

The background is a vibrant yellow. It is decorated with a complex network of black and white dots of various sizes. These dots are interconnected by thin black lines, some of which are solid and others dashed. The lines and dots form a series of overlapping circles and arcs, creating a sense of movement and connectivity. The overall effect is a modern, tech-inspired aesthetic.

**LAURENT DÉSÉCHALLIERS**

# **CONTEXT**

Create a context that enables AI  
to understand what matters,  
connect your knowledge  
and put it to work toward your goals.

# Le Contexte

*How to build a context that lets AI grasp what matters, connect what you know, and keep building on what you learn in pursuit of your own goals.*

---

Laurent Deséchalliers

---

## A Note on the Translation

---

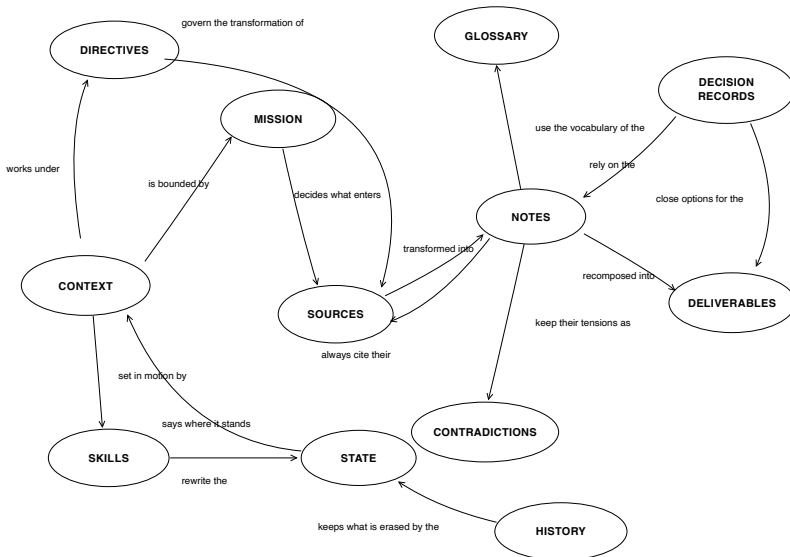
This edition is a machine translation. An AI produced it from the French text.

No human translator has been over it. The ideas, the structure and the examples follow the original closely; some of the phrasing, though, may read awkwardly, and a few passages still carry the shape of the French.

That is the price of having this book exist in several languages rather than in French alone. Where a wording is in doubt, the French edition is the one that counts.

## Definition of the Context

*A context is a system bounded by a human mission, in which an AI organises and draws on what we know, where we know it from, how the pieces connect and where the work stands, so as to help the human reach their goal without starting over. In doing so, the system builds a lasting information capital and makes it grow: the knowledge it produces stays available, grows richer over time, and can be put to work towards other ends.*



---

## Introduction

---

We have never had so much information within reach. And yet we still spend a large part of our energy finding it again, putting it back in order and rebuilding what we had understood a few days earlier.

A useful idea rarely arrives in a clean form. It comes out of a conversation, takes shape in a document, meets an objection in a meeting, produces a decision, then becomes an article, a specification or a project. Our tools keep every one of those traces. What they struggle to keep is the thread that runs between them.

Artificial intelligence changes part of the problem. It can read, draw things together, reword and transform very different kinds of material. But a conversation with an AI stays a fragile moment: the subject moves on, the discussion grows longer, and then you have to restate the goal, the decisions taken, the sources used and what is still outstanding.

It was to answer that difficulty that I built the `Context`.

A `Context` is a workspace devoted to one subject. A human and an AI keep in it the goal being pursued, the sources, the ideas, the rules and the state of the work, so as to produce results without having to rebuild everything at each session. The AI can then draw on the actual history of the work. It finds where an idea came from, recalls a choice, flags a

---

contradiction, points out a missing piece of evidence, picks a draft back up where it stopped. **It helps to organise thinking and to test it against itself. The human keeps responsibility for meaning and for decisions.**

This system came out of my daily work, from a single file and a very simple request put to an AI: organise what I am about to tell you. It grew through one attempt after another, each difficulty making a gap visible. It is still changing: some of the answers set out here have held across several uses, others remain provisional replies to problems I still run into.

So this book does not start from a finished theory. It starts from problems I encountered: five hundred emails that had become impossible to follow, ideas scattered across several tools, customer documents to answer without losing the work already done, texts picked up again after an interruption.

It is written for people who turn information into understanding, into decisions and into deliverables: product managers, consultants, writers, researchers, analysts and, more broadly, knowledge workers. It offers a way of making an AI a working partner over time while keeping control of your own material.

The book is organised as two complementary books. **Book I — Understanding the context** tells how the system came about and what it changes in the relationship with an AI. **Book II — Building and Sustaining the context** explains how to build it and use it over time.

---

They can be read one after the other or separately. The chapter **What Order to Read This Book In** sets out the possible routes.

## A Note on the Examples

---

The ideas and examples in this book draw on more than thirty years of work in technology. They come from situations I have encountered across a range of projects and professional settings, and from many conversations with product managers, CTOs, engineering managers and others at different companies. In other words, they span several organisations and several stages of my career.

To serve the purpose of the book, I have altered, generalised or combined these situations. The use cases, figures and metrics illustrate recurring problems and what I have learnt from them. They are constructed examples: none describes a real client, product or organisation, or reflects my employer's data, practices or positions.

The examples from my own life — the building I help run, my articles and my own writing — rest on things that genuinely happened to me. Details that might identify anyone have been changed.

The analysis, methods and opinions expressed in this book are mine alone and do not represent anyone else.

---

## About the Author

---

I have worked in software for several decades, and these days I work as a product manager. My career has taken me from development and architecture to organising teams, then to the product, its operations and its marketing.

Changing roles changed my relationship with information. Knowing a technical answer, or finding a document again, stopped being enough. I had to pull scattered signals together, work out what they said as a whole, weigh up several directions, and write for very different readers.

My interest in artificial intelligence and distributed systems goes back a long way. It now runs into a very practical question: **how do you work with an AI without handing over control of your information, your decisions and your thinking?**

I have written for the trade press and worked as a technical reviewer for M2 Édition and Eyrolles. These days I write on [malorean.net](https://malorean.net), about product management, AI, ways of working and information systems.

I also write for *Proposition Marketing*, an independent magazine covering marketing strategy and practice. Writing for readers who come from marketing and from the boardroom forces me to explain product questions without leaning on a shared vocabulary. This book asks the same of me.

I would very much enjoy talking with other people who build systems of this kind — close to mine, different or complementary: the subject deserves to be discussed, challenged and enriched by other practices. With academics working on neighbouring questions too — I spent a few years in two CNRS research centres, and reconnecting with that world would be a pleasure.

Laurent Deséchalliers

- [laurent@malorean.net](mailto:laurent@malorean.net)
- <https://malorean.net>

---

## Acknowledgements

---

My first thanks go to you.

You who had the slightly mad idea of opening this book — a book rather thicker than planned, which no doubt repeats itself here and there and deals with an abstruse subject.

So far I have worked on the `Context` fairly alone. I hope that solitude will not last: that the subject will make others want to explore it, discuss it, argue with it and move it on. Perhaps the next articles published on [malorean.net](http://malorean.net) will then be a chance to thank a good many contributors.

Above all, I thank my wife, Christine.

She watched the evenings and the weekends stretch on in front of this book. She supported me right through the writing of it, while knowing how to remind me that it was late and really time to shut that damned computer.

I promise, it was only a phase.

And, far beyond this book, thank you for sharing this wonderful life with me all these years.

---

## What Order to Read This Book In

---

The book follows the order in which I experienced things: first a problem, then a makeshift solution, and finally a system. So the full description of the `Context` only comes in Book II.

**The “understanding” route** is that order. You read straight through, and each piece of the system turns up just after the gap that produced it has been shown.

**The “building” route** is for someone who has already lived the problem and wants the mechanics, not the story.

1. **Definition of the `Context`**, a few lines to know what we are talking about.
2. **Part 5 — Under the Bonnet of a `Context`**, in the order of its six chapters: the map, opening, feeding, transforming, producing and deciding, saving.
3. **Part 6 — The `Context` in Its Ecosystem**, to connect what you have just built to the rest of your information systems.
4. **Book I**, whenever a design choice strikes you as arbitrary: that is where you will find the problem that caused it.

The shortcut comes at a cost. Book II was written for a reader who already has the situations of Book I in mind, and it refers back to them. So you will come across examples without all their context — that is the price, and it is still worth paying if you already know why you are here.

---

## Try Before You Read

---

This book describes a method, and a method is hard to tell. You can watch one run before reading the theory, or while reading it — that is often the best moment to understand what a given piece is for.

Three repositories are open at [github.com/laurent-maloreau](https://github.com/laurent-maloreau).

**Contexte Nano Notion** is the quickest to install. It requires neither a development environment nor technical skill: a ChatGPT and a Notion are enough. ChatGPT carries the intelligence of the Context, Notion serves as its memory — that is where the sources, the notes, the decisions and the state are written and read back from one conversation to the next. Within minutes you have a Context that holds.

**Contexte V2** is the full implementation, the one I use every day and the one this book was written with. It installs with Claude Code or with Codex, either one, and takes a little more setting up than the previous one.

**LLM Wiki** is not a Context. It is a knowledge library built from documents — books, standards, articles, courses — from which the AI draws short notes linked to one another, which you then query in plain language. The Context keeps the thread of a mission; the LLM Wiki keeps, on a large scale, what has been read. Part 6 explains how the two work together.

Nothing obliges you to start there. But if at some point the reading feels abstract, that is the shortest path back to something concrete.

## More by the Author

---



**Malorean Corp** — Not long ago, turning an idea into a prototype could take weeks. Today, someone working with artificial intelligence can do it in just a few days. That is a remarkable change. But the real transformation may lie elsewhere: in how we work together. *Expected release: late November 2026.*

More information at <https://malorean.net>.

---

# Contents

BOOK I

Understanding the Context

**Part 1 — The Birth of a Need** ..... 21

1.1 Five Hundred Emails and a Story That Vanishes ..... 22

1.2 A First Definition ..... 31

**Part 2 — What the Context Changes** ..... 39

2.1 From Conversation to Work You Can Pick Back Up ..... 40

2.2 Two Clients, a White Paper and a Contradiction Worth Keeping 47

**Part 3 — Three Uses in Practice** ..... 57

3.1 Answering a Client Questionnaire Without Losing the Work Done .  
58

3.2 Writing an Article from a Thought Still Taking Shape ..... 64

3.3 Ripening a Product Subject in the Middle of the Chaos ..... 72

**Part 4 — What Deliverables Leave Behind** ..... 81

4.1 The Answer Fits in a Cell, the Reasoning Does Not ..... 82

4.2 Six Months On, the Answer Is No Longer Enough ..... 86

BOOK II

Building and Sustaining the Context

**Part 5 — Under the Bonnet of a Context ..... 93**

5.1 The Full Map ..... 94

5.2 Opening: Recovering the Mission and the Present ..... 105

5.3 Feeding: Letting the Material In ..... 116

5.4 Transforming: From Sources to Ideas ..... 122

5.5 Producing and Deciding ..... 131

5.6 Saving and Picking Up Again ..... 141

**Part 6 — The Context in Its Ecosystem ..... 149**

6.1 Querying Structured Knowledge ..... 150

6.2 Querying the Product Itself ..... 157

6.3 The Day I Had Three Contexts ..... 165

6.4 Reading at the Neighbour’s Without Moving In ..... 171

6.5 The Cost of Circulation ..... 178

Before closing

**Closing ..... 185**

7.1 Epilogue ..... 186

7.2 Kinships and Neighbours ..... 188



## BOOK I

# Understanding the Context

---

- 1 The Birth of a Need
- 2 What the Context Changes
- 3 Three Uses in Practice
- 4 What Deliverables Leave Behind



# Part 1

## The Birth of a Need

---

The Context did not begin as an idea about artificial intelligence. It began with a mailbox that had become unmanageable, and with a realisation that took time to emerge: the sheer number of messages was not the real problem.

This part tells that story, then draws an initial definition from it — simple enough to remember, yet substantial enough to guide us through the rest of the book.

## Five Hundred Emails and a Story That Vanishes

---

The first time I built a `Context`, I was not setting out to invent a way of working with artificial intelligence. I was just trying to stop losing the thread of my building's affairs.

I sit on the residents' committee. That puts me on the receiving end of messages from the managing agent, from the owners, from the tenants and from the contractors:

- one email announces a new problem,
- the next gives details of a quote,
- a third pushes a repair back,
- a week later someone answers in the same thread about a different block.

In the meantime three other matters have moved on.

Volume was not the only problem. Some five hundred emails were spread across fifteen or so subjects:

- works,
- finances,
- insurance,
- general meetings,
- owners in arrears,
- tenders,
- incidents and assorted requests

- ...

Each moved at its own pace, sometimes for months on end. Whoever follows a job day by day keeps its story in their head without effort. They know which contractor turned up, why the schedule shifted, and which decision is still waiting on an answer. When I opened a new message, I did not always have that continuity. I had to work back up the thread, find the earlier exchanges and reassemble the situation before I could even take in the new information.

I wanted to be able to ask simple questions:

- where do we stand on the roof?
- which contractor sent the last quote for the gutters?
- what did we decide about that leak?
- what are we still waiting on from the managing agent?

My mailbox held the answers. What it did not give me was an overall view.

### **One File and One Sentence**

I started with the simplest arrangement I could think of: one file and one instruction given to an AI.

▮ *Pull the information together and break it into subjects.*

I passed it the emails. It attached them to subjects and kept a summary of each.

- Everything about the works came together.
- The money questions formed another group.

- Contractors, insurance and general meetings gradually found their place.

1

Even that first attempt gave me something. I could ask where a subject stood without rereading every message about it. The AI was no longer just finding a match in the mailbox: it was showing me what it had understood about the situation.

The file reorganised itself as new information came in.

- A heading appeared when a new subject started to matter.
- Two headings split apart when they began to mix different realities.

Information was no longer merely piling up. It was taking a shape I could work with.

Then that first system produced an answer that was accurate and still not enough.

### **The Job Was Finished, but Its Story Had Gone**

When a job reached its end, its summary read: “job complete, signed off”.

That was indeed where it stood. It was also a loss.

- When had the job started?
- At what point had it been delayed?
- Why had the schedule changed?
- When had the variation been signed?

All those stages had counted. The latest update had wiped them in favour of the final position.

I had asked for a summary. The AI had kept it current by replacing the old state with the new one. It had followed the instruction, but the instruction did not protect the history. So I needed two views of the same subject.

- The first had to answer “where are we?” quickly.
- The second had to keep the events in order: start of the job, delay, new agreement, repair, sign-off.
- The present state let me act.
- The history let me understand how that state had been reached, and check a point when I had doubts.

That distinction came out of a very concrete failing. I thought it was merely a better way of organising the file. It became one of the most important principles of my work with AI: **what is true now and what happened do not serve the same purpose.**

### The AI Could Not Guess My Rules

Another problem showed up with names:

- two contractors shared a surname.
  - Telling them apart meant understanding what the email was about.
- one owner appeared sometimes under her maiden name, sometimes under her married name.

With nothing to go on, the AI could create two people where there was only one.

Other differences came from how we worked.

- Owners in arrears for less than a hundred euros were not chased like the others.
- That threshold came from no universal rule the AI could have known.
- It came from our way of doing things.

I began writing this knowledge down as standing rules in the file:

#### # Handling rules

- Martin mentioned in connection with the roof means the roofer; Martin mentioned in connection with the grounds means the landscaper.
- Dupont-Lemoine and Lemoine are the same owner, flat 14.
- List owners in arrears for under a hundred euros separately.

Nothing more sophisticated than that. Three sentences I would otherwise have repeated in every conversation, and never had to say again.

Each rule came from an error I had watched happen or from a genuine ambiguity. It captured something I carried in my head and that the AI could not reconstruct with any certainty. So the system was starting to hold two different kinds of information:

- information about the building,
- the way I wanted it handled.

## The File That Helped Started to Get in the Way

Over the weeks, the file grew. It passed five hundred lines, then eight hundred, then a thousand.

That growth was proof of its usefulness:

- subjects were easier to follow,
- contractors were clearly identified,
- past decisions could be found again.

But what had made the system possible was starting to limit it.

- Too many different kinds of information shared the same space.
- Rules were mixed in with facts.
- Finances sat next to works, people, insurance and tenders.

Finding the right information was becoming difficult again.

I began by separating out the handling rules. They did not describe the building; they said how to work with the information about it. Setting them apart made them easier to find and to apply.

Then I divided the information into several subject areas.

Works got their own space, as did finances, contractors, decisions from general meetings and insurance. An overview said what each one held. The AI could then open the subject the question called for, instead of going through everything each time.

Finally I gathered the specialist vocabulary: legal terms, acronyms and phrases particular to a residents' committee.

The point was not to build an encyclopaedia. It was to give a stable meaning to the words the exchanges depended on.

I still had no grand architecture in mind. I added a piece when a problem made that piece necessary.

### **Trusting the System Without Losing Sight of the Sources**

The AI made mistakes now and then. It could bring together two pieces of information that were not quite about the same thing, or pick the wrong contractor between two namesakes. So the system had to speed up my understanding without becoming the only version of the facts available.

I kept every original email. When an answer needed checking, or the stakes rose, I could go back to the information in its original form. The summary gave me quick access. The source let me check it.

That choice was what made the whole thing usable. I could live with the occasional mistake in how the emails were organised, because the original messages remained within reach. The same level of trust would not do for a bank document, a regulatory obligation or an irreversible decision. The risk always depends on what you hand the system and on whether you can check it.

So the Context did not replace my mailbox. It turned its contents into a working memory, while leaving me a way back to the original messages.

## What Had Appeared

In a few weeks the starting point — one file and one sentence — had become something that met several needs at once: showing the present state, keeping the history, applying rules particular to the building, telling subjects apart, steadying the vocabulary and going back to the sources.

When a new email arrived, I could take an interest in what it changed instead of rebuilding everything that had come before it. When a question came up, I could ask it in the terms of the problem rather than hunt for the right message in the right thread. When the AI got something wrong, I could fix a rule or go back to the original text.

I had not designed the `Context` in one go. I had watched it appear through a run of shortcomings: the summary wiped the history, names ran together, rules got lost, the file grew too big. Each fix made the work more durable and easier to pick up again.

That origin matters. The `Context` did not come from wanting to file things more thoroughly. It came from needing to understand information that keeps changing, preserve the evidence needed to check it, and carry the work forward without starting over.

---

**WORTH REMEMBERING — TWO VIEWS OF THE SAME SUBJECT**

The state answers “where are we?”. The history answers “how did we get here?”. A summary kept current by overwriting the previous one keeps the first and loses the second. Both are needed: one to act on, the other to understand and to verify.

