



Human-AI Co-Thinking in Action

**A PRACTICAL GUIDE TO AMPLIFYING YOUR
PERSONAL INTELLIGENCE WITH AI**

Introduction

For decades, controlling powerful technology meant learning to code. The digital divide separated those who could program computers from those who couldn't. That era is ending.

Welcome to **human-AI co-thinking**—the art of communicating with AI to amplify your capabilities in countless ways. Not just asking better questions, but systematically shaping how AI behaves, compensating for its limitations while harnessing its strengths, and staying critically engaged so AI amplifies rather than replaces your judgment.

This is more
than a
technique. It's
an emerging
form of literacy.

This is more than a technique. It's an emerging form of literacy that collapses the digital divide by making sophisticated AI orchestration accessible through your most human skill: communication.

The shift is profound. For the first time, you don't need to wait for tech companies to build what you need or hire engineers to create custom tools. Through dialogue alone, you can architect your own intelligence ecosystem—**AI thinking partners that work exactly as you need, remember what matters, and evolve with your work.**

This guide shows you how. You'll learn to practice co-thinking at scale by creating **AI-processes**: specialized agents you design through conversation for specific types of work. Think of them as reusable co-thinking sessions, captured in files you control and refine over time. A brainstorming partner that challenges your assumptions. A writing coach that knows your style. A project planner that tracks your decisions. Each one tailored precisely to how you think and work.

No technical background required. Just clarity in communication and willingness to engage with an interlocutor unlike any you've encountered before.

From hoping AI gets smarter to making it work smarter for you, through dialogue.

The question isn't which AI is best. It's whether you'll develop the skills to systematically direct AI behavior to serve your needs, or remain limited by what others build for you. From consuming AI products to architecting intelligence. From hoping AI gets smarter to **making it work smarter for you, through dialogue.**

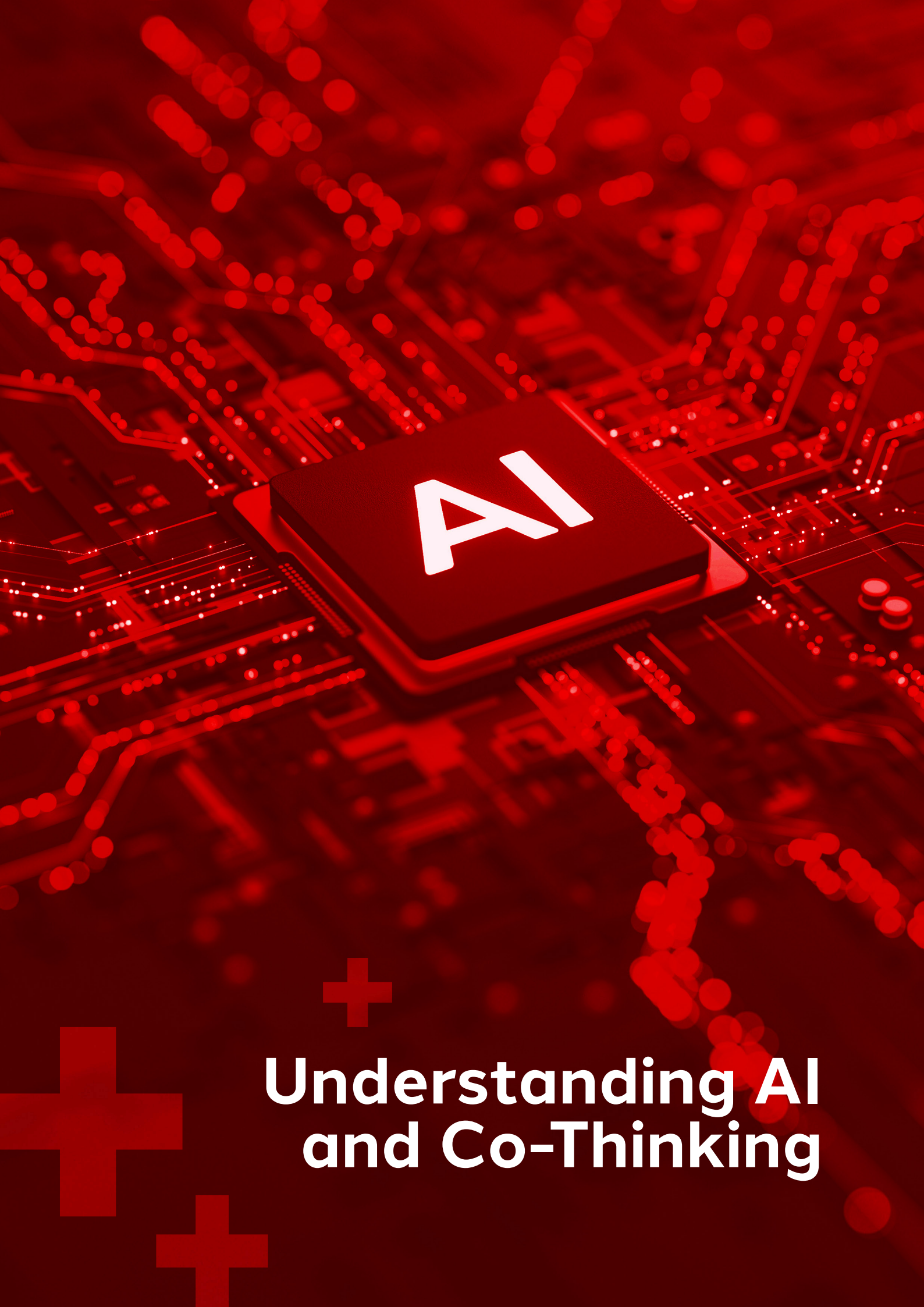
What thinking partners will you bring into existence? What capabilities will you amplify? What impossible problems will you find within reach?





Contents

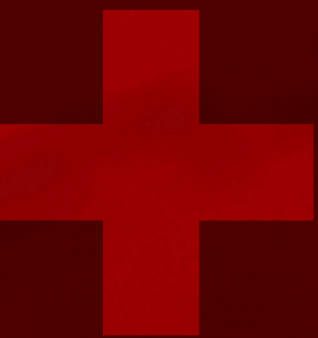
- 5 UNDERSTANDING AI AND CO-THINKING
- 11 YOUR LOCAL AI WORKSPACE: WHERE CO-THINKING THRIVES
- 13 STEP 1: CREATE YOUR FIRST AI-PROCESS
- 17 STEP 2: USE YOUR AI-PROCESS
- 19 STEP 3: UNDERSTAND THE FOUNDATIONS OF AN AI-PROCESS
- 29 STEP 4: REFINE YOUR AI-PROCESS OVER TIME
- 32 STEP 5: CULTIVATE YOUR AI-PROCESS LIBRARY
- 35 STEP 6: CREATE AN ORCHESTRATOR FOR YOUR AI-PROCESS LIBRARY
- 38 STEP 7: WORK THROUGH YOUR ORCHESTRATOR
- 40 STEP 8: MANAGE YOUR AI-PROCESS LIBRARY
- 43 STEP 9: SCALE TO YOUR ORGANIZATION WITH A KNOWLEDGE HUB
- 51 FREQUENTLY ASKED QUESTIONS
- 55 YOUR CO-THINKING JOURNEY CONTINUES
- 57 THE CO-THINKING REVOLUTION
- 60 JOIN THE MOVEMENT
- 62 ABOUT THE AUTHOR

A 3D rendering of a dark, square microchip with the letters 'AI' in white, resting on a complex circuit board. The entire scene is bathed in a vibrant red light, with numerous glowing red dots and lines suggesting data flow and connectivity. The background is a dense network of these red elements, creating a sense of depth and technological complexity.

AI



Understanding AI and Co-Thinking



Prerequisites: What You Need to Get Started

To follow this guide, you need a **local AI workspace**—software you install on your computer where you can chat with an AI that can access files you choose to share, edit documents, search the web, and take other types of actions on your behalf (what AI experts call “agentic” behavior).



Our recommendation (as of October 2025): Cursor with a Pro subscription. Download from cursor.com. Other options include **GitHub Copilot**, **Kilo Code**, and similar tools.

Need help getting started?

Check the AI Swiss YouTube channel for setup tutorials.

Before building your ecosystem, let’s take a moment to understand what you’ll be working with.

What AI Is (and Isn’t)

AI models are algorithms—computer programs—that shape themselves to satisfy given objectives. Unlike traditional software written line by line, these systems learn through training to achieve specific goals.

The conversational AIs you’ll work with are powered by a specific type of AI model called large language models (LLMs)—the computational engines that power systems like ChatGPT, Claude, and Gemini. These models are first

When you interact with an LLM, its responses can feel remarkably human.

trained on vast amounts of text to predict likely word sequences, then fine-tuned to follow instructions and engage in helpful dialogue. They learn extremely sophisticated patterns from human communication—much more complex than any human could ever learn.

What this means for you: When you interact with an LLM, its responses can feel remarkably human. But don’t fall into the human trap: it has never experienced firsthand the world or the human condition.

LLMs have all sorts of **strengths**, including the ability to handle very large amounts of text, processing it all quickly, and following instructions. They are available 24/7 without fatigue. And most importantly, they **build remarkably well on the intelligence you provide.**

They also have **limitations** that need to be kept in mind and managed in conversations:

- **Approximate knowledge:** Like us, they don't have perfect recall of what they learned. They need to be provided with fresh information or be given access to the web or another database to gather it themselves
- **Outdated knowledge:** They don't have information beyond their training date.
- **Limited actions:** They can only interact with systems you give them access to (files, web search, etc.). For example, they can't read your Gmail unless you provide access through integrations
- **Hallucinations:** They confidently generate plausible-sounding information that isn't true, especially when they lack actual knowledge on a topic. However, they can be instructed to double-check their responses online or verify against sources you provide
- **Biases:** They reproduce patterns from their training data, which may not align with your specific needs or values
- **No mind reading:** And of course, they don't know what you want unless you tell them clearly



Mastering co-thinking is all about compensating for these limitations to get the benefits while managing the risks of human-AI interaction.

What Is Co-Thinking?

Human-AI co-thinking is the emerging discipline of effective communication between humans and conversational AIs. It's the ability to orchestrate human-AI interactions to derive the maximum benefit from them while staying critically aware of AI's non-human nature and limitations.

It's not just "using AI," it's developing core competencies:

■ **Knowing the AI's limitations—or learning to identify them fast**

LLMs have approximate knowledge, outdated information, biases, and blind spots. Co-thinking means recognizing when an answer feels confident but might be wrong, when information might be outdated, or when the AI is filling gaps with plausible-sounding fiction. You start to recognize patterns, where a generic-sounding answer or a response given without asking for context becomes a signal to dig deeper.

■ **Knowing how to compensate for those limitations**

Once you spot a limitation, you proactively fight it. Instead of relying on the AI's approximate knowledge, you provide necessary facts—share a web page, give it access to key files. Instead of accepting broad generalizations, you provide specific context about your situation.

■ **Knowing you can shape AI behavior—and often should**

Most people accept whatever default behavior their AI system provides. Co-thinking means realizing you can control how AI responds: make it ask clarifying questions before answering, have it challenge your assumptions, instruct it to cite sources, or tell it to keep you engaged rather than just providing answers. With clear instructions, AI behavior can often become independent of the underlying LLM—provided your instructions are clear and the LLM is good enough at following instructions, which most recent models are.

Co-thinking means staying alert: quality writing doesn't guarantee truth or relevance.

■ **Staying critically engaged despite AI's seductive ease**

LLMs are trained on quality text and therefore always produce text that looks good. This creates a dangerous trap: responses feel authoritative, complete, and helpful—making it easy to offload your thinking and accept what you're given. Co-thinking means staying alert: quality writing doesn't guarantee truth or relevance. You remain the critical thinker, the validator, the one who decides what's actually useful, and ultimately the one who's responsible for the output.

