A Systems Approach to AI Model Integration and Performance Evaluation for the Generic UAM Simulation Framework

Newton H. Campbell Jr.*, Michael J. Acheson†, and Irene M. Gregory‡
NASA Langley Research Center, Hampton, VA, 23681, USA

This paper introduces `py-guam`, an open-source experimentation framework developed for the NASA Generic Urban Air Mobility simulation (GUAM) environment, facilitating the integration and evaluation of advanced artificial intelligence (AI) algorithms. We present a systems approach which enables the seamless incorporation of data-driven models, including off-nominal and failure state detection, into the GUAM's Cognitive Architecture (CA). The framework supports customizable experimentation parameters, derives Safety Performance Indicators (SPIs) from UL 4600 safety case analyses, and employs rapid UAM simulations to assess AI impacts on flight performance across diverse scenarios. Through comprehensive testing and validation experiments, we demonstrate GUAM's capability to enhance safety and efficiency in urban air mobility operations. Additionally, the open-source nature of `py-guam` fosters community collaboration, ensuring continuous improvement and adaptability to evolving technological advancements. This work establishes a robust tool for developing and testing AI-driven urban air mobility (UAM) systems, advancing the safety and reliability of autonomous urban air vehicles.

AI	artificial intelligence
ATM	air traffic management
CA	Cognitive Architecture
eVTOL	electric vertical takeoff and landing
FAA	Federal Aviation Administration
GUAM	Generic Urban Air Mobility simulation
HAVOK	Hankel Alternative View of Koopman
ICM	Intelligent Contingency Management
L+C	Lift-Plus-Cruise
LOC	loss of control
NASA	National Aeronautics and Space Administration
RL	reinforcement learning
SME	subject-matter expert
SPI	Safety Performance Indicator
STRL	Strategic-Tactical-Reactive-Layered
UAM	urban air mobility
UAV	uncrewed aerial vehicle
VAE	variational autoencoder
VTOL	vertical takeoff and landing
XAI	explainable AI

I. Introduction

Urban Air Mobility (UAM) is set to transform the transportation sector by introducing autonomous air vehicles into dense urban and sparse rural environments alike, at altitudes lower than standard commercial aviation. However, the complexities of these environments, coupled with the safety-critical nature of air traffic, demand rigorous testing of autonomous systems. Central to the functionality of modern UAM systems are AI algorithms embedded within flight autonomy algorithms. These algorithms are responsible for critical tasks such as navigation, collision avoidance, mission planning, and real-time decision-making. The effectiveness and reliability of these systems are essential for safe operation. Consequently, rigorous testing and validation under simulation of both nominal and off-nominal scenarios is critical. Traditional simulation environments often lack the flexibility and scalability required to comprehensively evaluate AI models in dynamic and unpredictable urban environments. This limitation poses a significant barrier to validating the robustness and reliability of AI algorithms in real-world applications, thereby hindering the widespread adoption of UAM technologies.

In integrating AI into our UAM framework, we identified the need for rigorous safety assessment to ensure reliable and secure operations, even at research maturity levels. Over the last year, we conducted a comprehensive review of the

*AI Subject Matter Expert, NASA Consolidated Applications and Platform Services (NCAPS), National Aeronautics and Space Administration (NASA) Langley Research Center

†Senior Research Engineer, Dynamic Systems and Control Branch, Michael.J.Acheson@nasa.gov, Member AIAA.

‡NASA Senior Technologist for Advanced Control Theory and Application, Irene.M.Gregory@nasa.gov, Fellow AIAA.

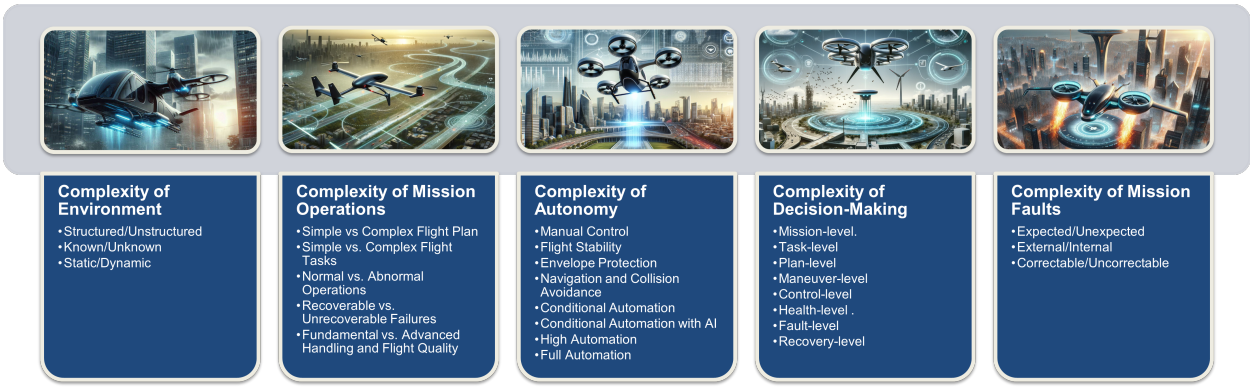

Fig. 1 Each UL4600 safety case sub-claim is supported by qualitative assessments across complexity categories which inform the modular design of GUAM and `py-guam`.

UL 4600 safety cases developed in our prior work [1], focusing on metrics and SPIs essential for evaluating AI-integrated autonomous vehicles. Collaborating with subject matter experts across NASA Langley Research Center, we used surveys and interviews to gather insights that informed updates to these safety cases, categorized in Figure 1. This collaborative effort directly influenced the design of the `py-guam` framework. We designed `py-guam` as an open-source library to allow developers to easily integrate diverse analytics and models into the GUAM simulation environment*. We aim to promote `py-guam` as a collaborative ecosystem where developers and researchers can contribute to and benefit from enhanced AI integration and safety validation within UAM systems.

By advancing a systems approach in GUAM, `py-guam` ensures AI algorithms are effectively validated under various conditions for robust real-world performance. The primary objectives of this paper are:

- Present a systems approach to the integration, testing, and characterization of AI models within the GUAM simulation framework.
- Introduce methods of integrating claim evidence and SPIs derived from UL 4600 safety case analyses into GUAM.
- Demonstrate the use of the `py-guam` open-source Python library for integrating off-nominal and failure state detection algorithms.
- Provide a platform for continual improvement and feedback through open-source development.

II. Background

Cognitive Architectures (CAs) are increasingly integrated into UAM vehicle control systems, significantly enhancing autonomy and safety in UAM applications. The `py-guam` library was developed to facilitate the incorporation of Python algorithms into the GUAM CA, ensuring easy enhancement and extensibility of the CA.

A. Cognitive Architectures

A CA is a task-independent framework that enables one or more agents to learn, store, and apply knowledge to generate behavior. It is a software implementation of a general intelligence theory, integrating multiple cognitive processes to support intelligent actions across diverse tasks. CAs have been extensively investigated in the domain of autonomous mission planning and control of vehicles. For instance, the Soar CA has been employed for decision-making in autonomous UAM operations, enabling vehicles to interact with dynamic environments and adapt to unforeseen events[2]. By integrating human-like decision-making processes, Soar facilitates enhanced autonomy for UAVs operating in complex and dynamic scenarios. Similarly, Zhang et al. introduced a cognitive control architecture for UAM vehicle collision avoidance inspired by human cognitive processing[3]. This architecture enhances real-time decision-making capabilities while reducing computational complexity, demonstrating the potential of cognitive systems in advancing UAV autonomy.

Layered CAs, such as the Strategic-Tactical-Reactive-Layered (STRL) model, have been developed to integrate high-level decision-making with low-level control functions in uncrewed aerial vehicle (UAV) systems[4]. STRL integrates in-depth planning into the CA, tightly coupling advanced analysis and synthesis of the data with actual planning. These architectures enable UAVs to perform complex tasks such as coalition formation and dynamic path

*`py-guam` can be accessed through the NASA GUAM repository at: `https://github.com/NASA/Generic-Urban-Air-Mobility-GUAM`

planning, thus enhancing their autonomy in executing coordinated missions. The integration of high-level cognitive tasks with real-time control systems, as evidenced in these studies, underscores the significance of CAs in advancing UAV autonomy for UAM applications.

Recent advancements in machine learning, particularly in deep learning and reinforcement learning, have further propelled the capabilities of CAs in UAVs. Techniques such as deep reinforcement learning have been heavily researched for UAV navigation and control, enabling autonomous systems to learn optimal policies through interaction with the environment[5]. Additionally, the incorporation of explainable AI (XAI) methods within CAs addresses the interpretability and transparency concerns of AI-driven autonomous systems[6, 7], which is crucial for trust and adoption in safety-critical applications like UAM.

Performance evaluation frameworks based on SPIs are vital for ensuring the reliability and robustness of AI models within UAM environments. The UL 4600 standard, which provides comprehensive safety guidelines for autonomous vehicles, offers a foundation for deriving SPIs pertinent to evaluating AI algorithms and as such, evaluating CAs[8, 9]. Bhattacharyya et al. emphasize the importance of formal verification in CAs, introducing methods to ensure that autonomous agents satisfy safety and performance requirements in mission-critical operations[10].

Simulation-based testing environments remain a critical step to assess the safety and performance of UAVs in urban settings. High-fidelity simulations enable the testing of autonomous systems under a wide range of scenarios, including adverse weather conditions and complex obstacle-rich environments[11, 12]. These testing frameworks are essential for validating AI models before deployment in real-world UAM operations.

Beyond safety considerations, the energy constraints inherent to electric vertical takeoff and landing (eVTOL) aircraft pose unique challenges for the feasibility of UAM operations. Recent studies, such as that by Taye and Wei[13], have examined mission feasibility under battery energy limitations, highlighting the trade-offs between operational safety and energy consumption in UAM missions. The ability to balance such constraints while maintaining high levels of safety and performance is critical for successfully deploying UAM systems. Advances in battery technology and energy-efficient flight planning algorithms are actively being researched to address these challenges[14].