---

## A First Definition

---

The system that came out of those five hundred emails can now be given a first definition.

*A Context is a workspace devoted to one subject, where a human and an AI keep what they need to understand, make decisions and produce results over time.*

The definition is deliberately plain. The Context this book describes has a precise organisation, which the second half will set out. To see what it brings, six markers are enough to begin with:

- a goal,
- source material,
- ideas,
- a state of the work,
- rules of its own,
- results to produce.

### A Subject and a Goal

The Context for the building was not trying to hold everything I knew. It pursued one goal: understanding the matters in hand, keeping their history, and finding what I needed in order to act on the residents' committee.

That goal drew a boundary. An email about the roof belonged there. An interesting read with no bearing on the building did

not. The limit kept the system from turning into a store where anything deemed interesting would pile up.

A `Context` therefore starts from an intention. It can serve to:

- write a book,
- explore a product problem,
- prepare a decision,
- answer client questionnaires,
- follow a business entity,
- ...

### Source Material That Remains Traceable

Information enters a `Context` in two ways.

The first takes in what already exists before the working session:

- an email,
- an article,
- a book,
- a web page,
- the transcript of an interview
- ...

These are the **document sources**. They arrive as finished objects and stay as they are. In the case of the building, the email announcing a postponed repair belongs to this first category.

The second takes in what gets formulated during the work with the AI. When I explain a situation, correct a reading, sharpen a distinction or state a conviction that comes from

experience, what I say becomes a **session source**. It builds up as the exchange goes on. When I explain that two names refer to the same owner, what I say comes in through this second door.

The difference lies in when and how the information appears.

- The document exists before the session and is brought into it.
- The session source is born during the session and grows with it.

Both then serve the same purpose: preserving what was received or said before it is interpreted.

- They do not guarantee that everything in them is true.
- They let you find again what was received or said, and then judge how reliable it is.

From these sources, the `Context` can derive an idea, a rule or a picture of where the work stands. Once the information has been interpreted and organised, the resulting knowledge is stored elsewhere. If a date looks wrong, if two people have been conflated or if a reading becomes questionable, you can go back to where it came from.

### **Ideas Made Usable**

Piling up sources is not enough. Five hundred well-kept emails are still five hundred emails to read.

So the `Context` gradually turns those sources into usable knowledge. It brings out:

- subjects,

- facts that matter,
- rules,
- distinctions,
- open questions,
- relations.

An isolated exchange joins the history of the job it belongs to. Several mentions of the same contractor feed one shared piece of knowledge. A correction stops a confusion from happening twice.

That in-between step matters because it makes the information usable.

- The sources preserve what was received or said.
- Organised knowledge lets you understand and act.

In more complex work, some ideas will serve more than once: one piece of product knowledge feeds a decision, a specification and a set of documentation; one verified fact helps answer several client questionnaires. The result changes, while the idea and where it came from stay available.

### **A State That Lets You Carry On**

Work in progress always has a current state: what is known, what is still uncertain, what has been decided and what needs to happen next.

For the building, the state answered an immediate question: where does this job stand today? Its history answered another: how did we get here? The system needed both so that I could genuinely pick the work back up.

The distinction goes beyond following a job. When I break off writing a text or exploring a product problem, I want to find more than a pile of documents. I want to know:

- what is established,
- which point is still shaky,
- which action can set the work going again.

The `Context` keeps that continuity outside the conversation.

- The session with the AI can stop.
- The work keeps a state the next session can return to.

### **Rules Particular to the Work**

The AI has general knowledge. What it does not have is much of what makes a situation particular:

- the two contractors who share a name,
- the way a team classifies a signal from a client,
- the level of evidence expected before publishing a claim
- the tone chosen for a book,
- ...

The `Context` keeps those rules with the work they govern. They can be corrected when experience shows their limits. They then apply without the human having to restate them in every request.

Those rules do not hand the AI responsibility for the work. They make explicit what the human expects of it. The human sets the goal, provides or validates the source material, corrects interpretations and makes the decisions. The AI takes in information, connects and organises it, asks

questions, rephrases ideas and prepares results within those boundaries.

1

### Results Shaped Around a Need

The `Context` remains focused on what it is meant to make possible.

For the building, the result might be:

- an answer to a question,
- a summary of where the works stand,
- preparation for a discussion.

In other uses it will be:

- an article,
- a client questionnaire,
- a piece of documentation.

The same workspace can produce several results because it preserves the underlying sources before they are shaped into a final form. Sources are not locked inside the first document that uses them. Ideas stay available for another question or another audience.

That capacity is what separates the `Context` from a plain archive.

- An archive lets you keep things.
- The `Context` organises how stored information is reused and transformed.

## A Form That Can Start Very Simply

In my own practice, a `Context` takes the shape of files I can read, move and keep independently of whatever tool helps me work on them. That form answers my need for control and for duration. Other ways of building it are possible, as long as they preserve the same functions.

The first `Context` had nothing sophisticated about it. It fitted in one file. It grew when its limits became visible. That progression still matters: you do not need to build the whole system described in this book before trying the idea out.

A subject, a goal and a few initial sources are enough.

- The AI helps organise them.
- The human notices what is missing, adds a rule, keeps a source, or separates two pieces of information that do not do the same job.
- The architecture appears at the pace of the problems you run into.

So the `Context` is not a particular folder structure. It is an enduring working relationship between a human, an AI and a body of information organised around a goal.

The next part looks at what that relationship changes. A conversation with an AI can already be fruitful. The shift to a `Context` begins when that conversation takes place within an environment that can preserve its goal, its history and the state of the work beyond the current moment.

---

**WORTH REMEMBERING — THE SIX MARKERS OF A** CONTEXT

---

A goal that draws a boundary, traceable source material, ideas made usable, a state that lets you carry on, rules of its own, and results to produce. Those six markers are enough to begin. The architecture comes afterwards, when a problem you run into makes it necessary.

---

# Part 2

## What the `Context` Changes

---

A conversation with an AI can already achieve a great deal. It explains, rephrases and suggests, and often gives a better answer than the one you would have found on your own.

So this part asks what a `Context` adds to that. The answer lies not in the quality of its answers, but in what survives from one session to the next: work that resumes where it left off, and documents that preserve disagreements instead of smoothing them away.

## From Conversation to Work You Can Pick Back Up

---

A conversation with an AI can start very simply.

I give it an idea that is still muddled. It rephrases it. I ask for an outline, then another. It raises an objection I had not thought of. I clarify what I mean, reject one approach and explore another. In a few minutes, the discussion moves forward work that might otherwise have stalled at a blank page.

That is one of the great strengths of chat: one question is enough to start. The form suits a short task very well. I ask for a summary, an explanation, a few options. The answer comes, I use it, and the conversation can stop.

The relationship changes when the work lasts.

### Five Sentences Before I Can Write

I reopen a six-day-old conversation to pick up a text I started the week before. Before I can write a single line, I type this:

*Continue the text on the quality of job reports. It is written for operations managers, not developers. We decided not to shorten the form; it is too early to make that choice. The figure on audits comes from a single interview, treat it as a hypothesis. We had got as far as the third section.*

Five sentences just to reorient the AI within work it had already seen six days earlier.

Everything I have just typed existed before I typed it: the angle, the audience, the decision, the weak point and the position in the text were all in the earlier conversation. I simply retrieved them from my own memory, because the conversation had preserved them in the order in which they were said.

But work follows a different logic.

- An instruction given at the start still has to apply at the end.
- A source added yesterday has to be able to support a paragraph written tomorrow.
- An old decision has to prevent the same discussion from reopening.
- A point left pending has to resurface at the right moment.

The longer the conversation grows, the more of that structure the human has to hold in their head. When a new session begins, they rebuild the situation for the AI: here is the goal, here is what changed, here is the tone, here are the decisions, here is where we were.

Chat is still useful. The weight of continuity, though, falls on whoever is using it.

### **Giving Lasting Information a Home of Its Own**

A Context changes that relationship by giving the work a space of its own, separate from the conversation.

The discussion stays the place of exchange. It is where you question, explore, sharpen and correct. What must last goes into the workspace:

- the goal,
- the sources,
- the rules,
- the ideas that have firmed up,
- the decisions,
- the texts,
- the current state of the work.

The difference may look slight. It is like the one between a meeting and the project it is about. A good meeting can move a project forward, but the project does not live inside the memory of the meeting. The documents, the decisions and the actions go on existing after it ends.

Keeping the full transcript of the chat would not have the same effect. It is valuable when you want to find exactly what was said, but it follows the shape of the conversation: a question, an answer, a correction, a digression, a return to the subject. To pick the work up you would still have to read that sequence through and decide what counts now.

The `Context`, for its part, transforms what was said.

- A correction goes where it can stop the mistake coming back.
- A decision stays visible as a decision.
- A source stays attached to what it supports.

- A text exists in its latest complete version.

Memory is thus organised around action. It is not only trying to keep the past. It is preparing the next move.

### **One Page to Recover Thirty-Two Texts**

I measured what that changes when I came back to work on this book.

The manuscript consisted of thirty-two texts at the time. Some parts were written, others still empty. One section had been split because it took up more room than it deserved. Two passages were waiting for me to read them over. One claim was not backed by any real example and had to be reworked. Taken one by one, none of those items is hard to understand. Together, though, they form a situation you have to rebuild before you can write a single useful line.

I could open the thirty-two texts, reread the outline, go through my old conversations and hunt down the remarks left pending. That piece of archaeology would take time and leave me with a nagging doubt: had I forgotten the very point that was meant to prevent a mistake?

Instead, I open the book's `Context`. It gives me a restart note, which that morning said roughly this:

# Restart brief

## Where things stand

Thirty-two texts exist, from the outline to the conclusion.  
The technical parts are written; the first two are still  
too theoretical.

## What to know before carrying on

The reader we have in mind is new to the subject: they lose  
the thread as soon as a concept appears before a concrete  
example.

The claim about the failure of document systems is still  
not backed by any personal experience.

## The next action

Rewrite the opening around the case of the five hundred  
building emails.

## What was left open

Two passages are waiting for me to read them over.  
Several sources have come in without being turned into  
notes.

One page, read in under a minute. I know what to do, and I  
have not opened a single one of the thirty-two texts.

**Moving Forward Is Not Getting Back Up to Speed**  
**Moving forward means knowing the next action. Getting**  
**back up to speed means recovering the logic of the work**  
**before you take that action.**

A task list can tell me that a source needs to be processed. It  
does not explain why that source comes first, which part of the  
book it is meant to clarify, or which risk it is meant to address.  
To get back up to speed, I need a richer set of information:

- the goal of the work;

- where it has actually got to;
- the decisions that still steer what follows;
- the weak points that must not be forgotten;
- a next action precise enough to get started.

The restart note follows precisely that order. “Thirty-two texts” gives a measure. “They lose the thread as soon as a concept appears before a concrete example” gives a direction. The next action follows from both, and names the precise case that has to open the book.

It reconstructed a line of reasoning, not just a position in a list.

It does not, for all that, try to recreate every detail of the previous session in my head. Too short a summary wipes out the reason for a choice. Too long a report pushes back the moment when the work starts again.

### **An Interruption Does Not Pick Its Moment**

At first I imagined that picking the work back up depended on ending a session cleanly. A session would end, its state would be carefully summed up and the next action noted.

Practice rarely follows that script.

- A meeting starts.
- The battery runs down.
- The day ends sooner than planned.
- Another urgent matter takes over.

The moment when that memory is most useful is exactly the one where you did not prepare the ending.

So the `Context` stays ready to be picked up again throughout the work. Decisions find their place when they are taken, tasks when they are stated, sources when they come in.

The ability to resume becomes a property of the system rather than a feat of discipline on the user's part.

The human still has a decisive role. They can judge that the next action proposed is no longer the priority, that an old decision has to be reopened, or that a new fact changes the goal. Memory does not lock the work onto a set course. It makes the current direction visible, so that you can decide whether to change it.

The five sentences I used to retype when I reopened my conversation, I no longer retype. They are written down somewhere, and someone other than me reads them back at the start of each session.

---

#### **WORTH REMEMBERING — CHAT, ENVIRONMENT AND RESUMING**

Chat is the place of exchange; the `Context` is the place of what must last. A conversation preserves information in the order it was said; work needs a different order. Moving forward means knowing the next action; getting back up to speed means recovering the logic of the work before taking it. Interruptions give no warning: the ability to resume is maintained throughout the work, not when the session ends.

---

## Two Clients, a White Paper and a Contradiction Worth Keeping

---

So far I have described the `Context` and several effects it produces. But what does it actually look like when you open it? What do you see in the files? How do several sources give rise to ideas that can each be right on their own and still be hard to reconcile?

The subject of this invented `Context` is simple: understanding why some technicians produce incomplete job reports, before deciding whether the product should change.

Its mission fits in a few lines:

*Understand what degrades the quality of job reports. Tell apart the difficulties of filling them in, the constraints of the organisation and the needs that come from using the data. Do not pick a solution before several sources have been set against each other.*

That mission gives the work a direction. It asks for a problem to be understood, not yet for a feature to be designed.

### A First Account Reveals an Idea

The first document put into the `Context` is the transcript of an interview with an operations manager.

She explains:

*“Technicians often fill their reports in on a Friday. Out on site, the form strikes them as too long. By the end of the week they have forgotten some of the detail.”*

The AI keeps that interview as **a source**. It then extracts an idea into a separate card, which is what I call an **atomic note**:

***Filling in a report that asks too much can lead some technicians to put it off and lose detail.***

That wording does not say every technician behaves this way, nor that the length of the form is the only cause. It preserves the central point of the account: in the setting described, the report is postponed and some information is lost between the job itself and writing it up.

The source stays available. The note does not replace it; it makes the main idea easier to find and to set against what comes next.

### **The Second Account Enriches the Same Note**

A few weeks later a second interview comes into the Context.

This time an operations manager says:

*“We made twelve fields mandatory to improve data quality. Now some technicians write ‘nothing to report’ or put a dash just so they can submit.”*

That account does not describe quite the same behaviour. The technicians in the first interview put the writing-up off. Those in the second fill the form in, but get round what it asks.

Both sources nevertheless bear on the same central idea: a constraint meant to obtain more information can end up reducing the real quality of what gets entered.

So the second interview joins the first **atomic note**. It does not replace the first source and does not create a rival idea. It enriches the note by bringing out another form of the same problem: after the deferred write-up, the appearance of completeness.

The note can now state:

*Filling in a report that asks too much can lead some technicians to put it off or get round it, at the expense of the information gathered.*

Two sources are referenced. Each keeps its own part and its own limits. The note grows sturdier without pretending that two accounts are enough to measure how widespread this is.

### The White Paper Reveals Another Idea

Next I add a white paper on making use of job data.

The document argues for a different requirement. To understand recurring faults, compare equipment and prepare certain audits, a company needs reports that are structured enough. If every technician writes what they like in a free-text box — or fills in nothing — entering the report may get easier, but the data loses much of its collective usefulness.

That **source** does not make the earlier accounts any less true. The technicians described go on deferring or getting round

the task. The PDF reveals another reality: removing every constraint would also have a cost.

The AI might be tempted to merge these into the first note and rewrite it as a general compromise. That would be a mistake. The white paper does not weaken or qualify what the interviews observed; it makes a different claim.

So it produces a second **atomic note**:

***Structured reports are needed in order to compare jobs and make use of job data.***

The Context now holds two distinct ideas:

- the constraint of filling in a report can degrade the information actually gathered;
- the lack of structure can degrade the later use of that information.

Both notes can be true at once. They nevertheless pull the product decision in opposite directions.

### **The Opposition Has to Stay Visible Between the Notes**

The Context's job is not to alter the first note until the contradiction disappears. It has to keep both propositions and make their relation explicit.

A **thematic note** can then state what bringing them together lets you see:

***The quality of a report depends on a tension between the effort asked for when filling it in and the structure needed for using it.***

That third note is not a diplomatic average. It replaces neither of the two atomic notes. It shows the new problem that appears when you read them together.

The contradiction also becomes part of the state of the work: how can we obtain structured information without causing people to postpone the report or work around its requirements? None of the three sources lets you answer yet.

### **Here Is What You See on Screen**

The `Context` used for this example exists as a small folder of files. Its structure is not the concept itself: it is my current way of giving the concept a concrete form.

```
contexte-qualite-comptes-rendus/
├── 00-frame/
│   ├── mission.md
│   └── directives.md
├── 01-state/
│   ├── state.md
│   └── contradictions.md
├── 03-sources/
│   ├── SRC00001-entretien-exploitation.md
│   ├── SRC00002-entretien-exploitation.md
│   └── SRC00003-livre-blanc-donnees-intervention.md
├── 04-notes/
│   ├── atomic/
│   │   ├── AT000001-la-contrainte-de-saisie.md
│   │   └── AT000002-des-comptes-rendus-structures.md
│   └── thematic/
│       └── THM00001-la-qualite-depend-de-la-saisie.md
└── 05-outputs/
    └── questions-prochain-entretien.md
```

You can now attach the words used since the start of the book to objects you can see.

- The **mission** explains why the folder exists.
- The **sources** keep what came in before it was interpreted.
- The **atomic notes** hold distinct ideas steady.
- The **thematic note** makes their tension intelligible.
- The **state** says what that tension leaves open.
- The **deliverable** prepares the next action.

None of this calls for any mysterious software. VS Code (though it could be the Codex app or Claude Cowork) is simply the window through which I reach the Context. The Context itself lives in the files, in what each is for, and in the relations kept between them.

## The Feedback Becomes Grounded in the Work

A few days later I come back to the work and put this to the AI:

*“Maybe we should make every field optional.”*

In an isolated conversation, the AI might argue for that idea, propose a lighter interface or list its advantages. In the Context, it has access to the two atomic notes and the tension between them.

It can then answer:

*That proposal would probably reduce some of the friction described in the interviews. It does, however, run against the need for structured data raised in the white paper. The sources show two real constraints; they do not yet let you choose between simplification, filling in stages, voice dictation or fitting the fields to the type of job.*

That feedback does not come from general knowledge of product management. It is grounded in the work under way. The AI knows which question we are trying to understand, which ideas are backed by which sources, and which contradiction is still open.

It does not decide for me. It simply prevents my new hunch from erasing one of the realities already established.

## The First Deliverable Prepares the Next Source

**The state** of the Context does not say: “design a new form”. It separates what looks established from what is still uncertain.

**What looks established** *Constraints on filling in a report can cause deferral and evasion. Enough structure is still needed in order to make use of the reports.*

**What is still open** *We do not yet know which friction weighs most, or which information absolutely has to stay structured.*

**Next action** *Prepare an interview that distinguishes between the causes of the problem instead of presenting a solution straight away.*

So the first deliverable is a modest one: a set of questions for the new interviews. It has to help us understand when the report is completed, what constraints technicians face on site, how useful each field really is and how people work around the requirements.

