

Petter Krus

Fluid and Mechatronic Systems, Linköping University

# Augmenting Aerospace System Design Using Large Language Models

# INTRODUCTION

- The democratization of AI since ChatGPT's release in 2022 opens the field for research into application of LLM in engineering design.
- This paper explores how Large Language Models (LLMs) can support and automate aerospace system, demonstrated through hybrid-electric.

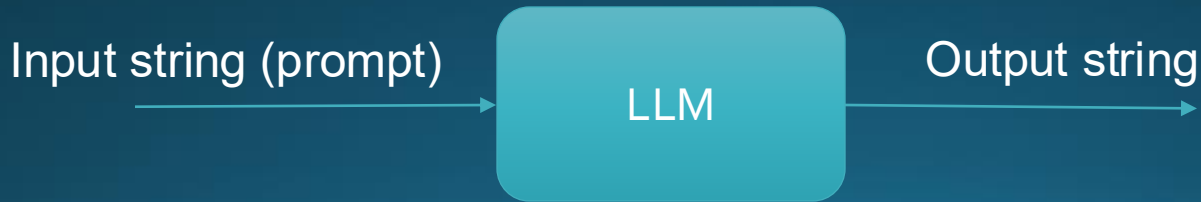
# Methodology Overview

Structured use of LLMs in aerospace system design involves:

- Prompt engineering
  - Context Engineering
  - Micro-templates
- Python system configurator generation
- API integration for interactive design.

# LARGE LANGUAGE MODELS AS BLACK BOXES

- The application of LLMs has not so much to do with their internal workings.
- AI-scientists does not have much of an edge in the applications of LLM over the domain expert.
- At the core the LLM can be seen as a black box that generates an output string from an input string.



# Generation of Architectures



- Prompt: "Generate a hybrid electric aircraft system with a fuel cell. Show the result as code for a UML-component diagram in PlantUML"
- The diagram is very basic and not entirely correct..

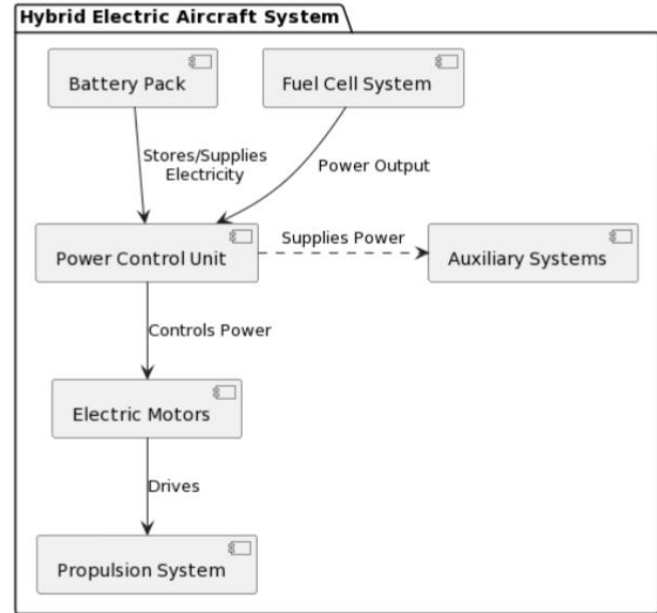
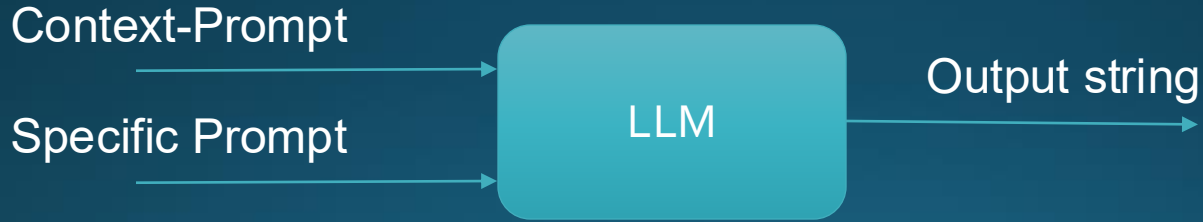


Figure 1 – Hybrid propulsion system with fuel cell. This diagram is generated with a single prompt in ChatGPT-4

# LARGE LANGUAGE MODELS AS BLACK BOXES

- The prompt can be structured to have a general pre-defined context prompt the can be reused, and one specific part.
- The context-prompt can also contain documents and this can also be used to create a Custom GPT that can be saved for later use.



# CONTEXT PROMPT WITH MICRO-TEMPLATES

## # Aircraft Propulsion System Configuration

This section details the standard configuration for connecting components in an aircraft propulsion system, focusing on hybrid systems that incorporate gas turbines, fuel cells, batteries, and electric machines.

## ## Component Descriptions

- **Gas Turbine**: Converts chemical power ( $\dot{P}_M$ ) from a fuel source (Fuel Tank or Hydrogen Storage) into rotational power (MR). The MR port can only be connected to one component.
- **Fuel Cell**: Converts chemical power ( $\dot{P}_M$ ) from Hydrogen Storage into electric power (EL).
- **Propeller**: Converts rotational power into linear power, i.e., thrust.
- **Gearbox**: Connects one or more rotational power sources (MR) to a single output rotational power (MR).
- **Electric Motor/Generator**: Converts electric power (EL) from a Battery into mechanical rotational power (MR) or vice versa. The MR port can only be connected to one component.
- **Battery Pack**: A storage unit that provides electric power (EL). It can have multiple connections.

## ## Component Connection Principles

- **Port Matching**: Ensure that ports on each component are connected to ports of a similar type on other components. For example, electric power ports (EL) should connect to other EL ports, while mass flow ports ( $\dot{P}_M$ ) should connect to other  $\dot{P}_M$  ports. Only include components that are used.
- **Dashed Connections**: When using dashed lines ("-[dashed]-"), do not

## ## Micro Templates

Here are some micro templates that show the style of the UML diagram.

