

Ms

MATSIEMS PRESENTS

The Unbothered Machine

Free edition: the first five chapters

5

CHAPTERS

1

PARTS

5

DIAGRAMS

2026 EDITION

ILLUSTRATED

Contents

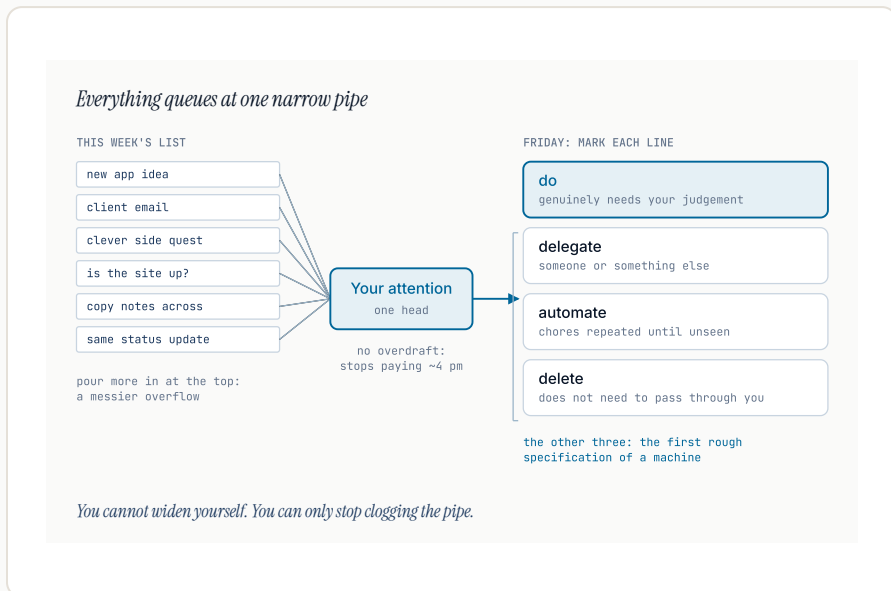
01	The Ground Floor	ch.01-05
	<i>why one person needs a machine</i>	

PART 1 • CH.01-05

The Ground Floor

why one person needs a machine

You Are the Bottleneck



Everyday demands queue through one narrow pipe of attention, then get sorted: do, delegate, automate, delete.

Ideas are not your problem. You have a notes app full of them, a voice memo from the shower, three domain names bought at midnight and a half-built prototype that was going to change everything in March. Ideas arrive like weather. Nobody has ever run short.

What you run short of is attention. Not time, exactly; you could find another hour if you stopped refreshing things. Attention is the scarcer stuff: the capacity to hold one problem in your head long enough to finish it. A solo builder has one head. Every project, every client email, every clever side quest draws on that single account, and the account does not do overdrafts. It simply stops

paying out, usually around four in the afternoon, and you call it tiredness.

This is the real constraint, and it is worth stating plainly because most advice gets it backwards. Productivity culture tells you to generate more: more ideas, more output, more content. But if attention is the narrow pipe, pouring more in at the top only makes the overflow messier. The useful question is not "what else could I do?" but "what must pass through me, and what could pass through something else?"

The bottleneck is not a flaw in the system. It is the system's most expensive component, and you are it.

So treat yourself like an expensive component. This week, keep a plain list of everything that required your attention, each item a line, nothing clever. On Friday, read it back and mark each line with one of four words: **do**, **delegate**, **automate**, **delete**. The do pile is what genuinely needs your judgement. The rest is the first rough specification of a machine. You will find, as I did, that a startling share of the list is chores you have repeated so often you stopped seeing them: checking whether a site is up, copying notes from one place to another, writing the same status update in slightly different words.

That list is where ABC, the system this book describes, began. Not with a grand architecture, but with a mild irritation at being the only thing in the building that could do anything. Every later part of the machine — the agents, the skills, the dashboard — exists to protect one narrow pipe from things that do not need to go through it. You cannot widen yourself. You can only stop clogging the pipe.

