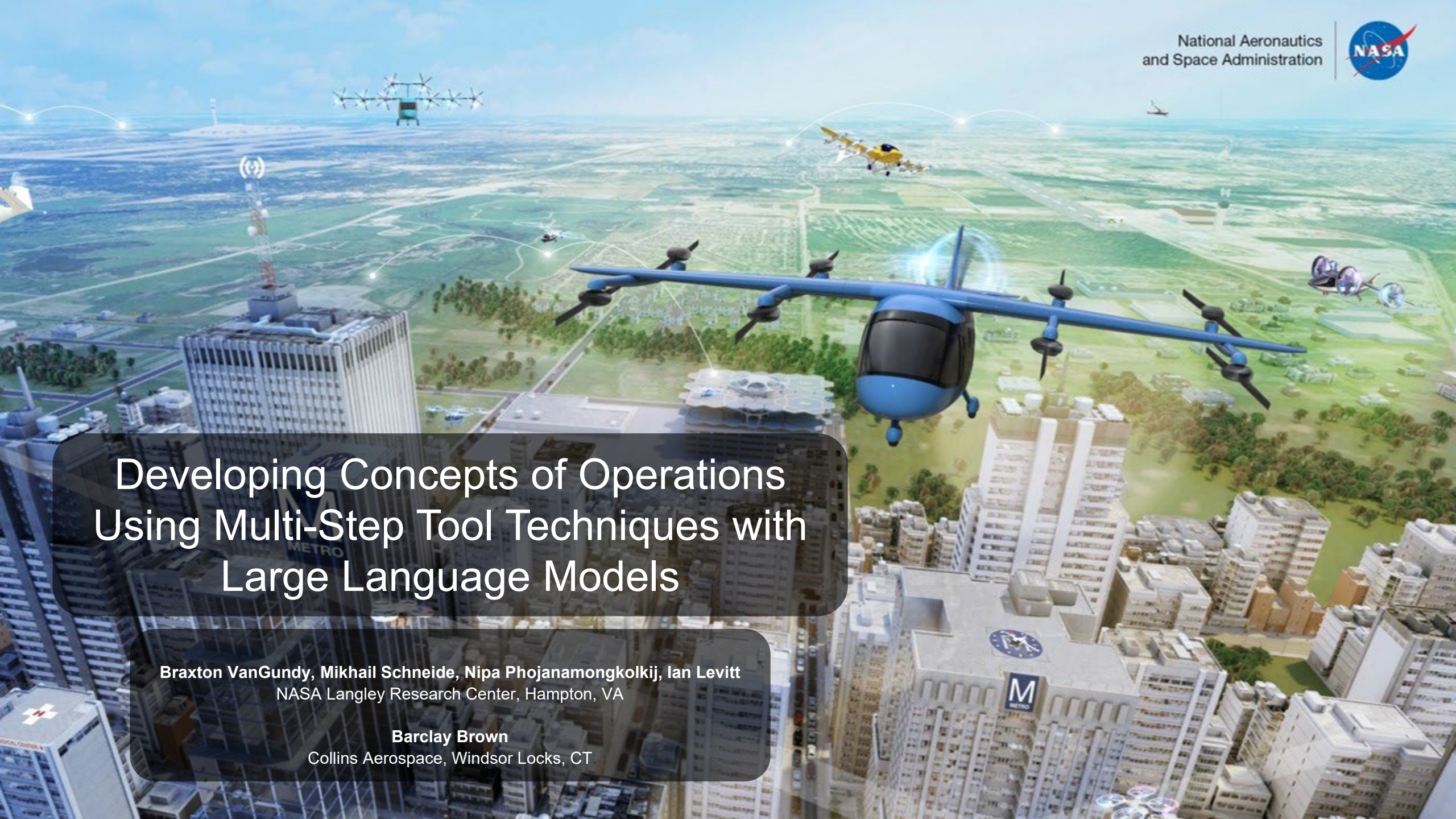


Developing Concepts of Operations Using Multi-Step Tool Techniques with Large Language Models

Braxton VanGundy, Mikhail Schneide, Nipa Phojanamongkolkij, Ian Levitt
NASA Langley Research Center, Hampton, VA

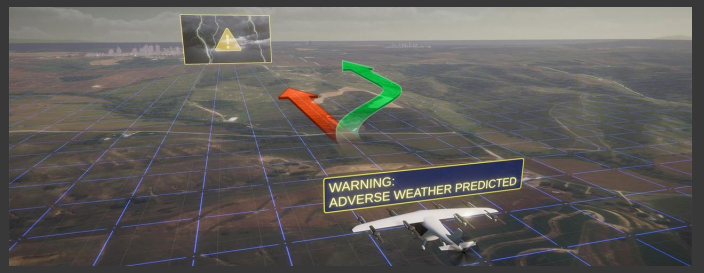
Barclay Brown
Collins Aerospace, Windsor Locks, CT

Introduction

- The National Aeronautics and Space Administration (NASA) Air Mobility Pathfinders (AMP) project is developing and evaluating concepts of operations (ConOps) for safe, secure, and scalable Urban Air Mobility (UAM) operations.

- The team's goal is to define structures and behaviors needed for system feasibility, readiness, and interoperability, establish a UAM knowledge base, and trace and validate assumptions and requirements relevant to Advanced Air Mobility (AAM).

AAM could be used in healthcare operations in the form of air taxi ambulances or medical supply delivery in the future. This concept graphic shows how a future AAM vehicle could aid in healthcare by carrying passengers to a hospital.

