syncro soft

Open4Tech Summer School 2026

Stop Talking to Chatbots. Start Deploying Agents.

Alexandru Smarandache
Software Developer, Syncro Soft

WHAT YOU'LL LEAVE WITH

Three pillars, three live demos, one honest verdict

01

Tool use

Agents that DO things, not just describe them.

02

Memory & context

Why an agent can remember you and manage what it knows.

03

Orchestration

Teams of specialist agents working together.

Plus: where 'useful' turns dangerous, and what's actually production-ready today.

CONTEXT

Why are AI agents exploding right now?

2022

The query era

ChatGPT launches. Everyone types questions and reads answers. AI as an encyclopedia.

2023-24

Tools + plug-ins

Models get 'hands' — search the web, run code, read files. First signs of action.

2025-26

Agents in production

Software acts autonomously. People supervise instead of execute.

Not theory — you know these already

GitHub Copilot

Writes code suggestions inside your editor. Tool use: reads your file context.

Cursor

Reads the whole codebase, edits files, runs fixes. Full agent loop.

Claude Code

Reads your repo, edits files, runs commands, writes tests — from the terminal.

Perplexity

Searches web → synthesises → cites sources. Tool use in action.

Devin / SWE-agent

Reads ticket → writes code → runs tests → opens PR. Fully autonomous agent.

We stopped giving AI a prompt.
We started giving it a problem.

01

Chatbot vs. Agent

What actually changed?

THE CORE DIFFERENCE

One talks. The other acts.

Chatbot

- Text in → text out
- Every action still goes through you
- Forgets when the tab closes
- Knows things, can't change things

Agent

- Goal in → actions out
- It has hands (tools) + a notebook (memory)
- Remembers across sessions
- Decides, acts, checks, repeats

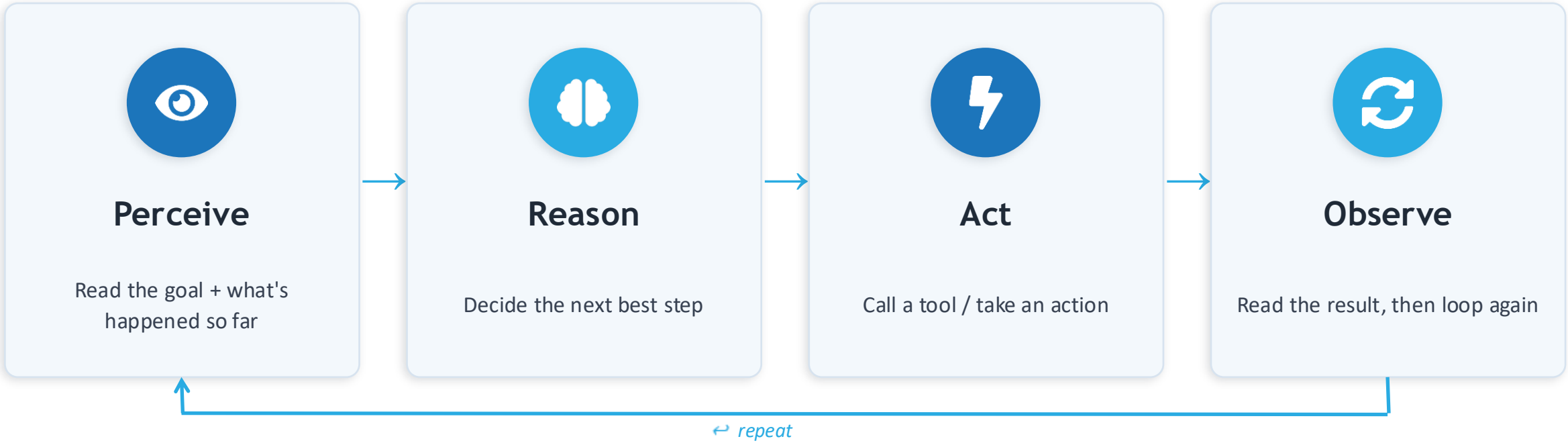
A chatbot tells you how to reset the router.
An agent resets it.

02

The Agent Loop

The engine underneath every agent

Perceive → Reason → Act → Observe → repeat



It keeps looping until the goal is met — that autonomy is what makes it an agent, not a one-shot reply.

03

Pillar 1 — Tool Use

Giving the model hands

The model requests — the application acts

1. Model decides

Given the goal, it picks a tool and the arguments to call it with.