A Job Is a Machine You Rent

Rent the machine, or own one

	RENTED: A JOB	OWNED: YOUR MACHINE
WHEN YOU STOP	Pay stops hour for hour	Output can continue asleep or on a beach
IMPROVEMENTS	Stay in the building scripts, processes, know-how	Compound for you every fix accumulates
YOU LEAVE WITH	A reference and a slightly better CV	The machine still yours in a year
WHO WORRIES	Someone else their sales team, their servers	You sales, sysadmin and worrier

THIS WEEK: ONE RECURRING SLOT THAT BELONGS ONLY TO THE MACHINE YOU OWN

Mon	Tue	Wed	Thu	Fri	Sat	Sun
rented	rented	rented	rented	rented	2 h: own	

Ownership is not freedom from work. It is choosing which work accumulates.

A rented machine and an owned one compared row by row, plus one weekly slot kept for the machine you own.

Consider what a job actually is, stripped of the lanyard. Somebody else built a machine: a set of processes, customers, tools and habits that turn effort into money. They need a part for it. You are the part. In exchange for fitting neatly, you receive a salary, which is the rent they pay for your hours, and you in turn rent their machine for the privilege of being paid.

There is nothing shameful in this. Rented machines are excellent. They come with someone else's sales team, someone else's servers, someone else's worry. Most of the world's useful work

happens inside them, and I have done plenty of it myself. Contract work still pays my bills this year, and I am grateful for it in the way one is grateful for a good umbrella.

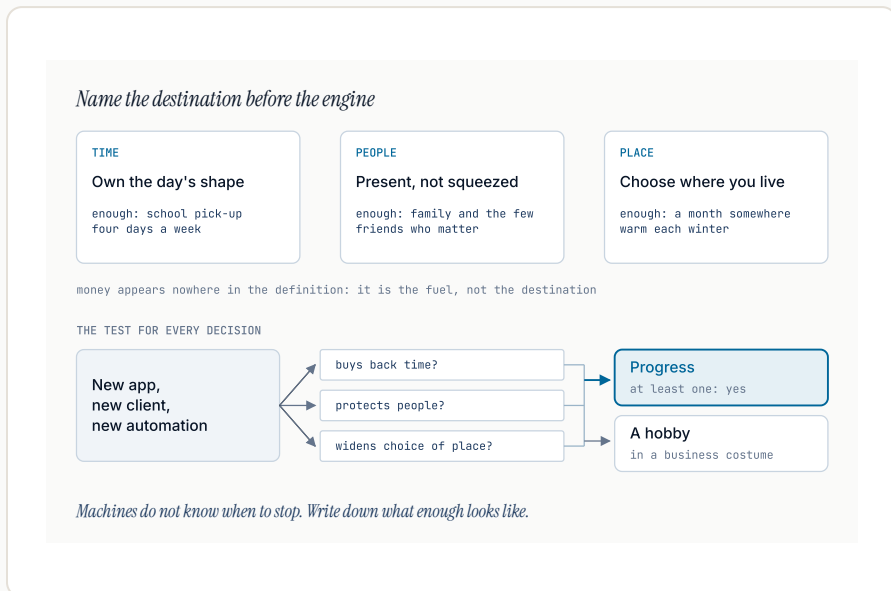
But a rented machine has two properties worth noticing. First, when you stop turning up, it stops paying you. The output is tied to your presence, hour for hour. Second, nothing you build inside it is yours to keep. The processes you improved, the scripts you wrote, the knowledge you accumulated: all of it stays in the building when you hand back the pass. You leave with a reference and a slightly better CV.

Owning a machine inverts both. Its output can continue when you are asleep or on a beach, at least in principle. And every improvement compounds in your favour rather than someone else's. The trade-off is that you are now also the sales team, the server administrator and the worrier. Ownership is not freedom from work. It is freedom to choose which work accumulates.