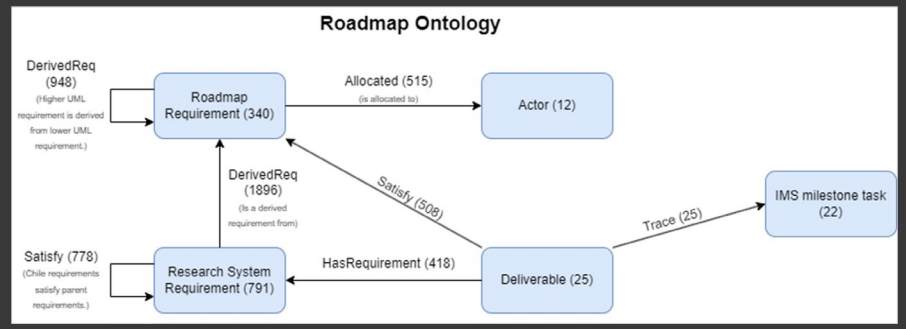


This NASA concept art represents how the addition of automated technologies on the aircraft like hazard avoidance could help ensure safe operation.

An idea for a future air taxi hovers over a municipal vertiport in this NASA illustration.

Problem Statement

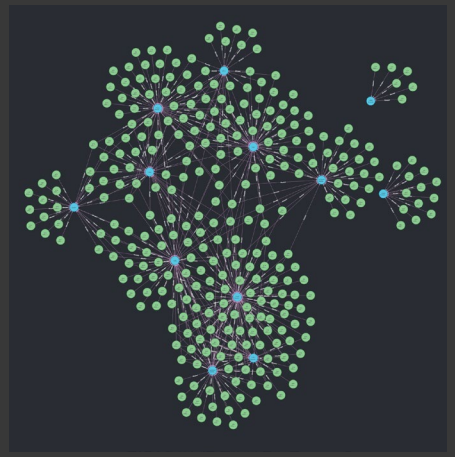
- Systems engineers (SEs) ingest information from many data sources:
  - Complex ontology mapped in modeling software, relational databases, and graph databases.
  - SE artifacts such as sequence and activity diagrams.
  - Large document repositories containing various document formats.
- SEs also need to interface with various tools to refine existing artifacts and models.



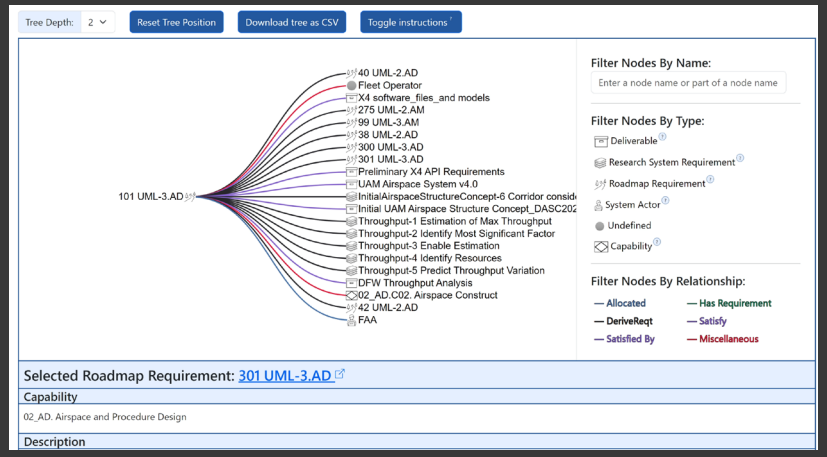
UAM Roadmap ontology used for the development of initial systems. Our team plans to expand our work to other AAM ontologies as they are developed.

Initial Solution

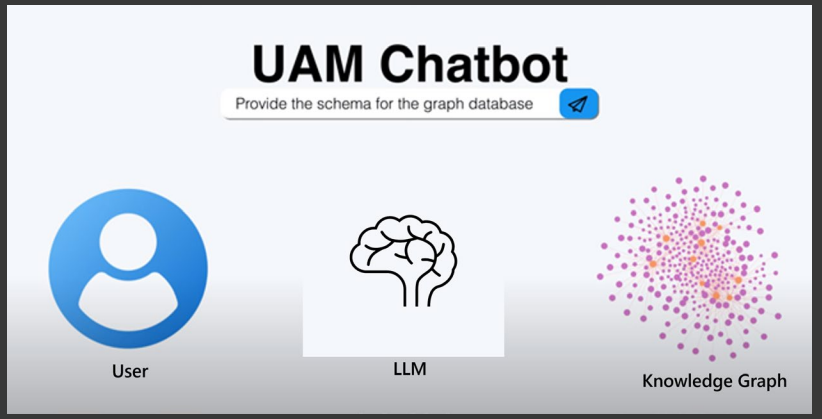
- NASA Langley Research Center (LaRC) is spearheading an innovative engineering approach known as the Knowledge-based Digital Platform (KbDP).
- This approach uses mathematical, data science, and system engineering principles to extract meaningful knowledge from relational and graph databases, document repositories, and system artifacts, which the human user leverages to greatly improve the efficiency and effectiveness of their research.
- Cutting edge machine learning (ML) algorithms and large language models (LLMs) play a large role in aiding our SEs in interrogating and navigating requirements and ConOps.



Graph database used to navigate and predict connections between requirements.