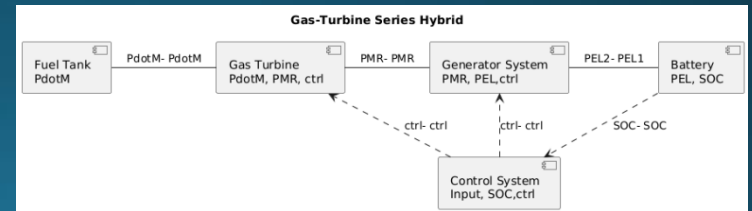
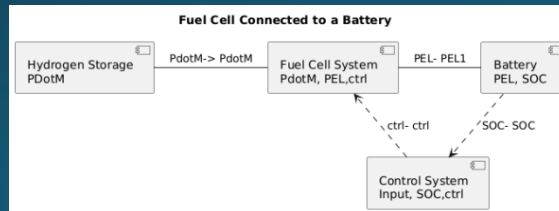
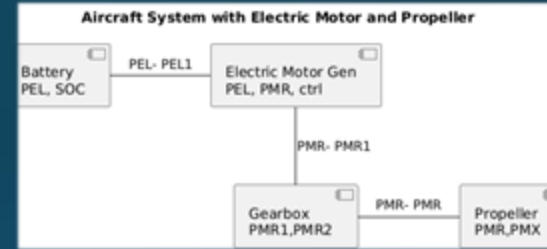
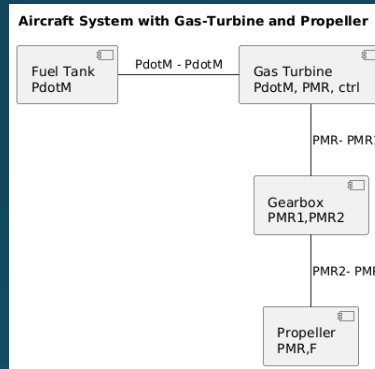
### ### Propeller Connected to an Aircraft

```
``plantuml
@startuml
' Define the components with ports
component "Aircraft Body\nPMX, Aero, states" as AircraftBody
component "Propeller\nPMR,PMX" as Propeller
' Connections
Propeller -down- AircraftBody: PMX- PMX
' Diagram Title
title Aircraft System with Aircraft Body and Propeller
@enduml
``
```

### ### Connection Between a Battery and a Motor

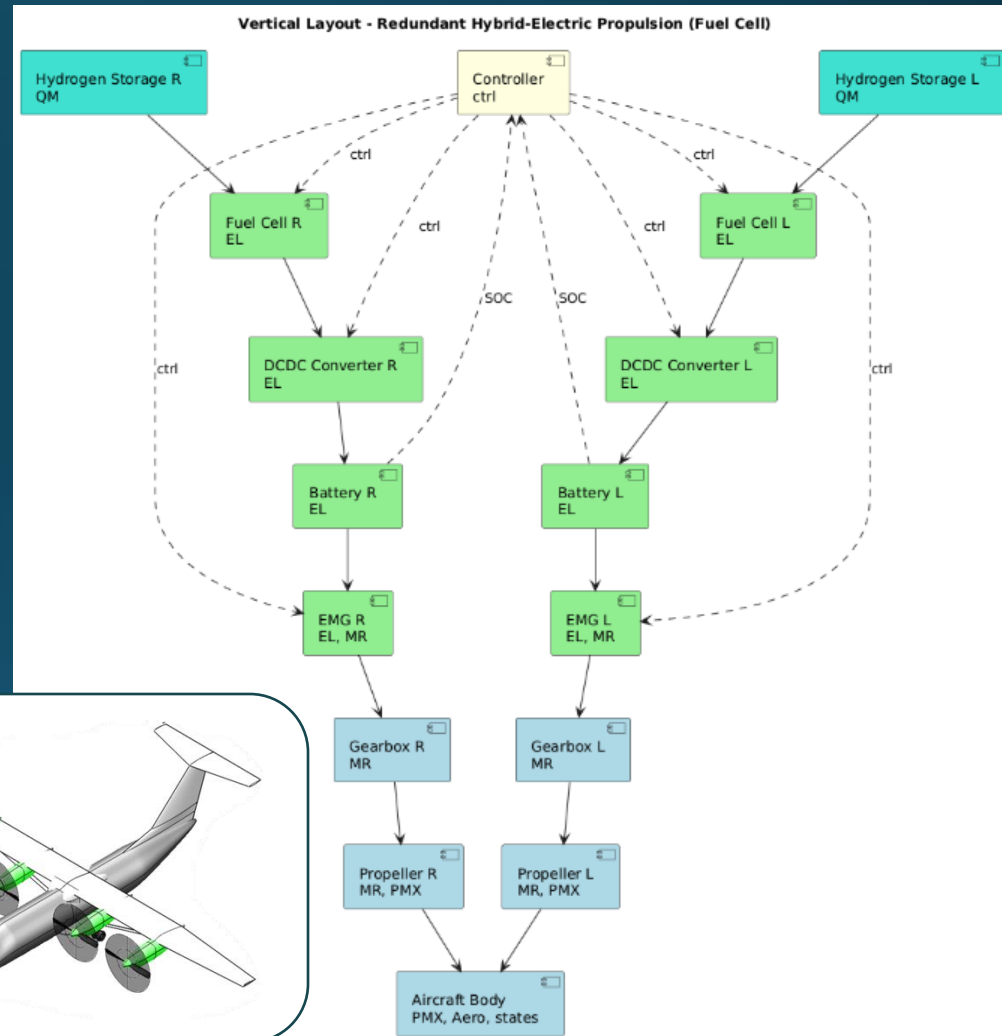
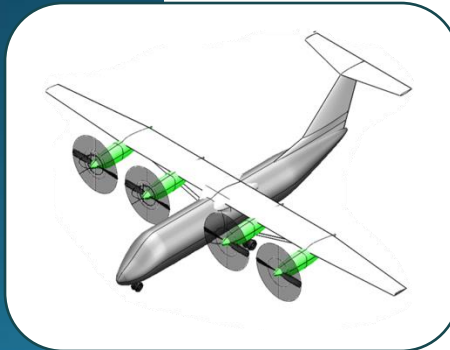
```
``plantuml
@startuml
' Define the components with ports
component "Battery\nPEL, SOC" as Battery
component "Electric Motor Gen\nPEL, PMR, ctrl" as ElMotorGen
component "Propeller\nPMR,PMX" as Propeller
component "Gearbox\nPMR1,PMR2" as Gearbox
' Connections
Battery -right- ElMotorGen: PEL- PEL1
ElMotorGen -down- Gearbox: PMR- PMR1
Gearbox -right- Propeller: PMR- PMR
' Diagram Title
title Aircraft System with Electric Motor and Propeller
@enduml
``
```

