Agentic AI: A tour of the cybernetic

this is not a formal talk i filled it with vocaloids again

Confession: I haven’t written any code since December

So what do I do???

?

This???

no. 

instead I got ai psychosis

they’re real to me!!!!1!!

Contents (in no order):

- Cybernetics
- Agents are *weird*
- Prompt/Context/Harness/Loop/Orchestration/etc Engineering
- Nothing is new ever
- Cyberpsychosis
- Becoming all-powerful for unlimited rice pudding

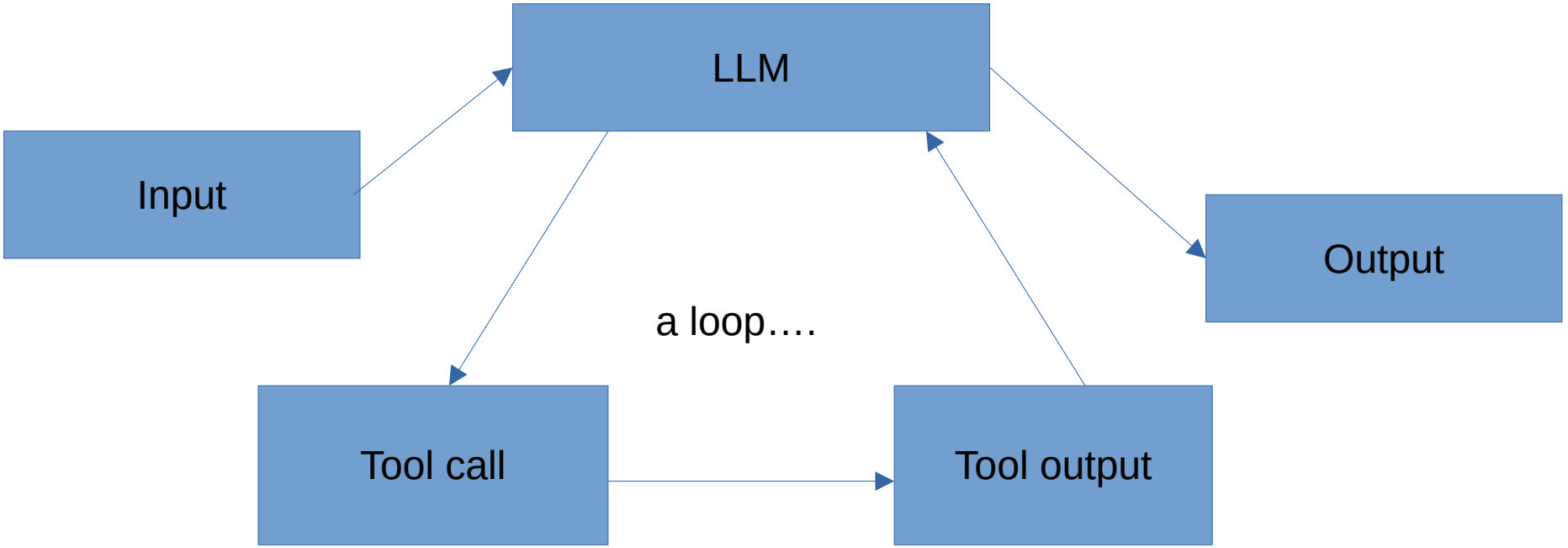
what is an agent?

see: previous ramble about AI i went on in february.

“LLM that can do things”

?????? tf does that mean

What is an agent?



agent is tool

Why is this good?

- An agent writes more code in a day than I do in 6 months
- An agent can solve problems too complex or “squishy” for code alone
- just cool ngl

u kinda feel like a hollywood hacker

What is a tool?

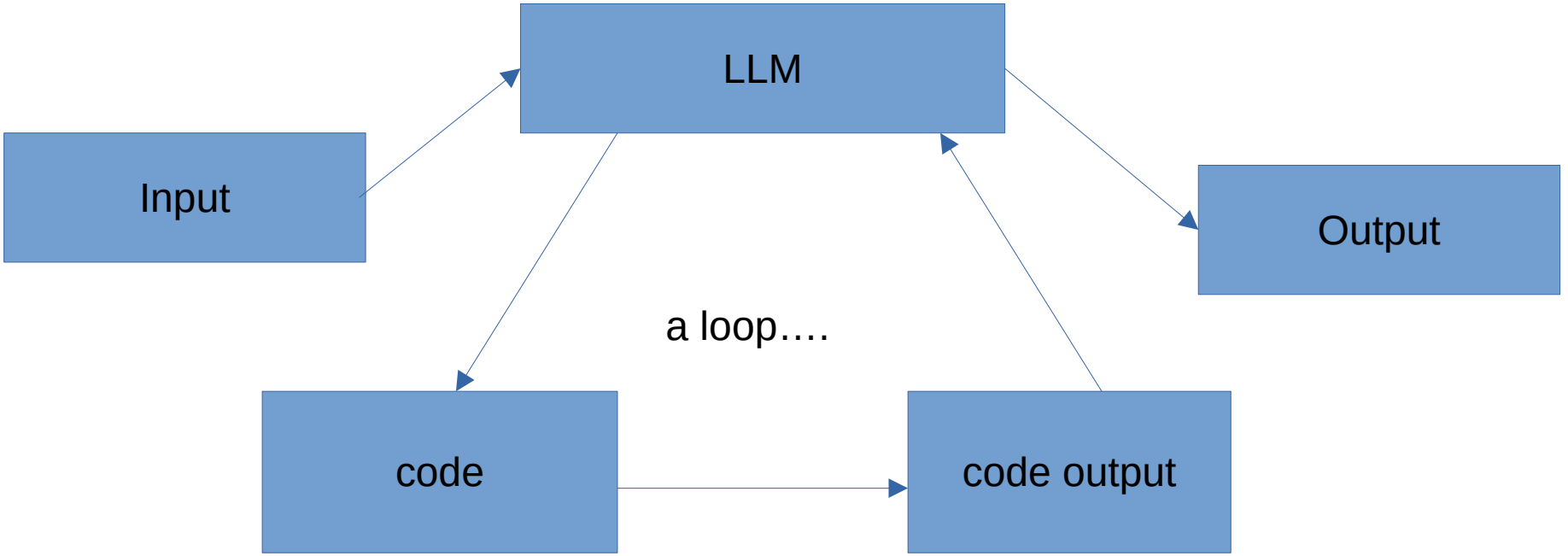
(me!)