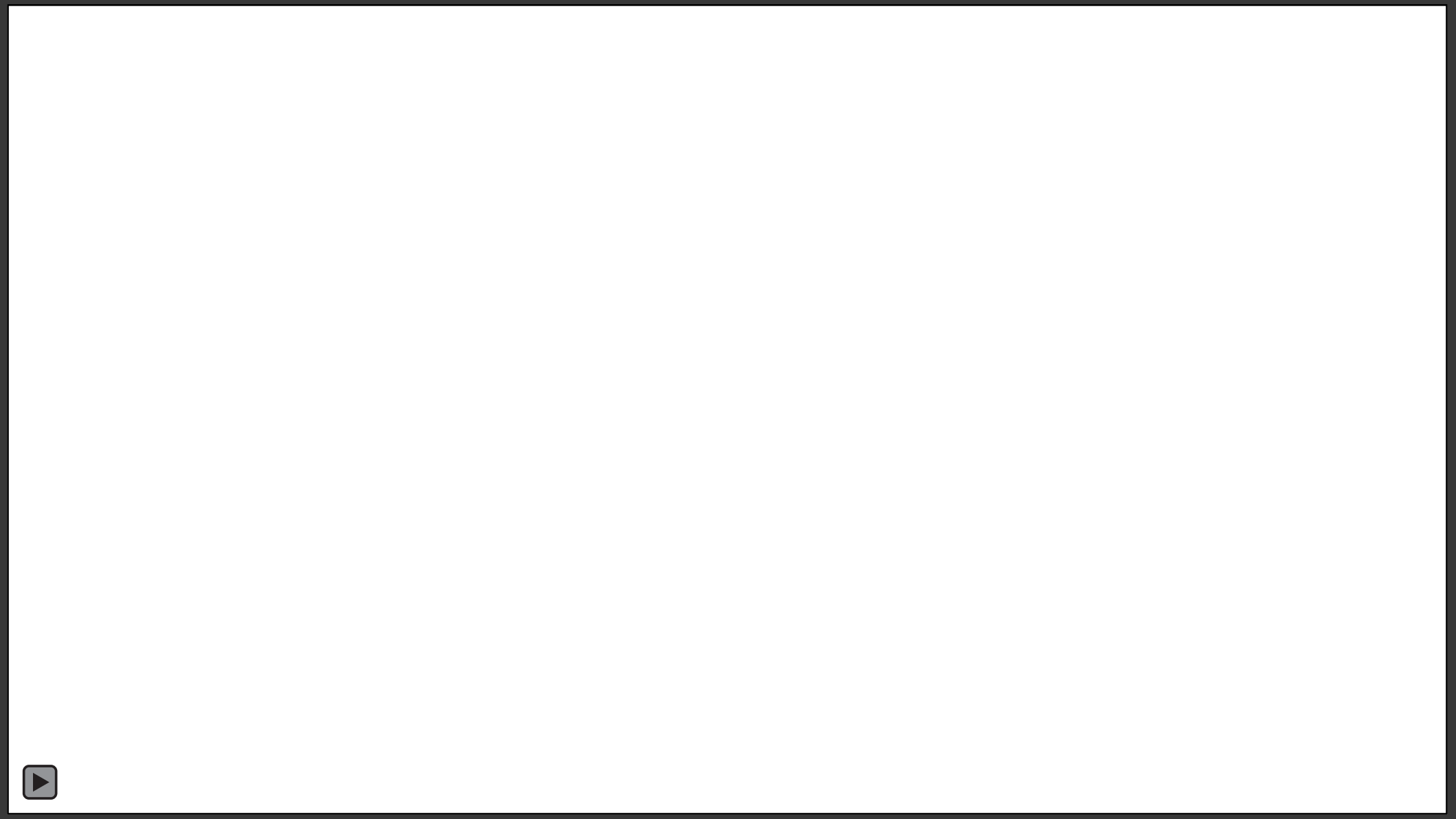


Advanced visualization capabilities are used to navigate the UAM ecosystem.



Integration of LLMs to allow the user to ask questions in plain English. No need to know complex query languages to access data and perform analysis.

