

Speeding up the past, or inventing the present

Most of what AI makes possible was never slow work. It was work nobody thought worth doing, so nobody missed it.

Isaac Correa, Madrid

DOI 10.5281/zenodo.23702117 · <https://isaac-correa.github.io/speeding-up-the-past/>
Licensed under CC BY 4.0. ORCID 0009-0003-0657-4419

I have never worked without AI, and yet for months I used it like someone who had spent twenty years working without it. I landed my first client a month after ChatGPT came out, at the end of 2022, while I was still studying marketing in Madrid. I had the newest tool there was, and I used it to do the same old things faster.

It took me a while to understand why, but it seems to me that almost everyone now uses AI to speed up the past, to do what was already being done, only faster, and very few use it to invent the present. And what AI lets you invent was almost never slow work. It simply wasn't worth doing, which is why nobody missed it.

A past that wasn't mine

Early 2023 brought ChatGPT Plus, which couldn't even search the web: if you wanted to work on a page, you had to paste the text in. I remember, with some amusement, that hardly anyone around me believed in it, while I spent so many hours with it that a classmate started calling me IsaacGPT. When I started working with clients I used it for everything, and for months I used it to build the same conventional solutions anyone would have built without AI, just faster.

So why did I do that, when I had no old habits to cling to? Because that past wasn't mine; it belonged to my profession. I have always believed that you learn by copying, the way you learn to paint by watching others paint. What I hadn't seen is that anyone who learns that way also inherits everything the people they copy never painted. Every profession comes with its list of tasks, and whatever isn't on the list never even comes up. With a new tool in hand, the instinct is to run through the same list faster, and the gap only shows once you stop copying the list and ask what is missing from it.

What was never worth it

I can't code, yet I've built most of the apps I use every day myself. I used to pay for a subscription to each of the apps they replaced and make do with software built for everyone, with hundreds of features I never touched. Nobody was going to build software for one person, because it wasn't worth it. Now I build it myself.

The same goes for companies. At the end of each month we send every client a report on how our work for them went. Preparing them all used to take one person the last ten days of the month; now they're automated, and she spends that time on other work. That is speeding up the past, and it's a good first step. But what the client needs is to find out what is happening while it can still be fixed, not thirty days later, and keeping a daily eye on every client never used to pay for itself.

More interesting still, a lot of routine work used to be sold as a separate service, because it took someone hours. Once it no longer takes those hours, what used to be a product becomes one feature of something much bigger, and what you end up selling is whatever no client could

afford before.

The other question

So if the tool is the same for everyone, what separates people is the question they ask about their own work. Someone speeding up the past asks how to do the same thing with less effort. Someone inventing the present asks what is worth doing now that wasn't before, or how to make a problem go away for good instead of solving it every time it comes back. This technology can only go as far as the imagination of the person using it, and imagination rarely strays beyond the list it inherited.

That's why I think younger people start with an advantage, though a smaller one than people make out. Someone learning a profession today has fewer habits to unlearn, but still inherits what that profession takes for granted, just as I did. And someone mid-career has more to unlearn, but can ask the other question too. All else being equal, experience will always win; what no longer makes sense is hiring someone to repeat the work of the past. In my essay "The End of Software" I wrote that junior roles would disappear. Today I would put it more precisely: the jobs that disappear are the ones built on repeating tasks, and young people will still be hired for everything else. Anyone who spends the day stringing routine tasks together has to change how they work, whatever their age, and their company has to give them time to do it. They will get things wrong at first, just as the person who held the job before them did.

Judgment

The real objection is judgment. If someone starting out today never does the work by hand, where does their judgment come from? Teachers used to complain that calculators would make us forget how to do arithmetic, and they were right: hardly anyone does long division by hand. Where they went wrong was in what that meant, because judgment was never in the arithmetic; it was in knowing which formula to use. The difference with AI is that it makes mistakes, sometimes with great confidence.

I don't write code, but I decide how the programs I build for myself should work, and I built that judgment by correcting my agents whenever they got something wrong. I don't have a full answer to how anyone will learn a profession this way, but I do know that the judgment that counts comes from judging the machine rather than repeating what it already does.

The bet

I'm not a neutral observer: at Hellomatik we do exactly this kind of work for other companies, the kind that never used to pay for itself. My bet is that companies swamped with work today will stop being swamped and will put the time they free up into growth, until they are swamped

again, but with clients, services and projects they couldn't afford before.

None of that comes from doing the same old things faster. I try to hold myself to this: every time I automate something, I try to ask what I could do now that I would never have considered before.

Earlier essays by the author: "The higher self", DOI [10.5281/zenodo.23663934](https://doi.org/10.5281/zenodo.23663934); "The End of Software", DOI [10.5281/zenodo.23468806](https://doi.org/10.5281/zenodo.23468806).