One of the things I have spent a lot of time doing is interviewing software engineers. This is obviously a hard task, and I don't claim to have a magic solution; but it's given me some time to reflect on what effective software engineers actually do.

1.The Software Engineering Loop

When you watch someone who knows what they are doing, you'll see them looping over the following steps:

1. Build a mental model of the requirements
2. Write code that (hopefully?!) does that
3. Build a mental model of what the code actually does
4. Identify the differences, and update the code (or the requirements).

There are lots of different ways to do these things, but the distinguishing factor of effective engineers is their ability to build and maintain clear mental models.

2.How about LLMs?

To be fair, LLMs are quite good at writing code. They're also reasonably good at updating code when you identify the problem to fix. They can also do all the things that real software engineers do: read the code, write and run tests, add logging, and (presumably) use a debugger.

But what they cannot do is maintain clear mental models.

LLMs get endlessly confused: they assume the code they wrote actually works; when test fail, they are left guessing as to whether to fix the code or the tests; and when it gets frustrating, they just delete the whole lot and start over.

This is exactly the opposite of what I am looking for.

Software engineers test their work as they go. When tests fail, they can check in with their mental model to decide whether to fix the code or the tests, or just to gather more data before making a decision. When they get frustrated, they can reach for help by talking things through. And although sometimes they do delete it all and start over, they do so with a clearer understanding of the problem.

3.But soon, right?

Will this change as models become more capable? Perhaps?? But I think it's going to require a change in how models are built and optimized. Software engineering requires models that can do more than just generate code.

When a person runs into a problem, they are able to temporarily stash the full context, focus on resolving the issue, and then pop their mental stack to get back to the problem in hand. They are also able to zoom out and focus on the big picture, allowing the details to temporarily disappear, diving into small pieces as necessary. We don't just keep adding more words to our context window, because it would drive us mad.

Even if it wasn't just too much context to deal with, we know that current generative models suffer from several issues that directly impact their ability to maintain clear mental models:

- **Context omission** : Models are bad at finding omitted context.
- **Recency bias** : They suffer a strong recency bias in the context window.
- **Hallucination** : They commonly hallucinate details that should not be there.

These are hopefully not insurmountable problems, and work is being done on adding memory to let them perform similar mental tricks to us. Unfortunately, for now, they cannot (beyond a certain complexity) actually understand what is going on.

They cannot build software because they cannot maintain two similar "mental models", identify the differences, and figure out whether or not to update the code or the requirements.

4.So, what now?

Clearly LLMs are useful to software engineers. They can quickly generate code, and they are excellent at synthesizing requirements and documentation. For some tasks

this is enough: the requirements are clear enough, and the problems are simple enough, that they can one-shot the whole thing.

That said, for anything non-trivial, they are not capable of maintaining enough context accurately enough to iterate to a working solution. You, the software engineer, are responsible for ensuring that the requirements are clear, and that the code actually does what it purports to do.

At Zed we believe in a world where people and agents can collaborate together to build software. But, we firmly believe that (at least for now) you are in the drivers seat, and the LLM is just another tool to reach for.