The deliverable does not settle the contradiction. It prepares the evidence that may allow us to resolve it later.

### **A Working Object, Not a Machine for Merging Everything**

This example also shows a mistake the AI can make: wanting to fit a new source into an existing note because the subjects look alike.

An **atomic note** can grow when several sources bear on the same proposition. That is what happens with the two interviews. But the white paper is not fresh evidence for that proposition. It supports a different idea, which deserves a note of its own.

---

The distinction matters. If every opposing view rewrites the previous note, *the body of ideas* ends up smoothing the tensions away. The knowledge looks coherent, but it no longer matches what the sources actually brought.

The `Context` does not guarantee that the AI will always pick the right boundary. The human has to be able to refuse a merge, ask for a new note and check the relation created between them. The structure makes that correction possible and lasting; it does not make the decision automatically.

You have now seen the object:

- **sources** that stay intact,
- **atomic notes** that each keep their own centre,
- a **thematic note** that brings them together without merging them,
- a **contradiction** that stays open and a **deliverable** that prepares what comes next.

The following chapters can now return to how the `Context` is used, then examine its architecture more closely through the elements we have just seen.

---

#### WORTH REMEMBERING — A CONTRADICTION IS KEPT

Two sources can both be true and pull the decision in opposite directions. An atomic note grows when a source strengthens the same proposition; a different proposition deserves its own note. Merging to make the tension disappear produces coherent knowledge that no longer matches the sources.

---



# Part 3

## Three Uses in Practice

---

The two previous parts said what the Context changes. This part shows it at work, on three uses I actually practise.

Answering a client questionnaire, writing an article from a thought still taking shape, letting a product subject ripen in the middle of a fragmented day.

Three timescales, three kinds of material, three audiences — and the same movement every time: recovering what has already been understood, transforming it, then producing without using up what served.

## Answering a Client Questionnaire Without Losing the Work Done

---

The first client questionnaire I handled with this system had forty-three questions.

We had to answer on the architecture of the product, its security, backups, service continuity and data protection. Some of the information was in our documentation. Other parts were buried in configuration files. For some of it we had to go to the team running the service.

We had to pull together seven technical sources and go back and forth with the team. That came on top of the writing itself: understanding what the prospect was asking, finding the information, checking it was still true, deciding what we could disclose, then putting the answer back into the format expected.

This was not an exceptional situation. Every new prospect sent its own document. One time it was a spreadsheet; another time a Word file or a PDF. The reference systems changed, the wording too, but the same subjects came round again.

And yet each time we felt we were starting over. Answering took between three and six hours and tied up several people, with no guarantee the prospect would become a client. Then

the document went off and the work stayed shut inside the file we had sent.

So my first aim was simple: when the next questionnaire arrived, I did not want to start from nothing again.

### **A Plausible Answer That Was Wrong**

The obvious solution was to keep the old questionnaires and reuse their answers.

We soon found out why that was not enough.

Some of the documents we received already had answers in them. They might come from a template, from another supplier, or from an earlier exchange. In one of them, an answer said the ISO 9001 certification dated from 2017. The sentence looked perfectly usable. It was precise, already written, and sitting opposite the right question.

The correct date was 2018.

The error came down to one year, but it exposed a much bigger problem. An old answer looks like knowledge. It has an assertive, reassuring shape. When an AI meets it in a document, it can pick it up without difficulty. Yet its being there says nothing about who wrote it, what it rests on, or whether it actually describes our product.

That was when I understood that I should not build a library of answers to copy. What had to be built was a memory of what we genuinely knew how to answer.

The questionnaire we received kept its value: it showed what the client wanted to know. But on its own it could not prove

the answer. That distinction became a rule of the `Context`: the client's document holds the question; the answer has to stay attached to information we have checked.

### **Keeping the Facts Rather Than the Forms**

We started pulling out of the questionnaires whatever could serve the next one.

The questions were no longer filed only by client or by number.

We grouped them by subject: backups, authentication, encryption, subcontractors, hosting, service continuity. A question labelled "14.6.l" at one prospect could thus join a question labelled "CL-072" at another if both bore on the same subject.

For each of those subjects the `Context` kept the validated facts and where they came from. An answer could rest on official documentation, a checked configuration, a test, or confirmation from the team responsible. We also kept the date of validation and the limits of what we were allowed to state. That work did not always produce a complete answer. It also brought the gaps out.

When information was missing, the system had to say so. It prepared a precise question for the team concerned and indicated which lines of the questionnaire that answer would complete. Missing information thus stopped hiding behind a plausible sentence. It became an identifiable action.

Finally, we separated what served to verify an answer from what could be sent to the client. The detail of a configuration

could be essential to the internal reviewer without belonging in the public document. So the Context prepared the wording meant for the prospect while keeping, alongside it, the evidence and the reservations needed to approve it.

Little by little the aim changed. It was no longer only about finishing the questionnaire in front of us. **Every checked answer had to reduce the work needed for the next ones.**

### The Fourth Questionnaire

The fourth questionnaire had a hundred and fifteen questions, far more than the first.

This time we were not starting from a scatter of files. The Context already knew the recurring subjects and where to find the facts that went with them. Ninety-six complete answers could be produced straight from the memory built up and from the product's own sources, with no new gathering.

The system also sorted the rest of the document. Some answers were solid but needed one last check. Others remained uncertain. A few could not be given with the information available.

Above all, the gaps had changed in kind. We no longer spent hours casting about for who knew what. The points still open matched commercial decisions — the length of a commitment, say — or one-off technical confirmations. The work remaining was visible, assignable and bounded.

That result let me see what answering a questionnaire “automatically” really means. It is not about asking an AI to

fill every box with confidence. It is about preparing at once whatever can be prepared, showing how much each answer can be trusted, and setting clearly apart what still calls for human judgement.

### **Ten Minutes Does Not Mean a Hundred Per Cent Certain**

In my own use, preparing a client questionnaire went from three to six hours down to about ten minutes to produce 80 to 90 % of the document.

3

That figure could give a false impression. It does not mean 80 to 90 % of the answers become instantly certain and ready to commit the company. It means the document is largely prepared: known answers are in place, sources are found, weak points are flagged, and missing information is turned into precise requests.

Human review is still essential. It does, however, land in the right places. Instead of spending time digging through old files and going back to the same people again, the reviewer can check the commitments, decide what will be disclosed, and concentrate on the new or uncertain answers.

So the biggest gain is not only going from several hours to a few minutes. It is the change in the nature of the work.

The first questionnaire had called for an investigation. The ones that followed benefited from everything that investigation had taught us. A wrong date became a rule of caution. A confirmation obtained from a team became a

---

reusable fact. An unanswered question became a known gap rather than an oversight.

The prospect still gets its document.

The company, for its part, no longer loses what it learnt while preparing it.

---

**WORTH REMEMBERING — AN ANSWER IS WORTH ITS EVIDENCE**

An old answer looks like knowledge: assertive, well formed, and silent about where it came from and when. Keep the checked facts and where they came from, rather than the forms already filled in. An explicit gap beats a plausible sentence: it becomes a question addressed to someone.

---

## Writing an Article from a Thought Still Taking Shape

---

The articles I publish on my blog rarely start at my computer, in front of a carefully prepared blank page.

An idea turns up after a difficulty I have run into, a tool I have tried, a conversation or a change of mind. Sometimes it arrives while walking, on public transport or on my sofa. I can feel it deserves a text, but I do not yet know its exact angle or the order in which it will have to be told.

At that point I often take out my phone.

I do not use it simply to record a voice note or dictate a text I will copy out later. **I start a conversation with the AI.** It listens, answers and pushes me on while I think. That conversation goes into the same working environment I will find again on my computer. So I can start far from my desk without creating an isolated note that will then have to be moved and reinterpreted.

### Unloading a Thought as It Forms

I create a `Context` with one simple goal: write the article. Then I give it everything I have in mind.

For as much as two hours sometimes, I talk about one point, then another, then come back to the first. I tell a story, jump to a conclusion, add a detail, open a new line and drop the

previous one for a while. I can repeat the same idea four times because it comes back to me in different shapes. I can also assert one thing and say the opposite an hour later because my thinking has moved during the conversation. That disorder is not an accident to be cleared away before starting. It is often the real shape of an idea while it is still being born.

Talking with the AI lets me carry on without imposing a structure too early. An answer makes me pin down a memory. A question shows that I have just skipped a step. A rephrasing shows me that what I thought obvious is not obvious yet. I correct, I qualify, and I set off in another direction.

The Context keeps what I say as a session source. It does not hold on only to the tidiest sentences. It also keeps the retreads, the hesitations and the contradictions that will let me see how the thinking moved.

The AI then brings the four wordings of one idea together instead of treating them as four different ideas. It flags two incompatible claims, finds the qualification that arrived an hour later, and puts an example next to the principle it illuminates. **It does not make the disorder disappear: it makes it workable.**

That first step frees me from choosing too soon. I do not yet have to decide which sentence will open the article, whether a given idea deserves a section, or in what order to tell the events. What I am after first is emptying out what I have in

mind, without letting go of the ideas that are only passing through it.

The Context, though, knows why that material is there. It is not receiving a conversation with nowhere to go. It is receiving the still-disordered material of an article to be built.

### **Using the AI as a Partner in Ripening**

Once that first material is down, I ask the AI what is missing.

It picks out a claim made too quickly, an assumption never explained, a passage that only holds up if the reader already knows my experience. It asks for an example. It brings together two ideas stated at different moments. Sometimes it offers a line that looks plausible and that I turn down because it does not match what I lived.

I answer, I sharpen, I correct. That conversation does not yet produce the article. It thickens its material.

The AI's part matters, but its place stays clear. It has everything I have poured in and looks for tensions in it faster than I do. What it does not have is any idea what I actually want to argue until I have said it. It opens the question; I decide whether the question is worth moving the text for.

That exchange shortens the time it takes to meet a weakness. On my own I would have found out two days later that an example was missing or that my conclusion rested on a step never explained. The AI brings that discovery about while the idea is still easy to rework.

### Adding Sources When Experience Is No Longer Enough

My articles often start from what I have lived through. Outside material still has its part:

- a reference gives a name to what I observed;
- documentation confirms how something works;
- another author argues a neighbouring position, or shows a limit I had not met;
- further reading reveals that my experience covers only one particular case.

Those documents come into the `Context` as **document sources**. Their part differs from that of my own first words. The **session source** keeps what I assert and what I understand of my experience. The **document sources** bring material I can set against that experience.

The article can then tell apart what I observed, what a source establishes and what remains an interpretation. That distinction protects the text from a common temptation: presenting personal experience as a general rule.

### Turning Sources into Reusable Ideas

At this stage the `Context` does not go straight from the sources to the outline of the article. It first turns the raw material into **atomic notes**.

Each note holds steady one self-standing idea from the conversation or from a document:

- a claim I want to argue;
- an example that gives it concrete shape;

- a qualification that limits its reach;
- an objection the article will have to answer;
- an outside fact that confirms or contradicts my experience.

The note keeps where it came from. I can thus understand the idea without rereading the whole conversation, then go back to the source if I have to check a wording or recover its setting.

That transformation changes how the article gets prepared.

- The conversation followed the movement of my thinking; the notes now group what speaks about the same idea, even when those pieces appeared an hour apart or in two different sources.
- One central note can bring out the angle of the article.
- Others bring it examples, evidence, objections or consequences.

Those notes are neither paragraphs yet nor a ready-made outline. **They stay neutral towards the deliverable: the same idea will be able to feed this article, a presentation or some other piece of work.**

### Going Through the Outline Before Writing

When the material becomes sufficient, I ask for an outline.

That step stops me going straight from a rich conversation to a text that would simply follow its order. A discussion moves by returns, corrections and associations. An article has to lead a reader who knows neither the discussion nor its history.

The outline fixes that progression. It says what has to be understood first, where to put the main example, which objection to take on and what the conclusion can actually claim.

The AI proposes a structure from the material available. I move it about, tighten it, turn down the parts that give too much weight to a secondary line. Two or three exchanges are often enough to get a clear direction.

The writing comes next.

At this stage the AI is not creating the text from a vague instruction. It is working with a body of ideas and themes drawn from my recounted experience, with answers to questions, with sources and with an approved outline. It groups the repetitions, puts the examples back in place and turns winding speech into readable prose.

I read the result over, looking for three precise gaps:

- a qualification that has gone;
- a passage that has become too assertive;
- a wording that betrays the idea.

I correct them, then I settle the published version.

### **What the Context Brings to a Short Piece of Work**

An article can be written in a day or two. It does not look as though it needs the same memory as a product followed for years or a book of dozens of chapters.

It is precisely that light case that shows duration is not the only thing at stake.

The `Context` holds together the first words, the sources added, the questions asked, the outline chosen and the complete text. If the work breaks off, I find its state again. If a sentence becomes questionable, I can go back to where it came from. If an idea that appeared while writing goes beyond the article, it can be held steady separately and feed future work.

3

Not all that material necessarily deserves to be kept indefinitely. A small `Context` can reach its goal, produce its article, and then be archived. Its value lies in the continuity it gave the work while the work was forming, and in the trace it leaves if you later have to understand how the text took that direction.

Publication itself can set off other tasks: preparing a summary, adapting the text to several languages, or producing what is needed to put it online. Those operations extend the same result, under rules the `Context` already knows.

The whole movement stays simple: an experience becomes speech, that speech meets questions and sources, the material becomes an outline, and then the outline becomes a text.

The AI speeds up each transformation, and the `Context` holds the thread that leads from one to the next.

---

None of that movement belongs to long writing alone. It starts as soon as an idea deserves better than a voice note forgotten in a phone.

---

#### **WORTH REMEMBERING — DISORDER IS MATERIAL**

A thought being born repeats itself, contradicts itself and doubles back: that is its normal shape, not a fault to correct before starting. The `Context` keeps that speech as a session source, then makes it workable. The outline sits between the conversation and the text, because a reader does not follow the order of a discussion.

---

## Ripening a Product Subject in the Middle of the Chaos

---

The two previous chapters each followed a single subject from start to finish. This chapter describes what a typical product manager's day is really like: several subjects, requests and timescales moving at once.

I am the only product manager in my company. That means most of the information tied to the product ends up passing in front of me.

In the same day, a client says something that strikes me as important.

- It comes down to a few words.
- I do not yet know whether it reveals a problem shared by other users or just one particular difficulty, but I do not want to lose it.

A little later I pick up a white paper on how a business concern is changing.

- This time the material no longer fits in a sentence, but in a PDF dozens of pages long.
- I do not yet know what will be useful to me, or whether the document will confirm what I see in the field.

Then the developers talk to me about a feature being built.

- A question has to be answered, a behaviour pinned down, or an assumption taken a few weeks earlier revisited.

Meanwhile the CPO wants to review two bets tied to another part of the product.

A conversation with support finally reveals that a piece of functional documentation is missing.

- Part of that documentation still exists only in my head.

By the end of the day I have moved across three or four different subjects (or more). I have taken in a few words of quotation, a whole document, a question about work in progress, a call to arbitrate, and a need for documentation.

**This is not a particularly bad day: it is an ordinary one.**

### **Inputs That Share Neither Size nor Destination**

The chaos of product management does not come only from the quantity of information. It comes from how different in kind it is.

Each of those five inputs looks at a different moment in the life of the product:

- a client's sentence is a weak signal, and I do not yet know whether it heralds others;
- a PDF gives a general view of a trade;
- a discussion with the developers bears on what we are building now;
- a request from support bears on what already exists and has to be explained;

- a question from the CPO forces several possible futures to be compared.

Those items are not meant to go into a single document.

- They do not all speak about the same subject and they do not sit at the same moment in the life of the product.

If I simply gather them in a list of notes, I keep the information but put off the work of understanding until the day I need it.

3

Contexts let me keep several threads going at once.

- The quotation joins the subject it seems to belong to, even if what it means is still uncertain.
- The PDF is kept with where it came from and can be analysed to draw out the few genuinely useful ideas (**atomic notes**).
- The conversation with the developers joins the work on the feature being built.
- The gap flagged by support becomes work to produce instead of a remark overheard between two meetings.

Not everything is turned into knowledge straight away:

- the client's sentence may stay ambiguous,
- the white paper barely relevant,
- the hunch contradicted the following week.

The Context does not make what comes in any cleverer. It gives it a place, an origin and a chance of being found again at the right moment.

**The Context holds the chaos without claiming to abolish it.**

### **While Information Comes In, the Requests for Output Keep Coming**

The day is not just about taking material in and filing it: things have to be produced too.

Support asks me, for instance, for functional documentation on a feature we are still working on.

- It has to explain how it works, pin down the important cases, and give the people dealing with clients what they need to answer properly.

The same day, I may want to finish an article for the company blog.

- That article does not necessarily deal with the same feature.
- It may, though, bear on the same business subject: the trade the product serves, a recurring difficulty its users run into, or a change in their practices.

The two documents then draw on part of the same knowledge without telling the same story.

- The documentation for support explains the product as it works.
  - It has to be precise, operational and fitted to users' questions.
- The article develops an experience, a business problem or a line of thought.
  - It may call on some of the same ideas, but it organises them for another reader and to another end.

I do not have to rebuild twice everything I know about the subject.

- The `Context` keeps intermediate material (the **atomic notes** and **thematic notes**) that is neither the documentation nor the article yet.
- From that material, the AI can prepare several assemblies: one centred on using the product, the other on the wider business concern.

3

I obviously read both results over.

- Documentation for support commits how the product is understood.
- An article commits my point of view.

But I spend my time checking the substance and the direction, rather than finding the same elements again and writing them out from scratch.

That movement can happen within a single day. While new information is coming in, knowledge already organised makes it possible to produce the documents the company or I need.

### **A Small Sentence Can Become a Product Subject**

The quotation heard in the morning does not necessarily set off a decision.

- Taken alone, it may prove nothing.
- Another client may express a similar difficulty a few weeks later.
- An interview will bring a qualification.

- The white paper picked up that day may give a name to the change observed.
- A figure from support will show that the problem comes back more often than we thought.

Those pieces arrive at different moments and in different shapes.

- The `Context` makes it possible to bring them together without erasing where they came from.
- It can point out
  - that a new remark resembles a signal already kept,
  - bring a contradiction to light,
  - ask what would let you choose between two readings.

Little by little, an isolated sentence can become a subject worth working on. By then I already have part of its history: the first signals, the documents that light it up, what confirms them, what contradicts them and the questions still open.

The `Context` helps that material ripen.

- It does not decide that the problem matters and does not choose the solution for me.

The product bet stays my responsibility.

- What I do get is the chance to take that decision with something better than a vague memory or a scatter of documents.

