A Safety-Driven Approach to Exploring and Comparing Air Traffic Management Concepts for Enabling Urban Air Mobility

Justin Poh, Dr Nancy G. Leveson
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA, USA
{jpoh, leveson}@mit.edu

Dr Natasha A. Neogi
Langley Research Center
National Aeronautics and Space Administration
Hampton, VA, USA
natasha.a.neogi@nasa.gov

***Abstract*— There is broad recognition that the high tempo and density of Urban Air Mobility (UAM) operations will require identifying new Air Traffic Management (ATM) concepts to safely integrate UAM air traffic into the airspace alongside existing air traffic. However, the simulation models used to compare ATM concepts today are difficult to apply during the early stages of concept development and do not offer enough support in identifying potential new concepts. In addition, they have limited ability to evaluate ATM concepts in terms of safety, security, and other key emergent properties. Instead of using simulation to evaluate ATM concepts, this paper demonstrates how a safety-driven systems engineering approach based on Systems-Theoretic Accident Model and Processes (STAMP) can be used to design properties such as safety into an ATM system from the earliest stages of development. Using a hazard analysis technique called Systems-Theoretic Process Analysis (STPA), system requirements and the desired ATM behavior are derived. As an example, two possible ATM concepts to implement that behavior are compared to identify their safety-related benefits and tradeoffs. This new approach enables (1) systematic exploration of alternative ATM concepts and (2) identification of the safety-related tradeoffs between concepts as early as possible in the development process.**

Keywords-System Safety, Systems Engineering, Architecture Development, Urban Air Mobility, Air Traffic Management

I. INTRODUCTION

Air Traffic Management (ATM) in the United States (US) has always been based on a centralized ATM concept [1]. Although numerous technological changes have been made over time to keep up with increases in air traffic, Air Traffic Control (ATC) has always been primarily responsible for managing the airspace and keeping aircraft safely separated, especially those flying under Instrument Flight Rules (IFR).

Although a centralized ATM concept has enabled a safe National Airspace System (NAS) in the US thus far, it will be challenging to simply adapt it to integrate Urban Air Mobility (UAM) because UAM introduces air traffic with significantly different characteristics than those that exist today [2], [3]. For example, today's air traffic can be centrally managed because the number of flight operations is relatively low and the air traffic consists primarily of commercial passenger flights that occur on a predictable schedule and flight path. However, UAM air traffic is anticipated to consist primarily of unscheduled, on-demand flights operating at a much higher traffic density, making it harder to manage using a centralized ATM concept.

Thus, to safely integrate UAM into the NAS, a fundamentally different ATM concept will need to be developed that can safely manage both UAM and existing air traffic. However, existing methods for evaluating new ATM concepts are limited in their ability to explore and compare new concepts in terms of key properties such as safety and security, especially during the early stages of development.

The goal of this research is to demonstrate that an alternative approach based on systems engineering and system safety principles can be used to generate useful design insights about potential ATM concepts early in the design lifecycle before developing any simulations. By modeling ATM concepts instead as NAS system *architectures*, alternative ways of managing air traffic can be explored and compared. This paper demonstrates how a new safety-driven approach to architecture development can enable safety and other desirable properties to be designed into the NAS from the beginning. It starts with a hazard analysis of the NAS using Systems-Theoretic Process Analysis (STPA) and uses the results to (1) systematically identify different possible NAS architectures representing potential new ATM concepts and (2) compare them to identify safety-related tradeoffs that can be used to inform how to architect the NAS to achieve system properties such as safety.

This paper is structured as follows. Section II reviews the limitations of existing methods for evaluating ATM concepts and Section III introduces the new safety-driven approach that

This research was funded by NASA contract 80NSSC20M0080

will be applied. Section IV then demonstrates how this new approach can be used to explore and compare possible NAS architectures for enabling UAM, and the benefits and tradeoffs of two possible architecture options are discussed. Finally, Section V concludes with a discussion of future work.

II. Existing Methods for Analyzing ATM Concepts

The methods used to analyze ATM concepts today can be divided into three main categories. The first category is physical NAS models that were developed to model specific areas of the NAS such as runways, airports, or terminal airspace at a high level of detail. These models are typically used to analyze the impact of changes to airspace structure or airport layout on performance metrics such as capacity or delays [4].