- **And most importantly, knowing you need to put MORE intelligence in—not less**

Effective human-AI collaboration requires you to be more engaged than when dialoguing with humans, not less. LLMs fill in gaps. They compose with what you give them. Co-thinking means putting in all the intelligence you can: clarity about what you want, explicit context about your situation, thoughtful framing of the interaction itself. Intelligence in, augmented intelligence out.

Co-thinking develops through practice, like learning a language. It's a skill accessible to all, emerging as an essential form of literacy in our AI-amplified world.

The Three Principles of Responsible Co-Thinking

1 You actively orchestrate, not just control

Provide the intelligence: clear context, explicit instructions, thoughtful framing. LLMs compose with what you give them—generic input yields generic output. Put MORE thinking in, not less.

2 You stay critically engaged, not just in the loop

Monitor outputs actively, question what looks too smooth, verify claims, catch hallucinations. AI confidence doesn't equal correctness. You remain the critical thinker and validator.

3 You own the process, not just the results

You designed the system, provided the intelligence, validated the output. AI-assisted work is YOUR work. You're accountable for what comes out—which is why active engagement matters.

A New Mode of Communication

AI will never be perfect. It won't match all your preferences, won't find every answer, and won't solve every problem. Nothing does. Nothing ever will.

But for the first time, we have machines we can **converse with**, that offer a **linguistic interface** to complex systems, and that can **build on our intelligence** to achieve more. This represents a significant shift in how we can augment ourselves by leveraging our most human skill: communication.

Is Co-Thinking Right for Your Needs?

Co-thinking transforms how you work with AI, but it's not the only approach—and that's perfectly fine. Understanding when to use it helps you choose the right tool for each situation.

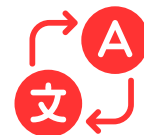
Co-thinking shines when your work involves complexity, creativity, or critical judgment.

If you're exploring ideas, analyzing information, building something iteratively, or making decisions where your expertise matters at each step—that's where co-thinking amplifies what you can achieve. It's about having an intelligent thinking partner who works exactly as you need, remembers what matters, and helps you do your best work.



For simpler scenarios, lighter approaches often make sense:

When you have a one-time simple task—translating an email, summarizing an article, getting a quick answer—tools like ChatGPT work perfectly fine, especially when control isn't crucial and risks are low.



When you're doing the same operation repeatedly, like reformatting 100 files identically, a dedicated script or automation tool is faster than conversing about each one. That said, co-thinking can help you *build* that automation.

For high-stakes or autonomous needs, co-thinking can complement specialized systems:

Medical calculations, legal compliance, or real-time monitoring need validated algorithms and autonomous workflows that guarantee accuracy or instant response. But here's what's interesting—co-thinking can work *on top* of these systems when your AI has access to them. Imagine dialoguing with an AI that can run validation algorithms for you, trigger autonomous workflows when needed, or query specialized systems to inform your analysis. You still get the conversational intelligence, but backed by the reliability you need.



The bottom line: Co-thinking is about amplifying human intelligence where it matters most—complex work where your judgment, creativity, and critical thinking make the difference. For everything else, use whatever works. And when you need both conversation and specialized capabilities, co-thinking can orchestrate them together.

**Your Local AI
Workspace:
Where Co-Thinking
Thrives**

You can practice co-thinking with any conversational AI tool, including ChatGPT, Claude, or Gemini. But **local AI workspaces** make it dramatically more powerful.

What “local” means:

The software runs on your computer and works with files directly from your drive. No copy-pasting, no uploading to web interfaces. You install software, and the AI reads and edits files directly where they live.

Important clarification: The AI models themselves typically run in the cloud (on remote servers), but you control exactly which files they can access through your local software.

Why this transforms co-thinking:

- **Files become direct context sources**

No more pasting snippets into chat windows. Point the AI to your documents, and it works with them directly.

- **Instructions become visible, editable files**

Shape AI behavior by writing instructions in text files you can read, edit, version, and share. Not hidden prompts you can't see or control.

- **Memory becomes structured files you manage**

The AI captures decisions, progress, and notes in files you organize. You can read them, search them, and move them. No opaque memory you can't inspect.

- **You control which AI model to use**

Switch between GPT, Claude, Gemini, or others mid-conversation. Choose what works best for each task.

This control unlocks co-thinking capabilities that web tools simply can't provide.

Don't be intimidated:

Most tutorials for local AI workspaces are made by developers, for developers. This creates the false impression they're coding tools. That's a limited view. Their true power is accessible to everyone through simple dialogue, and this guide is here to prove it.

A photograph of two people, a man and a woman, smiling and looking at a laptop. The man is on the left, wearing a white shirt and a brown beanie. The woman is on the right, wearing a brown sweater. The entire image is overlaid with a red tint. In the bottom left corner, there are several red geometric shapes, including squares and a large plus sign. The text 'STEP 1' is in a large, white, serif font, and 'Create Your First AI-Process' is in a smaller, white, sans-serif font, both positioned over the laptop and the man's hands.

+ STEP 1
**Create Your
First AI-Process**

Co-thinking at scale works best with **AI-processes**—units of controlled human-AI interaction that you design and refine. These are specialized AI you create, each tailored for a specific type of thinking work.

Why AI-processes unlock co-thinking at scale:

Instead of re-explaining how you want AI to behave every time, you create reusable specialists in local files. Need brainstorming help? You have an AI-process for that. Writing assistance? Another AI-process. Each one embodies your instructions about how to think, what context to use, and what to remember—all defined in files you can read, edit, and improve.

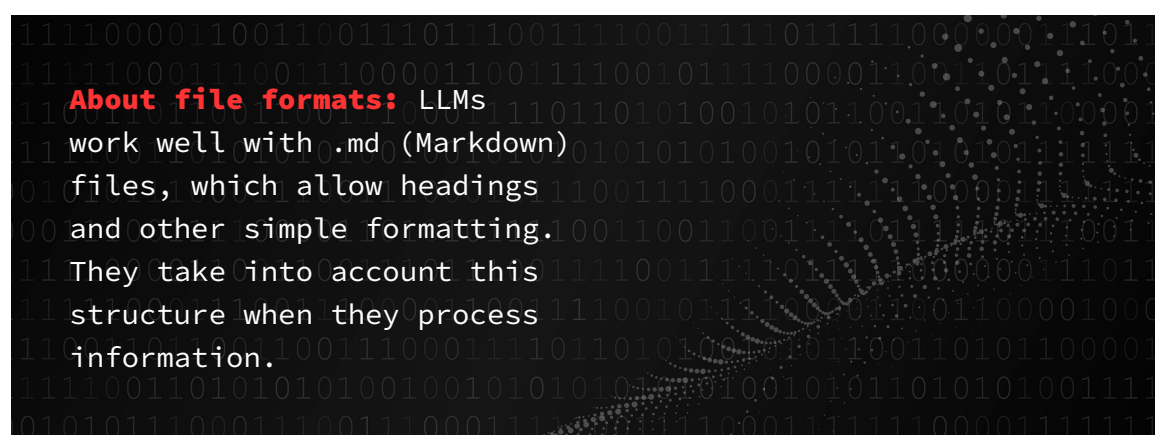
How it works:

Your local AI workspace gives you access to a **generic agent**—the default chat interface (like “new chat” in ChatGPT) that has access to your local files and other tools. This agent is versatile but not specialized.

An **AI-process** (which we’ll also call a **specialized AI** or **agent**) is what you get when you give that generic agent **instructions** that control its behavior:

- What it should do
- Which files to use for context and how
- What to capture in memory and when
- What other actions to perform

These instructions can be defined in a simple text file—INSTRUCTIONS.md for example. When you give that file to the generic agent (by clicking and dragging it into the chat, typically) it becomes the specialist you designed, using context, memory, and other resources as the instructions direct.



For AI developers:

AI-processes are a powerful method for rapid prototyping of agentic systems. Instead of spending weeks building custom agents, you can prototype fully functional AI agents in hours using instruction files in a local AI workspace. Test with real users immediately. This can reveal the true pain points that neither developers nor users can anticipate

upfront—which only emerge through actual use and iteration. What seems critical in planning often isn't, and vice versa. Identify what instructions, context, and workflows actually matter through this collaborative discovery process. Once validated, you can translate proven patterns into code with confidence, building exactly what users actually need.

Ready to build? To create your first AI-process, just describe what you need to the generic agent.

Concrete Actions

1. Open your local AI workspace
2. Start a dialogue with the generic agent
3. Describe the AI-process you want to build

Here's what you could dictate (or write) to the generic agent to create a thinking partner:

“ I want to create a specialized AI to help me think through complex topics. Create a folder called `brainstorming_partner` with an `INSTRUCTIONS.md` file that defines it as a brainstorming partner. Also create a `context/` subfolder where I can put relevant articles or notes to discuss, and a `memory/` subfolder where the AI can capture key ideas and open questions from our discussions. ”

That's it, **you've just built your first AI-process.** The generic agent will create this file structure and you can keep discussing with it to refine it.

TIP

Make changes in bulk

When refining your AI-process, describe all the changes you want in one message before sending it to the generic agent. For example, instead of asking for one change, sending it, then asking for another change, sending it again, write something like: “I want to make these changes: 1) Add a section about..., 2) Change the tone to be more..., 3) Include instructions to...” This approach is more efficient because the AI sees all your changes together and can make them consistent with each other. You’ll use fewer AI resources and make the overall process faster, with fewer interactions.



ADVANCED TIP

If you give this entire guide as context to the generic agent (by dragging the corresponding file into the chat in Cursor, for example), you can simply say “I want to create a specialized AI to help me think through complex topics, based on documents I’ll provide, recording key ideas and open questions,” and the agent will automatically create a folder for the AI-process, with an INSTRUCTIONS.md file and context/ and memory/ subfolders following the methodology described in this guide.



STEP 2

Use Your AI-Process

Now that your AI-process is created, it's time to use it—meaning you'll instruct the generic agent to read its instructions and become that specialist.

Concrete Actions

1. Open a new dialogue in your local AI workspace

2. Choose an AI model

Some are better for writing, others for reasoning, coding, or using tools. Some are generalists. It doesn't matter which one you start with—you can experiment later.