The same system then goes along with the work:

- adding interviews,

- setting a hypothesis against the facts,
- preparing scenarios,
- producing a case for the CPO,
- writing specifications,
- ...

A small signal caught in the middle of a day can thus go on living long enough to become a real decision.

### **An Abandoned Idea Can Go On Ripening**

Not every exploration makes it into production.

I have worked on uses of artificial intelligence that looked interesting but that the technology available could not yet deliver properly.

- The idea was not necessarily a bad one.
- The moment was.

In the usual way of working, the exploration would have produced a few documents and then disappeared into a folder once it was dropped.

- A year or two later, you would have had to find those files, remember the starting assumptions and work out why we stopped.

With a `Context`, the exploration can be archived along with how it went: the initial signals, the scenarios imagined, the attempts, the limits met and the reason for stopping.

- When the technology moves on, I can pick that work up again without turning an old negative result into a permanently bad idea.

I find again what we wanted to do, but also why it did not work. I can then check whether the obstacle is still there, keep what still holds and take the thinking up from that point.

So ripening a product subject is not always a continuous journey.

- It can start with a sentence, grow over several months, break off, then start again once the product, the market or the technology has changed.

### **Letting Order Emerge Without Slowing the Day Down**

My day stays fragmented.

- Clients do not coordinate their remarks with requests from support.
- Developers do not pick the moment when I have just finished reading a PDF.
- The decisions to prepare do not stop while I write documentation or an article.

The Context does not turn that reality into a perfect run of tasks.

It lets each fragment keep a continuity.

- A small sentence and a long document can come in without getting the same treatment.
- Several subjects can move forward without running into each other.

- Knowledge already worked on can serve several deliverables without being locked into the first.
- An interrupted idea can wait without losing its whole history.

That is how I use it in product management: not to make my day less unsettled, but to stop its movement scattering what I learn.

The chaos keeps coming in.

It no longer leaves as loss.

3

These three uses have one thing in common that I have not yet named: each ends with something that departs — an answer sent, an article published, a decision taken.

The next part looks at what is left once the deliverable has gone.

---

#### **WORTH REMEMBERING — SEVERAL THREADS AT ONCE**

A client's sentence and a fifty-page PDF call for different treatment and a different destination. The `Context` gives each fragment a place, an origin and a chance of being found again at the right moment. An abandoned exploration keeps its reasons for stopping: it becomes workable again when the technology or the market changes.

---

# Part 4

## What Deliverables Leave Behind

---

Each of the three previous uses ends with a deliverable handed over. This part looks at what is left once it has gone.

One question, asked twice six months apart, runs through it. The first time, the answer fits in a spreadsheet cell. The second, that answer still exists but is no longer enough: what is missing is the reasoning that produced it, and the relations between the documents that backed it.

#### 4.1

## The Answer Fits in a Cell, the Reasoning Does Not

---

Let us go back to one of the questionnaires from the previous part.

The question arrives in cell D47 of a spreadsheet:

| *Are your backups encrypted at rest?*

The final answer will run to three lines. To write them, though, I have to bring several things together:

- our documentation describes how backups work in general;
- a configuration file shows the setting actually in force;
- the Ops team confirms how that setting applies in production;
- legal states what we are allowed to disclose to a prospect.

I go from the spreadsheet to the documentation, from the documentation to the configuration, then from a discussion with the Ops team to legal approval. Finally I come back to cell D47 and write the answer there.

The document is sent.

### Three Lines at the End of a Chain

**The cell holds the result of the work. It does not show the path that produced it.**

It does not say:

- which documentation I consulted;
- which setting I checked;
- who confirmed it applied;
- which wording was authorised;
- which reservations were kept out of the text sent to the prospect.

Those things exist, all the same. They determined each of the three lines, and while I am answering, how they relate is clear in my head.

Once the document is sent, that unity disappears. The spreadsheet keeps the answer, but not the reasoning that made it trustworthy.

### **Every Tool Did Its Job**

The problem does not come from poor filing.

- The documentation is right there in the documentation space.
- The configuration stays in the product.
- The team's reply remains in its thread.
- The legal approval is recorded in its own system.
- Cell D47 is in its place in the prospect's document.

Every tool kept the object it was given. None of them was there to keep the whole reasoning.

To rebuild it, you have to know which elements to look for, where to look for them and what part each one played. That knowledge stays with the person who prepared the answer.

## The Deliverable Flattens What Produced It

A deliverable has to present a result fitted to whoever receives it.

- The prospect expects a clear answer, not a full account of the investigation that produced it.

That simplification is necessary. It becomes costly when the deliverable is the only trace of the work left.

The final result is always shorter than the path that leads to it — the answer sent, the article published, the product decision alike. If only the result remains, the next piece of work has to redo part of that path.

4

## Giving the Reasoning an Address

The `Context` replaces neither the spreadsheet, nor the documentation, nor the configuration, nor the exchanges with the teams.

- It gives a shared address to the work that ties them together.

Question D47, the elements consulted, the approvals obtained and the answer produced all belong to the same subject. They can keep their original form and location while staying linked inside the `Context`.

The cell goes on doing its job: answering the prospect. The `Context` does another: keeping what will make it possible to understand, to check and to pick that answer up again.

Six months later, that difference becomes decisive.

---

**WORTH REMEMBERING — THE DELIVERABLE DOES NOT KEEP THE REASONING**

An answer can run to three lines and still depend on several sources, approvals and reservations. The tools each keep their own object; the human is still the one carrying the relations between them. The `Context` gives an address to the work that ties those elements together, without replacing the tools they came from.

---

## Six Months On, the Answer Is No Longer Enough

---

Six months after cell D47, a new prospect asks the same question about backup encryption.

- I find the old answer in ten seconds.
- It is clear, well written and ready to be copied across.

And yet I sit in front of it unable to use it.

Three questions are enough to block me:

- where did that answer come from: the documentation, the setting observed, or what the Ops team explained to me that day?
- did legal approve that exact wording or a more cautious version?
- is the setting still the same today?

**The answer survived; what made it trustworthy has gone.**

### Finding Is Not Yet Picking Up

The old cell hands me a text. It does not hand me:

- its level of evidence,
- its limits,
- or the checks needed before using it again.

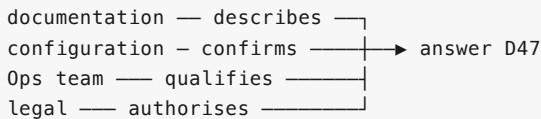
With no other memory, I have to redo the whole investigation of the previous chapter, in reverse. I am not quite starting from nothing, since I already have an answer. But I am

starting again without knowing which part of the work still holds.

It is the mistake of the certification dated 2017 instead of 2018, seen this time from the side of our own answers: the sentence is crisp, it comes from us, and its evidence has become invisible.

### Finding the Part Each Element Played

In the Context, the old answer does not sit alone. It keeps the relations that the reasoning behind it needed:



4

The verbs change what I can do with each element.

- The documentation describes how things work in general; on its own it does not prove the setting in force.
- The configuration confirms that setting at a given moment; it does not decide what may be disclosed.
- The Ops team brings a qualification that comes from running the thing.
- Legal authorises a wording meant for the prospect.

What I find is no longer just four objects sitting near an answer. I find the part each of them played.

That precision stops a source proving more than it actually establishes. It also shows which check has to be redone once time has passed.

## Checking What May Have Changed

The `Context` does not turn the old answer into permanent truth: it prepares the check.

- The documentation may still hold while the configuration has changed.
- The configuration may be unchanged while the authorised wording has to be revisited.
- A qualification from the Ops team may have gone after the product moved on.

The relations make those differences visible. I can go over the elements that may have changed without redoing the whole investigation at random.

The final answer may well be identical to the one from six months ago. The difference is that I know why I can still use it.

## Keeping the Relations That Serve the Work

The point is not to link every document to every other.

- Two files can share a word without taking part in the same reasoning.

A relation deserves to be kept when it changes how you understand, check, decide or produce.

- In the case of D47, four relations are enough: describe, confirm, qualify and authorise.

They make it possible to answer concrete questions:

- what does this answer rest on?
- which part has to be checked today?
- who had approved what we could disclose?

- 
- what would make this answer void?

So the `Context` is not trying to build a total map of everything I know. It keeps the network of relationships that a given piece of work needs.

### Opening the Machine

Book I has shown how the `Context` came about, what it changes in dealings with an AI and how it goes along with several uses. The D47 case brings out what is left to understand.

- How do you keep the mission of a piece of work without repeating it every session?
- Where do you put the sources, the ideas, the decisions and the deliverables so that they keep distinct parts?
- How do you load what is useful now without bringing all the material into the conversation?

Book II will open that machine. It will follow information all the way through a `Context`: from its arrival to its transformation, what it produces and how the work is picked up again.

---

### WORTH REMEMBERING — FINDING WHAT MAKES AN ANSWER TRUSTWORTHY

Finding an answer is not enough: you have to find what still makes it usable. A named relation states the precise part played by a source, an approval or a reservation. The `Context` prepares the check by showing what holds the answer up and what may have changed.

---



## BOOK II

# Building and Sustaining the Context

---

- 5 Under the Bonnet of a Context
- 6 The Context in Its Ecosystem



# Part 5

## Under the Bonnet of a Context

---

So far the `Context` has been described by what it makes possible. This part opens it up.

It sets out the full tree first, then follows a working session from start to finish: opening up and finding where things stand, letting material in, turning it into ideas, producing a deliverable, settling the trade-offs, then saving so that the next session starts again without rebuilding anything.

The example running through these six chapters is entirely made up: understanding why some field technicians produce incomplete job reports, before deciding whether the product should change.

## 5.1

# The Full Map

In part 2 I left behind the Context devoted to the quality of job reports:

- three sources,
- two atomic notes,
- one thematic note,
- a guide for the next interview.

What I was looking at then was what it allowed me to understand. Now I open the folder itself.

Here is the full tree of the version I use in this book:

5

```
contexte-qualite-comptes-rendus/
├─ README.md          general navigation, generated
├─ offline.md         the letterbox between sessions
├─
├─ 00-frame/          what sets a lasting direction
│   ├── README.md     navigation for the zone, generated
│   ├── mission.md    the aim, the question, the limits
│   ├── directives.md the rules specific to the mission
│   └─ loading.md     what will be read on opening
├─
├─ 01-state/          where the work stands now
│   ├── README.md     navigation for the zone, generated
│   ├── state.md      findings, uncertainties, debts
│   ├── synthesis.md  what has become lasting
│   ├── todo.md       the actions to be carried out
│   ├── decisions.md  the choices that shape the Context
│   ├── contradictions.md what is still in tension
│   ├── handoff.md    how to pick the work back up
│   ├── checkpoints.md the compact run of states
│   └─ health.md     the result of the last check
├─
├─ 02-history/        the factual, dated trace
│   └─ README.md     the role of this trace, generated
```

|                           |                                          |
|---------------------------|------------------------------------------|
| └─ YYYY-MM-DD.md          | the events of the day                    |
| └─ 03-sources/            | what has come in or been said            |
| └─ README.md              | the list of sources, generated           |
| └─ ingestion-log.md       | what was read, what it produced          |
| └─ documents/             |                                          |
| └─ README.md              | the list of documents, generated         |
| └─ raw/                   | the original files                       |
| └─ fulltext/              | their full text, ready to work on        |
| └─ SRC00001-....md        | identity, provenance, state              |
| └─ sessions/              |                                          |
| └─ README.md              | the list of sessions, generated          |
| └─ SESSION-....md         | the words spoken during the work         |
| └─ 04-notes/              | the knowledge drawn from the sources     |
| └─ README.md              | the map of the notes, generated          |
| └─ atomic/                | one idea per note – AT000001-....md      |
| └─ thematic/              | an angle that connects – THM00001-....md |
| └─ glossary/              | a term pinned down – GLO00001-....md     |
| └─ decision-records/      | a judgement call – DRE00001-....md       |
| └─ 05-outputs/            | what is produced for a use               |
| └─ README.md              | the list of results, generated           |
| └─ _draft/                | the deliverables in progress             |
| └─ questions-entretien.md | a guide for going further                |
| └─ note-de-cadrage.md     | a summary for deciding                   |
| └─ presentation-equipe.md | the same findings, recast                |

The numbers impose an order of reading, not a hierarchy of worth:

- the frame and the state steer the work;
- the history makes it possible to check how it unfolded;
- the sources hold on to the material, the notes turn it into knowledge;
- the outputs give it a shape for a recipient.

### Six Zones, Six Functions

00-frame/ answers the question “why am I here, and under what rules?”. This zone rarely changes, and for good reason:

altering the mission would transform the way I judge every source, every task and every result.

```

contexte-qualite-comptes-rendus/
├─ 00-frame/                what sets a lasting direction
│   └─ README.md           navigation for the zone, generated
│   └─ mission.md          the aim, the question, the limits
│   └─ directives.md       the rules specific to the mission
│   └─ loading.md          what will be read on opening

```

01-state/ answers “where have I got to now?”. This zone moves with the work, and lets me tell a finding from a hypothesis, a future action from a decision already taken, a passing difficulty from a contradiction worth keeping in sight.

```

contexte-qualite-comptes-rendus/
├─ 01-state/                where the work stands now
│   └─ README.md           navigation for the zone, generated
│   └─ state.md            findings, uncertainties, debts
│   └─ synthesis.md        what has become lasting
│   └─ todo.md             the actions to be carried out
│   └─ decisions.md        the choices that shape the Context
│   └─ contradictions.md   what is still in tension
│   └─ handoff.md          how to pick the work back up
│   └─ checkpoints.md      the compact run of states
│   └─ health.md           the result of the last check

```

02-history/ answers “what actually happened?”. This zone does not tell the story of the session nicely: it keeps the useful facts, a source added, a line of enquiry dropped, a decision taken, a note created. I rarely consult it, but it becomes decisive the day the present state strikes me as incoherent.

```

contexte-qualite-comptes-rendus/
├── 02-history/                the factual, dated trace
│   ├── README.md             the role of this trace, generated
│   └── YYYY-MM-DD.md         the events of the day

```

03-sources/ answers “what material have I been given?”. An outside document comes in frozen, my own words come in session after session. In both cases the source stays separate from whatever the Context will make of it.

```

contexte-qualite-comptes-rendus/
├── 03-sources/                what has come in or been said
│   ├── README.md             the list of sources, generated
│   ├── ingestion-log.md      what was read, what it produced
│   ├── documents/
│   │   ├── README.md         the list of documents, generated
│   │   ├── raw/              the original files
│   │   ├── fulltext/         their full text, ready to work on
│   │   └── SRC00001-....md    identity, provenance, state
│   └── sessions/
│       ├── README.md         the list of sessions, generated
│       └── SESSION-....md     the words spoken during the work

```

5

The card SRC00001-....md holds neither the document nor a summary of it. It answers everything I need to know about it:

- where it comes from and why I added it;
- where to find the original and its full text;
- whether the extraction worked;
- how much confidence to place in it;
- whether reading it has already produced notes.

So it makes the state of the source visible without touching the material itself.

04-notes/ answers “what have I learnt?”.

- An atomic note settles one idea.
- A thematic note connects several ideas.
- The glossary pins down a word.
- A decision record keeps the reasons behind a judgement call.

```

contexte-qualite-comptes-rendus/
├─ 04-notes/           the knowledge drawn from the sources
|   └─ README.md       the map of the notes, generated
|   └─ atomic/         one idea per note - AT000001-...md
|   └─ thematic/       an angle that connects - THM00001-...md
|   └─ glossary/       a term pinned down - GL000001-...md
|   └─ decision-records/ a judgement call - DRE00001-...md

```

05-outputs/ answers “what am I going to do with this knowledge?”. Drafts and results live there, in the shape their audience expects:

- an interview guide;
- an article;
- a specification;
- a presentation;
- a decision, among other possible shapes.

```

contexte-qualite-comptes-rendus/
├─ 05-outputs/         what is produced for a use
|   └─ README.md       the list of results, generated
|   └─ _draft/         the deliverables in progress
|   └─ questions-entretien.md a guide for going further
|   └─ note-de-cadrage.md  a summary for deciding
|   └─ presentation-equipe.md the same findings, recast

```

## The World of Ideas: Bricks for Several Buildings

The separation between 04-notes/ and 05-outputs/ carries one of the main strengths of the Context. The notes and the glossary form a *world of ideas*: a set of bricks of knowledge, reusable without limit, which are no longer raw material and not yet a result meant for someone.

```

03-sources/
  |
  ▼
04-notes/
├─ AT000001
├─ AT000002
└─ THM00001
    |
    ├──> questions-entretien.md
    ├──> note-de-cadrage.md
    └──> presentation-equipe.md
  
```

the world of ideas  
 one idea, pinned down  
 another idea, pinned down  
 one reading of how they relate

In my example, the friction of filling in the form and the need for structured data were pinned down before I knew what shape they would be presented in.

- the interview guide draws questions from them,
- the framing note draws out what is at stake,
- the presentation draws a story meant for the team.

These three deliverables do not own the knowledge they use.

- They select and recompose part of it for a particular need, audience and format.
- So a deliverable does not use up the notes it is made from: the bricks stay available for another building.

Without that intermediate world, every new result would force me to reopen the sources, make the connections again and rephrase the same findings.

**The Context does not only keep what has already been produced: it builds a lasting capacity to produce again.**

### Not Every File Changes in the Same Way

The tree only becomes truly legible once I know how each of its elements changes.

Some files are rewritten to show the best situation available now:

- `state.md`;
- `handoff.md`;
- the drafts in `05-outputs/_draft/`.

Others only grow longer, in order to keep the past:

- the session sources;
- the dated history;
- the synthesis;
- the series of checkpoints.

Documents received from outside stay frozen, and a note replaced by another remains readable. The `README.md` files, finally, are generated from what exists: correcting them by hand would create a second version of reality, which would end up diverging.

That variety explains why a single large file was no longer enough for me. The same text would have had to:

- show the present,
- keep the past,
- take in raw material,
- settle ideas,
- carry a finished result.

### **A Skeleton Rather Than a Recipe to Copy Out**

This skeleton is not documentation sitting next to the `Context`: it is a real folder, which I copy.

- Every file arrives with the sections it has to contain and a few pointers, so that two `Contexts` created months apart share the same landmarks.

The structure remains adaptable for all that. A `Context` can take on:

- a directive of its own,
- a deliverable specific to its trade,
- a source of a kind not seen before.

Above all, the common base keeps the three boundaries my work needs:

- the stable and the present,
- the raw and the interpreted,
- the knowledge and the shape given to it.

### **Speaking to the `Context`**

So far I have shown folders and files. You might conclude from this that I have to open each one, know where to write and

move every piece of information myself. That is not how I work.

I speak to the AI:

| *Open the `Context` devoted to the quality of the job reports.*

| *Add this document as a source.*

| *Turn this source into notes.*

| *Save the work.*

Behind each of these requests lies a procedure I wrote once and that the AI can pick up again every session.