The second category is functional NAS models that model the NAS as a whole but at a higher level of abstraction and for a specific function such as conflict detection. There are two main types of functional NAS models: (1) Control-Theoretic Models and (2) Human Performance Models. Control-theoretic models use mathematical abstraction to model air traffic management as a control system to identify algorithmic ways to enable multiple decision-making agents to collectively control and coordinate the movements of aircraft to resolve conflicts and avoid collisions [5], [6]. By contrast, human performance models were developed to model the cognitive functions and decision-making of air traffic controllers or flight crews to analyze the impact of new ATM concepts or procedures on human performance metrics such as workload [7], [8].

The third category is simulation frameworks that provide the infrastructure needed to integrate different models into a complete simulation of air traffic flowing through the NAS [9], [10], [11], [12], [13]. These frameworks enable performance metrics such as throughput, capacity, or closest point of approach between aircraft to be calculated.

More recently, a new type of simulation framework called Agent-Based Modeling and Simulation (ABMS) [14], [15] provides a more dynamic approach to simulation. By modeling the interactions and decision-making of each agent to match what would occur in the real world, it is used to predict overall system behavior and performance.

Despite the wide variety of methods that have been developed for analyzing ATM concepts, they have several common limitations. First, all these methods assume that an ATM concept already exists, and the goal is simply to model and analyze it. These methods therefore lack any design support for identifying alternative ATM concepts.

Second, these methods are difficult to apply during the early stages of development because many of the detailed design decisions needed to implement these simulations may not yet have been made. As a result, assumptions may need to be made to feasibly implement the simulation.

Finally, these methods are limited in their ability to analyze an ATM concept for properties such as safety or security. This is because they compare ATM concepts using quantitative metrics such as throughput, or financial cost whereas properties such as safety do not have easily quantifiable metrics that can be calculated before the system is developed.

Instead of using simulation to evaluate ATM concepts, we advocate an alternative safety-driven approach. Modeling ATM concepts as NAS *architectures* improves our ability to not only evaluate ATM concepts based on safety and other properties but also design these properties into the NAS from the beginning.

III. A STAMP-based Architecture Development Process

This paper proposes to use a new architecture development process that is founded on Systems-Theoretic Accident Model and Processes (STAMP). This section reviews STAMP and its associated hazard analysis, STPA, and then introduces the STAMP-based architecture development process itself.

A. Overview of STAMP and STPA

STAMP is an accident causality model that provides a framework for understanding how system properties and behaviors arise from the interactions between a system's components, and how those interactions can lead to accidents.

STAMP emphasizes the importance of viewing a system holistically to fully understand its behavior, including hardware, software, human operators, organizational aspects and the system's operating environment. STAMP also recognizes that emergent properties such as safety arise from the interactions between system components. This holistic view of a system enables STAMP to explain how accidents or undesirable system behavior might arise not only from component failure but also from requirement or human factors flaws.

STAMP treats safety as a control problem, not a reliability problem. Instead of focusing on preventing component failure, STAMP focuses on preventing accidents or unacceptable losses by identifying and enforcing sufficient constraints to avoid undesirable system behavior.

To model the controls in a system, STAMP uses a hierarchical safety control structure that contains a controlled process and the various controllers that can control the system's behavior. A controller (which may be hardware, software or human) thus enforces the safety constraints by applying appropriate control actions and receiving feedback about the effect of those controls on the system [16]. See Fig. 1.

STAMP also recognizes the importance of process models (or mental models for humans) because they are used by controllers to select appropriate control actions. Controllers therefore need appropriate feedback to maintain accurate process models over time to avoid making unsafe decisions.

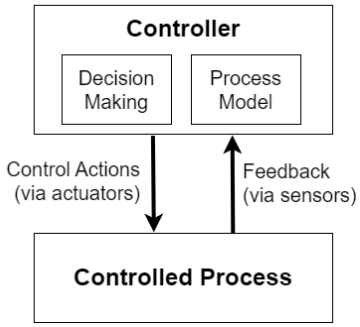

Figure 1: A simple control loop [16]

Using STAMP as a theoretical foundation, STPA is a hazard analysis method that analyzes the control loops in a system to determine how unsafe system behavior could occur due to both component failures as well as other control loop flaws [16]. Based on the causes of unsafe behavior identified by STPA, requirements can be generated to mitigate or prevent the unsafe behavior. When STPA is applied at an appropriate level of abstraction early in the development process, even a system as complex as the NAS can be analyzed to identify safety-related information to inform downstream design decisions.

This research is not the first to apply STPA to analyze ATM concepts. In [17], [18], Fleming demonstrates how STPA can be used to analyze and highlight the safety-related aspects of an ATM concept. However, that work assumes that an ATM concept already exists and focuses on addressing its flaws. This work extends Fleming's work by using STPA to generate possible new ATM concepts and compare them to inform the eventual selection of a final option.

