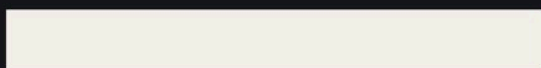
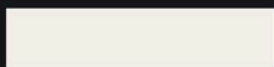
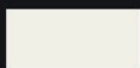


STEPS, PATTERNS AND CONCEPTS

AI AGENTS THAT DO THE WORK

How to build them and put them to work



FLORIS KNOL

From Zero to AI Agents That Do The Work

How to build them and put them to work

Floris Knol

Foreword

Something has changed in how work gets done, and almost nobody can tell you precisely what.

You hear that AI changes everything. You see examples that are impressive and that you can then do nothing with. You read posts by people who know, courses that promise, and articles that after three paragraphs tell you the main thing is to start. And then it is Monday morning, you are looking at your own work, and nothing has changed.

This book was written by someone who did do it, and it is about what he learned along the way.

What happened here

What this book is about is an environment where agents do work by themselves. Not one assistant helping you, but several running at once, each with an assignment, a limit, and somewhere to record what they did. And that in turn is not an end point: it is the step toward an organization that runs on AI rather than one that uses it.

I built that, from nothing, with what is simply available now. What made the difference was the way of choosing, and I found that on the way.

The unusual part is not the speed. It is that it got easier every day instead of harder.

With almost everything it is the other way round. What you build gets heavier: more parts, more exceptions, more things that can

break, and so less speed. Here the opposite happened, and that was not luck and not talent. It came from a small number of moves that looked like nothing on the day.

Those moves are what this book gives you.

A report from along the way, with the patterns pulled out.

I wrote everything down as it happened: what I did, why, what it cost, and what went wrong. Out of that pile come about ten patterns that kept repeating. Those patterns are the book; what I happened to be building is the example.

That distinction matters, because otherwise half the readers are watching instead of doing. What I built was technical, but none of the patterns is about technology. They are about the order in which you do things, about where your time goes, about what you can hand to a machine and what you cannot, and about how you notice you are working in the wrong direction. A bookkeeper, a photographer, a lawyer and a builder all run into them. Which is why every pattern also comes with an example that has nothing to do with software.

And you can do it yourself. Every number in this book comes from work that was really done, and next to every number is how it was measured. You run the exercises on your own screen, and then you see whether you come close. That is rare in this subject and it is deliberate: what you have seen work yourself, you do not have to take from me.

Including what went wrong. There is a day in this book on which I drew the wrong conclusion three times in a row, a moment when my own tools lied to me, and a measurement that proved my way of measuring was broken rather than the thing I was measuring.

Those are not in here out of modesty. They are in here because that is where most of the learning was.

Where this goes

What starts with one person and an empty machine ends in something that takes on work by itself in the real world.

Put like that it sounds like boasting, and I understand that. But it is literally the shape of this book, and not one of the steps in between is spectacular. It helps to see them in advance, because then you understand why the beginning is so small.

It starts with making one thing cheap. One task faster, one check that does itself, one mistake you do not make twice. That is the first half of the book and it is plain craft.

Then a hundred run at once. Not because that is impressive, but because the question changes as soon as there is more than one: what you take in at a glance with one has to be written down with a hundred, and what you do yourself with one has to happen without you with a hundred.

Then something goes wrong, and it corrects itself. Something that notices it is stuck and stops. Something that sees the same mistake coming back and changes the instruction rather than the mistake. Something that has a limit and does not cross it, not even when it would be convenient.

And then it becomes an organization. That word is not meant grandly. An organization is something with a goal nobody has to hand it again every morning, a budget that can run out, roles that know about each other, and a memory. Once those four are there, it is no longer a tool waiting for you.

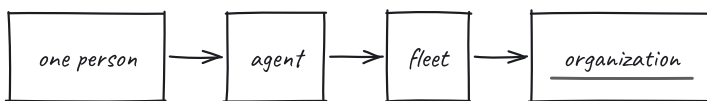
And then comes the part almost nobody mentions. Such an organization does things in the real world. It buys. It signs up to services and takes on obligations by doing so. It hires people — because the idea that people stop being involved is a fantasy: there is work a human has to do and work a human has to sign for. It adjusts its goals, and sometimes it scales itself up.