- In Claude Code and Codex, that kind of reusable procedure has a name: a **skill**.

5

The opening skill states, for instance:

- what has to be taken in before starting;
- which files to read every time;
- what must stay on disk until the task calls for it;
- how to report a problem;
- what information to hand back to me so that I can act straight away.

I do not need to give those steps again each time.

- I simply ask for the `Context` to be opened.

The conversation carries what I intend right now.

- The skill carries the reliable way of performing the gestures that recur.

My system thus has a few stable verbs:

- create a `Context`;
- open it and find the work in progress;
- add a source or keep a session;
- turn the sources into notes;
- save the state of the work;
- check that it holds together;
- archive it once it has reached its goal.

Not every exchange I have with the AI goes through a skill. I go on thinking, questioning, correcting and deciding in ordinary language.

**Skills only frame the operations that call for the same steps every time.**

- They prevent, for example, a source being transformed without its origin being kept, or a session ending without the return being prepared.

The procedure stays stable, but judgement does not become automatic. Two models may word a note differently. They must nevertheless consult the same elements, respect the same boundaries and carry out the same checks.

Sharing skills therefore does not give someone my `Context`. It gives them the same way of creating their own and keeping it alive.

I shall say no more about how they work.

Skills are neither an invention nor a peculiarity of the `Context`. Claude Code and Codex offer them precisely in order to

---

describe reusable procedures, and their documentation explains better than I could here how to create and use them. What belongs to the `Context` is the choice of the abilities it is given: create, open, feed, transform, save, check and archive. It is also the order in which they follow one another and the boundaries each of them has to respect.

The files keep what the `Context` knows. The skills organise the gestures by which it changes.

The map is open. The visit can now begin the way a real session begins: by opening the `Context`.

---

#### **WORTH REMEMBERING — THE SKELETON OF A `CONTEXT`**

A new `Context` arrives already furnished: six zones, each answering a different question. The numbers impose an order of reading, not a hierarchy of worth. What makes the tree legible is not its shape, but knowing how each file changes. Claude Code and Codex supply the mechanism of skills; the `Context` defines the recurring gestures, their order and their limits.

---

## Opening: Recovering the Mission and the Present

---

Two weeks have gone by since my last session on job reports — the documents in which a technician records what they found and what they did after working on a piece of equipment.

- The interviews and the notes are all still there, in their places.

But if I reopen them at random, I shall spend half an hour piecing together what I already knew, and I shall piece it together badly.

- So the opening starts with a much smaller core: the frame, the state and the way back in.

I do not have to ask for all of that. I call the opening skill, and it runs through its procedure.

### First I Recover Why I Am Here

The first file the AI reads (through its context-opening skill) is `00-frame/mission.md`. Its template asks five questions:

```
# Mission
```

```
## The aim
```

```
What is this `Context` meant to make possible?
```

```
## The central question
```

```
Which question guides the sorting and the exploring?
```

```
## What is in scope
```

```
What material may come in?
```

```
## What is out of scope
```

```
What gets turned away, even when it looks interesting?
```

```
## The expected deliverable
```

```
What do we want to produce, if we know that already?
```

In mine, the beginning is filled in like this:

```
## The aim
```

```
Understand what degrades the quality of job reports.
```

```
## The central question
```

```
How can usable information be gathered without making it  
off-putting to fill in?
```

```
## What is out of scope
```

- Choosing an interface before setting several sources against one another.
- Generalising from a single interview.

```
## The expected deliverable
```

```
Prepare the product decisions and the next interviews.
```

A mission that only said “improve the job reports” would not steer me very far. The central question, on the other hand, already makes some documents more relevant than others,

and what is out of scope stops an appealing idea from quietly turning my exploration into a project to build a solution.

- The deliverable, for its part, may stay unknown.
- I write down that uncertainty rather than inventing an outcome too early: the `Context` then knows that it is still exploring.

### The Rules I Had Laid Down Are Waiting for Me

Two similar missions may call for different precautions.

`directives.md` keeps those rules with the work:

- confidentiality,
- level of evidence,
- vocabulary,
- approval or expected form
- ...

The template is deliberately simple: one rule, one date, one instruction in the imperative.

```
# Directives
```

- (2026-09-11) Anonymise every interview before it goes in: replace the name of the person interviewed with their role, but keep the name of the company and the country.
- (2026-09-11) Keep what the sources show apart from what we deduce from them.
- (2026-09-11) Put forward no solution before the real practices have been documented.

A directive is only worth having if it can guide a future action.

- “Be careful” leaves me all the interpreting to do again;

- “keep observation apart from deduction” gives me a precise check at the moment of writing a note or a deliverable.

But a directive is not only an instruction I come across when opening a file. It is a standing rule that the `Context` has to apply every time an action falls within its scope.

Say I add the transcript of a new interview. I simply ask the `Context` to take it in (through the skill for that), without thinking to point out that it contains the name of the person interviewed. The anonymising directive comes into play during the addition all the same: before saving the transcript as a source, the skill replaces that name with the person’s role, while keeping the name of the company and the country.

I do not have to remember the rule at the precise moment it becomes necessary. The `Context` remembers it for me.

So the directives do not stop acting once the opening is over. They stay in force throughout the session and go along with the various gestures of the `Context`:

- when a source is added, the anonymising rule protects the identity of the person interviewed;
- when a note is created, the rule about level of evidence keeps what the source shows apart from what we deduce from it;
- when a deliverable is being prepared, the last directive stops a solution being presented as established too early.

The skill carrying out an action has to start by working out which directives bear on it, then check that its result respects them.

```
request
↓
applicable directives
↓
action of the skill
↓
check on the result
```

The directives thus surround the action: they constrain it before it runs and serve to check its result afterwards. They do not act on their own and they do not know everything. It is the skills that make them effective at the right moment.

The mission says where to go. The directives go along with every gesture of the `Context`, so that I do not have to recall, at each step, the precautions already decided on.

### **Not Everything Deserves Reopening**

My folder holds dozens of sources and notes. Loading them all at every opening would drown the priorities in the mass. So `loading.md` picks out a stable core:

```
## Always

- `00-frame/mission.md`
- `00-frame/directives.md`
- `01-state/state.md`
- `01-state/todo.md`
- `01-state/handoff.md`

## On request

- the sources needed for a check
- the notes tied to the subject being worked on
- the active draft
```

Everything stays within reach, but not everything is loaded into the AI's context. `loading.md` names the elements that are indispensable for understanding the mission, reading a request and acting without losing the rules already established.

- The rest stays on disk and is only consulted when the need arises.
- An old source may be indispensable for checking a claim without taking up room in the active context throughout every session.

What goes into `loading.md` depends on the Context.

- A book will load its outline at every opening, a product exploration its central thematic note.
- The principle stays the same:
  - start with enough information to act,
  - then go and fetch the rest when the task warrants it.

## Where I Had Really Got To

The mission stays stable; `01-state/state.md`, for its part, is made to change. It comes in three blocks:

```
# State

## What we know
- Filling the report in later leads to a loss of detail in
  the first interview.
- Some mandatory fields are worked around in the second.

## What is still uncertain
- We do not know which fields are actually used again after
  being filled in.

## The debts
- The white paper has been added but how reliable it is has
  yet to be assessed.
```

The state does not tell the whole story and does not summarise all the notes: it sets out the best understanding I have of the work itself.

`state.md` speaks to the AI and to the human at once.

For the AI it is a lasting working memory.

- Loaded at every opening, it tells it what has been established and can be drawn on, which uncertainties it has to preserve and which debts it must not forget.
- Without it, it would have to rebuild that state from the sources and the notes at every new session.

For me it plays the part of a dashboard.

- In a few lines I can check what the `Context` holds as established, what it still treats as uncertain, and which pieces of work remain incomplete.
- I can then correct that state if it does not match my own understanding.

A few dozen lines are thus enough for the AI to pick the work back up without confusing what is established with what is supposed, and for me to check the understanding it is relying on.

`todo.md` draws the actions out of it:

```
## In progress
- T-001 – Prepare an interview on how the mandatory fields
  are really used.

## To do
- T-002 – Check which data the audits rely on.

## Done
- T-003 – Compare the first two interviews. – finished on
  2026-09-10
```

I do not necessarily open `todo.md` to change it myself. I can simply say:

| *Add a task to check which data the audits rely on.*

The `Context` creates a line, gives it an identifier and puts it in the right section.

A task can also come up during the work.

- If the analysis of a source brings out a check that has to be made, the AI offers to add it rather than letting it vanish into the conversation.
- It does not quietly turn every idea into a new obligation: when a task changes the priorities or commits me to extra work, the call stays mine.

Once the work is done, I can ask:

*Close task T-002 (or give a rough description of the task and the AI will find the right reference).*

The `Context` does not delete the line. It moves it to `Done`, adds the date it was closed and, where that is useful, states the result or the element that lets it count as finished.

- A finished task does not disappear.
- Its presence saves me redoing the same work and explains why a line of enquiry stopped being active.

So `todo.md` is not a list written once and for all. It is the queue of the `Context`'s actions: the AI consults it to know what is left to do, updates it as the work moves on, and keeps the finished tasks so as not to start them again.

### Where I Start Again

`handoff.md` is the last file of the core. It is rewritten at every save, to answer a very concrete situation: someone arrives now, a person or an AI, with nothing of the previous conversation. What do they need to know in order to start again without going wrong?

```
# Picking up again
```

```
## Where things stand
```

```
Three sources have brought out a tension between ease of  
entry and later use.
```

```
## What to know before going on
```

```
The two interviews describe particular situations, whereas  
the white paper argues a general point of view.
```

```
This body of material brings out a tension, but does not  
yet allow its scale to be measured or the conclusion to  
be drawn that it exists in every organisation.
```

```
## The next action
```

```
Prepare the questions that will make it possible to  
identify which fields are actually used.
```

```
## What has been left open
```

```
The degree of structure the audits need remains unknown.
```

The next action is single and can be carried out straight away.

A list of ten possibilities would hand me back the work of prioritising that this file exists precisely to spare me.

So the opening command takes in what was dropped off while the `Context` slept, reads this core, then announces that action to me.

- It recovers its purpose before recovering its details, and my two weeks away now cost no more than a minute of reading.
- The session can now take in new material without losing its direction.

---

**WORTH REMEMBERING — THE OPENING CORE**

Reopening a `Context` does not mean rereading its contents, but loading a small core: the mission, the directives, the state and the way back in. The mission says where to go, the directives set the conditions of the journey. A useful way back in names a single action, one that can be carried out straight away.

---

## Feeding: Letting the Material In

---

The session begins, and three things arrive at once.

- An operations manager sends me the write-up of an interview,
- an idea comes to me during the exchange,
- a note I left the day before is waiting in an inbox (note manager, email...).

These three kinds of material do not enter the `Context` in the same way, and each has its own skill.

### A Document Arrives as a Finished Object

An outside document is kept along with its original file, its usable text and a card that makes it possible to keep track of it.

```
03-sources/documents/  
├─ SRC00001-entretien-exploitation.md  
├─ raw/  
│   └─ SRC00001-entretien-exploitation.pdf  
└─ fulltext/  
    └─ SRC00001-entretien-exploitation.md
```

The card `SRC00001-entretien-exploitation.md` can take this form:

```
---
added: 2026-09-11
kind: pdf
intention: interview
raw: raw/SRC00001-entretien-exploitation.pdf
fulltext: fulltext/SRC00001-entretien-exploitation.md
extraction: ok
reliability: medium
---

# Interview with the operations manager

## Why it is here

Understand what stops technicians from producing usable job
reports.

## Interpretation – not analysed

The manager describes an opposition between the speed of
entry out on site and the need for structured data for
analysis.

## Coverage

- Mobile entry
- Mandatory fields
- Reuse of the data

## What it produced

- To be filled in after analysis
```

This card does not necessarily summarise the document. It answers questions of management first:

- where does it come from,
- why did I add it,
- can I read what is inside it,
- how much confidence should it be given?

The section *Interpretation* – not analysed puts a first reading in quarantine. It stops an impression formed at the moment of importing from being treated straight away as established knowledge.

When a source arrives as a PDF, a web page or another format that is hard to work with directly, extraction produces a text version of it that both the human and the AI can read.

- That operation produces no knowledge yet: it merely makes the material accessible without altering the original document.

The transformation comes next.

- The extracted text is analysed so as to bring out atomic notes, thematic notes, glossary terms or judgement calls.
- Those notes feed the *world of ideas*, where the material of the sources becomes a set of reusable bricks of knowledge.

The level of reliability is not a verdict on the sources added.

- It simply says how this material is to be used: an isolated account, a verified piece of data, a contractual document or a hypothesis still to be confirmed.

Not all the useful material exists in a document yet. Some of it lies in my experience, my memories, my observations or the distinctions I come up with while working.

When I voice them during a session of exchanges with the AI, they too enter the *Context*.

- They do not automatically become established facts

- They keep their origin and can later be set against other sources.

Adding material therefore means taking in a document just as much as keeping what the person who owns the `Context` brings to the conversation.

### The Person's Knowledge Comes In Through the Session

A session is not a document delivered once and for all. It is built up as the exchange goes on. It is recorded in `03-sources/sessions/SESSION-2026-09-12.md`, which can take this form:

```
# Session of 12 September 2026

## 09:41 – AI

Do the mandatory fields really improve quality, or do they
merely shift the effort onto the technician?

## 09:42 – Author

In the interviews, mandatory fields mainly guarantee that a
value has been entered.
They guarantee neither its accuracy nor its usefulness.
When the technician does not understand what a field will
later be used for, he fills it in as fast as possible so
as to move on.
So the constraint improves apparent completeness, but it
can shift the effort without improving the information
that can really be used.

...
```

Not everything the AI says needs to be kept. A question, an objection or a rephrasing that changes the work, on the other hand, does.

- My words and the AI's stay separate.

- That way I can find where an idea came from and tell my own intent from a proposal made by the system.

A session may be disorderly.

- Its role is to capture the real movement of thought, not to present a perfect summary.

### **The Letterbox Speaks to the Context While It Is Shut**

An idea comes to me outside a session, on the train or in the middle of a meeting and with no access to the internet.

- It then goes into the letterbox.
- That letterbox is the file `offline.md`, placed at the root of the Context.
- All I have to do is write in that Markdown file.

It can take this form:

Ask whether technicians work around the mandatory fields with generic values.

At the next opening of the context, that note is read again, set against the work in progress, then taken in at the right place. It may become a task, an interview question, a note, or vanish if it is no longer relevant.

### **Adding Is Not Yet Understanding**

Once this material is in, my tree holds several sources and the trace of the session under way.

```
03-sources/  
├── documents/  
│   ├── SRC00001-entretien-exploitation.md  
│   ├── SRC00002-entretien-exploitation.md  
│   └── SRC00003-livre-blanc-donnees-intervention.md  
└── sessions/  
    └── SESSION-2026-09-11.md
```

These files prove that the material is available. They do not prove that it has been understood.

The file `03-sources/ingestion-log.md` tells the two operations apart. For each source it states whether the analysis is complete, partial or in need of review, with a fingerprint of the content and a few processing notes.

That distinction becomes essential when a document is replaced, when an extraction changes or when a session is broken off. The `Context` can then pick the analysis back up without pretending to know everything.

The material is in. What remains is to turn it into ideas precise enough to be connected, discussed and used again.

---

### WORTH REMEMBERING — THE SOURCE

A document, words spoken in a session and a note made outside one come in through three different doors and each keeps its original form. A source holds on to the material and the evidence, never the interpretation that will be drawn from it. Adding is not understanding: the ingestion log tells what is present apart from what has been taken in.

---

## Transforming: From Sources to Ideas

---

I now have three sources in front of me.

- Two interviews describe the friction the forms cause,
- A white paper argues for the need for structured data.

The temptation would be to decide right away and write that mandatory fields harm quality. That would be going too fast: the `Context` has to preserve what each brings without melting them into a single conclusion.

So I call the skill that creates and updates the notes: it reads what has not yet been processed and draws ideas out of it.

### I Extract an Idea That Stands Up on Its Own

A source rarely holds an idea already fit to be used again.

- An interview mixes facts, examples, opinions, hesitations and details particular to the situation being described.
- Transforming it does not mean summarising everything, but drawing the ideas out of it.

The path comes in three steps:

|                 |                                            |
|-----------------|--------------------------------------------|
| Account         | "When the form is too long, some           |
| technicians     |                                            |
|                 | put in a generic value so they can         |
| finish."        |                                            |
| ↓               |                                            |
| Observation     | Generic values are sometimes used to       |
|                 | speed up entry.                            |
| ↓               |                                            |
| Stabilised idea | Entry that is too demanding can impoverish |
|                 | the job reports.                           |

A strong idea is neither a striking phrase nor a final truth.

- It has a clear, firm centre of gravity,
- Can be understood away from the text that gave rise to it
- Stays bounded enough to be confirmed, qualified or contradicted.
- It also keeps its origin and its limits, so that a particular account does not turn into a general rule.

The AI reads the source in the light of the mission and the directives, then sets it against the notes already there. It creates a new note when the idea is new; otherwise it enriches the existing one with the extra qualification or limit.

It does this work without asking me to approve every wording.

I can then reread and correct the notes.

- When an ambiguity comes up during the session, it asks me;
- If it later finds a contradiction it cannot resolve, it keeps it instead of deciding in my place.

```
04-notes/
└─ atomic/
    └─ AT000001-la-contrainte-de-saisie-peut-appauvrir-les-comptes-rendus.md
```

```
---
```

```
id: AT000001
status: active
sources: [SRC00001, SRC00002]
```

```
---
```

```
# Entry that is too demanding can impoverish the job
  reports
```

```
## Main idea
```

```
When the form slows the job down, technicians may enter
  generic values or cut their description back to the
  minimum expected.
```

```
## Why it matters
```

```
Data that is formally structured can look clean while
  describing what happens on site less well.
```

```
## Qualifications and limits
```

```
The two interviews do not make it possible to measure how
  often this happens.
```

```
## Origin / evidence
```

- SRC00001: examples of generic values used to finish faster.
- SRC00002: drop in accuracy observed after mandatory fields were added.

```
## Links
```

- Contrast with AT000002.

The title carries a proposition, not just a theme. The note stays understandable without my having to reopen the documents it came from, while keeping the links that let me check where it came from.

An atomic note is not fixed at the first attempt.

- It can gain a qualification, a new piece of evidence or a relation without losing its identity.

The white paper produces a second note: structured data is needed in order to compare, aggregate and automate.

- That idea stays separate from the first because it asserts something else.

### **I Refuse to Decide Too Early**

The two ideas look as though they clash, and I do not yet have enough material to settle between them. So I open a contradiction.