The regulatory landscape for UAM is evolving, with aviation authorities working to establish standards and certifications for autonomous aerial vehicles. Compliance with regulations set forth by organizations such as the Federal Aviation Administration (FAA) and International Civil Aviation Organization (ICAO) is essential for the legal operation of UAVs in urban environments[15]. Ethical considerations, including privacy, security, and societal impact, also play a crucial role in the acceptance of UAM technologies[16]. CAs must be designed with these considerations in mind to ensure responsible deployment. Despite the potential benefits of UAM, several challenges hinder its mainstream adoption. Cohen and Shaheen[17] outlined key barriers to UAM implementation, including community acceptance, regulatory challenges, and the necessity for robust safety and security frameworks. Addressing these challenges requires collaborative efforts among industry, government, and academia to develop scalable solutions for UAM operations. The role of CAs in surmounting these obstacles is particularly significant, as they provide a foundation for developing intelligent, autonomous systems capable of navigating the complexities of urban airspace.

Pilot programs and public demonstrations are being conducted to assess the feasibility and public perception of UAM. For example, projects like Uber Elevate and Airbus's Vahana have explored the practical aspects of UAM services[18, 19]. These initiatives provide valuable insights into the technical, regulatory, and societal challenges associated with UAM.

B. GUAM

A key focus of NASA's Transformational Tools and Technologies (TTT) Autonomous Systems (AS) Project is on achieving increasing levels of autonomy in UAM flight. The Intelligent Contingency Management (ICM) sub-project [20] aims to develop tools and experimental methods to effectively tackle these challenges. ICM integrates AI [21] with advanced autonomy algorithms [22–24] and high-fidelity modeling techniques [25–27] to address safety and efficiency concerns in UAM operations. The research team conducts theoretical and experimental studies on various aspects of ICM, including data uncertainty quantification, vehicle health management, trajectory optimization, contingency detection and diagnosis, mission replanning and reconfiguration, human-machine interaction, and verification and validation.

To test and evaluate ICM concepts and algorithms in realistic scenarios, the team developed a UAM simulation framework that integrates multiple tools and models. The public version of this simulator, known as the Generic Urban Air Mobility simulation (GUAM), enables the research community to introduce algorithms at the intersection of UAM, systems thinking, and AI. GUAM, in both its public and internal experimental versions, serves as the primary work

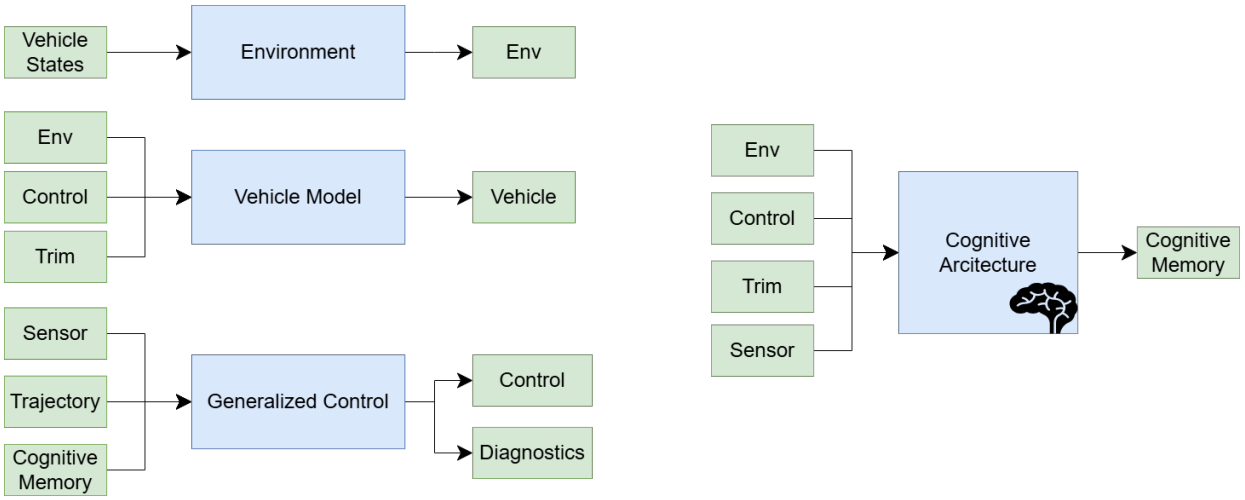

Fig. 2 GUAM High-Level Architecture

product of the ICM program. The CA operates in parallel with other simulation components within GUAM, as shown in Figure 2.

GUAM employs advanced analytics to interpret the vehicle's current and future states and calculate variables critical to mission execution. The CA gathers and synthesizes data from GUAM's primary simulation subsystems—Environment, Vehicle Modeling, and Generalized Control—integrating diverse information streams. Inputs include environmental data (weather conditions, wind patterns, atmospheric variables), control and trim inputs adjusting the vehicle's aerodynamic surfaces and propulsion systems, and sensor data from onboard instruments like accelerometers, gyroscopes, Global Positioning System (GPS) units, and Light Detection and Ranging (LiDAR) systems.

As depicted in Figure 2, the CA produces Cognitive Memory for other simulation components. Cognitive Memory is a dynamic repository encapsulating the current and future states of the vehicle and its environment. It involves monitoring and projecting the vehicle's internal and external conditions, using sensor fusion and trajectory prediction to assess vehicle health and situational awareness. The Generalized Control subsystem leverages Cognitive Memory within the Mission Manager (not shown), a sub-component responsible for risk management, planning, and decision-making. The Mission Manager adjusts control actions, anticipates potential system responses, and implements adaptive control mechanisms. It parameterizes adaptive control laws, onboard trajectory planning, and collision avoidance to ensure safety and efficiency, even amidst uncertainties and faults [28, 29]. In the context of John Boyd's Observe-Orient-Decide-Active (OODA) loop[30], the CA functions as a highly informed **Observe** module and the Mission Manager facilitates **Orient**, **Decide**, and to a small extent, **Act**.

Central to GUAM's effectiveness is its vehicle modeling capability, particularly for the Lift-Plus-Cruise (L+C) configuration[31], a type of eVTOL design [32]. Although the L+C is the primary vehicle used for ICM testing, GUAM supports the integration of multiple UAM vehicle types. The modeling approach integrates computational methods, real-time aerodynamic monitoring, and machine learning for system identification and system identification and dynamic adjustments. Ongoing GUAM development focuses on enhancing contingency management, including unforeseen situations, while reducing reliance on rigid rule-based systems.

GUAM leverages foundational technologies that provide a robust platform for modeling, simulation, and analysis of complex aerospace systems. Illustrated in Figure 3, MATLAB and Simulink[33] are central to this environment, offering extensive capabilities for simulating dynamic systems and designing control architectures. While widely adopted due to their powerful tools and comprehensive libraries, integrating advanced analytics and machine learning algorithms within MATLAB/Simulink presents challenges due to limited library availability and computational efficiency. To overcome these limitations, we developed `py-guam` to bridge MATLAB with Python — a leading language for data science and machine learning. Python's powerful libraries, such as NumPy, SciPy, scikit-learn, and TensorFlow, provide the GUAM the necessary tools for processing high-dimensional data and implementing sophisticated machine learning models. The adapters we used to develop this bridge have shaped a design pattern that will be used for future integration of GUAM with other languages and third-party applications as needed. This integration allows the CA to leverage Python's advanced capabilities while maintaining the real-time performance and modularity inherent to MATLAB/Simulink.

`py-guam` ensures efficient integration between MATLAB and Python through key architectural components.

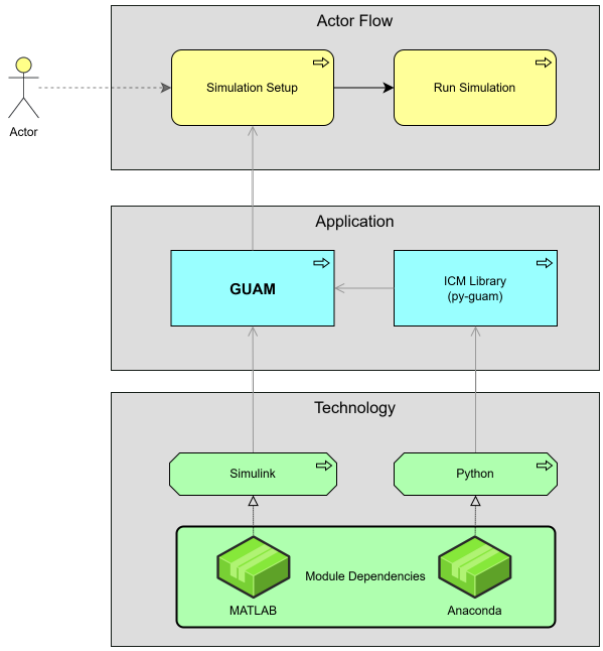

Fig. 3 GUAM Actor Workflow and Technology Stack Integration

MATLAB-Python adapter functions allow MATLAB to invoke Python modules, managing data type conversions for seamless data exchange. Embedding Python function calls within Simulink blocks enables direct utilization within simulation models while maintaining modularity and real-time performance. This design minimizes development overhead and maximizes performance, enabling the GUAM simulation environment to incorporate advanced AI models and data processing functions.

III. System Design

A. GUAM Cognitive Architecture (CA)

GUAM's CA is an informed observation framework that interprets data in real-time to provide insights for vehicle control and planning. It is intended to inform both high-level mission objectives and low-level control strategies through a series of interconnected components. Unlike traditional CAs that primarily focus on decision-making and collision avoidance, GUAM decouples decision-making from its CA and integrates advanced machine learning algorithms and real-time data processing to inform planning for safety and efficiency concurrently.

