LLMs for Direct Interaction with SysML v2

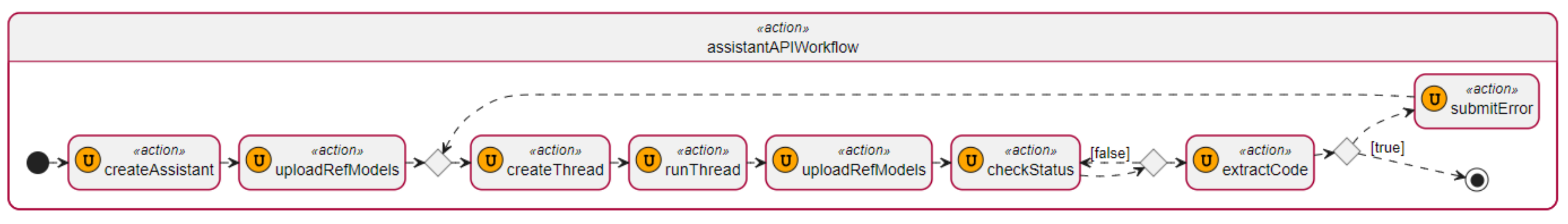
John K. DeHart
AVIAN

34th Annual INCOSE international symposium
hybrid event
Dublin, Ireland
July 2 - 6, 2024

PURPOSE / PROBLEM

Purpose: This study explores the integration of Large Language Models (LLMs) with Systems Modeling Language version 2 (SysML v2) to streamline systems engineering processes. The primary goal is to leverage LLMs to enhance the readability and usability of SysML v2, thereby reducing the technical barriers associated with traditional API interactions.

Problem: The interaction with SysML v2 typically requires navigating complex APIs, which can detract from its user-centric design. This complexity introduces a significant learning curve and potential inefficiencies in model management, hindering the overall effectiveness of systems engineering workflows.



OBJECTIVES / GOALS

Objectives:

Evaluate the feasibility of using LLMs to directly interact with SysML v2 models.

Understand the potential benefits and limitations of integrating LLMs with SysML v2.

Develop a streamlined methodology for systems engineers to manage models using natural language.

Goals:

Simplify SysML v2 model manipulation through natural language.

Enhance the accessibility and efficiency of systems engineering workflows.

Provide evidence-based insights into the integration of LLMs and SysML v2 for improved model management.

METHODS / DESCRIPTION

Methods:

Experimental Setup:

- Utilized SysML v2 for constructing text-based system models.
- Implemented OpenAI's Assistants to process natural language queries and generate Python code.
- Stored SysML v2 models in a file system and interfaced with the LLM through a scripting environment.

Workflow:

- Created an Assistant in the OpenAI Playground or via API.
- Uploaded reference SysML v2 models to the Assistant.
- Interacted with the models through natural language commands, which the Assistant processed to generate and execute code for model modifications.

Descriptions:

- **SysML v2:** Enhanced for human readability and model expressiveness, making it suitable for integration with LLMs.
- **LLMs:** Used for interpreting and generating SysML v2 constructs based on natural language inputs, facilitating a more intuitive interaction with system models.
- **Integration Approach:** Combined the structured data management capabilities of SysML v2 with the natural language processing strengths of LLMs to streamline the modeling process and reduce the need for specialized API knowledge.

DATA / RESULTS

Data:

Case Study 1: Modification of a SysML v2 model defining a physical beam's characteristics through natural language commands.

Case Study 2: Implementation of an API endpoint and testing front end for querying system models using natural language.

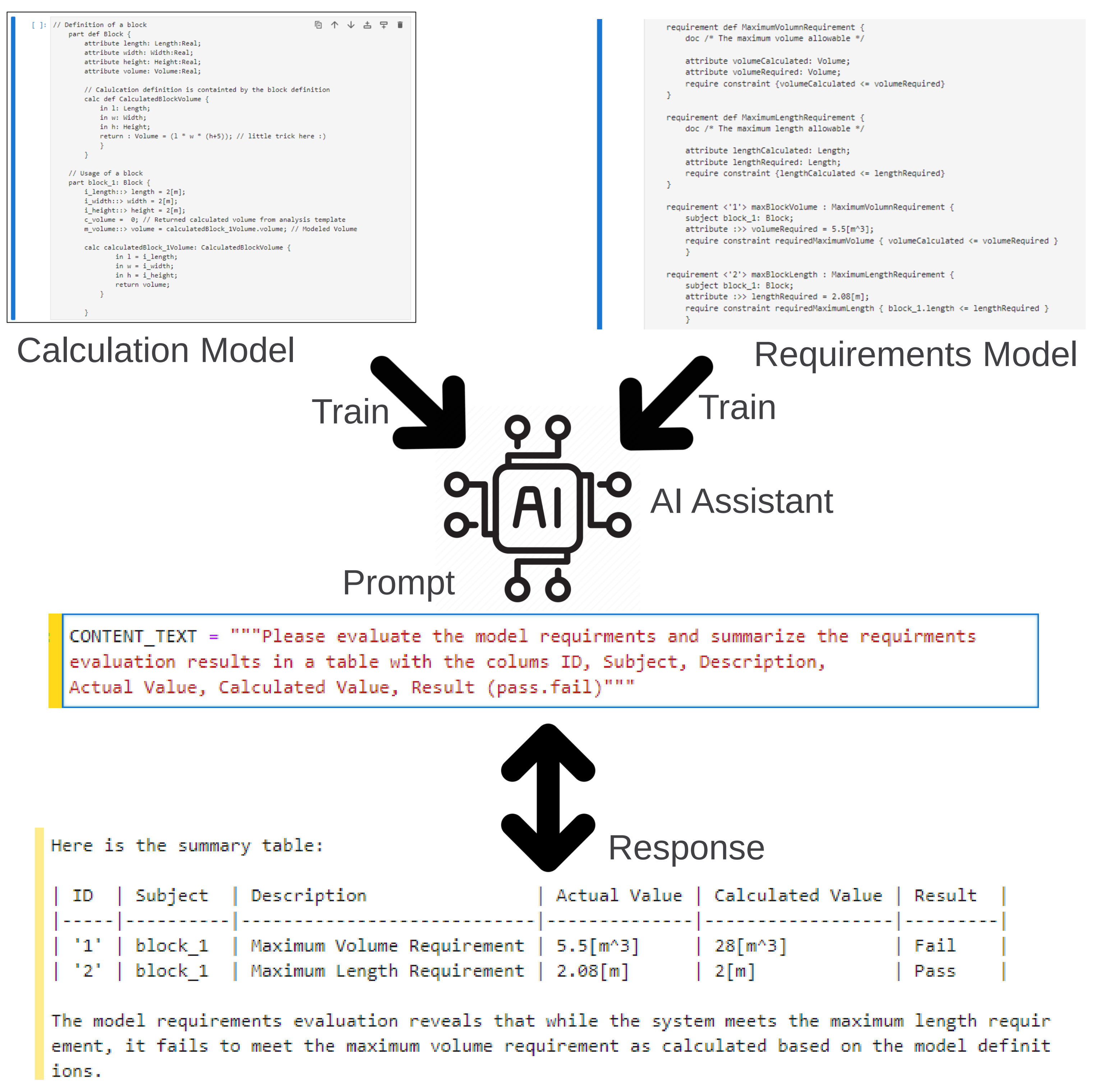
Case Study 3: Validation of a complex SysML v2 model, including non-standard calculations and requirements evaluation, using an AI assistant.