B. A STAMP-Based Architecture Development Process

A safety-driven architecture development process based on STAMP was developed by the authors in [19]. See Fig. 2.

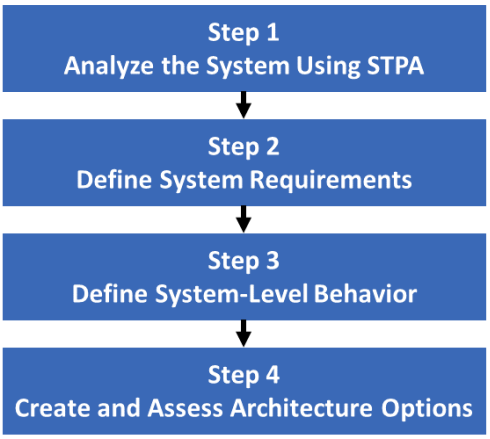

Figure 2: The STAMP-based architecture development process [19]

The goal of this process is to design a system to prevent unsafe or undesirable behavior. The first step is to analyze the system using STPA to determine how such behavior might occur. The STPA results are then used to generate system requirements that define the constraints needed to prevent unsafe behavior.

Once the system requirements have been generated, the third step is to define the system-level behavior needed to meet the requirements. Consistent with STAMP principles, defining the system-level behavior involves identifying the necessary control elements to adequately enforce the behavioral constraints described in the requirements [19]. These control elements include (1) responsibilities, (2) process model parts, (3) feedback and (4) control actions.

Once the system-level behavior has been created, the fourth step involves assigning the control elements to either existing components in the system or new components that will be added. Each architecture option is therefore defined by a unique set of assignments. Alternative architecture options can thus be systematically explored by considering how each responsibility and its associated control elements could be assigned. Each option can then be further analyzed using STPA and the results compared to determine the safety-related benefits and tradeoffs.

IV. Exploring and Comparing NAS Architectures Using a STAMP-Based Architecture Development Process

In this section, this architecture development process is applied to demonstrate how different NAS architectures representing potential ATM concepts can be explored and compared.

A. Step 1: Analyze the NAS Using STPA

The first step is to perform an initial STPA analysis of the NAS. As in any hazard analysis process, STPA starts by identifying the losses that are unacceptable to system stakeholders and the related high-level system hazards that could lead to them. The losses and hazards for the NAS are shown in Table 1.

Table 1: Losses and Hazards for the NAS

Losses	**Hazards**
L1: Loss of life or injury **L2:** Loss or damage to aircraft or equipment **L3:** Nonachievement of mission **L4:** Environmental effects (e.g. noise/visual pollution) exceed acceptable levels **L5:** Public safety is compromised	**H1:** Flights do not maintain minimum separation (e.g. to other flights, terrain) [L1, L2, L3, L5] **H2:** Air missions cannot be completed within acceptable operational limits [L3] **H3:** Environmental effects of flight operations exceed acceptable levels [L4] **H4:** Unauthorized aircraft enter an area that is restricted [L1, L3, L5]

Note that the losses and hazards in Table 1 account for safety as well as other desired emergent properties. This paper focuses on demonstrating how this development process can be used to ensure aircraft remain safely separated (i.e. control H1). Although controlling the other hazards is beyond the scope of this paper, this development process can be applied to do so.

Once the losses and hazards have been identified, the control structure model of the system is created. The high-level control structure for the NAS is shown in Fig. 3. Assuming the management of existing air traffic will remain similar to how ATC works today [2], the interactions between ATM and existing aviation aircraft and operators are modeled to reflect those interactions. However, for the interactions between ATM and UAM aircraft, no assumption is made about existing operations or a pre-existing operational concept. Instead, the interactions between ATM and UAM aircraft and operators are modeled abstractly using a control action called *Coordination* and generic feedback called *Requests* and *Reports*. Later in the analysis, these abstract control actions and feedback will be refined into more detailed ones based on the desired ATM behavior generated using this design process.