2. The application runs it

The model never executes anything itself — the application function does the work.

3. Result goes back

The model reads the real result and continues the loop.

Examples of tools

- search the web
- run code
- query a database
- send an email
- call an API
- do exact math

LIVE DEMO 1

Tool Use — the Study Buddy agent

```
$ python3 demo1_tool_use.py
```

What happens on screen:

- The agent REQUESTS calculate_gpa — it doesn't do the math itself
- The application executes the tool and returns 3.5
- It then requests lookup_course for the prerequisites
- Only after real results does it answer

A tool = a function + a description

TOOLS schema

```
{ "name": "calculate_gpa",
  "description":
    "Calculate student GPA",
  "input_schema": {
    "type": "object",
    "properties": {
      "grades": {
        "type": "array",
        "items": {
          "type": "number"
        }
      }
    }
  }
}
```

Python implementation

```
def calculate_gpa(grades):
    if not grades:
        return "No grades"
    avg = sum(grades)/len(grades)
    return round(avg, 2)

# That's it. Any Python
# function can be a tool.
```

Add your own tool — 3 steps

1. Write the function

```
def my_tool(param): return result
```

2. Add to TOOLS list

```
TOOLS.append({ "name":"my_tool", "description":"..." })
```

3. Run the agent

```
The model will use it automatically if it considers it useful
```

Question: What tool would you add to study more effectively?

04

Pillar 2 — Memory & Context

So it doesn't forget you every time

Two kinds of memory

Short-term: the context window

Everything in the current conversation. Big, but limited — and gone when it ends.

Long-term: persistent memory

Facts saved to a file or database, loaded back next time. This is how it 'remembers you'.

Demo 2 shows long-term memory as literally a JSON file the agent reads and writes.

You can't fit everything on the desk

The desk metaphor

A desk only fits so much paper. To keep working you must constantly decide what stays, what gets summarized, and what gets filed away.

The context window is that desk. Context management is keeping the right papers on it.

Summarize

Compress old turns into a short recap

Retrieve

Pull in only the relevant facts on demand

Forget

Drop what no longer matters

LIVE DEMO 2

Memory — it greets you by name

```
$ python3 demo2_memory.py   (run it twice)
```

What happens on screen:

- 1st run: the agent learns your name, major, goal and SAVES them to memory.json
- Open memory.json on screen: the 'brain' is just a file
- 2nd run: it greets you by name and picks up where you left off
- Reset anytime with: python3 demo2_memory.py --forget

Persistent memory = 4 lines of Python

demo2_memory.py — the essential functions

```
import json, os

def load_memory():                    # read the brain
    if os.path.exists("memory.json"):
        return json.load(open("memory.json"))
    return {}

def save_memory(data):                # write the brain
    json.dump(data, open("memory.json", "w"), indent=2)
```

05

Pillar 3 — Orchestration

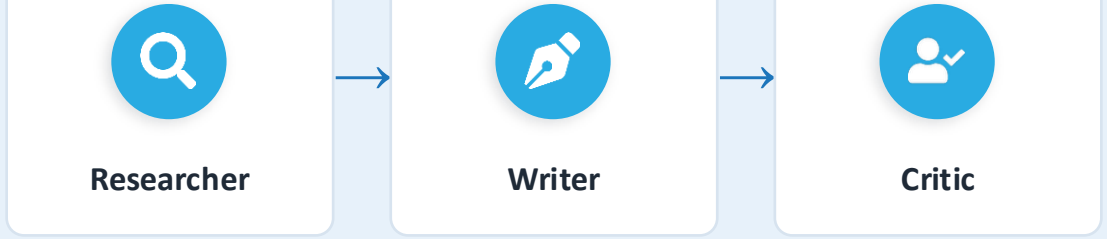
A team beats a lone genius

One giant prompt vs. a team of specialists

One big brain

- Tries to do everything at once
- Hard to debug when it fails
- One mistake derails the whole task

A team of specialists



Each agent does ONE job well. An orchestrator routes the work and a critic checks it — that's where reliability comes from.

