2nd International Conference on Foundation and Large Language Models (FLLM2024)

Prompt Engineering as Code (PEaC)

An approach for building modular, reusable, and portable prompts

gaetano.perrone@unina.it
spromano@unina.it

Outline

Large Language Models

A revolution for multiple fields:

- Education [1]
- Software Development [2]
- Writing and Content Creation [3]

Users can now interact with AI agents trained on vast amounts of data in **natural language.**

Human-like Responses: LLMs provide responses that **mimic human conversation**.

[1] Mesko, B. (2023). Prompt Engineering as an Important Emerging Skill for Medical Professionals: Tutorial. J Med Internet Res, 25, e50638.
[2] iray, L. Prompt Engineering with ChatGPT: A Guide for Academic Writers. Ann Biomed Eng 51, 2629–2633 (2023)
[3] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, & Fereshte Khani. (2024). Prompt Engineering a Prompt Engineer.

Prompt Engineering

A revolution for exploiting LLM capabilities

- A new discipline aimed at realizing “high-quality” prompts
- Optimize prompts to improve the quality of LLM responses
- Harness the model’s capabilities by exploring its hidden knowledge

No prompt engineering

Q: A baker bakes 20 muffins. Half of the muffins are chocolate, and half of the chocolate muffins have sprinkles. How many chocolate muffins have sprinkles?

A: The answer is 10. X

Zero-shot-CoT

Q: A baker bakes 20 muffins. Half of the muffins are chocolate, and half of the chocolate muffins have sprinkles. How many chocolate muffins have sprinkles?
A: ***Let’s think step by step.***
(Output) There are 20 muffins in total. Half of them are chocolate, so there are 10 chocolate muffins. Half of the chocolate muffins have sprinkles, so 10 / 2 = 5.

The answer is 5. ✔

Infrastructure as Code

A revolution for realizing complex IT infrastructures

- ➢ Automates the setup and maintenance of infrastructures
- ➢ Consistent, repeatable configurations across environments
- ➢ Simplifies version control and configuration for infrastructures

ANSIBLE

VAGRANT

HashiCorp Terraform

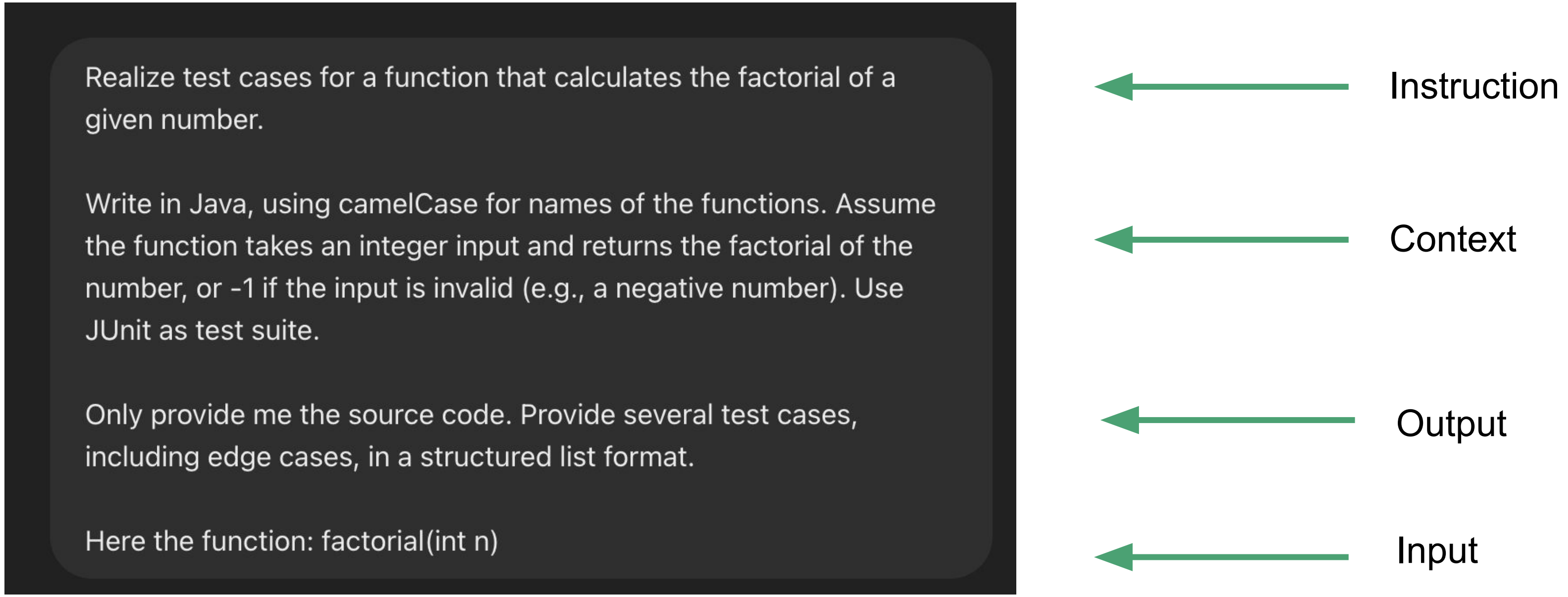
Prompt Engineering + **Infrastructure as Code**