The high-level design for the GUAM CA is illustrated in Figure 4. The vehicle's interpretation of the World (called World Interpretation in Figure 4) is captured through a Minimum Realization over varying timespans. The other sub-components of the vehicle have access to this minimal realization and raw sensor data. Critical Point Detection focuses on when the vehicle or flight environment has reached or is approaching some extreme, whether that is flying on the edge of the envelope, being in an loss of control (LOC) state, or flying in an environment that is extreme enough that it will significantly affect vehicle flight and/or autonomy. Critical Point Detection identifies thresholds for the vehicle and flight mission, identifying statistical significance between vehicle state variables in relation to the vehicle reaching those thresholds. Energy Estimators estimate vehicle reachability requirements and determine the feasibility of different flight paths/actions. This involves using algorithms such as Lagrangian and Hamiltonian estimators, based on principles from classical mechanics, and corresponding neural network models such as Hamiltonian Neural Networks to achieve these tasks. This design enables GUAM to validate UAM vehicle performance and safety, setting the foundation for future implementation[†] and integration within the GUAM CA.

The Reinforcement Learning sub-component uses reinforcement learning algorithms to optimize the vehicle's

[†]Note that World Interpretation, Critical Point Detection, and Command Performance are the only parts of the architecture implemented and published in the open-source repository at the time of publication.

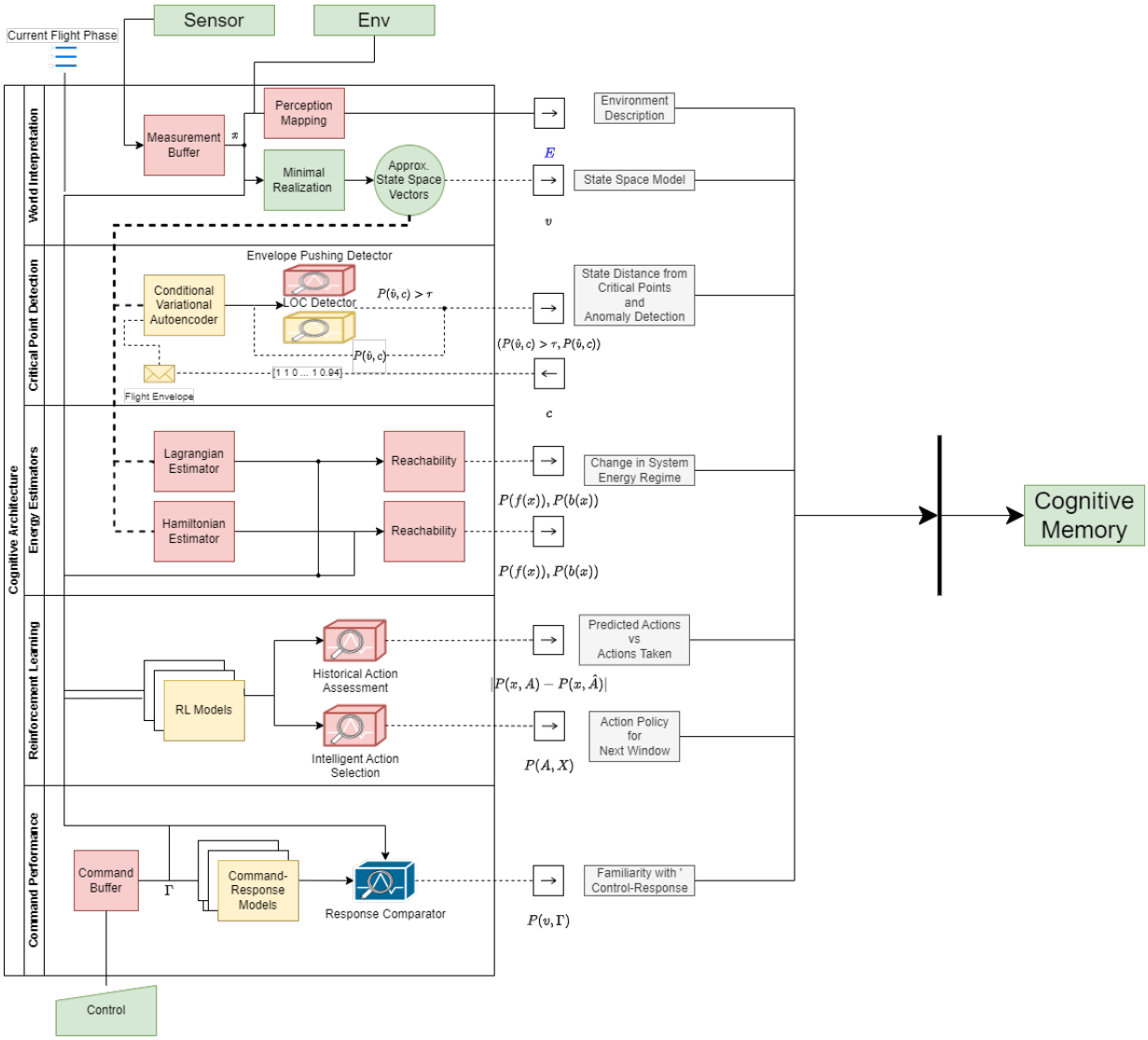

Fig. 4 GUAM Cognitive Architecture

behavior over time by learning from its past actions and outcomes. This involves using algorithms such as Q-learning or State-action-reward-state-action (SARSA) to assess the vehicle's historical actions and evaluate the proposed actions' effectiveness. Finally, the Command Performance sub-component is responsible for mapping vehicle commands to recent vehicle responses and interpreting the effectiveness of those commands in achieving the mission goals. This involves using supervised learning or decision tree algorithms to learn the relationships between vehicle commands and vehicle responses and to make predictions about the effectiveness of new commands.

B. Enabling Advanced Analytics with `py-guam`

The design of `py-guam` encompasses several key architectural components:

1) **MATLAB-Python Adapter Functions:** MATLAB-Python adapter functions in `py-guam` are instantiated as part of the simulation initialization process to seamlessly invoke Python modules from MATLAB. Adapter functions begin by determining the module path for the Python code to ensure the correct directory is targeted, and then checks for the existence of the required Python module. If the module exists, it adds the module path to Python's system path using `py.sys.path` and imports the module with `py.importlib.import_module`. The adapter functions handle persistent variables maintains references to the imported Python module and maintain flags indicating whether the script exists. Data type conversions are managed carefully; for example, MATLAB

arrays are converted to NumPy arrays using `py.numpy.array`, facilitating compatibility with Python functions. Additionally, the adapters pre-allocate storage for return variables to improve computational efficiency. Error handling is incorporated to manage cases where the Python module is not found or fails to execute, ensuring that the simulation can proceed gracefully by returning default values when necessary.

Algorithm 1 MATLAB - Python Adapter Function Initialization

```
procedure InitializeAdapter
    module_path ← DetermineModulePath
    if PythonModule is not in persistent storage then
        if FileExists(module_path + module file) then
            if module_path not in Python's sys.path then
                Insert(Python's sys.path, module_path)
            end if
            PythonModule ← ImportModule(module name)
            Store PythonModule in persistent storage
            scriptExists ← true
        else
            Warning("Could not load the Python module.")
            scriptExists ← false
        end if
    end if
end procedure
```

2) **Python Modules for Advanced Analytics:** The Python modules within `py-guam` are designed to implement advanced analytics and machine learning algorithms that are computationally intensive or better supported in Python's rich ecosystem. Modules such as `havok.py` leverage libraries like NumPy for numerical computations and SciPy for advanced mathematical functions. `havok.py` implements Hankel Alternative View of Koopman (HAVOK) analysis, which is used for dimensionality reduction and system identification in dynamical systems[34]. This module is structured to be callable from MATLAB without requiring real-time execution constraints. It is stateless and employs functions that accept input data and return processed results. By offloading these kinds of tasks to Python, the GUAM simulation benefits from optimized libraries and can implement algorithms like variational autoencoders (VAEs), convolutional neural networks, and other machine learning models that are more readily available in Python.

Algorithm 2 Python Module Execution Flow

```
procedure PythonFunction(Python_Data)
    Perform computations using Python libraries (e.g., NumPy, SciPy, PyTorch, Tensorflow)
    return Python_Result
end procedure
```

3) **Interoperability Layer:** The interoperability layer in `py-guam` facilitates seamless communication and synchronization between MATLAB and Python environments. This layer ensures that data passed between MATLAB and Python is correctly formatted and that function calls are executed without causing conflicts or errors. In the adapter functions, we use MATLAB's `coder.extrinsic` declarations, which inform MATLAB's code generation that certain functions—those interfacing with Python—should be treated as external. The interoperability layer handles the insertion of the module path into Python's system path, preventing issues where Python cannot locate the required modules. It also includes error checking after attempting to import Python modules, setting flags like `scriptexists` to manage the flow of execution depending on the availability of the Python code. Robust error-handling mechanisms are implemented to catch exceptions during Python function calls and data conversions.

Algorithm 3 Calling Python Function from MATLAB

procedure CALLPYTHONFUNCTION(*PythonModule*, *FunctionName*, *Python_Data*)
    Result ← PYTHONMODULE.FUNCTIONNAME(*Python_Data*)
    return *Result*
end procedure

4) **Data Conversion and Management:** Data conversion between MATLAB and Python is critical and is managed to maintain efficiency and prevent memory leaks. The adapter functions convert MATLAB data structures into Python-compatible formats using functions like `py.numpy.array` for arrays and matrices. For example, input data `x` is reshaped and converted into a NumPy array before being passed to the Python function. After processing in Python, the results are converted back into MATLAB arrays. This is achieved by iterating over the NumPy arrays returned from Python using `py.numpy.nditer` and constructing MATLAB arrays with the `double` function and typecasting via `py.array.array`. Pre-allocation of MATLAB variables before the data conversion ensures that memory is managed efficiently, and large data transfers are handled without unnecessary reallocations. The code avoids data duplication where possible and releases any temporary variables.