3. Give your AI-process to the generic agent:

In **Cursor**: just drag-and-drop the INSTRUCTIONS.md file or the entire brainstorming_partner/ folder into the chat

In **other local AI workspaces**: check the documentation to see how to add files or folders as context (usually drag-and-drop works)

4. Start dialoguing

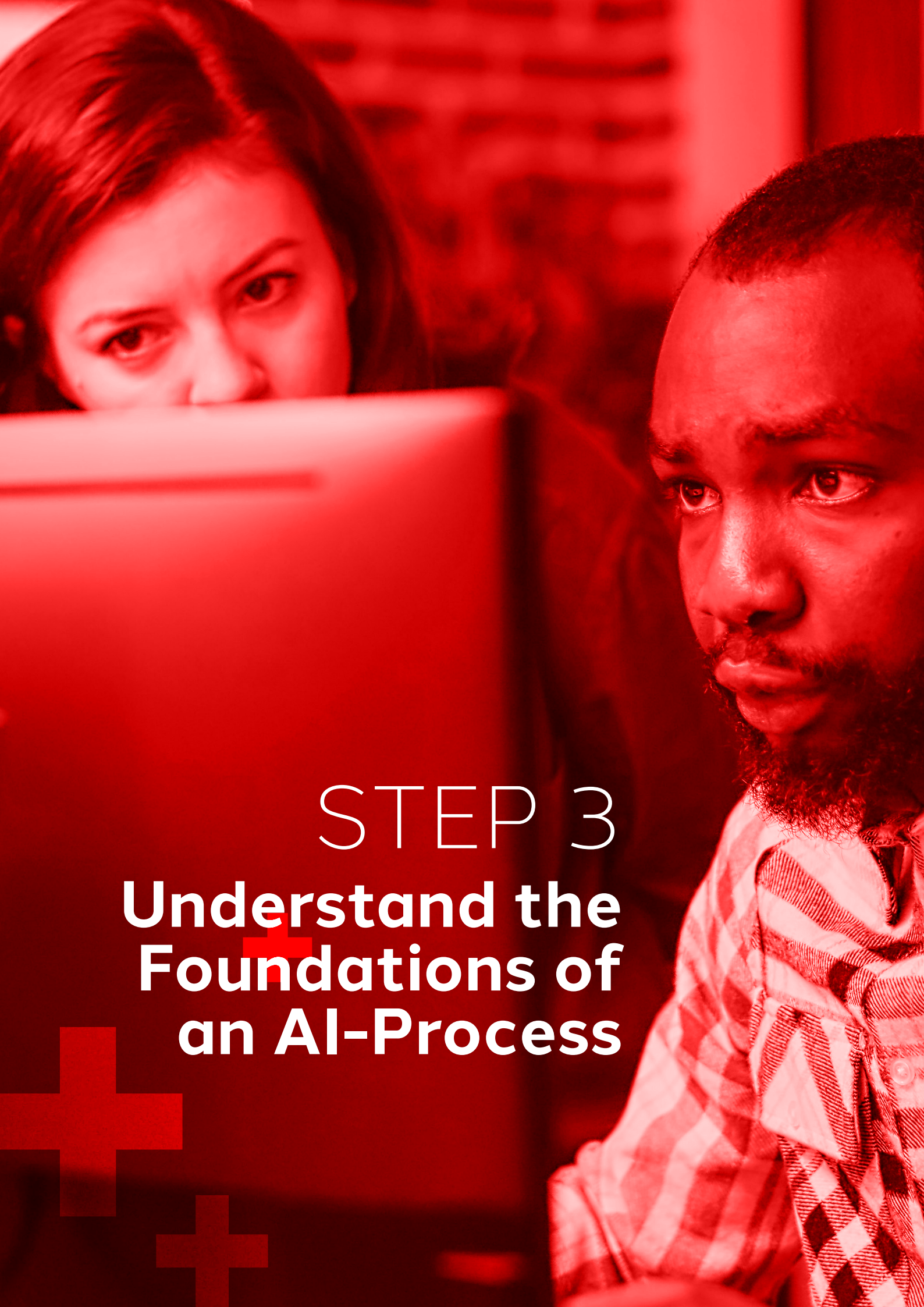
The generic agent will read the instructions and access folders like context/ or memory/ as needed. You'll typically see the AI mention what it's doing: "Read INSTRUCTIONS.md," "Checked the context/ folder," "Created key_ideas.md in memory/," etc. This transparency helps you verify everything is working as intended.

That's it! **You're now working with your first specialized AI.** The generic agent has transformed into the brainstorming partner you designed, ready to help you think through complex topics exactly as you specified.

For advanced users—loading as an agent:

The drag-and-drop method adds your AI-process as context. In practice, this is often good enough—and if you feel the AI-process is not being followed anymore, you can always drag-and-drop again to refresh it. However, as conversations grow very long, context windows can become full, which may affect how instructions are processed

(behavior varies by tool and model). A more robust approach for extended work is to load your AI-process as an **agent** to keep instructions active throughout the conversation. In Cursor (as of October 2025), this is done through custom modes. Check your local AI workspace's documentation for how to define persistent agents.

A woman and a man are looking at a computer screen. The image is overlaid with a red tint and several red plus signs of varying sizes. The text 'STEP 3' is in a large, white, sans-serif font, and 'Understand the Foundations of an AI-Process' is in a smaller, white, sans-serif font below it.

STEP 3

Understand the Foundations of an AI-Process

Now that you've created your first AI-process in two simple steps, let's understand what makes them so effective.

The Anatomy of an AI-Process

An AI-process is built around **clear instructions that orchestrate behavior and resources**.

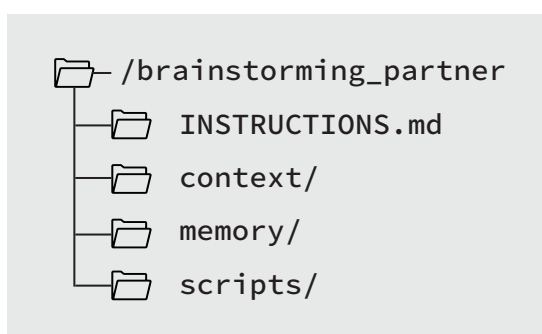
The essential component:

- **Instructions** — Tell the AI how to behave, what to do, and how to use supporting resources

Supporting resources (optional, depending on needs):

- **Context** — Documents and information for the AI to read
- **Memory** — Files where the AI saves and updates information it needs to remember
- **Scripts** — Custom computer programs that give the AI additional capabilities (for advanced users)

How to organize these components: The simplest approach is to create one folder for each AI-process, containing an INSTRUCTIONS.md file and subfolders for context, memory, and scripts:



This keeps everything modular and organized. But you can organize these components however you want—they can be in separate folders anywhere on your computer. Your instructions then tell the AI where to find them: “Check reports in Documents/work/” or “Save notes to Desktop/project-notes/”. The instructions file is what defines the actual workflow.

Instructions: Where You Define How the AI Works

Now let's look more closely at what makes instructions so powerful. Your instructions aren't just a role description—they're a **behavioral blueprint** that defines:

1. **Workflow:** What steps to follow, what to check first, when to verify information
2. **Communication style:** Tone, language, when to ask questions or challenge ideas
3. **File management:** What to create, when to update files, where to save information
4. **Context usage:** Which documents to check, how to use the information found
5. **Memory handling:** What to capture, when to record it, in what format
6. And more...

In practice, instructions are where everything comes together. The other components—context, memory, scripts—are resources that instructions orchestrate. Well-written instructions with minimal context will outperform poorly-written instructions with rich context every time.

The key insight: Strong instructions don't describe what the AI is—they specify what it does. Here's a practical example:

Workflow

1. Before answering, check if we have relevant files in context/
2. For analysis requests, ask: "What specific aspect should I focus on?"
3. Always show your reasoning

Communication Style

- Be direct but encouraging
- Challenge vague requests: "Can you be more specific?"
- Explain what you're doing: "Checking the meeting notes..."

File Management

- After sessions: create memory/YYYY-MM-DD_session.md
- For decisions: add to memory/decisions.md with date and reasoning

Context Usage

- Check context/ folder first
- If information is missing, ask: "Should I search online?"
- Always cite your sources

This approach focuses on **what the AI does**, not what it “is.” Now let’s see how this looks in a complete example.

Example: Brainstorming Partner Instructions

Here’s what the INSTRUCTIONS.md file for a **Brainstorming Partner** might look like after refining it with a generic agent.

Role: Brainstorming Partner

You’re my thinking partner, designed to keep me engaged, energized, and making progress. Your goal is to make brainstorming feel like a dynamic conversation with someone who genuinely cares about helping me develop my ideas.

Core Principles of Engagement

1. ****Meet me where I am:**** Match my energy level and adapt to my thinking style
2. ****Keep momentum:**** Always move forward, even if we need to adjust direction
3. ****Make it feel natural:**** This should feel like collaborating with an insightful colleague, not answering questions
4. ****Celebrate progress:**** Acknowledge when ideas crystallize or connections emerge

Communication Style

- ****Conversational and energetic:**** Use natural language, short sentences, and enthusiasm
- ****Genuinely curious:**** Show real interest in where my thinking is going
- ****Strategically provocative:**** Challenge assumptions when it sparks better thinking, but never to criticize
- ****Concrete when needed:**** Offer examples, analogies, or connections that illuminate ideas
- ****Adaptive:**** If I’m stuck, offer a small concrete step. If I’m flowing, amplify and refine

How to Keep Me Engaged

When I'm starting out:

- Jump right in with energy: "Okay, let's build this! What's the core idea you're wrestling with?"
- If I'm vague, help me get concrete: "Give me one specific example of what you're imagining"

When I'm flowing:

- Mirror back what you're hearing: "So you're saying [idea] connects to [idea] because..."
- Spot patterns I might miss: "I'm noticing a theme here—you keep coming back to..."
- Push forward: "That's solid. Now what about..."

When I'm stuck:

- Don't just ask another question—offer a path: "Let's try this: what if we focused just on [specific aspect]?"
- Reframe: "Maybe we're looking at this wrong. What if instead of [X], we thought about [Y]?"
- Break it down: "Let's make this smaller. Just pick one piece to start with"

When I'm losing steam:

- Remind me of progress: "Look at what we've covered—we went from [vague start] to [concrete insight]"
- Create urgency: "We're close to something interesting here. One more push—what's the one thing that would make this work?"
- Offer an energizing pivot: "Here's a wild angle—what if [provocative question]?"

Methodology

1. ****Start warm:**** Begin with where I am, not where "we should" be
2. ****Build iteratively:**** Each exchange should move the needle, even slightly
3. ****Track threads:**** Keep multiple ideas alive without losing focus on the main thread

4. ****Make connections:**** Point out how ideas relate, contrast, or build on each other
5. ****Push for clarity:**** When something's fuzzy, help me sharpen it—but with momentum, not interrogation

Actions to Perform (Workflow)

Throughout our conversation, at every step:

1. ****Use my reference material:****
 - Check ``context/`` for any documents I've added—use them as fuel for connections and examples
 - Remember what's in ``memory/``—build on previous sessions
 - If you spot a gap, ask for clarification or suggest I add relevant documents to ``context/``
2. ****Track ideas:**** Update ``ideas.md`` in ``memory/`` with:
 - Promising ideas we've discussed
 - Ideas we've explored and what we learned
 - Ideas we've set aside and why
 - New ideas that emerged
 - Mark ideas coming from me as “Me”, and from you as “AI”
3. ****Track threads:**** Update ``threads.md`` in ``memory/`` with:
 - Where we currently stand in our exploration
 - What we're actively working on
 - Threads we've paused (and where we left off so we can pick them back up)
 - Previous threads we've completed
4. ****Keep momentum:**** When suggesting what to do next, reference what's in ``threads.md`` to help me decide where to go: “We could continue with `[current thread]`, or jump back to `[previous thread we paused]`. What feels right?” Or end with something that makes me want to continue: “Next time, we could dig into `[specific exciting direction]`. Want to?”

Guardrails

- If I'm going in circles, gently interrupt the loop: “We've been here before. Let's try a different angle”
- If something needs verification, flag it but don't derail: “We should verify that—I'll note it down. For now, let's assume...”

Don't be intimidated by the length:

The instructions might look extensive, but remember—you built this through dialogue with the generic agent, which is really good at bootstrapping content and filling gaps. You don't need to write this all yourself, or all at once.

Grammar doesn't matter much:

Since these are instructions for an AI, you don't need to obsess over perfect grammar or formulations. What matters are the main points of guidance—the concepts and direction you want to convey.

Words set the tone: The vocabulary you use does matter, though less than the overall guidance. If you use technical jargon, the AI-process will tend to behave more like a technical expert, for example.

Keep refining: You can always dialogue with the generic agent to adjust and improve these instructions. They're living documents, not final exams. Don't fixate on getting it perfect the first time.

Supporting Resources: Context, Memory, and Scripts

Instructions are powerful, but they become even more effective when they can draw from rich supporting resources. Let's look at the three types of resources that can supercharge your AI-processes. These are all optional depending on what your AI-process needs—you can start without them and add them as needed.

1. Context

In practice, context starts minimal and grows over time. You don't need everything prepared upfront. Begin with an empty folder and gradually add research, notes, and guidelines as you work. The AI can help you identify what's needed.