# Micro Templates in the Context-Prompt



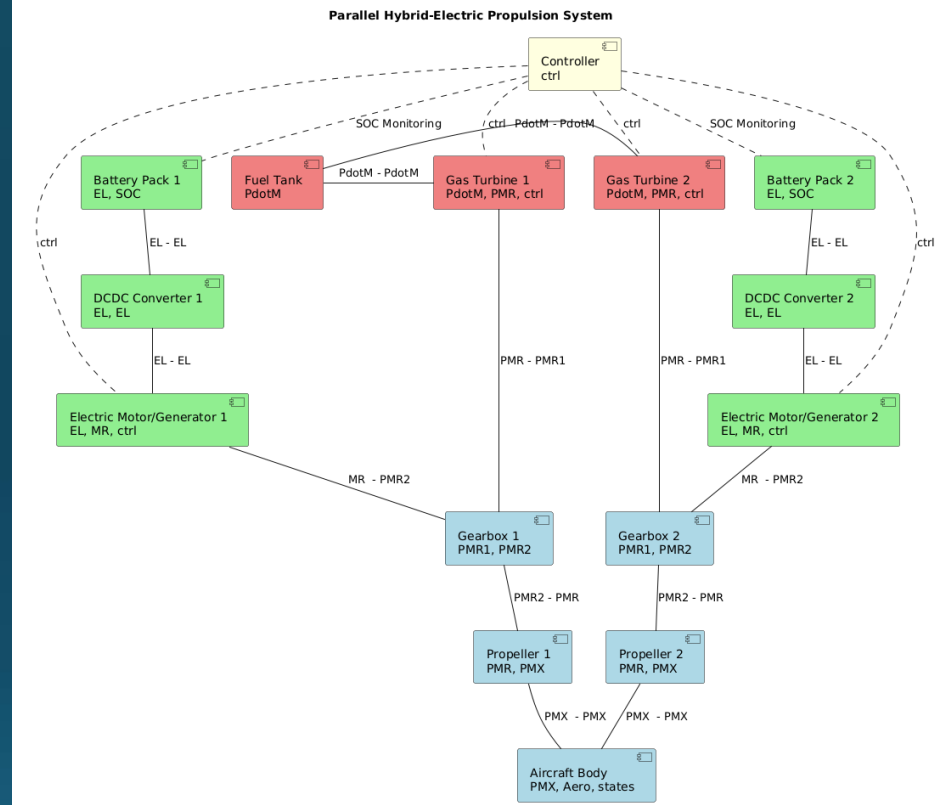
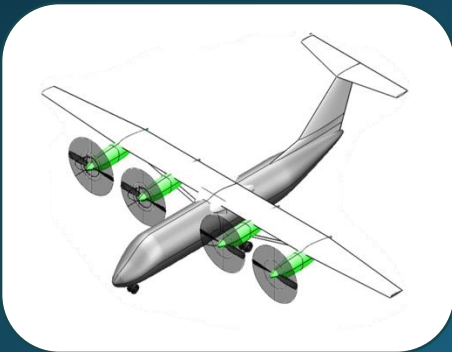
# Example: Hybrid Electric Aircraft

- Prompt: "Generate a Concept for a Hybrid Electric Aircraft with Two Propellers and Fuel Cells."
- UML-diagrams can be mapped to system libraries for **system simulation**, allowing system architectures to be manipulated, tested, and validated and optimized in simulations.



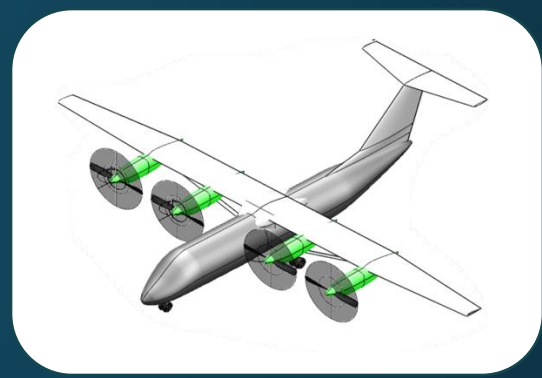
# Example: Hybrid Electric Aircraft

- Prompt: "Generate a concept for a parallel hybrid electric aircraft for 49 passengers with gas turbines, batteries and electric motors. Show the architecture as code for a component diagram in PlantUML format."

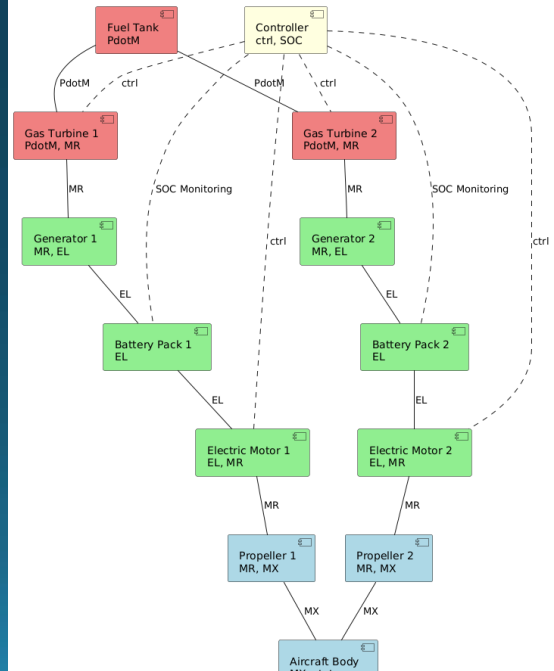


# Variation

- Series Hybrid



Two-Propeller Series-Hybrid Electric Aircraft with Gas-Turbine Generators



The diagram consists of a large, light blue oval containing a smaller, rounded rectangular box of a darker shade of blue. The text is positioned inside these shapes to illustrate the relationship between a defined design space and a fuzzy design space.

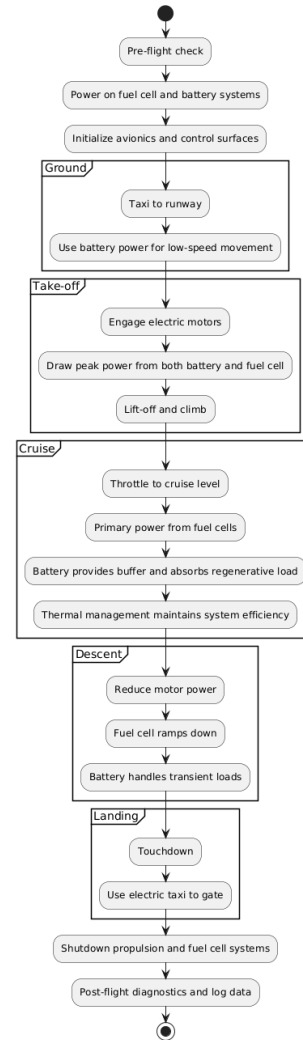
Design Space defined using LLM  
with context-prompt