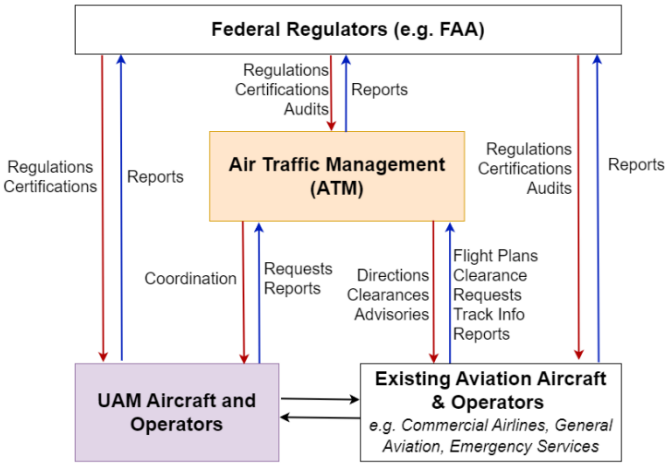

Figure 3: High-level NAS control structure

Once the control structure is created, it is analyzed to identify unsafe control actions (UCAs) that describe contexts in which providing or not providing a control action could be unsafe. There are four types of UCAs defined by STPA [16] and Table 2 shows examples of each type for the *Coordination* control action. Each UCA is traced to the resulting hazards labeled in square braces. Basically, Table 2 shows more detailed hazards stemming from the high-level ones in Table 1.

Once UCAs have been identified, STPA requires identifying loss scenarios that describe how each UCA might occur [16]. The left column of Table 3 shows three example loss scenarios associated with UCA-1.

B. Step 2: Generate NAS System Requirements

The second step of this process is to generate NAS system requirements to mitigate or prevent the identified loss scenarios. Consistent with STAMP principles, the system requirements describe the system-level behavioral constraints that, when adequately enforced, will prevent the undesirable behavior described in the loss scenarios from occurring. The satisfaction of these constraints ensures that the implemented requirements adequately control the hazards. The right column of Table 3 shows the derived system requirements that mitigate the three example loss scenarios.

C. Step 3: Defining NAS System-Level Behavior

Once the system requirements have been generated, the third step of this process is to define the system-level behavior necessary to satisfy these constraints. This process involves identifying the responsibilities and associated process model parts, feedback and control actions necessary to enforce the behavioral constraints described by each system requirement.

Table 4 shows the system-level behavior defined for four example system responsibilities that must be performed to keep aircraft safely separated. The first three responsibilities in Table 4 satisfy the three derived requirements in Table 3.

Table 2: Example UCAs associated with the "Coordination" control action issued by ATM

Not Providing Causes Hazard	Providing Causes Hazard	Providing Too Early/Too Late Causes Hazard	Stopped Too Soon/Applied Too Long Causes Hazard
UCA-1: ATM does not provide coordination when a collision between two aircraft is imminent [H1, H2] **UCA-2:** ATM does not provide coordination when UAM flights interfere with other flight operations [H1, H2, H4]	**UCA-3:** ATM provides coordination to allow a UAM aircraft to access the airspace when it cannot accommodate that aircraft [H1, H2] **UCA-4:** ATM provides coordination that imposes excessive operational impacts (e.g. delays) on flights [H2]	**UCA-5:** ATM provides coordination too late to assist a UAM aircraft experiencing an emergency [H1, H2] **UCA-6:** ATM provides coordination too late after UAM aircraft have interfered with the operations of another NAS user [H2]	**UCA-7:** ATM provides coordination to restrict UAM flights for too long after congestion has returned to acceptable levels [H2] **UCA-8:** ATM stops coordinating UAM aircraft too soon before an airspace restriction has been lifted [H1, H2, H4]

Table 3: Example loss scenarios and associated system requirements associated with UCA-1

Example Loss Scenarios Associated with UCA-1	**Derived System Requirement**
LS-1.1: ATM receives feedback about a potential collision but is preoccupied addressing other collisions and does not address this one.	**Req-1:** NAS must coordinate the movement of aircraft to prevent all potential conflicts between aircraft.
LS-1.2: ATM receives feedback about the potential collision but does not try to address it because it assumes at least one of the aircraft will recognize and address the potential collision on its own. However, neither aircraft does so and the conflict occurs.	**Req-2:** NAS must ensure that aircraft involved in a potential conflict have received coordination, are executing it correctly and that the risk of collision is no longer present.
LS-1.3: ATM receives feedback of the potential collision but does not act on it because there are more aircraft in the airspace than it can manage.	**Req-3:** NAS must only allow as many aircraft to access the airspace as it is capable of managing.