Initial Solution - Demo

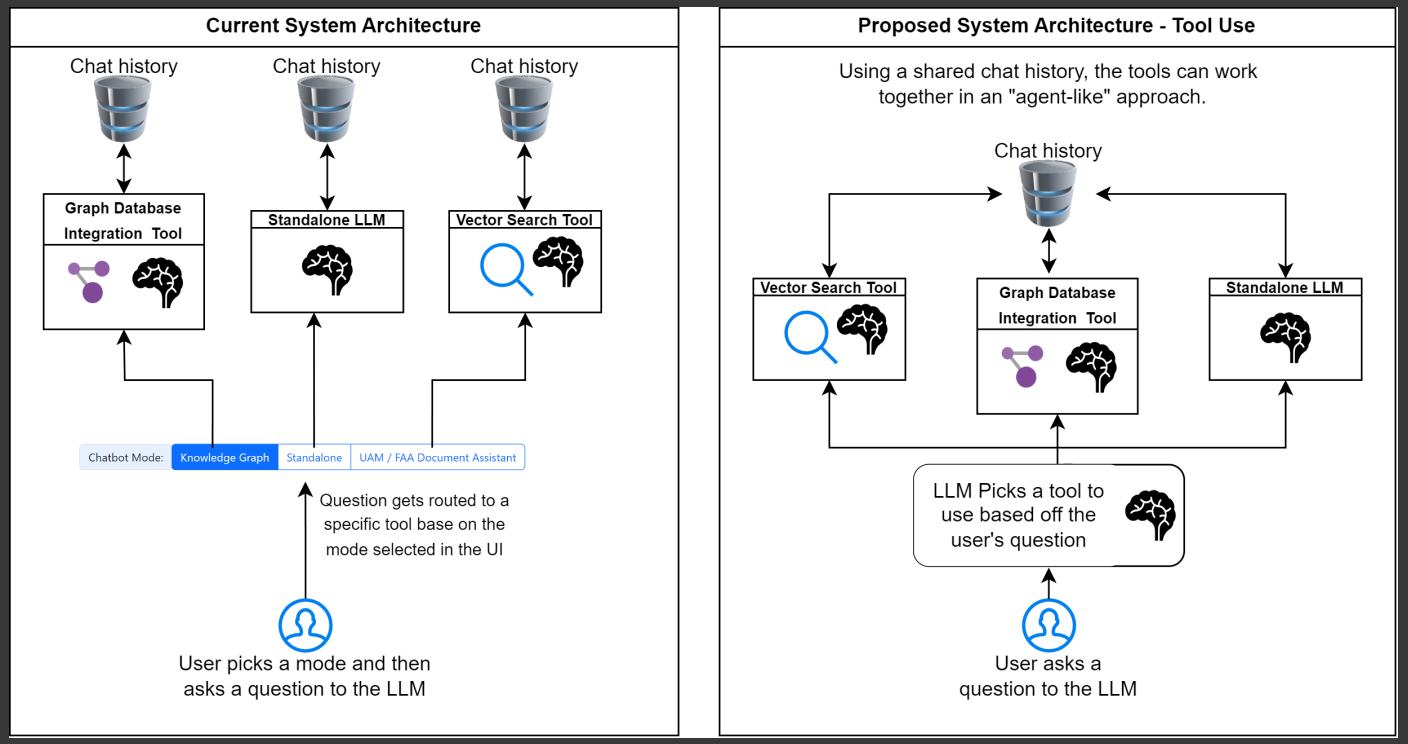


Improved Solution

- Our team is working on a new digital assistant architecture, seen on the right in the below figure, with the goal to address many of the user experience (UX) shortcomings of our original system while improving the accuracy and depth of responses from the LLM.

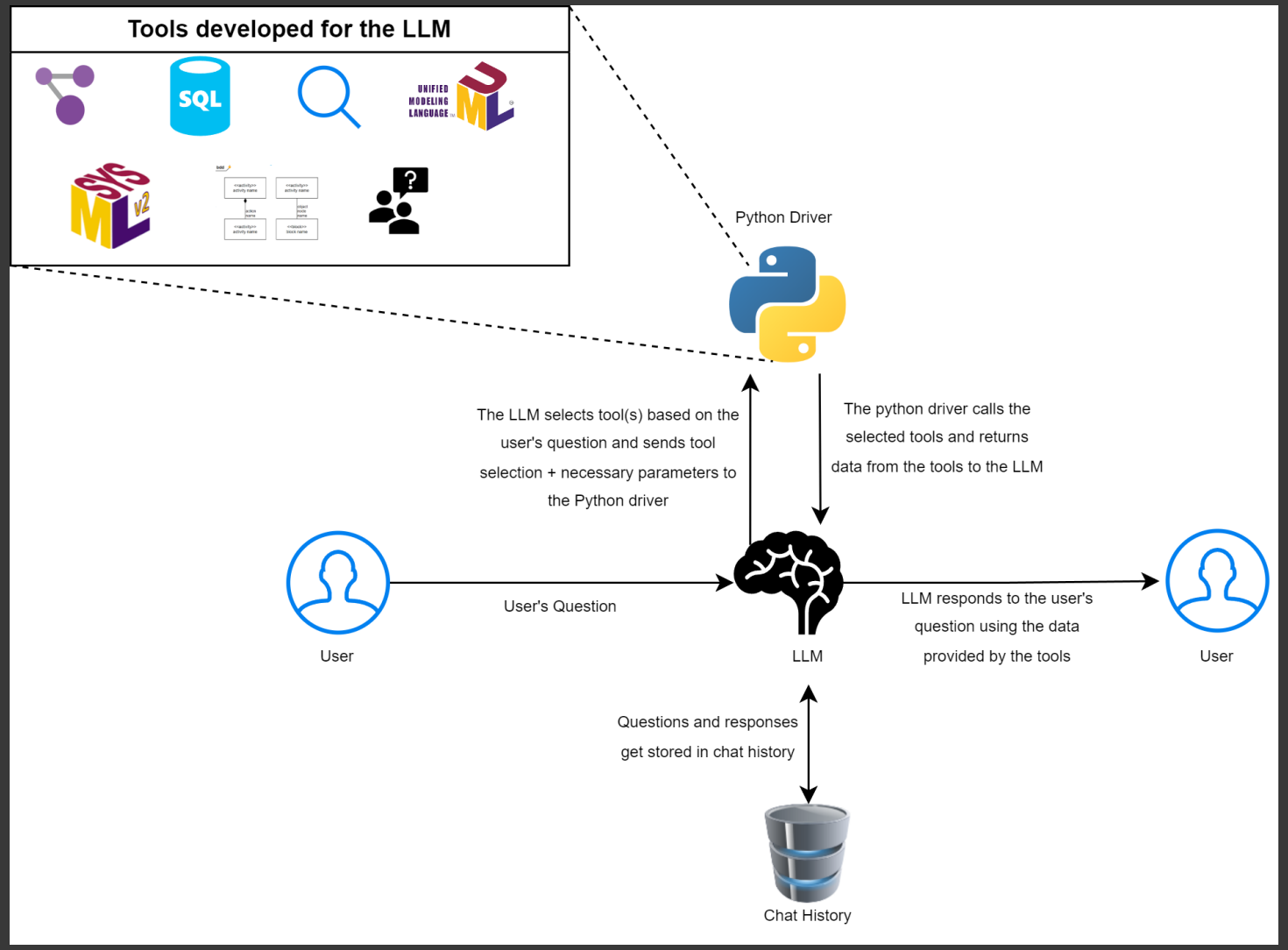
- This new architecture will allow the system to automatically select the appropriate tool to use based off the user's question.

- Will result in a collaborative pipeline where tools can pass data between other tools.

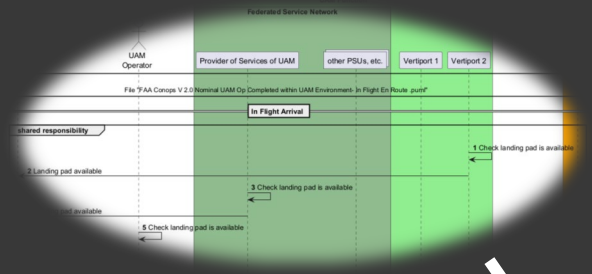


What is “Tool Use”?

