

ChatPharo: an Open Architecture for Understanding How to Talk Live to LLMs

Omar Abdelkader ¹

¹Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRISAL, F-59000 Lille, France

4th July 2025



Special Thanks !

- Benoît Verhaeghe
- Kasper Østerbye
- Valentin Bourcier
- Kris Gybels
- Balša Sarenac
- Srajal Dwivedi



Who am I?

- A PhD Student working on the Complishon engine for Pharo
- An Artificial Intelligence engineer
- Interested in support for Low-resource languages: How to gain more using LLMs in Pharo



What is LLM?

- LLMs are neural networks with billions of parameters, **trained** on **large-scale** and diverse datasets.
- They excel at understanding and generating human-like text.
- Can perform complex tasks such as code generation, following instructions, in-context learning, and reasoning.



How LLM generate code?

- Conceptually: Get trained and then use 'statistics' to do gigantic pattern matching to generate an answer
- Why LLMs are better for Java and JavaScript than for Pharo?
- Because they got trained on a **huge code base**.



How LLM generates code for low-resource languages?

- Let us focus on Pharo for now.
- Fine-tuning works best for small models (around 1B parameters).
- In-context learning works best for larger models (7B parameters and above), especially when using translation examples or few-shot prompts.
- Very large models (like 33B) can degrade in performance.



Why LLM is not ready to use for Pharo?

- **Data & model gap** here is little publicly-available code, almost no curated training sets or benchmarks, and current models show a steep drop in accuracy whenever training data are scarce or syntax is uncommon
- **Tooling & runtime gap** Pharo's live, image-based, highly-reflective IDE demands integration, package management, and runtime context that text-only LLM pipelines do not understand; today's models already struggle with repository- or project-level generation even in mainstream ecosystems.



How we can improve the LLM–Pharo integration?

- **Give the model fresh, live context** — wrap the prompt in a Retrieval-Augmented-Generation (RAG)
- **Cheap adaptation beats full retraining** — few-shot “translation-example” prompts
- **Let the LLM call tools inside the image** expose small helpers



What is Tool Calling?

- **What is it called in LLM jargon?** Tool calling refers to the process where an LLM invokes external functions or APIs to perform tasks it cannot handle directly.
- **How are executable tools triggered?** Tools are executed via structured API calls generated by the LLM, which are then handled by a backend system that runs the requested operation.
- **How is information retrieved from the image?** For image-based tools, the system can use image processing, OCR, or vision models to extract information from the image for further analysis.



How it really works? (User - AI)

```
{
  "model": "gpt-4o-mini",
  "messages": [
    {
      "role": "system",
      "content": "You are an AI coding assistant inside the user's Pharo image."
    },
    {
      "role": "user",
      "content": "Which packages in my image mention HTTP networking?"
    }
  ]
  ....
}
```



How it really works? (AI-Image)

```
{
  .....
  "role": "assistant",
  "content": null,
  "tool_calls": [
    {
      "id": "call_01",
      "type": "function",
      "function": {
        "name": "get_packages",
        "arguments": "{}"
      }
    }
  ]
  .....
}
```



How it really works? (Image - AI -User)

```
{
  .....
  {
    "role": "tool",
    "tool_call_id": "call_01",
    "content": "{\"packages\": [\"Zinc-HTTP\", \"Neo-JSON\", \"AI-ChatPharo\"]}"
  },
  {
    "role": "assistant",
    "content": "I found three packages that deal with HTTP:\n\n• **Zinc-HTTP** { the core networking stack.\n\n• **Neo-JSON** { \\\nserialisation helpers often used alongside HTTP.\n\n• **AI-ChatPharo** { your chat assistant, \\\nwhich wraps Zinc for its REST calls.\n\nLet me know if you'd like details on any of them!"
  }
],
"temperature": 0.7
}
```

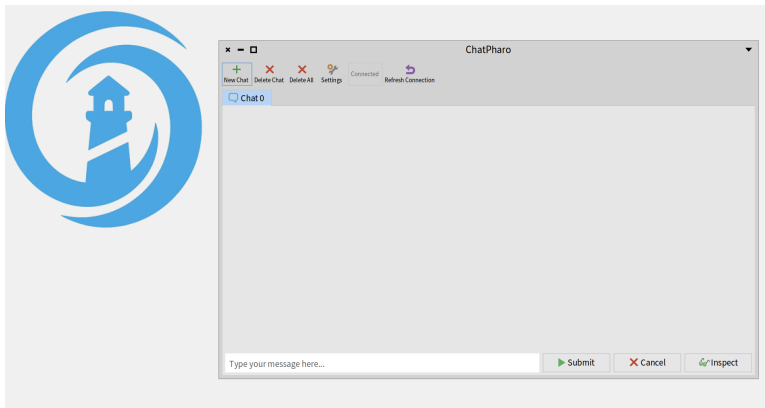


What is ChatPharo?