Table 4: Simplified system-level behavior for four example system responsibilities

(1) Responsibilities	**(2) Process Model Parts**	**(3) Feedback**	**(4) Control Actions**
Resp-1: Prevent conflicts between aircraft	For each aircraft: • Track and planned trajectory • Flight & navigation capabilities and constraints	• Aircraft tracks • Planned trajectories • Aircraft capabilities • Operational constraints	• Trajectory modifications
Resp-2: Ensure trajectory modifications are received, executed, and successfully resolve the conflict	• Potential collisions and associated trajectory modifications • Aircraft tracks • Trajectory modification acknowledged	• Potential collisions and trajectory modifications (from Resp-1) • Aircraft tracks • Acknowledge trajectory modification	• Request acknowledgement of trajectory modification • Request (to Resp-1) to revise trajectory modification
Resp-3: Manage airspace access to avoid exceeding the number of active flights that can be safely managed	• Flight plan for aircraft requesting access • Operational constraints (e.g. acceptable delay length)	• Flight plans & planned trajectories • Operational constraints	• Approve access request • Flight plan modifications
Resp-4: Ensure compliance with airspace restrictions	• Planned airspace restrictions. • Planned trajectory and flight plan for each aircraft	• Flight plans & planned trajectories • Planned airspace restrictions	• Trajectory modifications • Flight plan modifications

Note that the system-level behavior shown in Table 4 refines the abstract control actions and feedback shown in Fig. 3. For example, the abstract control action issued by ATM was originally called *Coordination* in Fig. 3 and Table 4 refines this into more specific control actions including *Trajectory Modifications* and *Flight Plan Modifications*. Similarly, the feedback available to ATM in Fig. 3 was *Requests* and *Reports* and Table 4 refines these into more specific feedback including *Aircraft Tracks* and *Planned* Trajectory.

D. Step 4: Create and Assess Architecture Options

The last step is to assign the identified responsibilities, process model parts, feedback and control actions to system components to create a NAS system architecture. This involves defining a set of architecture options, then analyzing and comparing them using STPA to identify benefits and tradeoffs.

1) Defining Architecture Options

In this STAMP-based approach, architecture options are created by assigning each responsibility to either an existing or new component in the system and then assigning that responsibility's associated process model parts, feedback and control actions to the same component. Each architecture option therefore defines the responsibilities performed by each controller and the control elements needed by each controller to carry out their assigned responsibilities. By creating architecture

options this way, a spectrum of architecture options can be systematically explored to determine the best way to assign responsibilities to achieve the desired emergent properties.

For example, to design the NAS to maintain separation between aircraft, the four responsibilities shown in Table 4 need to be assigned to either the aircraft or ATM. Although there are numerous ways to make these assignments, for length reasons, this paper presents and analyzes two opposing architecture options: (1) a **centralized** NAS architecture similar to today's ATM system and (2) a **decentralized** NAS architecture where the aircraft are primarily responsible for collision avoidance instead. Comparing these two opposing architecture options provides useful design information about what aspects of the NAS should be centralized or decentralized. The responsibility assignments for these two architecture options are shown in Table 5 and the corresponding control structures are shown in Fig. 4. To reduce clutter, only the refined control actions and feedback associated with the four responsibilities are shown.

2) *STPA Analysis of Architecture Options*

Having defined these two NAS architecture options, STPA can be used again to analyze each of them. Instead of performing a new STPA analysis for each option, the initial STPA analysis originally performed in step 1 of this process is updated.

To demonstrate how the two NAS architecture options shown in Fig. 4 are analyzed, consider UCA-1 originally shown in Table 2. *Trajectory* Modifications is one of the refined control actions associated with the abstract *Coordination* control action. So, UCA-1 can be refined to obtain UCA-1.1 and more detailed scenarios associated with UCA-1.1 can be generated. This STPA refinement should be done for each UCA identified in the initial analysis. Table 6 shows UCA-1.1 and examples of refined scenarios associated with UCA-1.1 for each architecture option.

3) *Comparison of NAS architecture options*

Having analyzed these two architectures, the causal factors described in the STPA scenarios can be compared. Comparing the loss scenarios allows system designers to make decisions about both the detailed design and larger conceptual design of the system. This paper demonstrates the latter by comparing a centralized and decentralized NAS architecture to inform decisions about which responsibilities should involve centralized or decentralized decision making.

