Llm can generate and then verify or get spec/constraints and then generate and compare to known knowledge or just RAG - key facts should have CSF verification riskiest assumptions and 6 fold quality metrics using prompt engineering + workflows - only generate quality metrics if needed and flag for human review.

Templating and higher order functions gives you the level of abstraction/flexibility vs functionality and speed - automation/solution tradeoff.

Self engineering systems by selecting only a handful of tools from the tool library. Only these are spun up at run time.

Iterative or prototyping mindset - find the MVP then build out functionality. Test as you go.

AI can upskill and use CSF to guide decisions and cognition based on constrained actor theory. Learning through any one of many different memory/learning tools - just pick one or two for a given use case.

Core library with customisability. We need to build out common workflows for maths, science, engineering etc have prebuilt code blocks so time to utility is kept as low as possible. + divergence convergence for creative problem solving.

Basically when we have a task, we build and agent and select tools for the agent and write workflows based on langgraph. We then chat with the workspace as a RAG pipeline. We use guardrails, handoffs etc. Just use openAI agent sdk where possible.

We use lite llm for token tracking with graphana and promethus for worker agent.

We can still start and stop the worker as previously or we can run continuously and can open a trace for each worker agent to view actions in real time.

In the demo, we build a set of task lists for the worker agent to solve. We have a list of action items which display in the chat as accordions for the user to review and verify + provide feedback to guide and train the AI. It also learns from experience autonomously I.e. when it tries to do something and fails it figures out why it failed using root cause analysis and figures out how to prevent this in future. It also highlight some abstract learning by assigning this particular instance yo some broader class of errors. It takes this into account when solving problems in future.

We need to build an update able RAG pipeline which stores explicit memories of the system to allow this to go into the solving loop.

If you can't figure something out just pass the problem into the llm with context and automate the solution process.

Dspy for programming with llms, agentsdk for programming agents, crewAI for multiagent systems, langgraph for complex AI logic and stateful execution, langchain for llm apps, smolagents for thinking in code, superAGI for long running agents? Multiagent convos with autogen, semantic kernel for AI middleware,

The 3 primary components are reasoning with tools (proof, math, logic, coding, optimisation, sat, utility/system, reflexive/self-awareness, file, API, quality assurance tooling, guardrails and csf, debugger), memory (RAG regex + experiential + database + context management + user psychology + collaborative intent discovery), learning (expert system, eurisko, agentic, workflows, upskill loop, RAG loop specialised knowledge, theory building, category theory, MDP control, fine-tuning, machine learning, evals, problem solving engine, governance, program synthesis, 5 fold decomp).

The only thing which cannot be automated is human intent discovery and alignment assurance.

With the agent upskill loop firstly build knowledge base from existing code. Then add tools as existing codebases based on research by system.

Chat agent tab 1 - workers tab 2

In chat agent, has tasks for human review (should guide user based on risks and what to focus on to.make these tasks as easy as possible) and chat with guardrails + stop button for generation and assets + progress bar and telemetry for chat responses.

Decompose task into a hierarchical goal tree and then backtrack with modularity and context management + forward projection of dependency info during solving. Combine with langgraph and taskmaster for this for complex control logic rather than simple sequential solving.

Test time inference compute scaling atomicity analysis.

Workers tab has current workers, metrics (status: on track/behind/ahead, paused for human feedback, stopped constraints exceeded, safety/ethics error, stopped no further progress possible) + milestones and assets + progress bar + continuous trace (thought + action + subgoal) + goal tree.

What if the AI worker can just ping your phone to review. Basically download the app and sync and it can notify you once it is done, how many tasks remaining and estimated time to complete + project status review.

Or even just get it to send an email and then reply to the email so you can leave it working on the go.

Demo version has free token limit - or may not be guaranteed only what compute is available. But otherwise users can use online version with their own backend or openrouter key as much as they want I.e. had all the functionality but is missing the ability to interact with desktop, only one worker at a time, and also user data only persisted for 1 month, each month we will delete your data stored on the server for privacy reasons or can upgrade to paid version to persist data indefinitely in online demo - demo only has one agent though. There is no upper limit to usage but LLM backend not guaranteed. Still able to run worker after logging out but will prompt the user to confirm. 1 worker, unlimited tasks or in paid unlimited workers and unlimited tasks.

Your greatest advantage is your lack of resources, it forces me to be more acutely aware of my needs as an SME trying to use AI in my own business to help me run it and offload responsibility to another system.

Essentially, it forces me to understand the customer because if I had all of these resources I wouldn't understand how hard this is and I wouldn't get my customers pain points. By being in this situation I can write up a detailed customer profile.

My customer is as SME which wants to get productivity and automation gains because right now they are struggling with their workload. They have to do everything and they are worn out, overloaded and struggling to balance the books. They have heard about AI and they are interested in its potential to help them address their biggest challenges but they don't know how yo use AI, don't trust it or their needs don't match the available products on the market. They don't have the money to build their own automation and chatbots don't have memory, hallucinate and can't reason reliably- also they want to be able to teach the AI how to do things in natural language rather than having to do prompt engineering which they don't understand. Existing solutions are expensive and free solutions like chatGPT are useful but fail to meet their specific business needs and are difficult to integrate into their workflows. They really need custom tooling and are interested in but don't really understand what AI agents are. They see the potential in the idea of having an AI coworker but there's no product on the market that would allow them to get this kind of out of the box functionality where there's no steep learning curve, complex AI jargon or difficult technical concepts to master. They want a low friction experience and the ability to set and forget so the AI can solve their biggest pain points safely, reliably and autonomously so they can focus on other parts of the business. For many of them, hiring is not an option due to the cost and they may be struggling to cope with their workload due to burnout so they really need the AI to deliver value fast and in a way that is easy to understand. The system should provide them with a level of support and personalised advice that makes it feel like you're working with a person and not an AI.

They don't want another automation snd they don't want to pay for all of these expensive and difficult to understand/learnt tools. They can't obviously build anything for themselves and they can't afford a forward deployment engineer.

Or could be a large Corp that wants to implement AI for its workers in frictionless way but doesn't want to have to build a complex tailored enterprise solution so they just give this to their workers so they can tailor the worker to their specific role needs without needing to hire an AI consultant to design custom solution ehich would be expensive and people may not like it/pilot may fail yo deliver value as per the MIT study. Also they don't want to have to constantly be using massive tokens amounts - so we need a system that gets less and less expensive the more you use it. Also not viable to be heavy user of token providers because it is too expensive to justify cost benefit - also need comprehensive specialised tooling depending on the domain vertical e.g. logistics, scientific or research. Need full powered solver.

The solution is software synthesis as machine learning- learn a program as tools for agent to get cheaper faster and better over time. And a problem.solving AI agent with a task list etc and simple familiar chat interface with cognitive tooling called by llm to address weaknesses of llms allowing this yo be unhobbled for business use and to improve quality of output.