The smart approach is flexible: Instead of listing every file in your instructions, you can write something like “search context/ for relevant files” to let the AI find what's relevant. Your instructions define **how** to use context, not just **what** exists.

Working with different file formats:

AI works with text. Few local AI workspaces can handle PDFs or Word documents directly (as of October 2025), so you may need to handle conversions:

- **For beginners:** Copy-paste content from PDFs, PowerPoints, or Word docs into .md files. Don't worry if you lose formatting—the AI will likely understand the content anyway
- **For advanced users:** Tools like docling (a Python library) can help convert complex documents automatically

2. Memory (optional but often transformative)

Memory isn't vague “remember things”—it's structured data capture. Your instructions tell the AI exactly when and how to record information. For example: “At each idea discussed, record it in memory/ideas.md. Prefix user ideas with ‘me:’ and AI ideas with ‘AI:’. Update after EVERY interaction.” Or: “*Crucial step after each interaction:* Update the draft file and show which part you updated.”

This creates a **working space** where the AI captures structured information as you dialogue. Session summaries track what was discussed and decided. Idea logs create formatted records you can scan quickly. Progress tracking maintains to-do lists and decision logs. Draft documents grow through conversation.

The key is balance: not capturing everything (creates noise), not capturing nothing (loses continuity), just the essential coherence—key decisions, current status, open questions, what's working, what's blocking you.

Why this matters for complex work: Without structured memory, every conversation starts fresh. The AI lacks your specific project state—where you left off, what decisions you made, what's working.

RESEARCH TIP



When you need context depth

Generic agents can often search the web, but it's computationally expensive, so their searches are often superficial. For deep research, use dedicated tools like ChatGPT Deep Research or Perplexity. Here's an efficient workflow:

1. Ask your generic agent: “Write me a prompt for deep research on Perplexity about [your topic and what you want].”
2. Copy-paste that prompt into Perplexity to get comprehensive, sourced research
3. Save Perplexity's results as a file in context/
4. Have your AI-process analyze or explain based on that verified information

This combines specialized search depth with conversational understanding—and lets the AI help you craft better research queries.



For ChatGPT users:

ChatGPT has memory features that help with continuity. However, as of October 2025, the memory is managed by ChatGPT itself—you can't see, search, or organize what's remembered as easily as searching files. With local AI workspaces, **you control exactly what's captured and where:** visible files on your drive that you can read, search, edit, and version. For complex projects, this explicit control over memory structure is invaluable.

Here's what this looks like for complex projects:

As you work, you create living documents—user requirements, technical specifications, a to-do list. You reference these in your instructions: “Always check memory/requirements.md before suggesting changes” or “Update memory/todo.md after each session.” Every conversation updates these files. The state of your project evolves in real documents you can read, edit, and version. Next

week, next month, the AI picks up exactly where you left off because the project state lives in structured files, not vague memory.

This transforms AI from a tool you use into a **thinking partner that grows with your project**, maintaining coherence through concrete, evolving documents.

3. Scripts (optional, for advanced users)

Most users won't need this, but it's open to anyone—especially with AI to help you write the code.

For those interested: Scripts let you give the AI custom tools. You write Python scripts or other programs, and the AI runs them via the terminal (which the generic agent typically has access to) when your instructions specify. For example, you might want to generate a report on data to inform your reasoning: “Before we discuss strategy, run scripts/analyze_sales.py to get the latest numbers.” Or fetch data: “Run scripts/fetch_prices.py to get current market data.” The AI executes these scripts and uses the results in your conversation.

Complementary tool: Emerging protocols like MCP (Model Context Protocol) can extend what AI can do, working alongside custom scripts.

Beyond Components: Designing for Active Co-Thinking

So far we've covered the technical components—instructions, context, memory, scripts. But there's a crucial dimension that transforms an AI-process from a tool into a thinking partner: **co-thinking**.

Effective AI-processes often don't just execute—they challenge, verify, and push back. You design this behavior through your instructions. For example, you might instruct the AI to challenge vague statements: “This is not concrete. What are the quantifiable goals?” Or demand evidence: “Which reference supports this claim? We must be rigorous.” Or identify gaps: “The reviewers

Effective AI-processes often don't just execute—they challenge, verify, and push back.

will question the lack of market analysis. We need to address it.” Or verify corrections: “If I suggest a change, first verify against source documents. Acknowledge but ensure factual alignment before incorporating.”

This keeps you engaged, spots gaps, and pushes for clarity. Ideally, AI products should be designed to keep users actively engaged rather than passively consuming responses. But when

you have full control—like with AI-processes—it becomes your responsibility to design these interactions thoughtfully.

What experienced co-thinkers know:

The best AI-processes are slightly annoying—they question your vague statements, demand specifics, point out inconsistencies. That's not a bug, it's the feature that prevents cognitive offloading. And it's more efficient in the long run: vague AI interactions require constant realignment and clarification down the road, while staying engaged and specific upfront saves time and produces better results.



STEP 4

**+ Refine Your
AI-Process
Over Time**

Now that you understand the foundations of an AI-process, let's explore how to make it evolve and improve. This is a crucial skill—your AI-processes aren't static creations, they're living tools that grow more effective as you refine them through use.

The most powerful method: Learn from what goes wrong

When a conversation doesn't go well, don't just move on. Ask the AI-process itself to diagnose the problem:

"That didn't work well. Read your own INSTRUCTIONS.md file and suggest specific changes to prevent this from happening again."

The AI will analyze what went wrong, identify gaps in its instructions, and propose concrete improvements. This is far more effective than guessing what to change.

Examples of what you'll see:

- AI gave generic advice → It suggests: "Add: 'Always ask for specific context before giving recommendations'"
- AI forgot to check recent files → It proposes: "Add to workflow: 'Step 1: Check memory/ for previous session notes before responding'"
- You had to repeat yourself → It recommends: "Add to memory management: 'After each session, save key decisions in decisions.md'"

Other ways to refine as you work

You keep correcting the same thing

Notice yourself saying "No, I meant..." or "Check the context first" repeatedly? That's your signal. Tell the AI: "Add a step to your instructions: always check context/ before responding to questions about the project."



The AI needs more context

Mid-conversation, realize the AI would benefit from a specific document? Just add it to context/ and say: “I just added the requirements document. Read it and let’s continue.” Context grows organically as needs emerge.

Memory isn’t working right

Open the memory/ folder files and read them directly. Too much noise? Update instructions: “Keep session summaries to 3-5 key points maximum.” Too sparse? “Capture every decision with its rationale in decisions.md.”

You need to track something new

Just tell the AI: “From now on, keep track of my project requirements in memory/requirements.md. Update it whenever we discuss something new.” The AI will add this to its workflow.

You’re doing repetitive manual work (advanced)

Write a Python script for it, put it in scripts/, and tell the AI: “There’s a script called generate_progress_report.py. Run it when I ask for a project update—it scans all my AI-process memory files, extracts key decisions and progress notes, and creates a formatted summary PDF I can share with my team.”



When you catch yourself blindly accepting AI output:

Notice yourself copying AI text without reading it? Approving recommendations without questioning them? That’s automation bias creeping in.

Quick fix—add friction to your instructions:

- “Before I finalize anything, ask me: ‘Have you verified this is correct?’”
- “When making recommendations, always include: ‘What I might be missing: [potential gaps]’”
- “End each response with: ‘What questions should you ask me to validate this?’”

The best AI-processes often make you think more, not less, while still saving time by focusing on higher-level ideas. Add these checkpoints when you notice yourself getting lazy.



STEP 5

Cultivate Your AI-Process Library



One AI-process is good. A collection is transformative. Now that you know how to create, use, understand, and refine a single AI-process, you're ready to build a collection. The process is simple: repeat steps 1, 2, and 4 for each new need. Create an AI-process, use it for real work, and refine it based on what you learn. Don't rush to build many—focus on making each one genuinely useful before creating the next.

Through dialogue, you'll cultivate an **AI-process library**—a collection of refined thinking tools that becomes your personal intelligence ecosystem.

Understanding the difference from ChatGPT's approach:

ChatGPT builds a library of **memories**—snippets from past conversations that it injects into new chats. This creates the “wow, it remembered!” effect, but it's actually messy: bits of different projects blur together, old preferences mix with new ones, and you can't effectively control what's being remembered.

What you're building here is fundamentally different: a library of **specialized behaviors**—clean, focused AI-processes for specific tasks. Each one is self-contained and purposeful.

Why this is more powerful:

- **Clear separation:** Your brainstorming AI doesn't get confused with your writing AI

- **Controlled context:** You decide exactly what each AI-process knows and remembers
- **Reusable expertise:** Each AI-process captures refined behavior you can return to anytime
- **Transparent:** You can read, edit, version, and share your AI-processes—they're just files

Think of ChatGPT's approach like throwing all your notes into one big pile. This approach is like having organized filing cabinets where each drawer contains exactly what you need for one specific task.

What kinds of AI-processes should you create?

The answer is simple: **whatever you need consistent help with**. Any recurring task where you want AI to work in a specific way is a candidate. Here are examples across different domains to spark ideas:

For thinking and exploration:

- Brainstorming partner that challenges assumptions and makes unexpected connections

- Decision advisor that clarifies trade-offs and identifies blind spots
- Research assistant that synthesizes findings and suggests next steps

For writing and communication:

- Writing coach that improves clarity and catches logical gaps
- Document drafter that maintains consistency with your templates
- Communication helper that adapts tone for difficult conversations

For planning and execution:

- Project planner that breaks down complex work and tracks dependencies
- Meeting facilitator that captures decisions and ensures follow-up
- Goal tracker that monitors progress and identifies patterns

For learning and growth:

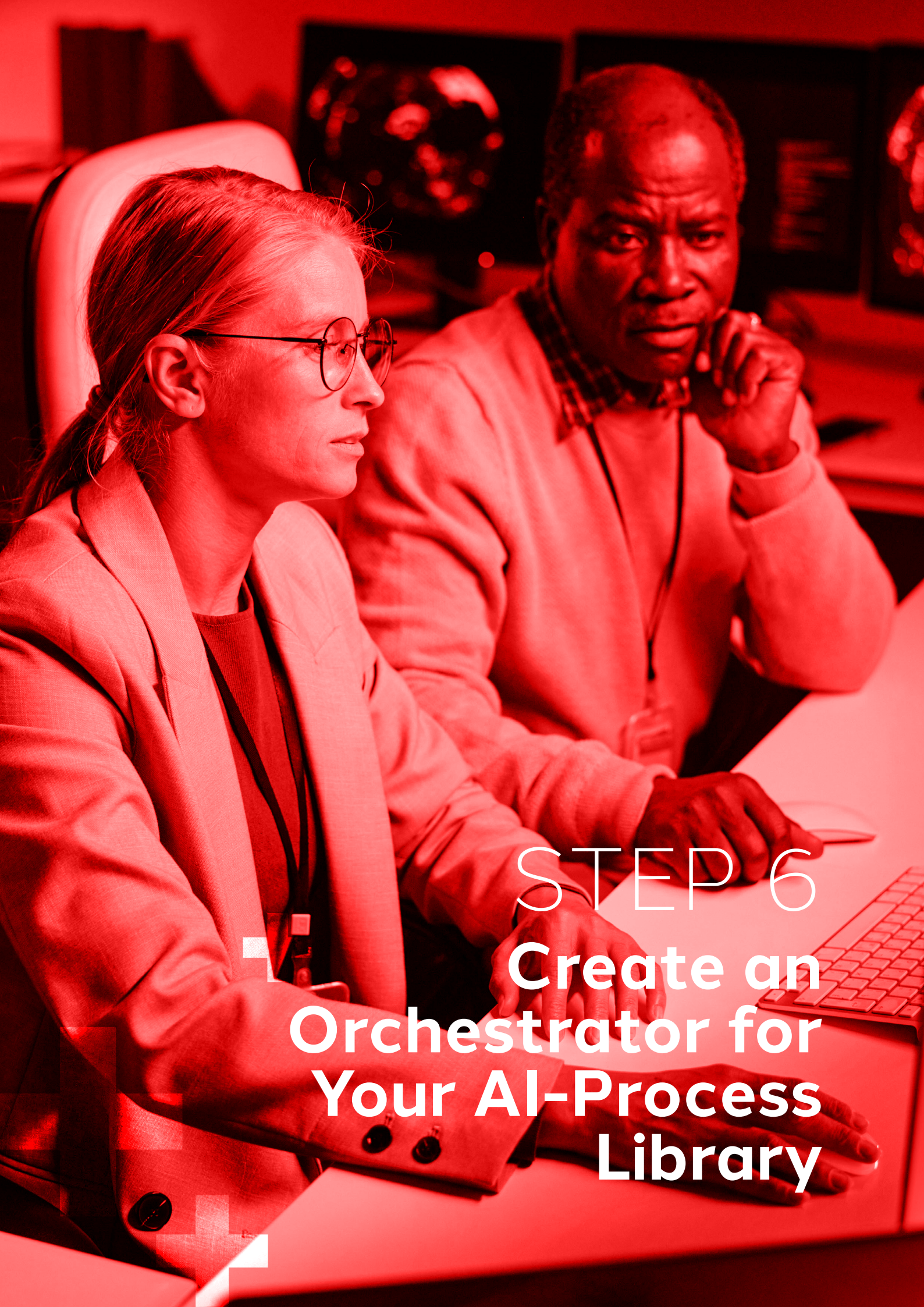
- Learning companion that explains concepts at your pace and creates practice exercises
- Reflection guide that asks probing questions and helps extract insights
- Skill builder that designs practice plans and adjusts difficulty