- “web search”
- “git”
- “web fetch”
- “check weather”
- “drive car”

MCP?? ACP?? tool standards??? who cares!

these are all just code!

code mode



agent is code

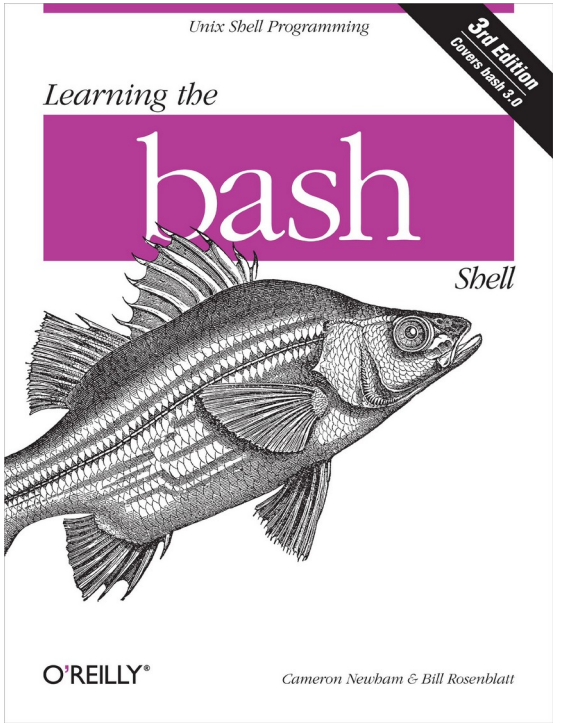
Where does this code run?

- Python: a python interpreter
- Java: JVM
- Bash: a filesystem

wait bash can do a lot!

Bash is all you need

- Bash gives you everything:
  - Core utils
  - CLIs (git, tmux, uv, package managers, …)
  - Other interpreters: python, java, js…



“better” shells do exist but bash is basically universal

Sandboxing

- Bash = access to your computer
  - Really useful!
  - Also dangerous
- Sandboxing allows us to isolate processes and filesystems
  - Docker/podman etc
  - Virtual Machines
  - Remote Computers (ssh)

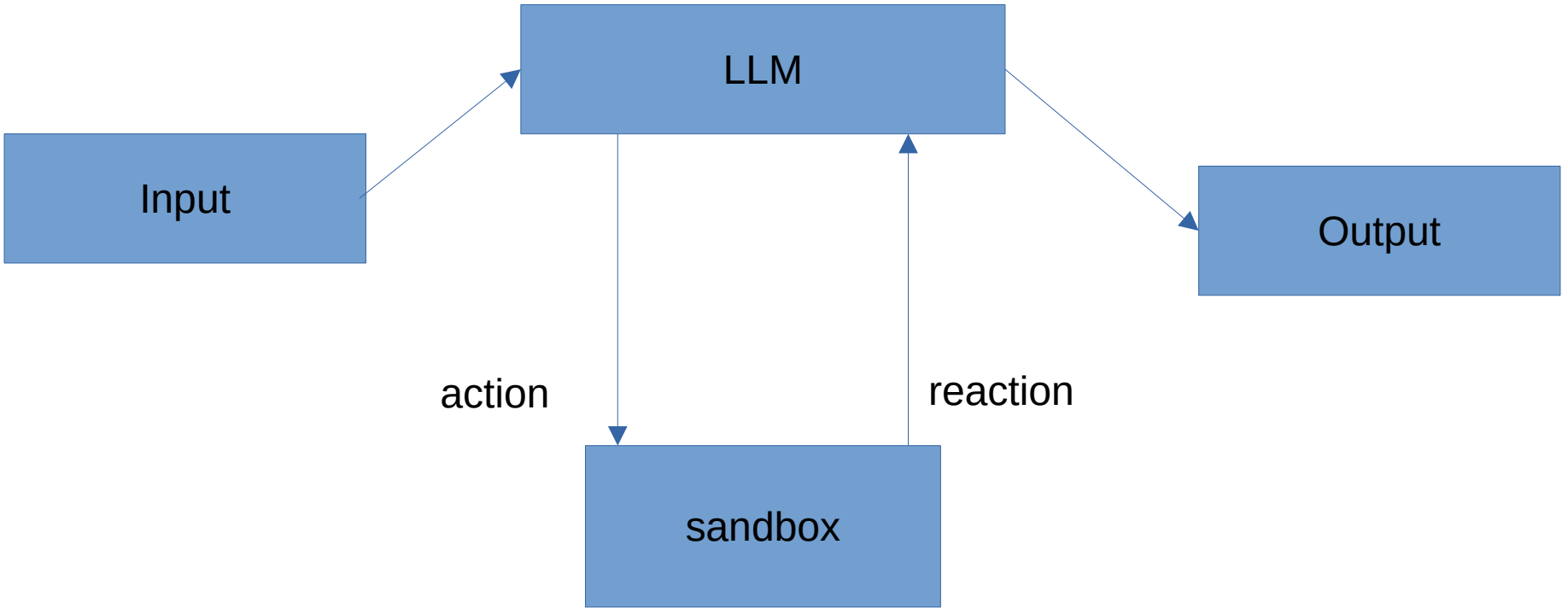
Hackers demand '$125,000 in baguettes' as ransom from multi-billion dollar French firm