- In the context of LLMs, a “tool use” implementation provides an LLM access to an assortment of pre-defined tools that it can choose from.
- The LLM chooses the correct tool, or tools, automatically based on the content within the user’s question.
- These tools can be used to fetch data related to a user’s question from a relational database or document repository, generate SE artifacts, enter data into a system, call on other LLMs, and much more.
- Tool use is also sometimes referred to as “function calling”.

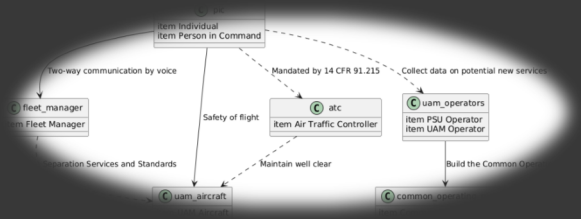


Our Tools



SE Artifact Finder

SysML V2 Generator

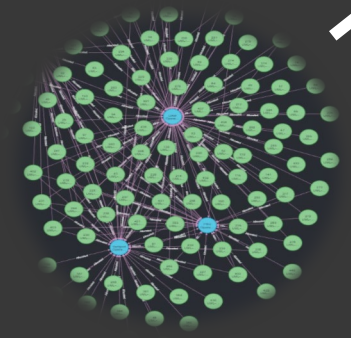


UML Diagram Generator

LLM

Relational Database

If it can be programed in code, it can be developed into a capability the LLM can use as a tool.