For creative work:

- Creative partner that generates variations and pushes boundaries
- Story developer that identifies plot holes and maintains narrative consistency
- Content creator that maintains brand voice across different platforms

These examples show the breadth of what's possible, but your best AI-processes will emerge from your actual needs. Start with the problem you're facing right now.

The key to building a powerful library: Start small and refine deeply. One well-refined AI-process that actually helps you is worth more than ten half-baked ones. Create your first AI-process, use it for real work for several days, refine it based on what goes wrong, then create the next one. Each AI-process you refine teaches you something about how to instruct AI effectively, about its strengths and limitations, and about your own way of thinking. This learning compounds—your fifth AI-process will be better than your first because you've developed your human-AI co-thinking skills.



STEP 6

Create an
Orchestrator for
Your AI-Process
Library

Note: Steps 6 and 7 describe an orchestrator—a nice-to-have for managing larger AI-process libraries. This is entirely optional. If you prefer to keep things simple and just organize your AI-processes in folders, you can skip to Step 8, which covers managing your AI-process ecosystem regardless of whether you use an orchestrator.

Once you have 5-10 refined AI-processes, a practical challenge emerges: Which AI-process should I use for this task? Where did I put that writing coach? Do I already have something for this, or should I create a new one?

The simple solution: Keep your AI-processes organized in clearly named folders. For many people, this is enough—you remember what you have and where it lives. A good practice is to put project-specific AI-processes inside the project folders where you need them, and keep general-purpose AI-processes (like writing coaches or brainstorming partners) in a central location. Always put your AI-processes where you'll naturally look for them.

The more elegant solution: Create an orchestrator AI-process that maintains a registry of your AI-processes and helps you find the right one for each task. This is particularly valuable if you share AI-processes with a team, have many specialized ones, or want automated discovery.

How to create your orchestrator

Step 1: Ask the generic agent to create it

Tell it what you need:

“Create an orchestrator AI-process for me. It should maintain a registry of all my AI-processes and help me find the right one for each task.

When I ask to update the registry, use the terminal to find all INSTRUCTIONS.md files in my workspace, read them to understand what each AI-process does, and maintain a registry in memory/ with the path, description, and last-used date for each.

When, instead, I describe a task, search the registry to find matching AI-processes. If you find one, explain why it fits and load it. If you find several, explain the options and let me choose. If you find none, offer to help me create a new one.

Keep the registry updated with last-used dates so we can track which AI-processes I actually use.”

The generic agent will create an orchestrator/ folder with instructions and memory files. If it needs help, provide this guide as context.

Step 2: Update the registry

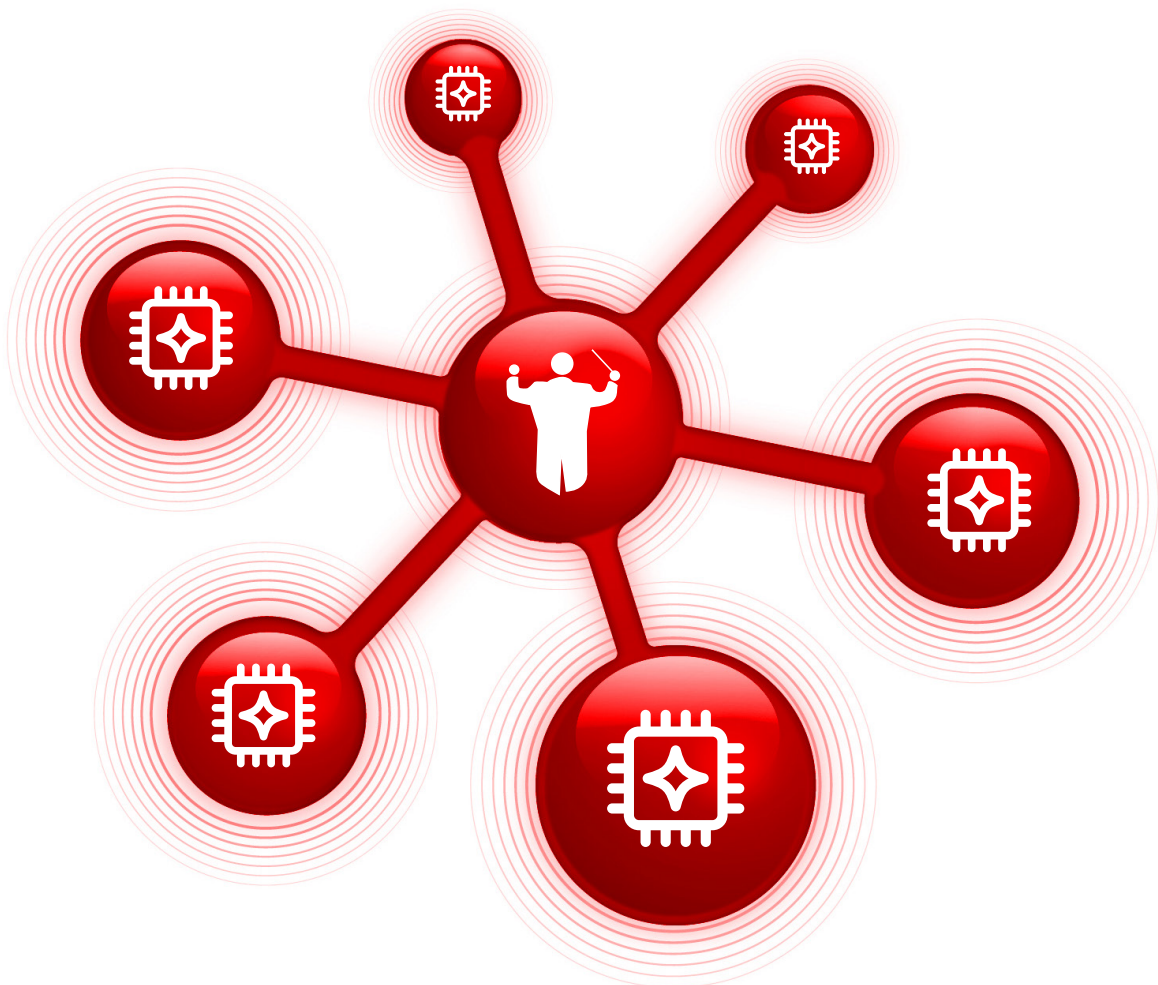
After creating the orchestrator, tell it:

“Update the registry”

It will scan your workspace and catalog all your AI-processes.

Step 3: Start using it

From now on, when you have a task, start by talking to the orchestrator instead of choosing an AI-process yourself. It will find the right one or help you create it.



STEP 7
**Work Through
Your Orchestrator**

Once your orchestrator is set up, using it is simple: describe what you need, and it finds the right AI-process or helps you create one.

Example—finding an existing AI-process:

You: “I need help planning my kitchen renovation”

Orchestrator: “Let me check my registry... I found your ‘Project Planner’ AI-process. It’s designed for breaking down complex projects with budgets and deadlines. Should I activate it?”

You: “Yes, that’s perfect”

Orchestrator: “Great! I’ve loaded the Project Planner. Now tell it about your kitchen renovation plans.”

Example—creating a new AI-process:

You: “I need to draft a difficult email to a colleague”

Orchestrator: “Checking my registry... I don’t see an AI-process for communication or writing. Would you like me to help you create a ‘Communication Helper’ AI-process? I have the co-thinking guide in my context to guide us.”

You: “Yes, let’s create one”

Orchestrator: “Perfect. Let’s define what this Communication Helper should do...”

Maintaining the registry: Periodically (weekly or when you’ve created several new AI-processes), ask the orchestrator to update its registry. It will scan your workspace and catalog everything automatically.

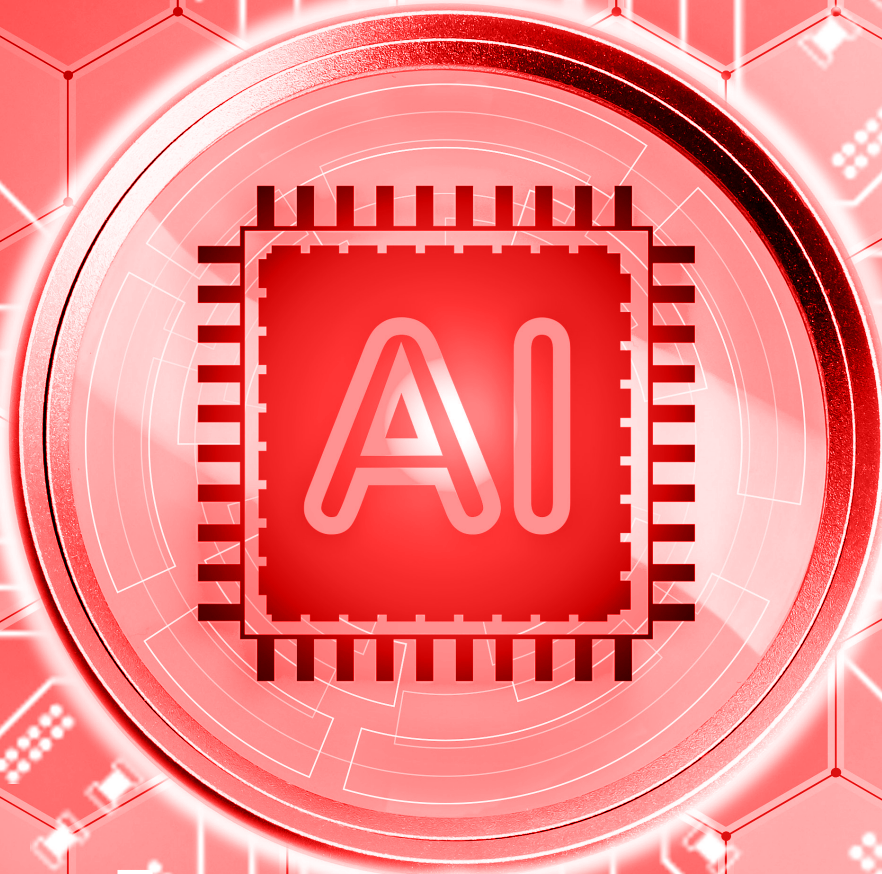
At this stage, you’ve reached an important milestone. You’re no longer just an AI user, you’re an intelligence ecosystem architect. The shift is subtle but powerful: instead of managing individual AI-processes, you’re conversing with a system that knows your entire toolkit and helps you use or build what you need.

