26th International Congress on Modelling and Simulation, Adelaide, SA, Australia, 30 Nov–4 Dec 2025
mssanz.org.au/modsim2025

Modelling buddy - making large language models competent assistants in environmental modelling

J.-M. Perraud [a], A. C. Freebairn [a], J. C. Bennett[b] and D. E. Robertson[b]

[a] *CSIRO Environment, Canberra, ACT*
[b] *CSIRO Environment, Clayton, VIC*
Email: jean-michel.perraud@csiro.au

Abstract: Large Language Models (LLMs) are sweeping through many aspects of our work and personal lives. The pace of innovation has been very fast over the past two years, bringing capabilities yet to realise their full potential, particularly in the development and deployment of software. Most environmental modelling software remains niche and of understated complexity. LLMs have little to no reliable information on these pieces of software on which to train. We observed that producing code for niche environmental modelling software using an LLM without additional context led at best to a refusal and at worst to sycophantic hallucinations. To make an LLM competent in a domain it was not trained on, we need to provide relevant context alongside prompts, typically -but not limited to- plain text using markdown format. This requirement is very generic, yet we initially found that tools were lacking for providing such information in a form useful for LLM inference. In this talk we outline our experience in how we can make LLMs more competent assistants for one niche piece of environmental modelling software: the **Swift2** python package (Perraud, 2025). Our aim is to use LLMs to help author streamflow modelling and forecasting workflows. We initially trialled adding context derived from a full-length markdown version of the **Swift2** reference documentation. We adapted an exemplar from a course in "dialog engineering" with the SolveIt platform (Howard and Whitaker, 2025). SolveIt is a read-eval-print (repl) system like jupyter that adds the capability to intersperse code cells with prompts for LLMs. We iteratively engineered our context, trying to make the backend model -at the time Claude 3.5 Sonnet- more competent for our goals. With a deliberate approach to let the LLM do the coding tasks without "spoon feeding", we managed to obtain a valid basic hydrologic simulation. We tried to continue to a more complicated task to perform a model calibration. To our surprise we ran into the maximum context window available for the LLM, despite it being a reasonable 200,000 tokens.

Longer context windows have appeared over the past year but come with extra compute resource and monetary costs and cannot be a substitute to proper context engineering. Our hypothesis is that it is preferable to dynamically provide more targeted context for each step of modelling task. We are experimenting crafting context material using a semi-manual form of Retrieval Augmented Generation (RAG). Initiatives such as the https://llmstxt.org/ convention and the Model Context Protocol (MCP) https://modelcontextprotocol.io/ have brought the foundations to streamline this process to more standard means to provide contexts to LLMs, and devise domain specific coding assistants e.g. https://github.com/awslabs/mcp. Our experiments have so far been done with the larger LLMs such as Claude Sonnet. It is worth noting that recent, smaller models at or below the ~30 billion parameters range (Devstral, Gemma-3 27B, Qwen-3, etc.) are increasingly capable and may offer affordable new avenues where fine-tuning is complementing context engineering.

1. REFERENCES

Howard J and Whitaker J (2025), SolveIt: The Thinking Developer's Environment, https://www.youtube.com/watch?v=DgPr3HVp0eg

Perraud J-M (2025), swift2 2.5.3 python package documentation, https://csiro-hydroinformatics.github.io/swift-py-doc

Keywords: *LLM, specialised software, context engineering, code generation*