Graph Database

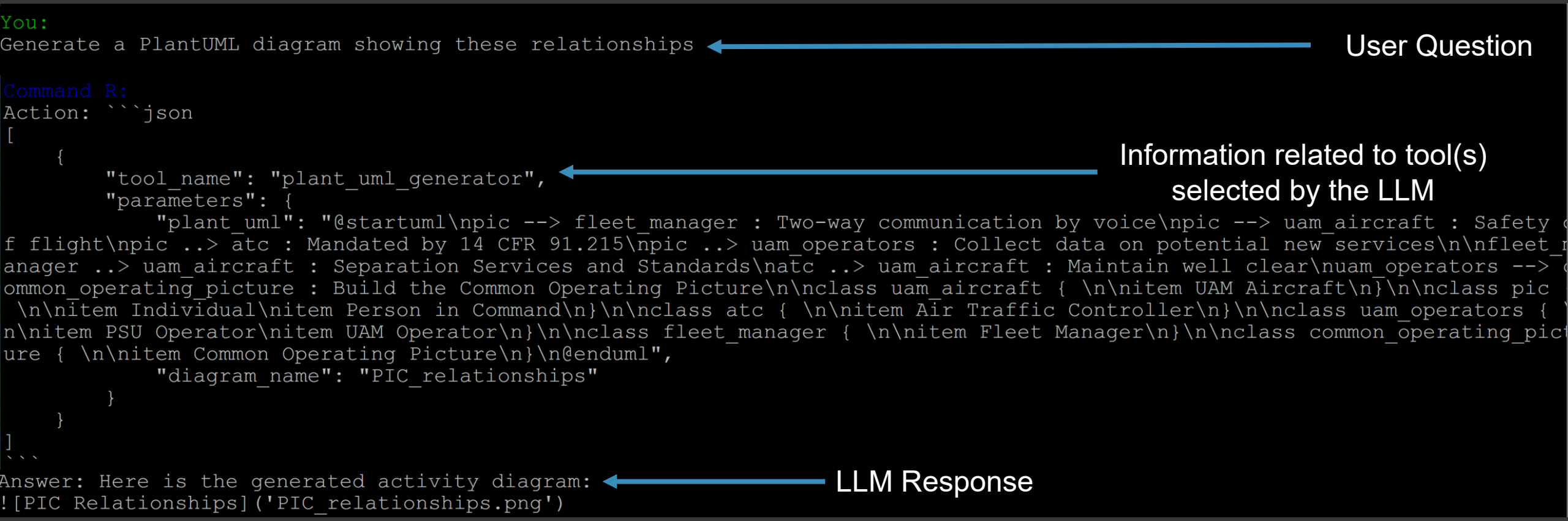
Generic question / answer

Document Search Engine

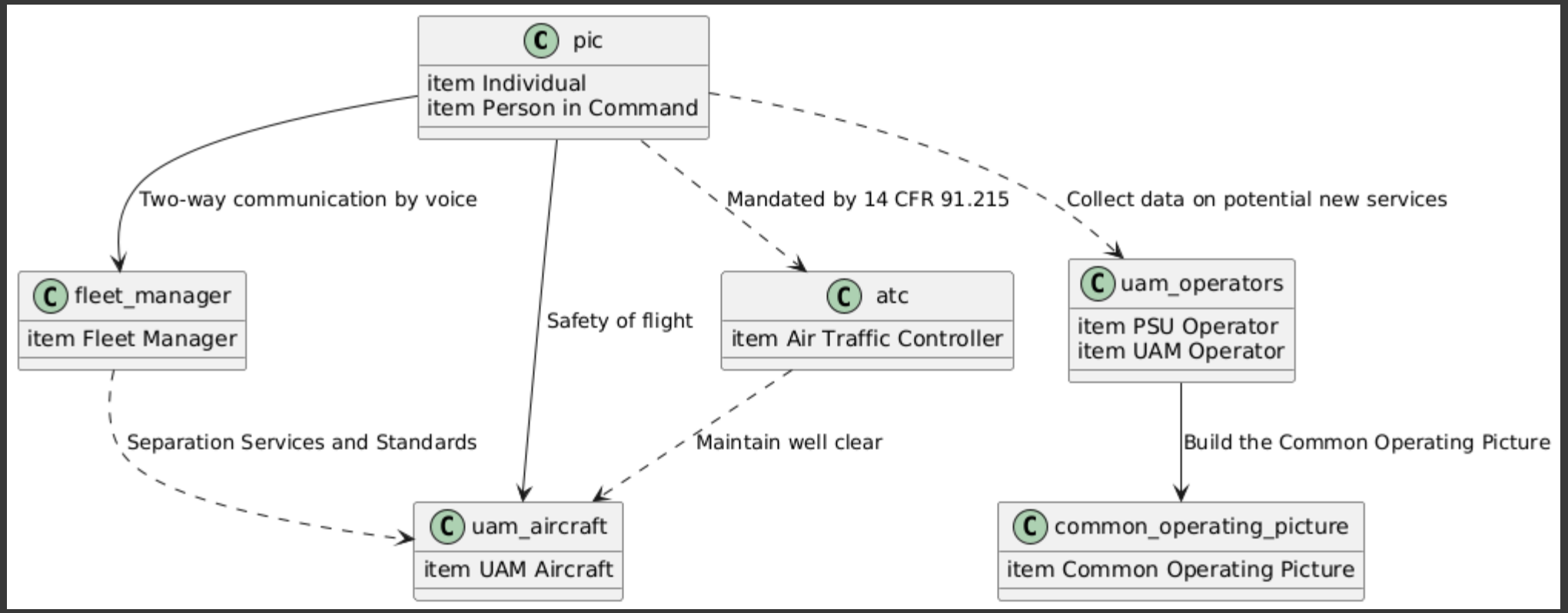
Prototype Demo

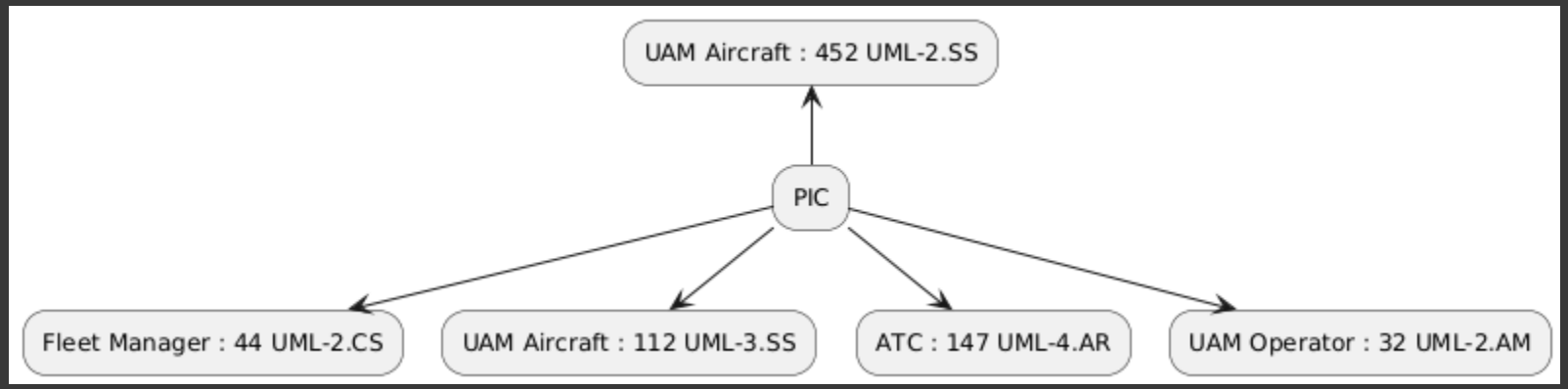
Picking the Right Tool

Generating SE Artifacts



Generating SE Artifacts





Finding SE Artifacts

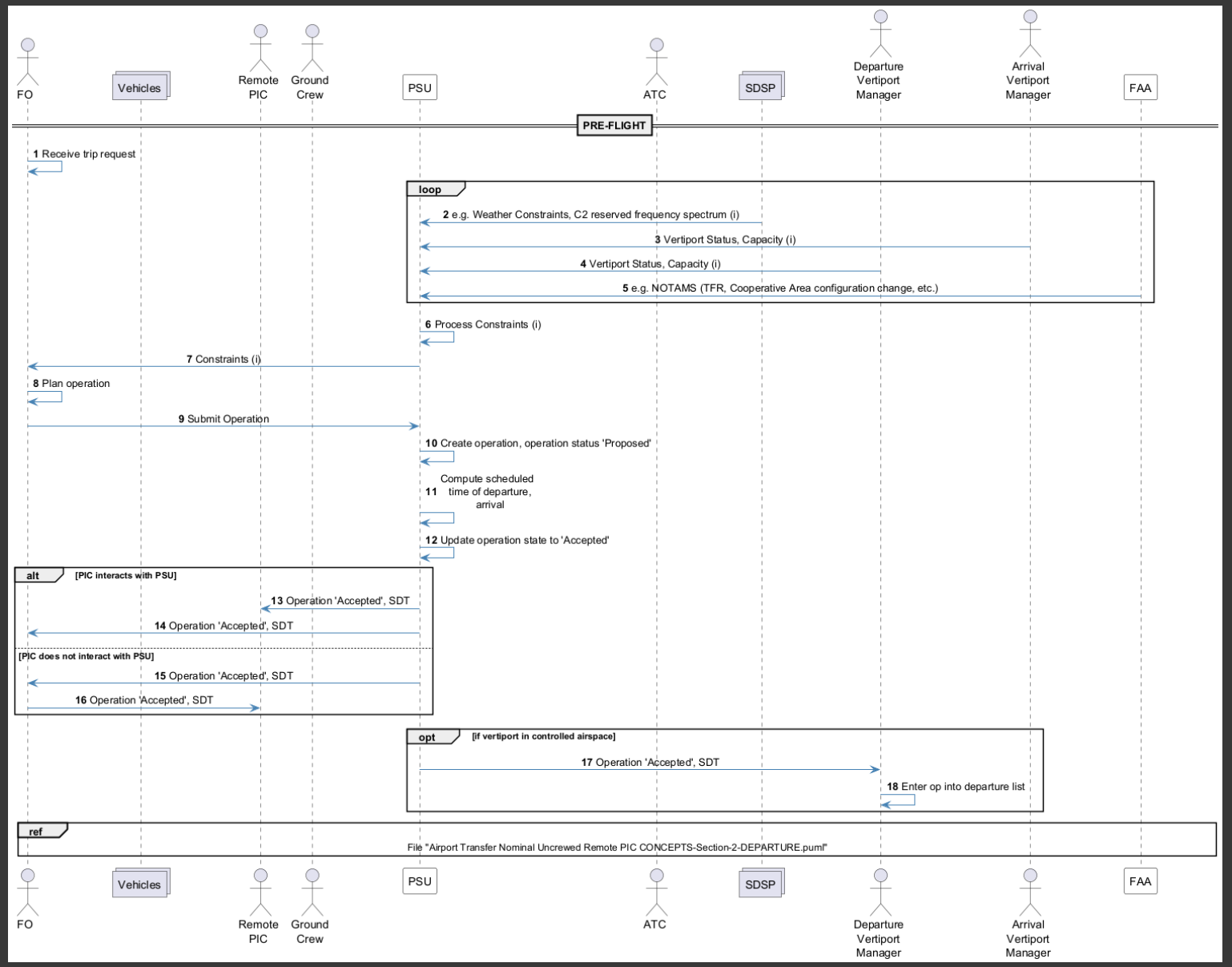

```
You:
Please show me a sequence diagram related to the Airport Transfer Nominal use case for pre-flight operations

Command R:
Action: ```json
[
    {
        "tool_name": "se_artifact_finder",
        "parameters": {
            "user_query": "Airport Transfer Nominal use case sequence diagram pre-flight"
        }
    }
]
```
<All keys matched successfully>
<All keys matched successfully>
Call: {'tool_name': 'se_artifact_finder', 'parameters': {'user_query': 'Airport Transfer Nominal use case sequence diagram p
re-flight'}}
Answer: Here is a diagram depicting the sequence of events for the Airport Transfer Nominal use case during pre-flight opera
tions:

![Airport Transfer Nominal Pre-flight Sequence Diagram]('Airport Transfer Nominal Uncrewed Remote PIC CONCEPTS (Section-1-P
RE-FLIGHT).png')

The above sequence diagram can be described as follows:
1. The Fleet Operator (FO) receives a trip request.
2. Pre-flight data is gathered by the Provider of Services for UAM (PSU) from various sources including the Supplemental Dat
a Service Providers (SDSP) for weather constraints and C2 reserved frequency spectrum, the Arrival and Departure Vertiport M
anagers for vertiport status and capacity, and The Federal Aviation Administration (FAA) for Notices to Airmen (NOTAMs) such
 as Temporary Flight Restrictions (TFR) and Cooperative Area configuration changes.
3. The PSU processes the gathered constraints.
4. The PSU sends the constraints to the FO.
5. The FO plans the operation using the provided constraints.
6. The FO submits the operation to the PSU.
7. The PSU creates the operation, setting the initial operation status to 'Proposed'.
8. The PSU computes the Scheduled Time of Departure (SDT) and Arrival.
9. The PSU updates the operation state to 'Accepted'.
10. If the Pilot in Command (PIC) interacts with the PSU:
- The PSU sends the 'Accepted' operation and SDT to the PIC.
- The PSU sends the 'Accepted' operation and SDT to the FO.
11. If the PIC does not interact with the PSU:
```
```

Finding SE Artifacts



Implementing Tool Use - Model

- First step is finding a LLM that supports tool use. Our team developed a prototype tool use system using a quantized version of Cohere's C4AI Command-R 35B LLM. We deployed this model on a local system using llama.cpp.

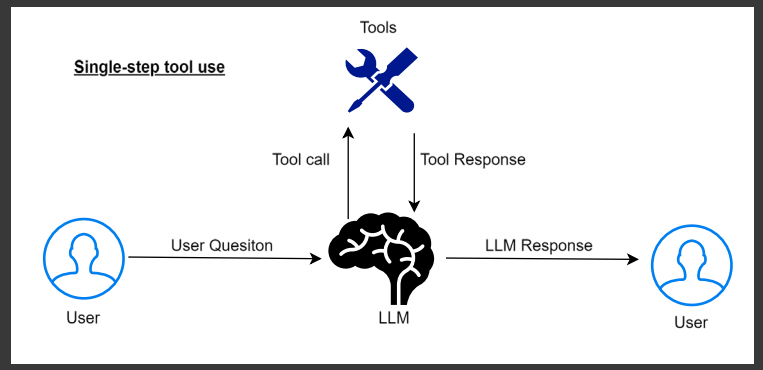
- For reliability, we restrict Command-R to only output the tool calls using a **grammar file**. Grammar files are used to constrain the LLM's output to a pre-defined set of responses. The grammar file is only used for tool calls, the LLM can bypass the grammar file by calling a "direct_response" tool to answer general questions. The grammar file is also bypassed when responding to the user after all tool responses have been received.