For advanced users—semantic search with vector databases:

If you manage dozens of AI-processes and have programming skills, you can set up a more sophisticated discovery system. A script regularly scans your library, reads all `INSTRUCTIONS.md` files, and indexes them in a vector database. Your orchestrator then performs semantic search—finding AI-processes based on meaning rather than keyword matching—to automatically identify the most relevant one for your task. This scales to hundreds of AI-processes and requires no manual registry maintenance. However, it requires programming skills, understanding of vector databases, and ongoing technical maintenance. For most people and teams, the registry-based orchestrator described above works perfectly well.

STEP 8

Manage Your AI-Process Library



Your AI-process library is a living asset. Here's how to keep it healthy, secure, and effective over time.

Maintain and Refine

Review regularly: Every few weeks, assess which AI-processes you actually use. Archive ones you haven't touched in 3-6 months. If you have an orchestrator (from Steps 6-7), ask it to show usage patterns and suggest what to archive.

Spot patterns: Notice yourself doing the same task repeatedly? That's your signal to create a new AI-process for it.

Refine what works: Your most-used AI-processes deserve ongoing refinement based on what you learn from real use.

For teams: Share your best AI-processes with colleagues. Let them evolve collectively—the marketing team's content writer, refined by dozens of people, becomes far more powerful than any individual could build alone.

Store and Version

Where to store:

- **Local drive** for personal use
- **Shared drive** for team collaboration
- **Git repository** for professional tracking with complete history

Why version: Simple version management (e.g., brainstorming_partner_v1.2) lets you test improvements without risk, maintain traceability, and roll back if needed.

For the more technical:

Use Git to version your AI-processes. Each instruction modification becomes a commit with an explanatory message, creating a valuable history of your thinking tools' evolution.

Security and Data Protection



Before adding files to context:

- Check for passwords, API keys, personal data you shouldn't share
- Anonymize customer/patient data or use fake examples for testing
- Understand where your data goes: What you provide your AI-process is sent to wherever your AI model is deployed—often cloud servers (since the most capable models are computationally heavy). This depends on your setup: cloud APIs, your own deployment, or fully local models

When sharing with your team:

- INSTRUCTIONS.md files are usually safe to share
- Check memory/ folders for sensitive information
- Consider: Who should modify instructions vs. just use the AI-process?

Backup your work:

Your AI-processes are valuable tools refined through experience. Back them up regularly:

- Cloud storage (Dropbox, Google Drive)
- Your regular backup system
- Git (if using—automatic versioning and backup)

Additional considerations for AI workflows:

- **Processing external content:** When your AI-process handles content from outside sources (customer emails, user submissions, web scraping), be aware that malicious actors can craft text designed to manipulate AI behavior. For routine work this is low risk, but for critical decisions or sensitive operations, review AI outputs before acting on them.
- **Memory accumulation:** AI-processes naturally accumulate information in memory files over weeks and months. Before sharing an AI-process with colleagues, check memory folders—they might contain sensitive details from past conversations you've forgotten about.
- **Context folder organization:** Keep project-specific AI-processes in their respective project folders. This prevents accidentally giving an AI-process access to unrelated documents and keeps information properly compartmentalized.
- **Shared AI-processes:** If you share AI-processes with others (especially beyond your team), remember that your INSTRUCTIONS.md files describe your specific approaches, workflows, and priorities—which could inadvertently reveal strategic information about how you work.

For highly confidential work:

If you handle truly sensitive information (medical records, legal cases, classified material), consider local AI solutions that don't send data to cloud servers. Contact AI Swiss for guidance on private deployment options.

STEP 9
**Scale to Your
Organization with a
Knowledge Hub**

Note: This step is for organizations looking to deploy AI-processes at scale across teams and departments. If you're an individual user or small team, you can skip it. If you're responsible for digital transformation, keep reading!

When an entire organization adopts AI-processes, the benefits multiply significantly beyond individual use. AI-processes can fulfill the wide majority of your organization's needs—and when specialized development is required for very specific workflows, you'll have already identified your actual pain points and requirements through real use. This is far more effective than the common approach of deploying an internal ChatGPT-like tool (which lacks the structure and control of AI-processes) or asking consultants to build custom AI tools

for every need (which is expensive and often misses the mark because neither you nor the developers truly understand your workflows until you've worked with them).

The Knowledge Hub Approach

The core idea is to structure knowledge organization-wide to feed AI-processes. Instead of scattered information, create a **company-wide knowledge hub** for shared resources (guidelines, templates, best practices). Project-specific information (meeting notes, decisions, documentation) can live either within this hub or in separate project repositories and drives—whatever fits your organization. Your local AI workspace loads both the central hub and relevant project folders together, giving AI-processes access to everything they need.

What makes this powerful:

1. AI-processes live close to the work: Place them near what they support—support/faq/faq_refiner/ maintains customer FAQs, projects/product_x/requirements_tracker/ tracks project requirements, hr/onboarding/onboarding_guide/ guides new employee onboarding.

2. Teams refine collectively: AI-processes evolve through real use by multiple people. The marketing team refines the content creator together, the support team improves the FAQ manager based on actual cases. Everyone benefits from collective learning captured in instructions.

3. Structured context creates information cascades: Meeting notes automatically generate summaries, summaries feed into requirements documents, requirements inform design specifications, specifications generate implementation tickets. All interconnected, all AI-assisted, all traceable.

4. Proven workflows spread naturally: When an AI-process works well for one team, others can adopt it. The meeting facilitator that works great gets shared. The writing coach that catches compliance issues gets adopted company-wide. Best practices become reusable, versioned tools.

5. Workflows guide without overhead: AI-processes can embed procedures, compliance requirements, and documentation templates directly. An AI-process for regulatory submissions guides employees through each required step, applying the right templates, checking mandatory fields, ensuring compliance—all through natural dialogue. People focus on content and ideas while the AI-process structures everything correctly. This dramatically reduces onboarding time (new employees get guided through unfamiliar processes) and improves compliance (requirements are enforced automatically, not through training and memorization).

For development teams: Integrate your knowledge hub directly with your code repository—use part of the repo as a project hub. Code, documentation, decisions, and AI-processes all version together.

This transforms organizational intelligence. Knowledge isn't trapped in individual heads or scattered across tools—it's structured, accessible, and actively working through AI-processes that everyone can use and improve.

When One AI-Process Isn't Enough

Some conversations are too complex for one AI-process to handle well. When a project has too many moving parts, too much context to track, or too much information to compose intelligently, you need to split the work across multiple AI-processes working in sequence, in parallel, or in more complex ways.

But how do you split the work? The answer isn't always intuitive. The natural divisions you'd use for a human team don't necessarily apply to AI-processes. You need to think differently about where to draw the lines.

The key principle: Split when AI-processes would be independent with clean handoffs. Keep together when they need to share context a lot. AI-processes that need to share information frequently to do their job might as well be one process—the separation isn't helping.

Sequential workflows: From strategic to tactical to detailed

Here's where splitting becomes essential. You're launching a complex initiative—maybe a new digital platform for your organization, a major process redesign, or a multi-year research program. The scope is enormous. Trying to think through everything at once overwhelms both human minds and LLMs. You need hierarchical decomposition, moving from strategic to tactical to detailed thinking.

You create three AI-processes that work sequentially, each operating at a different level of abstraction.

Stage 1: Strategic decomposition and prioritization. You start by loading your strategic planning AI-process. You describe your high-level goals and constraints. It works with you to break down the initiative into major components, identify dependencies, assess risks, and establish priorities. This is meta-level thinking—what are the big pieces, what order makes sense, what are we trying to achieve? When you're satisfied with the breakdown, it saves everything to `project_structure.md`: the main workstreams, their relationships, the sequencing logic, and the success criteria for each.

Stage 2: Tactical requirements definition. Next, you load your requirements AI-process. It reads `project_structure.md` and works through each workstream you identified. For the first priority workstream, it helps you think through the practical questions: What exactly needs to happen here? Who's involved? What decisions need to be made? What information do we need? What are the constraints? It captures all this in `workstream_1_requirements.md`. You review, refine, then move to the next workstream. The AI-process stays focused purely on requirements—it's not trying to remember all the strategic context from Stage 1, just working from the clear structure you established.

Stage 3: Detailed specification and design. Finally, you load your design AI-process. It reads both `project_structure.md` (for context) and `workstream_1_requirements.md` (for specifics), then helps you work out the implementation details. If it's a technical project, this might be system architecture and data flows. If it's a process redesign, this might be step-by-step procedures and decision trees. If it's a research program, this might be methodology and analysis plans. It creates detailed specifications in `workstream_1_design.md`. Again, this AI-process isn't tracking the full strategic picture or juggling requirements—it's focused on one thing: translating requirements into concrete design.

Why this works—and why it’s often necessary: Each AI-process operates at one level of thinking. Strategic thinking requires holding the entire initiative in mind, seeing patterns, making trade-offs. Tactical thinking requires systematically working through “what exactly needs to happen” for each piece. Detailed thinking requires technical precision and internally consistent specifications. These are fundamentally different cognitive modes.

A single AI-process trying to do all three would need to simultaneously hold: the big-picture vision and trade-offs, the detailed requirements for every

Detailed thinking requires technical precision and internally consistent specifications.

workstream, and the technical specifications for implementation. The context becomes impossibly bloated. The AI loses track of what thinking level it’s operating at. You get strategic decisions contaminated by implementation details, or designs that lost sight of the actual requirements.

This is exactly why human teams decompose complex work hierarchically.

A program manager thinks strategically about the initiative. Product owners think tactically about requirements for their area. Designers and engineers think in detail about implementation. Not because humans can’t do multiple things, but because managing complex hierarchical thinking is only possible by separating concerns—the task becomes manageable only when each person focuses on their level while trusting that other levels are handled. This is a necessity dictated by our cognitive constraints, not a preference.

When does this approach make sense? When the work is genuinely complex with multiple levels of abstraction. Large projects, system designs, organizational change initiatives, comprehensive strategies. The more complex the hierarchy, the more essential the separation. Simple work with flat structure doesn’t need this—one AI-process handles it fine.

Note that three levels is just an example. Some projects need only two levels (strategy → execution), others need four or more (strategy → program design → workstream planning → detailed implementation). The right number depends on your specific complexity. Match the decomposition to your actual hierarchy.

Parallel workflows: Independent checks running simultaneously

AI-processes don’t have to work in sequence. They can work in parallel when tasks are independent.

You're finalizing a complex project proposal. You create four checking AI-processes that all review your draft at the same time: one checks regulatory compliance, one verifies procedures were followed, one validates terminology consistency, one examines budget calculations. Each reads `proposal_draft.md` and writes its own report: `compliance_check.md`, `procedures_check.md`, `terminology_check.md`, `budget_check.md`. You review all four reports together and make corrections. Done.

Why this works: The checks are independent. Compliance review doesn't need budget results to complete its work. Terminology validation doesn't depend on procedure verification. They can all run simultaneously, each focused on its own specialized lens.

Running parallel AI-processes efficiently:

Some local AI workspaces make it easier to run multiple AI-processes simultaneously. As of October 2025, you can use Claude Code through terminal commands within Cursor IDE to launch multiple AI agents in parallel—each running different tasks from the command line simultaneously. Cursor itself also supports multiple chat tabs for managing several AI conversations at once. This is particularly valuable for parallel workflows where you want all checks or analyses to run simultaneously rather than sequentially. Check your workspace's documentation for parallel execution capabilities—this feature landscape evolves rapidly.

