OPACA Framework and Tools

Tutorial and Call for Feedback

Introduction

The **OPACA** Framework, developed in the course of the **Go-KI** project, aims at combining aspects of Multi-Agent Systems with Microservices and Container Technologies. You can read more about it in our paper (https://ieeexplore.ieee.org/document/10398198) and find the sources on GitHub. Those also contain some documentation for getting started.

- OPACA Framework: https://github.com/GT-ARC/opaca-core
- OPACA LLM UI: https://github.com/GT-ARC/opaca-llm-ui
- OPACA BPMN Editor: https://github.com/GT-ARC/opaca-bpmn

We held a workshop at ZEKI, introducing OPACA and its two major tools – a **BPMN Editor** and **LLM UI** – to the public and gathering some feedback. We want to invite you to also try out our tools and give us your honest feedback and suggestions for improvements! The Slides from the Workshop can be found here (useful background info, but not necessary).

You can find a permanent deployment of OPACA and its tools at http://10.42.6.107. The rest of this document explains a few simple tasks that you can try using the two above mentioned tools. Of course, you can also just play around with the tools and follow your own ideas and inspirations. Please note that some of the agents interface with real services that can e.g. open shelves in the ZEKI kitchen or write emails (from a Demo-Account in the DAI-LDAP). Please use those services responsively.

Feedback

Once you have completed the tasks on the next pages, please also participate in our Online-Survey (Google Forms, no signup necessary) in order to help us improve our tools. Your feedback is greatly appreciated!

https://forms.gle/cnFMnwDowDzoNC218

Hands-On Session: OPACA LLM

Go to **http://10.42.6.107** and open the **OPACA LLM** link.

Task 1: Connection

Connect to the OPACA platform at the above URL using the “Connect” button in the top bar. Once you are connected, it will give a rough outline of its capabilities.

Note: Some of the services in the deployment trigger real-world effects, like operating the shelves in the kitchen, or sending emails from a demo-account. Use those with care!

Once connected, you should make yourself familiar with the User-Interface. On the top-bar, you can select the current method that will be used to answer your requests. You can switch between methods even during a conversation. In the sidebar, you have the following views:

- **Information:** An LLM-generated outline of available services.
- **Prompt Library**: A collection of helpful use-cases, although some of them will not be applicable for the demo deployment.
- **Agents and Actions**: Will give you an overview of all the agents and their actions, running on the connected OPACA platform.
- **Configuration**: Here you can change some settings for the currently selected method. Some methods have more settings than others.
- **Logging**: Here you will find a detailed list of all LLM messages, debug output, including the tool calls, intermediate results, and the final message.
- **Help/FAQ**: A short description of the OPACA LLM and links with more information

Task 2: Getting Familiar with the Deployment

Prompt the application with the following request (or similar):

> *How can you assist me?*

The LLM should give a short overview of the available OPACA agents and what they can do. Since the answer is generated by the LLM, the exact contents can differ. Further, you can check in the Agents tab on the left to get more detailed information about a specific service.

Task 3: Prompting

Following are a list of simple prompts you can ask. You are of course free to test your own request or modify the requests, to see if they will change the behavior of our application.

> *What sensors are available?*
> *What is the weather forecast for Berlin for the next 3 days?*
> *What is the current temperature in the experience-hub?*
> *Create a line plot showing a sinus curve.*

Note: You can reset your current message history by clicking on the red “reset” button just next to the “Send” button next to the user input box. This will let the model forget all previous messages.

You can also test more complex requests, which would require multiple tool calls to be made. First, there are tool calls which could be executed in parallel:

> *Get me the current temperature, co2 value and noise for the experience hub.*
>
> *What are the stock prices of Microsoft, Nvidia and Amazon?*
>
> *Fetch and summarize my latest emails.*

Then there are requests, which would require sequential calls, since necessary information for one call needs to be acquired by another call first. This could include:

> *Check if any desk is available in the Robot Space and book the first one you find.*

Now you can think of your own requests.

Task 4: Testing Different Configurations

Try to make requests with different settings and use different strategies. One important setting is the model temperature. This can be a value between 0.0 - 2.0 and is used to determine the “creativity” of the model output. 0.0 indicates a very low creativity.

You could also use a strong model, in particular “gpt-4o”, to see if any failed requests are now being fulfilled properly.

Task 5: Debugging

Use the debug tab on the left or the debug button on each response to find out about the detailed process on how your response was generated, for instance:

- If your expected tools were actually called
- Whether the tools included proper parameter values
- If any additional/unnecessary tools were called
- What LLM component was responsible for an incorrect response

Hands-On-Session: OPACA BPMN Editor

Go to **http://10.42.6.107** and open the **BPMN Editor** link.

Task 1: Basic Modelling

Open the BPMN Editor and create a new BPMN diagram. Try to model the following process for a simple desk-booking system:

> *Ask the user in which room a desk should be booked. Then book one of the free desks in that room, if any. Finally, ask the user whether to book another desk.*

Hint: As a first step, you should use the Services widget in the top-right of the editor pane to connect to the OPACA-Platform (IP from above, with Port ":8000", no auth) to see what services are available. You can remove any irrelevant services from the list using the "trash" icon. Use those as a reference, but for now focus only on the BPMN nodes and edges.

There is no right or wrong, your process can be as simple or as complex as you like, but please stick to this general use case for better comparability of results.

Task 2: Making it Executable

Next, configure the User- and Service Tasks in your process. Add conditions (on Sequence Flows following a Gateway), assignments and temporary variables as needed. Don't forget to also mark the process itself as "executable". (You can find those elements, as well as the controls to create or modify them, in the Properties panel on the right side once you select an element that can contain them.)

Hint: Service parameters and results can be used just like local variables and remain in scope even after the Task that used that service, i.e. defining local variables to hold those values is not always necessary.

Hint: The language used in the assignments, conditions, etc. is a (reduced/safe) subset of JavaScript. Make sure to put any string parameters in quotes, including in User Tasks.

Task 3: Execution and Debugging

Now go to the Interpreter / Token Simulation view by clicking the icon on the top-left (in this mode, the process is read-only; click the icon again to go back to edit mode) and click the "Play" button on the Start Event to run the process. You can use the "Pause" button or spacebar to pause and resume the interpretation.

Important: Please note that in the current build, the Interpreter will show a scary Error screen on some exceptions. We noticed this too late and could not fix it for the workshop. If this happens to you, you can just close the error message using the "x" in the top-right corner of the editor and resume editing. Do NOT click "back", close or reload the tab, or your editing progress will be lost!

There will probably be some mistakes at first. Use the “Simulation Log” to view the state of all variables and try to fix the process until it actually does what you intended.

Task 4: LLM Editing Assistance

Use the “Download BPMN diagram” button on the bottom left if you want to save your BPMN diagram. (We would in fact be happy if you could send it to us by e-mail to *info@go-ki.org*, but that’s entirely up to you.) Create a new tab or reload the page to get back to the start. Use the text field to describe the same process you just modelled (in how much or how little detail as you seem appropriate) and click the “Generate BPMN” button to ask an LLM to generate the BPMN for you (this may take a few seconds).

Hint: There is also an “experimental” mode that will also try to generate the additional variables, assignments, etc. This works to some degree, but at the moment can’t reliably create Service invocations yet (in fact, at this point the LLM does not even know which services are available, so it could only “guess” useful services). You can also try this if you are interested in the results, but generation will take longer.

Inspect the generated process. Does it match your expectations? In what way does it differ from your own process? What parts are missing and have to be added (or fixed) by hand? Overall, how useful is the feature? How useful would you expect this feature to be if it was working better?