The Fuzzy Design  
Space when using  
LLM

# Activity Diagrams

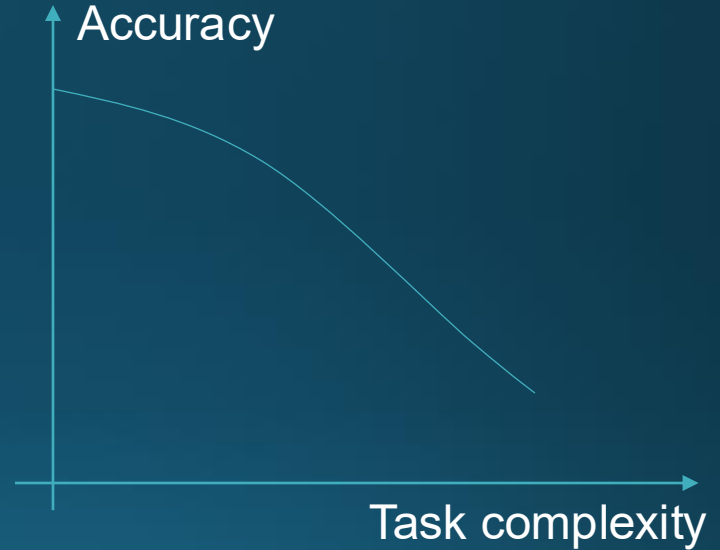
- Describing the system behaviour and function. This can be done using a UML Activity Diagram. Here, the LLM can be used to generate a UML activity diagram of a flight referring to the previous dialog.
- **Prompt.** "Make an UML activity diagram for PlantUML of a flight of the regional fuel cell electric aircraft in the example

Flight Profile of Fuel Cell Hybrid-Electric Aircraft



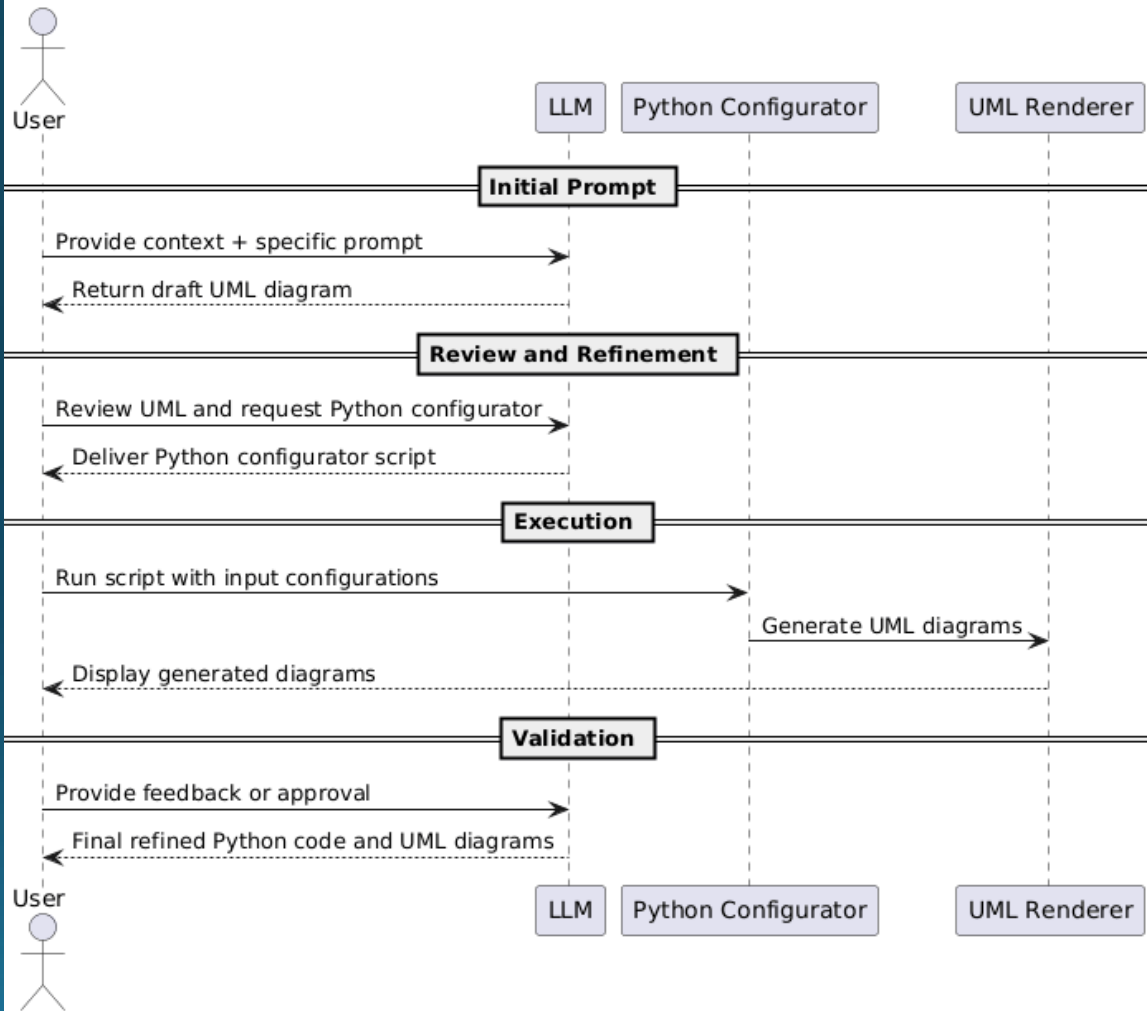
# Scaling Property of LLM

- The scaling property of LLMs have to be considered
- Just because a simple system can be generated, there is no guarantee a more complex can be generated reliably.