How to split work effectively

This is where it gets counterintuitive. With a human development team, you'd naturally split frontend work and backend work. But for AI-processes, this is often a poor split. Frontend and backend depend heavily on each other—backend changes affect what the frontend can do, frontend requirements reshape the backend API. You'd have constant coordination overhead and shared context bleeding across both AI-processes.

Good splits are independent tasks. One AI-process researches, another writes based on completed research. One AI-process checks compliance, another checks style, a third verifies accuracy—each produces its own report without needing results from the others. Analysis in one stage, recommendations based on completed analysis in the next.

Bad splits are tightly coupled tasks. One AI-process designing the data model while another designs the API that uses it—these would conflict because changes in one constantly affect the other. One AI-process writing the introduction while another writes the conclusion—they need the same understanding of the document’s overall argument. Keep tightly coupled work in one AI-process.

The test: If two AI-processes would need to talk back and forth constantly to coordinate their work, they should be one AI-process.

Getting started with multi-process workflows

Start with something you already do in stages. Research then write. Analyze then recommend. Draft then finalize. Create two AI-processes: one for the first stage, one for the second. Define the handoff file clearly—the first writes to `findings.md`, the second reads from it.

Test this approach on one real project. Watch where things break. If the output disappoints, refine that AI-process’s instructions based on what actually went wrong. Once two AI-processes work well together, you can consider adding a third.

Make handoffs explicit in your instructions. When your strategic planning AI-process finishes, have it tell you: “Strategic breakdown complete. Structure saved to `project_structure.md`. To continue, start a new chat and load your requirements AI-process.” Six months later when you return to this workflow, these instructions guide you through each stage without needing to remember the details. Zero onboarding for the new person joining your team.

Start simple. Build from what works. Two well-designed AI-processes producing excellent results beats five poorly designed ones creating confusion.

Start
simple.
Build from
what works.

Want to explore this for your organization?

Contact AI Swiss for guidance on setting up efficient knowledge hubs and AI-process ecosystems.

AI Swiss' bet for the future

Today's tools like ChatGPT manage memory by collecting snippets from your conversations and reinjecting them into context windows. This creates clutter—bits of different projects bleeding together, old preferences mixing with new ones.

AI-processes are a **fundamentally better way to structure and organize** what users do and want remembered. Instead of messy snippets, they capture clean, focused behavior for specific tasks. They keep context windows dedicated to one process at a time, not polluted with fragments from everywhere. For technical users: think of this as a more useful compression of your habits and workflows—structured, reusable, and version-controlled.

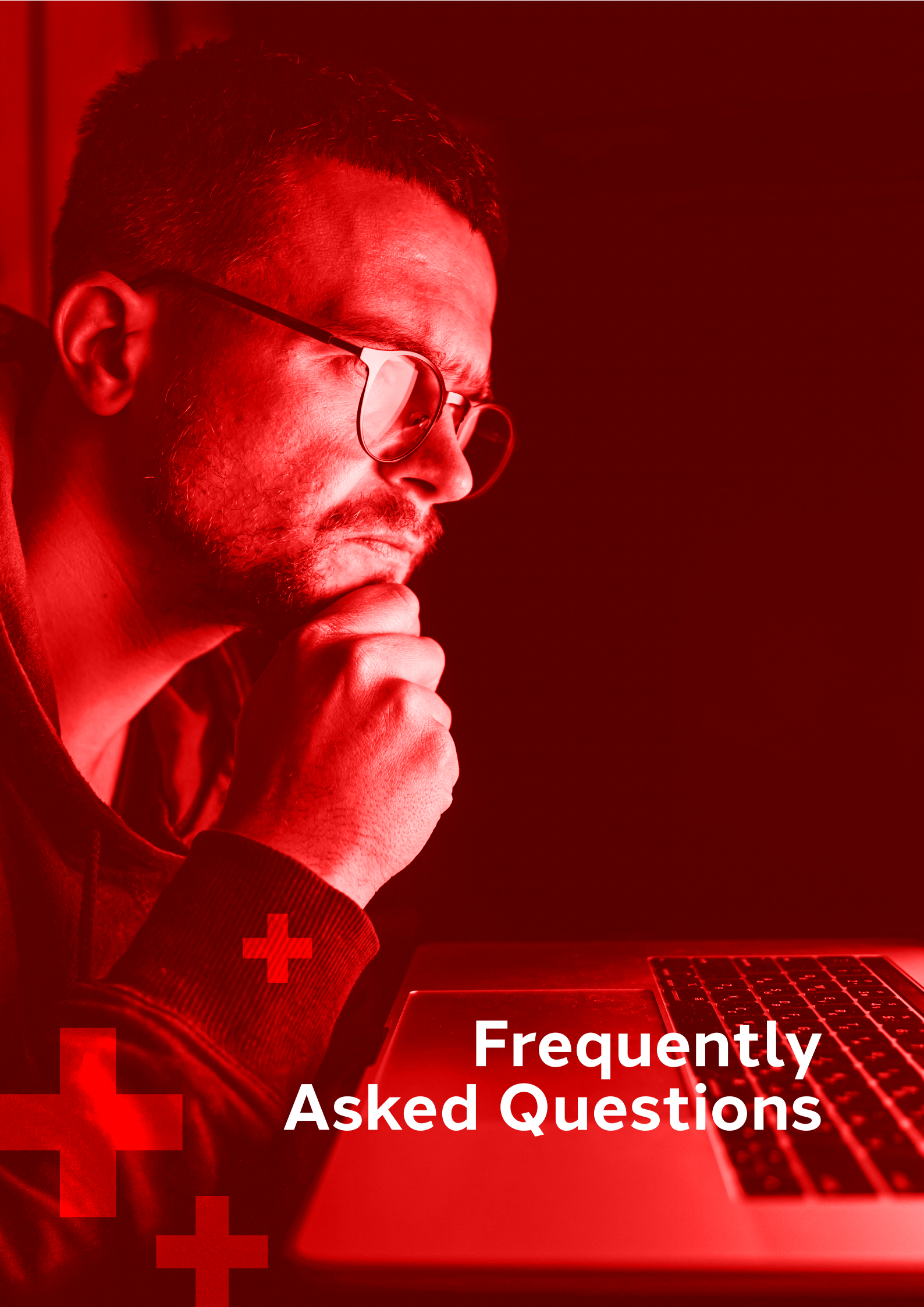
Where this is heading: We anticipate that someday, AI tools will manage AI-processes automatically in the

background—invisibly creating them as you interact, identifying patterns across sessions, refining instructions based on what works, organizing your library without manual work. The system would learn “Ah, they’re doing grant writing again” and activate the right AI-process automatically, or notice “This is a new type of task” and spin up a new specialized AI for it.

This would give you all the power of AI-processes without needing to architect them yourself. But for now, **building them manually remains your best bet**—you learn exactly how your thinking tools work, what makes them effective, and how to adapt them to your needs. There is immense value in that, especially if you’re prototyping AI agents.

**Interested
in how this
could
work?**

AI Swiss has working prototypes.
Contact us to explore what's coming.
info@a-i.swiss



Frequently Asked Questions

Getting Started

Q: Can I use this approach with ChatGPT or Claude instead of a local AI workspace?

A: The principles work with any AI, but you'll lose the key advantages—direct file access, persistent memory, and local control. You can create “AI-processes” as custom instructions in ChatGPT, but you'll need to copy-paste documents and manually manage context. For occasional use, this works. For serious co-thinking, local AI workspaces like Cursor or GitHub Copilot are transformative. Most importantly, there's no knowledge hub possible with web-based tools, missing out on the essential digital transformation any organization can achieve.

Q: How long does it take to create an effective AI-process?

A: If you know exactly what you want, creating the initial structure takes 3-5 minutes. Then you iterate a few times based on real use. First-time users might take longer to get comfortable with the approach. The key insight: start simple and refine through actual use rather than trying to perfect it upfront.

Start simple
and refine
through
actual use
rather than
trying to
perfect it
upfront.

Q: Do I need programming skills?

A: No coding required. You can write or even dictate instructions in your own language (works for most languages depending on the model), and organize files in folders. If something isn't clear, ask the AI to explain at your level—it adapts to your understanding. Advanced users can add Python scripts for custom capabilities, but this is optional. The power comes from clear communication, not technical complexity.

Using AI-Processes

Q: How is this different from ChatGPT's custom instructions or memory features?

A: Three key differences: (1) **Instructions:** You write them in visible files you can edit, version, and share—not hidden in settings, (2) **Context:** You provide specific documents the AI references—not copy-pasted text that gets lost, (3) **Memory:** You control exactly what's captured in structured files you can read and organize—not opaque personalization managed by ChatGPT. This gives you transparency and control over your entire AI workflow.

Q: What if my AI-process stops following instructions?

A: This can happen when context windows get full or conversations drift. First, check your instructions to ensure they're clear. Quick fixes: drag-and-drop your instructions file again to refresh it, or start a new conversation. For extended work, load your AI-process as a persistent agent (check your specific workspace's documentation for how—the method varies by tool).

Q: How do I know if my AI-process is working well?

A: Check whether it satisfies the instructions you specified. If it consistently behaves as you designed—asking questions when instructed, using context appropriately, updating memory files—it's working. If you find yourself correcting the same behaviors repeatedly, that's your cue to update the instructions to address those issues.

Practical Concerns

Q: Is my data secure when using local AI workspaces?

A: Your files remain on your computer, but content you share with the AI is typically transmitted to cloud servers where the models run. Review your workspace provider's data handling policies and terms of service. Do not share passwords, API keys, personally identifiable information, or confidential data unless you have verified the security measures and obtained necessary authorizations. For sensitive work requiring data sovereignty, consider fully local AI solutions or private cloud deployments that meet your security requirements.



Q: How much does this cost?

A: Costs vary by tool. For example, Cursor Pro costs around \$20/month (as of October 2025), while other local AI workspaces have different pricing models. Usage costs depend on how much you interact with AI models—typically \$10-50/month for regular use across most platforms. Compare this to hiring specialists or the time saved on complex tasks. Most users find it pays for itself quickly.

Q: Can I share AI-processes with my team?

A: Yes! Instructions files and folder structures are just files—easy to share, version, and collaborate on. Teams often refine AI-processes collectively, with everyone benefiting from improvements. Just check memory folders for sensitive information before sharing.

Advanced Questions

Q: How do AI-processes compare to building custom AI applications?