Which means it has to **take part in the formal world.** An accountant turns up. An audit turns up. There is a standard a process has to meet, and somebody asking: show me who approved this, when, and on what basis.

That is not an awkward extra but the condition. An organization that cannot show what it did is not autonomous — it is unsupervised. The difference between those two words is exactly the difference between something you may use inside a company and something you can only try out at home.

And the good part is that they are the same things. The recording you need in order to learn from it is the recording an auditor wants to see. The limit that stops the money running out is the limit a controller asks about. Do the one well and you get the other for free — skip it, and you will have to build it later at the worst possible moment.

I am not yet at the point of an autonomous organization. I am somewhere in the middle of it, and the book says exactly where. But the direction is clear, and every step toward it is one you can take today.



each step makes the next one possible

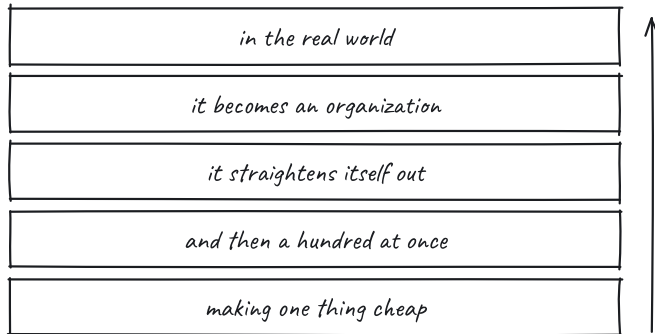
The shape of this book: what begins with one person ends with something that takes on work by itself.

How to read it

The five steps above are the five parts of the book, in exactly that order: making one thing cheap, then a hundred at once, then what happens when it goes wrong, then it becomes an organization, and then the real world. After that comes one more chapter that looks beyond where I stand — the only one that is not yet practice.

Every chapter comes down at least once to something you can try tonight. It sits in the text itself, with why it works and where it goes wrong. You can read past them; you can also do all of them, and then you will have built something by the end while reading.

And every chapter closes with one line: **what this made cheaper**. Those lines stack. Read them back one after another and you see what the book is actually about.



The five parts rest on each other: each one makes the next possible.

Why now

Because you are not late yet.

That is not a sales line but an observation. The things that make this possible are all here now, they are cheap, and almost nobody uses them together. That is exactly the moment when somebody who does builds a lead that money cannot close — because the lead is in what you have stacked, and stacking takes time nobody can skip.

In two years this will be ordinary. Not yet.

8. Why are you still doing that yourself?

It is not a reproach. It is a question you ask yourself while you are in the middle of something, and the first time the answer is usually still a good one.

Because I do it faster than I can explain it. Often true. *Because it only takes two minutes.* Also true. *Because it needs judgment.* Sometimes true.

What makes this question special is not that it changes the answer. It is that **it stays the same question while the answer shifts under your feet**. What could not be done last month can be done now. And you do not notice, because you are busy.

Why the answer shifts without your seeing it

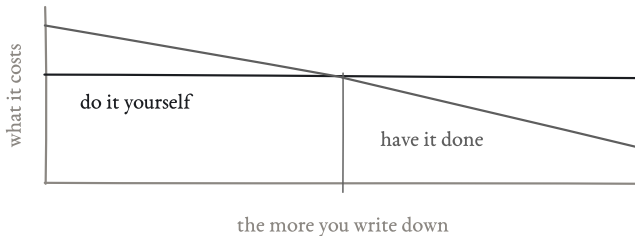
At some point you decided that something was your work. That decision was good — then. Afterwards it became a habit, and a habit does not account for itself.

Meanwhile three things change, and none of the three knocks.

What a machine can do grows. Not in leaps you notice, but a little each month. The task that last year had to be redone three times now works first time — and nobody sends you a message about it.