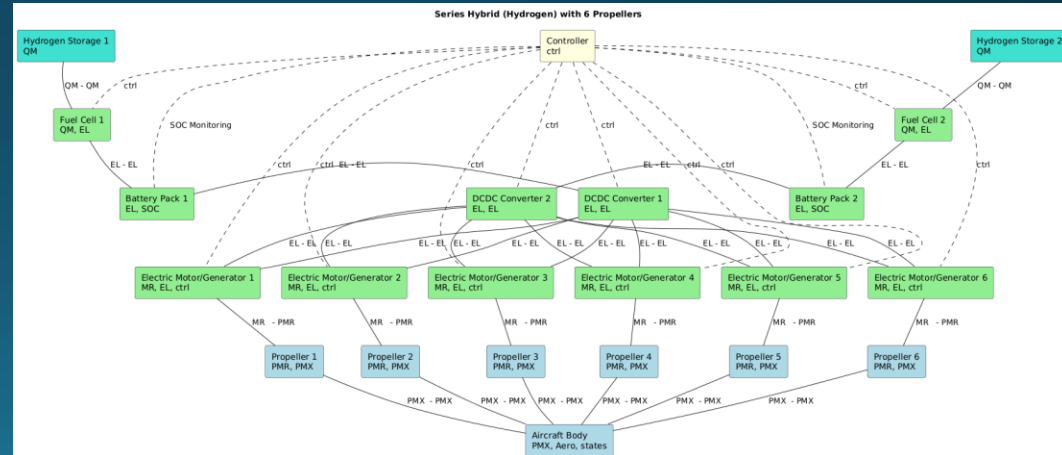
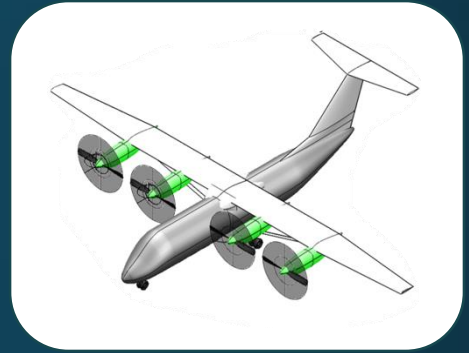
# Python Configurator Generation

Python Configurator Generation - Interaction Diagram



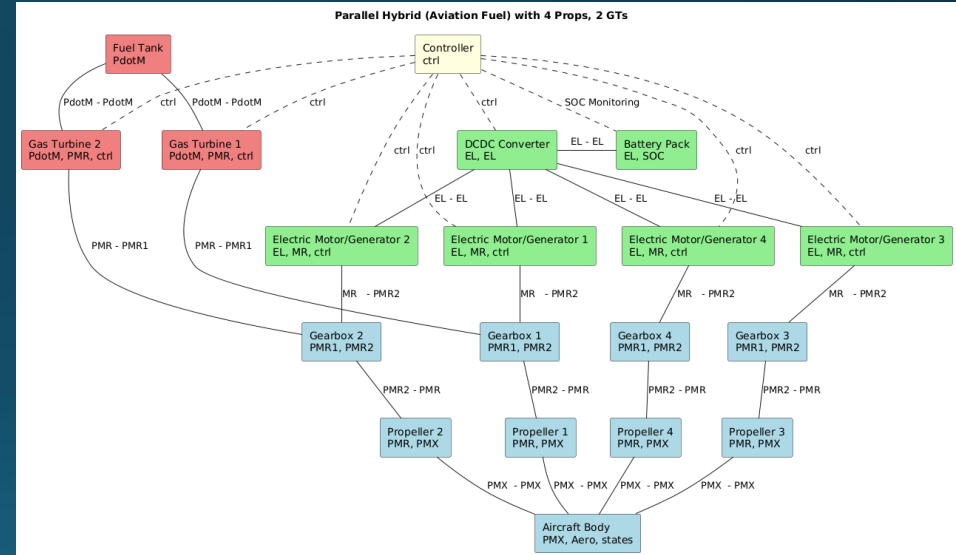
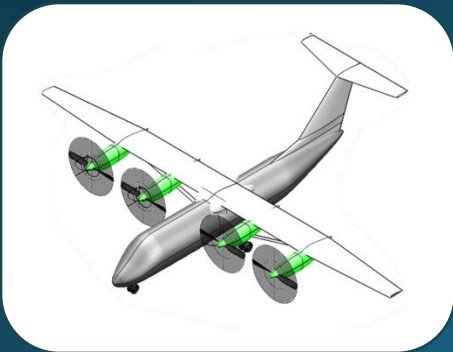
# Fuel Cell Aircraft

```
uml_series = generate_plantuml(  
    num_props=6,  
    hybrid_type='series',  
    primary_energy='hydrogen')
```



# Parallel Hybrid Aircraft

```
uml_parallel = generate_plantuml(  
    num_props=4,  
    hybrid_type='parallel',  
    primary_energy='aviation_fuel')
```