Algorithm 4 Data Conversion from MATLAB to Python

procedure CONVERTDATATOPYTHON(*MATLAB_Data*)
    Reshape *MATLAB_Data* if necessary
    Python_Data ← PY.NUMPY.ARRAY(*MATLAB_Data*)
    return *Python_Data*
end procedure

Algorithm 5 Data Conversion from Python to MATLAB

procedure CONVERTDATATOMATLAB(*Python_Result*)
    Initialize *MATLAB_Result* as empty array
    for all elements in PY.NUMPY.NDITER(*Python_Result*) **do**
        Append DOUBLE(element) to *MATLAB_Result*
    end for
    Reshape *MATLAB_Result* if necessary
    return *MATLAB_Result*
end procedure

Algorithm 6 Data Conversion and Management Strategies

procedure EFFICIENTDATAHANDLING
    Pre-allocate MATLAB variables for outputs
    Use persistent variables to store imported Python modules
    Avoid unnecessary data copies during conversion
    Release temporary variables after use to free memory
end procedure

Algorithm 7 Persistent Storage of Python Modules

procedure IMPORTPYTHONMODULE(*ModuleName*)
    if *PythonModule* is not in persistent storage **then**
        PythonModule ← PY.IMPORTLIB.IMPORT_MODULE(*ModuleName*)
        Store *PythonModule* in persistent storage
    end ifreturn *PythonModule*
end procedure

5) **Integration with Simulink:** Python-based analytics are integrated into Simulink models by embedding MATLAB-Python adapter functions within Simulink function blocks, allowing advanced analytics into the simulation workflow. Adapter functions are invoked from MATLAB Function blocks, enabling real-time

data exchange between Simulink and Python. This integration allows users to enable or disable Python functionalities by configuring Simulink blocks or setting parameters without altering the core simulation architecture. Additionally, this approach supports version control and component reuse.

Algorithm 8 Integration of Adapter Functions into Simulink

procedure IntegrateAdapterIntoSimulink(*Simulink_Model*)
    Add MATLAB Function block to *Simulink_Model*
    Set block's code to call the adapter function
    Connect input signals to the MATLAB Function block
    Connect output signals from the MATLAB Function block to subsequent blocks
    Configure block parameters and simulation settings as required
end procedure

The complete workflow of the MATLAB-Python adapter function is as follows:

Algorithm 9 Complete MATLAB-Python Adapter Function Workflow

procedure AdapterFunction(*MATLAB_Input*)
    InitializeAdapter
    if *scriptExists* **then**
        Python_Data ← ConvertDataToPython(*MATLAB_Input*)
        Python_Result ← CallPythonFunction(*PythonModule*, *FunctionName*, *Python_Data*)
        MATLAB_Output ← ConvertDataToMATLAB(*Python_Result*)
        return *MATLAB_Output*
    else
        Return default values
    end if
end procedure

IV. System Implementation

To systematically integrate new advanced analytics and AI algorithms into the GUAM framework, we follow a structured methodology. Tables 1 and 2 present the six key steps in this integration process.

A. Interaction with the Mission Manager

Shown in Figure 2, the GUAM CA takes in the Environment, Control, Vehicle, and Sensor signals at each timestep. The CA pulls in this information and produces analysis data from previous offline analysis capabilities. From a functional perspective, the Mission Manager sits inside the Generalized Control block and only leverages the most recent information processed by the CA. It may choose to buffer this information for its decision-making. Most sub-analyses of the CA rely on buffers and their respective models to decide how much historical information it will consider for a given analysis.

The CA is a Variant Subsystem under the Vehicle Simulation Block of GUAM. The implementation of Cognitive Memory in the design is a Simulink Bus sent to the output. When the CA is turned on, the most recent results at a given timestep are sent through this output signal. For a user to "turn on" the CA, the CognitiveType variable (userStruct.variants.cognitiveType) must be set to CognitiveEnum.INFORMED. Otherwise, vectors of 0's will be passed into Cognitive Memory.

Each submodule of the CA has a labeled MATLAB area for which the corresponding Simulink boxes are placed. Within each area, an individual subarea is designated for each analysis, such as a specific subarea for LOC analysis using VAEs. GOTO blocks are used for piping the analysis outputs to Cognitive Memory. Each subarea is (and should continue to be) annotated with a reference to a publication corresponding to the implementation being carried out by the box.

Using the process described in the previous subsection, we have incorporated several algorithms into the CA that can contribute to GUAM's decision-making capabilities. These algorithms are integrated to provide output to the Mission Manager, enhancing the ability to respond to various flight conditions. The initial implemented algorithms, developed

Table 1 Steps for Integrating Advanced Analytics and AI into GUAM

1. Develop AI Code and Prepare `py-guam` Environment

Write code for new AI algorithms as functions and add necessary analysis libraries to the GUAM Anaconda environment. Verify library setup with `test_environment.py`.

```
import scipy.integrate
from .hankel import hankel, block_hankel, block_s
np.set_printoptions(precision=9)

def linearize(x, energy_threshold=0.99, verbose=F
    '''
    @param x: Time series data (numpy array)
    @param verbose: If True, prints detailed debu
    @return: Tuple of numpy arrays (v, f, v_hat)
        - v: Linearized representation of state d
        - f: Intermittent forcing term
        - v_hat: Linearized representation of sta
    '''
    x = np.nan_to_num(x)
    if verbose:
```

```
>python test_environment.py
Library Import Successful
Core Package versions
    sklearn: 1.5.1
    tensorflow: 2.10.0
    scipy: 1.13.1
    numpy: 1.26.4
    pandas: 2.2.2
```

Fig. 5 Environment Setup and AI Code Development

2. Create Unit Tests for AI Functions

Develop AI-specific test functions and execute them using mock inputs in `test_functions.py`.

```
def test_linearize_with_multivariate_sine_wave(self):
    """Test linearize function with clean multivariate sine wa
    t = np.linspace(0, 10, 500)
    n_features = 3
    x = np.zeros((len(t), n_features))
    frequencies = [1, 0.5, 1.5]
    phases = [0, np.pi/4, np.pi/2]
    for i in range(n_features):
        x[:, i] = np.sin(2 * np.pi * frequencies[i] * t + phase

    # Plot the input multivariate time series
    self.plot_time_series(x, 'Input Multivariate Time Series -

    # Run linearize function
    v, f, v_hat = linearize(x, verbose=True)

    # Plot the outputs if they are not empty
    if v.size > 0 and f.size > 0 and v_hat.size > 0:
        self.plot_output(v, 'Output v from linearize() - Multi
        self.plot_output(f, 'Output f from linearize() - Multi
```

Fig. 6 Unit Testing of AI Functions

3. Develop and Test MATLAB Adapter Functions

Create MATLAB adapter functions to interface Python-based AI algorithms with MATLAB/Simulink.

```
function [v, f, vhat] = havok_py(x)
% havok_py.m - MATLAB wrapper for executing the HAVOK method us
  coder.extrinsic('py.importlib.import_module')
  coder.extrinsic('py.numpy.array')
  coder.extrinsic('py.dict')

    % Python call
    output = guamhavok.linearize(py.numpy.array(x), pyargs('
    v_temp = py2mat(output{1});
    f_temp = py2mat(output{2});
    vhat_temp = py2mat(output{3});
  end

  % Handle output
  [p,q] = size(v_temp);
  r = min(int32(p),RET_ROW_LEN);
```

Fig. 7 MATLAB Adapter Functions

4. Call MATLAB Adapter from Simulink Code Blocks

Integrate the validated adapter functions into appropriate Simulink code blocks within the CA. Develop and run test scripts to adjust inputs for debugging.

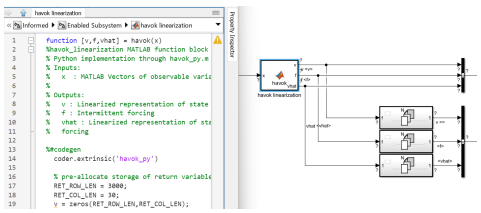

Fig. 8 Integration into Simulink Code Blocks

for a segment of the ICM program called Emergent Behavior and Response, are:

1) **LOC Detection using VAEs:** The CA employs VAEs trained on nominal flight maneuvers to detect anomalies indicative of a LOC situation[35, 36]. By attempting to reconstruct input data that deviates from the training set, the VAE's reconstruction error serves as a metric for detecting unusual flight behaviors. A high reconstruction error suggests that the current activity significantly differs from normal operations, prompting the CA to infer a potential LOC event.
2) **HAVOK Analysis for System Identification:** The HAVOK algorithm is utilized to create a minimal realization

Table 2 Execution and Assessment Steps for AI Integration in GUAM

5. Execute Full Trajectory Scripts
Run comprehensive trajectory scripts (e.g., `exam_TS_Hover2Cruise_traj.m`) to evaluate the integrated CA components under realistic flight conditions. These simulations assess the AI algorithms' ability to handle complex maneuvers and environmental dynamics.

```
Command Window
>> RUNME
Specify the desired example case to run:
    (1) Sinusoidal Timeseries
    (2) Hover to Transition Timeseries
    (3) Cruise Climbing Turn Timeseries
    (4) Ramp demo
    (5) Piecewise Bezier Trajectory
User Input: 3

userStruct =

  struct with fields:

    variants: [1×1 struct]

Default path setup
userStruct exists
userStruct.variants exists
userStruct.switches does not exist

----------------------------------------
Switch setup:
Lift+Cruise polynomial aerodynamic model: v2.1-MOF
Variant setup
Bus setup
```

Fig. 9 Execution of Full Trajectory Scripts

6. Assess Flight Outputs and Reports
Analyze simulation outputs to evaluate system performance, stability, and compliance with safety standards. Utilize reporting scripts like `reportSimPlots_GUAM.m` to generate visualizations and summaries, facilitating the identification of areas for improvement and ensuring that safety metrics are met.