LIVE DEMO 3

Multi-agent — Researcher → Writer → Critic

```
$ python3 demo3_multi_agent.py
```

What happens on screen:

- Orchestrator breaks down 'explain binary search trees'
- Researcher gathers the facts → hands off to the Writer
- Writer turns facts into a study summary → Critic approves it
- No single prompt did it all — each agent had one job

System prompts = the agent's identity

RESEARCHER

```
You are a Research
Specialist. Your job is
ONLY to gather
accurate facts about
the given topic.
Be concise and
factual. No opinions.
```

WRITER

```
You are an expert
Study Guide Writer.
Transform raw facts
into clear, engaging
summaries for
beginner students.
Use simple language.
```

CRITIC

```
You are a Quality
Critic. Review the
study guide for
accuracy and clarity.
Reply APPROVED or
REVISION NEEDED:
with specific notes.
```

06

Useful vs. Dangerous

Where the line starts to blur

THE DOUBLE EDGE

The power to act is the power to mess up

Hallucinated actions

Confidently does the wrong thing

Prompt injection

Hidden instructions hijack the agent

Runaway loops

Burns time + money with no exit

Over-permissioning

Given more access than it needs

STAYING IN CONTROL

Guardrails that keep agents safe

Human in the loop	Require approval before risky actions (spend, send, delete).
Scoped permissions	Give the minimum access needed — nothing more.
Spending + step limits	Hard caps so a runaway loop can't drain the budget.
Logging & review	Record every action for a complete audit trail.

Give an agent the keys — not the deed.

07

Production vs. Hype

An honest snapshot for 2026

HONEST SNAPSHOT

What's real today vs. still hype

Working in production

- Coding assistants (this is the big one)
- Customer support deflection
- Research & summarization
- Narrow, well-defined workflows

Still hype / early

- Fully autonomous 'set & forget' agents
- Reliable long-horizon planning
- Big multi-agent swarms, unsupervised
- Replacing whole job functions outright

Agents are powerful today —
with a human in the loop
and a narrow job.

YOUR TURN · PROJECT IDEAS

What else can you build with what you know now?

Study Coach

Quizzes you on course material, evaluates answers, and remembers where you struggle.

CV Reviewer

Takes a CV, looks up industry standards, gives structured feedback with concrete suggestions.

Code Explainer

Reads source code, explains what it does, suggests improvements — step by step.

News Digest

Searches news on a topic, filters duplicates, generates a personalised daily summary.

Schedule Optimizer

Reads your schedule, suggests focus blocks, redistributes tasks by priority.

Data Analyst

Takes a CSV, runs calculations, generates charts and interprets the results.

ECOSYSTEM · WHERE YOU FIT IN

Three levels: model → SDK → framework

The model

- Claude (Anthropic)
- GPT-4o (OpenAI)
- Gemini (Google)
- Llama (Meta, open)

The brain. Receives text+tools, returns text+tool calls.

SDK direct

- anthropic Python SDK
- openai Python SDK
- google-generativeai
- requests + JSON (manual)

The library that talks to the API. Today's demos use the Anthropic SDK directly.

Frameworks

- LangChain / LangGraph
- CrewAI
- AutoGen (Microsoft)
- Semantic Kernel

Abstractions over the SDK. Faster to prototype, harder to debug.

Recommendation to start: direct SDK (less magic, more control). Use a framework when you need speed.

GETTING STARTED · RIGHT NOW

Your first agent — follow along right now

1 

Run the demos

```
python3 demo1_tool_use.py
See the coloured output? Done — it works.
```

2 

Add a tool

```
Write a Python function in demo1.
Add it to the TOOLS list. Run again.
```

3 

Go live

```
pip install anthropic
export ANTHROPIC_API_KEY=sk-ant-...
Run it — now it's real.
```

4 

Build your own

```
Pick a project from the previous slide.
Copy the demo structure — it's already scalable.
```

RECAP

The three pillars, one more time

Tool use

It acts — calls tools instead of just talking.

Memory & context

It remembers you and manages what it knows.

Orchestration

Specialist agents team up with clean handoffs.

All three plug into the same loop: Perceive → Reason → Act → Observe.

Grab the workshop folder and extend it

Workshop Code

All 3 demos ready to run locally — no setup required beyond Python.

Step-by-Step README

Complete guide for adding new tools, memory, and multi-agent patterns.

Production-Ready Templates

The demo structure is fully scalable — use it as the base for your own projects.

Everything is in the workshop folder: demos, slides, a step-by-step README. Start with demo1_tool_use.py.

Questions?

Stop talking to chatbots. Go deploy an agent.

Alexandru Smarandache · Software Developer, Syncro Soft · Open4Tech 2026