on the call, hears the brief, and tries to remember it. By the time they write it up, half of it is reconstructed from memory.

## What good looks like

A single front door per work type. One intake form for blog content. One for video. One for ad creative. Each form enforces the fields you can't start without. Each form auto-assigns an owner based on work type or client.

Clients are trained — not asked, trained — to use the front door. "Please put that in the request form" said politely fifteen times until it's the only way work moves.

The brief lands as a complete unit, not a conversation thread. The next person who picks it up has everything they need to start.

### SYMPTOMS OF A BROKEN CAPTURE LAYER

*Read these out loud. If three or more describe your team this week, capture is your leak.*

- [ ] Every brief needs a follow-up Slack thread to clarify scope.
- [ ] Briefs sit untouched because no one knows who owns them.
- [ ] The same questions get asked of clients twice.
- [ ] The founder is the human router for "where does this go?"
- [ ] New team members can't pick up a brief without asking three people for context.

## THE FIX, IN ORDER

01 List every way work currently enters your system. Be honest. Email, Slack DM, Notion, verbal handoff — all of it.

02 Pick one work type to fix first. The one with the most volume.

03 Build a single intake form for that work type. Mandatory fields only.

04 Assign a single owner who triages incoming briefs daily.

05 Train clients to use the form by gently rejecting briefs that arrive any other way. "I'll get this moving as soon as it's in the request form" — said consistently for two weeks.

06 Once one work type is clean, do the next one.

# 02. Choreograph.

*The handoff layer. How work moves between strategist, writer, editor, designer, reviewer, client, scheduler — and how each handoff is made visible.*

## What it is

Once a brief enters the system, it moves. A blog post might pass through brief → outline → first draft → editor → revision → designer → client review → revision → schedule → publish. Eight handoffs. Each one is a place where work can stall, get lost, or come back wrong.

Choreography is how those handoffs are made explicit, owned, and visible. It's the difference between "the work is somewhere in the team's process" and "the work is in stage four, owned by Maya, due Wednesday."

## How it breaks

- → **Stages exist in the founder's head, not the system.** Everyone knows there's a review step but the system doesn't enforce it, so half the time it gets skipped.
- → **Handoffs happen via Slack ping.** "Hey can you take a look at this when you get a chance" is not a handoff — it's a hope. Work that's handed off this way slips invisibly.
- → **No WIP limits per stage.** The editor has 23 things in their queue. The designer has none. Throughput is bottlenecked at the editor and nobody can see it.
- → **Stage changes don't notify the next owner.** Work moves into "ready for design" and sits there for three days because the designer didn't know.
- → **No deadline visibility per stage.** Each stage has its own implicit deadline based on the publish date, but nobody's calculated backwards. The first stage runs long, every subsequent stage gets compressed, the final stage is rushed and quality drops.

## What good looks like

Explicit named stages. Every piece of work is in exactly one stage at any moment. Moving between stages is an action, not a passive drift.

Each stage has a named owner — the role, not the person. "Editor reviews" not "Maya does it." When Maya is sick, the role still exists.

WIP limits per stage, visible to everyone. If a stage is at limit, no new work moves into it until something moves out. This is uncomfortable at first and the right answer.

Stage changes trigger notification to the next owner automatically. No Slack pings. The system tells the next person it's their turn.

Each piece of work has a planned date for each stage, calculated backwards from the publish date. When a stage runs long, the system shows downstream stages getting squeezed before it happens.

### SYMPTOMS OF A BROKEN CHOREOGRAPHY LAYER

- [ ] Work sits in "ready for review" for days.
- [ ] People are surprised by deadlines that have been on the calendar for weeks.
- [ ] Reviewers don't know what changed since the last version they saw.
- [ ] The founder is the one who knows where everything is.
- [ ] Quality issues correlate with end-of-week or end-of-month — because everything compressed into the final stage.

### THE FIX, IN ORDER

01 Map your current stages on a whiteboard. Be specific. "Review" is not a stage — "internal editorial review" and "client approval" are two different stages with two different owners.

02 Name an owner for each stage. The role, not the person.

03 Set a WIP limit for each stage. Default to 3-5 per role until you have data.

04 Make stage changes visible — a board, a list, a tool, doesn't matter, but everyone can see where everything is at any moment.