What it costs to explain falls. That is the quieter of the two, and the more important one in this book. Every time you write something down — a rule, an example, a mistake in a log — the next explanation gets shorter. At some point *explaining* becomes cheaper than *doing*, and that moment passes without a sound.

And what you have left over in attention falls too. As more runs, your minute is worth more. The same two minutes that cost nothing last year cost something now — not because you are busier, but because they yield more elsewhere.



What a machine can do rises, what explaining costs falls — and somewhere they cross.

Three movements, all slow, all in the same direction. The decision you once made becomes a little less true every month. And because it is never wrong all at once, it stays.

Which is why it is a question and not a rule

A rule would be: *hand over everything a machine can do*. That is nonsense, because then you hand over things you do faster yourself, and things that need your judgment, and things you cannot check yet.

The question is better, for three reasons:

It does not ask for a system. You do not have to take an inventory of all your tasks. You ask it while working, about the thing you are working on.

It accepts "because it cannot be done yet". That is an honest answer, and a useful one: it names exactly what would have to

change. *Not yet* is a date without a number.

And it keeps itself current. The same question, a month from now, about the same work, can give a different answer without your having had to track anything.

The four answers, and what to do with each

In practice four kinds of answer come up, and each asks for something different.

"Because I am the only one who knows." This is the most expensive answer and it sounds the most like a compliment. What you know sits in your head and travels nowhere — not to a colleague, not to an agent, not to yourself in three months. **The action here is not to hand it over but to write it down.** Only then can the question be answered at all.

"Because explaining takes longer than doing." Almost always true for once, and almost never true for ten times. Work it out rather than feeling it: how often did I do this in the past month? At one, you are right. At five, you lost three times ago.

"Because I have to be able to trust it." A good answer, and it points at the real work: not the doing but the checking. Can you tell afterwards whether it went well? Can you tell cheaply? As long as you cannot, you should indeed do it yourself — and then building that check is the actual task.

"Because it cannot be done yet." The best answer, because it is concrete. Something is missing: an access, a key, a piece of tooling, an example. Write down *what* is missing. That is a ticket, not a feeling.

<i>only I know it</i>	<i>explaining takes longer</i>	<i>I have to trust it</i>	<i>it cannot be done yet</i>
write it down	count how often	build the check	note what is missing

Four answers, and each points at a different action.

Notice what is in none of the four: "because I enjoy it". That is a valid answer, and the only one that needs no action. But then call it that, instead of putting one of the other four in its place.

The one time I did not ask it

If I could redo one thing from those first days, it is this, and it is exactly the question of this chapter that I failed to ask myself.

I sat working out **the bootstrapping phase** myself. How the tooling could come to build itself, in what order, with which intermediate steps — the kind of puzzle you can spend hours on and feel proud of at the end of the evening.

That was work I need not have done. **I should have formulated it and handed it over.** Not "solve this", but: this is where we need to get to, this is what already exists, these are the conditions — now propose an order.

Why did I not? Because it felt like the interesting part. And that is exactly what is treacherous about this question: **you ask it automatically about dull work, and almost never about work you enjoy.** Dull work you are glad to hand over. Puzzle work you hold on to, and you call it thinking.

Sometimes it is. But the test is not whether it is enjoyable; the test is whether your head adds something that would not otherwise be

there. For that bootstrapping order the answer was no — I had no knowledge that was decisive there, I had a preference.

So if you ask this chapter's question once a day, ask it about the work you like most. That is where the answer stands still longest, and where it is most expensive.

Try it yourself · Ask what this arithmetic is still doing with you

What you do. Somebody puts the budgeted hours next to the actual ones every month. It takes an hour, it never goes wrong, and for that reason nobody has looked at it again. Have it recalculated and see whether the answer is still the same. Allow five minutes.

Type this

For each job, put the budgeted margin next to the actual one, from cost-control/ and post-calculations/. Handle the running jobs separately and show there how much of the budgeted cost has already been spent.

What you see. Six lines. Two finished jobs that stayed neatly within budget, one that lost sixteen percentage points of margin — **and a running job that has already spent 106 per cent of its budgeted cost while it is still going.** That last one appeared in no monthly overview, because a monthly overview looks at finished jobs.

