LLM-supported development and troubleshooting

20th International dCache Workshop, May 6-7, 2026, Nikhef

Artur Gottmann | May 6, 2026

Prologue: Some information about me

- After my Ph.D. in high-energy physics, I've worked since July 2021 as CMS representative at GridKa
- In addition to my regular duties: a focus on the interface between dCache and HPSS tape system
- In June 2025, I've joined the dCache admin team, taking over from Dr. Samuel Ambroj Pérez
- My preparation a few months before my onboarding as dCache admin:
  - Read The Book
  - Setup an all-in-one dCache VM, including a tape system imitation
  - Got familiar with the dCache configuration at GridKa
  - Read GridKa documentation of regular activities like storage downtimes, upgrades, pledge deployment etc.
- Obtained an overview quite quickly, but lack of years of experience

With some experience with LLMs from the past as user, I've started to use it for exploration also in the dCache context

LLM Overview

- Proprietary reasoning models, dominated by US companies
  - OpenAI: GPT-5.5 (xhigh)
  - Anthropic: Claude Opus 4.7 (max)
  - Google: Gemini 3.1 Pro Preview
- Open weight reasoning models, mostly from Chinese companies
  - Kimi: Kimi K2.6
  - Xiaomi: MiMo-V2.5-Pro
  - DeepSeek: DeepSeek V4 Pro (Max)
  - Z AI: GLM-5.1
  - MiniMax: MiniMax-M2.7
  - Alibaba: Qwen3.6 27B
  - Mistral (French): Mistral Medium 3.5

- Focusing on models with **reasoning**, since they perform usually better, when a multi-step approach is more suitable to complete a task.
- Proprietary models have strongest capabilities, but cost **a lot**
  - For occasional use, it might be worth getting Github Copilot
    $\rightarrow$ Pro plan is free for researchers - you need to expose your employee card and location, though
- Open weight models could be **self-hosted**.
  - However, not all of them fit on a usual power-computer, in particular those competitive to proprietary.
  - Typically, providers able to host them have cheaper costs
  - Public institutions like universities may be able to host those LLMs for free. Example: KI-Toolbox at KIT

Output quality depends strongly on the LLM - so pick your poison well!

Working with LLMs in the Software Area

Levels of Offloading work to LLMs (my definition)

- Level 1: Using only the chat interface → limited in capabilities, mostly useful to collect ideas
- Level 2: Chat with websearch and code interpreter → more powerful, but still inconvenient
- Level 3: Use CLI with full access to an **isolated** work directory → Quite powerful for the following use-cases:
  - Code summaries and code exploration
  - Scanning code for troubleshooting and creation of issue reports
  - Creating small scripts to resolve a certain task with clear inputs and outputs
- Level 4: Provide the CLI with extended set of agent tools (Github/Gitlab, Websearch, Browser, Databases, ...)
  → Allows for more targeted actions to complete a task. Example: Write tests for a repository, commit, and push
- Level 5: Assign roles to LLM agents, allow them to spawn further agents if necessary
  → If properly orchestrated (Human-in-the-Loop), can be very useful to plan and develop software projects
- Level 6: Setup a workflow for autonomous software development, which can spawn LLM agents with different roles
  → It is possible to mimic almost a complete development cycle (and yes, some people are going in that direction)
- Beyond Good and Evil: Setup OpenClaw connected to your subscriptions, and let it burn your money
  → just a joke, **DON'T DO THAT** for real...

The higher the level, the less control you have as a human, and the more difficult it is to follow the steps

My experience with LLMs so far

- For me, tasks at Level 3 were quite useful
  - Writing scripts to copy to `dCache` concurrently, release pins via API, gather billing information from opensearch
  - Study and explore certain aspects of `dCache` code: interplay core $\leftrightarrow$ zookeeper, debug level at runtime for certain components
  - Perform troubleshooting: what is leading to error 429 (too many requests) at WebDAV doors and frontend?
    $\rightarrow$ In fact, a high fraction of my `dCache` github issues had a troubleshooting iteration with LLM support beforehand
- Level 4 and 5 were leading to results too, but with more need of intervention
  - (Re-)Writing parts of existing code (changing a function, restructuring, tests) works quite well
  - Reorganizing entire codebase - even if small - may get more challenging, but possible
  - Adressing github/gitlab issues strongly depends on the complexity and information given
    $\rightarrow$ The clearer and the more concrete a given task, the better the results
  - Writing new code **is** challenging: you need to know what you want, otherwise you will be disappointed
    $\rightarrow$ But well, I was able to get decent and **working** HSM provider & staging orchestrator prototype without touching the code much
- Didn't go to Level 6 and beyond

Major challenges here: limited context windows and non-negligible halluzination rates of the LLMs

- LLMs *forget* often what you've said at the beginning
- If there is no concrete answer, but you insist, there is a problem, they often make up things

Summary, thoughts and points for discussion

- So far I'd say that at least for simple and concrete tasks, LLM support was quite useful for me and made me faster
  → Not sure about more complex tasks here - still trying to find a sweet-spot
- On my opinion, AI is not just a hype. It is challenging areas like science, IT, cybersecurity, medicine, law, ...
  → So we might need to think how to work with it already now
- Some questions:
  - What do you think of creating github issues with LLM-support?
  - What about addressing issues with LLMs?
  - What about pull requests with LLM-support?
  - What about development in general with LLM-support?