05 Set per-stage deadlines for one client's work as a pilot. Calculate backwards from publish date.

06 Run for two weeks. Look at where things stall. Adjust WIP limits. Roll out to more clients.

# 03. Compound.

*The memory layer. What the system remembers about each client, each project, each pattern — so the next project starts from learning, not from scratch.*

## What it is

Every project teaches you something. The client's no-go words. Their approval pattern (the marketing manager always wants three options; the founder always picks the second one). Their brand voice quirks. The fact that they want CTA buttons in sentence case, not title case.

Compound is the layer that captures these lessons and makes them available to the next project, the next team member, the next year. It's how an agency stops re-learning the same client every six months.

Most agencies don't have a compound layer. They have a Drive folder with brand guidelines that nobody reads, and the actual operational knowledge of how each client works lives in the head of whoever's been on that account longest. When that person leaves, the agency loses three years of learning in a week.

## How it breaks

- → **Brand context lives in static documents that don't get updated.** The brand guidelines were written in year one. Three years later they're aspirational, not descriptive.
- → **Client preferences are tribal knowledge.** New team members learn them by getting things wrong and being corrected.
- → **Project post-mortems don't happen.** Each project closes and whatever was learned dies with it.
- → **The same revisions keep happening across projects.** "Oh right, this client doesn't like that headline format" — said for the fourth time.
- → **When senior staff leave, knowledge leaves with them.** The replacement spends three months rebuilding context that should have been written down.

## What good looks like

Each client has a living memory document. Not a brand guidelines PDF — a working document maintained by the team, updated as new things are learned. Voice, tone, no-go list, approval pattern, asset preferences, common feedback.

Brand documents are versioned alongside projects. When the brand voice evolves, the document evolves with it.

Project close-out is a stage in the choreography, not an afterthought. Twenty minutes per project: what worked, what didn't, what should the next project know.

The memory document is read at the start of every project for that client. It's the brief's prerequisite, not its appendix.

### SYMPTOMS OF A BROKEN COMPOUND LAYER

- [ ] New team members re-learn client preferences from scratch.
- [ ] Brand voice drifts when the original strategist leaves.
- [ ] The same feedback comes back from clients across projects.
- [ ] Senior staff leaving feels like an extinction event.
- [ ] "Ask Sarah, she'll know" is the answer to most client-specific questions.

### THE FIX, IN ORDER

01 Pick your highest-revenue or longest-running client first. The one with the most accumulated learning.

02 Spend an hour with whoever knows the most about that client. Write down everything. Voice, no-go words, approval patterns, weird quirks.

03 Make that document the single source of truth. Delete or archive every other doc that contradicts it.

04 Make reading it a stage in your choreography. The brief stage requires the writer to have read the latest memory doc.

05 Make updating it a stage too. At project close, what new thing did we learn? Add it.

06 Once one client is done, do the next.

# 04.

# Capacity.

*The throughput layer. How much your team can actually deliver — not in theory, but in reality, with the work mix and people you have.*

## What it is

Capacity is how much your team can actually deliver. Not in theory — in reality, with the actual work mix you have, with the actual people you have, with the actual interruptions that real life produces.

Most agencies guess at capacity. They guess wrong, accept work the team can't deliver, and find out three weeks in when something slips. The capacity layer makes throughput visible, forecastable, and honest.

## How it breaks

- → **Capacity exists only in the founder's head.** The founder is the one who knows whether the team can take on a new client. When they're wrong, it's the team who suffers.
- → **Allocation is implicit.** Maya is "on Client A and Client B" but nobody knows whether that's 30% or 80% of her time.
- → **Hours are the unit of measurement instead of throughput.** "We have 40 hours a week from Maya" doesn't tell you how many blog posts she can ship.
- → **Holidays, sick leave, training, meetings, and admin aren't subtracted.** Theoretical capacity is 40 hours. Real capacity is closer to 25.
- → **Forecasting is reactive.** New work comes in and you check whether you can do it. By then it's already in the pipeline.

## What good looks like

Capacity is visible at a glance. Every team member has a known baseline of throughput per week — not hours, but units of work they can ship at sustainable quality.

Allocation is explicit. Maya is at 70% this week (3 blog posts, 1 video script). When something gets added, the allocation goes up and somewhere else it has to come down.

Capacity is forecast four weeks out, updated weekly. New work isn't accepted blind — it's checked against the forecast first.

Holidays, sick days, and known absences are baked into the forecast. The founder isn't doing this maths in their head.

When the team is at capacity, the founder's job is to say no to new work or to negotiate timelines. Not to absorb the overflow personally.

