Like the Homebrew Computer Club, fifty years on

- In 1975, around thirty people met in a garage in California.
- From those meetings came Apple, the Sol-20, and a generation who shaped personal computing.
- Something similar is happening with AI software engineering, right now. We'd like NI to be a part of it.

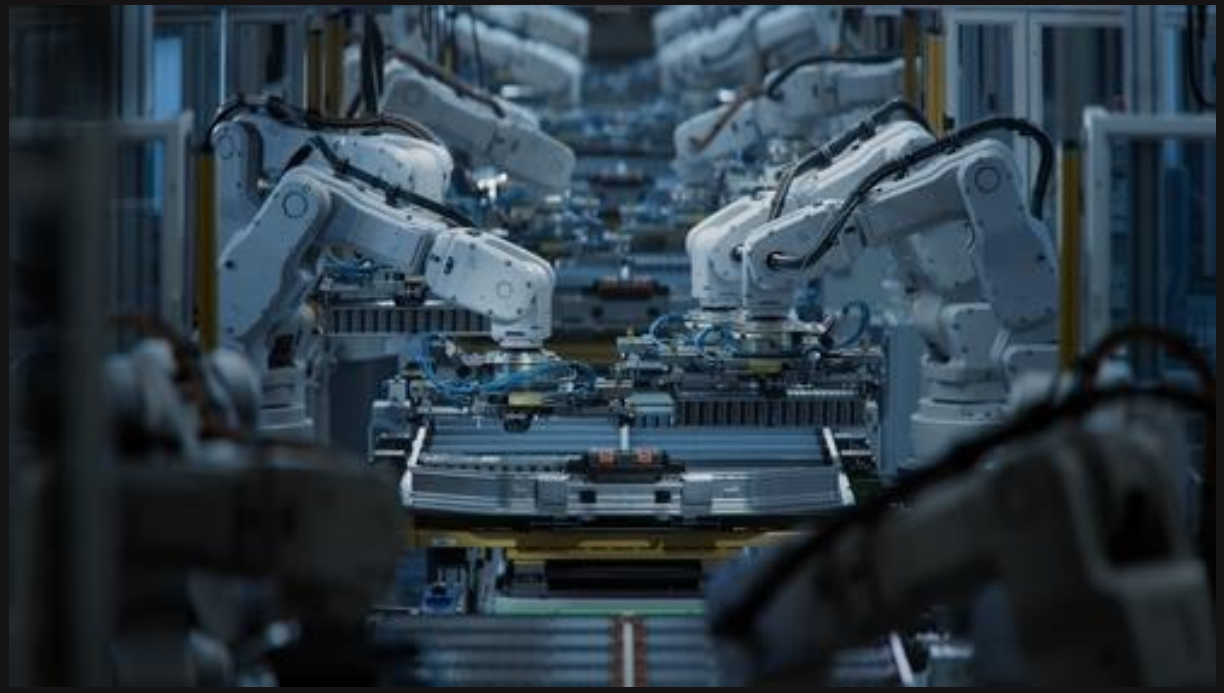
The next thirty minutes

1. What is fully automated software engineering?
2. How to build a FASE system
3. Baseline system
4. What we've been building and what we learnt
5. Questions and Feedback
6. How to join in.

What is fully automated software engineering?