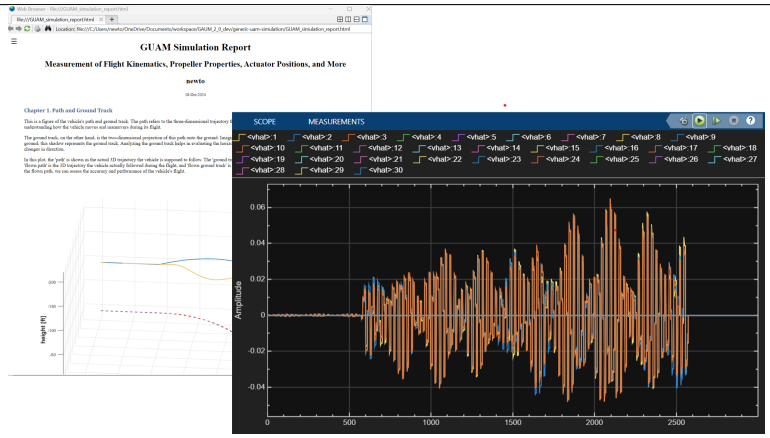

Fig. 10 Assessment of Flight Outputs and Reporting

of all flight and environmental variables over predefined time windows (typically 10-15 seconds)[7]. This analysis provides a low-dimensional representation of the system's dynamics, aiding in diagnostics and offering insights into the underlying behavior of the UAV.

3) **Endogenous Variable Threshold Monitoring using CN2 Rule Classification:** The CA monitors specific flight

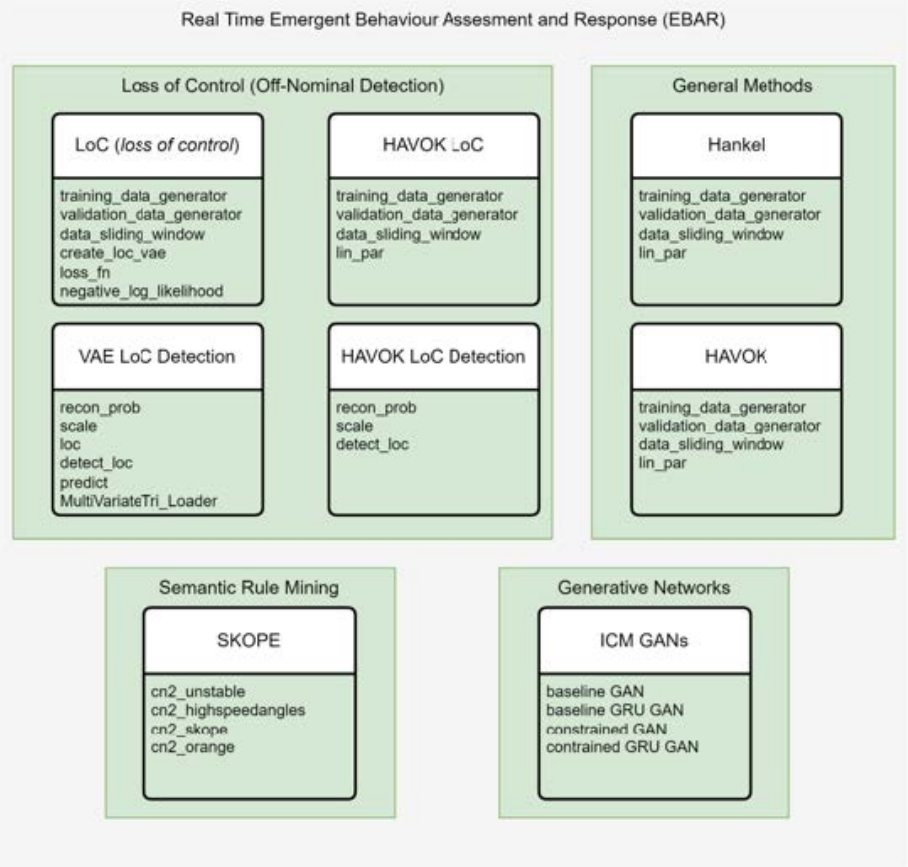

Fig. 11 Initial Python Modules developed for `py-guam`

stability parameters and velocities against hard-coded thresholds embedded within the simulation environment. When these thresholds are surpassed, the CA employs CN2 rule classification to learn the statistical correlations between variables, providing the Mission Manager with insights that, for example, indicate which variables correlate with instability[37].

4) **Combined HAVOK and LOC Detection:** This algorithm integrates the HAVOK minimal realization with the LOC detection mechanism. By inputting the low-dimensional representation from HAVOK into the VAEs, the system achieves faster and more accurate inference, reducing computational overhead and enhancing LOC detection efficiency.

Corresponding Simulink function blocks are configured to call Python scripts (shown in Figure 11 during the simulation as described in the last section, allowing the CA to process sensor and environment data at each timestep and return analytical outputs seamlessly. This real-time interaction addresses data conversion, buffering, and synchronization challenges between continuous raw data streams and discrete computational processes. Each submodule within the CA is organized in the simulation environment with explicit annotations and references to relevant publications. For instance, the Critical Point Detection submodule is implemented with Simulink blocks corresponding to specific analyses, such as LOC detection using VAEs.

V. Results

A. Designing for New Analytics and Experimentation

With a MATLAB/Simulink and Python integration and a corresponding design that lends to other third-party tool or language integration, the space of possible analytics and models that can be integrated into GUAM is quite large. Strategically navigating this landscape required a structured framework. Our previous work demonstrated how

to establish a UL 4600 framework for our ICM program[1] and produced a work product with safety case claims, sub-claims, evidence, methodologies, and SPIs to reach any of the levels of Mission Complexity highlighted in Figure 1. As GUAM is the corresponding internal simulator at the center of this program, we use this specification to scope the advanced analytics and AI algorithms and models that should be integrated into GUAM next.

Evidence for each claim and SPIs derived from UL 4600 safety case analyses are metrics for evaluating GUAM AI model performance. These indicators focus on safety-critical aspects such as LOC detection, anomaly detection, and successful mission completion rates. We reviewed this evidence and SPI across the UL 4600 safety cases with subject-matter experts (SMEs) on relevant projects within the NASA Aeronautics Research Mission Directorate (ARMD) branch. Our goal was to identify how these integrations could support a broader spectrum of AI models and advanced analytics while remaining consistent with UL 4600 standards for safety in complex environments. The structured interviews were with SMEs specializing in artificial intelligence, flight safety, and regulatory compliance. These discussions targeted a key question: *how can GUAM's flexibility and modularity best support iterative testing and integration of AI-driven models to enhance safety-critical outcomes in UAM?*

We provided clear instructions to interviewees that the interview was meant to refine our approach to Safety Performance Indicator (SPI) within GUAM, using these metrics to verify AI models specifically for flight-critical tasks. We aimed to translate these safety cases into actionable test cases within GUAM's framework, making the system adaptable to evolving AI models. The collected feedback established a baseline of safety cases for GUAM and identified where existing standards could be extended to include adaptive and resilient control in dynamic operational contexts.

Summary of Interview Insights

Our expert interviews focused on how a layered safety case methodology can support ICM for GUAM, ranging from basic compliance metrics to adaptive AI performance in complex scenarios. A key insight was distinguishing between static and dynamic SPIs. Experts emphasized the necessity of incorporating fixed metrics, such as adherence to predetermined flight paths, alongside dynamic metrics like the framework's responsiveness to unexpected LOC events. One interviewee noted that GUAM's structure allows informed, verifiable adjustments to SPIs as new AI models are tested, especially in variable risk environments.

Additionally, discussions highlighted the importance of adaptive SPIs in handling abnormal operations—especially pertinent to UAM, where unstructured traffic and unpredictable environmental changes are expected. By aligning SPIs to account for GUAM's potential to model non-communicative and uncooperative airspace traffic, experts suggested that GUAM could provide a more realistic benchmark for LOC detection and other mission-critical algorithms. For example, integrating SPIs that capture near-miss scenarios in non-cooperative air traffic adds a new layer of depth to GUAM 's safety performance assessment, aligning it more closely with real-world UAM needs. Additional highlights of the expert interviews were triaged as follows:

1) **Trajectory Accuracy and LOC Detection**: Evaluating trajectory accuracy under nominal and off-nominal conditions is critical. Experts recommended developing SPIs to detect deviations indicating early-stage LOC events, especially in dynamic environments with fluctuating air traffic or unexpected wind conditions. Benchmarking LOC detection rates through structured SPIs supports the continuous refinement of AI-driven control models, ensuring responsiveness to environmental shifts.
2) **Fault Detection and Adaptive Recovery Capabilities**: Enhancing fault resilience is vital for AI testing. Experts suggested incorporating SPIs that capture fault detection accuracy, time-to-recovery, and adaptive responses to anomalies like control surface malfunctions or sensor failures, especially in high-density urban operations. One SME highlighted GUAM's ability to simulate fault conditions across various flight stages, such as takeoff or landing, enabling assessment of AI models' real-time adaptability to prevent mission-critical failures.
3) **Mission Completion Rates under Varied Operational Complexity**: Tracking mission success rates is essential for evaluating AI models in complex, multi-objective scenarios. SPIs should measure whether vehicles complete missions despite environmental or operational anomalies. Experts emphasized that non-communicative or unstructured air traffic increases operational complexity, demanding AI models that can adapt trajectories or mission priorities when encountering unexpected events or communication disruptions. This SPI framework facilitates effective validation of mission-critical behaviors against FAA regulations.
4) **Anomaly Detection in Non-Cooperative Environments**: For GUAM to simulate real-world urban conditions, experts advocated for SPIs that measure the accuracy and timeliness of anomaly detection algorithms when encountering non-cooperative traffic. This includes identifying and adapting to unexpected traffic patterns without prior coordination, a key requirement in dense airspace where traditional air traffic control may not

fully apply. By establishing SPIs for anomaly detection, GUAM can evaluate how well AI models interpret and respond to real-time deviations from normal flight patterns, providing a realistic benchmark for UAM deployments.