### SYMPTOMS OF A BROKEN CAPACITY LAYER

- [ ] Founder is the only one who knows team load.
- [ ] New work gets accepted that the team can't deliver.
- [ ] Burnout shows up before the dashboard does.
- [ ] Client deadlines slip in the same week as a team member's planned holiday — because nobody planned for it.
- [ ] The question "can we take this on?" is answered by gut feel, not by data.

### THE FIX, IN ORDER

01 For each role on your team, define a unit of throughput. "Blog posts shipped per week at sustainable quality." "Video scripts completed per week." Get specific.

02 For each person, establish their baseline. Look at the last quarter — what did they actually ship? That's the real number, not the theoretical one.

03 Make allocation visible. A simple table is enough. Person × week × percentage allocated.

04 Update weekly. Subtract known absences.

05 Forecast four weeks out. New work checks against the forecast before it gets accepted.

06 When the forecast is full, say no or negotiate timeline. This is the hardest cultural change in the manual. It's also where the leverage is.

09 · THE DIAGNOSTIC

# Which pillar is *leaking*?

When the founder feels overload, the question is: which pillar is leaking? Run through this list. Don't think about it for too long — first instinct is usually right.

### 01 · CAPTURE

- [ ] Briefs arrive in more than two formats per week.
- [ ] You spend time clarifying scope after work has been "accepted."
- [ ] Work sits unowned because nobody triaged it.
- [ ] You've been the human router this week.

### 02 · CHOREOGRAPH

- [ ] Work stalls between stages without anyone noticing.
- [ ] You're surprised by deadlines.
- [ ] Reviewers see work without knowing what changed since last time.
- [ ] Quality drops at end-of-week or end-of-cycle.

### 03 · COMPOUND

- [ ] You answered the same client preference question twice.
- [ ] A new team member needed handholding on a long-standing client.
- [ ] Brand voice has drifted from where it was twelve months ago.
- [ ] You worry what happens when [senior person] leaves.

### 04 · CAPACITY

- [ ] You accepted work this month and aren't sure how it'll get delivered.
- [ ] Someone on the team is visibly overloaded.
- [ ] You've personally absorbed work the team couldn't take.
- [ ] You can't tell me what your team will be doing in three weeks.

Count the checkboxes per pillar. The pillar with the most boxes is your primary leak. If two are tied, fix Capture first — it's upstream of everything.

10 · THE ORDER OF OPERATIONS

# You can't fix all four *at once*.

You can't fix all four pillars at once. The intake order matters, because the pillars feed each other.

## The default order

**Capture first.** It's upstream of everything. A clean choreography layer is wasted on briefs that arrive incomplete. Compound is wasted on work that doesn't have a clear scope. Fix the front door first.

**Choreograph second.** Once briefs are entering cleanly, the next constraint is how they move. Make the stages explicit and the handoffs visible. This is usually where founders feel the biggest week-one improvement.

> *Slow is fast.*

**Capacity third.** With Capture and Choreograph working, you finally have data on what the team is actually shipping. That data is what makes Capacity forecasting honest instead of guessed.

**Compound fourth.** It's the longest-payback pillar. Worth doing, but the impact is felt at six months, not six weeks. Don't start here unless your only problem is staff turnover or context loss — most teams have more urgent leaks elsewhere first.

## When to break the default order

- → **Senior staff is leaving in the next quarter.** Compound first. Capture knowledge before it walks out the door. Then come back to Capture.
- → **You've just lost a major client because of inconsistent delivery.** Choreograph first. The damage is in the handoffs.
- → **You're about to take on a major new client.** Capacity first. Make sure the maths works before you sign.

## The two-week rule



Don't move to the next pillar until the current one is stable for two weeks. "Stable" means the symptoms in the diagnostic for that pillar have dropped to zero or near-zero.

11 · HOW TO BUILD YOUR OPERATION TOWARDS THIS

# How to build your *operation* towards this.

## Bend the tool you have, or buy a purpose-built one

You have two paths.

**Bend a generic tool.** Asana, ClickUp, Notion, Monday, Trello — all of them can hold the four pillars if you configure them carefully and discipline the team to use them as configured. The work is mostly setup and training. Cost: a couple of weeks of operator time, plus the ongoing cost of explaining "no, this goes in the form, not in Slack" forty times.