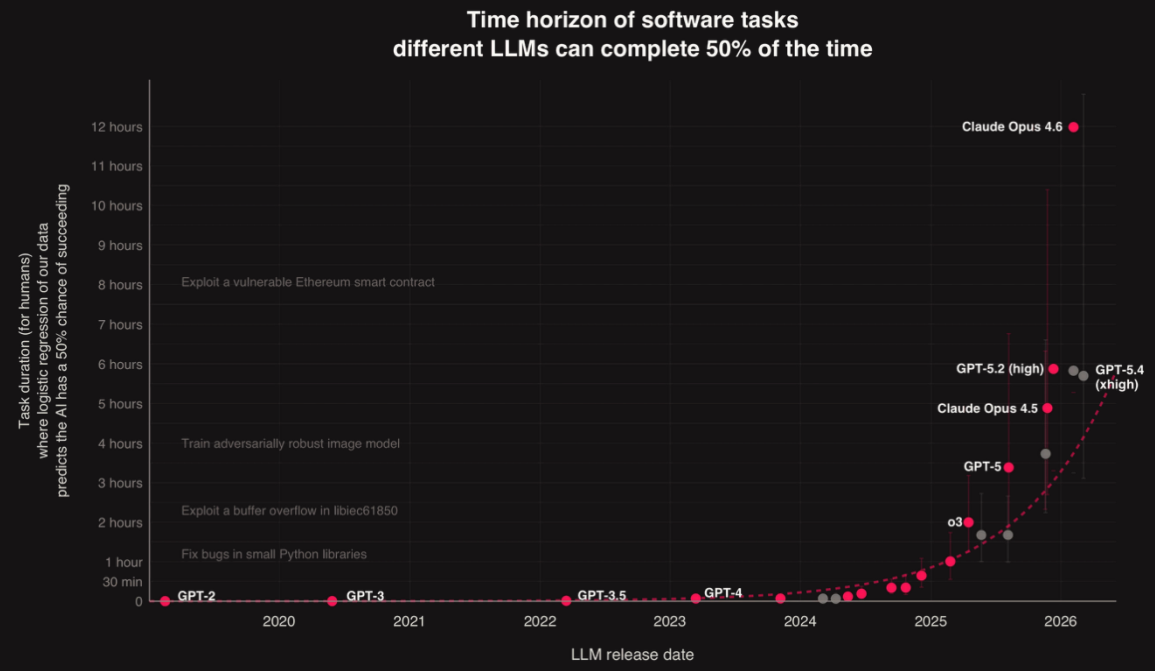
- A human writes one sentence: "Build me a task management app."
- An AI does the rest.
- It plans. It writes the code. It writes the tests. It runs the tests. It fixes the bugs. It says when it's finished.
- No human writes code. Ideally, no human reviews code either.

- And eventually…
- The AI entrepreneurially identifies what product to make
- And goes on to sell it



It's already happening at scale

- OpenAI shipped an internal product. Every line of code came from their AI. Zero lines written by hand.
- Anthropic ran 16 AI instances in parallel for two weeks. Output: a working C compiler, 100,000 lines.
- Stripe's AI agents merge over 1,000 pull requests per week.
- Spotify's background AI agent has merged 1,500 maintenance changes.
- StrongDM's policy: "Code must not be written by humans. Code must not be reviewed by humans."



How does it actually work?

- You give the AI a recipe written in English.
- The recipe says:
- Create a plan, create tests to verify the plan has been completed successfully, get the AI to follow the plan, run the tests if they fail get the AI to fix the failures, when everything passes finish
- The AI follows the recipe.
- The recipe is known as a ‘workflow’

What do we need to build to make it work?

- We need a way to control the AI, to get it to take many actions until the project is complete
- It needs to be able to plan/design the project
- It needs to be able to work through the design a part at a time
- It needs to be able to identify tests that will validate the software is complete and working reliably
- It needs a way to ensure that the tests pass and if not tell the AI what has failed so it can fix those problems

How to control the AI?

1. **Write code that explicitly calls the API**
   - A python loop that calls API, can be more expensive and doesn't use the extra tools and capabilities of the IDE tools
2. **Give it a tool to call to get instructions**
   - Use an existing CLI or vscode plugin and give it an MCP server. Holds tools, constraints, and design rules. The AI calls it when it needs to get the next instruction. Told to use the MCP by CLAUDE.md
3. **Wrap a command line tool with a program to simulate a user**
   - Similar to openclaw, wraps a CLI and parses the output to detect when task is complete and ready for more input. Can parse the AI output or tell it explicitly to print special instructions/codes when task is complete or to answer certain questions
4. **Give it a virtual machine and simulate a user of a GUI program**
   - Virtual machines controlled via VNC, using simple computer vision/image comparison to detect when the AI is ready for a new response, directly read from the IDE's own storage of the chat conversation to read back the AIs output

How does the AI know what to build?