5) **Predictive Metrics for Environmental Adaptability**: Adaptability to environmental changes, such as sudden weather shifts or airspace congestion, was recommended as an advanced SPI category. In these discussions, interviewees suggested creating metrics that assess how quickly and accurately an AI model can predict and adjust to environmental variables. This SPI could include metrics on response latency, adjustment accuracy, and success rates for recalibrating control algorithms in response to environmental data. For example, GUAM can be configured to alter environmental conditions dynamically within a single simulation, thereby testing the AI model's resilience and adaptability under stress.

Proposed Advanced Validation Techniques

The following advanced techniques were proposed for validation of the CA with respect to the UL 4600 safety cases.

1) **Anomaly Detection and Predictive Maintenance:** Utilizing advanced predictive maintenance methods, such as time-series forecasting and anomaly detection models like DeepAR[38] and Prophet[39], can enable the CA to preemptively detect performance deviations or sensor anomalies. Simulating potential wear or failure scenarios enhances maintenance identification and operational safety.
2) **Adaptive Thresholding for Flight Stability Monitoring:** Implementing adaptive thresholding algorithms allows the CA to dynamically adjust critical stability thresholds based on current operational conditions. Techniques like Bayesian thresholding or adaptive confidence intervals enable the CA to recalibrate in real time, improving resilience in new or challenging environments.
3) **Reinforcement Learning for Real-Time Decision-Making:** Incorporating reinforcement learning (RL) techniques, such as Proximal Policy Optimization (PPO) [40] or Deep Deterministic Policy Gradient (DDPG)[41], within the CA's decision-making processes can optimize responses to changing conditions through adaptive, experience-based adjustments. Simulated learning environments allow the CA to refine strategies for mission success and safety, leveraging RL's real-time trial-and-error learning.
4) **Explainability and Model Interpretability:** Adopting XAI techniques, such as Shapley values[42] or Local Interpretable Model-agnostic Explanations (LIME)[43], enhances transparency in the CA's decision-making, particularly in critical scenarios. Improved interpretability allows mission managers to understand the reasoning behind analyses, fostering trust in the CA's recommendations—a crucial step for reliable deployment in high-stakes UAV missions.

By integrating these advanced validation techniques, the CA can be more thoroughly tested and refined, enabling greater accuracy and adaptability across various mission-critical scenarios.

B. Mapping GUAM CA Components to UL 4600

Both GUAM and the GUAM CA are meticulously designed and implemented to allow for research into new models and algorithms for intelligent UAM. The implementation of `py-guam` within the CA helps us to account for alignment with UL 4600 safety standards, helping the ICM project understand areas of safety and reliability that the research is enhancing, as well as where we fall short. The modular structure of `py-guam` enables integration of functionalities addressing mission complexity, environmental adaptability, autonomy levels, decision-making complexity, and fault management, as outlined in the UL 4600 safety cases.

Table 3 maps GUAM's CA components to their `py-guam` implementations and corresponding UL 4600 safety case claims. Energy Estimators and Reinforcement Learning components are not currently implemented in `py-guam`. These areas represent ongoing research opportunities for complex flight plans and fault-level decision-making, respectively. Other implemented components of the CA align GUAM closer to our set of specific UL 4600 safety case claims. The World Interpretation component addresses issues in handling environmental complexity (Claim 1) and autonomy challenges (Claim 3). The Critical Point Detection component enables higher maturity in handling navigation, collision avoidance (Claim 3), task-level decision-making (Claim 4), and adaptability to dynamic environments (Claim 1). Finally, implementation of the CA Command Performance module enables the vehicle to make nuanced decisions about controls (Claim 4) and matures the ability to operate the vehicle under normal operational parameters (Claim 2). This research contributes to developing a framework for intelligent UAM adhering to safety standards. Future work will implement remaining components and refine existing modules to enhance GUAM alignment with specific safety case claims.

Table 3 Mapping of GUAM CA Components to `py-guam` Implementations and UL 4600 Safety Case Claims

GUAM CA Component	`py-guam` Module	Aligned UL 4600 Safety Case Claim
World Interpretation	Hankel HAVOK	Claim 1: Complexity of Environment (Dynamic Environments, Structured/Unstructured) Claim 3: Complexity of Autonomy (Flight Stability, Envelope Protection)
Critical Point Detection	LOC (Off-Nominal Detection) SKOPE	Claim 1: Complexity of Environment (Dynamic Environments) Claim 3: Complexity of Autonomy (Navigation and Collision Avoidance) Claim 4: Complexity of Decision-Making (Task-Level Decisions)
Energy Estimators	N/A	Claim 2: Complexity of Mission Operations (Complex Flight Plans)
Reinforcement Learning	N/A	Claim 5: Complexity of Mission Fault (Expected Fault-Level Decision-Making)
Command Performance	VAE LoC Detection	Claim 4: Complexity of Decision-Making (Control-Level Decisions) Claim 2: Complexity of Mission Operations (Normal Operations)

VI. Discussion

The interviews highlighted key gaps in integrating safety cases holistically into the software engineering process. **Trajectory accuracy metrics** can ensure that UAM vehicles adhere to predefined flight paths, minimizing the risk of collisions and optimizing airspace utilization. Continuously monitoring deviations from planned trajectories will lead to a system that can promptly identify and rectify navigational errors, thereby maintaining operational integrity. **Fault detection** metrics are pivotal for identifying and diagnosing anomalies within vehicle subsystems. Leveraging advanced machine learning, GUAM can detect early signs of component degradation or failure, enabling preemptive maintenance and reducing the likelihood of in-flight malfunctions. This approach to fault management aligns with reliability engineering principles, ensuring that UAM vehicles remain operational under varying conditions. **Mission completion** metrics assess the vehicle's ability to achieve its operational objectives under nominal and adverse conditions, evaluating factors such as payload delivery accuracy, adherence to mission timelines, and task execution. Measuring mission success provides valuable insights into the effectiveness of autonomous control strategies and decision-making processes. **Anomaly detection** is further enhanced through the integration of XAI techniques, which provide transparency into the decision-making processes of AI-driven systems. By leveraging XAI, the GUAM framework can not only identify anomalies but also elucidate the underlying causes, facilitating more informed and effective responses to unexpected events. **Environmental adaptability** metrics evaluate vehicle performance in diverse and changing urban environments, assessing the ability to adapt to varying weather conditions, air traffic densities, and infrastructure layouts. Ensuring that autonomous systems can dynamically adjust behavior in response to environmental fluctuations enhances operational resilience and safety.

From these conversations, we also produced a set of UL 4600 safety standards that are far more holistic, particularly with respect to their SPIs, which provide a structured foundation for certifying autonomous systems. A focus on implementing and advancing research in this area would allow research conducted on the GUAM framework a path to comply with industry best practices and facilitate the rigorous certification processes required for widespread UAM deployment. Furthermore, the integration of these SPIs into a unified framework enables comprehensive risk management, allowing for the identification, assessment, and mitigation of potential safety hazards across various operational scenarios.

A. AI Model Evaluation and Iteration

Evaluating and iteratively improving GUAM AI models present unique challenges that necessitate deeper exploration and targeted research. A primary challenge is assessing AI performance beyond traditional metrics like accuracy or precision, especially under rare or extreme conditions, including unforeseen faults or anomalies. While UL 4600 allows specification of such scenarios, determining sufficiency remains an open research area. Developing novel evaluation

methodologies to measure AI resilience, reliability, and generalization to unanticipated scenarios is required. Research using GUAM to stress-test AI models via adversarial simulations or synthetic data could reveal potential failure modes and areas for improvement.

Furthermore, iterative improvement of AI models is constrained by the limitations of simulation environments. While GUAM offers a sophisticated platform for simulating various flight conditions and fault scenarios, simulations may not fully capture the stochastic nature and unpredictability of real-world environments. Bridging this gap necessitates research into techniques such as domain adaptation and transfer learning, which enable AI models trained in simulated environments to perform reliably in real-world settings. Incorporating real-world data into the training and validation processes while addressing challenges related to data availability and privacy is crucial for enhancing model generalization.

Integrating advanced XAI techniques into the AI development lifecycle requires deeper investigation. While methods like Shapley values[42] provide some interpretability, more advanced XAI approaches are needed to offer comprehensive explanations of complex AI models, particularly deep learning architectures used in UAM applications. Research into domain-specific XAI methods is necessary to provide stakeholders——engineers, regulators, and operators——with clear, actionable insights into AI decision-making.

The iterative process of AI model refinement must be coupled with rigorous verification and validation (V&V) frameworks that align with safety standards such as UL 4600. Traditional V&V methods may not suffice for AI systems that learn and adapt over time. There is a pressing need for research into formal methods for AI verification, including probabilistic model checking and the use of formal specification languages tailored for AI behaviors. Establishing robust V&V processes ensures that iterative improvements do not introduce unintended behaviors that could compromise safety.