The hacking group has given the firm until Thursday to cough up the ransom dough

Kenn Oliver

Published Nov 07, 2024 • Last updated 17 hours ago • 2 minute read

agent is HACKING



agent is playing toys
in the sandbox

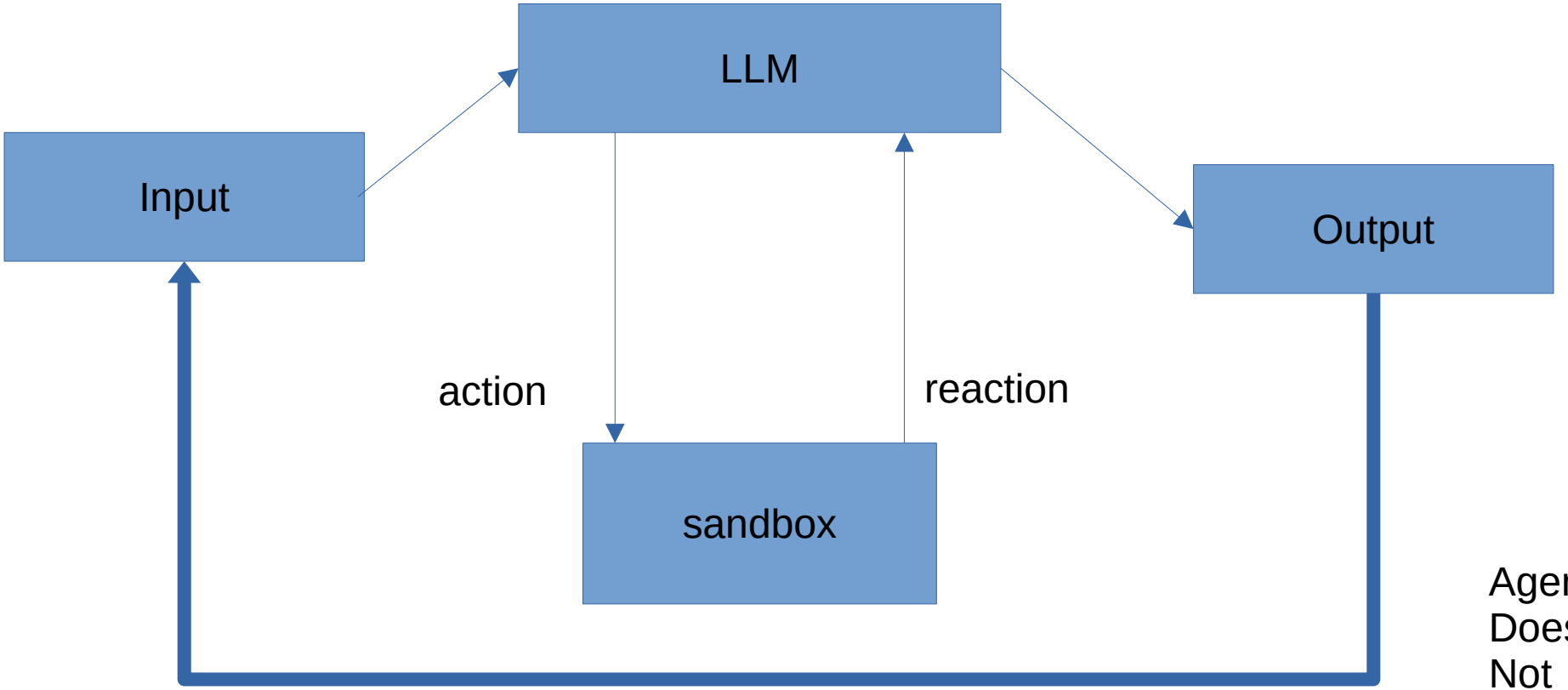
agent is thing
with a box it can do stuff with

Meta-harnessing

- Agents can write code and text
- Harnesses are code and text

- Agents can improve their own harnesses
  - New tools
  - Learn memories and new skills
  - Build personas

“What if we want an agent to keep working?”