Table 5: Responsibility Assignments for the Two NAS Architecture Options

Responsibility	**Option 1 Centralized Architecture**		**Option 2 Decentralized Architecture**	
	ATM	***UAM Aircraft***	***ATM***	***UAM Aircraft***
Resp-1: Avoid conflicts between aircraft	•			•
Resp-2: Ensure coordination instructions are received, executed, and successfully resolve the conflict	•			•
Resp-3: Manage access to the airspace	•		•	
Resp-4: Ensure conformance with airspace restrictions	•	•		•

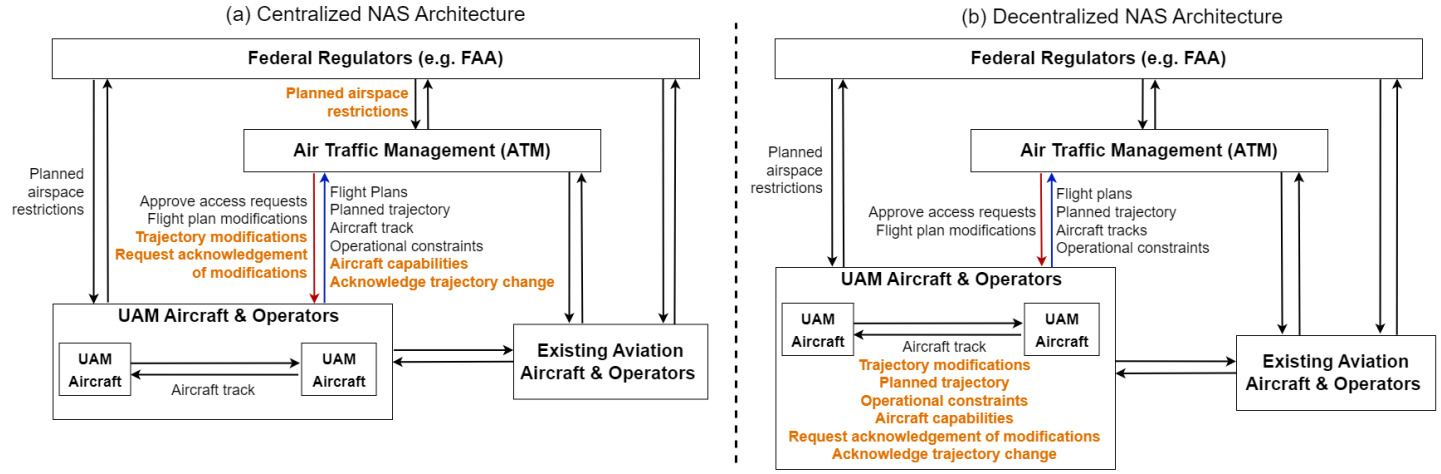

Figure 4: Simplified control structures representing the centralized (left) and decentralized (right) NAS architectures with differences highlighted in orange

Table 6: Examples of scenarios identified for the two architecture options for UCA-1.1

UCA-1.1: Trajectory Modifications are not provided when a collision between two aircraft is imminent	
Causal Scenarios for Architecture Option 1 *(ATM selects Trajectory Modifications)*	**Causal Scenarios for Architecture Option 2** *(Aircraft select Trajectory Modifications)*
LS-1.1.1: ATM receives feedback of the imminent collision but does not attempt to resolve it because it is preoccupied with resolving one set of conflicts and does not attend to the second set of conflicts. **LS-1.1.2:** ATM correctly recognizes the imminent collision and selects trajectory modifications to prevent it. However, due to communications failure or malicious interference, UAM aircraft do not receive them. **LS-1.1.3:** ATM does not realize a collision is imminent because it does not receive feedback when an aircraft's navigational capabilities are degraded (e.g. reduced GPS accuracy). Thus, ATM wrongly believes two UAM aircraft have sufficient separation to avoid a collision, but they do not.	**LS-1.1.4:** Two groups of UAM aircraft recognize that a collision is imminent for each of them. They independently select trajectory modifications to address their respective collisions but do not realize their selected trajectories conflict with each other. **LS-1.1.5:** A UAM aircraft experiences an emergency that requires a quick trajectory change to return to its departure aerodrome. If the UAM aircraft has to coordinate this trajectory change with numerous other nearby aircraft, it may take too long for all aircraft to agree on a set of acceptable trajectory modifications before a collision occurs.

Consider the assignment of Resp-1 and Resp-2. The scenarios identified in this analysis suggest that the air traffic context at a given point in time determines whether it is preferable for ATM or the aircraft to perform those responsibilities. If the airspace is densely populated or an aircraft experiences an in-flight emergency, a centralized architecture where ATM performs Resp-1 and Resp-2 (as in option 1) may be preferable. This is because under these conditions, any potential conflicts need to be resolved in a coordinated manner to avoid causing secondary conflicts with other aircraft. Compared to the aircraft, it is easier for ATM to have the necessary situational awareness of all air traffic in the airspace and their constraints on acceptable trajectories. ATM is therefore better equipped to make timely trajectory modification decisions to simultaneously coordinate the trajectories of numerous aircraft. By contrast, if the aircraft performed Resp-1 and Resp-2, they may not sufficiently coordinate their trajectory modifications and cause secondary collisions (e.g. scenario LS-1.1.4) or take so long to coordinate acceptable trajectory modifications that they do not resolve the conflict before it occurs (e.g. scenario LS-1.1.5).