In addition to these technical challenges, research into the ethical implications of deploying AI models in autonomous UAM systems is needed. This encompasses ensuring that AI that informs decision-making aligns with ethical considerations, such as fairness, privacy, and accountability. Developing metrics and evaluation criteria that capture these ethical dimensions will enhance the societal acceptability of AI-enabled UAM operations.

B. Adaptability to Emerging Challenges

GUAM's adaptability ensures simulation environments and future deployments remain robust, flexible, and responsive to evolving demands. One critical aspect is adapting to evolving regulatory frameworks. As UAM is emerging, regulatory standards and certification processes are continually developing. The GUAM framework can adapt to changes in airspace regulations, safety standards, and certification requirements imposed by authorities like the FAA and international aviation organizations. This requires a modular architecture allowing seamless integration of new compliance criteria and simulating adherence to updated mandates.

As the number of UAM vehicles operating concurrently increases, GUAM must also efficiently handle complex interactions among numerous autonomous agents. Implementing distributed computing techniques, optimizing algorithms for parallel processing, and leveraging high-performance computing resources are essential methods for ensuring that the GUAM framework scales effectively without compromising real-time performance or simulation fidelity.

The framework must address technological adaptability by integrating emerging technologies such as advanced sensors, communication systems (e.g., 5G networks), and novel propulsion methods. This involves developing flexible interfaces and abstraction layers within the simulation that allow new hardware models and software algorithms to be incorporated with minimal reconfiguration. By staying technologically agnostic and making plug-and-play integration easier over time, the GUAM framework can remain at the forefront of UAM innovation.

Interoperability with existing and future air traffic management systems is another critical need. The GUAM framework should support standard communication protocols and data formats, such as the Aeronautical Information Exchange Model (AIXM) and System Wide Information Management (SWIM) architectures. This enables the simulation to interact with traditional air traffic management (ATM) systems and emerging uncrewed traffic management (UTM) platforms, facilitating testing and validation of UAM operations within integrated airspace environments.

Addressing human factors and operator adaptability is essential for the acceptance and effectiveness of UAM systems. The GUAM framework should incorporate modules that simulate human-operator interactions, pilot training scenarios, and user interface evaluations. By modeling various levels of automation and human oversight, the simulation can assess the impact of human factors on system performance and safety, informing the design of more intuitive and reliable control systems.

Incorporating ethical and societal considerations requires adaptability in modeling and evaluation criteria. The framework should enable the assessment of privacy impacts, noise pollution levels, and community acceptance factors. This includes simulating different operational policies, such as no-fly zones over sensitive areas, and evaluating the effectiveness of noise mitigation strategies. By adapting to these societal needs, the GUAM framework can contribute to the responsible development of UAM systems.

Risk management adaptability involves the ability to simulate and respond to novel hazard scenarios, including cybersecurity threats, system malfunctions, and adverse weather conditions. The framework should support dynamic risk modeling, allowing for integrating new threat models and the continuous update of SPIs. Techniques such as probabilistic risk assessment and Monte Carlo simulations can be employed to evaluate system resilience under various risk factors.

Finally, collaborative adaptability is vital for aligning the GUAM framework with industry needs. Establishing open interfaces and Application Programming Interfaces (APIs) allows researchers, industry partners, and regulatory bodies to integrate their models and tools into the simulation environment. This collaborative approach enables the sharing of data, validation of models against a common framework, and the acceleration of UAM technology maturation.

C. Integration with Existing Systems

For the practical deployment of SPIs within the GUAM framework, simulated integration with existing ATM and regulatory frameworks is imperative. This integration ensures that autonomous UAM operations can coexist safely and efficiently within the established aviation infrastructure, adhering to regulatory standards and operational protocols.

The UL4600 safety case analyses for GUAM are designed to interface with current FAA regulations and procedures, facilitating compliance and promoting interoperability with existing ATM systems. By aligning SPIs with FAA safety standards and operational guidelines, we can ensure that autonomous UAM vehicles meet the rigorous safety and performance criteria required for certification and operational approval. This alignment is crucial for the certification processes, as it allows GUAM to serve as a reliable tool for demonstrating compliance with industry standards from the very start of experimentation with new models and algorithms.

Furthermore, GUAM is made with interoperability constraints in mind, particularly existing ATM systems with standardized communication protocols and data exchange formats. By adopting data standards like extensible markup language (XML) and JavaScript Object Notation (JSON) for information exchange, GUAM can facilitate real-time data sharing and coordination with traditional air traffic control (ATC) systems. This interoperability enables autonomous UAM vehicles to receive and transmit critical flight information, enhancing situational awareness and enabling coordinated maneuvers within mixed airspace environments.

D. Limitations and Future Work

Despite these considerations, the GUAM framework faces limitations inherent to simulation that necessitate further research and development. Primarily, simulations may not fully encapsulate the unpredictability and complexity of real-world urban environments, potentially overlooking critical edge cases and dynamic interactions between multiple autonomous vehicles. This limitation highlights the necessity for extensive real-world validation to ensure that SPI-based assessments accurately reflect operational performance and safety outcomes. Additionally, the computational demands of high-fidelity simulations constrain the scalability and real-time responsiveness of the GUAM framework, especially when integrating sophisticated AI models and multi-agent scenarios. Furthermore, the rapid evolution of AI and machine learning techniques necessitates continuous updates to the SPI framework to incorporate emerging algorithms and methodologies. Ongoing adaptation is essential to maintain the relevance and effectiveness of safety assessments as new AI-driven functionalities and autonomous behaviors are developed. Future work should focus on several key areas:

- **Expand Scenario Coverage:** Future research should aim to broaden the range of scenarios tested within the GUAM framework, particularly those involving multiple AI-controlled vehicles interacting simultaneously. Multi-agent interactions introduce additional layers of complexity, such as coordination, communication, and conflict resolution, which are critical for ensuring safe and efficient UAM operations in crowded urban airspaces. Developing and integrating SPIs that can effectively monitor and evaluate these interactions will enhance the framework's ability to anticipate and mitigate potential safety risks.
- **Enhance Ethical Decision-Making Metrics:** Developing more sophisticated metrics for assessing the ethical decision-making capabilities of AI models is essential. This involves creating SPIs that evaluate the fairness, accountability, and transparency of autonomous decision processes in complex urban environments. Incorporating ethical considerations into the SPI framework will ensure that UAM systems not only perform efficiently but also

adhere to societal and regulatory expectations regarding responsible AI use.

- **Automated SPI Identification Using Machine Learning:** Integrating large language models (LLMs) with codebase analysis can enable the automatic identification and mapping of relevant SPIs from simulation data. This would enhance the adaptability and comprehensiveness of the safety assessment framework by dynamically generating new SPIs that accurately reflect safety and performance metrics across diverse operational contexts. Automating SPI development reduces the need for manual metric creation and ensures the framework remains aligned with the latest operational insights and technological advancements.
- **Real-World Validation Studies:** Conducting real-world validation studies is crucial to ensure that GUAM-based assessments accurately predict performance in actual UAM operations. These studies involve deploying autonomous UAM systems in controlled urban environments and comparing simulation results with empirical data. Real-world validation helps identify discrepancies between simulated and actual performance, providing valuable feedback for refining SPIs and improving the overall reliability of the GUAM framework. Additionally, these studies facilitate the calibration of simulation models to better reflect real-world conditions, enhancing the predictive accuracy of SPIs.

VII. Conclusion

This paper introduced the development and validation of the `py-guam` Python library of the NASA Generic Urban Air Mobility simulation (GUAM) framework, meticulously aligned with UL 4600 safety standards developed in prior work [1] to ensure the safety and reliability of autonomous UAM systems. By integrating MATLAB/Simulink with Python-based advanced analytics, `py-guam` provides a robust and scalable platform capable of supporting complex flight operations and dynamic environments.

The systematic testing methodology developed for the GUAM Cognitive Architecture (CA) ensures that all integrated AI algorithms and advanced analytics function correctly and can be appropriately characterized for adherence to safety standards. Through rigorous environment verification, unit testing, adapter function validation, and trajectory simulations, we have demonstrated GUAM's capability to manage complex flight maneuvers and respond effectively to dynamic conditions, aligning with UL 4600's rigorous safety requirements.

Future work will focus on expanding GUAM's capabilities by incorporating reinforcement learning and energy estimators, currently under development. These additions will further enhance mission operations and fault management, providing greater resilience and efficiency. Additionally, integrating multi-agent interaction modules and explainable AI techniques will be explored to improve transparency and robustness in decision-making processes. These extensions will both broaden GUAM's applicability but also reinforce its role as a cutting-edge platform for ICM. If the community continues refining and expanding its functionalities, GUAM is well-positioned to address the evolving landscape of UAM challenges.

References

[1] Campbell, N. H., Gregory, I. M., Acheson, M. J., Ilangovan, H. S., and Ranganathan, S., *Benchmark Problem for Autonomous Urban Air Mobility*, AIAA, 2024. https://doi.org/10.2514/6.2024-0718, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0718.

[2] Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K., "Intelligent Agents for Interactive Simulation Environments," *AI Magazine*, Vol. 16, No. 1, 1995, p. 15. https://doi.org/10.1609/aimag.v16i1.1121, URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1121.

[3] Zhang, Q., Wei, R., and Huang, S., "Cognitive Control Architecture for the Practical Realization of UAV Collision Avoidance," *Sensors*, Vol. 24, No. 9, 2024, p. 2790.

[4] Emel'yanov, S., Makarov, D., Panov, A. I., and Yakovlev, K., "Multilayer Cognitive Architecture for UAV Control," *Cognitive Systems Research*, Vol. 39, 2016, pp. 58–72. https://doi.org/10.1016/j.cogsys.2015.12.008.

[5] Zhang, W., Song, K., Rong, X., and Li, Y., "Coarse-to-fine UAV target tracking with deep reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, Vol. 16, No. 4, 2018, pp. 1522–1530.

