AI in Software Development

Arthur Bit-Monnot

INSA Toulouse

2026-01-07

Disclaimer

- not an expert in AI-assisted coding
- not much hindsight yet (but some tendencies)

Will not cover energetic impact of wide-spread usage

- *Shift Proejct*, Intelligence artificielle, données, calculs : quelles infrastructures dans un monde décarboné ?

A provocative statement

Proposition 1 Your LLM is a better programmer than you.

LLM vs Computer Science Student

LLM:

- fast
- very knowledgeable
- restless
- cheap

CS Student:

LLM vs Computer Science Student

LLM:

- fast
- very knowledgeable
- restless
- cheap

CS Student:

Given enough time, will solve *any* problem:

- more creative
- better learning
- larger “context window”

LLM vs Computer Science Student

LLM:

- fast
- very knowledgeable
- restless
- cheap

CS Student:

Given enough time, will solve *any* problem:

- more creative
- better learning
- larger "context window"

Proposition 1 Actually an LLM is nothing without a developer giving it purpose and directions.

LLMs and Junior Developers

Proposition 2 An LLM is similar a good junior developper:

- brilliant, obedient

but still requires a senior developper for

- direction (prompting)
- reviewing

LLMs and Junior Developers

Proposition 2 An LLM is similar a good junior developper:

- brilliant, obedient

but still requires a senior developper for

- direction (prompting)
- reviewing

Proposition 3 Speed is not the bottleneck in producing quality software

LLMs and Productivity

Study on Experienced Programmers

Scientific study[1]

- experienced programmers
- on known open-source code base

Task:

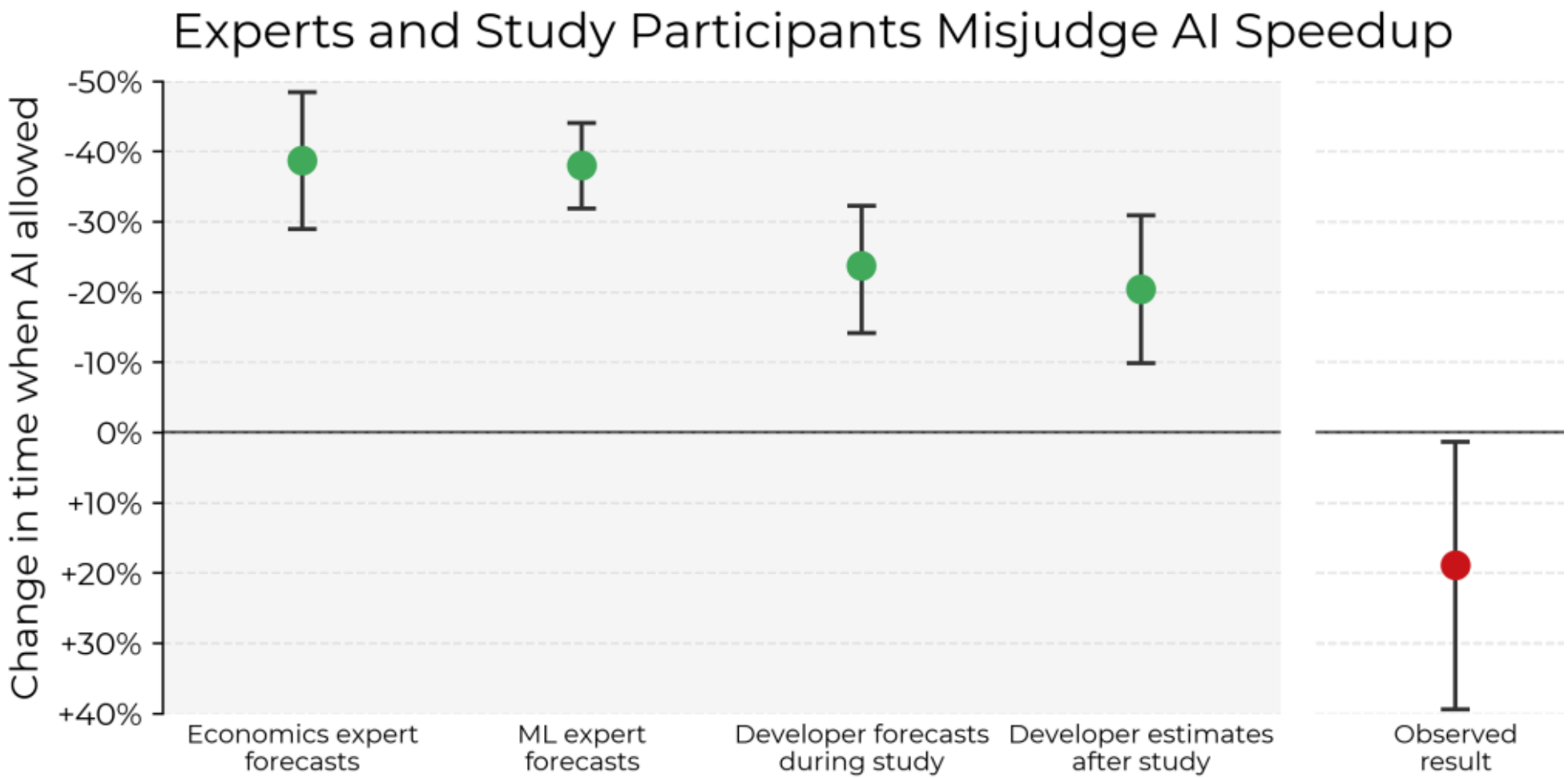
- fix a (real-world) bug in the code base (among 246)

Measures:

- time gained when given access to AI tools

[1] Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

Study on Experienced Programmers



Study on Experienced Programmer

Danger

An extreme case:

- experienced devs
- on large codebases their know

Key take-away:

- developers are not disappearing
- biased perception of productivity

The Productivity Bias

The 70% problem[1]

- Purely automated code generation can get you for an idea to 70% of a product in hours.
- But the last 30% requires experience and deep understanding of the code base (not acquired when building it).

Productivity & Excitement ⟶ Frustration ⟶ Abandon

[1] https://addyo.substack.com/p/the-70-problem-hard-truths-about

Usefulness of LLMs

- **Discoverability**
  - providing suggestions to developers (algorithms, frameworks, APIs)
- **Automating**: Writing boilerplate code
  - common functions (`toString`, ...)
  - bindings (exposing C API to python, ...)
- **Acceleration**: Writing ephemeral code
  - prototyping an idea
  - script to visualize data for a report
- **Assistant**: general programming task
  - under constant review and corrections
  - really empowering for *experienced programmers*

The dangers of LLMs

Assumptions:

- an LLM output quality is **strongly correlated** to the experience of the developer driving it
- the experience of a developer comes from actually writing code
- at this point of your career you will feel most productive when letting an LLM do all the work

The dangers of LLMs

Assumptions:

- an LLM output quality is **strongly correlated** to the experience of the developer driving it
- the experience of a developer comes from actually writing code
- at this point of your career you will feel most productive when letting an LLM do all the work
- you are learning and productivity should be the least of your concern
  - learning is tedious, but…
  - what you learn know will be useful for the rest of your career