However, one tradeoff of this centralized architecture is that, as the sole decision maker for collision avoidance, ATM is required to reliably make complex decisions, especially when a conflict involves a large group of aircraft or an unexpected situation. If ATM encounters a conflict that it cannot resolve in a timely manner (e.g. scenario LS-1.1.1), the ability of the NAS to keep aircraft safely separated will be significantly compromised. Another tradeoff of this centralized architecture is that maintaining adequate communication between ATM and the aircraft is critical because ATM is the sole decision maker for conflict avoidance. If there is a disruption in ATM's ability to communicate trajectory modifications to the aircraft in a timely manner (e.g. scenario LS-1.1.2), the ability of the NAS to keep aircraft safely separated will again be compromised.

In densely populated airspace, it may be necessary to incur these tradeoffs to gain the benefit of better coordinated conflict resolutions. However, under different circumstances, it may not be necessary to incur these tradeoffs. When the airspace is not densely populated, it may be more feasible to allow the aircraft to perform Resp-1 and Resp-2 (as in option 2) and coordinate among themselves to address any potential conflicts (i.e. self-separate). This is because, when the airspace is not densely populated, any conflicts are likely to involve relatively small groups of aircraft and different sets of conflicts are likely to occur far enough apart that they could be resolved independently with minimal risk of causing secondary collisions.

Thus, when the airspace is not densely populated, allowing self-separation might be preferable to address some of the tradeoffs associated with a centralized architecture. Distributing the decision making for conflict avoidance to the aircraft would alleviate the need for ATM to make complex and frequent decisions and the aircraft could resolve smaller conflicts in parallel. In addition, the ability of the NAS to avoid collisions is somewhat more resilient because the inability of some aircraft to select or coordinate trajectory modifications does not necessarily impact the ability of other aircraft to do so.

An additional benefit of an architecture where the aircraft perform Resp-1 and Resp-2 is that the aircraft can adapt their trajectory modification decisions more quickly to changing flight conditions (e.g. deteriorating weather, changes in GPS navigation accuracy). This is because the aircraft have direct access to feedback about these conditions and can more quickly account for them when making trajectory modification decisions. By contrast, if ATM performs Resp-1 and Resp-2, ATM must wait for the aircraft to provide feedback about those conditions to account for them (e.g. scenario LS-1.1.3).

Because the preferred assignment of Resp-1 and Resp-2 is so dependent on circumstances, instead of an architecture that

assigns Resp-1 and Resp-2 to either the aircraft or ATM alone, it may be preferrable to consider a more flexible architecture that allows for dynamic allocation of those responsibilities based on the circumstances. In such an architecture, either ATM or the aircraft could decide to perform Resp-1 and Resp-2, depending on who is better equipped to make the necessary decisions in each situation. For example, if only a few aircraft are involved in an isolated conflict, ATM could delegate responsibility for resolving the conflict to the aircraft. However, when the air traffic density is high or if an aircraft experiences an emergency, the aircraft could request ATM's assistance with resolving the conflict or ATM could take over that responsibility and make trajectory modification decisions instead of the aircraft.

This flexible architecture therefore represents a more collaborative approach to air traffic management where ATM and the aircraft will need to work together to adequately prevent collisions. However, enabling safe and effective collaboration between ATM and the aircraft requires adequate coordination between them [20]. Thus, this architecture will need to be further refined to define the interactions needed to enable effective coordination between ATM and the aircraft. This refinement of the NAS architecture is currently being developed.

This comparison illustrates that this process provides more design information than just a comparison of two discrete architecture options. It also provides insights into how system behavior changes with different responsibility assignments and those insights can inform decisions about how best to assign the responsibilities to achieve the desired system properties.

V. Conclusions & Future Work

Current approaches for evaluating ATM concepts are limited in their ability to explore and compare new ATM concepts for enabling UAM. This paper introduced a new safety-driven architecture development process that overcomes these limitations and enables emergent properties such as safety to be designed into an ATM system as early as possible.

Using this process, this paper showed how safety-related information obtained from STPA analyses can be used to both drive the generation of NAS system requirements and system-level behavior as well as to compare architecture options to identify their safety-related benefits and tradeoffs. In addition, this approach makes it easier to systematically explore architecture options by considering different possible responsibility assignments to identify those that will best achieve the desired emergent properties.