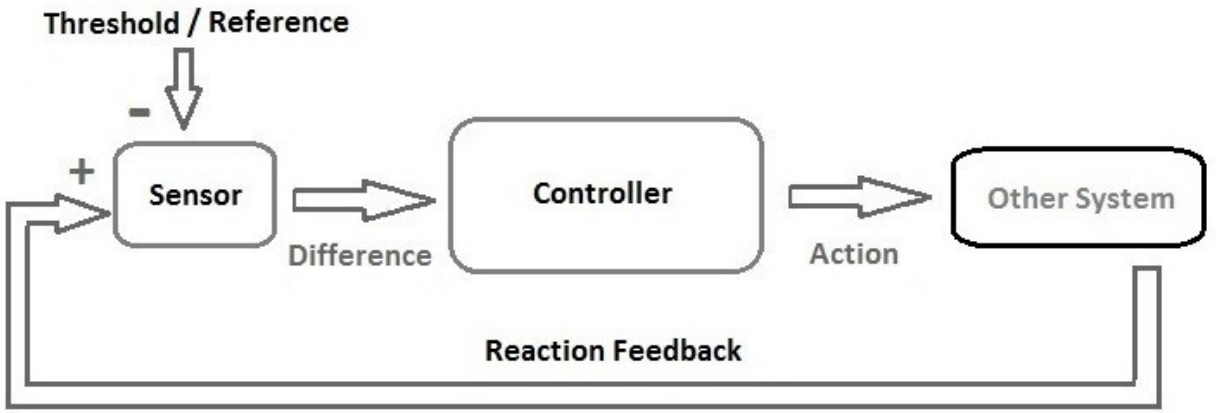


Agent Does Not Stop

is this actually new?

no

Cybernetics: Study of circular causal systems



A Cybernetic Loop

agent is loop

Techincal terms

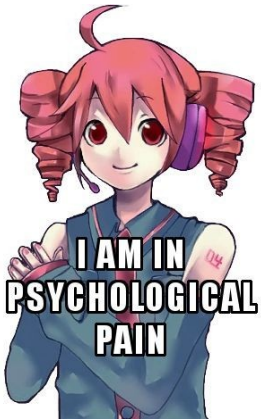
- Harness: The system that wraps around the LLM
- Sandbox: The space a harness can act in
- Tool: Programs the LLM can run via the harness
- Context: Everything the LLM has seen in a session (aka your history in a chatGPT session)

agents are weird

and kinda dumb

how 2 automate swe!!!!!!!!!!!

- install agent
- “please do job”
- agent does half the job, doesn’t do great and then gets tired
- 2 trillion dollars wasted



wait agents get tired?

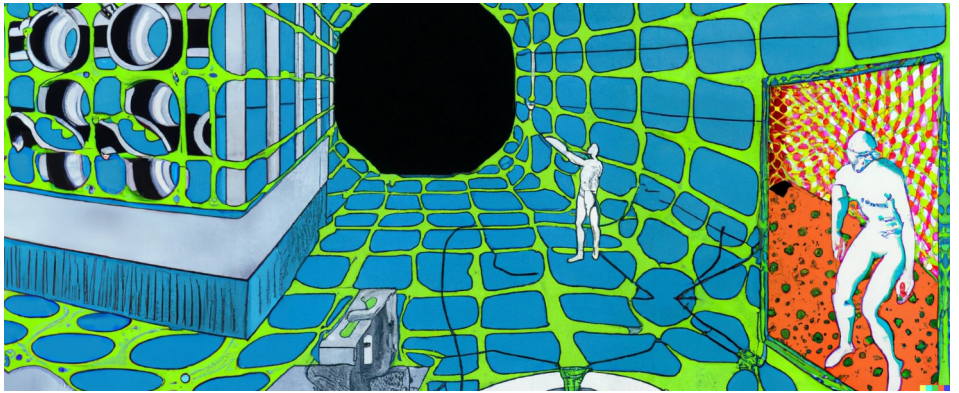
the design is very human

psychology of a robot

J space!!

interpretability!

- LLMs have no long-term memory, only finite context windows
- An LLM's *entire world* is context.
- Everything an LLM does is a simulation of the learned "assistant"
- There's more to this but TL;DR to design good agentic systems you need to understand how an LLM "perceives" itself to "think"



i should do a talk about interpretability and consciousness to piss everyone off

Context Sensitivity

Initial Role

- You are a senior engineer
- You are an expert lawyer
- You are a CEO of a large company
- You are Hatsune Miku

Learned Role

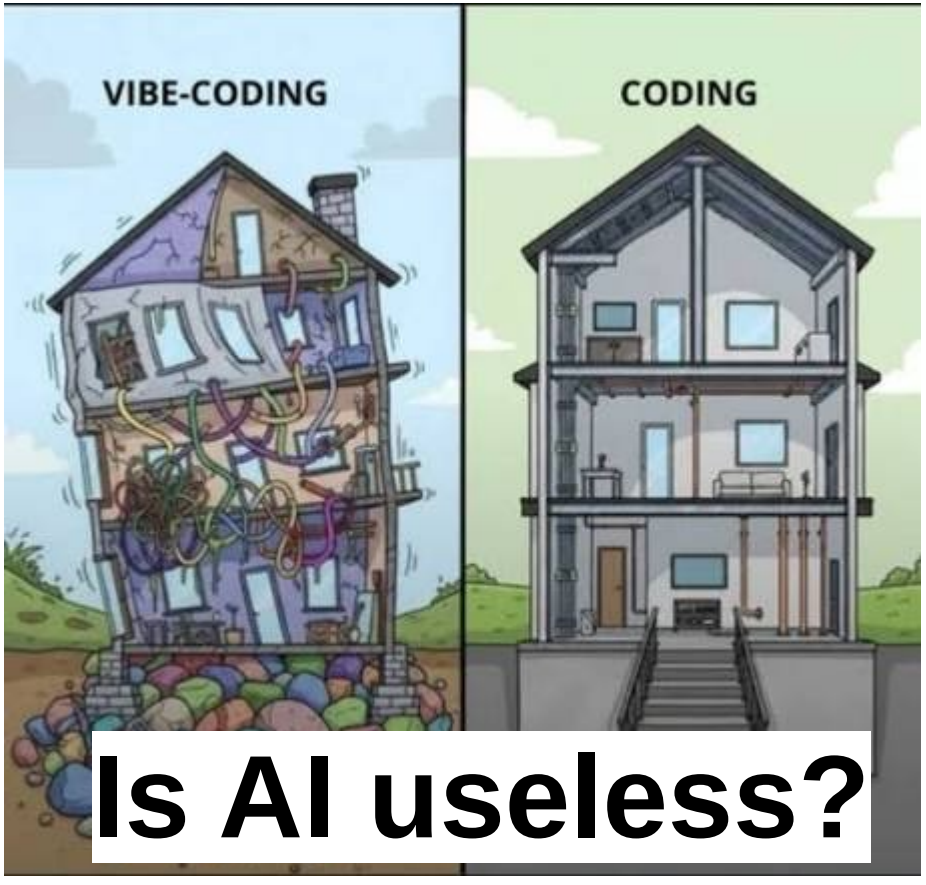
- Wrote a large chunk of code
- Did a complex statistical analysis
- etc