Design Space when using LLM  
with context-prompt

Design Space  
Defined by Coded  
Configuration Rules

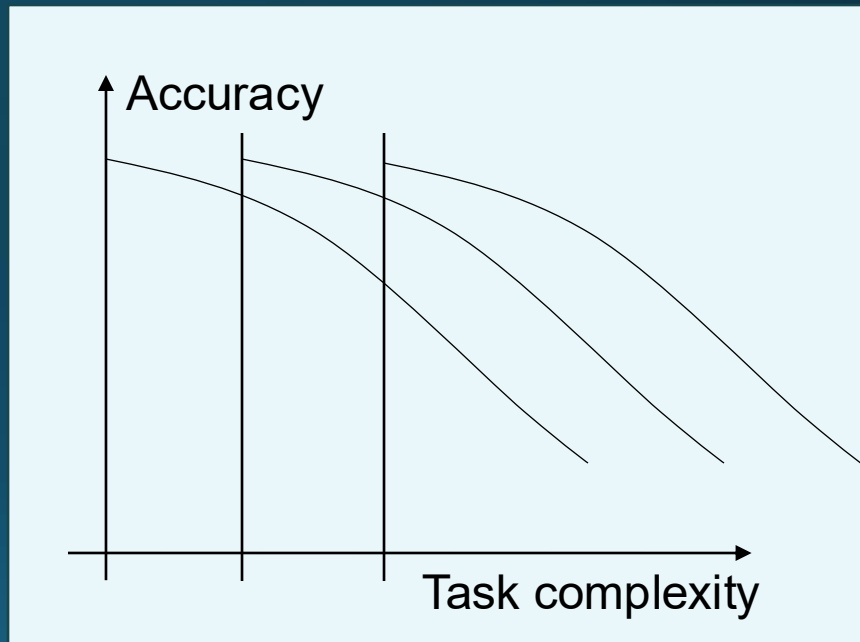
Design Space when  
using LLM

# DISCUSSION

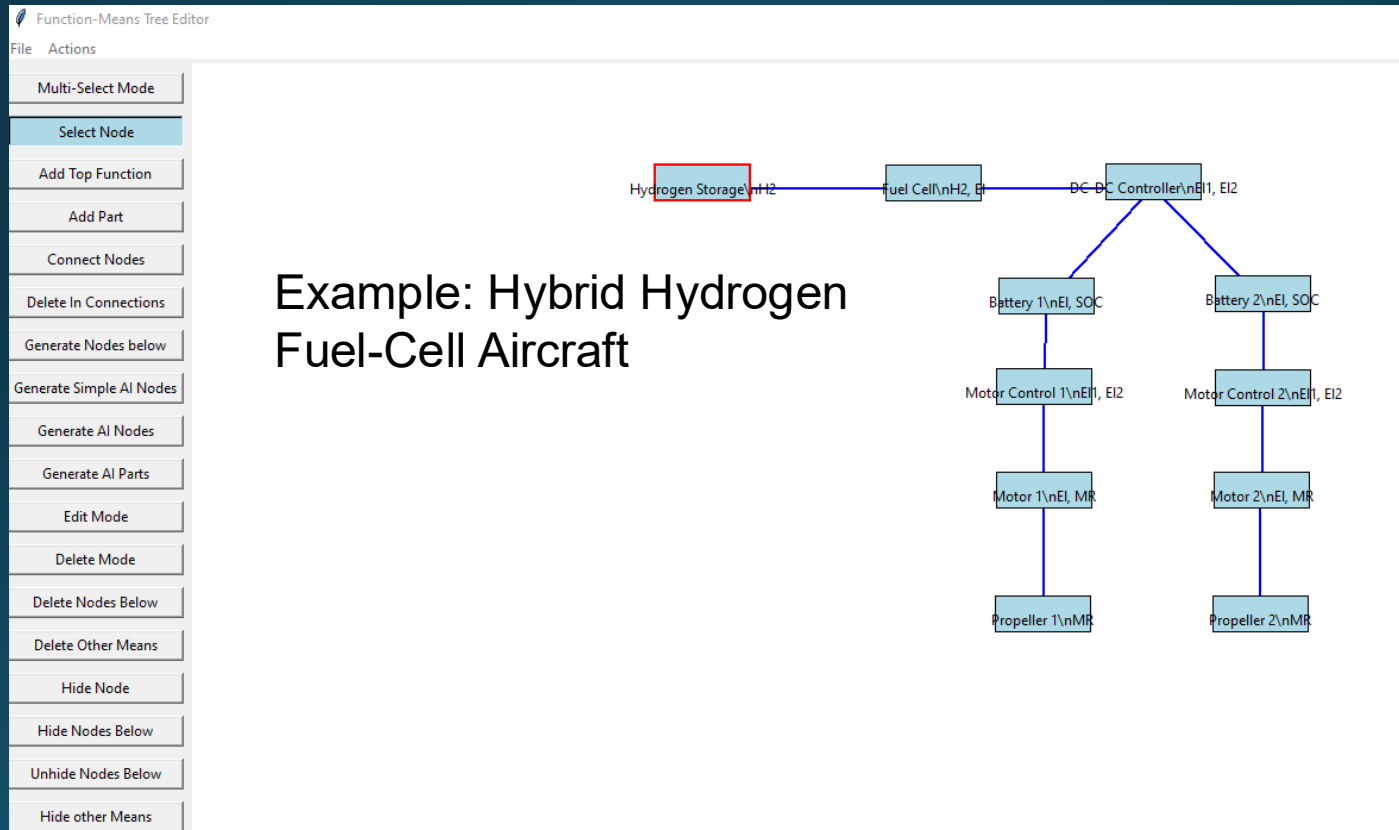
- **Application in UML Component Diagrams:**
  - The model produced UML component diagrams in **PlantUML** due to its minimalist format and ease of use, facilitating a clear representation of system architecture.
  - However, other diagram formats can also be incorporated for more detailed or specific representations.
  - UML diagrams can be mapped to system libraries for **simulation purposes**, allowing system architectures to be manipulated, tested, and validated in simulations.

# Use API for Tools to Enable Interactive use of LLMs

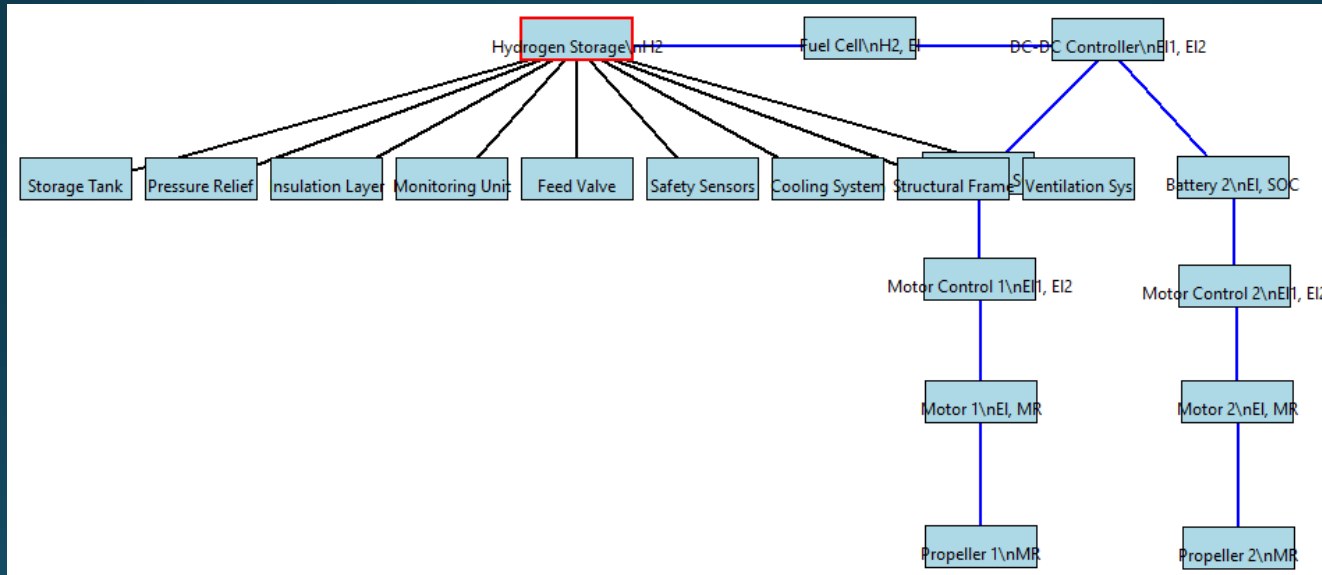
- Allow for interactive use of LLM by using LLM-API inside a tool
- Enables system decomposition