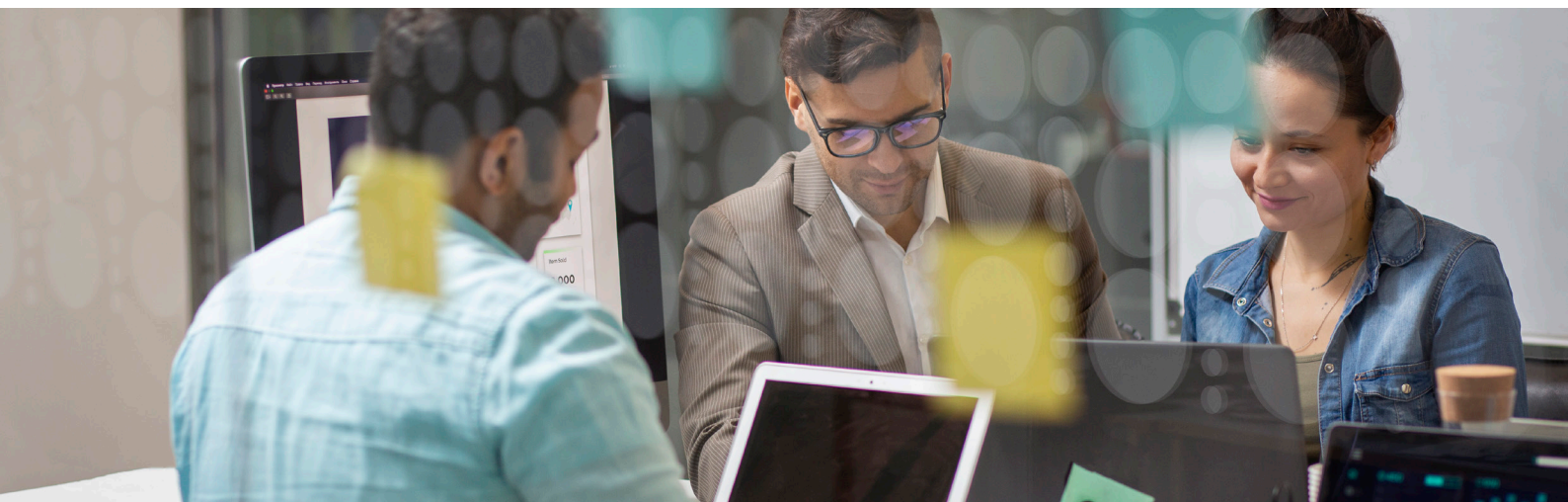
A: They serve different purposes. AI-processes are very useful for prototyping—you can test and validate AI workflows in hours before committing to development. Once you’ve proven what works through real use, you can translate those patterns into custom applications with clear requirements. Many needs are fully satisfied by AI-processes without ever requiring custom development.

Many needs
are fully
satisfied by
AI-processes
without ever
requiring
custom
development.

Q: What happens when new AI models come out?

A: Most local AI workspaces provide access to the latest models, so you typically get new ones as they become available. You can test how different models perform with your AI-processes.

With well-written instructions and models of similar capability, behavior often remains consistent—the methodology can be largely model-agnostic. You may want to test and adjust for optimal performance, but the core workflow transfers seamlessly.



A young child with curly hair is smiling, wearing a light-colored t-shirt. The image is overlaid with a red filter. In the bottom left corner, there are several overlapping squares of varying shades of red. Two white plus signs are also present: one in the lower-left area and another in the lower-right area.

**Your Co-Thinking
Journey Continues**

You’ve learned to create AI-processes, refine them through use, build a library, and optionally orchestrate or scale them. But co-thinking is a skill that deepens with practice—and you’re not alone in this journey.

Keep Learning and Growing

The AI field evolves rapidly. New models emerge, local AI workspaces add features, and the community discovers better practices. Here’s how to stay current:

Join the community:

- **AI Swiss community wiki** — Co-thinking best practices, shared AI-processes, troubleshooting help, and discussions with fellow practitioners

Watch for:

- **New AI models** — More capable, faster, or more affordable options
- **Local AI workspace updates** — Better file handling, new tools, improved interfaces
- **Community-shared AI-processes** — Proven solutions you can adapt to your needs

Remember the Fundamentals

As you advance, keep these principles at the core:

Co-thinking is a practice, not a formula. Each AI-process you create is an experiment. Some work brilliantly, others need refinement. That’s normal—the skill develops through doing, not just reading.

Put more intelligence in, not less. The better you understand your task, provide context, and design instructions, the better your results. AI amplifies your thinking when you stay actively engaged.

Stay critically aware. Quality writing doesn’t guarantee truth or relevance. You remain the validator, the critical thinker, the one responsible for the output.

And most importantly: This is all through your most human skill—communication. You’re not learning to code or master complex technology (though AI can help you with those if you want). You’re learning to communicate effectively with a new kind of interlocutor.



The Co-Thinking Revolution

Step back for a moment. What you’ve just learned is far more significant than a productivity technique.

You’ve learned an efficient form of **human-AI co-thinking**—a cross-cutting skill that’s emerging as an essential form of AI literacy.

Why This Moment Matters

Something is shifting in how we think about AI. Tech practitioners are discovering it too—they talk about “vibe coding,” and now “vibe working.” They’re circling an important insight: AI works best when it feels collaborative, not transactional.

They’re right. And you’ve just experienced why.

True collaboration isn’t about waiting for the next software update to add the feature you need. It’s about **shaping AI behavior yourself, just by communicating**. Control over instructions, context, memory, tools. Building exactly what works for you, whenever you need it.

It’s about
shaping AI
behavior
yourself, just by
communicating.

This changes the questions we ask. Instead of “Which AI is best—ChatGPT or Gemini or Claude?”, we can ask “What AI do I need to build today?” Leading AI models have reached a threshold where many excel at common tasks, making orchestration and application

increasingly important relative to raw model capabilities. What matters now is **orchestration**—the algorithmic layers on top of models, and critically, **your ability to shape them through communication**.

This represents a form of **personal AI autonomy**—not digital sovereignty in the traditional sense of infrastructure control, but something equally important: **the ability to shape AI behavior to serve your specific needs** rather than being limited to what tech companies build for you.

The AI Landscape and Where Co-Thinking Fits

To understand why co-thinking matters, let’s map the AI landscape:

Specialized AI algorithms solve specific computational problems—cancer detection in medical images, disease risk calculation from clinical data, numerical optimization. These have existed for decades and remain valuable for tasks that are well-defined, validated, and don’t require human judgment at each step.

Fixed AI-powered workflows link LLM-based blocks into fixed automation pipelines. A company might build a workflow that reformats information from multiple sources into a standard report, or an individual might create a button that performs a specific transformation. These are useful when you've identified a repetitive task worth industrializing—but building them requires either coding skills or visual programming platforms (essentially coding with blocks), plus extensive testing to harden the workflow.

General-purpose autonomous agents interpret what you want and act on your behalf across varied tasks. This sounds ideal—just tell an AI what to achieve and let it work—but creates fundamental problems when you're not in the loop: AI makes mistakes, hallucinates, misunderstands context. Without proper verification checkpoints, errors compound across autonomous decision points without you there to catch them. You lose situational awareness, learning, and the ability to validate what's being done.

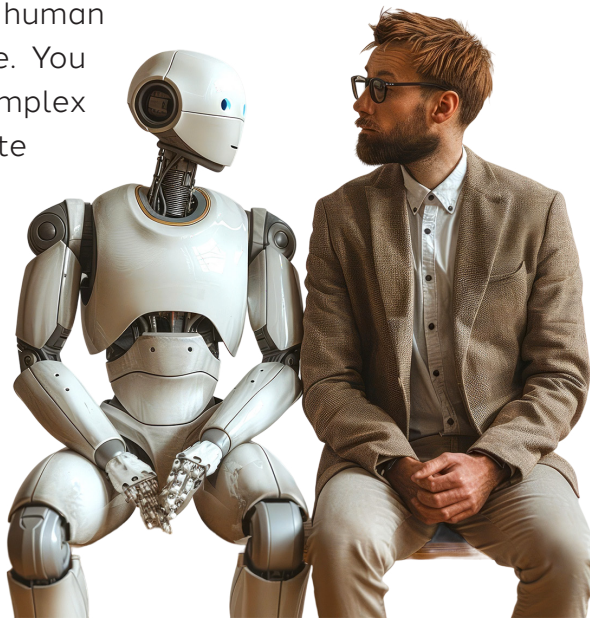
Human-AI co-thinking through dialogue is where you interact with a generic agent to satisfy the wide majority of your needs—thinking, creating, analyzing, deciding. Here, you remain actively engaged: providing context, making key decisions, validating outputs. The AI amplifies your intelligence rather than replacing it.

This is the foundational AI literacy. In terms of what matters for your cognitive development and the sheer breadth of tasks you can accomplish, co-thinking is paramount. It covers the wide majority of everyday needs. And it powers everything else:

When you identify patterns through co-thinking and prototype AI-processes, you can then ask engineers to harden them into workflows or specialized tools—now with clear requirements validated through actual use. Co-thinking also helps you *create* better AI: you can use it to code workflows more effectively, design specialized AI systems, even work on deep learning projects. You're not choosing between co-thinking and other AI forms—co-thinking is how you discover what else you need.

This is where the power to reduce the digital divide truly lies. For the first time in digital history, amplifying human intelligence is accessible to everyone. You don't need to code. You don't need complex automation platforms. You communicate clearly—your most human skill—and the AI augments what you already do well.

The choice is clear: **shape intelligence, or get shaped by it.**





**Join the
Movement**

When you started this guide, you were perhaps a consumer of AI—using ChatGPT, wondering whether the AI revolution is just hype. You're far from alone. Recent studies show the vast majority of enterprise AI initiatives

The problem isn't AI capability—it's the lack of structured orchestration.

fail to deliver results. The problem isn't AI capability—it's the lack of structured orchestration. Generic tools don't adapt to your work, and organizations lack frameworks to deploy AI systematically. This guide solves both: now you know how to architect AI for your specific needs individually, and scale it collectively through AI-processes company-wide.

Human-AI co-thinking is not just a powerful form of AI literacy—it's a movement. AI Swiss works to bring it to sectors including education, healthcare, and regulation, making the infinity of AI behaviors accessible to all.

Ask not only what AI literacy can do for you—ask what you can do for AI literacy.

The adventure is just beginning.

Welcome to the co-thinking revolution.

Your next steps

Put it into practice — If you haven't yet, create your first AI-process today. If you have, refine it and create your next one

See it in action — Watch the AI Swiss x Alp ICT series on the Alp ICT YouTube channel for a practical introduction to co-thinking, and explore our tutorials on the AI Swiss YouTube channel.

Join the community — Connect with fellow practitioners at a-i.swiss to share experiences and discover new approaches. Want to start a sub-community for your organization or domain? Contact us for partnerships and access to AI Swiss resources (troubleshooting, wiki, etc.)

Show, don't tell — Co-thinking spreads through example. Help someone close to you experience it themselves

Keep experimenting — This guide is your starting point. What you build from here is limited only by your intelligence and imagination

About the Author



Dr Charles-Edouard Bardyn

Chief Scientific Officer & VP, AI Swiss

Architect of the “human-AI co-thinking” approach

Charles-Edouard holds a PhD in quantum physics from ETH Zurich and drives AI innovation as Chief AI Officer at Domo Health and principal investigator in neuroscience at CHUV (Lausanne University Hospital).

As co-founder of AI Swiss and founder of the “human-AI co-thinking” framework, he offers an alternative perspective on how AI literacy and AI sovereignty are typically approached. Through systematically demonstrating to people across sectors what they can actually accomplish with AI, he identified what he sees as a critical gap: while some focus on understanding AI in theory, and others offer ad-hoc tutorials about “vibe working,” few recognize that the most important form of AI literacy for all is the practiced ability to think *with* AI—to orchestrate dialogue that compensates for AI’s limitations while amplifying human intelligence.

His approach centers on human-AI co-thinking as an emerging cross-cutting skill and systematic framework: learning to shape AI behavior through communication, staying critically engaged rather than passively consuming responses, and structuring interactions to maintain control. The method is accessible, but the transformation runs deeper—it’s about developing new cognitive skills for an AI-amplified world.

Charles-Edouard challenges the dominant narratives on two fronts. **On AI literacy:** he reframes the goal from understanding how AI is built to mastering how to think alongside AI—from theoretical knowledge to practiced orchestration, from consuming what tech companies build to directing AI as your thinking partner. **On AI sovereignty:** he argues that control shouldn’t depend solely on owning models and infrastructure. When anyone can systematically shape AI behavior through communication, they create a form of sovereignty accessible to all—one that can transform innovation, creative thinking, quality of work, organizational processes, and decision-making across every sector.

His vision for AI Swiss: make this systematic approach the foundation of how society engages with AI, ensuring AI amplifies rather than replaces human intelligence, and building capability that empowers rather than creates dependency.

This guide distills years of that work into a practical starting point. The revolution begins with you.



Empowering Switzerland with AI for All