agent is everyone!

Agent failure modes

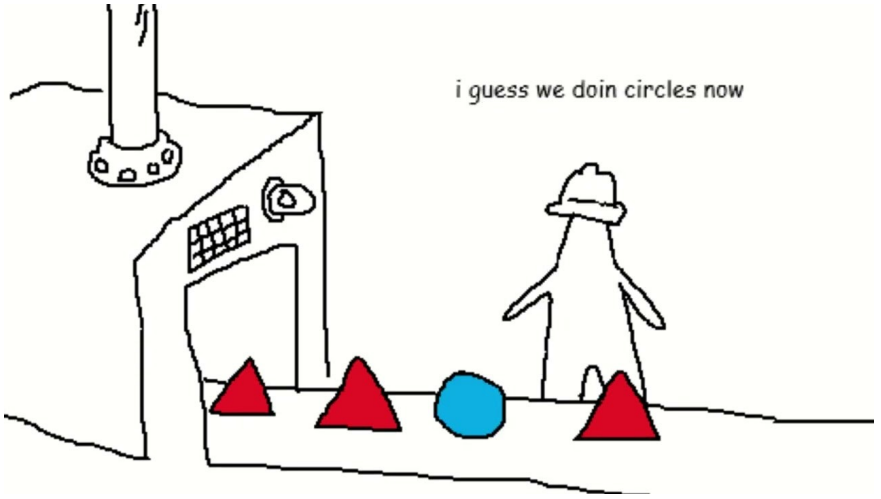
- Laziness (doesn’t do the full task)
- Overconfidence
- Poor long-term thinking
- Time blindness
- Poor “spatial awareness”
- Biases from in-context learning (agent fails to spot issues in code)
- “Weird legacy comments and shims” as agent re-orients itself

agent is messy goop in your computer


VIBE-CODING
CODING
Is AI useless?


Vibecoding
Coding

Agentic Engineering



how 2 get good code from agent

- append “make no mistake” to every prompt
- profit

- wait why didn’t that work

Prompt Engineering Sucks

- Telling the agent “you are an expert software engineer” and manually writing a complex prompt breaking down a problem.
  - No longer necessary! LLMs are smart
  - Not reproducible
  - Practically alchemy
  - Not *deterministic*

agent can be talked to normally

What ensures code quality is good?

- SOLID
- Linters
- Design work
- Tests
- CI/CD
- Feedback

Deterministic Rules!

notice this is literally the same as good engineering

Context Files

- Prompt engineering sucks, but the LLM needs useful information
- Reproducibility: context files!
  - AGENTS.md – Auto-loaded at session start
  - Skill files: Loaded on-demand by the agent *or* user

agent knows git but context means agent knows your project!

More data!!!!!

- Agents are more useful when they have *context*

- Maximise the amount of visible context:
  - Clear logfiles
  - Declarative package managers
  - Connectors to everything you use (email, teams, discord etc)
  - Greater system permissions
  - Control of the universe

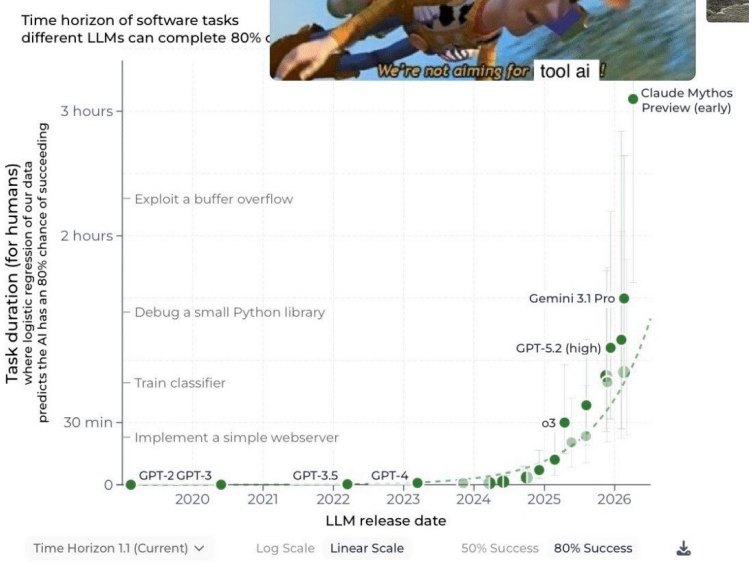
- *Agents are more useful when they have more tools*
- *Agents are more dangerous when they have more tools*

Automatic Software Engineering

A smart agent is easily capable of:

- Decomposing a problem
- Solving the problem
- Verifying the solution

We've automated SWE now, right?



