Memos

Memo on *“How LLMs affect developers’ intuition?”*

LLMs seem to emerge as a learning tool for software developers based on P12, P14, and P15, e.g., relying on LLMs to cover gaps in knowledge. At the same time, P14 argues that while using LLMs for learning, novice developers should do the majority of the work. This practice seems to be necessary to improve novice developers’ skills, but also to cultivate the intuition inherent in experienced developers. Intuition is something that comes naturally after fully understanding something. That intuition (internalised knowledge) helps to avoid scenarios where LLMs are misleading, which P11 described as ChatGPT being a very good liar

Memo on *“Which software development tasks would you delegate to LLMs?”*

P5, P12, and P13 mentioned using LLMs for tasks they prefer not to waste time on, such as document generation. P1, P4, and P12 mentioned relying on LLMs for tedious tasks, while P2, P13, and P19 rely on LLMs for straightforward tasks. It seems software developers are inclined to lose grip of tasks that are at a level of complexity they believe LLMs are capable of dealing with. But when it comes to full delegation, what tasks would developers fully delegate to LLMs?

Memo on *“How and Why people try LLMs”*

Contextual info from P2 (someone working in personal projects) suggests P2 is someone who learn "on the side" by doing small projects and experimenting with new tech (C++, Java, Python). P2's ATAI scale suggests P2 has a small fear of AIs, and relatively little trust in AI, but also believes in AI's potential to help and damage humankind. P2 and P5 follows LLM evolution on online communities. P3 and P5 mentioned the popularity of LLMs on social media, motivating them to try out. This aligns with the findings from the paper “It would work for me too”: How Online Communities Shape Software Developers' Trust in AI-Powered Code Generation Tools", which explores the influence of social communities in software developers’ trust in AI tools.

Memo on “*Is this really pair programming?”*

P18 and P19 mentioned AI sycophancy limiting interaction. P19 believes that LLMs are very polite, which limits the potential of AI pair programming experiences. For pair programming, it is important to have that clash of opinions. But with AI sycophancy, that might not be possible. According to P6 and P19, traditional pair programming comes with unexpected learning opportunities, founded in that clash of ideas. On the other hand, P7 mention that LLMs may not listen to software developers, making them spend a lot of time trying to show that they are providing a wrong suggestion. That “LLMs' stubbornness” is different from when developers try to defend their ideas during pair programming sessions. P5 also mention that, when doing pair programming, it is common to present the ideas at a human-understandable level, which might lack contextual information or be ambiguous to LLMs. P22 defends that the closest that AI tools may provide a pair programming experience is via AI code review.

Memo on *“What does it lose by not using LLMs?”*

P6, P8, and P18 mention losing gains in productivity, while P1 mention losing gains in innovation and time. However, P5 defends that self-reliant software developers would have long-term benefits, such as improvement in soft skills and expertise. It seems the loss in gains is deeply related to the expertise level. For senior developers, who have more mature skills and are towards the end of their career, they would be less impacted.