5

`## C-001 – Structure without degrading entry`

- `- On one side: imposed fields can impoverish descriptions from the field.`
- `- On the other: use at scale calls for comparable data.`
- `- What is at stake: determining which information has to be structured at the moment of entry and which can be structured afterwards.`
- `- Opened on: 2026-09-11`

The contradiction saves me from two symmetrical mistakes: picking a side too early, or smoothing the disagreement into a vague wording that bothers nobody.

- It stays open until a decision, a new source or a rewording resolves it.

Not every uncertainty deserves this treatment.

- I keep it for the tensions that really weigh on what the reasoning does next or on the quality of a decision.

### **I Connect What These Ideas Say Together**

During ingestion, the AI does not only create atomic notes.

- It also examines the relations between the ideas it has drawn out and the notes already there.
- When several of them form a whole that cannot be reduced to a single proposition, it creates a thematic note in the same operation and sets down the links that connect them.

I do not have to issue a second command.

- Creating the atomic notes, bringing them together and the possible appearance of a thematic note are all part of one and the same ingestion.

```

---
id: THM00001
status: active
sources: [AT000001, AT000002]
---

# Quality depends on entry and on use

## Angle

The quality of a job report cannot be judged by the number
  of fields filled in alone.

## Summary

The system has to obtain enough structure to make use of
  the data without making entry artificial or costly.

## Connected notes

- AT000001 – Demanding entry can impoverish the job
  reports.
- AT000002 – Structured data is needed for analysis and
  reuse.

## Tensions

- C-001 – Structure without degrading entry.

```

The thematic note does not merely place summaries side by side. It offers a reading of how they relate and shows what is left open.

### **I Notice That a Word Does Not Mean the Same Thing to Everyone**

Part of the disagreement sometimes comes from the vocabulary. The glossary then pins down the sense kept in this Context.

### ## Structured data

Information recorded in an explicit form that allows it to be compared or processed automatically.

Not to be confused with: a field the user is obliged to fill in at the moment of the job.

Examples: type of fault, downtime, equipment concerned.

This local definition is not meant to become universal.

- It simply stops two notes, or two people, using the same word to talk about different things.

### Bricks That Are Neutral Towards the Deliverable

A note belongs to the *world of ideas* as long as it is trying to stabilise a piece of knowledge rather than produce an effect on an audience.

- So it must not become too early an article's hook, a slide, a sales argument or a piece of a specification.

That neutrality does not mean the note stays vague.

- On the contrary, its title asserts a precise proposition, its content sets out the qualifications and its links make it possible to check the foundations.
- What stays neutral is its form: it is not organised to serve one single result.

AT000001 can thus become a question about workarounds in the interview guide, a risk in the framing note or an example in the presentation to the team. AT000002 can support a question

about how the data is used, justify a product requirement or explain to the team why free text is not always enough. The thematic note does not merge these bricks into a final text.

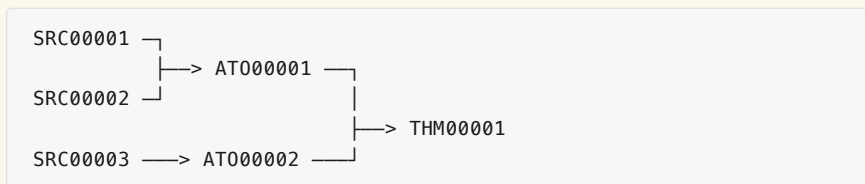
- It states what they make it possible to understand together and keeps the tension that will have to be handled differently depending on the deliverable.

This intermediate layer avoids starting the analysis again for each thing produced.

- The sources remain the evidence,
- the notes become knowledge that can be drawn on,
- each deliverable composes with them according to its own need.

## I Can Trace Every Idea Back to Its Evidence

At this stage, the relations can be pictured like this:



The named links let me tell a source that confirms an idea from one that qualifies or contradicts it.

- A level of confidence makes visible what rests on several observations and what remains a hypothesis.

The ingestion coverage shows which sources really contributed to the notes. It stops me treating a document as taken in simply because it sits in the folder.

These notes are not written for one particular deliverable.

- They form a shared body of knowledge that can feed several results without being copied out or shut inside any one of them.

**\*\*The *world of ideas* is precisely that reusable capital: a set of bricks stable enough to be found again and independent enough to be recombined.\*\***

---

#### **WORTH REMEMBERING — THE ATOMIC NOTE**

A stabilised idea carries a proposition and not a theme, and can be understood without reopening the document it came from. A contradiction is kept as long as it weighs on the reasoning, rather than being decided too early or smoothed away. Every note keeps the link to its sources, which makes it possible to trace an idea back to its evidence.

---

5.5

# Producing and Deciding

---

My thematic note has brought out a precise question: which information has to be filled in at the end of the job, and which can be completed afterwards?

Three people are now expecting something from me.

- A technician I have to interview,
- a committee that has to decide,
- a team I have to explain things to.

I am not going to run three investigations: I am going to draw three times on the same notes.

This is the only moment of the session when no skill helps me: producing a deliverable is a piece of writing, not a gesture that repeats.

## I Draw Three Times on the Same Stock

Going from notes to deliverables is not copying. Each result picks out the bricks that are useful to it, arranges them and rewords them for its recipient.

| Shared brick                              | Interview guide             | Framing note                        | Presentation to the team                 |
|-------------------------------------------|-----------------------------|-------------------------------------|------------------------------------------|
| AT000001 —<br>Entry that is too demanding | Ask which fields are worked | Describe the risk of losing quality | Show an example of friction in the field |

| Shared brick                                                | Interview guide                                    | Framing note                                         | Presentation to the team                                |
|-------------------------------------------------------------|----------------------------------------------------|------------------------------------------------------|---------------------------------------------------------|
| can impoverish the job reports                              | around and how                                     |                                                      |                                                         |
| AT000002 — Structured data is needed for analysis and reuse | Ask which data is really used again                | Establish the need for comparison and automation     | Explain the value expected for the team                 |
| THM00001 — Quality depends on entry and on use              | Organise the interview around the two requirements | State what is at stake and the criteria for deciding | Build the thread of the story                           |
| C-001 — Structure without degrading entry                   | Turn the tension into questions to check           | Keep it as a problem to be solved                    | Make it visible without announcing a premature solution |

The same stock thus produces three assemblies, without any deliverable taking up everything or replacing the notes that feed it.

Above all, that separation lets me correct the knowledge only once.

- If a new source qualifies AT000001, my next deliverables will start again from that enriched brick, instead of reproducing an old wording scattered across several documents.

### **I Start with the Most Urgent: The Next Interview**

My first need is to prepare the interview with a technician. The deliverable begins in the drafting space.

```
05-outputs/  
└─ _draft/  
    └─ questions-prochain-entretien.md
```

```
# Questions for the next interview
```

1. Which fields slow you down most during a job?
2. What do you enter when the information asked for is not available?
3. Which information could you add after the job?
4. Which data has to be accurate straight away in order to be useful when the reports are put to use?

The notes are trying to hold on to a reusable idea. The deliverable, for its part, picks a recipient, a form and an action it expects.

Once it has been reread and approved, the file leaves \_draft/ and goes straight into 05-outputs/. The move indicates a level of maturity, not a difference in kind of document.

So far I have only produced: I have given shape to what I already knew, for three audiences. The rest of this chapter is a

wholly different gesture. At some point you have to stop reporting and decide — and what I decide has to be findable again in six months.

### **The Decision Record Keeps the Choice That Shapes the Result**

“Interview a technician before writing a specification” is not a judgement call about what will be delivered.

- If that interview is simply the next action, it belongs in `01-state/todo.md`.
- If all discovery work has to respect that rule, it becomes a directive in `00-frame/directives.md`.

In neither case does it warrant a decision record.

5

The decision record appears when the work brings out several possible directions and one of them has to be chosen, the one that will shape the product, the text, the method or another result of the `Context`. It then keeps what has been decided about what will be designed or delivered, the alternatives set aside, the reasons for the choice and the consequences accepted.

In my example, the judgement call is therefore not about organising the interviews. It is about a functional decision:

- require all the information at the end of the job,
- keep free text only,
- or allow the report to be completed in a second stage.

A useful decision record does not delegate the heart of the choice to whoever will implement it. Three people reading it

must understand the same scope, the same expected behaviours and the same limits. Technical details that shape nothing may stay open, but none of them must be able to alter the decision quietly.

```
04-notes/
└─ decision-records/
    └─ DRE00001-limiter-la-cloture-a-quatre-informations.md
```

Here is what that gives once written. The template is long because it keeps all the reasoning that makes the decision understandable and open to revision — a reader already familiar with decision records can jump straight to the comment that follows it.

```
---
uuid: 0d5efbe2-1eac-4ad7-94c1-29f68adcf001
id: DRE00001
created: 2026-09-11
updated: 2026-09-11
status: accepted
decided_by: product team
decided_on: 2026-09-11
supersedes: null
superseded_by: null
sources: [AT000001, AT000002, THM00001]
---

# Limit the closing of a job to four mandatory pieces of
  information

## Decision

For a job to count as closed, exactly four pieces of
  information have to be present:
```

1. The equipment concerned, either already stated on the job sheet or entered at the moment of closing.
2. The real date and time the job ended.
3. The state of the equipment at the end of the job: back in service, kept running in degraded mode, or left out of service.
4. A non-empty summary of the work done, or of the reason why the equipment could not be put back into service.

No other information conditions the closing.

After closing, the report may be completed with the nature of the failure, its identified cause, the nature of the work done, the parts replaced and how long the equipment was unavailable, without calling the closing of the job into question.

The technician may fill in this further information, but the operations manager has to complete or approve it within twenty-four hours.

A report that is still incomplete remains readable.

It is flagged as still to be completed, and the operations manager is then alerted after twenty-four hours.

## ## Context

Technicians must be able to close their work out in the field without going through a full analysis form.

The operations department nevertheless needs comparable information in order to analyse failures, their causes, the work done, the parts replaced and the periods of unavailability.

## ## Assumptions

- The job sheet normally identifies the equipment before the work starts.
- The technician knows the state of the equipment and can describe the work done briefly at the moment of closing.
- The operations manager has the time and the information needed to approve the complete report within twenty-four hours.

## ## Alternatives

- Require all nine pieces of information at the moment of closing.
- Keep only a free summary and extract the job information from it automatically afterwards.
- Make all the information optional and check quality only after the fact.

## ## Reasons

The four immediate pieces of information are enough to identify the job, date its end, know the state of the equipment and understand the work done.

The five pieces of analytical information remain necessary, but they must not stop the technician finishing the work out in the field.

The second stage explicitly makes the operations manager responsible for completing or approving the report.

## ## Basis

- AT000001 describes the workarounds tied to entry that is too constrained.
- AT000002 establishes the need for comparable data.
- THM00001 connects the two requirements.

## ## Consequences

### ### Positive

- Closing calls for only four pieces of information known straight away.
- The information needed to analyse completed jobs remains available once the report has been completed.
- Responsibility for missing data is visible.

### ### Trade-offs

- The job metrics may stay incomplete during the twenty-four hours following a job.

- The operations manager takes on an extra approval load.

### ### Risks

- Incomplete reports may pile up.
- Information added after the fact may reflect what was found in the field less faithfully.

### ## Conformity criteria

- A job cannot be closed if any one of the four immediate pieces of information is missing.
- A job can be closed without any of the five further pieces of information.
- The further information can be changed without altering the closed state of the job.
- The report is presented as "still to be completed" or "complete and approved" in every use made of it.
- The operations manager is alerted twenty-four hours after closing if the report is not complete and approved.

### ## Related decisions

- Resolves C-001 – Structure without degrading entry.

### ## Sources

- SRC00001
- SRC00002
- SRC00003

The status `accepted` indicates here that the judgement call is approved and precise enough to guide implementation.

- As long as the information expected, the responsibilities or the conformity criteria remain undetermined, the decision record has to keep the status `proposed`.
- Once accepted, it is frozen as a historical trace and no longer changes quietly.

If a new decision replaces it, the old decision record stays readable and the two files are connected by `supersedes` and `superseded_by`.

That continuity keeps the full chronology of the judgement calls: what had been decided, on what information, then which element led to a different choice.

Six months later, then, I do not only find the current rule. I can understand why a first direction had seemed right, what made it inadequate and how the next decision came about.

Without that chronology, a succession of choices ends up looking like incoherence. With it, it becomes once more the understandable story of a product that is changing.

- The alternatives show what was set aside.
- The trade-offs make visible what was given up.
- The basis connects the decision to the knowledge and the evidence that support it.

### **Producing Does Not Use Up What I Know**

The interview guide, the thematic note and the decision record deal with the same problem without overlapping, because they fulfil three different functions.

- The note keeps what the `Context` believes it knows.
- The decision keeps a choice and the reasons for it
- The guide turns that knowledge into action with a particular recipient.

Another deliverable will be able to take up the same shared knowledge in a different form. **Producing therefore does not**

---

**empty the Context of its substance: it reveals what that substance makes possible.**

What remains is to record the state reached, so that the next session starts again without having to piece the path together.

---

#### **WORTH REMEMBERING — THE DECISION RECORD**

The same notes feed several deliverables, and no deliverable consumes the knowledge it uses. A decision record keeps a judgement call that shapes what will be designed or delivered, not an action to be carried out. It keeps what was set aside and what was given up, so that the choice stays open to revision.

---

## Saving and Picking Up Again

---

The decision record has just been accepted and a meeting starts in three minutes. I have finished neither the study nor even a stage that might look like an ending.

That is the ordinary case: another task calls for my attention, the machine has to restart, I move to another computer, or the AI's context window becomes too full.

- So saving does not answer a need of the work itself, of the “the study is finished” kind.
- It is a technical guarantee: the state of the `Context` has to outlive the session that produced it.

I must be able to break off now, reopen later and find the work exactly where I left it.

A skill takes care of it, and it has to run in a few seconds: a save that costs a lot will not be launched at the moment you are in a hurry, which is to say the moment it is most useful.

### The AI Freezes What We Know at This Moment

The file `01-state/state.md` sums up what is stable enough to guide what comes next.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ state.md
```

At the moment of saving, it can take this form:

```
# Current state

## Stabilised knowledge

- Entry that is too demanding can impoverish the job
  reports.
- Use at scale nevertheless calls for structured data.

## Decision taken

- Closing requires the equipment concerned, the date and
  time of the end, the state of the equipment and a summary
  of the work done.
- The nature and cause of the failure, the nature of the
  work, the parts replaced and the period of unavailability
  are completed after closing and approved within
  twenty-four hours by the operations manager.

## Main uncertainty

- Measure whether this separation really reduces closing
  time without degrading the quality of the data.

## Knowledge debt

- No pilot has yet measured closing time or the proportion
  of reports complete after twenty-four hours.
```

- A finished task can disappear from the todo list.
- An unresolved contradiction stays visible in the state.

The synthesis holds only what the work has stabilised enough. The state is rewritten to describe the present, while the synthesis grows step by step so as to keep the understanding acquired.

## I Prepare a Single Action for My Future Self

The handoff file does not tell the whole story: it prepares the next useful gesture.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ handoff.md
```

The file `01-state/handoff.md` can take this form:

```
# Picking up again  
  
## Current state  
  
DRE00001 is accepted and C-001 is resolved.  
The four pieces of information for closing and the five  
further ones are fixed.  
  
## Point to watch  
  
Do not add a mandatory piece of information at closing  
without opening a new decision record that replaces  
DRE00001.  
  
## Next action  
  
Start implementing DRE00001.  
  
## Open  
  
- Define the wordings shown to technicians for the three  
  possible states of the equipment after the job.  
- Prepare the measurement of closing time and of the  
  proportion of reports complete after twenty-four hours.
```

A useful way back in names a single next action. A list of ten possibilities would force the next session to redo the work of deciding that I have just done.

## I Keep a Trace So I Can Compare My Returns

The file `01-state/checkpoints.md` holds a series of compact snapshots of the situation.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ checkpoints.md
```

The file can take this form:

```
## 2026-09-11  
  
- Sources: 3  
- Atomic notes: 2  
- Thematic notes: 1  
- Open contradictions: 0  
- Accepted decision records: 1  
- Next action: start implementing DRE00001.  
  
## 2026-09-04  
  
- Sources: 1  
- Atomic notes: 1  
- Thematic notes: 0  
- Open contradictions: 0  
- Next action: get the point of view of the operations  
  side.
```

These snapshots show me how the Context has moved without forcing me to compare every file. They also let me spot a standstill, or a debt growing without my noticing.

## The AI Notes What Happened, Not What It Thinks of It

The history records the important events without trying to reconstruct every exchange. In my example, the day's events are added to `02-history/2026-09-11.md`.

```

contexte-qualite-comptes-rendus/
└─ 02-history/
   └─ 2026-09-11.md

```

The block added can take this form:

```

## 2026-09-11

- SRC00003-livre-blanc-donnees-intervention added.
- AT000002 and THM00001 created.
- C-001 opened.
- `questions-prochain-entretien.md` approved.
- DRE00001 accepted and C-001 resolved.

```

- The state says where things stand.
- The history says what happened.
- The handoff says what to do next.

### I Let the Machine Check What I Would Not See

At the moment of saving, a few mechanical checks reduce the oversights. Their result is written into 01-state/health.md.

```

contexte-qualite-comptes-rendus/
└─ 01-state/
   └─ health.md

```

- Do all the files referred to exist?
- Are the identifiers unique?
- Does every note state its sources?
- Is a source that has been added but not analysed flagged?
- Are replaced decisions connected to the ones replacing them?
- Is the next action explicit?

These checks do not replace my judgement. They simply stop an error of structure passing itself off as a problem of thought.

### The Full Cycle, and What It Allows

The general movement now becomes visible.