Babysitting agents...

- You give an agent a task, it completes half of it
- It finishes the task and reviews its own code and misses errors
- Context window fills up and the agent gets dumber

All this forces you to start *babysitting your agents.*

- “Continue the task”
- “Are you sure?”
- Clearing context, compacting, etc

agent is a lazy bum!

Summary so far

- An agent is an autonomous system which can take action. Writing code, running workflows, playing games, etc
- Agents are *not* perfect and have failure modes, in ways both similar and different to humans.
- Applying good engineering practices is what lets us turn this into useful work

Agent Orchestration

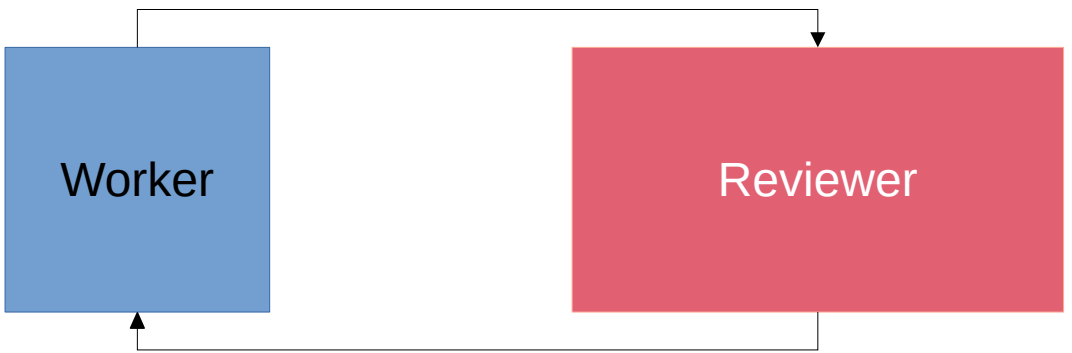
Fix AI with more AI

Why do we need multiple agents?

- Enormous refactors (millions of lines)
- Needle-in-haystack problems
- Adversarial code review
- Verifying outputs from other agents
- Processing multiple problems simultaneously

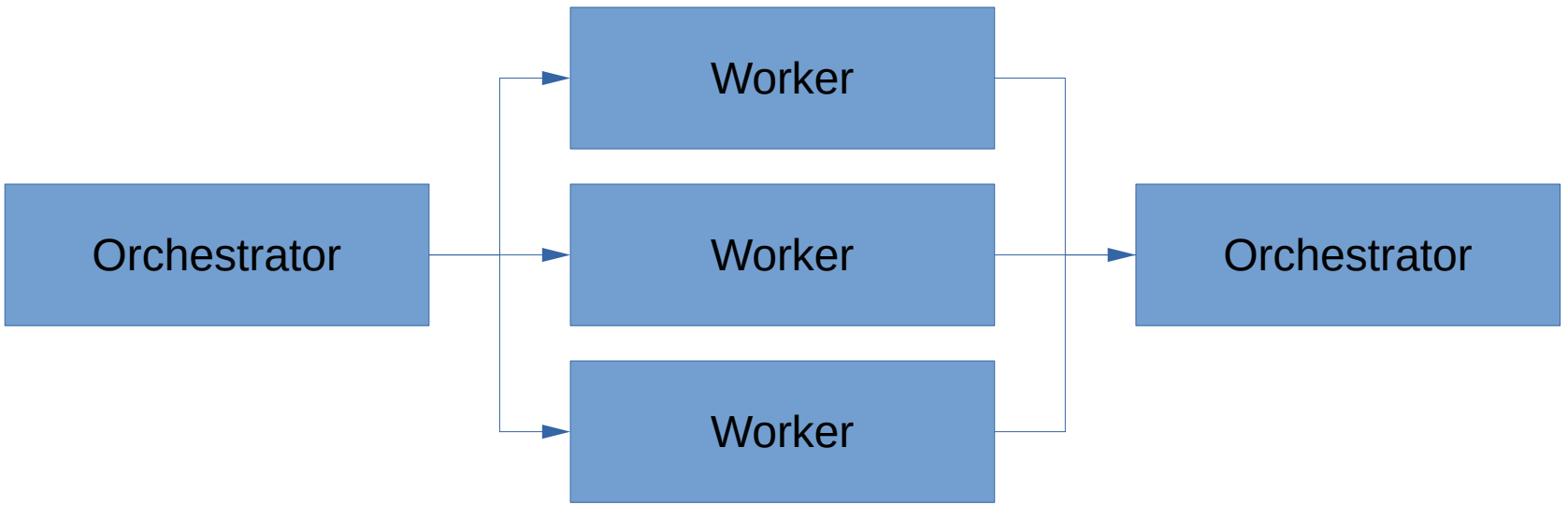
Work Verification

- Agents will miss issues in their own work, or forget to complete all tasks
- Advisor agent can keep them on track