**Buy a purpose-built one.** A tool designed for content operations specifically — Clipflow is the one we built; there are a handful of others — encodes the four pillars by default. The intake form is a content brief, not a generic ticket. Stages map to brief, outline, draft, edit, approve, publish. Memory lives with the client. Capacity is a first-class view, not something you build in a spreadsheet.

The honest answer on which to pick: bend if your team is under five and your work mix is narrow. Buy if you're 6+ and shipping more than 40 pieces of content a month. The tipping point is when you spend more time configuring the system than running through it.

## Tactics by pillar

**Capture.** One form per work type. Mandatory fields: scope, deadline, deliverable spec, success criteria, owner. Auto-assign owner from work type or client. Reject everything that arrives any other way. Train clients to use the form by saying "I'll get on this once it's in the form" until they do. (In Asana: Forms with rules. In Notion: a database with a form view. In Clipflow: the default brief object — pre-fielded, pre-routed.)

**Choreograph.** Explicit named stages. Each stage has a role-owner, a WIP limit, and a planned date calculated backwards from publish. Stage changes notify the next owner automatically. WIP-at-limit blocks new work into that stage until something moves out. (In ClickUp or Asana: status fields plus automations. In Notion: a board view with Slack integration. In Clipflow: stages are the data model — moves are events, the next owner is notified, capacity is checked.)

**Compound.** A living memory document per client. Voice, no-go list, approval pattern, weird quirks. Read at the start of every brief. Updated at the end of every project. Versioned alongside the work. (In Notion or Drive: a templated client doc, linked from every project. In Clipflow: client memory is a first-class object — voice and patterns sit next to the brief, not in a separate tab.)

**Capacity.** Throughput per role per week, not hours. Allocation as a percentage. Forecast four weeks out. Subtract holidays, sick days, and training. New work gets checked against the forecast before acceptance. (In a spreadsheet: rows of people, columns of weeks, cells of percentages. In ClickUp: workload view. In Clipflow: generated from actual throughput, updated weekly, surfaces warnings before you accept work you can't deliver.)

## What good early implementation looks like

Two weeks in: every brief comes through the form. The founder hasn't routed work in three days. The team can answer "where is this?" without asking anyone.

Six weeks in: stage stalls are visible before the founder feels them. WIP limits hold. Planned dates are within 10% of actual.

Three months in: a new team member onboards in a week. The founder spends a meaningful chunk of their week on work only they can do. The plateau breaks.

## What kills implementations

- → **Trying to fix all four pillars at once.** Pick one. Stable for two weeks. Move on.
- → **Over-automating before the patterns are stable.** Configure manually first. Automate the third repetition, not the first.
- → **Doing it secretly.** The team needs to know what's changing and why. The framework is a shared vocabulary or it's nothing.
- → **"Just this once."** The first time you accept a brief outside the form, you've taught the team that the system is optional.

12 · FIELD NOTES

# Three things *worth saying*.

Three things worth saying that don't fit anywhere else.

## The framework is a tool, not a religion

If a pillar doesn't fit your team, change it. Skip it. Rename it. The point isn't fidelity to the framework — the point is having a named operating system you can improve. If "Choreograph" is the wrong word for your team, call it something else.

The four pillars are stable across the agencies we've seen. The vocabulary around them isn't. Use the words that land for your team.

## The founder is the test case

The thing that tells you the framework is working is what the founder does this week.

Before the framework, the founder spends their time routing, clarifying, re-explaining, and absorbing overflow. After the framework — when it's running properly — the founder spends their time on the work that only they can do. New business. Strategy. Hiring. The things the system can't replace.

If you've implemented the framework and the founder's week looks the same, the system isn't working yet. Keep going.

## The point of all of this

Most agencies plateau at the size where the founder's bandwidth runs out. Eight people, twelve, fifteen — somewhere in there, the system that worked at five stops working, and instead of fixing the system the founder fills the gap personally.

The plateau is not about talent or ambition or work ethic. It's about the system. The agencies that grow past the plateau aren't the ones with better people — they're the ones who built an operating system that can hold more weight than the founder can.

> *The four pillars are that system. Run them, and the agency gets bigger than the founder.*

# A 30-minute *diagnostic*.

*If the framework named something you've been quietly fixing every week, there's a thirty-minute conversation worth having. Not a demo — a diagnostic.*

*We map your current loop, find the leak, and you decide if Clipflow is the system that closes it. We only run these with teams of 3 to 20.*

BOOK THE DIAGNOSTIC →