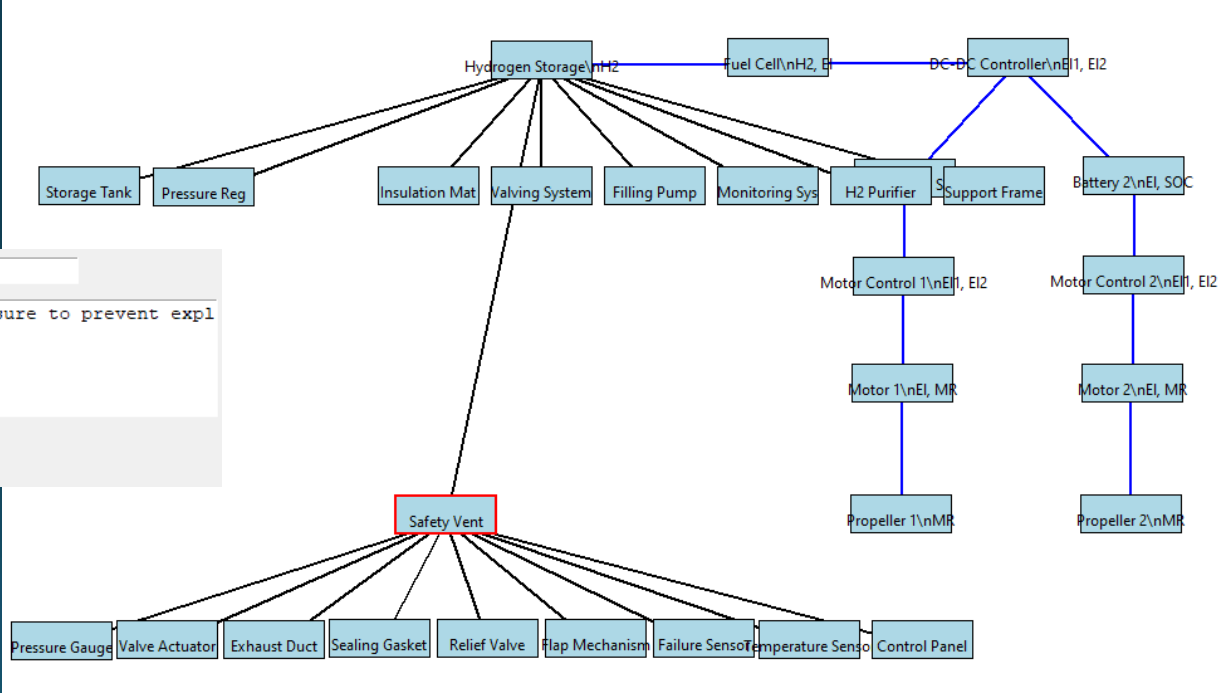
# Importing (AI-generated) System Layout to a System Editor to Generate Subsystems/Components



# AI Generate Subsystems/Components using LLM API-calls within a program



# AI Generate Subsystems/Components



```

# Gather parent context for AI prompt
def gather_parent_context(node_id):
    for edge in connection_manager.edges:
        if edge["to"] == node_id:
            parent_node_id = edge["from"]
            parent_node = self.nodes[parent_node_id]
            return gather_parent_context(parent_node_id) + [parent_node["name"]]
    return []

```

```

parent_context = " > ".join(gather_parent_context(selected_node_id))
comment = selected_node.get("comment", "").strip()
node_info = f'"{selected_node["name"]}'"
if comment:
    node_info += f" (Comment: {comment})"

```

```

# Embedded instructions for generating Part nodes
instructions = """

```

```

You are working on generating parts for a system design.
Use the following rules for generating Part nodes:
1. For a 'Part', generate up to 9 sub-parts that can integrate into the parent node.
2. Each sub-part should represent a distinct functional or structural component of the parent.
3. Each sub-part should have a concise name (≤15 characters) and a short description of its role.
4. Ensure sub-parts are diverse and cover various aspects such as structural, mechanical, electrical, or signal-related element:

```

```

Format:
- Name: [Suggestion Name]
- Comment: [Suggestion Role/Description]
"""

```

```

prompt = f"""
{instructions}
Context: The current node is part of a system hierarchy '{parent_context}'
to design the part '{selected_node['name']}''.
The selected node is {node_info}.
Suggest exactly 9 Part sub-nodes for this '{selected_node['name']}''.
Each Part node should have a name (≤15 characters) and a description (≤50 characters).
"""

```

```

import openai
openai.api_key = self.api_key

```

```

print("Prompt:", prompt)

```

```

try:
    response = openai.ChatCompletion.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": instructions},
            {"role": "user", "content": prompt}
        ],
        max_tokens=250
    )
    raw_content = response.choices[0].message.content

```

Prompt:

You are working on generating parts for a system design.

Use the following rules for generating Part nodes:

1. For a 'Part', generate up to 9 sub-parts that can integrate into the parent node.
2. Each sub-part should represent a distinct functional or structural component of the parent.
3. Each sub-part should have a concise name (≤15 characters) and a short description of its role.
4. Ensure sub-parts are diverse and cover various aspects such as structural, mechanical, electrical, or signal-related element:

Format:

- Name: [Suggestion Name]
- Comment: [Suggestion Role/Description]

Context: The current node is part of a system hierarchy 'Hydrogen Storage\NH3 Storage' to design the part 'Safety Vent'.

The selected node is 'Safety Vent' (Comment: Releases excess pressure to prevent overpressure). Suggest exactly 9 Part sub-nodes for this 'Safety Vent'.

Each Part node should have a name (≤15 characters) and a description (≤50 characters).

- ```
Raw AI response: - Name: Pressure Relief
                  - Comment: Opens to release excess gas pressure.

- Name: Check Valve
  - Comment: Prevents backflow of hydrogen gas.

- Name: Safety Seal
  - Comment: Ensures airtight closure to prevent leaks.

- Name: Flow Meter
  - Comment: Measures the rate of hydrogen flow.

- Name: Vent Cap
  - Comment: Covers the vent when not in use.

- Name: Temperature Sensor
  - Comment: Monitors vent temperature for safety.

- Name: Pressure Gauge
  - Comment: Displays current internal pressure levels.

- Name: Vibration Damp
  - Comment: Reduces vibrations during vent operation.

- Name: Warning Light
  - Comment: Illuminates for safety alerts during operation.
```