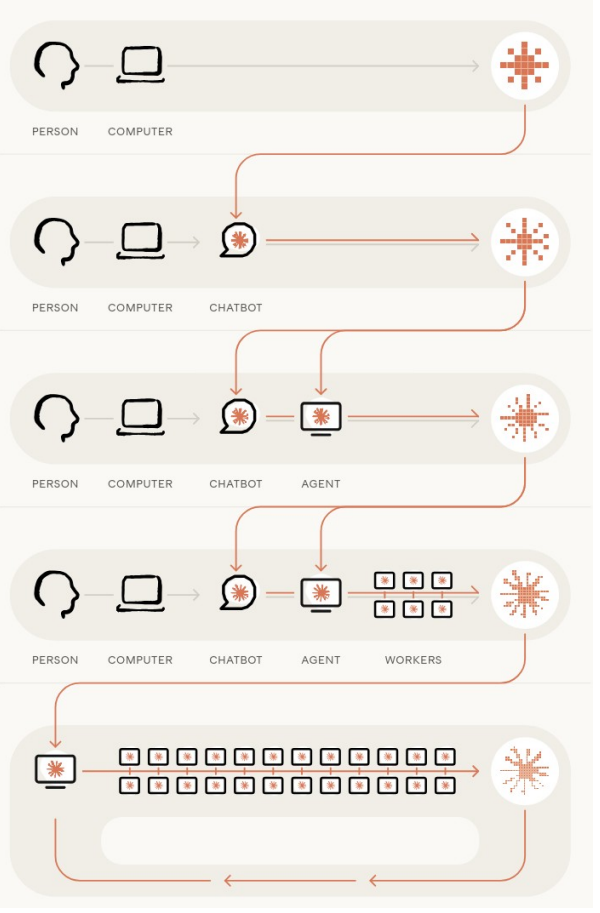
agent has a boss!

Parallel Workstreams



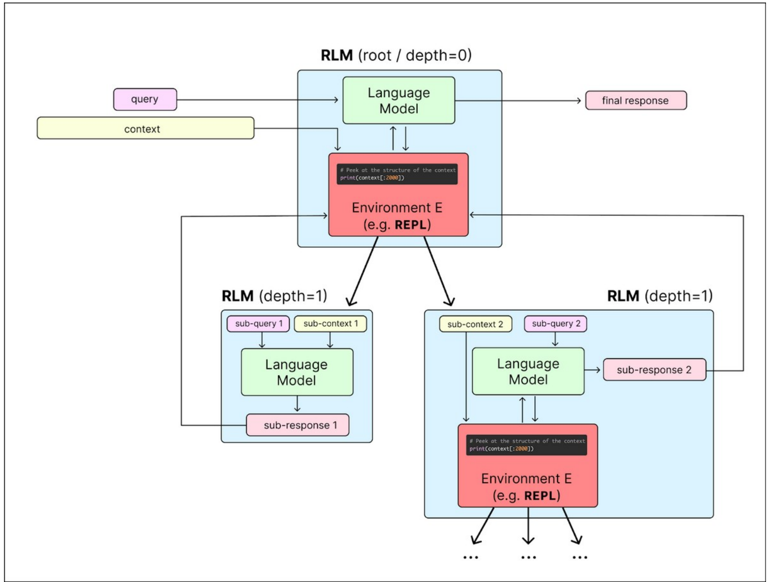
Recursive Language Models

- Agents are just programs
- Agents write code which can call programs
- Agents calling agents calling agents calling agents calling agents calling...



OH NO IT'S ALL AGENTS

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA



Let’s get back to reality

Parallelism is *hard*

In code:

- Race conditions
- Deadlocks
- Ahmdal’s Law
- Synchronisation
- Overhead

With agents, all of this again but also:

- Context mismatches
- Lack of visibility
- *Context* synchronisation
- Orchestration overhead

Human in the loop

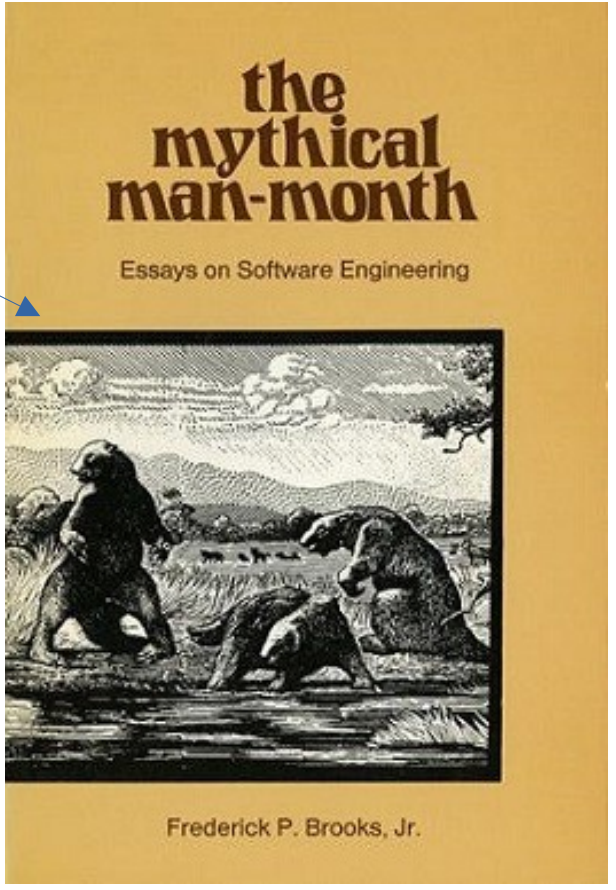
- Multiple agents working in tandem = increasing divergence from our reality

- Large-scale systems become impossible for one person to understand

Verification systems need to scale alongside swarms.
Everything needs to stay *grounded*

Orchestration must scale

Adding manpower to a late software project, makes it later



One agent solving a problem in an hour doesn’t mean 5 agents solve it in 12 minutes.