Results:

Case Study 1: Demonstrated successful modification of SysML v2 models via LLM-generated Python code.

Case Study 2: Showed effective interaction with SysML v2 models through an API, enabling complex queries and real-time model updates.

Case Study 3: Highlighted the AI assistant's ability to accurately interpret and validate complex model requirements, maintaining model integrity and enhancing usability.



```
[ ]: // Definition of a block
    part def Block {
        attribute length: Length:Real;
        attribute width: Width:Real;
        attribute height: Height:Real;
        attribute volume: Volume:Real;

        // Calulcation definition is containted by the block definition
        calc def CalculatedBlockVolume {
            in l: Length;
            in w: Width;
            in h: Height;
            return : Volume = (l * w * (h+5)); // little trick here :)
            }
        }

    // Usage of a block
    part block_1: Block {
        i_length::> length = 2[m];
        i_width::> width = 2[m];
        i_height::> height = 2[m];
        c_volume =  0; // Returned calculated volume from analysis template
        m_volume::> volume = calculatedBlock_1Volume.volume; // Modeled Volume

        calc calculatedBlock_1Volume: CalculatedBlockVolume {
                in l = i_length;
                in w = i_width;
                in h = i_height;
                return volume;
            }

        }
```

```
requirement def MaximumVolumnRequirement {
    doc /* The maximum volume allowable */

    attribute volumeCalculated: Volume;
    attribute volumeRequired: Volume;
    require constraint {volumeCalculated <= volumeRequired}
}

requirement def MaximumLengthRequirement {
    doc /* The maximum length allowable */

    attribute lengthCalculated: Length;
    attribute lengthRequired: Length;
    require constraint {lengthCalculated <= lengthRequired}
}

requirement <'1'> maxBlockVolume : MaximumVolumnRequirement {
    subject block_1: Block;
    attribute :>> volumeRequired = 5.5[m^3];
    require constraint requiredMaximumVolume { volumeCalculated <= volumeRequired }
    }

requirement <'2'> maxBlockLength : MaximumLengthRequirement {
    subject block_1: Block;
    attribute :>> lengthRequired = 2.08[m];
    require constraint requiredMaximumLength { block_1.length <= lengthRequired }
    }
```

```
CONTENT_TEXT = """Please evaluate the model requirments and summarize the requirments
evaluation results in a table with the colums ID, Subject, Description,
Actual Value, Calculated Value, Result (pass.fail)"""
```

```
Here is the summary table:

| ID  | Subject  | Description                  | Actual Value | Calculated Value | Result  |
|-----|----------|------------------------------|--------------|------------------|---------|
| '1' | block_1  | Maximum Volume Requirement   | 5.5[m^3]     | 28[m^3]          | Fail    |
| '2' | block_1  | Maximum Length Requirement   | 2.08[m]      | 2[m]             | Pass    |

The model requirements evaluation reveals that while the system meets the maximum length requir
ement, it fails to meet the maximum volume requirement as calculated based on the model definit
ions.
```

CONCLUSIONS

Conclusions: The integration of LLMs with SysML v2 offers significant improvements in usability, efficiency, and accessibility. By enabling direct, natural language interactions with system models, LLMs reduce the reliance on complex API navigation and allow systems engineers to focus on core design and analysis tasks. However, it is crucial to address potential challenges such as automation bias, error handling, and model reliability to ensure the successful adoption of this technology.

Future Work: Further research is needed to empirically validate these approaches and explore deeper interoperability with existing systems engineering ecosystems. Additionally, developing domain-specific LLM training methodologies and ethical guidelines will be essential for advancing this integration.

CONTACTS / REFERENCES

John K. DeHart
Sr. Principle System Engineer / Sr. Principle Digital Engineer
jdehart@avian.com