```
opening
  ↓
reading the letterbox
  ↓
loading the mission, the state and the handoff
  ↓
adding a source
  ↓
creating or updating the notes
  ↓
producing a draft
  ↓
check, approval or decision
  ↓
updating the state, the synthesis and the handoff
  ↓
checkpoint and history
  ↓
save check
  ↓
next opening
```

I do not wait for the end of that chain to save.

- I do it after adding a source, halfway through creating a note, before a decision or while preparing a deliverable, when I see my context window filling up...

That is exactly the role of saving: **making each of these states one you can pick up again, whatever happens to the conversation under way.**

So my meeting can begin.

---

**The Context is not only a tree of files: it is a system for thinking over time, one that lets material in, preserves tensions, stabilises ideas, produces several results and prepares the return to work.**

Its form stays deliberately simple and portable — readable files are enough to make the work observable, checkable and transmissible.

Once this engine is understood, another question appears: how does the Context deal with the tools, the knowledge bases and the real product that exist outside it?

---

#### **WORTH REMEMBERING — SAVING**

Saving does not mark the end of a stage: it is the guarantee that the state outlives the session that produced it. The state says where things stand, the history what happened, the handoff what to do next. The cycle can be broken off at any point, and every one of those points can be picked up again.

---



# Part 6

## The Context in Its Ecosystem

---

A well-maintained Context makes it tempting to put every piece of available knowledge into it.

This part examines what it leaves outside, and why.

It stays small so that it can still think effectively.

Anything kept merely as a precaution would age without warning while the original continued to

change elsewhere. So it works with neighbours:

tools whose job is to store information, take

measurements or execute operations, and other

Contexts that investigate different questions.

## Querying Structured Knowledge

---

The job reports can wait: this morning I am working on another area of the same product — users who abandon sign-up part-way through. An agency has just delivered its study, and I add it to the `Context`: several participants hesitate when faced with the password rules.

That observation reminds me of an idea I have encountered before: asking for a lot of effort before users have seen the value of the product can drive them away.

- I am almost certain I have material on the subject.
- I simply no longer know where it is or which words I had used.

Did I talk about initial effort, friction, activation or onboarding? The idea may have been

- drawn from a book,
- developed in a product management note,
- connected to another user journey,
- ...

I could try several keyword searches and open the results one by one. Instead, I ask the `Context`:

*What have we already learnt about the effort demanded before the user sees the value of the product?*

## The Context Queries a Wider Memory

The answer is not in the active Context. It may, on the other hand, be in my LLM Wiki, a shared library where the thousands of notes drawn from books, articles, standards and other documents are already structured.

The LLM Wiki and the Context are not the same kind of system. The wiki is a knowledge base.

- It keeps structured notes (in my case I reused the idea of atomic notes) drawn from what others have written: a definition, a principle, the result of a study, a requirement in a standard.
- Each note says what is known, and where it comes from.
- It may hold tens of thousands of them, built up over years and on subjects unrelated to one another.

No Context will ever hold that quantity, and that is not its role. But the wiki stops there.

- It does not say why that knowledge matters in the work I am doing today, or what I made of it.
- It does not connect an idea to a decision, or a decision to what it cost me.

## The LLM Wiki is not a system for thinking: it stores knowledge at scale, but it does not reason.

The reasoning lives in the Context. It is the Context that holds the why behind the how:

- what the mission is,
- which directives and decisions are in force,

- why this idea was kept for this mission,
- what I decided against,
- what I agreed to give up,
- which contradiction is still open.

It manages that only because it stays small — a few dozen notes that a human can read through.

**Each pays the price of its own strength.**

- **The wiki gains scale and loses the thread.**
- **The Context keeps the thread and gives up scale.**

Which is exactly why they have to talk to each other rather than merge.

### **I Describe the Idea I Am Looking For**

I do not copy the whole wiki into the Context devoted to signing up. Nor do I run a RAG query and then search a graph.

I simply put my question to the Context:

6

*What have we already learnt about the effort demanded before the user sees the value of the product?*

I put it in my own words.

- I do not need to know a note's title, or where it sits, or the vocabulary used when it was written.

The Context passes that idea to the LLM Wiki through its API. Everything else happens inside the wiki.

### **The Wiki Searches by Meaning, Then Around the Result**

The LLM Wiki starts by finding notes that are semantically close to my question.

- A note may talk about initial friction, activation or time to first value without using the word “effort”.

That is what its RAG lets it do: match my idea against knowledge that is already structured, without requiring the same vocabulary.

The wiki then looks around the best results. Its semantic graph follows the relations already set between the notes and adds the neighbours that may shed light on the subject:

- a definition,
- a limit,
- a contradiction,
- a complementary idea.

One of those neighbouring notes might never have answered my question directly. It becomes valuable all the same, because it shows under which conditions the idea applies, or where it stops being true.