Is it possible to realize prompt engineering by leveraging IaC techniques?

=

PEaC: Prompt Engineering as Code

Prompt structure



Realize test cases for a function that calculates the factorial of a given number.

Write in Java, using camelCase for names of the functions. Assume the function takes an integer input and returns the factorial of the number, or -1 if the input is invalid (e.g., a negative number). Use JUnit as test suite.

Only provide me the source code. Provide several test cases, including edge cases, in a structured list format.

Here the function: factorial(int n)

Giray, L. Prompt Engineering with ChatGPT: A Guide for Academic Writers. Ann Biomed Eng 51, 2629–2633 (2023). https://doi.org/10.1007/s10439-023-03272-4

Prompt Engineering as Code (PEaC)

Collect prompt sections in a modular and reusable approach
Follow the best practices of the Infrastructure as Code paradigm such as:

- Modular organization
- Import local and remote sections
- Data Serialization with YAML
- Extensibility with "extends"

YAML (Yet Another Multicolumn Layout)

YAML is a simple markup language commonly used to realize infrastructures by following an **Infrastructure as Code** approach

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: A AWS CloudFormation template to create an EC2
instance.

Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0c55b159cbfafe1f0
      KeyName: MyKeyPair
      SecurityGroups:
        - Ref: InstanceSecurityGroup
        - ...
```

Key

Value

Scalar

List

PEaC YAML format

```yaml
prompt:
  extends:
    - "<parent - yaml>"
    - ...
  context:
    base:
      - "string"
      - ...
    local?:
      name:
          preamble?: "string"
          source: "local path"
    web?:
      name:
          preamble?: "string"
          source: "remote - url"
          xpath: "string"
  output:
    base?:
      - "string"
    local?:
      name: "localname"