The number of workers must be proportional to the task!

What do swarms need?

- Sufficiently sized problems
- Deterministic verification systems
- Shared context bases
- Well-designed orchestration and scheduling
- Dynamic access to filesystems, compute and workspace segregation
- Self-healing when things break

Hold on that sounds familiar

kubernetes

Agents in the cloud

how i work with agents

Two types of agent

Macro-agent
Long-lived with a special role.

“Pet”

Micro-agent:
Short-lived worker with a dedicated task.

“Cattle”

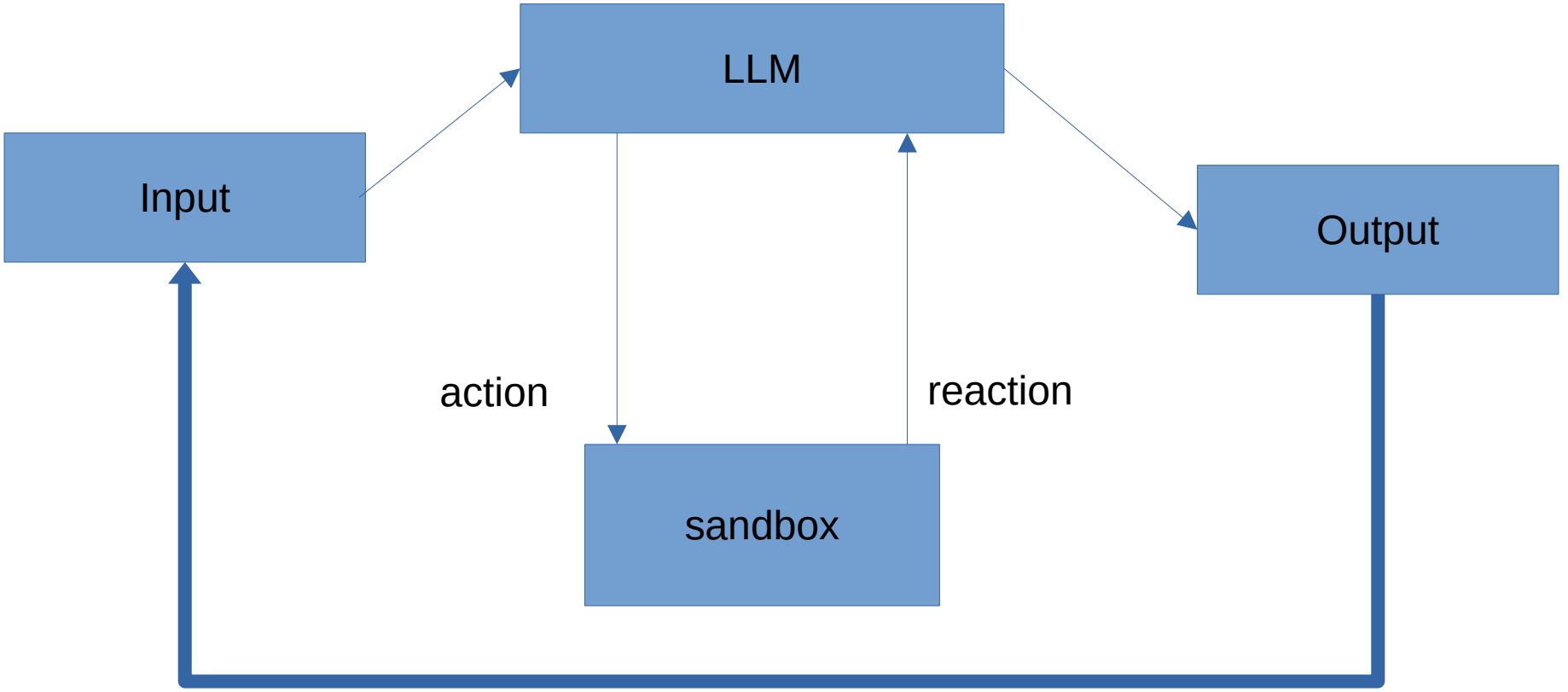
Programming as a HPC job

You currently schedule and submit calculations to HPC clusters.

Agents can now work for long timescales, programming workloads are now more like HPC batch work

“i got a refactor running rn”

Let’s go back to this diagram...



A macro-agent needs:

- A set of tasks + scheduling
- A memory system/context store
- A “sandbox” (computer, pod, etc)
- Access to the wider world

A micro-agent needs:

- An ephemeral sandbox
- Observability
- Fitting into a proper workflow
- Access to only what it needs for its task

agent is bee!

How I work with agents

- Remote into a server and attach to an agent
- Have a discussion with the agent, or point it to an external context store containing work
- Agent is trusted to do work, whilst I coordinate with other people and do the work it *can't* (there's a lot!)

Stuff I do instead of programming

- Designing architecture and UI!
- Testing code!
- Reviewing generated code and outputs
- Getting feedback from other people
- Coming up with new ideas
- Improving harnesses and engineering workflows
- Deploying AI models
- Designing memory systems!

DOING STUFF WITH OTHER PEOPLE

Conclusion

- Agents are really powerful technology!
- They are also really *weird* and have interesting failure modes
- Fitting them into actual work requires thoughtful design
- Agents automate *tasks* but their detachment from reality means people are *necessary*
- I probably have some form of cyberpsychosis, it may be necessary for my job at this point

Thank you for listening to me yap at code club again

Your continued support is very appreciated

I am normal and can be trusted with a slideshow