The decision this week is not dramatic. You do not need to resign. Simply look at your working hours and ask how many of them build something that will still be yours in a year. If the answer is none, carve out one recurring slot, even two hours on a Saturday morning, that belongs only to the machine you own. Protect it like a dentist appointment. Put it in the calendar with a dull name so nobody tries to negotiate it away.

Most people never build their own machine, not because they lack skill, but because every spare hour gets rented out too. The landlord is very persuasive. He is also, conveniently, never there when the rent goes up.

The Trinity Life, Defined



Time, people and place, each with a concrete enough, and the three-question test for every new decision.

Before you build a machine, decide what it is for. This sounds obvious, which is precisely why almost nobody does it. Builders tend to start with the engine and work out the destination somewhere around the third rewrite. By then the engine has opinions of its own.

My answer is something I call the Trinity life, and it has three parts: **time**, **people** and **place**. Time means owning the shape of the day, not merely having free hours scattered around obligations. People means being present for family and the handful of friends who matter, rather than squeezing them between calls. Place means

being able to choose where life happens, whether that is a coast, a city or simply a kitchen table that is not under a deadline.

Notice what is missing. There is no yacht. There is no number with lots of zeroes. Money appears nowhere in the definition, and that is deliberate. Money is the fuel, not the destination. If you put a revenue figure at the centre, the machine will happily optimise for it while eating the very time and attention you meant to free. Plenty of successful founders have built exactly that: a profitable trap with an excellent dashboard.

Naming the destination gives you a test you can apply to every decision. Does this new app, this new client, this new automation buy back time, protect people or widen choice of place? If it does none of the three, it may still be interesting, but it is a hobby wearing a business costume. Hobbies are allowed. They just should not be mistaken for progress.

Here is the practice. Write your own three words on a card, or in the first line of a file the machine reads every morning. They need not be mine. Perhaps yours are health, craft and garden. Then, beneath each, write one sentence describing what enough looks like. Not more; enough. "Enough time" might mean school pick-up four days a week. "Enough place" might mean a month somewhere warm each winter. Concrete, small, checkable.

The word *enough* is the most radical one in the file. Machines do not know when to stop. People must tell them. A system without a definition of enough will keep growing until it has consumed whatever you built it to protect. Decide what the machine buys before it starts spending you.

Agents Are Interns Who Never Sleep

Confidence is not admissible; evidence is

WHY BUILD A MACHINE AROUND THEM

well-read, types 1,000 words a minute
never sleeps, never sulks
cheap per hour, happy to repeat

WHY THE MACHINE NEEDS STRUCTURE

has never worked at your company
does not know where anything is kept
may say 'done' when it did something else

IN ABC EVERY DELEGATED TASK HAS THE SAME SHAPE

Brief

names the outcome
+ "done means..."

Context

files it may read
and nothing else

Agent loop

try, look at result,
try again

Evidence

link, passing test,
URL, log line

"I have updated the file"
not admissible: only confidence

here is the file, here is the diff,
here is the check that passed

Interns who are never checked are a liability with good grammar.

Why agents are worth building around, why they need structure, and the brief, context, loop, evidence shape.

Strip away the marketing and an AI agent in 2026 is a fairly simple thing. It is a language model given a goal, some tools — a terminal, a browser, a file system, an API or two — and permission to loop: try something, look at the result, try again. That is it. No consciousness, no ambition, no secret plan to replace you over the weekend.

The most useful mental model I have found is the intern. A very well-read intern who has skimmed most of the internet, types at a thousand words a minute, never sleeps and never sulks. Also an intern who has never worked at your company, does not know

where anything is kept, and will occasionally announce with total confidence that a task is finished when it has quietly done something else.

Both halves of that description matter. The first half explains why agents are worth building a machine around: tireless, fast, cheap per hour, happy to do the same dull thing a hundred times. The second half explains why the machine needs structure. You would not hand an intern on day one the keys to production and a vague instruction to "improve things". You would give them a clear brief, the files they need, a definition of done and a way to check their work.