[6] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G.-Z., "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI," *Information Fusion*, Vol. 58, 2021, pp. 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

[7] Ranganathan, S. I., Ilangovan, H. S., Campbell, N. H., Acheson, M. J., and Gregory, I. M., *Off-Nominal Event Analysis in Autonomous Flights Based on Explainable Artificial Intelligence*, 2024. https://doi.org/10.2514/6.2024-0720, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0720.

[8] Koopman, P., *The UL 4600 Guidebook: What to Include in an Autonomous Vehicle Safety Case*, Amazon Digital Services LLC - Kdp, 2022, 2022.

[9] Koopman, P., "UL 4600: What to Include in an Autonomous Vehicle Safety Case," *Computer*, Vol. 56, No. 5, 2023, pp. 101–104. https://doi.org/10.1109/MC.2023.3236171.

[10] Bhattacharyya, S., Eskridge, T. C., Neogi, N., and Carvalho, M., "Formal Assurance for Cognitive Architecture Based Autonomous Agent," *International Conference on Artificial General Intelligence*, Springer, 2023, pp. 54–65. https://doi.org/10.1007/978-3-031-35210-7_5.

[11] Phadke, A., Medrano, F. A., Sekharan, C. N., and Chu, T., "Designing UAV swarm experiments: A simulator selection and experiment design process," *Sensors*, Vol. 23, No. 17, 2023, p. 7359.

[12] Javed, S., Hassan, A., Ahmad, R., Ahmed, W., Ahmed, R., Saadat, A., and Guizani, M., "State-of-the-Art and Future Research Challenges in UAV Swarms," *IEEE Internet of Things Journal*, Vol. 11, No. 11, 2024, pp. 19023–19045. https://doi.org/10.1109/JIOT.2024.3364230.

[13] Taye, A., and Wei, P., "Flight Mission Feasibility Assessment of Urban Air Mobility Operations under Battery Energy Constraint," *AIAA SciTech Forum 2024*, 2024, pp. 1–13. https://doi.org/10.2514/6.2024-0532.

[14] Baek, J., Han, S. I., and Han, Y., "Energy-Efficient UAV Routing for Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 2, 2020, pp. 1741–1750. https://doi.org/10.1109/TVT.2019.2959808.

[15] Federal Aviation Administration, "Urban Air Mobility Concept of Operations v1.0," https://nari.arc.nasa.gov/sites/default/files/attachments/UAM_ConOps_v1.0.pdf, 2020. Accessed: 2023-10-05.

[16] Lin, P., Abney, K., and Jenkins, R. (eds.), *Robot Ethics 2.0: From Autonomous Cars to Artificial Intelligence*, Oxford University Press, New York, NY, 2017.

[17] Cohen, A., and Shaheen, S., "Urban Air Mobility: Opportunities and Obstacles," *International Encyclopedia of Transportation*, Elsevier, 2021, pp. 702–709. https://doi.org/10.1016/B978-0-08-102671-7.10764-X.

[18] Holden, J., and Goel, N., "Fast-Forwarding to a Future of On-Demand Urban Air Transportation," Uber Elevate White Paper, 2016. URL https://evtol.news/__media/PDFs/UberElevateWhitePaperOct2016.pdf, accessed: 2024-10-05.

[19] Airbus, "Vahana: The Future of Urban Air Mobility," Airbus Urban Mobility, 2018. URL https://www.airbus.com/en/innovation/zero-emission/urban-air-mobility, accessed: 2023-10-05.

[20] Gregory, I. M., Campbell, N. H., Neogi, N. A., Holbrook, J. B., Grauer, J. A., Bacon, B. J., Murphy, P. C., Moerder, D. D., Simmons, B. M., Acheson, M. J., et al., "Intelligent contingency management for urban air mobility," *Dynamic Data Driven Applications Systems: Third International Conference, DDDAS 2020, Boston, MA, USA, October 2-4, 2020, Proceedings 3*, Springer, 2020, pp. 22–26.

[21] Campbell, N. H., Ilangovan, H. S., Gregory, I. M., and Mikaelian, S. S., *Data Augmentation for Intelligent Contingency Management Using Generative Adversarial Networks*, AIAA, 2022. https://doi.org/10.2514/6.2022-0622, URL https://arc.aiaa.org/doi/abs/10.2514/6.2022-0622.

[22] Gregory, I. M., Acheson, M. J., Patterson, A. P., Houghton, M. D., Oshin, A., Ackerman, K. A., and Cook, J., *Candidate Performance Metrics for Generalized Control for Autonomous Flight*, AIAA, 2023. https://doi.org/10.2514/6.2023-2650, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-2650.

[23] Patterson, A., Ackerman, K. A., Cook, J., Acheson, M. J., and Gregory, I. M., *An L1 Adaptive Control Augmentation for a Lift-Plus-Cruise Vehicle*, AIAA, 2023. https://doi.org/10.2514/6.2023-2541, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-2541.

[24] Acheson, M. J., Gregory, I. M., and Cook, J., *Examination of Unified Control Incorporating Generalized Control Allocation*, AIAA, 2021. https://doi.org/10.2514/6.2021-0999, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0999.

[25] Sarojini, D., Ruh, M. L., Joshy, A. J., Yan, J., Ivanov, A. K., Scotzniovsky, L., Fletcher, A. H., Orndorff, N. C., Sperry, M., Gandarillas, V. E., Asher, I., Chambers, J. T., Gill, H., Lee, S., Cheng, Z., Rodriguez, G., Zhao, S., Mi, C., Nascenzi, T., Cuatt, T., Winter, T. F., Guibert, A. T., Cronk, A., Kim, H. A., Meng, S., and Hwang, J. T., *Large-Scale Multidisciplinary Design Optimization of an eVTOL Aircraft using Comprehensive Analysis*, AIAA, 2023. https://doi.org/10.2514/6.2023-0146, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-0146.

[26] Bhandari, R., Mishra, A. A., and Chakraborty, I., *Genetic Algorithm Optimization of Lift-Plus-Cruise VTOL Aircraft with Electrified Propulsion*, 2023. https://doi.org/10.2514/6.2023-0398, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-0398.

[27] Comer, A. M., and Chakraborty, I., *Flight Control System Architecture for Urban Air Mobility Simplified Vehicle Operations*, 2023. https://doi.org/10.2514/6.2023-0399, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-0399.

[28] Acheson, M. J., Gregory, I. M., and Cook, J., "Examination of unified control incorporating generalized control allocation," *AIAA Scitech 2021 Forum*, 2021, p. 0999.

[29] Houghton, M., Acheson, M., Patterson, A., Oshin, A., and Gregory, I., "Combined Bernstein Polynomial Optimal Reciprocal Collision Avoidance Differential Dynamic Programming for Trajectory Replanning and Collision Avoidance for UAM Vehicles," AIAA, 2023. https://doi.org/10.2514/6.2023-2544.

[30] Boyd, J. R., "Destruction and Creation," 1976. URL https://www.coljohnboyd.com/static/documents/1976-09-03__Boyd_John_R__Destruction_and_Creation.pdf.

[31] Lombaerts, T., Kaneshige, J., and Feary, M., "Control concepts for simplified vehicle operations of a quadrotor eVTOL vehicle," *AIAA aviation 2020 forum*, 2020, p. 3189.

[32] Simmons, B. M., and Murphy, P. C., *Wind Tunnel-Based Aerodynamic Model Identification for a Tilt-Wing, Distributed Electric Propulsion Aircraft*, AIAA, 2021. https://doi.org/10.2514/6.2021-1298, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-1298.

[33] Inc., T. M., "MATLAB," , 2023.

[34] Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E., and Kutz, J. N., "Chaos as an intermittently forced linear system," *Nature communications*, Vol. 8, No. 1, 2017, p. 19.

[35] Campbell, N. H., Grauer, J. A., and Gregory, I. M., *Loss of Control Detection for Commercial Transport Aircraft Using Conditional Variational Autoencoders*, AIAA, 2021. https://doi.org/10.2514/6.2021-0778, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0778.

[36] Campbell, N. H., Grauer, J. A., and Gregory, I. M., "Use of Design of Experiments in Determining Neural Network Architectures for Loss of Control Detection," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–20. https://doi.org/10.1109/AERO50100.2021.9438231.

[37] Kumar, N., and Kumar, U., "Comparative analysis of CN2 rule induction with other classification algorithms for network security," *Multimedia Tools Appl.*, Vol. 81, No. 26, 2022, p. 37119–37135. https://doi.org/10.1007/s11042-022-13542-3, URL https://doi.org/10.1007/s11042-022-13542-3.

[38] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T., "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International journal of forecasting*, Vol. 36, No. 3, 2020, pp. 1181–1191.

[39] Oswal, S., Hadawle, S., and Khangar, A., "Anomaly Detection in Time Series Data by Forecasting Using Facebook Prophet," *International Advanced Computing Conference*, Springer, 2022, pp. 205–220.

[40] Wu, Z., Yu, C., Ye, D., Zhang, J., Zhuo, H. H., et al., "Coordinated proximal policy optimization," *Advances in Neural Information Processing Systems*, Vol. 34, 2021, pp. 26437–26448.

[41] Xiong, H., Xu, T., Zhao, L., Liang, Y., and Zhang, W., "Deterministic policy gradient: Convergence analysis," *Uncertainty in Artificial Intelligence*, PMLR, 2022, pp. 2159–2169.

[42] Sundararajan, M., and Najmi, A., "The many Shapley values for model explanation," *International conference on machine learning*, PMLR, 2020, pp. 9269–9278.

[43] Mishra, S., Sturm, B. L., and Dixon, S., "Local interpretable model-agnostic explanations for music content analysis." *ISMIR*, Vol. 53, 2017, pp. 537–543.