  query?: "string"
```

PEaC Example

Use case scenario

Develop a system that simulates IoT smart devices sending health data to centralized dashboard.

IoT smart devices

- Python program

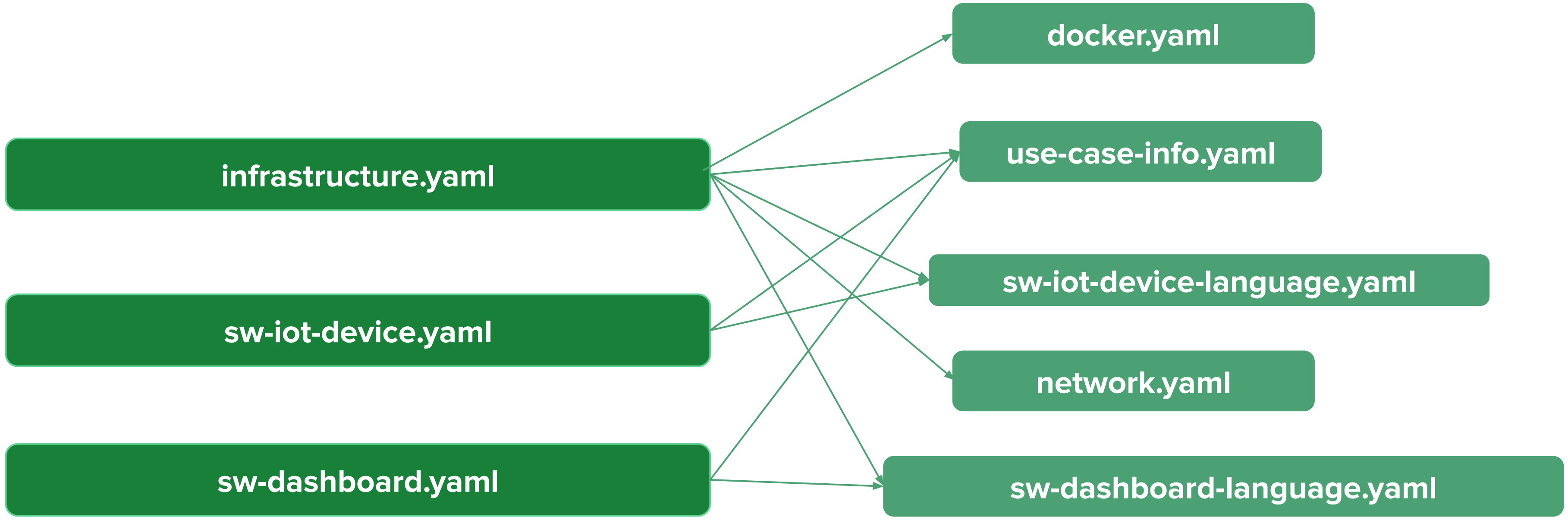
Dashboard

- Back-end
  - Java Spring Boot
  - Logs by using Log4J
  - Maven application
- Front-end
  - React

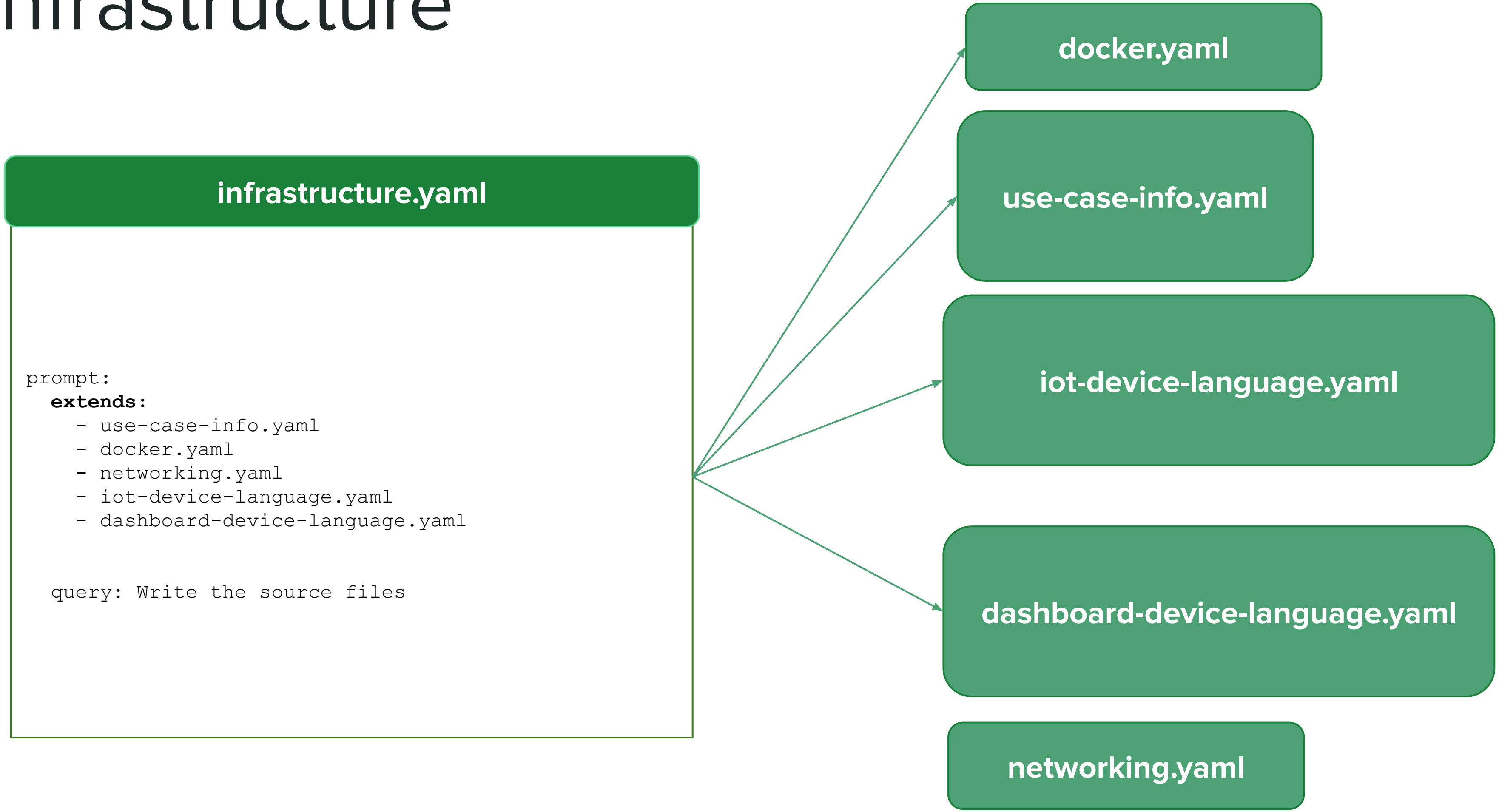
Infrastructure

- Use Docker
- Use a JSON example publicly available for simulating data
- Use EMQX as message broker

PEAC files dependencies



Infrastructure