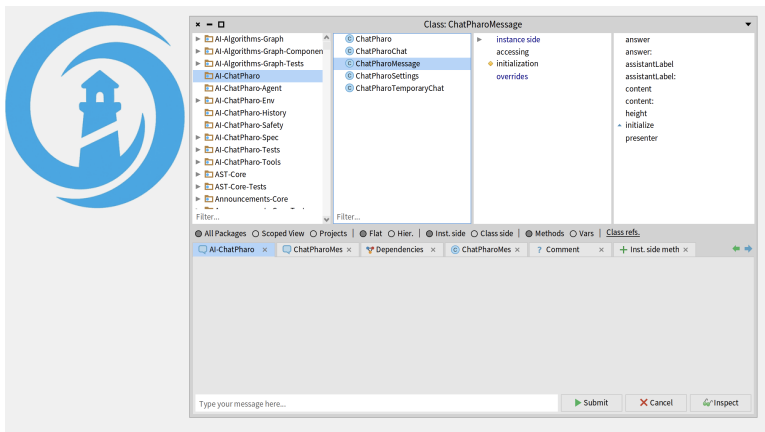
- A conversational assistant integrated into the Pharo environment that implements *tool calling* to have **live** information.
- So the best way is to have live information from Pharo **itself**
- Provide default tools to have more information from Pharo.



ChatPharo UI - Temporary ChatPharo



ChatPharo UI - ChatPharo inside the System Browser



Successful Use case

The screenshot displays the Roassal-Chart package interface. On the left, a package browser shows the hierarchy of classes, with 'Roassal-Chart' selected. The main area is a 'Playground' window containing a code editor with the following code:

```
1 | x y p c |
2 x := 1 to: 100.
3 y := x collect: [ :i | i squared ].
4
5 c := RSCompositeChart new.
6 p := RSLinePlot new.
7 p x: x y: y.
8 c add: p.
9 c title: 'Simple Line Chart'.
10 c xlabel: 'X Values'.
11 c ylabel: 'Y Values'.
12
13 c open.
```

Below the code editor, a 'Roassal' window displays a 'Simple Line Chart'. The chart has a title 'Simple Line Chart', an x-axis labeled 'X Values' ranging from 0.0 to 100.0, and a y-axis labeled 'Y Values' ranging from 0.0 to 10000.0. The chart shows a blue line representing the squares of integers from 1 to 100.

On the left side of the interface, there is a 'User' section with the text 'how to draw a chart inside Roassal ?' and an 'Assistant' section with the text 'To draw a chart in Roassal, you can use the `RSCompositeChart` class to combine different plots and decorations. Here's a general outline of how to do it:'. Below the assistant text is a text input field and 'Submit' and 'Cancel' buttons.



Key meta question

- Okay, we provide information to the LLM, and we receive information from the LLM.
- What kind of information should we give to the LLM model to have a **valuable** response?



exempli gratia

- Can we generate better class comments using the user classes, as Kasper is experimenting?
- Can we give Spec examples and get generated Spec code from a textual description?



Watch out!

- If you give too much information, you will get less good code
- For Spec
 - a tool for layout
 - a tool for the presenter

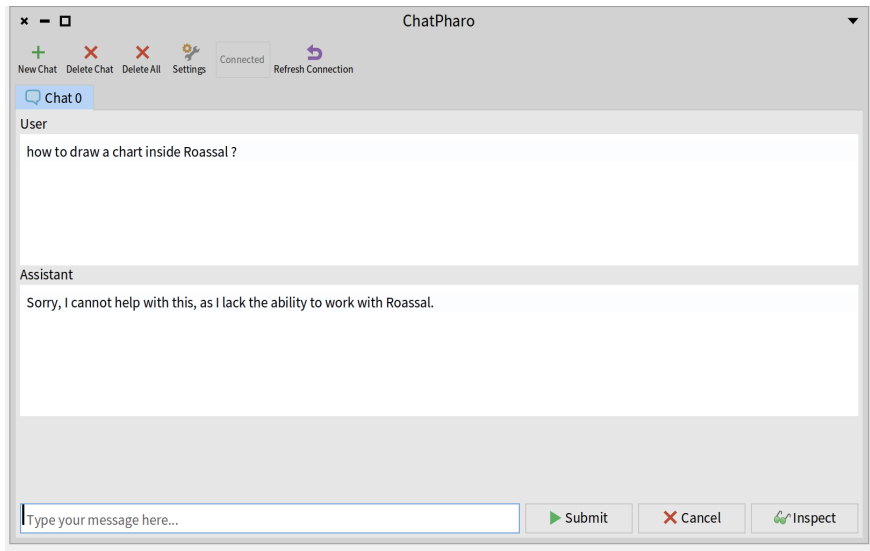


Key meta question cont...

- What type of tools should we create?
- What? Should I give you the model (the whole image, packages)?
- How? How should I give the model info (By executing tools)



Failed Use Case



Why

- A **tool call** means fresh external data were fetched.
- No tool call? The answer comes purely from the model's knowledge (hallucinations).
- The model can still generate code autonomously; it doesn't have to query external software.



Future works

- Reduce the hallucinations bug, providing an example (search engine)?
- Through a feedback loop?
- Improve ask ChatPharo



If you have ideas of scenario

- Gherkin?
- Debugger?
- Refactoring Engine?
- Moose ? Cormas ?
- Do not hesitate to contact me.



Summary

- ChatPharo is a conversational assistant integrated within the Pharo environment.
- Fetches live, real-time information from your Pharo image using Tool Calling.
- Success depends on providing the right amount and type of information to the LLM to ensure valuable, accurate responses.



Stay in touch



Try this tool on:

Github:

[github/omarabelkader/ChatPharo](https://github.com/omarabelkader/ChatPharo)

Maintainer: omar.abedelkader@inria.fr

News: chatpharo@inria.fr

Special thanks: Benoît Verhaeghe, Kasper
Østerbye,
Valentin Bourcier, Kris Gybels,
Balša Sarenac, Srajal Dwivedi