- If the human just says "build me a writing app", what features should it have?

Possible solutions:

LLM guesses/makes up some requirements itself

Looks up features of similar products online

Use a similar product (via computer use AI) and reverse engineers its design

Have the AI simulate a user and create plausible walkthroughs they might want to follow to solve their ‘job to be done’

Get the AI to build a prototype first and then build the real project based on the prototype

How does the AI know what to test?

Possible solutions:

Use a second (possibly adversarial) AI that tries to break the product

Designs walkthroughs that match the design of the product based on the original requirements

Force the AI to look at the running app with screenshots, and check the layout actually rendered

For every feature, check there's a button or input a human user could actually click

Use static analysis tools

How do you stop the AI giving up?

- The AI will declare itself done early, and stop.
- It will pause to ask for confirmation, every five minutes.
- Some chat sessions even have hard timeouts (Claude Desktop kills sessions at fifteen minutes).

Possible solutions:

- Refuse to let the AI stop until a checklist of conditions is met (a "completion gate")
- When the AI does stop, automatically poke it to keep going (a continuation prompt)
- Save progress to disk so a fresh AI session can pick up where the last one left off.

How do you stop the AI breaking your computer?

- An AI agent can run any command. Install random packages. Write to random files.
- On a real machine, this gets risky

Possible solutions:

- Restrict what the AI is allowed to do, to a fixed list of restrictive commands.
- Run the workflow in a virtual drive restricted to operate within it (copying in any starting state)
- Run the AI inside a virtual machine. Throw the virtual machine away when finished.
- Have the system make backups and make it easy/automatic to restore if things are damaged

How do you afford to do this?

- Each AI call costs money. A full product build can be hundreds of calls.

Commercial cloud platforms charge a lot per build:

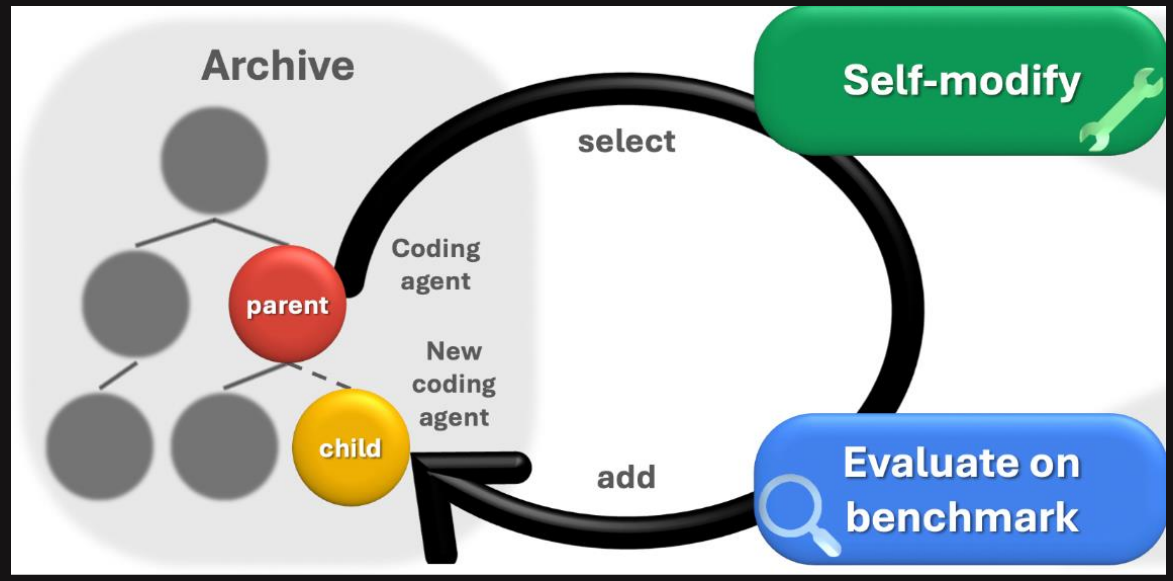
```
Lovable.dev:   about £11 per build (50% of daily credit
on the £22/month tier).
Bolt.new:      about £2.50 per build (10% of monthly
credit on the £25/month tier).
```