```
prompt:
  extends:
    - use-case-info.yaml
    - docker.yaml
    - networking.yaml
    - iot-device-language.yaml
    - dashboard-device-language.yaml


  query: Write the source files
```

PEaC and prompt engineering techniques

- Few-shot prompts (Ahmed et al., 2023)	The local section allows for the collection of few-shot prompts as txt files, while the web section allows for the retrieval of the information.
- Chain-of-thought (CoT) prompting (Diao et al., 2023)	The modular approach of PEaC can be adopted to break the prompt into several modules representing the reasoning steps.
- Contextual prompting (Liu et al., 2023)	The “context” section can be used to leverage this technique
- Zero-shot prompting (Kojima et al., 2022)	Simply insert instructions in the modular YAML structure.
- Reasoning and acting (ReAc) (Yang et al., 2023)	The context can contain the following sentence: “First, analyze the problem, then describe the actions needed to solve it”.
- Dynamic prompting (Wang et al., 2022)	YAML can easily be modified at runtime to dynamically change prompts.
- Reinforcement learning prompting (Zhang et al., 2022b)	This technique cannot be implemented through PEaC as it involves **soft prompts.**

Conclusions

PEaC: an approach for making prompts reusable and modular

- Organize prompts through the YAML syntax language
- Enable prompts sharing and collaboration
- Inherit and Extend Prompts
- Support diverse data sources

Future works

- Extend the PEaC language and define a formal specification language
- Experiment the approach in real-cases
- Integration with LLM Pipelines
- Explore Multi-Modal Capabilities

Thank you for your attention

https://github.com/giper45/peac.git

PEaC design concepts

Dynamic Instruction Adaptation

- For a given instruction and input, context and output shape and refine LLM responses.

Modularity and Reusability

- Context and output are structured into modular, reusable components, enabling efficient prompt management.

PEaC Approach

- Adopts YAML-based structure for human-readable, structured prompts.
- Facilitates sharing, portability, and scalability in prompt engineering.