Several refinements to this process are currently in development to improve the support it provides for developing system architectures. One is a more structured process to better guide architecture exploration. Another is a more rigorous process for using STPA to compare potential architecture options. These improvements will be presented in future work.

VI. References

[1] M. S. Nolan, *Fundamentals of air traffic control*, 5th ed. Clifton Park, N.Y: Delmar Cengage Learning, 2011.

[2] D. P. Thipphavong *et al.*, "Urban air mobility airspace integration concepts and considerations," presented at the Aviation Technology, Integration, and Operations Conference, 2018.

[3] E. R. Mueller, P. H. Kopardekar, and K. H. Goodrich, "Enabling airspace integration for high-density on-demand mobility operations," presented at the 17th AIAA Aviation Technology, Integration, and Operations Conference, 2017.

[4] A. R. Odoni and R. W. Simpson, "Review and Evaluation of National Airspace System Models," US Federal Aviation Administration, FAA-EM-79-12, Oct. 1979.

[5] G. Pappas, C. Tomlin, J. Lygeros, D. Godbole, and S. Sastry, "A next generation architecture for air traffic management systems," in *Proceedings of the 36th IEEE Conference on Decision and Control*, IEEE, 1997.

[6] P. K. Menon, G. D. Sweriduk, and K. D. Bilimoria, "New approach for modeling, analysis, and control of air traffic flow," *J. Guid. Control Dyn.*, 2004.

[7] K. M. Corker, "Human performance simulation in the analysis of advanced air traffic management," in *Proceedings of the 1999 Winter Simulation Conference*, 1999.

[8] B. F. Gore, B. L. Hooey, and D. C. Foyle, "NASA's Use of Human Performance Models for NextGen Concept Development and Evaluations," in *Proceedings of the 20th Behavior Representation in Modeling & Simulation (BRIMS) Conference*, 2011.

[9] K. D. Bilimoria, B. Sridhar, S. R. Grabbe, G. B. Chatterji, and K. S. Sheth, "FACET: Future ATM concepts evaluation tool," *Air Traffic Control Q.*, 2001.

[10] M. Peters, M. Ballin, and J. Sakosky, "A multi-operator simulation for investigation of distributed air traffic management concepts," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2002.

[11] S. George *et al.*, "Build 8 of the airspace concept evaluation system," in *AIAA Modeling and Simulation Technologies Conference*, 2011.

[12] J. E. I. Robinson, A. Lee, and C. F. Lai, "Development of a High-Fidelity Simulation Environment for Shadow-Mode Assessments of Air Traffic Concepts," presented at the Royal Aeronautical Society: Modeling and Simulation in Air Traffic Management Conference, London, UK, Nov. 2017.

[13] J. M. Hoekstra, R. N. H. W. van Gent, and R. C. J. Ruigrok, "Designing for safety: the 'free flight' air traffic management concept," *Reliab. Eng. Syst. Saf.*, vol. 75, no. 2, pp. 215–232, Feb. 2002

[14] A. R. Pritchett *et al.*, "Examining air transportation safety issues through agent-based simulation incorporating human performance models," in *Proceedings of The 21st Digital Avionics Systems Conference*, Oct. 2002.

[15] H. A. P. Blom and G. J. Bakker, "Safety Evaluation of Advanced Self-Separation Under Very High En Route Traffic Demand," *J. Aerosp. Inf. Syst.*, vol. 12, no. 6, pp. 413–427, Jun. 2015

[16] N. Leveson and J. P. Thomas, "STPA Handbook." Mar. 2018. [Online]. Available: psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

[17] C. H. Fleming and N. G. Leveson, "Including Safety during Early Development Phases of Future Air Traffic Management Concepts," *Elev. USAEurope Air Traffic Manag. Res. Dev. Semin.*, 2015.

[18] C. H. Fleming, M. Spencer, J. Thomas, N. Leveson, and C. Wilkinson, "Safety assurance in NextGen and complex transportation systems," *Saf. Sci.*, vol. 55, Jun. 2013

[19] J. Poh, "A Top-Down, Safety-Driven Approach to Architecture Development for Complex Systems," Masters, MIT, 2022.

[20] A. Kopeikin, N. Leveson, and N. A. Neogi, "System-Theoretic Analysis of Unsafe Collaborative Control in Teaming Systems," in *AIAA SCITECH 2024 Forum*, Orlando, FL, Jan. 2024.