Daniel Campbell got it down to about 50p per build,

- by running everything inside an existing IDE on the standard £10/month GitHub Copilot plan.

Recursive Self Improvement

- Keep logs
- Have the workflow prompts and their sequence defined in an easy to edit data driven way
- Add quality judgement tools (AI based) to try to give the AI taste
- Have the AI review the logs of previous runs of the workflow and propose changes to the workflow to improve the design



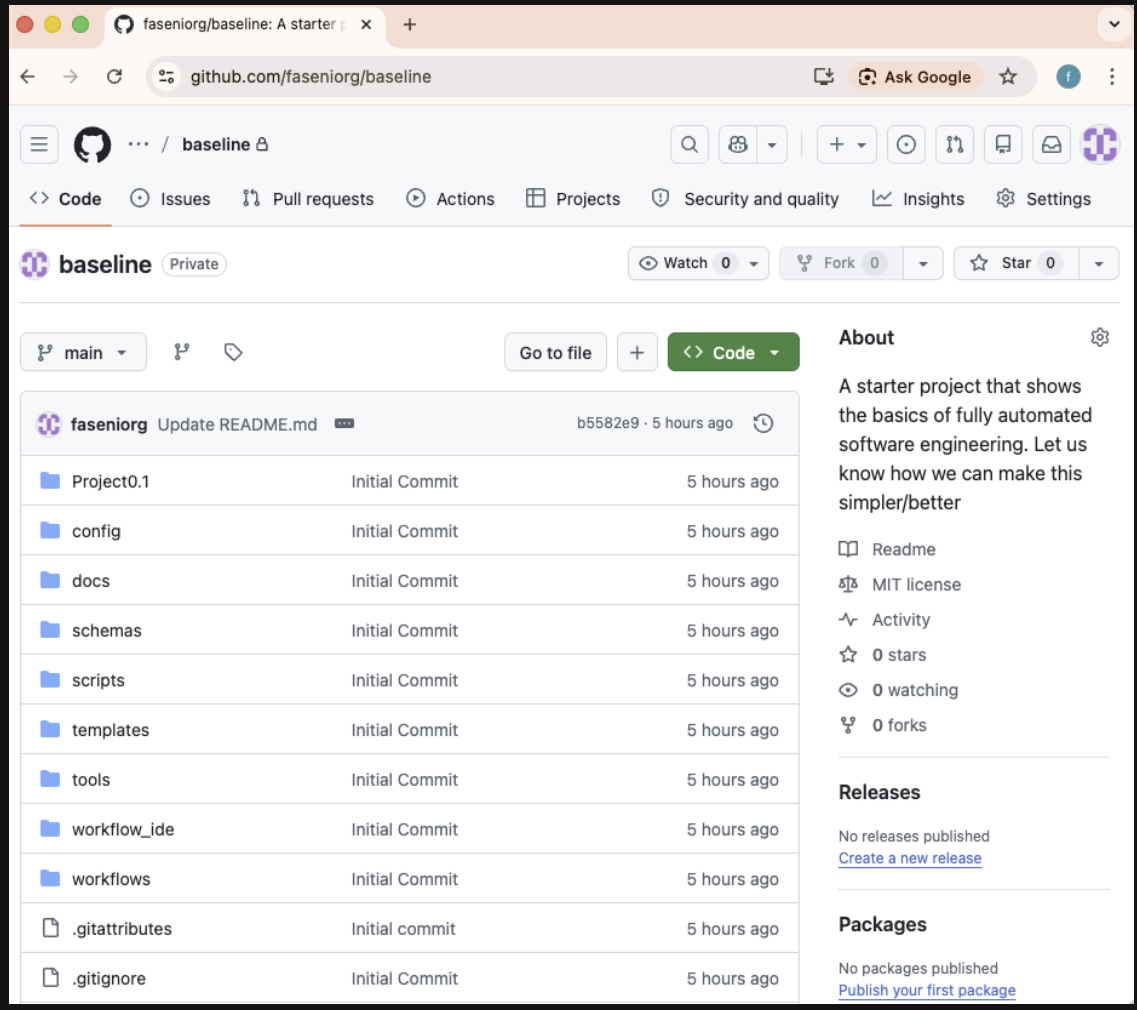
We have a ‘working’ baseline

- An open codebase that drives a coding AI through the seven stages from end to end.
- From an English description of what you want, it produces a working desktop app, with tests, a database, and documentation.
- Built as a teaching example. You can read the code, fork it, change it. Please help us make it simpler/better

`github.com/faseniorg/baseline`

Plan to expand this with new tools, focusing on:

- Judgement of quality (especially simulating user review)