What came back

ENJB-2026-001	budget	7.9%	actual	8.5%	+0.6
ENJB-2026-004	budget	15.2%	actual	8.6%	-6.6
ENJB-2026-010	budget	19.9%	actual	3.5%	-16.4

```
ENJB-2026-002 budget 8.2% running (106% of the
                                budgeted cost spent)
ENJB-2026-012 budget 14.9% running (54%)
ENJB-2026-007 budget 8.7% running (60%)
```

Where it goes wrong. **Ask explicitly for the running jobs separately.** Without that request it computes the margin across everything, and then a job about to derail disappears between three that went well. **Watch what is NOT in the source.** These figures come from cost-control/, and the outstanding supplier invoices are not in there — so the real cost is higher than what you see. That is not an arithmetic error but a source error, and chapter 14 is about that. **And do not ask for a judgment.** Whether sixteen percentage points is bad depends on things that are not in this folder.

Then like this

```
Add the outstanding supplier invoices per job to the cost so
far, and show which picture changes.
```

And then this

```
with outstanding supplier invoices included:
  ENJB-2026-007 60% -> 94% of budget
  ENJB-2026-012 54% -> 67% of budget
  ENJB-2026-002 106% -> 106% (none outstanding)
and: ENJB-2026-011 has a supplier invoice but is
      not in cost-control/ at all
```

And this is not about software

The question is older than the subject of this book, and that is exactly its strength.

- **An owner who does the books themselves** did so once because there was nobody else. Two years later there is somebody, and they still do it — not out of distrust, but because nobody asked the question.
- **A doctor who schedules their own appointments** defends it with exceptions that are real. The question is not whether those exceptions exist, but whether they should drive the whole diary.
- **A mechanic who checks every fault personally** is right about the difficult cases. The easy eighty per cent just comes along with them.
- **A writer who does their own layout** says nobody else does it the way they do. That is true, and the question is whether the difference is worth anything to the reader.

In all four it is not about laziness and not about delegation as a virtue. It is that a decision that was once right has quietly stayed put.

And the question is polite. That is not incidental. "Why are you still doing that yourself?" leaves room for a good answer, which is why you dare to ask it — of someone else, and of yourself.

Try this: once a week, ten minutes — have the five things listed that you repeated most often, each with the condition still missing before you no longer have to do it yourself. Ask for that missing condition and not for a solution: a solution you cannot yet check is a problem you have moved.

That round produces four kinds of answer. For three of them you now know the action; the most expensive one asks for something else, and it is not delegation.

Tip: when you hear yourself say “*I am the only one who knows that*”, have yourself questioned until it is on paper as a method somebody else can follow. Let the questions be asked rather than dictating: you automatically skip whatever is obvious to you, and that is exactly the step the other person is missing.

And there is one sentence that removes almost every blockage, provided you put it in the right place.

Put this in your standing instructions: *when in doubt, choose a plausible option yourself and carry on; say afterwards what you chose and why, and only ask when the choice is irreversible, costs money, or goes outside.* Most standstill comes not from hard questions but from small ones — what should this be called, where do I put it — and every time such a question reaches you, the work waits until you have time.

That is not a blank cheque, and it only works because the rest of this book sits underneath it. Checks run along, limits live in the execution and not in a request, and there is a trail of what was decided. **A plausible choice is safe as long as a wrong choice is visible and reversible** — and where it is not, the last clause of that instruction draws the line. Do insist that the choice is named afterwards; otherwise you get progress without a trail.

What this made cheaper: every time the question got a different answer, work moved that never came back onto my pile.

Read on

That was the foreword and chapter 8 of *From Zero to AI Agents That Do The Work*. The book has 27 chapters in five parts, from making one task cheap to an organization that takes part in the real world -- each with an exercise on the example administration.

Paperback € 39, e-book € 19: florisknol.com/buy. Buying the book gives you early access to Wantzel AI Studio.

Feel free to pass this sample on.