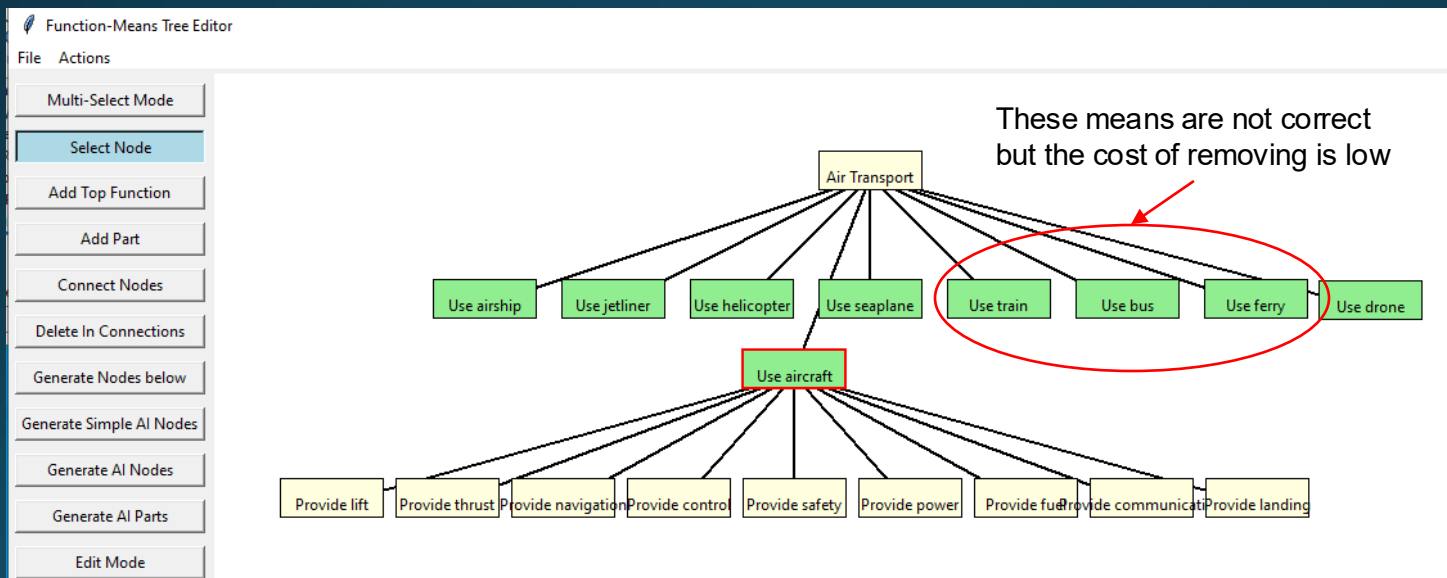
# Use API for Tools to Enable Interactive use of LLMs

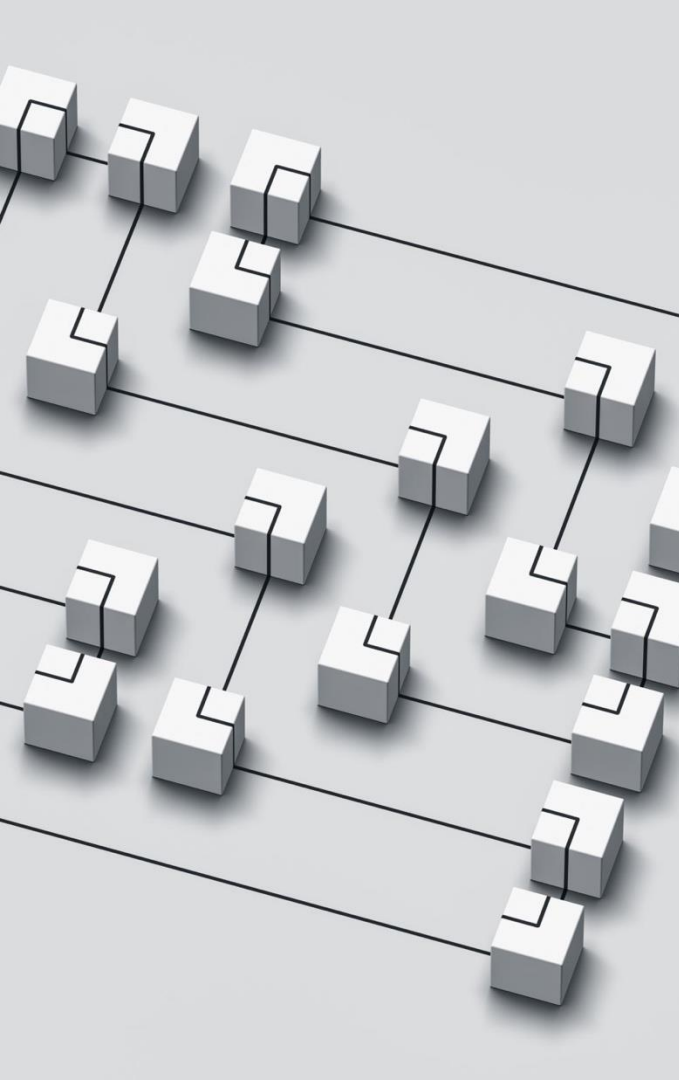
- Example 2: Function Means Tree
- LLM is used to generate suggestions for alternative solutions

The screenshot displays the 'Function-Means Tree Editor' interface. On the left is a vertical toolbar with buttons for 'Multi-Select Mode', 'Select Node', 'Add Top Function', 'Add Part', 'Connect Nodes', 'Delete In Connections', 'Generate Nodes below', 'Generate Simple AI Nodes', 'Generate AI Nodes', 'Generate AI Parts', 'Edit Mode', and 'Delete Mode'. The main workspace shows a tree diagram with a root node 'Air Transport' (highlighted with a red border) and nine child nodes: 'Use aircraft', 'Use airship', 'Use jetliner', 'Use helicopter', 'Use seaplane', 'Use train', 'Use bus', 'Use ferry', and 'Use drone'. An 'Edit Node Attributes' dialog box is open in the foreground, showing the 'Name' field with 'Air Transport' and the 'Comment' field with 'For 70 passengers'. A 'Save' button is at the bottom of the dialog.

# Use API for Tools to Enable Interactive use of LLMs

- Generating functions for the next level.





# Discussion and Conclusion

- Application of AI-generated UML Component Diagrams:
  - Diagrams can be mapped to system libraries for **system simulation**, allowing system architectures to be manipulated, tested, and validated and optimized in simulations.
- LLMs should be used when:
  - value of the correct answers > cost of the bad answers
- There is a gap between capabilities of the AI-tools and the application of of them in engineering .

# Announcements

**2025**

There is a Workshop on AI in Aeronautical Engineering at ITA, Dec 1-2

**Feb. 2026**

There is an on-line Postgraduate course "Engineering System Design Analytics and AI" starting in Feb 2026.