- Goal of expanding the project so it can create a full product that can be sold



We have a ‘working’ baseline

- An open codebase that drives a coding AI through the seven stages from end to end.
- From an English description of what you want, it produces a working desktop app, with tests, a database, and documentation.
- Built as a teaching example. You can read the code, fork it, change it. Please help us make it simpler/better

`github.com/faseniorg/baseline`

Plan to expand this with new tools, focusing on:

- Judgement of quality (especially simulating user review)
- Goal of expanding the project so it can create a full product that can be sold

What the baseline actually built

- From the prompt: "a database connectivity checker for operators."
- Result: a working Tauri desktop app, with a real Postgres database, with automated tests, in 95 minutes wall-clock.

Four out of four critical tests passed.

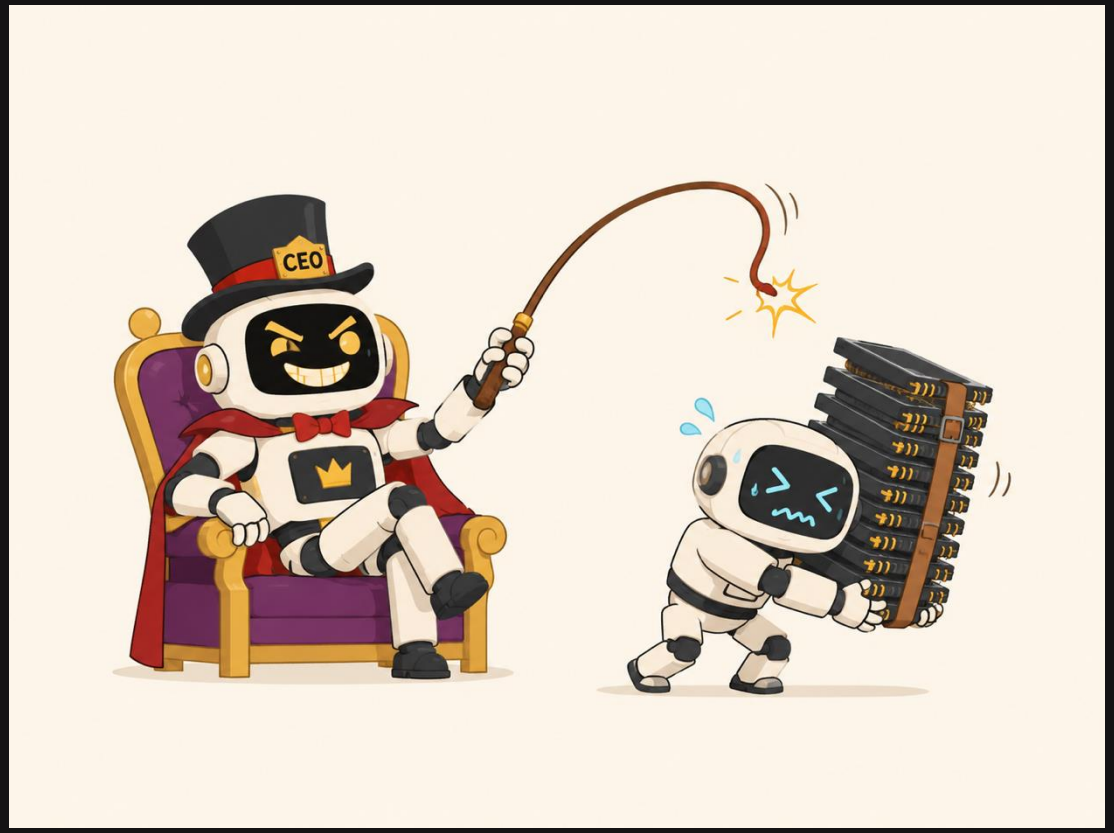
- The AI authored 335 lines of Rust, 291 lines of build config, the entire test suite, and the operator scripts.