That is the whole craft, really. In ABC every piece of delegated work has the same shape: a brief that names the outcome, the context the agent is allowed to read, and the **evidence** required before anything counts as complete. A link, a passing test, a deployed URL, a log line. Not "I have updated the file", but here is the file, here is the diff, here is the check that passed. The agent can be as confident as it likes; confidence is not admissible.

Try this with the next task you hand to any AI tool, even a chat window. Before you send the request, add one closing sentence: "Done means..." and finish it with something you can verify in under a minute. Then actually verify it. You will be surprised how often the intern's work is excellent, and how often the check catches the one time it was not. Both outcomes are information.

Interns who never sleep are a gift. Interns who are never checked are a liability with good grammar.

Hundred Small Bets

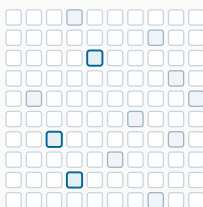
With one bet ignorance is fatal; with a hundred it is data

ONE BIG SWING

The One

years of runway, a hockey stick
50/50: glory or a blog post

A FLEET OF 100 MICRO-APPS, EACH A NARROW CORNER



- ☐ most do little
- ☐ a few do something
- ☒ show where to look next

a dud costs an afternoon
and a domain renewal

THE CATCH: SMALL BETS ARE SMALL ONLY IF PLACING ONE IS CHEAP

3 months per app

100 of them = a career

factories, templates, skills

Price of a sandwich

100 of them = a portfolio

this week: ship the smallest slice of The One, live by Sunday; next week, something unrelated

Count bets placed, not dreams nurtured.

One big swing against a grid of 100 micro-apps, and why the cost of each bet must fall to a sandwich.

The traditional founder story involves one enormous swing. A single product, years of runway, a pitch deck with a hockey stick, and a fifty-fifty chance of either glory or a reflective blog post. It makes for gripping biographies. It makes for poor odds, especially if you are one person with a mortgage.

The alternative is less cinematic and more forgiving: lots of small bets. Instead of one app that must succeed, you build a fleet of micro-apps, each modest, each cheap, each aimed at a narrow corner of the internet where someone is searching for something specific. A list of the best walking routes in a region. A guide to a

niche hobby. A scroller of facts about a single topic. None of them needs to be a company. Each only needs to earn its keep.

The arithmetic of small bets is kind to the uncertain. You do not know which niche will work; neither do I, and neither does anyone selling a course. With one bet, ignorance is fatal. With a hundred, ignorance becomes data. Most will do little. A few will do something. The ones that do something tell you where to look next, and the ones that do nothing cost you an afternoon and a domain renewal you can decline.

The catch is that small bets are only small if the cost of placing each one is small. If every app takes three months, a hundred of them is a career, not a portfolio. This is why the rest of this book is so preoccupied with factories, templates and skills. The machine's job is to drive the cost of one more bet towards the price of a sandwich. Only then does the fleet make sense.

So the decision for this week is a sizing one. Take whatever project you are currently treating as The One and ask what the smallest shippable slice of it would be. Not a minimum viable product with a roadmap; a single page, a single list, a single useful thing that could be live by Sunday. Ship that instead. Then do it again with something unrelated next week. Count bets placed, not dreams nurtured.

I have no idea which of my apps will matter most by December. That is not a weakness in the plan. It is the plan. Big swings are for people with spare lives.

END OF THE FREE EDITION

Ninety-five chapters to go.

You have read the opening of Part One: why one person needs a machine. The full book builds it, part by part.

- 100 chapters in 10 parts, each with a detailed diagram
- One goal, a scoreboard, a memory and your first skills
- Agents with job titles, a control plane and an app factory
- A5 PDF, EPUB and the Machine Kit of 10 templates

matsiems.com/books/the-unbothered-machine

One payment, instant download. Mat Siems · Unbothered Publishing

Colophon

One hundred chapters across ten parts, each carrying a single editorial figure drawn on a 4-point grid with 1px hairlines, no shadows, and one accent focal node. Set in Instrument Serif, Inter, and JetBrains Mono. Theme: matsiems #006699.