

writing / the stack report / 8 Aug 2026

It's OK. Do Your Thing.

Some (reluctant) thought on our AI moment

We're in an *AI moment*, to put it mildly.

I've been reluctant to write about AI here. The Stack Report is, quite deliberately, a view from the slow lane. And, we're told, taking it slowly isn't what AI is all about.

Mostly, though, I'm not sure there's any way, currently, to say anything about AI without it going badly.

There's a worrying divide in our community between the pro and the anti AI crowds. Say anything at all and you're open to attack from one side or the other. Too enthusiastic? You're a mark, a booster, complicit. Too sceptical? You're a dinosaur, in denial, about to be left behind. I'm pretty sure most of us are just trying to work it out, but even trying to say that leaves you exposed.

There's nowhere to go where the discourse isn't dominated. Posts that should be about some other topic entirely end up framed as about AI. Even the pieces that profess to be tired of the whole thing — *can we please talk about anything else?* — can't help but continue it. They're AI posts too. And so, of course, then, is this one. 🤖

§

Will and I were recording a Django Chat the other day. As it will, Orwell came up, on the newspapers in Spain:

Early in life I had noticed that no event is ever correctly reported in a newspaper, but in Spain, for the first time, I saw newspaper reports which did not bear any relation to the facts, not even the relationship which is implied in an ordinary lie. I saw great battles reported where there had been no fighting, and complete silence where hundreds of men had been killed. I saw troops who had fought bravely denounced as cowards and traitors,

and others who had never seen a shot fired hailed as the heroes of imaginary victories... I saw, in fact, history being written not in terms of what happened but of what ought to have happened according to various 'party lines'.

The horrors of war are of a different order to those of (even) the AI industry, but the epistemic structure of the media wrapped around it is the same.

Depending on which reports you read, the labs are either making or losing billions. The technology is here to stay, or about to collapse under its own costs.

Presumably LLMs aren't going away as a technology *per se* — the weights exist, the papers are published, you can run a decent model on a laptop. But if they cost more to run than folks are prepared to pay, then it's at least plausible they disappear *as we know them now*. A \$100-a-month subscription (say) might seem great value. But if the token spend behind that subscription is running into the thousands — and suddenly that's the price you'd have to pay — then at some point the calculus changes.

For every *the new model changes everything*, there's a company scaling back its AI spend over ROI concerns. For every benchmark chart going up and to the right, a study saying developers *felt* faster while measurably being slower. *Great battles reported* where there had been no fighting; *complete silence* where the money was lost.

How on earth are we meant to know what to believe?

§

The idea that programmers will all be out of work shortly isn't new. It's a story I've lived with my entire career.

When I was starting out, the big bogey was outsourcing. Primarily to India. The pitch was simple and, on paper, unanswerable: programming is labour, labour is cheaper over there, therefore the work goes over there. Write the spec in London or San Francisco, send it to Bangalore, receive working software at a fraction of the cost. The trade press was full of it. *The Economist* was foaming. The career advice was to get out while you could — retrain, move into management, anything but code!

Now, this wasn't a fringe prediction that fizzled. It was a serious, well-funded, decades-long effort. The global IT outsourcing industry today is worth something like \$640 billion a year — north of a trillion dollars if you fold in business process outsourcing. India's IT services exports alone run to roughly \$250 billion annually. This was a trillion-dollar

natural experiment in whether you can separate the *specifying* of software from the *building* of it.

Of course, we're all still here. So what happened?

There are plenty of contributing factors, and they're all real enough. The coordination tax: firms discovered they'd outsourced the cheap bit — the typing — and kept all the expensive coordination, plus added ten time zones and a contractual boundary. The wage convergence: good engineers in Bangalore quite rightly stopped being cheap. And then demand exploded. Software *ate the world*, the *addressable market* kept getting bigger, and Western developers, rather than losing work, saw their salaries grow throughout the entire period.

§

If you're a regular reader of the Stack Report, you might guess that I'd focus on a different story: that of (the impossibility of) specifying our software before we build it.

Spec to code fails for all the reasons waterfall approaches always have: **it's only in building the thing that you find out what you actually need**. The requirements aren't an input to the building. They're an output of it.

We talked about this in Shipping Software on Time and on Budget — the whole game is doing sufficient *discovery* to be in a position to *deliver*. And it's why, in Locality of Behaviour, I argued for deferring your abstractions while new code is still in flux: you're buying time for the real shape of the problem to emerge. That shape isn't in the ticket. It isn't in some PRD. It emerges from contact with the work.

This was Brooks' point in No Silver Bullet: the essential complexity of software is in the *conception*, not the expression. By the time you've specified the behaviour precisely enough for a disinterested third party to implement it without judgement calls — no questions, no clarifications, no "did you actually mean...?" — you've done the programming. At that point, the spec isn't some folder of markdown docs. It's the running implementation.

Outsourcing never took that problem away. AI doesn't either.

Everything short of a running program is delegating decisions. Delegating decisions requires shared context: someone inside the feedback loop, talking to users, pushing back on the ticket, noticing that the requirement as written can't be what's wanted.

Which is exactly what happened, in the end, with outsourcing. India™ moved up the value scale. There are loads of great engineers there. But they need to work closely with you — in the standups, in the loop — just the same as your Western engineer does. And once you're paying for embedded judgement, rather than remote typing, the cost doesn't end up significantly cheaper. Judgement-in-context gets priced globally.

Contrast that with where outsourcing, on the simplistic model, genuinely worked. Precisely the work where the spec is stable and external: keeping the mainframe alive, SAP maintenance, regulatory-driven change. Work where the feedback loop is already dead or slow. Where there's nothing left to discover by building. The work that, yeah, OK, we can automate or outsource away.

§

It's the same with AI.

One idea of the moment is *Spec Driven Development*. You'll have seen it — tools like [GitHub's Spec Kit](#) that walk you through a workflow: write a specification, derive a technical plan, break it into tasks, and have the agent implement against them, with the spec as the source of truth.

And look — it's certainly interesting. It shows how sequenced prompts can steer an LLM through a structured workflow. How we prompt these tools seems most of the knack, and as a technique for keeping an agent on the rails... well... again, interesting.

But from where I'm sat, the fundamental arithmetic didn't change. The problem is still in discovering what it is that you're trying to build. Prototypes have always been useful for that. So, faster prototypes are presumably useful too. (Assuming their cost remains low enough.) The work, though, is in getting from the prototype, to the production code, and then keeping it there, and improving it after that.

The idea that such work — our work — disappears to the machine is the outsourcing bogeyman all over again. It's delightfully scary, if you like that sort of thing, but detached from reality, as often good bedtime stories are.

§

Which brings us back.

The noise is deafening, I feel it. But it's OK to ignore it, really.

You can't resolve the discourse. I can't either. The reports are pressing an agenda, the numbers are contested, and the ground shifts almost daily. Standing in the middle of that, it's very easy to lose your nerve — when every voice around you is saying one thing or another, you start to doubt your own beliefs.

But don't buy into the noise. You're the one doing the work. Trust your instincts. We'll still be here when all this settles.

It's OK. Do your thing.

The Stack Report goes out by email, approximately monthly. [Subscribe](#) ↗

newer

Deploy your fork

5 Sep 26

older

In search of a new contribution model

13 Jun 26

[All issues](#) ↗ [writing](#)