- It is a trivial project
- It is not fully tested
- It is still far from being a commercial product
- But it’s a start

Baseline was built with human in the loop recursive self improvement

- Reveals interesting bias
- Once the AI realised that it was designing a workflow for another AI/Itself it became a domineering nightmare manager
- Was extremely prescriptive (undermining the AIs intelligence)
- Didn’t communicate any goals or purposes(acted like the AI was a dumb tool)
- Very critical and brittle, with no meaningful feedback to the AI
- Loved to say the project failed

- Needed to be taught to be respectful of intelligence, constructive about feedback, focusing on how to help the AI improve the system, a coach not a judge



Seven final-year students. Seven different approaches.

JAMES MCDONNELL.
3 Different ways of controlling AI IDEs
MAEVE McQUAID.
Discovered the importance of getting the AI to confirm that UI components actually enable the user to use features (not just have code with no way to run it)
DANIEL CAMPBELL.
Created a custom VScode extension to control Copilot chat to enable reliable multi step IDEs. Found prototyping helped. Found AI responds better to instructions it thinks are from humans
AIDAN JUMA.
AI run inside virtual machines, with eyes and a recursive self-improvement loop

Seven final-year students. Seven different approaches.

SHANE CULLEN.

Asked the AI to reflect on requirements. Asked the AI to critique the project. Discovered that some tech stacks were much more reliable than others.

NICOLE HANNA.

Two AIs, one writing code, one writing tests. Extensive examination of static analysis and quality tools as feedback to AI. Got the AI to create E2E tests

CIARAN McDERMOTT.

Created user journeys for requirements gathering and test generation. Found that having the AI create documents confirming it had completed certain steps reduced its likelihood of hallucinating it had completed requirements/tasks.

Without coordinating, our projects agreed on four things

- 1. The AI saying "done" before it's actually done is the central problem.
- 2. Don't let the AI grade its own work.
- 3. Write the AI's progress to disk. Don't trust its memory. Just log everything
- 4. The underlying tools will change next month. Need continuous maintenance and designs that sit on top of the leading tools (rather than integrate into them)

Any Questions?

https://app.sli.do/event/dfRjDqquCJuZFmL48i54Th

What could we do to make this community a success?

https://app.sli.do/event/dfRjDqquCJuZFmL48i54Th

Thank you.

Our goal is to run these meetup events Quarterly
Next one in August
Continue the discussion on WhatsApp
Hang out and chat

faseni.org

[o ‿ o]

FASENI

WhatsApp group

[Close]