- There are also many other locally deployable models that support tool use:
  - Mixtral-8x22B-Instruct
  - Mistral Large and Mistral Small
  - Llama 3.1

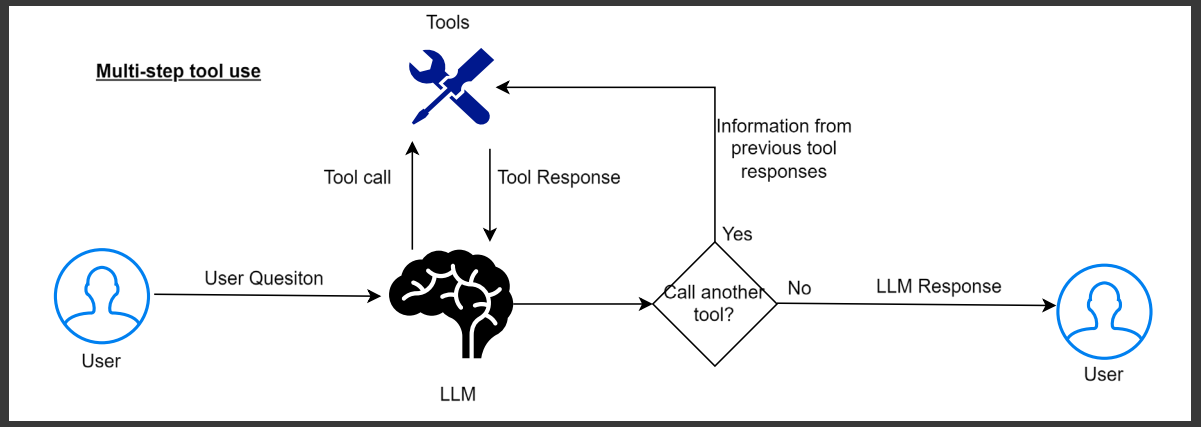
- Technically most LLMs support tool use, but those models not natively supporting it require extensive fine-tuning and / or prompt engineering.

Implementing Tool Use - Types

- There are two types of tool use implementations, Command-R supports both:
  - Single-step tool use where the model calls on the appropriate tool(s) once and then processes the data to generate a response.



  - Multi-step tool use can call on several tools in sequence, data from one tool can be passed to the tools called on next by the LLM. This is similar to an “Agent” based approach. Our implementation is a “pseudo” multi-step approach. Tools can see information from other tools via the chat history.



Implementing Tool Use - Prompt Structure

The general instructions define the LLM's role and provides a summary of when each tool should be used. They also include a safety preamble telling the LLM to avoid answering questions that are immoral / harmful.

The content of each dataset / database is defined in this section.

Each tool gets its own definition. The definitions contain the name of the tool, additional information about what the tool does, and the input parameters along with the expected output from the tool.

Tool Use Prompt for Command R - Summarized

General Instructions

If the user's query references or relates to system engineering and project management, prefer to use the "query_graph_database" to call the Neo4j graph database. If a user asks for a diagram or visualization, generate a diagram using the "plant_uml_generator" tool by sending the text representation of PlantUML to the tool as a parameter, use the chat history to generate the PlantUML. If the user asks for a SysML V2 text representation, generate a sysml text file by passing the sysml into the "sysmlv2_generator" function as a parameter. In all other circumstances, prefer to search through FAA/UAM documents using the "faa_uam_document_search" even if you know the answers so you are able to cite your sources. The only times you may directly answer are when you deem the subject in matter to be outside of the scope of FAA, UAM, and NASA specific domains. You may also directly answer if the information you need is already present within a <results> </results> block. .

Database Information

Document Database

Data within FAA/UAM document database
* NASA lessons learned
* UAM project specific documents

Graph Database

Data within the Graph database
* Roadmap Requirements
* System Actors
* Relationships between Roadmap Requirements and System Actors

Avaiable Tools

Document Search Tool

```python
def faa_uam_document_search(search_query: str) -> list[dict] | Error:

\"""Searches the document dataset.
Args:
    search_query (str): search query

Returns:
    list[dict] | Error: relevant documents or error
\"""
pass
```

Graph Database Search

```python
def graph_database_search(cypher_query: str) -> list[dict] | Error:

\"""Queries the graph database for project management and system engineering information.

We also include information about the node and link structure within the database here along with examples of how to perform cypher queries, but that text is too long to include here.

Args:
cypher_query(str): Input cypher query generated by the LLM

Returns:
    list[dict] | Error: graph database result or error
\"""
pass
```

Implementing Tool Use - Definition

- Tool definitions link to an equivalent Python function. Definitions contain the input parameters, expected return data, and a description of what the tool does. The description can also contain additional instruction, for example telling the LLM how to generate UML if you have a UML generation tool.

Tool name and input / output params.

Description of the tool

Information about the input arguments so the LLM knows what to send to the tool.

Information about the data the tool will return.

```python
def faa_uam_document_search(search_query: str) -> list[dict] | Error:
    \"""Searches the FAA/UAM document dataset.

    Args:
        search_query (str): search query

    Returns:
        list[dict] | Error: relevant documents or error
    \"""
    pass
```

Findings

- Tool approach reliably selects the correct tool based off the users question with the use of a grammar file.
- Command-R is extremely memory heavy; the cache is not quantized so context is very memory expensive. It supports up to 128K tokens, but we only had enough VRAM to run 10K (consumed 36GB of VRAM).
- Command-R needs a lot of instruction in the prompt to work properly, our prompt is over 5,000 tokens!
- Difficulty getting multi-step tool calling to work reliably. Command-R supports it, but we could not get it to consistently perform multi-step workflows where it should. Ended up deploying pseudo multi-step approach.
- Working integration into our production system, these tools will allow SEs to more easily, and intuitively, explore the AAM ecosystem, ultimately improving the efficiency and effectiveness of the SE's research and decision-making processes surrounding ConOps development and validation.

Questions

References

Cohere For AI . (n.d.). *Cohereforai/C4AI-command-R-V01 · hugging face*. CohereForAI/c4ai-command-r-v01 · Hugging Face. https://huggingface.co/CohereForAI/c4ai-command-r-v01

Cohere For AI . (n.d.). *Tool use with Cohere's models - cohere docs*. Cohere AI. https://docs.cohere.com/docs/tool-use

Phojanamongkolkij, N., VanGundy, B., Polavarapu, R., Levitt, I., & Brown, B. (2023, September). Requirement Discovery Using Embedded Knowledge Graph with ChatGPT. In AI4SE & SE4AI Research and Application Workshop.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., ... & Wang, C. (2023). Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155.