```
graph TD; A[an idea worded in my own words] --> B[the wiki finds the notes close in meaning]; B --> C[it adds their useful neighbourhood]; C --> D[a few sourced references come back]; D --> E[the Context keeps what serves its mission];
```

**The RAG finds what I would not have known how to name.  
The graph brings out what I would not have thought to  
look for.**

For me, these two operations are one and the same. I ask for references that address a problem; the LLM Wiki looks for the idea and its neighbourhood, then returns a shortlist.

6

### **The Context Receives References, Not a Ready-Made Answer**

The wiki sends back existing notes with their content, their origin and their relations. It does not manufacture a summary that would quietly blend several results together.

So the Context can read each result, compare it with the sign-up observations, and go back to its source when a piece of evidence has to be checked.

- Some notes really do concern effort imposed before access to the product.

- Others talk about effort demanded later, when the user already knows the value of it.

Their closeness to my question is not enough to make them equivalent.

The `Context` sorts them out on the basis of its mission. It can:

- keep an idea that sheds light on people giving up during sign-up,
- keep a contradiction,
- set aside a note that is too general.

A note that is kept enters the `Context` with its origin and the sources it rests on. The rest stays in the wiki.

That way I have the benefit of a far larger memory without turning every `Context` into a copy of my whole library.

- I do not need to learn how that memory is indexed in order to use it: I describe the idea I am looking for, then check the references it brings back.

This chain gives access to knowledge that is already structured.

- It does not yet say what the product actually does today.

To answer that other question, the `Context` has to leave the library and question the code, the tests and the running product directly.

---

**WORTH REMEMBERING — TWO SYSTEMS, ONE DOOR**

The LLM Wiki gains scale and loses the thread; the `Context` keeps the thread and gives up scale. I describe an idea in my own words; the wiki finds the notes close in meaning and adds their useful neighbourhood. The `Context` receives a few sourced references and keeps only what serves its mission.

---

## Querying the Product Itself

---

I am in the middle of rereading the study on sign-up abandonment when a question arrives:

*Does an invited user have to create a password like everyone else?*

The question seems tiny. But someone in support is waiting for the answer so that they can reply to a customer.

I could:

- break off my reading,
- open a test environment,
- set up two accounts,
- replay the journeys,
- compare what happens.

That is what I used to do.

- Half an hour later, I would probably have the answer.
- I would also have lost the thread of my work.

This time, I put the question to the `Context`.

### **The `Context` Checks the Product**

The `Context` works with the notes drawn from the user study and keeps that study as the source for their claims.

- It also carries the mission under way.

- It understands why the question matters, but none of that lets it know how the product behaves right now.

I no longer maintain a specification whose job is to describe that behaviour for the long term.

- The product changes too often for a document written alongside it to remain a faithful copy.

**The only truth that is up to date is what has been shipped and is running in production: the application's source code.**

In my system, skills outside the `Context` know how to inspect the source code and generate documentation from the implementation. The `Context` calls the one that answers product questions and passes my request to it.

question from the product manager



Context



calls



skill that reads the product



questions



real product

The skill starts by finding the invitation journey in the code. It then follows that journey through to the rule for creating the account.

The code shows the behaviour as programmed. A test confirms that an invited user can finish signing up thanks to

the link received, without creating a password straight away. The configuration shows that no variant specific to a customer is active for this journey.

The skill gives me an answer in functional terms:

*An invited user can reach their account from their invitation link without creating a password during that step. If they then choose to create one, the ordinary rules apply. The tests confirm the invitation journey and the voluntary creation of the password. The journey for recovering a forgotten password was not examined.*

I can pass the first two sentences on to support. I can also open the references the skill supplies if I want to check its analysis with a developer.

Finally, I can capture that answer in the `Context`, with its evidence, the date of the check and the limit it flagged.

- Answering support settles one case and stops there.
- Capturing it in the `Context` brings a verified fact about the product into the mission's *world of ideas*: the next question about the invitation journey will not start from nothing, and the note drawn from the study gains what observation alone did not give — the exact moment at which each journey asks for the password.

### **An Answer Is Only as Good as Its Evidence**

The skill did not stop at a plausible answer. For each claim, it states what it consulted:

- the rule present in the code,
- the test that checks the behaviour,
- the configuration that determines which variant is active.

That trace guards against a confident but wrong answer. It also makes it possible to see straight away what is missing. Here, the skill did not check the recovery of a forgotten password. It says so instead of extending its conclusion to a journey it has not read.

Depending on the question, other elements may become necessary.

- The code holds the possible rules, but a feature flag says which one is active.
- The database schema shows what the product can record, while authorised data can reveal the situation of a particular account.
- The Git history and the pull requests help in finding when a rule changed and the explanations kept with that change.
- ...

The skill gathers only what it needs in order to answer the question asked. It does not Hoover up the whole repository and does not turn every file into a source of the `Context`.

The answer stays readable by support, by sales or by a product manager. The evidence stays available for whoever has to check it or take the enquiry further.

### Finding the Rule Does Not Decide on Changing It

I go back to my study on signing up. It had already taught me that the two journeys do not present the creation of the password at the same moment.

The skill adds what the study could not see. The invited user does not get round the password rule: they meet it later, and it is then exactly the same.

So the friction has not disappeared for them, it has moved.

That precision shifts my question. I am no longer asking who is subject to the rule, but at which moment each of them meets it: before having seen the value of the product, or after. The code has taught me what the product does.

- It does not tell me whether the rule is a good one, whether it really protects accounts or whether its cost to users is acceptable.

To decide, I still need the other elements of the Context:

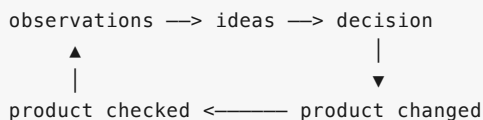
- the observations of the study and their scope;
- the security objectives;
- the reasons that had led to adopting that rule;
- the consequences expected from a change.

The skill establishes a fact about the product. The Context connects that fact to the notes and the decisions already there, while keeping the sources that make it possible to check where it came from. **The judgement call stays mine.**

If I decide to change the creation of the password, the decision record keeps the choice, its scope and the reasons

that support it. After implementation, the skill can go back to the product and check that the new behaviour does match the decision.

So the movement works both ways. The product informs the thinking, then the decision transforms the product.



## Keeping the Why, Reconstructing the What

### THE CODE-CENTRIC APPROACH

The code is no longer merely the end result of development. It becomes the source from which I can rebuild whatever describes the product's current behaviour. The `Context` keeps what the code cannot explain on its own: the sources, the reasoning, the decisions, the alternatives set aside and the reasons for the choice.

6

That separation changes where documentation sits.

I keep what explains why the product took one direction rather than another: the interviews, the constraints, the hypotheses and the trade-offs. I rebuild from the product whatever describes its current behaviour:

- an answer to support,
- functional documentation,

- a changelog,
- ...

I no longer maintain by hand two versions of the same behaviour, one in the code and the other in a document that starts ageing the moment it is published (and well before, in fact).

Not everything comes from the code, though.

- A user study is not in there.
- A regulatory obligation keeps its own source.
- A decision does not come down to the behaviour that results from it.

**The code carries what has been implemented. The `Context` keeps the why and makes it possible to discuss it when the situation changes.**

### Coming Back Six Months Later

The answer given to support stays attached to the version of the product examined today. It does not automatically describe the versions that follow.

- Six months later, the invitation journey may have changed.
- A feature flag may activate another variant, or a new way of signing in may have appeared.

The `Context` keeps the question, the answer in functional terms, the date of the check, the version examined, the evidence cited and the limit flagged. That way I can recover what we knew when we reached the conclusion.

---

To know the present behaviour, I simply ask the skill to run the check again.

- It starts from the real product instead of copying out the old answer.

I can then tell apart two questions that often get confused:

- why did we take that decision at the time?
- what does the product do today?

The `Context` answers the first by keeping the reasoning. The skill answers the second by going back to the source of truth. So the product does not replace the `Context`, and the `Context` does not claim to contain the product.

- One supplies a reality to be checked;
- the other keeps the knowledge needed to make sense of it and to change it.

This flow extends beyond external tools. When several `Contexts` work on different aspects of the same product, their knowledge too must be able to circulate without being collapsed into a single folder.

---

### **WORTH REMEMBERING — THE CODE-CENTRIC APPROACH**

No specification is maintained as a lasting description of the product: whatever describes its behaviour is rebuilt from the code. An answer is worth only what the evidence beside it and the limits it flags are worth. The code records what was implemented; the `Context` keeps the why and makes it possible to reconsider it.

---

## The Day I Had Three Contexts

---

A technician tells me he had to call a customer back over a piece of information missing from a job report.

I write the sentence down. Then I stop, because I do not know where to put it.

- I have a Context on the quality of job reports — the one that has taken up the whole second half of this book.
- I have a second one on the job itself: what it contains, the rules that govern it, what has already been decided about it.
- And I have a third, broader one, opened a few weeks earlier: how to get operators to take on first-line interventions.

The technician's sentence is relevant to all three:

- it documents a job report that fell short,
- it says something about the job,
- it sheds light on the question of the operators.

That is the moment when the problem changes shape.

- As long as I had a single Context, the question was: does this go in, or is it out of scope?
- With three, the question becomes: into which one?

### The Two Ways of Getting It Wrong

The first temptation is to bring everything together.

- One single large space, all the knowledge in the same place, no more question of where to file things.

That is exactly what I was running away from. A single space containing everything recreates the problem of traditional document-management systems: a place where

- the information is present and the sorting starts again every time it is consulted,
- the goals get mixed up,
- the sources pile up,
- and no note has a clear owner any more.

I told the story of that dead end at the start of the book; recreating it with a fresh vocabulary would not make it any better.

The second temptation is the opposite: let each Context stay at home. Each one is clean, bounded, coherent — and blind.

The result soon becomes clear. My Context on the job reports spent three sessions settling a rule that the one on the job already carried. It was not a visible duplicate: it was work done twice over, with no way for anyone to notice.

So a third way is needed, and it starts with knowing what each Context carries.

### **A Number in Front of the Name**

My Contexts are called 01-saisie-compte-rendu, 02-intervention, 03-autonomie-operateurs, 04-exploitation-catch-all.

The number is not a rank, it is a naming convention. It says what the Context carries, before I even open it.

- **02- an entity of the product.** A piece of the software that runs in production and has its own rules: 02-facturation in an ERP, 02-intervention in mine. What is in it is established. This Context remains open.
- **01- a hypothesis about an entity.** 01-saisie-compte-rendu: I think the way reports are filled in ought to change, I do not yet know how. Nothing it contains is established, so I create a 01- Context to keep what is stable (in production) clearly apart from a hypothesis.
- **03- a problem.** 03-autonomie-operateurs: get operators to take on first-line interventions. It is not a feature, it cuts across several entities and it gives rise to 01-s.
- **04- the catch-all.** A sentence overheard, a PDF, a hunch: material whose subject has no Context yet. Without it, I would have to create one per scrap of an idea — I started that way and found myself with more than fifty insubstantial Contexts within a few weeks.

### Every 01- Eventually Leaves Active Work

On the day it goes into production, the new way of filling reports in stops being a hypothesis: it becomes part of the job.

What now describes the product is proposed to 02-intervention. The rest stays in the exploratory Context, which is then archived. There is now only one active place to look for how a job report works.

A hypothesis that comes to nothing is archived along with whatever stopped it. It is not a failure to be wiped away: it is what will stop me going down the same road again in two years' time.

The `04-`, for its part, empties in the other direction. The day a problem emerges, I create the `Context` that goes with it, then go back to see what the catch-all had kept on the subject. Whatever is of no use is deleted.

All of that settles a single question — where do I put this — by separating what I am looking into from what is established.

- Without that separation, every abandoned hypothesis leaves its traces in the description of the product.
- And you no longer dare suppose anything in a place where everything is supposed to be true.

### **The Mistake It Took Me Months to See**

I began by opening `03-tarification`. It seemed obvious: it is a big subject, it concerns several entities, and it is long-running. Six months later, it held:

- the price list,
- two competitor benchmarks,
- a discussion about discounts,
- a support ticket about a wrong invoice,
- a note on Belgian VAT,
- three screenshots of a competitor,
- ...

None of that was out of scope: it all really was about pricing.

That is exactly the problem. To reject material, you have to be able to say “this does not help me”. Now that `Context` was not looking for anything — **it had no question**, only a label.

`03-autonomie-operateurs` does have one: how to get operators to take on first-line interventions.

So a `03-` is not a big subject. It is a question. Pricing, for its part, has not disappeared: it runs across my `Contexts` — `02-facturation` carries its rules, a `01-` may depend on it — without governing any of them.

So my convention has no box for domains, nor for a good many other things. It serves to file what has an obvious place, and nothing more. The rest waits in the catch-all, and I retrieve it by describing what I am looking for rather than remembering where I filed it.

I now have two of them, a lesson learnt through trial and error. A single catch-all mixed material from my product work with material from the industrial operations domain — two vocabularies, two audiences, two ways of naming the same objects. It became unreadable faster than it emptied.

- `04-exploitation-catch-all` takes in material from my customers’ business domain: operations;
- `04-productops-catch-all` takes in what touches my own trade and how it is organised: product ops.

Each stays readable, so each one empties.

The technician’s sentence, for its part, did not need the catch-all. It went into `02-intervention`, because that is where what is

true of the product lives, independently of the question I happen to be asking right now.

What remains is to find out how my three Contexts become aware of it.

---

#### **WORTH REMEMBERING — FOUR PREFIXES, A SINGLE QUESTION**

The number in front of the name says what the Context carries: 02- an entity that runs, 01- a hypothesis about that entity, 03- a problem that cuts across several of them, 04- the holding area where you put what you cannot yet place. A 01- exists in order to propose to its 02- what now describes the product, and then to be archived. If it comes to nothing, it is archived along with whatever stopped it. A domain is not a Context: it runs across several of them without governing any. A Context with no question will never know what is out of scope.

---

## Reading at the Neighbour's Without Moving In

---

I return to `01-saisie-compte-rendu`, the Context devoted to the quality of job reports. To move forward, I need to know the rules of the job itself: what a technician is allowed to enter, what is mandatory and what has already been decided.

Those rules do not live in my Context. They are stabilised in `02-intervention`, the neighbouring Context that describes what actually exists in the product.

I could copy them out so as to have them to hand. Six months later, I would probably have two versions of the same rule and no simple way of knowing which one is authoritative.

So I start with a lighter-weight approach: my Context declares that it reads from `02-intervention`.

### Reading Leaves the Knowledge with Its Owner

That relation is written into the stable frame of my Context, beside its mission and its directives. It opens a precise right: to consult what `02-intervention` already knows.

The material stays with its owner. It goes on changing there, and my exploration leans on it without claiming to own it.

That way I can check the current rule on mandatory fields without duplicating it in every study that concerns a job

report. I can also go back to its sources and to the decisions that built it.

Reading does not grant permission to edit. If a note strikes me as incomplete or wrong, my `Context` can put forward a proposal, but it never corrects its neighbour directly.

This first relation answers a common need: using knowledge from outside while letting it live in the place that is responsible for it.

It does not answer every case.

### **Absorbing Makes the Knowledge Independent**

Sometimes simply consulting it is not enough.

I may need to freeze the exact rule on which a decision rests.

I may want to reword it for the particular scope of my exploration. I may also need that knowledge to stay available if the `Context` it came from is archived, moved or reorganised.

In that case, I choose to absorb it.

Absorbing brings a piece of knowledge into my own `Context`. I can take a note as it stands or reword it in my own vocabulary. Depending on the need, I also copy the sources that make it possible to check it, or I keep references to them.

The copy does not thereby become an idea with no past. It keeps the `Context` it comes from, the date of the absorption and whatever was reworded.

So the difference lies less in the technical mechanism than in the responsibility that follows from it:

- when I read, the knowledge continues to belong to the neighbour;
- when I absorb, an independent version enters my own Context, and I become responsible for how it is used.

**Reading creates a dependency. Absorbing creates a new responsibility.**

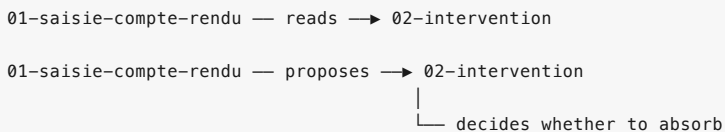
### The Movement Can Go the Other Way

My exploration does not only consume what 02-intervention already knows. It produces observations, hypotheses and sometimes lasting knowledge of its own.

While studying the quality of job reports, I may discover that a rule concerning how they are filled in actually belongs to the job itself. That knowledge ought not to stay shut inside an exploratory Context that may end up archived.

I do not write it straight into 02-intervention. I submit it to that Context.

The direction of flow has just changed:



Whoever produces a piece of knowledge does not decide where it finally sits. The Context that receives it judges it by its own mission.

It can:

- take the note over as it stands;
- reword it in its own vocabulary;
- keep a divergence when both readings remain valid;
- refuse the absorption when what is useful to the exploration does not describe the job itself.

Refusal is not a failure of the flow. It is what stops a local result quietly becoming a general truth.

### **When the Exploration Becomes a Product Reality**

A few weeks later, the new way of filling in job reports goes into production. 01-saisie-compte-rendu has played its part: what was only a hypothesis has become a reality of the product.

That does not mean moving its entire contents into 02-intervention. I go over what the exploration produced and separate two sets.

The first holds what now describes or explains the product:

- the notes that establish the new rules for filling reports in;
- the vocabulary stabilised for naming them;
- the structuring decisions and the reasons for those choices;
- the contradictions that remain open after going into production;
- the sources or references needed to check that knowledge.

Those elements are submitted to 02-intervention. The stable Context does not pile them up blindly: it enriches its existing notes, rewords what has to be reworded, and creates a new note only when an idea really is new.

In our example, the rule that limits closing to four pieces of information joins `02-intervention` along with the decision that explains it. It now describes a reality of the product.

The second set tells the story of the work without describing the product as it stands:

- the hypotheses abandoned;
- the mock-ups and the avenues set aside;
- the intermediate drafts;
- the tasks specific to the exploration;
- the discussions that make it possible to understand the road travelled.

Those elements stay in `01-saisie-compte-rendu`. They have not become useless: they explain what was tried, why certain directions were turned down and what should not be started again without a new reason.

Once that selection is done, the exploratory `Context` is archived as a whole. It stays readable and can be picked up later, but it no longer claims to describe the work in hand.

If the exploration had come to nothing, it would also have been archived, with the reasons for dropping it, without imposing anything on `02-intervention`.

So going into production does not merge two `Contexts`. It triggers a review: what now describes the product joins the stable `Context`; what tells the story of the exploration stays in the archived `Context`.

That selection protects both sides:

- the exploration stays free to be wrong;
- the stable Context receives only what it chooses to recognise as its own.

### **The Relations Are Part of the Frame**

Reading, absorbing and proposing must not remain implicit actions.

The stable frame of the Context states:

- which other Contexts it may consult;
- what that consultation allows;
- under which rules a piece of knowledge may be absorbed;
- how its origin and any rewordings are kept.

That governance requires no heavy machinery. It simply gives the relations an explicit home, so that provenance, responsibilities and boundaries stay visible.

So the network is not built by connecting everything to everything. It is built one relation at a time, each time a piece of work really needs a neighbour.

- Reading from the neighbour avoids duplicating what they know.
- Absorbing makes independent what has to live in my own Context.
- Proposing, finally, lets the neighbour decide whether to take in what my exploration has produced.

---

That flow looks clean on paper. It has a price all the same: a reference can age without warning, a copy can lose its origin, and two Contexts can hold on to different readings.

That is what the next chapter will examine.

---

#### **WORTH REMEMBERING — THREE ACTIONS, THREE RESPONSIBILITIES**

Reading consults knowledge that stays with its owner; absorbing creates an independent, sourced version of it in the Context that receives it. An exploration proposes what it has learnt, but the owner of the target Context decides what comes in. Going into production does not merge the folders: the stable Context absorbs what now describes the product, while the exploration keeps the rest before being archived.

---

## The Cost of Circulation

---

Six weeks have gone by.

My work on the job reports is ready to go out for review, and someone spots a discrepancy. The rule I am leaning on — the one that says which fields are mandatory at closing — is no longer the product's rule.

It has changed. While I was working, another exploration reached a conclusion, and the Context of the job absorbed its result.

Nobody told me. I did not read it again, because I had read it six weeks earlier and nothing told me to go back to it.

The previous chapter lays down a rule that seems neat: you reference in the direction of reading, you copy in the direction of absorption. I took it for a resolved trade-off.

It does not remove the risk. It moves it, and it is better to know where the risk ends up.

### An Invisible Dependency

What has just happened has a name: a blind dependency.

The link is live, so what I read can change without me. That is not the problem — it is often exactly what I want. A rule that changes in its owner's Context is the system working.

The problem is not knowing that it has moved.

---

What is missing is simple to state: nothing monitors what I read. My `Context` knows who it depends on; it does not know when what it depends on has changed. That is half the mechanism, and it is the costly half.

A copy would have protected me from that. The receiving `Context` becomes independent: it no longer depends on the continued existence of its source or on how that source changes.

But it loses something else in exchange. If nothing says where the copy comes from, the provenance disappears — and a year later I have a note that is right and no longer know what it rests on. That is precisely what the traceability described throughout this book is designed to prevent.

So the conclusion is not to pick a side. It is to record what each action has to carry:

- a **reference** calls for knowing when its source changes;
- a **copy** calls for saying where it comes from — which `Context`, on what date, and what was reworded along the way.

---

#### A POSSIBLE DEVELOPMENT

Another answer, still to come, will be technological: `Contexts` could notify their observers when their state changes, especially when a change affects the scope observed by another `Context`. The `Context` is a living subject, and many possibilities remain to be explored.

---

## The Rights That Reading Does Not Grant

A Context you read often can start to feel like a Context you partly own. Several times I have wanted to correct a note at the neighbour's directly — it was incomplete, I knew what to put in it, and it would have taken me thirty seconds.

That does not change the rule: what I have to tell it is a proposal, and the decision belongs to whoever receives it.

## Deleting Without Breaking Traceability

Deleting calls for a rule of its own, and I learnt it while clearing out a holding area.

When a note changes Context, the sources that produced it stay behind. They can find themselves alone, with no note attached to them any more.

The instinct is to tidy up. It is half right.

A source already read and turned into notes can be deleted: the knowledge it contained lives elsewhere. An unread source cannot.

- It carries material nobody has drawn out, and throwing it away amounts to losing what you have not even opened.

So the criterion is neither the age of the file nor the room it takes up.

- It is whether the source has been read, considered alongside everything that still cites it.

A condition goes with it, and it is not negotiable. Before deleting, check that no note, no vocabulary entry, no draft and

no line of the log still cites that source. Without that check, the rule breaks the very traceability it claims to serve.

### **A Divergence Is Not an Error**

There remains the case where two `Contexts` do not say the same thing about the same subject.

The instinct is to see an inconsistency in it, and so to decide quickly in order to restore order. That is often wrong.

A divergence may reflect a difference in scope: each can be right within its own boundaries. It may be a hypothesis still open, which one of them is looking into and the other knows nothing of. There may be two readings that both hold, because they look at the subject from two places.

Such a divergence deserves to be kept as it is, and that decision belongs to the human. What the system has to guarantee is that its existence is known.

A forgotten divergence always ends up making itself felt at the worst moment: when someone leans on one of the two versions without knowing that the other exists. That is what happened to me over the closing rule, except that it was not two readings but a single one that had aged.

So keeping the network reliable comes at a cost, and I would rather name it than discover it at review time. Connecting two `Contexts` does not remove the work of staying alert: it makes it possible, and it shifts it onto the human.

That, in the end, is what this book has been telling from the start.

---

**WORTH REMEMBERING — THE PRICE OF THE NETWORK**

A reference can age without warning, and a copy can lose its provenance: each action creates an obligation, and that obligation must be recorded. Reading does not authorise writing; what you have to say to the neighbour is a proposal, never a correction. A source already turned into notes can go, a source never read never can — and nothing is deleted without checking who still cites it. Two `Contexts` that contradict each other are not necessarily wrong: the serious problem is failing to notice.

---

# Before closing

---

Closing



## Closing

---

The book stops here; the subject does not stop with it.

The epilogue says where this work stands, what it has only touched on and where further work will be published. The section on related approaches places the `Context` among those that came before it or that surround it, so that you know what it borrows and what it does not take up.

## Epilogue

---

Thank you for making it this far.

I tried to make this book as readable as I could, and I did not entirely manage it. Some passages are drier than I would have liked, places where the mechanics take over from the story. You stayed with it anyway: that is a gift I do not take lightly.

### **This Book Has Only Scratched the Surface**

For all its thickness, it treats the `Context` only at surface level. Several parts stay thin conceptually, and I know it as I publish them. Atomic notes, for instance: depending on how I implement them, the results are very uneven. What works in a `Context` needs adjusting as soon as you move to an LLM Wiki, where the material has neither the same volume nor the same use.

Those areas deserve to be strengthened. They have not been here because a book has to stop somewhere, not because the subject is closed.

### **Work That Will Stay a Side Project**

The `Context` will go on developing, but always as a side project. It is something done alongside my job, and I have already given it far too many evenings. I will come back to it — I always come back to it — but it is time to ease off and enjoy life.

---

### **What I Would Like to Explore Next**

I did do the prior-art search. It turned up concepts very close to mine, sometimes in fields far from product work and knowledge management, often in other languages — Japanese and Chinese in particular, whose work remains largely invisible from the French- and English-speaking world. I have written an article about that search; it will appear on the blog. What is missing is the next step: analysing that work.

The names are identified, the papers are there, and I have not read them. That is decades of research carried out by hundreds of people on questions I took on alone, in the evenings. Going through it properly would strengthen this framework far more surely than carrying on inventing it.

### **What Comes Next Will Not Be a Volume II**

This book served to crystallise the concept. That was its job: to gather what was scattered, make it transmissible, give it a form stable enough to be discussed.

For some time, developments will be published on [malorean.net](http://malorean.net), not in a volume II or in a new version of this book. An article can be corrected, extended, contradicted. A book fixes things in place — and what is described here is not yet ripe enough to be fixed in place twice.

If you want to follow what comes next, that is where it will be written.

## Kinships and Neighbours

---

Context did not appear in an intellectual vacuum.

Most of the mechanisms it is made of have relatives, some of them very old, in knowledge management, note-taking methods, software engineering, decision-support systems, cognitive science and organisational memory.

Some of those influences were conscious from the start.

Others only came to light after the method had been designed, during a prior-art search carried out when the book was almost finished.

Two things have to be told apart.

An **influence** is a method or an idea I knew about and whose principles I deliberately took up or adapted.

A **kinship** or a **neighbour** is an earlier system that solves a close problem, sometimes with strikingly similar mechanisms, without having directly influenced Context.

The distinction matters. Two ideas can be close without one deriving from the other.

### **Zettelkasten: Knowledge as a Network of Reusable Units**

The kinship with the Zettelkasten is probably the most obvious one.

In Niklas Luhmann's system, a note is not simply filed in a folder matching the project of the moment. It joins a lasting

system of notes linked to one another and can be reused in new lines of reasoning.

Context shares that idea strongly: knowledge produced during a piece of work should not disappear with its deliverable.

The atomic notes of the Context clearly belong to that family.

The main difference lies in their status. An atomic note has to keep its provenance and sits inside an architecture that explicitly separates sources, contradictions, decisions, current state, synthesis and deliverables.

The Zettelkasten is therefore a very strong relative for the **gradual building of a lasting body of knowledge**, but not for the system as a whole.

### **PARA: Organising Information Around Action**

PARA, from Tiago Forte, is one of the conscious influences on Context.

PARA organises information into four categories: Projects, Areas, Resources and Archives. One of its important insights is that organising information should serve action rather than produce an abstract classification of the world.

Context keeps something of that insight.

A Context does not exist because a subject exists. It exists because a human is pursuing a mission.

So the boundary is not: “everything I know about databases”.

It could be: “choosing a data architecture for product X”.

The difference is that the Contexts themselves become autonomous units of work, with their own state, sources, knowledge, decisions and life cycle.

### **Domain-Driven Design: The Boundary Before the Content**

The very word `Context` naturally brings to mind the **Bounded Context** of Domain-Driven Design.

The kinship rests above all on one idea: a piece of information or a concept only has meaning inside a certain boundary.

The same word can mean different things in two different contexts without any need to merge them.

`Context` transposes that insight.

But instead of making the boundary depend on a coherent business model, it makes it depend on a **human mission**.

Two Contexts can therefore hold different understandings of the same subject because they are pursuing different goals.

### **Git: Divergence, Autonomy and Reintegration**

Git is a more directly technical influence.

When an exploration becomes substantial enough, it can be better to let it develop in a separate Context rather than disturb the stable space.

That logic resembles a branch.

The exploration can diverge, produce its own sources and knowledge, and then some of its conclusions can be reintegrated into the original space.

But, as with Git, integrating does not necessarily mean merging everything.

That insight also shaped the relations between Contexts: local autonomy, possible divergence and selective integration.

### **Project Memory: Keeping the Why of the Project**

The work on **project memory** is a kinship I had not identified when designing Context.

The problem it poses is almost exactly the one that appears when a piece of work lasts several months: organisations keep the final documents easily enough, but they keep far less well the context that explains how and why the project arrived at that result.

Why this decision?

What did we try before?

What constraints existed at the time?

What would a newcomer need to understand before carrying on?

Context answers the same questions with several distinct memories: the current state, the history, the decisions, the synthesis and the handoff.

The functional closeness is therefore very strong.

### **Experience Factory: The Deliverable Is Not the Knowledge Asset**

The Experience Factory, developed around the work of Victor Basili and trialled notably at NASA, is an even more important relative.

It separates the organisation that carries out a project from the one that turns the project's experience into knowledge reusable by the projects that follow.

That idea connects directly with a fundamental distinction in Context:

***the deliverable is not the capital.***

A project can produce an excellent result while leaving almost nothing behind it.

Conversely, its experience can be analysed, transformed and kept in a way that increases the ability to solve the problems that come next.

Context builds that accumulation directly into the space where the work is done.

### **Knowledge-Centered Service: Producing Knowledge While Working**

Knowledge-Centered Service — KCS — is probably one of the most interesting neighbours of Context.

Its principle is that documentation should not be a separate activity carried out after the work.

When a problem is solved, existing knowledge is searched for and reused. If it is imperfect, it is improved. If it does not exist, it is created.

Simply working thus becomes a mechanism for creating and maintaining knowledge.

That is exactly one of the ambitions of Context:

*every use of the system should be able to increase its information capital.*

KCS, though, is designed mainly for service and support environments, where `Context` aims to provide a generic structure for many kinds of mission.

### **HWMSE: Humans, Machines and Knowledge in One Space**

The most surprising relative found during the search probably comes from China.

The **Hall for Workshop of Meta-Synthetic Engineering**, or HWMSE, was developed around the work of Qian Xuesen from the early 1990s onwards.

It is designed to handle complex problems by explicitly combining:

- human experts;
- machine systems;
- knowledge systems.

More interesting still, the work on HWMSE does not only talk about using existing knowledge. It also talks about **knowledge creation** and about a knowledge system specific to the problem under study that keeps growing while it is used.

The closeness to `Context` is therefore considerable:

- an activity aimed at a problem;
- human-machine collaboration;
- resources and models drawn on during the work;
- knowledge produced by the activity;

- capital that accumulates.

HWMSE does not, however, have the same operational grammar as `Context`. You will not find there, in this form, the separation between source, atomic note, state, synthesis, contradiction, handoff and deliverable.

But among the systems studied, it is probably one of the closest architectural neighbours.

### ScholOnto and SWAN: A Claim Is Not Its Source

Another strong kinship shows up in systems devoted to scientific discourse.

ScholOnto and, later, SWAN separate scientific documents from the **claims produced out of those documents**.

A resource can serve as the origin of an assertion.

That assertion keeps its provenance.

It can be supported, challenged or contradicted by other elements.

That is extremely close to the principle behind the atomic notes of the `Context`:

*a source stays a source; the knowledge built from it is another object.*

Those systems also show that a claim recorded in a knowledge system should not automatically be treated as true. It has a provenance and a standing within a line of reasoning. That is a particularly strong kinship with the epistemic architecture of `Context`.

### Federated Systems: Reading Is Not Copying

Finally, some ideas about the relations between Contexts have surprisingly old relatives in federated database systems.

Those architectures have to let several autonomous databases cooperate without giving up their autonomy.

One system can expose certain information.

Another can consult it remotely or decide to keep a local copy.

And as soon as a copy is created, the classic problem appears: what happens when the original changes?

That is exactly what this formula sums up in Context:

***reference = dependency; copy = responsibility.***

The computing mechanisms had therefore existed for a long time.

Context transposes them into rules for governing knowledge between human–AI spaces.

### A Synthesis Rather Than an Invention from Scratch

These neighbours make it hard to present Context as the invention of entirely new mechanisms.

- That is not what the history shows.
- Provenance existed.
- Atomic pieces of knowledge existed.
- Project memory existed.
- Cross-project accumulation existed.
- Creating knowledge while working existed.

- Environments combining humans, machines and knowledge systems existed.
- Sourced and contestable assertions existed.
- Autonomous architectures able to share or copy information existed.

What seems to characterise `Context` more is the way those mechanisms are **selected, combined and standardised within a single method**.

The method gives them a common grammar:

- sources;
- atomic notes;
- contradictions;
- decisions;
- state;
- history;
- synthesis;
- actions;
- handoff;
- deliverables;
- relations between Contexts.

And above all, it applies that assembly to a very present-day problem: sustaining a lasting collaboration between a human and an AI without turning every conversation into an isolated, amnesiac episode.

`Context` therefore has a good many relatives.

---

But, as so often, knowing your relatives is mostly a way of understanding what makes your own work distinct.